Department of Computer Science
Series of Publications A
Report A-2018-5

Solving Optimization Problems via Maximum Satisfiability: Encodings and Re-Encodings

Jeremias Berg

To be presented with the permission of the Faculty of Science of the University of Helsinki, for public criticism in Auditorium CK112, Exactum, Gustaf Hällströmin katu 2b, on May 25th, 2018, at 12 o'clock noon.

University of Helsinki
Finland

Supervisors

Associate Professor Matti Järvisalo, University of Helsinki, Finland
Professor Petri Myllymäki, University of Helsinki, Finland

Pre-examiners

Professor Lakhdar Sais, Université d'Artois, France
Professor Peter Stuckey, University of Melbourne, Australia

Opponent

Associate Professor Inês Lynce, Universidade de Lisboa, Portugal

Custos

Associate Professor Matti Järvisalo, University of Helsinki, Finland

Contact information

Department of Computer Science
P.O. Box 68 (Gustaf Hällströmin katu 2b)
FI-00014 University of Helsinki
Finland

Email address: info@cs.helsinki.fi
URL: https://www.helsinki.fi/en/computer-science
Telephone: +358 2941 911

ISSN 1238-8645
ISBN 978-951-51-4241-2 (paperback)
ISBN 978-951-51-4242-9 (PDF)
Helsinki 2018
Unigrafia

Solving Optimization Problems via Maximum Satisfiability: Encodings and Re-Encodings

Jeremias Berg

Department of Computer Science
P.O. Box 68, FI-00014 University of Helsinki, Finland
jeremiasberg@gmail.com
http://www.jeremiasberg.com

PhD Thesis, Series of Publications A, Report A-2018-5
Helsinki, April 2018, 86 + 102 pages
ISSN 1238-8645
ISBN 978-951-51-4241-2 (paperback)
ISBN 978-951-51-4242-9 (PDF)

Abstract

NP-hard combinatorial optimization problems are commonly encountered in numerous different domains. As such efficient methods for solving instances of such problems can save time, money, and other resources in several different applications. This thesis investigates exact declarative approaches to combinatorial optimization within the maximum satisfiability (MaxSAT) paradigm, using propositional logic as the constraint language of choice. Specifically we contribute to both MaxSAT solving and encoding techniques.

In the first part of the thesis we contribute to MaxSAT solving technology by developing solver independent MaxSAT preprocessing techniques that re-encode MaxSAT instances into other instances. In order for preprocessing to be effective, the total time spent re-encoding the original instance and solving the new instance should be lower than the time required to directly solve the original instance. We show how the recently proposed label-based framework for MaxSAT preprocessing can be efficiently integrated with state-of-art MaxSAT solvers in a way that improves the empirical performance of those solvers. We also investigate the theoretical effect that label-based preprocessing has on the number of iterations needed by MaxSAT solvers in order to solve instances. We show that preprocessing does not improve best-case performance (in the number of iterations) of MaxSAT solvers, but can improve the worst-case performance. Going be-

yond previously proposed preprocessing rules we also propose and evaluate a MaxSAT-specific preprocessing technique called subsumed label elimination (SLE). We show that SLE is theoretically different from previously proposed MaxSAT preprocessing rules and that using SLE in conjunction with other preprocessing rules improves empirical performance of several MaxSAT solvers.

In the second part of the thesis we propose and evaluate new MaxSAT encodings to two important data analysis tasks: correlation clustering and bounded treewidth Bayesian network learning. For both problems we empirically evaluate the resulting MaxSAT-based solution approach with other exact algorithms for the problems. We show that, on many benchmarks, the MaxSAT-based approach is faster and more memory efficient than other exact approaches. For correlation clustering, we also show that the quality of solutions obtained using MaxSAT is often significantly higher than the quality of solutions obtained by approximative (inexact) algorithms. We end the thesis with a discussion highlighting possible further research directions.

Computing Reviews (2012) Categories and Subject Descriptors:

Mathematics of computing→Combinatorial optimization
Theory of computation→Constraint and logic programming
Theory of computation→Problems, reductions and completeness

General Terms:
Algorithms, Satisfiability, Combinatorial Optimization

Additional Key Words and Phrases:
constraint optimization, maximum satisfiability, MaxSAT, preprocessing

Acknowledgements

This work was done as part of the Constraint Reasoning and Optimization (CoReO) group of the Department of Computer Science at the University of Helsinki. First and foremost I would like to express my sincerest gratitude to my advisor, Associate Professor Matti Järvisalo for all of the guidance and advice I have received during my PhD studies. I especially want to thank you for believing in me enough to let me pursue my own research interests. I am also very grateful to Professor Petri Myllymäki for the numerous roles he has played in making this work possible. Many thanks go to all of my other co-authors as well: Paul Saikko, Brandon Malone, Tuukka Korhonen, Antti Hyttinen, Emilia Oikarinen, Kai Puolamäki, Kerstin Bunte and Samuel Kaski. Working with all of you has been most pleasant and educational.

The quality of this manuscript has been significantly improved by the valuable feedback I have received from various people. I would especially like to thank my pre-examiners Professor Lakhdar Sais and Professor Peter Stuckey for their valuable input as well as Associate Professor Inês Lynce for taking the time to come to Helsinki to be my opponent. I extend my gratitude to everyone else who has given me feedback and suggestions, Matti, Antti, Jonas and Brandon as well as all anonymous reviewers of each publication.

I am immensely grateful for the support I have received from the Doctoral School of Computer Science (DoCS). I thank the board and steering committee for giving me the financial stability needed complete my PhD. I especially want acknowledge Dr Pirjo Moen for helping me with most non-research related issues during my time as a PhD student. I also want to thank the Emil Aaltonen Foundation and the Nokia Foundation for financially supporting my PhD research.

I consider myself very lucky to have been a member of the CoReO research group. I'd like to thank past and present members of CoReO for the very inspiring research environment you have provided and the many interesting discussions we have had over the years. In addition to

my colleagues I would also like to thank my friends and other colleagues, both in and outside of Kumpula campus, for the welcome and necessary distractions from research. A special thanks goes to everyone that played cards with me in the coffee room or belayed me on the climbing wall. I am also immensely grateful for the opportunities given to me by my family and relatives. Without your support I would not be who I am today and probably would not have pursued a PhD in the first place. Finally I would like to thank Christina for being there for me through the ups and downs of day to day life.

Helsinki, April 2018

Jeremias Berg

Contents

Chapter 1

Introduction

Mathematical optimization is a rich field of study with numerous applications. Whenever we are given a problem and tasked with finding a solution that is "best", we are faced with an optimization problem. If the space of possible (feasible) solutions is discrete, we talk about a combinatorial optimization problem [1]. The exact definition of a solution being best (optimal) depends on the specific problem at hand. Commonly used quality measures include the length or cost of a solution. In this thesis, we focus on computationally challenging combinatorial optimization problems and, in particular, on developing maximum satisfiability [2] as a tool for solving them.

Computationally challenging optimization problems are common. Several of the well-known NP-complete decision problems correspond to NP-hard optimization problems. Consider, for example, the *traveling salesperson problem* (TSP) [3, 4]. An instance of TSP consists of a set of locations and the pairwise distances between them. A (feasible) solution to the instance is a route which visits all of the locations. The problem of deciding the existence of a route that has length at most some given bound is NP-complete. The corresponding NP-hard combinatorial optimization problem asks to find the shortest possible route.

NP-hard optimization problems are encountered in various settings, including, but not restricted to: telecommunications and network design [5], computational biology [6, 7], clustering [8–10], structure learning of probabilistic graphical models [11–13], argumentation [14], itemset mining [15–18], data visualization [19–21], planning [22–24], scheduling [25–30], routing [31], timetabling [32–36], hardware and software verification [37–39], covering [40], air traffic management [41, 42] and cancer therapy design [43].

The abundance and diversity of optimization problems suggests that efficient algorithms for can save time (e.g,. scheduling), money (e.g., net-

work design) or other resources in various applications. For example, an effective solution method to TSP could significantly decrease the delivery times and fuel costs of a delivery company.

The research field of combinatorial optimization is well-established and studied [1]. The solution approaches to combinatorial optimization problems can roughly be divided into four categories: approximation algorithms [44–48], local search algorithms [49–52], problem-specific exact algorithms [3, 53–56] and exact declarative methods [2, 57–62]. This thesis focuses on exact declarative methods for solving NP-hard combinatorial optimization problems.

Figure 1.1 overviews the declarative approach to solving an instance p of an NP-hard optimization problem $\mathcal{P}$. The first step of the declarative approach is the *encoding* of p into some mathematical constraint language $\mathcal{L}$. In other words, the declarative approach assumes the existence of a function (an encoding) $\mathcal{P} \to \mathcal{L}$ that maps each instance p of $\mathcal{P}$ to an instance $\mathcal{F}(p)$ of $\mathcal{L}$, i.e., a set of constraints in $\mathcal{L}$. The instance $\mathcal{F}(p)$ describes p in the sense that optimal solutions to $\mathcal{F}(p)$ correspond to optimal solutions to p. We assume that the constraint instance $\mathcal{F}(p)$ can be formed in polynomial time with respect to the size of p. This assumption is typical when working with declarative methods, although there has been some research into larger encodings, often for solving even more complex problems [63–65].

After encoding p into $\mathcal{F}(p)$, the next step in the declarative approach is *solving* $\mathcal{F}(p)$, i.e., computing an optimal assignment to the variables in $\mathcal{F}(p)$. We call such an assignment an *optimal solution* to $\mathcal{F}(p)$. Finally, the optimal solution to $\mathcal{F}(p)$ is used to reconstruct an optimal solution to p. Analogously to the encoding step, we assume that the reconstruction step is computable in polynomial time. Since $\mathcal{P}$ is NP-hard, these assumptions imply that $\mathcal{L}$ should be NP-hard as well. More specifically, we focus in this thesis on optimization problems and constraint languages with NP-complete decision counterparts. In the rest of the text, we use the term NP-hard in an informal manner to refer to specifically to such problems.

A notable characteristic of the pipeline in Figure 1.1 is that the (only) two computationally challenging steps are defining an encoding $\mathcal{P} \to \mathcal{L}$ and solving the constraint instance $\mathcal{F}(p)$. Assuming P $\neq$ NP, no complete solver for an NP-hard constraint language will run in polynomial time on every instance [66]. The efficiency of the declarative approach relies instead on designing solvers and encodings which ensure that the "interesting" instances of $\mathcal{P}$ are encoded into constraint instances $\mathcal{F}(p)$ on which the solver is able to avoid its worst case running time. By interesting instances we mean instances that are encountered in actual applications of the prob-

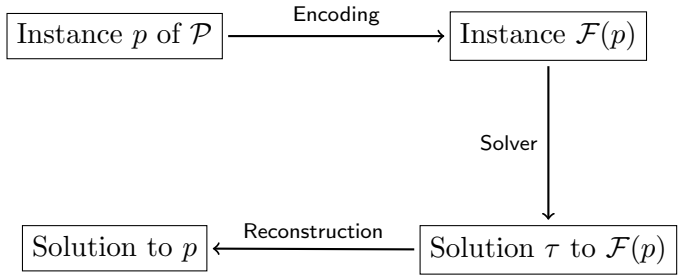

Figure 1.1: A declarative approach to solving an optimization problem $\mathcal{P}$.

lem. Consider for example a delivery company applying a solution method to TSP. A significant fraction of the theoretically possible instances (sets of locations) of TSP are never going to be encountered by the company in practice. Instead, an encoding and a solver which together are able to solve the instances corresponding to actually possible locations are enough to obtain a solution approach to TSP which is sufficient for the company's needs.

A significant benefit of the declarative approach to solving optimization problems is its generality. The computationally challenging step of solving $\mathcal{F}(p)$ is independent of the particular optimization problem $\mathcal{P}$ being solved. This means that improvements in solver technology of the chosen constraint language translate directly into more efficient algorithms to several different optimization problems, given the existence of well-performing encodings. Over the last decades, a number of different NP-hard constraint languages with varying features have been proposed and developed. A well known example is integer programming [57, 67, 68]. Others include constraint programming [59, 69], answer set programming [60, 61], maximum satisfiability [2] and its extensions to satisfiability modulo theories [70–72]. This thesis focuses on propositional logic as the underlying constraint language and maximum satisfiability as the corresponding constraint optimization problem.

1.1 Maximum Satisfiability

Maximum satisfiability (MaxSAT) is the optimization counterpart of the archetypical NP-complete propositional satisfiability (SAT) problem [66]. The expressive semantics of propositional logic, the constraint language underlying MaxSAT, allow encoding many NP-hard optimization problems as MaxSAT instances. At the same time, the relatively simple syntax also

allows the development of efficient solvers. The potential of propositional logic as the constraint language has been witnessed by the exceptional success of SAT solvers over the last decade [73, 74]. Recent improvements in MaxSAT solving technology and encodings have led to MaxSAT being applied in many different problem domains, including clustering [75], probabilistic modeling [76–79], data visualization [20], haplotype inference [80–82], game theory [83] treewidth computation [84], reasoning over biological networks [85, 86], electronic markets [87], routing [31], software verification and code debugging [37, 88–92], planning [24, 93, 94], cancer therapy design [43], computing covering arrays [95] scheduling [36], probabilistic reasoning [78], upgradeability [96], design debugging [97], analysis of other constraint satisfaction problems [98] and computer memory reconstruction [99].

The state of the art in MaxSAT solving techniques is evaluated annually in the MaxSAT Evaluations [100–102]. The evaluations have shown that the effectiveness of MaxSAT solvers for solving other optimization problems builds heavily on the effectiveness of SAT solvers. More specifically, many of the solvers that are most effective on MaxSAT instances that correspond to other optimization problems make extensive use of satisfiability solvers as subroutines. In the rest of the thesis such solvers are collectively called SAT-based MaxSAT solvers. SAT-based MaxSAT solvers can further be divided into roughly three subcategories: the model-guided [103–108], the core-guided [106, 109–120] and the implicit hitting set based [121–123] solvers. Most of the contributions of this thesis are developed in the context of core-guided and implicit hitting set based solvers, although many of the ideas are simple to extend to model-guided solvers as well.

It should be noted that in addition to SAT-based MaxSAT solvers, another commonly used approach to MaxSAT solving is branch and bound (B&B) [124–134]. B&B solvers tend to be most effective on random MaxSAT instances as well as challenging instances of smaller size. Such instances are encountered for example in combinatorics [100–102].

1.2 Contributions of the Thesis

This thesis is based on six peer-reviewed publications. The contributions of this thesis are divided into two interrelated research questions. In this section we first overview the publications and then discuss the research questions. We also briefly overview the specific contributions of the present author to each individual publication. The remaining chapters of the thesis will then discuss the contributions of each publication in more detail.

1.2.1 Original Publications

The following six peer-reviewed publications form the basis of this thesis. The papers are referred to as Papers I-VI in the rest of the text.

I Jeremias Berg, Paul Saikko, and Matti Järvisalo. **Improving the Effectiveness of SAT-Based Preprocessing for MaxSAT**. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 239-245. AAAI Press, 2015.

II Jeremias Berg, Paul Saikko, and Matti Järvisalo. **Re-using Auxiliary Variables for MaxSAT Preprocessing**. In *Proceedings of the IEEE 27th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 813-820. IEEE Computer Society, 2015.

III Jeremias Berg and Matti Järvisalo. **Impact of SAT-Based Preprocessing on Core-Guided MaxSAT Solving**. In *Proceedings of the 22nd International Conference on Principles and Practice of Constraint Programming (CP)*, volume 9892 of Lecture Notes in Computer Science, pages 66-85. Springer International Publishing, 2016.

IV Jeremias Berg, Paul Saikko, and Matti Järvisalo. **Subsumed Label Elimination for Maximum Satisfiability**. In *Proceedings of the 22nd European Conference on Artificial Intelligence (ECAI)*, volume 285 of Frontiers in Artificial Intelligence and Applications, pages 630-638. IOS Press, 2016.

V Jeremias Berg and Matti Järvisalo. **Cost-Optimal Constrained Correlation Clustering via Weighted Partial Maximum Satisfiability**. *Artificial Intelligence*. 244:110-142, 2017.

VI Jeremias Berg, Matti Järvisalo, and Brandon Malone. **Learning Optimal Bounded Treewidth Bayesian Networks via Maximum Satisfiability**. In *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 33 of JMLR Workshop and Conference Proceedings, pages 86-95. JMLR, 2014.

Reprints of the publications are included at the end of the thesis.

1.2.2 Research Questions

This thesis contributes to improving the effectiveness of using MaxSAT for solving combinatorial optimization problems by studying two distinct but connected research questions. The first question concerns the development

of MaxSAT solving methods, specifically in the form of solver-independent MaxSAT preprocessing [135]. The second question concerns the development of new MaxSAT encodings for two data analysis problems, correlation clustering [136] and bounded treewidth Bayesian network structure learning [137, 138].

Research Question 1: Preprocessing in MaxSAT solving

The first part of the thesis focuses on improving SAT-based MaxSAT solving. More specifically, Papers I-IV develop *preprocessing* techniques for MaxSAT. Preprocessing [139–141] extends the declarative pipeline (Figure 1.1) by adding a preprocessing step directly after the encoding step. During preprocessing, the constraint instance $\mathcal{F}(p)$ is re-encoded into another constraint instance $pre(\mathcal{F}(p))$ using polynomial-time computable inference rules. In this context, the inference rules are called *preprocessing rules* and the process of re-encoding $\mathcal{F}(p)$ is called *preprocessing* $\mathcal{F}(p)$. Analogously to the encoding, the preprocessing rules used should preserve optimal solutions. Informally, we say that preprocessing is *sound* if any optimal solution to $pre(\mathcal{F})$ can be used to reconstruct an optimal solution to $\mathcal{F}$ in polynomial time. The goal of (sound) preprocessing is to increase the applicability of MaxSAT for solving optimization problems by decreasing the overall time spent solving instances. In other words, effective preprocessing makes the total time spent preprocessing $\mathcal{F}(p)$ together with the time spent solving $pre(\mathcal{F}(p))$ lower than the time required to directly solve $\mathcal{F}(p)$. In this thesis, we focus on problem-independent preprocessing, i.e., preprocessing that does not depend on the particular optimization problem $\mathcal{P}$ being solved. In other words, we focus on preprocessing techniques that can be applied on any MaxSAT instance $\mathcal{F}$, regardless of the particular domain from which $\mathcal{F}$ was obtained.

In SAT solving, the importance of preprocessing is well-understood [140]. Many modern SAT solvers apply preprocessing before starting search [141–150]. The effectiveness of preprocessing in SAT solving suggests that similar effective preprocessing rules could be developed for MaxSAT solving as well. This possibility is especially interesting in the context of SAT-based MaxSAT solvers, since their effectiveness relies heavily on SAT solvers. Generalizing preprocessing rules proposed for SAT solving to MaxSAT is not straightforward. Direct application of many such rules to MaxSAT instances is not sound [135]. Informally, the reason is that, in order to be sound for SAT-solving, a preprocessing rule should preserve satisfiability, not the number of falsified clauses, and thus not optimal MaxSAT solutions either [151].

One approach to sound MaxSAT preprocessing is the so-called MaxSAT resolution rule [151]. Preprocessing rules based on MaxSAT resolution are indeed used by some B&B solvers [131, 152, 153]. However, such rules are difficult to use efficiently when solving MaxSAT instances that correspond to industrial applications. The reason is that each application of MaxSAT resolution adds several new clauses to the instance. Hence, MaxSAT resolution based preprocessing rules often increase the size of the already large industrial instances beyond what MaxSAT solvers can handle. In this thesis we focus on an alternative approach to MaxSAT preprocessing known as label-based preprocessing [135, 154]. Label-based preprocessing of MaxSAT instances allows generalizing several of the existing and well-established preprocessing rules proposed for SAT solving to MaxSAT by adding a single new variable (a label) to each soft clause of the instance before preprocessing.

In Papers I and II we develop label-based preprocessing further. In Paper I we show how label-based preprocessing can efficiently be incorporated with SAT-based MaxSAT solvers. The central insight of Paper I is that most SAT-based MaxSAT solvers add extra variables to the soft clauses of MaxSAT instances regardless of the use of preprocessing. In Paper I we show that the labels added during preprocessing can be reused in the solver, thus avoiding the need for the solver to add any new variables. We also show that reusing variables improves the empirical performance of LMHS, an at the time state-of-the-art SAT-based MaxSAT solver [123]. In Paper II we take the idea further and show that some variables in the input MaxSAT instance itself can be reused in the preprocessing and solving phases. This further reduces the number of new variables that need to be added when preprocessing and solving MaxSAT instances. We also show that identifying reusable variables from MaxSAT instances improves empirical performance of LMHS.

In Paper III we present a theoretical analysis on the effect of preprocessing on the number of SAT solver calls that SAT-based MaxSAT solvers require in order to terminate. An underlying motivation for the analysis is that SAT solver calls are the most computationally expensive step of such solvers. Thus insights into which factors influence the number of necessary calls can potentially significantly improve them. In Paper III we show that label-based generalizations of preprocessing rules for SAT solving can not reduce the minimum number of necessary SAT solver calls. We also show that preprocessing can ensure that the solver avoids worst-case executions, i.e., that preprocessing can decrease the maximum number of iterations required by SAT-based MaxSAT solvers.

Finally, in Paper IV we propose and analyze a new label-based MaxSAT preprocessing rule called subsumed label elimination (SLE). We analyze the theoretical differences between SLE and the generalizations of preprocessing rules for SAT solving. In particular, we show that including SLE amongst the preprocessing rules used during label-based preprocessing can result in more clauses and variables removed from the instance. We also report on an empirical evaluation on the effect of using SLE during label-based preprocessing. Our results show that SLE can improve the empirical performance of some state-of-the-art SAT-based MaxSAT solvers.

Research Question 2: Applications of MaxSAT in Data Analysis

The second part of this thesis focuses on developing MaxSAT encodings of other NP-hard combinatorial optimization problems. More specifically, Papers V and VI develop new MaxSAT encodings for two data analysis tasks: correlation clustering [136] (Paper V) and bounded treewidth Bayesian network structure learning [137, 155] (Paper VI).

Clustering is one of the central problems of unsupervised machine learning [156–159]. Given a set of data points, the goal of clustering is to partition the set in some meaningful way. The partitioning is typically called a clustering of the data and each set of a clustering is a cluster. This definition of clustering is very general, a number of different clustering problems and algorithms have been proposed over the years [160–164], including some constraint-based approaches [8–10, 165–167]. In Paper V, we focus on the correlation clustering problem [168–174]. Correlation clustering is a recently proposed clustering paradigm geared towards classifying data based on qualitative similarity information—as opposed to quantitative information—of pairs of data points. An instance of the correlation clustering problem consists of a set of data points and pairwise similarity information over them. The similarity information expresses preferences on whether or not the pair of points should be assigned to the same cluster. Informally, pairs of points that are similar should be assigned to the same cluster. At the same time, pairs of points that are dissimilar should be assigned to different clusters. An optimal solution to the instance balances these two conflicting objectives as well as possible. In contrast to other typical clustering paradigms, correlation clustering does not require the number of clusters as input. Instead, the optimal number should be learned during search. This makes correlation clustering especially well-suited for settings in which the true number of clusters is unknown. Consider for example the problem of clustering documents by topic without any prior knowledge on what those topics might be or how many of them there are [136, 175].

In Paper V we propose and prove the correctness of three MaxSAT encodings of correlation clustering. We also empirically compare the resulting MaxSAT-based solution approach with previously proposed exact and approximation (inexact) algorithms. Our results indicate that, within the scalability of exact approaches, the MaxSAT-based approach is often both faster and more memory-efficient than other exact approaches. We also show that the clusterings obtained using MaxSAT are of significantly better quality than the ones obtained by inexact algorithms, especially on sparse instances with missing similarity information.

Bayesian networks are an important class of probabilistic graphical models widely-used for representing joint probability distributions of sets of random variables [137, 176]. A Bayesian network structure is a directed acyclic graph (DAG) in which each node corresponds to a random variable. The graph represents the conditional dependencies between the variables. Often, a Bayesian network structure that represents given data well is not known *a priori*, and needs to be learned from observations (data) instead. Learning the optimal structure is a well-known optimization problem called the Bayesian network structure learning problem (BNSL) [177–180]. There are two main frameworks for BNSL: the score-based framework, and the independence test-based framework. In the score-based framework, each possible DAG structure is assigned a score that measures how well the structure explains the observations. The goal of BNSL is to compute a best-scoring network. For several commonly used scoring functions, the BNSL problem is NP-hard [181]. As is typical for challenging optimization problems, early solution methods to the problem tended to focus on polynomial-time inexact algorithms [182–187] while interest in exact algorithms for BNSL has increased within the last decade [54, 180, 188–191].

