Convex Optimization for Discrete Graphical Models

Valentin Durante

THÈSE

En vue de l'obtention du
DOCTORAT DE L'UNIVERSITÉ DE TOULOUSE

Délivré par l'Université Toulouse 3 - Paul Sabatier

Présentée et soutenue par

Valentin DURANTE

Le 15 décembre 2023

Optimisation convexe pour les Modèles Graphiques discrets

Ecole doctorale : **EDMITT - Ecole Doctorale Mathématiques, Informatique et Télécommunications de Toulouse**

Spécialité : **Mathématiques et Applications**

Unité de recherche :
MIAT - Mathématiques et Informatique Appliquées Toulouse

Thèse dirigée par
Thomas SCHIEX

Jury

Mme Renata Sotirov, Rapporteure
M. Paul Swoboda, Rapporteur
Mme Amélie Lambert, Examinatrice
M. Clément Royer, Examinateur
M. Martin Cooper, Examinateur
M. Edouard Pauwels, Examinateur
M. Thomas SCHIEX, Directeur de thèse
M. George Katsirelos, Co-directeur de thèse

À mon grand-père,
À mes parents,

Acknowledgments

Ce manuscrit est non seulement l'aboutissement de 3 ans de travail mais aussi l'aboutissement de tout mon parcours académique et humain jusqu'à mon arrivée dans l'équipe MIAT à l'INRAE de Toulouse. Le moment est donc venu de remercier toutes les personnes qui ont eu une influence sur cette belle partie de ma vie et qui ont rendu ce travail possible.

Tout d'abord, je tiens à remercier mes deux directeurs de thèse Thomas Schiex et George Katsirelos. Thomas et George ont toujours apporté un regard bienvaillant sur mon travail et m'ont également laissé beaucoup de libertés pour explorer les pistes qui me plaisaient. J'ai beaucoup gagné en maturité scientifique à leur côté et les réunions hebdomadaires du mercredi risque de me manquer. Thomas et George forment un super duo et ils ont même fini par me rendre un peu geek sur les bords!

Thank you to all the jury members: Renata Sotirov, Paul Swoboda, Amélie Lambert, Clément Royer, Martin Cooper and Edouard Pauwels. Thank you for the insightful feedbacks and for the interesting discussions we had about my work.

Thank you to my two thesis committee members: Angelika Wiegele and Jean Bernard Lasserre. Thank you Angelika for giving me the opportunity to come to Klagenfurt.

À partir de maintenant, je vais essayer de suivre un ordre chronologique. Merci à tous mes amis qui me connaissent déjà depuis tout petit: Sam, Bilal, Liam, Tifenn et Quentin avec qui j'ai toujours autant de plaisir à discuter et à faire une descente du parc à chien à l'occasion. Merci aussi à toute l'équipe qui me suit depuis le lycée: Quentin, Hugo, Lucas, Arthur, Matthieu, Thomas, Léo et Thibaut. Très content de voir que tout le monde a trouvé sa voie et j'attends toujours la semaine de vacances en été avec autant d'impatience. J'aimerai également remercier ceux que je côtoie depuis la prépa et l'école d'ingénieur: Martin, Théo, Victor, Mathis, Pierre, Adrien et Isma.

Merci aux personnes qui m'ont inspiré à poursuivre cette voie scientifique. En particulier mes professeurs de mathématiques Jérôme Borel pour l'enthousiasme qu'il m'a apporté à faire des maths et également Michel Carré pour la rigueur. Je veux aussi remercier Serge Gratton qui m'a permis de rentrer en contact avec Thomas.

Le moment est venu de remercier toute l'équipe du laboratoire MIAT. J'ai reçu un acceuil chalereux dès le début de mon stage et je tiens à remercier toutes les personnes du secrétariat et en particulier Fabienne qui font en sorte que tout roule correctement dans le labo. Merci au directeur du labo Sylvain, merci pour tes précieux conseils malgrès le fait que j'ai suivi la voie du doctorat (peut-être par pur esprit de contradiction?). Évidemment je veux remercier tous les docteurs et futurs docteurs avec qui j'ai eu l'honneur de partager mon bureau: Manon, Paul, Marianne, Loïc et Vincent. Merci à Lise qui m'a fait confiance (peut-être par totale inconscience) pour prendre soin du cactus de l'unité. Un grand merci à tous les joueurs de tarot et ceux avec qui nous avons fait de belles sorties au baraka: Prasanna, Julien, encore Julien, Samuel (promis un jour je viendrai faire de l'escalade), Philippe, Benjamin,

Raphael, Paul, Jean, Khaoula (bonne chance pour ta soutenance), Aurélie, Joan, Bessam, et malheureusement, je risque d'en oublier quelques-uns. Pour certains d'entre vous j'espère que l'on se retrouvera rapidement sur les terrains de volley. Thank you to all the PhD students I have met during my staying at Klagenfurt, I had a really good time there. Merci à Joséphine, même si elle déteste perdre au tarot et qu'elle me trouvait un peu lent à vélo, ça m'a fait plaisir de discuter avec toi sur les derniers mois de thèse (j'attends encore ta super idée de start-up). Merci beaucoup à Hanna d'être toujours aussi sympa et de bonne humeur, trop forte en art créatif merci au passage de m'avoir offert une règle pour mon anniversaire (j'espère aussi que l'on croisera bientôt les gants). Et enfin merci à mon Duo de thèse, Pierre, avec qui je suis très content d'avoir pu partager cette expérience de plus de trois ans. Pierre c'est en quelque sorte le maître de jeu de l'unité. J'espère que l'on aura encore l'occasion de se croiser par la suite.

Last but not least, j'aimerais remercier tous les membres de ma famille. Tout d'abord mes parents qui m'ont toujours soutenu et ont eu confiance en moi. Merci d'avoir été là pendant ce long parcours qui ne fait au final que commencer. Merci à mon petit frère Arthur qui n'a pas pu résister à l'envie de me suivre à l'INRAE même si tu fais de la biologie. J'aimerai remercier mes grands-parents et en particulier mon grand-père, Christian, qui a été une grande source d'inspiration pour moi tout au long de ma vie.

Contents

List of Acronyms **ix**

Introduction **1**

1 Background **5**
1.1 Discrete Graphical Models . . . 6
1.1.1 Notation, definitions . . . 6
1.2 Queries and exact optimization . . . 9
1.2.1 Optimization over graphical models . . . 10
1.2.2 Connections with Max Cut . . . 11
1.2.3 Integer linear programming . . . 13
1.2.4 Quadratic optimization . . . 14
1.3 Exact algorithms . . . 15
1.3.1 Dynamic programming . . . 15
1.3.2 Branch and bound . . . 17
1.4 Bounding procedures with linear relaxations . . . 18
1.4.1 Definitions . . . 18
1.4.2 Sherali-Adams hierarchy . . . 20
1.4.3 Local polytope, duality and local consistencies . . . 21
1.5 Bounding procedures with semidefinite relaxations . . . 22
1.5.1 Definitions . . . 22
1.5.2 MAP inference as constrained binary quadratic problem . . . 31
1.5.3 Log-encoding of the discrete variables . . . 34
1.5.4 Moment-SOS hierarchy . . . 37
1.5.5 First and second order methods . . . 38
1.5.6 Low-rank methods . . . 44
1.6 Integer solutions with rounding . . . 46

2 Coordinate vs block coordinate descent methods **49**
2.1 LR-LAS . . . 50
2.1.1 Gangster constraints . . . 53
2.1.2 Dual feasible points . . . 56
2.2 LR-BCD . . . 58
2.2.1 From sparsity to block optimization . . . 58
2.2.2 Optimization on the unit-sphere through trigonometry . . . 61
2.2.3 Computational complexity . . . 69
2.2.4 Strong duality and optimality . . . 69
2.2.5 Descent property . . . 70
2.2.6 Producing an integer primal solution . . . 74
2.3 Experiments . . . 74

2.3.1 Description of solvers . . . 74
2.3.2 Random instances . . . 75
2.3.3 Sparse problems . . . 80
2.3.4 Real world problems . . . 81
2.4 Discussion . . . 84
2.4.1 Update rule for the cost matrix . . . 85
2.4.2 On proving convergence of LR-BCD . . . 87
2.4.3 Strengthening the LR-BCD formulation . . . 89

3 Tighter bounds through constraints 93
3.1 A better representation of the feasible set . . . 94
3.2 Resolution with ADMM . . . 98
3.2.1 Primal ADMM . . . 99
3.2.2 Recovering an integer solution . . . 101
3.2.3 Constraint set projectors . . . 101
3.3 Global constraints . . . 104
3.4 Results . . . 105
3.4.1 CPU-time . . . 106
3.4.2 Bounds . . . 106
3.4.3 Results on protein instances . . . 107
3.4.4 Faster projection onto the semidefinite cone . . . 108
3.5 Discussion . . . 109

4 Solving MaxCut exactly with low rank SDP bounds 111
4.1 Introduction . . . 112
4.2 Related work and our contributions . . . 113
4.3 Background . . . 114
4.3.1 Semidefinite relaxations for MaxCut . . . 114
4.3.2 Other exact solvers based on semidefinite programming . . . 116
4.4 Bounding procedure . . . 118
4.4.1 Combining the mixing method with the proximal bundle method 118
4.4.2 Safe upper bounds . . . 119
4.5 Branch and cut . . . 120
4.5.1 Bounding routine . . . 120
4.5.2 Branching rules . . . 121
4.5.3 Problem reformulation and warm start . . . 122
4.6 Biased hyperplane . . . 124
4.7 Implementation . . . 125
4.8 Experiments . . . 127
4.9 Discussion . . . 134

Conclusion 135

Appendices 139

A MixCut parameters 141

B Exactly-one and diagonal projections 143

C French summary 145

Bibliography 159

List of Acronyms

The following abbreviations are used in this manuscript:

AI	Artificial Intelligence
ADMM	Alternating Direction Method of Multipliers
B&B	Branch And Bound
CFN	Cost Function Network
CP	Constraint Programming
CSP	Constraint Satisfaction Problem
CPD	Computational Protein Design
DNN	Doubly Non-Negative program
DL	Deep Learning
ER	Erdős–Rényi
ILP	Integer Linear Programming
IPM	Interior Point Methods
GM	Graphical Model
LP	Linear Programming
MAP	Maximum A Posteriori
MRF	Markov Random Field
QCQP	Quadratically Constrained Quadratic Program
SAT	Boolean SATisfiability Problem
SDP	Semidefinite Programming
WCSP	Weighted Constraint Satisfaction Problem

Notation

In this section, we recall the notation and conventions that are commonly used in linear programming [Dantzig, 1963] and semidefinite programming [Wolkowicz et al., 2012]. These notations are used throughout the entire manuscript.

1. Sets
   - $[n]$ sequence of integers from 1 to n.
   - $\mathbb{R}^n$ the n dimensional Euclidean space.
   - $\mathbb{R}^{m\times n}$ the space of $(m \times n)$ real matrices.
   - $\mathcal{S}^n$ the space of $(n \times n)$ real symmetric matrices.
   - $\mathcal{S}^n_+ := \{X \in \mathcal{S}_n : X \succeq 0\}$ the convex cone of $(n \times n)$ symmetric positive semidefinite matrices.
2. Vectors
   - x a vector.
   - $x^\top$ the transpose of vector x.
   - x_i the i-th component of vector x.
   - $x^\top y$ the inner product of two vectors.
   - $\|x\|$ the Euclidean norm of vector x.
   - e_i the i-th vector of the canonical basis of $\mathbb{R}^n$.
   - $1_n \in \mathbb{R}^n$ the vector of all ones.
   - 0_n the zero vector in $\mathbb{R}^n$.
3. Matrices
   - X a matrix.
   - $X^\top$ the transpose of matrix X.
   - X_k the k-th row vector of matrix X.
   - I_n the identity matrix in $\mathbb{R}^{n\times n}$.
   - $0^{m\times n}$ the zero matrix in $\mathbb{R}^{m\times n}$.
   - E_{ij} element of the canonical basis of $\mathbb{R}^{m\times n}$.
   - $A = \mathrm{Diag}(x)$ diagonal matrix with entries $A_{ii} = x_i,\ x \in \mathbb{R}^n$.
   - $x = \mathrm{diag}(A)$ the vector of $\mathbb{R}^n$ made of the diagonal entries of A.
   - A_i the i-th matrix in a set of matrices $\{A_i\}_{i\in\mathcal{I}}$.
   - A^{-1} the inverse of the square matrix A.
   - A^+ the Moore-Penrose inverse of $A \in \mathbb{R}^{m\times n}$.

- $A_{\succeq 0}$ the projection of $A \in \mathcal{S}^n$ onto the positive semidefinite cone.

The notations for the matrix rows and the matrix indices overlap, but it should be clear from the context which object we are referring to.

4. Operations
   - Tr(.)
   $$\mathrm{Tr}(X) = \sum_{i=1}^{n} X_{ii},$$
   the trace of a $(n \times n)$ square matrix.
   - $\langle ., . \rangle$
   $$\langle X, Y \rangle = \mathrm{Tr}(X^\top Y) = \sum_{i=1}^{m} \sum_{j=1}^{n} X_{ij} Y_{ij},$$
   the standard inner product on two $(m \times n)$ real matrices.
5. Operators on matrices
   - $\mathcal{A} : \mathcal{S}^n \to \mathbb{R}^m$ a linear operator.
   $$\mathcal{A}_i(X) = \langle A_i, X \rangle, \ i = 1, \ldots, m.$$
   - $\mathcal{A}^* : \mathbb{R}^m \to \mathcal{S}^n$ the adjoint of $\mathcal{A}$.
   $$\mathcal{A}^*(y) = \sum_{i=1}^{m} y_i A_i.$$
6. Functions
   Let $f : U \subset \mathbb{R}^n \longrightarrow \mathbb{R}$.
   - $\nabla f(x)$ gradient of f at $x \in \mathbb{R}^n$.
   - $\nabla^2 f(x)$ Hessian of f at $x \in \mathbb{R}^n$.

We now introduce the notations for discrete graphical models [Koller and Friedman, 2009, Cooper et al., 2020]. Some notations may overlap with previously defined objects, but the context clearly indicates which object we are referring to.

1. Variables
   - x_i a discrete variable. Variables can be assigned values from their finite domain.
   - D_i domain of a variable x_i.
   - d_i size of the domain D_i.
   - $a, b, c, g, r \ldots$ actual values.
2. Sequences

- $X, Y, Z \ldots$ a sequence of variables.
- $v, w \ldots$ a sequence of values.
- D_X the domain of a sequence of variables, the Cartesian product of the domains of all variables in X.
- v_X an element of D_X which defines an assignment for all the variables in X.

3. Operators

   - $v_X[Y]$ the projection of v_X on $Y \subseteq X$: the sequence of values that the variables in Y take in v_X.

Introduction

Practically solving combinatorial problems has been a challenge for many decades. In principle, these problems can be found everywhere. They are at the heart of certain disciplines of AI, such as automated reasoning. The computer learns "how to reason" given a set of rules specified by the programmer. These rules are guided by formal logic. This paradigm has been successfully used to formally prove mathematical conjectures that had remained open for decades [Heule et al., 2016, Brakensiek et al., 2022].

In this thesis, our work focuses on a framework that can be seen as an extension of the Constraint Satisfaction Problem (CSP). For these problems, we have to deal with three kind of objects: variables, discrete domains and constraints. Variables are associated with discrete domains from which they take their values. Constraints describe requirements over a set of one or multiple variables. For example, a constraint may prohibit a particular combination of values. Then our goal is to ask the following question: can we find an assignment of all the variables which fulfills all the constraints? With these ingredients we can model a large range of problems going from radio frequency assignment [Koster, 1999] to the well known Sudoku. The constraints are considered as hard constraints. If at least one of the constraints is violated, then the answer to the question for that particular assignment is false.

Our framework, graphical models (GM), have a couple of properties that make them interesting to study. GMs have the three same ingredients as CSP but the constraints are replaced by "soft" constraints. These constraints are integer or real valued functions which take a finite set of non-negative values. The value returned by a constraint for a particular assignment of the variables corresponds to a violation, or a cost that we have to pay if we choose this assignment. A GM defines a joint function over all the variables which gives the total cost for all the constraints. Then, a natural query on graphical models is to find an assignment that minimizes the joint function. The special feature of this joint function is that it can be described as combination of "small" functions which makes it interesting for optimization purposes.

The combinatorial aspect of these problems makes them hard to solve. Indeed the number of combinations to ask the queries is exponential in the number of variables. One idea to overcome this issue is to dive into the continuous world. Finite domains are now replaced by continuous ones, giving us access to the whole continuous optimization toolbox. This transformation only provides an approximation of the original problem. However, it gives some information that can help us to solve the problem exactly. In this thesis we decided to approximate combinatorial problems with Semidefinite Programming (SDP). SDP is a sub-class of convex optimization where the variables are matrices.

Our main motivation is to explore the efficiency of these methods to approximate the optimum of the joint function defined by a GM. In this regard, we usually have

to make a trade-off between the quality of the approximation and the effort spent to solve it. In general, working with matrices is costly in terms of memory and number of operations. However, we will see that there exists methods to handle them efficiently.

Organisation of the manuscript

This thesis is organized in 4 chapters.

The background, Chapter 1, is an introduction to Graphical Models (GM). We give a short study of the theoretical aspects and queries on graphical models. In particular, we are interested in an optimization query which asks to minimize a certain joint function defined by the GM. This task is NP-Complete. We give an overview of the different theoretical and practical tools that were developed to practically solve it.

Chapter 2 presents two new relaxations for the Maximum A Posteriori (MAP) problem on pairwise discrete graphical models. The original combinatorial problem is relaxed into two semidefinite programs, which we solve by using efficient low-rank methods. The first solver is based on the mixing method, a low-rank solver dedicated to diagonally constrained SDPs. For the second relaxation, we developed a dedicated low-rank solver that performs block coordinate descent steps. To make a coordinate descent step, we have to solve an optimization problem on the unit sphere. We show that we can solve it efficiently by using trigonometry and the Newton algorithm. We compare our two methods with state-of-the-art solvers for the MAP problem. Among other things, we demonstrate the quality of SDP solutions for certain types of problem. There is also evidence that our second method has good scalability properties. Finally, we applied it to two real-world use cases: genome assembly and computational protein design. The work of this chapter was published at the International Conference on Machine Learning (ICML) [Durante et al., 2022].

In Chapter 3, we explore a new set of constraints to tighten the SDP relaxation of the MAP problem. To keep a good trade-off between the effort spent and the quality of the solution, we decided to use a first-order method to approximately solve the relaxation. This method also allows for computing a safe dual bound. We show some results against the previous low-rank solvers.

In Chapter 4, we are interested in solving MaxCut to optimality which is a central combinatorial problem in graph theory. Solving MaxCut is relevant in the context of this thesis since it shares tight links with the MAP problem on discrete GM. We propose a new SDP solver based on one of the low-rank methods we used in the second chapter. We show how this method can be incorporated into a branch and cut algorithm. This work is the result of a collaboration begun while visiting the Klagenfurt Mathematical Institute. It will be submitted to a journal before the end of the year.

Funding

This thesis has been supported by the Artificial and Natural Intelligence Institute (ANITI) of Toulouse, France through ANR/PIA grant ANR-19-PI3A-0004 and by the EUR BioEco (grant ANR-18-EURE-0021). We would like to thank these two organisations. Without them, this work would not have been possible.

Chapter 1

Background

Contents

1.1 Discrete Graphical Models 6
1.1.1 Notation, definitions 6
1.2 Queries and exact optimization 9
1.2.1 Optimization over graphical models 10
1.2.2 Connections with Max Cut 11
1.2.3 Integer linear programming 13
1.2.4 Quadratic optimization 14
1.3 Exact algorithms 15
1.3.1 Dynamic programming 15
1.3.2 Branch and bound 17
1.4 Bounding procedures with linear relaxations 18
1.4.1 Definitions 18
1.4.2 Sherali-Adams hierarchy 20
1.4.3 Local polytope, duality and local consistencies 21
1.5 Bounding procedures with semidefinite relaxations 22
1.5.1 Definitions 22
1.5.2 MAP inference as constrained binary quadratic problem . . . 31
1.5.3 Log-encoding of the discrete variables 34
1.5.4 Moment-SOS hierarchy 37
1.5.5 First and second order methods 38
1.5.6 Low-rank methods 44
1.6 Integer solutions with rounding 46

1.1 Discrete Graphical Models

Graphical models (GM) define a mathematical framework to concisely describe a multivariate function using a certain kind of factorization. Our study is restricted to functions of discrete variables. A large number of problems in Computer Science, Logic, Constraint Satisfaction/Programming, Machine Learning, Statistical Physics and Artificial Intelligence can be modeled using graphical models. A GM is defined by a set of variables and a finite set of small functions which are combined together into a joint multivariate function. The small functions usually involve few variables or can be represented in a compact way using a dedicated language.

For example, graphical models with binary functions and Boolean variables are used in automated reasoning. Small functions are defined by disjunction of variables or their negation. Those functions called clauses are in turn combined with conjunction to define the Conjunctive Normal Form. This framework is useful for describing logical properties such as the logical circuits built into computer hardware. We can consider different queries on this type of graphical model, such as the SAT problem, which seeks to show the existence of an assignment that satisfies the conjunctive Boolean formula. If we instead consider Boolean functions over sets of variables with finite domains, combined with conjunction, we obtain a constraint network (CN). Finding an assignment of the variables that optimizes the joint function is referred to as the Constraint Satisfaction Problem (CSP). Given the size of the input, in the worst case there is no known polynomial-time algorithm to solve these problems. SAT was the first known NP-complete problem [Cook, 2023], meaning that any problem in NP can be reduced to it using polynomial-time reductions.

Small functions can also be described as real-valued tensors. Combined with addition or multiplication, we can model a discrete probability distribution as it is done with Bayesian Networks (BNs). After normalization, Markov Random Fields (MRFs) also define a joint distribution over discrete variables [Koller and Friedman, 2009, Bishop and Nasrabadi, 2006].

To summarize, the general definition of graphical models covers a variety of well-studied frameworks, including constraint networks [Rossi et al., 2006], propositional logic [Biere et al., 2009], generalized additive independence models [Bacchus and Grove, 2013], weighted propositional logic, Markov Random Fields [Kindermann and Snell, 1980, Koller and Friedman, 2009], Bayesian networks [Koller and Friedman, 2009] (MRFs with normalized tensors describing conditional probabilities, organized according to direct acyclic graphs), Possibilistic and Fuzzy Constraint Networks [Dubois et al., 1993].

1.1.1 Notation, definitions

We now give the formal definition of a discrete graphical model [Cooper et al., 2020].

Definition 1. *Graphical model. A discrete graphical model $\mathcal{M} = \langle X, \Phi \rangle$ with co-domain B and a combination operator $\oplus$ is defined by:*

- *a sequence of n variables X, each with an associated finite domain.*

- *A set of functions (or potentials)* Φ*. Each function* $\theta_S \in \Phi$ *is a function from* $D_S \to B$*.* S *is called the scope of the function and* $|S|$ *its arity.*

$\mathcal{M}$ *defines a joint function:*

$$\begin{array}{rccc} \Theta_{\mathcal{M}} : & D_X & \longrightarrow & B \\ & v & \longmapsto & \bigoplus\limits_{\theta_S \in \Phi} \theta_S(v[S]). \end{array}$$

Together with the co-domain B, the operator $\oplus$ defines a so-called valuation structure.

Definition 2. *Valuation structure. The pair* $(B, \oplus)$ *is a valuation structure if:*

- *the co-domain* B *is totally ordered by* $\preceq$*, and contains a minimum* $\bot$ *and a maximum element* $\top$*.*
- *The operator* $\oplus$ *is commutative:* $\forall a, b \in B,\ a \oplus b = b \oplus a$*.*
- *The operator* $\oplus$ *is associative:* $\forall a, b, c \in B,\ (a \oplus b) \oplus c = a \oplus (b \oplus c)$*.*
- *The operator* $\oplus$ *is monotone:* $\forall a, b, c \in B,\ (a \preceq b \implies (a \oplus c) \preceq (b \oplus c))$*.*
- *The minimum element* $\bot$ *is neutral for* $\oplus$*:* $\forall a \in B,\ a \oplus \bot = a$*.*
- *The maximum element* $\top$ *is absorbing for* $\oplus$*:* $\forall a \in B, a \oplus \top = \top$*.*

In this thesis, our work is focused mainly on additive graphical models also called Cost Function Networks (CFNs) [Dechter, 2022, Cooper et al., 2020]. The co-domain B is the set of non-negative integers bounded by $\top \in \mathbb{N} \cup \{+\infty\}$. The operator $\oplus$ is defined by the bounded addition $a +^{\top} b = \min(\top, a + b)$.

Definition 3. *Cost function network. A cost function network (CFN) is a graphical model* $\mathcal{C} = \langle X, C \rangle$ *with:*

- *a set* $X = (x_1, \ldots, x_n)$ *of* n *discrete variables. Each variable* x_i *takes its values in domain* D_i*.*
- *A set* C *of cost functions. A cost function* $c_S \in C$ *maps tuples of* D_S *to the co-domain* $\{0, \ldots, \top\}$*.*

$\mathcal{C}$ *defines a joint function:*

$$\begin{array}{rccc} C_{\mathcal{C}} : & D_X & \longrightarrow & \{0, \ldots, \top\} \\ & v & \longmapsto & \sum\limits_{c_S \in C}^{\top} c_S(v[S]). \end{array} \tag{1.1}$$

Due to their expressivity, CFNs can be used to model many types of graphical models. Cost functions with values in $\{0, \ldots, \top\}$ are soft constraints. For a particular $c_S \in C$ and a sequence $v \in D_X$, the value $c_S(v[S])$ can be seen as a violation of the constraint. The maximum element $\top$ encodes forbidden tuples, that is, partial assignments with cost $\top$ are forbidden. When a cost function takes its values in $\{0, \top\}$ it is a hard constraint.

Function representation A graphical model defines a joint function as a combination of simple functions. These functions are simple because they involve a small number of variables. Since variables have finite domains, each function can be represented as a multi-dimensional table also called a tensor. For a function $\theta_S \in \Phi$, this representation allows to map any sequence of values $v_S \in D_S$ to an element of the co-domain. To store the information, $O(d^{|S|})$ space is needed where d is the maximum domain size for the variables in S.

Throughout the manuscript, we restrict our study to pairwise graphical models, *i.e.*, the small functions all have an arity of at most 2. We then have a natural representation of the functions using vectors and matrices with values in the co-domain B. When they contain a small number of values different from the minimum element $\bot$, one can use sparse structures to compactly store the information.

Another important class is that of stochastic graphical models for which the co-domain is the real interval $[0, +\infty]$. In this case, the operator $\oplus$ is the multiplication and the functions describe non-normalized marginal probabilities over subset of variables.

Definition 4. *Markov Random Field. A Markov Random Field (MRF) [Koller and Friedman, 2009] is a graphical model $\mathcal{M} = \langle X, \Phi \rangle$ with:*

- *a set $X = (x_1, \ldots, x_n)$ of discrete random variables. Each random variable x_i takes its values in domain D_i.*
- *A set Φ of potentials. A potential $\theta_S \in \Phi$ maps tuples of D_S to the co-domain $[0, +\infty]$.*

$\mathcal{M}$ defines a joint function:

$$\begin{array}{rccc} \Theta_{\mathcal{M}} : & D_X & \longrightarrow & [0, +\infty] \\ & v & \longmapsto & \displaystyle\prod_{\theta_S \in \Phi} \theta_S(v[S]). \end{array} \tag{1.2}$$

Graph representation A pairwise graphical model can be represented as a graph, hence the terminology. For a graphical model $\mathcal{M} = \langle X, \Phi \rangle$ we have a corresponding graph $G = (V, E)$ with a vertex for each variable. For a function $\theta_S \in \Phi$ with scope $S = (x_i, x_j)$ we have an edge $e \in E$ with endpoints x_i and x_j:

$$V = X, \quad E = \{(x_i, x_j) \mid S = (x_i, x_j),\ \theta_S \in \Phi\}.$$

Discrete MRFs and CFNs are closely related. One can use logarithmic or exponential transformations to transform one into the other and vice versa. However, there exist some disparities:

- The valuation structure of CFNs is defined over (possibly bounded) non-negative integers while MRFs use non-negative real numbers.

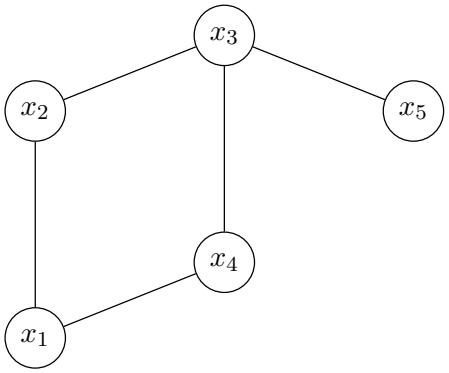


Figure 1.1: Graph representation of a pairwise graphical model $\mathcal{M} = \langle X, \Phi \rangle$ with 5 variables $X = \{x_1, x_2, x_3, x_4, x_5\}$ and 5 functions $\Phi = \{\theta_{12}, \theta_{23}, \theta_{14}, \theta_{34}, \theta_{45}\}$.

- CFNs have the ability to encode hard constraints with the maximum element $\top \in \mathbb{N} \cup \{+\infty\}$. For the case $\top = +\infty$, MRFs mimic these constraints by using zero potentials. Whether it is allowed or not, the introduction of zero potentials has led to the development of different algorithms.

For completeness, we give the two transformations. A CFN $\mathcal{C} = \langle X, C \rangle$ can be transformed into a MRF $\mathcal{M} = \langle X, \Phi \rangle$ using exponentiation. The sum operator becomes a multiplication and the costs of $\top$ are mapped to zero potentials.

$$\Phi = \left\{ \begin{array}{c} c_S \in C \\ \theta_S : v_S \mapsto \exp\left(-c_S(v_S)\right) \end{array} \right\}$$

Conversely, a MRF $\mathcal{M} = \langle X, \Phi \rangle$ can be expressed as a CFN $\mathcal{C} = \langle X, C \rangle$ with a logarithmic transformation. Note that a CFN uses integer costs, so the probabilities are shifted and rounded up to a certain precision.

$$C = \left\{ \begin{array}{c} \theta_S \in \Phi \\ c_S : v_S \mapsto \lfloor -\log\left(\theta_S(v_S)\right) \times 10^m \rfloor \end{array} \right\}$$

with m a precision parameter. Zero potentials are mapped to the maximum element $\top$.

1.2 Queries and exact optimization

In the first section, we presented discrete graphical models, a powerful framework that can express many different problems. Queries over graphical models ask one to compute simple information on the joint function such as its minimum or its mean value. Those queries cover a broad range of applications in Artificial Intelligence or Computer Science, for example.

1.2.1 Optimization over graphical models

One usual query over an MRF is to find the most probable assignment also called maximum a posteriori (MAP). A probability distribution can be recovered from the joint function by computing a normalization constant also called the partition function.

Definition 5. *Partition function. The partition function Z of a Markov Random Field $\mathcal{M} = \langle X, \Phi \rangle$ is defined as the sum of the potentials over all possible assignment $v \in D_X$:*

$$Z = \sum_{v \in D_S} \prod_{\theta_S \in \Phi} \theta_S(v[S]).$$

Computing the partition function is also a query on the MRF which is known to be #P-complete [Valiant, 1979]. We can then define the probability distribution associated with the MRF:

$$\begin{array}{rccl} p_{\mathcal{M}}: & D_X & \longrightarrow & [0,1] \\ & v_X & \longmapsto & P_{\mathcal{M}}(v) = \frac{1}{Z} \prod\limits_{\theta_S \in \Phi} \theta_S(v[S]). \end{array}$$

Definition 6. *Maximum A Posteriori. The Maximum A Posteriori (MAP) problem on a Markov Random Field $\mathcal{M} = \langle X, \Phi \rangle$ consists in finding an assignment $v \in D_X$ with maximum probability:*

$$\max_{v \in D_X} P(v) = \frac{1}{Z} \prod_{\theta_S \in \Phi} \theta_S(v[S]).$$

Note that for optimization purposes the partition function is not relevant and one can directly work with the non-normalized joint distribution. There exists an equivalent query on CFNs, namely, the weighted constraint satisfaction problem.

Definition 7. *Weighted constraint satisfaction problem. The Weighted Constraint Satisfaction Problem (WCSP) on a CFN $\mathcal{C} = \langle X, C \rangle$ consists in finding an assignment $v \in D_X$ with a minimum cost:*

$$\min_{v \in D_X} C_{\mathcal{C}}(v) = \sum_{c_S \in C}{}^{\top} c_S(v).$$

Due to the monotonicity of the logarithm and the exponential, the transformations presented above essentially show that MAP and WCSP are the same problem. As a result, the two terms are used interchangeably in the remainder of the manuscript.

Example 1. *Let us consider a CNF $\mathcal{C} = \langle X, C \rangle$ with:*

- $X = (x_1, x_2, x_3)$.
- $D_1 = \{a, b, c\},\ D_2 = \{a, b\},\ D_3 = \{a, b\}$.

- $C = \{c_\emptyset, c_1, c_{12}, c_{23}\}$.

The function $c_\emptyset$ with empty scope defines a constant term for the CFN. Since all functions have nonnegative costs, $c_\emptyset$ is a natural lower bound for the optimal value of the WCSP on $\mathcal{C}$.

In this example, we describe the cost functions as tensors and we set the maximum element $\top = +\infty$:

$c_\emptyset = 4$

c_1	
a	0
b	2
c	1

c_{12}	a	b
a	$+\infty$	3
b	0	2
c	1	0

c_{23}	a	b
a	$+\infty$	0
b	0	$+\infty$

The weighted constraint satisfaction problem asks to find the assignment of all variables that minimizes the joint function:

$$\min_{v \in D_X} C_{\mathcal{C}}(v) = c_\emptyset + c_1(v_1) + c_{12}(v_{12}) + c_{23}(v_{23}).$$

Note that the partial assignments $(x_1 = a, x_2 = a)$, $(x_2 = a, x_3 = a)$ and $(x_2 = b, x_3 = b)$ are forbidden since they all give a cost of $+\infty$. One can see that the unique solution of the WCSP is given by $v^* = (c, b, a)$ and the corresponding optimal cost is $C_{\mathcal{C}}(v^*) = 5$. For this small example, the solution can be found by complete enumeration.

1.2.2 Connections with Max Cut

MaxCut is a central problem in combinatorial optimization, it is one of the 21 Karp's NP-complete problems [Karp, 2010]. Given a weighted graph, it aims to find a partition of the vertices into two disjoint sets such that the sum of the edges between the sets is maximized. It gained a lot of attention in the past decades due to its broad range of applications in physics and computer science. It is sometimes interesting to reduce other combinatorial problems to this one, given the large number of algorithms that have been developed in recent years. In this section, we want to highlight the fact that MaxCut and the WCSP share tight links.

Notations Let $G = (V, E)$ be an undirected graph with $n = |V|$ vertices and $m = |E|$ edges. We denote by w_e the weight of an edge $e \in E$. We call a cut of the graph G a partitioning of its vertices into two disjoint subsets $V = (S, T)$. We denote by $(S : T) \coloneqq \{(x_i, x_j) \in E \mid x_i \in S,\ x_j \in T\}$ the crossing edges. With each cut, we associate a weight that corresponds to the weighted sum of the crossing edges

$$w(S, T) = \sum_{e \in (S:T)} w_e.$$

Definition 8. *MaxCut. The MaxCut problem aims to find a partition of the vertices $V = (S^*, T^*)$ such that the weighted sum $w(S^*, T^*)$ is maximized.*

Example 2. *Let us consider a graph* $G = (V, E)$ *with*

- $V = \{x_1, x_2, x_3, x_4, x_5\}$.
- $E = \{(x_1, x_2), (x_1, x_4), (x_2, x_3), (x_3, x_4), (x_3, x_5)\}$.

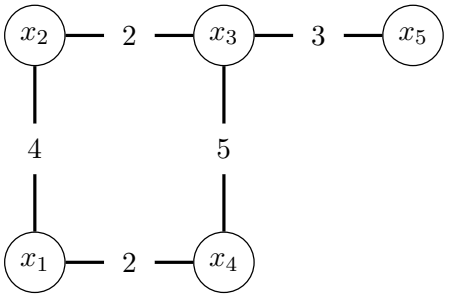


Figure 1.2: Representation of G with edge weights $w_{12} = 4$, $w_{14} = 2$, $w_{23} = 2$, $w_{34} = 5$ and $w_{35} = 3$.

The unique solution of MaxCut is given by the partition $V = (S^*, T^*)$ *with* $S^* = \{x_1, x_3\}$, $T^* = \{x_2, x_4, x_5\}$ *and value* $w(S^*, T^*) = 16$.

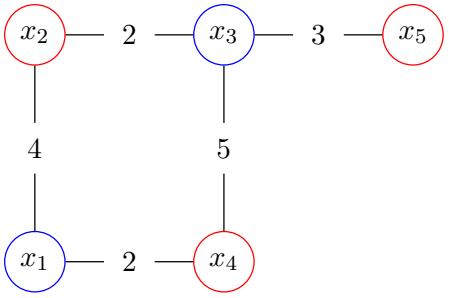


Figure 1.3: Unique solution of MaxCut on G, vertices in T^* and S^* are drawn in red and blue respectively.

Cost function networks have the ability to model MaxCut. It suffices to consider a CFN $\mathcal{C} = \langle X, C \rangle$ with $X = V$ and cost functions $C := \{c_{ij} \mid (x_i, x_j) \in E\}$. X contains Boolean variables that give the side of the cut and the cost functions are given by

c_{ij}	a	b
a	w_{ij}	0
b	0	w_{ij}

Then MaxCut is equivalent to the weighted constraint satisfaction problem on $\mathcal{C}$. We have seen that MaxCut can be modeled as a WCSP. The opposite transformation is introduced later in the manuscript. Consequently, the arsenal of optimization tools for CFNs can be used to solve MaxCut and vice versa.

1.2.3 Integer linear programming

In this section, we show that the pairwise WCSP can be reduced to an integer linear program. For simplicity, we consider the maximum element $\top = +\infty$. An integer linear program (ILP) is an optimization problem over integer variables with a linear objective function and linear constraints.

Definition 9. *Standard integer linear program. Let $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$ and a variable $x \in \mathbb{R}^n$, an integer linear program in standard form is given by the following minimization problem:*

$$\begin{aligned} \min \ & c^\top x \\ s.t \ & Ax = b \\ & x \geqslant 0 \\ & x \in \mathbb{Z}^n. \end{aligned} \tag{ILP}$$

Modeling the WCSP as an ILP Let us consider a CFN $\mathcal{C} = \langle X, C \rangle$ with $|X| = n$ variables,

- for each variable $x_i \in X$ with domain D_i we consider a set of Boolean decision variables $y_{iq} \in \{0, 1\}$ where $q \in D_i$ enumerates all possible assignments of x_i. With these notations, $y_{iq} = 1$ if and only if $x_i = q$.
- For each binary cost function $c_{ij} \in C$ we introduce a set of Boolean decision variables $z_{iq,jr} \in \{0, 1\}$ where $(q, r) \in D_i \times D_j$ enumerates all possible assignments of the pair (x_i, x_j). With these notations, $z_{iq,jr} = 1$ if and only if $(x_i, x_j) = (q, r)$.

We consider that for each variable $x_i \in X$ there exists a (potentially zero) unary cost function $c_i \in C$. The WCSP objective can be rewritten as:

$$\min \sum_{c_i \in C,\, q \in D_i} c_i(q) y_{iq} + \sum_{\substack{c_{ij} \in C \\ q \in D_i,\, r \in D_j}} c_{ij}(q, r) z_{iq,jr}.$$

Let $d = \sum_{c_S \in C} |D_S|$ and let $w \in \{0, 1\}^d$ be the vector of stacked decision variables. The WCSP is equivalent to an ILP over the set of feasible w's.

Feasible set for the WCSP First, we introduced the decision variables and the linear objective function to model the WSCP as an equivalent ILP. However, not all sequences of $\{0, 1\}^d$ variables correspond to valid assignment of the original CFN. We need to introduce some constraints to make the model exact.

- For each variable, we need to ensure that only one value is assigned:

$$\sum_{q \in D_i} y_{iq} = 1, \quad i = 1, \ldots, n.$$

- Binary assignments have to be consistent with the variables involved, these are called marginal consistency constraints:

$$\sum_{r \in D_j} z_{iq,jr} = y_{ir}, \qquad c_{ij} \in C,\ q \in D_i$$
$$\sum_{q \in D_i} z_{iq,jr} = y_{jr}, \qquad c_{ij} \in C,\ r \in D_j.$$

Finally, the WCSP is equivalent to the following ILP in $\{0,1\}$-variables:

$$\begin{aligned}
\min \quad & \sum_{c_i \in C,\, q \in D_i} c_i(q) y_{iq} + \sum_{\substack{c_{ij} \in C \\ q \in D_i,\, r \in D_j}} c_{ij}(q,r) z_{iq,jr} & \\
\text{s.t} \quad & \sum_{q \in D_i} y_{iq} = 1, & i = 1, \ldots, n \\
& \sum_{r \in D_j} z_{iq,jr} = y_{iq}, & c_{ij} \in C,\ q \in D_i \\
& \sum_{q \in D_i} z_{iq,jr} = y_{jr}, & c_{ij} \in C,\ r \in D_j \\
& y_{ia},\ z_{iq,jr} \in \{0,1\}. &
\end{aligned} \tag{1.3}$$

Without loss or generality, let us consider that each variable $x_i \in X$ has the same domain size e and the cardinality of the set of binary cost functions is f. Then, the above optimization problem has $ne + fe^2$ variables and $n + 2fe$ constraints. Note that the third and fourth set of constraints ensure that

$$\sum_{(q,r) \in D_i \times D_j} z_{iq,jr} = 1, \quad c_{ij} \in C,$$

that is, for each binary cost function, only one pair is assigned. Solving a linear integer program is itself NP-Hard, hence this transformation does not overcome the hardness of the original WCSP. However, a lot of work has focused on solving these problems in practice. For a given instance of WCSP, ILP solvers may be more efficient than solvers dedicated to WCSP, and vice versa.

1.2.4 Quadratic optimization

Due to their intrinsic linear constraints, ILPs are not suitable for expressing binary cost functions. A set of auxiliary variables must be introduced and made consistent with the original ones, which leads to new constraints. To overcome this issue, one can compactly reformulate the WCSP using Boolean quadratic optimization. Boolean quadratic optimization asks to optimize a polynomial of degree at most 2 over a (possibly constrained) set of Boolean variables.

Modeling the WCSP with quadratic optimization Let us consider a CFN $\mathcal{C} = \langle X, C \rangle$ with $|X| = n$ variables, using the same notations as above, the WCSP is equivalent to the following Boolean quadratic optimization problem:

$$\begin{aligned} \min \quad & \sum_{c_i \in C,\, q \in D_i} c_i(q) y_{iq} + \sum_{\substack{c_{ij} \in C \\ q \in D_i,\, r \in D_j}} c_{ij}(q,r) y_{iq} y_{jr} \\ \text{s.t} \quad & \sum_{q \in D_i} y_{iq} = 1, \qquad i = 1, \dots, n \\ & y_{ia} \in \{0, 1\}. \end{aligned} \tag{1.4}$$

Compared to (1.3) this problem has only ne variables and n constraints. Since the cost functions take their values in $\{0, \dots, \top\}$ the degree 2 polynomial only has positive coefficients, thus, it is called a posiform [Boros and Hammer, 2002]. Once again, solving (1.4) is NP-Hard but we show later that using quadratic optimization is a first step to transform the WCSP into an equivalent MaxCut problem.

1.3 Exact algorithms

1.3.1 Dynamic programming

Non-serial dynamic programming [Bertele and Brioschi, 1973], also known as Variable Elimination (V-E) [Koller and Friedman, 2009] is an exact method for inference in discrete graphical models. It is based on sequential reductions of the graphical model by introducing new cost functions called *messages* [Cooper et al., 2020]. More precisely, let us consider a CFN $\mathcal{C} = \langle X, C \rangle$ for which we have to solve the WCSP. Let T be the neighbors of variable $x_i \in X$ in the graph of the problem and let C_{x_i} be the set of functions c_S such that $x_i \subset S$. We can remove variable x_i from the graph by *combining* all the functions in C_{x_i}. This new function is defined over the set $T \cup x_i$. By *eliminating* variable x_i from its scope, we obtain an equivalent inference problem with one variable less.

- *Combination*: let $c_S, c_{S'} \in C$, the combination of c_S and $c_{S'}$ is a new cost function with scope $S \cup S'$:

$$c_{S \cup S'} \coloneqq (c_S + c_{S'})(v) = c_S(v[S]) + c_{S'}(v[S']).$$

- *Elimination*: let $c_S \in C$ and $x_i \in X$ such that $x_i \in S$. We denote by T the set $S \setminus x_i$, variable x_i can be eliminated from c_S resulting in a new cost function with scope T:

$$c_T(u) \coloneqq \min_{v \in D_i} c_S(u \cup v).$$

This sequence of operations has exponential time and space complexity $O(d^{|T|+1})$ with d the maximum domain size for the subset of variables $T \cup x_i$.

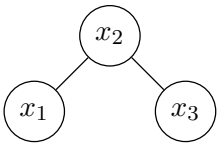
This procedure is sequentially applied until all variables are eliminated. It results in a unique cost function with empty scope $c_\emptyset$. The value of $c_\emptyset$ is the value of the

solution to the optimization query. For certain type of graphs, like trees, variable elimination can be applied efficiently. However, in the general case, the algorithm is sensitive to the elimination ordering.

Example 3. *Let us consider a CFN* $\mathcal{C} = \langle X, C \rangle$ *with:*

- $X = \{x_1, x_2, x_3\}$.
- $D_1 = \{a, b\},\ D_2 = \{a, b\},\ D_3 = \{a, b\}$.
- $C = \{c_1, c_{12}, c_{23}\}$,

and graph representation:



We describe the cost functions as tensors:

c_1	
a	1
b	3

c_{12}	a	b
a	1	1
b	2	0

c_{23}	a	b
a	0	1
b	1	2

Our query is to solve the WCSP on $\mathcal{C}$*. To do so, we show how variable elimination works on this simple problem. The elimination order for our example is* $x_1 \to x_2 \to x_3$*. First we eliminate variable* x_1 *using the procedure described above. The neighborhood of* x_1 *is* $T = \{x_2\}$ *and* $C_{x_1} = \{c_1, c_{12}\}$*. First we combine the two functions* c_1 *and* c_{12} *resulting in the new function or message:*

m_{12}	a	b
a	2	2
b	5	3

Then we eliminate variable x_1 *to obtain the equivalent WCSP:*

m_2	
a	2
b	2

c_{23}	a	b
a	0	1
b	1	2

The sequential elimination of x_2 *and* x_3 *leads to the cost function with empty scope:*

m_{23}	a	b
a	2	3
b	3	4

$\to$

m_3	
a	2
b	3

$\to \quad c_\emptyset = 2$

Which gives the optimal cost of the optimization problem.

1.3.2 Branch and bound

A basic approach to answer the minimization query (WCSP/MAP) on a discrete graphical model is based on tree-search over the space of all possible assignments. If all variables are assigned, we can compute the joint function and find the optimum by comparing the different values. The root node corresponds to the original graphical model with space state D_X. For a certain node, the space state can be split into two subsets. If we consider a variable $x_i \in X$ with domain size $d_i > 1$ we reduce the optimization query to two sub-problems, one for which x_i takes value $j \in D_i$ and one for which value j is forbidden. The algorithm exhaustively explores the *branches* of the binary search tree. Along the search, the algorithm maintains a global upper bound to keep track of the best known solution. In general, feasible assignments are calculated early in the tree by calls to heuristics. This procedure helps for computing possibly good upper bounds. At each node of the tree, bounding procedures allow for computing local lower bounds over the partially assigned space state. For minimization query, it is then possible to prune sub-trees if the local lower bound is strictly greater than the current best known solution. If it is lower than the global upper bound, the node is added to a list of open nodes. The way the tree is explored depends on the order used to arrange the sub-problems.

Breadth-First Search Nodes are stored in a first in, first out (FIFO) queue. All nodes present at a certain depth are explored before moving to a lower depth of the tree. This data structure allows for finding an optimal solution that is closest to the root of the tree. However, if a good quality solution is not found early during the search, the algorithm will not benefit from pruning. This can lead to a potentially large number of open nodes and therefore a large memory footprint.

Depth-First Search When the structure is a stack, nodes are processed in a last in, first out (LIFO) order corresponding to a depth-first search. If two subproblems are created as a result of a branching decision, they are added to the top of the stack. This strategy has low memory requirements but can get stuck on arbitrarily long bad paths until it finds a path that leads to an optimal solution.

Best-First Search One can also use a priority queue based on different policies. A fairly common strategy, called Best-First Search, consists of sorting the nodes based on their lower bounds. Compared to Depth-First Search, Best-First Search is not tied to exploring a single path of the tree. While it usually needs more memory, good solutions are found earlier in the tree.

When a leaf node is reached, it is evaluated and the global upper bound can be updated. In practice, the overall algorithm heavily depends on the quality of the bounding procedure in order to eliminate larger parts of the search tree. The efficiency is also sensitive to the branching rules, *i.e.*, the choice of variable, and the choice of value to branch on. The algorithm finds a candidate solution when it reaches a leaf node for which the value of the lower bound equals the value of the

global upper bound. An optimality proof is returned, and the algorithm terminates globally when the list of remaining open nodes is all pruned.

In most cases, Branch and Bound is the method of choice to solve WCSPs exactly. As a central part of the algorithm, our work mainly focused on the bounding procedure. One of the motivations of this thesis was to find a good compromise between the quality of the bounds and the computational efficiency of the methods. In the next sections, we present bounding procedures that are commonly used in the literature to solve combinatorial problems. Some of them are tailored to the special structure of the CFNs/MRFs.

1.4 Bounding procedures with linear relaxations

In Section 1.2.3 we showed that the WCSP can be cast as an equivalent integer linear program. One way to solve this problem is to use a continuous relaxation. The integrality constraints on the decision variables ($y_i \in \{0, 1\}$) are relaxed. The optimization problem becomes a so-called linear program (LP) [Dantzig, 1963]. This section covers a variety of methods related to linear programming in order to approximately solve the ILP (1.3) introduced previously. The reader can refer to [Kannan et al., 2019] for a recent study of this subject.

1.4.1 Definitions

Before introducing linear programming, we recall the notion of relaxation and why it is useful for our bounding procedure.

Definition 10. *Relaxation. An optimization problem* $R : z^R \coloneqq \min\{f^R(x) \mid x \in T \subseteq \mathbb{R}^n\}$ *is a relaxation of* $P : z^P \coloneqq \min\{f(x) \mid x \in X \subseteq \mathbb{R}^n\}$ *if*

- $X \subseteq T$.
- $f^R(x) \leqslant f(x), \forall x \in X$.

Usually one can obtain a relaxation of $P : z^P \coloneqq \min\{f(x) \mid x \in X \subseteq \mathbb{R}^n\}$ by weakening or removing constraints in the search space X.

Proposition 1.4.1. *If* R *is a relaxation of* P *then* $z^R \leqslant z^P$.

This proposition is straightforward considering the two assumptions in Definition 10. The optimal value of the relaxed problem z^R gives a lower bound on the optimal value of the original problem which can in turn be used for the bounding procedure. The idea behind a relaxation is to transform the original problem into a problem easier to solve. A question that attracted a lot of research is the tightness of the gap between the two solutions $|z^P - z^R|$.

Let $\mathbb{R}^n$ be the n-dimensional Euclidean space, linear programming (LP) asks to optimize a linear function over the intersection of $\mathbb{R}^n$ and a set of linear constraints.

Definition 11. *Standard linear program. Let $c \in \mathbb{R}^n$, $A \in \mathbb{R}^{m \times n}$ and a variable $x \in \mathbb{R}^n$, a linear program in standard form is given by the following minimization problem:*

$$\begin{aligned} \min \ & c^\top x \\ s.t \ & Ax = b \\ & x \geqslant 0. \end{aligned} \tag{LP}$$

The last constraint is a component-wise inequality: $x_i \geqslant 0,\ i = 1, \ldots, n$. The feasible set of (LP) is a polyhedron $\mathcal{P}$, if $\mathcal{P}$ is bounded, it is called a polytope. This problem is referred as the primal problem and x is called the primal variable.

Example 4. *Relaxation of the ILP (1.3). The LP relaxation of (1.3) is obtained by removing the integrality constraints:*

$$\begin{aligned} \min \quad & \sum_{c_i \in C,\, q \in D_i} c_i(q) y_{iq} + \sum_{\substack{c_{ij} \in C \\ q \in D_i,\, r \in D_j}} c_{ij}(q,r) z_{iq,jr} & & \\ s.t \quad & \sum_{q \in D_i} y_{iq} = 1, & & i = 1, \ldots, n \\ & \sum_{r \in D_j} z_{iq,jr} = y_{iq}, & & c_{ij} \in C,\ q \in D_i \\ & \sum_{q \in D_i} z_{iq,jr} = y_{jr}, & & c_{ij} \in C,\ r \in D_j \\ & y_{ia}, z_{iq,jr} \geqslant 0. & & \end{aligned} \tag{1.5}$$

This LP relaxation is known in the graphical model literature as the *local polytope*. Note that this relaxation is no longer an exact reformulation of the WCSP, indeed, it can have fractional solutions.

