Symmetries and Distances : two intriguing challenges in Mathematical Programming

Gustavo Dias da Silva

NNT : 2017SACLX008

Thèse de doctorat de l'Université Paris-Saclay préparée à l'École Polytechnique

École doctorale n°580
Sciences et Technologies de l'Information et de la Communication (STIC)
Spécialité de doctorat : Informatique

Par

M. Gustavo Dias da Silva

Symétries et Distances : deux défis fascinants dans la Programmation Mathématique

Thèse présentée et soutenue à Palaiseau, le 24 Janvier 2017.

Composition du Jury :

To my beloved family.

ABSTRACT

This thesis is mostly dedicated to study and discuss two important challenges existing not only in the field of Mathematical Programming: symmetries and distances. In the background we take a look into Semidefinite Programming and its pertinency as one of the major tools employed nowadays to solve hard Mathematical Programs. After the introductory Chapter 1, we discuss about symmetries in Chapter 2 and about distances in Chapter 4. In between them we present a chapter that we actually prefer to call as *entr'acte*: its content is not necessarily worthy of publication yet (it does not provide any innovation so far), but it does provide a connection between the two seemingly separate Chapters 2 and 4, which are the ones containing the main contributions of this thesis.

It is widely known that symmetric Mathematical Programs are harder to solve to global optimality using Branch-and-Bound type algorithms, given that the solution symmetry is reflected in the size of the Branch-and-Bound tree. It is also well-known that some of the solution symmetries are usually evident in the formulation, which makes it possible to attempt to deal with symmetries as a preprocessing step. Implementation-wise, one of the simplest approaches is to break symmetries by adjoining Symmetry-Breaking Constraints to the formulation, thereby removing some symmetric global optima, then solve the reformulation with a generic solver. Sets of such constraints can be generated from each orbit of the action of the symmetries on the variable index set. It is unclear, however, whether and how it is possible to choose two or more separate orbits to generate Symmetry-Breaking Constraints which are compatible with each other (in the sense that they do not make all global optima infeasible). In Chapter 2 we discuss and test a new concept of Orbital Independence that clarifies this issue. The numerical experiences conducted using public Mixed-Integer Linear Programs and Mixed-Integer Nonlinear Programs emphasize the correctness and usefulness of the Orbital Independence theory.

In the sequel we continue to examine the impact of symmetries under the particular scope of Binary Quadratic Programming, which encompasses binary programs with a quadratic objective function and quadratic constraints, and Semidefinite Programming, which is considered to be one of the most significant developments in Mathematical Programming in the last decades. Semidefinite Programming importance stems from the fact that many practical problems in engineering and operations research can be modelled or approximated as Semidefinite Programs, and also from the fact that such programs

provide tighter convex relaxations to several nonconvex combinatorial optimization problems when compared to other methods. Particularly, Semidefinite Programming suits Quadratically Constrained Quadratic Programming. The Entr'acte 3 employs Binary Quadratic Programming (a specialization of Quadratically Constrained Quadratic Programming) to investigate symmetries and Semidefinite Programming all together. This is of concern first because Binary Quadratic Programming is in and on itself a relevant subfield of Mathematical Programming, and second because Semidefinite Programming is a typical tool used to cope with problems related to Distance Geometry, which we introduce in Chapter 4. We generate symmetric Binary Quadratic Programs having a certain symmetry structure and use them to exemplify the conditions under which the usage of Symmetry-Breaking Constraints is majoritarily advantageous. Moreover, we try to grasp initial impressions on the impact of symmetries and their breaking devices in the performance of Semidefinite Programming and Diagonally Dominant Programming solvers when handling this particular class of (symmetric) Mathematical Programs.

Finally, we properly enter the Distance Geometry subject. In Chapter 4 we cope with the most fundamental problem arising in the field of Distance Geometry, the one of realizing graphs in Euclidean spaces: it asks to find a realization of an edge-weighted undirected graph in $\mathbb{R}^K$ for some given K such that the positions for adjacent vertices respect the distance given by the corresponding edge weight. The Euclidean Distance Geometry Problem is of great importance since it has many applications to science and engineering. It is notoriously difficult to solve computationally, and most of the methods proposed so far either do not scale up to useful sizes, or unlikely identify good solutions. In fact, the need to constrain the rank of the matrix representing feasible solutions of the Euclidean Distance Geometry Problem is what makes the problem so hard. Intending to overcome these issues, we propose a two-steps heuristic algorithm based on Semidefinite Programming (or more precisely based on the very recent Diagonally Dominant Programming paradigm) and the explicitly modelling of Rank Constraints. We provide extensive computational testing against randomly generated instances as well as against feasible realistic protein conformation instances taken from the Protein Data Bank to analyze our method.

RÉSUMÉ

Cette thèse est principalement consacrée à l'étude et à la discussion de deux questions importantes qui se posent, entre autres, dans le domaine de la Programmation Mathématique : les symétries et les distances. En arrière-plan, nous examinons la Programmation Semidéfinie et sa pertinence comme l'un des principaux outils employés aujourd'hui pour résoudre les Programmes Mathématiques difficiles. Après le chapitre introductif, nous discutons des symétries au Chapitre 2 et des distances au Chapitre 4. Entre ces deux chapitres, nous présentons un court chapitre que nous préférons en fait appeler entr'acte : leur contenu ne mérite pas d'être publié pour le moment (il ne fournit aucune innovation à ce jour), mais il fournit un lien entre les deux Chapitres 2 et 4 apparemment distincts, qui sont ceux qui contiennent les principales contributions de cette thèse.

Il est bien connu que les Programmes Mathématiques symétriques sont plus difficiles à résoudre pour l'optimalité globale en utilisant des algorithmes du type Branch-and-Bound, étant donné que la symétrie de solution est reflétée dans la taille de l'arbre Branch-and-Bound. Il est également bien connu que certaines des symétries de solution sont habituellement évidentes dans la formulation, ce qui permet d'essayer de traiter les symétries en tant qu'étape de prétraitement. En termes de mise en œuvre, l'une des approches les plus simples consiste à rompre les symétries en associant les contraintes de rupture de symétrie à la formulation, en supprimant ainsi des optima globaux symétriques, puis à résoudre la reformulation avec un solveur générique. Des ensembles de ces contraintes peuvent être générés à partir de chaque orbite de l'action des symétries sur l'ensemble des indices des variables. Cependant, il est difficile de savoir si et comment il est possible de choisir deux ou plus orbites distinctes pour générer des Contraintes de Rupture de Symétrie qui sont compatibles les unes avec les autres (en ce sens qu'elles ne rendent pas tous les optima globaux infaisables). Dans le Chapitre 2, nous discutons et testons un nouveau concept d'Indépendance Orbitale qui clarifie cette question. Les expériences numériques réalisées à l'aide de Programmes Linéaires et Non-linéaires Mixtes-Entiers soulignent l'exactitude et l'utilité de la théorie de l'Indépendance Orbitale.

Dans la suite, nous continuons à examiner l'impact des symétries dans le cadre particulier de la Programmation Quadratique Binaire, qui englobe les programmes binaires avec une fonction objectif quadratique et des contraintes quadratiques, et la Programmation Semidéfinie, considérée comme l'un des développements les plus significatifs en Programmation Mathématique dans les dernières décennies. L'importance de la Programmation Semidéfinie découle du fait que

de nombreux problèmes pratiques de recherche d'ingénierie et d'exploitation peuvent être modélisés ou approchés sous forme de Programmes Semidefinis, et aussi du fait que de tels programmes fournissent des relaxations convexes plus serrées à plusieurs problèmes d'optimisation combinatoire nonconvexes comparés à d'autres méthodes. En particulier, la Programmation Semidéfinie s'adapte à la Programmation Quadratique. L'Entr'acte 3 emploie la Programmation Quadratique Binaire (une spécialisation de la Programmation Quadratique) pour étudier ensemble les symétries et la Programmation Semidéfinie. Ce sont des sujets d'intérêt d'abord parce que la Programmation Quadratique Binaire est en elle-même un sous-domaine pertinent de la Programmation Mathématique, et deuxièmement parce que la Programmation Semidéfinie est un outil typique utilisé pour faire face aux problèmes liés à la Géométrie de Distance que nous introduisons dans le Chapitre 4. Nous générons des programmes quadratiques binaires symétriques ayant une certaine structure de symétrie et nous les utilisons pour illustrer les conditions dans lesquelles l'utilisation des Contraintes de Rupture de Symétrie est avantageuse. De plus, nous essayons de saisir les impressions initiales sur l'impact des symétries et de leurs dispositifs de rupture dans la performance des solveurs de Programmation Semidéfinie et de Programmation Diagonale Dominante en manipulant cette classe particulière de Programmes Mathématiques symétriques.

Pour finir nous entrons proprement dans le champ de la Géométrie de Distance. Dans le Chapitre 4, nous abordons le problème le plus fondamental qui se pose dans le domaine de la Géométrie de Distance, celui de la réalisation des graphes dans les espaces euclidiens : il s'agit de trouver une réalisation d'un graphe pondéré non orienté dans $\mathbb{R}^K$ pour un certain K donné, de sorte que les positions pour les sommets adjacents respectent la distance donnée par le poids de l'arête correspondante. Le Problème de la Géométrie de Distance Euclidienne est d'une grande importance car il a de nombreuses applications en science et en ingénierie. Il est notoirement difficile de calculer numériquement des solutions, et la plupart des méthodes proposées jusqu'à présent ne sont pas adaptées à des tailles utiles ou sont peu susceptibles d'identifier de bonnes solutions. En fait, la nécessité de contraindre le rang de la matrice représentant des solutions réalisables du Problème de la Géométrie de Distance Euclidienne est ce qui rend le problème si difficile. Dans le but de surmonter ces problèmes, nous proposons un algorithme heuristique en deux étapes basé sur la Programmation Semidéfinie (et la Programmation Diagonale Dominante) et la modélisation explicite de Contraintes de Rang. Nous fournissons de nombreux tests informatiques comprenant des instances générées de façon aléatoire ainsi que des exemples réalistes de conformation de protéines (prises auprès de la banque de données de protéine) pour analyser notre méthode.

PUBLICATIONS

Some of the ideas presented in this thesis have appeared previously in the following publications:

[1] Gustavo Dias and Leo Liberti. "Orbital Independence in Symmetric Mathematical Programs." In: *Proceedings of the 9th Annual International Conference on Combinatorial Optimization and Applications*. Ed. by Zaixin Lu, Donghyun Kim, Weili Wu, Wei Li, and Ding-Zhu Du. Vol. 9486. Lecture Notes in Computer Science. Springer International Publishing, 2015, pp. 467–480.

[2] Gustavo Dias and Leo Liberti. "Diagonally dominant programming in distance geometry." In: *Proceedings of the 4th International Symposium on Combinatorial Optimazation*. Ed. by Raffaele Cerulli, Satoru Fujishige, and Ridha Mahjoub. Vol. 9849. Lecture Notes in Computer Science. Springer-Verlag, 2016, pp. 225–236.

[3] Gustavo Dias and Leo Liberti. "New methods for the Distance Geometry Problem." In: *Proceedings of the 14th Cologne-Twente Workshop on Graphs and Combinatorial Optimization*. Ed. by Alberto Ceselli, Roberto Cordone, and Giovanni Righini. Electronic Notes in Discrete Mathematics. Gargnano, Italy, 2016, pp. 57–60.

[4] Gustavo Dias, Leo Liberti, and Nelson Maculan. "Modelling Rank Constraints in Mathematical Programming." In: *Proceedings of the 13th Cologne-Twente Workshop on Graphs and Combinatorial Optimization*. Ed. by Ekrem Duman and ALi Fuat Alkaya. Istanbul, Turkey, 2015, pp. 193–196.

"Someone has said something" — Someone.

ACKNOWLEDGMENTS

This thesis is actually the result of the collaboration of many people: some were directly or indirectly involved in the research itself; others made Paris a better place to live during the four years I spent there. That being so, I would like to thank all of them!

To Leo Liberti for having accepted me as his PhD student, for his guidance and for sharing with me both his everlasting motivation to broaden the boarders of scientific knowledge, and his experience with respect to several aspects of a successful scientific carrer.

To Nelson Maculan for his imense patience and attention during the whole length of my PhD journey, and particularly for the many inspirational conversations we used to have during his stays in Paris. I will carry your teachings with me forever, Mestre.

To Sourour Elloumi, Amélie Lambert, Fabio Furini, Frédéric Messine and Marcia Fampa for accepting to be members of the jury and also for their questions, comments and suggestions, which undoubtely helped me to improve the quality of this work.

To the Brazilian people and the National Council for Scientific and Technological Development (CNPq) for the crucial financial support.

To Claudia, Sonia, Pierre-Louis, Raouia, Luca, Olivier and Claire for making the Sysmo team at LIX a welcoming and pleasant place to work, a true multinational team of researchers!

To my dear friends Eduardo, Kostas, Ky, Naveen and Youcef for the daily coffee breaks we used to take at LIX to chill out. We had many funny discussions, I will miss those days forever, guys!

To my dear friends Mariana, Bruno, Pedro, Gérson and many others for all the amazing moments we have been through together during my stay in the Maison du Portugal in the Cité Universitaire, :)

And to my beloved family, for their love and support during my whole lifetime. Without you, I would not have succeeded. Amo vocês.

CONTENTS

ACRONYMS

MP	Mathematical Programming
$\mathbb{MP}$	Mathematical Program
LP	Linear Programming
$\mathbb{LP}$	Linear Program
ILP	Integer Linear Programming
MILP	Mixed-Integer Linear Programming
$\mathbb{MILP}$	Mixed-Integer Linear Program
NLP	Nonlinear Programming
$\mathbb{NLP}$	Nonlinear Program
INLP	Integer Nonlinear Programming
MINLP	Mixed-Integer Nonlinear Programming
$\mathbb{MINLP}$	Mixed-Integer Nonlinear Program
SDP	Semidefinite Programming
$\mathbb{SDP}$	Semidefinite Program
SOCP	Second-Order Cone Programming
$\mathbb{SOCP}$	Second-Order Cone Program
DDP	Diagonally Dominant Programming
$\mathbb{DDP}$	Diagonally Dominant Program
QCQP	Quadratically Constrained Quadratic Programming
$\mathbb{QCQP}$	Quadratically Constrained Quadratic Program
BQP	Binary Quadratic Programming
$\mathbb{BQP}$	Binary Quadratic Program
GO	Global Optimization
PD	Positive Definite
PSD	Positive Semidefinite
DD	Diagonal Dominant

LMI	Linear Matrix Inequality
EDM	Euclidean Distance Matrix
BB	Branch-and-Bound
IPM	Interior-Point Method
DSB	Dynamic Symmetry Breaking
SSB	Static Symmetry Breaking
OS	Orbital Shrinking
SBC	Symmetry-Breaking Constraint
TC	Trace Constraint
RC	Rank Constraint
MIPLIB	Mixed Integer Problem Library
MINLPLib	MINLP Library
OI	Orbital Independence
AMPL	A Mathematical Programming Language
GAMS	General Algebraic Modeling System
API	Application Programming Interface
CLI	Command Line Interface
DAG	Directed Acyclic Graph
SMT	Steiner Minimal Tree
MST	Minimum Spanning Tree
DG	Distance Geometry
EDGP	Euclidean Distance Geometry Problem
DMCP	Distance Matrix Completion Problem
EDMCP	Euclidean Distance Matrix Completion Problem
SCP	Set Covering Problem
GSCP	Geometric Set Covering Problem
ESTP	Euclidean Steiner Tree Problem
MWCP	Maximum Weight Clique Problem
MSTP	Minimum Spanning Tree Problem

GCP	Graph Coloring Problem
QAP	Quadratic Assignment Problem
MCP	Molecular Conformation Problem
PDB	Protein Data Bank
NMR	Nuclear Magnetic Resonance
PCA	Principal Component Analysis
LNCS	Lecture Notes in Computer Science

LIST OF TABLES

Part I

OVERVIEW

1 INTRODUCTION

In this thesis we dedicate most of our effort to study and discuss two very intriguing challenges existing not only in the field of Mathematical Programming: symmetries and distances. The second most important subject adressed would be Semidefinite Programming.

In general terms, mathematical symmetry has to do with objects that held an invariant, i.e., a property that remains unchanged when operations (or transformations) of a certain type are applied to the objects. Particularly, the mathematical objects, property and operations in which we are interested in this work are, respectively, the global optimal solutions (or optima) of Mathematical Programs, the objective function value attained by these solutions, and permutations of finite sets. In this sense, we can formally describe symmetric optima as global optimal solutions that (by definition have the same objective function value and) are mapped into each other by permuting the values attributed to (some of) the decision variables. As we explain later on, the presence of symmetries acting within Mathematical Programming might bring forth undesired effects when it comes to solvers performance. On the other hand, it also prompts several theoretical and practical demanding questions, notably as concerns the development of techniques aiming to detect and exploit them efficiently, hopefully in our favor.

Distance is a numerical description of how far apart two objects are. In physics, distance refers to a physical length. In Mathematics, distance refers to the value assigned by a metric (or distance function) to each pair of elements in a set. Obviously, distances are intrinsic to our lifes and, as a consequence, they are present in most of the practical situations faced by humans on a daily basis. This natural relation with the notion of distance is deceptive and gives us the false impression that it is always trivial to estimate or calculate them. For concrete cases, perhaps. Carrying a scale or the closed form of a distance function, for instance, one can easily compute the distance between two windows in a living room or the distance between two given points in an Euclidean space. But if one needs to solve problems by means of computer algorithms involving distance calculations, derivative of distance functions and so forth, several mathematical and computational challenges arise. This work provides an account of our attempt to solve the most fundamental problem in Distance Geometry using Mathematical Programming tools whilst, of course, trying to adress these issues.

1.1 MATHEMATICAL PROGRAMMING

Mathematical Programming (MP) is a descriptive language used to formalize several types of optimization problems in terms of parameters (input), decision variables (output), constraints and objective functions, by defining a set of corresponding Mathematical Programs ($\mathbb{MP}$s) [48]. This work focus exclusively on the single-objective optimization paradigm. In this context, we consider problems $P \in \mathbb{MP}$ in the following general form:

$$\left.\begin{array}{rl} \min\limits_{x} & f(x) \\ \forall i \in \mathcal{I}_I & g_i(x) \leqslant 0, \\ \forall i \in \mathcal{I}_E & g_i(x) = 0, \\ & x \in B. \end{array}\right\} \qquad (1)$$

In Eq. (1), $f, g_i : \mathbb{R}^n \to \mathbb{R}$ are functions for which we have closed form expressions f, g_i for each $i \in \mathcal{I}_E \cup \mathcal{I}_I$. The expressions are written in terms of a formal language $\mathcal{L}$ based on an alphabet $\mathcal{A}$ consisting of a finite number of operators (e.g. sum, difference, product, fractions, powers, square roots, basic transcendental functions such as logarithm and exponentials, and possibly more complicated operators depending on the application at hand), a countable supply of variable symbols $x_1, \dots, x_n$ representing the decision variables $x_1, \dots, x_n$, and the rational numbers. The set B might contain nonfunctional constraints such as ranges $[x^L, x^U]$ for the decision variables, and/or integrality constraints, encoded as an index set $Z \subseteq N = [n] = \{1, \dots, n\}$ such that $x_j \in \mathbb{Z}$ for each $j \in Z$. As is well-known, this paradigm contains Linear Programming (LP), Integer Linear Programming (ILP), Mixed-Integer Linear Programming (MILP), Nonlinear Programming (NLP), Integer Nonlinear Programming (INLP), Mixed-Integer Nonlinear Programming (MINLP) and Semidefinite Programming (SDP) if $x_1, \dots, x_n$ are matrices.

1.2 SYMMETRIES IN MATHEMATICAL PROGRAMMING

By far, the most widely used technique for solving optimization problems formulated as Eq. (1) is the Branch-and-Bound (BB) algorithm paradigm. In brief words, BB type algorithms consist of a tree-based search in the solution set (or feasible region) of a given problem for the best solution. Explicit enumeration is normally impossible due to the exponentially increasing number of potential solutions, and thus the search is wisely performed, from the root to the leaf nodes, using branching and prunning rules based on lower and upper bounds on the value of the objective function.

Since all the global optima have the same value in terms of objective function, it is immediate to note that should many leaf nodes in the BB tree contain symmetric global optima, BB type algorithms may converge even slower on problems whose solution set has many of them [49]: all the symmetric leaf nodes must be visited (or explored) in order to verify and assert convergence. In fact, it was shown in [49] that roughly 18% of the MP instances in commonly employed public libraries, such as Mixed Integer Problem Library (MIPLIB) and MINLP Library (MINLPLib), have nontrivial symmetry. Therefore, symmetries are investigated in MP mainly to reduce the computation time of BB type algorithms.

Generally, we can break down any strategy designed to cope with symmetries in MP into two main phases:

1. Symmetry *detection*;

2. Symmetry *exploitation*.

As we shall expose in Chapter 2, when it comes to symmetry detection, it is not difficult to cite several combinatorial optimization problems whose MP formulations inevitably exhibit (trivial) symmetries due to their intrinsic symmetric nature, an example being Packing Problems with identical objects [19]. Facing such cases, researchers and practitioners typically uncover the presence of formulation symmetries "manually", meaning that they identify the symmetries by inspecting explicit properties of the problem. Nevertheless, formulation symmetries are not always evident to human perception and, therefore, the development of automated procedures aiming to detect them systematically becomes critical. And these algorithmic approaches are largely based on Abstract Algebra, or more precisely, on the algebraic structures studied in Group Theory [16]: *groups*. In Chapter 2 we provide a short review of the Group Theory concepts related to the study of symmetries in MP.

Similarly, in terms of symmetry exploitation, the most common paradigm employed in the literature is to eliminate (or, as we prefer to say, to break) the symmetries. This predilection for symmetry breaking should entail smaller search trees with respect to the number of symmetric branches (subtrees). In order to accomplish such a feat, the information obtained during the detection phase is normally used either to derive constraints to be adjoined to original formulation or to derive branching rules to be applied during the execution of the BB algorithm itself.

If both phases (detection and exploitation) are performed before running the solution algorithm, i.e., as a pre-solve procedure, we call such strategy as a Static Symmetry Breaking (SSB) approach. Otherwise, we call it a Dynamic Symmetry Breaking (DSB) approach [62]. The work developed herein concerns a general-purpose automated SSB strategy that advocates the use of Symmetry-Breaking Constraints (SBCs), tailored specifically to eliminate symmetries. Overall, we can describe our methodoly in a subtle higher level of detail (with respect to the exploitation phase) as follows:

1. *Detect* formulation symmetries;
2. *Generate* new constraints and;
3. *Reformulate* the original problem.

The main contribution of this research concerns the second step: constraint generation. We devise theoretical conditions that allow us to exploit the symmetry properties of a problem P as much as possible, by generating as many SBCs as possible to be adjoined to P in the reformulation step. The expectation is, as a consequence, to cut out the largest possible number of symmetric optima from the solution set of P. We then proceed and design an algorithm that implements all these conditions, and we embed it within a SSB strategy.

Hereupon, one can observe how intimately related are symmetries and reformulations when it comes down to SSBs. For this reason, we dedicate a few words in the following section to briefly introduce the Theory of Reformulations, provided that it plays an important part as regards symmetries, not to mention in MP as a whole.

1.2.1 *Reformulations*

When a given problem P is cast into a different one, say $P' \in \mathbb{MP}$, we call P' a reformulation of P. Furthermore, it is fairly well-known that different formulations may share numerical properties (e.g., the set of optima). In this sense, reformulations play an essential role in MP since casting a problem formulation into the specific form required by a certain algorithm is crucial to guarantee the efficiency of the solution process. Such specific forms are called the *standard form* of the problem with respect to the algorithm. We allude to [48] for a list of the most common standard forms.

Yet, despite of this essential role and its frequent use, there were few attempts to formally define what a reformulation is in MP until 2009. Among these few attempts, [82] proposes a definition that requires the existence of a bijection σ between the feasible regions of P and P' and [8] a definition based on complexity theory tools such as instance, polynomial amount of time, and other concepts. Both are quite limited however. The first work that objectively contributes towards creating an unified Theory of Reformulations is [48], by providing several definitions as regards this concept. We present the most fundamental ones in what follows.

Although more quantifiers and sets could be eventually involved, consider the binary variables $x \in \{0,1\}^{|I|\times|J|}$ and the packing constraints (or packing polytope [40])

$$\forall i \in I \quad \sum_{j\in J} x_{ij} \leqslant 1, \tag{2}$$

written using quantifiers $\sum, \forall$ and sets of indices I, J. It does not depend on the particular instance (i.e. numerical data) being solved. But given one, e.g., where $I = \{1,2\}$ and $J = \{1,2,3\}$, the above constraints translate into

$$\left.\begin{array}{l} x_{11}+x_{12}+x_{13} \leqslant 1, \\ x_{21}+x_{22}+x_{23} \leqslant 1. \end{array}\right\} \tag{3}$$

The first formulation (2) is called a *structured formulation* and the second one (3) is called a *flat formulation*. Formulations are commonly written by researchers and practitioners in structured form so as to decouple the model and data layers. Modelling softwares [30, 75], in turn, obligatorily convert structured formulations into flat formulations before passing them to solvers, given that solvers implement numerical algorithms.

Formally, a flat problem P is defined as [48, 56]: given an alphabet $\mathcal{A}$ consisting of countably many alphanumeric names $N_{\mathcal{A}}$ and operator symbols $O_{\mathcal{A}}$, a flat P is a 7-tuple $(\mathcal{P}, \mathcal{V}, \mathcal{E}, \mathcal{O}, \mathcal{C}, \mathcal{B}, \mathcal{T})$ where:

- $\mathcal{P} \subseteq N_{\mathcal{A}}$ is the sequence of parameter symbols;
- $\mathcal{V} \subseteq N_{\mathcal{A}}$ is the sequence of variable symbols;
- $\mathcal{E}$ is the set of mathematical expressions;
- $\mathcal{O} \subseteq \{-1,1\} \times \mathcal{E}$ is the sequence of objective functions;
- $\mathcal{C} \subseteq \mathcal{E} \times \{-1,0,1\} \times \mathbb{R}$ is the sequence of constraints;
- $\mathcal{B} \subseteq \mathbb{R}^{|\mathcal{V}|} \times \mathbb{R}^{|\mathcal{V}|}$ is the sequence of variable bounds and;
- $\mathcal{T} \subseteq \{0,1,2\}^{|\mathcal{V}|}$ is the sequence of variable types.

The subsets like $\{n_1, ..., n_i\}$ with $n_i \in \mathbb{Z}$, identify, respectively, optimization directions (minimization or maximization), types of constraints ($\leqslant, =$ or $\geqslant$) and types of decision variables (continuous, integer or binary).

An *expression tree* $e = (V_e, A_e) \in \mathcal{E}$ can be represented by a Directed Acyclic Graph (DAG), a natural data structure to store flat formulations. Nodes may represent parameters, decision variables or operators whereas edges may represent the parent-child relationships between the nodes. For a few examples, see [48]. Moreover, *reformulation schemas* define searching patterns that allow the manipulation of these expression trees, i.e., which allow the execution of the reformulations in practical terms. Such graphs can be evaluated by the algorithm proposed in [56].

The most commom flat reformulations in the literature are: opt-reformulations, narrowings and relaxations. In basic terms (see [48] for precise definitions):

Definition 1. *Opt-reformulations are reformulations that guaranteedly preserve both local and global optimality.*

Definition 2. *Narrowings are reformulations that guaranteedly preserve at least one global optimum.*

Definition 3. *Relaxations are reformulations whose feasible regions contain the feasible region of the original problem but do not keep track of optimality.*

Relaxations are far and away the most used reformulations, mainly to provide bounds on the objective function value at the optimum. They are derived either by removing some constraints (Langrangian relaxations, continuous relaxations, etc) or using simpler constraints (such as convex relaxations [47]). A taxonomy of several flat reformulations is given in [56].

Finally, a structured P is a P defined over structured entities [51]. Consider a sequence $\mathcal{I} = \{I_\beta \subseteq \mathbb{N} \mid \beta \leqslant \alpha\}$ of finite subsets of integers and a multi-index $\mathbf{i} = (i_1, ..., i_\alpha)$ where $i_\beta \in I_\beta$, for each $\beta \leqslant \alpha$. A structured parameter p is a jagged array of parameter symbols $p_\mathbf{i}$, with an assigned value $p_\mathbf{i}$ and $\mathbf{i} \in \mathcal{I}$. Similarly, an structured decision variable x is a jagged array of decision variables symbols $x_\mathbf{i}$, with $\mathbf{i} \in \mathcal{I}$. And lastly, a structured expression is defined just as the flat version, with parameters and variables replaced by their structured versions, but now resting on $O_\mathcal{A}$ enriched with quantifiers $\sum$, $\prod$, among others.

1.3 DISTANCES IN MATHEMATICAL PROGRAMMING

In Mathematics, distance functions stem from the concept of norms, meaning that they are formally defined by means of norms. In Linear Algebra and Functional Analysis, a norm (denoted as $\|\cdot\|$) is a function that assigns a strictly positive length (or size) to each vector x in a normed vector space $(\mathcal{X}, \|\cdot\|)$, except for the zero vector, which is assigned a length of zero. In Chapter 4 we provide a short review of further Linear Algebra concepts which shall be used in this work.

Definition 4. *Given vectors x, y residing in $(\mathcal{X}, \|\cdot\|)$, the distance between x, y values*

$$d(x, y) = \|x - y\|. \tag{4}$$

There exist many different mathematical norms, each one inducing different metrics and topological vector spaces. In this thesis we focus our attention to a particular family of norms, namely the p-norms.

Definition 5. *The p-norm of a vector $x \in \mathcal{X}$ is given by*

$$\|x\|_p = \left(\sum_{i \in N} |x_i|^p \right)^{1/p}. \tag{5}$$

Our interest in p-norms is obvious and twofold: first, we target two theoretical problems that are largely used to model many real-life applications involving distance functions. This means that the n-dimensional Euclidean space (denoted as $\mathbb{R}^n$) is the normed vector space which we will be working with, and the Euclidean norm is exactly the 2-norm; second, we wish to explore the equivalence relation between p-norms in the $\mathbb{R}^n$.

Two norms $\|\cdot\|_\alpha$ and $\|\cdot\|_\beta$ on a vector space $\mathcal{X}$ are said to be equivalent if there exist positive real numbers a and b such that for all $x \in \mathcal{X}$

$$a\|x\|_\alpha \leqslant \|x\|_\beta \leqslant b\|x\|_\alpha.$$

In particular, for the $(1, 2, \infty)$-norms in the $\mathbb{R}^n$, it is true that

$$\left.\begin{array}{l} \|x\|_2 \leqslant \|x\|_1 \leqslant \sqrt{n}\|x\|_2, \\ \|x\|_\infty \leqslant \|x\|_2 \leqslant \sqrt{n}\|x\|_\infty, \\ \|x\|_\infty \leqslant \|x\|_1 \leqslant n\|x\|_\infty. \end{array}\right\} \tag{6}$$

In the context of MP, the features that researchers and practitioners wish the most are certainly the convexity and differentiability of the functions appearing in Eq. (1). Each p-norm induces distance functions with specific characteristics regarding these two properties. For instance, the Euclidean norm defines a convex nonlinear nondifferentiable distance function when plugged into Eq. (4). And nondifferentiability is a serious issue in terms of optimization algorithms. On the other hand, the $(1, \infty)$-norms (also known as the Manhattan and the Maximum norms, respectively) define distance functions which are linearizable in a particular MP setting. Provided that Eq. (6) holds, we can use the $(1, \infty)$-norms to approximate and compute bounds for the Euclidean norm.

Now from a computational standpoint, it is well-know that, when dealing with real numbers, numerical issues originate from the usage of floating point representation in computer systems: there is a natural precision limitation as far as it concerns the representation of the real numbers, particularly of the irrational ones, which prohibits the exact representation of most of them, calls for an ϵ-rounding representation subterfuge for $\epsilon > 0$ and, as a result, causes floating point arithmetic to be error-prone [41]. As a matter of curiosity, to catch a glimpse of this fact, one can run in the terminal of any computer the command

```
$ echo 1.0E45 | awk '{printf "%.0f\n", $1}'.
```

The result depends on the technical standard used, but the outcome in the author's computer (which most likely employs the IEEE 754 Standard [39]) is "9999999999999999297572890245355512199307591 68". All of this means that testing whether two floats (i.e. real number representations in a computer) are equal might result to be false because they differ on digits that actually are not significative to the application in hand. So, in effect, one should always test whether two floats are ϵ-proximal instead.

This problematic naturally extends to floating point arithmetic. So, for instance, is it true that 0.05E45 + 0.95E45 = 1.0E45 under floating point representation? Obviously, it should be, but again the result one gets when running

```
$ echo 0.05E45 0.95E45 | awk '{printf "%.0f\n", $1+$2}'
```

is "1000000000000000088213614053064226407018659840". It is obvious that the order of magnitude here is far beyond what users routinely need in practice, but this can truly affect two key features of MP algorithms: feasibility and convergence. As for feasibility, it is clear that checking whether a solution belongs to a feasible region defined mostly by equality constraints may never result to be true, notedly when the MIP contains integer and continuous decision variables. As for convergence, it is also clear that lower and upper bounds may never converge precisely to the same value.

Altogether, the mathematical properties of the distance function induced by the 2-norm and the use of floating point representation in computer systems pose serious challenges for those willing to devise fast and accurate MP methods for tackling problems that are intimately related to distance computations.

1.3.1 *Applications*

As might be expected, next we introduce four of these problems that put the concept of distance at their core. One of them is explored in this thesis. The others are briefly presented because we judge that they are quite interesting and should be divulged to the readers regardless. But also because, most likely, they will become part of the author's research projects in the future.

1.3.1.1 *Euclidean Distance Geometry Problem*

Our first problem arises from the field of Distance Geometry (DG) [11, 68]. Given a positive integer K and a simple edge-weighted undirected graph $G = (V, E)$, the Euclidean Distance Geometry Problem (EDGP) questions the existence of a vertex realization function $V \to \mathbb{R}^K$ such that each vertex pair adjacent to an edge is placed at a distance which is equal to the edge weight. This problem has many applications to science and engineering, and many methods have been proposed to solve it. Under the action of Euclidean groups, the EDGP contains infinitely many (symmetric) congruent solutions. In Chapter 4 one can find more details about the EDGP and also of the new heuristic approaches that we are proposing to handle this notoriously difficult problem.