After having learnt a Bayesian network structure, the network is typically used for *probabilistic inference* tasks, such as inferring the probability distribution of some variables, possibly given the values of others. For general Bayesian network structures, this inference task is NP-hard [192]. However, it is becomes tractable whenever the underlying network structure has *bounded (fixed) treewidth* [193, 194]. Treewidth is a well-known graph-theoretic measure [195]. Informally, treewidth measures how "close" a given graph is to being a tree. All trees have treewidth 1 and all complete graphs with n nodes have treewidth $n - 1$. Treewidth has important connections to (in)tractability. Many NP-hard problems become tractable when restricted to instances that can be modeled using graphs with bounded treewidth [196, 197]. The fact that inference is tractable in Bayesian networks with low treewidth motivates the development of

algorithms that learn optimal Bayesian network structures with bounded treewidth, a problem known as bounded treewidth Bayesian network structure learning (BTBNSL). Compared to the recent progress in practical algorithms for optimally solving BNSL, fewer algorithms have been proposed for BTBNSL [138, 198–201]. In Paper VI we study BTBNSL in the score-based framework. It should be noted that the extra constraint bounding the treewidth of the solution network structure is a non-trivial addition to BNSL. BTBNSL is also an NP-hard optimization problem [199]. In fact, computing the treewidth of any graph is NP-hard [202].

In Paper VI we propose a MaxSAT encoding of BTBNSL. We compare the resulting MaxSAT-based solution approach to a previously proposed dynamic programming algorithm as the only other practical exact solution algorithm to BTBNSL available at the time of the publication of Paper VI [198]. We show that the MaxSAT-based method is more memory-efficient and scales noticeably better than the dynamic programming algorithm.

1.2.3 Specific Contributions by the Present Author

All publications were jointly co-written by all of their authors. Other contributions by the present author are as follows.

Paper I: The idea of reusing labels in a SAT-based MaxSAT solver was first proposed by the present author. The modifications required for label reusing in LMHS were done by the second author of the paper as the author of the LMHS solver. The present author modified a SAT preprocessor to be usable as a MaxSAT preprocessor and ran all of the experiments.

Paper II: The idea of identifying literals from the input formula that can be used as labels in preprocessing and assumptions in MaxSAT solving was a natural extension of Paper I. The present author implemented the modifications to the external preprocessor used in the publication and ran all of the experiments.

Paper III: The theoretical analysis was conducted by the present author under the guidance of the second author.

Paper IV: The idea of subsumed label elimination was developed by the present author with assistance from the other authors. The present author implemented the technique into the preprocessor and ran all of the experiments.

Paper V: The MaxSAT encoding of correlation clustering was jointly developed by the authors of the publication. The present author ran all of the experiments.

Paper VI: The MaxSAT encoding of bounded treewidth Bayesian network learning was co-developed by the authors of the publication. The present author ran all experiments presented in the paper.

1.3 Organization of the Thesis

The rest of the thesis is organized as follows. In Chapter 2 we give the background information relevant to this thesis. The contributions to the first and second research question are then overviewed in more detail in Chapters 3 and 4, respectively. We conclude the thesis with a summarizing discussion in Chapter 5.

Chapter 2

Preliminaries

In this chapter we give the relevant definitions and background information for understanding the main results of this thesis. First we give precise definitions of the satisfiability and maximum satisfiability problems in Sections 2.1 and 2.2, respectively. We then proceed by overviewing *cardinality constraints* in Section 2.3 as an important class of higher level constraints commonly used in both SAT-based MaxSAT solving and MaxSAT encodings of other optimization problems. We end the chapter by overviewing SAT-based MaxSAT solvers in Section 2.4. In our discussion we assume familiarity with propositional logic.

2.1 Propositional Satisfiability

We identify the truth value true with 1 and false with 0. A Boolean variable x has the domain $\{0, 1\}$. A literal l is a Boolean variable x or its negation $\neg x$. For a literal l, it holds that $\neg\neg l = l$. A clause C is a disjunction ($\vee$) of literals and a formula in conjunctive normal form (CNF) is a conjunction ($\wedge$) of clauses. We will mostly treat clauses as sets of literals and CNF formulas as sets of clauses. We will also simplify set notation when modifying formulas. Specifically, given a clause C and a CNF formula F, $F \setminus C$ is identified with $F \setminus \{C\}$ and $F \cup C$ with $F \cup \{C\}$. We denote the set of variables and literals of a clause C by $\textsc{Var}(C)$ and $\textsc{Lit}(C)$, respectively. The set of variables $\textsc{Var}(F)$ and literals $\textsc{Lit}(F)$ of a formula F are defined by $\textsc{Var}(F) = \bigcup_{C \in F} \textsc{Var}(C)$ and $\textsc{Lit}(F) = \bigcup_{C \in F} \textsc{Lit}(C)$, respectively. For a set L of literals, we use $\neg L$ to denote the set of negations of the literals in L, i.e., $\neg L = \{\neg l \mid l \in L\}$. L is a set of *assumptions* if either $x \notin L$ or $\neg x \notin L$ for each variable $x \in \textsc{Var}(F)$. Given a literal l, we denote by $\textsc{Cl}_F(l)$ the set of clauses of F which contain l, dropping the subscript

whenever clear from context. The clauses $\textsc{Cl}(L)$ containing literals from the set $L \subseteq \textsc{Lit}(F)$ are defined by $\textsc{Cl}(L) = \cup_{l \in L} \textsc{Cl}(l)$.

Given a set V of Boolean variables, a truth assignment τ over V is a function $\tau \colon V \to \{0, 1\}$. A truth assignment is extended to literals, clauses and CNF formulas in the standard way: $\neg x$ is true ($\tau(\neg x) = 1$) if x is false ($\tau(x) = 0$), a clause C is true ($\tau(C) = 1$) if $\tau(l) = 1$ for at least one literal $l \in C$, and a CNF formula F is true ($\tau(F) = 1$) if $\tau(C) = 1$ for all clauses $C \in F$. A truth assignment τ satisfies a clause C if $\tau(C) = 1$ and a formula if $\tau(F) = 1$, else it falsifies them. A CNF formula F is satisfiable if there exists a truth assignment τ which satisfies it, else F is unsatisfiable. Two formulas F_1 and F_2 are *equivalent* if $\tau(F_1) = \tau(F_2)$ for any truth assignment τ over $\textsc{Var}(F_1) \cup \textsc{Var}(F_2)$. The formulas are *equisatisfiable* if F_1 is satisfiable if and only if F_2 is. The well-known propositional satisfiability (SAT) problem asks if a given CNF formula F is satisfiable. As is common in most practical applications, we treat the SAT problem as the problem of computing a satisfying assignment to F or proving that one does not exist. Essentially all modern SAT solvers can provide a satisfying assignment whenever invoked on a satisfiable formula F.

A truth assignment $\tau \colon S \to \{0, 1\}$ over a proper subset $S \subset \textsc{Var}(F)$ is a partial assignment of the formula F. The simplification of F under a partial truth assignment τ is another formula F^τ obtained by removing all clauses satisfied by τ from the formula and all literals falsified by τ from the remaining clauses. When convenient, we will treat a (partial) truth assignment τ as a set of literals by $l \in \tau$ if and only if $\tau(l) = 1$. Similarly, each set $L \subseteq \textsc{Lit}(F)$ of assumptions can be treated as a (partial) truth assignment. In this thesis we use partial truth assignments in the context of satisfiability checking under assumptions [203]. Given a formula F and a set of assumptions $L \subseteq \textsc{Lit}(F)$, we say that F is satisfiable under L if F^L is satisfiable. For an alternative view, F is satisfiable under L if there exists a satisfying assignment τ to F that sets $\tau(l) = 1$ for all $l \in L$.

Given a CNF formula F, a *SAT solver* is an algorithm that computes a satisfying assignment to F or proves that one does not exist. The development of SAT solvers is an active area of research [73, 74, 204–210]. Besides pure satisfiability checking, SAT solvers are commonly used as subroutines in more complex algorithms, for example in SAT-based MaxSAT solvers [116, 203]. Most modern SAT solvers that are used in SAT-based MaxSAT solving implement the conflict-driven clause learning (CDCL) algorithm [204, 211, 212]. CDCL solvers have in turn evolved from the older Davis-Putnam-Logemann-Long search procedure [213]. In this thesis we only use CDCL SAT solvers as black boxes in SAT-based MaxSAT solvers

and as such will not discuss the details of how they operate here. The only requirement we make of a SAT solver is that it supports satisfiability querying under assumptions, and that it is able to compute subsets of the assumptions which explain unsatisfiability. More precisely, given a formula F and a set of assumptions $L \subseteq \text{LIT}(F)$ for which F^L is unsatisfiable, we assume that the SAT solver can extract a subset $L' \subseteq L$ such that $F^{L'}$ is unsatisfiable as well. Most modern CDCL SAT solvers support these features through the so-called *assumption interface*.

2.2 Maximum Satisfiability

An instance $\mathcal{F}$ of weighted partial maximum satisfiability (or MaxSAT for short) is a triplet $\mathcal{F} = (F_h, F_s, w)$ consisting of two CNF formulas, the *hard clauses* F_h and the *soft clauses* F_s, and a weight function $w \colon F_s \to \mathbb{N}$. The literals $\text{LIT}(\mathcal{F})$ and variables $\text{VAR}(\mathcal{F})$ of MaxSAT instances are the literals and variables of $F_h \wedge F_s$, respectively. Given a MaxSAT instance $\mathcal{F} = (F_h, F_s, w)$, any truth assignment τ which satisfies F_h is a solution to $\mathcal{F}$. The cost $\text{COST}(\mathcal{F}, \tau)$ of a solution τ to $\mathcal{F}$ is the sum of the weights of soft clauses it falsifies, i.e.,

$$\text{COST}(\mathcal{F}, \tau) = \sum_{C \in F_s} w(C) \cdot (1 - \tau(C)).$$

A solution τ^o to $\mathcal{F}$ is optimal if $\text{COST}(\mathcal{F}, \tau^o) \leq \text{COST}(\mathcal{F}, \tau)$ for all solutions τ to $\mathcal{F}$. The (optimal) cost of the instance $\mathcal{F}$ is the cost of the optimal solutions to $\mathcal{F}$. We denote the optimal cost of an instance $\mathcal{F}$ by $\text{COST}(\mathcal{F})$. In the rest of the thesis, we assume that all MaxSAT instances have solutions, or equivalently, that F_h is satisfiable.

The MaxSAT solvers we work with in this thesis make extensive use *unsatisfiable cores*. Given a MaxSAT instance $\mathcal{F} = (F_h, F_s, w)$, a subset $\kappa \subseteq F_s$ is an unsatisfiable core if the formula $F_h \wedge \kappa$ is unsatisfiable. A core κ is a minimal if $F_h \wedge \kappa_s$ is satisfiable for all $\kappa_s \subset \kappa$. Minimal cores are abbreviated by MUS (minimal unsatisfiable subformula). A set $M \subseteq F_s$ is a correction set (of $\mathcal{F}$) if the formula $F_h \wedge (F_s \setminus M)$ is satisfiable. The correction set M is minimal (an MCS) if $F_h \wedge (F_s \setminus M_s)$ is unsatisfiable for all $M_s \subset M$. We denote the set of MUSes and MCSes of $\mathcal{F}$ by $\text{MUS}(\mathcal{F})$ and $\text{MCS}(\mathcal{F})$, respectively.

The MCSes and MUSes of MaxSAT instances are related to each other via *hitting sets*. Given a collection of sets $\mathcal{K}$, a set H is a hitting set over $\mathcal{K}$ if $H \cap K \neq \emptyset$ for all $K \in \mathcal{K}$. A hitting set H is *irreducible* if no $H_s \subset H$ is a hitting set over $\mathcal{K}$. For a MaxSAT instance $\mathcal{F}$, the well known hitting set duality theorem establishes a connection between $\text{MUS}(\mathcal{F})$ and $\text{MCS}(\mathcal{F})$.

Theorem 1 (Hitting Set Duality [214])**.** *A set κ is an MUS of a MaxSAT instance $\mathcal{F}$ if and only if it is an irreducible hitting set over* $\mathrm{MCS}(\mathcal{F})$*. Similarly, a set M is an MCS of $\mathcal{F}$ if and only if it is an irreducible hitting set over* $\mathrm{MUS}(\mathcal{F})$*.*

Minimal correction sets provide an alternative definition of the MaxSAT problem. For a solution τ to a MaxSAT instance $\mathcal{F} = (F_h, F_s, w)$, let $\mathrm{U}(\tau) \subseteq F_s$ be the set of soft clauses falsified by τ. We say that τ is a *minimal solution* to $\mathcal{F}$ if $\mathrm{U}(\tau)$ is set-minimal, i.e., if there does not exist a solution τ^2 to $\mathcal{F}$ for which $\mathrm{U}(\tau^2) \subset \mathrm{U}(\tau)$. Notice that all optimal solutions to $\mathcal{F}$ are minimal but the converse does not hold. It is simple to show that there exists a many-to-one correspondence between minimal solutions and the MCSes of $\mathcal{F}$. More specifically, a solution τ of $\mathcal{F}$ is minimal if and only if $\mathrm{U}(\tau) \in \mathrm{MCS}(\mathcal{F})$. We say that a minimal solution τ to $\mathcal{F}$ corresponds to an MCS M^τ of $\mathcal{F}$ if $M^\tau = \mathrm{U}(\tau)$. The correspondence is not one-to-one, instead each $M \in \mathrm{MCS}(\mathcal{F})$ corresponds to a set of minimal solutions of $\mathcal{F}$. However, if two minimal solutions τ^1 and τ^2 to $\mathcal{F}$ correspond to the same $M \in \mathrm{MCS}(\mathcal{F})$, then τ^1 and τ^2 satisfy (and hence also falsify) the exact same clauses of $\mathcal{F}$. This implies that

$$\textsc{Cost}(\mathcal{F}, \tau^1) = \sum_{\substack{C \in F_s \\ \tau^1(C)=0}} w(C) = \sum_{C \in M} w(C) = \sum_{\substack{C \in F_s \\ \tau^2(C)=0}} w(C) = \textsc{Cost}(\mathcal{F}, \tau^2).$$

In this thesis we will treat minimal solutions that correspond to the same MCSes as equivalent. We say that an $M \in \mathrm{MCS}(\mathcal{F})$ corresponds to a solution τ^M if τ^M is a minimal solution of $\mathcal{F}$ that corresponds to M. A set $M \in \mathrm{MCS}(\mathcal{F})$ is optimal if it corresponds to an optimal solution of $\mathcal{F}$.

The relationship between MCSes and minimal solutions of MaxSAT instances suggests an alternative definition of the MaxSAT problem. Let $\mathcal{F} = (F_h, F_s, w)$ be a MaxSAT instance with the weight function w extended to sets $S \subseteq F_s$ of soft clauses by $w(S) = \sum_{C \in S} w(C)$. Denote the set of solutions and minimal solutions to $\mathcal{F}$ by $\textsc{sol}(\mathcal{F})$ and $\textsc{msol}(\mathcal{F})$, respectively. The optimal cost $\mathrm{COST}(\mathcal{F})$ of $\mathcal{F}$ can be expressed in terms of the MCSes of $\mathcal{F}$ by

$$\begin{aligned} \mathrm{COST}(\mathcal{F}) &= \min_{\tau \in \textsc{sol}(\mathcal{F})} \mathrm{COST}(\mathcal{F}, \tau) = \min_{\tau \in \textsc{msol}(\mathcal{F})} \mathrm{COST}(\mathcal{F}, \tau) \\ &= \min_{\tau \in \textsc{msol}(\mathcal{F})} w(M^\tau) = \min_{M \in \mathrm{MCS}(\mathcal{F})} w(M). \end{aligned}$$

In other words, an $M \in \mathrm{MCS}(\mathcal{F})$ is optimal if $w(M) \leq w(M')$ for all $M' \in \mathrm{MCS}(\mathcal{F})$. Thus the MaxSAT problem can be reformulated as the

problem of computing an $M^o \in \arg\min_{M \in \mathrm{MCS}(\mathcal{F})}\{w(M)\}$. By hitting set duality, such M^o is also a *minimum-cost hitting set* over $\mathrm{MUS}(\mathcal{F})$, i.e., a hitting set over $\mathrm{MUS}(\mathcal{F})$ which minimizes $w(M^o)$ over all hitting sets of $\mathrm{MUS}(\mathcal{F})$. Notice that a minimum-cost hitting set is guaranteed to be irreducible. The following theorem shows that a satisfiability query can be used in order to verify that a hitting set over any collection of cores of $\mathcal{F}$ is an optimal MCS without computing the entire $\mathrm{MUS}(\mathcal{F})$.

Theorem 2 (Adapted from [122])**.** *Let $\mathcal{F} = (F_h, F_s, w)$ be a MaxSAT instance and $\mathcal{C}$ a collection of cores of $\mathcal{F}$. Let M be a minimum cost hitting set over $\mathcal{C}$ and assume that $F_h \wedge (F_s \setminus M)$ is satisfiable. Then M is an optimal MCS of $\mathcal{F}$.*

The implicit hitting set solvers we work with in this thesis are based on Theorem 2.

2.3 Cardinality Constraints

Despite the simple syntax, several types of complex constraints can be modeled with CNF formulas. One such class of constraints commonly used in both SAT-based MaxSAT solving and MaxSAT encodings of other problems are *cardinality constraints*, an important special case of the more general class of *pseudo-boolean constraints*. Given a set $L = \{l_1, \ldots, l_n\}$ of n literals, a set $W = \{w_1, \ldots, w_n\}$ of weights, a constant k and $\circ \in \{\leq, \leq, \geq\geq, =\}$, a pseudo-boolean constraint is a linear constraint over L of form $\sum_{i=1}^{n} w_i l_i \circ k$. A truth assignment τ satisfies the constraint whenever $\sum_{i=1}^{n} w_i \tau(l_i) \circ k$ is true. We denote the set of clauses resulting from encoding a pseudo-boolean constraint $\sum_{i=1}^{n} w_i l_i \circ k$ to CNF by $\mathrm{CNF}(\sum_{i=1}^{n} w_i l_i \circ k)$. A pseudo-boolean constraint is a cardinality constraint if $w_i = 1$ for all $1 \leq i \leq n$. The numerous applications of cardinality constraints have motivated the development several different CNF encodings of them [215–220].

Example 1. *Let $L = \{l_1, \ldots, l_N\}$ be a set of literals and consider the* at-most-one *cardinality constraint*

$$\sum_{i=1}^{N} l_i \leq 1$$

enforcing that at most one of the literals in L must be set to true. The at-most-one constraint is commonly used in SAT-based MaxSAT solving [111, 115, 116, 120] as well as MaxSAT encodings of other problems, including correlation clustering and bounded treewidth Bayesian network structure

learning. A simple way of encoding this constraint in CNF is with $\mathcal{O}(n^2)$ *clauses of form* $(\neg l_i \vee \neg l_j)$ *for every distinct* l_i *and* l_j *in* L*. As an example of a more compact encoding, the ladder encoding uses* $n-1$ *auxiliary variables* $y_1, \ldots, y_{n-1}$ *and clauses corresponding to* $l_i \leftrightarrow (\neg y_i \wedge y_{i+1})$ *as well as* $y_i \rightarrow y_{i+1}$*. All in all the ladder encoding uses* $\mathcal{O}(n)$ *auxiliary variables and* $\mathcal{O}(n)$ *clauses.*

2.4 SAT-based MaxSAT Solvers

In this section we overview and discuss the two types of MaxSAT solvers on which the rest of the thesis focuses on. The contributions of this thesis to MaxSAT preprocessing are not specific to a single MaxSAT solver, but instead two classes of MaxSAT solvers that we call core-guided solvers [111, 115–117, 119, 120] and implicit hitting set based solvers [121–123]. We discuss these solvers in terms of two abstract MaxSAT solving algorithms: *CG*, representing core-guided solvers and *IHS*, representing implicit hitting set based solvers. In the rest of the thesis, we use the term MaxSAT algorithm to refer to abstractions and MaxSAT solver to refer to concrete implementations of MaxSAT algorithms.

The CG and IHS algorithms are presented in pseudocode in Figure 2.1 on the left and right side, respectively. These abstractions cover several modern MaxSAT solvers, including Fu-Malik (WPM1, WMSU1) [116, 221, 222], PMRES [115], OLL [111, 223] and ONE (K) [119] (the CG algorithm), as well as MaxHS [121, 122] and LMHS [123] (the IHS algorithm). It should be noted that solvers implementing CG or IHS often also make use of several different additional heuristics and search strategies [224–227] that are not included in the pseudocodes of Figure 2.1.

Both CG and IHS rely extensively on the ability to extract unsatisfiable cores from MaxSAT instances. Let (F_h, F_s) be two sets of clauses such that $F_h \wedge F_s$ is unsatisfiable. In both CG and IHS, a core $\kappa \subseteq F_s$ is extracted using the assumption interface of the underlying SAT solver. Let $F_s^A = \{C \vee a_C \mid C \in F_s\}$ be the set of all clauses in F_s, each extended with a unique *assumption variable* a_C. Let also $\mathcal{A}(F_s) = \textsc{Var}(F_s^A) \setminus \textsc{Var}(F_s)$ be the set of all assumption variables and consider a subset $\mathcal{A}_s \subseteq \mathcal{A}(F_s)$. Core extraction using assumptions is based on the fact that the simplification of $F_h \wedge F_s^A$ under $\neg\mathcal{A}_s$ is the formula $F_h \wedge \{C \mid C \vee a_C \in \textsc{Cl}(\mathcal{A}_s)\}$. In order to see this, consider a clause $C \vee a_C \in F_s^A$. If $a_C \in \mathcal{A}_s$, the partial assignment $\neg\mathcal{A}_s$ reduces $C \vee a_C$ to C. If $a_C \notin \mathcal{A}_s$, the clause $C \vee a_C$ can be trivially satisfied by setting a_C to true. Hence we can check the satisfiability of $F_h \wedge F_s$ by querying a SAT solver for the satisfiability of $F_h \wedge F_s^A$ under $\neg\mathcal{A}(F_s)$.

```
1  CG(F_h, F_s, w)
2  (F_h^w, F_s^w) ← (F_h, F_s)
3  while true do
4  |  (result, κ, τ) ← IsSAT(F_h^w, F_s^w)
5  |  if result="satisfiable" then
6  |  |  return τ
7  |  else
8  |  |  F_s^w = (F_s^w \ κ)
9  |  |  F_s^w ← F_s^w ∧ CLONE(κ)
10 |  |  (F_h^w, F_s^w) ← RELAX(F_h^w, F_s^w, κ)
```

```
1  IHS(F_h, F_s, w)
2  C ← ∅
3  while true do
4  |  H ← MinCostHittingSet(C)
5  |  (result, κ, τ) ← IsSAT(F_h, (F_s \ H))
6  |  if result="satisfiable" then
7  |  |  return τ
8  |  else
9  |  |  C ← C ∪ {κ}
```

Figure 2.1: Abstractions of the two types of MaxSAT algorithms we work with in this thesis.

If the formula is satisfiable, the returned truth assignment (restricted to $\textsc{Var}(F_h \wedge F_s)$) is also a satisfying assignment of $F_h \wedge F_s$. Otherwise, the subset $\neg\mathcal{A}_\kappa \subseteq \neg\mathcal{A}(F_s)$ of the assumptions returned by the solver can be mapped to an unsatisfiable core $\kappa = \{C \mid C \vee a_C \in \textsc{Cl}(\mathcal{A}_\kappa)\}$ of (F_h, F_s). In Figure 2.1 we abstract this functionality into the function IsSAT. The result of a query $\text{IsSAT}(F_h, F_s)$ is a triplet $(\mathit{result}, \kappa, \tau)$, where *result* is true if and only if $F_h \wedge F_s$ is satisfiable. If $F_h \wedge F_s$ is satisfiable, then τ is a satisfying assignment to it. Otherwise κ is an unsatisfiable core of $F_h \wedge F_s$. In the IHS algorithm the assumption interface is also used for removing clauses from the SAT-solver queries. More specifically, for a subset $H \subseteq F_s$, the satisfiability of $F_h \wedge (F_s \setminus H)$ is equivalent to the satisfiability of $F_h \wedge F_s^A$ under $\neg\mathcal{R}^H = \neg(\mathcal{A}(F_s) \setminus \mathcal{A}(H))$. This enables clause removal from the formula without the need to reset the internal state of the SAT solver. Notice that if $F_h \wedge F_s^A$ is unsatisfiable under $\neg\mathcal{R}^H$, the core returned by the SAT solver is guaranteed to be a subset of $F_s \setminus H$.