Definition 12. *Local polytope [Schlesinger, 1976, Koster, 1999, Werner, 2007]. Let us consider a GM $\mathcal{M} = \langle X, \Phi \rangle$, the local polytope is the polytope defined by the LP relaxation (1.5). We denote by $\mathbb{L}(\mathcal{M})$ the local polytope associated with $\mathcal{M}$.*

Duality From the primal problem (LP) we can derive another LP called the dual problem. Using the same notations as in Definition 11, the dual is given by the following maximization problem:

$$\begin{aligned} \max \ & b^\top y \\ \text{s.t} \ & A^\top y \leqslant c. \end{aligned} \tag{LP-Dual}$$

The weak duality theorem [Dantzig, 1963] states that for any feasible point of (LP-Dual), $b^\top y$ is a lower bound on the optimal value of (LP). There exists a stronger version of this theorem summarized below.

Theorem 1.4.2. *Duality theorem [Dantzig, 1963]. If both (LP) and (LP-Dual) have a feasible solution, then both have an optimal solution, and if x^* is an optimal solution of (LP) and y^* is an optimal solution of (LP-Dual), then*

$$c^\top x^* = b^\top y^*.$$

Example 5. *Dual of the local polytope. For simplicity, we consider a pairwise CFN $\mathcal{C} = \langle X, C \rangle$ with $|X| = n$ variables. The dual of (1.5) is given by the maximization problem:*

$$\begin{aligned} \max \ & \sum_{i=1}^{n} \lambda_i \\ s.t \ & c_i(q) - \lambda_i + \sum_{(S \in C) \cap (i \in S)} \delta_i^S(q) \geqslant 0, && i = 1, \dots, n, \ \forall q \in D_i \\ & c_S(t) - \sum_{i \in S} \delta_i^S(t_i) \geqslant 0, && \forall S \in C \text{ such that } |S| > 1, \ \forall t \in D_S. \end{aligned} \tag{1.6}$$

Solving a linear program To solve LPs, Dantzig [1963] introduced the simplex method which is regarded as one of the first numerical optimization method over constrained sets. Assuming that (LP) has a solution, this solution is necessarily located at a vertex of the polyhedron $\mathcal{P}$. The simplex algorithm iterates through vertices to find the one with minimal objective cost. In theory, this algorithm can be quite slow because $\mathcal{P}$ can have an exponential number of vertices. This method is still embedded in state-of-the-art LP solvers like CPLEX or GUROBI [Cplex, 2009, Gurobi Optimization, LLC, 2023]. Subsequently, researchers developed interior-point methods [Dikin, 1967, Karmarkar, 1984], a polynomial-time algorithm for solving LPs. The solution is found by following a path in the interior of the feasible set $\mathcal{P}$. In practice, the simplex method is sometimes preferred because it can be warm-started. This gives a non-negligible advantage when the structure of the solution is approximately known or when it is used in a branch and bound algorithm.

1.4.2 Sherali-Adams hierarchy

Let us consider a graphical model $\mathcal{M} = \langle X, \Phi \rangle$, we denote by $F \subseteq \{0,1\}^n$ the feasible set of the ILP (1.3). For our optimization purpose, it is interesting to know if the local polytope $\mathbb{L}(\mathcal{M})$ gives a tight representation of F. For example, if $\mathbb{L}(\mathcal{M}) = \text{conv}(F)$ then the LP relaxation is exact, where $\text{conv}(F)$ is the convex hull of the set F, see Definition 20. A general method for refining the LP relaxation is to introduce new constraints called cuts.

Definition 13. *Valid inequality. Let us consider a polyhedron $\mathcal{P}$, an inequality $a^\top x \leqslant b$ is valid for $\mathcal{P}$ if it is valid for all $x \in \mathcal{P}$.*

Intuitively, an inequality is valid if it does not exclude any element of the feasible set $\mathcal{P}$. These inequalities can be found by exploiting the combinatorial structure of the underlying problem. The aim is then to find cuts that are valid for F and that reduce the search space $\mathbb{L}(M)$. A celebrated class of cuts for linear program relaxations are the Gomory-Chvátal cuts [Balas et al., 1996]. In 1990, Sherali and Adams [Sherali and Adams, 1990] presented a hierarchy of continuous relaxations to approximate the feasible set of $\{0,1\}$-linear programs such as (1.3). Let $t \in$

$\{1, \ldots, ne\}$. Valid inequalities for F are obtained by multiplying the inequality constraints in $\mathbb{L}(\mathcal{M})$ by the products

$$f(I, J) = \prod_{i \in I} y_i \prod_{j \in J} (1 - y_j) \tag{1.7}$$

with I, J disjoint subsets of $\{1, \ldots, ne\}$ such that $|I \cup J| = t$. To this set of constraints, we add the inequalities $f(I, J) \geqslant 0$ with I, J disjoint subsets of $\{1, \ldots, ne\}$ such that $|I \cup J| = \min(t+1, ne)$. After linearization of the polynomials, we obtain a sequence of nested outer relaxations indexed by t:

$$F = \mathbb{L}_{ne}(\mathcal{M}) \subseteq \ldots \subseteq \mathbb{L}_{t+1}(\mathcal{M}) \subseteq \mathbb{L}_t(\mathcal{M}) \subseteq \ldots \subseteq \mathbb{L}(\mathcal{M}).$$

Note that for $t = ne$ the relaxation is exact. Linearization of the polynomials introduces a huge number of variables. As a result, this method becomes numerically impractical when we move to higher levels of the hierarchy.

1.4.3 Local polytope, duality and local consistencies

Instead of using general-purpose LP solvers for solving (1.5), research has explored specialized methods to better exploit the structural properties of graphical models. A first idea to approximate the solution of the local polytope is to switch to the dual problem 1.6). As early as the 1970s, Schlesinger worked on the local polytope and its dual for a particular graphical model with a lattice structure [Schlesinger, 1976]. It has been shown that the dual shares close links with the structure of the graphical model itself: an optimal dual solution can be interpreted as a way to transform a GM into an equivalent GM with maximum nullary potential $\theta_\emptyset$ using so-called CFN equivalence-preserving transformations (EPTs), or MRF reparametrizations.

Definition 14. *Equivalent graphical models [Cooper et al., 2020]. Two graphical models $\mathcal{M} = \langle X, \Phi \rangle$ and $\mathcal{M}' = \langle X, \Phi' \rangle$, with the same variables and valuation structure, are equivalent iff they define the same joint function: $\forall v \in D_X$, $\Theta_{\mathcal{M}}(v) = \Theta_{\mathcal{M}'}(v)$.*

Such a dual-optimal-solution-based transformation has also been called Optimal Soft Arc Consistency enforcing (OSAC [Cooper et al., 2007]). Despite the recent advances in LP solving, exact LP solvers remain too slow for many practical instances. Later, Prusa and Werner [2013] proved that solving the dual to optimality is not easier than solving any LP, which brought LP and MAP even closer together. This result partially justifies why the approaches that worked are inexact.

Cooper et al. [2010] present different classes of properties that can be enforced to obtain weaker lower bounds for the MAP problem. Informally, these consistencies are enforced through a sequence of operations, which aim to increase the value of the nullary potential $\theta_\emptyset$ without necessarily reaching dual-optimality while preserving equivalence and non-negativity, hence increasing the lower bound [Schiex, 2000]. Virtual Arc Consistency (VAC) [Cooper et al., 2008]) and message passing

algorithms such as TRW-S [Kolmogorov, 2005] can be seen as fixed-point algorithms converging towards a feasible point of the dual of the local polytope (1.6). TRW-S performs block-coordinate ascent on the dual variables until a fixed point is reached. Instead, VAC aims to increase the lower bound $\theta_\emptyset$ by planning sequences of EPTs until a fixed point is reached. These sequences are found by enforcing arc consistency, a polynomial time proof technique from constraint programming [Rossi et al., 2006]. Without loss of generality for a GM with f binary potentials and maximum domain size e, each iteration has linear complexity $O(fe^2)$ in the size of the problem for both algorithms. Note that the fixed points of VAC and TRW-S share the same properties. The reader is referred to [Wang and Koller, 2013] for a broader study of message-passing algorithms.

1.5 Bounding procedures with semidefinite relaxations

In this section, we give an introduction to the rich field of semidefinite programming (SDP). Semidefinite programming is a class of convex optimization problem for which one has to optimize a linear function over the intersection of the semidefinite cone and a set of linear constraints. In the early 1990s, SDP gained a lot of attention due to its wide application in combinatorial optimization or operations research among others. For example, the spectrahedron defined by the SDP constraints was shown to be much tighter than LP approaches to approximate the feasible set of some discrete NP-Hard problems. This section is organized in three parts:

1. Definitions 1.5.1: theoretical aspects of semidefinite programming.

2. MAP inference as constrained quadratic problem 1.5.2, Moment-SOS hierarchy 1.5.4: applications, how it is linked to our problems of interest.

3. First and second-order methods 1.5.5, Low-rank methods 1.5.6: algorithms, state-of-the-art methods to efficiently solve semidefinite programs.

1.5.1 Definitions

Semidefinite programming is interesting from a practical point of view because of its wide application in many use cases, and also as a theory, since it brings together notions from different fields such as linear algebra, and continuous and convex analysis. Compared with linear programming where one has to optimize over a subset of $\mathbb{R}^n$, semidefinite programming is a matrix optimization problem. As a central element of the SDP theory, it is then useful to recall some standard results on the set of real matrices. Let $\mathbb{R}^{n\times n}$ be the space of real matrices of size $(n \times n)$. $\mathbb{R}^{n\times n}$ is an Euclidean space equipped with the inner product

$$A, B \in \mathbb{R}^{n\times n},\ \langle A, B\rangle = \mathrm{Tr}(A^\top B) = \sum_{i,j}^{n} A_{ij}B_{ij}$$

We denote by $\|.\|_F$ the norm induced by the inner product, also referred to as the Frobenius norm

$$\|A\|_F = \langle A, A\rangle^{\frac{1}{2}} = \sqrt{\sum_{i,j=1}^{n} |A_{ij}|^2}$$

Due to its Euclidean structure, the convex analysis results on $\mathbb{R}^n$ can be extended to the space of $(n \times n)$ real matrices. In particular, for semidefinite programming, we are interested in the $\mathbb{R}^{n\times n}$ subspace of symmetric matrices. A matrix $A \in \mathbb{R}^{n\times n}$ is symmetric if $A^\top = A$. There exists a fundamental theorem that characterizes the set of symmetric matrices and their associated linear mapping.

Theorem 1.5.1. *Spectral theorem for matrices.*
Let $A \in \mathcal{S}^n$, there exists a matrix U such that $UU^\top = U^\top U = I_n$ and a diagonal matrix D with real coefficients such that

$$A = UDU^\top.$$

D is a diagonal matrix containing the eigenvalues of A

$$D_{ii} = \lambda_i,\ i = 1, \ldots, n,$$

the columns U_i of U are the corresponding eigenvectors of A

$$AU_i = \lambda_i U_i$$

A matrix $U \in \mathbb{R}^{n\times n}$ such that $UU^\top = U^\top U = I_n$ is called a unitary matrix. From its definition, the family of row or column vectors of U is an orthonormal basis of $\mathbb{R}^n$.

Positive definite matrices In semidefinite programming, the matrix variables are constrained to be positive semidefinite. We can draw a parallel with the nonnegative orthant constraint in linear programming. More generally, this constraint is a special case of a broader class known as cone constraints [Boyd and Vandenberghe, 2004]. Positive definiteness (respectively positive semidefiniteness) is a central notion in the SDP theory. We now give three equivalent definitions for real matrices.

Definition 15. *Let $A \in \mathcal{S}^n$. A is said to be positive definite if $x^\top Ax > 0$ for all non-zero $x \in \mathbb{R}^n$.*

$$A \text{ positive definite} \iff x^\top Ax > 0,\ x \in \mathbb{R}^n \setminus \{0\}$$

Let $A \in \mathcal{S}^n$, A is said to be positive semidefinite if $x^\top Ax \geqslant 0$ for all $x \in \mathbb{R}^n$.

$$A \text{ positive semidefinite} \iff x^\top Ax \geqslant 0,\ x \in \mathbb{R}^n$$

Definition 16. *Eigenvalues.*
Let $A \in \mathcal{S}^n$, A is said to be positive definite (respectively positive semidefinite) if and only if all of its eigenvalues are positive (respectively non-negative).

From the spectral theorem, since A is symmetric, its eigenvalues are necessarily real values.

Definition 17. *Decomposition.*
Let $A \in \mathcal{S}^n$, A is said to be positive semidefinite if and only if there exists a real matrix B such that

$$A = B^\top B$$

A is positive definite if and only if there exists such a decomposition with the additional constraint that B is invertible.

There exists a more general version of the last definition with matrices of rank r.

Definition 18. *Fixed rank decomposition.*
Let $A \in \mathcal{S}^n$, A is positive semidefinite with rank r if and only if there exists a full rank $(r \times n)$ matrix B such that

$$A = B^\top B$$

This last definition will be a central idea used in the low-rank section. Indeed, for certain types of SDPs, such as those with few constraints, there exist theorems that give the existence of low-rank solutions.

Convex analysis Optimization problems arise naturally in many places. For example, the search for stable 3D conformations of proteins can be cast into a discrete optimization problem. However, depending on the nature of the problem and its size *e.g.*, its number of variables or constraints, it can quickly become intractable. For some special subclasses of optimization problems like linear programming or least-square minimization, efficient methods are known to recover the optimum even for problems with more than thousands of variables.

During the second half of the 20th century, convex optimization gained a lot of attention. Indeed, using convex analysis, deterministic conditions can be derived to check for optimality. Usually, convex optimization methods aim to produce sequences converging towards points satisfying those conditions. Semidefinite programming falls in the class of convex optimization. In the next section, we recall some standard convex analysis results that are used throughout the manuscript.

Convex sets

Definition 19. *Convex set.*
Let E be a vector space over real numbers, a set $C \subseteq E$ is convex if for every $x, y \in C$ the line segment connecting x and y lies in C. Formally,

$$\forall (x, y) \in C^2,\ \forall \lambda \in [0, 1],\ \lambda x + (1 - \lambda) y \in C$$

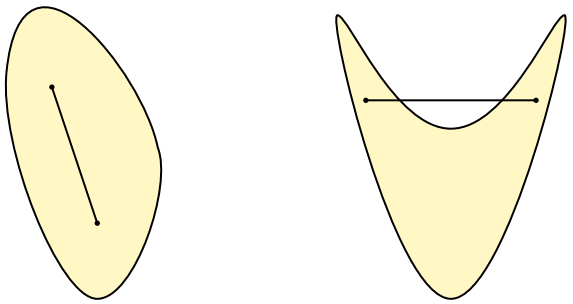


Figure 1.4: Example of convex and non-convex sets. *Left*, a simple convex set. *Right*, a non-convex set since the line segment connecting the two points of the set has elements not contained in the set.

Definition 20. *Convex combination and convex hull.*
A convex combination of the points $x_1, \ldots, x_n$ is a combination of the form

$$\{\lambda_1 x_1 + \ldots + \lambda_n x_n \mid \sum_{i=1}^{n} \lambda_i = 1,\ \lambda_i \geqslant 0,\ i = 1, \ldots, n\}$$

The convex hull of a set $C \subseteq E$ is the set defined by all the convex combinations of elements of C, denoted ***conv*** *C.*

$$\textbf{\textit{conv}}\ C := \{\lambda_1 x_1 + \ldots + \lambda_n x_n \mid x_i \in C,\ \sum_{i=1}^{n} \lambda_i = 1,\ \lambda_i \geqslant 0,\ i = 1, \ldots, n\}$$

From its definition, the convex hull is a convex set, which is also the smallest convex set containing C.

Definition 21. *Cones.*
Let E be a vector space over real numbers, a set $C \subseteq E$ is a cone if for every $x \in C$ and for every non-negative scalar λ, $\lambda x \in C$.

A set C is a convex cone if it is convex and a cone. Now it is interesting to see that the set $\mathcal{S}_+^n$ of positive semidefinite matrices is a convex cone of $\mathbb{R}^{n \times n}$. Using the spectral theorem, it is straightforward to check that for $X \in \mathcal{S}_+^n,\ \lambda \geqslant 0,\ \lambda X \in \mathcal{S}_+^n$. Moreover with the first definition of positive semidefiniteness, for $X, Y \in \mathcal{S}_+^n,\ x \in \mathbb{R}^n, \lambda \in [0, 1]$

$$x^\top(\lambda X + (1-\lambda)Y)x = \lambda x^\top X x + (1-\lambda) x^\top Y x \geqslant 0$$

Since $\mathcal{S}_+^n$ is a convex cone, it defines a partial ordering $\preceq$ on $\mathbb{R}^{n \times n}$. For $X, Y \in \mathbb{R}^{n \times n}$

$$X \preceq Y \iff Y - X \in \mathcal{S}_+^n$$

As a convex subset of $\mathbb{R}^{n \times n}$, we can define the projection onto the cone of positive semidefinite matrices $\mathcal{S}_+^n$.

Theorem 1.5.2. *Projection onto $\mathcal{S}_+^n$ [Boyd and Vandenberghe, 2004].*
Let $P_{\mathcal{S}_+^n}(X_0)$ be the Euclidean projection of a matrix X_0 into the convex cone $\mathcal{S}_+^n$.

$$P_{\mathcal{S}_+^n}(X_0) = \sum_{i=1}^{n} \max\{0, \lambda_i\} U_i U_i^\top$$

where $X_0 = \sum_{i=1}^{n} \lambda_i U_i U_i^\top$ is the spectral decomposition of X_0.

This latter result will be useful for projective optimization methods such as projected gradients or augmented Lagrangian. The projection basically amounts to removing the negative eigenvalues in the spectral decomposition of the symmetric matrix.

Convex functions

Definition 22. *Convex functions.*
Let E be a vector space over real numbers, a function $f : C \to \mathbb{R}$ is convex if C is convex and for all $x, y \in C,\ \lambda \in [0, 1]$

$$f(\lambda x + (1 - \lambda)y) \leqslant \lambda f(x) + (1 - \lambda) f(y).$$

A function f is strictly convex if the inequality above holds strictly for $x \neq y$ and $0 < \lambda < 1$. A function f is concave (respectively strictly concave) if $-f$ is convex (respectively strictly convex).

Intuitively, a function is convex over a convex set C if for all points x and y in C, the image of the mean of x and y is smaller than the mean of the images.

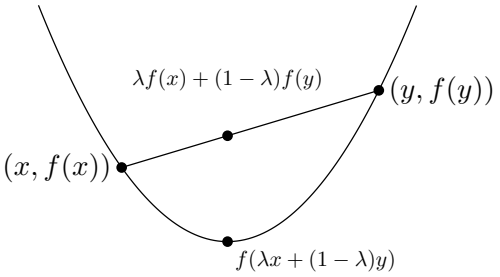


Figure 1.5: Example of a simple convex function. The image $f(\lambda x + (1 - \lambda)y)$ is always below the segment defined by the points $(x, (f(x))$ and $(y, f(y))$ which is called a cord of f.

First and second order conditions

For differentiable and twice differentiable functions there exist necessary and sufficient conditions to check for convexity. In practice, those conditions are fundamental

for convex optimization. Let f be a differentiable function, that is, the gradient ∇f exists for each point in the domain of f, f is convex if and only if

$$f(y) \geqslant f(x) + \nabla f(x)^\top (y - x) \tag{1.8}$$

for every x, y in the domain of f. Intuitively, f is above its tangents at every point.

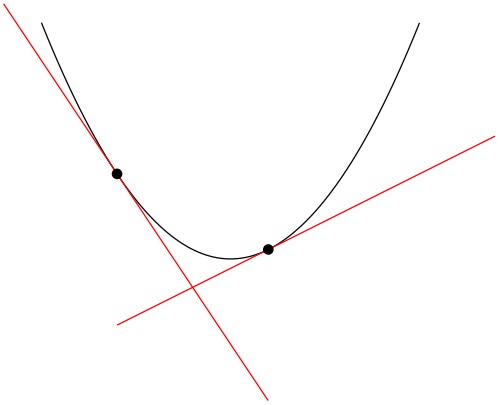


Figure 1.6: Example of a simple convex function. The tangent at each point is a global linear underestimator of the function.

Interestingly, for a point x such that $\nabla f(x) = 0$, using the equation (1.8), for all y in the domain of f, $f(y) \geqslant f(x)$. It means that $f(x)$ is a global lower bound on f. In the convex settings, first-order optimization algorithms aim to converge towards points satisfying $\nabla f(x) = 0$. Those points are called critical points of f. If the inequality is strict for $x \neq y$ then the function is strictly convex. Note that for convex functions, critical points are not necessarily unique while strict convexity gives uniqueness of the global minimizer.

Now, let f be a twice differentiable function, that is, the Hessian $\nabla^2 f$ exists at each point of the domain of f, f is convex if and only if

$$\nabla^2 f(x) \succeq 0$$

for every x in the domain of f. In other words, the Hessian is positive semidefinite.

Semidefinite programming We now introduce the concept of semidefinite programming. Semidefinite programming (SDP) asks to optimize a linear function over the intersection of the semidefinite cone and an affine space defined by a set of linear constraints [Wolkowicz et al., 2012]. Semidefinite programming can be seen as an extension of linear programming where the nonnegative orthant is replaced by the convex cone of positive semidefinite matrices $\mathcal{S}_+^n$. Due to their similarities, some of the optimization methods used in linear programming were extended to semidefinite programs.

Example 6. *Let us consider a linear program in its normal form*

$$\begin{aligned} \min\ & c^\top x \\ s.t\ & Ax \leqslant b \\ & x \geqslant 0 \end{aligned} \tag{LP}$$

where $x, c, b \in \mathbb{R}^n$ and $A \in \mathbb{R}^{m\times n}$ is the matrix of constraints. If we consider the matrix $X = \mathrm{Diag}(x)$ then the constraint $x \geqslant 0$ is equivalent to the positive semidefiniteness of X. In the same way, if we introduce the matrices

$$A_i = \begin{pmatrix} a_{i1} & & \\ & \ddots & \\ & & a_{in} \end{pmatrix} \qquad C = \begin{pmatrix} c_1 & & \\ & \ddots & \\ & & c_n \end{pmatrix}$$

then the linear program (LP) is equivalent to the following semidefinite program

$$\begin{aligned} \min\ & \langle C, X\rangle \\ s.t\ & \langle A_i, X\rangle \leqslant b_i \qquad i = 1, \dots, m \\ & X_{ij} = 0 \qquad i, j = 1, \dots, n,\ i \neq j \\ & X \succeq 0 \end{aligned}$$

Although LPs can easily be written as SDPs, it's generally not a good idea to perform this transformation, as optimization methods for SDPs are generally more computationally intensive.

Duality

As in the case of linear programming, duality theory also applies to semidefinite programming. However all the results from LP do not necessarily hold for the SDP case. We now derive some usual SDP duality results. We consider a generic SDP in its standard form

$$\begin{aligned} \min\ & \langle C, X\rangle \\ \text{s.t}\ & \langle A_i, X\rangle = b_i \quad i = 1, \dots, m \\ & X \succeq 0 \end{aligned} \tag{SDP}$$

This problem is usually referred to as the primal problem and the variable $X \succeq 0$ is called the primal variable. The Lagrangian for the minimization problem (SDP) is

$$\mathcal{L}(X, \mu, S) = \langle C, X\rangle + \sum_{i=1}^{m} \mu_i (b_i - \langle A_i, X\rangle) - \langle X, S\rangle$$

where $S \succeq 0$ is the dual variable for the conic constraint $-X \preceq 0$. The corresponding dual function is then given by

$$g(S, \mu) = \min_X \mathcal{L}(X, \mu, S) = \begin{cases} \mu^\top b & \text{if } C - \sum_{i=1}^{m} \mu_i A_i - S = 0 \\ +\infty & \text{otherwise} \end{cases}$$

The dual problem is

$$\begin{aligned} \max_{\mu} \ & \mu^\top b \\ \text{s.t} \ & \sum_{i=1}^{m} \mu_i A_i + S = C \\ & S \succeq 0 \end{aligned} \tag{SDPD}$$

Proposition 1.5.3. *Weak duality.*
Let p^* be the optimal value of (SDP) and d^* be the optimal value of (SDPD). Weak duality states that

$$p^* \geqslant d^*.$$

Moreover, for every primal-dual feasible pair $(X, (\mu, S))$

$$\langle C, X \rangle \geqslant \mu^\top b$$

Before going through the proof of this theorem, we introduce a useful lemma which is a well-known result for positive semidefinite matrices.

Lemma 1.5.4. *Let X be a $(n \times n)$ symmetric matrix, X is positive semidefinite if and only if for all $Y \in \mathcal{S}_+^n$, $\langle X, Y \rangle \geqslant 0$.*

Proof. $\Longleftarrow$ By contrapositive statement, if X is not positive semidefinite, there exists $y \in \mathbb{R}^n$ such that $y^\top X y = \langle X, yy^\top \rangle < 0$. But $Y = yy^\top$ is positive semidefinite thus there exists a matrix $Y \in \mathcal{S}_+^n$ such that $\langle X, Y \rangle < 0$.
$\Longrightarrow$ Let $X \in \mathcal{S}_+^n$, by the spectral theorem, there exists a unitary matrix $U \in \mathbb{R}^{n \times n}$ such that $X = UDU^\top$. D is the diagonal matrix containing the eigenvalues of X which are all nonnegative since $X \succeq 0$. Let $Y \in \mathcal{S}_+^n$ and $\hat{Y} = U^\top Y U$, *i.e.*, $Y = U\hat{Y}U^\top$. $\hat{Y}$ is positive semidefinite, indeed, for $v \in \mathbb{R}^n$

$$v^\top \hat{Y} v = v^\top U^\top Y U v = (Uv)^\top Y (Uv) \geqslant 0.$$

Then we have

$$\begin{aligned} \langle X, Y \rangle &= \operatorname{Tr}(UDU^\top Y) = \operatorname{Tr}(UDU^\top U\hat{Y}U^\top) \\ &= \operatorname{Tr}(UD\hat{Y}U^\top) = \operatorname{Tr}(D\hat{Y}) = \sum_{i=1}^{n} D_{ii}\hat{Y}_{ii}. \end{aligned}$$

Since $\hat{Y}$ is positive semidefinite, its diagonale entries are nonnegative hence

$$\sum_{i=1}^{n} D_{ii}\hat{Y}_{ii} \geqslant 0$$

which completes the proof. □

The proof of the weak duality proposition follows naturally from the lemma.

Proof. Suppose that X and (μ, S) are primal and dual feasible, we have

$$\langle X, C\rangle = \langle X, \sum_{i=1}^{m} \mu_i A_i + S\rangle$$

$$= \sum_{i=1}^{m} \mu_i \langle X, A_i\rangle + \langle X, S\rangle = \mu^\top b + \langle X, S\rangle.$$

From Lemma 1.5.4, $\langle X, S\rangle \geqslant 0$ which gives the desired result

$$\langle X, C\rangle \geqslant \mu^\top b$$

□

Proposition 1.5.5. *Strong duality [Vandenberghe and Boyd, 1996].*
We have $p^* = d^*$ if either of the following conditons holds

1. *The primal problem (SDP) is strictly feasible, i.e., there exists a X with $X \succ 0$ and $\langle A_i, X\rangle = b_i,\ i = 1, \ldots, m$.*
2. *The dual problem (SDPD) is stricly feasible, i.e., there exists a μ with $C - \sum_{i=1}^{m} \mu_i A_i \succ 0$.*

If both conditions hold, the optimal sets of the two problems are nonempty.

For a proof of this theorem, see [Nesterov and Nemirovskii, 1994, Chapter 4.2]. Those two conditions are referred to as Slater's condition for the primal and dual problem respectively. The dual problem gives relevant information on the optimal value of the primal problem (SDP). Indeed, we showed that any dual feasible point (μ, S) produces a lower bound on the original problem with guarantees. These certificates are crucial in exact optimization methods such as branch-and-bound. Moreover, duality is symmetric, one can show using classic linear algebra arguments that (SDP) is also the dual problem of (SDPD).Thus, when strong duality holds, it is sometimes convenient to switch from one formulation to the other. However, it is important to notice that, compared with linear programming, strong duality does not always holds, even for small problems.

In the remainder of the manuscript, we mainly use semidefinite programs in the standard form. We now give the two forms with their duals. Let $C, (A_i)_{i=1,\ldots,m} \in \mathcal{S}^n$.

$$\begin{aligned} \min\ & \langle C, X\rangle \\ \text{s.t}\ & \langle A_i, X\rangle = b_i \quad i = 1, \ldots, m \\ & X \succeq 0, \end{aligned}$$

with its dual:

$$\begin{aligned} \max\ & \sum_{i=1}^{m} b_i y_i \\ \text{s.t}\ & \sum_{i=1}^{m} A_i y_i + S = C \\ & S \succeq 0. \end{aligned}$$

Equivalently, SDP can be formulated using the linear operator $\mathcal{A}$.

$$\begin{aligned} \min \ & \langle C, X \rangle \\ \text{s.t} \ & \mathcal{A}(X) = b \\ & X \succeq 0, \end{aligned}$$

with its dual:

$$\begin{aligned} \max \ & b^\top y \\ \text{s.t} \ & \mathcal{A}^*(y) + S = C \\ & S \succeq 0. \end{aligned}$$

1.5.2 MAP inference as constrained binary quadratic problem

As we saw previously, MAP inference on pairwise discrete graphical models can be transformed into a linear programming problem. However, for binary potentials, one has to introduce a variable for each pair within the scope of the function. This leads to a quadratic number of variables. Due to their innate ability to deal with quadratic terms, binary quadratic formulations are well suited to express the pairwise MAP problem in a more compact way.

We consider a discrete pairwise GM $\mathcal{M} = \langle X, \Phi \rangle$ with $|X| = n$ variables and its associated graph $G = (V, E)$. Each variable $x_i \in X$ can take d_i possible values, we note $d = \sum_{i=1}^{n} d_i$ the total number of values. We recall that the GM $\mathcal{M}$ defines a joint function Θ_M over X as

$$\Theta_M = \sum_{\theta_S \in \Phi} \theta_S.$$

The goal of the Maximum A Posteriori (MAP) problem is to find an assignment v of all the variables which minimizes the joint function [Cooper et al., 2020]

$$\arg\min_{v \in D_X} \Theta_M(v) = \sum_{\theta_S \in \Phi} \theta_S(v[S]) \tag{MAP}$$

Using the associated graph, we can rewrite (MAP) as

$$\arg\min_{v \in D_X} \Theta_M(v) = \sum_{x_i \in X} \theta_i(x_i) + \sum_{(x_i, x_j) \in E} \theta_{ij}(x_i, x_j)$$

In order to obtain a quadratic reformulation, we rewrite each variable $x_i \in X$ as a boolean vector $b_i \in \{0, 1\}^{d_i}$. This encoding is usually referred to as 1-hot or direct encoding of the variable x_i. The j^{th} element of b_i will take value 1 when variable $x_i = j$ and value 0 otherwise. Usually, for general arities, potentials are represented as tensors. For unary and binary potentials we can naturally represent them as real vectors and matrices respectively. We denote by $\{c_i |\ x_i \in X\}$ and $\{Q_{ij} |\ (x_i, x_j) \in E\}$ the corresponding vectors and matrices. The value of $\theta_{ij}(x_i, x_j)$

is $b_i^\top Q_{ij} b_j$ and the value of $\theta_i(x_i)$ is then $c_i^\top b_i$. We now introduce the stacked vector $b^\top = [b_1^\top, \dots, b_n^\top] \in \{0,1\}^d$, the joint function becomes

$$\Theta_M(b) = b^\top Q b + c^\top b.$$

Where $c^\top = [c_1^\top, \dots, c_n^\top]$ is the stacked vector of all unary costs and Q is the $(d \times d)$-block matrix which contains all the binary cost matrices, *i.e.*, Q has block Q_{ij} if $(x_i, x_j) \in E$ and $0^{d_i \times d_j}$ otherwise.

When optimizing over b instead of v, one must enforce the fact that only one element of every Boolean vector b_i is set to 1, *i.e.*, that $\forall x_i \in X$, $\sum_{j=1}^{d_i} (b_i)_j = 1$. This set of constraints can be gathered in a linear constraint $Ab = 1_n$ where A is a Boolean matrix of size $(n \times d)$ with

$$A_i^\top = [0_{d_1}^\top \ \cdots \ 0_{d_{i-1}}^\top \ 1_{d_i}^\top \ 0_{d_{i+1}}^\top \ \cdots \ 0_{d_n}^\top], \ i = 1, \dots, n. \tag{1.9}$$

In the rest of the manuscript, this constraint will be referred to as the "exactly-one" constraint. Finally, minimizing the joint function Θ_M becomes equivalent to solving the following constrained quadratic program

$$\begin{aligned} \min_{b \in \{0,1\}^d} \quad & b^\top Q b + c^\top b \\ \text{s.t} \quad & Ab = 1_n \end{aligned} \tag{1.10}$$

Since the binary quadratic constrained problem (1.10) is NP-Hard in general, numerous methods were proposed to approximately solve it. One idea is to relax the integrality constraint $x \in \{0,1\}^d$ into the box constraint $x \in [0,1]^d$

$$\begin{aligned} \min_{b \in [0,1]^d} \quad & b^\top Q b + c^\top b \\ \text{s.t} \quad & Ab = 1_n \end{aligned} \tag{1.11}$$

The idea is then to use continuous optimization methods in order to solve the relaxed problem or its dual. Unfortunately, if the cost matrix Q is not positive semidefinite, (1.11) falls in the class of non-convex problems. There exist methods in order to relax (1.10) as convex quadratic problems using the equality $b_i^2 = b_i$ which always holds for binary variables. In [Hammer and Rubin, 1970] the authors showed that it is possible to introduce a shifting of the cost matrix with a real parameter γ such that $Q + \gamma I_d$ is positive semidefinite and the two functions $f(b) = b^\top Q b$ and $g(b) = b^\top (Q + \gamma I_d) b - \gamma 1_n^\top b$ coincide on $\{0,1\}^d$. Another simple idea from [Billionnet, 2007] is to reformulate the non-convex products $x_i x_j$ as the convex or concave functions $\frac{1}{2}((x_i + x_j)^2 - x_i - x_j)$ or $\frac{1}{2}(-(x_i + x_j)^2 + x_i + x_j)$. While being easy to implement, these reformulations can lead to poor continuous relaxations.

In the literature, two convex relaxations are commonly known to produce tight bounds for quadratically constrained quadratic programming (QCQP). The first one is based on semidefinite programming and the second one is the reformulation

linearization technique (RLT) [Anstreicher, 2009, Sherali and Adams, 2013]. If we consider a quadratically constrained quadratic program of the form:

$$\begin{aligned} \min \;& x^\top Q_0 x + c_0^\top x \\ \text{s.t} \;& x^\top Q_i x + c_i^\top x \leqslant b_i, \quad i \in \mathcal{I} \\ & x^\top Q_i x + c_i^\top x = b_i, \quad i \in \mathcal{E} \\ & l \leqslant x \leqslant u \end{aligned} \tag{QCQP}$$

For which (1.11) is a special case, the idea is to replace variable products $x_i x_j$ by a new variable X_{ij}. For the SDP relaxation, the change of variable $X = xx^\top$ is used. This introduces the new cone constraint $X - xx^\top \succeq 0$ which is known to be equivalent to

$$\bar{X} := \begin{pmatrix} 1 & x^\top \\ x & X \end{pmatrix} \succeq 0$$

We can then rewrite (QCQP) as

$$\begin{aligned} \min \;& \langle \bar{Q}_0, \bar{X} \rangle \\ \text{s.t} \;& \langle \bar{Q}_i, \bar{X} \rangle \leqslant b_i, \quad i \in \mathcal{I} \\ & \langle \bar{Q}_i, \bar{X} \rangle = b_i, \quad i \in \mathcal{E} \\ & l \leqslant x \leqslant u, \quad \bar{X} \succeq 0 \end{aligned} \tag{1.12}$$

with

$$\bar{Q}_i = \begin{pmatrix} 0 & \frac{1}{2} c_i^\top \\ \frac{1}{2} c_i & Q_i \end{pmatrix}$$

This relaxation has shown good approximation results for some problems of interest, see [Luo et al., 2010] for an extensive study.

For the RLT relaxation, the idea is to use products of bound constraints on the variable x to create new valid inequalities for (QCQP). Indeed, let us consider the two variables $l_i \leqslant x_i \leqslant u_i$ and $l_j \leqslant x_j \leqslant u_j$ and the new variable $X_{ij} = x_i x_j$. If we multiply the bound constraints regarding x_i with the bound constraints regarding x_j we can obtain the four following valid inequalities

$$\begin{aligned} X_{ij} - l_i x_j - l_j x_i &\geqslant -l_i l_j \\ X_{ij} - u_i x_j - u_j x_i &\geqslant -u_i u_j \\ X_{ij} - l_i x_j - u_j x_i &\leqslant -l_i u_j \\ X_{ij} - l_j x_i - u_i x_j &\leqslant -l_j u_i \end{aligned}$$

If we add the symmetry constraint to the new variable $X = xx^\top$, the two last

constraints become redundant. Thus, we can also rewrite (QCQP) as

$$\begin{aligned}
\min \;& \langle Q_0, X\rangle + c_0^\top x \\
\text{s.t} \;& \langle Q_i, X\rangle + c_i^\top x \leqslant b_i, && i \in \mathcal{I} \\
& \langle Q_i, X\rangle + c_i^\top x = b_i, && i \in \mathcal{E} \\
& X - lx^\top - xl^\top \geqslant -ll^\top \\
& X - ux^\top - xu^\top \geqslant -uu^\top \\
& X - lx^\top - xu^\top \leqslant -lu^\top \\
& l \leqslant x \leqslant u, && X = X^\top
\end{aligned} \tag{1.13}$$

Due to the symmetry of X, this problem can be seen as a linear program with $n(n+3)/2$ variables and $m + n(2n+3)$ constraints. The additional constraints obtained in RLT can be used to further constrained the SDP relaxation. For recent advances in solving QCQP, the reader is referred to the work of Elloumi Sourour and Amélie Lambert [Elloumi and Lambert, 2019].

1.5.3 Log-encoding of the discrete variables

When using 1-hot encoding the size of the quadratic problem reformulation is given by the sum d of the domain size of every discrete variable. Scalability is always a critical issue when using optimization methods, which is why a significant resolution time can be spent pre-processing the problem. The pre-processing steps aim to reduce the problem size by finding symmetries or using smart variable eliminations for example. Naturally, the question arises as to whether there is a more concise way of representing the MAP problem than using 1-hot encoding.

To this end, we have explored a method derived from constraint programming called log-encoding [Walsh, 2000]. The idea is to find a mapping between two NP-Hard problems, namely constraint satisfaction problems (CSPs) and propositional satisfiability (SAT). On the one hand, propositional satisfiability are problems with Boolean variables and non-binary constraints. On the other hand, constraint satisfaction problems generally have binary constraints on variables with non-Boolean domains. Log-encoding allows for representing finite non-Boolean domains as a set of Boolean variables. With this encoding, CSPs for which weighted constraint satisfaction problems (WCSPs) are a generalization, can be mapped to SAT.

Let us consider a GM variable $x_i \in X$ with domain size d_i. We associate a Boolean variable b_j which is assigned to 1 if the jth bit of x_i is set. We need at most $\lceil log_2(d_i)\rceil$ variables to fully represent the domain of x_i. Note that if the domain size d_i is not a power of 2, we can add additional constraints on the b_j's to prohibit the remaining values at the top of the domain. Moreover, it is not necessary to enforce the "exactly-one" constraint as in (1.10) since it is implicitly satisfied by the log-encoding. Without loss of generality, let us consider a GM $M = \langle X, \Phi\rangle$, such that each variable $x_i \in X$ has the same domain size l and such that $\lceil log_2(l)\rceil = log_2(l)$. To represent all the variables using log-encoding, we need $nlog_2(l)$ Boolean

variables while nl Boolean variables are needed using the direct 1-hot encoding. One can see that we benefit asymptotically from the log-encoding representation, but we already have gains for domains of moderate size.

Variables are concisely represented using this new encoding, now we need to check whether it is also the case for the constraints, and therefore for the objective function to be minimized. To do so, we introduce some notions of pseudo-Boolean optimization. Let $V = \{1, 2, \ldots, n\}$, we denote by $\mathbb{B} = \{0, 1\}$ and $\mathbb{B}^n$ the set of Boolean vectors of size n. Mappings $f : \mathbb{B}^n \mapsto \mathbb{R}$ such that f is given explicitly by an algebraic expression are called pseudo-Boolean functions. Let us consider a Boolean vector $b = (b_1, \ldots, b_n) \in \mathbb{B}^n$. For a variable $b_i \in \mathbb{B}$, since values 0 and 1 are symmetric, it is convenient to introduce the complement $\bar{b_i} = 1 - b_i$ and the set of literals $L = \{b_1, \bar{b_1}, \ldots, b_n, \bar{b_n}\}$. In [Boros and Hammer, 2002] the authors have shown that all pseudo-Boolean functions can be uniquely represented as multi-linear polynomials of the form

$$f(b_1, \ldots, b_n) = \sum_{S \subseteq V} c_S \prod_{j \in S} b_j \tag{1.14}$$

The degree of f is the largest subset $S \subseteq V$ such that $c_S \neq 0$. The size of f is the total number of variable occurrences in it

$$\text{size}(f) = \sum_{S : c_S \neq 0} |S|$$

Finally, pseudo-Boolean functions can be represented as posiforms over the set of literals

$$f(b_1, \ldots, b_n) = \sum_{T \subseteq L} a_T \prod_{u \in T} u \tag{1.15}$$

where $a_T \geqslant 0$. Now, we go back to our original problem of representing GM potentials. Let us first consider a unary potential $\theta_i \in \Phi$. Considering all non zero costs, θ_i has a trivial posiform representation. Indeed, for a variable x_i with domain size d_i we can compute the unary cost

$$\theta_i(x_i) = \sum_{c_j \neq 0} c_j bin(j)$$

Where $bin(j)$ is the bit representation of value $j \in [d_i]$ using $\lceil log_2(d_i) \rceil$ Boolean variables.

Example 7. *Log encoding of a unary potential.*

x_i	$\theta_i(x_i)$
0	*0*
1	*4*
2	*1*
3	*0*

In this case, we have a variable with domain size 4 *and a corresponding unary potential. To represent the domain of* x_i *we introduce* $l = 2$ *Boolean variables* b_0 *and* b_1 *as well as their complements* $\bar{b_0}$ *and* $\bar{b_1}$*. The posiform for* θ_i *is then*

$$\theta_i(x_i) = 4b_0\bar{b_1} + \bar{b_0}b_1$$

Working on this posiform, $\bar{b_0}$ *and* $\bar{b_1}$ *can be replaced by* $1 - b_0$ *and* $1 - b_1$ *to obtain a polynomial in the variables* b_0 *and* b_1*. By doing so, the polynomial can possibly have non-positive coefficients. Note that as long as the unary potential is not identically equal to zero, the degree of the polynomial is* $\lceil log_2(d_i) \rceil$*.*

Following the same ideas, for a binary potential θ_{ij} with scope (x_i, x_j), considering all non zero costs, the posiform is given by

$$\theta_{ij}(x_i, x_j) = \sum_{Q_{kl} \neq 0} Q_{kl} bin(k) bin(l)$$

Where $bin(k)$ and $bin(l)$ are the bit representation of value $k \in [d_i]$ and $l \in [d_j]$ using $\lceil log_2(d_i) \rceil$ and $\lceil log_2(d_j) \rceil$ Boolean variables respectively.

Example 8. *Log encoding of a binary potential.*

x_i	x_j	$\theta_i(x_i, x_j)$
0	*0*	*2*
0	*1*	*0*
0	*2*	*0*
0	*3*	*0*
1	*0*	*0*
1	*1*	*1*
1	*2*	*0*
1	*3*	*0*
2	*0*	*0*
2	*1*	*0*
2	*2*	*3*
2	*3*	*0*
3	*0*	*0*
3	*1*	*0*
3	*2*	*0*
3	*3*	*4*

In order to represent the domains of x_i *and* x_j *we introduce four Boolean variables as well as their complements,* $L = \{b_0, \bar{b_0}, b_1, \bar{b_1}, b_2, \bar{b_2}, b_3, \bar{b_3}\}$*. The posiform for* θ_{ij} *is given by*

$$\theta_{ij}(x_i, x_j) = 2\bar{b_0}\bar{b_1}\bar{b_2}\bar{b_3} + b_0b_2\bar{b_1}\bar{b_3} + 3b_1b_3\bar{b_0}\bar{b_2} + 4b_0b_1b_2b_3$$

As in the previous case, it is easy to see that if the binary potential is not identically equal to zero, its degree is $\lceil log_2(d_i) \rceil \times \lceil log_2(d_j) \rceil$*.*

Optimizing over general pseudo-Boolean functions of degree higher than two is a hard task even for small problems. However, there exists a procedure to reduce any problem to a quadratic optimization problem. The generic algorithm is shown in Algorithm 1. The overall idea relies on the fact that quadratic terms can be linearized. Indeed, we can replace a quadratic term with a new variable by adding a certain penalization to the objective function with a large penalty coefficient. The downside of this procedure is the introduction of a potentially large number of new variables.

Algorithm 1 ReduceMin(f)

input: A pseudo-Boolean function f given by its multi-linear polynomial form (1.14).

initialization: Set $M = 1 + 2\sum_{S\subseteq V} |c_S|$, $m = n$, and $f^n = f$.

while There exists a subset $S^* \subseteq V$ for which $|S^*| > 2$ and $c_{S^*} \neq 0$ **do**

1. Choose two elements i and j from S^* and update
   $c_{i,j} = c_{i,j} + M$,
   set $c_{i,m+1} = c_{j,m+1} = -2M$ and
   $c_{m+1} = 3M$ and
   for all subsets $S \supseteq \{i,j\}$ with $c_S \neq 0$ define $c_{(S\setminus\{i,j\})\cup\{m+1\}} = c_S$ and set $c_S = 0$.
2. Define $f^{m+1}(b_1,\ldots,b_{m+1}) = \sum_{S\subseteq V} c_S \prod_{k\in S} b_k$, and set $m = m+1$.

end while

output: Set $g = f^m$.

Theorem 1.5.6. *[Rosenberg, 1975].*
ReduceMin(f) terminates in polynomial time in the size of f, and produces a pseudo-Boolean function g in m variables, the size of which is polynomially bounded in size(f), and such that

$$\min_{y\in\mathbb{B}^n} g(y) = \min_{x\in\mathbb{B}^n} f(x).$$

For a proof of this theorem see [Boros and Hammer, 2002, Chapter 4.4]. As suggested by the algorithm, the reduction is not unique. Finding the sequence of operations that minimizes the number of new variables m is an NP-hard problem [Boros and Hammer, 2002]. Thus, for unary and binary potentials, log-encoding followed by reduction to quadratic functions does not necessarily lead to a more concise formulation of the MAP problem, and even if it does, it is not computationally easy to find it.

1.5.4 Moment-SOS hierarchy

In the last section, we showed that we can write the MAP inference problem as a binary quadratic problem. It is also known that MAP inference on pairwise

discrete graphical models can be transformed into an LP over the so-called *marginal polytope* [Wainwright et al., 2008]. We denote by $\mathbb{M}(\mathcal{M})$ the marginal polytope associated with the pairwise graphical model $\mathcal{M}$. Solving this LP is intractable in general due to an exponential number of constraints. To overcome this issue, the standard LP relaxation over the local polytope, see Definition 12, was introduced. We denote by $\mathbb{L}(\mathcal{M})$ the local polytope associated with $\mathcal{M}$, $\mathbb{L}(\mathcal{M})$ form a convex polyhedral outer bound on $\mathbb{M}(\mathcal{M})$. Now a question naturally arises: Can we get a tighter approximation of $\mathbb{M}(\mathcal{M})$ using SDP relaxations and how do they compare with the local polytope formulation.

Without loss of generality, let us consider a GM $\mathcal{M}$ with n discrete variables with domain size r and let $d = nr$. Let G be its associated graph. For an even integer m we denote by $\binom{d-1}{\leqslant m}$ the set of subsets of $\{0, \dots, d-1\}$ with size less than m. We introduce the real random vector X indexed by $\binom{d-1}{\leqslant m}$ with $X_\emptyset = 1$. Finally, we define the moment matrix $M(X)$ indexed by the two sets $S, T \in \{0, \dots, d-1\}$, $|S|, |T| \leqslant \frac{m}{2}$ with entries $M(X)_{S,T} = X_{S \Delta T}$ where $S \Delta T$ is the symmetric difference of the two sets S and T. Enforcing positive semidefiniteness of the moment matrix $M(X)$ with parameter m is equivalent to enforcing the pseudo marginals on hyperedges of G of size $\frac{m}{2}$ [Wainwright et al., 2008]. Those sets define a sequence of outer bounds on the marginal polytope $\mathbb{M}(G)$ also known as Lasserre sequence of relaxations [Lasserre, 2002, Laurent, 2003]. In [Erdogdu et al., 2017] the authors used a partial Sum-of-Square lifting with terms of degree $m = 4$ on pairwise graphical models with Boolean variables. The moment-SOS hierarchy has found recent practical applications for approximately solving discrete optimization problems like MaxCut or Max2Sat [Sinjorgo and Sotirov, 2023, Campos et al., 2022, Wang et al., 2022]. It is done by partially lifting the relaxation to higher orders of the hierarchy.

1.5.5 First and second order methods

We now give an overview of the first and second-order methods used to solve SDPs. The modern primal-dual interior point algorithm was proposed by Karmarkar in 1984 to solve linear programs [Karmarkar, 1984]. It was later extended to more general non-linear convex programs. These methods are useful in practice, as they enable us to obtain solutions with desired accuracy in polynomial time. The extension to non-linear convex programs have been made possible by the study of a new type of convex functions called self-concordant functions [Nesterov and Nemirovskii, 1994]. Basically, Nesterov and Nemirovsky derived a new class of convex functions that can be efficiently minimized by Newton's method. A self-concordant function is a function such that its third derivative is bounded by a constant and a term that depends on the norm defined by its Hessian.

In the 1990's it was successfully applied to the growing field of semidefinite programming and it became the method of choice compared to the slower ellipsoid method. Modern SDP solvers (MOSEK [ApS, 2022], SDPT3 [Tütüncü et al., 2003]) still use sophisticated versions of the algorithm for its polynomial-time guarantees

and robustness. The primal-dual interior point algorithm is based on the strict feasibility of the primal (SDP) and dual (SDPD) problems.

Assumption 1.5.7. *There exists a primal-dual pair* $(X, (\mu, S))$ *of (SDP) and (SDPD) such that*

$$X \succ 0, \;\; \langle A_i, X\rangle = b_i, \; i = 1, \dots, m, \;\; C - \sum_{i=1}^{m} \mu_i A_i \succ 0$$

We also assume that the linear equations $\langle A_i, X\rangle = b_i, \; i = 1, \dots, m$ are linearly independent.