1.3.1.2 *Distance Matrix Completion Problem*

A related problem, the Distance Matrix Completion Problem (DMCP), asks whether a partially defined matrix can be completed to a distance matrix D, i.e., completed to a symmetric matrix whose entries D_{ij} represent the distance between the points $i, j \in V$. When the completion is required to be to an Euclidean Distance Matrix (EDM) [25], i. e. where distances are given by 2-norms, this problem is called Euclidean Distance Matrix Completion Problem (EDMCP) [3]. The difference is that while K is part of the input in the EDGP, it is part of the output in the EDMCP.

1.3.1.3 *Euclidean Steiner Tree Problem*

The Euclidean Steiner Tree Problem (ESTP) [59] asks for a shortest possible network interconnecting a given set V of points in the K-dimensional Euclidean space. The ESTP is related to the Minimum Spanning Tree Problem (MSTP), the difference relying in the fact that one is allowed to use extra (not given) points to construct the network in the ESTP; these extra points are known as Steiner points. Indeed, the value of the Steiner ratio, the ratio of the lenght of the Minimum Spanning Tree to the Steiner Minimal Tree for the same set V, conjectured to be $2/\sqrt{3}$, to the best of our knowledge, is still an open question. A very interesting account on the history of the problem can be found in [15].

1.3.1.4 *Geometric Set Covering Problem*

The Geometric Set Covering Problem (GSCP) consists of a broad class of problems that essentially aims to cover a (compact) geometric set by (usually a limited number of) other compact geometric sets. The GSCP is a Set Covering Problem (SCP) in geometric settings. Consider, for instance, the problem of covering cubic lattices with spheres [54], the problem of filling the 3D Euclidean space with tilings (arrangements of polyhedrons) [17] or the problem of covering ellipsoids with spheres [69]. The fact is that to ascertain whether a point $x \in \mathbb{R}^n$ belonging to the target set is covered or not, one needs to verify whether the distance between x and the center of the set that supposedly covers x, for example, lies below a threshold value. And this check is done most often via distance functions.

1.4 SEMIDEFINITE PROGRAMMING

Semidefinite Programming is a relatively new subfield of Convex Optimization [14] concerned with optimization problems defined by a linear objective function subject to the intersection of the cone of Positive Semidefinite (PSD) matrices with an affine space. Simply put, SDP is linear programming over PSD matrices. Actually, it is not hard to see that SDP encompasses LP as a special case.

SDP is considered to be one of the most important developments in MP in the last decades, and is of growing interest for several reasons: first, the existence of polynomial algorithms with efficient implementations that make Semidefinite Programs (SDPs) tractable in many situations; second, the vast list of different and important fields of applications, where SDP has proved to be a useful tool; and third, the beauty and depth of the underlying theory, that links in a natural way different and usually unrelated areas of Mathematics [12].

One of the most important applications of SDP is its use in the formulation of convex relaxations of nonconvex optimization problems. And we shall apply it accordingly in Entr'acte 3 and Chapter 4. In this context, we present the SDP programs in primal and dual standard forms. The problem in primal form can be written as

$$\left.\begin{array}{rl} \min\limits_{X} & A_0 \bullet X \\ \forall i \in \mathcal{I}_I & A_i \bullet X \leqslant b_i, \\ \forall i \in \mathcal{I}_E & A_i \bullet X = b_i, \\ & X \succeq 0, \end{array}\right\} \tag{7}$$

where A_i with $i \in \{0\} \cup \mathcal{I}_E \cup \mathcal{I}_I$ are real symmetric $n \times n$ matrices, b is a real $(|\mathcal{I}_E| + |\mathcal{I}_I|)$-vector and the variable X is a real symmetric $n \times n$ matrix. Notationwise, the Frobenius inner prodcut is denoted by $U \bullet V = \mathrm{tr}(U^\top V) = \sum_j \sum_k u_{jk} v_{jk}$ for any two matrices of the same dimensions and $\mathrm{tr}(U)$ is the trace of U. Moreover, $U \succeq 0$ means that the matrix U is positive semidefinite and $U \succ 0$ means that U is Positive Definite (PD). $U \succeq V$ means that $U - V \succeq 0$; and similarly $U \succ V$ means that $U - V \succ 0$. See Section 4.2 for further details.

Given an SDP in the form of Eq. (7), using the same data and some linear algebra manipulation, we can convert it into a SDP in dual form as

$$\left.\begin{array}{rl} \max\limits_{u} & b^\mathsf{T} u \\ & A_0 - \sum\limits_{i \in \mathcal{I}_E \cup \mathcal{I}_I} u_i A_i \succeq 0, \\ \forall i \in \mathcal{I}_I & u_i \geqslant 0. \end{array}\right\} \tag{8}$$

The dual program (9) has u as decision variables, one for each constraint of the primal program, and thus u is a real $(|\mathcal{I}_E| + |\mathcal{I}_I|)$-vector. The second constraint in (9) is a Linear Matrix Inequality (LMI) on the variables u: a requirement that a matrix depending linearly (or affinely) on some variables be positive semidefinite. It is convenient however to introduce a slack matrix Q and rewrite the problem (9) as

$$\left.\begin{array}{rl} \max\limits_{u,Q} & b^T u \\ & Q = A_0 - \sum\limits_{i \in \mathcal{I}_E \cup \mathcal{I}_I} u_i A_i, \\ \forall i \in \mathcal{I}_I & u_i \geqslant 0, \\ & Q \succeq 0. \end{array}\right\} \tag{9}$$

Here Q is a real symmetric $n \times n$ matrix.

The three most important mathematical properties of SDPs are probably the fact that (a) both forms represent convex Nonlinear Programs (NLPs), that (b) the dual of the dual is the primal and, lastly, that (c) under Slater's constraint qualification, strong duality holds for feasible SDPs. The condition on strong duality requires that there exists a matrix $X \succ 0$ which satisfies the constraints in Eq. (7). In this case, the SDP is said to be strictly feasible.

Many practical problems in engineering and operations research can be modeled or approximated as SDP problems. In fact, it is well-known that SDPs provide tighter relaxations to several combinatorial optimization problems when compared to other methods [88, 91]. Particularly, it suits well (nonconvex) Quadratically Constrained Quadratic Programming (QCQP) problems (i. e. problems which have quadratic constraints and a quadratic objective function) and it is largely used to cope with DG related problems, which are themselves QCQPs. Incidentally, we experiment with SDP and two particular cases of QCQP respectively in Entr'acte 3 and Chapter 4. Moreover, it is widely accepted that Interior-Point Methods (IPMs) [4] are very robust and accurate for solving general SDP programs of moderate size, which implies that SDPs (in both forms) can be solved efficiently in such cases.

However, one notable limitation of SDP for practical purposes is that current technology still does not allow us to scale up to large-scale instance sizes unless we trade accuracy for scaling. For example, folklore says that IPMs (which are second-order methods) for SDPs are supposed to work well up to sizes of "around" 1000 variables, i.e., a matrix variable of around 33×33, which is hardly "large-scale". On top of this, SDP is not applicable to MPs containing integrality constraints. We refer the reader to [5, 88, 91] for further details.

1.4.1 *Diagonally Dominant Programming*

As mentioned previously, one serious drawback of SDP is that current solving technology is limited to instances of fairly low to moderate sizes. Notwithstanding, Ali Ahmadi and Georgina Hall recently remarked [2] that diagonal dominance provides a useful tool for inner approximating the PSD cone.

Definition 6. *A real symmetric matrix* $A = (A_{ij})$ *is Diagonal Dominant (DD) if*

$$A_{ii} \geqslant \sum_{\substack{j \in N \\ j \neq i}} |A_{ij}| \quad \forall i \in N. \tag{10}$$

Using Gershgorin's theorem, one can easily show that all DD matrices are PSD. And that the converse does not hold, hence the inner approximation. See Section 4.2 for further details. This means that

$$\left.\begin{array}{rl} \min\limits_{X} & A_0 \bullet X \\ \forall i \in \mathcal{I}_I & A_i \bullet X \leqslant b_i, \\ \forall i \in \mathcal{I}_E & A_i \bullet X = b_i, \\ & X \text{ is DD}, \end{array}\right\} \tag{11}$$

is a Diagonally Dominant Program (DDP) whose feasible region is a inner approximation of that of Eq. (7).

The crucial observation here is that Eq. (10) is easy to linearize exactly, and so the constraint X is DD in Eq. (11). In order to do so, define a continuous nonnegative symmetric variable $T = (T_{ij})$ as

$$T_{ij} \geqslant |A_{ij}|$$

and observe that, if we sum both the right and left hand sides over all $j \in N$ for a given $i \in N$, the inequality

$$\sum_{\substack{j \in N \\ j \neq i}} T_{ij} \geqslant \sum_{\substack{j \in N \\ j \neq i}} |A_{ij}|$$

holds. We can then reformulate the DD constraint into the system

$$\left.\begin{array}{rl} \forall i \in N & X_{ii} \geqslant \sum\limits_{\substack{j \in N \\ j \neq i}} T_{ij}, \\ \forall i,j \in N & T_{ij} \geqslant X_{ij} \geqslant -T_{ij}, \\ \forall i,j \in N & T_{ij} \geqslant 0, \end{array}\right\} \tag{12}$$

which yields a linear Diagonally Dominant Programming (DDP) formulation when replaced in Eq. (11).

1.4.1.1 *DDP from the dual*

Since Eq. (11) is an inner approximation of Eq. (7), there might conceivably be cases where the feasible region of Eq. (11) is empty while the feasible region of Eq. (7) is nonempty (quite independently of whether the original SDP instance has a solution or not). For such cases, the authors recall that the dual of any SDP is another SDP (and that strong duality holds under Slater's condition). The alternative would be to derive a DDP from the dual SDP as

$$\left.\begin{array}{rl} \max\limits_{u,Q} & b^\top u \\ & Q = A_0 - \sum\limits_{i \in \mathcal{I}_E \cup \mathcal{I}_I} u_i A_i, \\ \forall i \in \mathcal{I}_I & u_i \geqslant 0, \\ & Q \text{ is DD}, \end{array}\right\} \tag{13}$$

and its corresponding LP formulation by means of Eq. (12). Note however that the DDP from the dual can still result infeasible.

1.4.1.2 *Iterative improvement of the DDP formulation*

Furthermore, an iterative method to improve the DDP inner approximation for general SDPs is provided in [2]. For any $n \times n$ matrix U, we have $U^\top U \succeq 0$ since any Gram matrix is PSD. By the same reason, $U^\top Z U \succeq 0$ for any $Z \succeq 0$. This implies that

$$\mathcal{D}(U) = \{U^\top Z U \mid Z \text{ is DD}\} \tag{14}$$

is a subset of the PSD cone. We can therefore replace the constraint X is DD by $X \in \mathcal{D}(U)$ in Eq. (11). Note that this means the LP formulation is now parametrized on U, which offers the opportunity to choose U so as to improve the approximation. More precisely, we define a sequence of DDP formulations:

$$\left.\begin{array}{rl} \min\limits_{X} & A_0 \bullet X \\ \forall i \in \mathcal{I}_I & A_i \bullet X \leqslant b_i, \\ \forall i \in \mathcal{I}_E & A_i \bullet X = b_i, \\ & X \in \mathcal{D}(U^h), \end{array}\right\} \tag{15}$$

for each $h \in \mathbb{N}$, with

$$U^0 = I, \tag{16}$$

$$U^h = \mathsf{factor}(\bar{X}^{h-1}), \tag{17}$$

where $\mathsf{factor}(\cdot)$ indicates a factor of the argument matrix and $\bar{X}^h$ is the solution of Eq. (15) for a given h. Note that the interior of the PSD cone contains PD matrices only, and so the authors suggest using Choleski factors for efficiency.

The iterative method ensures that, for each h, the feasible region of Eq. (15) contains the feasible region for $h-1$. This is easily seen to be the case since, if U^h is a factor of $\bar{X}^{h-1}$, we trivially have $(U^h)^\top I U^h = (U^h)^\top U^h = \bar{X}^{h-1}$, and since I is trivially DD, $\bar{X}^{h-1} \in \mathcal{D}(U^h)$. Moreover, $\bar{X}^{h-1}$ is feasible in Eq. (15), which proves the claim.

The transformation of the constraint $X \in \mathcal{D}(U)$ into a set of linear constraints is also straightforward. $X \in \mathcal{D}(U)$ is equivalent to $X = U^\top Z U \ \wedge\ Z$ is DD, i.e., to

$$\left.\begin{array}{rrcl} & X &=& U^\top Z U, \\ \forall i \in [n+K] & Z_{ii} &\geqslant& \sum\limits_{\substack{j\in[n+K]\\ j\neq i}} T_{ij}, \\ & T &\geqslant& Z \ \geqslant\ -T, \\ & T &\geqslant& 0, \end{array}\right\} \qquad (18)$$

as observed above. Naturally, one may also apply the iterative procedure to the DDP of the dual SDP given by Eq. (13).

Lastly, it is important to point out that Ahmadi and Hall also propose techniques to inner approximate the PSD cone via Second-Order Cone Programming (SOCP). Bottom line, DDP can be described as a technique for obtaining feasible solutions to SDPs via sequences of inner approximating Linear Programs (LPs) or Second-Order Cone Programs (SOCPs). In this thesis we focus on the LP variant, since there exists vastly superior technology for solving large-scale LPs than SDPs or SOCPs.

1.5 THESIS STRUCTURE

This thesis is based mostly on the two Lecture Notes in Computer Science (LNCS) papers [22] and [23], the contents of which are discussed (and somewhat extended) in Chapters 2 and 4. The former exposes our work on the Orbital Independence theory; the latter describes our work on the Euclidean Distance Geometry Problem. In order to give a sense of unity to two otherwise quite different topics, the two main chapters were separated by a further chapter, which we prefer to call *entr'acte*, about Binary Quadratic Programming.

The idea is to create a chain of topics that begins with the study of orbital symmetry in Mathematical Programs (or more precisely Mixed-Integer Linear and Nonlinear Programs) in Chapter 2, then moves on to the analysis of the effects of symmetries in Semidefinite Programming formulations for Binary Quadratic Programs in Entr'acte 3, and finally reaches our work on developing Semidefinite Programming/Diagonally Dominant Programming based methods for the Euclidean Distance Geometry Problem in Chapter 4.

Nevertheless, it is important to urge the reader to consider the difference, in terms of relevance, between the *chapters* and the *entr'acte*. The former are based on refereed publications, and contribute original material. The latter is a light presentation of topics that the author thinks would be worth deeper studies. It has not been submitted for publication yet since its original content is still very limited; it is work in progress.

Part II

CONTRIBUTIONS

2 ORBITAL INDEPENDENCE

It is well-known that symmetric Mathematical Programs are harder to solve to global optimality using Branch-and-Bound type algorithms, since the solution symmetry is reflected in the size of the Branch-and-Bound tree. It is also well-known that some of the solution symmetries are usually evident in the formulation, making it possible to attempt to deal with symmetries as a preprocessing step. One of the easiest approaches is to break symmetries by adjoining some Symmetry-Breaking Constraints to the formulation, thereby removing some symmetric global optima, then solve the reformulation with a generic solver. Sets of such constraints can be generated from each orbit of the action of the symmetries on the variable index set. It is unclear, however, whether and how it is possible to choose two or more separate orbits to generate Symmetry-Breaking Constraints which are compatible with each other (in the sense that they do not make all global optima infeasible). In this chapter we discuss and test a new concept of orbit independence which clarifies this issue.

2.1 INTRODUCTION

An important issue that arises when breaking symmetries of MPs in view of solving them using BB type algorithms is addressed in the following sections. Symmetry-breaking devices are usually derived from orbits of the action of the formulation group on the decision variables. However, one cannot simply use such devices for all the orbits simultaneously: some orbits depend on each other, in a very precise mathematical sense, and hence it may be impossible to use more than one orbit for symmetry-breaking purposes. Next, we discuss a notion of orbit independence which permits to break symmetries from distinct orbits concurrently. Briefly, a short theory describing sufficient conditions is developed and used to devise an algorithm that potentially identifies the largest independent set of orbits of any Mathematical Program. We then provide extensive computational results to showcase the correctness and usefulness of the Orbital Independence (OI) ideas. The tests are performed against symmetric instances taken from the public libraries MIPLIB2010 and MINLPLib2.

The rest of this chapter is organized as follows: in Section 2.2 we introduce notation, recall concepts of Group Theory and review previous work related to symmetries in MP; in Section 2.3 we present all the theoretical developments concerning the OI framework; in Section 2.4 we describe in details the SBCs generator algorithm devised based on the theory recently constructed; and finally, computational experiments are provided and analysed in Section 2.5.

2.2 NOTATION AND PREVIOUS WORK

2.2.1 *Group Theory*

From now on, we consider that groups act on vectors in $\mathbb{R}^n$ by permuting its components and that permutations act on sets of vectors by acting on each vector individually. Standard group nomenclature is employed: S_n and C_n are the symmetric and cyclic group of order n, respectively. $\mathsf{Sym}(\cdot)$ is the symmetric group on the ground set $\cdot$ (e.g. $S_n = \mathsf{Sym}([n])$). Let H and G be groups. If H is a subgroup of G, we write $H \leqslant G$; if H is a normal subgroup of G, then we write $H \lhd G$. Finally, if H is isomorphic to G, we write $H \cong G$. $\langle \Delta \rangle$ denotes the group generated by the set Δ of generators. For a group $G \leqslant S_n$ and a set X of row vectors, $XG = \{xg \mid x \in X \wedge g \in G\}$. A similar definition is valid for a set Y of column vectors: $GY = \{gy \mid y \in Y \wedge g \in G\}$.

2.2.2 *Symmetry detection*

We emphasize that Eq. (1) subsumes the description of *two* mathematical entities: the MP itself, denoted by P, and its formal description P in the language $\mathcal{L}$, which we obtain when replacing x, f, g by their representing symbols x, f, g. It is well-known that P can be parsed into a Directed Acyclic Graph data structure T (an elementary graph contraction of the well-known parsing tree) using a fairly simple context-free grammar [10, 18]. The leaf nodes of T are labelled by constants or decision variable symbols, whereas the other nodes of T are labelled by operator symbols. The incidence structure of T encodes the parent-child relationships between operators, variables and constants. In practice, we can write P using a modelling language such as AMPL [30] and use an unpublished but effective AMPL API to derive T [32]. Since T is a labelled graph, we know how to compute the group $\mathcal{G}$ of its label-invariant isomorphisms (which must also respect a few other properties, such as noncommutativity of certain operators) [64, 65]. In addition, we can use the software codes NAUTY or TRACES [65] to obtain $\mathcal{G}$ and the set Θ of the orbits of its action on the nodes $V(T)$ of the DAG.

2.2.3 *Formulation and solution groups*

Liberti has shown in [49] that:

(a) the action of $\mathcal{G}$ can be projected to the leaf nodes of $V(T)$, which represent decision variables;

(b) this projection induces a group homomorphism ϕ mapping $\mathcal{G}$ to a certain group image G_P;

(c) G_P is a group of permutations of the indices of the variable symbols $x_1, \ldots, x_n$;

(d) G_P is precisely the group of variable permutations of P which keeps $f(x)$ and $\{g_i(x) \mid i \leqslant m\}$ invariant.

In other words, [49] provides (among other things) a practical methodology for computing the *formulation group* G_P of a MP given as in Eq. (1). Since it is not hard to show that G_P is a subgroup of the *solution group* of P, meant as the group of permutations which keeps the set $\mathscr{G}(P)$ of global optima of P invariant, this methodology can be used to extract symmetries from P prior to solving it.

2.2.4 *Symmetry exploitation*

Once the formulation symmetries are known, their most efficient exploitation appears to be their usage within the BB algorithm itself [60, 61, 71, 72]. Such approaches are, unfortunately, difficult to implement, as each solver code must be addressed separately. Their simplest exploitation is the SSB [62, §8.2] which, simply put, consists in adjoining some Symmetry-Breaking Constraints to the original formulation Eq. (1) in the hope of making all but one of the symmetric global optima infeasible. Following the usual trade-off between efficiency and generality, approaches which offer provable guarantees of removing symmetric optima are limited to special structures [40], whereas approaches which hold for any MP in the large class Eq. (1) are mostly common-sense constraints designed to work in general [50]. The consensus seems to be that sets of SBCs are derived from each orbit of the action of G_P on X.

Remark 7. *This is not the only possibility provided that SBCs can also be derived from cyclic subgroups of G_P or single permutations.*

Though breaking symmetries may work well with BB type algorithms, local-search based heuristics usually find optima faster if there are many of them. So it may not always be worth eliminating them [50]. The first attempt to propose a paradigm shift as concerns exploiting symmetries in MPs is Orbital Shrinking (OS) [29]. This recent philosophy sustains that symmetry shall be exploited as much as possible and broken as a last resort. Developed at first to Mixed-Integer Linear Programs (MILPs), the idea behind the technique is to derive relaxations which are at the same time smaller (fewer variables) and symmetry free. This is achieved by aggregating and replacing the original x variables by means of new z variables, defined as

$$\forall \omega \in \Omega \quad z_\omega = \sum_{j \in \omega} x_j,$$

where Ω is the set of orbits of a particular subgroup of the formulation group of P. Simply put, the x variables indexed by the orbit ω are replaced by a single z_ω variable.

The OS paradimg has already unveiled proofs of its promising future [79, 80]. A survey on the subject will become public soon where one will be able to find an extension of the OS method to convex Mixed-Integer Nonlinear Programs (MINLPs) and to nonconvex MINLPs having a special structure.

Finally, we would like to refer the reader to a very recent survey paper [73], which contains an extensive and detailed assessment of the state of the art in symmetry handling methods in Mathematical Programming.

2.2.5 *Orbits*

We recall that an orbit is an equivalence class of the quotient set $X/\sim$, where $i \sim j$ if there is $g \in G_P$ such that $g(i) = j$. This way, G_P partitions X into a set Ω_{G_P} of orbits $\omega_1, \dots, \omega_p$, each of which can be used to generate SBCs. The projection homomorphism ϕ defined above for $\mathcal{G}$ and the leaf nodes of the parsing tree can be restricted to act on G_P and generalized to project its action to any subset $Y \subseteq X$ as follows: for each $\pi \in G_P$ let $\phi(\pi)$ be the product of the cycles of π having all components in Y. We denote by ϕ_Y this generalized action projection homomorphism. The image of ϕ_Y, when Y is some orbit $\omega \in \Omega_{G_P}$, is a group $G_P[\omega]$ called the *transitive constituent* of ω. A group action is *transitive* on a set S if $s \sim t$ for each $s, t \in S$.

2.2.6 *Symmetry-Breaking Constraints*

We borrow the square bracket notation to localize vectors: if $x^* \in \mathscr{G}(P)$ is a global optimum of P, then $x^*[\omega]$ is a projection of x^* on the coordinates indexed by ω. If $G_P[\omega]$ is the full symmetric group $\mathrm{Sym}(\omega)$ on the orbit, it means that $\mathscr{G}(P)$ contains vectors which, when projected onto ω, yield every possible order of $x^*[\omega]$. This implies that we can arbitrarily choose one order, e.g.:

$$\forall \ell < |\omega| \quad x_{\omega(\ell)} \leqslant x_{\omega(\ell+1)}, \tag{19}$$

where $\omega(\ell)$ is the ℓ-th element of ω (stored as a list), enforce this order by means of SBCs, and still be sure that at least one global optimum remains feasible. The SBCs in Eq. (19) are called *strong SBCs*. If $G_P[\omega]$ has any other structure, we observe that, by transitivity of the transitive constituents, at least one permutation in $G_P[\omega]$ will map the component having minimum value in $x^*[\omega]$ to the first component.

Remark 8. *The choice of minimum value and first components are arbitrary. Alternative SBC sets can occur by choosing maximum and/or any other component.*

This, nonetheless, yields the *weak SBCs*:

$$\forall \ell \in \omega \setminus \{\omega(1)\} \quad x_{\omega(1)} \leqslant x_{\omega(\ell)}. \tag{20}$$

Strong SBCs select one order out of $|\omega|!$ many, and hence are able to break all the symmetries in $G_P[\omega]$. Weak SBCs , on the other hand, are unlikely to be able to achieve that. We let $g(x[B]) \leqslant 0$ denote SBCs involving only variables x_j with j in a given set B.

2.2.7 *Stabilizers*

Let $Y \subseteq X$. We recall that the pointwise stabilizer of Y with respect to G_P (or any group G) is defined as the subgroup of elements of G_P fixing each element of Y, i.e., $G^Y = \{g \in G_P \mid \forall y \in Y\ (gy = y)\}$. The setwise stabilizer of Y with respect to G_P is the subgroup of those elements of G_P under which Y is invariant, i.e., $\mathsf{stab}(Y, G_P) = \{g \in G_P \mid \forall y \in Y\ (gy \in Y)\}$. By definition, if Y is an orbit of G_P, then G^Y is the kernel of ϕ_Y and $\mathsf{stab}(Y, G_P) = G_P$.

2.3 ORBITAL INDEPENDENCE NOTIONS

In this section we introduce our main results regarding OI. First we illustrate how SBCs built from different orbits may cut global optima from a MP; then we recall the conditions of OI originally introduced in [49], and finally we present a new concept of OI based on pointwise stabilizers.

2.3.1 *Incompatible SBCs*

In general, one may only adjoin to P the SBCs from *one* orbit. Adjoining SBCs from two or more orbits chosen arbitrarily may result in all global optima being infeasible, as Example 9 shows.

Example 9. *Let* P *be the following MILP:*

$$\min_{x \in \{0,1\}^4} \quad x_1 + x_2 + 2x_3 + 2x_4$$
$$\begin{pmatrix} 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ -1 & 0 & -1 & 0 \\ 0 & -1 & 0 & -1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} \leqslant \begin{pmatrix} 1 \\ 1 \\ -1 \\ -1 \end{pmatrix}.$$

This problem has as set of optima $\mathcal{G}(P) = \{(0,1,1,0),(1,0,0,1)\}$. *In addition, it has formulation group* $G_P = \langle(1\ 2)(3\ 4)\rangle$ *and orbits* $\Omega_{G_P} = \{\omega_1, \omega_2\} = \{\{1,2\},\{3,4\}\}$. *Valid* SBCs *for* ω_1 *(resp.* ω_2*) are* $x_1 \leqslant x_2$ *(resp.* $x_3 \leqslant x_4$*). By simple inspection of the optima set, whereas adjoining either of the two* SBCs *yields valid narrowings, adjoining both simultaneously leads to an infeasible problem.*

Yet, breaking symmetries from only one orbit does not generally make a strong computational impact in MPs of the form Eq. (1). In what follows, we explore the concept of Orbital Independence meant as sufficient conditions to guarantee that SBCs from many orbits preserve at least one global optimum of P feasible.

2.3.2 *Some existing OI conditions*

In order to concurrently combine sets of SBCs generated by two orbits $\omega, \theta \in \Omega_{G_P}$ into a valid narrowing of a MINLP (see Section 1.2.1), two sufficient conditions were provided in [49]:

1. There is a subgroup $H \leqslant G_P[\omega \cup \theta]$ such that $H[\omega] \cong C_{|\omega|}$ and $H[\theta] \cong C_{|\theta|}$;
2. $\mathsf{gcd}(|\omega|, |\theta|) = 1$.

Two orbits with these properties are called *coprime*. Coprime orbits occur relatively rarely in practice [49].

Another set of conditions for OI was hinted at in [55], by means of the following iterative procedure. Initially, one sets $G \leftarrow G_P$ and picks an orbit $\omega \in \Omega_{G_P}$; then adjoins SBCs for ω to P, and then replaces G by G^ω. Termination occurs when G is the trivial group. At each iteration, the SBCs from different orbits can be concurrently adjoined to P. On the other hand, the orbits refer to the action of different groups: G_P initially, then the groups in a normal chain of pointwise stabilizers. In the following, we expand on this idea.

2.3.3 *New conditions for OI*

Our goal now is to introduce the concept of *independent* set of orbits and provide conditions that will help us to identify such sets. These new necessary conditions for OI will be established based on pointwise stabilizers.

First, let $\omega, \theta \in \Omega_{G_P}$. We look at what happens to θ when ω is pointwise stabilized: either G^ω fixes θ, or a subset of θ, or it does not fix any element of θ at all. So three cases follow:

(a) for any subset $\sigma \subseteq \theta$, $\sigma \notin \Omega_{G^\omega}$;

(b) there is a subset $\sigma \subsetneq \theta$ such that $\sigma \in \Omega_{G^\omega}$;

(c) $\theta \in \Omega_{G^\omega}$.

We can thus state the following binary *dependence* relations on the set Ω_{G_P}.

Definition 10. *The orbit θ is dependent of ω, denoted by $\theta \to \omega$, if θ is stabilized when ω is stabilized (case (a) above).*

Definition 11. *The orbit θ is semi-dependent of ω, denoted by $\theta \rightsquigarrow \omega$, if θ splits when ω is stabilized (case (b) above).*

Definition 12. *The orbit θ is independent of ω, denoted by $\theta \Lsh \omega$, if θ is not stabilized when ω is stabilized (case (c) above).*

Next, let Γ^ω be the set of permutations of G_P which move elements of the orbit ω nontrivially. By definition, Γ^ω does not contain the identity permutation e of G_P and thus it is not itself a group. Moreover, the following properties trivially hold: $G^\omega \cap \Gamma^\omega = \varnothing$ and $\mathrm{stab}(\omega, G_P) = G^\omega \cup \Gamma^\omega = G_P$. Moreover:

Lemma 13. *For $\omega \in \Omega_{G_P}$, $G_P[\omega] = \phi_\omega(\Gamma^\omega) \cup \{e\}$.*

Proof. For $\omega \in \Omega_{G_P}$, we have that $G_P[\omega] = \phi_\omega(G_P)$ and that $G_P = G^\omega \cup \Gamma^\omega$; thus, $G_P[\omega] = \phi_\omega(G^\omega \cup \Gamma^\omega)$. From elementary set theory, we also have that $\phi_\omega(G^\omega \cup \Gamma^\omega) = \phi_\omega(G^\omega) \cup \phi_\omega(\Gamma^\omega)$. Now take $\pi \in G_P$. Because $G^\omega \cap \Gamma^\omega = \varnothing$, either $\pi \in G^\omega$ or $\pi \in \Gamma^\omega$. By definition, $\phi_\omega(\pi) = e$ for all $\pi \in G^\omega$; otherwise, $\phi_\omega(\pi) \neq e$ for all $\pi \in \Gamma^\omega$. It thus follows that $G_P[\omega] = \phi_\omega(\Gamma^\omega) \cup \{e\}$. □

Theorem 14 establishes the dependence relation between two orbits $\omega, \theta \in \Omega_{G_P}$ by comparing the sets Γ^ω and Γ^θ.

Theorem 14. *The following statements are true:*

(1) If $\Gamma^\theta = \Gamma^\omega$ then $\theta \to \omega$ and $\omega \to \theta$;

(2) If $\Gamma^\theta \subset \Gamma^\omega$ then $\theta \to \omega$ and either $\omega \Lsh \theta$ or $\omega \rightsquigarrow \theta$;

(3) If $\Gamma^\theta \cap \Gamma^\omega \neq \varnothing$ then ($\theta \Lsh \omega$ or $\theta \rightsquigarrow \omega$) and ($\omega \Lsh \theta$ or $\omega \rightsquigarrow \theta$);

(4) If $\Gamma^\theta \cap \Gamma^\omega = \varnothing$ then $\theta \Lsh \omega$ and $\omega \Lsh \theta$.

Proof. (1) Assume $\Gamma^\theta = \Gamma^\omega$ and consider ω. Then $G^\omega = G_P \setminus \Gamma^\omega \Rightarrow G^\omega \cap \Gamma^\theta = \varnothing \Rightarrow \theta \notin \Omega_{G^\omega}$ and $\theta \to \omega$. Since the same argument holds if we consider θ, we also have $\omega \to \theta$.

(2) Assume $\Gamma^\theta \subset \Gamma^\omega$ and consider ω. Then $G^\omega = G_P \setminus \Gamma^\omega \Rightarrow G^\omega \cap \Gamma^\theta = \varnothing \Rightarrow \theta \notin \Omega_{G^\omega}$ and $\theta \to \omega$. Considering θ, we have that $G^\theta = G_P \setminus \Gamma^\theta \Rightarrow G^\theta \cap \Gamma^\omega \neq \varnothing$. If the action of G^θ is transitive on ω, we have $\omega \Lsh \theta$. Otherwise, we have $\omega \rightsquigarrow \theta$.

(3) Assume $\Gamma^\theta \cap \Gamma^\omega \neq \varnothing$ but neither set is wholly contained in the other, and consider ω. Then $G^\omega = G_P \setminus \Gamma^\omega \Rightarrow G^\omega \cap \Gamma^\theta \neq \varnothing$. If the action of G^ω is transitive on θ, we have $\theta \Lsh \omega$. Otherwise, we have $\theta \rightsquigarrow \omega$. The same argument holds if we consider θ.

(4) Assume $\Gamma^\theta \cap \Gamma^\omega = \varnothing$ and consider ω. Then $G^\omega = G_P \setminus \Gamma^\omega \Rightarrow G^\omega \supset \Gamma^\theta \Rightarrow \theta \in \Omega_{G^\omega}$ and $\theta \mathrel{↳} \omega$. The argument is similar if we consider θ, thus $\omega \mathrel{↳} \theta$.

$\square$

Lemma 15. *The premise $\Gamma^\theta \cap \Gamma^\omega = \varnothing$ to condition (4) in Theorem 14 never holds.*

Proof. Let Δ be the set of generators of G_P. If there is $g \in \Delta$ such that $g[\omega]$ and $g[\theta]$ are nontrivial, then $g \in \Gamma^\theta \cap \Gamma^\omega$. Otherwise, let $\Delta^\theta = \{g \in \Delta \mid g[\omega] = e\}$ and $\Delta^\omega = \{g \in \Delta \mid g[\theta] = e\}$. Because every element of G_P can be expressed as the combination (under the group operation) of finitely many elements of Δ, there is $g \in G_P$ such that $g = g_\omega g_\theta$ where $g_\omega \in \Delta^\omega$ and $g_\theta \in \Delta^\theta$. Thus $g \in \Gamma^\theta \cap \Gamma^\omega$. $\square$

Based on the above definitions and results, the following lemmata hold.