Given an input MaxSAT instance $\mathcal{F} = (F_h, F_s, w)$, the CG algorithm maintains a working formula (F_h^w, F_s^w), initialized to (F_h, F_s) on Line 2. The algorithm iteratively queries the internal SAT solver using the function $\text{IsSAT}(F_h^w, F_s^w)$ (Line 4), obtaining a triplet $(\mathit{result}, \kappa, \tau)$. Whenever the SAT solver returns "satisfiable", CG terminates and returns the assignment τ, guaranteed to be an optimal solution to $\mathcal{F}$ (Line 6). Otherwise, a core κ of (F_h^w, F_s^w) is obtained. The algorithm proceeds by *relaxing* the working instance and *compiling* information about the core into it (Line 10). Most of the implementations of RELAX that we are aware of assume that all of the soft clauses in the core have equal weight. To handle cores κ with varying clause weights, the solvers use a standard technique known as clause cloning [109, 221] (Line 9). First the smallest weight among all clauses in κ is computed, $w_{\min}^\kappa = \min\{w(C) \mid C \in \kappa\}$. Then each clause $C \in \kappa$ for which $w(C) > w_{\min}^\kappa$ is cloned; a duplicate clause $\textsc{Clone}(C)$ is added

to F_s^w, the weight of the original clause is set to $w_{\min}^\kappa$, and the duplicate $\textsc{Clone}(C)$ is given the residual weight $w(\textsc{Clone}(C)) = w(C) - w_{\min}^\kappa$. All duplicates are left in the working instance as soft clauses and the function $\text{RELAX}(F_h^w, F_s^w, \kappa)$ is invoked using the original clauses of κ which now all have equal weight. The exact manner in which the formula is modified, i.e., the implementation of RELAX, depends on the concrete MaxSAT solver. A classical example is the Fu-Malik solver [116] in which each clause $C \in \kappa$ is extended with a fresh relaxation variable r_C to form the extended clause $C \vee r_C$. The extended clauses are left in the formula as soft and a cardinality constraint $\text{CNF}(\sum r = 1)$ is added as hard clauses. Several of the early core-guided solvers relax the soft clauses in the core and add cardinality constraints as hard clauses. In contrast, more recently proposed core-guided solvers harden the soft clauses in the core and add cardinality constraints as soft clauses [111, 115, 119, 120].

In contrast to the CG algorithm, the IHS algorithm does not add or remove any clauses at all during execution and instead only works on the input hard and soft clauses. Given an input MaxSAT instance $\mathcal{F} = (F_h, F_s, w)$, the IHS algorithm maintains a set $\mathcal{C}$ of cores of $\mathcal{F}$, initialized to $\emptyset$ on Line 2. At each iteration, a minimum-cost hitting set over $\mathcal{C}$ is computed (Line 4). Then a SAT solver is invoked on all of the clauses in the working formula, except for the ones in H (Line 5). If the formula is satisfiable, H is an optimal MCS of $\mathcal{F}$ (Theorem 2) and IHS terminates, returning the optimal solution satisfying $F_h \wedge (F_s \setminus H)$ (Line 7). Otherwise, a new core is obtained and added to the set $\mathcal{C}$ (Line 9), after which the algorithm reiterates. In two solvers implementing the IHS algorithm, MaxHS [122] and LMHS [123], a minimum-cost hitting set is obtained by solving the current hitting set problem using an integer programming solver.

Beyond the scope of this thesis, a third class of SAT-based MaxSAT solvers are the so-called model-guided solvers [103–108, 110]. When invoked on a MaxSAT instance $\mathcal{F} = (F_h, F_s, w)$, a model-guided solver initializes an upper and lower bound UB and LB of the optimal cost of $\textsc{Cost}(\mathcal{F})$. The exact manner in which the bounds are initialized depends on the solver, a simple example sets $\text{LB} = 0$ and $\text{UB} = \sum_{C \in F_s} w(C)$. During search, the solver queries a SAT solver for the satisfiability of $F_h \wedge F_s^A \wedge \text{CNF}(\sum_{C \in F_s} (w(C) \cdot a_C) \leq k)$ where k is some constant satisfying $LB \leq k \leq UB$. If the formula is satisfiable, then $\textsc{Cost}(\mathcal{F}) \leq k$ and the value of the upper bound is lowered. Similarly, if the formula is unsatisfiable, then $\textsc{Cost}(\mathcal{F}) > k$ and the value of the lower bound is increased. The solver iterates until an optimal solution is found. Different model-guided solvers make use of several different search strategies and

encodings of cardinality constraints. Some also use unsatisfiable cores for more precise control on which soft clauses are relaxed and how much the bounds are updated [106, 107].

Finally we mention that in addition to SAT-based MaxSAT solvers a central approach to exact MaxSAT solving is branch and bound (B&B) [124–133]. Given an input MaxSAT instance $\mathcal{F}$, B&B solvers search for an optimal solution to $\mathcal{F}$ by branching on the two possible values of each variable in the formula. In order to avoid exhaustive search over all possible assignments of the variables, B&B solvers make use of several different bound computation and other inference rules [131, 134, 152, 228, 229] designed to allow effective pruning of the search tree. Some B&B solvers also make use of restricted forms of unsatisfiable cores in their bound computations [125].

Chapter 3

Preprocessing for Maximum Satisfiability Solving

In this chapter we discuss the contributions of this thesis to MaxSAT preprocessing techniques, overviewing Papers I-IV. While the importance of preprocessing in SAT solving is well-established [140–144, 146–150], the role of preprocessing in MaxSAT solving is not as developed [135, 151]. Here we focus on the label-based approach to MaxSAT preprocessing [135] and the CG and IHS MaxSAT algorithms presented in Figure 2.1 of Section 2.4. The empirical results presented in this chapter focus on the LMHS MaxSAT solver [123], a from-scratch instantiation of the IHS algorithm by the second author of Papers I, II and IV. All experiments were performed on a cluster of 2.53-GHz Intel Xeon quad core machines with 32 GB memory and Ubuntu Linux, using a per-instance memory limit of 30 GB. Since the time limit used in the experiments varied between papers, we will specify them in the relevant sections. For the formal proofs and complete empirical results, we direct the reader to the reprints of the papers at the end of the thesis.

This chapter is organized as follows. In Section 3.1 we give preliminaries on label-based preprocessing of MaxSAT instances. In Section 3.2 we discuss how label-based preprocessing can be integrated into SAT-based MaxSAT solving in a manner that significantly decreases the number extra variables and clauses that are added (Papers I and II). We demonstrate that tighter integration between the preprocessing and solving steps results in improved empirical performance of LMHS. In Section 3.3 we overview a theoretical analysis of the effect of preprocessing on the number of iterations required by CG and IHS (Paper III). Finally, in Section 3.4 we present a MaxSAT-specific preprocessing technique that we call subsumed label elimination (SLE) (Paper IV). We give theoretical results on the differ-

ences between SLE and the MaxSAT generalizations of preprocessing rules for SAT solving. We also show that using SLE in conjunction with previously proposed preprocessing rules leads to further simplifications during preprocessing as well as improved empirical performance of LMHS.

3.1 Label-based MaxSAT Preprocessing

Most of the contributions of this thesis to MaxSAT preprocessing build on previous work [135] on lifting four central preprocessing rules proposed for SAT to MaxSAT using the so called labeled CNF (LCNF) framework [154, 230]. More specifically, the rules lifted are bounded variable elimination, subsumption and self-subsuming resolution [141], as well as blocked clause elimination [231]. In this chapter we focus on the same four rules and call them *SAT-based preprocessing rules*. It should, however, be noted that, in addition to these four, several other preprocessing rules have been proposed for SAT solving [140, 142, 143, 145–150].

For some intuition on why SAT-based preprocessing rules can not directly be applied on MaxSAT instances, consider the subsumption elimination (SE) rule. Let F be a SAT formula and C, D two clauses of F. We say that C subsumes D if $C \subseteq D$. A clause D is *subsumed* if some other clause subsumes it. The SE rule allows removing subsumed clauses from F. Let $pre(F)$ be the formula resulting after an application of SE on F. Then F and $pre(F)$ are equisatisfiable since any assignment τ that satisfies the former will satisfy the latter and vice versa. More generally, we say that a preprocessing rule is *sound* for SAT-solving if (i) applying the rule to a formula F gives an equisatisfiable formula $pre(F)$ and (ii) a satisfying assignment to F can be reconstructed from any satisfying assignment to $pre(F)$ in polynomial time. Even if SE is sound for SAT solving, the next example demonstrates that directly removing subsumed clauses from the hard and soft clauses of MaxSAT instances can alter the costs of solutions and thus also the optimal solutions.

Example 2. *Let $\mathcal{F} = (F_h, F_s, w)$ be a MaxSAT instance with*

$$F_h = \{(\neg x_1), (\neg x_2), (\neg x_3 \vee \neg x_4)\},$$
$$F_s = \{(x_1 \vee x_3), (x_2 \vee x_3), (x_3), (x_4)\}$$

and $w(C) = 1$ for each $C \in F_s$. An optimal solution τ to $\mathcal{F}$ sets $\tau(x_1) = \tau(x_2) = \tau(x_4) = 0$ and $\tau(x_3) = 1$, falsifying one soft clause. Direct application of SE on $F_h \wedge F_s$ removes two soft clauses. The resulting instance

$\mathcal{F}^2 = (F_h^2, F_s^2, w^2)$ *has*

$$F_h^2 = \{(\neg x_1), (\neg x_2), (\neg x_3 \vee \neg x_4)\} \text{ and } F_s^2 = \{(x_3), (x_4)\}.$$

One optimal solution τ^2 *to* $\mathcal{F}^2$ *sets* $\tau^2(x_1) = \tau^2(x_2) = \tau^2(x_3) = 0$ *and* $\tau^2(x_4) = 1$. *This solution falsifies one soft clause in* $\mathcal{F}^2$ *but three in* $\mathcal{F}$.

Example 2 illustrates the fact that instead of only preserving satisfying assignments, MaxSAT preprocessing should preserve the *optimal solutions* of instances.

Definition 1. *Let* $\mathcal{F}$ *be a MaxSAT instance,* $\mathcal{R}$ *a preprocessing rule, and* $pre(\mathcal{F})$ *the instance obtained by preprocessing* $\mathcal{F}$ *with* $\mathcal{R}$. *Assume* τ^p *is an optimal solution to* $pre(\mathcal{F})$. *The preprocessing rule* $\mathcal{R}$ *is sound for MaxSAT if an optimal solution* τ *to* $\mathcal{F}$ *can be reconstructed from* τ^p *in polynomial time.*

Procedure 3.1 describes label-based preprocessing of a MaxSAT instance $\mathcal{F} = (F_h, F_s, w)$ using SAT-based preprocessing rules. First, each soft clause $C \in F_s$ is extended with a unique new *label* (Boolean variable) l_C to form the *labeled clause* $C \vee l_C$ and the set of labeled soft clauses $F_s^L = \{C \vee l_C \mid C \in F_s\}$. Notice the similarity between labels and assumption variables used in SAT-based MaxSAT solving (recall Section 2.4). Let $\mathcal{L}(\mathcal{F}) = \textsc{Var}(F_s^L) \setminus \textsc{Var}(F_s)$ be the set of all added labels. The next step of label-based preprocessing is preprocessing the formula $F_h \wedge F_s^L$ with SAT-based preprocessing rules, thereby obtaining the formula $pre(F_h \wedge F_s^L)$. In order to guarantee soundness for MaxSAT, bounded variable elimination and self-subsuming resolution are restricted from removing any variables in $\mathcal{L}(\mathcal{F})$ during preprocessing [135]. Finally, the preprocessed MaxSAT instance $pre(\mathcal{F}) = (F_h^p, F_s^p, w^p)$ has $F_h^p = pre(F_h \wedge F_s^L)$ as hard clauses. The soft clauses F_s^p contain unit clauses with negations of labels that appear among the hard clauses: $F_s^p = \{(\neg l_C) \mid l_C \in \textsc{Lit}(F_h^p) \cap \mathcal{L}(\mathcal{F})\}$. The weight of each soft clause $w^L((\neg l_C))$ is equal to the weight $w(C)$ of the soft clause to which l_C was added in the first step. We emphasize that soft clauses are only added for labels $l_C \in \textsc{Lit}(F_h^p)$. Even if bounded variable elimination or self subsuming resolution can not remove any labels, a label can still be removed from the instance during preprocessing. For example, if a soft clause is subsumed by a hard clause, SE can remove the labeled soft clause during preprocessing together with the corresponding label.

The basis for the soundness of label-based preprocessing with SAT-based preprocessing rules is the following theorem.

Preprocess $\mathcal{F} = (F_h, F_s, w)$

1. Let $F_s^L = \{C \vee l_C \mid C \in F_s, l_C \text{ new}\}$ and $\mathcal{L}(\mathcal{F}) = \textsc{Var}(F_s^L) \setminus \textsc{Var}(F_s)$.
2. Preprocess the CNF formula $F_h \wedge F_s^L$ using SAT-based preprocessing rules.
   - Do not remove any $l \in \mathcal{L}(\mathcal{F})$ with bounded variable elimination nor self-subsuming resolution.
3. Return
$$pre(\mathcal{F}) = (F_h^p, F_s^p, w^p)$$
with $F_h^p = pre(F_h \wedge F_s^L)$, $F_s^p = \{(\neg l_C) \mid l_C \in \textsc{Lit}(F_h^p) \cap \mathcal{L}(\mathcal{F})\}$ and $w^p((\neg l_C)) = w(C)$.

Procedure 3.1: Label-based preprocessing of a MaxSAT instance $\mathcal{F}$.

Theorem 3. *(Adapted from [135]) Assume that an instance $\mathcal{F} = (F_h, F_s, w)$ is preprocessed using label-based preprocessing with SAT-based preprocessing rules to obtain $pre(\mathcal{F}) = (F_h^p, F_s^p, w^p)$. For each soft clause $C \in F_s$, let l_C be the label added to C during preprocessing. Then the following hold.*

(i) The optimal costs of $\mathcal{F}$ and $pre(\mathcal{F})$ are equal.

(ii) $M \in \text{MUS}(\mathcal{F})$ if and only if $\{(\neg l_C) \mid C \in M\} \in \text{MUS}(pre(\mathcal{F}))$.

We show that label-based preprocessing with SAT-based preprocessing rules is sound for MaxSAT using Theorem 3 and hitting set duality (Theorem 1).

Theorem 4. *Label-based preprocessing with SAT-based preprocessing rules is sound for MaxSAT.*

Proof. (Sketch) Let $\mathcal{F} = (F_h, F_s, w)$ be a MaxSAT instance and $pre(\mathcal{F}) = (F_h^p, F_s^p, w^p)$ an instance obtained by label-based preprocessing of $\mathcal{F}$ using SAT-based preprocessing rules. Consider an optimal solution τ^p to $pre(\mathcal{F})$ and let M^{τ^p} be the MCS corresponding to τ^p. By Theorem 3 and hitting set duality, the set $M = \{C \mid (\neg l_C) \in M^{\tau^p}\}$ is an MCS of $\mathcal{F}$. Since $w^p((\neg l_C)) = w(C)$ for all $(\neg l_C) \in M^{\tau^p}$, it follows that $w^p(M^{\tau^p}) = w(M)$. Since M^{τ^p} is optimal for $pre(\mathcal{F})$ and $\text{COST}(pre(\mathcal{F})) = \text{COST}(\mathcal{F})$, the

MCS M is optimal for $\mathcal{F}$. Hence the solution τ^M corresponding to M is an optimal solution to $\mathcal{F}$. The fact that τ^M can be reconstructed from τ^p in polynomial time follows from the fact that SAT-based preprocessing rules allow reconstruction of satisfying assignments to CNF formulas [135, 140]. □

3.2 Integrating Label-based Preprocessing into SAT-based Solving

As the first contribution to MaxSAT preprocessing of this thesis we investigate label-based preprocessing in conjunction with SAT-based MaxSAT solving. In Paper I we show how to improve the empirical performance of LMHS [123], a MaxSAT solver implementing the IHS algorithm, by reusing labels as assumptions.

More generally, we show that if a preprocessed MaxSAT instance $pre(\mathcal{F})$ is solved with a SAT-based MaxSAT solver, the labels introduced during preprocessing can be reused as the assumption variables used for core extraction within the internal SAT solver. Since a similar number of assumption variables would otherwise be introduced by the solver, reusing labels as assumptions removes the need to add extra variables when using label-based preprocessing with SAT-based MaxSAT solving. In more detail, assume that the IHS algorithm instructed to reuse labels as assumptions is invoked on a preprocessed MaxSAT instance $pre(\mathcal{F}) = (F_h^p, F_s^p, w^p)$. Then the internal SAT solver of IHS is first initialized with the clauses in F_h^p (and specifically not F_s^p). During search, the cores in $\mathcal{C}$ are maintained in terms of label variables. Each computed hitting set H is the set of label variables that should not be assumed to be false in the next SAT solver call. Each unsatisfiable SAT solver call obtains a new subset of the label variables and the augmented IHS algorithm terminates as soon as a SAT solver call returns "satisfiable". While Paper I focuses on the IHS algorithm, a similar idea is applicable to the CG algorithm as well. Notice that labels l can be treated as soft clauses by introducing a unit clause $(\neg l)$ on demand.

Informally, the correctness of reusing labels as assumptions follows by considering an execution of IHS not reusing labels as assumptions invoked on a preprocessed MaxSAT instance $pre(\mathcal{F}) = (F_h, F_s, w)$. Initially each soft clause $(\neg l_C) \in F_s$ is extended with an assumption variable a to form the clause $(\neg l_C \vee a)$. Notice that this clause is equivalent to the implication $\neg a \rightarrow \neg l_C$. Let $\mathcal{A}(pre(\mathcal{F}))$ be the set of all assumption variables. During an iteration of the while-loop (Lines 3-9), IHS will first compute H as a minimum-cost hitting set over the set of cores $\mathcal{C}$ identified so far. In

practice, H is a subset of $\mathcal{A}(pre(\mathcal{F}))$ containing the assumption variables a of all clauses $\neg l_C \vee a$ which will be removed from the instance in the next SAT solver call. Afterwards a SAT solver is invoked on $F_h \wedge F_s^A$ under $\neg(\mathcal{A}(pre(\mathcal{F})) \setminus H)$. In this call, each soft clause $(\neg l_C \vee a)$ for which $a \notin H$ is reduced to $(\neg l_C)$ due to assuming a to be false. Thus the value of l_C is propagated to false as well. Similarly, if $a \in H$, the value of a is not assumed to be anything at all. However, as a only appears in a single clause, the SAT solver can assign it to true in order to satisfy the clause $(\neg l_C \vee a)$. As $\neg l_C$ does not appear in any other clause, the SAT solver can also assign l_C to true, satisfying all clauses in $\textsc{Cl}_{F_h}(l_C)$. Thus the assumptions only affect the values of the corresponding label variables through the implications $\neg a \rightarrow \neg l_C$. An alternative description of reusing labels as assumptions is hence to not introduce the implications $\neg a \rightarrow \neg l_C$ at all, but instead directly assume the values of the l_C variables. In Paper I we give a more direct proof of soundness using the formal LCNF framework [135].

In addition to the theoretical analysis, Paper I also reports on an experimental evaluation of the effect that reusing labels as assumptions has on LMHS. The evaluation was performed using the weighted partial industrial and crafted benchmarks of the 2014 MaxSAT evaluation [101] using a per-instance time limit of 1 h. Figure 3.2 shows a summary of the results. The line MaxHS-2.5 corresponds to the newest version of the MaxHS solver at the time [121, 122]. MaxHS is included to give a baseline comparison to LMHS.

The line LMHS+pre of Figure 3.2 of shows the performance of LMHS using preprocessing *without reusing* labels as assumptions. We note that preprocessing without reusing labels actually degrades overall performance of the solver. The best overall performance is achieved by LMHS+R-pre, corresponding to LMHS using preprocessing and reusing labels as assumptions. In the rest of this chapter, we will refer to LMHS+R-pre simply as LMHS, explicitly mentioning whenever it is used *without* reusing labels as assumptions.

Reusing Literals from MaxSAT Instances

In Paper II we show how the number of variables that need to be introduced to a MaxSAT instance $\mathcal{F}$ during preprocessing and SAT-based solving can be decreased further. We prove that literals $l \in \textsc{Lit}(\mathcal{F})$ that satisfy three easily identifiable criteria can be reused as labels in preprocessing and assumptions in SAT-based MaxSAT solving. We also propose *group detection* as a simple pattern-matching procedure to identify such literals. We experimentally demonstrate that reusable literals can be iden-

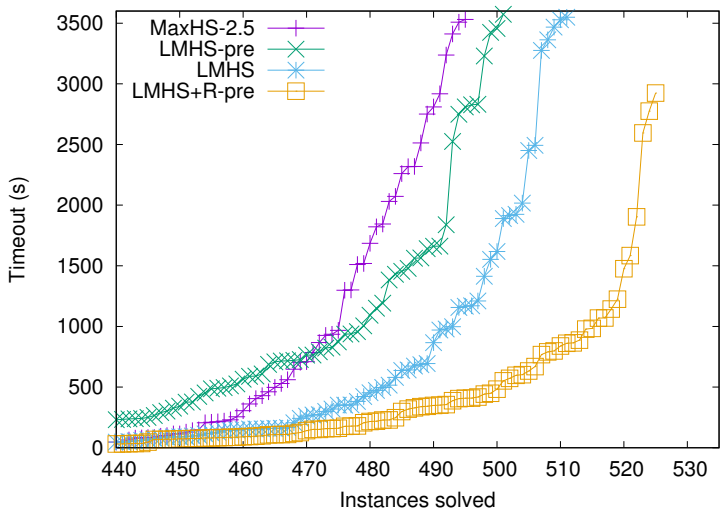

Figure 3.2: The effect of reusing labels as assumptions on LMHS (from Paper I).

tified in a significant fraction of the MaxSAT evaluation benchmarks and that using group detection leads to modest improvements in the empirical performance of LMHS.

In more detail, we introduce the concept of a *group-detectable* literal.

Definition 2. *Let* $\mathcal{F} = (F_h, F_s, w)$ *be a MaxSAT instance and* $l \in \textsc{Lit}(\mathcal{F})$. *The literal* l *is* group-detectable *if it satisfies the following three criteria.*

1. $(\neg l) \in F_s$.
2. $\neg l \notin \textsc{Lit}(F_h \wedge (F_s \setminus (\neg l)))$.
3. $l \notin \textsc{Lit}(F_s)$.

In words, a literal l is group-detectable in an instance $\mathcal{F} = (F_h, F_s, w)$ if l is not a member of any soft clauses of $\mathcal{F}$ and its negation $\neg l$ does not appear in any clause in $F_h \wedge F_s$ except for one unit soft clause. We say that a soft clause $(\neg l)$ is group-detectable if the literal l is.

In Paper II we show that given *any* MaxSAT instance $\mathcal{F}$ (preprocessed or not), all group-detectable literals $l \in \textsc{Lit}(\mathcal{F})$ can be reused as labels for preprocessing and assumptions for core extraction. For some intuition on the connection between Paper I and II, notice that if an instance $\mathcal{F}$ is preprocessed to obtain a preprocessed instance $pre(\mathcal{F})$, then every soft

clause in $pre(\mathcal{F})$ is group-detectable. Hence group detection could be seen as a generalization of reusing labels as assumptions after preprocessing.

In Paper II we propose group detection as a pattern matching procedure for identifying group-detectable literals and reusing them as labels during preprocessing and assumptions during solving. Intuitively, the correctness of group detection should be clear. In the paper we give a formal proof of correctness using the LCNF framework [135]. The name group detection stems from a setting in which we are given a group G of clauses and wish to encode the soft group constraint $\bigwedge_{C \in G} C$ of weight w_g in CNF. One possible encoding is to: (i) introduce a single new *group variable* g, (ii) extend each $C \in G$ with the same g variable to form the clause $C \vee g$, (iii) treat all extended clauses $C \vee g$ as hard, and (iv) introduce the soft clause $(\neg g)$ with weight c_w. Notice that using this encoding the literal g is group-detectable. An observation similar to Paper II was made in [232] where the authors study *group MaxSAT* as an alternative approach to handling soft group constraints.

In Paper II we report on the results of an empirical evaluation of group detection. Figure 3.3 shows the fraction of soft clauses group-detectable in the weighted partial industrial and crafted benchmarks of the 2014 MaxSAT evaluation. As the figure illustrates, all soft clauses are group-detectable in over 40% of the crafted and over 30% of the industrial instances, suggesting that a significant fraction of the soft clauses in the considered MaxSAT benchmarks correspond to encodings of group constraints. The effect of group detection on the total solving time of LMHS is shown in Figure 3.4 on the same benchmark set. All runs were performed using a per-instance time limit of 1 h. The base algorithm (the line LMHS in the plot), using neither preprocessing nor group detection, exhibits the worst performance. Interestingly, the variant using only group detection and no preprocessing (LMHS-G) is competitive with the variant using preprocessing without group detection (LMHS-pre). This observation highlights the importance of minimizing the number of extra variables and clauses during label-based preprocessing and SAT-based MaxSAT solving. Best performance is achieved by using both preprocessing and group detection (LMHS-G-pre), even if the improvement over LMHS-G and LMHS-pre is modest. In Paper II we offer one possible explanation for the modesty of the improvement to be the fairly strong connection between group detection and bounded variable elimination. Apart from the LMHS solver, the paper also includes empirical results for the Eva solver [115] as an example of a solver that implements the CG algorithm.