Starting from a strictly feasible point, the algorithm follows a central path scheme. To remain in the interior of the feasible set along the iterations, the objective function is augmented with a so-called barrier term. Self-concordant barriers are a subclass of self-concordant functions. For a nonempty bounded closed convex set, a self-concordant barrier is a self-concordant function that varies slowly in the interior of the set and quickly tends to infinity as we approach the boundaries. For semidefinite programming, a self-concordant barrier for the cone $\mathcal{S}_+^n$ is given by [Nesterov et al., 2018]

$$F(X) = -ln \det(X) = -\sum_{i=1}^{n} ln\lambda_i$$

with $\{\lambda_i \mid i = 1, \dots, n\}$ the set of eigenvalues of X. Then, we can introduce the barrier problem parameterized by a positive barrier parameter

$$\begin{array}{lll} \min & f(X,t) = \langle C, X\rangle - tF(X) & \\ \text{s.t} & \langle A_i, X\rangle = b_i & i = 1, \dots, m \\ & X \succ 0. & \end{array} \qquad \text{(SDPB(t))}$$

Since the objective function is strictly convex there exists a unique solution X_t^*. Knowing that the derivative for the barrier function is $\nabla F(X) = -X^{-1}$, we can derive the Karush-Kuhn-Tucker (KKT) conditions for (SDPB(t)):

$$\begin{cases} \langle A_i, X\rangle = b_i & i = 1, \dots, m \\ X \succ 0 & \\ C - tX^{-1} = \sum_{i=1}^{m} \mu_i A_i. & \end{cases}$$

We then define $S = tX^{-1}$ or equivalently $SX = tI_n$. The system of equations becomes

$$\begin{cases} \langle A_i, X\rangle = b_i & i = 1, \dots, m \\ X \succ 0 & \\ C - \sum_{i=1}^{m} \mu_i A_i = S & S \succ 0 \\ SX - tI_n = 0. & \end{cases} \qquad \text{(KKT)}$$

Any feasible solution X and (μ, S) to (KKT) is a feasible solution for the primal problem (SDPB(t)) and its dual. Moreover, under Assumption 1.5.7 and constraint qualification, the solution is unique. For a given barrier parameter $t > 0$, the third and fourth equations give information on the current duality gap

$$\langle C, X\rangle - \mu^\top b = \langle S, X\rangle = nt.$$

We define

$$X^*(t) = \underset{X \in \mathrm{dom} f}{\arg\min}\, f(X, t) \tag{1.16}$$

This trajectory is called the central path associated with the optimization problem. Let X^* be the solution of (SDP), it is known that $X^*(t) \to X^*$ as $t \to 0$. The idea is then to solve a sequence of problems with $t \to 0$ while staying close to this trajectory. Starting from a feasible primal-dual point $v = (X, \mu, S)$, Newton's method is applied to $f(., t)$ in order to find a promising search direction. Multiple methods were introduced to efficiently compute this search direction see [Todd, 1999], however, we restrict our study to a straightforward strategy. Let us assume that $\bar{X}$ is a feasible solution to (SDPB(t)), the second order approximation of $f(., t)$ at $\bar{X}$ is given by:

$$f(X, t) = f(\bar{X}, t) + \left\langle C - \bar{X}^{-1}, X - \bar{X}\right\rangle + \frac{1}{2} t \left\langle \bar{X}^{-1}(X - \bar{X}), \bar{X}^{-1}(X - \bar{X})\right\rangle.$$

If we define $\Delta = X - \bar{X}$, a descent direction can by computed as the solution of the following convex optimization problem

$$\begin{aligned} \min_{\Delta} \quad & \left\langle C - \bar{X}^{-1}, \Delta\right\rangle + \frac{1}{2} t \left\langle \bar{X}^{-1}\Delta\bar{X}^{-1}, \Delta\right\rangle \\ \text{s.t} \quad & \langle A_i, \Delta\rangle = 0 \qquad\qquad i = 1, \dots, m. \end{aligned} \tag{1.17}$$

Since it is convex, KKT conditions are necessary and sufficient and a solution to (1.17) can be found using the following system of linear equations

$$\begin{aligned} C - \bar{X}^{-1} + t\bar{X}^{-1}\Delta\bar{X}^{-1} &= \sum_{i=1}^{m} \mu_i A_i \\ \langle A_i, \Delta\rangle &= 0, \qquad i = 1, \dots, m. \end{aligned} \tag{1.18}$$

With variables $\Delta \in \mathbb{R}^{n\times n}$ and $\mu \in \mathbb{R}^m$. We get

$$\begin{aligned} \Delta &= \frac{1}{t}\bar{X}\left(\sum_{i=1}^{m} \mu_i A_i + \bar{X}^{-1} - C\right)\bar{X} \\ \sum_{j=1}^{m} \mu_i \langle A_i, XA_jX\rangle &= \langle A_i, X(C - \bar{X}^{-1})X\rangle, \qquad i = 1, \dots, m \end{aligned} \tag{1.19}$$

The complexity of a Newton step and thus the efficiency of the overall method depends on the number of operations needed to solve the linear system (1.19).

In [Nesterov et al., 2018] the authors showed that it does not exceed $O(n^2(m+n)m)$ operations. This estimation can be reduced if the constraint matrices A_i have some structural properties. To solve the second equation, one has to compute and store the matrices $XA_jX,\ j = 1, \ldots, m$. While the matrices A_j are generally sparse, the product can become dense and it leads to memory issues for moderately to highly constrained problems. Finally, for a given parameter t, the Newton step gives a new point closer to the central path. The algorithm then iteratively reduces the value of t and repeats the procedure described above.

Interior point methods quickly show their limitations for solving constrained large-scale problems. Indeed if the number of constraints is of order $O(n^2)$, the overall time complexity for solving the SDP is $O(n^6)$. As a result, they have shown little practical interest in solving discrete problem relaxations. However, they continue to serve as a baseline for their robustness and ability to produce high-quality solutions. To overcome those issues, alternative approaches were proposed to solve SDPs. Researchers began to explore algorithms based on first-order information. Although they offer theoretically weaker guarantees, they are better suited to handle sparse structures of the data. For example, the state of the art solvers [Rendl et al., 2010, Gusmeroli et al., 2022], which solve SDP relaxations for MaxCut within branch-and-cut, use the so-called bundle method. It is an extension of the sub-gradient method for non-smooth convex problems. On first glance, using interior point methods is fine because relaxations of discrete problems are often sparse and highly structured. But these solvers introduce many cuts that break these properties, so the interior point method is no longer able to handle the relaxation. Therefore, additional cuts are dualised and a call to an interior point solver is only used for a simple SDP with diagonal constraints.

We now present two first-order methods that were used in the literature for solving different MAP SDP relaxations.

Augmented Lagrangian The first one is based on an augmented Lagrangian technique. The simple idea behind augmented Lagrangian methods is to add a penalty term to the Lagrangian in order to smooth the dual function. Optimization then becomes simpler and robust methods can be applied to optimize the corresponding dual function. In [Wang et al., 2016], the authors derive an SDP relaxation of the MAP problem for which the positive semidefinite variable is constrained to have a fixed trace, *i.e*,

$$\begin{aligned} \min\ & \langle C, X\rangle & & \\ \text{s.t}\ & \langle A_i, X\rangle = b_i & & i = 1, \ldots, k \\ & \langle A_i, X\rangle = b_j & & j = k+1, \ldots, m \\ & \mathrm{Tr}(X) = \eta & & \\ & X \succeq 0, & & \end{aligned} \tag{1.20}$$

with C the symmetric matrix containing the binary and unary potentials of a given GM and some $\eta \in \mathbb{N}$. The augmenting term is based on a theoretical result over the

following set:

$$\Omega(\eta) := \{X \in \mathcal{S}_+^n \mid \mathrm{Tr}(X) = \eta\}.$$

It is known [Wang et al., 2013] that for $X \in \Omega(\eta)$, $\|X\|_F \leqslant \eta$, and equality holds if and only if rank(X) = 1. This constraint is added as a penalty term in the objective function of the MAP SDP relaxation

$$\begin{aligned} \min \ & \langle C, X\rangle + \sigma(\|X\|_F^2 - \eta^2) & \\ \text{s.t} \ & \langle A_i, X\rangle = b_i & i = 1, \dots, k \\ & \langle A_i, X\rangle = b_j & i = k+1, \dots, m \\ & X \succeq 0 & \end{aligned} \tag{1.21}$$

with a penalty parameter $\sigma > 0$. To get an efficient bounding procedure for the MAP problem, [Wang et al., 2016] propose to solve the dual problem of (1.21) which does not involve an SDP but only a convex optimization problem in $\mathbb{R}^m$. The dual function is continuously differentiable but not necessarily twice differentiable so they use a quasi-Newton method called L-BFGS-B [Zhu et al., 1997] to maximize it. Note that at some point, a projection into the SDP cone is involved, and thus the computation of a set of positive eigenvalues. This operation is computationally intensive and generally costs $O(n^3)$.

Alternating Direction Method of Multipliers The Alternating Direction Method of Multipliers (ADMM) works on convex problems with decomposable objective functions. First we give a generic presentation of the method and then we present some recent extensions to SDPs. For a broader study of the algorithm, see [Boyd et al., 2011]. The aim of ADMM is to solve problems of the form

$$\begin{aligned} \min \ & f(x) + g(z) \\ \text{s.t} \ & Ax + Bz = c \end{aligned} \tag{1.22}$$

with f and g two convex functions. Note that any convex optimization problem

$$\begin{aligned} \min \ & f(x) \\ \text{s.t} \ & x \in \mathcal{C} \end{aligned} \tag{1.23}$$

can be cast into the form (1.22). Indeed if we consider g to be the indicator function for the constraint set C, (1.23) is equivalent to

$$\begin{aligned} \min \ & f(x) + g(z) \\ \text{s.t} \ & x - z = 0 \end{aligned} \tag{1.24}$$

We then introduce a Lagrangian that penalizes the linear constraint between the two variables. The augmented Lagrangian is given by

$$L_\rho(x, z, y) = f(x) + g(z) + y^\top(Ax + Bz - c) + \frac{\rho}{2}\|Ax + Bz - c\|^2. \tag{1.25}$$

The algorithm is based on a 2-blocks optimization of L_ρ and an update of the dual variable. The primal and dual variables are sequentially updated while the others are kept fixed:

$$\begin{aligned} x^{k+1} &:= \arg\min_x L_\rho(x, z^k, y^k) \\ z^{k+1} &:= \arg\min_z L_\rho(x^{k+1}, z, y^k) \\ y^{k+1} &:= y^k + \rho(Ax^{k+1} + Bz^{k+1} - c). \end{aligned} \tag{1.26}$$

In [Wen et al., 2010], the authors proposed a variation of ADMM applied to the augmented Lagrangian of the dual SDP (SDPD)

$$L_\rho(X, \mu, S) = \mu^\top b + \left\langle X, \sum_{i=1}^m \mu_i A_i + S - C \right\rangle + \frac{1}{2\rho} \left\| \sum_{i=1}^m \mu_i A_i + S - C \right\|_F^2 \tag{1.27}$$

The optimization steps now read

$$\mu^{k+1} := \arg\min_\mu L_\rho(X^k, \mu, S^k) \tag{1.28}$$

$$S^{k+1} := \arg\min_{S \succeq 0} L_\rho(X^k, \mu^{k+1}, S) \tag{1.29}$$

$$X^{k+1} := X^k + \frac{\sum_{i=1}^m \mu_i^{k+1} A_i + S^{k+1} - C}{\rho}. \tag{1.30}$$

The same method was later applied in [Huang et al., 2014] for the MAP problem. Note that in the two papers, the authors are considering extensions of ADMM with more than 2 blocks of optimization (1.26). It was recently shown [Chen et al., 2016] that ADMM extended to multi-block optimization problems is not necessarily convergent. These multi-block problems often arise when dealing with discrete problem relaxations. Indeed one has to introduce different kinds of linear constraints to make the relaxation tighter. However, it is possible to recover convergence for problems with "orthogonal" constraints. It means that the constraint operators apply on non-overlapping entries of the variables. In this particular case, the algorithm can be shown to be equivalent to the original 2-blocks ADMM.

A bottleneck of this method when applied to SDPs is the optimization step (1.29). It is equivalent to the following convex optimization problem

$$\min_{S \in \mathcal{S}_+^n} \|S - U\|_F^2 \tag{1.31}$$

for a particular symmetric matrix U depending on the fixed variables. It corresponds to the projection of U onto the semidefinite cone $\mathcal{S}_+^n$. Using theorem 1.5.2, there exists a unique solution that can be expressed using the spectral decomposition U. Computing the full spectrum of a symmetric matrix in $\mathbb{R}^{n \times n}$ with direct methods has complexity $O(n^3)$. Both papers [Wen et al., 2010, Huang et al., 2014] try to overcome this issue by computing partial eigendecomposition with iterative methods such as the Lanczos algorithm [Cullum and Willoughby, 2002]. More recently, people started to look at a low-rank decomposition of S to get rid of the cone constraint

in (1.29) [Cerulli et al., 2021]. This decomposition was also applied to the primal variable X in [Zhao et al., 2022, Chen and Goulart, 2023]. This idea of decomposing a positive semidefinite matrix into a product of low-rank matrices is not new, we discuss it in the next section.

1.5.6 Low-rank methods

We now give an overview of the low-rank methods used to solve SDPs. We have already seen that interior point solvers quickly reach their limits for large constrained problems. In early 2000, Burer and Monteiro [2003, 2005] introduced a new way to solve SDPs through a change of variable. In the light of definition 18, a positive semidefinite matrix X with rank r can be factored as the product $X = VV^\top$ with $V \in \mathbb{R}^{n\times r}$. Applying this change of variable to the standard SDP (SDP) we get the following optimization problem

$$\begin{aligned} \min \ & \langle C, VV^\top \rangle \\ \text{s.t} \ & \langle A_i, VV^\top \rangle = b_i \quad i = 1, \dots, m \\ & V \in \mathbb{R}^{n\times r} \end{aligned} \tag{BM-SDP}$$

At this point, we can make two observations on (BM-SDP). On the one hand, the cone constraint was removed and the number of variables has changed from n^2 to nr. On the other hand the problem is no longer convex and thus convex optimization methods cannot be directly applied anymore. An important question is the choice of the rank for the factorization. Barvinok [1995] and Pataki [1998] independently proved the following theorem:

Theorem 1.5.8. *There exists an optimum X^* of (SDP) with rank r such that $\frac{r(r+1)}{2} \leqslant m$, with m the number of constraints.*

A natural choice for the rank of V is $r = \lceil \sqrt{2(m+1)} \rceil$. Until recently, the question of whether the optimal solution of (SDP) could be recovered from the non-convex problem (BM-SDP) was open. Using Riemannian analysis, the paper [Boumal et al., 2020] gives deterministic guarantees on the rank r and the constraints of (SDP) so (BM-SDP) does not have any spurious second-order critical points. It was also shown that the bound $r \geqslant \sqrt{2(m+1)}$ is tight [Waldspurger and Waters, 2020]. The article [Frostig et al., 2014] was one of the first to exploit low-rank relaxations for MAP-MRF inference. They restrict their study to pairwise graphical models with Boolean variables. Using the same transformations as in Subsection 1.5.2, the MAP inference task in this case is equivalent to $\min_{x\in\{0,1\}^n} x^\top Q x$. After the change of variable $y = 2x - 1_n$ the problem is equivalent to $\min_{y\in\{-1,1\}^n} y^\top Q y$. Note that this last problem is similar to MaxCut for which we already have the standard SDP relaxation

$$\begin{aligned} \min \ & \langle Q, X \rangle \\ & \operatorname{diag}(X) = 1_n \\ & X \succeq 0. \end{aligned} \tag{1.32}$$

Instead of solving (1.32) they propose to solve the low-rank relaxation

$$\begin{aligned} \min_{V \in \mathbb{R}^{n \times r}} \quad & \langle Q, VV^\top \rangle \\ & \|V_i\| = 1 \qquad i = 1, \dots, n. \end{aligned} \tag{1.33}$$

They efficiently solve this last problem by either a projected gradient descent or a simple coordinate descent method. The coordinate descent method consists in optimizing the objective function regarding the row vector V_i while the others are being fixed. This method was later studied in [Wang et al., 2017] where the authors provide global convergence results under certain assumptions. The paper [Hu and Carlone, 2019] deals with more general graphical models, *i.e.*, each variable can take K possible states. However, the binary potentials considered are the following

$$\theta_{ij}(x_i, x_j) = \begin{cases} 0 & \text{if } x_i = x_j \\ \delta_{ij} & \text{otherwise.} \end{cases}$$

The resulting graphical model is known as the *Potts Model* [Potts, 1952]. The case of the k-class *Potts model* with Boolean variables was studied in [Pabbaraju et al., 2020] using the mixing method introduced in [Wang et al., 2017]. Since variables are non-Boolean, the MAP estimation problem is equivalent to (1.10). They derive the two following SDP relaxations

$$\begin{aligned} \min_{X \in \mathbb{R}^{(d+1)\times(d+1)}} \quad & \langle Q, X \rangle \\ \text{s.t} \quad & \langle U_i, X \rangle = 2 - K, \quad \forall i = 1, \dots, n \\ & \operatorname{diag}(X) = 1_{d+1} \\ & X \succeq 0 \end{aligned} \tag{DARS}$$

with matrix $U_i = \begin{pmatrix} 0^{d\times d} & A_i \\ A_i^\top & 0 \end{pmatrix}$ and A_i as defined in (1.9).

$$\begin{aligned} \min_{Z \in \mathbb{R}^{(d+1)\times K}} \quad & \langle Q, Z \rangle \\ \text{s.t} \quad & \operatorname{diag}([Z]_{tl}) = 1_{d+1} \\ & [Z]_{br} = I_K \\ & Z \succeq 0. \end{aligned} \tag{FUSES}$$

Where $[Z]_{tl}$ is the $(d+1) \times (d+1)$ top left corner of Z and $[Z]_{br}$ is the $K \times K$ bottom right corner of Z. Both relaxations are solved using Burer and Monteiro factorization. For (DARS) the method is a mix of dual ascent and Riemannian Staircase presented in [Boumal et al., 2016]. For (FUSES), since the additional linear constraints on Z are no longer needed, the Reimannian Staircase can be directly applied.

1.6 Integer solutions with rounding

So far, we have analyzed continuous relaxations for computing lower bounds. As a reminder, these bounds play a crucial role in the branch-and-bound algorithm. In this section, we focus on heuristics to compute feasible integer solutions for the original problem.

Approximation ratio for MaxCut Let us consider a weighted graph $G = \langle V, E \rangle$ with $|V| = n$ vertices. We have shown that the basic low-rank SDP relaxation for MaxCut is given by:

$$\begin{aligned} \min \ & \langle Q, X \rangle \\ & \operatorname{diag}(X) = 1_n \\ & X \succeq 0. \end{aligned} \tag{1.34}$$

In 1995, Goemans and Williamson [Goemans and Williamson, 1995] presented their celebrated 0.87856 approximation ratio for MaxCut on graphs with non-negative weights. They have shown that by solving the problem (1.34) we can build an integer solution with expected value 0.87856 times the optimum. The method can be quickly explained as follows:

- let $X^* \in \mathbb{R}^{n \times n}$ be a solution of (1.34). Using a partial Cholesky factorization, compute $X^* = {V^*}^\top V^*$ for some full rank matrix $V^* \in \mathbb{R}^{m \times n}$.
- Draw a random vector r uniformly distributed on the unit sphere in $\mathbb{R}^m$.
- Build $x \in \{-1, 1\}^n$ with $x_i = \operatorname{sgn}(r^\top V_i^*),\ i = 1, \ldots, n$.

After using the rounding heuristic, one can apply a local search to improve the solution x. A usual method consists of sequentially flipping the signs of x until a better solution cannot be found. The pseudo-code for this heuristic is presented in algorithm 2. For our SDP relaxation, we extend this scheme for non-Boolean variables, the method is further explored in Chapter 2. While the theoretical approximation ratio does not apply in our case, we found out that this simple heuristic is still able to find good quality solutions, sometimes better than specialized methods. We provide some empirical results in Chapter 2.

Algorithm 2 1-OPT search

```
x ∈ {−1, 1}^n, changed := true, f := evaluate(x)
while changed do
  changed = false
  for i = 1, ..., n do
    x_i ← −x_i
    if evaluate(x) < f then
      f = evaluate(x), changed = true
    else
      x_i ← −x_i
    end if
  end for
end while
return x, f
```

Chapter 2

Coordinate vs block coordinate descent methods

Contents

2.1 LR-LAS 50
2.1.1 Gangster constraints 53
2.1.2 Dual feasible points 56
2.2 LR-BCD 58
2.2.1 From sparsity to block optimization 58
2.2.2 Optimization on the unit-sphere through trigonometry 61
2.2.3 Computational complexity 69
2.2.4 Strong duality and optimality 69
2.2.5 Descent property 70
2.2.6 Producing an integer primal solution 74
2.3 Experiments 74
2.3.1 Description of solvers 74
2.3.2 Random instances 75
2.3.3 Sparse problems 80
2.3.4 Real world problems 81
2.4 Discussion 84
2.4.1 Update rule for the cost matrix 85
2.4.2 On proving convergence of LR-BCD 87
2.4.3 Strengthening the LR-BCD formulation 89

In this chapter, we present two methods for the 0/1 constrained quadratic reformulation of the MAP problem on pairwise discrete graphical models. The first method is based on the idea that one can reformulate a 0/1 quadratic problem with linear constraints as a pure MaxCut problem using constraint penalization. We then present a new dedicated method that avoids the introduction of large penalty coefficients to model the problem. Both discrete optimization problems are relaxed by means of low-rank semidefinite programs. Compared to what has been done in the literature, both low-rank approaches can be applied to discrete graphical models with an arbitrary number of states and arbitrary binary potentials. The goal underlying those two methods is to compute tight bounds on the discrete optimization problem using SDP relaxations while getting closer to the computational efficiency of linear methods. We aim to bridge the gap between the two paradigms to tackle large-scale problems that are out of reach for current solvers. Both methods are then compared to state-of-the-art solvers on a pool of crafted and real-world problems. This chapter is organized in four parts:

- LR-LAS 2.1: we present our first low-rank SDP method for the MAP problem on discrete MRFs. The binary quadratic formulation (1.10) is reduced to a pure MaxCut problem using Lasserre's penalization result [Lasserre, 2016]. Then we solve the basic SDP relaxation of the MaxCut problem with the mixing method [Wang et al., 2017].
- LR-BCD 2.2: we introduce a new SDP formulation as well as a dedicated block-coordinate descent algorithm which takes advantage of the structural properties of our problem. We derive some theoretical properties of this new low-rank SDP algorithm.
- Experiments 2.3: our two methods are tested against state-of-the-art message passing and LP solvers on a range of random and real-world instances. Some parameters are discussed, such has the rank of the low-rank relaxation.
- Discussion 2.4: finally, we discuss some features and additional theoretical properties of LR-BCD.

2.1 LR-LAS

Approximation techniques for MaxCut based on semidefinite programming have been extensively studied since the seminal work of Goemans and Williamson [1995]. Compared to linear programming approaches they often provide much tighter bounds on a variety of combinatorial optimization problems. However, SDP has found limited application in MAP/WCSP. The main reason lies in the fact that the Goemans and Williamson's SDP relaxation of MaxCut, a specialization of MAP with binary variables, requires cubic complexity for usual interior point solvers. The algorithm doesn't scale well and becomes impractical except on small very hard problems. For this reason, approximate LP methods are usually preferred because they achieve a

better trade off between tightness and computational cost.

The suitability of SDP for MAP and other applications has improved with the non-convex low-rank Burer-Monteiro approach. They exploit the results of Barvinok [1995] and Pataki [1998] who showed that the rank of solutions of SDP problems is bounded by $O(\sqrt{m})$ with m the number of constraints. This approach uses a low-rank factorization $X = VV^\top$ of the semidefinite matrix X in the SDP relaxation of these combinatorial problems. It has since been observed that, in practice, the rank of the solution is often lower, with some software using a constant $O(1)$ rank.

Let us write the constrained quadratic 0/1 formulation of the MAP/MRF problem. We consider a pairwise GM $\mathcal{M} = \langle X, \Phi \rangle$ with associated graph (X, E). Each variable $x_i \in X$ can take d_i possible states, we note $d = \sum_{i=1}^{n} d_i$ the total number of states and $s_1 = 1, s_i = s_{i-1} + d_{i-1}, i = 2, \dots, n$ the indices of the domains in the stacked vector. Using the 1-hot encoding introduced previously in Subsection 1.5.2, the MAP problem is equivalent to the following quadratic optimization problem:

$$\begin{aligned} \min_{b \in \{0,1\}^d} \quad & b^\top Q b + c^\top b \\ \text{s.t} \quad & Ab = 1_n \end{aligned} \tag{2.1}$$

Where $A \in \mathbb{R}^{n \times d}$ encodes the exactly-one constraint for each GM variable:

$$A_i^\top = [0_{d_1}^\top \ \cdots \ 0_{d_{i-1}}^\top \ 1_{d_i}^\top \ 0_{d_{i+1}}^\top \ \cdots \ 0_{d_n}^\top], \ i = 1, \dots, n$$

To obtain a pure MaxCut problem, we use the -1/1 change of variable $y = 2b - 1_d$. We introduce new vectors and matrices $R = Q/4$, $e = (c + Q1_d)/2$, $F = A/2$ and $u = 1_n - A1_d/2$. Problem (2.1) is now equivalent to:

$$\begin{aligned} \min_{y \in \{-1,1\}^d} \quad & y^\top R y + e^\top y \\ \text{s.t} \quad & Fy = u \end{aligned} \tag{2.2}$$

Here, the initial exactly-one constraint $\sum_{i=s_k}^{s_k+d_k-1} b_i = 1$ becomes $2F_i^\top y = 2 - d_i$ with $F_i \in \mathbb{R}^d, 2F_i^\top = [0_{d_1}^\top \ \cdots \ 0_{d_{i-1}}^\top \ 1_{d_i}^\top \ 0_{d_{i+1}}^\top \ \cdots \ 0_{d_n}^\top]$. This change of variables creates a constant term in the objective function. Indeed for the quadratic part, we have

$$y^\top R y = \frac{1}{4}(2y - 1_d)^\top Q (2y - 1_d) = y^\top Q y - y^\top Q 1_d + 1_d^\top Q 1_d.$$

This constant term can be safely ignored for optimization purposes. In this formulation, the matrix R has the same exact sparsity as Q.

In order to get an unconstrained quadratic program, we must remove the linear constraints in (2.2). A usual approach to do this is to penalize the linear constraints with a penalty ρ that must be appropriately chosen [Lasserre, 2016]. This reduces the problem to a pure MaxCut problem on which the Burer-Monteiro approach can

be directly applied and solved using efficient row-by-row updates [Wang et al., 2017]. The problem becomes:

$$\min_{y\in\{-1,1\}^d} \; y^\top R y + y^\top e + (2\rho+1)\|Fy-u\|^2 \tag{2.3}$$

As soon as $\rho \geqslant \max\{|y^\top R y + y^\top e| \;:\; y \in \{-1,1\}^d\}$, the solution of (2.3) and (2.2) are the same [Lasserre, 2016]. One can note at this point that if any potential function in the GM at hand contains potentials of large amplitude, ρ will need to take a large value to ensure the above property.

We next homogenize the problem by converting linear terms to quadratic terms. We introduce the extended vector $z^\top = [y^\top \; 1] \in \{-1,1\}^{d+1}$ and reformulate (2.3) in terms of z. Then, (2.3) equivalently asks to minimize $z^\top B z$ where $z \in \{-1,1\}^{d+1}$ and B is the symmetric matrix:

$$B \coloneqq \begin{pmatrix} (2\rho+1)F^\top F + R & \frac{1}{2}(e^\top - 2(2\rho+1)u^\top F)^\top \\ \frac{1}{2}(e^\top - 2(2\rho+1)u^\top F) & (2\rho+1)u^\top u \end{pmatrix} \tag{2.4}$$

The usual SDP relaxation of the MaxCut problem can then be written using the new rank 1 matrix variable $X = zz^\top \in \mathbb{R}^{(d+1)\times(d+1)}$. Dropping the rank-1 constraint, the SDP relaxation is:

$$\min_X \{\langle B, X\rangle : X \succeq 0;\; X_{ii} = 1,\; i = 1,\dots,d+1\} \tag{2.5}$$

Burer and Monteiro [2003] introduced the idea of using a low-rank factorization of X to solve the SDP (2.5). This factorization was motivated by a proof by Barvinok [1995] and Pataki [1998] of the following theorem:

Theorem 2.1.1. *[Barvinok, 1995, Pataki, 1998] There exists an optimum X^* of (2.5) with rank r such that $\frac{r(r+1)}{2} \leqslant m$, with m the number of constraints.*

Every positive semi-definite matrix of rank r can be factorized as a product of two rank-r matrices: $X = VV^\top$, $V \in \mathbb{R}^{(d+1)\times r}$. Then, (2.5) becomes:

$$\min_{V\in\mathbb{R}^{(d+1)\times r}} \langle B, VV^\top\rangle \;\text{ s.t }\; \|V_i\| = 1,\; i = 1,\dots,d+1 \tag{2.6}$$

With $V_i \in \mathbb{R}^r$ the row vector i of V. The norm constraints on the rows are inherited from the diagonal constraints of the original problem. If we consider the factorization $X = VV^\top$, the entries of matrix X are $X_{ij} = V_i^\top V_j$. Thus $X_{ii} = 1$ implies that $V_i^\top V_i = \|V_i\|^2 = 1$, hence the norm constraints. The low-rank reformulation of the problem reduces the number of variables from $(d+1)^2$ to $r(d+1)$ and the semi-definite constraint $X \succeq 0$ now becomes implicit as $VV^\top$ is always positive semi-definite. Several approaches exploit this idea. To solve (2.6) we use the mixing method [Wang et al., 2017]. It consists in efficient $O(r(d+1))$ cyclic updates of the row vectors V_i, all other row vectors being fixed:

$$V_i = -\frac{g_i}{\|g_i\|}, \quad g_i = \sum_{j=1}^{d+1} B_{ij} V_j \;\; \forall i = 1,\dots,d+1.$$

Algorithm 3 Mixing method

Initialize V_i randomly on the unit sphere
while not converged yet **do**
  for $i = 1, \ldots, d+1$ **do**
    $g_i \coloneqq VC_i$
    $V_i \coloneqq -\frac{g_i}{\|g_i\|}$
  end for
end while

The pseudo-code for the mixing method is given in Algorithm 3.

The mixing method is known to recover the optimum of the convex SDP (2.5) with random initialization as long as the rank of V satisfies the bound provided in (2.1.1) *i.e.*, $r > \sqrt{2(d+1)}$. One issue with using the mixing method in the presence of dualized constraints generated by Lasserre's approach is that the row-by-row updates are sensitive to the magnitude of the penalty parameter ρ. As we observed previously, ρ may be large if the input GM has large terms. The large value of ρ will create large coefficients in the objective matrix B. If we consider a particular

$$g_i = \sum_{j=1}^{d+1} B_{ij} V_j$$

the sum will be dominated by a few terms with high magnitude, see Figure 2.1. The update will promote descent in the direction of the rows corresponding to these terms, possibly slowing down convergence.

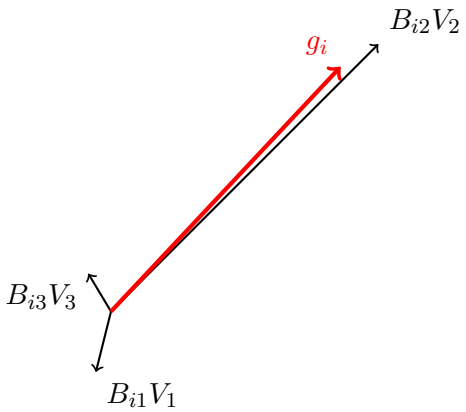


Figure 2.1: Predominant direction: $B_{i2} \gg B_{i1}$ and $B_{i2} \gg B_{i3}$.

2.1.1 Gangster constraints

A feasible solution of (2.1) always satisfies the so-called *gangster* constraint [Zhao et al., 1998]. This constraint was introduced to tighten numerous discrete optimization relaxations. It specifies that the off-diagonal entries of the matrix $b_k b_k^\top$ must

be all zeros. If we consider the Boolean vector b_k that corresponds to the variable $x_k \in X$, then $b_k^i b_k^j = 0,\ \forall i \neq j$, since at least one of the two terms is equal to zero. It is interesting to notice that if the solution to (2.6) nullifies the quadratic penalty term induced by the Lasserre penalization, it will also satisfy this gangster constraint.

Lemma 2.1.2. *If* $\mathrm{Tr}(PVV^\top) = 0$, *where* $P = \begin{pmatrix} F^\top F & -F^\top u \\ -u^\top F & u^\top u \end{pmatrix}$, *then* V *satisfies the gangster constraint.*

Before going through the proof, we describe how the gangster constraint can be formulated in our SDP. First, if we consider the Boolean vector b_k that corresponds to the variable $x_k \in X$, the goal of the gangster constraint is to ensure that the off-diagonal entries of the matrix $b_k b_k^\top$ are all zeros. To ensure that b_k will satisfy the gangster constraint, we can use the following matrix $H_k \in \mathbb{R}^{d_k \times d_k}$

$$H_k = \begin{pmatrix} 0 & 1 & \cdots & 1 \\ 1 & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & 1 \\ 1 & \cdots & 1 & 0 \end{pmatrix}$$

Thus, for each $k = 1, \ldots, n$, we must have, $b_k^\top H_k b_k = 0$ or equivalently $\mathrm{Tr}(H_k b_k b_k^\top) = 0$. If we think in terms of the concatenated vector $b \in \mathbb{R}^d$ we must have $\mathrm{Tr}(Hbb^\top) = 0$ with

$$H = \begin{pmatrix} \boxed{H_1} & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \boxed{H_n} \end{pmatrix},\ H \in \mathbb{R}^{d \times d}$$

Now, doing the -1/1 change of variable $y = 2b - 1_d$,

$$b^\top H b = 0 \iff y^\top G y + y^\top q + r = 0$$

with $G = \frac{1}{4}H,\ q = \frac{1}{2}H1_d,\ r = \frac{1}{4}1_d^\top H 1_d$. Using the augmented vector $z^\top = [y^\top\ 1] \in \{-1, 1\}^{d+1}$:

$$\mathrm{Tr}(Rzz^\top) = 0 \text{ with } R = \begin{pmatrix} G & \frac{1}{2}q \\ \frac{1}{2}q^\top & r \end{pmatrix} \in \mathbb{R}^{(d+1)\times(d+1)}$$

Then we can use the usual relaxation and the usual low-rank factorization:

$$X = zz^\top \in \mathbb{R}^{(d+1)\times(d+1)},\ \mathrm{Tr}(RX) = 0$$

$$X = VV^\top,\ \mathrm{Tr}(RVV^\top) = 0$$

Proof of Lemma 2.1.2. We want to prove the following implication which characterizes the solutions of the low-rank problem (2.5):

$\mathrm{Tr}(PVV^\top) = 0 \Rightarrow V$ satisfies the gangster constraint, with $P = \begin{pmatrix} F^\top F & -F^\top u \\ -u^\top F & u^\top u \end{pmatrix} \succeq 0$.

We can factorize the matrix P, $P = S^\top S$ with $S = \begin{pmatrix} F & -u \end{pmatrix} \in \mathbb{R}^{n\times(d+1)}$. Thus, $tr(PVV^\top) = 0$ becomes:

$$\begin{aligned} \mathrm{Tr}(S^\top SVV^\top) = 0 &\iff \mathrm{Tr}(V^\top S^\top SV) = 0 \\ \mathrm{Tr}((SV)^\top SV) &= 0 \\ \|SV\|_F^2 &= 0 \\ SV &= 0 \end{aligned}$$

One can see that this last equality is equivalent to the exactly-one constraint as in the BCD formulation. Now we will prove that

$$\mathrm{Tr}(PVV^\top) = 0 \Rightarrow \mathrm{Tr}(RVV^\top) = 0$$

with $R = \begin{pmatrix} G & \frac{1}{2}q \\ \frac{1}{2}q^\top & r \end{pmatrix} \in \mathbb{R}^{(d+1)\times(d+1)}$ the gangster constraint matrix. We will rewrite $P = M + R$ with M a matrix we will describe later. The objective is to show that:

$$\mathrm{Tr}(PVV^\top) = 0 \Rightarrow \mathrm{Tr}(MVV^\top) = 0$$

Thus, given the result above:

$$\begin{aligned} \mathrm{Tr}(PVV^\top) = 0 &\Rightarrow \mathrm{Tr}((M+R)VV^\top) = 0 \\ \mathrm{Tr}(MVV^\top) + \mathrm{Tr}(RVV^\top) &= 0 \\ \mathrm{Tr}(RVV^\top) &= 0 \end{aligned}$$

First,

$$M = P - R = \begin{pmatrix} D & -F^\top u - \frac{1}{2}q \\ -u^\top F - \frac{1}{2}q^\top & u^\top u - r \end{pmatrix}$$

With $D = \frac{1}{4}\,\mathrm{Diag}(1_d) \in \mathbb{R}^{d\times d}$. For $i = 1, ..., d$

$$(-F^\top u - \frac{1}{2}q)_i = \frac{d_i}{4} - \frac{1}{2} - \frac{1}{4}(d_i - 1) = -\frac{1}{4}$$

and

$$\begin{aligned} u^\top u - r &= \sum_{i=1}^{n}(1 - \frac{d_i}{2})^2 - \frac{1}{4}\sum_{i=1}^{n} d_i(d_i - 1) \\ &= n - d + \frac{d}{4} = n - \frac{3}{4}d \end{aligned}$$

Next, $\mathrm{Tr}(MVV^\top) = \sum_i \sum_j M_{ij} V_i^\top V_j$ with $\begin{cases} M_{ij} = \frac{1}{4},\ i = j,\ i, j = 1, ..., d \\ M_{ij} = -\frac{1}{4},\ i = d+1,\ j = 1, ..., d \\ M_{ij} = -\frac{1}{4},\ j = d+1,\ i = 1, ..., d \\ M_{ij} = n - \frac{3}{4}d,\ i = j = d+1 \\ M_{ij} = 0 \text{ otherwise} \end{cases}$

Thus,

$$\mathrm{Tr}(MVV^\top) = \sum_{i=1}^{d} M_{ii}\|V_i\|^2 + 2\sum_{j=1}^{d} M_{d+1,j}V_j^\top V_{d+1} + n - \frac{3}{4}d$$

$$= \frac{1}{4}d + n - \frac{3}{4}d - \frac{1}{2}\sum_{j=1}^{d} V_{d+1}^\top V_j$$

We showed that $\mathrm{Tr}(PVV^\top) = 0 \Rightarrow V$ satisfies the exactly-one constraint. We can simplify the last term:

$$\sum_{j=1}^{d} V_{d+1}^\top V_j = \sum_{j=1}^{n}(2 - d_j)$$

Finally,

$$\mathrm{Tr}(MVV^\top) = \frac{1}{4}d + n - \frac{3}{4}d - \frac{1}{2}\sum_{j=1}^{n}(2 - d_j)$$

$$= \frac{1}{4}d + n - \frac{3}{4}d - n + \frac{1}{2}d = 0$$

We can now conclude that $\mathrm{Tr}(PVV^\top) = 0 \Rightarrow \mathrm{Tr}(MVV^\top) + \mathrm{Tr}(RVV^\top) = \mathrm{Tr}(RVV^\top) = 0$, so V satisfies the gangster constraint. □

2.1.2 Dual feasible points

Within a branch and bound scheme, for the bounding procedure we are interested in computing bounds with guarantees. Since the mixing method works on the primal problem, we cannot directly use the returned value as a safe bound. We now give some hints on how to compute a good dual feasible solution given information on the primal iterates. The proposed method is inexpensive and does not require dual SDP resolution. First we recall the primal and dual problems :

$$\underset{V\in\mathbb{R}^{(d+1)\times r}}{\text{minimize}}\ f(V) := \langle B, VV^\top\rangle,\ \text{s.t}\ ||V_i|| = 1 \tag{2.7}$$

$$\underset{\lambda\in\mathbb{R}^{d+1}}{\text{maximize}}\ D(\lambda) := -1_{d+1}^\top\lambda,\ \text{s.t}\ B + \mathrm{Diag}(\lambda) \succeq 0 \tag{2.8}$$

Let V^* be the solution for the primal problem (2.7) and let $\lambda^* \in \mathbb{R}^{d+1}$ be the vector

$$\lambda_i^* = \|V^{*\top}B_i\|,\ \ i = 1, \ldots, d+1,$$

with $B_i \in \mathbb{R}^{d+1}$ the i-th row vectors of B. In [Wang et al., 2017], the authors show that (V^*, λ^*) is a primal-dual feasible pair that closes the duality gap, *i.e.*, $\langle B, V^*V^{*\top}\rangle = -1_{d+1}^\top\lambda^*$.

We now introduce the function:

$$\begin{array}{rccc} h: & \mathbb{R}^{d+1\times r} & \longrightarrow & \mathbb{R}^r \\ & V & \longmapsto & \|V^\top B_i\|. \end{array} \tag{2.9}$$

The application $g(V) : V \longmapsto V^\top B_i$ is linear thus Lipschitz on the finite space $\mathbb{R}^{(d+1)\times r}$ and the 2-norm is also Lipschitz on $\mathbb{R}^r$. As a composition of two Lipschitz functions, h is Lipschitz and using the Cauchy-Schwarz inequality, one can derive the following inequality:

$$\left|\|V^\top B_i\| - \|V^{*\top} B_i\|\right| \leqslant \|V^\top B_i - V^{*\top} B_i\| \leqslant \|B_i\|\|V^\top - V^{*\top}\|_F. \tag{2.10}$$

At this point, if we compute a point V close enough to the optimum V^*, $\|V^{*\top} - V^\top\|_F \leqslant \frac{\varepsilon}{\|B_i\|_2}$, we have:

$$|\lambda_i^* - \lambda_i| = \left|\|V^\top B_i\| - \|V^{*\top} B_i\|\right| \leqslant \|V^\top B_i - V^{*\top} B_i\| \leqslant \|B_i\|\|V^\top - V^{*\top}\|_F \leqslant \varepsilon. \tag{2.11}$$

To conclude, if we are able to compute a primal iterate as close as we want from the primal solution, we can compute a point $\lambda \in \mathbb{R}^{d+1}$ as close as we want from the dual optimum λ^*.

The question that remains is whether the point $\lambda \in \mathbb{R}^{d+1}$ with

$$\lambda_i = \|V^\top B_i\|, \;\; i = 1, \ldots, d+1,$$

is dual feasible, that is, $B + \mathrm{Diag}(\lambda) \succeq 0$. We denote by

$$K \coloneqq \max \|V^\top B_i\|, \; i = 1, \ldots, d+1,$$

using the result in (2.11) for $V \in \mathbb{R}^{(d+1)\times r}$ such that $\|V^{*\top} - V^\top\|_F \leqslant \frac{\varepsilon}{K}$, we have

$$|\lambda_i^* - \lambda_i| \leqslant \varepsilon, \; i = 1, \ldots, d+1.$$

Now we can consider the new point $\lambda' = \lambda + \varepsilon 1_{d+1}$ which we rewrite:

$$\lambda' = \lambda - \lambda^* + \lambda^* + \varepsilon 1_{d+1}.$$

We have

$$B + \mathrm{Diag}(\lambda') = B + \mathrm{Diag}(\lambda^*) + \mathrm{Diag}(\lambda - \lambda^* + \varepsilon 1_{d+1}).$$

Since λ^* is dual feasible, we already know that $B + \mathrm{Diag}(\lambda^*) \succeq 0$. Next, since

$$|\lambda_i^* - \lambda_i| \leqslant \varepsilon, \; i = 1, \ldots, d+1$$
$$\lambda_i - \lambda_i^* + \varepsilon \geqslant 0, \; i = 1, \ldots, d+1.$$

Thus, $\mathrm{Diag}(\lambda - \lambda^* + \varepsilon 1_{d+1}) \succeq 0$ and we can conclude that $B + \mathrm{Diag}(\lambda') \succeq 0$. We found a dual feasible point λ' with a dual objective value $-1_{d+1}^\top \lambda'$ arbitrarily close to the dual optimum as we show next.

Bounding the dual value Thanks to the result above, we can bound the dual value for λ'. Indeed, since

$$-\varepsilon \leqslant \lambda_i - \lambda_i^* \leqslant \varepsilon \text{ for all } i = 1, ..., d+1$$

We have

$$0 \leqslant \lambda_i - \lambda_i^* + \varepsilon \leqslant 2\varepsilon$$
$$0 \geqslant -(\lambda_i - \lambda_i^* + \varepsilon) \geqslant -2\varepsilon$$
$$-1_{d+1}^\top(\lambda - \lambda^* + \varepsilon 1_{d+1}) \geqslant -2(d+1)\varepsilon.$$

Let d^* be the optimal value for the dual problem (2.8), we have

$$d^* \geqslant -1_{d+1}^\top \lambda' = -1_{d+1}^\top \lambda^* - 1_{d+1}^\top(\lambda - \lambda^* + \varepsilon 1_{d+1}) \geqslant d^* - 2(d+1)\varepsilon, \tag{2.12}$$

and we get the following bounds

$$d^* \geqslant -1_{d+1}^\top \lambda' \geqslant d^* - 2(d+1)\varepsilon. \tag{2.13}$$

With the right term converging to 0 when $\varepsilon \longrightarrow 0$.

2.2 LR-BCD

2.2.1 From sparsity to block optimization

For the MAP/MRF problem, the objective matrix in the 0/1 quadratic formulation enjoys some structural properties. The goal of this section is to show how we can use those sparsity patterns to extend the coordinate descent method to a block coordinate descent method. First, let us write again the constrained quadratic 0/1 formulation of the MAP/MRF problem. We consider a pairwise GM $M = \langle X, \Phi \rangle$ with n discrete variables and associated graph (X, E). Each variable $x_i \in X$ can take d_i possible states, we note $d = \sum_{i=1}^n d_i$ the total number of states and $s_1 = 1, s_i = s_{i-1} + d_{i-1}, i = 2, \ldots, n$ the indices of the domains in the stacked vector. Using the 1-hot encoding introduced previously, the MAP/MRF problem is equivalent to the following quadratic optimization problem:

$$\begin{aligned} \min_{b \in \{0,1\}^d} \quad & b^\top Q b + c^\top b \\ \text{s.t} \quad & Ab = 1_n \end{aligned} \tag{2.14}$$

The objective matrix Q stores the information for the binary potentials while vector c contains the stacked unary potentials. When two variables $(x_i, x_j) \in X^2$ are independent, *i.e.*, there exists no binary potential that applies to the two variables, then the corresponding block in the objective matrix Q is a zero-block of size $d_i \times d_j$. There is a relation between the original problem sparsity and the sparsity of the objective matrix. We can distinguish between two types of sparsity:

1. Potential sparsity: Whenever a potential contains a small amount of non-zero entries, the corresponding block in Q will be sparse. This can be the case for attractive or repulsive potentials used in Potts models. Namely, the potential θ_{ij} between variables x_i and x_j will take positive or negative values whenever $x_i = x_j$ and 0 otherwise. This results in a block matrix with only non-zero entries on the diagonal in Q.

2. Terms sparsity: If the GM M has only a small number of potentials then matrix Q will be sparse by blocks.

Such information is heavily used in modern SDP solvers. Indeed sparsity can lead to computational and storage gains: the problem can be formulated in a more compact way and some specialized solvers use specific forms of sparsity to speed up the resolution. Now, if we consider the pairwise GM M, by construction, there is no binary potential $\theta_S \in \Phi$ such that $S = \{x_i, x_i\},\ i \in [n]$. There is no binary potential that links a GM variable to itself. From this observation, one can see that the objective matrix Q has a zero block-diagonal structure. This central idea allows us to consider block-coordinate descent instead of the coordinate descent previously used in the mixing method.

$$Q = \begin{pmatrix} \boxed{0^{d_1 \times d_1}} & & & \\ & \boxed{0^{d_2 \times d_2}} & & \\ & & \ddots & \\ & & & \boxed{0^{d_n \times d_n}} \end{pmatrix}$$

We now go back to the optimization problem (2.14). We use the same transformation as in the previous section, see (2.2). First we do a change of variable and then a homogenization to get the following constrained -1/1 quadratic problem:

$$\begin{aligned} \min_{z \in \{-1,1\}^{d+1}} \quad & z^\top R z \\ \text{s.t} \quad & F_i^\top z = 2 - d_i, \quad \forall i = 1, \dots, n \end{aligned} \tag{2.15}$$

In this formulation, matrix R has the same exact sparsity as Q, meaning that R still has the zero block-diagonal property. This time we do not dualize the linear constraints to write (2.15) as an equivalent MaxCut problem. Using the rank one matrix variable $X = zz^\top$ and dropping the rank one constraint we get the following SDP relaxation:

$$\begin{aligned} \min_{X \in \mathbb{R}^{(d+1)\times(d+1)}} \quad & \langle R, X \rangle \\ \text{s.t} \quad & \langle U_i, X \rangle = 2 - d_i, \quad \forall i = 1, \dots, n \\ & \operatorname{diag}(X) = 1_{d+1} \\ & X \succeq 0 \end{aligned} \tag{2.16}$$

With matrix

$$U_i = \begin{pmatrix} 0^{d \times d} & F_i \\ F_i^\top & 0 \end{pmatrix}$$

Once again we use the low-rank factorization $X = VV^\top$ to transform (2.16) into the non convex optimization problem:

$$\begin{aligned} \min_{V \in \mathbb{R}^{(d+1)\times r}} \quad & \langle R, VV^\top \rangle \\ \text{s.t} \quad & \langle U_i, VV^\top \rangle = 2 - d_i, \qquad \forall i = 1, \dots, n \\ & \|V_i\| = 1, \qquad\qquad \forall i = 1, \dots, d+1 \end{aligned} \tag{2.17}$$

We now focus on the rows of the low-rank matrix V. We denote $f(V) = \langle R, VV^\top \rangle$ as the objective function of the optimization problem. The objective function with respect to the rows V_i of V is:

$$f(V) = \sum_{i=1}^{d+1} V_i^\top \sum_{j=1}^{d+1} R_{ij} V_j$$

As in the previous section, we note $g_i = \sum_{j=1}^{d+1} R_{ij} V_j$. In the mixing method, the idea was to minimize f regarding a row V_i of V while fixing all the other rows $V_j,\ j = 1, \dots, d+1,\ j \neq i$.

$$R = \begin{pmatrix} \overset{\leftarrow d_1 \rightarrow}{\lfloor 0 \rfloor} & & & \\ & \overset{\leftarrow d_2 \rightarrow}{\lfloor 0 \rfloor} & & \\ & & \ddots & \\ & & & \overset{\leftarrow d_n \rightarrow}{\lfloor 0 \rfloor} \\ & & 0 & \end{pmatrix} \qquad V = \begin{pmatrix} \left[\text{——} \ V_1^\top \ \text{——} \right] \\ \left[\text{——} V_{d_1+1}^\top \text{——} \right] \\ \vdots \\ \left[\text{——} \ V_d^\top \ \text{——} \right] \\ V_{d+1}^\top \end{pmatrix} \begin{matrix} \updownarrow d_1 \\ \updownarrow d_2 \\ \\ \updownarrow d_n \\ \updownarrow 1 \end{matrix}$$

For each discrete variable, if we consider the set of rows that corresponds to its domain, the unit vectors are now independent in the objective function. Without loss of generality for the first discrete variable $x_1 \in X$, if we look at the first d_1 rows of V which correspond to the d_1 values of x_i, in the objective function we have by construction $R_{ij} = 0$ for $(i, j) \in [1, d_1]^2$. It means that there are no monomials with terms $V_i^\top V_j,\ (i, j) \in [1, d_1]^2$ in the objective function. This observation is the same for all variables. Thus, for a particular row V_i which belongs to the set of rows that correspond to the domain of a variable, the vector $g_i = \sum_{j=1}^{d+1} R_{ij} V_j$ does not contain any row of that block. We can draw an elegant bridge between the discrete problem and this low-rank relaxation. For a discrete variable, we will now optimize over the whole domain of the variable instead of considering each value within the domain independently. Instead of doing row-by-row updates, for a given GM variable x_k, we simultaneously optimize all the rows of V that correspond to x_k while keeping

all other rows fixed. To simultaneously update all these rows, we have to solve:

$$\min_{V_i, s_k \leqslant i < s_{k+1}} \sum_{i=s_k}^{s_k+d_k-1} V_i^\top g_i \quad \text{s.t.} \begin{cases} V_{d+1}^\top (\sum_{j=s_k}^{s_k+d_k-1} V_j) = 2 - d_k \\ \|V_i\| = 1, \ s_k \leq i < s_{k+1} \end{cases} \tag{2.18}$$

This problem is more difficult to solve than the simple row update of the Mixing method. We now give some low-level implementation details of our method called *LR-BCD*: Low Rank Block Coordinate Descent. First, the rank chosen for the low-rank factorization $X = VV^\top$ differs from the rank previously used in the LR-LAS method. Compared to the LR-LAS formulation, the SDP formulation (2.16) has n additional equality constraints. Thus the rank r is updated as follows $r = \lceil \sqrt{2(n+d+1)} \rceil$ to meet the result of the Barvinok and Pataki theorem. This rank is the default rank used in the LR-BCD implementation. Later on, we will discuss other choices of ranks for the factorization and its effect on the method. The generic algorithm is shown in Algorithm 4.

Algorithm 4 LR-BCD for MAP/MRF

```
Initialize V_i randomly on the unit sphere
while not converged yet do
  for i = 1, ..., n do
    for j = s_i, ..., s_{i+1} − 1 do
      g_i := V C_i
    end for
    V_{s_i}, ..., V_{s_{i+1}} := solve (2.18)
  end for
end while
```

The method consists in cyclic updates of blocks of rows $(V_k), k \in [s_i, s_{i+1}]$ all other row vectors being fixed. The last row vector V_{d+1} of $V \in \mathbb{R}^{(d+1)\times r}$ that was added due to homogenization lifting in (2.15) is never updated during the procedure. This choice will be discussed in the following sections.

2.2.2 Optimization on the unit-sphere through trigonometry

At each step of the LR-BCD algorithm, we have to solve the following optimization problem on the unit sphere:

$$\min_{V_i, s_k \leqslant i \leqslant s_{k+1}} \sum_{i=s_k}^{s_k+d_k-1} V_i^\top g_i \quad \text{s.t.} \begin{cases} V_{d+1}^\top (\sum_{j=s_k}^{s_k+d_k-1} V_j) = 2 - d_k \\ \|V_i\| = 1, \ s_k \leq i < s_{k+1} \end{cases} \tag{2.19}$$

In the following, we make the assumption that the vectors g_i and V_{d+1} are never colinear. By the second constraint, every solution row vector V_i must lie in the

unit spherical manifold. This second constraint makes the problem non-convex. However, the next proposition allows us to implicitly include this constraint in a new formulation using geometric reasoning on a 2-dimensional subspace of $\mathbb{R}^r$.