Lemma 16. *The relation $\rightarrow$ is reflexive and the relations $\rightsquigarrow$ and $↳$ are irreflexive.*

Lemma 17. *The relation $\rightarrow$ is symmetric iff $\Gamma^\theta = \Gamma^\omega$ and asymmetric iff $\Gamma^\theta \subset \Gamma^\omega$.*

Lemma 18. *The relation $\rightarrow$ is transitive.*

Proof. Let $\theta, \omega, \tau \in \Omega_{G_P}$ be distinct orbits satisfying $\theta \rightarrow \omega$ and $\omega \rightarrow \tau$. From Theorem 14, $\theta \rightarrow \omega$ implies that either $\Gamma^\theta = \Gamma^\omega$ or $\Gamma^\theta \subset \Gamma^\omega$. Similarly, $\omega \rightarrow \tau$ implies that either $\Gamma^\omega = \Gamma^\tau$ or $\Gamma^\omega \subset \Gamma^\tau$. Then:

(a) $\Gamma^\theta = \Gamma^\omega \wedge \Gamma^\omega = \Gamma^\tau \Rightarrow \Gamma^\theta = \Gamma^\tau \Rightarrow \theta \rightarrow \tau$;

(b) $\Gamma^\theta = \Gamma^\omega \wedge \Gamma^\omega \subset \Gamma^\tau \Rightarrow \Gamma^\theta \subset \Gamma^\tau \Rightarrow \theta \rightarrow \tau$;

(c) $\Gamma^\theta \subset \Gamma^\omega \wedge \Gamma^\omega = \Gamma^\tau \Rightarrow \Gamma^\theta \subset \Gamma^\tau \Rightarrow \theta \rightarrow \tau$;

(d) $\Gamma^\theta \subset \Gamma^\omega \wedge \Gamma^\omega \subset \Gamma^\tau \Rightarrow \Gamma^\theta \subset \Gamma^\tau \Rightarrow \theta \rightarrow \tau$.

$\square$

Whenever the dependence relations are symmetric, we write $\omega \leftrightarrow \theta$ or $\omega \leftrightsquigarrow \theta$ or $\omega \mathrel{↲↳} \theta$. Using this notation, we set forth that:

Definition 19. *Two orbits $\omega, \theta \in \Omega_{G_P}$ are dependent if $\omega \leftrightarrow \theta$, semi-dependent if $\omega \leftrightsquigarrow \theta$ and independent if $\omega \mathrel{↲↳} \theta$.*

Following, we extend the dependence relations presented above to sets of orbits. In this sense, consider a set $\Omega \subseteq \Omega_{G_P}$ and let $\Omega^\omega = \Omega \setminus \omega$ for $\omega \in \Omega$. We look at what happens to ω when the set Ω^ω is pointwise stabilized, i.e., when all the orbits in Ω^ω are (simultaneously) pointwise stabilized. Similar cases to (a)-(c) may occur and suitable definitions can be stated.

Definition 20. *The orbit ω is dependent of Ω^{ω}, denoted by $\omega \hookrightarrow \Omega^{\omega}$, if ω is stabilized when all orbits of Ω^{ω} are stabilized.*

Definition 21. *The orbit ω is semi-dependent of Ω^{ω}, denoted by $\omega \rightsquigarrow \Omega^{\omega}$, if ω splits when all orbits of Ω^{ω} are stabilized.*

Definition 22. *The orbit ω is independent of Ω^{ω}, denoted by $\omega \looparrowright \Omega^{\omega}$, if ω is not stabilized when all orbits of Ω^{ω} are stabilized.*

Lemma 23 establishes necessary conditions to have $\omega \looparrowright \Omega^{\omega}$. The pointwise stabilizer of a set Ω of orbits is denoted as G^{Ω} hereafter.

Lemma 23. *If $\omega \looparrowright \Omega^{\omega}$, then $\omega \mathrel{↳} \theta$ for all $\theta \in \Omega^{\omega}$.*

Proof. By definition, $\omega \looparrowright \Omega^{\omega}$ implies that the action of $G^{\Omega^{\omega}}$ on ω is transitive. Since $G^{\Omega^{\omega}}$ is a subgroup of G^{θ} for every $\theta \in \Omega^{\omega}$, G^{θ} also acts transitively on ω and thus $\omega \mathrel{↳} \theta$. □

Finally we can define an independent set of orbits. We remark that, although we do not state them explicitly, corresponding definitions can be laid down concerning the concepts of dependent and semi-dependent sets of orbits.

Definition 24. *A set Ω is said to be independent if $\omega \looparrowright \Omega^{\omega}$ for all $\omega \in \Omega$.*

Corollary 25 provides necessary conditions so as to a set Ω be independent.

Corollary 25. *If the set Ω is independent, then $\omega \mathrel{↲\!↳} \theta$ for all $\omega, \theta \in \Omega$.*

Proof. By Definition 24 and Lemma 23. □

Example 26 illustrates that the conditions presented in Corollary 25 are not sufficient to guarantee that a set Ω is independent.

Example 26. *Let P be the following MILP:*

$$\min_{x \in \{0,1\}^6} \quad x_1 + x_2 + 2x_3 + 2x_4 + 3x_5 + 3x_6$$

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \end{pmatrix} \leqslant \begin{pmatrix} 1 \\ 1 \\ 1 \\ 2 \\ 2 \\ 2 \\ 2 \end{pmatrix}.$$

$\mathcal{G}(P) = \{(1,0,1,0,0,1),(0,1,0,1,0,1),(0,1,1,0,1,0),(1,0,0,1,1,0)\}$ *contains its optima. It has formulation group $G_P = \langle (1\ 2)(3\ 4), (3\ 4)(5\ 6) \rangle$, which induces the orbits $\Omega_{G_P} = \{\omega_1, \omega_2, \omega_3\} = \{\{1,2\},\{3,4\},\{5,6\}\}$. One can check that the elements in Ω_{G_P} are pairwise independent ($\omega_1 \mathrel{↲\!↳} \omega_2 \wedge \omega_1 \mathrel{↲\!↳} \omega_3 \wedge \omega_2 \mathrel{↲\!↳} \omega_3$); however $\omega_1 \hookrightarrow \{\omega_2, \omega_3\}$ and $\omega_2 \hookrightarrow \{\omega_1, \omega_3\}$ and $\omega_3 \hookrightarrow \{\omega_1, \omega_2\}$.*

2.3.4 *SBCs from independent sets*

Let Ω_I denote an independent set of orbits. Similarly to the results presented in [49], the following propositions set appropriate conditions to build weak and strong SBCs, respectively, from independent sets of orbtis.

Proposition 27. *The constraints* (20) *are SBCs for* P *and* $G^{\Omega_I^\omega}$ *with respect to* $\omega \in \Omega_I$.

Proof. Let $y \in \mathscr{G}(P)$. Since $G^{\Omega_I^\omega}$ acts transitively on ω, there exists $\pi \in G^{\Omega_I^\omega}$ mapping $\min y[\omega]$ to $y_{\omega(1)}$. $\square$

Proposition 28. *Provided that* $G^{\Omega_I^\omega}[\omega] = \mathsf{Sym}(\omega)$, *the constraints* (19) *are SBCs for* P *and* $G^{\Omega_I^\omega}$ *with respect to* $\omega \in \Omega_I$.

Proof. Let $y \in \mathscr{G}(P)$. Since $G^{\Omega_I^\omega}[\omega] = \mathsf{Sym}(\omega)$, there exists $\pi \in G^{\Omega_I^\omega}$ such that $(\pi y)[\omega]$ is ordered by $\leqslant$. Therefore πy is feasible with respect to the contraints (19). $\square$

2.4 ORBITAL INDEPENDENCE ALGORITHM

In this section we describe the methodology used to find an independent set of orbits of a mathematical program. We present how to model and solve the problem of finding such a set by means of a classical combinatorial optimization problem. Moreover, we describe in details the algorithm proposed to build SBCs from all orbits contained in an independent set.

2.4.1 *Independence graph*

Our interest relies in finding the largest $\Omega_I \subseteq \Omega_{G_P}$. Nevertheless, so far we do not have theoretical results providing sufficient conditions to find such a set. Yet we can use the necessary conditions provided by Corollary 25 and search for the largest set $\Omega_K \subseteq \Omega_{G_P}$ whose elements are pairwise independent. Having obtained Ω_K, we can then search for the largest $\Omega_I \subseteq \Omega_K$.

Hence we propose to find Ω_K by encoding the independence relation between orbits of G_P as an undirected graph $G_I = (V, E)$, as of now called the *independence graph* of P, where $V = \Omega_{G_P}$ and E is the set of pairs of independent orbits in Ω_{G_P}. In this manner we reduce the problem of finding Ω_K to the problem of finding the maximum clique in G_I.

2.4.2 *OI reformulations*

We expect that the larger the number of SBCs adjoined to the original formulation, the stronger their computational impact. Particularly, the larger the number of strong SBCs, the better. We thus propose two different reformulations based on the concept of OI: the first prioritizing the total number of SBCs generated and the second prioritizing the total number of strong SBCs generated. In this sense, we look for cliques in G_I that either involve large orbits or involve mostly orbits which may satisfy the conditions to build strong SBCs.

In order to find such cliques, we associate a weight function $w : V \to W$ to $G_I = (V, E, w)$ and solve the Maximum Weight Clique Problem (MWCP) for G_I using the MP formulation described in [13]. In the first reformulation, which we call *orbital independence narrowing*, we have $W = \{|\omega_1|, \dots, |\omega_{|V|}|\}$ and $w(\omega_i) = |\omega_i|$ for all $\omega_i \in V$. In the second, which we call *strong orbital independence narrowing*, $W = \{w_1, w_2\}$. It is worth pointing out that the strong orbital independence narrowing prioritizes cliques having mostly orbits which satisfy $G_P[\omega] = \mathsf{Sym}(\omega)$; this is a necessary condition to have $G^{\Omega_I^\omega}[\omega] = \mathsf{Sym}(\omega)$ since $G^{\Omega_I^\omega}[\omega]$ is a subgroup of $G_P[\omega]$ for every $\omega \in \Omega_I$.

2.4.3 *Algorithm description*

The Algorithm 1 generates a set C containing SBCs derived from the largest independent set of orbits of P. It takes as inputs a nontrivial formulation group (parameter G_P) and a reformulation strategy (parameter σ). The following list of functions simplify the pseudocode of Alg. 1:

- $\mathsf{computeOrbits}(G_P)$ returns the orbits of the group G_P;
- $\mathsf{computePointStab}(\omega)$ returns the pointwise stabilizer of orbit ω;
- $\mathsf{pos}(\omega)$ returns the position of orbit ω in the list Ω_{G_P};
- $\mathsf{isTransitive}(G, \omega)$ returns true if the action of the group G is transitive on the orbit ω and false otherwise;
- $\mathsf{buildGraph}(V, E, \sigma)$ returns a graph with vertices V, edges E and weights appropriate to the strategy σ;
- $\mathsf{solveMWCP}(G_I)$ returns a solution of the MWCP for the graph G_I.

If G_P has more than one orbit ($|\Omega_{G_P}| > 1$), the algorithm first iteratively looks for all the pairs of independent orbits in order to build the set E. Provided that the Condition (3) in Theorem 14 is not sufficient to ascertain whether two orbits $\omega, \theta \in \Omega_{G_P}$ satisfy $\omega \mathrel{\Ldsh\Rdsh} \theta$, ultimately we must check if the action of the stabilizers G^ω and G^θ is transitive on θ and ω, respectively. Thus the algorithm does not compare the sets Γ^ω and Γ^θ but rather directly checks whether the actions are transitive.

Algorithm 1 Orbital Independence SBC generator

```
Require: nontrivial $G_P$ and reformulation strategy $\sigma$
 1: Let $C = \varnothing$ and $\Omega_I = \varnothing$
 2: Let $\Omega_{G_P} = \mathsf{computeOrbits}(G_P)$
 3: if $|\Omega_{G_P}| > 1$ then
 4:   for $\omega \in \Omega_{G_P}$ do
 5:     Let $G^\omega = \mathsf{computePointStab}(\omega)$
 6:     for $\theta \in \Omega_{G_P}$ such that $\mathsf{pos}(\theta) > \mathsf{pos}(\omega)$ do
 7:       Let $G^\theta = \mathsf{computePointStab}(\theta)$
 8:       if $\mathsf{isTransitive}(G^\omega, \theta) \wedge \mathsf{isTransitive}(G^\theta, \omega)$ then
 9:         Let $E = E \cup \{\{\omega, \theta\}, \{\theta, \omega\}\}$
10:       end if
11:     end for
12:   end for
13:   if $|E| \geqslant 2$ then
14:     Let $G_I = \mathsf{buildGraph}(\Omega_{G_P}, E, \sigma)$
15:     Let $\Omega_K = \Omega_I = \mathsf{solveMWCP}(G_I)$
16:     for $\omega \in \Omega_K$ do
17:       if not $\mathsf{isTransitive}(G^{\Omega_I^\omega}, \omega)$ then
18:         Let $\Omega_I = \Omega_I \setminus \omega$
19:       end if
20:     end for
21:     for $\omega \in \Omega_I$ do
22:       Let $g(x[\omega]) \leqslant 0$ be some SBCs for $P$ and $G^{\Omega_I^\omega}$ w.r.t. $\omega$
23:       Let $C = C \cup \{g(x[\omega]) \leqslant 0\}$
24:     end for
25:   end if
26: end if
27: return $C$
```

Following the first loop, if at least one pair of independent orbits is found ($|E| \geqslant 2$), the algorithm builds the independence graph G_I according to the reformulation strategy σ and calls a third party MILP solver to solve the MWCP for G_I. Once Ω_K is known, the algorithm converges to a set Ω_I by iteratively removing (from a copy of Ω_K stored as Ω_I) the orbits that do not satisfy $\omega \not\leftrightarrow \Omega_I^{\omega}$.

Remark 29. *Our approach here is not optimal in the sense that the resulting Ω_I may not be the largest one; the point is that evaluating all possible $\Omega_I \subseteq \Omega_K$ would most likely require a huge computational effort owing to many stabilizer computations.*

Then, for each orbit in the set Ω_I, the algorithm builds and adds SBCs to the set C. It is important to emphasize that if $|\Omega_{G_P}| = 1$ (unique orbit) or $|E| = 0$ (no pair of independent orbits in Ω_{G_P}), no reformulation is carried out.

Theorem 30. *The constraint set $C_{\Omega_I} = \{g(x[\omega_k]) \leqslant 0 \mid \omega_k \in \Omega_I\}$ is an SBC system for P.*

Proof. If P is infeasible then adjoining the constraints in C_{Ω_I} to P does not change its infeasibility, so assume P is feasible. Since $g(x[\omega_k]) \leqslant 0$ are SBCs for P and $G^{\Omega_I^{\omega_k}}$ with respect to ω_k, there exist $y \in \mathscr{G}(P)$ and $\pi_{\omega_k} \in G^{\Omega_I^{\omega_k}}$ such that $\pi_{\omega_k} y$ satisfies $g((\pi_{\omega_k} y)[\omega_k]) \leqslant 0$. But $\pi_{\omega_k} \in G_P$ for all $\omega_k \in \Omega_I$ and, due to the closure of the group operation, there exists $\pi \in G_P$ such that $\pi = \prod \pi_{\omega_k}$. So $\pi y \in \mathscr{G}(P)$. But $\pi[\omega_k] = \pi_{\omega_k}[\omega_k]$ since $\pi_{\omega_{k'}}$ stabilizes ω_k pointwise for every $k' \neq k$ and thus $(\pi y)[\omega_k] = (\pi_{\omega_k} y)[\omega_k]$. Therefore πy satisfies $g((\pi y)[\omega_k]) \leqslant 0$ for all $\omega_k \in \Omega_I$. □

2.5 COMPUTATIONAL EXPERIMENTS

In this section we show the computational impact on the resolution of general MILPs and MINLPs when adjoining SBCs arising from different orbits simultaneously. We describe the computational environment involved (the instances, machinery, solvers, setups, procedures, etc) and analyze the results obtained from the conducted experiments.

2.5.1 *Datasets*

Our test bed consists of two groups of instances, all of them taken from public libraries. The first group is comprised of symmetric MILPs found in MIPLIB2010, and the second group is comprised of symmetric MINLPs found in MINLPLib2. We have submitted all instances constituting these two libraries to Algorithm 2 (see below) and we have kept only those exhibiting at least one pair of independent orbits ($|\Omega_K| \geqslant 2$). As a result, we have found 89 instances in total, 47 from

MIPLIB2010 and 42 from MINLPLib2. We have tested François Margot's instances as well (see `wpweb2.tepper.cmu.edu/fmargot/lpsym.html`), but none of them has a single pair of independent orbits ($\Omega_K = \varnothing$). Again, we allude to [73] for a detailed description of the presence of symmetries in public MP instances.

2.5.2 *Environment*

The reformulations were obtained on a 4-CPU Intel Xeon at 2.66GHz with 24Gb RAM. Automatic group detection is carried out using the ROSE reformulator [51] and the software TRACES [65]. Other group computations are carried out using GAP v. 4.7.4 [87]. The MP results were obtained on a 24-CPU Intel Xeon at 2.53GHz with 48Gb RAM. All problems were solved under the AMPL [30] environment. CPLEX 12.6 [37] was used to solve the $\mathbb{MILP}$s and SCIP 3.0.1 [1] to solve the $\mathbb{MINLP}$s.

The computation time was limited to 7200 seconds of user cpu time. In order to try and provide a fair assessment of our methodology, we disabled the symmetry handling methods built into CPLEX. We also ran CPLEX in single thread mode to impose its sequential (and deterministic) behaviour and increase the chances of measuring performance differences. Unfortunately we were not able to disable SCIP's symmetry handling features as well since, so far, to the best of our knowledge, its API or CLI does not provide a setup parameter to switch off this feature.

2.5.3 *Reformulation algorithm*

Currently, given an ASCII file containing a problem $P \in \mathbb{MP}$ written in AMPL, the overall SSB procedure which we employ to detect and exploit the symmetries of P is depicted in Algorithm 2.

Algorithm 2 Orbital Independence reformulator

```
Require: $P$
1: Parse $P$'s AMPL model into its DAG representation;
2: Compute the generators of the DAG's automorphism group;
3: Use the projection operator to obtain the generators of $G_P$;
4: Run Algorithm 1 to obtain the set $C$;
5: Reformulate $P$ into $P'$ by adjoining $C$ to $P$.
6: return $P'$
```

As mentioned in Section 1.2, we can group the steps of Alg. 2 into the two main phases: steps 1 to 3 (symmetry detection) and steps 4 to 5 (symmetry exploitation). Step 1 is performed by ROSE; step 2 is performed by TRACES; steps 3 and 4 are performed by GAP, and the whole process, which naturally includes step 5, is coordinated by a shell script code. Note that the output of Alg. 2 is another ASCII file containing an AMPL model corresponding to P'.

2.5.4 *Results*

We first comment the results regarding the reformulation process. Tables 1 and 2 report, per instance, the number of variables (n) and orbits ($|\Omega_{G_P}|$) of the original formulation, and the total number of variables indexed by the orbits Ω_{G_P} (#svar); for each OI narrowing type, the Tables report the size of the maximum clique ($|\Omega_K|$), the size of the largest independent set ($|\Omega_I|$), the total number of variables indexed by all the orbits in Ω_I (#var), the number of weak (#wea) and strong (#str) SBCs generated, and the parameters σ, ρ and υ, which are described in the sequel.

We would like to remark that both reformulation strategies yielded the same narrowings for the most part of the instances. In these cases, we do not present results concerning the strong orbital independence reformulation. Additionally, we also point out that the size of the maximum cliques is equal to the size of the largest independent sets for all instances.

Apart from the structure of the group G_P, intuitively, the ratio $\sigma = (\#svar/n)$ may also indicate how symmetric a formulation P is. Similarly, the ratios $\rho = (|\Omega_I|/|\Omega_{G_P}|)$ and $\upsilon = (\#var/\#svar)$ may indicate how extensively we have exploited the symmetries of P. All together, we expect SBCs to make a strong computational impact whenever the triplet (σ, ρ, υ) tends to $(1, 1, 1)$. Both Tables 1 and 2 show that the generic symmetric instances tested so far have, in general, at most two high ratios, which suggests that the impact of the SBCs may not be too significative. Indeed, we can easily recognize three patterns in which the majority of the instances fit into: either the instance is highly symmetric ($\sigma \approx 1$) and we cannot explore much of its symmetries $(\rho, \upsilon) \approx (0, 0)$, or the instance does not exhibit many symmetries ($\sigma \approx 0$) and we explore almost all of them $(\rho, \upsilon) \approx (1, 1)$, or eventually $(\sigma, \rho, \upsilon) \approx (0, 0, 0)$. The few exceptions (among 89 cases) being instances ex9_2_6, hmittelman, lop97icx and st_rv9.

Table 3 provides aggregated solution statistics. Per dataset and for each formulation, the table reports the number of best performances and the total time comsumed in hours to solve all instances. We do not report in this table the cases where no method performed better than the other. There were only 9 ties for the MIPLIB2010 dataset but 19 (almost half of the tested nonlinear instances) for the MINLPLib2. From this point of view, the statistics are somewhat more expressive regarding the MIPLIB2010 library, most likely because we could, as mentioned previously, switch off CPLEX's internal symmetry handling procedures during the experiments and thus adequately detect the computational impact of the SBCs on this particular dataset.

	Original formulation			OI-narrowing							
Instance	n	$\lvert\Omega_{G_P}\rvert$	#svar	$\lvert\Omega_K\rvert$	$\lvert\Omega_I\rvert$	#var	#wea	#str	σ	ρ	υ
bab5	21600	1936	3872	4	4	8	0	4	.17	0*	0*
blp-ar98	16017	2	4	2	2	4	0	2	0*	1.00	1.00
blp-ic97	8445	2	4	2	2	4	0	2	0*	1.00	1.00
core2536-691	15288	88	187	12	12	29	3	14	.01	.13	.15
core4872-1529	24605	505	1046	46	46	96	0	50	.04	.09	.09
gmu-35-40	842	40	111	4	4	13	0	9	.13	.10	.11
gmu-35-50	1177	40	111	4	4	13	0	9	.09	.10	.11
gmut-75-50	36164	64	242	6	6	19	0	13	0*	.09	.07
gmut-77-40	13140	70	280	7	7	26	0	19	.02	.10	.09
iis-bupa-cov	345	2	7	2	2	7	0	5	.02	1.00	1.00
lectsched-4-obj	3513	267	557	17	17	36	0	19	.15	.06	.06
macrophage	2260	251	566	18	18	42	5	19	.25	.07	.07
map06	46015	107	245	10	10	20	0	10	0*	.09	.08
map10	46015	107	245	10	10	20	0	10	0*	.09	.08
map14	46015	107	245	10	10	20	0	10	0*	.09	.08
map18	46015	107	245	10	10	20	0	10	0*	.09	.08
map20	46015	107	245	10	10	20	0	10	0*	.09	.08
mcsched	1669	45	90	15	15	30	0	15	.05	.33	.33
mzzv11	10240	155	310	16	16	32	0	16	.03	.10	.10
neos-1311124	1092	52	1092	4	4	84	0	80	1.00	.07	.07
neos-1426635	520	52	520	4	4	40	0	36	1.00	.07	.07
neos-1426662	832	52	832	4	4	64	0	60	1.00	.07	.07
neos-1436709	676	52	676	4	4	52	0	48	1.00	.07	.07
neos-1440460	468	52	468	4	4	36	0	32	1.00	.07	.07
neos-1442119	728	52	728	4	4	56	0	52	1.00	.07	.07
neos-1442657	624	52	624	4	4	48	0	44	1.00	.07	.07
neos-555424	3815	132	3810	8	8	190	107	75	.99	.06	.04
neos-826841	5516	156	5436	3	3	200	191	6	.98	.01	.03
neos-849702	1737	128	1737	2	2	36	34	0	1.00	.01	.02
neos-911880	888	259	888	7	7	24	0	17	1.00	.02	.02
neos-952987	31329	37	81	4	4	8	0	4	0*	.10	.09
neos18	963	53	248	5	5	26	0	21	.25	.09	.10
ns1631475	22696	105	210	11	11	22	0	11	0*	.10	.10
ns2081729	661	300	600	3	3	6	0	3	.90	.01	.01
p2m2p1m1p0n100	100	25	92	3	3	12	0	9	.92	.12	.13
protfold	1835	558	1800	2	2	4	0	2	.98	0*	0*
rocII-4-11	3409	2	27	2	2	27	0	25	0*	1.00	1.00
rococoC10-001000	2566	41	82	4	4	8	0	4	.03	.09	.09
rvb-sub	33765	113	226	12	12	24	0	12	0*	.10	.10
satellites1-25	9013	200	400	20	20	40	0	20	.04	.10	.10
seymour-disj-10	1209	49	106	5	5	12	0	7	.08	.10	.11
seymour	1255	55	156	5	5	41	29	7	.12	.09	.26
swath	6404	21	163	2	2	8	0	6	.02	.09	.04
transportmoment	9099	85	189	17	17	38	0	21	.02	.20	.20
toll-like	2883	386	1091	26	26	91	44	21	.37	.06	.08
uc-case3	36921	2687	5374	2	2	4	0	2	.14	0*	0*
uct-subprob	2236	136	306	7	7	14	0	7	.13	.05	.04

	Original formulation			SOI-narrowing							
Instance	n	$\lvert\Omega_{G_P}\rvert$	#svar	$\lvert\Omega_K\rvert$	$\lvert\Omega_I\rvert$	#var	#wea	#str	σ	ρ	υ
core2536-691	15288	88	187	12	12	27	0	15	.01	.13	.14
macrophage	2260	251	566	18	18	39	0	21	.25	.07	.06
neos-555424	3815	132	3810	8	8	145	58	79	.99	.06	.03
neos-826841	5516	156	5436	4	4	46	0	42	.98	.02	0*
neos-849702	1737	128	1737	2	2	9	0	7	1.00	.01	0*
toll-like	2883	386	1091	26	26	59	0	33	.37	.06	.05

Table 1: OI-narrowings of symmetric instances from MIPLIB2010. 0* indicates values of $O(10^{-3})$ or less.

	Original formulation			OI-narrowing							
Instance	n	$\lvert\Omega_{G_P}\rvert$	#svar	$\lvert\Omega_K\rvert$	$\lvert\Omega_I\rvert$	#var	#wea	#str	σ	ρ	υ
arki0002	2456	384	2304	2	2	12	0	10	.93	0*	0*
arki0005	2370	9	18	9	9	18	0	9	0*	1.00	1.00
arki0006	2370	9	18	9	9	18	0	9	0*	1.00	1.00
autocorr_bern25-03	26	12	24	2	2	4	0	2	.92	.16	.16
carton7	230	49	162	3	3	13	8	2	.70	.06	.08
carton9	266	83	266	3	3	13	8	2	1.00	.03	.04
cecil_13	733	18	36	9	9	18	0	9	.04	.50	.50
chp_partload	2080	82	164	5	5	10	0	5	.07	.06	.06
crudeoil_li21	1236	134	268	2	2	4	0	2	.21	.01	.01
ex9_2_6	16	7	16	2	2	6	3	1	1.00	.28	.37
gastrans	89	6	12	2	2	4	0	2	.13	.33	.33
hmittelman	16	3	6	3	3	6	0	3	.37	1.00	1.00
kport20	98	25	55	5	5	11	0	6	.56	.20	.20
kport40	217	48	150	8	8	28	0	20	.69	.16	.18
lop97ic	1626	3	127	3	3	127	0	124	.07	1.00	1.00
lop97icx	986	8	777	8	8	777	0	769	.78	1.00	1.00
mbtd	210	61	210	2	2	12	9	1	1.00	.03	.05
netmod_kar1	456	48	132	3	3	9	0	6	.28	.06	.06
netmod_kar2	456	48	132	3	3	9	0	6	.28	.06	.06
powerflow2383wpr	15882	12	24	3	3	6	0	3	0*	.25	.25
powerflow2383wpp	15882	12	24	3	3	6	0	3	0*	.25	.25
risk2bpb	434	12	72	12	12	72	0	60	.16	1.00	1.00
routingdelay_bigm	1115	18	36	12	12	24	0	12	.03	.66	.66
routingdelay_proj	1115	18	36	12	12	24	0	12	.03	.66	.66
sepasequ_complex	485	5	27	5	5	27	9	13	.05	1.00	1.00
st_rv9	50	10	20	10	10	20	0	10	.40	1.00	1.00
super1	1263	12	26	12	12	26	0	14	.02	1.00	1.00
super2	1274	11	24	11	11	24	0	13	.01	1.00	1.00
super3	1281	11	24	11	11	24	0	13	.01	1.00	1.00
super3t	1032	11	24	11	11	24	0	13	.02	1.00	1.00
syn15m	55	2	5	2	2	5	0	3	.09	1.00	1.00
torsion100	5004	2	4	2	2	4	0	2	0*	1.00	1.00
torsion25	1254	2	4	2	2	4	0	2	0*	1.00	1.00
torsion50	2504	1227	2454	3	3	6	0	3	.98	0*	0*
torsion75	3754	2	4	2	2	4	0	2	0*	1.00	1.00
transswitch2383wpr	18768	15	30	3	3	6	0	3	0*	.20	.20
transswitch2383wpp	18768	15	30	3	3	6	0	3	0*	.20	.20
turkey	512	4	8	4	4	8	0	4	.01	1.00	1.00
unitcommit1	738	2	30	2	2	30	0	28	.04	1.00	1.00
unitcommit2	738	2	30	2	2	30	0	28	.04	1.00	1.00
waste	1425	30	76	15	15	38	0	23	.05	.50	.50
waterund28	760	106	216	2	2	4	0	2	.28	.01	.01

	Original formulation			SOI-narrowing							
Instance	n	$\lvert\Omega_{G_P}\rvert$	#svar	$\lvert\Omega_K\rvert$	$\lvert\Omega_I\rvert$	#var	#wea	#str	σ	ρ	υ
carton7	230	49	162	3	3	8	0	5	.70	.06	.04
carton9	266	83	266	3	3	8	0	5	1.00	.03	.03

Table 2: OI-narrowings of symmetric instances from MINLPLib2. 0* indicates values of $O(10^{-3})$ or less.

	Original formulation		OI-narrowing		SOI-narrowing	
Dataset	# Best	Time (h)	# Best	Time (h)	# Best	Time (h)
MIPLIB2010	16	49.52	19	48.16	3	48.15
MINLPLib2	8	52.00	11	51.3	2	51.29
Total	24	101.52	30	99.46	5	99.44

Table 3: Aggregated solution statistics for the public datasets MIPLIB2010 and MINLPLib2.

Finally, Tables 4 and 5 report details of the optimization results. Per instance and for each formulation, the table exhibits the best solution found, the user cpu time (in seconds), the gap (%) and the solver status at termination (opt = optimum found, lim = time limit reached, inf = infeasible instance). Best values are emphasized in boldface. Two intances (namely powerflow2383wpp and transswitch2383wpp) from Table 2 do not appear in Table 5 because we could not solve them with SCIP 3.0.1 in view of the presence of unsupported AMPL operands in their formulations (SCIP error message: "AMPL operand number 46 not supported so far").

As expected, we do not observe cases of infeasible narrowings on account of the usage of SBCs derived from distinct orbits simultaneously. It is important to highlight that the SBCs reduced the total computation time of the OI-narrowings in about 2 hours when compared to the total computation time of the original formulations. We observe small but consistent improvements in favor of the Orbital Independence narrowings. In 35 out of 87 instances, the SBCs slightly helped to improve the performance of the solvers. On the other hand, in 24 cases, the SBCs were harmful and in 28 other instances, they made no difference at all. Although they provided good results, the few SOI-narrowings did not achieve outstanding performances. As regards the four instances that are closer to satisfy $(\sigma, \rho, \upsilon) \approx (1, 1, 1)$, except for ex9_2_6, the narrowings performed better, as expected. Interestingly, the SBCs were particularly harmful to instances of two families, map# among the MILPs and torsion# among the MINLPs. We shall investigate the reasons behind these results in order to gain more insights into the detrimental impact of SBCs.