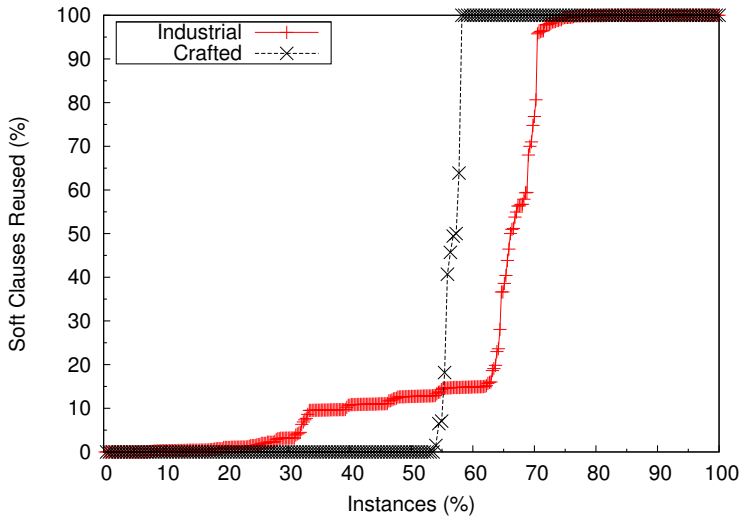

Figure 3.3: The fraction of soft clauses reusable as labels or assumptions (from Paper II).

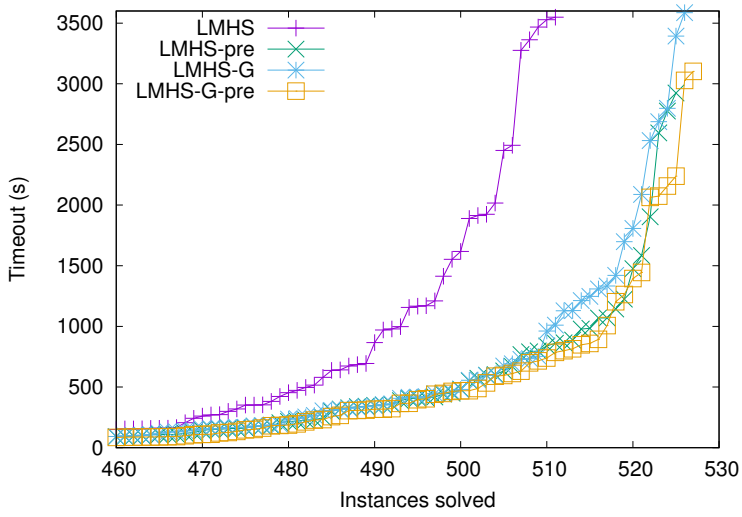

Figure 3.4: The effect of group detection on the LMHS MaxSAT solver (from Paper II).

3.3 Effect of Preprocessing on SAT-based MaxSAT Solving

Paper III focuses on improving the theoretical understanding of the effect of label-based preprocessing on the number of SAT solver calls made by SAT-based MaxSAT algorithms. We note that theoretical analysis of many SAT-based MaxSAT algorithms is in general challenging. For example, the number of SAT solver calls necessary for the CG algorithm to solve any MaxSAT instance is not known. The difficulty in determining the number stems to some extent from the fact that the instance is modified during search. Thus the core extracted on each iteration is a core of the current instance but not necessarily a core of the original instance. More generally, the effect that the formula rewriting step of the CG algorithm has on the core structure of the input instance remains an interesting open question, even if some results are known [233]. Analysis of the IHS algorithm is simpler since it only extracts cores of the original instance. It is known that the number of SAT solver calls necessary for the IHS algorithm to solve MaxSAT instances can be exponential in the number of soft clauses, even when restricted to unweighted instances [234].

In Paper III we provide a full characterization of the effect of label-based preprocessing with SAT-based preprocessing rules on the number of necessary SAT solver calls of two algorithms: IHS and CG^H. CG^H is an abstract MaxSAT algorithm first studied in [233]. In particular CG^H is similar to CG except for the fact that CG^H only considers implementations of Relax in which the soft clauses of the core remain in the instance as *soft clauses* and new cardinality constraints are added as *hard clauses*. In [233] the authors provide a characterization of the cores in the ith working formula of CG^H in terms of the cores of the input instance and the added (hard) cardinality constraints. Our results in Paper III for CG^H make use of this characterization. As a by-product of the main result, we also develop a similar characterization of the MUSes, thus sharpening the main results of [233].

In order to simplify the discussion, we will from now on focus on what we call *normalized MaxSAT instances*, a view on MaxSAT instances similar to [119]. A MaxSAT instance $\mathcal{F}^N = (F_h^N, F_s^N, w)$ is normalized if each soft clause $C \in F_s^N$ is group-detectable. We say that a literal $l \in \text{Lit}(\mathcal{F}^N)$ is a *soft literal* if $(\neg l) \in F_s^N$. The results of Papers I and II imply that we can assume all MaxSAT instances to be normalized. In more detail, given any MaxSAT instance $\mathcal{F} = (F_h, F_s, w)$ we construct a normalized instance $\mathcal{F}^N$ by replacing each $C \in F_s$ which is not group detectable by a hard clause

$C \vee g_C$ and a soft clause $(\neg g_C)$ of weight $w(C)$. The results of Papers I and II imply that $\mathcal{F}^N$ has the same optimal solutions as $\mathcal{F}$ and reusing all the group-detectable literals of $\mathcal{F}^N$ as assumptions allows solving $\mathcal{F}^N$ without introducing any extra variables compared to solving $\mathcal{F}$. Normalized MaxSAT instances are convenient for the analysis conducted in Paper III. If a normalized MaxSAT instance $\mathcal{F} = (F_h, F_s, w)$ is preprocessed with group detection to obtain the instance $pre(\mathcal{F}) = (F_h^p, F_s^p, w)$, then $F_s^p \subseteq F_s$, and Theorem 3 can be restated as follows.

Corollary 1. *Let $\mathcal{F}$ be a normalized MaxSAT instance and $pre(\mathcal{F})$ the instance resulting after preprocessing $\mathcal{F}$ using label-based preprocessing with group detection and SAT-based preprocessing rules. Then* $\mathrm{MUS}(\mathcal{F}) = \mathrm{MUS}(pre(\mathcal{F}))$*, which implies* $\mathrm{MCS}(\mathcal{F}) = \mathrm{MCS}(pre(\mathcal{F}))$.

While Paper III focuses on SAT-based preprocessing rules, the results hold for any preprocessing rules that satisfy Corollary 1.

In Paper III we analyze four variants of a fixed $\mathrm{A} \in \{\mathrm{CG}^H, \mathrm{IHS}\}$.

- A: the base algorithm.
- A_{pre}: A applied after label-based preprocessing.
- $\mathrm{A}^{\mathrm{MUS}}$: A using an idealized SAT solver that is guaranteed to extract an MUS when invoked on an unsatisfiable formula.
- $\mathrm{A}_{pre}^{\mathrm{MUS}}$: $\mathrm{A}^{\mathrm{MUS}}$ applied after label-based preprocessing.

We investigate the relative performance of these variants from two separate points of view, the *best case* and *worst case* in the number of iterations (SAT solver calls). Let $\mathcal{F}$ be a normalized MaxSAT instance and $\mathrm{A} \in \{\mathrm{CG}^H, \mathrm{IHS}\}$. Due to non-deterministic heuristics of SAT solvers, there are several different possible executions of A invoked on $\mathcal{F}$. We define the *length* of an execution of A on $\mathcal{F}$ as the number of cores extracted by A during that execution before termination. A *best-case* execution of A on $\mathcal{F}$ has length equal to the minimum over all possible executions of A on $\mathcal{F}$. Similarly, a *worst-case* execution has length equal to the maximum over all possible executions. Let $\textsc{Minlen}(\mathrm{A}, \mathcal{F})$ and $\textsc{Maxlen}(\mathrm{A}, \mathcal{F})$ denote the length of best-case and worst-case executions of A on $\mathcal{F}$, respectively.

Figures 3.5 and 3.6 overview the results of our analysis. We establish that for any $\mathrm{A} \in \{\mathrm{CG}^H, \mathrm{IHS}\}$, label-based preprocessing using SAT-based preprocessing rules can not decrease the length of the best-case executions of A on any instance (Figure 3.5). On the other hand, preprocessing can decrease the length of the worst-case executions on some instances. Intuitively, the results follow from the inability of SAT-based preprocessing

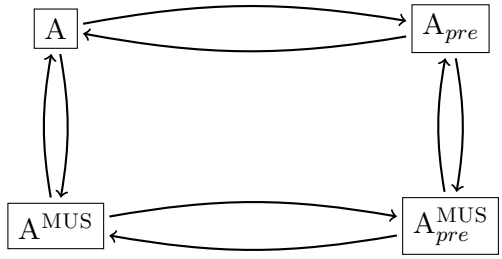

Figure 3.5: Best-case performance in the number of iterations of $A \in \{CG^H, IHS\}$. Here $X \to Y$ indicates that $\textsc{Minlen}(X, \mathcal{F}) \leq \textsc{Minlen}(Y, \mathcal{F})$ on all MaxSAT instances $\mathcal{F}$ (from Paper III).

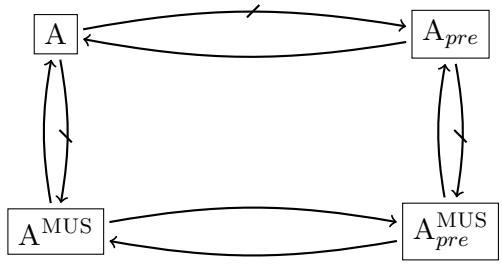

Figure 3.6: Worst-case performance in the number of iterations of $A \in \{CG^H, IHS\}$. Here $X \to Y$ indicates that $\textsc{Maxlen}(X, \mathcal{F}) \leq \textsc{Maxlen}(Y, \mathcal{F})$ on all MaxSAT instances $\mathcal{F}$. $X \nrightarrow Y$ indicates that $X \to Y$ does not hold (from Paper III).

rules to affect the MUS structure of MaxSAT instances, as stated in Corollary 1. In essence, we show that the best-case executions of A on any instance $\mathcal{F}$ correspond to executions where the SAT solver only returns MUSes. Since SAT-based preprocessing can not affect the MUSes of $\mathcal{F}$, any executions, where the SAT solver only returns MUSes, are valid for both A and A_{pre}. However, as preprocessing can still remove some of the soft literals of MaxSAT instances, applying SAT-based preprocessing before solving might in some cases allow the algorithm to avoid bad executions following from extracting cores that are not MUSes.

Finally, we note that, in addition to CG^H, IHS and the results of Paper III, it is known that $\mathcal{O}(\log(\mathrm{COST}(\mathcal{F})))$ calls to a SAT solver are required to solve any MaxSAT instance [235]. The result is obtained using a model-guided MaxSAT solver which uses a combination of linear and binary search for the optimal cost (recall Section 2.4). As preprocessing does not affect the optimal cost of instances, the same result implies that label-based preprocessing with SAT-based preprocessing rules can not affect the minimum number of SAT solver calls required by the model-guided solver studied in [235] either.

The focus in this section was on the number of iterations of SAT-based solvers. Even if preprocessing does not decrease the minimum number of iterations, the empirical evaluations conducted in Papers I and II demonstrate that preprocessing does improve the empirical performance of SAT-based MaxSAT solvers on some instances. While a complete theoretical understanding of how preprocessing affects SAT-based MaxSAT solvers remains an interesting open research question, we note that the analysis conducted in Paper III does not cover the effect of preprocessing on the individual SAT-solver calls executed by IHS and CG^H.

3.4 Subsumed Label Elimination

As the final contribution of this thesis to MaxSAT preprocessing, we investigate a MaxSAT-specific preprocessing rule which does not satisfy Corollary 1, but still guarantees sound MaxSAT preprocessing. More specifically, in Paper IV we propose and analyze subsumed label elimination (SLE). We prove correctness of SLE, give theoretical analysis comparing SLE to SAT-based preprocessing rules, and show empirically that incorporating SLE in the preprocessing step further improves performance of several MaxSAT solvers implementing CG and IHS. In the paper we also briefly overview the connection between SLE and the so-called *column dominance rule* proposed in the early 90s in conjunction with branch-and-bound approaches

for the *binate covering problem* [236]. We will not detail the binate covering problem here, except to say that while SLE can be seen as the MaxSAT counterpart of the column dominance rule, Paper III reports on the first study of such a rule in the context of MaxSAT that we are aware of.

Similarly to the previous section, we assume that all MaxSAT instances in this section are normalized. This allows identifying MCSes with sets of soft literals. More specifically, let $\mathcal{F} = (F_h, F_s, w)$ be a normalized MaxSAT instance and consider a subset $K \subseteq F_s$ of soft clauses. We are interested in determining the satisfiability of the formula $F_K = F_h \wedge (F_s \setminus K)$. Let $C \in K$ and $D \in (F_s \setminus K)$ be two soft clauses of $\mathcal{F}$. Then $C = (\neg l_1)$ and $D = (\neg l_2)$ for two soft literals l_1 and l_2. Now $\neg l_1 \notin \textsc{Lit}(F_K)$ since C is the only clause of $F_h \wedge F_s$ containing $\neg l_1$. Thus all clauses in $\textsc{Cl}_{F_K}(l_1)$ can be trivially satisfied by assigning l_1 to true. On the other hand, as $D \in F_K$, any potential satisfying assignment τ to F_K assigns l_2 to false. In other words, any potential satisfying assignment to F_K can set all soft literals l for which $(\neg l) \in K$ to true and has to set all literals l for which $(\neg l) \in F_s \setminus K$ to false. Hence the satisfiability of F_K is equivalent to the satisfiability of the simplification $F_h^{K^L}$ of F_h under the partial assignment $K^L = \{l \mid (\neg l) \in K\} \cup \{\neg l \mid (\neg l) \in F_s \setminus K\}$. Thus K is a correction set of $\mathcal{F}$ if and only if $F_h^{K^L}$ is satisfiable. In this section, we identify sets $M \subseteq F_s$ of soft clauses of normalized MaxSAT instances with the partial assignment $M^L = \{l \mid (\neg l) \in M\} \cup \{\neg l \mid (\neg l) \in F_s \setminus M\}$. Specifically, we define the simplification F_h^M of F_h under $M \subseteq F_s$ to be equal to $F_h^{M^L}$.

Next we give an informal description of SLE. Let $\mathcal{F} = (F_h, F_s, w)$ be a normalized MaxSAT instance, and l_1 and l_2 two soft literals of $\mathcal{F}$. We say that l_2 subsumes l_1 if (i) $\textsc{Cl}(l_1) \subseteq \textsc{Cl}(l_2)$, i.e., l_2 appears in all of the same clauses as l_1, and (ii) $w((\neg l_1)) \geq w((\neg l_2))$. SLE allows removing subsumed literals from $\mathcal{F}$. More formally, SLE allows enforcing all subsumed literals to false and simplifying the instance accordingly. The soundness of SLE follows from the fact that if l_1 is subsumed by l_2, then there exists an optimal $M \in \text{MCS}(\mathcal{F})$ which does not contain $(\neg l_1)$. Hence there also exists an optimal solution τ^M to $\mathcal{F}$ that assigns l_1 to false. More specifically, we show that if $(\neg l_1)$ is a member of some $M^1 \in \text{MCS}(\mathcal{F})$, then $M^2 = (M^1 \setminus (\neg l_1)) \cup (\neg l_2)$ is a correction set of $\mathcal{F}$. Thereby M^2 contains an MCS M_s of $\mathcal{F}$ for which $(\neg l_1) \notin M_s$. Notice that the assumption $w((\neg l_1)) \geq w((\neg l_2))$ implies that $w(M^1) \geq w(M^2) \geq w(M_s)$, so either M^1 is not optimal, or all three are. To see that M^2 is a correction set, notice that the simplifications of F_h under M^1 and M^2 satisfy $F_h^{M^2} \subseteq F_h^{M^1}$ and recall that $F_h^{M^1}$ is satisfiable[1].

[1]After publication of Paper IV we have discovered a minor error in the proof of the

In Paper IV we provide a formal proof of correctness of SLE on the LCNF level. In more detail, for any normalized MaxSAT instance $\mathcal{F} = (F_h, F_s, w)$, we show that the soft literal l_1 is subsumed by the soft literal l_2 if (i) $(\neg l_2)$ appears in the same MUSes of $\mathcal{F}$ as $(\neg l_1)$ and (ii) $w((\neg l_1)) \geq w((\neg l_2))$. However, as checking which literals belong to which MUSes is NP-hard, the first condition is probably not checkable in polynomial time. Instead, we show that $\textsc{Cl}(l_1) \subseteq \textsc{Cl}(l_2)$ is a sufficient condition for $(\neg l_2)$ appearing in the same MUSes as $(\neg l_1)$.

In addition to its proof of correctness, Paper IV also contains additional theoretical analysis of SLE. We show that, in contrast to the SAT-based preprocessing rules, SLE does not satisfy Corollary 1. Instead, we show that the MUSes of MaxSAT instances $pre(\mathcal{F})$ preprocessed with SLE are restrictions of the MUSes of $\mathcal{F}$ onto the soft clauses of $pre(\mathcal{F})$. The contrast between SLE and SAT-based preprocessing is further exemplified in Paper IV by MaxSAT instances on which no SAT-based preprocessing rules can be applied but SLE can. The possibility of SLE affecting the MUSes of MaxSAT instances also means that preprocessing with SLE can in some cases remove optimal MaxSAT solutions. More precisely, assume that two labels l_1 and l_2 subsume each other, i.e., that $\textsc{Cl}(l_1) = \textsc{Cl}(l_2)$ and $w((\neg l_1)) = w((\neg l_2))$. Then, if there exists an optimal $M_1 \in \text{MCS}(\mathcal{F})$ containing l_1 and not l_2, there also exists an optimal $M_2 \in \text{MCS}(\mathcal{F})$ containing l_2 and not l_1. Even so, SLE can soundly remove either l_1 or l_2, thus also removing the corresponding MCS and optimal MaxSAT solutions. However, as implied by the soundness of SLE, it will never remove all optimal solutions nor create new ones.

It should be noted that only using SLE during preprocessing might not be very effective. The reason is the precondition $\textsc{Cl}(l_1) \subseteq \textsc{Cl}(l_2)$. In order for this condition to be met, the formula needs to contain clauses with more than one soft literal. However, many normalized MaxSAT instances encountered in practical applications need not contain such clauses. Consider for example an un-normalized MaxSAT instance $\mathcal{F} = (F_h, F_s, w)$ that does not contain any group-detectable soft clauses. Then every clause in the normalized instance $\mathcal{F}^N = (F_h^N, F_s^N, w)$ obtained from $\mathcal{F}$ following Section 3.3 contains at most one soft literal. Even so, SLE can still be used when preprocessing $\mathcal{F}^N$ as long as other techniques capable of distributing soft literals among clauses are used as well. In our work the main preprocessing rule capable of this is bounded variable elimination.

---

theorem corresponding to this discussion, Theorem 6. The last sentence of the proof should read: "By assumption, $R' = (R \setminus \{l_2\}) \cup \{l_1\}$, a subset of $Lbls(\Phi^{pre})$, is a hitting set of LMUS(Φ) and hence contains an irreducible hitting set of LMUS(Φ), i.e, an LMCS of Φ."

In Paper IV we present the result of an experimental evaluation on the effect of SLE together with SAT-based MaxSAT solvers. As benchmarks we used the industrial and crafted benchmarks of the 2015 MaxSAT evaluation. Figure 3.7 demonstrates the effect that SLE has on the fraction of soft literals remaining after preprocessing with and without SLE. We found that especially on weighted instances, SLE can significantly increase the number of soft literals that are removed during preprocessing. For example, for one third of the weighted partial industrial instances ($x = 0.3$), with SLE close to 80% of the soft literals are eliminated ($y \approx 0.2$, i.e., some 20% of the soft literals remain afterwards). In comparison, without SLE only $\approx 45\%$ are eliminated.

Figure 3.8 gives a break-down of the effect that SLE has on the running time of LMHS on the different families of the weighted partial industrial benchmarks. The experiments were run using a per-instance timeout of 30 min. We see that, for a majority of the instances, SLE improves the total solving time of LMHS, both compared to using no preprocessing, and only using SAT-based preprocessing. In Paper IV, we also provide results for Eva [115], Open-WBO [113], and Primal-Dual [120].

After the publication of Paper IV we have generalized SLE, resulting in a technique called group-subsumed label elimination (GSLE) [237]. A soft literal l of a MaxSAT instance $\mathcal{F}$ is group-subsumed by a set L of soft literals if (i) $\textsc{Cl}(l) \subseteq \textsc{Cl}(L)$ and (ii) $w((\neg l)) \geq \sum_{l_g \in L} w((\neg l_g))$. GSLE allows removing group-subsumed literals from $\mathcal{F}$. GSLE is a straightforward generalization of SLE, the proof of correctness is essentially identical to SLE. Both SLE and GSLE are implemented in the MaxSAT preprocessor MaxPre [237], developed after publication of Paper IV. Interestingly, we have found that using GSLE during preprocessing does not significantly increase total preprocessing time compared to using SLE. In addition to SLE and GSLE, MaxPre also includes all other algorithmic ideas for preprocessing proposed in this thesis, namely label reuse and group detection.

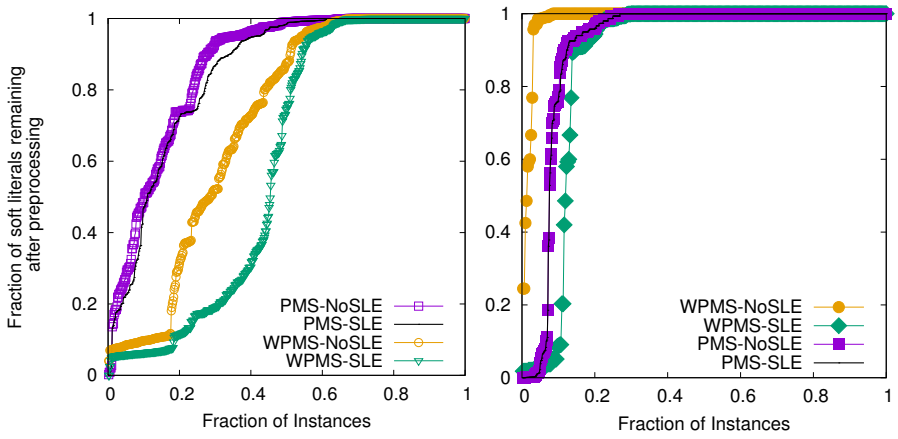

Figure 3.7: Fraction of soft literals remaining in industrial (left) and crafted (right) unweighted (PMS) and weighted (WPMS) benchmarks after preprocessing with and without SLE (from Paper IV).

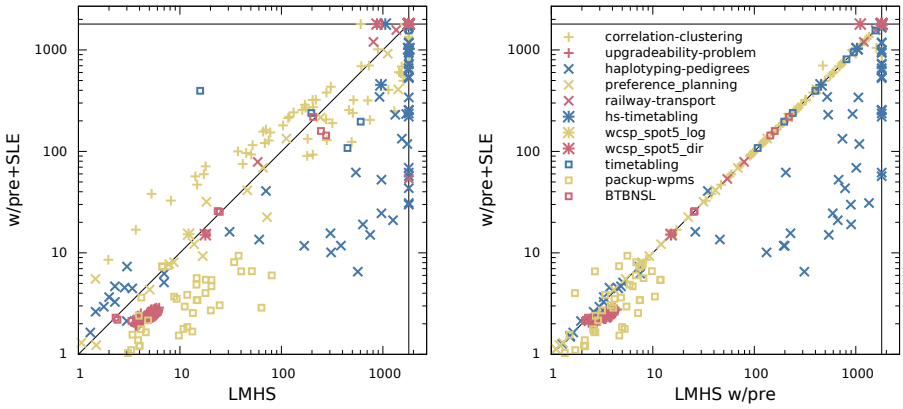

Figure 3.8: Effect of SLE on runtimes without (left) and with (right) SAT-based preprocessing of LMHS on industrial weighted partial instances (from Paper IV).