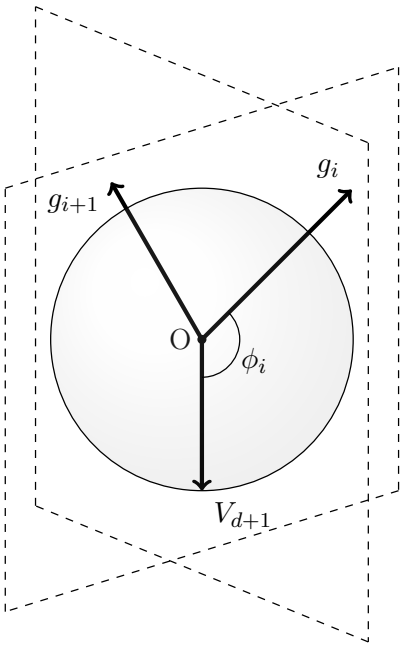


Figure 2.2: 2-dimensional subspace generated by V_{d+1} and the g_i's.

Theorem 2.2.1. *At the optimum of (2.19), every optimized vector V_i lies in the 2-dimensional subspace generated by g_i and V_{d+1}.*

Proof. Let $u = V_{d+1}$, we denote by P_i the 2-dimensional subspace generated by g_i and u, *i.e.*, $P_i := vec(g_i, u)$ and $P_i^\perp$ the orthogonal complement of P_i in $\mathbb{R}^r$. Let V_i^* be the vector V_i in an optimum of the problem above and assume that V_i^* does not lie in P_i. Let $\mathcal{B}_i := \{u, v\}$ the orthonormal basis of P_i such that g_i has a positive coordinate over v, let $\mathcal{C}_i := \{c_0, \ldots, c_{r-3}\}$ an orthonormal basis of $P_i^\perp$. Then, V_i can be written as

$$V_i = \alpha u + \beta v + \sum_{i=0}^{r-3} \gamma_i c_i$$

with at least one coefficient $\gamma_i \neq 0$. Since, when $\beta > 0$, changing the sign of β improves the criteria without changing the status of the "exactly-one" and norm-1 constraints, we must have $\beta < 0$. Let

$$\beta' = -\sqrt{\beta^2 + \sum_{i=0}^{r-3} \gamma_i^2}$$

and

$$V_i^{*\prime} = \alpha u + \beta' v.$$

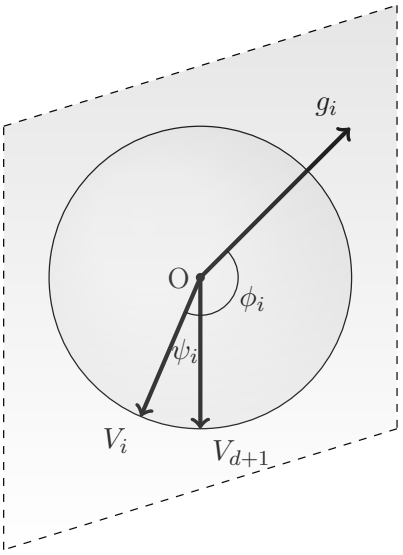


Figure 2.3: Optimizing V_i on the 2-dimensional subspace generated by V_{d+1} and g_i.

If we now consider a solution where V_i^* has been replaced by $V_i^{*\prime}$, the "exactly-one" constraint is still satisfied because the scalar product of $V_i^{*\prime}$ and u is unchanged. The norm-1 constraint is still satisfied as $\|V_i^{*\prime}\|^2 = \alpha^2 + \beta^2 + \sum_{i=0}^{r-3} \gamma_i^2 = 1$. Then, the new solution improves the criteria given that $\beta' < \beta < 0$ and that $g_i^\top v > 0$. □

Given that V_i lies on the unit sphere and the 2-dimensional subspace generated by g_i and V_{d+1}, it is therefore entirely defined by the angle it makes with V_{d+1}. With vectors on the sphere, the problem above can be rewritten using trigonometric functions, omitting the d_i norm-1 constraints which are implicitly satisfied. Let ϕ_i be the fixed angle between g_i and V_{d+1} and let ψ_i be the angle between V_i and V_{d+1} in this same plane. Problem (2.19) can be written as follows:

$$\min_{\psi_i \in [0,\pi]} \sum_{i=s_k}^{s_k+d_k-1} \|g_i\| \cos(\phi_i + \psi_i) \quad \text{s.t} \quad \sum_{i=s_k}^{s_k+d_k-1} \cos(\psi_i) = 2 - d_k \tag{2.20}$$

To deal with the remaining exactly-one constraint, we introduce the Lagrangian using multiplier $\lambda \in \mathbb{R}$:

$$L(\Psi, \lambda) = \sum_{i=s_k}^{s_k+d_k-1} \|g_i\| \cos(\phi_i + \psi_i) + \lambda \Big(\sum_{i=s_k}^{s_k+d_k-1} \cos(\psi_i) + d_k - 2 \Big) \tag{2.21}$$

Theorem 2.2.2. *The Lagrange dual function of the block optimization problem (2.20) is given by:*

$$h(\lambda) = \min_{\Psi} L(\Psi, \lambda) = - \sum_{i=s_k}^{s_k+d_k-1} \|g_i + \lambda V_{d+1}\| + (d_k - 2)\lambda$$

Proof. The partial derivatives of L are

$$\frac{\partial L}{\partial \psi_i} = -\|g_i\| \sin(\phi_i + \psi_i) - \lambda \sin(\psi_i)$$

which must all be equal to zero at the optimum. We use the fact that the sum of sinusoids with the same period and different phases is also a sinusoid:

$$A\sin(\omega t+\phi)+B\sin(\omega t) = \sqrt{A^2 + B^2 + 2AB\cos(\phi)} \sin\left(\omega t + \arctan\left(\frac{A\sin(\phi)}{A\cos(\phi)+B}\right)\right)$$

We have:

$$-\|g_i\|\sin(\phi_i+\psi_i)-\lambda\sin(\psi_i) = \sqrt{\|g_i\|^2 + \lambda^2 + 2\lambda\|g_i\|\cos(\phi_i)} \sin\left(\psi_i + \arctan\left(\frac{\|g_i\|\sin(\phi_i)}{\|g_i\|\cos(\phi_i)+\lambda}\right)\right)$$

which implies that

$$\psi_i + \arctan\left(\frac{\|g_i\|\sin(\phi_i)}{\|g_i\|\cos(\phi_i)+\lambda}\right) = 0 \text{ or } \psi_i + \arctan\left(\frac{\|g_i\|\sin(\phi_i)}{\|g_i\|\cos(\phi_i)+\lambda}\right) = \pi.$$

We take

$$\psi_i = \pi - \arctan\left(\frac{\|g_i\|\sin(\phi_i)}{\|g_i\|\cos(\phi_i)+\lambda}\right),$$

this choice is motivated later on in the primal feasibility therorem 2.2.6. By trigonometric identities, noting $\gamma_i = \frac{\|g_i\|\sin(\phi_i)}{\|g_i\|\cos(\phi_i)+\lambda}$, we have:

$$\cos(\psi_i) = -\cos(\arctan(\gamma_i)) = -\frac{1}{\sqrt{1+\gamma_i^2}}$$

Developing and simplifying $(1+\gamma_i^2)$ and plugging the result back above, we get:

$$\cos(\psi_i) = -\frac{\|g_i\|\cos(\phi_i)+\lambda}{\|g_i + \lambda V_{d+1}\|}$$

Similarly, one can derive:

$$\cos(\phi_i + \psi_i) = -\frac{\|g_i\| + \lambda\cos(\phi_i)}{\|g_i + \lambda V_{d+1}\|}$$

Plugging this back into the Lagrangian, we get:

$$\begin{aligned} L(\lambda, \Psi) &= \sum_{i=s_k}^{s_k+d_k-1} \left(-\|g_i\| \frac{\|g_i\| + \lambda\cos(\phi_i)}{\|g_i + \lambda V_{d+1}\|}\right) + \lambda \left(\sum_{i=s_k}^{s_k+d_k-1} \left(-\frac{\|g_i\|\cos(\phi_i)+\lambda}{\|g_i + \lambda V_{d+1}\|}\right) + d_k - 2\right) \\ &= -\sum_{i=s_k}^{s_k+d_k-1} \|g_i + \lambda V_{d+1}\| + \lambda(d_k - 2) \end{aligned}$$

So the dual problem is to find λ that maximizes

$$h(\lambda) \coloneqq -\sum_{i=s_k}^{s_k+d_k-1} (\|g_i + \lambda V_{d+1}\|) + \lambda(d_k - 2)$$

The first derivative of h is:

$$h'(\lambda) = -\sum_{i=s_k}^{s_k+d_k-1} \left(\frac{g_i^\top V_{d+1} + \lambda}{\|g_i + \lambda V_{d+1}\|} \right) + (d_k - 2)$$

and the second derivative:

$$h''(\lambda) = -\sum_{i=s_k}^{s_k+d_k-1} \left(\frac{\|g_i\|^2 - (g_i^\top V_{d+1})^2}{\|g_i + \lambda V_{d+1}\|^3} \right)$$

One can see that $h''(\lambda) < 0$ whenever g_i and V_{d+1} are not colinear which proves that h is strictly concave and guarantees the uniqueness of the optimum. Being strictly concave, we just have to find λ^* such that $h'(\lambda^*) = 0$. □

We could not exhibit a closed form of the maximizer λ^*. However, the function $f = -h$ can be optimized using the Newton algorithm.

Theorem 2.2.3. *The second derivative of f is Lipschitz on $\mathbb{R}$ and is strictly positive whenever the g_i's and V_{d+1} are not colinear.*

Proof. We need to show that the second derivative of f is Lipschitz on $\mathbb{R}$. In order to prove this, we will show that the derivative of f'' is bounded on $\mathbb{R}$. Let $u = V_{d+1}$, the first and second derivative of f are

$$f'(\lambda) = \sum_{i=s_k}^{s_k+d_k-1} \frac{g_i^\top u + \lambda}{\|g_i + u\|} + 2 - d_k$$

and

$$f''(\lambda) = \sum_{i=s_k}^{s_k+d_k-1} \frac{\|g_i\|^2 - (g_i^\top u)^2}{\|g_i + \lambda u\|^3}$$

respectively.

We denote by

$$f_i(\lambda) = \frac{\|g_i\|^2 - (g_i^\top u)^2}{\|g_i + \lambda u\|^3}$$

each term in the sum of the second derivative. We will show that the derivative of each f_i is bounded on $\mathbb{R}$.

$$f_i'(\lambda) = -\frac{3(\|g_i\|^2 - (g_i^\top u)^2)(g_i^\top u + \lambda)}{\|g_i + \lambda u\|^5}$$

$$= -\frac{3(\|g_i\|^2 - (g_i^\top u)^2)(g_i^\top u)}{\|g_i + \lambda u\|^5} - \frac{3(\|g_i\|^2 - (g_i^\top u)^2)\lambda}{\|g_i + \lambda u\|^5}$$

Using the assumption that g_i and u are not colinear there exists $k_i > 0$ such that $\|g_i + \lambda u\|^5 \geqslant k_i > 0$. Thus, we have :

$$\left|\frac{3(\|g_i\|^2 - (g_i^\top u)^2)(g_i^\top u)}{\|g_i + \lambda u\|^5}\right| \leqslant \frac{|3(\|g_i\|^2 - (g_i^\top u)^2)(g_i^\top u)|}{k_i}$$

Now we need to show that the second term is also bounded. Let $a_i = 3(\|g_i\|^2 - (g_i^\top u)^2)$, we can rewrite the absolute value of the second term as :

$$\left|\frac{a_i\lambda}{\|g_i + \lambda u\|^5}\right| = |a_i|\frac{(\lambda^2)^{\frac{1}{2}}}{(\|g_i\|^2 + 2g_i^\top u\lambda + \lambda^2)^{\frac{5}{2}}}$$

$$= |a_i|\left(\frac{\lambda^2}{(\|g_i\|^2 + 2g_i^\top u\lambda + \lambda^2)^5}\right)^{\frac{1}{2}}$$

If we consider the function :

$$l(\lambda) = \frac{\lambda^2}{(\|g_i\|^2 + 2g_i^\top u\lambda + \lambda^2)^5}$$

We know that l is a continuous function on $\mathbb{R}$. Moreover, we have:

$$l(\lambda) \underset{\lambda\to\pm\infty}{\sim} \frac{\lambda^2}{\lambda^{10}} = \frac{1}{\lambda^8} \underset{\lambda\to\pm\infty}{\longrightarrow} 0$$

We can conclude that l is a continuous function with finite limits at $\pm\infty$ thus it is bounded. So, we have the result that there exists $M \in \mathbb{R}_+^*$ such that the second term is bounded by $|a_i|\sqrt{M}$. □

To reach quadratic convergence with the Newton method, one has to find a starting point λ_0 which is close enough to the solution λ^*. Thankfully, we were able to derive a relaxation of the BCD problem (2.19) which produced good-quality solutions. Instead of working with each row separately, we introduce the stacked vector $V^\top = [V_{s_k}^\top \dots V_{s_k+d_k-1}^\top] \in \mathbb{R}^{(d_k\times r)}$. The norm constraints on the rows are then relaxed to $\|V\|_2^2 = d_k$. We now have to solve the following optimization problem:

$$\min_{V\in\mathbb{R}^{(d_k\times r)}} V^\top g \quad \text{s.t} \quad V^\top U = 2 - d_k, \;\; \|V\|_2^2 = d_k, \tag{2.22}$$

where $g^\top = [g_{s_k}^\top \dots g_{s_k+d_k-1}^\top] \in \mathbb{R}^{(d_k\times r)}$ and $U^\top = [V_{d+1}^\top \dots V_{d+1}^\top] \in \mathbb{R}^{(d_k\times r)}$. Once again this problem is non-convex due to the norm constraint but it is possible to derive a closed-form solution as stated in the following proposition.

Proposition 2.2.4. *Let us define* $p = 4 - \frac{4}{d_k}$ *and* $q = \|g\|^2 - \frac{1}{d_k}(U^\top g)^2$. *The solution to (2.22) is given by* $V^* = \alpha U + \beta g$ *with*

$$\alpha = -\frac{1}{d_k}(\beta U^\top g + d_k - 2), \;\; \beta = -\sqrt{\frac{p}{q}}$$

Let us first introduce a lemma that will be useful for the proof of this theorem.

Lemma 2.2.5. *The solution V^* to (2.22) must lie in the subspace $vec(U, g)$.*

Proof. Let us consider the sets

$$P_1 : \{y \in \mathbb{R}^{(d_k \times r)} \mid U^\top y = 2 - d_k\},\ D_1 : \{y \in \mathbb{R}^{(d_k \times r)} \mid \|y\|^2 = 2 - d_k\}.$$

For $y \in P_1$, we have $y = y_0 + z$ with y_0 a particular solution to the linear equation $U^\top y_0 = 2 - d_k$ and z a vector such that $U^\top z = 0$. For y_0, we take

$$y_0 = \frac{(2 - d_k)}{\|U\|^2} U$$

which gives us the desired result

$$U^\top y_0 = \frac{(2 - d_k)}{\|U\|^2} U^\top U = 2 - d_k.$$

The objective function is now equal to

$$g^\top y = \frac{(2 - d_k)}{\|U\|^2} U^\top g + z^\top g.$$

The first term is constant for all y in the feasible set $P_1 \cap D_1$. We now check the value of the second term. Let Π_{P_1} be the projector onto the hyperplane orthogonal to U. We can write the dot product

$$z^\top g = z^\top (g - \Pi_{P_1}(g) + \Pi_{P_1}(g)) = z^\top \Pi_{P_1}(g) \text{ since } z^\top (g - \Pi_{P_1}(g)) = 0.$$

Thus we only worry about the value of the dot product $z^\top \Pi_{P_1}(g)$ which is minimal in the direction $-\Pi_{P_1}(g)$. Since $\Pi_{P_1}(g)$ lies in the space generated by U and g we proved that the solution to (2.22) must lie in $vec(U, g)$.
Let us consider $V \in vec(U, g)$ e.g $V = \alpha U + \beta g$ for some $\alpha, \beta \in \mathbb{R}$. The first constraint gives

$$\alpha = -\frac{1}{d_k}(\beta U^\top g + d_k - 2).$$

By developing the squared norm we have

$$\beta^2 \left(\frac{1}{d_k} (U^\top g)^2 - \frac{2}{d_k} (U^\top g)^2 + \|g\|^2 \right) + \frac{(d_k - 2)^2}{d_k} = d_k$$

$$\beta^2 \left(\|g\|^2 - \frac{1}{d_k} (U^\top g)^2 \right) = d_k - \frac{(d_k - 2)^2}{d_k}$$

$$\beta^2 \left(\|g\|^2 - \frac{1}{d_k} (U^\top g)^2 \right) = 4 - \frac{4}{d_k}$$

Using the Cauchy-Schwartz inequality we know that $\|g\|^2 - \frac{1}{d_k}(U^\top g)^2 > 0$ whenever U and g are not colinear. Thus using the notation $p = 4 - \frac{4}{d_k}$

$$\beta = \mp \sqrt{\frac{p}{\gamma}} \text{ with } \gamma = \|g\|^2 - \frac{1}{d_k} (U^\top g)^2$$

One can see that the objective value is smaller for $\beta = -\sqrt{\frac{p}{\gamma}}$ which gives us the result. □

Let $V^* = [(V^*_{s_k})^\top, \dots, (V^*_{s_k+d_k-1})^\top$ be the solution to (2.22). We can use the stationarity condition of the initial problem (2.19) to compute an initial dual point λ_0. Without loss of generality, we can consider one of the equality:

$$\frac{\partial L(\Psi, \lambda)}{\partial \psi_i} = -\|g_i\| \sin(\phi_i + \psi_i) - \lambda \sin(\psi_i) = 0, \ i \in [s_k, s_k + d_k - 1]$$

Which gives

$$\lambda = \frac{-\|g_i\| \sin(\phi_i + \psi_i)}{\sin(\psi_i)}$$

Then we have to compute the angles between the relaxed solution V_i^* and the problem vectors V_{d+1} and g_i. Namely, we have:

$$\phi_i + \psi_i = \arccos\left(\frac{g_i^\top V_i^*}{\|g_i\|\|V_i^*\|}\right)$$

$$\psi_i = \arccos\left(\frac{V_{d+1}^\top V_i^*}{\|V_{d+1}\|\|V_i^*\|}\right)$$

And thus one can compute a starting dual point λ_0 for the Newton method. Empirically we found out that this candidate is close to the optimal solution and optimization methods like the Newton method or BFGS are able to converge to the maximizer of the dual Lagrange function within very few iterations (< 10).

Computing the solution vectors Using Theorem 2.2.1, we know that at the optimum of (2.19), there exist $\alpha_i, \beta_i \in \mathbb{R}$ such that $V_i = \alpha_i V_{d+1} + \beta_i g_i$. After computing the optimal dual multiplier λ^*, we can compute the angles ψ_i^* between V_i and V_{d+1} with the relation

$$\psi_i^* = \pi - \arctan\left(\frac{\|g_i\| \sin(\phi_i)}{\|g_i\| \cos(\phi_i) + \lambda^*}\right).$$

In order to determine α_i and β_i, we solve the following system of equations:

$$\begin{cases} v_i^\top g_i &= \alpha_i g_i^\top V_{d+1} &+ \beta_i ||g_i||^2 \\ v_i^\top V_{d+1} &= \alpha_i &+ \beta_i g_i^\top V_{d+1} \end{cases}$$

$$\begin{cases} ||g_i|| \cos(\phi_i + \psi_i^*) &= \alpha_i g_i^\top V_{d+1} &+ \beta_i ||g_i||^2 \\ \cos(\psi_i^*) &= \alpha_i &+ \beta_i g_i^\top V_{d+1} \end{cases}$$

$$\begin{cases} \alpha_i &= \cos(\psi_i^*) &- \beta_i g_i^\top V_{d+1} \\ \|g_i\| \cos(\phi_i + \psi_i^*) &= (\cos(\psi_i^*) - \beta_i g_i^\top V_{d+1}) g_i^\top V_{d+1} &+ \beta_i \|g_i\|^2 \end{cases}$$

$$\begin{cases} \alpha_i &= \cos(\psi_i^*) &- \beta_i g_i^\top V_{d+1} \\ \|g_i\| \cos(\phi_i + \psi_i^*) &- \cos(\psi_i^*) g_i^\top V_{d+1} &= \beta_i (\|g_i\|^2 - (g_i^\top V_{d+1})^2) \end{cases}$$

Using the Cauchy-Schwarz inequality we know that $\|g_i\|^2 > (g_i^\top V_{d+1})^2$ whenever V_{d+1} and g_i are not colinear, thus we can simplify the second equation:

$$\begin{cases} \alpha_i &= \cos(\psi_i^*) - \beta_i g_i^\top V_{d+1} \\ \beta_i &= \frac{\|g_i\|\cos(\phi_i+\psi_i^*) - \cos(\psi_i^*) g_i^\top V_{d+1}}{\|g_i\|^2 - (g_i^\top V_{d+1})^2} \end{cases} \tag{2.23}$$

Finally, we have completely determined the vectors V_i.

2.2.3 Computational complexity

Using a low-rank transformation, we are able to reduce a large-scale semi-definite program to a sequence of optimization problems on $\mathbb{R}$. In the previous section, we showed some properties of those optimization problems which allows us to use efficient second-order methods such as the Newton method. For our use case, the update rule for the Newton method is:

$$\lambda^{j+1} = \lambda^j - [\nabla^2 f(\lambda^j)]^{-1} \nabla f(\lambda^j)$$

and the first and second derivative of f are:

$$f'(\lambda) = \sum_{i=s_k}^{s_k+d_k-1} \frac{g_i^\top V_{d+1} + \lambda}{\|g_i + \lambda V_{d+1}\|} - (d_k - 2)$$

$$f''(\lambda) = \sum_{i=s_k}^{s_k+d_k-1} \frac{\|g_i\|^2 - (g_i^\top V_{d+1})^2}{\|g_i + \lambda V_{d+1}\|^3}$$

Given that all the vectors we are dealing with have size r, by pre-computing the dot products with the g_i's, computing the first derivative requires $O(d_k r)$ operations. For the second derivative, we can pre-compute the numerators $\|g_i\|^2 - (g_i^\top V_{d+1})^2$ and the evaluation of the second derivative is again $O(d_k r)$. Overall, one update step of the Newton method will require only $O(d_k r)$ operations, which is also what a single round of row-by-row updates over the d_k rows associated with x_k would require. The BCD updates however benefit from the efficient Newton updates. In practice, we observe that only a few iterations of the Newton method are needed in our experiments.

2.2.4 Strong duality and optimality

The block optimization problem (2.20) is not necessarily convex, that is, KKT conditions are not sufficient to prove optimality. However, for our problem, it is possible to show that the primal-dual pair (Ψ^*, λ^*) closes the duality gap, yielding optimality.

Theorem 2.2.6. *At the optimum,* $\Psi^* = (\psi_{s_k}^*, \ldots, \psi_{s_k+d_k-1}^*)$ *is primal feasible:*

$$\sum_{i=s_k}^{s_k+d_k-1} \cos(\psi_i^*) = 2 - d_k.$$

Proof. Using the proof of Theorem 2.2.2, we showed that at the optimum we have:

$$\cos(\psi_i^*) = -\frac{g_i^\top V_{d+1} + \lambda^*}{\|g_i + \lambda^* V_{d+1}\|}, \ i = s_k, \dots, s_k + d_k - 1.$$

The first order optimality condition for the dual Lagrangian gives us:

$$h'(\lambda^*) = 0 = -\sum_{i=s_k}^{s_k+d_k-1} \left(\frac{g_i^\top V_{d+1} + \lambda^*}{\|g_i + \lambda^* V_{d+1}\|} \right) + (d_k - 2),$$

which directly gives the desired result:

$$\sum_{i=s_k}^{s_k+d_k-1} \cos(\psi_i^*) = 2 - d_k. \tag{2.24}$$

□

Using this theorem, we can now state our main result for optimality.

Theorem 2.2.7. *The primal-dual pair* (Ψ^*, λ^*) *is a feasible pair that closes the duality gap.*

Proof. We denote by $f(\Psi) := \sum_{i=s_k}^{s_k+d_k-1} \|g_i\| \cos(\phi_i + \psi_i)$ the primal objective, Theorem 2.2.6 ensures feasibility of Ψ^* at the optimum. By a simple transformation, we have:

$$\begin{aligned} f(\Psi^*) &= \sum_{i=s_k}^{s_k+d_k-1} \|g_i\| \cos(\phi_i + \psi_i^*) \\ &= \sum_{i=s_k}^{s_k+d_k-1} \|g_i\| \cos(\phi_i + \psi_i^*) + \lambda^* \left(\sum_{i=s_k}^{s_k+d_k-1} \cos(\psi_i^*) + d_k - 2 \right) \\ &= h(\lambda^*). \end{aligned}$$

□

2.2.5 Descent property

In the last section, we showed that we can derive a relaxation of the MAP/MRF problem using the following non-convex optimization problem:

$$\begin{aligned} \min_{V \in \mathbb{R}^{(d+1)\times r}} \quad & \langle R, VV^\top \rangle \\ \text{s.t} \quad & \langle U_i, VV^\top \rangle = 2 - d_i, \qquad \forall i = 1, \dots, n \\ & \|V_i\| = 1, \qquad \forall i = 1, \dots, d+1 \end{aligned} \tag{2.25}$$

All along the LR-BCD procedure, the last row of the low-rank matrix V is never updated. We recall that this row corresponds to the extra variable that was added to the quadratic model for the homogenization of the linear part. If V_{d+1} remains fixed, we have the following property:

Theorem 2.2.8. *The block coordinate descent with fixed last row V_{d+1} is strictly decreasing.*

Before proving this theorem, we need an intermediate result. Once we have computed the optimal angles $\psi_i's$ we can compute the corresponding $V_i's$. Indeed since each V_i lies in the plane generated by g_i and V_{d+1} we can write it as a linear combination of g_i and V_{d+1} : $V_i = \alpha_i V_{d+1} + \beta_i g_i$.

Remark At the optimum $\psi_i^* = \pi - \arctan\left(\frac{\|g_i\|\sin(\phi_i)}{\|g_i\|\cos(\phi_i)+\lambda^*}\right) \neq 0$ and so is β_i.

Lemma 2.2.9. *Equality of the coefficient ratios.*
For each block $k \in [1, n]$ corresponding to a WCSP variable, the ratios $\frac{\alpha_i}{\beta_i}$, $i \in [s_k, s_k + d_k - 1]$ are equal.

Without loss of generality, we will work on the first block corresponding to the first WCSP variable.

First reminder Using the stationarity condition of (2.20), at the optimum the solution $\Psi^* = (\psi_1^*, \dots, \psi_{d_1}^*)$ must verify:

$$\frac{\partial L}{\partial \psi_i^*} = -\|g_i\|\sin(\phi_i + \psi_i^*) - \lambda \sin(\psi_i^*) = 0, \ \forall i \in [1, d_1].$$

So we have the result that at the optimum, the values $\lambda^* = \frac{-\|g_i\|\sin(\phi_i+\psi_i^*)}{\sin(\psi_i^*)}$ are the same for all the variables.

Second reminder $\forall i \in [1, d_1]$:

$$\begin{cases} \alpha_i &= \cos(\psi_i^*) - \beta_i g_i^\top V_{d+1} \\ \beta_i &= \frac{\|g_i\|\cos(\phi_i+\psi_i^*) - \cos(\psi_i^*) g_i^\top V_{d+1}}{\|g_i\|^2 - (g_i^\top V_{d+1})^2} \end{cases}$$

Proof of Lemma 2.2.9.

$$\frac{\alpha_i}{\beta_i} = \frac{\cos(\psi_i^*)}{\beta_i} - g_i^\top V_{d+1}$$

$$= \frac{\cos(\psi_i^*)(\|g_i\|^2 - (g_i^\top V_{d+1})^2)}{\|g_i\|\cos(\phi_i + \psi_i^*) - \cos(\psi_i^*) g_i^\top V_{d+1}} - g_i^\top V_{d+1}$$

$$= \frac{\cos(\psi_i^*)\|g_i\|^2 - \cos(\psi_i^*)(g_i^\top V_{d+1})^2 - g_i^\top V_{d+1}\|g_i\|\cos(\phi_i + \psi_i^*) + \cos(\psi_i^*)(g_i^\top V_{d+1})^2}{\|g_i\|\cos(\phi_i + \psi_i^*) - \cos(\psi_i^*) g_i^\top V_{d+1}}$$

$$= \frac{\cos(\psi_i^*)\|g_i\|^2 - g_i^\top V_{d+1}\|g_i\|\cos(\phi_i + \psi_i^*)}{\|g_i\|\cos(\phi_i + \psi_i^*) - \cos(\psi_i^*) g_i^\top V_{d+1}}.$$

Now let us work with the denominator:

$$\|g_i\|\cos(\phi_i + \psi_i^*) - \cos(\psi_i^*) g_i^\top V_{d+1} = \|g_i\|\cos(\phi_i + \psi_i^*) - \cos(\psi_i^*)\cos(\phi_i)\|g_i\|$$

$$= \|g_i\|(\cos(\phi_i + \psi_i^*) - \cos(\psi_i^*)cos(\phi_i))$$
$$= -\|g_i\|sin(\phi_i)\sin(\psi_i^*).$$

Then we transform the numerator as follows:

$$\cos(\psi_i^*)\|g_i\|^2 - g_i^\top V_{d+1}\|g_i\|\cos(\phi_i + \psi_i^*) = \cos(\psi_i^*)\|g_i\|^2 - \|g_i\|^2\cos(\phi_i)\cos(\phi_i + \psi_i^*)$$
$$= \|g_i\|^2(\cos(\psi_i^*) - \cos(\phi_i)\cos(\phi_i + \psi_i^*))$$
$$= \|g_i\|^2(\cos(\psi_i^*) - \cos(\phi_i)(\cos(\phi_i)\cos(\psi_i^*) - \sin(\phi_i)\sin(\psi_i^*))$$
$$= \|g_i\|^2(\cos(\psi_i^*) - \cos(\phi_i)^2\cos(\psi_i^*) + \cos(\phi_i)\sin(\phi_i)\sin(\psi_i^*)).$$

Then using $\cos(x)^2 = 1 - \sin(x)^2$:

$$= \|g_i\|^2(\cos(\psi_i^*) - (1 - \sin(\phi_i)^2)\cos(\psi_i^*) + \cos(\phi_i)\sin(\phi_i)\sin(\psi_i^*))$$
$$= \|g_i\|^2\sin(\phi_i)(\sin(\phi_i)\cos(\psi_i^*) + \cos(\phi_i)\sin(\psi_i^*)).$$

Then using $\sin(x + y) = \sin(x)\cos(y) + \cos(x)\sin(y)$:

$$= \|g_i\|^2\sin(\phi_i)sin(\phi_i + \psi_i^*).$$

To conclude we have:

$$\frac{\alpha_i}{\beta_i} = \frac{\|g_i\|^2\sin(\phi_i)\sin(\phi_i + \psi_i^*)}{-\|g_i\|\sin(\phi_i)\sin(\psi_i^*)} = \frac{-\|g_i\|\sin(\phi_i + \psi_i^*)}{\sin(\psi_i^*)}.$$

Which is exactly the Lagrange multiplier which is constant for each variable within the block. □

Proof of Theorem 2.2.8. Let us show that the objective difference before and after a BCD update is always positive. For a given row V_i the objective difference before and after the update is:

$$f(V_i) - f(\hat{V}_i) = g_i^\top(V_i - \hat{V}_i)$$

with

$$\hat{V}_i = \alpha_i V_{d+1} + \beta_i g_i$$

the newly updated row. We then have

$$g_i = \frac{1}{\beta_i}(\hat{V}_i - \alpha_i V_{d+1})$$

$$g_i^\top(V_i - \hat{V}_i) = \frac{1}{\beta_i}(\hat{V}_i - \alpha_i V_{d+1})^\top(V_i - \hat{V}_i)$$
$$= \frac{1}{\beta_i}\left((\hat{V}_i{}^\top V_i - 1) - \alpha_i(V_i - \hat{V}_i)^\top V_{d+1}\right).$$

The first term can be simplified as follows:

$$\frac{1}{\beta_i}(\hat{V}_i{}^\top V_i - 1) = -\frac{1}{2\beta_i}\|V_i - \hat{V}_i\|^2.$$

Moreover:

$$\beta_i = \frac{\|g_i\| \cos(\phi_i + \psi_i) - \cos(\psi_i) g_i^\top V_{d+1}}{\|g_i\|^2 - (g_i^\top V_{d+1})^2}$$

Using the Cauchy-Schwartz inequality, we already know that $\|g_i\|^2 - (g_i^\top V_{d+1})^2 > 0$. Next we will rewrite the numerator:

$$\begin{aligned}\|g_i\| \cos(\phi_i + \psi_i) - \cos(\psi_i) g_i^\top V_{d+1} &= \|g_i\|(\cos(\phi_i + \psi_i) - \cos(\psi_i)\cos(\phi_i)) \\ &= -\|g_i\| \sin(\phi_i)\sin(\psi_i)\end{aligned}$$

Since $\phi_i \in]0, \pi[$ and $\psi_i \in]0, \pi[$ the sinus are positive so we can conclude that $\beta_i < 0$. If we sum up for all the rows within the block:

$$\sum_{i=s_k}^{s_k+d_k-1} \frac{1}{\beta_i}(\hat{V_i}^\top V_i - 1) = -\sum_{i=s_k}^{s_k+d_k-1} \frac{1}{2\beta_i}\|V_i - \hat{V_i}\|^2 > 0.$$

We proved that the first term is positive, let us now check the second terms. For a particular optimization block, we showed that the terms $\frac{\alpha_i}{\beta_i}$ are constant. We denote by K this constant term, then if we sum up for all the block:

$$\begin{aligned}-\sum_{i=s_k}^{s_k+d_k-1} \frac{\alpha_i}{\beta_i}(V_i - \hat{V_i})^\top V_{d+1} &= -K \sum_{i=s_k}^{s_k+d_k-1} (V_i - \hat{V_i})^\top V_{d+1} \\ &= -K \left(\sum_{i=s_k}^{s_k+d_k-1} V_i^\top V_{d+1} - \sum_{i=s_k}^{s_k+d_k-1} \hat{V_i}^\top V_{d+1} \right),\end{aligned}$$

using primal feasibility of the $V_i's$ and $\hat{V_i}'s$:

$$= -K\,(2 - d_k - (2 - d_k)) = 0.$$

Thus, the second term is equal to 0 so we proved that

$$\sum_{i=s_k}^{s_k+d_k-1} f(V_i) - f(\hat{V_i}) > 0.$$

□

Stopping criteria Using the fixed row assumption, we proved that both function values and low-rank iterates converge towards a unique limit point. For this reason, we use these two pieces of information to compute the stopping criterion for LR-BCD. We work with relative stopping criteria so that we are not dependent on problems that may have different weightings. Let ε be a given tolerance specified by the user, LR-BCD stops whenever one of these two conditions is met:

- function criterion:

$$(f(V^k) - f(V^{k+1}))/(1 + |f(V^k)|) < \varepsilon.$$

- Iterate criterion:
$$\|V^k - V^{k+1}\|_F/(1 + \|V^k\|_F) < \varepsilon.$$

With V^k the low-rank matrix at iteration k. Note that since LR-BCD is decreasing, $f(V^k) - f(V^{k+1})$ is always positive. The user also have the choice to stop the method earlier by using the $-it$ parameter which controls the maximum number of iterations.

2.2.6 Producing an integer primal solution

To produce a primal integer solution from the optimal solution of the low-rank relaxation, we use a slightly modified version of the Goemans and Williamson rounding scheme presented in Section 1.6. Indeed, for LR-LAS and LR-BCD, the integer solution must satisfy the exactly-one constraint. Let V_r be a normalized vector sampled from a Gaussian distribution. We assign each variable to the value $x_k = \left(\arg\max_{s_k \leqslant i \leqslant s_{k+1}} V_i^T V_r\right) - s_k$. The integer solution obtained is then submitted to a simple greedy search where, for every variable, the best improving change of state (if any) is applied. This is done on every variable repeatedly until a local minimum is reached. For LR-BCD and LR-LAS, we repeat this process with several random vectors and we return the best integer solution found. The total number of roundings can be controlled by the user with the parameter *-nbR*.

2.3 Experiments

In this section, we discuss the experimental results of our two methods against state-of-the-art message passing and LP solvers on a variety of crafted and real-world instances. We also explore a key parameter for the efficiency of both low-rank methods, namely the rank used for the relaxation.

2.3.1 Description of solvers

We implemented the row-by-row updates with dualized exactly-one constraint (LR-LAS) as well as the BCD update method (LR-BCD) in C++ with the Eigen3 [Guennebaud et al., 2010] linear algebra library. The code is available online at `https://github.com/ValDurante/LR-BCD`. We use rank $r = \lceil\sqrt{2}(d+1)\rceil$ by default for LR-LAS and $r = \lceil\sqrt{2}(n+d+1)\rceil$ for LR-BCD. This corresponds to the number of constraints for the two SDP formulations thus satisfying the Barvinok and Pataki requirement. We also tried variants with fixed lower ranks. For LR-LAS, the penalty coefficient ρ is set as the sum of the maximum of all the binary and unary functions. Thus, it satisfies the theoretical assumption of the Lasserre dualization result, but it may be larger than needed, leading to slower convergence. They are compared with exact LP (local polytope) bounds and message-passing MAP/MRF algorithms Min-Sum and TRW-S. The LP bounds are computed using

CPLEX 20.1.0.0. We used Open-GM2 [Kappes et al., 2015], an efficient C++ MIT-licensed library, in release 3.3.7, available at https://github.com/opengm for Min-Sum and TRW-S. For Min-Sum we used the following parameters: maximum number of iterations of 100, minimal message distance of 0.01, and damping of 0.8. For TRW-S: maximum number of iterations of $100,000$, tolerance 10^{-5} and TABLE mode. Min-Sum provides only upper bounds.

Overall, we therefore compare five solvers: LR-LAS and LR-BCD (parameterized by their rank), LP, Min-Sum and TRW-S. We also considered using ECOS interior point method available in CVXPY [Agrawal et al., 2018], but preliminary tests showed that it ran several orders of magnitude slower than the Mixing Method on small problems with 120 binary variables from the BiqMac library available at http://biqmac.uni-klu.ac.at/biqmaclib.html.

All experiments were run on a single thread on a server equipped with a Xeon®Gold 6248R CPU@3.00GHz and 1TB of RAM running Debian Linux 4.19.98-1.

2.3.2 Random instances

Following [Park et al., 2019], we first used random pairwise MAP problems. The pairwise potentials are sampled uniformly from $[0, s]$ with $s = 5$. The unary potentials are sampled uniformly from $[0, 1]$ using fixed precision numbers with 9 digits precision. The magnitude of s controls the importance of the pairwise couplings which are the source of NP-hardness. Empirically, we found that solution time and quality were insensitive to the value of s except for very low s (where coupling effects become negligible). We generate random problems varying the number of variables from 100 to 1000 for the largest instances. All variables have the same number of states ranging from 3 to 10. All the random problems have complete graphs. For all figures presented in this section, for each point, the results are averaged over 10 instances. We observed that the standard deviations of the measures were very low on the 10 generated samples and we therefore do not report it.

2.3.2.1 CPU-time

We present in Figure 2.4 a summary of the CPU-time performance of all solvers on problems of increasing size (100 to 1000 MRF variables) and increasing number of states (and therefore increasing size d). TRW-S is clearly the most efficient algorithm here, often one order of magnitude faster than LR-BCD. LR-LAS instead, is considerably slower than LR-BCD, often by more than two orders of magnitude. On instances of 300 MRF variables and more than 3 states ($d > 900$), it times out. The CPLEX-based exact local polytope solver is even worse. It times-out on 100 variables instances with 7 states.

2.3.2.2 Bounds

Figure 2.5 shows normalized upper and lower bounds on the same families of instances. Considering upper bounds, message-passing algorithms show slightly infe-

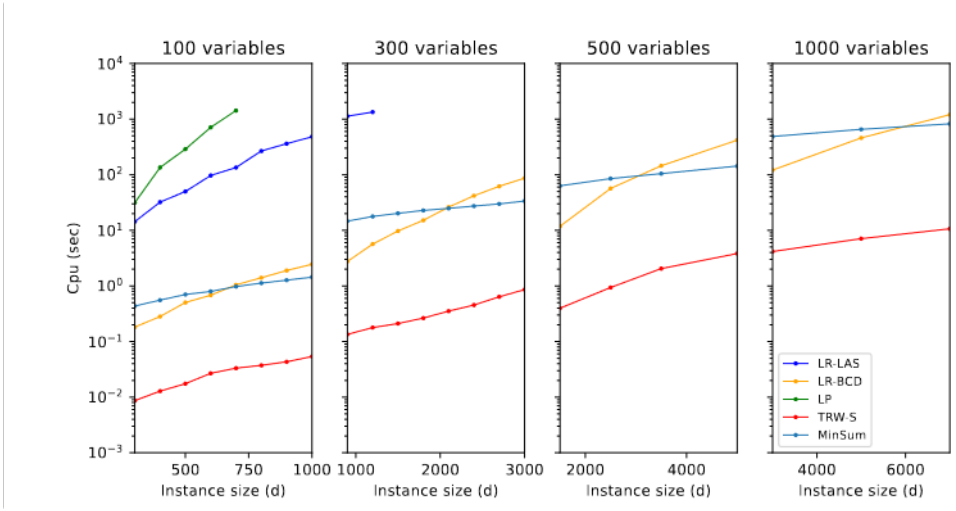


Figure 2.4: Cpu-time for all solvers on problems of increasing size (100, 300, 500, and 1000 variables) as a function of the problem size (number of states $\times$ number of variables).

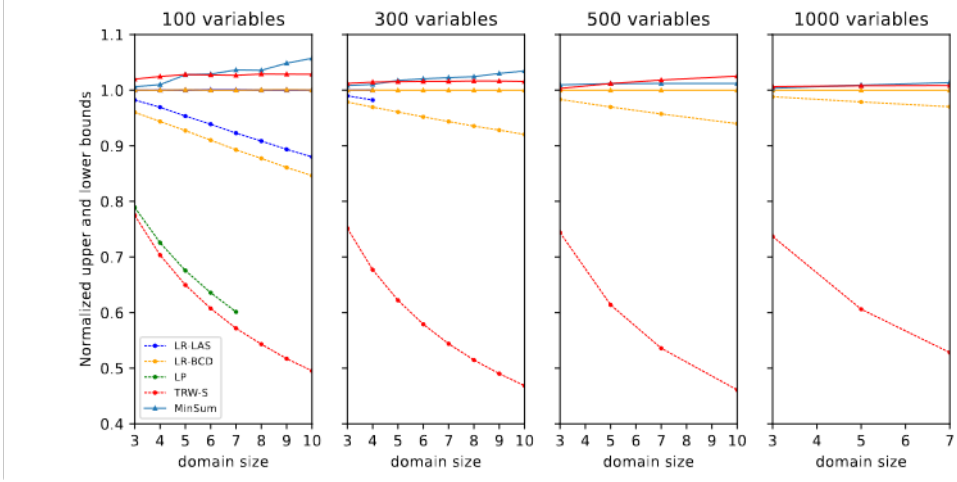


Figure 2.5: Normalized upper and lower bounds for all solvers on problems of increasing size (100, 300, 500, and 1000 variables) as a function of the number of states (domain size). The best integer primal solution value is taken as a reference.

rior performance compared to other solvers. The most important differences appear in the lower bounds, where LP only slightly improves over TRW-S and where LR-LAS and LR-BCD really stand out. Thanks to its implicit enforcing of the gangster constraints, LR-LAS offers slightly improved lower bounds, but this extra tightness comes at an extreme computational cost. It is interesting to notice that for LR-BCD, the optimality gap decreases as the ratio of (number of variables)/(number of states) increases. This means that for a fixed number of states, increasing the number of variables will make the LR-BCD bounds more precise.

2.3.2.3 Very low-rank relaxations

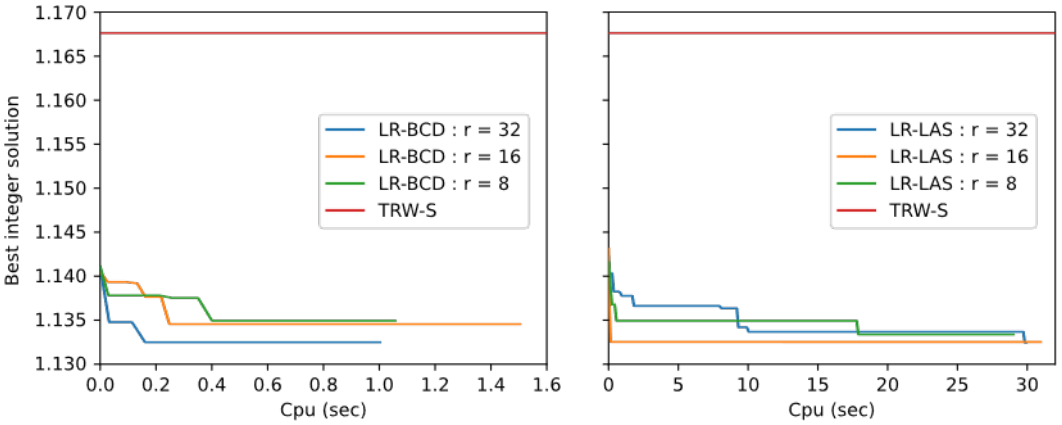


Figure 2.6: Comparison of the best integer solution as a function of time for TRW-S and LR-BCD (left) and TRW-S and LR-LAS (right) on a 100 variables instance with five states. Note the different time scales.

One interesting feature of Burer-Monteiro style methods is the ability to control their efficiency through the rank r. As far as integer solutions are concerned, it is perfectly fine to use low or very low ranks, below the theoretical limits of $\lceil\sqrt{2(d+1)}\rceil$ and $\lceil\sqrt{2(n+d+1)}\rceil$. Indeed, rounding will still provide a feasible integer solution. To compare the ability of TRW-S, LR-LAS and LR-BCD to quickly provide an interesting integer solution, we ran all three algorithms with increasing time-outs and measured the cost of the integer solution produced as a function of time. Because LR-LAS times-out on all but the smallest instances, we tested this on a 100 variables, 5 states instance (we observed very consistent behavior among all samples from a given family of instances). This is illustrated in Figure 2.6. TRW-S is extremely fast and immediately produces its best possible integer solution. Surprisingly, LR-LAS and LR-BCD are able to produce a better integer solution, even from the first iteration. This integer solution also improves in quality as the number of iterations grows.

The effect of strong rank reduction is different between LR-LAS and LR-BCD.

For best performance, LR-BCD is best used with a large rank: it reaches its best solution after less than 25 iterations (0.2 seconds). Instead, LR-LAS may benefit from an intermediate rank, where it produces a comparable solution well before convergence. With the highest rank, the quality of the integer solutions produced by LR-LAS improves very slowly. This could be explained by the fact that the primal relaxed solutions that LR-LAS computes do not exactly satisfy the exactly-one constraints of each MRF variable until the very end.

2.3.2.4 Empirical convergence results

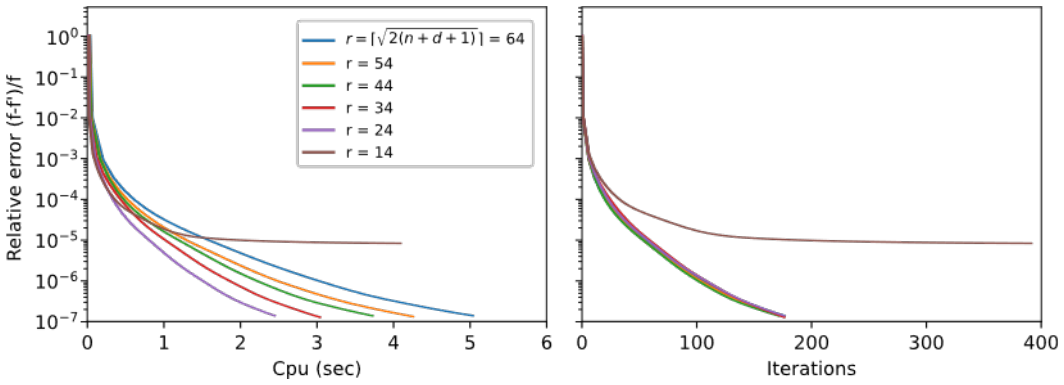


Figure 2.7: Relative error with a high-quality solution vs. cpu-time (left) and number of iterations (whole matrix update, right) using different ranks for the relaxation. The experiment was made on a 500 variables, three states instance ($d = 1500$) and is representative of the behavior on all instances.

In this section, we empirically study the convergence of LR-BCD and the effect of the rank r on the value of the optimized criteria with increasing cpu-time and number of iterations. For an instance with 500 variables and three states, we computed a tight solution of (2.16) using Mosek [ApS, 2022]. Figure 2.7 shows the relative error of the solutions produced by LR-BCD compared to Mosek solution as a function of cpu-time. We observe that, even when the rank is high, LR-BCD converges very quickly to solutions with very low tolerance, a tolerance of 10^{-3} being reached in less than 0.5 seconds. As the rank decreases, the convergence to a certain precision accelerates without any loss in precision, until a very low rank is reached where the quality of the solution cannot be pushed to tight tolerance. On the right, we also plot the relative error against the number of BCD iterations. Each iteration corresponds to an entire update of the low-rank matrix V. As the rank reduces, the number of iterations grows to the point where faster iterations are almost compensated by the increased number of iterations. In comparison, the industrial solver Mosek, with default settings, took 111 seconds to solve this relatively small instance.

Figure 2.8 shows the relative error of Mosek and LR-BCD compared to Mosek so-

lution as a function of the time. Mosek was run with default settings and `nb_threads` = 10 threads. For LR-BCD we set the rank to be the quotient of $\lceil\sqrt{2(n+d+1)}\rceil/2$. Since Mosek uses multi-threading, we used the wallclock-time for Mosek and the cpu-time for LR-BCD. As a comparison, LR-BCD achieves a speed-up of $\sim 10^2$ to reach a relative error of 10^{-5}. At the end, the total cpu-time was 27.57 seconds for LR-BCD and 2448.79 seconds for Mosek.

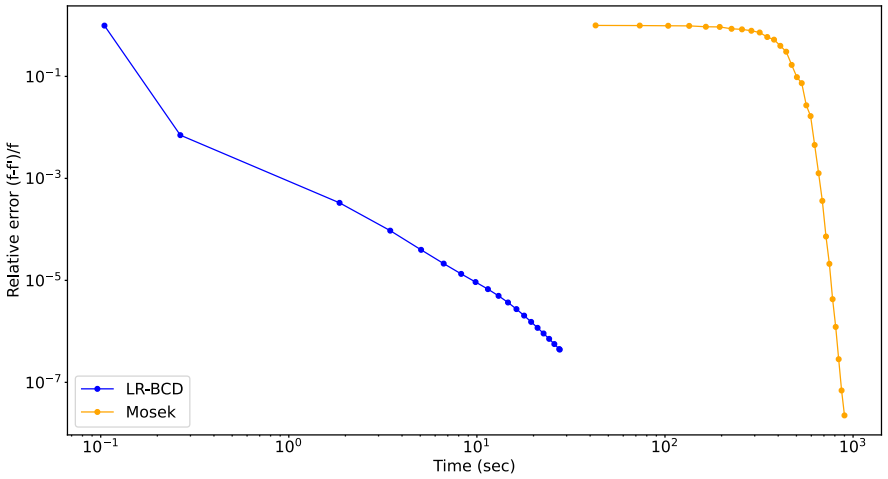


Figure 2.8: Relative error with a high-quality solution vs. time for LR-BCD and Mosek on a 1000 variables, three states instance ($d = 3000$).

2.3.2.5 Relaxing the LR-LAS penalty parameter

Let us recall the original 0/1 constrained quadratic reformulation of the MAP problem on pairwise discrete graphical models:

$$\begin{aligned} \min_{b\in\{0,1\}^d} \quad & b^\top Q b + c^\top b \\ \text{s.t} \quad & Ab = 1_n \end{aligned}$$

and its penalized -1/1 counterpart using the Lasserre transformation:

$$\min_{y\in\{-1,1\}^d} \quad y^\top R y + y^\top e + (2\rho+1)\|Fy-u\|^2$$

As soon as $\rho \geqslant \max\{|y^\top R y + y^\top e| \ : \ y \in \{-1,1\}^d\}$, the solution of the two problems are the same [Lasserre, 2016]. In the original paper, the author suggests computing the scalars:

$$r^1_{e,R} = \min\{e^\top y + \langle X, R\rangle : \begin{pmatrix} 1 & y^\top \\ y & X \end{pmatrix} \succeq 0;\ X_{ii} = 1,\ i = 1,\ldots,d\}$$

$$r^2_{e,R} = \max\{e^\top y + \langle X, R\rangle : \begin{pmatrix} 1 & y^\top \\ y & X \end{pmatrix} \succeq 0;\ X_{ii} = 1,\ i = 1, \dots, d\}$$

and set

$$\rho := \max_{i=1,2} |r^i_{e,R}|$$

which clearly satisfies the constraint above. However, we experimentally found that introducing coefficients with a relatively high magnitude compared to other coefficients in the cost matrix can drastically reduce the convergence speed of the mixing method. To obtain better efficiency and scalability with LR-LAS, one can try different settings for the penalty parameter ρ. We tried different versions of LR-LAS by incrementally decreasing the penalty parameter below the theoretical bound introduced above. To ensure that the modified version is still a relaxation of the original problem, we use the following proposition [Gilbert, 2021]:

Proposition 2.3.1. *Penalization monotony*
Let X_s be a non-empty set, f and $p: X_s \to \mathbb{R}$ two functions, $r \in \mathbb{R}$ and $\Theta_r := f + rp$. If $r_1 < r_2$ are two reals and if $\bar{x}_{r_i} \in \arg\min\{\Theta_{r_i}(x) : x \in X_s\}$ $(i = 1, 2)$, then:

1. $p(\bar{x}_{r_1}) \geqslant p(\bar{x}_{r_2})$,
2. $f(\bar{x}_{r_1}) \leqslant f(\bar{x}_{r_2})$ if $r_1 \geqslant 0$,
3. $\Theta(\bar{x}_{r_1}) \leqslant \Theta(\bar{x}_{r_2})$ if $p(\bar{x}_{r_2}) \geqslant 0$.