But we can conjecture why SBCs are occasionally harmful: BB type algorithms are complex systems whose performance depend on many factors (LP solutions, branching policies, cut generation schemes and so on). We expect the effect of SBCs to be, theoretically, the reduction of the number of symmetric subtrees during the BB search. Since there is no guarantee that this reduction will actually happen, failure is a possibility. Yet their presence may change LPs solutions computed in the nodes of the BB tree. In this sense, SBCs can also unduly impact on branching policies and on cut generation schemes, since LP solutions are the most important inputs for these two key features. However,

	Original formulation				OI-narrowing			
Instance	Best	Time (s)	Gap (%)	St.	Best	Time (s)	Gap (%)	St.
bab5	-106412	3879.57	0	opt	-106412	**930.53**	0	opt
blp-ar98	6205.21	**1319.87**	0	opt	6205.21	3736.05	0	opt
blp-ic97	4025.02	5517.71	0	opt	4025.02	**3122.49**	0	opt
core4872-1529	1459	7200.11	**1.70**	lim	1467	7200.14	2.21	lim
gmu-35-40	-2406600	63.59	0	opt	-2406600	**53.12**	0	opt
gmu-35-50	-2607780	7208.25	0.01	lim	-2607780	7209.47	0.01	lim
gmut-75-50	-14178800	7200.61	0.01	lim	-14179300	6597.76	0	**opt**
gmut-77-40	-14170700	**3439.64**	0	opt	-14170700	3582.83	0	opt
iis-bupa-cov	36	7200.08	4.34	lim	36	6017.99	0	**opt**
lectsched-4-obj	4	9.10	0	opt	4	**8.05**	0	opt
map06	-289	**705.17**	0	opt	-289	806.19	0	opt
map10	-495	**616.85**	0	opt	-495	699.47	0	opt
map14	-674	684.74	0	opt	-674	**664.73**	0	opt
map18	-847	**322.72**	0	opt	-847	328.57	0	opt
map20	-922	**147.98**	0	opt	-922	169.07	0	opt
mcsched	211913	**317.08**	0	opt	211913	364.21	0	opt
mzzv11	-21718	**20.27**	0	opt	-21718	34.15	0	opt
neos-1311124	-181	7200.79	0.55	lim	-181	7200.49	0.55	lim
neos-1426635	-176	7201.01	1.14	lim	-176	7200.77	**0.57**	lim
neos-1426662	-44	7200.67	14.40	lim	-44	7201.45	**12.94**	lim
neos-1436709	-128	7200.38	0.78	lim	-128	7200.40	0.78	lim
neos-1440460	-179.25	7200.59	0.42	lim	-179.25	7200.37	**0.05**	lim
neos-1442119	-181	7200.35	0.55	lim	-181	7200.34	0.55	lim
neos-1442657	-154.5	7200.59	0.97	lim	-154.5	7200.40	0.97	lim
neos-911880	54.76	7.61	0	opt	54.76	**7.12**	0	opt
neos-952987	∞	7200.37	∞	lim	∞	7200.32	∞	lim
neos18	13	25.43	0	opt	13	**16.00**	0	opt
ns1631475	21450	7200.10	**91.03**	lim	∞	7200.06	∞	lim
ns2081729	9	**391.90**	0	opt	9	815.13	0	opt
p2m2p1m1p0n100	0	0.00	0	opt	0	0.00	0	opt
protfold	-26	7200.04	37.68	lim	-27	7200.04	**32.19**	lim
rocII-4-11	-5.65564	400.50	0	opt	-5.65564	**385.86**	0	opt
rococoC10-001000	11460	140.67	0	opt	11460	**138.33**	0	opt
rvb-sub	27.4683	7200.48	58.58	lim	27.4683	7200.39	**58.56**	lim
satellites1-25	-5	**191.84**	0	opt	-5	421.96	0	opt
seymour-disj-10	287	7200.08	**1.22**	lim	288	7200.10	1.56	lim
seymour	306	7200.11	**1.35**	lim	307	7200.11	1.69	lim
swath	467.408	7200.37	10.14	lim	467.408	7200.42	**9.95**	lim
transportmoment	∞	2.68	∞	inf	∞	**2.50**	∞	inf
uc-case3	6931.2	7200.36	0.05	lim	6931.2	7200.38	0.05	lim
uct-subprob	315	7200.20	**3.29**	lim	315	7200.21	5.02	lim

	Original formulation				OI-narrowing				SOI-narrowing			
Instance	Best	Time (s)	Gap (%)	St.	Best	Time (s)	Gap (%)	St.	Best	Time (s)	Gap (%)	St.
core2536-691	683	56.90	0	opt	683	65.38	0	opt	683	**50.90**	0	opt
macrophage	374	868.50	0	opt	374	372.68	0	opt	374	**271.38**	0	opt
neos-555424	1286800	**5.76**	0	opt	1286800	6.79	0	opt	1286800	6.55	0	opt
neos-826841	29.0082	7200.13	3.45	lim	29.0082	7200.18	3.45	lim	29.0082	7200.14	3.45	lim
neos-849702	0	730.88	0	opt	0	**8.65**	0	opt	0	90.03	0	opt
toll-like	614	7200.07	18.52	lim	611	7200.08	17.02	lim	613	7200.07	**16.76**	lim

Table 4: MIPLIB2010 results obtained with CPLEX 12.6.

	Original formulation				OI-narrowing			
Instance	Best	Time (s)	Gap (%)	St.	Best	Time (s)	Gap (%)	St.
arki0002	12.9734	7200.07	∞^*	lim	12.9734	7200.06	∞^*	lim
arki0005	10434.4	7200.31	∞^*	lim	10434.4	7200.14	∞^*	lim
arki0006	116.815	7200.22	∞^*	lim	116.815	7200.54	∞^*	lim
autocorr_bern25-03	-92	0.02	0	opt	-92	0.02	0	opt
cecil_13	-115656	7304.46	2.66	lim	-115656	7300.49	2.66	lim
chp_partload	∞	7202.05	∞	lim	∞	7202.55	∞	lim
crudeoil_li21	∞	7203.18	∞	lim	∞	7202.59	∞	lim
ex9_2_6	-1	0.02	0	opt	-1	0.02	0	opt
gastrans	89.0858	0.15	0	opt	89.0858	**0.11**	0	opt
hmittelman	13	0.03	0	opt	13	**0.02**	0	opt
kport20	26.9093	6489.09	0	opt	26.9093	**4002.44**	0	opt
kport40	**32.3661**	7443.03	35.35	lim	32.5547	7436.03	35.09	lim
lop97ic	4973.17	7200.83	94.03	lim	4830.18	7201.11	**88.70**	lim
lop97icx	4306	7208.88	49.98	lim	4323.03	7205.98	**46.45**	lim
mbtd	10.1668	7201.70	306.67	lim	8.50003	7206.50	**240.00**	lim
netmod_kar1	-0.419789	10.51	0	opt	-0.419789	**7.76**	0	opt
netmod_kar2	-0.419789	10.59	0	opt	-0.419789	**7.79**	0	opt
powerflow2383wpr	∞	7204.88	∞	lim	∞	7204.91	∞	lim
risk2bpb	-55.8761	**0.11**	0	opt	-55.8761	0.15	0	opt
routingdelay_bigm	146.626	**12.27**	0	opt	146.626	13.76	0	opt
routingdelay_proj	∞	7201.98	∞	lim	∞	7201.87	∞	lim
sepasequ_complex	578.744	7229.26	106.34	lim	492.415	7235.63	**70.75**	lim
st_rv9	-120.153	0.32	0	opt	-120.153	**0.22**	0	opt
super1	∞	7211.69	∞	lim	∞	7209.83	∞	lim
super2	∞	7209.98	∞	lim	∞	7212.45	∞	lim
super3	∞	7224.12	∞	lim	∞	7209.58	∞	lim
super3t	∞	7209.06	∞	lim	∞	7208.77	∞	lim
syn15m	-853.283	0.16	0	opt	-853.283	0.16	0	opt
torsion100	-0.337993	7200.57	**39539.62**	lim	0	7200.67	∞^*	lim
torsion25	-0.341737	7200.11	**10069.16**	lim	0	7200.13	∞^*	lim
torsion50	-0.333206	7200.32	**16362.80**	lim	0	7200.32	∞^*	lim
torsion75	-0.338488	7200.61	**25863.95**	lim	0	7200.44	∞^*	lim
transswitch2383wpr	∞	7205.02	∞	lim	∞	7205.78	∞	lim
turkey	1766.82	7200.18	∞^*	lim	1766.82	7200.20	∞^*	lim
unitcommit1	578177	2.79	0	opt	578177	**2.66**	0	opt
unitcommit2	578177	**6.89**	0	opt	578177	7.62	0	opt
waste	609.134	7216.32	101.88	lim	609.134	7217.47	101.88	lim
waterund28	∞	7200.37	∞	lim	∞	7200.84	∞	lim

	Original formulation				OI-narrowing				SOI-narrowing			
Instance	Best	Time (s)	Gap (%)	St.	Best	Time (s)	Gap (%)	St.	Best	Time (s)	Gap (%)	St.
carton7	191.73	98.15	0	opt	191.73	92.05	0	opt	191.73	**84.58**	0	opt
carton9	205.137	103.60	0	opt	205.137	97.24	0	opt	205.137	**42.99**	0	opt

Table 5: MINLPLib2 results obtained with SCIP 3.0.1.

since there are elements of arbitrary choice regarding the generation of SBCs, we suspect that forcing these choices may, in some cases, prevent the BB algorithm to take the correct decisions.

Overall, we understand that the results are at most reasonable, but they support our motivation and encourage a more extensive experimental evaluation against a larger set of instances that exhibit nontrivial symmetries. In this sense, we will investigate a particular class of symmetric Quadratically Constrained Quadratic Programs that favors the OI framework and yields interesting outcomes in the following entr'acte. Conversely, the results presented so far may also be an evidence that we have reached the limit of what we can do in terms of SSB, since we are exploiting as much as we can, but not getting expressive results for the general case. Perhaps this is not the most impactful way to pursue in general, and so it is time to try and exploit the OI ideas dynamically.

2.6 conclusions

In this chapter we have discussed the notion of Orbital Independence by presenting theoretical results that establish sufficient conditions to break symmetries from different orbits of Mathematical Programs concurrently: we have defined the concept of an independent set of orbits. These conditions were employed in the design of an algorithm that potentially identifies the largest independent set of orbits of a $\mathbb{MP}$ and generates SBCs to all orbits of this set. We have evaluated the impact of our methodology by conducting experiments with symmetric instances taken from the libraries MIPLIB2010 and MINLPLib2. Altogether, we have observed that the results were coherent in theoretical terms but at most reasonable in practical terms.

3 BINARY QUADRATIC PROGRAMMING

In this entr'acte we will continue to examine the impact of symmetries in Mathematical Programming by focusing on a particular combinatorial specialization of Quadratically Constrained Quadratic Programming, namely Binary Quadratic Programming. This relevant subfield of Mathematical Programming encompasses binary programs with a quadratic objective function and quadratic constraints, and has many real-world and combinatorial optimization applications. The choice of Binary Quadratic Programming here is twofold: first, it allow us to use the Orbital Independence framework once more to showcase the conditions under which the usage of Symmetry-Breaking Constraints is majoritarily advantageous; and second, it allow us to reach other topic of our interest, Semidefinite Programming. Since it is well-known that Semidefinite Programming suits Quadratically Constrained Quadratic Programming perfectly in technical terms, this entr'acte uses Binary Quadratic Programs to investigate symmetries and Semidefinite Programming all together. This is of concern because Semidefinite Programming is vastly employed nowadays to solve (approximately) all sorts of hard Mathematical Programs; that being so, it is frequently used to cope with problems related to Distance Geometry, introduced in Chapter 4.

3.1 INTRODUCTION

Even though solving general Binary Quadratic Programs (BQPs) is known to be **NP**-hard (with a few known exceptions [46]), Binary Quadratic Programming (BQP) has attracted a lot of attention amongst the optimization community in recent years. It has become a very important subfield of Mathematical Programming because a large number of theoretical and practical applications can be modelled using BQPs [42]. Among several interesting examples, we would like to mention one that relates to modelling and solving the Molecular Conformation Problem (MCP) [70] by means of a quadratic assignment formulation [74]. The Quadratic Assignment Problem (QAP) is a BQP and the MCP is one of the many applications of the Euclidean Distance Geometry Problem (see Chapter 4).

In this entr'acte we randomly generate symmetric Binary Quadratic Programs having a certain symmetry structure with two main purposes. First, we want to grasp initial impressions on the impact of symmetries and their breaking devices in the performance of Semidefinite Programming and Diagonally Dominant Programming solvers.

Second, we want to exemplify the conditions under which the usage of Symmetry-Breaking Constraints is majoritarily advantageous. As regards the former, after generating OI narrowings by means of the Algorithm 2 proposed in Chapter 2, we derive the primal and dual $\mathbb{SDP}$s (as well as their respective $\mathbb{DDP}$s) for both the original $\mathbb{BQP}$s and their narrowings; then we employ standard software codes available in the literature to solve the resulting programs. Note that as a byproduct of our experiments, we are also able to explicitly compare the quality of the bounds generated by SDP and DDP. As regards the latter, we just conduct similar experiments to those presented in Chapter 2.

The remainning of this entr'acte is structured as follows: in Section 3.2 we introduce the standard BQP form. The theoretical derivation of the Semidefinite Programs and Diagonally Dominant Programs associated to Binary Quadratic Programs are presented in Sections 3.3 and 3.4, respectively. We describe how we generate symmetric Binary Quadratic Programs in Section 3.5 and, to conclude, computational experiments are presented in Section 3.6.

3.2 BINARY QUADRATIC PROGRAMS

Notationwise, recall that $N = [n] = \{1, \ldots, n\}$. We are interested in studying $\mathbb{BQP}$s in the following general form:

$$\left.\begin{array}{rl} \min\limits_{x} & x^\top A_0 x + a_0{}^\top x \\ \forall i \in \mathcal{I}_I & x^\top A_i x + a_i{}^\top x \leqslant b_i, \\ \forall i \in \mathcal{I}_E & x^\top A_i x + a_i{}^\top x = b_i, \\ & x \in \{0,1\}^n. \end{array}\right\} \tag{21}$$

where A_i denotes a $n \times n$ real (possibly indefinite) symmetric matrix for all $i \in \{0\} \cup \mathcal{I}_I \cup \mathcal{I}_E$, b is a vector of dimension $(|\mathcal{I}_E| + |\mathcal{I}_I|)$ and x is a vector of decision variables of dimension n.

Now let $\mathsf{diag}(x)$ represent the square diagonal matrix that has the elements of vector x on its main diagonal. Note that since $x_i x_i = x_i$ holds for all $i \in N$, it is also true that $a_i{}^\top x = x^\top \mathsf{diag}(a_i) x$ for all $i \in \{0\} \cup \mathcal{I}_I \cup \mathcal{I}_E$. If we define $\tilde{A}_i = A_i + \mathsf{diag}(a_i)$, we can actually describe $\mathbb{BQP}$s using the more compact form:

$$\left.\begin{array}{rl} \min\limits_{x} & x^\top \tilde{A}_0 x \\ \forall i \in \mathcal{I}_I & x^\top \tilde{A}_i x \leqslant b_i, \\ \forall i \in \mathcal{I}_E & x^\top \tilde{A}_i x = b_i, \\ & x \in \{0,1\}^n. \end{array}\right\} \tag{22}$$

The inconvenient of representing $\mathbb{BQP}$s using the formulation given by Eq. (22) is that linear constraints which may describe the feasible region of Eq. (21) are necessarily reformulated as quadratic constraints.

Finally, since it is of our interest to work with SBCs later on, we remark that any SBC written as defined in Section 2.2.6 can be easily translated into one of the inequality constraint forms present in Eq. (21) or in Eq. (22).

3.3 SDP FOR BQP

Next we derive the primal and the dual standard SDPs (see Sect. 1.4) for BQPs written as Eq. (21). To this end, the crucial step is to observe that

$$x^\top A x = \mathrm{tr}(x^\top A x) = \mathrm{tr}(A x x^\top),$$

where $\mathrm{tr}(A)$ represents the trace of the matrix A. Defining a matrix X of decision variables as $X = xx^\top$, one gets

$$x^\top A x = \mathrm{tr}(AX) = A^\top \bullet X = A \bullet X,$$

where $A \bullet B$ denotes the Frobenius inner product. The same procedure can be applied to the linear terms of the functions appearing in Eq. (21) since $a^\top x = \mathrm{tr}(a^\top x)$ trivially holds. The equation $X - xx^\top = 0$ is known as a Rank Constraint (RC) over the matrix X: $\mathrm{rk}\, X = \mathrm{rk}\, xx^\top = \mathrm{rk}\, x$ for any $x \in \mathbb{R}^{n \times m}$. In the BQP case, $x \in \{0,1\}^n$.

The next step is to require $\mathrm{diag}(X) = x$, where $\mathrm{diag}(X)$ represents the column vector whose components are the elements of the main diagonal of X. Note that $X = x^\top x \wedge \mathrm{diag}(X) = x \Rightarrow x \in \{0,1\}^n$. Indeed, for $i = j \in N$, $X_{ii} = x_i x_i \wedge X_{ii} = x_i \Rightarrow x_i x_i = x_i \Rightarrow x_i(x_i - 1) = 0 \Rightarrow x_i \in \{0,1\}$. We may thus reformulate Eq. (21) into:

$$\left.\begin{array}{rl} \min\limits_{x,X} & A_0 \bullet X + a_0 \bullet x \\ \forall i \in \mathcal{I}_I & A_i \bullet X + a_i \bullet x \leqslant b_i, \\ \forall i \in \mathcal{I}_E & A_i \bullet X + a_i \bullet x = b_i, \\ & \mathrm{diag}(X) = x, \\ & X - x^\top x = 0. \end{array}\right\} \tag{23}$$

Finally, observe that the constraint $\mathrm{diag}(X) = x$ can be rewritten using the Frobenius inner product because $X_{ii} = x_i \Rightarrow X_{ii} - x_i = 0 \Rightarrow A_i \bullet X + a_i \bullet x = 0$ for $i \in N$, where $A_i = \mathrm{diag}(e_i)$, $a_i = -e_i$ and $e_i \in \mathbb{R}^n$ is the ith canonical unit vector.

The most common way to derive a SDP relaxation for Eq. (23) is by relaxing the Rank Constraint $X - x^\top x = 0$ into $X - x^\top x \succeq 0$ and use the fact that

$$X - x^\top x \succeq 0 \iff \begin{pmatrix} 1 & x^\top \\ x & X \end{pmatrix} \succeq 0$$

by the Schur complement condition for positive semidefiniteness to formulate the SDP:

$$\left.\begin{array}{rl} \min\limits_{x,X} & A_0 \bullet X + a_0 \bullet x \\ \forall i \in \mathcal{I}_I & A_i \bullet X + a_i \bullet x \leqslant b_i, \\ \forall i \in \mathcal{I}_E & A_i \bullet X + a_i \bullet x = b_i, \\ \forall i \in \mathcal{I}_D & A_i \bullet X + a_i \bullet x = 0, \\ & \begin{pmatrix} 1 & x^\top \\ x & X \end{pmatrix} \succeq 0. \end{array}\right\} \tag{24}$$

Following up, it is necessary to define a symmetric matrix Y of decision variables as

$$Y = \begin{pmatrix} 1 & x^\top \\ x & X \end{pmatrix}$$

and notice that all constraints in Eq. (24) are linear with respect to Y. It is then possible to rewrite the SDP given by Eq. (24) in primal standard form like:

$$\left.\begin{array}{rl} \min\limits_{Y} & P_0 \bullet Y \\ \forall i \in \mathcal{I}_I & P_i \bullet Y \leqslant b_i, \\ \forall i \in \mathcal{I}_E \cup \mathcal{I}_D & P_i \bullet Y = b_i, \\ & Y \succeq 0, \end{array}\right\} \tag{25}$$

where

$$P_k = \begin{pmatrix} 0 & 0 \\ a_k & A_k{}^\top \end{pmatrix}$$

for $k \in \{0\} \cup \mathcal{I}_E \cup \mathcal{I}_I \cup \mathcal{I}_D$. According to what was described in Section 1.4, the dual SDP of Eq. (25) may be formulated as:

$$\left.\begin{array}{rl} \max\limits_{u,Q} & b^\top u \\ & P_0 - \sum\limits_{i \in \mathcal{I}_E \cup \mathcal{I}_I \cup \mathcal{I}_D} u_i P_i = Q, \\ \forall i \in \mathcal{I}_I & u_i \geqslant 0, \\ & Q \succeq 0. \end{array}\right\} \tag{26}$$

3.4 DDP FOR BQP

The DDPs associated to the SDPs represented by Eq. (25) and Eq. (26) are immediately obtained by substituting the PSD constraints for DD constraints as in

$$\left.\begin{array}{rl} \min\limits_{Y} & P_0 \bullet Y \\ \forall i \in \mathcal{I}_I & P_i \bullet Y \leqslant b_i, \\ \forall i \in \mathcal{I}_E \cup \mathcal{I}_D & P_i \bullet Y = b_i, \\ & Y \text{ is DD}, \end{array}\right\} \tag{27}$$

and

$$\left.\begin{array}{rl} \max\limits_{u,Q} & b^\top u \\ & P_0 - \sum\limits_{i \in \mathcal{I}_E \cup \mathcal{I}_I \cup \mathcal{I}_D} u_i P_i = Q, \\ \forall i \in \mathcal{I}_I & u_i \geqslant 0, \\ & Q \text{ is DD}. \end{array}\right\} \tag{28}$$

The linearization of the DD constraints is achieved by means of Eq. (12), and it yields the Linear Programs:

$$\left.\begin{array}{rl} \min\limits_{Y,T} & P_0 \bullet Y \\ \forall i \in \mathcal{I}_I & P_i \bullet Y \leqslant b_i, \\ \forall i \in \mathcal{I}_E \cup \mathcal{I}_D & P_i \bullet Y = b_i, \\ \forall i \in [n+1] & Y_{ii} \geqslant \sum\limits_{\substack{j \in [n+1] \\ j \neq i}} T_{ij} \\ & T \geqslant Y \geqslant -T. \end{array}\right\} \tag{29}$$

and

$$\left.\begin{array}{rl} \max\limits_{u,Q,T} & b^\top u \\ & P_0 - \sum\limits_{i \in \mathcal{I}_E \cup \mathcal{I}_I \cup \mathcal{I}_D} u_i P_i = Q, \\ \forall i \in \mathcal{I}_I & u_i \geqslant 0, \\ \forall i \in [n+1] & Q_{ii} \geqslant \sum\limits_{\substack{j \in [n+1] \\ j \neq i}} T_{ij} \\ & T \geqslant Q \geqslant -T. \end{array}\right\} \tag{30}$$

Finally, consider an auxiliary matrix Z of decision variables to state the iterative forms of Eq. (29) and Eq. (30) as:

$$\left.\begin{array}{rl} \min\limits_{Y,Z,T} & P_0 \bullet Y \\ \forall i \in \mathcal{I}_I & P_i \bullet Y \leqslant b_i, \\ \forall i \in \mathcal{I}_E \cup \mathcal{I}_D & P_i \bullet Y = b_i, \\ & Y = U^\top Z U, \\ \forall i \in [n+K] & Z_{ii} \geqslant \sum\limits_{\substack{j \in [n+K] \\ j \neq i}} T_{ij}, \\ & T \geqslant Z \geqslant -T, \\ & T \geqslant 0. \end{array}\right\} \tag{31}$$

and

$$\left.\begin{array}{rl} \max\limits_{u,Q,Z,T} & b^T u \\ & P_0 - \sum\limits_{i \in \mathcal{I}_E \cup \mathcal{I}_I \cup \mathcal{I}_D} u_i P_i = Q, \\ \forall i \in \mathcal{I}_I & u_i \geqslant 0, \\ & Y = U^\top Z U, \\ \forall i \in [n+K] & Z_{ii} \geqslant \sum\limits_{\substack{j \in [n+K] \\ j \neq i}} T_{ij}, \\ & T \geqslant Z \geqslant -T, \\ & T \geqslant 0. \end{array}\right\} \qquad (32)$$

We would like to point out that our coding of the iterative DDP method implements four termination criteria:

1. Maximum number of iterations reached;
2. Time limit reached;
3. Solution is feasible but not Positive Definite;
4. Problem is infeasible.

An execution is globally limited by a maximum number of iterations and an execution time limit. Moreover, the execution stops if, at any iteration, a feasible solution that is not PD is returned or if the approximation results infeasible. The former has to do with the fact that a feasible non-PD solution means that the inner approximation has converged to a face of the PSD cone, which is the ultimate objective of the method. The latter happens in case the DD cone does not intersect the original feasible region; this can only happen however in the first iteration of the method, since after one successful (feasible) iteration, the inner approximation can no longer become empty (see Section 1.4.1.2).

Recall that the DD constraint is parameterized by means of the U matrices. In case the first program of the sequence is infeasible, an alternative to construct a subsequent feasible approximation is to run a feasibility-recovery subroutine (similar to the first phase of the simplex method): run the iterative DDP method on a lifted program containing slack and surplus variables whilst minimizing the sum of these variables until all of them are zero. Then the main algorithm can carry on minimizing the original objective function using the U matrices from the last iteration of the recovery subroutine (the iterative method on the surplus/slack formulation).

3.5 BQP GENERATION

In this section we will expose the most important properties of the symmetric $\mathbb{BQP}$s proposed in this entr'acte.

The first property (P1) relates to their feasible region: every instance has one single equality constraint of the type

$$a^\top x = \lceil n/2 \rceil. \tag{33}$$

The usage of constraint Eq. (33) aims at imposing a balance between zeros and ones in the optimal solutions; as a consequence, we expect to create symmetric subtrees mostly around the central axis of the search tree. Moreover, being invariant to permutations, this constraint allows us to specify the structure of the formulation groups by controlling the strutucture of the matrix A_0 alone.

The indices of the decision variables are divided into a partition $\mathcal{P}$, and the subsets s of the partition are randomly selected to be an orbit or not.

The second property (P2) refers to the matrix A_0: it is a block diagonal matrix. Each subset of the partition corresponds to a block in the matrix A_0. If a subset $s \in \mathcal{P}$ is not chosen to become an orbit, the entries of the block B_s are computed by sampling a $(|s| \times |s|)$-matrix M_s and defining

$$B_s = {M_s}^\top M_s. \tag{34}$$

These blocks are Gram matrices. If a subset s is selected to be an orbit, the entries of the block B_s are computed by sampling a pair (z_1, z_2) of natural numbers and defining

$$B_s = \begin{cases} z_1 + (|s|-1)z_2 & \text{if } i = j, \\ -z_2 & \text{if } i \neq j. \end{cases} \tag{35}$$

These blocks are DD matrices. Since all blocks of A_0 are PSD, the matrix A_0 is PSD as well and the continuous relaxations of the $\mathbb{BQP}$s are convex.

The blocks defined by Eq. (35) are the third and last property (P3). Eq. (35) specifies matrices whose off-diagonal entries have the same value and can be permuted at will among them; similarly, the diagonal entries are identical to each other and can be permuted freely.

When P1, P2 and P3 are put together, they induce the following symmetry properties on the formulation group of the $\mathbb{BQP}$s:

1. $G_P[\omega] = \mathsf{Sym}(\omega)$ for every orbit $\omega \in \Omega_P$;

2. $\Omega_I = \Omega_{G_P}$.

According to the theory presented in Chapter 2, these two conditions allow us to concurrently use SBCs derived from all orbits of $\mathbb{BQP}$s that satisfy P1, P2 and P3. Example 31 illustrates one of these programs.

Example 31. *Let* P *be the following* BQP*:*

$$\min_{x\in\{0,1\}^9} \quad x^\top A_0 x$$
$$a_1{}^\top x = 5,$$

where

$$A_0 = \begin{pmatrix} 6 & -3 & -3 & 0 & 0 & 0 & 0 & 0 & 0 \\ -3 & 6 & -3 & 0 & 0 & 0 & 0 & 0 & 0 \\ -3 & -3 & 6 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 12 & -6 & -6 & 0 & 0 & 0 \\ 0 & 0 & 0 & -6 & 12 & -6 & 0 & 0 & 0 \\ 0 & 0 & 0 & -6 & -6 & 12 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 & 1 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 5 & 2 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 & 2 & 5 \end{pmatrix}$$

and $a_1{}^\top = (1\ 1\ 1\ 1\ 1\ 1\ 1\ 1\ 1)$*. It does satisfy properties P1, P2 and P3. Moreover,* $\Omega_{G_P} = \{\omega_1, \omega_2\} = \{\{1,2,3\},\{4,5,6\}\}$*, with* $\omega_1 \perp\!\!\!\perp \omega_2$ *holding. And the transitive constituent of both orbits is the full symmetric group.*

3.6 COMPUTATIONAL EXPERIMENTS

In this section we describe the computational environment involved (instances, machinery, solvers, setups, procedures, etc) and analyze the results obtained in our experiments.

3.6.1 *Dataset*

Our set of randomly generated symmetric BQPs is divided in two groups, according to the size of the instances. The first group contains 40 small instances (with up to 50 variables) which are used in the SDP/DDP experiments. The second group constains 97 medium sized instances (with up to 100 variables) which are used in the OI experiments. The BQPs are named bqp_n_oxs, where n represents the number of variables, o the number of orbits and s the orbits' size (R means orbits with random sizes).

3.6.2 *Environment*

We implemented the Mathematical Programs presented in Sections 3.3 and 3.4 in Python 2.7 using PICOS 1.1.1 [77] to interface with the solvers MOSEK 7.1 [7] and CPLEX 12.6 [37], which were used to solve the SDPs and the DDPs, respectively. The tests were carried out on a

24-CPU Intel Xeon at 2.53GHz with 48Gb RAM running GNU/Linux. We disabled CPLEX's built-in symmetry handling methods and ran it in single thread mode (similar to what was done in Chapter 2) to obtain the OI results presented in Section 3.6.4.

3.6.3 *Results: SDP/DDP*

To evaluate the quality of the bounds computed in this section, we computed the optima of the instances in our testbed. Table 6 reports the optimization results for Eq. (21). Per instance, it exhibits the best solution found, the execution time, the optimality gap and the solver status at termination (opt = optimum found, lim = time limit reached, inf = infeasible instance). All instances were solved to optimality.

Instance	Original formulation			
	Best	Time (s)	Gap (%)	St.
bqp_25_2xR	140	0.54	0	opt
bqp_25_3xR	100	0.03	0	opt
bqp_25_4x5	68	0.06	0	opt
bqp_30_2x10	235	0.06	0	opt
bqp_30_2xR	375	0.38	0	opt
bqp_30_3x6	27	0.04	0	opt
bqp_30_3xR	114	0.03	0	opt
bqp_30_4x5	30	0.02	0	opt
bqp_30_4x6	69	0.06	0	opt
bqp_30_5x5	45	0.02	0	opt
bqp_35_2xR	3960	116.42	0	opt
bqp_35_3x7	87	0.04	0	opt
bqp_35_3xR	79	0.03	0	opt
bqp_35_4x7	108	0.05	0	opt
bqp_35_4xR	54	0.01	0	opt
bqp_35_6x5	56	0.10	0	opt
bqp_40_2x10	110	0.01	0	opt
bqp_40_2xR	200	0.02	0	opt
bqp_40_3x10	70	0.00	0	opt
bqp_40_3x8	160	0.11	0	opt
bqp_40_3xR	184	0.11	0	opt
bqp_40_4x8	116	0.15	0	opt
bqp_40_5xR	96	0.06	0	opt
bqp_40_7x5	30	0.02	0	opt
bqp_45_2x15	780	4.37	0	opt
bqp_45_2xR	1014	0.04	0	opt
bqp_45_3x9	290	0.22	0	opt
bqp_45_3xR	469	0.12	0	opt
bqp_45_4x9	242	0.40	0	opt
bqp_45_4xR	260	0.13	0	opt
bqp_45_5xR	172	0.37	0	opt
bqp_45_8x5	40	0.24	0	opt
bqp_50_2xR	842	1.04	0	opt

bqp_50_3x10	255	0.74	0	opt
bqp_50_3xR	296	0.09	0	opt
bqp_50_4x10	105	0.23	0	opt
bqp_50_4xR	155	0.06	0	opt
bqp_50_6xR	80	0.04	0	opt
bqp_50_7x5	40	0.01	0	opt
bqp_50_9x5	40	0.04	0	opt

Table 6: Optimization results obtained with CPLEX 12.6 for the set of small BQP instances.

We start by analyzing the impact of SBCs in SDP. Tables 7 and 8 report details of the optimization results for the primal and dual SDPs, respectively. We derived the primal and dual relaxations for both the original formulation and its OI-narrowing. Per instance and for each formulation, the table exhibits the best solution found, the user cpu time (in seconds), the gap (%), the rank of the solution and its maximum possible value (displayed like rank/rank$_{\max}$), and the solver status at termination (opt = optimum found, nopt = near optimal, unkn = unknown, lim = time limit reached, inf = infeasible). Best values are emphasized in boldface. According to MOSEK's online documentation [7], the near optimal status occurs when the IPM terminates having found a solution not too far from meeting the optimality conditions (for instance when the solver stalls and can make no more significant progress towards the optimal solution). The same applies for the unknown status, but in this case, the solution is less likely to be optimal. Yet MOSEK's documentation recommends to check its quality; an illustrative description can be found in [6].

As Table 7 exhibits, all executions of the primal SDPs terminated due to stalling, either with near optimal or unknown status. Since we are not dealing with BB type algorithms in this case, we will evaluate the impact of the SBCs according to the ranking: quality of bound, status and execution time. In terms of value, the bounds differ in six cases (bqp_25_2xR, bqp_25_3xR, bqp_30_3xR, bqp_45_2xR, bqp_45_3xR and bqp_45_5xR); in terms of status, the bounds differ in three instances, two (bqp_25_2xR and bqp_25_3xR) in favor of the narrowing relaxation (the status changed from "unkn" to "nopt"). Timewise, we observe 30 cases were the use of SBCs were detrimental. But this is expected since more constraints are added to the SDPs. As regards the rank, we notice that it is considerably reduced in three cases when using SBCs: bqp_40_2xR, bqp_45_3xR and bqp_50_3x10.

Instance	Primal SDP							
	Original formulation				OI-narrowing			
	Best	Time (s)	Rank	St.	Best	Time (s)	Rank	St.
bqp_25_2xR	0.16	0.12	26/26	unkn	0.17	0.12	26/26	**nopt**

bqp_25_3xR	3.45	0.08	26/26	unkn	3.44	0.11	26/26	**nopt**
bqp_25_4x5	13.00	**0.07**	26/26	nopt	13.00	0.09	26/26	nopt
bqp_30_2x10	38.25	**0.12**	24/31	nopt	38.25	0.13	28/31	nopt
bqp_30_2xR	15.00	**0.09**	30/31	nopt	15.00	0.12	31/31	nopt
bqp_30_3x6	0.00	0.13	30/31	unkn	0.00	**0.12**	30/31	unkn
bqp_30_3xR	**2.81**	0.15	30/31	unkn	2.80	0.13	28/31	unkn
bqp_30_4x5	7.23	0.11	31/31	nopt	7.23	0.11	31/31	nopt
bqp_30_4x6	0.00	**0.09**	31/31	unkn	0.00	0.13	31/31	unkn
bqp_30_5x5	0.00	**0.09**	30/31	unkn	0.00	0.12	31/31	unkn
bqp_35_2xR	18.00	**0.14**	36/36	nopt	18.00	0.15	36/36	nopt
bqp_35_3x7	36.00	**0.13**	35/36	nopt	36.00	0.18	35/36	nopt
bqp_35_3xR	14.33	**0.11**	36/36	nopt	14.33	0.13	36/36	nopt
bqp_35_4x7	0.00	0.11	36/36	**nopt**	0.00	0.14	36/36	unkn
bqp_35_4xR	0.00	**0.11**	36/36	unkn	0.00	0.13	36/36	unkn
bqp_35_6x5	0.00	**0.11**	36/36	unkn	0.00	0.14	36/36	unkn
bqp_40_2x10	80.00	**0.13**	40/41	nopt	80.00	0.16	41/41	nopt
bqp_40_2xR	3.18	**0.13**	41/41	nopt	3.18	0.17	23/41	nopt
bqp_40_3x10	60.00	**0.16**	41/41	nopt	60.00	0.18	41/41	nopt
bqp_40_3x8	0.00	**0.14**	40/41	unkn	0.00	0.15	40/41	unkn
bqp_40_3xR	60.00	**0.13**	41/41	nopt	60.00	0.16	40/41	nopt
bqp_40_4x8	0.00	**0.13**	41/41	unkn	0.00	0.20	41/41	unkn
bqp_40_5xR	6.61	**0.13**	41/41	nopt	6.61	0.21	41/41	nopt
bqp_40_7x5	20.00	**0.14**	40/41	nopt	20.00	0.15	40/41	nopt
bqp_45_2x15	92.00	**0.14**	45/46	nopt	92.00	0.19	45/46	nopt
bqp_45_2xR	5.69	0.13	44/46	unkn	**5.70**	0.16	45/46	unkn
bqp_45_3x9	0.00	**0.17**	45/46	unkn	0.00	0.19	45/46	unkn
bqp_45_3xR	**0.10**	0.19	40/46	unkn	0.08	0.20	25/46	unkn
bqp_45_4x9	46.00	**0.18**	45/46	nopt	46.00	0.21	45/46	nopt
bqp_45_4xR	0.00	**0.15**	46/46	unkn	0.00	0.19	45/46	unkn
bqp_45_5xR	**53.03**	0.16	46/46	nopt	53.02	0.19	43/46	nopt
bqp_45_8x5	0.00	**0.13**	45/46	unkn	0.00	0.26	45/46	unkn
bqp_50_2xR	75.00	**0.18**	50/51	nopt	75.00	0.24	50/51	nopt
bqp_50_3x10	0.35	**0.18**	40/51	unkn	0.35	0.23	21/51	unkn
bqp_50_3xR	6.42	**0.18**	49/51	nopt	6.42	0.34	46/51	nopt
bqp_50_4x10	0.00	0.26	51/51	unkn	0.00	**0.23**	50/51	unkn
bqp_50_4xR	0.00	**0.19**	49/51	unkn	0.00	0.24	51/51	unkn
bqp_50_6xR	0.00	**0.17**	50/51	unkn	0.00	0.23	51/51	unkn
bqp_50_7x5	0.00	**0.17**	51/51	unkn	0.00	0.21	51/51	unkn
bqp_50_9x5	0.00	**0.15**	51/51	unkn	0.00	0.19	51/51	unkn

Table 7: Primal SDP results obtained with MOSEK 7.1 for the set of small BQP instances.