Chapter 4

Maximum Satisfiability for Data Analysis

In this chapter we discuss the contributions of this thesis to the development of new MaxSAT encodings for two NP-hard data analysis problems, correlation clustering (Paper V) and bounded treewidth Bayesian network structure learning (BTBNSL) (Paper VI). Correlation clustering is discussed in Section 4.1 and BTBNSL in Section 4.2. In both sections we give a precise definition of the data analysis task as a combinatorial optimization problem, overview the MaxSAT encodings we propose, and present a summary of the results of an empirical evaluation of the resulting MaxSAT-based approach.

In order to simplify the discussion, we will present all MaxSAT encodings in this chapter in terms of general propositional logic. This can be done without loss of generality as it is well-known that for any formula $\mathcal{G}$ in propositional logic, there exists an equivalent CNF formula $F^{\mathcal{G}}$. Furthermore, the size of $F^{\mathcal{G}}$ can be assumed to be polynomial in the size of $\mathcal{G}$. More specifically, applying the well-known Tseitin encoding [238] on $\mathcal{G}$ results in an equivalent CNF formula $F^{\mathcal{G}}$ the number of variables and clauses of which are linear in the number of constraints and variables of $\mathcal{G}$.

4.1 Correlation Clustering

In Paper V we present and evaluate three MaxSAT encodings for the correlation clustering problem [239]. Under the original formulation, an instance of correlation clustering consists of an undirected graph with the nodes corresponding to a set of data points and each edge labeled as either positive or negative. Two points with a positive edge between them are similar,

and points with a negative edge between them are dissimilar. The goal of correlation clustering is to cluster the nodes of the graph in a way that correlates as well as possible with the edge labeling. More specifically, an optimal clustering balances two conflicting objectives. On one hand, similar points should be assigned to the same cluster. On the other hand, dissimilar points should be assigned to different clusters. Notice that there exists "trivial" clusterings that maximize either individual objective. The number of similar points assigned to the same cluster is maximized by a clustering that assigns all nodes to the same cluster. Similarly, the number of dissimilar points assigned to different clusters is maximized by a clustering that assigns all nodes to different clusters. However, no such trivial clustering is in general optimal with respect to both objectives. Balancing the conflicting objectives is an important characteristic of correlation clustering. The lack of a "trivial" clustering that balances both objectives makes correlation clustering well-suited for situations in which the true number of clusters is unknown.

The rest of the Section is organized as follows. In Section 4.1.1 we detail the general setting under which we study correlation clustering. The three MaxSAT encodings we propose for the problem are presented in Section 4.1.2 and an overview of the results of the experimental evaluation reported on in Paper V is given in Section 4.1.3.

An interest in correlation clustering has continued after the publication of Paper V [240–243]. We especially note a recently published paper that develops one of the MaxSAT encodings that we propose in Paper V in the context of the clique partitioning problem [244].

4.1.1 Problem Setting

An instance of correlation clustering consists of a set $V = \{v_1, \ldots, v_N\}$ of N data points and a symmetric *similarity matrix* $W \in \overline{\mathbb{R}}^{N \times N}$ where $\overline{\mathbb{R}} = \mathbb{R} \cup \{\infty, -\infty\}$. From now on, we fix V and say that an instance consists only of the matrix W. We denote the element on row i column j in W by $W(i, j)$. The values of W represent similarities of pairs of points, the points v_i and v_j are *similar* if $W(i, j) > 0$ and *dissimilar* if $W(i, j) < 0$. An instance of correlation clustering can equivalently be viewed as an weighted undirected graph $G = (V, E)$ where $\{v_i, v_j\} \in E$ if $W(i, j) \neq 0$, and the weight of each edge $\{v_i, v_j\}$ is equal to $W(i, j)$. Figure 4.1 gives an example of a similarity matrix W on the left and the corresponding graph on the right.

A function $cl \colon V \to \mathbb{N}$ is a clustering of W if $cl(v_i) = cl(v_j)$ for all $W(i, j) = \infty$ and $cl(v_i) \neq cl(v_j)$ for all $W(i, j) = -\infty$. These kinds of

$$W = \begin{bmatrix} \infty & 0 & -2 & 5 \\ 0 & \infty & -\infty & -3 \\ -2 & -\infty & \infty & 1 \\ 5 & -3 & 1 & \infty \end{bmatrix}$$

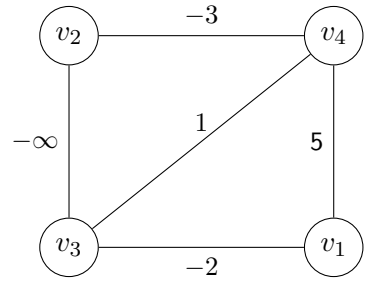

Figure 4.1: A Similarity matrix and its graph representation.

constraints enforcing two points to the same or to different clusters are commonly called *must-link* and *cannot-link* constraints, respectively [245].

Given an instance W of correlation clustering, the cost $\textsc{Cost}(W, cl)$ of a clustering $cl\colon V \to \mathbb{N}$ is

$$\textsc{Cost}(W, cl) = \sum_{\substack{cl(v_i)=cl(v_j) \\ i<j}} (\mathcal{I}[-\infty < W(i,j) < 0] \cdot |W(i,j)|) + \sum_{\substack{cl(v_i)\neq cl(v_j) \\ i<j}} (\mathcal{I}[\infty > W(i,j) > 0] \cdot W(i,j)),$$

where $\mathcal{I}[b]$ is an indicator function which takes the value 1 if the condition b is true, else $I[b] = 0$. A clustering cl is optimal if $\textsc{Cost}(W, cl) \leq \textsc{Cost}(W, cl')$ for all clusterings cl' of W.

Example 3. *Consider the similarity matrix W in Figure 4.1 (left). An optimal clustering cl of this instance assigns $cl(v_1) = cl(v_4) = 1$, $cl(v_2) = 2$ and $cl(v_3) = 3$. The cost $\textsc{Cost}(W, cl)$ of cl is 1.*

An important factor to note here is that the cost of a clustering does not depend on the specific cluster indexes. Hence the search for an optimal clustering of W can be restricted to functions $cl\colon V \to [N]$ where $[N] = \{0, \ldots, N-1\}$. More generally, given a similarity matrix W, a clustering $cl\colon V \to \mathbb{N}$ and a permutation $\sigma\colon \mathbb{N} \to \mathbb{N}$, the function $cl^\sigma = \sigma \circ cl$ is also a clustering of V with $\textsc{Cost}(W, cl) = \textsc{Cost}(W, cl^\sigma)$. In other words, the space of clusterings is highly symmetric.

4.1.2 MaxSAT Encodings of Correlation Clustering

In this section we overview our three related MaxSAT encodings of correlation clustering, the transitive encoding, the unary encoding and the binary

encoding. For the remaining of this section, we fix an instance W of correlation clustering to an $N \times N$ similarity matrix with E non-zero elements. In other words, we assume that there are E pairs of distinct $1 \leq i < j \leq N$ for which $W(i,j) \neq 0$.

For each encoding, we describe the MaxSAT instance $\mathcal{F}(W)$ resulting after applying the encoding on W and a procedure for converting an optimal solution τ of $\mathcal{F}(W)$ to an optimal clustering cl^τ of W. In the rest of the section, let $\textsc{Transitive}(W)$, $\textsc{Unary}(W)$ and $\textsc{Binary}(W)$ denote the MaxSAT instances produced by the transitive, unary and binary encodings on W, respectively. Even though the specifics of each instance differ, the overall structure of them is similar. Let $\mathcal{F}(W) \in \{\textsc{Transitive}(W), \textsc{Unary}(W), \textsc{Binary}(W)\}$ and $\mathcal{F}(W) = (F_h^W, F_s^W, w^W)$. The hard clauses F_h^W represent a conjunction of two complex constraints: $F_h^W = \textsc{IsFunc}(W) \wedge \textsc{Sol}(W)$. The constraint $\textsc{IsFunc}(W)$ is satisfied if cl^τ is a function $cl^\tau \colon V \to \mathbb{N}$, and $\textsc{Sol}(W)$ is satisfied if cl^τ is a clustering of W. The constraint $\textsc{Sol}(W)$ is further divided into two parts corresponding to the must-link ($\textsc{SameCl}(i,j)$) and cannot-link ($\textsc{DiffCl}(i,j)$) constraints, respectively:

$$\textsc{Sol}(W) = \bigwedge_{W(i,j)=\infty} \textsc{SameCl}(i,j) \wedge \bigwedge_{W(i,j)=-\infty} \textsc{DiffCl}(i,j).$$

The constraint $\textsc{SameCl}(i,j)$ is satisfied if and only if $cl^\tau(v_i) = cl^\tau(v_j)$ and the constraint $\textsc{DiffCl}(i,j)$ is satisfied if and only if $cl^\tau(v_i) \neq cl^\tau(v_j)$. Since all hard clauses are satisfied by any solution to the MaxSAT instance, the semantics of the constraints ensure that any solution to the MaxSAT instance corresponds to a clustering of W.

The soft clauses F_s^W are defined in a way which ensures that

$$\textsc{Cost}(\mathcal{F}, \tau) = \textsc{Cost}(W, cl^\tau)$$

for any MaxSAT solution τ. The soft clauses contain $\textsc{SameCl}(i,j)$ with weight $w^W(\textsc{SameCl}(i,j)) = W(i,j)$ for each $0 < W(i,j) < \infty$ and a constraint $\textsc{DiffCl}(i,j)$ with weight $w^W(\textsc{DiffCl}(i,j)) = |W(i,j)|$ for each $-\infty < W(i,j) < 0$; all in all,

$$F_s^W = \bigwedge_{0<W(i,j)<\infty} \textsc{SameCl}(i,j) \wedge \bigwedge_{0>W(i,j)>-\infty} \textsc{DiffCl}(i,j).$$

In Paper V, the correctness of each MaxSAT encoding is established by showing that $\textsc{Cost}(\mathcal{F}, \tau) = \textsc{Cost}(W, cl^\tau)$ and that for *any* clustering cl of W there exists a solution τ_{cl} to $\mathcal{F}(W)$ for which $cl = cl^{\tau_{cl}}$.

The transitive encoding of correlation clustering can be seen as a MaxSAT reformulation of a previously proposed integer programming model for correlation clustering, a model originally proposed for the clique partitioning problem [168, 175, 246]. The variables of TRANSITIVE(W) are of form x_{ij} for each distinct pair $i, j = 1, \ldots, N$. Given a solution τ to TRANSITIVE(W), the corresponding clustering cl^τ is constructed by assigning all v_i and v_j for which $\tau(x_{ij}) = 1$ to the same cluster. With these variables, the constraint ISFUNC(W) is encoded using $\theta(N^3)$ constraints of form $(x_{ij} \wedge x_{jk}) \rightarrow x_{ik}$ for distinct i, j, and k. Each such constraint corresponds to the clause $(\neg x_{ij} \vee \neg x_{jk} \vee x_{ik})$; all in all,

$$\text{ISFUNC}(W) = \bigwedge_{i,j,k \text{ distinct}} (\neg x_{ij} \vee \neg x_{jk} \vee x_{ik}).$$

The two other constraints, SAMECL(i, j) and DIFFCL(i, j), are encoded with unit clauses: SAMECL$(i, j) = (x_{ij})$ and DIFFCL$(i, j) = (\neg x_{ij})$. In total TRANSITIVE(W) contains $\theta(N^2)$ variables and $\theta(N^3)$ clauses.

The unary encoding of correlation clustering resembles to some extent a previously proposed quadratic integer programming formulation of correlation clustering [247]. In contrast to the transitive encoding, the unary encoding is parametrized on K, the maximum number of clusters in the solution clustering. The value of K needs to be set before creating the instance UNARY(W). In Paper V, we used $K = N$ in all experiments. This ensures that the produced clustering is an optimal solution to the general correlation clustering problem (recall the discussion at the end of Section 4.1.1). The main variables of UNARY(W) are the $\theta(N \cdot K)$ variables of form y_i^k for $1 \leq i \leq N$ and $1 \leq k \leq K$. Given a solution τ to UNARY(W), the clustering cl^τ is constructed using those variables by setting $cl^\tau(v_i) = k$ if and only if $\tau(y_i^k) = 1$. The constraint ISFUNC(W) is encoded using cardinality constraints:

$$\text{ISFUNC}(W) = \bigwedge_{i=1}^{N} \text{CNF}(\sum_{k=1}^{K} y_i^k = 1).$$

In Paper V we used the so-called *sequential encoding* [215] to convert the cardinality constraints to CNF. The other two constraints, SAMECL(i, j) and DIFFCL(i, j), are encoded in a straightforward manner by

$$\text{SAMECL}(i, j) = \bigvee_{k=1}^{K} (y_i^k \wedge y_j^k) \text{ and } \text{DIFFCL}(i, j) = \bigwedge_{k=1}^{K} \neg(y_i^k \wedge y_j^k).$$

Including all clauses and variables introduced by the Tseitin encoding, the instance UNARY(W) contains $\theta(E \cdot K + N \cdot K)$ variables and $\theta(E \cdot K)$ clauses.

The third MaxSAT encoding we consider, the binary encoding, is essentially a bitwise reformulation of the unary encoding. Similarly to the unary encoding, the binary encoding is also parametrized on K. The main variables of $\textsc{Binary}(W)$ are $\theta(N \cdot \log_2(K))$ variables of form b_i^a for $1 \leq i \leq N$ and $1 \leq a \leq \lceil \log_2(K) \rceil$. Given a solution τ to $\textsc{Binary}(W)$, the clustering cl^τ is constructed by interpreting $\tau(b_i^{\lceil \log_2(K) \rceil}), \ldots, \tau(b_i^1)$ as a binary number c and setting $cl^\tau(v_i) = c$. We note that this construction results in a clustering $cl^\tau \colon V \to 2^{d'}$ for the smallest d' for which $2^{d'} \geq K$. In Paper V, we present extra constraints that can be added to $\textsc{Binary}(W)$ to ensure that $cl^\tau(v_i) \leq K < N$ for all $1 \leq i \leq N$. Notice that such constraints are not needed if $K = N$, instead the cluster indexes of cl^τ can be permuted to the interval $0, \ldots, N-1$ as a post-processing step.

A convenient consequence of the construction of $\textsc{Binary}(W)$ is that $\textsc{IsFunc}(W) = \emptyset$, i.e., no clauses are required in order to ensure that cl^τ is a function. The encoding of $\textsc{SameCl}(i,j)$ and $\textsc{DiffCl}(i,j)$ is straightforward: two points v_i and v_j are assigned to the same cluster by cl^τ if and only if all bits in the binary representation of their cluster numbers are equal, i.e.,

$$\textsc{SameCl}(i,j) = \bigwedge_{k=1}^{\log_2(K)} (b_i^k \leftrightarrow b_j^k) \text{ and } \textsc{DiffCl}(i,j) = \bigvee_{k=1}^{\log_2(K)} \neg(b_i^k \leftrightarrow b_j^k).$$

In total $\textsc{Binary}(W)$ contains $\theta(E + N \cdot \log_2 K)$ variables and $\theta(E \cdot \log_2 K)$ clauses.

In Paper V we also consider different types of redundant constraints designed to reduce the symmetries in the binary encoding. As discussed in the previous section, the space of clusterings of W is very symmetric. Several of the symmetries are transferred to the space of MaxSAT solutions of $\textsc{Binary}(W)$. Some of the symmetries can be broken by adding extra constraints to $\textsc{Binary}(W)$. For a simple example of such an constraint, we can enforce the point v_1 to be assigned to cluster 0 with the constraint $\bigwedge_{k=1}^{\lceil \log_2(K) \rceil}(\neg b_1^k)$. Adding extra symmetry breaking constraints to the instance is non-trivial in general. While such constraints have the potential of decreasing the solving time of the instance, adding too many extra constraints can instead increase the size of the instance enough to degrade the performance of the MaxSAT solver. Notice for example that the transitive encoding naturally breaks all symmetries related to cluster indexing while being significantly larger than the other two instances. In Paper V we report on an experimental evaluation of some possible symmetry breaking constraints that could be used in conjunction with the binary encoding.

4.1.3 Experimental Evaluation

In Paper V we report on an experimental evaluation of the applicability of MaxSAT for solving correlation clustering. As benchmarks we used four sets of similarity values between amino-acid sequences of proteins [248], and seven different benchmark sets from the UCL machine learning repository [249]. The number of data points in the benchmark sets ranges from 327 to 990. In this section we overview the most significant results of the experiments using two of the four protein datasets, which we will from now on refer to as protein 1 and 2. The protein 1 dataset contains 669 data points and the protein 2 dataset contains 586 data points. The similarity values between the amino acid sequences in the sets were originally computed using BLAST [250], and the datasets were obtained from `http://www.paccanarolab.org/scps`.

In our experiments we compare our MaxSAT encodings with other previously proposed exact approaches to correlation clustering: an the integer linear programming (ILP) model [168, 175] and an quadratic integer programming (QIP) model [247]. All ILP and QIP models were solved with IBM CPLEX (version 12.6) and Gurobi Optimizer (version 6.0) using default settings. In our evaluation, all MaxSAT instances were solved using the 2013 evaluation version of MaxHS [121, 122]. The choice of MaxHS was motivated by it performing better than Eva500 [115], MsUnCore BCD2 version [110, 112], OpenWBO [113, 117] and ILP2013 [251] in preliminary experimentation. A per-instance time limit of 8 h was enforced on all solver runs. In addition to exact approaches we also compare our MaxSAT encodings with four polynomial-time inexact algorithms: KwickCluster (KC) [168], SDPC [252] and SCPS and CCA [248]. Out of these, the KC and SDPC algorithms were proposed for the general correlation clustering problem, while SCPS and CCA were proposed specifically to cluster the protein datasets. The semidefinite programs of SDPC were solved with the Matlab package SeDuMi 1.3 [253].

The first experiment we report on investigated the scalability of the exact approaches with respect to the number of data points in the input instance. For an increasing n, we formed a pruned similarity matrix W^n by taking the n first data points of the protein 1 dataset. Figure 4.2 shows the result of this test with n, the number of datapoints used, on the x-axis and the time required to solve W^n on the y-axis. The reason for the QIP model missing from the plot is that neither CPLEX nor Gurobi could solve any of the instances within 8 h, an observation we verified using the non-commercial SCIP [254] solver as well. From the figure we can clearly see that the binary MaxSAT encoding scales the best and is the only one able

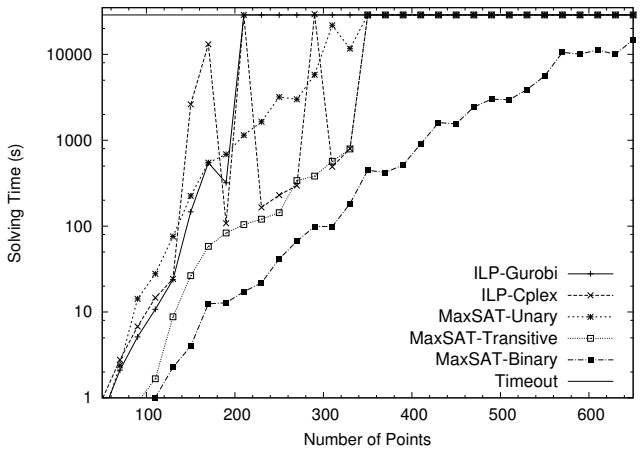

Figure 4.2: Point scalability of the exact approaches on the Protein 1 dataset (from Paper V).

to solve the full protein 1 dataset. Furthermore, many of the failed runs of the other exact approaches were due to memory-outs, suggesting that the algorithms would not terminate regardless of the time limit used.

The second experiment we report on was designed to investigate the scalability and quality of solutions obtained by the binary MaxSAT encoding with respect to the number of nonzero values in the similarity matrix. For $p \in \{0.05, 0.10, \ldots, 1\}$, we pruned the input similarity matrix W generated from either the protein 1 or protein 2 dataset by independently putting each nonzero value $W(i, j)$ to 0 with probability p. This approach results in a similarity matrix W' where the expected number of nonzero entries is $(1 - p) \cdot 100\%$ of the number of nonzero entries in W. As can be seen from the top row of Figure 4.3, instances with fewer nonzero values are faster to solve with MaxHS. While this is hardly surprising, a more interesting observation of this experiment concerns the cost of the clusterings obtained by solving sparse instances. The bottom row of Figure 4.3 shows the cost $\textsc{Cost}(W, cl^p)$ of the optimal clustering cl^p of the pruned instance W' with respect to the complete instance W. The costs are plotted both for the exact MaxSAT method and the approximative algorithms KC, SDPC, SCPS and CCA. For both protein 1 and 2, we found that the clustering obtained by MaxSAT invoked on an instance with 60% ($p = 0.4$) of the non-zero values pruned was of lower cost than the clustering obtained by any of the inexact algorithms when invoked on the complete instance. These results suggest that first pruning a significant amount of the non-zero values of

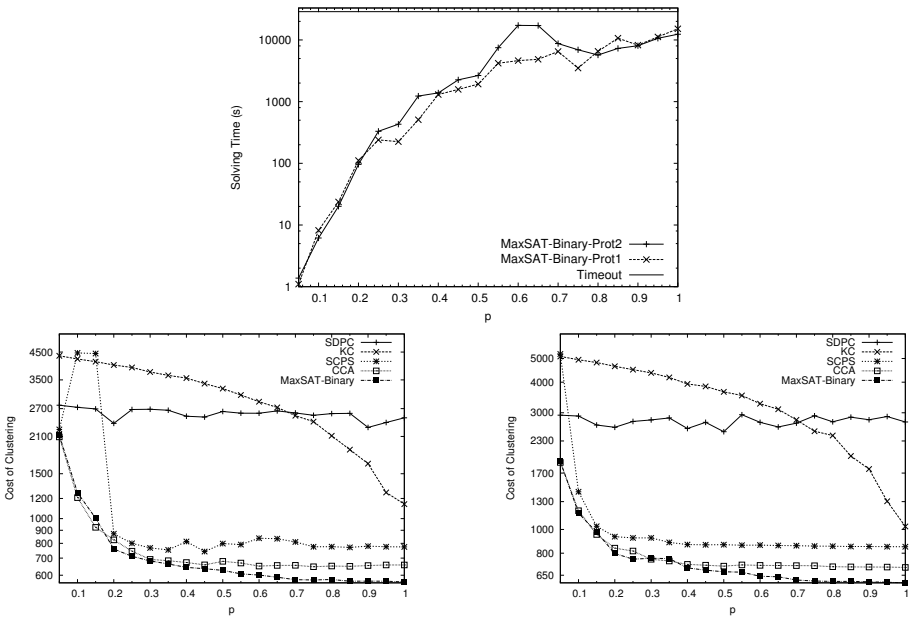

Figure 4.3: Top: Evolution of running times. Bottom: Cost of the clusterings obtained on sparse matrices. Bottom left: Protein 1 (669 datapoints), bottom right: Protein 2 (586 datapoints).

the matrix and then solving the sparser matrix using the binary MaxSAT encoding results in a competitive inexact approach to applications of correlation clustering with medium-size instances as long as running times in the order of a few minutes are acceptable. For applications where faster running times are required or larger instances need to be solved, the other inexact algorithms should be considered. All runs of KC, SDPC, SCPS and CCA reported on in Paper V were completed within a few seconds.

In addition the experiments on unconstrained correlation clustering, Paper V also includes results of experiments on the *constrained correlation clustering* problem. Constrained correlation clustering extends the correlation clustering problem by allowing extra hard constraints in the instance. We report on experiments evaluating the effect of extra symmetry breaking constraints as well as user specified must-link and cannot-link constraints which might come from an domain expert. We found that in many settings, adding extra constraints to the instance decreases the running time of the MaxSAT solver while not significantly increasing the cost of the produced clustering. As far as we are aware, adding similar constraints to the in-

exact algorithms is non-trivial. This further highlights the benefits of a declarative approach to solving correlation clustering.

4.2 Bounded Treewidth Bayesian Network Structure Learning

In Paper VI we propose and evaluate a MaxSAT encoding of the bounded treewidth Bayesian network structure learning problem (BTBNSL). We compare the resulting MaxSAT-based approach to the dynamic programming algorithm of [198], which at the time of publication of Paper VI was the only other known implementation of a solution algorithm to BTBNSL. Since the publication of Paper VI there has been a continued interest in BTBNSL [200, 255–257]. For example, an integer programming-based solution algorithm was published concurrently with Paper VI [199].

Given a set of observations (data) $\mathcal{D}$ over some set X of random variables, the goal of Bayesian network structure learning is to compute a Bayesian network structure which summarizes statistical dependencies and independencies in the data. BTBNSL further restricts the set of feasible solutions to networks that have treewidth less than some given bound $k \in \mathbb{N}$. In the score-based approach to BTBNSL, which we focus on, a scoring function Score is precomputed based on the data. The scoring function assigns a score $\textsc{Score}(G)$ to each possible network structure $G = (X, E)$. The score measures how well G explains the data, an optimal network minimizes Score over all possible networks. Our MaxSAT encoding is applicable under any *decomposable* scoring function. We give a precise definition of a scoring function being decomposable in the next section and note here that several commonly used scoring functions are decomposable, including MDL [258], BD [259], and fNML [11]. In the rest of the section, we assume that all scores are given as input and work with a generic decomposable scoring function Score.