The last item gives the desired property since the penalty function is the squared 2-norm in $\mathbb{R}^d$ which is always positive. However, decreasing the penalty parameter ρ also makes the relaxation less tight. In Figure 2.9 we show the effect of ρ on the running time and solution quality of LR-LAS. Each data point corresponds to the average over 10 runs of LR-LAS with a fixed penalty parameter. The results are compared with the outputs of LR-BCD which do not depend on ρ.

We observe that, even by decreasing the penalty parameter, LR-LAS is dominated by LR-BCD in terms of cpu performance. The solution quality of LR-LAS remains better than the one of LR-BCD until a certain value of ρ after which it quickly deteriorates.

2.3.3 Sparse problems

Until this point, all the experiments were run on dense MAP/MRF instances, that is, the graph corresponding to the GM $\mathcal{M}$ is complete. We also decided to explore sparse problems for which message-passing algorithms are usually the methods of choice. We generated Erdős–Rényi graphs with various probability parameters p. Each edge in the graph, or equivalently binary potential, is included with probability p, independently from every other edge. As in the previous experiments, the corresponding binary potentials are sampled uniformly from $[0, s]$ with $s = 5$. In Figure 2.10 we show the gap between the values of LR-BCD and LR-LAS primal solutions and the values of the integer solutions given by the Goemans and Williamson

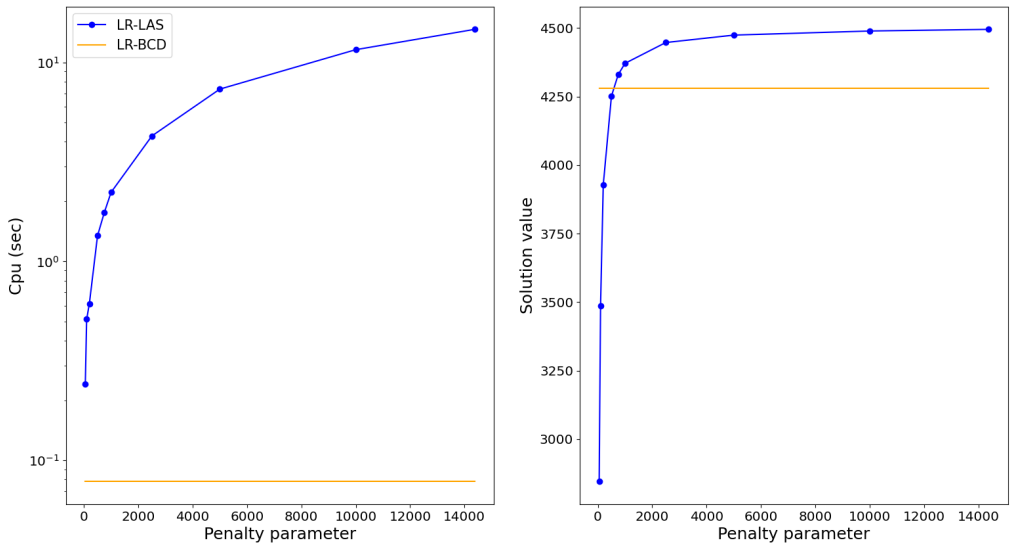


Figure 2.9: Cpu-time vs penalty parameter (left) and solution value vs penalty parameter (right). The experiment was made on a 100 variables, three states instance ($d = 300$).

heuristic. One can see the clear limitation of LR-BCD on these instances. As the probability p decreases and the number of states increases, the quality of the LR-BCD solutions becomes worse. As a reminder, for the MAP/MRF problem with positive costs, we can always bound the optimal value with the trivial lower bound $lb = 0$. At a certain point, the relaxation solution yields negative values, making it useless for a branch and bound scheme.

2.3.4 Real world problems

Genome assembly We applied the same algorithms to a problem occurring in the context of genome sequencing and more precisely at the genome assembly step. In this case study, biologists are interested in sequencing the DNA of a fish that has undergone genomic duplications. Under the pressure of evolutive forces and breeding, some animal and plant species can undergo whole genome duplication (WGD), an event where the genome is duplicated in two identical copies. As time passes, neutral and beneficial mutations accumulate in these copies, which can slowly drift away.

For genome sequencing, DNA is extracted and cut into pieces that are read at their extremities, resulting in a set of short DNA sequences called reads. Because DNA sequencing is generally not totally reliable, standard methods repeat this process to get a very large set of reads, for a total amount of data that on average covers the genome several times. Strongly overlapping reads are then merged into larger

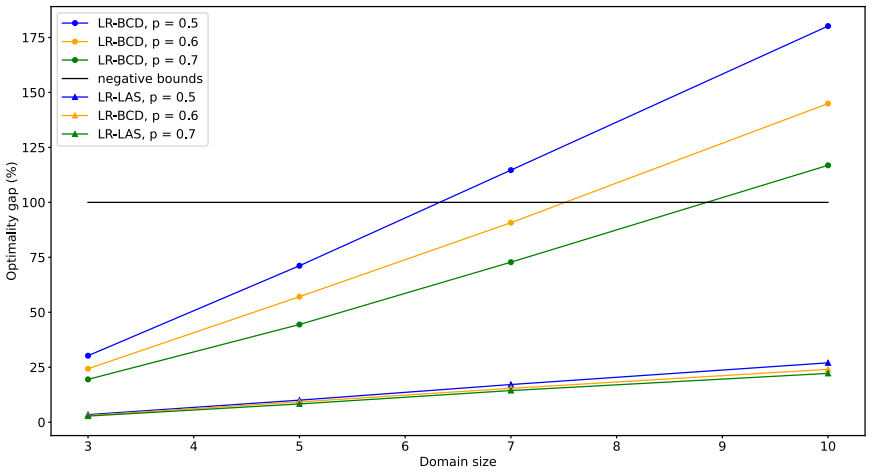


Figure 2.10: Optimality gap vs Domain size on ER-instances with varying probability parameter p. The experiments were made on instances with 100 variables and domain sizes varying from 3 to 10.

regions called contigs. These contigs are in turn used to reconstruct the genome. For our problem, we work on a fish species that has undergone two WGD in the course of its evolution. Each duplication creates two copies which we denote as D_1 and D_2 respectively. Copies create "false" overlaps between regions which hinder the assembly process, leading to the creation of chimeric assemblies where a region from one copy is merged with a region from another copy. This is especially true for recent whole genome duplications, which have not yet drifted because of evolution.

Our objective is to identify the copies resulting from a duplication in a given set of contigs. Fortunately, contigs contain information that can help us. Indeed, the genome contains specific regions (gene encoding regions) which should be still visible in both copies. These regions are known and can be identified in each contig. If two given contigs share a sufficient number of them, this is a strong indication that these two regions come from different regions. We can now model our problem:

- We consider a set C of n contigs.
- Each contig $i \in C$ has a set T_i of indices corresponding to the identified regions.
- For each pair of contigs $(i, j) \in C^2$ we associate a weight $w_{ij} = |T_i \cap T_j|$.

If we consider a pair of contigs $(i, j) \in C^2$, if $|T_i \cap T_j|$ is large, this indicates that these contigs must belong to different copies. We can now introduce the weighted graph $G = \langle X, E \rangle$ with $X = C$, $E = \{(i, j) \mid (i, j) \in C^2,\ w_{ij} \neq 0\}$ and each edge $(i, j) \in E$ has weight w_{ij}. Reconstructing the two copies resulting from one

duplication can be modeled by a MaxCut problem instance on G, where each side of the cut corresponds to one of the two copies. The solution to this problem can be approximated using the basic MaxCut relaxation.

In practice, dealing with a single duplication in the context of a genome that has undergone two duplications is already very satisfactory, as the most recent duplication is the one that raises the most issues because the two induced copies are highly similar. If needed, the problem of directly dealing with two duplications can be modeled using a graphical model $\mathcal{M} = \langle X, \Phi \rangle$. X is a set of n discrete variables corresponding to the n contigs. Each variable has domain size $d_i = 4$ (the number of copies). Using the notations defined above, for each pair of contigs $(i, j) \in C^2$ such that $w_{ij} \neq 0$ we introduce a binary potential:

θ_{ij}	1	2	3	4
1	w_{ij}	0	0	0
2	0	w_{ij}	0	0
3	0	0	w_{ij}	0
4	0	0	0	w_{ij}

Then our problem reduces naturally to the MAP problem on $\mathcal{M}$. Our fish instance has $8,574$ variables having 4 states ($34,296$ Boolean variables) and $80,763$ binary potentials. It has been made available in the Cost Function Library[1], in the `real/fish` category. On this instance,TRW-S provides a lower bound of 0 and an integer solution with cost $127,878$ in 0.069 seconds. The LP bound is also 0 with a running time of 80.04 seconds. Min-Sum provides an integer solution of cost $479,924$ in 0.270 seconds (and seems stuck there, with no improvement with more iterations). LR-BCD produces a bound of $102,900$ and an integer solution of cost $122,694$, reducing the optimality gap from 100% (TRW-S) to 16.1% (LR-BCD). LR-BCD took 68.6 minutes to process this instance, while LR-LAS did not finish after several hours.

Computational protein design Proteins are one of the key components of all living organisms. Computational protein design (CPD) aims to conceive new proteins with properties that can be useful for a wide range of applications. For the past 30 years, CPD has been a hot topic at the interface between biology and computer science. The function of a protein is tightly related to its 3D structure. Formally, a protein is a sequence of smaller blocks called amino acids which folds into a certain 3D shape. Recently, Deep learning methods were successfully applied to the problem of predicting this final fold [Jumper et al., 2021]. Given a sequence of amino acids, the deep neural network tries to estimate the 3D conformation of the protein.

A team in our lab is instead interested in the inverse problem. They try to find sequences of amino acids (or residues) that fold into a certain conformation. To do so, Marianne Defresne, a PhD student at INRAE under the supervision of Thomas

[1] `https://forgemia.inra.fr/thomas.schiex/cost-function-library/-/tree/master/real/fish`

Schiex and Sophie Barbe, explored a method that combines automated reasoning (based on discrete optimization) and deep learning. CPD can be cast into a discrete energy minimization problem. If we only consider physics interactions between pairs of amino acids, this problem can in turn be modeled as a MAP problem on a pairwise GM.

The idea proposed by Marianne is to first train a neural network on existing protein structures to predict the GM mentioned above. The joint function defined by the GM is called Effie [Defresne et al., 2023]. Once Effie is learned, it is used for protein design. In other words, Effie replaces traditional energy functions for design [Pavlovicz et al., 2020]. The optimization of Effie can be modeled as follows:

- We consider a protein backbone with a set R of n residues.
- Each residue $i \in R$ defines a variable with a domain containing the possible 20 canonical amino acids.
- For each pair (i, j) of residues in the backbone, we have a binary potential θ_{ij} learned by the neural net. Every potential has a size of 20×20.

Then the minimization of Effie naturally reduces to a MAP problem on the pairwise GM $\mathcal{M} = \langle X, \Phi \rangle$, with $X = R$ and $\Phi = \{\theta_{ij} \mid (i, j) \in [n]^2,\ i < j\}$. In their work, Defresne et al. first minimized Effie with the exact solver toulbar2 [Hurley et al., 2016]. However, for large proteins ($n > 100$), solving the instance optimally becomes a hard task for toulbar2. Therefore, they started to use approximate methods. The first idea is to call toulbar2 with a limited number of backtracks. For example, the solver is stopped after reaching the first leaf node. A second idea is to use our low-rank solver LR-BCD with multiple calls to the rounding and greedy local search. The best solution returned is kept. For the largest proteins, LR-BCD is run with a limited number of iterations.

To assess the quality of the overall protein design method, the sequence returned by toulbar2 or LR-BCD is compared to the native sequence. The comparison is made using the Native Sequence Recovery rate (NSR), *i.e.*, the percentage of similarity between the two sequences. Note that a designed sequence with an NSR between 30 and 40% is already considered as satisfactory. Figure 2.11 shows the comparison of the two methods on proteins with less than $n = 150$ residues.

One can see that the results returned by toulbar2 and LR-BCD are similar. However, LR-BCD is much faster even when toulbar2 is set to use 0 backtrack as shown in Figure 2.12. As a result, LR-BCD was chosen as the default solver for *in silico* validation. The authors used it to optimize proteins with up to $n = 1500$ residues ($d = 1500 \times 20 = 30000$).

We thank Marianne Defresne and her co-authors for sharing their results.

2.4 Discussion

In this section, we discuss open questions and further improvements regarding our low-rank solver.

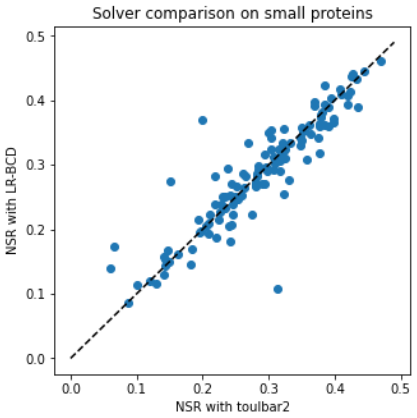


Figure 2.11: Similarity to the native sequence (NSR) of sequences designed with toulbar2 or LR-BCD.

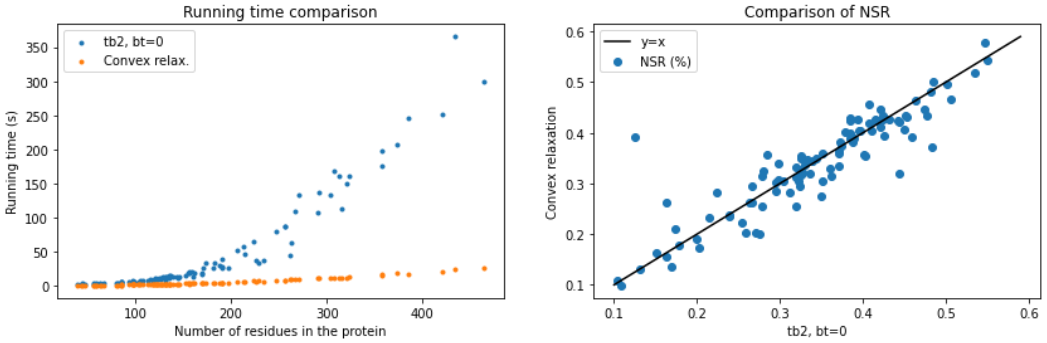


Figure 2.12: Comparison between LR-BCD and toulbar2 with no backtrack. Left: inference time on each instance. Right: NSR on each instance.

2.4.1 Update rule for the cost matrix

The starting point of our study is the resolution of a combinatorial problem. A usual method to solve such a problem is to use a branch and bound algorithm introduced in subsection 1.3.2. Now we discuss how to efficiently reformulate the problem after a branching decision. Let us consider a pairwise graphical model $\mathcal{M} = \langle X, \Phi \rangle$. In the following, we distinguish between two cases:

- The penalized formulation, corresponding to the LR-LAS solver.
- The constrained formulation, corresponding to the LR-BCD solver.

These two formulations have different cost matrices, which must be taken into account when reformulating the problem. Let us consider a variable $x_i \in X$ and a value $a \in D_i$, branching on variable x_i and value a creates two new subproblems:

1. A subproblem where x_i is assigned to value $a \in D_i$.
2. A subproblem where value $a \in D_i$ is forbidden or removed from the domain.

Constrained formulation In this case, we work with the matrix $R \in \mathbb{R}^{(d+1)\times(d+1)}$ introduced in the formulation (2.16). We consider that the branching decision is $x_i = a$ or value a is removed from the domain D_i. We denote by j the row/column index corresponding to variable x_i and value a in the objective cost matrix R. We denote by R_j and C_j the j-th row and column of R and $r, c \in \mathbb{R}^{d+1}$ the last row and column of R.

1. Remove value a from D_i: if we remove value a from the domain of x_i then the matrix R is updated as follows:

$$\begin{aligned} r &\leftarrow r - 2R_j \\ c &\leftarrow c - 2C_j \\ R &\leftarrow R \setminus \{R_j \cup C_j\} \in \mathbb{R}^d, \end{aligned}$$

   where $R \setminus \{R_j \cup C_j\}$ corresponds to matrix R with row and column j removed. The exactly-one constraint corresponding to variable x_i becomes:

$$\langle U_i, X \rangle = 2 - (d_i - 1) = 3 - d_i.$$

2. Assign $x_i = a$: the matrix R is updated as follows:

$$\begin{aligned} r &\leftarrow r - 2R_{s_i} - \cdots - 2R_{j-1} - R_j - 2R_{j+1} - \cdots - 2R_{s_i+d_i-1} \\ c &\leftarrow c - 2C_{s_i} - \cdots - 2C_{j-1} - C_j - 2C_{j+1} - \cdots - 2C_{s_i+d_i-1} \\ R &\leftarrow R \setminus \left\{ \bigcup_{l=s_i}^{s_i+d_i-1} R_l \cup \bigcup_{l=s_i}^{s_i+d_i-1} C_l \right\} \in \mathbb{R}^{(d+1-d_i)\times(d+1-d_i)}. \end{aligned}$$

   In this case the exactly-one constraint corresponding to variable x_i can be removed.

Note that when we remove a value from a domain, the size of the problem is reduced by one. It is reduced by d_i if we assign variable x_i.

Penalized formulation In this case we work with the penalized objective matrix:

$$B := \begin{pmatrix} (2\rho+1)F^\top F + R & \frac{1}{2}(e^\top - 2(2\rho+1)u^\top F)^\top \\ \frac{1}{2}(e^\top - 2(2\rho+1)u^\top F) & (2\rho+1)u^\top u \end{pmatrix}.$$

The update steps for matrix R are the same as previously. However, since we use a penalization of the exactly-one constraint, new terms appear in the objective matrix. We recall that $F \in \mathbb{R}^{n\times(d+1)}$ and $u \in \mathbb{R}^n$ are the left and right-hand side of the exactly-one constraint. Thus, when we remove a value from a domain, the matrix F and the vector u need to be updated.

1. Remove value a from D_i: for matrix F it is straightforward to see that removing value a is equivalent to removing column j of F. For the right-hand side u,
$$u_i \leftarrow \frac{3-d_i}{2}.$$
At the end, we have the update:
$$(F^\top u)_i \leftarrow (F^\top u)_i + \frac{1}{4}$$
$$u^\top u \leftarrow u^\top u + \frac{1}{4}(5-2d_i).$$

2. Assign $x_i = a$: in this case, row i of F and entry i of u are removed and the bottom right term of B is updated as follows:
$$u^\top u \leftarrow u^\top u - \frac{1}{4}(2-d_i)^2.$$

2.4.2 On proving convergence of LR-BCD

Theoretically proving the global convergence of LR-BCD to the solution of (2.16) is still an open question. We saw that in practice it can quickly converge to a solution with a small relative error compared to interior point solutions. However, for completeness, it would be interesting to show the global convergence of the method, or conversely, find pathological instances for which LR-BCD fails to converge. In this section, we give some hints towards a convergence proof.

Convergence of the iterates A first step to show convergence of LR-BCD is to show convergence of the sequence of iterates $\{V^i\}_{i\in\mathbb{N}}$ generated by the algorithm. In this regard, we introduce a lemma that allows us to show the convergence of the sequence of function values.

Lemma 2.4.1. *Convergence of the sequence $\{f(V)^i\}_{i\in\mathbb{N}}$. Let us consider the low-rank relaxation introduced previously:*

$$\begin{aligned} \min_{V\in\mathbb{R}^{(d+1)\times r}} \quad & f(V) \coloneqq \langle R, VV^\top\rangle \\ s.t \quad & \langle U_i, VV^\top\rangle = 2-d_i, \qquad \forall i=1,\dots,n \\ & \|V_i\| = 1, \qquad\qquad \forall i=1,\dots,d+1 \end{aligned} \tag{2.26}$$

Proof. Due to the norm constraint, it is straightforward to show that the constraint set of (2.26) is a compact set of $\mathbb{R}^{(d+1)\times r}$. Since f is continuous on a compact set it is bounded. We showed in Theorem 2.2.8 that the sequence $\{f(V)^i\}_{i\in\mathbb{N}}$ generated by LR-BCD is decreasing. Since it is decreasing and bounded, the sequence converges. □

In the proof of Theorem 2.2.8 we showed that the objective difference before and after an update of a row V_i is equal to

$$f(V_i) - f(\hat{V}_i) = -\frac{1}{2\beta_i}\|V_i - \hat{V}_i\|^2$$

with $\beta_i < 0$. Summing up the differences from 1 to d, an overall update gives

$$f(V) - f(\hat{V}) = \sum_{i=1}^{d} -\frac{1}{2\beta_i}\|V_i - \hat{V}_i\|^2.$$

Finally, lemma 2.4.1 ensures convergence of $\{f(V)^i\}_{i\in\mathbb{N}}$. Due to the compactness of the feasible set (2.26), the sequence of iterates $\{V^i\}_{i\in\mathbb{N}}$ has at least one cluster point which is a fixed point for the mapping we present below.

LR-BCD mapping A second step to show convergence of LR-BCD is to write the low-rank matrix update as a corresponding matrix mapping. We denote by

$$\mathcal{M} : \mathbb{R}^{(d+1)\times r} \to \mathbb{R}^{(d+1)\times r}$$

the mapping of LR-BCD, *i.e.*, $\hat{V} = \mathcal{M}(V)$. Since LR-BCD converges to a unique limit point $\bar{V}$, it must be a fixed point for the mapping $\bar{V} = \mathcal{M}(\bar{V})$. We now write the mapping in matrix form. Using Theorem 2.2.1, we know that there exist $\alpha_i, \beta_i \in \mathbb{R}$ such that $\hat{V}_i = \alpha_i V_{d+1} + \beta_i g_i$ with

$$g_i = \sum_{j<s_k} R_{ij}\hat{V}_j + \sum_{j\geqslant s_{k+1}} R_{ij}V_j.$$

We can write the update on the whole matrix as follows:

$$\hat{V} = D_\beta\left[L\hat{V} + L^\top V\right] + D_\alpha KV. \tag{2.27}$$

With:

- $L^\top \in \mathbb{R}^{(d+1)\times(d+1)}$ the upper triangular part of the objective matrix R.
- $L \in \mathbb{R}^{(d+1)\times(d+1)}$ the lower triangular part of R.
- $D_\beta \in \mathbb{R}^{(d+1)\times(d+1)}$ the diagonal matrix with entries $(\beta_i)_i$.
- $D_\alpha \in \mathbb{R}^{(d+1)\times(d+1)}$ the diagonal matrix with entries $(\alpha_i)_i$.
- $K = \begin{pmatrix} 0 & \dots & 0 & 1 \\ \vdots & 0 & \vdots & \vdots \\ 0 & \dots & 0 & 1 \end{pmatrix} \in \mathbb{R}^{(d+1)\times(d+1)}$.

Note that we fix $\beta_{d+1} = 0$ and $\alpha_{d+1} = 1$ since $V_{d+1} = V_{d+1} + 0.g_{d+1}$. The idea is now to regroup the updated terms:

$$\hat{V} = D_\beta L\hat{V} + D_\beta L^\top V + D_\alpha KV$$
$$(I_{d+1} - D_\beta L)\hat{V} = (D_\beta L^\top + D_\alpha K)V.$$

Since L is lower triangular, matrix $I_{d+1} - D_\beta L$ is nonsingular and we can write the mapping as follows:

$$\mathcal{M}(V) = \hat{V} = (I_{d+1} - D_\beta L)^{-1}(D_\beta L^\top + D_\alpha K)V \tag{2.28}$$

We can now characterize the fixed point of the mapping (2.28):

$$\mathcal{M}(\bar{V}) = \bar{V} \iff (D_\beta(L + L^\top) + D_\alpha K - I_{d+1})\hat{V} = 0,$$

or equivalently:

$$(D_\beta R + D_\alpha K - I_{d+1})\bar{V} = 0. \tag{2.29}$$

Let us consider the Moore-Penrose inverse of the matrix D_β:

$$D_\beta^+ = \begin{pmatrix} \frac{1}{\beta_1} & & & \\ & \ddots & & \\ & & \frac{1}{\beta_{d+1}} & \\ & & & 0 \end{pmatrix},$$

we can write the last equation as:

$$(D_\beta^+ D_\beta R + D_\beta^+ D_\alpha K - D_\beta^+)\bar{V} = 0. \tag{2.30}$$

Finally, we hope that this characterization will help us to further analyze the convergence properties of LR-BCD. In particular, it would be interesting to see if these points correspond to first-order critical points.

2.4.3 Strengthening the LR-BCD formulation

Along the experiments, we realized that the LR-BCD formulation was loose on certain kind of instances, especially on the sparse problems. As we already seen for LR-BCD, extending coordinate descent to general constraints is not straightforward. We were able to include the exactly-one constraints, but it's not clear whether it is possible to do the same with other types of constraints. Let us consider a pairwise GM $\mathcal{M} = \langle X, \Phi \rangle$ with $|X| = n$ variables and $d = \sum_{i=1}^{n} d_i$. We recall that LR-BCD works on the following SDP relaxation:

$$\begin{aligned} \min_{X \in \mathbb{R}^{(d+1)\times(d+1)}} \quad & \langle R, X \rangle \\ \text{s.t} \quad & \langle U_i, X \rangle = 2 - d_i, \quad \forall i = 1, \ldots, n \\ & \text{diag}(X) = 1_{d+1} \\ & X \succeq 0. \end{aligned} \tag{2.31}$$

On some Erdős–Rényi instances, we saw that the solution of (2.31) was negative. To overcome this issue, we can introduce a new set of constraints that we call the positivity constraints. These constraints are presented in detail later in the Subsection 3.1. Basically it means that for any pair of indices $(k, l) \in [d]^2$ corresponding to a non-zero binary cost in the objective matrix R, the variable X must enforce:

$$X_{kl} + X_{kd+1} + X_{d+1l} \geqslant -1. \tag{2.32}$$

These constraints are also referred to as triangular inequalities and will be later used in Chapter 4.

At this point, our idea is to find a method to strengthen the LR-BCD formulation while keeping the efficiency and scalability of LR-BCD. To do so, we thought about partial Lagrangian methods where additional constraints can be dualized. We denote by $\mathcal{C}$ the constraint set of (2.31) and $\mathcal{A} : \mathcal{S}^{d+1} \longrightarrow \mathbb{R}^m$ the linear operator for the constraints (2.32). We would like to use LR-BCD to solve the strengthened relaxation:

$$\begin{aligned} \min_{X \in \mathbb{R}^{(d+1)\times(d+1)}} \quad & \langle R, X \rangle \\ \text{s.t} \quad & \langle U_i, X \rangle = 2 - d_i, \quad \forall i = 1, \dots, n \\ & \mathcal{A}(X) \leqslant 1_m \\ & \operatorname{diag}(X) = 1_{d+1} \\ & X \succeq 0. \end{aligned} \tag{2.33}$$

We now introduce the partial Lagrangian where we only dualize the positivity constraints:

$$L(X, \lambda) = \langle R, X \rangle + \lambda^\top (\mathcal{A}(X) - 1_m), \tag{2.34}$$

with $\lambda \in \mathbb{R}^m_+$. Let p^* be the optimal value of the primal problem (2.33). By weak duality, we always have that

$$\begin{aligned} p^* &\geqslant \max_{\lambda \geqslant 0} \min_{X \in \mathcal{C}} \; -\lambda^\top 1_m + \langle R + \mathcal{A}^*(\lambda), X \rangle \\ &= \max_{\lambda \geqslant 0} \; -\lambda^\top 1_m + \min_{X \in \mathcal{C}} \langle R + \mathcal{A}^*(\lambda), X \rangle \end{aligned}$$

If strong duality holds the inequality is an equality. Let

$$f(\lambda) \coloneqq -\lambda^\top 1_m + \min_{X \in \mathcal{C}} \langle R + \mathcal{A}^*(\lambda), X \rangle.$$

The goal is to maximize f with regard to the variable $\lambda \in \mathbb{R}^m_+$. To do so, since f is concave but nonsmooth, we can use nonsmooth optimization methods like subgradient methods or the bundle method. An evaluation of the function and a subgradient at a certain point $\lambda \in \mathbb{R}^m_+$ are given by

$$\begin{aligned} f(\lambda) &= -\lambda^\top 1_m + \langle R + \mathcal{A}^*(\lambda), X^* \rangle \\ \partial f(\lambda) &= -1_m + \mathcal{A}(X^*). \end{aligned} \tag{2.35}$$

With

$$X^* = \arg\min_X \langle R + \mathcal{A}^*(\lambda), X \rangle, \text{ s.t } X \in \mathcal{C}. \tag{2.36}$$

We can then solve the minimization problem using LR-BCD and the maximization problem using the bundle method. The bundle method, used in combination with a low-rank solver, is discussed later in Chapter 4.

Chapter 3

Tighter bounds through constraints

Contents

3.1 A better representation of the feasible set 94
3.2 Resolution with ADMM 98
3.2.1 Primal ADMM 99
3.2.2 Recovering an integer solution 101
3.2.3 Constraint set projectors 101
3.3 Global constraints 104
3.4 Results 105
3.4.1 CPU-time 106
3.4.2 Bounds 106
3.4.3 Results on protein instances 107
3.4.4 Faster projection onto the semidefinite cone 108
3.5 Discussion 109

In the last chapter, we introduced two dedicated methods to solve two different relaxations of the MAP problem on discrete graphical models. We have seen that for certain kind of instances, the relaxation solved by LR-BCD can be loose. Sometimes, the solution of the SDP relaxation has a negative value, which is of no use since we know that the optimal value of the MAP problem is always non-negative. In this chapter, we overcome this issue by introducing new constraints for our SDP relaxation. An idea to obtain a tighter relaxation is to better approximate the feasible set of the MAP problem. Adding new constraints to our formulation comes at a certain cost, in this chapter we have chosen to trade efficiency for tightness of the bounds. Since it is unclear whether we can generalize LR-BCD to handle other constraints than the exactly-one constraint, we decided to explore other methods to solve the SDP relaxation. Since we are now introducing a potentially large number of constraints to represent our feasible set, interior point methods are computationally too expensive for the relaxation we are considering. Instead, we focused on first-order methods, in particular the Alternating Direction Method of Multipliers (ADMM). In general, these methods can take many iterations to converge to the exact optimum. However, they are very effective at returning good approximate solutions in a relatively short time compared to interior point methods. This chapter is organized in 4 parts:

- A better representation of the feasible set 3.1: we present the new constraints that we have decided to add to our relaxation. These constraints enjoy some properties that are useful for the resolution with ADMM.
- Resolution with ADMM 3.2: we solve the new SDP relaxation of the MAP problem with a variant of ADMM.
- Global constraints 3.3: we discuss the addition of global constraints, which may be interesting for modeling certain problems.
- Results 3.4: we compare ADMM with LR-LAS and LR-BCD on a set of crafted and real-world problems. For some of the instances we also include a comparison with the exact solver toulbar2. We also discuss the efficiency of iterative methods for computing the projection onto the semidefinite cone.

3.1 A better representation of the feasible set

Let us consider a pairwise GM $\mathcal{M} = \langle X, \Phi \rangle$ with $|X| = n$ discrete variables. Each discrete variable $x_i \in X$ has domain size d_i. We note $d = \sum_{i=1}^{n} d_i$ the total number of values. Let $b \in \{0,1\}^d$ be the stacked vector which represents the 1-hot or direct encoding of the variables, $b^\top = [b_1^\top \ \dots \ b_i^\top \ \dots \ b_n^\top]$, and $s_1 = 1, s_i = s_{i-1} + d_{i-1}, i = 2, \dots, n$ the indices of the domains in the stacked vector. To avoid any confusion with the variable indices, we use a superscript to denote the entry j of the vector

$b_i : b_i^j$. Using the results of Subsection 1.5.2, we have seen that the MAP problem on $\mathcal{M}$ can be cast as the following constrained binary quadratic problem:

$$\begin{aligned} \min_{b\in\{0,1\}^d} \quad & b^\top Q b + c^\top b \\ \text{s.t} \quad & Ab = 1_n, \end{aligned} \tag{3.1}$$

with Q and c the matrix and vector representations of the binary and unary potentials. We recall that the linear constraint $Ab = 1_n$ encodes the exactly-one constraints for each discrete variable. For LR-BCD and LR-LAS we used a change of variables to transform (3.1) into an equivalent quadratic problem in $\{-1, 1\}$ variables. For LR-LAS this was convenient because we could reduce it to a pure Max-Cut problem. For LR-BCD we were able to handle the resulting diagonal constraint efficiently. This time we focus on the $\{0, 1\}$ formulation. To derive the basic SDP relaxation of (3.1), we homogenize the problem with the extended vector $y^\top = [b^\top 1] \in \{0, 1\}^{d+1}$. For the exactly-one constraints we use the symmetric matrices

$$U_i = \begin{pmatrix} 0^{d\times d} & F_i \\ F_i^\top & 0 \end{pmatrix} \in \mathbb{R}^{(d+1)\times(d+1)},\ i = 1, \dots, n,$$

with $F_i \in \mathbb{R}^d, 2F_i^\top = [0_{d_1}^\top \ \cdots \ 0_{d_{i-1}}^\top \ 1_{d_i}^\top \ 0_{d_{i+1}}^\top \ \cdots \ 0_{d_n}^\top]$. Let

$$B = \begin{pmatrix} Q & \frac{1}{2}c \\ \frac{1}{2}c^\top & 0 \end{pmatrix},$$

the constrained binary quadratic problem is equivalent to:

$$\begin{aligned} \min_{y\in\{0,1\}^{d+1}} \quad & y^\top B y \\ \text{s.t} \quad & y^\top U_i y = 1,\ i = 1, \dots, n. \end{aligned} \tag{3.2}$$

Using the trace property, we reformulate the problem as:

$$\begin{aligned} \min_{y\in\{0,1\}^{d+1}} \quad & \mathrm{Tr}(B y y^\top) \\ \text{s.t} \quad & \mathrm{Tr}(U_i y y^\top) = 1,\ i = 1, \dots, n, \end{aligned} \tag{3.3}$$

We now introduce the rank one positive semidefinite matrix

$$Y = yy^\top = \begin{pmatrix} bb^\top & b \\ b^\top & 1 \end{pmatrix},$$

by dropping the rank one constraint we obtain the semidefinite relaxation:

$$\begin{aligned} \min \quad & \langle B, Y\rangle \\ \text{s.t} \quad & \langle U_i, Y\rangle = 1,\ i = 1, \dots, n \\ & Y \succeq 0. \end{aligned} \tag{3.4}$$

This relaxation equivalent to the ones solved by LR-LAS and LR-BCD.

Diagonal constraints From the equality $Y = yy^\top = \begin{pmatrix} bb^\top & b \\ b^\top & 1 \end{pmatrix}$, we can directly add a new set of constraints. Indeed, for $(i,j) \in [d]^2$, $(bb^\top)_{ij} = b^i b^j$. Since the $b^{i\prime}s$ are $\{0,1\}$ variables, ${b^i}^2 = b^i$. Thus for the diagonal entries of $bb^\top$, we have:

$$(bb^\top)_{ii} = {b^i}^2 = b^i, \ i = 1, \dots, d.$$

We can now introduce the corresponding symmetric constraint matrices:

$$D_i = \begin{pmatrix} E_{ii} & -\frac{1}{2}e_i \\ -\frac{1}{2}e_i^\top & 0 \end{pmatrix} \in \mathbb{R}^{(d+1)\times(d+1)}.$$

Matrix Y is now constrained by this new set of diagonal constraints:

$$\langle D_i, Y \rangle = 0, \ i = 1, \dots, d. \tag{3.5}$$

Due to homogenization, the bottom right entry of matrix Y is also constrained to be equal to 1, $Y_{d+1d+1} = 1$. These constraints are equivalent to the diagonal constraints in the $\{-1,1\}$ formulation.

Gangster constraints Because of the combinatorial structure of our problem, we can enforce some entries of the matrix Y to be equal to 0. Without loss of generality, let us consider the 1-hot or direct encoding of the first variable $b_1 \in \{0,1\}^{d_1}$. Because of the exactly-one constraint, only one entry of the vector b_1 must be equal to 0. This means that

$$b_1^i \times b_1^j = 0, \ i \neq j, \ (i,j) \in [d_1]^2,$$

since one of the two terms necessarily equals 0. All off-diagonal entries of the matrix $b_1 b_1^\top \in \mathbb{R}^{d_1 \times d_1}$ must be equal to 0. This result can be extended to all variables and thus, to all diagonal blocks of the matrix $bb^\top \in \mathbb{R}^{d \times d}$.

$$bb^\top = \begin{pmatrix} \begin{bmatrix} \ddots & 0 \\ 0 & \ddots \end{bmatrix} & & \\ & \ddots & \\ & & \begin{bmatrix} \ddots & 0 \\ 0 & \ddots \end{bmatrix} \end{pmatrix} \tag{3.6}$$

These constraints are known as gangster constraints and have been studied for many SDP relaxations of combinatorial problems [Zhao et al., 1998, Wolkowicz and Zhao, 1999]. We now define the index set:

$$\mathcal{J} \coloneqq \bigcup_{i=1}^{n} \left\{ (i,j) \in [s_i, s_i + d_i - 1]^2, i \neq j \right\}, \tag{3.7}$$

and the corresponding gangster operator:

$$\begin{array}{rccl} \mathcal{G}_{\mathcal{J}}: & \mathcal{S}^{d+1} & \longrightarrow & \mathcal{S}^{d+1} \\ & Y & \longmapsto & (\mathcal{G}_{\mathcal{J}}(Y))_{ij} \coloneqq \begin{cases} Y_{ij} & \text{if } (i,j) \text{ or } (j,i) \in \mathcal{J} \\ 0 & \text{otherwise} \end{cases} \end{array} \tag{3.8}$$

This operator is used to tighten our original relaxation (3.4).

Positivity constraint The joint function defined by the graphical model $\mathcal{M}$ is always non-negative, *i.e.*, it can always be bounded below with the trivial lower bound $lb = 0$. To be informative, it is important that the solution of our SDP relaxation remains non-negative. To ensure this property, we can constrain some entries of the matrix Y to be non-negative. Let $Y = yy^\top = \begin{pmatrix} bb^\top & b \\ b^\top & 1 \end{pmatrix}$, for any binary potential, $\theta_{ij} \in \Phi$, $\theta_i(x_i, x_j) \geqslant 0$. Let $Q_{ij} \in \mathbb{R}^{d_i \times d_j}$ be the matrix representation of the binary potential θ_{ij}, $b_i \in \mathbb{R}^{d_i}$ and $b_j \in \mathbb{R}^{d_j}$ the 1-hot encoding of variables x_i and x_j respectively. Then it must hold that $b_i^\top Q_{ij} b_j \geqslant 0$ or equivalently, $\mathrm{Tr}(Q_{ij} b_j b_i^\top) \geqslant 0$. Then we can enforce the block $b_j b_i^\top \in \mathbb{R}^{d_i \times d_j}$ to be non-negative and we can extend this result for all the following blocks:

$$b_j b_i^\top \geqslant 0, \ (i,j) \in [n]^2, \ \theta_{ij} \in \Phi. \tag{3.9}$$

Note that blocks for which $\theta_{ij} \notin \Phi$ do not need to ensure this property, in fact they do not appear in the objective function since the corresponding entries in B are equal to 0. Now let $c_i \in \mathbb{R}^{d_i}$ be the vector representation of the unary potential $\theta_i \in \Phi$, it must also hold that $c_i^\top b_i \geqslant 0$. However, we do not need to ensure that $b_i \geqslant 0$ since it is already implicitly satisfied by the diagonal constraints (3.1) and the positive semidefiniteness of Y:

$$Y \succeq 0 \implies Y_{ii} = Y_{i(d+1)} = Y_{(d+1)i} \geqslant 0.$$

Now, if we gather all these constraints, we have the following SDP relaxation for the MAP problem:

$$\begin{array}{rl} \min & \langle B, Y \rangle \\ \text{s.t} & \langle U_i, Y \rangle = 1, \ i = 1, \dots, n \\ & \langle D_i, Y \rangle = 0, \ i = 1, \dots, d \\ & \mathcal{G}_{\mathcal{J}}(Y) = 0 \\ & Y_{[ij]} \geqslant 0, \ \theta_{ij} \in \Phi \\ & Y_{(d+1)(d+1)} = 1 \\ & Y \succeq 0, \end{array} \tag{3.10}$$

where $Y_{[ij]}$ is the block matrix corresponding to the sub-matrix $b_i b_j^\top$. These constraints can be extended to $\{-1, 1\}$ variables for the LR-BCD formulation (2.16).

Indeed, let us consider the change of variable for the Boolean vector $x = 2b - 1_d$. For $(i, j) \in [n]^2, \theta_{ij} \in \Phi$:

$$b_i b_j^\top \geqslant 0 \iff \frac{1}{4}(x_i + 1_{d_i})(x_j + 1_{d_j})^\top \geqslant 0$$
$$x_i x_j^\top + x_i 1_{d_j}^\top + 1_{d_i} x_j^\top \geqslant -1_{d_i} 1_{d_j}^\top.$$

If we consider the matrix variable for the LR-BCD formulation $X = \begin{pmatrix} xx^\top & x \\ x^\top & 1 \end{pmatrix}$, it means that for any pair of indices $(k, l) \in [d]^2$ corresponding to a non-zero binary cost in the objective matrix we have

$$X_{kl} + X_{k(d+1)} + X_{(d+1)l} \geqslant -1. \tag{3.11}$$

Example 9. *Tightness of the bound on a small pairwise graphical model. In this example we discuss the relevance of the different constraints on a small MAP instance from the Cost Function Library*[1]*. We used the queens-5-5-4 instance from the* `crafted/coloring` *benchmark. This instance has* 100 *variables with domain size* 4 *and* $|\Phi| = 185$ *potentials. The SDP relaxations with the different constraint sets were solved using Mosek. The optimal value for this instance is* $p^* = 12$*. For the basic SDP relaxation, we include the exactly-one, the diagonal and the bottom right constraints.*

Optimal value	Basic	Gangster	Positivity
5.44	✓		
8.12	✓	✓	✓

Table 3.1: Optimal value of the SDP relaxation for different set of constraints.

In this section, we have introduced new valid constraints to tighten up our SDP relaxation of the MAP problem on pairwise GM. These constraints have an important property, apart from the diagonal and the exactly-one constraints, they do not apply to the same entries of the matrix Y. This is an important feature for the efficiency of the SDP method we have chosen, as the projections can be performed simultaneously.

3.2 Resolution with ADMM

We remind the reader that a brief introduction to ADMM in the context of SDPs was presented in Subsection 1.5.5. Unlike interior point methods, first-order methods are usually better at taking into account the structural properties of the problem [Wen et al., 2010]. Our SDP relaxation (3.10) has the additional constraint that some entries of the matrix Y must be non-negative. These SDPs are often referred to as Doubly Non-Negative programs (DNN) [Cerulli et al., 2021]. First, we tried to find an efficient method to solve these problems.

[1] https://forgemia.inra.fr/thomas.schiex/cost-function-library

3.2.1 Primal ADMM

During my one-month stay at Klagenfurt, the optimization team suggested the ADMM method presented in [de Meijer et al., 2023]. We like to call it *primal* ADMM because it splits the primal variable and the augmented Lagrangian is optimized with respect to the block of primal variables. We decided to use this method to solve the relaxation (3.10). We denote by $\mathcal{C}$ the constraint set of (3.10) without the cone constraint. We reformulate our original SDP as the equivalent problem:

$$\begin{aligned} \min_{Y,R} \ & \langle B, Y \rangle \\ \text{s.t} \ & Y \in \mathcal{C} \\ & R \succeq 0 \\ & Y = R. \end{aligned} \tag{3.12}$$

The augmented Lagrangian with respect to the equality constraint is given as follows:

$$L_\sigma(Y, R, Z) = \langle B, Y \rangle + \langle Z, Y - R \rangle + \frac{\sigma}{2} \|Y - R\|_F^2, \tag{3.13}$$

with Z the dual variable for the equality constraint and σ the penalty parameter. Then, ADMM performs a sequential optimization of the augmented Lagrangian with respect to the variables Y and R followed by an update of the dual variable Z:

$$R^{p+1} = \arg\min_{R \succeq 0} \ L_{\sigma^p}(R, Y^p, Z^p) \tag{a}$$

$$Y^{p+1} = \arg\min_{Y \in \mathcal{C}} \ L_{\sigma^p}(R^{p+1}, Y, Z^p) \tag{b}$$

$$Z^{p+1} = Z^p + \gamma \sigma^p (Y^{p+1} - R^{p+1}), \tag{c}$$

with R^p, Y^p and Z^p the variables at iteration $p \in \mathbb{N}$, σ^p the penalty parameter at iteration p and $\gamma \in \left(0, \frac{1+\sqrt{5}}{2}\right)$ a given stepsize. The two steps (a) and (b) are essentially projections onto the semidefinite cone and the convex set $\mathcal{C}$. For the two minimization problems, the linear terms can be combined together with the squared norms:

$$\begin{aligned} (a)\ R^{p+1} &= \arg\min_{R \succeq 0} \ \langle Z^p, Y^p - R \rangle + \frac{\sigma^p}{2} \|Y^p - R\|_F^2 \\ &= \arg\min_{R \succeq 0} \ \frac{\sigma^p}{2} \left(\|Y^p - R\|_F^2 + \frac{2}{\sigma^p} \langle Z^p, Y^p - R \rangle + \frac{1}{(\sigma^p)^2} \|Z^p\|_F^2 \right) \\ &= \arg\min_{R \succeq 0} \ \left\| \left(Y^p + \frac{1}{\sigma^p} Z^p \right) - R \right\|_F^2 \\ &= \left(Y^p + \frac{1}{\sigma^p} Z^p \right)_{\succeq 0}. \end{aligned} \tag{3.14}$$

Similarly for the second step:

$$\begin{aligned}(b)\ Y^{p+1} &= \underset{Y\in\mathcal{C}}{\arg\min}\ \langle B + Z^p, Y\rangle + \frac{\sigma^p}{2}\|Y - R^{p+1}\|_F^2 \\ &= \underset{Y\in\mathcal{C}}{\arg\min}\ \left\|Y - \left(R^{p+1} - \frac{1}{\sigma^p}(B + Z^p)\right)\right\|_F^2 \\ &= \mathcal{P}_{\mathcal{C}}\left(R^{p+1} - \frac{1}{\sigma^p}(B + Z^p)\right),\end{aligned} \tag{3.15}$$

with $\mathcal{P}_{\mathcal{C}}$ the orthogonal projection onto the feasible set $\mathcal{C}$.

Stopping criteria and stepsize For both the stopping criteria and the stepsize, we used the ones suggested by [de Meijer et al., 2023]. Given a certain tolerance $\varepsilon_{\text{ADMM}}$, ADMM is stopped when

$$\max\left\{\frac{\|Y^p - R^p\|_F}{1 + \|Y^p\|_F},\ \sigma^p\frac{\|Y^{p+1} - Y^p\|_F}{1 + \|Z^p\|_F}\right\} < \varepsilon_{\text{ADMM}}. \tag{3.16}$$

Note that these two terms are strictly equivalent to the primal and dual residuals introduced in the original ADMM [Boyd et al., 2011]. As for the step size $\gamma\sigma^p$, it can either be fixed or be updated along the iterations. We have found that an adaptive stepsize can greatly improve CPU time on the instances we have tested.

Dual feasible solution Compared with the two low-rank solvers, ADMM allows for computing bounds with guarantees. At the end of the ADMM procedure, we can compute a dual feasible point using the solution returned by the algorithm. This result was introduced in the paper [Li et al., 2021] and used in [de Meijer et al., 2023]. Let $L(Y, R, Z) \coloneqq L_0(Y, R, Z)$ be the Lagrangian for our problem (3.12), the Lagrangian dual is given by

$$\begin{aligned}g(Y, R, Z) &\coloneqq \max_{Z\in\mathcal{S}^{d+1}}\ \min_{Y\in\mathcal{C},\ R\succeq 0} L(Y, R, Z) \\ &= \max_{Z\in\mathcal{S}^{d+1}}\left\{\min_{Y\in\mathcal{C}}\ \langle B + Z, Y\rangle + \min_{R\succeq 0}\ \langle Z, R\rangle\right\} \\ &= \max_{Z\in\mathcal{S}^{d+1}}\left\{\min_{Y\in\mathcal{C}}\ \langle B + Z, Y\rangle - \operatorname{Tr}(R)\lambda_{\max}(Z)\right\},\end{aligned} \tag{3.17}$$

with $\lambda_{\max}(Z)$ the maximum eigenvalue of Z. The last step follows from the Rayleigh Principle. Therefore, for any feasible $Z \in \mathcal{S}^{d+1}$, the solution to

$$\min_{Y\in\mathcal{C}}\ \langle B + Z, Y\rangle - \operatorname{Tr}(R)\lambda_{\max}(Z) \tag{3.18}$$

yields a valid lower bound for the primal problem (3.12). This problem turns out to be an LP that can be efficiently solved. For the experiments, we used Mosek [ApS, 2022] to solve it.

3.2.2 Recovering an integer solution

For this work, we are also interested in heuristics to find good feasible integer solutions. We can not apply a method similar to the Goemans and Williamson heuristic. Instead, we directly use the entries of the approximate solution of (3.10). Let $Y^* = \begin{pmatrix} W^* & b^* \\ (b^*)^\top & 1 \end{pmatrix}$ be the solution returned by ADMM where W^* is the top-left submatrix of Y^* of size $(d \times d)$. We use the vector $b^* \in \mathbb{R}^d$ to compute our integer solution. We assign each GM variable to the value $x_k = \left(\arg\max_{s_k \leqslant i \leqslant s_{k+1}} (b^*)^i\right) - s_k$. This rounding heuristic is followed by a greedy search as the one we have presented for LR-BCD and LR-LAS, see Subsection 2.2.6.

3.2.3 Constraint set projectors

The goal of this subsection is to derive the projectors onto the different feasible sets defined by the constraints. These projections are needed for the step (b) of the ADMM algorithm. It is clear that all the feasible sets are nonempty closed convex subsets of $\mathbb{R}^{(d+1)\times(d+1)}$. Thus, by the Hilbert projection theorem [Rudin, 1991], we have existence and uniqueness of the projections.

Exactly-one projection Without loss of generality, let us consider the first exactly-one constraint, *i.e.*, we focus on the first d_1 entries of the vector b. Let $M = \begin{pmatrix} bb^\top & b \\ b^\top & 1 \end{pmatrix}$, we aim to find the unique solution to the minimization problem:

$$\begin{aligned} \min_Y \quad & \|Y - M\|_F^2 \\ \text{s.t} \quad & \langle U_1, Y \rangle = 1. \end{aligned} \tag{3.19}$$

If M is already feasible, then we have the trivial solution $Y = M$. Otherwise we work only with the constrained entries of Y. In fact, we can set the others to the corresponding entries of M. We rewrite the optimization problem (3.19):

$$\begin{aligned} \min_Y \quad & 2\sum_{i=1}^{d_1} (Y_{i(d+1)} - M_{i(d+1)})^2 \\ \text{s.t} \quad & \sum_{i=1}^{d_1} Y_{i(d+1)} = 1, \end{aligned} \tag{3.20}$$

with the associated Lagrangian:

$$L(\lambda, Y) = 2\sum_{i=1}^{d_1} (Y_{i(d+1)} - M_{i(d+1)})^2 + \lambda(\sum_{i=1}^{d_1} Y_{i(d+1)} - 1). \tag{3.21}$$

Since the problem is convex, KKT conditions are necessary and sufficient for

optimality. Therefore, we have to solve the following system of equations:

$$\begin{cases} \frac{\partial L}{\partial Y_{i(d+1)}}(\lambda, Y) = 4(Y_{i(d+1)} - M_{i(d+1)}) + \lambda = 0, \ i = 1, \dots, d_1 \\ \sum_{i=1}^{d_1} Y_{i(d+1)} = 1 \end{cases} \tag{3.22}$$

From the stationnarity conditions, we get:

$$Y_{i(d+1)} - M_{i(d+1)} = -\frac{\lambda}{4} \iff Y_{i(d+1)} = M_{i(d+1)} - \frac{\lambda}{4}.$$

Plugging it back in the constraint:

$$\sum_{i=1}^{d_1} \left(M_{i(d+1)} - \frac{\lambda}{4} \right) = 1$$

$$\lambda = \frac{4}{d_1} \Big(\sum_{i=1}^{d_1} M_{i(d+1)} - 1 \Big).$$

We can already see that λ is related to the violation of the exactly-one constraint for the entries of the matrix M. Finally, since $4(Y_{i(d+1)} - M_{i(d+1)}) + \lambda = 0$:

$$\begin{aligned} Y_{i(d+1)} - M_{i(d+1)} + \frac{1}{d_1} \Big(\sum_{i=1}^{d_1} M_{i(d+1)} - 1 \Big) = 0 \\ Y_{i(d+1)} = M_{i(d+1)} - \frac{1}{d_1} \Big(\sum_{i=1}^{d_1} M_{i(d+1)} - 1 \Big). \end{aligned} \tag{3.23}$$

Intuitively, we see that the projection of M onto the feasible set is equivalent to removing $\frac{1}{d_1}$ times the exactly one violation of all corresponding entries. Let δ be the exactly-one constraint violation, the distance from the solution $\|Y^* - M\|_F$ is equal to $\frac{\sqrt{2}|\delta|}{\sqrt{d_1}}$.