Table 8 shows that the results related to the dual SDPs are slightly better. MOSEK has converged to optmimal solutions 30 times, what did not happen with the primal programs; and the lower bounds obtained are the same per instance. In relative terms, the results have also improved in favor of the SDPs relaxations of the narrowings. As regards the termination status, they differ in 17 intances with 6 improvements (either "unkn" to "nopt" or "nopt" to "opt") in favor of the

SBCs; as regards reduction in CPU time, we observe 5 cases out of 23 in favor of the SBCs. No favorable trend is observed as concerns the rank of the solutions.

Overall, it may be interesting to further investigate why there are so many cases of unknown or near optimality solutions. Moreover, it seems to be clear that solving the SDPs corresponding to the OI-narrowing is not quite advantegeous. In any case, provided that the times do not differ too much in absolute values, we believe that if the non-SBC version provides a weak bound, one should add the SBCs and re-solve it hoping for better bounds. Specially because these results refer to a particular application, symmetric BQPs. They may be better in a different context.

Instance	Dual SDP							
	Original formulation				OI-narrowing			
	Best	Time (s)	Rank	St.	Best	Time (s)	Rank	St.
bqp_25_2xR	0.16	0.74	26/26	nopt	0.16	**0.63**	26/26	nopt
bqp_25_3xR	3.44	**0.49**	26/26	nopt	3.44	0.58	26/26	nopt
bqp_25_4x5	13.00	0.47	25/26	**opt**	13.00	1.02	26/26	nopt
bqp_30_2x10	38.24	1.05	31/31	nopt	38.24	0.92	30/31	**opt**
bqp_30_2xR	15.00	0.92	30/31	**opt**	15.00	1.74	31/31	nopt
bqp_30_3x6	0.00	1.65	31/31	nopt	0.00	**1.26**	31/31	nopt
bqp_30_3xR	2.78	1.43	31/31	nopt	2.78	**1.12**	31/31	nopt
bqp_30_4x5	7.23	1.14	31/31	opt	7.23	**0.82**	31/31	opt
bqp_30_4x6	0.00	0.78	30/31	**opt**	0.00	1.39	31/31	nopt
bqp_30_5x5	0.00	**0.92**	31/31	nopt	0.00	1.43	31/31	nopt
bqp_35_2xR	18.00	1.52	35/36	**opt**	18.00	2.77	35/36	nopt
bqp_35_3x7	36.00	**1.45**	35/36	opt	36.00	3.02	36/36	opt
bqp_35_3xR	14.32	**1.36**	35/36	opt	14.32	1.87	35/36	opt
bqp_35_4x7	0.00	**1.44**	35/36	opt	0.00	2.27	35/36	opt
bqp_35_4xR	0.00	**1.54**	36/36	nopt	0.00	3.16	36/36	nopt
bqp_35_6x5	0.00	**2.14**	36/36	nopt	0.00	2.97	36/36	nopt
bqp_40_2x10	80.00	**2.80**	40/41	opt	80.00	3.71	41/41	opt
bqp_40_2xR	3.18	2.88	41/41	nopt	3.18	2.37	40/41	**opt**
bqp_40_3x10	60.00	3.02	40/41	**opt**	60.00	4.02	41/41	nopt
bqp_40_3x8	0.00	2.57	41/41	unkn	0.00	5.64	41/41	**nopt**
bqp_40_3xR	60.00	**2.19**	39/41	opt	60.00	4.04	41/41	opt
bqp_40_4x8	0.00	**2.88**	41/41	nopt	0.00	4.13	41/41	nopt
bqp_40_5xR	6.61	2.15	40/41	opt	6.61	**2.13**	40/41	opt
bqp_40_7x5	20.00	2.57	40/41	**opt**	20.00	5.17	41/41	nopt
bqp_45_2x15	92.00	**5.51**	45/46	nopt	92.00	7.26	46/46	nopt
bqp_45_2xR	5.67	4.00	46/46	**nopt**	5.67	5.28	45/46	unkn
bqp_45_3x9	0.00	**4.80**	46/46	nopt	0.00	7.27	45/46	nopt
bqp_45_3xR	0.05	**4.45**	46/46	unkn	0.05	5.37	46/46	unkn
bqp_45_4x9	46.00	**3.84**	45/46	opt	46.00	5.94	46/46	opt
bqp_45_4xR	0.00	**4.29**	46/46	nopt	0.00	6.73	45/46	nopt
bqp_45_5xR	53.02	**3.05**	46/46	opt	53.02	3.78	46/46	opt
bqp_45_8x5	0.00	**4.83**	45/46	nopt	0.00	7.17	45/46	nopt
bqp_50_2xR	75.00	5.09	50/51	**opt**	75.00	11.14	50/51	nopt

bqp_50_3x10	0.35	5.30	50/51	**opt**	0.35	6.72	51/51	unkn
bqp_50_3xR	6.41	5.70	51/51	nopt	6.41	5.38	51/51	**opt**
bqp_50_4x10	0.00	5.88	51/51	**nopt**	0.00	8.50	51/51	unkn
bqp_50_4xR	0.00	6.93	50/51	unkn	0.00	10.04	51/51	**nopt**
bqp_50_6xR	0.00	**7.04**	51/51	nopt	0.00	9.96	51/51	nopt
bqp_50_7x5	0.00	6.78	51/51	unkn	0.00	10.77	50/51	**nopt**
bqp_50_9x5	0.00	4.61	51/51	**opt**	0.00	8.42	51/51	unkn

Table 8: Dual SDP results obtained with MOSEK 7.1 for the set of small BQP instances.

Next we analyze the impact of SBCs in DDP. Tables 9 and 10 report details of the optimization results for primal and dual DDPs, respectively. Again we derived the primal and dual relaxations for both the original formulation and its OI-narrowing. Per instance and for each formulation, the table exhibits the best solution found, the user cpu time (in seconds), number of DDP iterations, the rank of the solution and its maximum possible value (displayed like rank/rank$_{\max}$) and the termination status of the algorithm (not = solution is feasible but not PD, lim = time limit reached, inf = infeasible). Best values are emphasized in boldface.

With respect to the primal DDPs, the results are quite poor. Although the bounds obtained were the same for each instance, the usage of SBCs was detrimental in 28 out of 40 cases in terms of the execution time; they were helpful in 7 out of 40 cases. Furthermore, we notice that the first inner aproximation already provides a feasible non-PD solution for all instances (Iter. "1" and St. "not"). This means that the inner approximation intersects one of the faces of the PSD cone in the first iteration. As a consequence, we could not verify the usefulness of the iterative method in terms of improving the value of the bounds after each iteration. We note however that the presence of the SBCs alter the sequence of approximations since the rank of the solutions differs (i. e. a different face of the cone is reached) per instance when the SBCs are used. Fortunately this particular investigation has no requirements regarding the rank of the solutions; the context is rather different in Chapter 4 though.

Instance	Primal DDP									
	Original formulation					OI-narrowing				
	Best	Time (s)	Iter.	Rank	St.	Best	Time (s)	Iter.	Rank	St.
bqp_25_2xR	520.00	0.48	1	1/26	not	520.00	**0.39**	1	8/26	not
bqp_25_3xR	715.00	**0.37**	1	1/26	not	715.00	0.43	1	6/26	not
bqp_25_4x5	364.00	**0.35**	1	1/26	not	364.00	0.36	1	5/26	not
bqp_30_2x10	645.00	**0.55**	1	1/31	not	645.00	0.56	1	10/31	not
bqp_30_2xR	2160.00	0.67	1	1/31	not	2160.00	**0.57**	1	6/31	not
bqp_30_3x6	45.00	**0.55**	1	1/31	not	45.00	0.56	1	1/31	not

bqp_30_3xR	210.00	0.66	1	1/31	not	210.00	**0.56**	1	6/31	not
bqp_30_4x5	285.00	**0.66**	1	1/31	not	285.00	0.67	1	1/31	not
bqp_30_4x6	465.00	0.68	1	1/31	not	465.00	**0.56**	1	6/31	not
bqp_30_5x5	315.00	**0.55**	1	1/31	not	315.00	0.58	1	1/31	not
bqp_35_2xR	918.00	**0.82**	1	1/36	not	918.00	0.84	1	5/36	not
bqp_35_3x7	342.00	0.83	1	1/36	not	342.00	0.83	1	7/36	not
bqp_35_3xR	936.00	**0.82**	1	1/36	not	936.00	0.83	1	1/36	not
bqp_35_4x7	216.00	**1.04**	1	1/36	not	216.00	1.05	1	7/36	not
bqp_35_4xR	882.00	**0.83**	1	1/36	not	882.00	1.05	1	7/36	not
bqp_35_6x5	558.00	1.05	1	1/36	not	558.00	1.05	1	3/36	not
bqp_40_2x10	3500.00	**1.20**	1	1/41	not	3500.00	1.22	1	4/41	not
bqp_40_2xR	200.00	1.21	1	1/41	not	200.00	1.21	1	1/41	not
bqp_40_3x10	2000.00	**1.20**	1	1/41	not	2000.00	1.58	1	6/41	not
bqp_40_3x8	720.00	**1.18**	1	1/41	not	720.00	1.22	1	8/41	not
bqp_40_3xR	480.00	**1.20**	1	1/41	not	480.00	1.22	1	5/41	not
bqp_40_4x8	520.00	**1.17**	1	1/41	not	520.00	1.59	1	8/41	not
bqp_40_5xR	480.00	**1.18**	1	1/41	not	480.00	1.21	1	1/41	not
bqp_40_7x5	180.00	**1.16**	1	1/41	not	180.00	1.21	1	5/41	not
bqp_45_2x15	4002.00	**1.65**	1	1/46	not	4002.00	1.71	1	7/46	not
bqp_45_2xR	667.00	**2.24**	1	1/46	not	667.00	2.28	1	8/46	not
bqp_45_3x9	1840.00	**1.63**	1	1/46	not	1840.00	1.69	1	9/46	not
bqp_45_3xR	6049.00	**1.64**	1	1/46	not	6049.00	1.71	1	1/46	not
bqp_45_4x9	1518.00	**1.64**	1	1/46	not	1518.00	1.67	1	9/46	not
bqp_45_4xR	1656.00	**1.64**	1	1/46	not	1656.00	1.65	1	7/46	not
bqp_45_5xR	1150.00	1.64	1	1/46	not	1150.00	1.64	1	7/46	not
bqp_45_8x5	368.00	**1.65**	1	1/46	not	368.00	1.68	1	2/46	not
bqp_50_2xR	5675.00	3.07	1	1/51	not	5675.00	**2.28**	1	12/51	not
bqp_50_3x10	1075.00	**2.26**	1	1/51	not	1075.00	2.29	1	10/51	not
bqp_50_3xR	1725.00	**2.22**	1	1/51	not	1725.00	3.12	1	9/51	not
bqp_50_4x10	1450.00	2.68	1	1/51	not	1450.00	**2.26**	1	1/51	not
bqp_50_4xR	850.00	**2.25**	1	1/51	not	850.00	3.15	1	9/51	not
bqp_50_6xR	1150.00	**2.28**	1	1/51	not	1150.00	3.51	1	8/51	not
bqp_50_7x5	1375.00	2.28	1	1/51	not	1375.00	2.28	1	5/51	not
bqp_50_9x5	525.00	3.08	1	1/51	not	525.00	**2.29**	1	1/51	not

Table 9: Primal DDP results obtained with CPLEX 12.6 for the set of small BQP instances.

As refers to the dual DDPs, the results are slightly better. Again the lower bounds obtained are the same for each instance, but the SBCs impact improved in relative terms: we observe a reduction in the execution time in 15 cases, and a growth in 20 cases. For the same reason, feasible and non-PD solutions found in the first iteration, we could not observe the iterative method in action and improving the quality of the bounds after performing at least two iterations successfully. In the case of the dual programs, however, the presence of the SBCs did not change the rank of the solutions, but again, this is not relevant in this context.

Instance	Dual DDP									
	Original formulation					OI-narrowing				
	Best	Time (s)	Iter.	Rank	St.	Best	Time (s)	Iter.	Rank	St.
bqp_25_2xR	-2184.00	0.54	1	25/26	not	-2184.00	**0.53**	1	25/26	not
bqp_25_3xR	-13858.00	**0.42**	1	25/26	not	-13858.00	0.47	1	25/26	not
bqp_25_4x5	-6552.00	**0.49**	1	25/26	not	-6552.00	0.57	1	25/26	not
bqp_30_2x10	-40425.00	0.68	1	30/31	not	-40425.00	0.68	1	30/31	not
bqp_30_2xR	-20385.00	0.82	1	30/31	not	-20385.00	**0.67**	1	30/31	not
bqp_30_3x6	-21435.00	0.70	1	30/31	not	-21435.00	**0.68**	1	30/31	not
bqp_30_3xR	-29835.00	**0.75**	1	30/31	not	-29835.00	0.82	1	30/31	not
bqp_30_4x5	-1260.00	**0.68**	1	30/31	not	-1260.00	0.69	1	30/31	not
bqp_30_4x6	-6330.00	0.83	1	30/31	not	-6330.00	**0.75**	1	30/31	not
bqp_30_5x5	-660.00	0.69	1	30/31	not	-660.00	0.69	1	30/31	not
bqp_35_2xR	-40302.00	0.97	1	35/36	not	-40302.00	**0.95**	1	35/36	not
bqp_35_3x7	-31068.00	**1.05**	1	35/36	not	-31068.00	1.24	1	35/36	not
bqp_35_3xR	-2754.00	1.27	1	35/36	not	-2754.00	**1.01**	1	35/36	not
bqp_35_4x7	-29574.00	1.06	1	35/36	not	-29574.00	**1.03**	1	35/36	not
bqp_35_4xR	-19926.00	1.30	1	35/36	not	-19926.00	**1.03**	1	35/36	not
bqp_35_6x5	-4284.00	1.12	1	35/36	not	-4284.00	**1.05**	1	35/36	not
bqp_40_2x10	-130340.00	**1.48**	1	40/41	not	-130340.00	1.88	1	40/41	not
bqp_40_2xR	-900.00	**1.46**	1	40/41	not	-900.00	1.48	1	40/41	not
bqp_40_3x10	-126800.00	2.08	1	40/41	not	-126800.00	**1.51**	1	40/41	not
bqp_40_3x8	-96640.00	**1.46**	1	40/41	not	-96640.00	1.57	1	40/41	not
bqp_40_3xR	-52780.00	**1.45**	1	40/41	not	-52780.00	1.51	1	40/41	not
bqp_40_4x8	-67880.00	**1.46**	1	40/41	not	-67880.00	1.57	1	40/41	not
bqp_40_5xR	-800.00	**1.50**	1	40/41	not	-800.00	1.59	1	40/41	not
bqp_40_7x5	-41420.00	1.90	1	40/41	not	-41420.00	**1.57**	1	40/41	not
bqp_45_2x15	-739496.00	2.59	1	45/46	not	-739496.00	**2.11**	1	45/46	not
bqp_45_2xR	-234669.00	**2.06**	1	45/46	not	-234669.00	2.14	1	45/46	not
bqp_45_3x9	-183954.00	**2.12**	1	45/46	not	-183954.00	2.13	1	45/46	not
bqp_45_3xR	-253644.00	**2.05**	1	45/46	not	-253644.00	2.08	1	45/46	not
bqp_45_4x9	-105800.00	**2.70**	1	45/46	not	-105800.00	2.72	1	45/46	not
bqp_45_4xR	-73853.00	**2.11**	1	45/46	not	-73853.00	2.70	1	45/46	not
bqp_45_5xR	-23253.00	**2.14**	1	45/46	not	-23253.00	2.71	1	45/46	not
bqp_45_8x5	-48530.00	2.16	1	45/46	not	-48530.00	2.16	1	45/46	not
bqp_50_2xR	-183075.00	3.76	1	50/51	not	-183075.00	**2.86**	1	50/51	not
bqp_50_3x10	-103075.00	2.96	1	50/51	not	-103075.00	2.96	1	50/51	not
bqp_50_3xR	-24650.00	3.84	1	50/51	not	-24650.00	**2.93**	1	50/51	not
bqp_50_4x10	-3600.00	3.91	1	50/51	not	-3600.00	**2.99**	1	50/51	not
bqp_50_4xR	-389725.00	3.06	1	50/51	not	-389725.00	3.06	1	50/51	not
bqp_50_6xR	-190725.00	**3.05**	1	50/51	not	-190725.00	3.09	1	50/51	not
bqp_50_7x5	-58100.00	**3.10**	1	50/51	not	-58100.00	3.12	1	50/51	not
bqp_50_9x5	-1550.00	**3.09**	1	50/51	not	-1550.00	3.13	1	50/51	not

Table 10: Dual DDP results obtained with CPLEX 12.6 for the set of small BQP instances.

Table 11 displays a direct comparison of the bounds obtained during our experiments. In theory, for the same instance, we expect the bounds to satisfy two possible chains, either

$$\mathrm{DDP_P} \geqslant \mathrm{BQP_O} \geqslant \mathrm{SDP_P} = \mathrm{SDP_D} \geqslant \mathrm{DDP_D} \tag{C1}$$

or

$$\mathrm{BQP_O} \geqslant \mathrm{DDP_P} \geqslant \mathrm{SDP_P} = \mathrm{SDP_D} \geqslant \mathrm{DDP_D}, \tag{C2}$$

where the subscripts P, O and D mean primal, original and dual, respectively. Both chains are correct because both the $\mathrm{DDP_P}$ and the $\mathrm{BQP_O}$ can be seen as two unrelated inner approximations of the $\mathrm{SDP_P}$ (recall that SDPs are convex relaxations of nonconvex MPs). As regards the SDPs, the equality holds because of strong duality.

In most of the cases, the bounds satisfy the chain C1, whilst the chain C2 is verified in only two cases: bqp_35_2xR and bqp_45_2xR. The most important dicovery with this comparison is that the bounds provided by the dual DDPs are very weak. Most likely, even though the first inner approximation ($U^0 = I$, see Eq. (16)) is feasible, it is not good for the application at hand. Nevertheless, since this is a parameterized method, one can always consider to use a different starting value for the U matrices, seeking for improvements in the bounds generated by the dual DDPs.

Instance	$\mathrm{BQP_O}$	$\mathrm{DDP_P}$	$\mathrm{SDP_P}$	$\mathrm{SDP_D}$	$\mathrm{DDP_D}$	Chain
bqp_25_2xR	140	520.00	0.16	0.16	-2184.00	C1
bqp_25_3xR	100	715.00	3.45	3.44	-13858.00	C1
bqp_25_4x5	68	364.00	13.00	13.00	-6552.00	C1
bqp_30_2x10	235	645.00	38.25	38.24	-40425.00	C1
bqp_30_2xR	375	2160.00	15.00	15.00	-20385.00	C1
bqp_30_3x6	27	45.00	0.00	0.00	-21435.00	C1
bqp_30_3xR	114	210.00	2.81	2.78	-29835.00	C1
bqp_30_4x5	30	285.00	7.23	7.23	-1260.00	C1
bqp_30_4x6	69	465.00	0.00	0.00	-6330.00	C1
bqp_30_5x5	45	315.00	0.00	0.00	-660.00	C1
bqp_35_2xR	3960	918.00	18.00	18.00	-40302.00	C2
bqp_35_3x7	87	342.00	36.00	36.00	-31068.00	C1
bqp_35_3xR	79	936.00	14.33	14.32	-2754.00	C1
bqp_35_4x7	108	216.00	0.00	0.00	-29574.00	C1
bqp_35_4xR	54	882.00	0.00	0.00	-19926.00	C1
bqp_35_6x5	56	558.00	0.00	0.00	-4284.00	C1
bqp_40_2x10	110	3500.00	80.00	80.00	-130340.00	C1
bqp_40_2xR	200	200.00	3.18	3.18	-900.00	C1
bqp_40_3x10	70	2000.00	60.00	60.00	-126800.00	C1
bqp_40_3x8	160	720.00	0.00	0.00	-96640.00	C1
bqp_40_3xR	184	480.00	60.00	60.00	-52780.00	C1
bqp_40_4x8	116	520.00	0.00	0.00	-67880.00	C1
bqp_40_5xR	96	480.00	6.61	6.61	-800.00	C1
bqp_40_7x5	30	180.00	20.00	20.00	-41420.00	C1

bqp_45_2x15	780	4002.00	92.00	92.00	-739496.00	C1
bqp_45_2xR	1014	667.00	5.69	5.67	-234669.00	C2
bqp_45_3x9	290	1840.00	0.00	0.00	-183954.00	C1
bqp_45_3xR	469	6049.00	0.10	0.05	-253644.00	C1
bqp_45_4x9	242	1518.00	46.00	46.00	-105800.00	C1
bqp_45_4xR	260	1656.00	0.00	0.00	-73853.00	C1
bqp_45_5xR	172	1150.00	53.03	53.02	-23253.00	C1
bqp_45_8x5	40	368.00	0.00	0.00	-48530.00	C1
bqp_50_2xR	842	5675.00	75.00	75.00	-183075.00	C1
bqp_50_3x10	255	1075.00	0.35	0.35	-103075.00	C1
bqp_50_3xR	296	1725.00	6.42	6.41	-24650.00	C1
bqp_50_4x10	105	1450.00	0.00	0.00	-3600.00	C1
bqp_50_4xR	155	850.00	0.00	0.00	-389725.00	C1
bqp_50_6xR	80	1150.00	0.00	0.00	-190725.00	C1
bqp_50_7x5	40	1375.00	0.00	0.00	-58100.00	C1
bqp_50_9x5	40	525.00	0.00	0.00	-1550.00	C1

Table 11: SDP and DDP bounds for BQP.

3.6.4 *Results: OI*

Again we start off commenting the results related to the OI reformulation process. Table 12 reports, per instance, the number of variables (n) and orbits ($|\Omega_{G_P}|$) of the original formulation, and the total number of variables indexed by the orbits Ω_{G_P} (#svar); for each OI narrowing type, the table reports the size of the maximum clique ($|\Omega_K|$), the size of the largest independent set ($|\Omega_I|$), the total number of variables indexed by all the orbits in Ω_I (#var), the number of weak (#wea) and strong (#str) SBCs generated, and the parameters σ, ρ and υ, which were described in Section 2.5.4.

As the results in Table 12 indicate, the BQPs generated according to what was exposed in Section 3.5 are highly symmetric. In general, we observe that $(\sigma, \rho, \upsilon) = ([0.5, 1], 1, 1)$ holds for all cases, meaning that at least half of the variables of each instance belongs to an orbit ($\sigma \in [0.5, 1]$) and that we explore them all ($(\rho, \upsilon) = (1, 1)$). Moreover, the BQPs were generated so as to guarantee that every orbit satisfies the conditions in Proposition 28; as one may also observe in Table 12, we could build nothing but strong SBCs for all instances.

	Original formulation			OI-narrowing							
Instance	n	$\lvert\Omega_{G_P}\rvert$	#svar	$\lvert\Omega_K\rvert$	$\lvert\Omega_I\rvert$	#var	#wea	#str	σ	ρ	υ
bqp_55_2xR	55	2	37	2	2	37	0	35	.67	1.00	1.00
bqp_55_3xR	55	3	40	3	3	40	0	37	.72	1.00	1.00
bqp_55_4x11	55	4	44	4	4	44	0	40	.80	1.00	1.00
bqp_55_4xR	55	4	46	4	4	46	0	42	.83	1.00	1.00
bqp_55_5xR	55	5	55	5	5	55	0	50	1.00	1.00	1.00
bqp_55_6xR	55	6	45	6	6	45	0	39	.81	1.00	1.00

bqp_60_2x20	60	2	40	2	2	40	0	38	.66	1.00	1.00
bqp_60_2xR	60	2	45	2	2	45	0	43	.75	1.00	1.00
bqp_60_3x15	60	3	45	3	3	45	0	42	.75	1.00	1.00
bqp_60_3x20	60	3	60	3	3	60	0	57	1.00	1.00	1.00
bqp_60_3xR	60	3	47	3	3	47	0	44	.78	1.00	1.00
bqp_60_4x12	60	4	48	4	4	48	0	44	.80	1.00	1.00
bqp_60_4xR	60	4	47	4	4	47	0	43	.78	1.00	1.00
bqp_60_5x10	60	5	50	5	5	50	0	45	.83	1.00	1.00
bqp_60_5xR	60	5	49	5	5	49	0	44	.81	1.00	1.00
bqp_65_11x5	65	11	55	11	11	55	0	44	.84	1.00	1.00
bqp_65_2xR	65	2	47	2	2	47	0	45	.72	1.00	1.00
bqp_65_3xR	65	3	49	3	3	49	0	46	.75	1.00	1.00
bqp_65_4x13	65	4	52	4	4	52	0	48	.80	1.00	1.00
bqp_65_4xR	65	4	54	4	4	54	0	50	.83	1.00	1.00
bqp_65_5x13	65	5	65	5	5	65	0	60	1.00	1.00	1.00
bqp_65_5xR	65	5	54	5	5	54	0	49	.83	1.00	1.00
bqp_65_6xR	65	6	54	6	6	54	0	48	.83	1.00	1.00
bqp_70_2xR	70	2	49	2	2	49	0	47	.70	1.00	1.00
bqp_70_3xR	70	3	45	3	3	45	0	42	.64	1.00	1.00
bqp_70_4x14	70	4	56	4	4	56	0	52	.80	1.00	1.00
bqp_70_4xR	70	4	70	4	4	70	0	66	1.00	1.00	1.00
bqp_70_5xR	70	5	58	5	5	58	0	53	.82	1.00	1.00
bqp_70_6x10	70	6	60	6	6	60	0	54	.85	1.00	1.00
bqp_70_7xR	70	7	63	7	7	63	0	56	.90	1.00	1.00
bqp_70_9x7	70	9	63	9	9	63	0	54	.90	1.00	1.00
bqp_75_2x25	75	2	50	2	2	50	0	48	.66	1.00	1.00
bqp_75_2xR	75	2	39	2	2	39	0	37	.52	1.00	1.00
bqp_75_3xR	75	3	60	3	3	60	0	57	.80	1.00	1.00
bqp_75_4x15	75	4	60	4	4	60	0	56	.80	1.00	1.00
bqp_75_4xR	75	4	54	4	4	54	0	50	.72	1.00	1.00
bqp_75_5x15	75	5	75	5	5	75	0	70	1.00	1.00	1.00
bqp_75_5xR	75	5	66	5	5	66	0	61	.88	1.00	1.00
bqp_75_6xR	75	6	67	6	6	67	0	61	.89	1.00	1.00
bqp_75_7xR	75	7	63	7	7	63	0	56	.84	1.00	1.00
bqp_75_8xR	75	8	66	8	8	66	0	58	.88	1.00	1.00
bqp_80_2x20	80	2	40	2	2	40	0	38	.50	1.00	1.00
bqp_80_2xR	80	2	55	2	2	55	0	53	.68	1.00	1.00
bqp_80_3x20	80	3	60	3	3	60	0	57	.75	1.00	1.00
bqp_80_3xR	80	3	64	3	3	64	0	61	.80	1.00	1.00
bqp_80_4x16	80	4	64	4	4	64	0	60	.80	1.00	1.00
bqp_80_4xR	80	4	65	4	4	65	0	61	.81	1.00	1.00
bqp_80_5x16	80	5	80	5	5	80	0	75	1.00	1.00	1.00
bqp_80_5xR	80	5	70	5	5	70	0	65	.87	1.00	1.00
bqp_80_6xR	80	6	72	6	6	72	0	66	.90	1.00	1.00
bqp_80_7x10	80	7	70	7	7	70	0	63	.87	1.00	1.00
bqp_80_8xR	80	8	68	8	8	68	0	60	.85	1.00	1.00
bqp_85_12x5	85	12	60	12	12	60	0	48	.70	1.00	1.00
bqp_85_16x5	85	16	80	16	16	80	0	64	.94	1.00	1.00
bqp_85_2x17	85	2	34	2	2	34	0	32	.40	1.00	1.00
bqp_85_2xR	85	2	59	2	2	59	0	57	.69	1.00	1.00
bqp_85_3xR	85	3	68	3	3	68	0	65	.80	1.00	1.00

bqp_85_4x17	85	4	68	4	4	68	0	64	.80	1.00	1.00
bqp_85_4xR	85	4	67	4	4	67	0	63	.78	1.00	1.00
bqp_85_5xR	85	5	64	5	5	64	0	59	.75	1.00	1.00
bqp_85_6xR	85	6	75	6	6	75	0	69	.88	1.00	1.00
bqp_85_7xR	85	7	76	7	7	76	0	69	.89	1.00	1.00
bqp_85_8xR	85	8	80	8	8	80	0	72	.94	1.00	1.00
bqp_85_9xR	85	9	85	9	9	85	0	76	1.00	1.00	1.00
bqp_90_2x30	90	2	60	2	2	60	0	58	.66	1.00	1.00
bqp_90_2xR	90	2	65	2	2	65	0	63	.72	1.00	1.00
bqp_90_3x30	90	3	90	3	3	90	0	87	1.00	1.00	1.00
bqp_90_3xR	90	3	75	3	3	75	0	72	.83	1.00	1.00
bqp_90_4x18	90	4	72	4	4	72	0	68	.80	1.00	1.00
bqp_90_4xR	90	4	73	4	4	73	0	69	.81	1.00	1.00
bqp_90_5x15	90	5	75	5	5	75	0	70	.83	1.00	1.00
bqp_90_5xR	90	5	77	5	5	77	0	72	.85	1.00	1.00
bqp_90_6xR	90	6	76	6	6	76	0	70	.84	1.00	1.00
bqp_90_7xR	90	7	70	7	7	70	0	63	.77	1.00	1.00
bqp_90_8x10	90	8	80	8	8	80	0	72	.88	1.00	1.00
bqp_90_9x9	90	9	81	9	9	81	0	72	.90	1.00	1.00
bqp_95_18x5	95	18	90	18	18	90	0	72	.94	1.00	1.00
bqp_95_2xR	95	2	51	2	2	51	0	49	.53	1.00	1.00
bqp_95_3xR	95	3	77	3	3	77	0	74	.81	1.00	1.00
bqp_95_4x19	95	4	76	4	4	76	0	72	.80	1.00	1.00
bqp_95_4xR	95	4	90	4	4	90	0	86	.94	1.00	1.00
bqp_95_5xR	95	5	88	5	5	88	0	83	.92	1.00	1.00
bqp_95_6xR	95	6	89	6	6	89	0	83	.93	1.00	1.00
bqp_95_7xR	95	7	95	7	7	95	0	88	1.00	1.00	1.00
bqp_95_8xR	95	8	95	8	8	95	0	87	1.00	1.00	1.00
bqp_95_9xR	95	9	86	9	9	86	0	77	.90	1.00	1.00
bqp_100_2xR	100	2	70	2	2	70	0	68	.70	1.00	1.00
bqp_100_3x25	100	3	75	3	3	75	0	72	.75	1.00	1.00
bqp_100_3xR	100	3	77	3	3	77	0	74	.77	1.00	1.00
bqp_100_4x20	100	4	80	4	4	80	0	76	.80	1.00	1.00
bqp_100_4xR	100	4	81	4	4	81	0	77	.81	1.00	1.00
bqp_100_5x20	100	5	100	5	5	100	0	95	1.00	1.00	1.00
bqp_100_5xR	100	5	93	5	5	93	0	88	.93	1.00	1.00
bqp_100_6xR	100	6	96	6	6	96	0	90	.96	1.00	1.00
bqp_100_7xR	100	7	88	7	7	88	0	81	.88	1.00	1.00
bqp_100_8xR	100	8	89	8	8	89	0	81	.89	1.00	1.00
bqp_100_9x10	100	9	90	9	9	90	0	81	.90	1.00	1.00

Table 12: OI narrowings of symmetric BQPs. 0* indicates values of $O(10^{-3})$ or less.

Some comments about the optimization results are in order. Table 13 presents the aggregated statistics for the BQP dataset. In the majority of the cases, 67 out of 97, the narrowings performed better, against 27 of the original formulations; in three cases, the SBCs usage was indifferent. We were expecting a better ratio in favor of the narrowings. Note however the large difference in terms of execution

	Original formulation		OI-narrowing	
Dataset	# Best	Time (h)	# Best	Time (s)
BQP	27	12.22	67	19.84

Table 13: Aggregated solution statistics for the BQP dataset.

time: 12 hours in total for the original problems against 20 seconds for the OI narrowings. Table 14 exhibits detailed results. All instances were solved to optimality.