This section is organized as follows. In Section 4.2.1 we detail BTBNSL and discuss how the treewidth of a Bayesian network can be computed. The MaxSAT encoding of BTBNSL is presented in Section 4.2.2 and an overview of the results of the experimental evaluation conducted in Paper VI is given in Section 4.2.3.

4.2.1 Problem Setting

Let $X = \{X_1, \ldots, X_N\}$ be a set of N random variables, and for each $i = 1, \ldots, N$, let $\mathcal{P}_i = 2^{X \setminus \{X_i\}}$ be the set of *candidate parent sets* of X_i.

An instance of BTBNSL consists of X, a bound $k \in \mathbb{N}$, and for each $i = 1, \ldots, N$ the set $\mathcal{P}_i$ as well as a *local score function* $s_i \colon \mathcal{P}_i \to \mathbb{N}$ associating a positive cost $s_i(P)$ to each $P \in \mathcal{P}_i$. Picking a single $P_i \in \mathcal{P}_i$ for each X_i gives rise to the directed graph $G = (X, E)$ in which $(X_j, X_i) \in E$ if and only if $X_j \in P_i$. We say that any X_j for which $(X_j, X_i) \in E$ is a parent of X_i, and X_i is a child of X_j. The graph G is a solution to the BTBNSL instance if it is acyclic and has treewidth less than k, i.e., if $\mathrm{TW}(G) \leq k$. The score $\textsc{Score}(G)$ of a solution G is equal to the sum of the local scores of each node and its parent set:

$$\textsc{Score}(G) = \sum_{X_i \in X} s_i(P_i). \tag{4.1}$$

As a side note, we say that any scoring function $\textsc{Score}$ for which the value $\textsc{Score}(G)$ can be computed similarly to Equation 4.1, is *decomposable*. A solution G^o is optimal if $\textsc{Score}(G^o) \leq \textsc{Score}(G)$ for any solution G.

Figure 4.4 illustrates the definition and computation of the treewidth of a Bayesian network $G = (X, E)$ [202, 260]. The treewidth of G is equal to the treewidth of its (undirected) moralized graph $\textsc{Moral}(G) = (X, E^M)$, obtained from G by adding an edge between any two nodes X_i and X_k that share a common child and dropping the direction of all edges. Given a linear ordering $\prec$ of X and two nodes $X_i, X_j \in X$, the node X_i is a *predecessor* of X_j (under $\prec$) if $X_i \prec X_j$ and $\{X_i, X_j\} \in E^M$. The undirected triangulation $\Delta(\textsc{Moral}(G), \prec)$ of $\textsc{Moral}(G)$ under $\prec$ is obtained by iteratively adding edges to E^M between pairs X_j and X_k of nodes that share a common predecessor. The edges are added until fix point. Finally, the directed ordered graph $\vec{\Delta}(\textsc{Moral}(G), \prec)$ is obtained from the resulting triangulation by ordering all edges according to $\prec$. The width of $\prec$ is the maximum out-degree of any node of $\vec{\Delta}(\textsc{Moral}(G), \prec)$. The treewidth $\mathrm{TW}(G)$ of G is the minimum-width of all linear orderings of X.

4.2.2 MaxSAT Encoding of BTBNSL

In this section we overview our MaxSAT encoding of BTBNSL. Given an instance of BTBNSL over $X = \{X_1, \ldots, X_N\}$ and a bound $k \in \mathbb{N}$, the encoding produces the MaxSAT instance $\textsc{Bayes}(X, k) = (F_h^X, F_s^X, w)$. For each variable $X_i \in X$ and potential parent set $S \in \mathcal{P}_i$, the instance $\textsc{Bayes}(X, k)$ includes a variable P_i^S. Given a solution τ to $\textsc{Bayes}(X, k)$, the graph $G^\tau = (X, E^\tau)$ corresponding to τ has $(X_j, X_i) \in E^\tau$ if and only if $X_j \in S$ for a $S \in \mathcal{P}_i$ for which $\tau(P_i^S) = 1$. The hard clauses of $\textsc{Bayes}(X, k)$ enforce that G^τ is a solution to X. The soft clauses ensure that $\textsc{Cost}(\mathcal{F}, \tau) = \textsc{Score}(G^\tau)$.

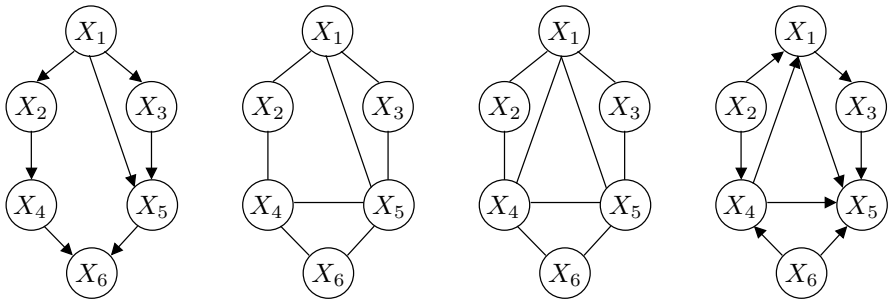

Figure 4.4: Computing the treewidth of a Bayesian network structure $G = (X = \{X_1, \ldots, X_6\}, E)$. (a) G; (b) the moralized graph $\textsc{Moral}(G) = (X, \mathrm{M}(E))$ of G; (c) the triangulation $\Delta(\textsc{Moral}(G), \prec)$ of the moralized graph under the linear ordering $X_6 \prec X_2 \prec X_4 \prec X_1 \prec X_3 \prec X_5$; (d) the ordered graph $\vec{\Delta}(\textsc{Moral}(G), \prec)$.

Next we overview the structure of $\textsc{Bayes}(X, k)$. Compared to the MaxSAT encodings for correlation clustering, the CNF conversions of the constraints in $\textsc{Bayes}(X, k)$ are somewhat involved. We refer the reader to Paper VI for the details. The hard clauses F_h^X of $\textsc{Bayes}(X, k)$ enforce that the graph G^τ corresponding to a solution τ of $\textsc{Bayes}(X, k)$ is a solution to the BTBNSL instance X. The clauses in F_h^X represent a conjunction of three different complex constraints:

$$F_h^X = \bigwedge_{i=1}^{N} \mathrm{CNF}\Big(\sum_{S \in \mathcal{P}_i} P_i^S = 1\Big) \wedge \mathrm{ACYC}(X) \wedge \mathrm{TWCNF}(X, k).$$

The constraint $\mathrm{CNF}(\sum_{S \in \mathcal{P}_i} P_i^S = 1)$ is satisfied if and only if G^τ has a single parent set S for each X_i. In Paper VI, we used the improved sequential counter [261] to encode this cardinality constraint to CNF.

The constraint $\mathrm{ACYC}(X)$ is satisfied if and only if the graph G^τ is acyclic. The CNF encoding of $\mathrm{ACYC}(X)$ assigns a level number $l(X_i) \in \mathbb{N}$ to each node $X_i \in X$ and enforces that the level number of a parent X_j of a node X_i satisfies $l(X_j) \leq l(X_i)$. In more detail, we model the level number $l(X_i)$ of each node X_i in binary using $\log_2(N)$ variables $b_i^1, \ldots, b_i^{\log_2(N)}$. We enforce $l(X_j) \leq l(X_i)$ by modeling the statement "the most significant bit in which the binary representations of $l(X_i)$ and $l(X_j)$ differ is 1 in $l(X_i)$".

The constraint $\mathrm{TWCNF}(X, k)$ is satisfied if and only if $\mathrm{TW}(G^\tau) \leq k$. The CNF translation of $\mathrm{TWCNF}(X, k)$ follows the SAT encoding for computing the treewidth of a fixed graph presented in [261]. Essentially,

we enforce the existence of a linear ordering of the variables in X that has width at most k. This is enough to ensure that $\mathrm{TW}(G^\tau) \leq k$ as the treewidth is equal to the minimum width over all possible orderings.

The soft clauses F_s^X contain a unit negation $(\neg P_i^S)$ with weight $s_i(S)$ for all $i = 1, \ldots, N$ and $S \in \mathcal{P}_i$. These ensure that the cost incurred by selecting S as a parent for X_i is $s_i(S)$, as expected. These clauses ensure that $\textsc{Cost}(\mathcal{F}, \tau) = \textsc{Score}(G^\tau)$ as required to make sure that the Bayesian network structure G^τ corresponding to an optimal solution τ to $\textsc{Bayes}(X, k)$ is an optimal solution to BTBNSL.

4.2.3 Experimental Evaluation

In Paper VI we report on an experimental evaluation evaluating of the applicability of MaxSAT for solving BTBNSL. As benchmark sets we used eight well-known UCl datasets [249] over 9–29 random variables, as well as two datasets (Adult and Housing) from [198]. Table 4.1 summarizes the benchmark sets. For each benchmark, we used the (decomposable) MDL scoring function [258]. We compare our MaxSAT encoding with the dynamic programming (DP) algorithm for BTBNSL of [198], the only other implementation of an exact algorithm for BTBNSL we were aware of at the time. All MaxSAT instances were solved with the MaxHS [121] solver. A per instance time limit of 8 h and memory limit of 30 GB were enforced on all benchmarks and solver runs. We used $k = 2, 3$ and 4 as bounds for the treewidth of the solution network.

Table 4.1 overviews the results of our experiment. For each treewidth bound, the best running time to find an optimal solution is highlighted in boldface. We observe that the dynamic programming approach is competitive with our MaxSAT approach only for the smallest dataset with 9 variables. Apart from the multiple timeouts ("> 28 800"), we observe that DP most often runs out of memory ("mo") on the datasets with more variables, especially for treewidth bounds greater than 2. In contrast, the MaxSAT approach (MS) timeouts on only two instances, and, especially, does not run out of memory. For a clear 2/3 majority of the instances, MS produces an optimal solution within half-an-hour; and for half of the instances within around 10 minutes.

Table 4.1: Running times in seconds of our MaxSAT-based approach (**MS**) and the dynamic programming (**DP**) approach [198] for different UCI datasets and treewidth bounds $k = 2, 3, 4$. Explanations: "mo" denotes a memory out; N denotes the number of variables (nodes); **#fails** denotes the number of times the memory or time limit was exceeded.

		treewidth ≤ 2		**treewidth** ≤ 3		**treewidth** ≤ 4		**#fails**	
Dataset	N	**MS** (s)	**DP** (s)	**MS** (s)	**DP** (s)	**MS** (s)	**DP** (s)	**MS**	**DP**
Abalone	9	64	**7**	166	**57**	**215**	536	0	0
Housing	14	**2 226**	6 927	**2 329**	> 28 800	**2 991**	mo	**0**	2
Wine	14	**27**	6 924	**22**	> 28 800	**171**	mo	**0**	2
Adult	15	**998**	> 28 800	**1 623**	> 28 800	**1 782**	mo	**0**	3
Voting	17	**22 909**	> 28 800	**26 419**	mo	> 28 800	mo	**1**	3
Zoo	17	**410**	> 28 800	**412**	mo	**105**	mo	**0**	3
Hepatitis	20	**315**	mo	**100**	mo	**1 164**	mo	**0**	3
Heart	23	**1 198**	mo	**2 186**	mo	**41**	mo	**0**	3
Horse	28	**192**	mo	> 28 800	mo	**544**	mo	**1**	3
Flag	29	**1 418**	mo	**11 148**	mo	**1 356**	mo	**0**	3
#fails:		**0**	7	**1**	9	**1**	9	**2**	25

Chapter 5

Conclusion

This thesis contributed to declarative methods for exactly solving combinatorial optimization problems. We focused on MaxSAT encodings and re-encodings of combinatorial optimization problems with the aim of solving instances that correspond to real-world applications.

In Papers I-IV we studied MaxSAT solving technology in the form of solver independent re-encodings, i.e., preprocessing, of MaxSAT instances $\mathcal{F}$ to other instances $pre(\mathcal{F})$ with the aim of making the time required to re-encode $\mathcal{F}$ and solve $pre(\mathcal{F})$ less than the time required to solve $\mathcal{F}$. In Papers I and II we further developed the previously proposed labeled CNF framework for MaxSAT preprocessing. In Paper I we showed that the extra label variables introduced during label-based preprocessing can be reused as assumption variables in many core-guided and implicit hitting set MaxSAT solvers, thus avoiding all variables that otherwise would be introduced by the solvers. We also showed that reusing labels as assumptions is necessary in order to improve the empirical performance of LMHS, an implicit hitting set based MaxSAT solver.

In Paper II we generalized the idea proposed in Paper I further by showing that some literals from the input MaxSAT instance itself can be used as labels during preprocessing and assumptions during solving. We demonstrated that such literals can be identified using simple pattern matching, resulting in a procedure we call group detection. Our empirical results indicate that group detection identifies a significant fraction of the literals in the evaluation benchmarks and that reusing detected literals results in modest further improvements to the empirical performance of LMHS.

Even though the ideas presented in Papers I and II are theoretically applicable to both core-guided and implicit hitting set based MaxSAT solvers, in practice we observed a more significant benefit of label-based preprocessing in conjunction with implicit hitting set based MaxSAT solvers. The

relationship between label-based preprocessing and the formula rewriting performed by core-guided solvers remains an open and interesting question. A better understanding of the formula rewriting could result in improved performance of core-guided solvers and label-based preprocessing. Another approach to further improving the performance label-based preprocessing in SAT-based MaxSAT solvers could be via some form of *inprocessing*, i.e via preprocessing steps interleaved with the execution of the solving algorithm. It could also be interesting to investigate if similar ideas could be used in order to develop preprocessing in constraint programming [262, 263] or other constraint optimization paradigms.

In Paper III we presented the results of a theoretical analysis on the effect of label-based preprocessing with SAT-based preprocessing rules on core-guided and implicit hitting set based MaxSAT solvers. We showed that preprocessing can not decrease the number of iterations (SAT solver calls) required by either algorithm, but can help them avoid some long executions. As discussed at the end of Section 3.3, the results of Paper III highlight some potentially beneficial approaches to the further development of MaxSAT preprocessing techniques. Since preprocessing has limited effect on the number of iterations of MaxSAT solvers, the overall benefit of preprocessing on MaxSAT solving could be further improved by developing preprocessing techniques that allow faster core-extraction from unsatisfiable instances. Furthermore, even though label-based preprocessing with SAT-based preprocessing rules does not affect the MCS structure of MaxSAT instances, it can still affect the solutions corresponding to the MCSes. A better understanding of the effect that preprocessing has on the minimal solutions that correspond to optimal MCSes could result in improved preprocessing techniques.

It should also be mentioned that the base of the results presented in Paper III (Corollary 1 in Section 3.3) is actually stronger than what is required for sound MaxSAT preprocessing. An argument similar to the proof of Theorem 4 can be used to show that label-based preprocessing with SAT-based preprocessing rules preserves all minimal solutions to MaxSAT instances, not only the optimal ones. Thus preprocessing rules that are sound but do not satisfy Corollary 1 could affect the number of iterations of SAT-based MaxSAT solvers more significantly. Finally, it should be noted that the abstraction of core-guided solvers considered Paper III does not cover more recently proposed solvers, specifically the ones that introduce soft cardinality constraints. The effect of soft cardinality constraints on the MUS structure of MaxSAT instances remains an interesting open question. Developing a better understanding of how the formula rewrit-

ings used by core-guided solvers affect the MCSes of the instance could potentially improve both MaxSAT preprocessing and MaxSAT solving.

As the final contribution to MaxSAT preprocessing of this thesis, we proposed subsumed label elimination (SLE) in Paper IV. We showed that SLE is theoretically orthogonal to the SAT-based preprocessing rules in the sense that using SLE together with the SAT-based preprocessing rules can result in additional clauses and variables removed during preprocessing. We also demonstrated that, even though SLE is sound for MaxSAT, it does not preserve all MCSes of MaxSAT instances. Hence, an interesting further research direction would be to investigate the effect that SLE has on the number of iterations of SAT-based MaxSAT solvers. We hypothesize that the effect of SLE is similar to that of SAT-based preprocessing rules, but do not have a proof at this time. In addition to the theoretical results, Paper IV also demonstrated empirically that using SLE together with SAT-based preprocessing rules results in more variables and clauses being removed during preprocessing and in a decrease of the overall solving time of LMHS. These observations motivate further development of MaxSAT-specific preprocessing rules that make direct use of the label variables and weights of the soft clauses.

Papers V and VI proposed MaxSAT encodings of two data analysis tasks: correlation clustering and bounded treewidth Bayesian network structure learning. We empirically compared our MaxSAT-based solution approach with other, previously proposed exact algorithms. For both problems, we observed that the MaxSAT-based approach was faster and more memory efficient than the other considered approaches on several benchmarks. After the publication of Papers V and VI we have compiled a set of MaxSAT benchmarks of both correlation clustering and BTBNSL to each MaxSAT evaluation organized since 2015, i.e., the 2015, 2016 and 2017 evaluations. Interestingly, the solver that was most successful on those benchmarks in each evaluation was implicit hitting set based, an observation we made already in the original publications. This seems to suggest that these benchmarks exhibit some form of specific structure which is more easily exploited implicit hitting set based solvers compared to core-guided solvers. One possible explanation is the high diversity of weights of the soft clauses in the instances, which means that core-guided solvers need to perform a significant amount of clause cloning when solving them. A deeper understanding of the similarities and differences between core-guided and implicit hitting set based solvers remains an interesting open question for developing more effective encodings and MaxSAT solvers for combinatorial optimization problems at large.

References

[1] Christos H. Papadimitriou and Kenneth Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice-Hall, Inc., 1982.

[2] Chu Min Li and Felip Manyà. MaxSAT, hard and soft constraints. In *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, chapter 19, pages 613–631. IOS Press, 2009.

[3] Richard Bellman. Dynamic programming treatment of the travelling salesman problem. *Journal of the ACM*, 9(1):61–63, 1962.

[4] Nico L.J. Ulder, Emile H.L. Aarts, Hans-Jürgen Bandelt, Peter J.M. Van Laarhoven, and Erwin Pesch. Genetic local search algorithms for the traveling salesman problem. In *Proceedings of the 1st Workshop on the Parallel Problem Solving from Nature*, volume 496 of *Lecture Notes in Computer Science*, pages 109–116. Springer, 1990.

[5] Mauricio G.C. Resende and Panos M. Pardalos. *Handbook of Optimization in Telecommunications*. Springer Science & Business Media, 2008.

[6] Zhibin Wang, Chongzhi Zang, Jeffrey A. Rosenfeld, Dustin E. Schones, Artem Barski, Suresh Cuddapah, Kairong Cui, Tae-Young Roh, Weiqun Peng, Michael Q. Zhang, and Keji Zhao. Combinatorial patterns of histone acetylations and methylations in the human genome. *Nature Genetics*, 40:897–903, 2008.

[7] David Allouche, Isabelle André, Sophie Barbe, Jessica Davies, Simon de Givry, George Katsirelos, Barry O'Sullivan, Steven David Prestwich, Thomas Schiex, and Seydou Traoré. Computational protein design as an optimization problem. *Artificial Intelligence*, 212:59–79, 2014.

[8] Thi-Bich-Hanh Dao, Khanh-Chuong Duong, and Christel Vrain. Constrained clustering by constraint programming. *Artificial Intelligence*, 244:70–94, 2017.

[9] Ian Davidson, S. S. Ravi, and Leonid Shamis. A SAT-based framework for efficient constrained clustering. In *Proceedings of the SIAM International Conference on Data Mining*, pages 94–105. SIAM, 2010.

[10] Sean Gilpin and Ian Davidson. A flexible ILP formulation for hierarchical clustering. *Artificial Intelligence*, 244:95–109, 2017.

[11] Tomi Silander, Teemu Roos, Petri Kontkanen, and Petri Myllymäki. Factorized normalized maximum likelihood criterion for learning Bayesian network structures. In *Proceedings of the 4th European Workshop on Probabilistic Graphical Models*, pages 257–272, 2008.

[12] Dag Sonntag, Matti Järvisalo, José M. Peña, and Antti Hyttinen. Learning optimal chain graphs with answer set programming. In *Proceedings of the 31st Conference on Uncertainty in Artificial Intelligence*, pages 822–831. AUAI Press, 2015.

[13] Antti Hyttinen, Paul Saikko, and Matti Järvisalo. A core-guided approach to learning optimal causal graphs. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 645–651. AAAI Press, 2017.

[14] Andreas Niskanen, Johannes Peter Wallner, and Matti Järvisalo. Optimal status enforcement in abstract argumentation. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, pages 1216–1222. IJCAI/AAAI Press, 2016.

[15] Tias Guns, Anton Dries, Siegfried Nijssen, Guido Tack, and Luc De Raedt. MiningZinc: A declarative framework for constraint-based mining. *Artificial Intelligence*, 244:6–29, 2017.

[16] Tias Guns, Siegfried Nijssen, and Luc De Raedt. K-pattern set mining under constraints. *IEEE Transactions on Knowledge and Data Engineering*, 25(2):402–418, 2013.

[17] Benjamin Négrevergne, Anton Dries, Tias Guns, and Siegfried Nijssen. Dominance programming for itemset mining. In *Proceedings of the 13th International Conference on Data Mining*, pages 557–566. IEEE Computer Society, 2013.

[18] John O. R. Aoga, Tias Guns, and Pierre Schaus. An efficient algorithm for mining frequent sequence with constraint programming. In *Machine Learning and Knowledge Discovery in Databases*, pages 315–330, Cham, 2016. Springer International Publishing.

[19] Thomas Hofmann and Joachim M. Buhmann. Pairwise data clustering by deterministic annealing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(1):1–14, 1997.

[20] Kerstin Bunte, Matti Järvisalo, Jeremias Berg, Petri Myllymäki, Jaakko Peltonen, and Samuel Kaski. Optimal neighborhood preserving visualization by maximum satisfiability. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, pages 1694–1700. AAAI Press, 2014.

[21] Sigurdur Olafsson, Xiaonan Li, and Shuning Wu. Operations research and data mining. *European Journal of Operational Research*, 187(3):1429–1448, 2008.

[22] Dana Nau, Malik Ghallab, and Paolo Traverso. *Automated Planning: Theory & Practice*. Morgan Kaufmann Publishers Inc., 2004.

[23] Jussi Rintanen. Planning with SAT, admissible heuristics and A*. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pages 2015–2020. AAAI Press, 2011.

[24] Lei Zhang and Fahiem Bacchus. MaxSAT heuristics for cost optimal planning. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence*. AAAI Press, 2012.

[25] Luis C. Rabelo and Albert Jones. Job shop scheduling. In *Encyclopedia of Operations Research and Management Science*, pages 817–830. Springer, 2013.

[26] Mirko Stojadinovic. Air traffic controller shift scheduling by reduction to CSP, SAT and SAT-related problems. In *Proceedings of the 20th International Conference on Principles and Practice of Constraint Programming*, volume 8656 of *Lecture Notes in Computer Science*, pages 886–902. Springer, 2014.

[27] Miquel Bofill, Marc Garcia, Josep Suy, and Mateu Villaret. MaxSAT-based scheduling of B2B meetings. In *Proceedings of the 12th International Conference on the Integration of AI and OR Techniques in Constraint Programming*, volume 9075 of *Lecture Notes in Computer Science*, pages 65–73, 2015.

[28] Jan K. Lenstra, A.H.G. Rinnooy Kan, and Peter Brucker. Complexity of machine scheduling problems. *Annals of Discrete Mathematics*, 1:343–362, 1977.

[29] Marius M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254–265, 1987.

[30] Derya E. Akyol and G. Mirac Bayhan. A review on evolution of production scheduling with neural networks. *Computers & Industrial Engineering*, 53(1):95 – 122, 2007.

[31] Hui Xu, Rob A. Rutenbar, and Karem A. Sakallah. Sub-SAT: a formulation for relaxed Boolean satisfiability with applications in routing. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 22(6):814–820, 2003.

[32] Samuel S. Brito, George H.G. Fonseca, Tulio A.M. Toffolo, Haroldo G. Santos, and Marcone J.F. Souza. A SA-VNS approach for the high school timetabling problem. *Electronic Notes in Discrete Mathematics*, 39:169–176, 2012.

[33] Andrea Schaerf. A survey of automated timetabling. *Artificial Intelligence Review*, 13(2):87–127, 1999.

[34] Edmund K. Burke, Barry McCollum, Amnon Meisels, Sanja Petrovic, and Rong Qu. A graph-based hyper-heuristic for educational timetabling problems. *European Journal of Operational Research*, 176(1):177–192, 2007.

[35] Alain Hertz. Tabu search for large scale timetabling problems. *European Journal of Operational Research*, 54(1):39–47, 1991.