Diagonal projection As mentioned before, the diagonal constraints and the exactly-one constraints are the only constraints that share overlapping entries of the matrix Y. First, we focus only on the diagonal constraints. Let $M = \begin{pmatrix} bb^\top & b \\ b^\top & 1 \end{pmatrix}$, We aim to find the unique solution to the minimization problem:

$$\begin{aligned} \min_Y \ & \|Y - M\|_F^2 \\ \text{s.t} \ & \langle D_i, Y \rangle = 0. \end{aligned} \tag{3.24}$$

In the same way as for the exactly-one constraint, if M is already feasible, we have the trivial solution $Y = M$. Otherwise, we rewrite the optimization problem (3.24):

$$\begin{aligned} \min_Y \ & 2(Y_{i(d+1)} - M_{i(d+1)})^2 + (Y_{ii} - M_{ii})^2 \\ \text{s.t} \ & Y_{i(d+1)} = Y_{ii}, \end{aligned} \tag{3.25}$$

Or equivalently,

$$\min f(Y_{id+1}) \coloneqq 2(Y_{i(d+1)} - M_{i(d+1)})^2 + (Y_{i(d+1)} - M_{ii})^2 \tag{3.26}$$

At the optimum, the solution must satisfy $f'(Y_{i(d+1)}) = 0$, *i.e.*,

$$\begin{aligned} 6Y_{i(d+1)} - 4M_{i(d+1)} - 2M_{ii} &= 0 \\ Y_{i(d+1)} &= \frac{2}{3}M_{i(d+1)} + \frac{1}{2}M_{ii}. \end{aligned} \tag{3.27}$$

However, for computational efficiency, the exactly-one and diagonal constraints can be performed simultaneously. Without loss of generality, let us consider the first exactly-one constraint, the solution to the following optimization problem:

$$\begin{aligned} \min_{Y} \quad & \|Y - M\|_F^2 \\ \text{s.t} \quad & \langle U_1, Y \rangle = 1 \\ & \langle D_i, Y \rangle = 0, \ i = 1, \dots d_1, \end{aligned} \tag{3.28}$$

is given by:

$$Y_{i(d+1)} = \frac{2}{3}M_{i(d+1)} + \frac{1}{3}M_{ii} - \frac{1}{d_1}\left(\sum_{i=1}^{d_1}\left(\frac{2}{3}M_{i(d+1)} + \frac{1}{3}M_{ii}\right) - 1\right), \ i = 1, \dots, d_1. \tag{3.29}$$

See Appendix B for a complete proof of this result.

Gangster and positivity projections These two projections are easily computed by either:

- Applying the gangster operator to the matrix Y to set the corresponding entries to 0.
- Setting the negative entries within the blocks $Y_{[ij]}$, $\theta_{ij} \in \Phi$ to 0.

Projection onto the semidefinite cone This step is the most computationally intensive of the whole ADMM algorithm. Let us consider a symmetric matrix $M \in \mathcal{S}^n$, by theorem 1.5.2, the projection of M onto the semidefinite cone is given by:

$$M_{\succeq 0} = \sum_{i=1}^{d+1} \max\{0, \lambda_i\} u_i u_i^\top, \tag{3.30}$$

where $M = \sum_{i=1}^{d+1} \lambda_i u_i u_i^\top$ is the spectral decomposition of M. Without any additional assumptions on the structure of the matrix M, computing the set of eigenvalues and eigenvectors is cubic in the size of the matrix $O((d+1)^3)$. For our implementation, we used the Eigen3 [Guennebaud et al., 2010] linear algebra library. The implementation of their eigenvalue solver is based on a symmetric QR algorithm [Golub

and Van Loan, 2013]. They claim that the computational cost is about $9d^3$ when eigenvalues and eigenvectors are required. This can quickly become a barrier to scalability.

It is possible to reduce the total number of operations. Indeed, for the projection onto the semidefinite cone, we are only interested in computing the positive eigenpairs of M. Depending on the proportion of positive or negative eigenvalues, it is more efficient to compute only one of the two subsets. If we compute the negative eigenpairs, we can recover the projection by applying the simple operation:

$$M_{\succeq 0} = M - M_{\preceq 0}, \tag{3.31}$$

with

$$M_{\preceq 0} = \sum_{i=1}^{d+1} \min\{0, \lambda_i\} u_i u_i^\top. \tag{3.32}$$

When only a subset of eigenpairs is required, one can use iterative methods. The two papers [Goulart et al., 2020, Rontsis et al., 2022] explore approximate methods for the projection onto the semidefinite cone. In the context of ADMM, it is known that the algorithm still converges when the projection errors are summable, *i.e.*, when the sum of the projection errors over all the ADMM iterations is bounded [Eckstein and Bertsekas, 1992]. The first paper gives sharp bounds on the projection errors so the ADMM algorithm maintains its convergence properties. Based on these results, the second paper provides an efficient iterative method to approximately compute a subset of eigenpairs. It relies on a block Krylov method [Knyazev, 2001] with a $O((d+1)^2 m)$ per-iteration cost where m is the number of computed eigenpairs. The authors suggest that this method is preferable when ADMM converges to a solution with a small proportion of negative or positive eigenpairs. Otherwise, the usual QR algorithm is more efficient in terms of number of operations.

We have presented the projections onto the different feasible sets for our SDP relaxation (3.10). For the constraint set $\mathcal{C}$, considering that the exactly-one and diagonal constraints can be performed simultaneously, all constraints apply to non-overlapping entries of the primal matrix Y. In the end we have a closed form solution for the projection into $\mathcal{C}$ so the second step of the ADMM algorithm (b) can be computed efficiently.

3.3 Global constraints

Some of the problems encountered in graphical models include global constraints, *i.e.*, a constraint involving all or an unbounded number of variables. Let us consider a discrete pairwise GM $\mathcal{M} = \langle X, \Phi \rangle$ with $|X| = n$ variables. We use the same notations as in Section 3.1.

All different We denote by d_{max} the maximum domain size $d_{max} := \max_{i \in [n]} d_i$. To satisfy the all different constraint, we must enforce all variables of the collection X

to take distinct values. The constraint can be extended to subsets of variables. To enforce this global constraint, we need d_{max} linear constraints. Let $A_i \in \{0,1\}^d$ be the boolean vector which take value 1 at position i for each variable within the stacked vector b and 0 otherwise. To enforce the fact that at most one variable will take value i we can use the linear constraint $A_i^T b \leqslant 1$. Thus we can encode the all different constraint as

$$A_i^T b \leqslant 1,\ i = 0, \ldots, d_{max}.$$

k-values Using the same notations, to satisfy the k-values constraint, the variables of the collection X must take at most k distinct values. To do so, we will use the vectors A_i again. We introduce d_{max} new Boolean variables $y_i \in \{0,1\}$. The fact that a value i is used by one of the variables can be encoded by the constraint

$$\frac{A_i^T x}{n} \leqslant y_i.$$

Indeed, the fraction $\frac{A_i^T x}{n}$ is always upper bounded by 1. It is upper bounded by 0 if and only if none of the variables takes value i. We can then add the k-values constraint by enforcing the linear constraint

$$\sum_{i=1}^{d_{max}} y_i \leqslant k.$$

Finally we relax the Boolean constraint on variables y_i by using the quadratic constraint

$$y_i^2 = y_i,\ i = 1, \ldots, d_{max}.$$

3.4 Results

In this subsection, we compared the three SDP methods on GMs with Erdős–Rényi (ER) graphs. The GMs were generated using the same method as in Subsection 2.3.3. We also compared our results with the latest version of the exact solver toulbar2 available at `https://github.com/toulbar2/toulbar2`. Toulbar2 was run with default parameters and a time limit of 1 hour. For all figures presented in this section, for each point, results are averaged over 10 instances. We observed that the standard deviations of the measures were very low on the 10 generated samples and we therefore do not report them. This section bears witness to the trade-off between method efficiency and the quality of the bounds.

The two solvers [Huang et al., 2014, Wang et al., 2016] introduced in the background section also use SDP relaxations to tackle the MAP problem on pairwise discrete GMs. However, we could not find an implementation of these two methods, so we could not compare them with our solvers. These two solvers use constraint sets that are similar to the one we used in our relaxation (3.10).

All experiments were run on a single thread on a server equipped with a Xeon®Gold 6248R CPU@3.00GHz and 1TB of RAM running Debian Linux 4.19.98-1.

3.4.1 CPU-time

First, we compared the CPU times of the three SDP solvers on ER instances with different probability parameters and sizes. The exact solver toulbar2 was able to find the optimum for some of the smallest instances with probability parameter $p = 0.1$ and reached the time limit for all other instances. Therefore, we have not reported the time results for toulbar2. Note that the CPU time for ADMM also includes the time to solve the LP for the dual feasible point. However, in all the experiments we have done, this time is negligible, being less than 1% of the total CPU time. Figure 3.1 shows that the low-rank solvers are faster than ADMM, from one order of magnitude for LR-LAS to two order of magnitude for LR-BCD. We also found that ADMM tends to be faster as the instance density increases, which is not the case for the low-rank methods.

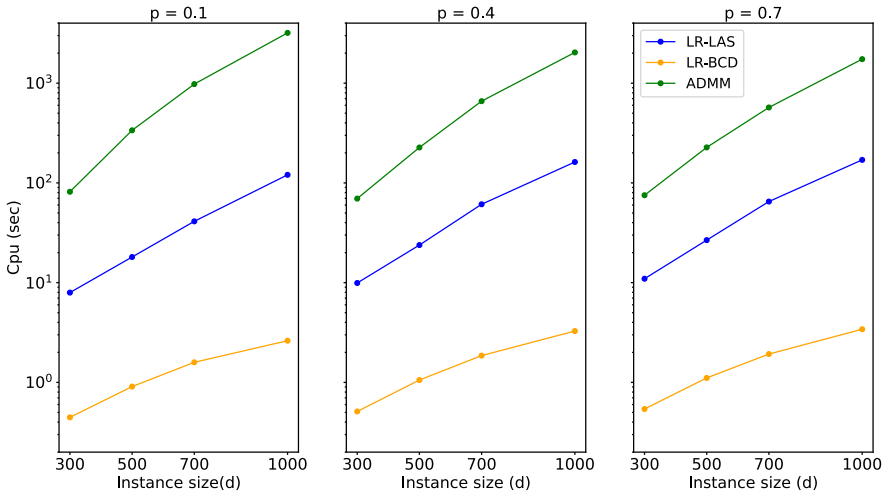


Figure 3.1: Cpu-time for the three SDP solvers on Erdős–Rényi instances with varying edge probability as a function of the problem size (number of states × number of variables).

3.4.2 Bounds

We now focus on the bounds returned by the four solvers. As mentioned earlier, ADMM has the advantage of returning dual bounds with guarantees, which is not the case for LR-BCD and LR-LAS. We recall that by using rounding and greedy search heuristics, we can produce integer feasible solutions for the three SDP methods. These integer solutions serve as upper bounds for our experiments. Note that when the exact solver toulbar2 reached the time limit, we used the global lower bound and the best integer solution found so far. Figure 3.2 shows the normalized upper

and lower bounds on ER instances with different edge probabilities and domain sizes. All considered instances have $n = 100$ discrete variables. While being much faster than the other methods, the lower bounds returned by LR-BCD collapse as the domain sizes increase. The bounds are also loose for instances with a small edge probability, being all negatives for $p = 0.1$. ADMM has the strongest lower bounds among the four solvers. This clearly shows the benefit of adding new constraints compared to the LR-BCD and LR-LAS formulations. Overall, the integer solutions are fairly similar between the 4 methods. Note that for certain instances with $p = 0.1$, toulbar2 was able to find the optimum within the 1 hour time limit. This means that for SDP solvers, the integer solution computed by the heuristics may be close to the true optimum.

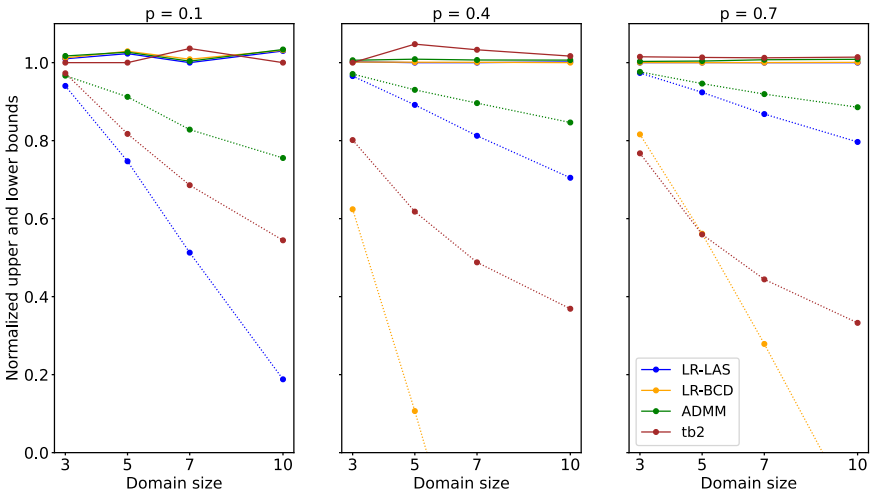


Figure 3.2: Normalized upper and lower bounds for all four solvers on Erdős–Rényi instances with varying edge probability as a function of domain size. The best integer primal solution value is taken as a reference. All considered instances have $n = 100$ discrete variables.

3.4.3 Results on protein instances

In this subsection, we tried our implementation of ADMM on small proteins from Marianne Defresne's benchmark, see Paragraph 2.3.4. These instances are dense, the GMs considered have a non-zero binary potential for each pair of variables. We denote by n the size of the matrix and by m the total number of constraints in the SDP relaxation. The optimality gap is computed by using the integer solution and the dual lower bound returned by the algorithm. We denote by f_{int} the value of the

Instance	n	m	Cpu (sec)	Optimality gap (%)
1a32	1701	2890086	5733.68	13.45
1aal	1141	1299658	1200.30	10.02
1aaz	1741	3027688	6261.98	15.09
1ail	1401	1960071	2814.56	11.35
1b6q	1101	1210056	1197.88	9.80
1b35	1141	1299658	1150.23	12.26
1bcc	1321	1742467	2120.37	10.88
1bct	1381	1904470	2351.06	12.87
1beo	1961	3841699	12091.91	16.56
1bf0	1201	1440061	1626.74	8.49
1bha	1361	1849669	3240.10	5.83
1bun	1221	1488462	1535.25	10.54
1cxz	1721	2958487	6143.69	16.85
1cyu	1961	3841699	12021.10	15.51
1d0d	1221	1488462	1581.34	9.15
1d1r	1661	2755684	5554.83	12.17
1d2d	1121	1254457	1414.42	10.05

Table 3.2: ADMM results on small protein instances.

integer solution and lb the dual lower bound,

$$\text{Optimality gap} = \frac{|f_{\text{int}} - lb|}{|f_{\text{int}}|} \times 100.$$

The results are given in Table 3.4.3. Even on the smallest instances, the commercial solver Mosek ran out of memory. For both LR-LAS and LR-BCD, we reported negative lower bounds on these instances.

3.4.4 Faster projection onto the semidefinite cone

After a certain number of iterations, if we find that either the proportion of positive or negative eigenvalues is relatively small, we switch to a partial eigenvalue solver. To this end, we used Spectra `https://spectralib.org/doc/`, a header-only library built on top of Eigen. Spectra is an eigen solver inspired by the software ARPACK [Lehoucq et al., 1998] written in FORTRAN. ARPACK stands for ARnoldi PACKage and relies on the Implicitly Restarted Arnoldi Method proposed in [Lehoucq and Sorensen, 1996]. We recall that the Arnoldi method has a $O((d+1)^2 m)$ per-iteration cost where m is the number of computed eigenpairs. Figure 3.3 compares the Cpu times of two implementations:

- first implementation: at each ADMM iteration, compute the full eigen decomposition with a direct solver.
- Second implementation: if the proportion of positive or negative eigenvalues is less than a specified threshold, switch to a partial eigen decomposition.

The first experiment was run on a ER instance with edge probability $p = 0.1$. We set the threshold to 15% of the matrix size. In Figure 3.3, we can clearly see when

the second method switches from full to partial eigen decomposition. In the end, the second implementation was 5.4 times faster to reach the same solution.

As we already discussed in Subsection 3.2.3, using an iterative eigen solver may not be a good option when the number of eigenpairs to compute is large. For another ER instance we pushed the threshold to 25%, the results are presented in Figure 3.4. This time, the second implementation was 3.6 times slower to reach the same solution.

Finally, it might be interesting to use an iterative eigenvalue solver if the considered SDP has a low-rank solution. Indeed, the null space of the matrix is then large and there is a small number of non-zero eigenvalues to compute. These low-rank solutions appear, for example, in the SDP relaxation of MaxCut.

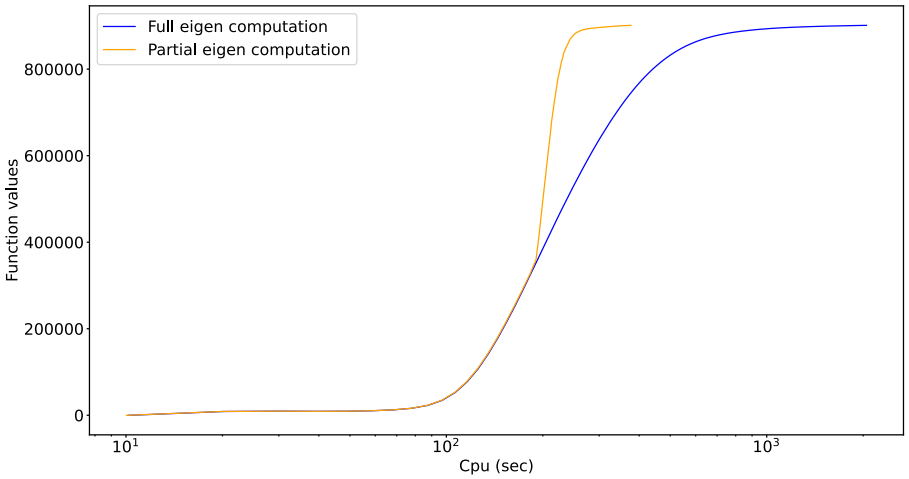


Figure 3.3: Function values against Cpu time (sec) on a Erdős–Rényi instance with edge probability $p = 0.1$. The instance has $n = 300$ variables with domain sizes 3 ($d = 900$). The time axis is on a logarithmic scale.

3.5 Discussion

In this chapter, we tried to trade efficiency for quality of the bounds by reinforcing the relaxation we considered in Chapter 2. By adding the gangster and positivity constraints, the bounds are now much tighter on a variety of different problems. Since the number of constraints can become quite large, interior point methods are not suitable for dealing with these SDP relaxations [Wiegele and Zhao, 2022]. Therefore, we used an ADMM algorithm which can efficiently handle our set of constraints. However, ADMM is still limited by calls to eigen solvers for projections onto the semidefinite cone. As shown in the experiments with the protein problems,

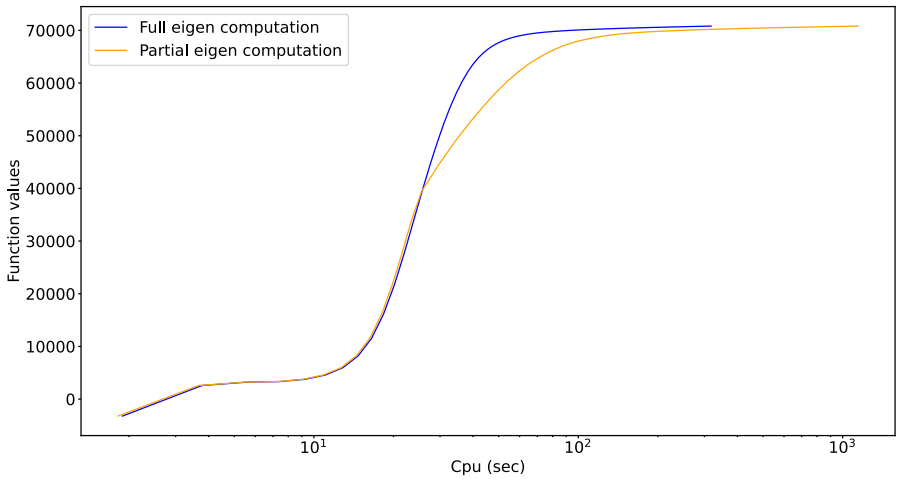


Figure 3.4: Function values against Cpu time (sec) on a Erdős–Rényi instance with edge probability $p = 0.1$. The instance has $n = 100$ variables with domain sizes 5 ($d = 500$). The time axis is on a logarithmic scale.

instances with $d > 1000$ become hard for ADMM. We discussed the usage of iterative eigen solvers to accelerate the projections but it is not suited for all kind of instances. Finally, we hope to combine low-rank methods with ADMM to get the best of both worlds.

Chapter 4

Solving MaxCut exactly with low rank SDP bounds

Contents

4.1 Introduction . . . 112
4.2 Related work and our contributions . . . 113
4.3 Background . . . 114
4.3.1 Semidefinite relaxations for MaxCut . . . 114
4.3.2 Other exact solvers based on semidefinite programming . . . 116
4.4 Bounding procedure . . . 118
4.4.1 Combining the mixing method with the proximal bundle method 118
4.4.2 Safe upper bounds . . . 119
4.5 Branch and cut . . . 120
4.5.1 Bounding routine . . . 120
4.5.2 Branching rules . . . 121
4.5.3 Problem reformulation and warm start . . . 122
4.6 Biased hyperplane . . . 124
4.7 Implementation . . . 125
4.8 Experiments . . . 127
4.9 Discussion . . . 134

This work is a joint work with Jan Schwiddessen. Jan is a PhD student whom I met during my stay at the Mathematical Institute of Klagenfurt. We were both interested in low-rank methods for solving SDPs. After some discussion, Jan proposed the idea of developing a new solver and the journey began. It was a great pleasure to collaborate with him on this project, and we plan to submit a paper before the end of this year.

In this chapter, we are now interested in solving combinatorial problems to optimality. We present `MixCut` an exact solver for MaxCut based on semidefinite programming. It uses a branch-and-cut approach with hypermetric inequalities to strengthen the basic MaxCut relaxation. The bounding procedure is efficiently carried out by a low-rank SDP algorithm, the mixing method [Wang et al., 2017]. We propose a set of new branching decisions based on information from both primal and dual solutions as well as a new heuristic to compute good integer solutions. Experimentally, we show that our method is able to reach a better trade-off between the tightness of the bounds and the total number of explored nodes in the search tree. We demonstrate our results against state-of-the-art algorithms by means of an in-depth study on dense MaxCut instances from the BiqMac library [Wiegele, 2007].

4.1 Introduction

MaxCut is a central problem in combinatorial optimization. Given a weighted graph, it aims to find a partition of the vertices into two disjoint sets such that the sum of the edges between the sets is maximized. It gained a lot of attention in the past decades due to its broad range of applications in physics and computer science. In the previous chapters, we saw that MAP inference on discrete GMs can be reduced to a pure MaxCut problem by means of Lasserre's exact penalization [Lasserre, 2016]. MaxCut is known to be NP-hard [Karp, 2010], and researchers have tried various methods to solve it. These include algorithms based on LP relaxations, which are more efficient on sparse problems. Goemans and Williamson [1995] gave a celebrated 0.87856 approximation ratio for graphs with nonnegative weights using an SDP relaxation. Nowadays, for dense instances, the best-performing solvers almost all use semidefinite relaxations. Even though those methods showed great results, solving MaxCut to optimality is still challenging even for problems of moderate size. Although not covered in depth in the manuscript, heuristics have been developed to find good solutions for large-scale problems [Benlic and Hao, 2013]. Recently, MaxCut has seen a resurgence of interest with quantum computers and the Quantum Approximate Optimization Algorithm (QAOA) [Farhi et al., 2014, Guerreschi and Matsuura, 2019].

4.2 Related work and our contributions

In the early 2000s, Burer and Monteiro introduced the idea of using a low-rank factorization to solve semidefinite programs [Burer and Monteiro, 2003]. This method quickly showed great experimental results. Indeed, it has made it possible to approximately solve large-scale structured SDPs several orders of magnitude faster than the usual primal-dual interior-point methods. The factorization significantly reduces the number of variables and the cone constraint is implicitly satisfied. However, it comes at the cost of losing convexity. Recently, Boumal *et al.* [Boumal et al., 2016] showed that non-convexity is not an issue as long as some assumptions are satisfied. Several methods were developed to efficiently solve those low-rank problems. For example, we can cite the Riemannian trust-region method (RTR) [Absil et al., 2007], or the mixing method [Wang et al., 2017] which is a coordinate descent algorithm. Goemans and Williamson showed that MaxCut has a natural rank-1 SDP relaxation for which they derived the approximation ratio. In addition to high efficiency, this motivated us to use a low-rank method for our solver's bounding procedure.

Due to its large number of applications in different fields, MaxCut is regarded as a central problem in combinatorial optimization. One line of work is to find strong relaxations in order to obtain the best possible approximation of the problem. Such relaxations can be derived from the moment-SOS hierarchy [Lasserre, 2001]. While offering strong convergence guarantees towards the optimizer of MaxCut, going to higher orders of the hierarchy is usually too computationally expensive. Recent works overcome this issue by using partial lifting to higher orders to obtain tight bounds on MaxCut [Wang et al., 2022, Campos et al., 2022]. Together with LP-based methods, they perform well on sparse instances. For example, the recent LP solver McSparse [Charfreitag et al., 2022] starts to show its limitations for graphs with a density greater than 10%.

SDP solvers have shown their superiority on dense problems. Tighter bounds are typically provided, making the branch and bound procedure more efficient. The top performing solvers all include hypermetric inequalities [Deza and Laurent, 2010]. BiqMac [Rendl et al., 2010] and BiqCrunch [Krislock et al., 2017] use triangular inequalities while BiqBin [Gusmeroli et al., 2022] and MADAM [Hrga and Povh, 2021] add pentagonal and heptagonal inequalities. These solvers essentially differ in the method used to solve the SDP relaxation at each node. Basically, there is always a trade-off between the quality of the bounding procedure and the time spent to solve the underlying optimization problem. In this respect, we have explored low-rank methods that can find good-quality solutions orders of magnitude faster than the usual primal-dual interior point methods.

The main contribution of our work is the introduction of the exact solver `MixCut` for solving MaxCut. It is based on the mixing method, a recent coordinate ascent method introduced in [Wang et al., 2017]. This chapter is organized in six parts:

- Background 4.3: we present the basic SDP relaxation for the MaxCut problem. Additionally, we incorporate a subset of hypermetric inequalities, namely

triangular inequalities, to strengthen the basic relaxation. Then, we introduce the current state-of-the-art solvers based on SDP approaches.

- Bounding procedure 4.4: we combine the mixing method with a proximal bundle approach to efficiently solve the strengthened SDP relaxation of MaxCut.
- Branch and cut 4.5: we develop a new branch and cut algorithm for MaxCut that exploits, among other things, new branching decisions, warm-starting and cut updates.
- Biased hyperplane 4.6: we propose a rounding procedure inspired by the renowned Goemans and Williamson approach. This procedure rounds the low-rank solution of the primal problem into feasible solutions, effectively yielding a valid lower bound for the optimal value of the MaxCut problem.
- Implementation 4.7: we discuss some of the implementation details regarding our solver.
- Experiments 4.8: we test `MixCut` against BiqCrunch, BiqBin and MADAM on several MaxCut instances from the BiqMac library. We carried out experiments on dense and sparse problems to demonstrate the flexibility of our solver in handling different types of instances.

4.3 Background

In this section we give a quick overview of the usual SDP relaxations for MaxCut. In addition, we introduce a low-rank, non-convex relaxation that can be solved efficiently using a coordinate descent algorithm. We then present the existing exact SDP solvers with their different features.

4.3.1 Semidefinite relaxations for MaxCut

In this subsection, we use the notations introduced in Subsection 1.2.2. Let $G = (V, E)$ be a weighted undirected graph with $n = |V|$ vertices and $m = |E|$ edges. Let $A \in \mathbb{S}^n$ be the adjacency matrix of G, *i.e.*, $A_{ij} = w_{ij}$ if $(i, j) \in E$ and 0 otherwise. We also define the Laplacian of G

$$L = \mathrm{Diag}(A1_n) - A.$$

It is straightforward to show that MaxCut is equivalent to the following binary quadratic optimization problem:

$$\begin{array}{ll} \text{maximize} & x^\top L x \\ \text{subject to} & x \in \{-1, 1\}^n. \end{array} \tag{4.1}$$

Although being NP-Hard, SDP-based methods have been successfully applied to solve MaxCut instances up to moderate sizes. However, scalability is still an issue

even for modern state-of-the-art solvers. To derive the basic SDP relaxation of (4.1) we can use the trace property to reformulate the problem as

$$\begin{array}{ll} \text{maximize} & \text{Tr}(Lxx^\top) \\ \text{subject to} & x \in \{-1,1\}^n. \end{array} \tag{4.2}$$

We now introduce the rank one positive semidefinite matrix $X = xx^\top$. Since x is a $\{-1,1\}^n$-vector we have the following constraint on the diagonal of X, $diag(X) = 1_n$. Finally, by dropping the rank one constraint, we obtain the semidefinite relaxation:

$$\begin{array}{ll} \text{maximize} & \langle L, X\rangle \\ \text{subject to} & diag(X) = 1_n \\ & X \succeq 0. \end{array} \tag{MC$_{BASIC}$}$$

Different algorithms have been developed to tackle diagonally constrained SDPs. BiqBin uses a primal-dual interior point method tailored to this kind of problems. Low-rank approaches also perform very well and can surprisingly scale to large instances [Wang et al., 2017, Yurtsever et al., 2021] up to a size of $n = 10^7$. These methods work on a low-rank reformulation of (MC$_{BASIC}$). The positive semidefinite matrix X can be factorized as a product of two rank-k matrices: $X = V^\top V$, $V \in \mathbb{R}^{k\times n}$. Then, (MC$_{BASIC}$) becomes:

$$\max_{V\in\mathbb{R}^{k\times n}} \langle L, V^\top V\rangle \;\text{ s.t } \|V_i\| = 1,\; i = 1,\dots,n, \tag{MC$_{LR}$}$$

with $V_i \in \mathbb{R}^k$ the i-th column vector of V. The norm constraints on the columns are inherited from the diagonal constraints of the original problem. As mentioned in Chapter 2, $X_{ii} = 1$ implies that $V_i^T V_i = \|V_i\|^2 = 1$, hence the norm constraints. The low-rank reformulation reduces the number of variables from n^2 to kn and the semi-definite constraint $X \succeq 0$ now becomes implicit as $V^\top V$ is always positive semi-definite. While we have considerably reduced the number of variables, the problem (MC$_{LR}$) is no longer convex. However, we have already seen that there are deterministic guarantees for which the solution of (MC$_{BASIC}$) can be recovered from the low-rank problem [Boumal et al., 2020].

A stronger relaxation Unfortunately, the basic SDP relaxation may be too loose for medium and large instances. As a result, the number of nodes within the branch and bound scheme increases sharply with the size of the problem, and the overall algorithm does not scale well. To overcome this issue, we strengthen the basic SDP relaxation by adding valid cuts to the formulation. Various categories of cuts have been considered in the literature for the MaxCut problem. In this work, we are interested in hypermetric inequalities [Deza and Laurent, 2010]. These inequalities can be derived from the so-called *gap inequalities* [Laurent and Poljak, 1996] given by:

$$\langle bb^\top, X\rangle \geqslant B, \tag{4.3}$$

with $b \in \mathbb{Z}^n$ and $B := \min\{|b^\top x| : x \in \{-1,1\}^n\}$. Hypermetric inequalities are inequalities for which the right hand side $B = 1$ and $b \in \mathcal{B} := \{b \in \mathbb{Z}^n : \sum b_i \text{ odd}\}$. by setting only three entries of b to be either 1 or -1 we can derive the so called triangular inequalities:

$$\begin{aligned} X_{ij} + X_{ik} + X_{jk} &\geqslant -1 \\ X_{ij} - X_{ik} - X_{jk} &\geqslant -1 \\ -X_{ij} + X_{ik} - X_{jk} &\geqslant -1 \\ -X_{ij} - X_{ik} + X_{jk} &\geqslant -1, \end{aligned} \tag{4.4}$$

for distinct $i, j, k \in [n]$. Intuitively, a cut can only pass through either 0 or 2 edges for a cycle of length 3 in the graph. There exists $4\binom{n}{3}$ such inequalities which are embedded in the linear operator $\mathcal{A}(X) \leqslant b$. This allows us to derive a strengthened relaxation:

$$\begin{aligned} \text{maximize} \quad & \langle L, X \rangle \\ \text{subject to} \quad & diag(X) = 1_n \\ & \mathcal{A}(X) \leqslant b \\ & X \succeq 0. \end{aligned} \tag{MC$_{TRI}$}$$

Note that each triangular inequality involves only three entries of the matrix X. As a result, $\mathcal{A}(X)$ can be computed efficiently in $O(m)$ operations with m the number of triangular inequalities considered.

4.3.2 Other exact solvers based on semidefinite programming

In this section, we provide a brief summary of other exact solvers used to address the MaxCut problem. As mentioned earlier, semidefinite programming proves to be the most effective approach for solving general MaxCut instances, employed by the top-performing solvers such as BiqMac [Rendl et al., 2010], BiqCrunch [Krislock et al., 2017], BiqBin [Gusmeroli et al., 2022] and MADAM [Hrga and Povh, 2021].

4.3.2.1 BiqMac and BiqBin

BiqMac and BiqBin are two SDP-based solvers using partial Lagrangian methods. BiqBin was originally designed to handle more general problems, namely binary quadratic problems with linear constraints. These problems are reduced to MaxCut instances using the exact penalization developed by Lasserre [Lasserre, 2016]. This transformation was further studied in [Gusmeroli and Wiegele, 2022]. Both solvers use hypermetric inequalities to tighten the basic MaxCut relaxation. Since the number of constraints can become quite large, dual-primal interior point solvers are limited to small instances. The idea behind BiqMac is to dualize only triangular inequalities. An upper bound is obtained by minimizing the following partial dual function

$$g(y) = y^\top 1_n + \max_{\substack{\text{diag}(X) = 1_n \\ X \succeq 0}} \langle L - \mathcal{A}^*(y), X \rangle, \tag{4.5}$$

with y the non negative dual variable. This nonsmooth optimization problem is solved with the bundle method. In order to compute subgradients and function evaluations, one has to solve the inner maximization problem. This can be done efficiently by interior point solvers adapted to diagonally constrained SDPs. Compared to BiqMac, BiqBin uses pentagonal and heptagonal inequalities to strengthen the relaxation. In addition, the latter solver features a parallelized branch and bound for improved scaling.

4.3.2.2 BiqCrunch

BiqCrunch works with an augmented Lagrangian version of (MC_{TRI}) with only triangular inequalities. The augmenting term is based on a theoretical result over the following set:

$$\Omega(n) := \{X \in \mathcal{S}_+^n \mid \mathrm{Tr}(X) = n\}.$$

It is known that for $X \in \Omega(n)$, $\|X\|_F \leqslant n$, and equality holds if and only if rank(X) = 1. The new problem with penalty parameter α is

$$\begin{array}{ll} \text{maximize} & \langle L, X\rangle + \dfrac{\alpha}{2}(\|X\|_F^2 - n^2) \\ \text{subject to} & \mathrm{diag}(X) = 1_n \\ & \mathcal{A}(X) \leqslant b \\ & X \succeq 0. \end{array} \tag{4.6}$$

Using projection onto the semidefinite cone, the dual minimization problem can be simplified to

$$\begin{array}{ll} \text{minimize} & \dfrac{1}{2\alpha}\,\|[L - \mathrm{Diag}(y) - \mathcal{A}^*(z)]_{\succeq 0}\|_F^2 + 1_n^\top y + b^\top z + \dfrac{\alpha}{2}n^2 \\ \text{subject to} & y,\ z \geqslant 0. \end{array} \tag{4.7}$$

Note that this problem is no longer a SDP. In [Malick and Roupin, 2013] the authors state that the dual function is convex and differentiable thus, it can be efficiently minimized using first order methods such as L-BFGS-B [Zhu et al., 1997]. Tighter bounds are obtained by sequentially decreasing the penalty parameter α.

4.3.2.3 MADAM

MADAM is a parallel SDP solver using the alternating direction method of multipliers (ADMM) for the bounding procedure. ADMM is applied to solve a slightly modified version of (MC_{TRI}). The authors introduce a slack variable for the inequality constraints

$$\begin{array}{ll} \text{maximize} & \langle L, X\rangle \\ \text{subject to} & \mathrm{diag}(X) = 1_n \\ & \mathcal{A}(X) + s = b \\ & X \succeq 0,\ s \geqslant 0. \end{array} \tag{4.8}$$

Solvers	SDP relaxation			Resolution	Parallelization
	Triangular	Pentagonal	Heptagonal		
BiqMac	✓			Bundle and IPM	
BiqCrunch	✓			L-BFGS-B	
BiqBin	✓	✓	✓	Bundle and IPM	✓
Madam	✓	✓	✓	ADMM	✓

Table 4.1: SDP solvers with their main features.

Due to the special structure of the MaxCut relaxation (4.8), they are able to prove convergence of the multi-block ADMM [Hrga and Povh, 2021]. The formulation includes triangular, pentagonal and heptagonal inequalities. The algorithm can be warm-started during the cutting plane scheme. At each cutting plane iteration, inactive inequalities are removed from the model and the values of the remaining variables are stored to provide a new starting point for the next iteration. As far as we know, MADAM is the best existing exact solver on the BiqMac library instances [Hrga and Povh, 2021].

The solvers with their different features are summarized in Table 4.3.2.3.

4.4 Bounding procedure

The bounding procedure is an important feature of the branch and cut framework. More subtrees can be pruned in the search space by computing tight bounds. However, computing such bounds can become too computationally intensive, making the algorithm inefficient compared to other cheaper methods. Although SDP methods provide tight bounds, they are more demanding than LP-based methods in terms of memory requirements and number of operations. Even for first order methods like ADMM, the computational complexity is at least $O(n^3)$. This has changed with the introduction of the low-rank mixing method which consists of efficient $O(n^2k)$ updates of the matrix V with $k \ll n$.

4.4.1 Combining the mixing method with the proximal bundle method

While being efficient on diagonally constrained SDPs, the mixing method is not able to handle general constraints such as hypermetric inequalities. It is then impossible to directly use it in our branch and cut framework. In the following, we make the implicit assumption that (MC_{TRI}) is strictly feasible, *i.e.*, strong duality holds. To include the inequalities, we use a method similar to the one used in BiqMac [Rendl et al., 2010] and BiqBin [Gusmeroli et al., 2022]. We use a partial Lagrangian approach and only dualize the affine constraints $\mathcal{A}(X) \leq b$. The partial Lagrangian is given by

$$\mathcal{L}(X, y) = \langle L, X\rangle - y^\top(\mathcal{A}(X) - b) = b^\top y + \langle L - \mathcal{A}^*(y), X\rangle,$$

where $y \in \mathbb{R}^m_+$ are the dual variables corresponding to the constraints $\mathcal{A}(X) \leqslant b$. The dual function is

$$f(y) = \max_{X \in \mathcal{E}} \mathcal{L}(X, y) = b^\top y + \max_{X \in \mathcal{E}} \langle L - \mathcal{A}^*(y), X \rangle,$$

where

$$\mathcal{E} \coloneqq \{X \in \mathbb{R}^{n \times n} \colon \operatorname{diag}(X) = 1_n,\ X \succeq 0\}$$

denotes the so-called *elliptope*. By weak duality, we have $\operatorname{opt}(\mathrm{MC}_{TRI}) \leq f(y)$ for all $y \in \mathbb{R}^m_+$. In order to compute a good upper bound on the optimal value of (MC_{TRI}), we approximately solve the dual problem

$$\min_{y \geq 0} f(y) = \min_{y \geq 0} \left\{ b^\top y + \max_{X \in \mathcal{E}} \langle L - \mathcal{A}^*(y), X \rangle \right\}. \tag{4.9}$$

Since the dual fonction f is convex but nonsmooth we use the bundle method, a first order method based on subgradient information. It is straightfoward to verify that an evaluation of the function and a subgradient at certain point $y \in \mathbb{R}^m_+$ are given by

$$\begin{aligned} f(y) &= b^\top y + \langle L - \mathcal{A}^*(y), X^* \rangle \\ \partial f(y) &= b - \mathcal{A}(X^*), \end{aligned} \tag{4.10}$$

with

$$X^* \coloneqq \arg\max_X \langle L - \mathcal{A}^*(y), X \rangle,\ \text{s.t}\ X \in \mathcal{E}. \tag{4.11}$$

The advantage of our solver over BiqBin and BiqMac is that we solve (4.11) efficiently using the mixing method. Note that at each iteration, the mixing method always returns a primal feasible point, $X^k = {V^k}^\top V^k \in \mathcal{E}$. Even if (4.11) is solved approximately, the solution still provides a valid subgradient and the bundle algorithm converges.

4.4.2 Safe upper bounds

When working with the branch and cut paradigm, it is important that the bounding procedure returns bounds with guarantees. Indeed, if this is not the case, we can potentially prune parts of the tree where the solution lies, and the branch and cut algorithm will not be exact. The mixing method is essentially a primal method, *i.e.*, we cannot directly use the returned value as a safe upper bound. The dual problem for (MC_{BASIC}) is

$$\begin{aligned} \text{minimize} \quad & 1_n^\top \mu \\ \text{subject to} \quad & \operatorname{Diag}(\mu) - L \succeq 0. \end{aligned} \tag{MCD_{BASIC}}$$

Let p^* be the optimal value of (MC_{BASIC}), by weak duality we know that for any dual feasible point

$$p^* \leqslant 1_n^\top \mu.$$

Let V^* be the solution for the primal problem (MC_{LR}) and let $\mu^* \in \mathbb{R}^n$ be the vector

$$\mu_i^* = \|V^* L_i\|, \quad i = 1, \dots, n,$$

with $L_i \in \mathbb{R}^n$ the column vectors of L. In [Wang et al., 2017], the authors show that (V^*, μ^*) is a primal-dual feasible pair that closes the duality gap, *i.e.*, $\langle L, V^{*\top} V^* \rangle = e^\top \mu^*$. This result is interesting because it allows us to compute a dual feasible point by using only the primal solution of the low-rank problem and the upper bound is exact. However, it holds only if we are able to compute the exact solution of (MC_{LR}). Since the mixing method returns an approximate solution $\hat{V}$, the vector $\hat{\mu}_i = \|\hat{V} L_i\|_2$ is not always dual feasible. In order to tackle this issue, we compute the dual slack matrix $S = \text{Diag}(\hat{\mu}) - L$. If S is already positive semidefinite, nothing has to be done and $1_n^\top \hat{\mu}$ gives us the desired upper bound. If this is not the case, we apply the following offset

$$S \leftarrow S - \lambda_{\min}(S) I_n$$

which makes the matrix S positive semidefinite. The new point

$$\hat{\mu}_i \leftarrow \hat{\mu}_i - \lambda_{\min}(S), \quad i = 1, \dots, n$$

is now dual feasible. Note that this transformation deteriorates the dual objective value by the quantity $-n\lambda_{\min}(S)$. Let p^*_{TRI} be the optimal value of (MC_{TRI}), by weak duality, for any feasible y, we have

$$p^*_{TRI} \leqslant b^\top y + \max_{X \in \mathcal{E}} \langle L - \mathcal{A}^*(y), X \rangle. \tag{4.12}$$

We can then compute a safe upper bound using the result presented above

$$p^*_{TRI} \leqslant b^\top y + \max_{X \in \mathcal{E}} \langle L - \mathcal{A}^*(y), X \rangle \leqslant b^\top y + 1_n^\top \hat{\mu}. \tag{4.13}$$

4.5 Branch and cut

Branch and cut is a method that has been widely used to solve combinatorial problems [Mitchell, 2002]. It is an exact algorithm that combines a cutting plane method with a branch and bound algorithm. The goal of the cutting plane method is to better approximate the feasible set of the integer problem by adding valid inequalities. In our solver it consists in solving a sequence of SDP relaxations with additional triangular inequalities. Overall, the solver has a better performance than the use of branch and bound alone with the basic MaxCut relaxation. In particular, we have better scalability because we can prune far more nodes.

4.5.1 Bounding routine

For our tree exploration, we use a best-first search (BFS) strategy. Nodes are sorted in a priority queue based on their upper bound. Nodes with the largest upper bound

are explored first. Since it is a maximization problem, nodes with an upper bound smaller than the global lower bound can be pruned. Each subproblem is evaluated using a cutting plane algorithm. First, we separate the triangular inequalities by complete enumeration and the most violated cuts are added to the SDP formulation. Then, we efficiently solve the relaxation with the method we described in 4.4.1. At the end of each cutting plane iteration, we check if the node can be pruned. If it is not the case, we remove the cuts that are almost inactive and we start another cutting plane iteration until a stopping criterion is satisfied. Almost inactive cuts are cuts for which the dual multiplier is close to 0, the tolerance to remove them is controlled by a parameter of our solver. At the end of the cutting plane algorithm, if the node cannot be pruned, we compute the safe upper bound (4.13). The generic algorithm is shown in Algorithm 5.

Algorithm 5 MixCut bounding procedure

```
Initialize bundle method by computing the solution of (MC_BASIC)
if node not prunable then
  Call to primal heuristic (4.6)
end if
while not done do
  Remove inactive cuts and add most violated ones
  Compute new gradient and function evaluation (4.10)
  Call to bundle method to solve (4.9)
  if node not prunable then
    Call to primal heuristic (4.6)
  end if
end while
if node not prunable then
  Compute safe bound (4.13)
end if
```

4.5.2 Branching rules

Besides the bounding procedure, branching rules are one of the main features of the branch and cut algorithm. In our case, we branch on edges of the graph. For a particular edge, we divide the search space in two, depending on whether the vertices are on the same or opposite sides of the cut. In order to choose which edge to branch on, we use the following rules:

- First vertex: we choose the first vertex of the edge based on dual information. Considering equation (4.13), at the end of the bounding procedure, we have two vectors of dual variables. A first vector $\hat{\mu} \in \mathbb{R}^n$ which corresponds to the diagonal constraint $\text{diag}(X) = 1_n$ and second vector $y \in \mathbb{R}^m$ for the hypermetric inequalities. We merge these two vectors into a single vector

$\nu \in \mathbb{R}^n$ with the following transformation:

$$\nu_i = \hat{\mu}_i + w_1 \sum_{\substack{cut[j] \text{ applies} \\ \text{on row } X_i}} y_j, \quad i = 1, \dots, n \tag{4.14}$$

with w_1 a constant weight and $cut[j]$ the j-th triangular inequality in our list of cuts. Then we keep the vertex with highest dual value:

$$k = \arg\max_{i=1,\dots,n} \ \nu_i.$$

A motivation for this choice is the fact that

$$\hat{\mu}_i = \|VL_i\| - \lambda_{\min}(S), \ i = 1, \dots, n,$$

which means that the vertex with the maximum dual variable $\hat{\mu}_i$ is the one that concentrates the most weights in the column vector L_i. This vertex should have a greater influence on the cut than other vertices.

- Second vertex: for the second vertex, we also use the primal solution. We are looking at entries X_{ki} such that $|X_{ki}|$ is not too close to 1. Intuitively, when the entry $|X_{ki}|$ is close to one, it means that vertices k and i are likely to be on the same side of the cut. The second vertex is chosen as follows:

$$l = \arg\max_{i=1,\dots,n} \ (\nu_k + \nu_i) \times (1 + w_2|X_{ki}|^{1.5}), \tag{4.15}$$

with w_2 a constant weight.

At the end of the procedure, we branch on the edge (k, l). This creates two new subproblems with size reduced by one. Those subproblems are added to a priority queue following a best first strategy, that is, nodes with the highest upper bound are evaluated first.

4.5.3 Problem reformulation and warm start

We now discuss how to reformulate the MaxCut problem (4.1) after a branching decision. As mentioned in the last paragraph, we branch on edges of the graph. Without loss of generality, let us consider that we branch on the edge $(n-1, n) \in E$. Whether we set $x_{n-1} = x_n$ or $x_{n-1} \neq x_n$, the entries of $X = xx^\top$ must satisfy $X_{k(n-1)} = X_{kn}$ or $X_{k(n-1)} \neq X_{kn}$ for $k = 1, \dots n$. This allows us to reformulate the problem with one variable less. Let $L \in \mathbb{R}^{n \times n}$ be the Laplacian matrix of the graph:

$$L = \begin{pmatrix} \bar{L} & c_1 & c_2 \\ c_1^\top & \alpha & \beta \\ c_2^\top & \beta & \gamma \end{pmatrix}, \tag{4.16}$$

with $\bar{L} \in \mathbb{R}^{(n-2)\times(n-2)}$, $c_1, c_2 \in \mathbb{R}^{n-2}$ and $\alpha, \beta, \gamma \in \mathbb{R}$. For $x_{n-1} = x_n$:

$$L_S = \begin{pmatrix} \bar{L} & c_1 + c_2 \\ c_1^\top + c_2^\top & \alpha + 2\beta + \gamma \end{pmatrix} \in \mathbb{R}^{(n-1)\times(n-1)}. \tag{4.17}$$

For $x_{n-1} \neq x_n$:

$$L_O = \begin{pmatrix} \bar{L} & c_1 - c_2 \\ c_1^\top - c_2^\top & \alpha - 2\beta + \gamma \end{pmatrix} \in \mathbb{R}^{(n-1)\times(n-1)}. \tag{4.18}$$

In our implementation, the branching history is stored at each node of the branch and bound tree. The corresponding Laplacian matrix is built when the node is evaluated. These transformations were already presented in [Mitchell, 2001] for a slightly modified MaxCut SDP relaxation.

Warm starting the cutting plane algorithm In order to reduce the number of cutting plane iterations at each node, active cuts from the parent node are passed down to the child nodes. Without loss of generality, let us consider that we branch on the edge $(j,k) \in E$. Any triangular inequality sharing an index with the edge (j,k) will be affected by the branching decision. For completeness, we give the reformulation of the constraint for the four types of triangle inequalities. Let us consider a triangular inequality with indices (i,j,k), for compactness, we only use the submatrix with indices (i,j,k) to represent the constraints. Matrices with the index S correspond to the branching decision $x_j = x_k$, for $x_j \neq x_k$ we use the index O. The transformations are basically the same as for the Laplacian matrix. We add or subtract row and column k to row and column j and then we remove them. This results in a constraint matrix with a size reduced by one.

- $X_{ij} + X_{ik} + X_{jk} \geqslant -1$:

$$A_1 = \frac{1}{2}\begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}, \tag{4.19}$$

$$A_{1S} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}, \qquad A_{1O} = \begin{pmatrix} 0 & 0 \\ 0 & -1 \end{pmatrix}. \tag{4.20}$$

- $X_{ij} - X_{ik} - X_{jk} \geqslant -1$:

$$A_2 = \frac{1}{2}\begin{pmatrix} 0 & 1 & -1 \\ 1 & 0 & -1 \\ -1 & -1 & 0 \end{pmatrix}, \tag{4.21}$$

$$A_{2S} = \begin{pmatrix} 0 & 0 \\ 0 & -1 \end{pmatrix}, \qquad A_{2O} = \begin{pmatrix} 0 & 1 \\ 1 & 1 \end{pmatrix}. \tag{4.22}$$

- $-X_{ij} + X_{ik} - X_{jk} \geqslant -1$:

$$A_3 = \frac{1}{2}\begin{pmatrix} 0 & -1 & 1 \\ -1 & 0 & -1 \\ 1 & -1 & 0 \end{pmatrix}, \tag{4.23}$$

$$A_{3S} = \begin{pmatrix} 0 & 0 \\ 0 & -1 \end{pmatrix}, \qquad A_{3O} = \begin{pmatrix} 0 & -1 \\ -1 & 1 \end{pmatrix}. \tag{4.24}$$

- $-X_{ij} - X_{ik} + X_{jk} \geqslant -1$:

$$A_4 = \frac{1}{2}\begin{pmatrix} 0 & -1 & -1 \\ -1 & 0 & 1 \\ -1 & 1 & 0 \end{pmatrix}, \tag{4.25}$$

$$A_{4S} = \begin{pmatrix} 0 & -1 \\ -1 & 1 \end{pmatrix}, \qquad A_{4O} = \begin{pmatrix} 0 & 0 \\ 0 & -1 \end{pmatrix}. \tag{4.26}$$

At the end, the reformulation for the child nodes give a new set of 3 constraints:

- $-X_{jj} \geqslant -1$.
- $2X_{ij} + X_{jj} \geqslant -1$.
- $-2X_{ij} + X_{jj} \geqslant -1$.