	Original formulation				OI-narrowing			
Instance	Best	Time (s)	Gap (%)	St.	Best	Time (s)	Gap (%)	St.
bqp_55_2xR	2708	137.96	0	opt	2708	**0.05**	0	opt
bqp_55_3xR	232	**0.05**	0	opt	232	0.07	0	opt
bqp_55_4x11	846	3.88	0	opt	846	**0.08**	0	opt
bqp_55_4xR	219	**0.04**	0	opt	219	0.07	0	opt
bqp_55_5xR	285	2.07	0	opt	285	**0.08**	0	opt
bqp_55_6xR	63	**0.02**	0	opt	63	0.05	0	opt
bqp_60_2x20	1443	10.98	0	opt	1443	**1.44**	0	opt
bqp_60_2xR	1189	0.08	0	opt	1189	**0.05**	0	opt
bqp_60_3x15	60	0.01	0	opt	60	0.01	0	opt
bqp_60_3x20	4790	1757.17	0	opt	4790	**0.12**	0	opt
bqp_60_3xR	1485	2.05	0	opt	1485	**0.08**	0	opt
bqp_60_4x12	420	2.08	0	opt	420	**0.07**	0	opt
bqp_60_4xR	240	0.21	0	opt	240	**0.10**	0	opt
bqp_60_5x10	180	**0.02**	0	opt	180	0.06	0	opt
bqp_60_5xR	212	0.11	0	opt	212	**0.09**	0	opt
bqp_65_11x5	77	0.21	0	opt	77	**0.11**	0	opt
bqp_65_2xR	1137	0.15	0	opt	1137	**0.10**	0	opt
bqp_65_3xR	303	**0.06**	0	opt	303	0.08	0	opt
bqp_65_4x13	1717	11.34	0	opt	1717	**0.11**	0	opt
bqp_65_4xR	266	0.12	0	opt	266	**0.08**	0	opt
bqp_65_5x13	241	2.09	0	opt	241	**0.10**	0	opt
bqp_65_5xR	236	**0.04**	0	opt	236	0.08	0	opt
bqp_65_6xR	204	0.35	0	opt	204	**0.10**	0	opt
bqp_70_2xR	580	7212.02	14.69	lim	580	**0.14**	0	opt
bqp_70_3xR	1132	2.04	0	opt	1132	**0.58**	0	opt
bqp_70_4x14	427	2.85	0	opt	427	**0.13**	0	opt
bqp_70_4xR	648	16.13	0	opt	648	**0.12**	0	opt
bqp_70_5xR	197	**0.12**	0	opt	197	0.17	0	opt
bqp_70_6x10	155	0.68	0	opt	155	**0.13**	0	opt
bqp_70_7xR	112	**0.05**	0	opt	112	0.09	0	opt
bqp_70_9x7	70	**0.02**	0	opt	70	0.05	0	opt
bqp_75_2x25	763	**0.07**	0	opt	763	0.08	0	opt
bqp_75_2xR	646	**0.05**	0	opt	646	0.11	0	opt
bqp_75_3xR	651	**0.06**	0	opt	651	0.17	0	opt
bqp_75_4x15	949	14.61	0	opt	949	**0.20**	0	opt
bqp_75_4xR	981	0.26	0	opt	981	**0.11**	0	opt
bqp_75_5x15	931	93.00	0	opt	931	**0.15**	0	opt

bqp_75_5xR	604	0.84	0	opt	604	**0.20**	0	opt
bqp_75_6xR	210	**0.07**	0	opt	210	0.12	0	opt
bqp_75_7xR	172	**0.06**	0	opt	172	0.09	0	opt
bqp_75_8xR	100	**0.06**	0	opt	100	0.08	0	opt
bqp_80_2x20	500	**0.01**	0	opt	500	0.02	0	opt
bqp_80_2xR	1760	59.90	0	opt	1760	**0.06**	0	opt
bqp_80_3x20	100	**0.01**	0	opt	100	0.02	0	opt
bqp_80_3xR	853	0.25	0	opt	853	**0.24**	0	opt
bqp_80_4x16	976	75.01	0	opt	976	**0.18**	0	opt
bqp_80_4xR	715	3.42	0	opt	715	**0.19**	0	opt
bqp_80_5x16	936	27.07	0	opt	936	**0.40**	0	opt
bqp_80_5xR	305	0.66	0	opt	305	**0.22**	0	opt
bqp_80_6xR	78	**0.04**	0	opt	78	0.12	0	opt
bqp_80_7x10	170	**0.04**	0	opt	170	0.07	0	opt
bqp_80_8xR	100	**0.06**	0	opt	100	0.09	0	opt
bqp_85_12x5	147	1.25	0	opt	147	**0.66**	0	opt
bqp_85_16x5	69	1.76	0	opt	69	**1.06**	0	opt
bqp_85_2x17	2924	0.04	0	opt	2924	0.04	0	opt
bqp_85_2xR	5189	3.98	0	opt	5189	**0.06**	0	opt
bqp_85_3xR	695	649.55	0	opt	695	**0.10**	0	opt
bqp_85_4x17	714	27.94	0	opt	714	**0.15**	0	opt
bqp_85_4xR	1374	61.84	0	opt	1374	**0.21**	0	opt
bqp_85_5xR	827	1.75	0	opt	827	**0.86**	0	opt
bqp_85_6xR	233	1.65	0	opt	233	**0.17**	0	opt
bqp_85_7xR	141	0.07	0	opt	141	**0.06**	0	opt
bqp_85_8xR	160	0.65	0	opt	160	**0.06**	0	opt
bqp_85_9xR	52	0.14	0	opt	52	**0.10**	0	opt
bqp_90_2x30	9420	7212.55	35.51	lim	9420	**0.11**	0	opt
bqp_90_2xR	1872	7212.53	32.07	lim	1872	**0.08**	0	opt
bqp_90_3x30	6585	7212.67	34.19	lim	6585	**0.16**	0	opt
bqp_90_3xR	2735	4.49	0	opt	2735	**0.18**	0	opt
bqp_90_4x18	576	40.24	0	opt	576	**0.11**	0	opt
bqp_90_4xR	1047	3.22	0	opt	1047	**0.22**	0	opt
bqp_90_5x15	225	**0.01**	0	opt	225	0.03	0	opt
bqp_90_5xR	215	0.61	0	opt	215	**0.26**	0	opt
bqp_90_6xR	183	**0.12**	0	opt	183	0.17	0	opt
bqp_90_7xR	283	2.90	0	opt	283	**0.55**	0	opt
bqp_90_8x10	925	65.20	0	opt	925	**1.17**	0	opt
bqp_90_9x9	117	**0.04**	0	opt	117	0.06	0	opt
bqp_95_18x5	95	2.22	0	opt	95	**1.40**	0	opt
bqp_95_2xR	4226	3.26	0	opt	4226	**0.11**	0	opt
bqp_95_3xR	1843	8.24	0	opt	1843	**0.12**	0	opt
bqp_95_4x19	636	29.29	0	opt	636	**0.09**	0	opt
bqp_95_4xR	480	0.08	0	opt	480	**0.07**	0	opt
bqp_95_5xR	468	**0.07**	0	opt	468	0.37	0	opt
bqp_95_6xR	220	0.05	0	opt	220	**0.04**	0	opt
bqp_95_7xR	468	1.33	0	opt	468	**0.17**	0	opt
bqp_95_8xR	1425	3194.67	0	opt	1425	**0.12**	0	opt
bqp_95_9xR	209	1.06	0	opt	209	**0.29**	0	opt
bqp_100_2xR	8606	7211.99	43.86	lim	8606	**0.32**	0	opt
bqp_100_3x25	700	0.12	0	opt	700	**0.07**	0	opt

bqp_100_3xR	6942	186.10	0	opt	6942	**0.49**	0	opt
bqp_100_4x20	4230	1238.27	0	opt	4230	**0.22**	0	opt
bqp_100_4xR	400	1.36	0	opt	400	**0.16**	0	opt
bqp_100_5x20	1280	161.01	0	opt	1280	**0.79**	0	opt
bqp_100_5xR	884	**0.16**	0	opt	884	0.31	0	opt
bqp_100_6xR	725	**0.09**	0	opt	725	0.23	0	opt
bqp_100_7xR	358	**0.10**	0	opt	358	0.17	0	opt
bqp_100_8xR	294	0.16	0	opt	294	**0.14**	0	opt
bqp_100_9x10	70	0.02	0	opt	70	0.02	0	opt

Table 14: BQP results obtained with CPLEX 12.6.

	Original formulation			
Instance	Best	Time (s)	Gap (%)	St.
bqp_70_2xR	580	7212.03	14.62	lim
bqp_90_2x30	9420	7212.37	35.51	lim
bqp_90_2xR	1872	7212.13	32.07	lim
bqp_90_3x30	6585	7212.88	34.19	lim
bqp_100_2xR	8606	7212.29	43.86	lim

Table 15: Extended results for hard BQP instances obtained with CPLEX 12.6.

In five occasions, CPLEX could not reach optimality: bqp_70_2xR, bqp_90_2x30, bqp_90_2xR, bqp_90_3x30 and bqp_100_2xR. Given the level of difficulty displayed by these BQPs, we decided to perform a second round of tests restricted to them, but now using CPLEX's full power in terms of symmetry exploitation, setting the parameter *symmetry* of CPLEX's API to level 5. The results are shown in Table 15. Remarkably, there is no significative change in the final gaps when compared with the previous results, meaning that these five instances are indeed hard to solve, except if one employs the OI reformulations.

3.7 CONCLUSIONS

In this entr'acte we have continued to examine the impact of symmetries in the context of Binary Quadratic Programming and Semidefinite Programming. At first, our findings indicate that it may not be necessary to consider Symmetry-Breaking Constraints when solving Semidefinite Programs. But these results represent a first attempt in this sense and involve a particular setting. So it remains important to conduct further investigations in the topic. In order to carry out our experiments, we have also established a procedure to generate symmetric Binary Quadratic Programs. In particular, these Binary Quadratic Programs have proved to be very relevant to the Orbital Independence Theory since they embed the conditions under which the use of Symmetry-Breaking Constraints is majoritarily advantageous.

4 EUCLIDEAN DISTANCE GEOMETRY PROBLEM

Entering the Distance Geometry subject, in this chapter we cope with the most fundamental problem arising in the field of Distance Geometry, the one of realizing graphs in Euclidean spaces: it asks to find a realization of an edge-weighted undirected graph in $\mathbb{R}^K$ for some given K such that the positions for adjacent vertices respect the distance given by the corresponding edge weight. The Euclidean Distance Geometry Problem is of great importance since it has many applications to science and engineering. It is notoriously difficult to solve computationally, and most of the methods proposed so far either do not scale up to useful sizes, or unlikely identify good solutions. In fact, the need to constrain the rank of the matrix representing feasible solutions of the Euclidean Distance Geometry Problem is what makes the problem so hard. In view of overcoming such issues, we propose a two-steps heuristic algorithm based on Semidefinite Programming (or, precisely, based on the recent Diagonally Dominant Programming paradigm) and the explicitly modelling of Rank Constraints. We analyze our method via extensive computational testing against randomly generated instances and against feasible realistic protein conformation instances taken from the Protein Data Bank.

4.1 INTRODUCTION

In this chapter we are interested in solving the following decision problem:

> Euclidean Distance Geometry Problem (EDGP). Given an integer $K \geqslant 1$ and a simple, edge-weighted, undirected graph $G = (V, E, d)$, where $d : E \to \mathbb{R}_+$, is there a *realization function* $x : V \to \mathbb{R}^K$ such that:
>
> $$\forall \{i, j\} \in E \quad \|x_i - x_j\|_2 = d_{ij}? \tag{36}$$

The EDGP is a fundamental problem in Distance Geometry and was formally introduced by Yemini in 1978 [94]. In [81], the problem was shown to be **NP**-hard (with 2-norm and $K = 1$) by reduction from Partition. Nevertheless, due to its vast range of practical applications, the three most investigated being clock syncronization ($K = 1$) [83], sensor network localization ($K = 2$) [85] and molecular conformation ($K = 3$) [44], the problem has attracted the attention of the scientific community in the last decades. For example, very recently, the EDGP was applied to model and solve special Graph Coloring Problems (GCPs) involving distance constraints as weighted edges [31]. We refer to [58] for an extensive survey on the EDGP subject.

Usually, the squared version of Eq. (36) is employed, for two reasons: first, as pointed out in [25], the squared EDM $D^2 = (d_{ij}^2)$ has rank at most $K+2$, a fact which might come in useful for all sorts of reasons (e. g. to devise cuts for the ESTP); secondly, and most important, since the vast majority of algorithmic implementations employ floating point representations, as we pointed in Section 1.3, there is a risk that $\sum_k (x_{ik} - x_{jk})^2 = 0$ might be represented by a tiny negative floating point scalar, resulting in a computational error when extracting the square root in Eq. (5) for $p = 2$. Obviously solving the squared system yields exactly the same set of solutions as the original system.

Most MP methods for solving Eq. (36) thus do not address the original system explicitly, but rather a Global Optimization (GO) problem defined by means of a penalty function:

$$\min_{x \in \mathbb{R}^{n \times K}} \sum_{\{i,j\} \in E} (\|x_i - x_j\|_2^2 - d_{ij}^2)^2, \tag{37}$$

which has global optimum x^* with value zero if and only if x^* satisfies Eq. (36). This formulation is convenient because most local NLP solvers find it easier to improve the cost of a feasible nonoptimal solution, rather than achieving feasibility from an infeasible point, and relevant because such solvers are often employed to solve EDGP instances. Moreover, Eq. (37) can be easily adjusted to deal with imprecise distances represented by intervals, which is useful considering that practical methods employed to gauge physical distances (such as the Nuclear Magnetic Resonance (NMR) in the case of molecules, for instance) typically provide bounds on the values of the distances and not precise values. A survey on continuous methods for the EDGP is given in [57].

Semidefinite Programming is quite often used to devise relaxations of EDGPs [3, 85, 93]. In this setting, presented in details in Section 4.3, solutions of the EDGP are usually represented by real symmetric matrices which are required to have rank at most K. As it is widely known, the rank of a matrix $A \in \mathbb{M}^{m \times n}$ is defined as the size of the largest collection of linearly independent columns (or rows) of A. However, requiring linear independence of a set of vectors $\{v_1, ..., v_n\} \subseteq \mathbb{R}^m$ of decision variables of a Mathematical Program is a nontrivial feat since the elementary definition,

$$\forall \alpha \in \mathbb{R}^n \quad \sum_{i=1}^{n} \alpha_i v_i = 0 \Rightarrow \alpha = 0,$$

requires in general uncountably many nonlinear constraints, each involving either binary variables or complementarity terms. In view of this difficulty, it is absolutely reasonable to opt to relax the Rank Constraint instead.

On the other hand, we adventure to explore an equivalent definition of the rank of A (its number of nonzero eigenvalues) and propose general Rank Constraint formulations consisting of a finite set of equations. Perhaps unwisely, because these models are nonlinear and yield MINLPs when adjoined to any MP; and MINLPs represent undoubtedly the most difficult class of MPs to be solved. In our case, based on the characteristics of the EDGP, we manage to at least simplify our MINLPs into nonconvex NLPs. Such MPs are quite challenging to solve anyway, particularly because NLP solvers are complex algorithms sensitive to many factors, notably to the quality of the starting points. Since we are actually dealing with SDPs, our idea is to use DDP to find cheap and hopefully good starting points for the rank constrained SDPs.

We thus propose a simple two-steps heuristic algorithm to tackle the EDGP based on Mathematical Programming: first solve an LP obtained by means of DDP and afterwards use its solution as a starting point to solve the rank constrained SDPs.

The rest of this chapter is organized as follows: in Section 4.2 we introduce notation and recall some classical definitions of Linear Algebra; in Section 4.3 we present some of the many formulations for the EDGP; in Section 4.4 we describe in details all the tools we apply in our heuristic to solve the EDGP; and finally, computational results are provided and discussed in Section 4.5.

4.2 NOTATION AND DEFINITIONS

We start off by reviewing well-known concepts of Linear Algebra. All vectors in this chapter are contained in the vector space $\mathbb{R}^n$ endowed with an inner product, denoted by $\langle \cdot, \cdot \rangle$. We let $\mathbb{R}^{m \times n}$ be the set of $m \times n$ real matrices and $\mathbb{S}^n$ the set of real symmetric matrices of dimension n.

Definition 32. *An orthonormal basis $\mathcal{B}$ of the vector space $\mathbb{R}^n$ is a set of n orthogonal unit vectors that spans $\mathbb{R}^n$.*

Definition 33. *A linear operator $\mathcal{A} : \mathbb{R}^n \to \mathbb{R}^n$ is called* self-adjoint *when $\langle \mathcal{A}u, v \rangle = \langle u, \mathcal{A}v \rangle$ for any vectors $u, v \in \mathbb{R}^n$.*

A self-adjoint linear operator $\mathcal{A}$ may be represented by a matrix $A \in \mathbb{S}^n$ corresponding to an orthonormal basis $\mathcal{B}$ of $\mathbb{R}^n$. We remark that A differs as $\mathcal{B}$ changes.

Definition 34. *A nonzero vector $v \in \mathbb{R}^n$ is an* eigenvector *of a matrix $A \in \mathbb{S}^n$ when there is $\lambda \in \mathbb{R}$ such that*

$$Av = \lambda v.$$

The number λ *is an* eigenvalue *of* A *corresponding to* v.

Theorem 35 (Spectral Theorem)**.** *For any self-adjoint linear operator* $\mathcal{A} : \mathbb{R}^n \to \mathbb{R}^n$*, there is an orthonormal basis for* $\mathbb{R}^n$ *consisting of the eigenvectors of a matrix* $A \in \mathbb{S}^n$.

Let $N = \{1, \dots, n\}$. By the Spectral Theorem, for any matrix $A \in \mathbb{S}^n$, there exists a set of orthonormal eigenvectors $\{v_1, ..., v_n\} \subset \mathbb{R}^n$ satisfying $Av_i = \lambda_i v_i$ for $i \in N$, where $\lambda_i \in \mathbb{R}$. Let $\vartheta \in \mathbb{R}^{n\times n}$ be the square matrix whose columns are the eigenvectors of A and $\Lambda \in \mathbb{S}^n$ be the diagonal matrix whose entries are the eigenvalues of A. Using matrix notation, we can write $A\vartheta = [Av_1\ Av_2\ ...\ Av_n] = [\lambda_1 v_1\ \lambda_2 v_2\ ...\ \lambda_n v_n]$, and finally

$$A\vartheta = \vartheta\Lambda. \tag{38}$$

Eq. (38) constitutes the *eigensystem* of the matrix A. $\mathcal{B}$ is an *eigenbasis* for A. Since ϑ is orthogonal, it has an inverse and $\vartheta^{-1} = \vartheta^\top$. Therefore $A\vartheta\vartheta^{-1} = \vartheta\Lambda\vartheta^{-1}$ and

$$A = \vartheta\Lambda\vartheta^\top. \tag{39}$$

Eq. (39) represents the *eigendecomposition* of A.

Definition 36. *A matrix* $A \in \mathbb{S}^n$ *is said to be Positive Semidefinite (PSD) if* $v^\top A v \geqslant 0$ *for every nonzero vector* $v \in \mathbb{R}^n$*. The standard notation* $A \succeq 0$ *is applied.*

It is well-known that the eigenvalues of PSD matrices are nonnegative. Indeed, if $A \succeq 0$, it follows that $v_i{}^\top A v_i = v_i{}^\top \lambda_i v_i = \lambda_i \|v_i\|_2^2 \geqslant 0 \Rightarrow \lambda_i \geqslant 0$ for $i \in N$.

Definition 37. *A matrix* $A \in \mathbb{S}^n$ *is said to be Positive Definite (PD) if* $v^\top A v > 0$ *for every nonzero vector* $v \in \mathbb{R}^n$*. The standard notation* $A \succ 0$ *is applied.*

Similarly, PD matrices have positive eigenvalues.

Definition 38. *A matrix* $A \in \mathbb{S}^n$ *is called a* Gram matrix *if it arises from a set of vectors* $\{a_1, ..., a_n\} \subseteq \mathbb{R}^m$ *as* $A_{ij} = \langle a_i, a_j \rangle$ *for* $i, j \in N$.

Every Gram matrix A is PSD. In fact, writing $A = a^\top a$, we have that $v^\top a^\top a v = (av)^\top av = \|av\|_2^2 \geqslant 0$ for every nonzero $v \in \mathbb{R}^n$, where $a \in \mathbb{R}^{m\times n}$ is the matrix whose columns are the generating vectors of A.

Definition 39. *Given a matrix* $A \in \mathbb{S}^n$ *and* $i \in N$*, the closed discs* $B(A_{ii}, R_i)$ *centered at* A_{ii} *with radius*

$$R_i = \sum_{\substack{j\in N\\ j\neq i}} |A_{ij}| \tag{40}$$

are called Gershgorin discs.

Theorem 40 (Gershgorin Circle Theorem)**.** *Every eigenvalue of* $A \in \mathbb{S}^n$ *lies within at least one of the Gershgorin discs* $B(A_{ii}, R_i)$*, for* $i \in N$.

As mentioned earlier, it follows from Gershgorin's theorem that all DD matrices are PSD. Indeed, from Eq. (10), $A_{ii} \geqslant R_i \geqslant 0$. Since $\lambda_i \in B(a_{ii}, R_i)$ implies $|\lambda_i - A_{ii}| \leqslant R_i$, we have that $\lambda_i - A_{ii} \geqslant -R_i \Rightarrow \lambda_i \geqslant A_{ii} - R_i \geqslant 0$. But the converse does not hold, i.e. PSD does not imply DD.

Definition 41. *A real symmetric* interval *matrix is defined as the family of matrices*

$$\mathbf{A} = \{A \in \mathbb{S}^n : \underline{A} \leqslant A \leqslant \overline{A}\},$$

where $\underline{A}, \overline{A} \in \mathbb{R}^{n \times n}$ *are given matrices satisfying* $\underline{A} \leqslant \overline{A}$ *and the inequality is considered element-wise.*

The *midpoint* and the *radius* of $\mathbf{A}$ are defined respectively by

$$A_c = \frac{1}{2}(\underline{A} + \overline{A}) \text{ and } A_r = \frac{1}{2}(\overline{A} - \underline{A}).$$

The spectral radius (i.e. the supremum among the absolute values of all eigenvalues) is denoted by $\rho(\cdot)$. Moreover, consider the matrices $A_C = \frac{1}{2}(A_c + {A_c}^\top)$ and $A_R = \frac{1}{2}(A_r + {A_r}^\top)$. Let $\lambda_i(A)$ denote the ith eigenvalue of A.

Theorem 42 ([35], Thm. 1)**.** *Let* $\mathbf{A}$ *be a real symmetric interval matrix. For* $i \in N$,

$$\lambda_i(\mathbf{A}) \subseteq [\lambda_i(A_C) - \rho(A_R), \lambda_i(A_C) + \rho(A_R)]. \tag{41}$$

4.3 EDGP FORMULATIONS

In this section we present the feasibility formulation that we aim to solve and one of the MP formulations that we use in our heuristic.

4.3.1 *Feasibility formulations*

We represent a realization x as a matrix $x \in \mathbb{R}^{n \times K}$ where each of the $n = |V|$ rows is a vector $x_i \in \mathbb{R}^K$ that gives the position of vertex $i \in V$.

It was shown in [85] that the system of constraints given by Eq. (36) can be reformulated exactly to a pure feasibility MP as

$$\left.\begin{array}{ll} \forall \{i,j\} \in E & (e_i - e_j)^\top X (e_i - e_j) = d_{ij}^2, \\ & X = xx^\top, \end{array}\right\} \tag{42}$$

where $e_i \in \mathbb{R}^n$ is the i-th canonical unit vector.

From this point forward, the conventional way to pursue would be to derive a SDP relaxation of Eq. (42) by carrying out the same procedure described in Section 3.3, and solve the resulting formulation via standard SDP algorithms.

In the opposite way, we want to solve Eq. (42) exactly. In order to fulfill our goal, from the definition of X, we first observe that $X \in \mathbb{S}^n$ and $\operatorname{rk} X = \operatorname{rk} x \leqslant \min\{n; K\}$. In addition, since X is a Gram matrix, it is PSD. These facts permit us to reformulate Eq. (42) into

$$\left.\begin{array}{ll} \forall\{i,j\} \in E & (e_i - e_j)^\top X (e_i - e_j) = d_{ij}^2, \\ & \operatorname{rk} X \leqslant \min\{n; K\}, \\ & X \succeq 0. \end{array}\right\} \tag{43}$$

The mapping between solutions of Eq. (42) and Eq. (43) is simple: if there is X satisfying Eq. (43), there are vectors $x_1, \ldots, x_n \in \mathbb{R}^K$ satisfying Eq. (42) with $\langle x_i, x_j \rangle = X_{ij}$ holding for all $\{i,j\} \in E$.

There are polynomial time algorithms to retrieve the vectors x associated to a solution X. Briefly, we can use Eq. (39) to write $X = \vartheta \Lambda \vartheta^\top$. Since $X \succeq 0$, Λ has a nonnegative diagonal and $\sqrt{\Lambda}$ exists; so we can perform Principal Component Analysis (PCA) and factor X into $x = \vartheta \sqrt{\Lambda_\phi^+}$, where ϑ is the eigenvector matrix of X and Λ_ϕ^+ is the diagonal matrix with the $\phi = \min\{n; K\}$ largest positive eigenvalues and zeros elsewhere; hence our interest in solving Eq. (43) exactly.

4.3.2 *MP formulations*

The EDGP is solved using MP techniques commonly by means of error minimization formulations. Apart from Eq. (37), there are several equivalent alternatives [20, 66]. Due to our own empirical evidences, we are particularly interested in the following GO problem:

$$\min_{x \in \mathbb{R}^{n \times K}} \sum_{\{i,j\} \in E} \left| \|x_i - x_j\|_2^2 - (d_{ij})^2 \right|. \tag{44}$$

Similarly to what happens to Eq. (37), we are interested in finding global optima of Eq. (44), which is a nonconvex NLP. Nonlinear Programs are challenging to solve in practical terms, not only because they do not scale well as instances increase in size, but also due to numerical issues. The Rank Constraint models presented in Section 4.4.2 adjoin nonlinear constraints to whatever formulation one chooses as base model. Therefore, prior to using Eq. (44) to derive a MP equivalent to Eq. (43), we first derive a simpler version of Eq. (44) with respect to nonlinearities: we reformulate it into a LP using SDP.

In that regard, we start off remodelling the absolute value function by defining a continuous nonnegative symmetric variable $S = (S_{ij})$ as

$$S_{ij} = \left| \|x_i - x_j\|_2^2 - (d_{ij})^2 \right|,$$

which measures the deviation from the computed and the respective given distance for all $\{i,j\} \in E$. Then using the definition of the absolute value function ($y = |a| \Rightarrow y = a \vee y = -a$) and the fact that Eq. (44) is a minimization problem, we formulate the following constrained NLP:

$$\left.\begin{array}{ll} \min\limits_{x \in \mathbb{R}^{n \times K}, S \in \mathbb{S}^n} & \sum\limits_{\{i,j\} \in E} S_{ij} \\ \forall \{i,j\} \in E & S_{ij} \geqslant \|x_i - x_j\|_2^2 - (d_{ij})^2, \\ \forall \{i,j\} \in E & S_{ij} \geqslant -\|x_i - x_j\|_2^2 + (d_{ij})^2, \\ & S \geqslant 0. \end{array}\right\} \tag{45}$$

The contraints in Eq. (45) together guarantee that the objective function (total deviation) tends to zero through positive values whenever $\|x_i - x_j\|_2^2 \neq (d_{ij})^2$ for any $\{i,j\} \in E$. Now we use the squared expansion of the Euclidean norm

$$\|x_i - x_j\|_2^2 = \langle (x_i - x_j), (x_i - x_j) \rangle = \langle x_i, x_i \rangle - 2\langle x_i, x_j \rangle + \langle x_j, x_j \rangle$$

and the fact that the solutions of Eq. (42) and Eq. (43) are related by $X_{ij} = \langle x_i, x_j \rangle$ (and also that X is symmetric) to obtain the LP relaxation:

$$\left.\begin{array}{ll} \min\limits_{X, S \in \mathbb{S}^n} & \sum\limits_{\{i,j\} \in E} S_{ij} \\ \forall \{i,j\} \in E & S_{ij} \geqslant X_{ii} - 2X_{ij} + X_{jj} - (d_{ij})^2, \\ \forall \{i,j\} \in E & S_{ij} \geqslant -X_{ii} + 2X_{ij} - X_{jj} + (d_{ij})^2, \\ & S \geqslant 0. \end{array}\right\} \tag{46}$$

Besides being a LP, we remark that two positive features of the formulation above are (a) the fact that all of its constraints (except the symmetry constraints $X, S \in \mathbb{S}^n$) are inequalities, and (b) the fact that we can conveniently check the quality of our solutions by direct comparison of their objective function value with the global optimum value zero. A drawback is that Eq. (46) actually has twice more constraints (two per edge) when compared to the LPs that we could derive from alternative EDGP formulations.

It remains however to model the rank and PSD constraints over X to attain the MP equivalent to Eq. (43). This contribution is adressed in the next section.

4.4 EFFORTS TO SOLVE THE EDGP

This section describes the ingredients which we put together in our two-steps heuristic; first we derive the LP that will provide the starting points; after we present our modelling regarding the Rank Constraints; in the sequence we describe how we compute bounds for the variables of our models and present some extra (and hopefully helpful to improve performance) constraints; finally we explain how we exploit characteristics of the EDGP to simplify our MINLPs into NLPs, and how we tweak our heuristic in terms of accuracy so as to reduce computation times when dealing with large EDGP instances.

4.4.1 *Starting points via DDP*

Aiming to find good initial solutions, we derive a LP formulation that inner-approximates a SDP relaxation of Eq. (42) based on diagonal dominance (see Section 1.4.1). As mentioned previously, the relaxation is obtained by relaxing the Rank Constraint to $X - xx^\top \succeq 0$. We then use once more the fact that

$$X - xx^\top \succeq 0 \iff \begin{pmatrix} I_K & x^\top \\ x & X \end{pmatrix} \succeq 0$$

to get the pure feasibility SDP:

$$\left.\begin{aligned} \forall \{i,j\} \in E \quad & X_{ii} + X_{jj} - 2X_{ij} = d_{ij}^2, \\ & \begin{pmatrix} I_K & x^\top \\ x & X \end{pmatrix} \succeq 0. \end{aligned}\right\} \tag{47}$$

This means that

$$\left.\begin{aligned} \forall \{i,j\} \in E \quad & X_{ii} + X_{jj} - 2X_{ij} = d_{ij}^2, \\ & \begin{pmatrix} I_K & x^\top \\ x & X \end{pmatrix} \text{ is DD} \end{aligned}\right\} \tag{48}$$

is a DDP formulation that inner approximates Eq. (47). Now recall that we can linerize the DD constraint. We exploit this idea to derive a new DDP formulation related to the EDGP, which is in fact an LP for the EDGP. Let

$$Y = \begin{pmatrix} I_K & x^\top \\ x & X \end{pmatrix}$$

and consider a continuous nonnegative variable $T \in \mathbb{S}^{n+K}$, then:

$$\left.\begin{aligned} \forall \{i,j\} \in E \quad & X_{ii} + X_{jj} - 2X_{ij} = d_{ij}^2 \\ & Y = \begin{pmatrix} I_K & x^\top \\ x & X \end{pmatrix}, \\ \forall i \in [n+K] \quad & Y_{ii} = \sum_{\substack{j \in [n+K] \\ j \neq i}} T_{ij}, \\ & T \geqslant Y \geqslant -T, \\ & T \geqslant 0. \end{aligned}\right\} \tag{49}$$

Note that the only existing LP formulation for the EDGP is the linear relaxation of Eq. (37), in which every monomial $m(x)$ of the quartic polynomial in the objective is linearized to a variable μ subject to a linear outer approximation of the nonconvex constraint $\mu = m(x)$. It is well-known [43] that this relaxation is much weaker than the obvious (tight) lower bound zero. Although the new formulation Eq. (49) is linear, it may not necessaryly improve this situation since it actually inner approximates the SDP relaxation given by Eq. (47).

Imediatelly, if we consider a continuous variable $Z \in \mathbb{S}^n$, we can state the iterative form of Eq. (49) as:

$$\left.\begin{array}{rl} \forall\{i,j\} \in E & X_{ii} + X_{jj} - 2X_{ij} = d_{ij}^2 \\ & Y = \begin{pmatrix} I_K & x^\top \\ x & X \end{pmatrix} \\ & Y = U^\top Z U, \\ \forall i \in [n+K] & Z_{ii} \geqslant \sum\limits_{\substack{j \in [n+K] \\ j \neq i}} T_{ij}, \\ & T \geqslant Z \geqslant -T, \\ & T \geqslant 0. \end{array}\right\} \tag{50}$$

These are still pure feasibility MPs. We then use Eq. (46) to cast Equations (49) and (50) as LPs as follows:

$$\left.\begin{array}{rl} \min\limits_{X,S \in \mathbb{S}^n} & \sum\limits_{\{i,j\} \in E} S_{ij} \\ \forall\{i,j\} \in E & S_{ij} \geqslant X_{ii} - 2X_{ij} + X_{jj} - (d_{ij})^2, \\ \forall\{i,j\} \in E & S_{ij} \geqslant -X_{ii} + 2X_{ij} - X_{jj} + (d_{ij})^2, \\ & Y = \begin{pmatrix} I_K & x^\top \\ x & X \end{pmatrix}, \\ \forall i \in [n+K] & Y_{ii} \geqslant \sum\limits_{\substack{j \in [n+K] \\ j \neq i}} T_{ij}, \\ & T \geqslant Y \geqslant -T, \\ & S, T \geqslant 0, \end{array}\right\} \tag{51}$$

and

$$\left.\begin{array}{rl}
\min\limits_{X,S\in\mathbb{S}^n} & \sum\limits_{\{i,j\}\in E} S_{ij} \\
\forall\{i,j\}\in E & S_{ij} \geqslant X_{ii} - 2X_{ij} + X_{jj} - (d_{ij})^2, \\
\forall\{i,j\}\in E & S_{ij} \geqslant -X_{ii} + 2X_{ij} - X_{jj} + (d_{ij})^2, \\
 & Y = \begin{pmatrix} I_K & x^\top \\ x & X \end{pmatrix}, \\
 & Y = U^\top Z U, \\
\forall i \in [n+K] & Z_{ii} \geqslant \sum\limits_{\substack{j\in[n+K]\\ j\neq i}} T_{ij}, \\
 & T \geqslant Z \geqslant -T, \\
 & S, T \geqslant 0.
\end{array}\right\} \qquad (52)$$

Our implementation of the iterative DDP method for the EDGP is similar to the one described in Section 3.4 as concerns the termination criteria and the feasibility recovery subroutine.

