Applications of MaxSAT in Data Analysis

Jeremias Berg, Antti Hyttinen and Matti Järvisalo

HIIT, Department of Computer Science, University of Helsinki, Finland

Abstract

We highlight important real-world optimization problems arising from data analysis and machine learning, representing somewhat atypical applications of SAT-based solver technology. To address the problem of current lack of heterogeneity in benchmark sets available for evaluating MaxSAT solvers, we provide a benchmark library of MaxSAT instances encoding different data analysis and machine learning problems. By doing so, we also advocate extending MaxSAT solvers to support real-valued weights for soft clauses via the presented problem domains in which the costs are naturally real-valued.

1 Introduction

Due to recent advances in maximum satisfiability (MaxSAT) solving techniques, MaxSAT, and especially its weighted partial generalization, are being used for solving a widening range of optimization problems. Having good benchmark problem sets is a critical requirement for modern algorithmic research, including MaxSAT solver development. Constructing large, meaningful benchmark sets is a time-consuming task that requires care. Indeed, there is an acknowledged shortage of good benchmark problem sets for weighted partial MaxSAT, which can be observed by looking at the weighted partial MaxSAT benchmark sets used in the recent MaxSAT Evaluations [4] (see `http://maxsat.ia.udl.cat/`).

In this paper, we give an overview of three important optimization problems arising from data analysis and machine learning, and how we have recently approached them via MaxSAT. As a contribution to the community at large, our emphasis is on providing the (Max)SAT community novel types of real-world benchmarks.

The data analysis problem domains described are:

- Cost-optimal correlation clustering (Section 3),
- Learning optimal bounded-treewidth Bayesian network structures (Section 4),
- Causal structure learning (Section 5).

Each of these problems represents a somewhat "non-standard" application domain for SAT-based techniques. However, our recent work [6, 7, 8, 26, 25]—published mainly *outside the SAT/constraints community*—on applying MaxSAT solvers to these problems shows that MaxSAT is a very reasonable current alternative to approaching these challenging problems:

- MaxSAT allows for learning noticeably better correlation clusterings than more traditional algorithms (w.r.t. costs of clusterings) and allows for constrained correlation clustering (which is novel in itself) [6, 7].
- MaxSAT is currently the most efficient way to learning optimal bounded-treewidth Bayesian networks [8]. This includes comparisons with e.g. competing state-of-the-art integer programming (IP) based approaches for the problem.

- MaxSAT provides the currently most general exact approach to learning optimal causal structures [26, 25].

The motivations for this paper are three-fold:

1. **Highlighting potential of and challenges for Boolean optimization in data analysis.** We hope to draw more attention of the SAT community to develop novel SAT-based approaches to hard search and optimization problems arising from data analysis, which due to the well-observed "big data revolution" is gaining increasing importance. The MaxSAT encodings for the problem domains described in this paper have already been successful in providing novel approaches for the particular problems. Furthermore, despite these recent advances, we believe the SAT community could make further progress in the state-of-the-art algorithmic solutions for such data analysis problems, particularly by developing novel techniques (such as incremental partial grounding) for dealing with data-oriented problems under very large amounts of data.

2. **Novel real-world benchmarks for MaxSAT solver developers and evaluations.** As—we believe—MaxSAT solver developers and MaxSAT Evaluation organizers and participants would agree, to an extent there is currently a lack of heterogenous benchmark sets for real-world applications of MaxSAT. Furthermore, optimization problems arising from data analysis are based on underlying real-world data, and the number of openly available datasets is bound to only increase, yielding opportunities for generating high numbers of interesting MaxSAT benchmarks from such domains. With all this in mind, we have made available a large set of MaxSAT instances encoding problem instances from the presented problem domain at

   `http://cs.helsinki.fi/group/coreo/benchmarks/`

   for the use of the SAT community and e.g. the MaxSAT Evaluations. The web page provides readmes for each of the benchmark sets, with specifics e.g. on the way the weights for the individual MaxSAT instances were obtained, the datasets based on which the instances were made, as well as explanations of the various parameterizations and the associated naming conventions used for the instance files. Furthermore, we provide both real-weighted (using the original weights) and integer-weighted (obtained using from the original real-values using multiplication and rounding) WCNF DIMACS instance files for the use of the community at large. We have also submitted the benchmark sets (with integer weights) to the 2015 MaxSAT Evaluation. We will work on expanding this benchmark collection further in the future.