The first constraint is implicitly satisfied by the diagonal constraint on X. The last two constraints can be simplified to a bound constraint on the entry X_{ij} of X:

$$1 \geqslant X_{ij} \geqslant -1.$$

However, this constraint is also implicitly satisfied by the two constraints $\operatorname{diag}(X) = 1_n$ and $X \succeq 0$. Indeed, let us consider the two vectors $u_1 = e_i + e_j$ and $u_2 = e_i - e_j$,

$$u_1^\top X u_1 = X_{ii} + X_{jj} + 2X_{ij} \geqslant 0 \implies X_{ij} \geqslant -1.$$

$$u_2^\top X u_2 = X_{ii} + X_{jj} - 2X_{ij} \geqslant 0 \implies 1 \geqslant X_{ij}.$$

Since the inequalities are redundant, they are not passed down to the child nodes. Experimentally, we noticed that a large proportion of the active cuts at the parent node are also active at the child nodes. This has motivated us to warm start the cutting plane algorithm with the active cuts of the parent node.

4.6 Biased hyperplane

In order to find good integer solutions, we use the random hyperplane rounding proposed by Goemans and Williamson. Those feasible solutions are generated at each node using the low rank solution V of (MC_{TRI}). We draw a random vector r on the unit sphere and compute the dot products with the columns of V. The sign of the dot product $r^\top V_i$ gives the side of the cut for the entries of the resulting vector v. This first step is followed by a local search (1-OPT) that flips the sign of each entry until the solution can no longer be improved. We generate multiple random vectors and repeat the scheme described above.

Intuitively, columns of V colinear with the random vector r have a high chance of being fixed to either 1 or -1. For vectors close to the cutting hyperplane the result is unsure. Therefore, vectors for which the absolute value of the dot product $r^\top V_i$ is small are good candidates to apply the 1-OPT. Now, a question naturally arises,

can we heuristically find a "good" vector that will give us better integer solutions in general? If we bias the vector selection, we lose the theoretical approximation guarantee of Goemans and Williamson. However, we have observed some gains in practice.

Following our intuition, we want to find a vector that will maximize the absolute values of the dot products with the columns of V. We can estimate such vector by solving the maximization problem:

$$\max_{r \in \mathbb{R}^k} \ \|V^\top r\|, \ \text{s.t} \ \|r\| = 1. \tag{4.27}$$

The optimal value of (4.27) is well known since it is the spectral norm of the matrix $V^\top \in \mathbb{R}^{n \times k}$, that is:

$$\|V^\top\|_2 = \max\{\|V^\top r\| \ : \ \|r\| = 1\} = \sqrt{\lambda_{\max}(VV^\top)}.$$

The optimum is attained for the normalized eigenvector corresponding to the maximum eigenvalue of $VV^\top \in \mathbb{R}^{k \times k}$. This eigenvector can be efficiently computed using iterative methods such as the Power Iteration method [Golub and Van Loan, 2013]. For a $(k \times k)$ matrix, the cost of an iteration of this method is $O(k^2)$. An example of the biased hyperplan idea is given in Figure 4.6.

Experimentally, we have found that the solver with the biased hyperplane heuristic is more stable than the one with the random heuristic. This means that the solution is usually found higher up in the tree, regardless of the seed.

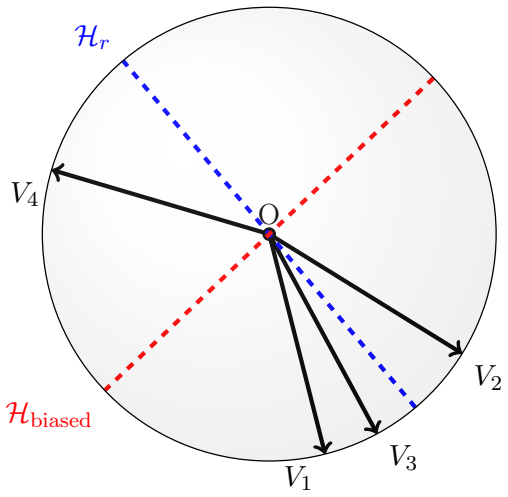


Figure 4.1: Random ($\mathcal{H}_r$) and biased ($\mathcal{H}_{\text{biased}}$) hyperplanes on a small example.

4.7 Implementation

Stopping criteria The mixing method is a primal first order method, it is very efficient at finding good quality solutions, but it can take a long time to reach the

optimum. Wang et al. showed that the mixing method has a linear convergence rate to the global optimum whenever the solution is close enough [Wang et al., 2017]. For our solver, we decided to use four stopping criteria. These criteria use information from the objective function values and the primal iterates. Let $\Delta = f^{k+1} - f^k \geqslant 0$ be the objective difference before and after a complete mixing method iteration and V^k be the primal iterate at iteration k:

- $\Delta < \varepsilon_{\Delta_{abs}}$.
- $|\Delta|/(1 + |f^k|) < \varepsilon_{\Delta_{rel}}$.
- $\|V^{k+1} - V^k\|_F < \varepsilon_{V_{abs}}$.
- $\|V^{k+1} - V^k\|_F/(1 + \|V^k\|_F) < \varepsilon_{V_{rel}}$.

We use absolute and relative measures. In the latter case, the stopping criterion is less sensitive to the magnitude of the entries of the Laplacian matrix. Experimentally, we have found that we can recover a good solution of (4.11) within a few iterations of the mixing method. We achieve a good trade-off between quality of the bounds and time spent with stopping criteria of order 10^{-2}.

Cuts For our branch and cut algorithm, cuts are stored in a C-style array. Formally, a cut is a C-struct where we store the following information:

double	value	value of the violation
double	y	dual multiplier
int	i	first index
int	j	second index
int	k	third index
int	type	one of the four types of triangular inequalities
bool	inherited	cut inherited from the parent node or not

To handle the separation step efficiently, we use two standard C++ libraries. To store the most violated inequalities, we use a priority queue based on constraint violation. The cuts are stored in ascending order, *i.e.*, the least violated cut is at the top of the queue. Since we only add a limited number of inequalities at each iteration of the cutting plane algorithm, this allows us to easily keep track of the least violated cut. Indeed the priority queue gives a constant time access to the top element.

After the separation, we use an unordered_set from the boost C++ library `https://www.boost.org/` to add the most violated cuts to our set of cuts. This data structure is a hash table which has, on average, constant time complexity to access the data. This implementation can be easily extended to other inequalities such as pentagonal and heptagonal inequalities.

Parameters A parameter file is provided to the user to run the solver. If the parameters are not provided or are commented out, they are set to their default values. The various parameters are listed in Appendix A.

Solvers	SDP relaxation			Resolution	Parallelization
	Triangular	Pentagonal	Heptagonal		
MixCut	✓			Bundle and mixing method	
BiqCrunch	✓			L-BFGS-B	
BiqBin	✓	✓	✓	Bundle and IPM	✓
Madam	✓	✓	✓	ADMM	✓

Table 4.2: SDP solvers with their main features.

4.8 Experiments

Solvers Our solver is compared with three state of the art MaxCut solvers: BiqCrunch [Krislock et al., 2017], BiqBin [Gusmeroli et al., 2022] and Madam [Hrga and Povh, 2021]. After a discussion with the authors of these solvers, we did not include BiqMac [Rendl et al., 2010] in the experiments as it has been surpassed by BiqBin, which uses the same approach. We recall that our solver and BiqCrunch only include triangular inequalities, while BiqBin and Madam add pentagonal and heptagonal inequalities to their formulations. To ensure a fair comparison, we used the serial versions of BiqBin and Madam. All solvers are implemented in C and we run them using default parameters. For Madam, we used the penalty parameter suggested by the authors [Hrga and Povh, 2021]. The solvers and their main features are listed in Table 4.8.

All experiments were run on a single thread on a server equipped with a Xeon®Gold 6248R CPU@3.00GHz and 1TB of RAM running Debian Linux 4.19.98-1.

Instances For our experiments, we used dense MaxCut instances from the BiqMac library [Wiegele, 2007]. We focused on medium sized instances with $n = 100$ nodes. A short description of the instances is given bellow:

- $g05\_100.*$: unweighted graphs with edge probability $1/2$.
- $pm1d\_100.*$: weighted graphs with edge weights chosen uniformly from $\{-1, 0, 1\}$ and density 0.99.
- $wd\_100.*$: graphs with integer edge weights chosen from $[-10, 10]$ and density $d = 0.5, 0.9$.
- $pwd\_100.*$: graphs with integer edge weights chosen from $[0, 10]$ and density $d = 0.5, 0.9$.

Solving the root node relaxation First, we compare the four solvers for solving the root node relaxation on two different instances. Figure 4.2 shows the results on the pw05_100.0 instance. We evaluate the bound quality against the cpu time. Each data point corresponds to the value of the upper bound after a cutting plane

iteration. As expected, the bounds returned by BiqBin and Madam, which use pentagonal and heptagonal inequalities, cannot be reached by BiqCrunch and MixCut. The beneficial property of our solver is its ability to return a good quality solution in a relatively short time compared to other solvers. The results are also relevant for the instance g05_100.1 as shown in Figure 4.3.

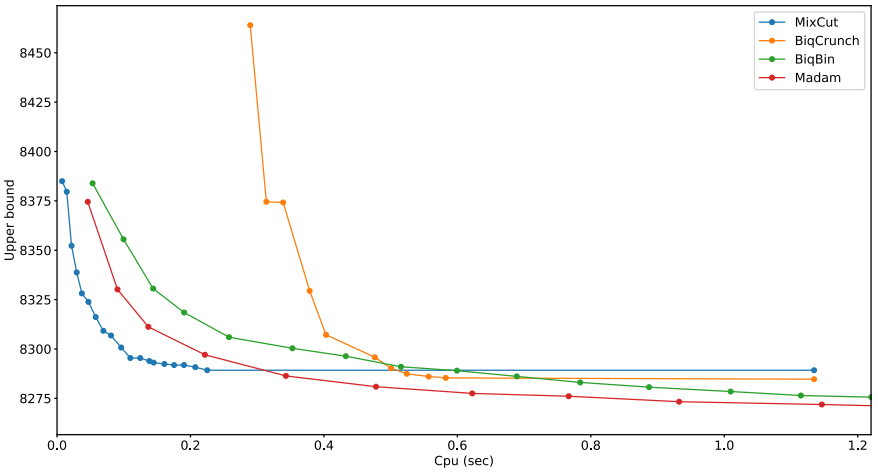


Figure 4.2: Upper bounds against Cpu (sec) at the root node of the pw05_100.0 instance.

Results on the dense BiqMac instances We now present the overall results on the dense instances of the BiqMac library. All solvers are set to solve the instances to optimality. We are interested in two pieces of information, the CPU time and the total number of branch and bound nodes. The results are shown in Tables 4.3, 4.4, 4.6 and 4.5. In terms of CPU time, MixCut outperforms the previous baseline on every type of instance. It is 5× to 10× faster than the serial version of Madam. The efficiency of MixCut, BiqCrunch, BiqBin and Madam to solve dense instances is shown in the cactus plot of Figure 4.4. A cactus plot shows how many problems can be solved in a given time. For each method separately it consists in

- Solving each problem p_i, noting time t_i.
- Sorting the t_i in increasing order.
- Plotting the point $(t_1, 1)$, $(t_1 + t_2, 2)$ and in general $(\sum_i^k t_i, k)$.

In the end, a point (x, y) for a solver means that if we set the timeout to y, it will prove optimality for x instances.

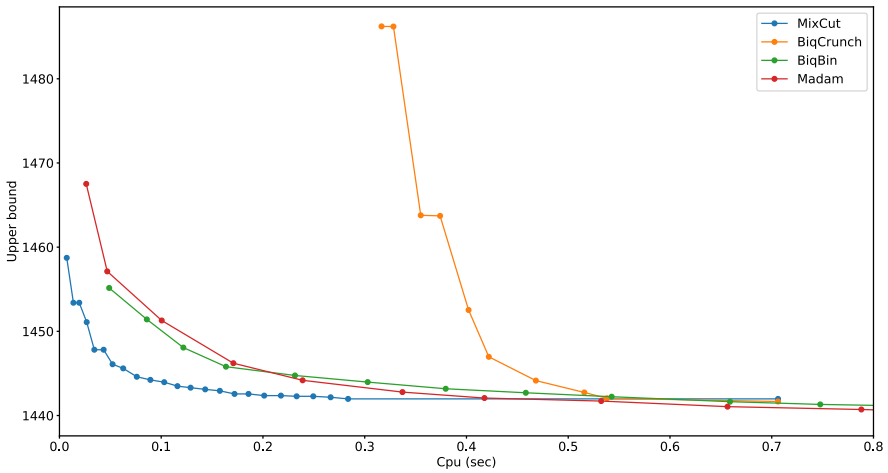


Figure 4.3: Upper bounds against Cpu (sec) at the root node of the g05_100.1 instance.

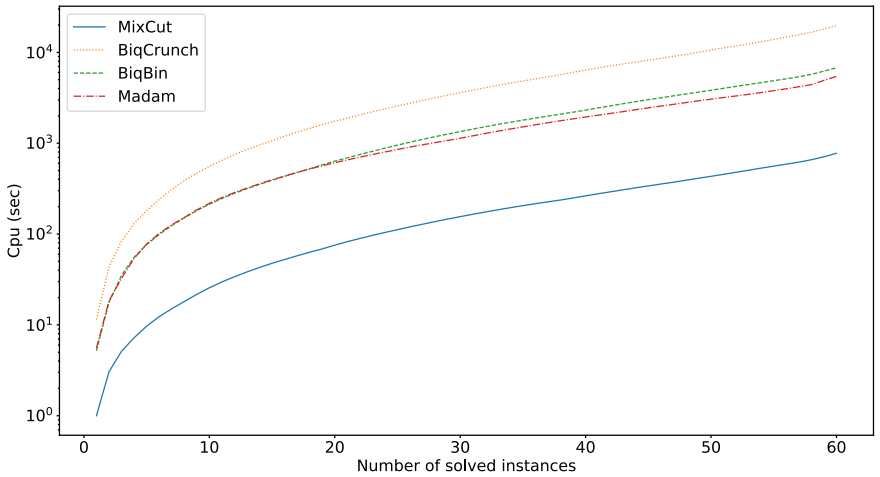


Figure 4.4: Number of dense instances solved by each solver as time passes. The time axis is on a logarithmic scale.

MixCut has the highest number of branch and bound nodes among the four solvers. For these medium sized instances we achieve a better trade-off between the quality of the bounds and the size of the explored tree. The bounds are less tight in our case since we only include triangular inequalities. While the mixing method is not suited to solve SDPs to a high precision, it excels at finding good quality solutions in a really short amount of time. As a result, we are able to prune fewer nodes, but the amount of time we spend at each node is drastically reduced. For SDP based methods, the computational complexity of the bounding procedure is often a bottleneck.

Problem	BiqCrunch		BiqBin - serial		MADAM - serial		`MixCut`	
	Time	Nodes	Time	Nodes	Time	Nodes	Time	Nodes
g05_100.0	253.89	325	107.48	99	88.09	195	**14.20**	743
g05_100.1	1447.89	1779	554.74	465	522.70	863	**66.06**	3615
g05_100.2	92.26	97	33.03	29	36.99	55	**4.19**	193
g05_100.3	454.63	659	195.09	209	127.56	389	**24.39**	1253
g05_100.4	31.85	31	12.43	7	12.85	11	**2.05**	87
g05_100.5	103.74	93	30.42	19	24.48	25	**4.77**	219
g05_100.6	99.20	99	36.37	25	43.01	33	**5.42**	245
g05_100.7	212.91	205	86.21	65	84.37	85	**9.27**	453
g05_100.8	143.52	165	60.57	41	48.29	79	**7.22**	361
g05_100.9	169.25	237	61.87	57	52.47	155	**8.02**	393

Table 4.3: CPU times (s) and B&B nodes to solve "g05" problems.

Problem	BiqCrunch		BiqBin - serial		MADAM - serial		`MixCut`	
	Time	Nodes	Time	Nodes	Time	Nodes	Time	Nodes
pm1d_100.0	581.78	643	221.99	231	117.49	267	**19.72**	1013
pm1d_100.1	1041.60	1107	346.44	411	229.42	425	**38.57**	2167
pm1d_100.2	816.22	929	254.11	367	156.53	319	**29.05**	1589
pm1d_100.3	242.89	227	56.74	63	42.45	71	**8.09**	399
pm1d_100.4	589.04	619	199.17	289	130.30	277	**21.95**	1191
pm1d_100.5	165.20	157	77.74	45	44.42	61	**6.90**	347
pm1d_100.6	143.09	133	46.58	33	34.30	41	**5.59**	277
pm1d_100.7	87.43	65	31.83	13	33.49	17	**3.59**	167
pm1d_100.8	46.71	41	20.42	9	14.48	15	**2.08**	91
pm1d_100.9	225.98	239	93.99	59	45.27	79	**8.82**	461

Table 4.4: CPU times (s) and B&B nodes to solve "pm" problems.

Result on sparse BiqMac instances We also compared the solvers on the sparse MaxCut instances of the BiqMac library. Once again we focused on instances with $n = 100$ nodes. A short description of the instances is given bellow:

- pm1s_100.*: weighted graphs with edge weights chosen uniformly from $\{-1, 0, 1\}$ and density 0.1.

Problem	BiqCrunch		BiqBin - serial		MADAM - serial		`MixCut`	
	Time	Nodes	Time	Nodes	Time	Nodes	Time	Nodes
w05_100.0	354.92	369	96.58	139	78.46	149	**10.55**	507
w05_100.1	105.44	91	44.63	37	39.88	29	**4.40**	199
w05_100.2	70.79	61	34.81	15	32.36	15	**3.03**	135
w05_100.3	255.61	267	102.12	81	85.60	69	**9.83**	463
w05_100.4	431.33	423	146.65	131	120.45	145	**15.81**	761
w05_100.5	364.58	337	146.83	107	112.10	135	**15.17**	729
w05_100.6	132.98	119	38.80	19	25.81	33	**4.09**	177
w05_100.7	50.10	37	20.95	9	23.66	9	**2.72**	107
w05_100.8	360.93	369	145.79	137	95.72	185	**14.34**	653
w05_100.9	40.29	29	26.83	9	37.77	15	**2.08**	89
w09_100.0	253.50	229	91.23	69	76.69	91	**10.42**	505
w09_100.1	1549.04	1395	484.83	577	505.20	567	**50.67**	2727
w09_100.2	594.82	559	167.49	247	128.79	233	**19.47**	991
w09_100.3	852.80	855	205.05	233	186.41	463	**27.66**	1407
w09_100.4	360.15	301	133.62	103	109.18	133	**12.59**	625
w09_100.5	11.51	7	5.19	1	5.56	1	**1.00**	33
w09_100.6	76.00	49	22.30	9	34.03	13	**2.47**	117
w09_100.7	186.86	165	76.13	69	50.12	55	**6.86**	327
w09_100.8	131.96	95	70.95	33	70.94	35	**4.62**	207
w09_100.9	284.46	283	91.10	75	77.21	115	**10.36**	487

Table 4.5: CPU times (s) and B&B nodes to solve "w" problems.

Problem	BiqCrunch		BiqBin - serial		MADAM - serial		`MixCut`	
	Time	Nodes	Time	Nodes	Time	Nodes	Time	Nodes
pw05_100.0	865.32	1013	255.79	403	208.51	383	**30.76**	1567
pw05_100.1	301.84	353	94.75	105	84.28	103	**10.26**	501
pw05_100.2	347.32	387	131.55	119	88.57	149	**14.01**	637
pw05_100.3	81.43	83	25.18	15	21.32	29	**3.93**	191
pw05_100.4	409.96	393	155.39	117	90.00	191	**16.57**	827
pw05_100.5	106.67	103	35.03	29	24.26	33	**4.76**	213
pw05_100.6	706.84	729	220.62	223	178.79	195	**25.59**	1247
pw05_100.7	187.14	171	73.49	31	54.64	39	**7.00**	331
pw05_100.8	59.81	39	17.50	7	47.46	13	**2.65**	111
pw05_100.9	228.22	237	68.37	47	57.18	59	**8.34**	441
pw09_100.0	347.90	341	119.11	101	99.56	139	**16.83**	809
pw09_100.1	520.71	545	178.75	205	128.10	227	**21.02**	995
pw09_100.2	145.29	139	55.14	47	45.11	57	**7.81**	361
pw09_100.3	130.77	115	40.31	29	34.17	19	**5.22**	245
pw09_100.4	261.37	267	85.42	69	53.34	77	**10.09**	469
pw09_100.5	379.24	319	142.95	85	112.16	87	**15.77**	745
pw09_100.6	193.28	223	77.75	91	54.10	109	**10.38**	469
pw09_100.7	580.26	571	164.95	157	138.99	201	**27.44**	1317
pw09_100.8	217.10	177	90.01	43	75.31	57	**9.60**	457
pw09_100.9	159.83	133	56.87	25	61.07	35	**7.43**	335

Table 4.6: CPU times (s) and B&B nodes to solve "pw" problems.

- wd_100.*: graphs with integer edge weights chosen from $[-10, 10]$ and density $d = 0.1$.

- pwd_100.*: graphs with integer edge weights chosen from $[0, 10]$ and density $d = 0.1$.

We also did a comparison with the recent LP-based method presented in [Charfreitag et al., 2023]. The source code for this solver is publicly available at `https://github.com/CharJon/ScientificMaxcutSolver`. For instances with density $d = 0.1$, the LP-based solver is slower than the SDP approaches. The number of branch and bound nodes is higher than the other methods. As the results of the paper suggest [Charfreitag et al., 2023], it shines on sparser instances where SDP solvers are much less efficient. Results for the LP solver are given in Table 4.9. MixCut is also faster than the other SDP solvers on sparse instances. The results are shown in Tables 4.7 and 4.8. Overall, the time differences between the solvers tend to be closer than for the dense instances. For the easiest pw01 and w01 problems, BiqCrunch is competitive with BiqBin and Madam. For solvers including pentagonal and heptagonal inequalities, sparse instances are usually solved at the root node. However, even when problems are solved at the root, the trade-off using the mixing method is still in our favour. The efficiency of MixCut, BiqCrunch, BiqBin and Madam to solve sparse instances is shown in the cactus plot of Figure 4.5.

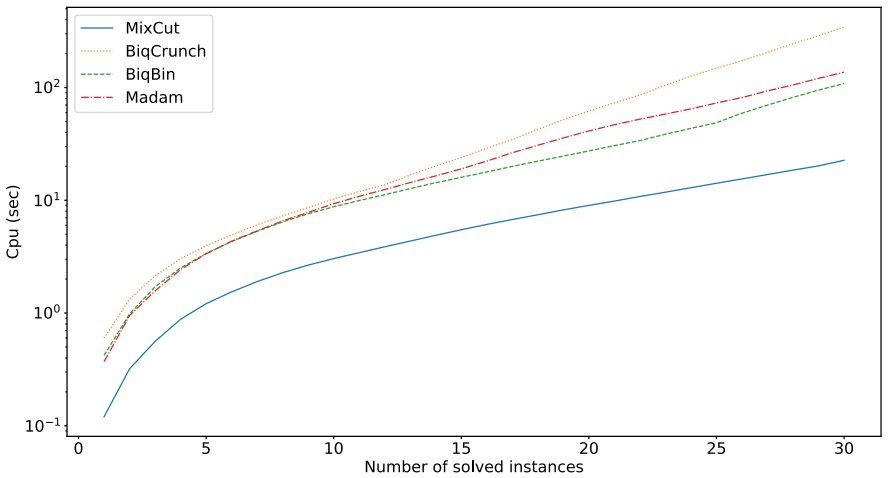


Figure 4.5: Number of sparse instances solved by each solver as time passes. The time axis is on a logarithmic scale.

Problem	BiqCrunch		BiqBin - serial		MADAM - serial		`MixCut`	
	Time	Nodes	Time	Nodes	Time	Nodes	Time	Nodes
pm1s_100.0	3.02	3	1.61	1	1.60	1	**0.54**	17
pm1s_100.1	31.91	31	13.46	9	11.85	31	**1.68**	67
pm1s_100.2	3.38	3	1.00	1	1.01	1	**0.39**	11
pm1s_100.3	42.96	45	12.01	7	12.13	23	**1.46**	61
pm1s_100.4	21.60	21	4.90	1	6.55	7	**0.99**	37
pm1s_100.5	5.59	7	1.24	1	0.99	1	**0.38**	15
pm1s_100.6	22.07	25	10.64	3	5.62	11	**0.97**	33
pm1s_100.7	0.60	1	0.42	1	0.37	1	**0.12**	1
pm1s_100.8	1.35	1	0.86	1	0.86	1	**0.32**	9
pm1s_100.9	1.67	1	1.13	1	2.10	1	**0.36**	13

Table 4.7: CPU times (s) and B&B nodes to solve sparse "pm" problems.

Problem	BiqCrunch		BiqBin - serial		MADAM - serial		`MixCut`	
	Time	Nodes	Time	Nodes	Time	Nodes	Time	Nodes
w01_100.0	1.79	1	1.50	1	1.52	1	**0.48**	15
w01_100.1	1.69	1	1.86	1	5.59	3	**0.63**	19
w01_100.2	40.24	33	10.49	7	8.59	17	**1.59**	59
w01_100.3	5.03	3	2.42	1	8.15	5	**0.69**	21
w01_100.4	0.90	1	0.80	1	0.62	1	**0.33**	9
w01_100.5	3.80	3	1.67	1	1.92	1	**0.59**	17
w01_100.6	0.91	1	0.94	1	1.24	3	**0.38**	11
w01_100.7	11.37	5	2.15	1	2.53	1	**0.64**	21
w01_100.8	0.81	1	0.56	1	4.14	1	**0.20**	3
w01_100.9	1.14	1	1.15	1	0.92	1	**0.44**	11
pw01_100.0	9.90	7	2.25	1	3.28	1	**0.79**	23
pw01_100.1	24.55	15	4.70	1	15.10	7	**1.30**	49
pw01_100.2	9.39	7	3.14	1	4.97	5	**0.83**	29
pw01_100.3	18.96	15	5.29	1	5.49	9	**1.23**	41
pw01_100.4	0.97	1	1.17	1	1.21	1	**0.33**	9
pw01_100.5	1.21	1	1.20	1	1.53	1	**0.38**	11
pw01_100.6	7.95	7	2.74	1	4.34	1	**0.78**	27
pw01_100.7	12.62	7	3.31	1	5.78	1	**1.13**	41
pw01_100.8	0.73	1	0.73	1	0.58	1	**0.24**	5
pw01_100.9	57.33	43	13.08	7	16.41	15	**2.44**	97

Table 4.8: CPU times (s) and B&B nodes to solve sparse "pw" and "w" problems.

Problems	SMS						MixCut					
	Time			Nodes			Time			Nodes		
	min	avg.	max	min	avg.	max	min	avg.	max	min	avg.	max
pm1s_100	9.34	25.45	46.64	37	240	559	**0.12**	**0.72**	**1.68**	**1**	**27**	**67**
w01_100	5.52	9.63	23.59	**1**	55	319	**0.2**	**0.60**	**1.59**	3	**19**	**59**
pw01_100	9.08	18.33	28.22	13	129	295	**0.24**	**0.95**	**2.44**	**5**	**34**	**97**

Table 4.9: CPU times (s) and B&B nodes to solve sparse MaxCut instances with the LP-based solver Scientific MaxCut Solver (SMS) and MixCut.

4.9 Discussion

To further improve our solver, there are several strategies we would like to explore. First, with respect to cuts, only triangular inequalities are included. For larger instances, the relaxation may not be tight enough and many nodes will not be pruned. As a result, the number of open nodes in the branch-and-bound tree quickly increases and the search space becomes too large to efficiently prove optimality. This can also lead to memory issues since we are storing information at each node. To tighten the relaxation and improve the bounding procedure, we are interested in adding other hypermetric inequalities. Following the work done in BiqBin and Madam, we saw that the two solvers provide better bounds by including pentagonal and heptagonal inequalities. Separation of triangular inequalities can be done by complete enumeration in $O(n^3)$, for the other two types there are a total of $16\binom{n}{5}$ and $64\binom{n}{7}$ inequalities respectively. To keep our solver efficient, a next step is to find a good heuristic for separating promising violated pentagonal and heptagonal inequalities. Note that in BiqBin and Madam this is done by approximately solving a Quadratic Assignment Problem (QAP) with simulated annealing.

A natural idea to extend our solver is to use parallelization for the branch-and-bound scheme. This method is already exploited in many exact solvers such as BiqBin and Madam.

Conclusion

In this thesis we explored different methods to approximate the solution of the MAP problem on pairwise discrete graphical models. Practically solving combinatorial problems is a major challenge in many fields. In the background chapter, we introduced graphical models, a framework which covers a variety of applications from constraint programming to computational protein design. Our main motivation was to design an optimization method which offers a good trade-off between efficiency and quality. By efficiency, we mean the ability to scale to large problems. By quality, we mean the ability to give a tight approximation of the discrete solution.

In Chapter 2 we introduced two continuous relaxations of the MAP problem on discrete pairwise graphical problems. For the first relaxation we were driven by the recent advances in solving MaxCut. We used an exact reduction to MaxCut proposed by Lasserre. Instead of using the usual interior point methods to solve the SDP relaxation of MaxCut, we used a low-rank method based on coordinate descent. However we realized that this method was not suited to handle the large entries introduced by the penalization. To overcome this issue, we proposed a second relaxation as well as a new low-rank SDP method. Our method relies on block coordinate descent. To perform a block coordinate descent step, we solve an optimization problem on the unit sphere with the Newton method.

We were able to derive some properties of our algorithm. LR-BCD is always decreasing and it converges toward a unique limit point. We also have the advantage that at each iteration, the low rank matrix is always primal feasible, *i.e.*, it always satisfies all the constraints. Experimentally, we compared LR-BCD to the commercial interior point solver Mosek and we found out that it quickly converges to the global optimum of the SDP problem. We even reached a speed-up of two order of magnitudes to converge to a good quality solution and we showed the ability of LR-BCD to scale to large instances. In order to find feasible integer solutions, we also slightly modified the Goemans and Williamson heuristic. Followed by a simple local search, we were able to produce strong solutions, sometimes better than specialized algorithms. Because of its different qualities, LR-BCD was used in different projects. In particular, it was added to a deep learning pipeline developed by Marianne Defresne for computational protein design.

During the experiments we realized that for certain instances our relaxation was not tight enough. We decided to add more constraints to have a better representation of the feasible set of the combinatorial problem. Since it was not clear whether LR-BCD could be extended to handle more general constraints we decided to use a different method to solve the corresponding SDP relaxation. This method was suggested by the optimization team at the Klagenfurt mathematics laboratory during my 1 month staying at the institute. It is a variant of the alternating direction method of multipliers, a first-order method that has shown good results for solving SDPs.

While in Klagenfurt, I also started a collaboration with Jan Schwiddessen, a PhD student under the supervision of Angelika Wiegele. This collaboration led to the work of the last chapter. This time, we were interested in solving combinatorial problems to optimality. We proposed a new exact MaxCut solver based on a low rank SDP method. We use a branch and cut algorithm with a bounding procedure that is efficiently performed by a combination of the mixing method and the proximal bundle method. For our solver, we developed new branching decisions and a derandomized version of the Goemans and Williamson rouding procedure. We demonstrate the superiority of our solver over existing methods on a variety of MaxCut instances from the BiqMac library.

Perspectives

Future work on LR-BCD The central optimization algorithm presented in this thesis can be extended in multiple directions. First, it would be interesting to derive more convergence properties for LR-BCD, *e.g.*, convergence to the global optimum. At the end of Chapter 2, we characterized the fixed points of the algorithm. Similar to what was done in the mixing method paper, a starting point for the convergence proof would be to analyze the Jacobian of the LR-BCD mapping. However it is not an easy task since we do not have a closed form solution for the block optimization problems.

Next, if we want to use LR-BCD in a branch and bound algorithm, we need to compute bounds with guarantees. For now, LR-BCD is a primal algorithm, so the returned value does not give a certificate for the lower bound. We aim to find an efficient method to recover a dual feasible point. In the Chapter 4, we saw that we can produce a dual feasible point for a particular SDP relaxation of MaxCut by using the primal solution returned by the mixing method. Ideally, we would like to have a similar method for LR-BCD.

Another direction is to improve the tightness of the relaxation solved by LR-BCD. Since the MAP relaxation solved by LR-BCD is not always tight, we would like to extend LR-BCD to handle more constraints. A natural way is to use a combination of LR-BCD and the bundle method. For example, we could add the positivity constraints to avoid negative solutions. In the $\{-1, 1\}$ setting, these constraints are similar to hypermetric inequalities. These constraints can be dualized as in the MixCut chapter and we would use LR-BCD to solve the inner SDP with the exactly-one constraint.

We also thought about combining ADMM with low-rank methods. Indeed, the low-rank factorization would allow us to get rid of the projections onto the semidefinite cone. While the problem becomes non-convex, researchers began to derive convergence properties for ADMM in non-convex settings [Wang et al., 2018]. Recently, Chen et al. [Chen and Goulart, 2023] proposed a Burer-Monteiro ADMM to solve the basic MaxCut SDP relaxation. However this method is limited to diagonally constrained SDPs and the extension to general constraints seems to be a hard

problem.

Boosting LR-BCD with parallelization Finally, regarding LR-BCD, we thought about a preprocessing step to accelerate the algorithm. The idea is to use parallelization to solve multiple block optimization problems at the same time. Once again, this result relies on the structure of the MAP problem. If we consider a pairwise GM with associated graph $G = (X, E)$, vertices that do not share an edge can be updated in parallel. Indeed, in the SDP relaxation we presented, the two corresponding variables share a zero block in the objective matrix. In this case, therefore, the update of one variable has no effect on the update of the other variable. Independent variables can be detected by applying a coloring algorithm on the graph G. For the coloring problem, there exists an efficient heuristic called DSATUR with complexity $O(n^2)$ where n is the number of vertices. At the end, variables with the same colors can be updated in parallel. The number of parallel blocks corresponds to the number of colors returned by the heuristic. Then, this preprocessing step would be more efficient on sparse instances where the chromatic number of the graph is more likely to be small.

LR-BCD will soon be integrated in the exact solver toulbar2 [Hurley et al., 2016] as a preprocessing step. The operations within LR-BCD are not too complex, *e.g.*, matrix-vector products, matrix-matrix products. We implemented our own linear algebra routines to avoid additional dependencies in toulbar2.

Improvement of the MixCut solver Regarding the MixCut solver we are developing with Jan Schwiddessen, we plan to add pentagonal and heptagonal inequalities to our relaxation. This work is the subject of Jan's forthcoming visit to Toulouse. Once completed, our work will be submitted to a journal, hopefully before the end of the year.

Appendices

Appendix A

MixCut parameters

A.1 MixCut parameters

type	parameter	usage
int	seed	seed for the random number generator
int	problem_type	0: Maxcut 1: QUBO in $\{-1, 1\}$ variables 2: QUBO in $\{0, 1\}$ variables
int	objective_is_integer	0: objective can take non-integer values 1: objective takes integer values only
int	objective_sense	0: minimization problem 1: maximization problem
double	X_abs_lower_bound	only branch on X_{ij} if $\lvert X_{ij}\rvert \geqslant$ X_abs_lower_bound
double	branching_X_abs_weight	impact of $\lvert X_{ij}\rvert$ for branching decision
int	early_branching	0: no early branching 1: early branching
int	min_iter	minimum number of iterations of the Mixing Method
double	tol_V_rel_early	stopping tolerance if early branching is used
double	tol_delta_abs	stop Mixing Method when $\Delta <$ tol_delta_abs
double	tol_delta_rel	stop Mixing Method when $\lvert\Delta\rvert/(1 + \lvert fval\rvert) <$ tol_delta_rel
double	tol_V_abs	stop Mixing Method when $\lVert V_{new} - V_{old}\rVert_F <$ tol_V_abs
double	tol_V_rel	stop Mixing Method when $\lVert V_{new} - V_{old}\rVert_F/(1 + \lVert V_{old}\rVert_F) <$ tol_V_rel
int	primal_heuristics	0: no primal heuristics 1: GW heuristic is applied once per subproblem 2: GW heuristic is applied more often for smaller subproblems 3: GW heuristic is applied k times (k is the rank for the factorization) 4: GW heuristic is applied n times (n is the size of the subproblem)
int	good_hyperplane	0: no "good"/biased hyperplane 1: the first hyperplane in GW heuristic always is a "good"/biased one
int	local_search	try to improve every primal feasible solution found by GW via local search 0: no local search 1: one-opt local search 2: one-opt and two-opt local search
int	variable_fixing	0: no variable fixing 1: variable fixing

Appendix B

Exactly-one and diagonal projections

B.1 Exactly-one and diagonal projections

Without loss of generality, let us consider the first exactly-one constraint, *i.e.*, we focus on the first d_1 entries of the vector b. Let $M = \begin{pmatrix} bb^\top & b \\ b^\top & 1 \end{pmatrix}$, We aim to find the unique solution to the minimization problem:

$$\begin{aligned} \min_{Y} \ & \|Y - M\|_F^2 \\ \text{s.t} \ & \langle U_1, Y \rangle = 1 \\ & \langle D_i, Y \rangle = 0, \ i = 1, \dots, d_1. \end{aligned} \tag{B.1}$$

If M is already feasible, then we have the trivial solution $Y = M$. We rewrite the optimization problem (B.1):

$$\begin{aligned} \min_{Y} \ & 2\sum_{i=1}^{d_1}(Y_{id+1} - M_{id+1})^2 + \sum_{i=1}^{d_1}(Y_{ii} - M_{ii})^2 \\ \text{s.t} \ & \sum_{i=1}^{d_1} Y_{id+1} = 1 \\ & Y_{id+1} = Y_{ii}, \ i = 1, \dots, d_1, \end{aligned} \tag{B.2}$$

or equivalently:

$$\begin{aligned} \min_{Y} \ & 2\sum_{i=1}^{d_1}(Y_{id+1} - M_{id+1})^2 + \sum_{i=1}^{d_1}(Y_{id+1} - M_{ii})^2 \\ \text{s.t} \ & \sum_{i=1}^{d_1} Y_{id+1} = 1 \end{aligned} \tag{B.3}$$

The associated Lagrangian is:

$$L(\lambda, Y) = 2\sum_{i=1}^{d_1}(Y_{id+1} - M_{id+1})^2 + \sum_{i=1}^{d_1}(Y_{id+1} - M_{ii})^2 + \lambda(\sum_{i=1}^{d_1} Y_{id+1} - 1). \tag{B.4}$$

Since the problem is convex, KKT conditions are necessary and sufficient for optimality. Therefore, we have to solve the following system of equations:

$$\begin{cases} \frac{\partial L}{\partial Y_{id+1}}(\lambda, Y) = 4(Y_{id+1} - M_{id+1}) + 2(Y_{id+1} - M_{ii}) + \lambda = 0, \ i = 1, \dots, d_1 \\ \sum_{i=1}^{d_1} Y_{id+1} = 1 \end{cases} \tag{B.5}$$

From the stationnarity conditions, we get:

$$Y_{id+1} = \frac{2}{3} M_{id+1} + \frac{1}{3} M_{ii} - \frac{\lambda}{6}.$$

Plugging it back in the constraint:

$$\sum_{i=1}^{d_1} \left(\frac{2}{3} M_{id+1} + \frac{1}{3} M_{ii} - \frac{\lambda}{6} \right) = 1$$

$$\lambda = \frac{6}{d_1} \left(\sum_{i=1}^{d_1} \left(\frac{2}{3} M_{id+1} + \frac{1}{3} M_{ii} \right) - 1 \right).$$

Finally we get:

$$Y_{id+1} = \frac{2}{3} M_{id+1} + \frac{1}{3} M_{ii} - \frac{1}{d_1} \left(\sum_{i=1}^{d_1} \left(\frac{2}{3} M_{id+1} + \frac{1}{3} M_{ii} \right) - 1 \right). \tag{B.6}$$

Appendix C

French summary

C.1 Introduction

La résolution pratique de problèmes combinatoires est un défi depuis de nombreuses décennies. En principe, ces problèmes sont omniprésents. Ils sont au cœur de certaines disciplines de l'intelligence artificielle, comme le raisonnement automatique. L'ordinateur apprend "comment raisonner" à partir d'un ensemble de règles spécifiées par le programmeur. Ces règles sont guidées par la logique formelle. Ce paradigme a été utilisé avec succès pour prouver formellement des conjectures mathématiques qui étaient restées ouvertes pendant des décennies [Heule et al., 2016, Brakensiek et al., 2022].

Dans cette thèse, notre travail se concentre sur un cadre qui peut être considéré comme une extension du problème de satisfaction de contraintes (CSP). Pour ces problèmes, nous devons traiter trois types d'objets : les variables, les domaines discrets et les contraintes. Les variables sont associées à des domaines discrets dans lesquels elles prennent leurs valeurs. Les contraintes décrivent des exigences sur un ensemble d'une ou plusieurs variables. Par exemple, une contrainte peut interdire une combinaison particulière de valeurs. Notre objectif est alors de poser la question suivante : pouvons-nous trouver une affectation de toutes les variables qui satisfasse toutes les contraintes ? Avec ces ingrédients, nous pouvons modéliser un large éventail de problèmes allant de l'assignation de fréquences radio [Koster, 1999] au célèbre Sudoku. Les contraintes sont considérées comme des contraintes dures. Si au moins une des contraintes est violée, la réponse à la question pour cette affectation particulière est fausse.

Notre cadre, les modèles graphiques (GM), possède quelques propriétés qui les rendent intéressants à étudier. Les GM sont définis par les trois mêmes ingrédients que les CSP, mais les contraintes sont remplacées par des contraintes "douces". Ces contraintes sont des fonctions à valeur entière ou réelle qui prennent un ensemble fini de valeurs non négatives. La valeur retournée par une telle contrainte pour une affectation particulière des variables correspond à une violation, ou à un coût que nous devons payer si nous choisissons cette affectation. Un GM définit une fonction jointe sur l'ensemble des variables qui donne le coût total de toutes les contraintes. Une question naturelle sur les modèles graphiques consiste à trouver une affectation des variables qui minimise la fonction jointe. La particularité de cette fonction jointe est qu'elle peut être décrite comme une combinaison de "petites" fonction, ce qui la rend intéressante à des fins d'optimisation.

L'aspect combinatoire de ces problèmes les rend difficiles à résoudre. En effet, le

nombre de combinaisons possibles pour évaluer les questions d'intérêt est exponentiel dans le nombre de variables. Une idée pour surmonter ce problème est de plonger dans le monde continu. Les domaines finis sont maintenant remplacés par des domaines continus, ce qui nous donne accès à toute la boîte à outils de l'optimisation continue, et notamment, à des algorithmes de résolution en temps polynomial. Cette transformation ne fournit qu'une approximation du problème original. Cependant, elle fournit certaines informations qui peuvent nous aider à résoudre le problème exactement. Dans cette thèse, nous avons décidé d'approximer les problèmes combinatoires avec la programmation semi-définie (SDP). La programmation semi-définie est une sous-classe de l'optimisation convexe où les variables sont des matrices.

Notre principale motivation est d'explorer l'efficacité de ces méthodes pour approcher l'optimum de la fonction conjointe définie par un GM. A cet égard, nous avons généralement un compromis entre la qualité de l'approximation et l'effort consacré à sa résolution. pour la résoudre. En général, travailler avec des matrices est coûteux en termes de mémoire et de nombre d'opérations. nombre d'opérations. Cependant, nous verrons qu'il existe des méthodes pour les traiter efficacement. les manipuler efficacement.

C.2 Organisation du manuscrit

Cette thèse est organisée en 4 chapitres.

Le Chapitre 1 est une introduction aux modèles graphiques (GM). Nous donnons une brève étude des aspects théoriques et des requêtes sur les modèles graphiques. En particulier, nous nous intéressons à une requête d'optimisation qui correspond à minimiser la fonction jointe définie par le GM. Ce problème d'optimisation est NP-Complet. Nous donnons un aperçu des différents outils théoriques et pratiques qui ont été développés pour résoudre ce problème en pratique.

Le Chapitre 2 présente deux nouvelles relaxations pour le problème du Maximum A Posteriori (MAP) sur les modèles graphiques discrets binaires. Le problème combinatoire original est relaché en deux programmes semi-définis, que nous résolvons en utilisant des méthodes efficaces de rang faible. Le premier solveur est basé sur la mixing method, une méthode de rang faible dédiée aux SDP à contraintes diagonales. Pour la deuxième relaxation, nous avons développé un solveur qui effectue des étapes de descente en coordonnées par bloc. Pour réaliser une étape descente en coordonnées, nous devons résoudre un problème d'optimisation sur la sphère unité. Nous montrons que nous pouvons résoudre ce problème efficacement en utilisant, entre autres, des notions de trigonométrie et l'algorithme de Newton. Nous comparons nos deux méthodes avec des solveurs de pointe pour le problème MAP. Nous démontrons la qualité des solutions SDP pour certains types de problèmes. Nous démontrons également que notre deuxième méthode a de bonnes propriétés de scalabilité. Enfin, nous avons appliqué cette méthode à deux cas d'utilisation réels : l'assemblage de génomes et la conception computationnelle de protéines. Le travail de ce chapitre a été publié à la Conférence internationale sur l'apprentissage

automatique (ICML) [Durante et al., 2022].

Dans le Chapitre 3, nous explorons un nouvel ensemble de contraintes pour renforcer la relaxation SDP du problème MAP. Pour maintenir un bon compromis entre le coût de calcul et la qualité de la solution, nous avons décidé d'utiliser une méthode du premier ordre pour résoudre approximativement la relaxation. Cette méthode permet également de calculer une borne duale avec garantie. Nous comparons les résultats de cette méthode avec les deux solveurs de rang faible présentés précédemment.

Dans le Chapitre 4, nous nous intéressons à la résolution optimale de MaxCut, qui est un problème combinatoire central en théorie des graphes. La résolution de MaxCut est pertinente dans le contexte de cette thèse puisqu'il partage des liens étroits avec le problème MAP sur les GM discrets. Nous proposons un nouveau solveur SDP basé sur l'une des méthodes de rang faible que nous avons utilisées dans le deuxième chapitre. Nous montrons comment cette méthode peut être incorporée dans un algorithme de branch and cut. Ce travail est le résultat d'une collaboration commencée lors d'une visite à l'Institut Mathématique de Klagenfurt. Il sera soumis dans un journal avant la fin de l'année.

C.3 Modèles graphiques discrets

Les modèles graphiques (GM) définissent un cadre mathématique permettant de décrire de manière concise une fonction multivariée à l'aide d'un certain type de factorisation. Notre étude se limite aux fonctions de variables discrètes. Un grand nombre de problèmes en informatique, Logique, Programmation par contraintes, Apprentissage automatique, Physique statistique et Intelligence artificielle peuvent être modélisés à l'aide de modèles graphiques. Un GM est défini par un ensemble de variables et un ensemble fini de petites fonctions qui sont combinées ensemble en une fonction multivariée jointe. Les petites fonctions impliquent généralement peu de variables ou peuvent être représentées de manière concise à l'aide d'un langage spécifique.

Par exemple, les modèles graphiques avec des fonctions binaires et des variables booléennes sont utilisés dans le raisonnement automatisé. Les petites fonctions sont définies par une disjonction de variables ou de leur négation. Ces fonctions, appelées clauses, sont à leur tour combinées avec l'opérateur de conjonction pour définir la forme normale conjonctive. Ce cadre est utile pour décrire des propriétés logiques telles que les circuits logiques intégrés dans le matériel informatique. Nous pouvons envisager différentes requêtes sur ce type de modèle graphique, comme le problème SAT, qui cherche à démontrer l'existence d'une affectation satisfaisant la formule booléenne conjonctive. Si nous considérons plutôt des fonctions booléennes sur des ensembles de variables avec des domaines finis, combinées avec la conjonction, nous obtenons un réseau de contraintes (CN). La recherche d'une affectation des variables qui optimise la fonction conjointe est appelée problème de satisfaction de contraintes (CSP). Compte tenu de la taille des données d'entrée, il n'existe pas, dans le pire des cas, d'algorithme connu en temps polynomial pour résoudre ces

problèmes. SAT est le premier problème NP-complet connu [Cook, 2023], ce qui signifie que tout problème NP peut y être réduit à l'aide de réductions en temps polynomial.

Les petites fonctions peuvent également être décrites comme des tenseurs de valeurs réelles. Combinées avec l'addition ou la multiplication, nous pouvons modéliser une distribution de probabilités discrète comme cela est fait avec les réseaux bayésiens (BN). Après normalisation, les champs aléatoires de Markov (MRF) définissent également une distribution jointe sur des variables discrètes [Koller and Friedman, 2009, Bishop and Nasrabadi, 2006].

En résumé, la définition générale des modèles graphiques couvre une variété de cadres bien étudiés, y compris les réseaux de contraintes [Rossi et al., 2006], la logique propositionnelle [Biere et al., 2009], les modèles d'indépendance additive généralisée [Bacchus and Grove, 2013], la logique propositionnelle pondérée, les champs aléatoires de Markov [Kindermann and Snell, 1980, Koller and Friedman, 2009], les réseaux bayésiens [Koller and Friedman, 2009] (MRF avec des tenseurs normalisés décrivant des probabilités conditionnelles, organisés selon des graphes acycliques directs), réseaux de contraintes possibilistes et floues [Dubois et al., 1993].

C.3.1 Notations, définitions

Nous donnons maintenant la définition formelle des modèles graphiques discrets [Cooper et al., 2020].

Definition 23. *Modèle graphique. Un modèle graphique discret $\mathcal{M} = \langle X, \Phi \rangle$ avec un co-domaine B et un opérateur de combinaison $\oplus$ est défini par:*

- *Un ensemble X de n variables, chaque variable est associée à un domaine discret.*
- *Un ensemble de fonctions (ou potentiels) Φ. Chaque fonction $\theta_S \in \Phi$ est une fonction de $D_S \to B$. S est appelé le scope de la fonction et $|S|$ est son arité.*

$\mathcal{M}$ définit une fonction jointe:

$$\begin{array}{rccc} \Theta_{\mathcal{M}}: & D_X & \longrightarrow & B \\ & v & \longmapsto & \bigoplus\limits_{\theta_S \in \Phi} \theta_S(v[S]). \end{array}$$

Dans cette thèse, notre travail se concentre principalement sur les modèles graphiques additifs également appelés Réseaux de Fonctions de Coûts (CFNs) [Dechter, 2022, Cooper et al., 2020]. Le co-domaine B est l'ensemble des entiers positifs borné par $\top \in \mathbb{N} \cup \{+\infty\}$. L'opérateur $\oplus$ est défini par l'addition bornée $a +^{\top} b = \min(\top, a + b)$.

Definition 24. *Réseau de Fonctions de Coûts. Un réseau de fonctions de coûts (CFN) est un modèle graphique $\mathcal{C} = \langle X, C \rangle$ avec:*

- *Un ensemble $X = (x_1, \ldots, x_n)$ de n variables discrètes. Chaque variable x_i prend ses valeurs dans le domaine D_i.*

- *Un ensemble C de fonctions de coûts. Une fonction $c_S \in C$ associe à chaque tuple de D_S un élément du co-domaine $\{0, \ldots, \top\}$.*

$\mathcal{C}$ définit une fonction jointe:

$$\begin{array}{rccc} C_{\mathcal{C}} : & D_X & \longrightarrow & \{0, \ldots, \top\} \\ & v & \longmapsto & \sum_{c_S \in C}^{\top} c_S(v[S]). \end{array} \tag{C.1}$$

En raison de leur expressivité, les CFN peuvent être utilisés pour modéliser de nombreux types de modèles graphiques. Les fonctions de coûts dont les valeurs sont comprises dans $\{0, \ldots, \top\}$ sont des contraintes souples. Pour un $c_S \in C$ particulier et une séquence $v \in D_X$, la valeur $c_S(v[S])$ peut être considérée comme une violation de la contrainte. L'élément maximum $\top$ encode les tuples interdits, c'est-à-dire que les affectations partielles avec un coût $\top$ sont interdites. Lorsqu'une fonction de coûts prend ses valeurs dans $\{0, \top\}$, il s'agit d'une contrainte dure.

Tout au long du manuscrit, nous limitons notre étude aux modèles graphiques binaires, *i.e.*, les petites fonctions ont toutes une arité d'au plus 2. Nous disposons alors d'une représentation naturelle des fonctions à l'aide de vecteurs et de matrices dont les valeurs se trouvent dans le co-domaine B. Lorsqu'ils contiennent un petit nombre de valeurs différentes de l'élément minimal $\bot$, on peut utiliser des structures éparses pour stocker l'information de manière concise.