As stated previously, it is common knowledge that local NLP solvers find it easier to improve the cost of a feasible nonoptimal solution, rather than achieving feasibility from an infeasible point. So it is important to point out that a solution X' computed by solving either Eq. (51) or Eq. (52) is not necessarily feasible in Eq. (42): the interior of the PSD cone contains PD matrices, which have full rank, and so $\text{rk}\, X' > \min\{n; K\}$ may hold. We thereby employ a global NLP solver instead of a local one to handle our rank constrained models, hoping to better exploit the cheap but most likely rank infeasible (with respect to Eq. (43)) solutions obtained via the LPs above. In this sense, the second step of our method may be interpreted as a rank reduction procedure.

4.4.2 *Modelling the rank*

In Sections 4.4.2.1 and 4.4.2.2 the rank is modeled as the number of nonzero eigenvalues of $X \in \mathbb{S}^n$. The eigensystem and eigendecompositon of X (see Section 4.2) are used, respectively, to encode the eigenvalues and eigenvectors of X as decision variables of the MP. We emphasize that several problems in optimization and control involve a matrix of decision variables to be subject to a Rank Constraint, and so other applications may benefit of the models presented hereafter.

There are three middling attractive reasons for modeling the rank using eigenvalues and eigenvectors: first, it enables us to model both the RC and the PSD constraint appearing in Eq. (43) with the same set of equations; second, given a solution X' obtained through the DDP models, we can compute its eigenvalues and eigenvectors to warm start the rank constrained models; and third, we obtain as a byproduct of our heuristic the eigenvectors and eigenvalues of the matrix X

satisfying $\operatorname{rk} X \leqslant \min\{n; K\}$, and so we can easily retrieve the matrix x using the procedure described in Section 4.3.1.

4.4.2.1 *Eigensystem*

In order to model the eigensystem of X, we assume that eigenvalues of X are bounded and satisfy $\lambda_i \subseteq [\lambda_i^L, \lambda_i^U]$ for $i \in N$, with $\lambda_i^L, \lambda_i^U \in \mathbb{R}$, and that the nonzero eigenvalues have absolute value greater than a given $\epsilon_\lambda > 0$. Next we define binary decision variables like $z : N \to \{0, 1\}$ that take value 1 if the respective eigenvalue is nonzero and 0 otherwise. Let δ_{ij} be the Kronecker delta for $i, j \in N$ and consider the system of constraints:

$$\forall i, j \in N \quad \sum_{k \in N} X_{ik} v_{kj} = \lambda_j v_{ij}, \tag{53}$$

$$\forall i, j \in N \quad \sum_{k \in N} v_{ki} v_{kj} = \delta_{ij}, \tag{54}$$

$$\forall i \in N \quad \lambda_i^L z_i \leqslant \lambda_i \leqslant \lambda_i^U z_i, \tag{55}$$

$$\forall i \in N \quad |\lambda_i| \geqslant \epsilon_\lambda z_i. \tag{56}$$

Eq. (53) represents the eigensystem of X. Eq. (54) states that the set of eigenvectors $\{v_1, ..., v_n\}$ is an orthonormal basis of $\mathbb{R}^n$ associated to X. Eq. (55) and Eq. (56) together require $\lambda_i = 0 \Leftrightarrow z_i = 0$ for $i \in N$. We call MES the model represented by the system (53)-(56) and point out that it can be solved over the reals in case X is required to be symmetric as well (which is our case). An alternative to constraint (56) is the constraint $\lambda_i^2 \geqslant \epsilon_\lambda^2 z_i$ for all $i \in N$.

4.4.2.2 *Eigendecomposition*

The next model follows directly from substituting Eq. (53) by constraints that represent the eigendecomposition of X:

$$\forall i, j \in N \quad X_{ij} = \sum_{k \in N} v_{ik} v_{jk} \lambda_k. \tag{57}$$

We call MED the model given by the system (54)-(57). But Eq. (57) implicitly guarantees that X is symmetric since

$$X_{ij} = \sum_{k \in N} v_{ik} v_{jk} \lambda_k = \sum_{k \in N} v_{jk} v_{ik} \lambda_k = X_{ji}. \tag{58}$$

Hence it is possible remove the symmetry constraints on X when using MED. Should we maintain the symmetry constraints, the total number of constraints in Eq. (57) can be reduced as follows

$$\forall i \leqslant j \in N \quad X_{ij} = \sum_{k \in N} v_{ik} v_{jk} \lambda_k, \tag{59}$$

yielding a third model given by the system (54)-(56),(59), named MED$_{RS}$.

4.4.2.3 *Rank constraints*

Finally, Rank Constraints can be constructed by means of the z variables like, e.g.:

$$\sum_{i \in N} z_i \leqslant r \tag{60}$$

with $r \in \mathbb{N}$. Other types of constraints (e.g. the rank is bounded below by r) and objectives (e.g. the rank must be maximum) can be constructed similarly. In the EDGP, $r = \min\{n; K\}$ in Eq. (60).

4.4.3 *Bounds on the variables x, X and λ*

The next result is simple but important to establish bounds for the variables x, X and λ.

First, we introduce some notation and an useful definition which will serve to shorten our presentation. Given a weighted graph $G = (V, E, d)$, let $P_{[s,t]}$ denote a simple path of G with endpoints $s, t \in V$ and $\mathsf{len}(P_{[s,t]})$ its length. Furthermore, let P_G denote the longest simple path of G and $P^{-}_{[s,t]}$ the shortest path between s, t.

Definition 43. *The* center point c *of a path* $P_{[s,t]}$ *is the point located along* $P_{[s,t]}$ *satisfying* $\mathsf{len}(P_{[s,c]}) = \mathsf{len}(P_{[c,t]}) = \mathsf{len}(P_{[s,t]})/2$.

Note that the center point of $P_{[s,t]}$ may or may not coincide with one of its original vertices.

Proposition 44. *Let* $G = (V, E, d)$ *be an YES instance of the* EDGP *having no prelocated vertices and a* P_G *with* $\mathsf{len}(P_G) = L$. *There exists* x *bounded by* $B(0, L/2)$.

Proof. We prove it by contradiction. Let w_s, w_t be the endpoints and w_c the center point of P_G. Take a realization x of G. Because G has no anchors, x can be translated at will to satisfy $\|x(w_c)\|_2 = 0$ (center point of P_G at the origin). Now assume that there is a different vertex $w_o \in V$ such that $\|x(w_o)\|_2 > L/2$. In this case, since G is connected, there must exist a path $P_{[w_c,w_o]}$ satisfying $\mathsf{len}(P_{[w_c,w_o]}) > L/2$. But then $\mathsf{len}(P_{[w_s,w_o]}) = \mathsf{len}(P_{[w_s,w_c]}) + \mathsf{len}(P_{[w_c,w_o]}) > \mathsf{len}(P_G)$, which contradicts our assumption. □

The proof of Prop. 44 provides straight foward bounds to the norm of the x vectors. For the realization x satisfying $c = 0$, it is easy to see that no vector x_i $(i \in N)$ can be located further than the length of its shortest path to the origin. So

$$\|x_i\|_2 \leqslant \mathsf{len}(P^{-}_{[i,0]}). \tag{61}$$

Now for the X variables, recall that $X_{ij} = \langle x_i, x_j \rangle$ for $i, j \in N$. By the Cauchy-Schwarz inequality, $\|X_{ij}\|_2 = \|\langle x_i, x_j \rangle\|_2 \leqslant \|x_i\|_2 \|x_j\|_2$;

and in turn, if we denote $\gamma_{ij} = \|x_i\|_2\|x_j\|_2$, we have $-\gamma_{ij} \leqslant X_{ij} \leqslant \gamma_{ij}$. But these bounds can be improved. First, from the inner product definition, we know that $X_{ii} \geqslant 0$. Second, for $\{i,j\} \in E$, any global optima of the EDGP must satisfy $X_{ii} - 2X_{ij} + X_{jj} - (d_{ij})^2 = 0$; since the elements of the diagonal of X are nonnegative, $X_{ii} + X_{jj} \geqslant 0$ and

$$X_{ij} \geqslant -\frac{d_{ij}^2}{2}$$

must hold. We can then define the parameter

$$\eta_{ij} = \begin{cases} 0 & \text{if } i = j, \\ \max(-\frac{d_{ij}^2}{2}, -\gamma_{ij}) & \text{if } \{i,j\} \in E, \\ -\gamma_{ij} & \text{otherwise,} \end{cases}$$

and set forth that $\eta_{ij} \leqslant X_{ij} \leqslant \gamma_{ij}$ for $i,j \in N$. As a consequence, the matrix X of decision variables can be interpreted as a real symmetric interval matrix like

$$X = \begin{pmatrix} [\eta_{11}, \gamma_{11}] & [\eta_{12}, \gamma_{12}] & \dots & [\eta_{1n}, \gamma_{1n}] \\ [\eta_{21}, \gamma_{12}] & [\eta_{22}, \gamma_{22}] & \dots & [\eta_{2n}, \gamma_{2n}] \\ \vdots & \vdots & \ddots & \vdots \\ [\eta_{n1}, \gamma_{n1}] & [\eta_{n2}, \gamma_{n2}] & \dots & [\eta_{nn}, \gamma_{nn}] \end{pmatrix}.$$

And this condition allow us to compute bounds for the λ variables according to Eq. (41).

4.4.4 *Additional constraints*

4.4.4.1 *Trace constraints*

This is possibly the most popular equation relating the eigenvalues and the diagonal elements of a matrix X:

$$\sum_{i \in N} X_{ii} = \sum_{i \in N} \lambda_i. \tag{62}$$

However here we have floating point numbers appearing in an equality expression, which may not be interesting computationally for the reasons mentioned in Sect. 1.3. So alternatively to adjoining the Trace Constraint (TC) as given by Eq. (62) directly to the MPs, one might derive bounds for the sum of the λ values since the following inequalities trivially hold:

$$\sum_{i \in N} \eta_{ii} \leqslant \sum_{i \in N} \lambda_i \leqslant \sum_{i \in N} \gamma_{ii}. \tag{63}$$

In fact, we do observe experimentally that these sums over η and γ are tighter than the sums over the bounds computed using Eq. (41). And also that these inequalities are beneficial.

4.4.4.2 *Symmetry-Breaking Constraints*

Provided that there is no relation between the eigenvalues of a matrix $X \in \mathbb{S}^n$, it is usual to assume them sorted in a nondecreasing order like

$$\lambda_1(X) \leqslant \lambda_2(X) \leqslant \cdots \leqslant \lambda_n(X).$$

These are strong SBCs which might remove symmetric global optima and quicken the performance of BB type algorithms (see Section 2.2.6). Of course, we intend to test them.

4.4.5 *Modelling simplifications*

In order to write Eq (43) as a MP, we must put together the LP relaxation Eq.(46), one of the RC models (MES, MED or MED_{RS}), the RC given by Eq. (60), and set the lower bounds of the λ variables to zero since $X \succeq 0$. Some variations include adjoining the TCs or the SBCs or both. Notwithstanding, all the resulting formulations are MINLPs. Needless to say, some preliminary tests indicated that it would be impossible to scale up with these programs in practical terms [24].

We therefore proceed to simplify our MINLPs into NLPs based on the following observation: with probability 1, any matrix $x \in \mathbb{R}^{n\times K}$ would have full rank if sampled from an uniform space, particularly for $n \gg K$. This latter condition $(n \gg K)$ is common in real-life applications of the EDGP, since one is usually interested in localizing many n points in the low $(1, 2, 3)$-dimensional Euclidean spaces. Whenever this condition is known to hold *a priori*, one can (safely) narrow the search space by assuming that:

$$\operatorname{rk} X = K.$$

Now recall that the eigenvalues of X keep no relation between them; since we have just assumed that the rank of X equals K, we can actually fix the $n - K$ first z-variables to 0 and the remaining K to 1, so as to satisfy the SBCs described in the previous section. This allow us to eliminate the binary z-variables and the Rank Constraint from our formulations.

4.4.6 *Tackling large instances*

A successful computational method has to be fast and accurate, regardless of its application domain. The most successful algorithms in the literature for the EDGP are the DGSol [67] (an homotopy method) and the Branch-and-Prune [45, 53] (a pure feasibility combinatorial-type method). Both have many positive and few negative traits.

Recall that, for the EDGP, a 100% accurate solution has optimum value zero either in Eq. (37) or in Eq. (44). So in terms of MP methods, being fast and accurate means converging fast to global optima. It is common-sense however that exact MP solvers are accurate, but that they do not necessarily scale well to large sizes, at least timewise. This means that we are setting the bar too high in attempting to solve the EDGP exactly. It is thus desirable to endow our heuristic with some speed capability, specially to tackle large instances. And the idea is to turn it into an approximative method, in a controlled way though, to avoid compromising the accuracy of the solutions provided by the MP formulations. Or in other words, to accept convergence to certain local optima.

In this regard, recall that solvers usually provide (via API or CLI) setup parameters which permit users to fine-tune the stopping (or convergence) criteria. Most likely, there is a parameter (say $absgap$) representing the absolute gap tolerance, meaning that the solver stops whenever the absolute gap between the best feasible solution and the best lower bound (upper bound in case of maximization problems) drops below $absgap$ (or when the time limit is reached, whatever comes first).

The point is that if the value of $absgap$ is fixed and equal for all instances, we end up searching for realizations with the same level of accuracy, oblivious to the fact that number of edges differs from instance to instance (we are obviously concerned with increasing trends). A simple alternative to adjust the convergence criteria according to the size of the instance being solved is to parameterize the absolute gap tolerance in terms of $|E|$; and we can do this defining a parameter that fixes the "maximum allowed error per edge", say epe, and then setting $absgap = epe * |E|$. We can eventually experiment and calibrate epe so as to converge to local optima with acceptable accuracy values.

4.5 COMPUTATIONAL EXPERIMENTS

In this section we describe the computational environment involved (instances, machinery, solvers, setups, procedures, etc) and analyze the results obtained in our experiments.

4.5.1 *Dataset*

Our test set consists of 40 feasible instances of the EDGP, divided into two groups. The first group consists of 9 instances for the EDGPs in $K = 2$, named euclid-n_p, which were generated randomly as follows:

1. place n points in a square, uniformly at random;

Instance	$\|V\|$	$\|E\|$	Instance	$\|V\|$	$\|E\|$
$K = 2$					
euclid-15_0.5	15	60	euclid-50_0.3	50	412
euclid-20_0.5	20	111	euclid-50_0.4	50	535
euclid-30_0.5	30	240	euclid-50_0.5	50	642
euclid-40_0.5	40	429	euclid-50_0.6	50	772
euclid-50_0.2	50	290			
$K = 3$					
C0700odd.2	8	28	lavor30_6-3	30	195
lavor11	11	40	lavor30_6-4	30	191
lavor11_7	11	47	lavor30_6-5	30	195
lavor11_7-2	11	47	lavor30_6-6	30	195
C0150alter.1	26	191	lavor30_6-7	30	195
tiny	27	252	lavor30_6-8	30	193
lavor30_6-1	30	192	C0700.odd.G	36	308
lavor30_6-2	30	202	2erl-frag-bp1	39	406
C0080create.1	60	681	res_3000	108	1487
C0080create.2	60	681	res_5000	108	1392
names	82	840	1guu	150	955
C0020pdb	107	999	1guu-1	150	959
pept	107	999	1guu-4000	150	968
res_0	108	1410	2kxa	177	2711
res_1000	108	1506	res_2kxa	177	2627
res_2000	108	1404			

Table 16: Description of the EDGP instances.

2. generate the cycle $1, \dots, n$ to ensure biconnectedness;

3. for each other vertex pair i, j, decide whether $\{i, j\} \in E$ with probability p;

4. record the Euclidean distance d_{ij} between pairs of points in E;

obviously, all such instances are feasible.

The second group consists of 31 instances for the EDGPs in $K = 3$, which are protein instances taken from the Protein Data Bank (PDB) [9]. For these instances, only edges smaller than 5Å were kept, which is realistic with respect to NMR experiments.

Overall, we consider instances having $|V| \leqslant 40$ as small instances. Table 16 gives details (name and dimensions) of our test set.

4.5.2 *Environment*

The LPs were solved with CPLEX 12.6 [37] and the NLPs with BARON 14.4.0 [78, 86] under the GAMS environment [63, 75] on a 24-CPU Intel Xeon at 2.53GHz with 48Gb RAM running GNU/Linux. Both solvers ran with default configurations. Execution time was limited to 3 hours of wall clock time. We set GAMS absolute gap parameter *optca* according to the discussion in Section 4.4.6, and preliminary trial and error type experiments indicated that $epe = 10$ leads to a good trade-off between accuracy and time consumption. We set $\epsilon_\lambda = 0.001$ and use Eigen [34], a C++ template library for Linear Algebra, to compute the bounds on the λ variables using Eq. (41). We also added the constraints Eq. (63) to all formulations.

4.5.3 *Results*

The most used quality indicators of a realization x of the EDGP are the (scaled) largest distance error, defined as

$$\mathsf{lde}(x) = \max_{\{i,j\} \in E} (|\,\|x_i - x_j\|_2 - d_{ij}\,| / d_{ij}),$$

and the (scaled) mean distance error, defined as:

$$\mathsf{mde}(x) = \frac{1}{|E|} \sum_{\{i,j\} \in E} (|\,\|x_i - x_j\|_2 - d_{ij}\,| / d_{ij}).$$

These are the metrics used to analyze our results, with lower values meaning better solutions. But note that they are defined in terms of the original vectors $x \in \mathbb{R}^{n \times K}$. Therefore, first and foremost, we need to apply PCA on every SDP solution X' by factoring X' into $x' = \vartheta\sqrt{\Lambda_K^+}$, and then compute and record the mde and lde values for x'.

4.5.3.1 *DDP results*

We start off presenting the results regarding the DDP programs. Table 17 reports per instance, for the noniterative $\mathbb{DDP}$s, the mde, lde and rank of the solutions, and the total wall clock time time (in seconds); for the iterative $\mathbb{DDP}$s, it reports the mde, lde and rank of the solutions (we also display the maximum possible value of the rank like in Chapter 3), and the total wall clock time (in seconds), number of DDP iterations and the termination status of the algorithm (not = solution is feasible but not PD, lim = time limit reached, inf = infeasible).

Overall, our experiments show that the DDP methods are fast and scale well (at least the noniterative method), as expected, given that we solved $\mathbb{DDP}$s with matrices of size up to 177×177, apparently, without major difficulties. The exception being the fruitless attempts to solve instances res_0, res_1000, res_2000, res_3000, 2kxa, res_2kxa with the iterative method, which reached the time limit of 3 hours whilst solving the second program of the DDP sequence.

Differently to what happened with the $\mathbb{BQP}$s (see Section 3.6.3), we judge that the iterative method finally displayed its importance and usefulness when applied to solve the EDGP: it improved the quality of the solutions whenever at least two programs of the DDP sequence

	DDP				Iterative DDP					
Instance	mde	lde	Rank	Time (s)	mde	lde	Rank	Time (s)	Iter	St.
$K = 2$										
euclid-15_0.5	0.531	2.125	14/15	0.095	0.531	2.125	14/15	0.095	1	not
euclid-20_0.5	0.817	8.226	19/20	0.126	0.817	8.226	19/20	0.126	1	not
euclid-30_0.5	0.688	3.806	29/30	0.274	0.688	3.806	29/30	0.274	1	not
euclid-40_0.5	0.863	10.822	39/40	0.584	0.863	10.822	39/40	0.584	1	not
euclid-50_0.2	0.873	8.669	46/50	0.567	0.873	8.669	46/50	0.567	1	not
euclid-50_0.3	0.816	8.562	47/50	0.733	0.816	8.562	47/50	0.733	1	not
euclid-50_0.4	0.939	9.344	47/50	0.709	0.939	9.344	47/50	0.709	1	not
euclid-50_0.5	1.265	181.876	48/50	0.924	1.265	181.876	48/50	0.924	1	not
euclid-50_0.6	1.051	30.699	48/50	0.676	1.051	30.699	48/50	0.676	1	not
$K = 3$										
C0700odd.2	0.256	0.986	7/8	0.079	0.256	0.986	7/8	0.079	1	not
lavor11	0.379	1.496	11/11	0.085	**0.222**	0.888	8/11	0.388	3	not
lavor11_7	0.457	1.722	11/11	0.085	**0.267**	1.317	10/11	0.347	3	not
lavor11_7-2	0.448	1.487	11/11	0.085	**0.331**	1.477	10/11	0.195	2	not
C0150alter.1	0.587	2.338	25/26	0.192	0.587	2.338	25/26	0.192	1	not
tiny	0.608	2.476	27/27	0.227	**0.495**	2.994	25/27	2.021	2	not
lavor30_6-1	0.694	2.029	30/30	0.183	**0.580**	1.471	27/30	2.165	2	not
lavor30_6-2	0.651	3.078	30/30	0.154	**0.487**	2.430	29/30	1.741	2	not
lavor30_6-3	0.665	2.151	30/30	0.190	**0.534**	1.833	29/30	1.754	2	not
lavor30_6-4	0.642	1.539	30/30	0.209	**0.477**	0.950	28/30	2.761	2	not
lavor30_6-5	0.605	4.447	30/30	0.178	**0.409**	0.971	27/30	14.50	3	not
lavor30_6-6	0.610	2.029	29/30	0.222	0.610	2.029	29/30	0.222	1	not
lavor30_6-7	0.717	5.194	30/30	0.222	**0.527**	4.112	28/30	2.943	2	not
lavor30_6-8	0.637	5.516	30/30	0.167	**0.428**	1.175	25/30	2.049	2	not
C0700.odd.G	0.697	1.502	35/36	0.414	0.697	1.502	35/36	0.414	1	not
2erl-frag-bp1	0.682	3.268	39/39	0.557	**0.433**	1.513	34/39	94.36	3	not
C0080create.1	0.741	1.723	60/60	0.879	**0.605**	1.034	58/60	154.41	2	not
C0080create.2	0.741	1.723	60/60	0.889	**0.605**	1.034	58/60	154.49	2	not
names	0.859	1.325	82/82	2.141	**0.730**	0.993	79/82	2385.58	2	not
C0020pdb	0.871	1.197	106/107	3.879	0.871	1.197	106/107	3.879	1	not
pept	0.849	1.067	106/107	3.729	0.849	1.067	106/107	3.729	1	not
res_0	0.795	1.875	108/108	6.103	0.795	1.875	108/108	10868.27	2	lim
res_1000	0.792	1.110	108/108	6.182	0.792	1.110	108/108	10829.29	2	lim
res_2000	0.805	1.765	108/108	4.914	0.805	1.765	108/108	10929.82	2	lim
res_3000	0.823	1.551	108/108	5.947	0.823	1.551	108/108	10842.22	2	lim
res_5000	0.808	2.180	107/108	5.102	0.808	2.180	107/108	5.102	1	not
1guu	0.870	1.120	149/150	3.330	0.870	1.120	149/150	3.330	1	not
1guu-1	0.913	2.270	148/150	3.277	0.913	2.270	148/150	3.277	1	not
1guu-4000	0.900	1.558	149/150	3.114	0.900	1.558	149/150	3.114	1	not
2kxa	0.913	4.486	177/177	17.359	0.913	4.486	177/177	11392.98	2	lim
res_2kxa	0.925	4.696	177/177	15.283	0.925	4.696	177/177	11661.45	2	lim

Table 17: Results obtained with CPLEX 12.6 from all EDGP instances and both DDP methods.

were solved successfully; this is clearly the case among the sample of small $K = 3$ instances. Moreover, we point out that the method does take few iterations (3 at most in our tests) and converges very quickly to a face of the PSD cone, regardless of the instance size (see rows with status "not"). Unfortunately, but again as foreseen, the rank of the solutions found are rather greater than the dimensions of the embedding spaces (2 or 3, accordingly), meaning that we definitely need to solve a rank reduction problem (i.e. employ a rank reduction procedure) to find realizations with the correct rank, at least as it concerns the EDGP. The results pertaining to this task are exposed in the next section.

4.5.3.2 *NLP results: RC models*

We tested the simplified rank constrained NLPs obtained by combining Eq. (46) with either MES, MED or MED$_{RS}$, and their respective narrowings, obtained by adjoining the SBCs on the λ variables (see Section 4.4.4.2), resulting in a total of 6 models per instance. Tables 18, 19 and 20 report the results. Per instance and for each formulation, the tables exhibit the mde and lde of the solutions found, the elapsed wall clock time (in seconds) and the solver status (opt = optimal, feas = feasible, nsf = no solution found). Per instance, we emphasize in boldface the best realizations found according to the ranking criteria: solver status, mde and computational time.

Unfortunately, but not surprisingly, although the DDP methods provided starting points for instances with up to 177 points, we could not scale up too much with our MP heuristic, and no approximate solutions were found for instances having more than 60 points: our method could pull out solutions for just 27 out of the 40 tested instances, basically the small ones. Yet our rank reduction step managed to improve the quality of the initial DDP solutions in all 27 cases, which is a positive achievement.

We move on to draw preliminary conclusions on which family of RC models performs the better. We observe a slight advantage to the MED based formulations: optimal solutions were found for instances with up to 30 points and they produced 15 best performances. In second place comes the MED$_{RS}$ models, which were also capable of providing optimal solutions for instances of size at most 30, but produced only 10 best performances. This is not quite as expected, since the MED$_{RS}$ models have less nonlinear terms overall per instance, but possible, considering that the trilinear terms in Eq. (57) are identical for X_{ij} and X_{ji} (as seen in Eq. (58)): these constraints are kind of redundant and so, in theory, they induce no overhead in terms of convexification techniques or feasibility matters. On the other hand, the MES based formulations performed the worse: they could solve instances with at most 11 points and produced only one best performance (lavor11).

		Original RC formulation				Narrowing			
Instance	Model	mde	lde	Time (s)	St.	mde	lde	Time (s)	St.
euclid-15_0.5	mes	-	-	10808.13	nsf	-	-	10808.17	nsf
	med	0.065	0.680	10808.10	fea	0.065	0.680	10808.12	fea
	medrs	0.065	0.680	10808.14	fea	0.065	0.680	10808.11	fea
euclid-20_0.5	mes	-	-	10808.15	nsf	-	-	10808.12	nsf
	med	**0.124**	3.772	10808.11	fea	0.124	3.772	10808.12	fea
	medrs	0.124	3.772	10808.15	fea	0.150	4.828	10808.16	fea
euclid-30_0.5	mes	-	-	10808.28	nsf	-	-	10808.14	nsf
	med	0.049	1.505	103.84	fea	**0.023**	0.713	175.29	fea
	medrs	0.039	0.918	52.66	fea	0.039	0.865	95.94	fea
euclid-40_0.5	mes	-	-	10808.22	nsf	-	-	10808.18	nsf
	med	0.011	1.027	**981.53**	fea	0.011	1.027	1022.62	fea
	medrs	0.011	1.027	1680.93	fea	0.011	1.027	1774.48	fea
euclid-50_0.2	mes	-	-	10809.28	nsf	-	-	10809.25	nsf
	med	0.017	0.715	7620.37	fea	0.253	6.531	10808.31	fea
	medrs	0.017	0.715	**7295.73**	fea	0.018	1.202	5580.57	fea
euclid-50_0.3	mes	-	-	10814.81	nsf	-	-	10877.01	nsf
	med	0.011	1.081	**4763.35**	fea	0.012	1.265	8574.81	fea
	medrs	0.012	1.322	9848.76	fea	0.026	2.553	2148.76	fea
euclid-50_0.4	mes	-	-	10809.44	nsf	-	-	10806.06	nsf
	med	0.029	1.706	4574.68	fea	0.029	1.706	3650.53	fea
	medrs	0.029	1.706	**2217.35**	fea	0.033	2.887	3720.59	fea
euclid-50_0.5	mes	-	-	10800.66	nsf	-	-	10808.32	nsf
	med	0.126	43.278	3982.96	fea	0.126	43.278	**1694.92**	fea
	medrs	-	-	10808.57	nsf	0.147	43.278	10808.31	fea
euclid-50_0.6	mes	-	-	10808.60	nsf	-	-	10841.86	nsf
	med	0.040	3.981	7813.21	**fea**	-	-	10808.20	nsf
	medrs	-	-	10808.40	nsf	-	-	10808.31	nsf

Table 18: Results for $K = 2$ EDGP instances obtained from all RC formulations with BARON 14.4.0. 0* indicates values of $O(10^{-5})$ or less.

		Original RC formulation				Narrowing			
Instance	Model	mde	lde	Time (s)	St.	mde	lde	Time (s)	St.
C0700odd.2	mes	0^*	0^*	3.55	opt	0^*	0^*	4.15	opt
	med	0^*	0^*	0.88	opt	0^*	0^*	**0.08**	opt
	medrs	0^*	0^*	1.71	opt	0^*	0^*	0.93	opt
lavor11	mes	0.022	0.265	33.78	fea	0^*	0^*	28.05	**opt**
	med	0.003	0.082	0.47	fea	0.103	0.768	0.24	fea
	medrs	0.003	0.082	0.58	fea	0.103	0.768	0.50	fea
lavor11_7	mes	0^*	0^*	32.37	opt	0^*	0^*	51.85	opt
	med	0^*	0^*	0.25	opt	0^*	0^*	0.26	opt
	medrs	0^*	0^*	0.21	opt	0^*	0^*	**0.19**	opt
lavor11_7-2	mes	0^*	0^*	21.98	opt	0.007	0.049	31.17	fea
	med	0^*	0^*	0.22	opt	0^*	0^*	0.19	opt
	medrs	0^*	0^*	0.28	opt	0^*	0^*	**0.18**	opt
tiny	mes	-	-	10808.16	nsf	-	-	10808.59	nsf
	med	0.038	0.601	23.91	fea	0.033	0.893	79.65	fea
	medrs	**0.024**	0.544	28.25	fea	0.033	0.893	113.34	fea
C0150alter.1	mes	-	-	10808.31	nsf	-	-	10808.19	nsf
	med	0.016	0.401	223.25	fea	**0.003**	0.050	77.19	fea
	medrs	0.016	0.401	61.70	fea	0.047	0.860	36.91	fea
lavor30_6-1	mes	-	-	10808.44	nsf	-	-	10808.18	nsf
	med	0.019	0.481	1748.64	fea	0.001	0.040	1020.75	fea
	medrs	0^*	0^*	756.32	**opt**	0.020	0.498	403.95	fea
lavor30_6-2	mes	-	-	10808.54	nsf	-	-	10808.17	nsf
	med	0.008	0.472	342.76	fea	**0.006**	0.484	157.61	fea
	medrs	0.011	0.490	284.97	fea	0.011	0.490	147.05	fea
lavor30_6-3	mes	-	-	10808.20	nsf	-	-	10808.47	nsf
	med	0.009	0.421	63.34	fea	0.013	0.484	601.13	fea
	medrs	0.010	0.456	43.66	fea	**0.001**	0.068	265.19	fea
lavor30_6-4	mes	-	-	10808.18	nsf	-	-	10808.19	nsf
	med	0.017	0.469	137.28	fea	**0.006**	0.469	99.62	fea
	medrs	0.017	0.484	151.72	fea	0.031	0.430	33.00	fea
lavor30_6-5	mes	-	-	10808.18	nsf	-	-	10808.21	nsf
	med	0^*	0^*	72.31	opt	0^*	0^*	170.73	opt
	medrs	0^*	0^*	**55.79**	opt	0.020	0.491	281.05	fea
lavor30_6-6	mes	-	-	10808.39	nsf	-	-	10808.16	nsf
	med	0^*	0^*	224.43	opt	0^*	0^*	433.66	opt
	medrs	0.014	0.457	130.49	fea	0^*	0^*	**178.94**	opt
lavor30_6-7	mes	-	-	10808.58	nsf	-	-	10808.38	nsf
	med	**0.004**	0.203	981.34	fea	0.008	0.426	64.14	fea
	medrs	0.007	0.455	170.63	fea	0.008	0.502	109.46	fea
lavor30_6-8	mes	-	-	10808.20	nsf	-	-	10808.56	nsf
	med	0.022	0.587	35.20	fea	0^*	0^*	72.20	**opt**
	medrs	0.007	0.459	156.00	fea	0.008	0.516	113.49	fea
C0700.odd.G	mes	-	-	10808.68	nsf	-	-	10808.19	nsf
	med	0.004	0.239	563.83	fea	0.006	0.278	331.08	fea
	medrs	0.001	0.046	530.14	fea	**0^***	0.007	828.07	fea
2erl-frag-bp1	mes	-	-	10808.20	nsf	-	-	10808.22	nsf
	med	0.029	0.622	**148.39**	fea	0.029	0.622	225.98	fea
	medrs	0.029	0.622	153.89	fea	0.029	0.622	257.93	fea

Table 19: Results for small $K = 3$ EDGP instances obtained from all RC formulations with BARON 14.4.0. 0^* indicates values of $O(10^{-5})$ or less.

		Original RC formulation				Narrowing			
Instance	Model	mde	lde	Time (s)	St.	mde	lde	Time (s)	St.
C0080create.1	mes	-	-	10808.53	nsf	-	-	10827.49	nsf
	med	**0.039**	2.191	10808.43	fea	-	-	10808.26	nsf
	medrs	-	-	10808.48	nsf	0.080	1.440	6678.16	fea
C0080create.2	mes	-	-	10812.33	nsf	-	-	10828.08	nsf
	med	**0.038**	2.182	10808.42	fea	0.042	1.970	10808.56	fea
	medrs	-	-	10808.08	nsf	0.080	1.440	4683.39	fea

Table 20: Results for large $K = 3$ EDGP instances obtained from all RC formulations with BARON 14.4.0. 0* indicates values of $O(10^{-5})$ or less.

	Original RC formulation		Narrowing		Total
Model	Euclid	Protein	Euclid	Protein	
mes	-	1	-	-	1
med	4	4	2	5	15
medrs	2	3	-	5	10
Total	14		12		26

Table 21: Aggregated solution statistics for the RC models.

Proportionally, the narrowings were not helpful as regards the reduction of computational times, most likely because we set to zero the value of $N - K$ eigenvalues a priori; that is, we are actually adding solely one strong SBC for the $K = 2$ instances and two strong SBCs for the $K = 3$ instances, which is insignificant in total numbers. Yet the narrowings are responsible for 12 out of the 26 best performances; in 14 cases the original formulations performed better, and there was one case in which no influence of the SBCs was detected at all (instance euclid-15_0.5). Table 21 depicts a summary of the performance statistics.