3. **Advocating extending MaxSAT solvers to support real-valued weights.** Rather than expecting the end-user of MaxSAT solvers to multiply the weights into integral values (hopefully keeping as much of the real-valued resolution as possible), we note that, at least in principle, a majority of modern MaxSAT algorithms, including the classical Fu–Malik [20] algorithm as well as a variety of recently proposed algorithms such as [2, 32, 33, 31], could allow floating-point representations of weights on soft clauses as input. This could be implemented—depending on the algorithm in question—either by using floats as the internal representation type, or—in order to avoid potential numerical instability issues—by simply internally converting the input weights into integers using as high precision as available. Indeed, we believe it would be beneficial to extend the standard DIMACS WCNF input format to accept floating-point representations of

weights, stepping forward from the more traditional but in many case unnecessary and (from the end-user perspective) a restrictive requirement of integer-weighted soft clauses.[1] To corroborate this view, in each of the problem domains described in this paper the use of real-valued weights is a natural choice. More generally, we believe that by directly supporting real-valued weights, MaxSAT could become a more attractive choice as a Boolean optimization paradigm for a wider base of end-users[2]. Note that, while floats can be converted using a simple script with high precision to integer weights by multiplying by appropriate constants, here we want to emphasize the *user's perspective*: there is no reason *not* to support floating-point representations of weights, since in many problems domains the actual weights are indeed real-valued. The benchmarks we have made available showcase such domains.

The rest of the paper is organized as follows. After a short recap of MaxSAT (Section 2), we explain each of the considered data analysis problems one by one in Sections 3–5), with overviews on the MaxSAT encodings on which the provided benchmarks are based on, and with details on the filenaming conventions used for presenting the underlying parameters of the benchmarks. We then conclude with some lessons learned from working on the MaxSAT-based approaches to these problems.

2 Maximum Satisfiability

For a Boolean variable x, there are two literals, x and $\neg x$. A clause is a disjunction ($\vee$, logical OR) of literals. A truth assignment is a function from Boolean variables to $\{0, 1\}$. A clause C is satisfied by a truth assignment τ ($\tau(C) = 1$) if $\tau(x) = 1$ for a literal x in C, or $\tau(x) = 0$ for a literal $\neg x$ in C. A set F of clauses is satisfiable if there is an assignment τ satisfying all clauses in F ($\tau(F) = 1$), and unsatisfiable ($\tau(F) = 0$ for every assignment τ) otherwise. An instance $F = (F_h, F_s, c)$ of the *weighted partial MaxSAT* problem consists of two sets of clauses, a set F_h of *hard* clauses and a set F_s of *soft* clauses, and a function $c : F_s \to \mathbb{R}^+$ that associates a non-negative cost with each of the soft clauses. Note here that the weight function above allows for *assigning non-negative real values* from $\mathbb{R}^+$ as weights, as opposed to the more traditional definition $c : F_s \to \mathbb{N}$ restricting to integer valued weights.

Any truth assignment τ that satisfies F_h is a *solution* to F. The *cost* of a solution τ to F is

$$\textsc{cost}(F, \tau) = \sum_{\substack{C \in F_s: \\ \tau(C)=0}} c(C),$$

i.e., the sum of the costs of the soft clauses not satisfied by τ. A solution τ is (globally) *optimal* for F if $\textsc{cost}(F, \tau) \leq \textsc{cost}(F, \tau')$ holds for any solution τ' to F. Given an instance F, the weighted partial MaxSAT problem asks to find an optimal solution to F. We will refer to weighted partial MaxSAT instances simply as MaxSAT instances.