Une autre classe importante est celle des modèles graphiques stochastiques pour lesquels le co-domaine est l'intervalle réel $[0, +\infty]$. Dans ce cas, l'opérateur $\oplus$ est la multiplication et les fonctions décrivent des probabilités marginales non normalisées sur un sous-ensemble de variables.

Definition 25. *Champs de Markov Aléatoires. Un champ de Markov aléatoire (MRF) [Koller and Friedman, 2009] est un modèle graphique $\mathcal{M} = \langle X, \Phi \rangle$ avec:*

- *Un ensemble $X = (x_1, \ldots, x_n)$ de variables aléatoires discrètes. Chaque variable aléatoire x_i prend ses valeurs dans le domaine D_i.*
- *Un ensemble Φ de potentiels. Un potentiel $\theta_S \in \Phi$ associe à chaque tuple de D_S un élément du co-domain $[0, +\infty]$.*

$\mathcal{M}$ définit une fonction jointe:

$$\begin{array}{rccc} \Theta_{\mathcal{M}} : & D_X & \longrightarrow & [0, +\infty] \\ & v & \longmapsto & \prod_{\theta_S \in \Phi} \theta_S(v[S]). \end{array} \tag{C.2}$$

C.4 Requêtes et optimisation exacte

Dans la première section, nous avons présenté les modèles graphiques discrets, un cadre puissant qui permet d'exprimer de nombreux problèmes différents. Les requêtes sur les modèles graphiques demandent de calculer des informations simples

sur la fonction jointe, telles que son minimum ou sa valeur moyenne. Ces requêtes couvrent un large éventail d'applications en intelligence artificielle ou en informatique, par exemple.

C.4.1 Optimisation sur les modèles graphiques

Une requête habituelle sur les MRF consiste à trouver l'affectation la plus probable, également appelée Maximum À Posteriori (MAP). Une distribution de probabilité peut être récupérée à partir de la fonction jointe en calculant une constante de normalisation également appelée fonction de partition.

Definition 26. *Fonction de partition. La fonction de partition Z d'un champ de Markov aléatoire $\mathcal{M} = \langle X, \Phi \rangle$ est définie comme la somme des potentiels sur l'ensemble de toutes les affectations $v \in D_X$:*

$$Z = \sum_{v \in D_S} \prod_{\theta_S \in \Phi} \theta_S(v[S]).$$

Calculer la fonction de partition est également une requête sur les MRF. Cette requête est #P-complete [Valiant, 1979]. Par la suite, nous pouvons définir la distribution de probabilité associée au MRF:

$$\begin{array}{rccl} p_{\mathcal{M}}: & D_X & \longrightarrow & [0,1] \\ & v_X & \longmapsto & P_{\mathcal{M}}(v) = \frac{1}{Z} \prod\limits_{\theta_S \in \Phi} \theta_S(v[S]). \end{array}$$

Definition 27. *Maximum À Posteriori. Le problème du Maximum À Posteriori (MAP) sur un champ de Markov aléatoire $\mathcal{M} = \langle X, \Phi \rangle$ consiste à trouver une affectation $v \in D_X$ avec la probabilité maximale:*

$$\max_{v \in D_X} P(v) = \frac{1}{Z} \prod_{\theta_S \in \Phi} \theta_S(v[S]).$$

Il convient de noter qu'à des fins d'optimisation, la fonction de partition n'est pas pertinente et que l'on peut directement travailler avec la distribution jointe non normalisée. Il existe une requête équivalente sur les CFN, à savoir le problème de satisfaction de contraintes pondérées.

Definition 28. *Problème de Satisfaction de Contraintes Pondérées. Le Problème de Satisfaction de Contraints Pondérées (WCSP) sur un CFN $\mathcal{C} = \langle X, C \rangle$ consiste à trouver une affectation $v \in D_X$ avec un coût minimal:*

$$\min_{v \in D_X} C_{\mathcal{C}}(v) = \sum_{c_S \in C}{}^{\top} c_S(v).$$

C.5 Méthode de descente en coordonnées vs descente par bloc

Dans ce chapitre, nous présentons deux méthodes pour la reformulation quadratique 0/1 avec contraintes du problème MAP sur des modèles graphiques discrets

binaires. La première méthode est basée sur l'idée que l'on peut reformuler un problème quadratique 0/1 avec des contraintes linéaires comme un problème MaxCut en utilisant une pénalisation des contraintes. Nous présentons ensuite une nouvelle méthode dédiée qui évite l'introduction de coefficients de pénalité importants pour modéliser le problème. Les deux problèmes d'optimisation discrets sont relachés au moyen de programmes semi-définis de rang faible. Par rapport à ce qui a été fait dans la littérature, les deux approches de rang faible peuvent être appliquées à des modèles graphiques discrets avec un nombre arbitraire d'états et des potentiels binaires arbitraires. L'objectif sous-jacent à ces deux méthodes est de calculer des bornes fortes sur le problème d'optimisation discret en utilisant des relaxations SDP tout en se rapprochant de l'efficacité de calcul des méthodes de programmation linéaire. Les deux méthodes sont ensuite comparées à des solveurs de pointe sur un ensemble de problèmes aléatoires et réels. Ce chapitre est organisé en quatre parties :

- LR-LAS 2.1 : nous présentons notre première méthode SDP de rang faible pour le problème MAP sur les MRF discrets. La formulation quadratique binaire est réduite à un problème MaxCut en utilisant le résultat de pénalisation de Lasserre [Lasserre, 2016]. Nous résolvons ensuite la relaxation SDP de base du problème MaxCut à l'aide de la mixing method [Wang et al., 2017].

- LR-BCD 2.2 : nous introduisons une nouvelle formulation SDP ainsi qu'un algorithme de descente par blocs de coordonnées dédié qui tire parti des propriétés structurelles de notre problème. Nous démontrons certaines propriétés théoriques de ce nouvel algorithme SDP de rang faible.

- Expériences 2.3 : nous comparons nos deux méthodes par rapport à des solveurs de pointe basés sur la programmation linéaire et sur des méthodes de message passing. Pour les expériences, nous considérons des problèmes aléatoires ainsi que des problèmes tirés d'applications réelles. Certains paramètres sont discutés, comme le rang de la relaxation de rang faible.

- Discussion 2.4 : enfin, nous discutons de certaines caractéristiques et propriétés théoriques supplémentaires de LR-BCD.

Les techniques d'approximation pour MaxCut basées sur la programmation semi-définie ont été largement étudiées depuis le travail fondateur de [Goemans and Williamson, 1995]. Comparées aux approches de programmation linéaire, elles fournissent souvent des bornes beaucoup plus fortes pour une variété de problèmes d'optimisation combinatoire. Cependant, l'utilisation des SDP reste limitée pour les problèmes MAP/WCSP. La raison principale réside dans le fait que les algorithmes de résolution usuels pour les SDPs, les méthodes de point intérieur, nécessitent une complexité cubique dans la taille du problème. Ainsi, ces méthodes ne passent pas à l'échelle et sont essentiellement utilisées pour de petits problèmes très difficiles. Pour cette raison, les méthodes LP approximatives sont généralement préférées car elles permettent un meilleur compromis entre la qualité des bornes et le coût de calcul.

L'utilisation de la programmation semi-définie pour le problème MAP et d'autres applications s'est améliorée grâce à l'introduction d'une approche de rang faible, non convexe, par Burer et Monteiro. Cette approche exploite les résultats de [Barvinok, 1995] et [Pataki, 1998] qui ont montré que tout problème SDP possède une solution de rang $O(\sqrt{m})$ avec m le nombre de contraintes. Cette méthode utilise une factorisation de rang faible $X = VV^\top$ de la matrice semi-définie X dans la relaxation SDP de ces problèmes combinatoires. En pratique, il a été observé depuis que le rang de la solution est souvent inférieur et certains logiciels utilise un rang constant de $O(1)$ pour la factorisation.

C.6 Renforcer les bornes avec de nouvelles contraintes

Dans le chapitre précédent, nous avons introduit deux méthodes dédiées à la résolution de deux relaxations différentes du problème MAP sur des modèles graphiques discrets. Nous avons vu que pour certains types d'instances, la relaxation résolue par LR-BCD peut être faible. Parfois, la solution de la relaxation SDP a une valeur négative, ce qui n'est pas utile puisque nous savons que la valeur optimale du problème MAP est toujours au moins positive. Dans ce chapitre, nous résolvons ce problème en introduisant de nouvelles contraintes pour notre relaxation SDP. Une idée naturelle pour obtenir une relaxation plus forte est de mieux caractériser l'ensemble réalisable du problème MAP. L'ajout de nouvelles contraintes à notre formulation a un certain coût. Dans ce chapitre, nous avons choisi de favoriser la qualité des bornes à l'efficacité de calcul. Comme il n'est pas certain que nous puissions généraliser LR-BCD pour traiter d'autres contraintes que la contrainte "exactement un", nous avons décidé d'explorer d'autres méthodes pour résoudre la relaxation SDP. Comme nous introduisons maintenant un nombre potentiellement important de contraintes pour représenter notre ensemble réalisable, les méthodes de points intérieurs sont trop coûteuses en termes de calcul pour la relaxation que nous considérons. Nous nous sommes donc concentrés sur les méthodes du premier ordre, en particulier la méthode des directions alternées (ADMM). En général, ces méthodes peuvent nécessiter de nombreuses itérations pour converger vers l'optimum exact. Cependant, elles sont très efficaces pour obtenir de bonnes solutions approximatives en un temps relativement court par rapport aux méthodes de points intérieurs. Ce chapitre est organisé en 4 parties :

- Une meilleure représentation de l'ensemble réalisable 3.1 : nous présentons les nouvelles contraintes que nous avons décidé d'ajouter à notre relaxation. Ces contraintes bénéficient de certaines propriétés qui sont utiles pour la résolution avec ADMM.

- Résolution avec ADMM 3.2 : nous résolvons la nouvelle relaxation SDP du problème MAP avec une variante d'ADMM.

- Contraintes globales 3.3 : nous discutons de l'ajout de contraintes globales, qui peuvent être intéressantes pour modéliser certains problèmes.

- Résultats 3.4 : nous comparons ADMM avec LR-LAS et LR-BCD sur un ensemble de problèmes aléatoires et réels. Pour certains cas, nous incluons également une comparaison avec le solveur exact toulbar2. Nous discutons également de l'efficacité des méthodes itératives pour calculer la projection sur le cône semi-défini.

C.6.1 Discussion

Dans ce chapitre, nous avons essayé de favoriser la qualité des bornes par rapport à l'efficacité de calcul en renforçant la relaxation que nous avons considérée au chapitre 2. En ajoutant les contraintes gangster et de positivité, les bornes sont maintenant beaucoup plus fortes sur une variété de problèmes. Comme le nombre de contraintes peut devenir très important, les méthodes de points intérieurs ne conviennent pas pour traiter ces relaxations SDP [Wiegele and Zhao, 2022]. Nous avons donc utilisé un algorithme ADMM qui peut traiter efficacement notre ensemble de contraintes. Cependant, ADMM est toujours limité par des appels à un algorithme de calcul de valeurs et vecteurs propres pour les projections sur le cône semi-défini. Comme le montrent les expériences avec les problèmes de protéines, les instances de taille $d > 1000$ deviennent difficiles pour ADMM. Nous avons discuté de l'utilisation de méthodes itératives pour le calcul de vecteurs et valeurs propres afin d'accélérer les projections, mais cela ne convient pas à tous les types d'instances. Finalement, nous espérons combiner les méthodes de rang faible avec ADMM pour obtenir le meilleur des deux mondes.

C.7 Résolution de MaxCut avec des bornes SDP de rang faible

Ce travail est réalisé en collaboration avec Jan Schwiddessen. Jan est un étudiant en doctorat que j'ai rencontré pendant mon séjour à l'Institut mathématique de Klagenfurt. Nous nous intéressions tous deux aux méthodes de résolution SDP de rang faible. Après quelques discussions, Jan a proposé l'idée de développer un nouveau solveur et l'aventure a commencé. Ce fut un grand plaisir de collaborer avec lui sur ce projet, et nous prévoyons de soumettre un article avant la fin de cette année.

Dans ce chapitre, nous nous intéressons maintenant à la résolution exacte de problèmes combinatoires. Nous présentons `MixCut` un solveur exact pour le problème MaxCut basé sur la programmation semi-définie. Il utilise une approche branch-and-cut avec des inégalités hypermétriques pour renforcer la relaxation de base de MaxCut. La procédure de bornage est efficacement réalisée par un algorithme SDP de rang faible, la mixing method [Wang et al., 2017]. Nous proposons un ensemble de nouvelles décisions de branchement basées sur des informations provenant des solutions primales et duales, ainsi qu'une nouvelle heuristique pour calculer de bonnes solutions entières. Expérimentalement, nous montrons que notre méthode est capable d'atteindre un meilleur compromis entre la qualité des bornes et le nombre total de nœuds explorés dans l'arbre de recherche. Nous comparons nos résultats

à un ensemble d'algorithmes de pointe grâce à une étude approfondie des instances MaxCut denses de la bibliothèque BiqMac [Wiegele, 2007].

C.7.1 Introduction

MaxCut est un problème central de l'optimisation combinatoire. Étant donné un graphe pondéré, il vise à trouver une partition des sommets en deux ensembles disjoints de sorte que la somme des poids des arêtes entre les ensembles soit maximisée. Il a fait l'objet d'une grande attention au cours des dernières décennies en raison de son large éventail d'applications en physique et en informatique. Dans les chapitres précédents, nous avons vu que l'inférence MAP sur les GM discrets peut être réduite à un problème MaxCut au moyen de la pénalisation exacte de Lasserre [Lasserre, 2016]. MaxCut est connu pour être NP-Difficile [Karp, 2010], et les chercheurs ont essayé plusieurs méthodes pour le résoudre, notamment des algorithmes basés sur des relaxations LP, qui sont efficaces pour les problèmes peu denses. [Goemans and Williamson, 1995] ont démontré le célèbre ratio d'approximation de 0,87856 pour les graphes avec des poids non négatifs en utilisant une relaxation SDP. Aujourd'hui, pour les instances denses, les solveurs les plus performants utilisent presque tous des relaxations semi-définies. Même si ces méthodes ont donné d'excellents résultats, la résolution optimale de MaxCut reste un défi, même pour des problèmes de taille modérée. Bien qu'elles ne soient pas traitées en profondeur dans le manuscrit, des heuristiques ont été développées pour trouver de bonnes solutions pour les problèmes à grande échelle [Benlic and Hao, 2013]. Récemment, MaxCut a connu un regain d'intérêt avec les ordinateurs quantiques et l'algorithme d'optimisation approximative quantique (QAOA) [Farhi et al., 2014, Guerreschi and Matsuura, 2019].

C.7.2 Discussion

Pour améliorer encore notre solveur, nous aimerions explorer plusieurs stratégies. Tout d'abord, en ce qui concerne les contraintes, seules les inégalités triangulaires sont incluses. Pour les instances plus grandes, la relaxation peut ne pas être assez forte, par conséquent, le nombre de nœuds ouverts dans l'arbre de recherche augmente rapidement et l'espace de recherche devient trop grand pour prouver efficacement l'optimalité. Cela peut également entraîner des problèmes de mémoire puisque nous stockons des informations à chaque nœud. Afin de renforcer la relaxation et d'améliorer la procédure de bornage, nous souhaitons ajouter d'autres inégalités hypermétriques. Suite au travail effectué dans BiqBin et Madam, nous avons vu que les deux solveurs fournissent de meilleures bornes en incluant des inégalités pentagonales et heptagonales. La séparation des inégalités triangulaires peut être effectuée par énumération complète en $O(n^3)$, pour les deux autres types il y a un total de $16\binom{n}{5}$ et $64\binom{n}{7}$ inégalités respectivement. Pour que notre solveur reste efficace, l'étape suivante consiste à trouver une bonne heuristique pour séparer les inégalités pentagonales et heptagonales violées. Dans le cas de BiqBin et Madam, cela est fait en résolvant approximativement un problème d'affectation quadratique (QAP) avec un recuit simulé.

Une idée naturelle pour àméliorer les performances de notre solveur est d'utiliser

la parallélisation pour l'algorithme de branch-and-cut. Cette méthode est déjà exploitée dans de nombreux solveurs exacts tels que BiqBin et Madam.

C.8 Conclusion

Dans cette thèse, nous avons exploré différentes méthodes pour approximer la solution du problème MAP sur des modèles graphiques discrets binaires. La résolution pratique de problèmes combinatoires est un défi majeur dans de nombreux domaines. Dans le chapitre sur l'état de l'art, nous avons présenté les modèles graphiques, un cadre qui couvre une variété d'applications allant de la programmation par contraintes à la conception computationnelle de protéines. Notre principale motivation était de concevoir une méthode d'optimisation qui offre un bon compromis entre efficacité et qualité. Par efficacité, nous entendons la capacité à s'adapter à des problèmes de grande taille. Par qualité, nous entendons la capacité à donner une approximation précise de la solution discrète.

Dans le chapitre 2, nous avons introduit deux relaxations continues du problème MAP sur les modèles graphiques discrets binaires. Pour la première relaxation, nous avons été guidés par les progrès récents de la résolution du problème MaxCut. Nous avons utilisé une réduction exacte du problème MAP vers MaxCut proposée par Lasserre. Plutôt que d'utiliser les méthodes habituelles de points intérieurs pour résoudre la relaxation SDP de MaxCut, nous avons utilisé une méthode de rang faible basée sur une descente en coordonnées. Cependant, nous avons réalisé que cette méthode n'était pas adaptée pour traiter les grands facteurs introduits par la pénalisation. Pour résoudre ce problème, nous avons proposé une deuxième relaxation ainsi qu'une nouvelle méthode SDP de rang faible. Notre méthode repose sur une descente par blocs de coordonnées. Pour effectuer une étape de descente par bloc de coordonnées, nous résolvons un problème d'optimisation sur la sphère unité à l'aide de la méthode de Newton.

Nous avons démontré quelques propriétés de notre algorithme. LR-BCD est toujours décroissant et les points d'adhérence de la suite des itérés sont des points fixes pour la méthode. Nous avons également montré qu'à chaque itération, la matrice de rang faible appartient toujours à l'ensemble réalisable, *i.e.*, elle satisfait toujours toutes les contraintes. Expérimentalement, nous avons comparé LR-BCD au solveur commercial Mosek basé sur une méthode de points intérieurs. Nous avons constaté que LR-BCD convergeait rapidement vers l'optimum global du problème SDP. Sur certaines instances, nous obtenons une accélération de deux ordres de grandeur pour converger vers une solution de bonne qualité. Afin de trouver des solutions entières réalisables, nous avons également légèrement modifié l'heuristique de Goemans et Williamson. En utilisant une simple recherche locale, nous avons pu produire de très bonnes solutions, parfois meilleures que celles des algorithmes spécialisés. En raison de ses différentes qualités, LR-BCD a été utilisé dans différents projets. En particulier, il a été ajouté à un pipeline d'apprentissage profond développé par Marianne Defresne pour la conception computationnelle de protéines.

Cependant, au cours des expériences, nous nous sommes rendu compte que

pour certains problèmes notre relaxation n'était pas assez forte. Nous avons décidé d'ajouter des contraintes supplémentaires afin d'obtenir une meilleure représentation de l'ensemble réalisable du problème combinatoire. Comme il n'était pas certain que LR-BCD puisse être étendu pour traiter des contraintes plus générales, nous avons décidé d'utiliser une méthode différente pour résoudre la relaxation SDP correspondante. Cette méthode a été suggérée par l'équipe d'optimisation du laboratoire de mathématiques de Klagenfurt pendant mon séjour d'un mois à l'institut. Il s'agit d'une variante de la méthode des multiplicateurs alternés, une méthode du premier ordre qui donne de bons résultats pour la résolution des SDP.

Pendant mon séjour à Klagenfurt, j'ai également entamé une collaboration avec Jan Schwiddessen, étudiant en doctorat sous la direction d'Angelika Wiegele. Cette collaboration a donné lieu au travail du dernier chapitre. Nous nous sommes intéressés à la résolution exacte de problèmes combinatoires. Nous avons proposé un nouveau solveur MaxCut exact basé sur une méthode SDP de rang faible. Nous utilisons un algorithme de branch-and-cut avec une procédure de bornage qui est efficacement réalisée grâce à la combinaison de la mixing et de la bundle method. Pour notre solveur, nous avons développé de nouvelles décisions de branchement ainsi qu'une version non aléatoire de l'heuristique de Goemans et Williamson. Nous démontrons la supériorité de notre solveur par rapport aux méthodes existantes sur une variété d'instances MaxCut de la bibliothèque BiqMac.

C.9 Perspectives

Travaux futurs sur LR-BCD L'algorithme d'optimisation central présenté dans cette thèse peut être amélioré suivant plusieurs directions. Tout d'abord, il serait intéressant de dériver d'autres propriétés de convergence pour LR-BCD, *e.g.*, la convergence vers l'optimum global. À la fin du chapitre 2, nous avons caractérisé les points fixes de l'algorithme. Comme dans l'article de la mixing method, un point de départ pour la preuve de convergence consisterait à analyser le jacobien de l'application qui représente la méthode LR-BCD. Cependant, ce n'est pas une tâche facile car nous n'avons pas de solution analytique pour les problèmes d'optimisation par blocs.

Ensuite, si nous voulons utiliser LR-BCD dans un algorithme branch-and-bound, nous devons calculer des bornes avec des garanties. Pour l'instant, LR-BCD est un algorithme primal, de telle sorte que la valeur renvoyée ne donne pas de certification sur la valeur de la borne inférieure. Notre objectif est de trouver une méthode efficace pour obtenir une solution réalisable du problème dual. Dans le chapitre 4, nous avons vu que nous pouvons produire une solution réalisable duale pour une relaxation SDP particulière de MaxCut en utilisant la solution primale renvoyée par la mixing method. Idéalement, nous aimerions disposer d'une méthode similaire pour LR-BCD.

Une autre direction consiste à améliorer la qualité de la relaxation résolue par LR-BCD. Puisque la relaxation MAP résolue par LR-BCD n'est pas toujours forte, nous aimerions étendre LR-BCD pour traiter plus de contraintes. Un moyen naturel

est d'utiliser une combinaison de LR-BCD et de la bundle method. Par exemple, nous pourrions ajouter des contraintes de positivité pour éviter les solutions négatives. Dans le cadre de problèmes avec des variables $\{-1, 1\}$, ces contraintes sont similaires aux inégalités hypermétriques. Ces contraintes peuvent être dualisées comme dans le chapitre sur le solveur MixCut et nous utiliserions LR-BCD pour résoudre un SDP avec la contrainte "exactement un".

Nous avons également pensé à combiner ADMM avec des méthodes de rang faible. En effet, la factorisation de rang faible nous permettrait de nous débarrasser des projections sur le cône semi-défini. Les chercheurs ont commencé à dériver des propriétés de convergence pour ADMM dans des contextes non-convexes [Wang et al., 2018]. Récemment, Chen et al. [Chen and Goulart, 2023] ont proposé une combinaison de l'agorithme ADMM avec la factorisation de Burer-Monteiro pour résoudre la relaxation SDP de base du problème MaxCut. Cependant, cette méthode est limitée aux SDP à contraintes diagonales et l'extension aux contraintes générales semble être un problème difficile.

Accélération de LR-BCD grâce à la parallelisation Enfin, en ce qui concerne le LR-BCD, nous avons pensé à une étape de prétraitement pour accélérer l'algorithme. L'idée est d'utiliser la parallélisation pour résoudre plusieurs problèmes d'optimisation par blocs en même temps. Une fois de plus, ce résultat repose sur la structure du problème MAP. Si nous considérons un GM binaire avec un graphe associé $G = (X, E)$, les sommets qui ne partagent pas une arête peuvent être mis à jour en parallèle. En effet, dans la relaxation SDP que nous avons présentée, les deux variables correspondantes partagent un bloc zéro dans la matrice objective. Dans ce cas, la mise à jour d'une variable n'a donc aucun effet sur la mise à jour de l'autre variable. Les variables indépendantes peuvent être détectées en appliquant un algorithme de coloration sur le graphe G. Pour le problème de coloration, il existe une heuristique efficace appelée DSATUR avec une complexité $O(n^2)$ où n est le nombre de sommets. A la fin, les variables ayant les mêmes couleurs peuvent être mises à jour en parallèle. Le nombre de blocs parallèles correspond au nombre de couleurs retournées par l'heuristique. Cette étape de prétraitement serait donc plus efficace sur des instances peu denses où le nombre chromatique du graphe a plus de chance d'être petit.

LR-BCD sera bientôt intégré dans le solveur exact toulbar2 [Hurley et al., 2016] en tant qu'étape de prétraitement. Les opérations au sein de LR-BCD ne sont pas trop complexes, *e.g.*, produits matrice-vecteur, produits matrice-matrice. Nous avons implémenté nos propres routines d'algèbre linéaire pour éviter des dépendances supplémentaires dans toulbar2.

Amélioration du solveur MixCut En ce qui concerne le solveur MixCut que nous développons avec Jan Schwiddessen, nous prévoyons d'ajouter des inégalités pentagonales et heptagonales à notre relaxation. Une fois terminé, notre travail sera soumis à un journal, si possible avant la fin de l'année.

Bibliography

P-A Absil, Christopher G Baker, and Kyle A Gallivan. Trust-region methods on Riemannian manifolds. *Foundations of Computational Mathematics*, 7(3):303–330, 2007. (Cited on page 113.)

Akshay Agrawal, Robin Verschueren, Steven Diamond, and Stephen Boyd. A rewriting system for convex optimization problems. *Journal of Control and Decision*, 5 (1):42–60, 2018. (Cited on page 75.)

Kurt M Anstreicher. Semidefinite programming versus the reformulation-linearization technique for nonconvex quadratically constrained quadratic programming. *Journal of Global Optimization*, 43:471–484, 2009. (Cited on page 33.)

MOSEK ApS. *MOSEK Fusion API for Python 9.3.12*, 2022. URL `https://docs.mosek.com/latest/pythonfusion/index.html`. (Cited on pages 38, 78 and 100.)

Fahiem Bacchus and Adam J Grove. Graphical models for preference and utility. *arXiv preprint arXiv:1302.4928*, 2013. (Cited on pages 6 and 148.)

Egon Balas, Sebastian Ceria, Gérard Cornuéjols, and N Natraj. Gomory cuts revisited. *Operations Research Letters*, 19(1):1–9, 1996. (Cited on page 20.)

Alexander I. Barvinok. Problems of distance geometry and convex properties of quadratic maps. *Discrete & Computational Geometry*, 13(2):189–202, 1995. (Cited on pages 44, 51, 52 and 152.)

Una Benlic and Jin-Kao Hao. Breakout local search for the max-cut problem. *Engineering Applications of Artificial Intelligence*, 26(3):1162–1173, 2013. (Cited on pages 112 and 154.)

Umberto Bertele and Francesco Brioschi. On non-serial dynamic programming. *J. Comb. Theory, Ser. A*, 14(2):137–148, 1973. (Cited on page 15.)

Armin Biere, Marijn Heule, and Hans van Maaren. *Handbook of satisfiability*, volume 185. IOS press, 2009. (Cited on pages 6 and 148.)

Alain Billionnet. *Optimisation Discrète, de la modélisation à la résolution par des logiciels de programmation mathématique*. Dunod, January 2007. URL `https://hal.science/hal-01125300`. (Cited on page 32.)

Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006. (Cited on pages 6 and 148.)

Endre Boros and Peter L Hammer. Pseudo-Boolean optimization. *Discrete applied mathematics*, 123(1-3):155–225, 2002. (Cited on pages 15, 35 and 37.)

Nicolas Boumal, Vlad Voroninski, and Afonso Bandeira. The non-convex Burer-Monteiro approach works on smooth semidefinite programs. *Advances in Neural Information Processing Systems*, 29, 2016. (Cited on pages 45 and 113.)

Nicolas Boumal, Vladislav Voroninski, and Afonso S Bandeira. Deterministic Guarantees for Burer-Monteiro Factorizations of Smooth Semidefinite Programs. *Communications on Pure and Applied Mathematics*, 73(3):581–608, 2020. (Cited on pages 44 and 115.)

Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. Distributed optimization and statistical learning via the alternating direction method of multipliers. *Foundations and Trends® in Machine learning*, 3(1):1–122, 2011. (Cited on pages 42 and 100.)

Stephen P Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004. (Cited on pages 23 and 26.)

Joshua Brakensiek, Marijn Heule, John Mackey, and David Narváez. The resolution of Keller's conjecture. *Journal of Automated Reasoning*, 66(3):277–300, 2022. (Cited on pages 1 and 145.)

Samuel Burer and Renato DC Monteiro. A nonlinear programming algorithm for solving semidefinite programs via low-rank factorization. *Mathematical Programming*, 95(2):329–357, 2003. (Cited on pages 44, 52 and 113.)

Samuel Burer and Renato DC Monteiro. Local minima and convergence in low-rank semidefinite programming. *Mathematical programming*, 103(3):427–444, 2005. (Cited on page 44.)

Juan S Campos, Ruth Misener, and Panos Parpas. Partial Lasserre relaxation for sparse Max-Cut. *Optimization and Engineering*, pages 1–22, 2022. (Cited on pages 38 and 113.)

Martina Cerulli, Marianna De Santis, Elisabeth Gaar, and Angelika Wiegele. Improving ADMMs for solving doubly nonnegative programs through dual factorization. *4OR*, 19:415–448, 2021. (Cited on pages 44 and 98.)

Jonas Charfreitag, Michael Jünger, Sven Mallach, and Petra Mutzel. McSparse: Exact solutions of sparse maximum cut and sparse unconstrained binary quadratic optimization problems. In *2022 Proceedings of the Symposium on Algorithm Engineering and Experiments (ALENEX)*, pages 54–66. SIAM, 2022. (Cited on page 113.)

Jonas Charfreitag, Sven Mallach, and Petra Mutzel. Integer Programming for the Maximum Cut Problem: A Refined Model and Implications for Branching. In *SIAM Conference on Applied and Computational Discrete Algorithms (ACDA23)*, pages 63–74. SIAM, 2023. (Cited on page 132.)

Caihua Chen, Bingsheng He, Yinyu Ye, and Xiaoming Yuan. The direct extension of ADMM for multi-block convex minimization problems is not necessarily convergent. *Mathematical Programming*, 155(1-2):57–79, 2016. (Cited on page 43.)

Yuwen Chen and Paul Goulart. Burer-Monteiro ADMM for large-scale SDPs, 2023. (Cited on pages 44, 136 and 157.)

Stephen A Cook. The complexity of theorem-proving procedures. In *Logic, Automata, and Computational Complexity: The Works of Stephen A. Cook*, pages 143–152. 2023. (Cited on pages 6 and 148.)

Martin Cooper, Simon de Givry, and Thomas Schiex. Graphical models: queries, complexity, algorithms. *Leibniz International Proceedings in Informatics*, 154:4–1, 2020. (Cited on pages xii, 6, 7, 15, 21, 31 and 148.)

Martin C Cooper, Simon De Givry, and Thomas Schiex. Optimal Soft Arc Consistency. In *IJCAI*, volume 7, pages 68–73, 2007. (Cited on page 21.)

Martin C Cooper, Simon De Givry, Martí Sánchez-Fibla, Thomas Schiex, and Matthias Zytnicki. Virtual Arc Consistency for Weighted CSP. In *AAAI*, pages 253–258, 2008. (Cited on page 21.)

Martin C Cooper, Simon De Givry, Martı Sánchez, Thomas Schiex, Matthias Zytnicki, and Tomas Werner. Soft arc consistency revisited. *Artificial Intelligence*, 174(7-8):449–478, 2010. (Cited on page 21.)

IBM ILOG Cplex. V12. 1: User's manual for cplex. *International Business Machines Corporation*, 46(53):157, 2009. (Cited on page 20.)

Jane K Cullum and Ralph A Willoughby. *Lanczos algorithms for large symmetric eigenvalue computations: Vol. I: Theory*. SIAM, 2002. (Cited on page 43.)

George Dantzig. *Linear programming and extensions*. Princeton university press, 1963. (Cited on pages xi, 18, 19 and 20.)

Frank de Meijer, Renata Sotirov, Angelika Wiegele, and Shudian Zhao. Partitioning through projections: Strong SDP bounds for large graph partition problems. *Computers & Operations Research*, 151:106088, 2023. (Cited on pages 99 and 100.)

Rina Dechter. *Reasoning with probabilistic and deterministic graphical models: Exact algorithms*. Springer Nature, 2022. (Cited on pages 7 and 148.)

Marianne Defresne, Sophie Barbe, and Thomas Schiex. Scalable Coupling of Deep Learning with Logical Reasoning. In Edith Elkind, editor, *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, pages 3615–3623. International Joint Conferences on Artificial Intelligence Organization, 8 2023. doi: 10.24963/ijcai.2023/402. URL https://doi.org/10.24963/ijcai.2023/402. Main Track. (Cited on page 84.)

Michel Marie Deza and Monique Laurent. *Geometry of cuts and metrics*, volume 15 of *Algorithms and Combinatorics*. Springer, Heidelberg, 2010. ISBN 978-3-642-04294-2. doi: 10.1007/978-3-642-04295-9. URL https://doi.org/10.1007/978-3-642-04295-9. First softcover printing of the 1997 original [MR1460488]. (Cited on pages 113 and 115.)

II Dikin. Iterative solution of problems of linear and quadratic programming. In *Doklady Akademii Nauk*, volume 174, pages 747–748. Russian Academy of Sciences, 1967. (Cited on page 20.)

Didier Dubois, Hélene Fargier, and Henri Prade. The calculus of fuzzy restrictions as a basis for flexible constraint satisfaction. In *[Proceedings 1993] Second IEEE International Conference on Fuzzy Systems*, pages 1131–1136. IEEE, 1993. (Cited on pages 6 and 148.)

Valentin Durante, George Katsirelos, and Thomas Schiex. Efficient Low Rank Convex Bounds for Pairwise Discrete Graphical Models. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 5726–5741. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/durante22a.html. (Cited on pages 2 and 147.)

Jonathan Eckstein and Dimitri P Bertsekas. On the Douglas-Rachford splitting method and the proximal point algorithm for maximal monotone operators. *Mathematical programming*, 55:293–318, 1992. (Cited on page 104.)

Sourour Elloumi and Amélie Lambert. Global solution of non-convex quadratically constrained quadratic programs. *Optimization methods and software*, 34(1):98–114, 2019. (Cited on page 34.)

Murat A Erdogdu, Yash Deshpande, and Andrea Montanari. Inference in graphical models via semidefinite programming hierarchies. *Advances in Neural Information Processing Systems*, 30, 2017. (Cited on page 38.)

Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028*, 2014. (Cited on pages 112 and 154.)

Roy Frostig, Sida Wang, Percy S Liang, and Christopher D Manning. Simple MAP inference via low-rank relaxations. *Advances in Neural Information Processing Systems*, 27, 2014. (Cited on page 44.)

Jean Charles Gilbert. Fragments d'Optimisation Différentiable - Théories et Algorithmes. Lecture, March 2021. URL https://inria.hal.science/hal-03347060. (Cited on page 80.)

Michel X Goemans and David P Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming.

Journal of the ACM (JACM), 42(6):1115–1145, 1995. (Cited on pages 46, 50, 112, 151 and 154.)

Gene H Golub and Charles F Van Loan. *Matrix computations*. JHU press, 2013. (Cited on pages 103 and 125.)

Paul J Goulart, Yuji Nakatsukasa, and Nikitas Rontsis. Accuracy of approximate projection to the semidefinite cone. *Linear Algebra and its Applications*, 594: 177–192, 2020. (Cited on page 104.)

Gaël Guennebaud, Benoît Jacob, et al. Eigen v3. http://eigen.tuxfamily.org, 2010. (Cited on pages 74 and 103.)

Gian Giacomo Guerreschi and Anne Y Matsuura. QAOA for Max-Cut requires hundreds of qubits for quantum speed-up. *Scientific reports*, 9(1):6903, 2019. (Cited on pages 112 and 154.)

Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2023. URL `https://www.gurobi.com`. (Cited on page 20.)

Nicolo Gusmeroli and Angelika Wiegele. EXPEDIS: An exact penalty method over discrete sets. *Discrete Optimization*, 44:100622, 2022. (Cited on page 116.)

Nicolo Gusmeroli, Timotej Hrga, Borut Lužar, Janez Povh, Melanie Siebenhofer, and Angelika Wiegele. BiqBin: a parallel branch-and-bound solver for binary quadratic problems with linear constraints. *ACM Transactions on Mathematical Software (TOMS)*, 48(2):1–31, 2022. (Cited on pages 41, 113, 116, 118 and 127.)

Peter L Hammer and Abraham A Rubin. Some remarks on quadratic programming with 0-1 variables. *Revue française d'informatique et de recherche opérationnelle. Série verte*, 4(V3):67–79, 1970. (Cited on page 32.)

Marijn JH Heule, Oliver Kullmann, and Victor W Marek. Solving and verifying the Boolean Pythagorean triples problem via cube-and-conquer. In *International Conference on Theory and Applications of Satisfiability Testing*, pages 228–245. Springer, 2016. (Cited on pages 1 and 145.)

Timotej Hrga and Janez Povh. `MADAM`: a parallel exact solver for max-cut based on semidefinite programming and ADMM. *Comput. Optim. Appl.*, 80(2):347–375, 2021. ISSN 0926-6003,1573-2894. doi: 10.1007/s10589-021-00310-6. URL `https://doi.org/10.1007/s10589-021-00310-6`. (Cited on pages 113, 116, 118 and 127.)

Siyi Hu and Luca Carlone. Accelerated inference in Markov Random Fields via smooth Riemannian optimization. *IEEE Robotics and Automation Letters*, 4(2): 1295–1302, 2019. (Cited on page 45.)

Qixing Huang, Yuxin Chen, and Leonidas Guibas. Scalable semidefinite relaxation for maximum a posterior estimation. In *International Conference on Machine Learning*, pages 64–72. PMLR, 2014. (Cited on pages 43 and 105.)

Barry Hurley, Barry O'Sullivan, David Allouche, George Katsirelos, Thomas Schiex, Matthias Zytnicki, and Simon De Givry. Multi-language evaluation of exact solvers in graphical model discrete optimization. *Constraints*, 21(3):413–434, 2016. (Cited on pages 84, 137 and 157.)

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with AlphaFold. *Nature*, 596(7873):583–589, 2021. (Cited on page 83.)

Hariprasad Kannan, Nikos Komodakis, and Nikos Paragios. Tighter continuous relaxations for MAP inference in discrete MRFs: A survey. In *Handbook of Numerical Analysis*, volume 20, pages 351–400. Elsevier, 2019. (Cited on page 18.)

Jörg H Kappes, Bjoern Andres, Fred A Hamprecht, Christoph Schnörr, Sebastian Nowozin, Dhruv Batra, Sungwoong Kim, Bernhard X Kausler, Thorben Kröger, Jan Lellmann, et al. A comparative study of modern inference techniques for structured discrete energy minimization problems. *International Journal of Computer Vision*, 115(2):155–184, 2015. (Cited on page 75.)

Narendra Karmarkar. A new polynomial-time algorithm for linear programming. In *Proceedings of the sixteenth annual ACM symposium on Theory of computing*, pages 302–311, 1984. (Cited on pages 20 and 38.)

Richard M Karp. *Reducibility among combinatorial problems*. Springer, 2010. (Cited on pages 11, 112 and 154.)

Ross Kindermann and J Laurie Snell. *Markov random fields and their applications*, volume 1. American Mathematical Society, 1980. (Cited on pages 6 and 148.)

Andrew V Knyazev. Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method. *SIAM journal on scientific computing*, 23(2):517–541, 2001. (Cited on page 104.)

Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009. (Cited on pages xii, 6, 8, 15, 148 and 149.)

Vladimir Kolmogorov. Convergent tree-reweighted message passing for energy minimization. In *International Workshop on Artificial Intelligence and Statistics*, pages 182–189. PMLR, 2005. (Cited on page 22.)

A M C A. Koster. *Frequency assignment: Models and Algorithms*. PhD thesis, University of Maastricht, The Netherlands, November 1999. Available at www.zib.de/koster/thesis.html. (Cited on pages 1, 19 and 145.)

Nathan Krislock, Jérôme Malick, and Frédéric Roupin. Biqcrunch: A semidefinite branch-and-bound method for solving binary quadratic problems. *ACM Transactions on Mathematical Software (TOMS)*, 43(4):1–23, 2017. (Cited on pages 113, 116 and 127.)

Jean B Lasserre. Global optimization with polynomials and the problem of moments. *SIAM Journal on optimization*, 11(3):796–817, 2001. (Cited on page 113.)

Jean B Lasserre. An explicit equivalent positive semidefinite program for nonlinear 0-1 programs. *SIAM Journal on Optimization*, 12(3):756–769, 2002. (Cited on page 38.)

Jean B Lasserre. A MAX-CUT formulation of 0/1 programs. *Operations Research Letters*, 44(2):158–164, 2016. (Cited on pages 50, 51, 52, 79, 112, 116, 151 and 154.)

Monique Laurent. A comparison of the Sherali-Adams, Lovász-Schrijver, and Lasserre relaxations for 0–1 programming. *Mathematics of Operations Research*, 28(3):470–496, 2003. (Cited on page 38.)

Monique Laurent and Svatopluk Poljak. Gap inequalities for the cut polytope. *European Journal of Combinatorics*, 17(2):233–254, 1996. (Cited on page 115.)

Richard B Lehoucq and Danny C Sorensen. Deflation techniques for an implicitly restarted Arnoldi iteration. *SIAM Journal on Matrix Analysis and Applications*, 17(4):789–821, 1996. (Cited on page 108.)

Richard B Lehoucq, Danny C Sorensen, and Chao Yang. *ARPACK users' guide: solution of large-scale eigenvalue problems with implicitly restarted Arnoldi methods*. SIAM, 1998. (Cited on page 108.)

Xinxin Li, Ting Kei Pong, Hao Sun, and Henry Wolkowicz. A strictly contractive Peaceman-Rachford splitting method for the doubly nonnegative relaxation of the minimum cut problem. *Computational Optimization and Applications*, 78(3): 853–891, 2021. (Cited on page 100.)

Zhi-Quan Luo, Wing-Kin Ma, Anthony Man-Cho So, Yinyu Ye, and Shuzhong Zhang. Semidefinite relaxation of quadratic optimization problems. *IEEE Signal Processing Magazine*, 27(3):20–34, 2010. (Cited on page 33.)

Jérôme Malick and Frédéric Roupin. On the bridge between combinatorial optimization and nonlinear optimization: a family of semidefinite bounds for 0–1 quadratic problems leading to quasi-Newton methods. *Mathematical Programming*, 140(1): 99–124, 2013. (Cited on page 117.)

John E Mitchell. Restarting after branching in the SDP approach to MAX-CUT and similar combinatorial optimization problems. *Journal of Combinatorial Optimization*, 5:151–166, 2001. (Cited on page 123.)

John E Mitchell. Branch-and-cut algorithms for combinatorial optimization problems. *Handbook of applied optimization*, 1(1):65–77, 2002. (Cited on page 120.)

Yurii Nesterov and Arkadii Nemirovskii. *Interior-point polynomial algorithms in convex programming*. SIAM, 1994. (Cited on pages 30 and 38.)

Yurii Nesterov et al. *Lectures on convex optimization*, volume 137. Springer, 2018. (Cited on pages 39 and 41.)

Chirag Pabbaraju, Po-Wei Wang, and J Zico Kolter. Efficient semidefinite-programming-based inference for binary and multi-class MRFs. *Advances in Neural Information Processing Systems*, 33, 2020. (Cited on page 45.)

Sejun Park, Eunho Yang, Se-Young Yun, and Jinwoo Shin. Spectral approximate inference. In *International Conference on Machine Learning*, pages 5052–5061. PMLR, 2019. (Cited on page 75.)

Gábor Pataki. On the rank of extreme matrices in semidefinite programs and the multiplicity of optimal eigenvalues. *Mathematics of operations research*, 23(2): 339–358, 1998. (Cited on pages 44, 51, 52 and 152.)

Ryan E Pavlovicz, Hahnbeom Park, and Frank DiMaio. Efficient consideration of coordinated water molecules improves computational protein-protein and protein-ligand docking discrimination. *PLoS Computational Biology*, 16(9):e1008103, 2020. (Cited on page 84.)

Renfrey Burnard Potts. Some generalized order-disorder transformations. In *Mathematical proceedings of the Cambridge philosophical society*, volume 48, pages 106–109. Cambridge University Press, 1952. (Cited on page 45.)

Daniel Prusa and Tomas Werner. Universality of the local marginal polytope. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1738–1743, 2013. (Cited on page 21.)

Franz Rendl, Giovanni Rinaldi, and Angelika Wiegele. Solving max-cut to optimality by intersecting semidefinite and polyhedral relaxations. *Mathematical Programming*, 121:307–335, 2010. (Cited on pages 41, 113, 116, 118 and 127.)

Nikitas Rontsis, Paul Goulart, and Yuji Nakatsukasa. Efficient semidefinite programming with approximate ADMM. *Journal of Optimization Theory and Applications*, pages 1–29, 2022. (Cited on page 104.)

Ivo G Rosenberg. Reduction of bivalent maximization to the quadratic case. 1975. (Cited on page 37.)

Francesca Rossi, Peter Van Beek, and Toby Walsh. *Handbook of constraint programming*. Elsevier, 2006. (Cited on pages 6, 22 and 148.)

Walter Rudin. Functional analysis, mcgrawhill. *Inc, New York*, 45(46):4, 1991. (Cited on page 101.)

Thomas Schiex. Arc consistency for soft constraints. In *International Conference on Principles and Practice of Constraint Programming*, pages 411–425. Springer, 2000. (Cited on page 21.)

Michail I Schlesinger. Syntactic analysis of two-dimensional visual signals in noisy conditions. *Kibernetika*, 4(113-130):1, 1976. (Cited on pages 19 and 21.)

Hanif D Sherali and Warren P Adams. A hierarchy of relaxations between the continuous and convex hull representations for zero-one programming problems. *SIAM Journal on Discrete Mathematics*, 3(3):411–430, 1990. (Cited on page 20.)

Hanif D Sherali and Warren P Adams. *A reformulation-linearization technique for solving discrete and continuous nonconvex problems*, volume 31. Springer Science & Business Media, 2013. (Cited on page 33.)

Lennart Sinjorgo and Renata Sotirov. On solving the MAX-SAT using sum of squares. *arXiv preprint arXiv:2302.06931*, 2023. (Cited on page 38.)

Todd. A study of search directions in primal-dual interior-point methods for semidefinite programming. *Optimization methods and software*, 11(1-4):1–46, 1999. (Cited on page 40.)

Reha H Tütüncü, Kim-Chuan Toh, and Michael J Todd. Solving semidefinite-quadratic-linear programs using SDPT3. *Mathematical programming*, 95:189–217, 2003. (Cited on page 38.)

Leslie G Valiant. The complexity of computing the permanent. *Theoretical computer science*, 8(2):189–201, 1979. (Cited on pages 10 and 150.)

Lieven Vandenberghe and Stephen Boyd. Semidefinite programming. *SIAM review*, 38(1):49–95, 1996. (Cited on page 30.)

Martin J Wainwright, Michael I Jordan, et al. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305, 2008. (Cited on page 38.)

Irene Waldspurger and Alden Waters. Rank optimality for the Burer-Monteiro factorization. *SIAM journal on Optimization*, 30(3):2577–2602, 2020. (Cited on page 44.)

Toby Walsh. SAT v CSP. In *International Conference on Principles and Practice of Constraint Programming*, pages 441–456. Springer, 2000. (Cited on page 34.)

Huayan Wang and Daphne Koller. Subproblem-tree calibration: A unified approach to max-product message passing. In *International Conference on Machine Learning*, pages 190–198. PMLR, 2013. (Cited on page 22.)

Jie Wang, Victor Magron, Jean B Lasserre, and Ngoc Hoang Anh Mai. CS-TSSOS: Correlative and term sparsity for large-scale polynomial optimization. *ACM Transactions on Mathematical Software*, 48(4):1–26, 2022. (Cited on pages 38 and 113.)

Peng Wang, Chunhua Shen, and Anton Van Den Hengel. A fast semidefinite approach to solving binary quadratic problems. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1312–1319, 2013. (Cited on page 42.)

Peng Wang, Chunhua Shen, Anton van den Hengel, and Philip HS Torr. Efficient semidefinite branch-and-cut for MAP-MRF inference. *International Journal of Computer Vision*, 117:269–289, 2016. (Cited on pages 41, 42 and 105.)

Po-Wei Wang, Wei-Cheng Chang, and J Zico Kolter. The mixing method: low-rank coordinate descent for semidefinite programming with diagonal constraints. *arXiv preprint arXiv:1706.00476*, 2017. (Cited on pages 45, 50, 52, 56, 112, 113, 115, 120, 126, 151 and 153.)

Yu Wang, Wotao Yin, and Jinshan Zeng. Global Convergence of ADMM in Nonconvex Nonsmooth Optimization, 2018. (Cited on pages 136 and 157.)

Zaiwen Wen, Donald Goldfarb, and Wotao Yin. Alternating direction augmented lagrangian methods for semidefinite programming. *Mathematical Programming Computation*, 2(3-4):203–230, 2010. (Cited on pages 43 and 98.)

T. Werner. A Linear Programming Approach to Max-sum Problem: A Review. *IEEE Trans. on Pattern Recognition and Machine Intelligence*, 29(7):1165–1179, July 2007. URL `http://dx.doi.org/10.1109/TPAMI.2007.1036`. (Cited on page 19.)

Angelika Wiegele. Biq Mac Library—A collection of Max-Cut and quadratic 0-1 programming instances of medium size. *Preprint*, 51, 2007. (Cited on pages 112, 127 and 154.)

Angelika Wiegele and Shudian Zhao. SDP-based bounds for graph partition via extended ADMM. *Computational Optimization and Applications*, 82(1):251–291, 2022. (Cited on pages 109 and 153.)

Henry Wolkowicz and Qing Zhao. Semidefinite programming relaxations for the graph partitioning problem. *Discrete Applied Mathematics*, 96:461–479, 1999. (Cited on page 96.)

Henry Wolkowicz, Romesh Saigal, and Lieven Vandenberghe. *Handbook of semidefinite programming: theory, algorithms, and applications*, volume 27. Springer Science & Business Media, 2012. (Cited on pages xi and 27.)

Alp Yurtsever, Joel A Tropp, Olivier Fercoq, Madeleine Udell, and Volkan Cevher. Scalable semidefinite programming. *SIAM Journal on Mathematics of Data Science*, 3(1):171–200, 2021. (Cited on page 115.)

Qing Zhao, Stefan E Karisch, Franz Rendl, and Henry Wolkowicz. Semidefinite programming relaxations for the quadratic assignment problem. *Journal of Combinatorial Optimization*, 2(1):71–109, 1998. (Cited on pages 53 and 96.)

Shipu Zhao, Zachary Frangella, and Madeleine Udell. NysADMM: faster composite convex optimization via low-rank approximation. In *International Conference on Machine Learning*, pages 26824–26840. PMLR, 2022. (Cited on page 44.)

Ciyou Zhu, Richard H Byrd, Peihuang Lu, and Jorge Nocedal. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on mathematical software (TOMS)*, 23(4):550–560, 1997. (Cited on pages 42 and 117.)

RÉPUBLIQUE FRANÇAISE
Liberté
Égalité
Fraternité

Université de Toulouse

Titre : Optimisation convexe pour les modèles graphiques discrets

Mots clés : Programmation par contraintes, Optimisation combinatoire, Modèles graphiques, Optimisation convexe, Relaxation convexe

Résumé : Les modèles graphiques définissent une famille de formalismes et d'algorithmes utilisés en particulier pour le raisonnement logique et probabiliste, dans des domaines aussi variés que l'analyse d'image ou le traitement du langage naturel. Ils sont capables d'être appris à partir de données, donnant une information probabiliste qui peut ensuite être combinée avec des informations logiques. L'objectif de la thèse est d'améliorer l'efficacité des algorithmes de raisonnement sur ces modèles afin d'augmenter la puissance du mécanisme de raisonnement fondamental utilisé dans ces outils (le calcul de minorant) en exploitant les progrès réalisés ces dernières années dans le domaine de l'optimisation convexe. Ceci devrait permettre de résoudre des problèmes jusqu'ici hors de portée de nos outils les plus efficaces.

Title: Convex Optimization for Discrete Graphical Models

Key words: Constraint programming, Combinatorial optimization, Graphical models, Convex optimization, Convex relaxation

Abstract: Graphical models define a family of formalisms and algorithms used for logical and probabilistic reasoning, in fields as varied as image analysis or natural language processing. They can be learned from data, giving probabilistic information which can later be combined with logical information. The objective of the thesis is to improve the efficiency of the reasoning algorithms on these models in order to increase the power of the fundamental reasoning mechanism used in these tools (minorant computation) by exploiting the progress made in the field of convex optimisation. We should be able to solve problems that are currently beyond the reach of our most efficient tools.