An interesting observation is that the set of euclid instances seems to be harder solve than the protein set. If we compare the elapsed times of the instances having less than 40 vertices, apart from the MED models, almost all protein instances were solved in less than ten minutes, whilst the euclid instances consumed one to two hours of elapsed time, reaching the time limit in many cases (see e.g. euclid-15_0.5 and euclid-20_0.5). One could possibly argue that this does not make sense because the realization matrices $x \in \mathbb{R}^{n \times K}$ are smaller for the $K = 2$ instances of the euclid set. However, recall that $\mathbb{SDP}$s and $\mathbb{DDP}$s are oblivious to the dimension K since $X \in \mathbb{R}^{n \times n}$, so in terms of DDP methods for the EDGP, we can affirm with certainty that the "hardness" of an instance is purely related to the set of distance values, and not to the dimension of the embedding spaces at all.

Overall, the NLP results are not expressive but can be taken as a proof of concept of our methodology, particularly with respect to the RC models, considering that global optima were found for many instances. Timewise the method is not competitive when compared to the main algorithms in the literature (DGSol and BP). A direct comparison is barely necessary: even focusing on the best performances of the MED formulations, the elapsed times are already surpassing 2 hours for the small instances (see euclid-50_0.6), and reaching 3 hours for the large instances (see C0080create.1 and C0080create.2). This is the price one pays for applying exact methods to solve difficult MP problems.

4.5.3.3 *NLP results: GO models*

To scrutinize even further the accomplishments of our heuristic, we decided to test both Eq. (37) and Eq. (44), which we name respectively Squared and Absval, for simplicity, and compare the results. Note that, in this round of experiments, both programs (Squared and Absval) are written in terms of the original $x \in \mathbb{R}^{n \times K}$ variables, so we warm started them with the output x' of the PCA of the DDP solutions. We added constraints Eq. (61) to both formulations and also employed the stopping criterion described in Section 4.4.6.

Table 22 reports per instance and per formulation, the mde and lde of the solutions found, the elapsed wall clock time (in seconds) and the solver status (opt = optimal, feas = feasible, nsf = no solution found). Again, we rank and emphasize in boldface the best solution per instance according to the solver status, the mde value and the computational time.

First of all, we would like to remark that both programs provided approximate solutions for all instances in good elapsed times, particularly the Squared model. This is an indication that perhaps it is possible to increase their accuracy (by reducing the value of epe) and find even better solutions, yet in reasonable computational times. In general, we observe that the Squared model performed better on the set of euclid instances (8 vs 1), while the Absval model performed better on the protein instances (21 vs 9). A perceptible trend among the large protein instances is that the Absval model converged slower to lower mde values.

Besides, contrary to what happened with the RC models, the GO solver clearly took advantage of the value of the dimension K and solved to optimality all intances of the euclid set (the exception being euclid50_0.2), what did not happen with the protein set. It would be very insteresting indeed if we could somehow explore this lower dimensionality when employing Semidefinite Programming.

In comparison with the quality of the RC results, the GO results are superior in all senses: scale, accuracy and computational time. The point is, while the RC formulations have $n \times n$ matrices as decision variables, the GO formulations have $n \times K$ matrices. Given that $n \gg K$ in basically all real-world applications of the EDGP, $n^2 \gg nK$ also holds, and one ends up inevitably with many more variables when

	Squared				Absval			
Instance	mde	lde	Time (s)	St.	mde	lde	Time (s)	St.
				$K = 2$				
euclid-15_0.5	0^*	0^*	0.05	**opt**	0^*	0^*	0.20	fea
euclid-20_0.5	0^*	0^*	0.13	**opt**	0^*	0^*	0.88	fea
euclid-30_0.5	0^*	0^*	0.08	**opt**	0^*	0^*	1.08	fea
euclid-40_0.5	0^*	0^*	3.11	**opt**	0^*	0^*	1.84	fea
euclid-50_0.2	0^*	0.001	4.24	fea	0^*	0.001	**1.04**	fea
euclid-50_0.3	0^*	0^*	4.30	**opt**	0^*	0^*	1.17	fea
euclid-50_0.4	0^*	0^*	4.58	**opt**	0^*	0^*	2.55	fea
euclid-50_0.5	0^*	0^*	6.20	**opt**	0.021	3.935	2.25	fea
euclid-50_0.6	0^*	0^*	4.86	**opt**	0^*	0^*	2.79	fea
				$K = 3$				
C0700odd.2	0.226	0.986	0.01	fea	0^*	0^*	0.02	**opt**
lavor11	**0.004**	0.042	0.06	fea	0.029	0.293	0.10	fea
lavor11_7	0^*	0^*	0.07	opt	0^*	0^*	0.07	opt
lavor11_7-2	0^*	0^*	**0.05**	opt	0^*	0^*	0.15	opt
C0150alter.1	0^*	0^*	1.54	opt	0^*	0^*	**1.22**	opt
tiny	0^*	0^*	1.65	**opt**	0.033	0.893	0.72	fea
lavor30_6-1	0.013	0.347	1.30	fea	0^*	0^*	1.01	**opt**
lavor30_6-2	0.053	0.511	0.42	fea	**0.012**	0.430	0.73	fea
lavor30_6-3	0.045	0.548	2.20	fea	**0.006**	0.477	1.31	fea
lavor30_6-4	0.035	0.539	2.57	fea	**0.025**	0.445	0.67	fea
lavor30_6-5	0.013	0.354	3.47	fea	0^*	0^*	0.63	**opt**
lavor30_6-6	0^*	0^*	1.83	opt	0^*	0^*	**1.61**	opt
lavor30_6-7	0.038	0.555	2.28	fea	**0.011**	0.446	0.81	fea
lavor30_6-8	0.013	0.359	4.19	fea	**0.008**	0.520	0.95	fea
C0700.odd.G	0.015	0.419	5.46	fea	0^*	0^*	1.90	**opt**
2erl-frag-bp1	**0.014**	0.331	2.92	fea	0.029	0.622	2.24	fea
C0080create.1	0^*	0^*	15.38	**opt**	0.127	3.634	33.35	fea
C0080create.2	0^*	0^*	15.27	**opt**	0.127	3.634	33.22	fea
names	0.108	0.903	3.33	fea	**0.036**	0.777	120.92	fea
C0020pdb	0.073	0.782	41.49	fea	**0.051**	0.974	183.22	fea
pept	0.059	0.930	21.28	fea	**0.048**	1.009	154.56	fea
res_0	0.080	0.803	10.11	fea	**0.001**	0.596	276.53	fea
res_1000	0.073	1.513	16.75	fea	**0.020**	1.750	319.17	fea
res_2000	0.085	1.465	5.76	fea	**0.051**	1.983	345.85	fea
res_3000	**0.021**	0.534	46.22	fea	0.026	2.310	246.78	fea
res_5000	**0.080**	0.815	17.85	fea	0.139	3.396	135.32	fea
1guu	0.035	0.885	48.01	fea	**0.027**	1.011	17.41	fea
1guu-1	0.037	0.884	81.24	fea	**0.021**	0.776	18.17	fea
1guu-4000	0.048	0.818	81.14	fea	**0.039**	1.113	15.53	fea
2kxa	**0.046**	0.889	9.63	fea	0.120	3.322	810.46	fea
res_2kxa	0.058	0.821	4.89	fea	**0.048**	2.513	798.82	fea

Table 22: Results for all EDGP instances obtained from the GO formulations with BARON 14.4.0. 0^* indicates values of $O(10^{-5})$ or less.

employing SDP approaches. Adding a bunch of nonlinear constraints on top of this enlarged set of decision variables seems to be a recipe for failure then. Maybe not in theoretical applications where $n \approx K$. But if this is not the case or if one simply does not want to resort to SDP procedures, based on our experience, it seems to be more attractive computationally to attack the GO formulations at once, above all if a powerful Global Optimization solver is available; or develop algorithms that are not heavily based on MP solvers. Lesson learned.

4.6 CONCLUSIONS

In this chapter we have presented a two-steps approximative heuristic algorithm based on Mathematical Programming to try and solve the notoriously difficult Euclidean Distance Geometry Problem. We have provided new Linear Programs for the EDGP based on the recent Diagonally Dominant Programming paradigm, which we used to obtain as good as possible (but yet rank infeasible) starting solutions to the problem. Furthermore, we have also provided three Rank Constraint models based on the classical concepts of eigensystem and eigendecomposition, which we used as rank reduction devices. We have then described an heuristic that consists simply of using the solution of the LPs to warm start the rank constrained formulations. Computational experiments, performed with some randomly generated instances as well as with some realistic protein instances taken from the Protein Data Bank, allowed us to validate our method and draw conclusions regarding its performance and accuracy.

5 CONCLUSIONS

In this thesis we have explored theoretical and practical aspects of Mathematical Programming mainly from the standpoint of symmetries and distances. Some mathematical and computational issues were adressed with the intention of effectively overcoming them (or at least taking a few steps forward in this direction) by providing new ideas which may be further explored and developed in the future.

As regards the Symmetry-Breaking Constraints paradigm, we expect that the larger the amount of orbits exploited and SBCs generated, the tigther the resulting narrowing and the faster the convergence of Branch-and-Bound type algorithms. Previously, in general, adjoining SBCs from two or more orbits chosen arbitrarily could result in all global optima being infeasible. To overcome this issue, we have established the Orbital Independence Theory. In short words, given the set Ω_{G_P} of orbits of the formulation group G_P of a problem P, we have devised a procedure to identify an independent set $\Omega_I \subseteq \Omega_{G_P}$ of orbits and we have shown that we can use the SBCs generated from each of the orbits in Ω_I concurrently to reformulate the original formulation without losing global optima. We have evaluated the impact of our methodology by conducting experiments with symmetric instances taken from the libraries MIPLIB2010 and MINLPLib2, as well as with randomly generated Binary Quadratic Programs. The results related to the public instances were at most reasonable. Nevertheless, the results are also considered as an evidence of the fact that we may have reached the limit of what we can do in terms of Static Symmetry Breaking, since we are exploiting as much as we can, but not getting expressive results for the general case; we therefore consider that it is long past time for trying and exploiting the Orbital Independence ideas dynamically.

Next we have explored the symmetry subject in Mathematical Programming by discussing about Binary Quadratic Programming and the performance of Semidefinite Programming (and Diagonally Dominant Programming) when solving symmetric Binary Quadratic Programs. Our preliminary findings indicate that it may not be necessary to adjoin Symmetry-Breaking Constraints when solving Semidefinite Programs or Diagonally Dominant Programs. Yet we recommend its use first because we observed a few cases of improvement and second because these results originate from what is a preliminary attempt to study these subjects together. That being so, we judge that it remains important to further investigate it; at least from the point of view of other symmetry handling strategies available in the literature.

However, in order to carry out our experiments, we have established a procedure to generate symmetric Binary Quadratic Programs. In particular, these tailored Binary Quadratic Programs have proved to be quite relevant to the Orbital Independence Theory since they embed the conditions under which the use of Symmetry-Breaking Constraints produces very interesting outcomes in terms of reduction of Branch-and-Bound execution times.

Finally, we have adventured to tackle the fundamental Euclidean Distance Geometry Problem by means of Semidefinite Programming as a final act. We have presented a two-steps approximative heuristic algorithm to try and solve this notoriously difficult problem. Employing the recent Diagonally Dominant Programming paradigm, we have derived new Linear Programming relaxations for the Euclidean Distance Geometry Problem, used to obtain starting solutions which are rank infeasible with high probability. We have also devised rank reduction devices, namely Rank Constraint models based on the classical concepts of eigensystem and eigendecomposition. Using these two tools, we have described a heuristic that consists simply of using the solution of the Linear Programs to warm start the rank constrained formulations. Computational experiments, performed with some randomly generated instances as well as with some realistic protein instances taken from the Protein Data Bank; these tests allowed us to validate our method and draw conclusions regarding its performance and accuracy when comparing to the results obtained from solving the original Mathematical Programming formulations.

Taking all the content of this thesis into account, we judge that the results presented in this work are far from being exceptional, but they are quite reasonable. Naturally, we hope that the ideas presented herein may be useful and/or insightful in other contexts, eventually.

BIBLIOGRAPHY

[1] Tobias Achterberg. "SCIP: Solving constraint integer programs." In: *Mathematical Programming Computation* 1.1 (2009), pp. 1–41.

[2] Amir Ahmadi and Georgina Hall. *Sum of Squares Basis Pursuit with Linear and Second Order Cone Programming*. Tech. rep. 1510.01597v2. arXiv, 2016.

[3] Abdo Alfakih, Amirand Khandani, and Henry Wolkowicz. "Solving Euclidean Distance Matrix Completion Problems Via Semidefinite Programming." In: *Computational Optimization and Applications* 12.1 (1999), pp. 13–30.

[4] Farid Alizadeh. "Interior Point Methods in Semidefinite Programming with Applications to Combinatorial Optimization." In: *SIAM Journal on Optimization* 5.1 (1995), pp. 13–51.

[5] Miguel Anjos and Jean Lasserre. *Handbook on semidefinite, conic and polynomial optimization*. First edition. Vol. 166. International Series in Operations Research and Management Science. Springer US, 2012.

[6] MOSEK ApS. *What if the solver stalls?* 2014. URL: http://blog.mosek.com/2014/06/what-if-solver-stall.html.

[7] MOSEK ApS. *The MOSEK command line tool. Version 7.1 (Revision 60)*. 2016. URL: http://docs.mosek.com/7.1/tools/The_optimizers_for_continuous_problems.html.

[8] Charles Audet, Pierre Hansen, Brigitte Jaumard, and Gilles Savard. "Links between linear bilevel and mixed 0-1 programming problems." In: *Journal of Optimization Theory and Applications* 93.2 (1997), 273–300.

[9] Helen Berman, Kim Henrick, and Haruki Nakamura. "Announcing the worldwide Protein Data Bank." In: *Nature Structural Biology* 10.12 (2003), p. 980.

[10] Jean Berstel and Luc Boasson. "Context-free Languages." In: *Handbook of Theoretical Computer Science*. Ed. by Jan van Leeuwen. Vol. B. Cambridge, MA, USA: MIT Press, 1990, pp. 59–102.

[11] Leonard Blumenthal. *Theory and Applications of Distance Geometry*. Oxford: Oxford University Press, 1953.

[12] Walter Bofill and Juan Gomez. "Linear and Nonlinear Semidefinite Programming." In: *Pesquisa Operacional* 34.3 (Dec. 2014), pp. 495–520.

[13] Immanuel Bomze, Marco Budinich, Panos Pardalos, and Marcello Pelillo. "The Maximum Clique Problem." In: *Handbook of Combinatorial Optimization, Supp. A*. Ed. by Ding-Zhu Du and Panos Pardalos. Vol. Supp. A. Dordrecht: Kluwer Academic Publishers, 1998, pp. 1–74.

[14] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. First edition. Cambridge, UK: Cambridge University Press, 2004.

[15] Marcus Brazil, Ronald Graham, Doreen Thomas, and Martin Zachariasen. "On the history of the Euclidean Steiner tree problem." In: *Archive for History of Exact Sciences* 68.3 (2014), pp. 327–354.

[16] Allan Clark. *Elements of Abstract Algebra*. New York: Dover Publications, 1984.

[17] John Conway, Yang Jiao, and Salvatore Torquato. "New family of tilings of three-dimensional Euclidean space by tetrahedra and octahedra." In: *Proceedings of the National Academy of Sciences* 108.27 (2011), pp. 11009–11012.

[18] Alberto Costa, Pierre Hansen, and Leo Liberti. "Formulation symmetries in circle packing." In: *Proceedings of the 1st International Symposium on Combinatorial Optimization*. Ed. by Ridha Mahjoub. Vol. 36. Electronic Notes in Discrete Mathematics. Amsterdam, Netherlands: Elsevier, 2010, pp. 1303–1310.

[19] Alberto Costa, Pierre Hansen, and Leo Liberti. "On the Impact of Symmetry-breaking Constraints on Spatial Branch-and-Bound for Circle Packing in a Square." In: *Discrete Applied Mathematics* 161.1-2 (2013), pp. 96–106.

[20] Claudia D'Ambrosio, Vu Khac Ky, Carlile Lavor, Leo Liberti, and Nelson Maculan. "Computational experience on distance geometry problems 2.0." In: *Proceedings of the XIII Global Optimization Workshop*. Ed. by Leocadio Casado, Eligius Hendrix, and Inmaculada Garcia. Malaga, 2014, pp. 101–104.

[21] Claudia D'Ambrosio, Marcia Fampa, Jon Lee, and Stefan Vigerske. "On a Nonconvex MINLP Formulation of the Euclidean Steiner Tree Problem in n-Space." In: *Proceedings of the 14th International Symposium on Experimental Algorithms*. Ed. by Evripidis Bampis. Springer International Publishing, 2015, pp. 122–133.

[22] Gustavo Dias and Leo Liberti. "Orbital Independence in Symmetric Mathematical Programs." In: *Proceedings of the 9th Annual International Conference on Combinatorial Optimization and Applications*. Ed. by Zaixin Lu, Donghyun Kim, Weili Wu, Wei Li, and Ding-Zhu Du. Vol. 9486. Lecture Notes in Computer Science. Springer International Publishing, 2015, pp. 467–480.

[23] Gustavo Dias and Leo Liberti. “Diagonally dominant programming in distance geometry.” In: *Proceedings of the 4th International Symposium on Combinatorial Optimazation*. Ed. by Raffaele Cerulli, Satoru Fujishige, and Ridha Mahjoub. Vol. 9849. Lecture Notes in Computer Science. Springer-Verlag, 2016, pp. 225–236.

[24] Gustavo Dias, Leo Liberti, and Nelson Maculan. “Modelling Rank Constraints in Mathematical Programming.” In: *Proceedings of the 13th Cologne-Twente Workshop on Graphs and Combinatorial Optimization*. Ed. by Ekrem Duman and ALi Fuat Alkaya. Istanbul, Turkey, 2015, pp. 193–196.

[25] Ivan Dokmanic, Reza Parhizkar, Juri Ranieri, and Martin Vetterli. “Euclidean Distance Matrices: Essential theory, algorithms, and applications.” In: *IEEE Signal Processing Magazine* 32.6 (2015).

[26] Marcia Fampa, Jon Lee, and Nelson Maculan. “An overview of exact algorithms for the Euclidean Steiner tree problem in n-space.” In: *International Transactions in Operational Research* 23.5 (2016), pp. 861–874.

[27] Marcia Fampa, Jon Lee, and Wendel Melo. “A specialized branch-and-bound algorithm for the Euclidean Steiner tree problem in n-space.” In: *Computational Optimization and Applications* 65.1 (2016), pp. 47–71.

[28] Marcia Fampa and Nelson Maculan. “Using a Conic Formulation for Finding Steiner Minimal Trees.” In: *Numerical Algorithms* 35.2 (2004), pp. 315–330.

[29] Matteo Fischetti and Leo Liberti. “Orbital Shrinking.” In: *Proceedings of the 2nd International Symposium on Combinatorial Optimization*. Ed. by Ridha Mahjoub, Vangelis Markakis, Ioannis Milis, and Vangelis Paschos. Vol. 7422. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 48–58.

[30] Robert Fourer, David Gay, and Brian Kernighan. *The AMPL Book*. Second edition. California, USA: Cengage Learning, 2002.

[31] Rosiane de Freitas, Bruno Dias, Nelson Maculan, and Jayme Szwarcfiter. *Distance geometry approach for special graph coloring problems*. Tech. rep. 1606.04978v1. arXiv, 2016.

[32] S. Galli. *Parsing AMPL internal format for linear and non-linear expressions*. B.Sc. dissertation, DEI, Politecnico di Milano, Italy. 2004.

[33] Michael Garey, Ronald Graham, and David Johnson. “The complexity of computing Steiner minimal trees.” In: *SIAM Journal on Applied Mathematics* 32.4 (1977), 835–859.

[34] Gaël Guennebaud, Benoît Jacob, et al. *Eigen - a C++ template library for linear algebra*. http://eigen.tuxfamily.org. v3.2.3. 2010.

[35] Milan Hladík. “Bounds on eigenvalues of real and complex interval matrices.” In: *Applied Mathematics and Computation* 219.10 (2013), 5584–5591.

[36] Frank Hwang, Dana Richards, and Pawel Winter. *The Steiner Tree Problem*. 1st. Vol. 53. Annals of Discrete Mathematics. Elsevier, 1992.

[37] IBM. *ILOG CPLEX 12.6 - User's manual*. 2014.

[38] 11th DIMACS Implementation Challenge in Collaboration with ICERM. *Steiner Tree Problems*. http://dimacs11.zib.de/. 2014.

[39] “IEEE standard Floating-Point Arithmetic.” In: *IEEE Std 754-2008* (2008), pp. 1–58.

[40] Volker Kaibel and Marc Pfetsch. “Packing and partitioning orbitopes.” In: *Mathematical Programming* 114.1 (2008), pp. 1–36.

[41] Donald Knuth. *The Art of Computer Programming, Part II: Seminumerical Algorithms*. 1st. Reading, MA: Addison-Wesley, 1981.

[42] Gary Kochenberger, Jin-Kao Hao, Fred Glover, Mark Lewis, Zhipeng Lü, Haibo Wang, and Yang Wang. “The unconstrained binary quadratic programming problem: a survey.” In: *Journal of Combinatorial Optimization* 28.1 (2014), pp. 58–81.

[43] Carlile Lavor, Leo Liberti, and Nelson Maculan. “Computational Experience with the Molecular Distance Geometry Problem.” In: *Global Optimization: Scientific and Engineering Case Studies*. Ed. by Janos Pintér. Berlin: Springer, 2006, pp. 213–225.

[44] Carlile Lavor, Leo Liberti, Nelson Maculan, and Antonio Mucherino. “Recent advances on the discretizable molecular distance geometry problem.” In: *European Journal of Operational Research* 219.3 (2012), pp. 698–706.

[45] Carlile Lavor, Leo Liberti, Nelson Maculan, and Antonio Mucherino. “The discretizable molecular distance geometry problem.” In: *Computational Optimization and Applications* 52.1 (2012), pp. 115–146.

[46] Duan Li, Xiaoling Sun, Shenshen Gu, Jianjun Gao, and Chunli Liu. “Polynomially Solvable Cases of Binary Quadratic Programs.” In: *Optimization and Optimal Control: Theory and Applications*. Ed. by Altannar Chinchuluun, Panos Pardalos, Rentsen Enkhbat, and Ider Tseveendorj. New York, NY: Springer New York, 2010, pp. 199–225.

[47] Leo Liberti. “Writing Global Optimization Software.” In: *Global Optimization: from Theory to Implementation*. Ed. by Leo Liberti and Nelson Maculan. Vol. 84. Nonconvex Optimization and Its Applications. Berlin: Springer, 2006, pp. 211–262.

[48] Leo Liberti. "Reformulations in Mathematical Programming: Definitions and Systematics." In: *RAIRO-RO* 43.1 (2009), pp. 55–86.

[49] Leo Liberti. "Reformulations in Mathematical Programming: Automatic symmetry detection and exploitation." In: *Mathematical Programming A* 131 (2012), pp. 273–304.

[50] Leo Liberti. "Symmetry in Mathematical Programming." In: *Mixed Integer Nonlinear Programming*. Ed. by Sven Leyffer and John Lee. Vol. 154. The IMA Volumes in Mathematics and its Applications. New York: Springer, 2012, pp. 263–286.

[51] Leo Liberti, Sonia Cafieri, and David Savourey. "Reformulation Optimization Software Engine." In: *Mathematical Software*. Ed. by K. Fukuda, J. van der Hoeven, M. Joswig, and N. Takayama. Vol. 6327. Lecture Notes in Computer Science. New York: Springer, 2010, pp. 303–314.

[52] Leo Liberti, Sonia Cafieri, and Fabien Tarissan. "Reformulations in Mathematical Programming: A Computational Approach." In: *Foundations of Computational Intelligence Volume 3: Global Optimization*. Ed. by Ajith Abraham, Aboul-Ella Hassanien, Patrick Siarry, and Andries Engelbrecht. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 153–234.

[53] Leo Liberti, Carlile Lavor, and Nelson Maculan. "A branch-and-prune algorithm for the molecular distance geometry problem." In: *International Transactions in Operational Research* 15.1 (2008), pp. 1–17.

[54] Leo Liberti, Nelson Maculan, and Yue Zhang. "Optimal configuration of gamma ray machine radiosurgery units: the sphere covering subproblem." In: *Optimization Letters* 3.1 (2009), pp. 109–121.

[55] Leo Liberti and James Ostrowski. "Stabilizer-based symmetry breaking constraints for mathematical programs." In: *Journal of Global Optimization* 60 (2014), pp. 183–194.

[56] Leo Liberti, Carlile Lavor, Nelson Maculan, and Marco Antonio Nascimento. "Reformulation in mathematical programming: an application to quantum chemistry." In: *Discrete Applied Mathematics* 157.6 (2009), 1309–1318.

[57] Leo Liberti, Carlile Lavor, Antonio Mucherino, and Nelson Maculan. "Molecular distance geometry methods: from continuous to discrete." In: *International Transactions in Operational Research* 18 (2010), pp. 33–51.

[58] Leo Liberti, Carlile Lavor, Nelson Maculan, and Antonio Mucherino. "Euclidean distance geometry and applications." In: *SIAM Review* 56.1 (2014), pp. 3–69.

[59] Nelson Maculan, Philippe Michelon, and Adilson Xavier. "The Euclidean Steiner tree problem in Rn: A mathematical programming formulation." In: *Annals of Operations Research* 96.1 (2000), pp. 209–220.

[60] François Margot. "Pruning by isomorphism in branch-and-cut." In: *Mathematical Programming* 94 (2002), pp. 71–90.

[61] François Margot. "Exploiting orbits in symmetric ILP." In: *Mathematical Programming B* 98 (2003), pp. 3–21.

[62] François Margot. "Symmetry in Integer Linear Programming." In: *50 Years of Integer Programming*. Ed. by Michael Jünger, Thomas Liebling, Denis Naddef, George Nemhauser, William Pulleyblank, Gerhard Reinelt, Giovanni Rinaldi, and Laurence Wolsey. Berlin: Springer, 2010, pp. 647–681.

[63] Bruce McCarl. *McCarl GAMS user's guide*. GAMS Development Corporation. Washington, 2016.

[64] Brendan McKay. "Practical Graph Isomorphism." In: *Congressus Numerantium* 30 (1981), pp. 45–87.

[65] Brendan McKay and Adolfo Piperno. "Practical graph isomorphism, II." In: *Journal of Symbolic Computation* 60 (2014), pp. 94–112.

[66] Luca Mencarelli, Youcef Sahraoui, and Leo Liberti. "A multiplicative weights update algorithm for MINLP." In: *EURO Journal on Computational Optimization* (), accepted.

[67] Jorge Moré and Zhijun Wu. "Distance geometry optimization for protein structures." In: *Journal of Global Optimization* 15 (1999), pp. 219–234.

[68] Antonio Mucherino, Carlile Lavor, Leo Liberti, and Nelson Maculan. *Distance Geometry: Theory, Methods and Applications*. New York: Springer, 2013.

[69] Roberto do Nascimento, Ana Flavia Macambira, Lucidio Cabral, and Renan Pinto. "The discrete ellipsoid covering problem: A discrete geometric programming approach." In: *Discrete Applied Mathematics* 164.1 (2014), 276–285.

[70] Arnold Neumaier. "Molecular Modeling of Proteins and Mathematical Prediction of Protein Structure." In: *SIAM Review* 39.3 (1997), pp. 407–460.

[71] James Ostrowski, Jeff Linderoth, Fabrizio Rossi, and Stefano Smriglio. "Orbital branching." In: *Proceedings of the 1st Conference on Integer Programming and Combinatorial Optimization*. Ed. by Matteo Fischetti and David Williamson. Vol. 4513. Lecture Notes in Computer Science. Springer, 2007, pp. 104–118.

[72] James Ostrowski, Jeff Linderoth, Fabrizio Rossi, and Stefano Smriglio. "Constraint orbital branching." In: *Proceedings of the 2nd Conference on Integer Programming and Combinatorial Optimization*. Ed. by Andrea Lodi, Alessandro Panconesi, and Giovanni Rinaldi. Vol. 5035. Lecture Notes in Computer Science. Springer, 2008, pp. 225–239.

[73] Marc Pfetsch and Thomas Rehn. *A Computational Comparison of Symmetry Handling Methods for Mixed Integer Programs*. Tech. rep. 5209. Optimization Online, 2015.

[74] A. Phillips and J. Rosen. "A quadratic assignment formulation of the molecular conformation problem." In: *Journal of Global Optimization* 4.2 (1994), pp. 229–241.

[75] Richard Rosenthal. *GAMS - A user's guide*. GAMS Development Corporation. Washington, 2014.

[76] Joachim Rubinstein, Doreen Thomas, and Nicholas Wormald. "Steiner trees for terminals constrained to curves." In: *SIAM Journal on Discrete Mathematics* 10.1 (1997), 1–17.

[77] Guillaume Sagnol. *Picos - A Python Interface to Conic Optimization Solvers*. http://picos.zib.de/. v1.1.1. 2016.

[78] Nikolaos Sahinidis. *BARON 14.4.0: Global Optimization of Mixed-Integer Nonlinear Programs - User's manual*. 2014.

[79] Domenico Salvagnin. "Orbital Shrinking: A New Tool for Hybrid MIP/CP Methods." In: *CPAIOR 2013 Proceedings*. Ed. by Carla Gomes and Meinolf Sellmann. Vol. 7874. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 204–215.

[80] Domenico Salvagnin and Toby Walsh. "A Hybrid MIP/CP Approach for Multi-activity Shift Scheduling." In: *CP 2012 Proceedings*. Ed. by Michela Milano. Vol. 7514. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 633–646.

[81] James Saxe. "Embeddability of weighted graphs in k-space is strongly **NP**-hard." In: *Proceedings of 17th Allerton Conference in Communications, Control and Computing* (1979), pp. 480–489.

[82] Hanif Sherali. "Personal communication." 2007.

[83] Amit Singer. "Angular synchronization by eigenvectors and semidefinite programming." In: *Applied and Computational Harmonic Analysis* 30.1 (2011), pp. 20–36.

[84] Warren Smith. "How to find Steiner minimal trees in Euclidean d-space." In: *Algorithmica* 7.1 (1992), 137–177.

[85] Anthony Man-Cho So and Yinyu Ye. "Theory of semidefinite programming for sensor network localization." In: *Mathematical Programming B* 109.2-3 (2007), pp. 367–384.

[86] Mohit Tawarmalani and Nikolaos Sahinidis. "A polyhedral branch-and-cut approach to global optimization." In: *Mathematical Programming* 103.2 (2005), p. 2005.

[87] The GAP Group. *GAP - Groups, Algorithms and Programming*. Version 4.7.4, 2014.

[88] Lieven Vandenberghe and Stephen Boyd. "Semidefinite Programming." In: *SIAM Review* 38.1 (1996), pp. 49–95.

[89] David Warme, Pawel Winter, and Martin Zachariasen. "Exact Algorithms for Plane Steiner Tree Problems: A Computational Study." In: *Advances in Steiner Trees*. Ed. by Ding-Zhu Du, James Smith, and Joachim Rubinstein. Boston, MA: Springer US, 2000, pp. 81–116.

[90] Pawel Winter and Martin Zachariasen. "Euclidean Steiner minimum trees: An improved exact algorithm." In: *Networks* 30.3 (1997), pp. 149–166.

[91] Henry Wolkowicz, Romesh Saigal, and Lieven Vandenberghe. *Handbook of semidefinite programming: Theory, Algorithms and Applications*. 1st. Vol. 27. International Series in Operations Research and Management Science. Springer, 2000.

[92] Guoliang Xue and Yinyu Ye. "An efficient algorithm for minimizing a sum of Euclidean norms with applications." In: *SIAM Journal of Optimization* 7.4 (1997), 1017–1036.

[93] Yasutoshi Yajima. "Positive semidefinite relaxations for distance geometry problems." In: *Japan Journal of Industrial and Applied Mathematics* 19.1 (2002), pp. 87–112.

[94] Yechiam Yemini. "The positioning problem - a draft of an intermediate summary." In: *Proceedings of the Conference on Distributed Sensor Networks* (1978). Carnegie-Mellon University, Pittsburgh, 137–145.

ÉCOLE DOCTORALE
Sciences et technologies de l'information et de la communication (STIC)

Titre : Symétries et Distances : deux défis fascinants dans la Programmation Mathématique

Mots clefs : Programmation Mathématique, Groupes de symétrie, Programmation Semidéfinie, Géométrie de Distances

Résumé : Cette thèse est principalement consacrée à l'étude et à la discussion de deux questions importantes qui se posent, entre autres, dans le domaine de la Programmation Mathématique : les symétries et les distances. En arrière-plan, nous examinons la Programmation Semidéfinie et sa pertinence comme l'un des principaux outils employés aujourd'hui pour résoudre les Programmes Mathématiques difficilles. Après le chapitre introductif, nous discutons des symétries au Chapitre 2 et des distances au Chapitre 4. Entre ces deux chapitres, nous présentons un chapitre que nous préférons en fait appeler entr'acte : leur contenu ne mérite pas d'être publié pour le moment (il ne fournit aucune innovation à ce jour), mais il fournit un lien entre les deux Chapitres 2 et 4 apparemment distincts, qui sont ceux qui contiennent les principales contributions de cette thèse. Les conclusions de la thèse sont présentées au Chapitre 5.

Title : Symmetries and distances: two intriguing challenges in Mathematical Programming

Keywords : Mathematical Programming, Symmetry groups, Semidefinite Programming, Distance Geometry

Abstract : This thesis is mostly dedicated to study and discuss two important challenges existing not only in the field of Mathematical Programming: symmetries and distances. In the background we take a look into Semidefinite Programming and its pertinency as one of the major tools employed nowadays to solve hard Mathematical Programs. After the introductory Chapter 1, we discuss about symmetries in Chapter 2 and about distances in Chapter 4. In between them we present a chapter that we actually prefer to call as entr'acte: its content is not necessarily worthy of publication yet (it does not provide any innovation so far), but it does provide a connection between the two seemingly separate Chapters 2 and 4, which are the ones containing the main contributions of this thesis. The concluding remarks of the thesis are presented in Chapter 5.

Université Paris-Saclay
Espace Technologique / Immeuble Discovery
Route de l'Orme aux Merisiers RD 128 / 91190 Saint-Aubin, France