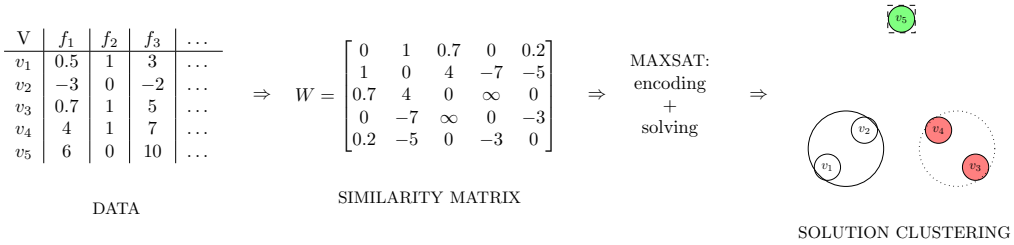

Figure 1: The correlation clustering problem by example

3 Cost-Optimal Correlation Clustering

Correlation clustering is a well-studied [1, 14, 41, 22, 19, 7, 6] NP-hard clustering problem that finds important applications in various domains such as biosciences [5], and social network analysis and information retrieval [12, 11, 13].

Figure 1 outlines the correlation clustering task via an example. To begin with, we are given a dataset consisting of N data points $V = \{v_1, v_2, \ldots, v_N\}$ with some features $f_1, f_2, \ldots, f_n$. Figure 1 (left) gives an example datasets. From the data, first a symmetric *similarity matrix* $W^{N \times N}$ is computed; see Figure 1 (middle) for an example similarity matrix. The similarity matrix W can equivalently be seen as a pairwise similarity function over V, i.e., the value at row i column j, denoted by $W(i, j)$, indicates whether the data points v_i and v_j are similar ($W(i, j) > 0$) or dissimilar ($W(i, j) < 0$). The absolute value $|W(i, j)|$ represents confidence in the (dis)similarity. The special case $W(i, j) = 0$ is used for representing *missing information* of the (dis)similarity between v_i and v_j in case of incomplete data. This definition of similarity matrices is thus rather general, and includes often-studied special cases, such as assuming complete similarity information with similarity values restricted to $\{-1, 1\}$.

The objective of the arising combinatorial optimization problem of correlation clustering is to partition (cluster) the set V in a way that correlates as well as possible with V, i.e., minimizing the number of similar pairs of points assigned to different clusters (partition) and the number of dissimilar pair of points assigned to the same cluster; an example clustering is shown in Figure 1 (right). An important contrast between correlation clustering and other clustering paradigms is that the number of clusters is not assumed as input. This makes correlation clustering especially well-suited for datasets for which the true number of clusters is unknown, which is in fact often the case when dealing with real-world data.

3.1 Problem Definition

A more precise formulation of the correlation clustering problem is as follows. Given a symmetric similarity matrix W, the task is to find a cost-optimal clustering, i.e., a function $cl^* \colon V \to \mathbb{N}$

[1]The classical definition of MaxSAT with integer weights, to our best understanding, arises from the fact that before the developments of practical MaxSAT solvers, the problem was mainly of interest from the perspective of computational complexity analysis.

[2]We acknowledge that supporting floats does not in itself solve all problems related to usability of MaxSAT from the end-users' perspective; another challenge is to lower the entry barrier to modelling with MaxSAT by e.g. providing more sophisticated modelling tools.

minimizing the correlation clustering cost function:

$$cl^* \in \arg\min_{cl\colon V\to\mathbb{N}} \sum_{\substack{cl(v_i)=cl(v_j)\\ i<j}} (\mathcal{I}[-\infty < W(i,j) < 0] \cdot |W(i,j)|) + \sum_{\substack{cl(v_i)\neq cl(v_j)\\ i<j}} (\mathcal{I}[\infty > W(i,j) > 0] \cdot W(i,j))$$

where the indicator function $\mathcal{I}[b] = 1$ iff the condition b is true. Figure 1 (right) illustrates an example solution of the correlation clustering problem. The cost of this (optimal) solution is 4.9.

3.2 Computing Weights

In our MaxSAT-based approach to correlation clustering, the input to the MaxSAT encoding is the similarity matrix W. The exact method in which W is computed depends on the particular application at hand. A simple example is to assume that all data points are in the n-dimensional Euclidean space, i.e., $v_i \in \mathbb{R}^n$ for all i. In this setting, the Euclidean distances between pairs of data points can be transformed into a similarity matrix. More precisely, given the pairwise Euclidean distances $d(v_i, v_j)$ and some (hand-picked) threshold α, let

$$W(i,j) = \begin{cases} d(i,j) - \alpha & \text{if } d(i,j) > \alpha \\ -(\alpha - d(i,j)) & \text{else.} \end{cases} \tag{1}$$

This is a natural (although by no means the only possible) similarity measure. Our MaxSAT encoding is of course general in the sense that we do not make any assumptions on the way the values of the similarity matrix have been computed.