[36] Roberto J.A. Achá and Robert Nieuwenhuis. Curriculum-based course timetabling with SAT and MaxSAT. *Annals of Operations Research*, 218(1):71–91, 2014.

[37] Manu Jose and Rupak Majumdar. Cause clue clauses: Error localization using maximum satisfiability. *ACM SIGPLAN Notices*, 46(6):437–446, 2011.

[38] Charlie Shucheng Zhu, Georg Weissenbacher, and Sharad Malik. Post-silicon fault localisation using maximum satisfiability and backbones. In *Proceedings of the 11th International Conference on Formal Methods in Computer-Aided Design*, pages 63–66. FMCAD Inc, 2011.

[39] Morten Mossige, Arnaud Gotlieb, and Hein Meling. Deploying constraint programming for testing ABB's painting robots. *AI Magazine*, 38(2):94–96, 2017.

[40] Javier Barbas and Angel Marin. Maximal covering code multiplexing access telecommunication networks. *European Journal of Operational Research*, 159(1):219 – 238, 2004.

[41] Dimitris Bertsimas, Guglielmo Lulli, and Amedeo R. Odoni. An integer optimization approach to large-scale air traffic flow management. *Operations Research*, 59(1):211–227, 2011.

[42] Arthur Richards and Jonathan P. How. Aircraft trajectory planning with collision avoidance using mixed integer linear programming. In *Proceedings of the 2002 American Control Conference*, volume 3, pages 1936–1941. IEEE, 2002.

[43] Pey-Chang Lin and Sunil Khatri. Application of MaxSAT-based ATPG to optimal cancer therapy design. *BMC Genomics*, 13(6), 2012.

[44] Tung-Wei Kuo, Kate Ching-Ju Lin, and Ming-Jer Tsai. On the construction of data aggregation tree with minimum energy cost in wireless sensor networks: NP-completeness and approximation algorithms. *IEEE Transactions on Computers*, 65(10):3109–3121, 2016.

[45] Anupam Gupta, Viswanath Nagarajan, and R. Ravi. Approximation algorithms for optimal decision trees and adaptive TSP problems. *Mathematics of Operations Research*, 42(3):876–896, 2017.

[46] Anton Milan, Seyed Hamid Rezatofighi, Ravi Garg, Anthony R. Dick, and Ian D. Reid. Data-driven approximations to NP-hard problems. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, pages 1453–1459. AAAI Press, 2017.

[47] David S. Johnson. Approximation algorithms for combinatorial problems. In *Proceedings of the 5th Annual ACM Symposium on Theory of Computing*, pages 38–49. ACM, 1973.

[48] Vijay V. Vazirani. *Approximation Algorithms*. Springer-Verlag New York, Inc., 2001.

[49] Richard E. Korf. Depth-first iterative-deepening: An optimal admissible tree search. *Artificial Intelligence*, 27:97–109, 1985.

[50] Jordan Thayer and Wheeler Ruml. Anytime heuristic search: Frameworks and algorithms. In *Proceedings of the 3rd Annual Symposium on Combinatorial Search*, pages 121–128. AAAI Press, 2010.

[51] Hisao Ishibuchi and Takashi Yamamoto. Fuzzy rule selection by multi-objective genetic local search algorithms and rule evaluation measures in data mining. *Fuzzy sets and Systems*, 141(1):59–88, 2004.

[52] Eric A. Hansen and Rong Zhou. Anytime heuristic search. *Journal of Artificial Intelligence Research*, 28:267–297, 2007.

[53] Gerhard J. Woeginger. *Exact Algorithms for NP-Hard Problems: A Survey*, pages 185–207. Springer Berlin Heidelberg, 2003.

[54] Mikko Koivisto and Kismat Sood. Exact Bayesian structure discovery in Bayesian networks. *Journal of Machine Learning Research*, 5:549–573, 2004.

[55] Fedor V. Fomin and Petteri Kaski. Exact exponential algorithms. *Communications of the ACM*, 56(3):80–88, 2013.

[56] Fedor V. Fomin and Dieter Kratsch. *Exact Exponential Algorithms*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 2010.

[57] George L. Nemhauser and Laurence A. Wolsey. *Integer and combinatorial optimization*. Wiley Interscience Series in Discrete Mathematics and Optimization. Wiley, 1988.

[58] T.C. Hu and Andrew B. Kahng. Linear and integer programming in practice. In *Linear and Integer Programming Made Easy*, pages 117–130. Springer, 2016.

[59] Francesca Rossi, Peter Van Beek, and Toby Walsh. *Handbook of Constraint Programming*. Elsevier, 2006.

[60] Ilkka Niemelä. Logic programs with stable model semantics as a constraint programming paradigm. *Annals of Mathematics and Artificial Intelligence*, 25(3-4):241–273, 1999.

[61] Piero Bonatti, Francesco Calimeri, Nicola Leone, and Francesco Ricca. Answer set programming. In *A 25-Year Perspective on Logic Programming*, pages 159–182. Springer-Verlag, 2010.

[62] Luc De Raedt, Tias Guns, and Siegfried Nijssen. Constraint programming for data mining and machine learning. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*. AAAI Press, 2010.

[63] Ronald de Haan, Martin Kronegger, and Andreas Pfandler. Fixed-parameter tractable reductions to SAT for planning. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, pages 2897–2903. AAAI Press, 2015.

[64] Edmund Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, and Hel. Counterexample-guided abstraction refinement for symbolic model checking. *Journal of the ACM*, 50(5):752–794, 2003.

[65] Mikoláš Janota, William Klieber, João Marques-Silva, and Edmund Clarke. Solving QBF with counterexample guided refinement. *Artificial Intelligence*, 234:1–25, 2016.

[66] Stephen A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing*, pages 151–158. ACM, 1971.

[67] Alexander Schrijver. *Theory of linear and integer programming*. John Wiley & Sons, 1998.

[68] James F. Campbell. Integer programming formulations of discrete hub location problems. *European Journal of Operational Research*, 72(2):387–405, 1994.

[69] Michela Milano and Francesca Rossi. Constraint programming. *Intelligenza Artificiale*, 3(1-2):28–34, 2006.

[70] Daniel Larraz, Kaustubh Nimkar, Albert Oliveras, Enric Rodríguez-Carbonell, and Albert Rubio. Proving non-termination using MaxSMT. In *Proceedings of the 26th International Conference on Computer Aided Verification*, volume 8559 of *Lecture Notes in Computer Science*, pages 779–796, 2014.

[71] Roberto Sebastiani and Patrick Trentin. On optimization modulo theories, MaxSMT and sorting networks. In *Proceedings of the 23rd International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, volume 10206 of *Lecture Notes in Computer Science*, pages 231–248. Springer, 2017.

[72] Roberto Sebastiani and Patrick Trentin. OptiMathSAT: A tool for optimization modulo theories. In *Proceedings of the 27th International Conference on Computer Aided Verification*, volume 9206 of *Lecture Notes in Computer Science*, pages 447–454. Springer, 2015.

[73] Tomas Balyo, Marijn Heule, and Matti Järvisalo. SAT Competition 2016: Recent developments. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, pages 5061–5063. AAAI Press, 2017.

[74] Matti Järvisalo, Daniel Le Berre, Olivier Roussel, and Laurent Simon. The international SAT solver competitions. *AI Magazine*, 33(1):89–92, 2012.

[75] Jeremias Berg and Matti Järvisalo. Cost-optimal constrained correlation clustering via weighted partial maximum satisfiability. *Artificial Intelligence*, 244:110–142, 2017.

[76] Jeremias Berg, Matti Järvisalo, and Brandon Malone. Learning optimal bounded treewidth Bayesian networks via maximum satisfiability. In *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics*, volume 33 of *JMLR Workshop and Conference Proceedings*, pages 86–95. JMLR, 2014.

[77] Antti Hyttinen, Patrik O. Hoyer, Frederick Eberhardt, and Matti Järvisalo. Discovering cyclic causal models with latent variables: A general SAT-based procedure. In *Proceedings of the 29th Conference on Uncertainty in Artificial Intelligence*. AUAI Press, 2013.

[78] James D. Park. Using weighted MaxSAT engines to solve MPE. In *Proceedings of the 18th National Conference on Artificial Intelligence*, pages 682–687. AAAI Press / The MIT Press, 2002.

[79] Tian Sang, Paul Beame, and Henry A. Kautz. A dynamic approach for MPE and weighted MaxSAT. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 173–179, 2007.

[80] Ana Graça, Inês Lynce, João Marques-Silva, and Arlindo L. Oliveira. Efficient and accurate haplotype inference by combining parsimony and pedigree information. In *Revised Selected Papers of the 4th International Conference on Algebraic and Numeric Biology*, volume 6479 of *Lecture Notes in Computer Science*, pages 38–56. Springer, 2012.

[81] Ana Graça, João Marques-Silva, Inês Lynce, and Arlindo L. Oliveira. Haplotype inference with pseudo-Boolean optimization. *Annals of Operations Research*, 184(1):137–162, 2011.

[82] Inês Lynce and João Marques-Silva. Haplotype inference with Boolean satisfiability. *International Journal on Artificial Intelligence Tools*, 17(2):355–387, 2008.

[83] Xiaojuan Liao, Miyuki Koshimura, Hiroshi Fujita, and Ryuzo Hasegawa. MaxSAT encoding for MC-net-based coalition structure generation problem with externalities. *IEICE Transactions*, 97-D(7):1781–1789, 2014.

[84] Jeremias Berg and Matti Järvisalo. SAT-based approaches to treewidth computation: An evaluation. In *Proceedings of of the 26th International Conference on Tools with Artificial Intelligence*, pages 328–335. IEEE Computer Society, 2014.

[85] João Guerra and Inês Lynce. Reasoning over biological networks using maximum satisfiability. In *Proceedings of the 18th International Conference on Principles and Practice of Constraint Programming*, volume 7514 of *Lecture Notes in Computer Science*, pages 941–956. Springer, 2012.

[86] Dawn M. Strickland, Earl R. Barnes, and Joel S. Sokol. Optimal protein structure alignment using maximum cliques. *Operations Research*, 53(3):389–402, 2005.

[87] Tuomas Sandholm. An algorithm for optimal winner determination in combinatorial auctions. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pages 542–547. Morgan Kaufmann, 1999.

[88] João Marques-Silva, Mikolas Janota, Alexey Ignatiev, and Antonio Morgado. Efficient model based diagnosis with maximum satisfiability. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, pages 1966–1972. AAAI Press, 2015.

[89] Alessandro Bezerra Trindade, Renato De Faria Degelo, Edilson Galvão Dos Santos Junior, Hussama Ibrahim Ismail, Helder Cruz Da Silva, and Lucas Carvalho Cordeiro. Multi-core model checking and maximum satisfiability applied to hardware-software partitioning. *International Journal of Embedded Systems*, 9(6):570–582, 2017.

[90] Yu Feng, Osbert Bastani, Ruben Martins, Isil Dillig, and Saswat Anand. Automated synthesis of semantic malware signatures using maximum satisfiability. In *Proceedings of the 24th Annual Network and Distributed System Security Symposium*. The Internet Society, 2017.

[91] Sean Safarpour, Hratch Mangassarian, Andreas G. Veneris, Mark H. Liffiton, and Karem A. Sakallah. Improved design debugging using maximum satisfiability. In *Proceedings of the 7th International Conference on Formal Methods in Computer-Aided Design*, pages 13–19. IEEE Computer Society, 2007.

[92] Xujie Si, Xin Zhang, Radu Grigore, and Mayur Naik. Maximum satisfiability in software analysis: Applications and techniques. In *Proceedings of the 29th International Conference on Computer Aided Verification*, volume 10426 of *Lecture Notes in Computer Science*, pages 68–94. Springer, 2017.

[93] Christian Muise, J. Christopher Beck, and Sheila A. McIlraith. Optimal partial-order plan relaxation via MaxSAT. *Journal of Artificial Intelligence Research*, 57:113–149, 2016.

[94] Marcel Kevin Tiepelt and Tilak Raj Singh. Finding pre-production vehicle configurations using a MaxSAT framework. In *Proceedings of the 18th International Configuration Workshop*, page 117. École des Mines d'Albi-Carmaux, 2016.

[95] Carlos Ansótegui, Idelfonso Izquierdo, Felip Manyà, and José Torres-Jiménez. A MaxSAT-based approach to constructing optimal covering arrays. In *Proceedings of the 16th International Conference of the Catalan Association for Artificial Intelligence*, volume 256 of *Frontiers in Artificial Intelligence and Applications*, pages 51–59. IOS Press, 2013.

[96] Josep Argelich, Daniel Le Berre, Inês Lynce, João Marques-Silva, and Pascal Rapicault. Solving Linux upgradeability problems using Boolean optimization. In *Proceedings of the 1st International Workshop on Logics for Component Configuration*, volume 29 of *Electronic Proceedings in Theoretical Computer Science*, pages 11–22. Open Publishing Association, 2010.

[97] Yibin Chen, Sean Safarpour, João Marques-Silva, and Andreas G. Veneris. Automated design debugging with maximum satisfiability.

IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 29(11):1804–1817, 2010.

[98] Inês Lynce and João Marques-Silva. Restoring CSP satisfiability with MaxSAT. *Fundamenta Informaticae*, 107(2-3):249–266, 2011.

[99] Xiaojuan Liao, Hui Zhang, and Miyuki Koshimura. Reconstructing AES key schedule images with SAT and MaxSAT. *IEICE Transactions on Information and Systems*, 99(1):141–150, 2016.

[100] Josep Argelich, Chu Min Li, Felip Manyà, and Jordi Planes. The first and second MaxSAT evaluations. *Journal on Satisfiability, Boolean Modeling and Computation*, 4(2-4):251–278, 2008.

[101] Josep Argelich, Chu Min Li, Felip Manyà, and Jordi Planes. MaxSAT Evaluations. `http://maxsat.ia.udl.cat/`.

[102] Carlos Ansótegui, Fahiem Bacchus, Matti Järvisalo, and Ruben Martins. MaxSAT Evaluation 2017, 2017. `http://mse17.cs.helsinki.fi/`.

[103] Antonio Morgado, Federico Heras, and João Marques Silva. Model-guided approaches for MaxSAT solving. In *Proceedings of the 25th IEEE International Conference on Tools with Artificial Intelligence*, pages 931–938. IEEE Computer Society, 2013.

[104] Miyuki Koshimura, Tong Zhang, Hiroshi Fujita, and Ryuzo Hasegawa. QMaxSAT: A partial MaxSAT solver. *Journal of Satisfiability, Boolean Modeling and Computation*, 8(1/2):95–100, 2012.

[105] Daniel Le Berre and Anne Parrain. The SAT4J library, release 2.2, system description. *Journal on Satisfiability, Boolean Modeling and Computation*, 7:59–64, 2010.

[106] Carlos Ansótegui and Joel Gabàs. WPM3: An (in) complete algorithm for weighted partial MaxSAT. *Artificial Intelligence*, 250:37–57, 2017.

[107] João Marques-Silva and Jordi Planes. Algorithms for maximum satisfiability using unsatisfiable cores. In *Proceedings of Design, Automation and Test in Europe*, pages 408–413. ACM, 2008.

[108] Ruben Martins, Vasco M. Manquinho, and Inês Lynce. Improving linear search algorithms with model-based approaches for MaxSAT solving. *Journal of Experimental & Theoretical Artificial Intelligence*, 27(5):673–701, 2015.

[109] Vasco M. Manquinho, João Marques-Silva, and Jordi Planes. Algorithms for weighted Boolean optimization. In *Proceedings of the 12th International Conference on Theory and Applications of Satisfiability Testing*, volume 5584 of *Lecture Notes in Computer Science*, pages 495–508. Springer, 2009.

[110] Federico Heras, Antonio Morgado, and João Marques-Silva. Core-guided binary search algorithms for maximum satisfiability. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence*. AAAI Press, 2011.

[111] António Morgado, Carmine Dodaro, and João Marques-Silva. Core-guided MaxSAT with soft cardinality constraints. In *Proceedings of the 20th International Conference on Principles and Practice of Constraint Programming*, volume 8656 of *Lecture Notes in Computer Science*, pages 564–573. Springer, 2014.

[112] António Morgado, Federico Heras, and João Marques-Silva. Improvements to core-guided binary search for MaxSAT. In *Proceedings of the 15th International Conference on Theory and Applications of Satisfiability Testing*, volume 7317 of *Lecture Notes in Computer Science*, pages 284–297. Springer, 2012.

[113] Ruben Martins, Saurabh Joshi, Vasco M. Manquinho, and Inês Lynce. Incremental cardinality constraints for MaxSAT. In *Proceedings of the 20th International Conference on Principles and Practice of Constraint Programming*, volume 8656 of *Lecture Notes in Computer Science*, pages 531–548. Springer, 2014.

[114] António Morgado, Federico Heras, Mark H. Liffiton, Jordi Planes, and João Marques-Silva. Iterative and core-guided MaxSAT solving: A survey and assessment. *Constraints*, 18(4):478–534, 2013.

[115] Nina Narodytska and Fahiem Bacchus. Maximum satisfiability using core-guided MaxSAT resolution. In *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, pages 2717–2723. AAAI Press, 2014.

[116] Zhaohui Fu and Sharad Malik. On solving the partial MaxSAT problem. In *Proceedings of the 9th International Conference on Theory and Applications of Satisfiability Testing*, volume 4121 of *Lecture Notes in Computer Science*, pages 252–265. Springer, 2006.

[117] Ruben Martins, Vasco M. Manquinho, and Inês Lynce. Open-WBO: A modular MaxSAT solver. In *Proceedings of the 17th International Conference on Theory and Applications of Satisfiability Testing*, volume 8561 of *Lecture Notes in Computer Science*, pages 438–445. Springer, 2014.

[118] Carlos Ansótegui, Maria Luisa Bonet, and Jordi Levy. SAT-based MaxSAT algorithms. *Artificial Intelligence*, 196:77 – 105, 2013.

[119] Mario Alviano, Carmine Dodaro, and Francesco Ricca. A MaxSAT algorithm using cardinality constraints of bounded size. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pages 2677–2683. AAAI Press, 2015.

[120] Nikolaj Bjørner and Nina Narodytska. Maximum satisfiability using cores and correction sets. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, pages 246–252. AAAI Press, 2015.

[121] Jessica Davies and Fahiem Bacchus. Exploiting the power of MIP solvers in MaxSAT. In *Proceedings of the 16th International Conference on Theory and Applications of Satisfiability Testing*, volume 7962 of *Lecture Notes in Computer Science*, pages 166–181. Springer, 2013.

[122] Jessica Davies and Fahiem Bacchus. Solving MaxSAT by solving a sequence of simpler SAT instances. In *Proceedings of the 17th International Conference on Principles and Practice of Constraint Programming*, volume 6876 of *Lecture Notes in Computer Science*, pages 225–239. Springer, 2011.

[123] Paul Saikko, Jeremias Berg, and Matti Järvisalo. LMHS: a SAT-IP hybrid MaxSAT solver. In *Proceedings of the 19th International Conference on Theory and Applications of Satisfiability Testing*, volume 9710 of *Lecture Notes in Computer Science*, pages 539–546. Springer, 2016.

[124] Brian Borchers and Judith Furman. A two-phase exact algorithm for MaxSAT and weighted MaxSAT problems. *Journal of Combinatorial Optimization*, 2(4):299–306, 1998.

[125] Chu Min Li, Felip Manya, and Jordi Planes. Exploiting unit propagation to compute lower bounds in branch and bound MaxSAT solvers. In *Proceedings of the 11th International Conference on Principles and*

Practice of Constraint Programming, volume 3709 of *Lecture Notes in Computer Science*, pages 403–414. Springer, 2005.

[126] Chu Min Li and Zhe Quan. An efficient branch-and-bound algorithm based on MaxSAT for the maximum clique problem. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, volume 10, pages 128–133. AAAI Press, 2010.

[127] André Abramé and Djamal Habet. AHMAXSAT: Description and evaluation of a branch and bound MaxSAT solver. *Journal on Satisfiability, Boolean Modeling and Computation*, 9:89–128, 2015.

[128] Yan-Li Liu, Chu-Min Li, Kun He, and Yi Fan. Breaking cycle structure to improve lower bound for MaxSAT. In *Proceedings of the 10th International Workshop on Frontiers in Algorithmics*, volume 9711 of *Lecture Notes in Computer Science*, pages 111–124. Springer, 2016.

[129] André Abramé and Djamal Habet. Learning nobetter clauses in MaxSAT branch and bound solvers. In *Proceedings of the 28th International Conference on Tools with Artificial Intelligence*, IEEE Computer Society, pages 452–459, 2016.

[130] David R. Morrison, Sheldon H. Jacobson, Jason J. Sauppe, and Edward C. Sewell. Branch-and-bound algorithms: A survey of recent advances in searching, branching, and pruning. *Discrete Optimization*, 19:79–102, 2016.

[131] Chu Min Li, Felip Manya, and Jordi Planes. New inference rules for MaxSAT. *Journal of Artificial Intelligence Research*, 30(1):321–359, 2007.

[132] Gintaras Palubeckis. A new bounding procedure and an improved exact algorithm for the Max-2-SAT problem. *Applied Mathematics and Computation*, 215(3):1106–1117, 2009.

[133] Zhao Xing and Weixiong Zhang. MaxSolver: An efficient exact algorithm for (weighted) maximum satisfiability. *Artificial intelligence*, 164(1-2):47–80, 2005.

[134] Han Lin, Kaile Su, and Chu Min Li. Within-problem learning for efficient lower bound computation in MaxSAT solving. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*, pages 351–356. AAAI Press, 2008.

[135] Anton Belov, António Morgado, and João Marques-Silva. SAT-based preprocessing for MaxSAT. In *Proceedings of the 19th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning*, volume 8312 of *Lecture Notes in Computer Science*, pages 96–111. Springer, 2013.

[136] Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation clustering. *Machine Learning*, 56(1-3):89–113, 2004.

[137] Judea Pearl. *Probabilistic reasoning in intelligent systems: networks of plausible inference*. Morgan Kaufmann Publishers Inc., 1988.

[138] Gal Elidan and Stephen Gould. Learning bounded treewidth Bayesian networks. *Journal of Machine Learning Research*, 9:2699–2731, 2008.

[139] Martin W. P. Savelsbergh. Preprocessing and probing techniques for mixed integer programming problems. *INFORMS Journal on Computing*, 6(4):445–454, 1994.

[140] Matti Järvisalo, Marijn Heule, and Armin Biere. Inprocessing rules. In *Proceedings of the 6th International Joint Conference on Automated Reasoning*, volume 7364 of *Lecture Notes in Computer Science*, pages 355–370. Springer, 2012.

[141] Niklas Eén and Armin Biere. Effective preprocessing in SAT through variable and clause elimination. In *Proceedings of the 8th International Conference on Theory and Applications of Satisfiability Testing*, volume 3569 of *Lecture Notes in Computer Science*, pages 61–75. Springer, 2005.

[142] Marijn Heule, Matti Järvisalo, and Armin Biere. Covered clause elimination. In *Short papers for the 17th International Conference on Logic for Programming Artificial Intelligence, and Reasoning*, volume 13 of *EPiC Series in Computing*, pages 41–46. EasyChair, 2013.

[143] Cédric Piette, Youssef Hamadi, and Lakhdar Saïs. Vivifying propositional clausal formulae. In *Proceedings of the 18th European Conference on Artificial Intelligence*, volume 178 of *Frontiers in Artificial Intelligence and Applications*, pages 525–529. IOS Press, 2008.

[144] Matti Järvisalo and Armin Biere. Reconstructing solutions after blocked clause elimination. In *Proceedings of the 13th International Conference on Theory and Applications of Satisfiability Testing*, volume 6175 of *Lecture Notes in Computer Science*, pages 340–345. Springer, 2010.

[145] Inês Lynce and João Marques-Silva. Probing-based preprocessing techniques for propositional satisfiability. In *Proceedings of the 15th IEEE International Conference on Tools with Artificial Intelligence*, pages 105–110. IEEE Computer Society, 2003.

[146] Fahiem Bacchus and Jonathan Winter. Effective preprocessing with hyper-resolution and equality reduction. In *Selected Revised Papers of the 6th International Conference on Theory and Applications of Satisfiability Testing*, volume 2919 of *Lecture Notes in Computer Science*, pages 341–355. Springer, 2004.

[147] Sathiamoorthy Subbarayan and Dhiraj K. Pradhan. NiVER: Non-increasing variable elimination resolution for preprocessing SAT instances. In *Online Proceedings of the 7th International Conference on Theory and Applications of Satisfiability Testing*, pages 276–291. Springer, 2004.

[148] Roman Gershman and Ofer Strichman. Cost-effective hyper-resolution for preprocessing CNF formulas. In *Proceedings of the 8th International Conference on Theory and Applications of Satisfiability Testing*, volume 3569 of *Lecture Notes in Computer Science*, pages 423–429. Springer, 2005.