3.3 MaxSAT Encoding

Next we give a short overview of the encoding of correlation clustering to MaxSAT. In [6, 7] we proposed three different encodings. Here we give an overview of the so-called *binary encoding* of [7] as it was the best performing one in our experiments. We refer the reader to the references for details on the two other, the so-called *unary* and *transitive* encodings.

Given a set of data points $V = \{v_1, \ldots, v_N\}$ and a similarity matrix W over V, the binary encoding includes $\log N$ variables $b_i^1, \ldots, b_i^{\log N}$ for each data point v_i. The semantics of these variables is that the value of $b_i^{\log N} \ldots b_i^1$ when interpreted as a binary number (least significant bit to the right) indicates which cluster the data point v_i should be assigned to by the solution clustering. Using these variables, the constraint requiring that two data points v_i and v_j should (not) be assigned to the same cluster is equivalent to requiring that the value of b_i^k and b_j^k should (not) be the same for all k. This is encoded by defining auxiliary variables S_{ij} using the hard constraints $S_{ij} \leftrightarrow \bigwedge_{k=1}^{\log N} (b_i^k \leftrightarrow b_j^k)$ and including the soft unit clause S_{ij} $(\neg S_{ij})$ with weight $|W(i,j)|$ for all (dis)similar pairs of points v_i and v_j. In total, an instance formed by the binary encoding contains $\mathcal{O}(E + N \cdot \log_2 N)$ variables and $\mathcal{O}(E \cdot \log_2 N)$ clauses, where $E \leq \mathcal{O}(N^2)$ is the number of non-zero values in the input similarity matrix.

3.4 Available Benchmarks

We generated the correlation clustering benchmarks based on nine real world datasets. The first four are protein datasets containing pre-computed similarity values between amino acid sequences of proteins. The protein data was obtained from `http://www.paccanarolab.org/scps`. The pre-computed similarity values for these datasets were in the range $[0, 1]$. In order to fit them into our clustering setting we subtracted -0.5 from all values. The four protein datasets contain 669, 586, 567, and 654 data points, respectively.

The benchmarks also include five other datasets:

ORL, the AT&T ORL database of images of faces, obtained from `http://www.cl.cam.ac.uk/research/dtg/attarchive/facedatabase.html`. The dataset contains 400 data points.

Ionosphere, the UCI ionosphere dataset for classification of radar returns from the ionosphere, obtained from `http://archive.ics.uci.edu/ml/`. The dataset contains 351 data points.

Breastcancer, the LIBSVM breast-cancer dataset, obtained from `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`. The dataset contains 683 data points.

Ecoli, the UCI Ecoli dataset, containing protein localization sites, Obtained from `http://archive.ics.uci.edu/ml/`. The dataset contains 327 data points.

Vowel, The LIBSVM Vowel dataset, originally from UCI, with 10 features. Obtained from `http://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`. The dataset contains 990 data points.

For these datasets we created similarity matrices in the way already described. First we calculated the normalized Euclidean distances between pairs of data points, and then created the similarity matrix W as detailed in Equation 1 with $\alpha = 0.5$. Finally, in order to simulate sparsity of the data, we pruned all similarity values in the range $[-0.2, 0.2]$. This step was omitted for the protein data as it was incomplete to begin with.

In the benchmark set we varied the number N of data points picked from each dataset in order to create MaxSAT instances with a runtime between 5 seconds and 2 hours, always using the N first data points from each dataset. The parameter values and filename convention used for the benchmarks are detailed in Table 1.

Table 1: Overview of the benchmarks encoding correlation clustering

Filename:	<*r*_>CorrelationClustering_*dataset*_*enc*_N*n*_D*d*.wcnf	
Parameter	**Description**	**Range**
r	Indicates that the weights of the soft clauses are rounded to integers	"Rounded" "
dataset	Name of the dataset the instance is based on	
enc	Name of the encoding used	UNARY TRANSITIVE BINARY
n	Number of datapoints used from the dataset	$n \in [150, 400]$
d	Every similarity value $W(i, j)$ for which $\|W(i, j)\| < d$ has been pruned from the data before creating the instance.	$d \in [0, 0, 2]$

3.5 Solver Performance

In [7] we conducted an extensive experimental evaluation of the feasibility of solving correlation clustering using MaxSAT. We compared our encoding with previously proposed (exact) integer-linear and quadratic programming formulations, and used the state-of-the-art solvers Cplex, Gurobi, and SCIP for solving. For solving the MaxSAT instances we used the MaxHS solver [17]. We found that this SAT-IP hybrid MaxSAT-solving approach scales significantly better on the benchmarks. For example, using a timeout of 8 hours MaxHS was able to solve 3 out of the 4 protein datasets completely (i.e., the entire datasets). In comparison, the ILP solvers at best managed to cluster 75% of the datapoints available in the Protein 3 dataset, but ran out of memory at 50% for the other datasets. The choice of MaxHS as the MaxSAT solver here is motivated by experiments with a slightly different set-up where we compared MaxHS to other more standard MaxSAT algorithms. On a benchmark set consisting of 140 instances, MaxHS solved 119 while the next best solver, Eva [34] (with the original real-valued weights from the similarity matrices multiplied and truncated to integer values using greatest available resolution) solved only 65. Other state-of-the-art MaxSAT solvers, such as OpenWBO [32], did not exhibit competitive performance.

4 Learning Optimal Bounded Treewidth Bayesian Network Structures

Bayesian networks are an important and widely-used class of probabilistic graphical models for representing joint probability distributions, i.e., probabilistic relationships among a set of variables of interest [37]. A Bayesian network consists of a network structure, represented as an acyclic directed graph (DAG), and the parameters associated with each node (i.e., variable) in the DAG. Given a set of random variables $X = \{X_1, \dots X_N\}$ the Bayesian network structure learning problem (BNSL) asks to find a DAG G with the node set X minimizing a given non-negative scoring function $s(G)$. Several different scoring function are well-known. The only—standard—assumption our MaxSAT encoding makes is that the score $s(G)$ of any possible structure $G = (X, E)$ is *decomposable* [23]. A decomposable score can be expressed as the sum $s(G) = \sum_{i=1}^{N} s_i(P_i)$ where $P_i = \{X_j \in X \setminus \{X_i\} \mid (X_j, X_i) \in E\}$ is the *parent set* of node X_i and $s_i(P_i)$ the local score of selecting the parent set P_i for X_i. Indeed, as explained in Section 4.2, commonly applied scoring functions are typically decomposable.

To begin with, in the BNSL problem, we are given a set of data in the form of observations over the set X. An example is shown in Figure 2 (left). Based on the observations, a local score $s_i(P_i)$ is calculated for each variable X_i and candidate parent set P_i; see the example in Figure 2 (middle). The goal of the structure learning problem is to learn a DAG G, i.e., pick parents for each node X_i *while maintaining acyclicity*, such that G minimizes objective function s. An example is shown in Figure 2 (right).

After learning a Bayesian network, the network is typically used for probabilistic inference tasks, such as determining the most likely joint assignments of a set of variables under given evidence. Exact Bayesian inference is in general NP-hard [15], but for *bounded (fixed) treewidth* networks exact inference becomes tractable [30]. Treewidth is a fundamental and important graph theoretic property, which intuitively characterizes "how far" an undirected graph is from being a tree. In general, the treewidth of graphs on n nodes ranges from 1 (for trees) to $n - 1$ (including the complete graph K_n). Computing the treewidth of an undirected graph is in general an NP-hard problem. From the computational perspective, treewidth has important connections to (in)tractability. Many NP-hard problems on graphs, when restricted to input