[149] Hyojung Han and Fabio Somenzi. Alembic: An efficient algorithm for CNF preprocessing. In *Proceedings of the 44th annual Design Automation Conference*, pages 582–587. ACM, 2007.

[150] Marijn Heule, Matti Järvisalo, and Armin Biere. Efficient CNF simplification based on binary implication graphs. In *Proceedings of the 14th International Conference on Theory and Applications of Satisfiability Testing*, volume 6695 of *Lecture Notes in Computer Science*, pages 201–215. Springer, 2011.

[151] Maria Luisa Bonet, Jordi Levy, and Felip Manyà. Resolution for MaxSAT. *Artificial Intelligence*, 171(8-9):606–618, 2007.

[152] Javier Larrosa, Federico Heras, and Simon de Givry. A logical approach to efficient MaxSAT solving. *Artificial Intelligence*, 172(2-3):204–233, 2008.

[153] Josep Argelich, Chu Min Li, and Felip Manyà. A preprocessor for MaxSAT solvers. In *Proceedings of the 11th International Conference on Theory and Applications of Satisfiability Testing*, volume 4996 of *Lecture Notes in Computer Science*, pages 15–20. Springer, 2008.

[154] Anton Belov, Matti Järvisalo, and João Marques-Silva. Formula preprocessing in MUS extraction. In *Proceedings of the 19th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, volume 7795 of *Lecture Notes in Computer Science*, pages 108–123. Springer, 2013.

[155] Byron Ellis and Wing Hung Wong. Learning causal bayesian network structures from experimental data. *Journal of the American Statistical Association*, 103(482):778–789, 2008.

[156] Aristidis Likas, Nikos Vlassis, and Jakob J. Verbeek. The global K-means clustering algorithm. *Pattern recognition*, 36(2):451–461, 2003.

[157] Leonard J. Schulman. Clustering for edge-cost minimization. *Electronic Colloquium on Computational Complexity (ECCC)*, 6(35), 1999.

[158] Anil K. Jain, M. Narasimha Murty, and P. J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[159] Anil K. Jain and Richard C. Dubes. *Algorithms for Clustering Data*. Prentice-Hall, Inc., 1988.

[160] Andrew Y. Ng, Michael I. Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in neural information processing systems*, pages 849–856, 2002.

[161] Robert C. Edgar. Search and clustering orders of magnitude faster than blast. *Bioinformatics*, 26(19):2460–2461, 2010.

[162] Daniel Aloise, Pierre Hansen, and Leo Liberti. An improved column generation algorithm for minimum sum-of-squares clustering. *Mathematical Programming*, 131(1):195–220, 2012.

[163] Weifeng Zhi, Buyue Qian, and Ian Davidson. Scalable constrained spectral clustering via the randomized projected power method. In *Proceedings of the 2017 IEEE International Conference on Data Mining*, pages 1201–1206. IEEE Computer Society, 2017.

[164] Kamal Jain and Vijay V. Vazirani. Primal-dual approximation algorithms for metric facility location and k-median problems. In *Proceedings of the 40th Annual Symposium on Foundations of Computer Science*, pages 2–13. IEEE Computer Society, 1999.

[165] Rajkumar Jain and Narendra S. Chaudhari. Formulation of 3-clustering as a 3-SAT problem. In *Proceedings of the 5th Indian International Conference on Artificial Intelligence*, pages 465–472. IICAI, 2011.

[166] Jean-Philippe Métivier, Patrice Boizumault, Bruno Crémilleux, Mehdi Khiari, and Samir Loudni. Constrained clustering using SAT. In *Proceedings of the 11th International Conference on Advances in Intelligent Data Analysis*, volume 7619 of *Lecture Notes in Computer Science*, pages 207–218. Springer, 2012.

[167] Marianne Mueller and Stefan Kramer. Integer linear programming models for constrained clustering. In *Proceedings of the 13th International Conference on Discovery Science*, volume 6332 of *Lecture Notes in Computer Science*, pages 159–173. Springer, 2010.

[168] Nir Ailon, Moses Charikar, and Alantha Newman. Aggregating inconsistent information: Ranking and Clustering. *Journal of the ACM*, 55(5):23:1–23:27, 2008.

[169] Moses Charikar, Venkatesan Guruswami, and Anthony Wirth. Clustering with qualitative information. *Journal of Computer and System Sciences*, 71(3):360–383, 2005.

[170] Ron Shamir, Roded Sharan, and Dekel Tsur. Cluster graph modification problems. *Discrete Applied Mathematics*, 144(1-2):173–182, 2004.

[171] Ioannis Giotis and Venkatesan Guruswami. Correlation clustering with a fixed number of clusters. *Theory of Computing*, 2(1):249–266, 2006.

[172] Erik D. Demaine, Dotan Emanuel, Amos Fiat, and Nicole Immorlica. Correlation clustering in general weighted graphs. *Theoretical Computer Science*, 361(2-3):172–187, 2006.

[173] Nir Ailon and Edo Liberty. Correlation clustering revisited: The "true" cost of error minimization problems. In *Proceedings of the 36th International Colloquium on Automata, Languages and Programming*, volume 5555 of *Lecture Notes in Computer Science*, pages 24–36. Springer, 2009.

[174] Erik D. Demaine and Nicole Immorlica. Correlation clustering with partial information. In *Proceedings of the 6th International Workshop on Approximation Algorithms for Combinatorial Optimization*

Problems and 7th International Workshop on Randomization and Approximation Techniques in Computer Science, volume 2764 of *Lecture Notes in Computer Science*, pages 1–13. Springer, 2003.

[175] Jurgen Van Gael and Xiaojin Zhu. Correlation clustering for crosslingual link detection. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, pages 1744–1749. AAAI Press, 2007.

[176] Adnan Darwiche. Chapter 11 Bayesian networks. In *Handbook of Knowledge Representation*, volume 3 of *Foundations of Artificial Intelligence*, pages 467 – 509. Elsevier, 2008.

[177] Sebastian Ordyniak and Stefan Szeider. Parameterized complexity results for exact Bayesian network structure learning. *Journal of Artificial Intelligence Research*, 46:263–302, 2013.

[178] Cassio P. de Campos and Qiang Ji. Efficient learning of Bayesian networks using constraints. *Journal of Machine Learning Research*, 12:663–689, 2011.

[179] Luis M. de Campos. A scoring function for learning Bayesian networks based on mutual information and conditional independence tests. *Journal of Machine Learning Research*, 7:2149–2187, 2006.

[180] Mark Bartlett and James Cussens. Integer linear programming for the Bayesian network structure learning problem. *Artificial Intelligence*, 244:258–271, 2017.

[181] David M. Chickering. Learning Bayesian networks is NP-complete. In *Learning from Data: Artificial Intelligence and Statistics V*, pages 121–130. Springer-Verlag, 1996.

[182] Nir Friedman and Daphne Koller. Being Bayesian about network structure. A Bayesian approach to structure discovery in Bayesian networks. *Machine Learning*, 50:95–125, 2003.

[183] David M. Chickering. Learning equivalence classes of Bayesian-network structures. *Journal of Machine Learning Research*, 2:445–498, 2002.

[184] Tommi Jaakkola, David Sontag, Amir Globerson, and Marina Meila. Learning Bayesian network structure using LP relaxations. In *Proceedings of the 13th International Conference on Artificial Intelli-*

gence and Statistics, volume 9 of *JMLR Proceedings*, pages 358–365. JMLR, 2010.

[185] James Cussens. Bayesian network learning by compiling to weighted MaxSAT. In *Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence*, pages 105–112. AUAI Press, 2008.

[186] Alexandra M. Carvalho, Teemu Roos, Arlindo L. Oliveira, and Petri Myllymäki. Discriminative learning of Bayesian networks via factorized conditional log-likelihood. *Journal of Machine Learning Research*, 12:2181–2210, July 2011.

[187] Daniel Eaton and Kevin Murphy. Bayesian structure learning using dynamic programming and MCMC. In *Proceedings of the 23rd Conference on Uncertainty in Artificial Intelligence*, pages 101–108. AUAI Press, 2007.

[188] Sascha Ott and Satoru Miyano. Finding optimal gene networks using biological constraints. *Genome Informatics*, 14:124–133, 2003.

[189] Tomi Silander and Petri Myllymäki. A simple approach for finding the globally optimal Bayesian network structure. In *Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence*, pages 445–452. AUAI Press, 2006.

[190] James Cussens. Bayesian network learning with cutting planes. In *Proceedings of the 27th Conference on Uncertainty in Artificial Intelligence*, pages 153–160. AUAI Press, 2011.

[191] Changhe Yuan and Brandon Malone. Learning optimal Bayesian networks: A shortest path perspective. *Journal of Artificial Intelligence Research*, 48:23–65, 2013.

[192] Gregory F. Cooper. The computational complexity of probabilistic inference using Bayesian belief networks. *Artificial Intelligence*, 42(2-3):393 – 405, 1990.

[193] Steffen L. Lauritzen and David J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. In Glenn Shafer and Judea Pearl, editors, *Readings in Uncertain Reasoning*, pages 415–448. Morgan Kaufmann Publishers Inc., 1990.

[194] Johan Kwisthout, Hans L. Bodlaender, and Linda C. van der Gaag. The necessity of bounded treewidth for efficient inference in Bayesian networks. In *Proceedings of the 19th European Conference on Artificial Intelligence*, volume 215 of *Frontiers in Artificial Intelligence and Applications*, pages 237–242. IOS Press, 2010.

[195] Neil Robertson and Paul D. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *Journal of Algorithms*, 7(3):309–322, 1986.

[196] Hans L. Bodlaender. A partial k-arboretum of graphs with bounded treewidth. *Theoretical Computer Science*, 209(1):1 – 45, 1998.

[197] Umberto Bertele and Francesco Brioschi. *Nonserial Dynamic Programming*. Academic Press, Inc., Orlando, FL, USA, 1972.

[198] Janne H. Korhonen and Pekka Parviainen. Exact learning of bounded tree-width Bayesian networks. In *Proceedings of the 16th International Conference on Artificial Intelligence and Statistics*, volume 31 of *JMLR Workshop and Conference Proceedings*, pages 370—378. JMLR, 2013.

[199] Pekka Parviainen, Hossein Shahrabi Farahani, and Jens Lagergren. Learning bounded tree-width Bayesian networks using integer linear programming. In *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics*, volume 33 of *JMLR Workshop and Conference Proceedings*, pages 751–759. JMLR, 2014.

[200] Siqi Nie, Denis D. Mauá, Cassio P. De Campos, and Qiang Ji. Advances in learning Bayesian networks of bounded treewidth. In *Advances in Neural Information Processing Systems*, pages 2285–2293, 2014.

[201] Mukund Narasimhan and Jeff Bilmes. PAC-learning bounded tree-width graphical models. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence*, pages 410–417. AUAI Press, 2004.

[202] Hans L. Bodlaender. Discovering treewidth. In *Proceedings of the 31st Conference on Current Trends in Theory and Practice of Computer Science*, volume 3381 of *Lecture Notes in Computer Science*, pages 1–16. Springer, 2005.

[203] Niklas Eén and Niklas Sörensson. Temporal induction by incremental SAT solving. *Electronic Notes in Theoretical Computer Science*, 89(4):543–560, 2003.

[204] João Marques-Silva and Karem A. Sakallah. GRASP - a new search algorithm for satisfiability. In *Proceedings of the 1996 IEEE/ACM International Conference on Computer-Aided Design*, pages 220–227. IEEE Computer Society, 1996.

[205] Frank Hutter, Marius Lindauer, Adrian Balint, Sam Bayless, Holger Hoos, and Kevin Leyton-Brown. The configurable SAT solver challenge (CSSC). *Artificial Intelligence*, 243:1–25, 2017.

[206] Forrest Sheng Bao, Chris Gutierrez, Jeriah Jn Charles-Blount, Yaowei Yan, and Yuanlin Zhang. Accelerating Boolean satisfiability (SAT) solving by common subclause elimination. *Artificial Intelligence Review*, pages 1–15, 2017.

[207] Niklas Eén and Niklas Sörensson. An extensible SAT-solver. In *Proceedings of the 6th International Conference on Theory and Applications of Satisfiability Testing*, volume 2919 of *Lecture Notes in Computer Science*, pages 502–518. Springer, 2003.

[208] Gilles Audemard and Laurent Simon. Predicting learnt clauses quality in modern SAT solvers. In *Proceedings of the 21st International Joint Conference on Artifical Intelligence*, pages 399–404. Morgan Kaufmann Publishers Inc., 2009.

[209] Armin Biere. Lingeling, Plingeling and Treengeling entering the SAT competition 2013. In *Proceedings of SAT Competition*, volume B-2013-1 of *Department of Computer Science Series of Publications B*, pages 51–52. University of Helsinki, 2013.

[210] Gilles Audemard and Laurent Simon. Glucose in the SAT 2014 competition. *SAT COMPETITION 2014*, page 31, 2014.

[211] Lintao Zhang, Conor F. Madigan, Matthew H. Moskewicz, and Sharad Malik. Efficient conflict-driven learning in a Boolean satisfiability solver. In *Proceedings of the 2001 IEEE/ACM International Conference on Computer-Aided Design*, pages 279–285. IEEE Computer Society, 2001.

[212] João Marques-Silva, Inês Lynce, and Sharad Malik. Conflict-driven clause learning SAT solvers. In *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, chapter 4, pages 131–153. IOS Press, 2009.

[213] Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5(7):394–397, 1962.

[214] Raymond Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32(1):57–95, 1987.

[215] Carsten Sinz. Towards an optimal CNF encoding of boolean cardinality constraints. In *Proceedings of the 11th International Conference on Principles and Practice of Constraint Programming*, volume 3709 of *Lecture Notes in Computer Science*, pages 827–831. Springer, 2005.

[216] Roberto Asín, Robert Nieuwenhuis, Albert Oliveras, and Enric Rodríguez-Carbonell. Cardinality networks: a theoretical and empirical study. *Constraints*, 16(2):195–221, 2011.

[217] Olivier Bailleux and Yacine Boufkhad. Efficient CNF encoding of Boolean cardinality constraints. In *Proceedings of the 9th International Conference on Principles and Practice of Constraint Programming*, volume 2833 of *Lecture Notes in Computer Science*, pages 108–122. Springer Berlin Heidelberg, 2003.

[218] Soukaina Hattad, Saïd Jabbour, Lakhdar Sais, and Yakoub Salhi. Enhancing pigeon-hole based encoding of Boolean cardinality constraints. In *Proceedings of the 9th International Conference on Agents and Artificial Intelligence*, volume 2, pages 299–307. SciTePress, 2017.

[219] Toru Ogawa, Yangyang Liu, Ryuzo Hasegawa, Miyuki Koshimura, and Hiroshi Fujita. Modulo based CNF encoding of cardinality constraints and its application to MaxSAT solvers. In *Proceedings of the 25th IEEE International Conference on Tools with Artificial Intelligence*, pages 9–17. IEEE Computer Society, 2013.

[220] Ignasi Abío, Valentin Mayer-Eichberger, and Peter J. Stuckey. Encoding linear constraints with implication chains to CNF. In *Proceedings of the 21st International Conference on the Principles and Practice of Constraint Programming*, volume 9255 of *Lecture Notes in Computer Science*, pages 3–11. Springer, 2015.

[221] Carlos Ansótegui, Maria Luisa Bonet, and Jordi Levy. Solving (weighted) partial MaxSAT through satisfiability testing. In *Proceedings of the 12th International Conference on Theory and Applications of Satisfiability Testing*, volume 5584 of *Lecture Notes in Computer Science*, pages 427–440. Springer, 2009.

[222] Carlos Ansótegui, Maria Luisa Bonet, and Jordi Levy. A new algorithm for weighted partial MaxSAT. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence*. AAAI Press, 2010.

[223] Benjamin Andres, Benjamin Kaufmann, Oliver Matheis, and Torsten Schaub. Unsatisfiability-based optimization in clasp. In *Technical Communications of the 28th International Conference on Logic Programming*, LIPIcs, pages 211–221. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2012.

[224] Carlos Ansótegui, Maria Luisa Bonet, Joel Gabàs, and Jordi Levy. Improving SAT-based weighted MaxSAT solvers. In *Proceedings of the 18th International Conference on Principles and Practice of Constraint Programming*, volume 7514 of *Lecture Notes in Computer Science*, pages 86–101. Springer, 2012.

[225] Jeremias Berg and Matti Järvisalo. Weight-aware core extraction in SAT-based MaxSAT solving. In *Proceedings of the 23rd International Conference on Principles and Practice of Constraint Programming*, volume 10416 of *Lecture Notes in Computer Science*, pages 652–670. Springer, 2017.

[226] Jessica Davies and Fahiem Bacchus. Postponing optimization to speed up MaxSAT solving. In *Proceedings of the 19th International Conference on Principles and Practice of Constraint Programming*, volume 8124 of *Lecture Notes in Computer Science*, pages 247–262. Springer, 2013.

[227] Fahiem Bacchus, Antti Hyttinen, Matti Järvisalo, and Paul Saikko. Reduced cost fixing in MaxSAT. In *Proceedings of the 23rd International Conference on Principles and Practice of Constraint Programming*, volume 10416 of *Lecture Notes in Computer Science*, pages 641–651. Springer, 2017.

[228] Federico Heras, Javier Larrosa, and Albert Oliveras. MiniMaxSAT: An efficient weighted MaxSAT solver. *Journal of Artificial Intelligence Research*, 31:1–32, 2008.

[229] Rolf Niedermeier and Peter Rossmanith. New upper bounds for maximum satisfiability. *Journal of Algorithms*, 36(1):63–88, 2000.

[230] Anton Belov and João Marques-Silva. Generalizing redundancy in propositional logic: Foundations and hitting sets duality. *CoRR*, abs/1207.1257, 2012.

[231] Matti Järvisalo, Armin Biere, and Marijn Heule. Blocked clause elimination. In *Proceedings of the 16th International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, volume 6015 of *Lecture Notes in Computer Science*, pages 129–144. Springer, 2010.

[232] Federico Heras, Antonio Morgado, and João Marques Silva. MaxSAT-based encodings for Group MaxSAT. *AI Communications*, 28(2):195–214, 2015.

[233] Fahiem Bacchus and Nina Narodytska. Cores in core based MaxSAT algorithms: An analysis. In *Proceedings of the 17th International Conference on Theory and Applications of Satisfiability Testing*, volume 8561 of *Lecture Notes in Computer Science*, pages 7–15. Springer, 2014.

[234] Jessica Davies. *Solving MaxSAT by Decoupling Optimization and Satisfaction.* PhD thesis, University of Toronto, 2013.

[235] Alexey Ignatiev, Antonio Morgado, Vasco Manquinho, Ines Lynce, and João Marques-Silva. Progression in maximum satisfiability. In *Proceedings of the 21st European Conference on Artificial Intelligence*, volume 263 of *Frontiers in Artificial Intelligence and Applications*, pages 453–458. IOS Press, 2014.

[236] Olivier Coudert and Jean Christophe Madre. New ideas for solving covering problems. In *Proceedings of the 32st Conference on Design Automation*, pages 641–646. ACM Press, 1995.

[237] Tuukka Korhonen, Jeremias Berg, Paul Saikko, and Matti Järvisalo. MaxPre: An extended MaxSAT preprocessor. In *Proceedings of the 20th International Conference on Theory and Applications of Satisfiability Testing*, volume 10491 of *Lecture Notes in Computer Science*, pages 449–456. Springer, 2017.

[238] Grigorii S. Tseitin. On the complexity of derivation in propositional calculus. In *Automation of Reasoning: 2: Classical Papers on Computational Logic 1967–1970*, pages 466–483. Springer Berlin Heidelberg, 1983.

[239] Nikhil Bansal, Avrim Blum, and Shuchi Chawla. Correlation clustering. In *Proceedings of the 43rd Symposium on Foundations of Computer Science*, page 238. IEEE Computer Society, 2002.

[240] Divya Pandove, Rinkle Rani, and Shivani Goel. Local graph based correlation clustering. *Knowledge-Based Systems*, 138:155–175, 2017.

[241] Yixin Zhuang, Hang Dou, Nathan Carr, and Tao Ju. Feature-aligned segmentation using correlation clustering. *Computational Visual Media*, 3(2):147–160, 2017.

[242] Nate Veldt, Anthony Ian Wirth, and David F. Gleich. Correlation clustering with low-rank matrices. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1025–1034. ACM, 2017.

[243] Evgeny Levinkov, Alexander Kirillov, and Bjoern Andres. A comparative study of local search algorithms for correlation clustering. In *Proceedings of the 39th German Conference on Pattern Recognition*, volume 10496 of *Lecture Notes in Computer Science*, pages 103–114. Springer, 2017.

[244] Atsushi Miyauchi and Tomohiro Sonobeand Noriyoshi Sukegawa. Exact clustering via integer programming and maximum satisfiability. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, pages ??? – ??? AAAI Press, 2018. To appear.

[245] Kiri Wagstaff and Claire Cardie. Clustering with instance-level constraints. In *Proceedings of the 17th International Conference on Artificial Intelligence*, pages 1103–1110. AAAI Press / The MIT Press, 2000.

[246] Martin Grötschel and Yoshiko Wakabayashi. A cutting plane algorithm for a clustering problem. *Mathematical Programming*, 45(1):59–96, 1989.

[247] Paola Bonizzoni, Gianluca Della Vedova, Riccardo Dondi, and Tao Jiang. On the approximation of correlation clustering and consensus clustering. *Journal of Computer and System Sciences*, 74(5):671–696, 2008.

[248] Tamás Nepusz, Rajkumar Sasidharan, and Alberto Paccanaro. SCPS: a fast implementation of a spectral method for detecting protein families on a genome-wide scale. *BMC Bioinformatics*, 11:120, 2010.

[249] A. Frank and A. Asuncion. UCI machine learning repository, 2010.

[250] Stephen F. Altschul, Warren Gish, Webb Miller, Eugene W. Myers, and David J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, 1990.

[251] Carlos Ansótegui and Joel Gabàs. Solving (weighted) partial MaxSAT with ILP. In *Proceedings of the 10th International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, volume 7874 of *Lecture Notes in Computer Science*, pages 403–409. Springer, 2013.

[252] Moses Charikar and Anthony Wirth. Maximizing quadratic programs: Extending grothendieck's inequality. In *Proceedings of the 45th Symposium on Foundations of Computer Science*, pages 54–60. IEEE Computer Society, 2004.

[253] Jos F. Sturm. Using SeDuMi 1.02, a MATLAB toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11–12:625–653, 1999. Version 1.05 available from `http://fewcal.kub.nl/sturm`.

[254] Tobias Achterberg, Timo Berthold, Thorsten Koch, and Kati Wolter. Constraint integer programming: A new approach to integrate CP and MIP. In *Proceedings of the 5th International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, volume 5015 of *Lecture Notes in Computer Science*, pages 6–20. Springer, 2008.

[255] Siqi Nie, Cassio P. De Campos, and Qiang Ji. Learning bounded treewidth Bayesian networks via sampling. In *European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty*, pages 387–396. Springer, 2015.

[256] Mauro Scanagatta, Giorgio Corani, Cassio P. de Campos, and Marco Zaffalon. Learning treewidth-bounded Bayesian networks with thousands of variables. In *Advances in Neural Information Processing Systems*, pages 1462–1470, 2016.

[257] Siqi Nie, Cassio P. de Campos, and Qiang Ji. Efficient learning of Bayesian networks with bounded tree-width. *International Journal of Approximate Reasoning*, 80:412–427, 2017.

[258] Wai Lam and Fahiem Bacchus. Learning Bayesian belief networks: An approach based on the MDL principle. *Computational Intelligence*, 10:269–293, 1994.

[259] Gregory F. Cooper and Edward Herskovits. A Bayesian method for the induction of probabilistic networks from data. *Machine Learning*, 9:309–347, 1992.

[260] Rina Dechter. Bucket elimination: A unifying framework for reasoning. *Artificial Intelligence*, 113(1-2):41–85, 1999.

[261] Marko Samer and Helmut Veith. Encoding treewidth into SAT. In *Proceedings of the 12th International Conference on Theory and Applications of Satisfiability Testing*, volume 5584 of *Lecture Notes in Computer Science*, pages 45–50. Springer, 2009.

[262] Peter J. Stuckey. Lazy clause generation: Combining the power of SAT and CP (and MIP?) solving. In *Proceedings of the 7th International Conference on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, volume 6140 of *Lecture Notes in Computer Science*, pages 5–9. Springer, 2010.

[263] Broes de Cat, Marc Denecker, Maurice Bruynooghe, and Peter J. Stuckey. Lazy model expansion: Interleaving grounding with search. *Journal of Artificial Intelligence Research*, 52:235–286, 2015.