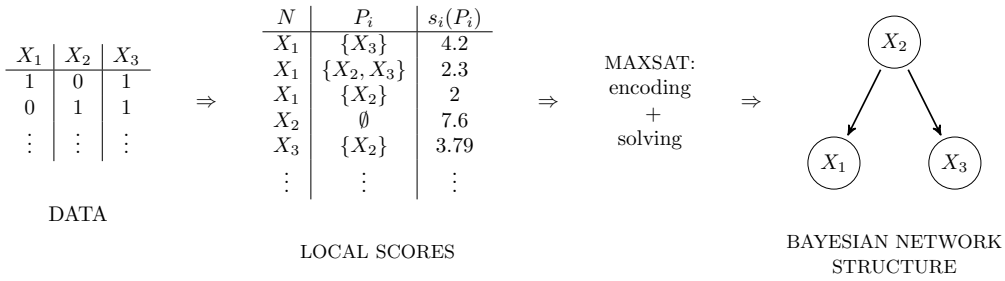

Figure 2: The Bayesian network structure learning problem by example

graphs of bounded treewidth, can be solved in polynomial time via dynamic programming, including Bayesian inference. This motivates the study of algorithms for the problem of learning optimal bounded treewidth Bayesian network structures (BTW-BNSL). Here it is important to note that BTW-BNSL is NP-hard for any treewidth bound $W > 1$ [27].

4.1 Problem Definition

A more precise formulation of the bounded treewidth Bayesian network structure learning problem is as follows. Given a set $X = \{X_1, \ldots, X_N\}$ of nodes, for each node X_i a set $\mathcal{P}_i \subset \mathcal{P}(X \setminus \{X_i\})$ of candidate parent sets for X_i, a non-negative local score (cost) $s_i(P_i)$ for each candidate parent set P_i, and a treewidth bound W, find a DAG G^* such that

$$G^* \in \operatorname*{arg\,min}_{G \in \mathrm{DAG}_W(N)} \sum_{i=1}^{N} s_i(P_i),$$

where $\mathrm{DAG}_W(N)$ is the set of DAGs over N nodes having treewidth at most W.

4.2 Computing Weights

As already mentioned, there are a number of ways to calculate the score of a graph structure, which results in the local scores s_i required as input to the MaxSAT encoding. Intuitively, a score should favor structures that explain the data well (e.g., maximize likelihood), but at the same time the score should penalize unnecessarily dense structures. One of the simplest examples of scores is the log (Bayesian) posterior probability of the structure G given the observed dataset D, i.e., $s(G) = \log P(G \mid D)$. With suitable choices of priors, this score is decomposable and computable in closed form [16]. Commonly used (decomposable) scores include BDE [16, 24], BIC [40], and scores based on information theoretical concepts such as MDL [29] and fNML [42].

4.3 MaxSAT Encoding

Next we briefly overview the MaxSAT encoding of bounded treewidth Bayesian network learning. For more details we refer the reader to [8].

Assume that we are given a set $X = \{X_1, \ldots, X_N\}$ of nodes, and, for each X_i, a non-negative local score (cost) $s_i(S)$ for each candidate parent set S of X_i. Let K stand for the maximum number of candidate parent sets the individual nodes in X have. Our MaxSAT encoding of the

BTW-BNSL problem under a fixed treewidth bound W includes one variable P_i^S for each node X_i and its candidate parent set S. The semantics of these variables is $P_i^S = 1$ iff the set S is chosen as the parent set of node X_i. Using these variables we encode the following constraints.

Each node in the learned network has exactly one (possibly empty) parent set. For a fixed node X_i, this corresponds to the cardinality constraint $\sum_{S \in \mathcal{P}_i} P_i^S = 1$. In the benchmarks, we encode this using the *improved sequential counter CNF encoding* of [39], resulting in $\mathcal{O}(N \cdot K)$ clauses and variables[3].

The learned network is acyclic. In order to ensure that the learned network is acyclic, we encode for each node X_i a *level number* $l(X_i) \in \{0, 1, \ldots, N-1\}$ in binary. Using the level numbers, acyclicity can be ensured by enforcing that $l(X_j) > l(X_i)$ for all parents X_j of X_i. This constraint results in $\mathcal{O}(WNK \cdot \log(N))$ variables and $\mathcal{O}(WNK \cdot \log(N))$ clauses.[4]

Optimality of the learned network. In order to encode the objective function of (BTW-)BNSL, we need to ensure that selecting a parent set S for the node X_i results in a cost of $s_i(S)$. This is encoded as soft unit clauses $(\neg P_i^S)$ with cost $s_i(S)$ for all nodes X_i and candidate parent sets S, resulting in $\mathcal{O}(NK)$ soft clauses.

Bounding the treewidth of the learned network: This constraint is the most involved of the whole encoding. We view treewidth computation as a problem of finding an optimal linear ordering of the nodes [18, 10]. More precisely, our encoding enforces that there exists a linear ordering over the nodes of the network that has width at most W. This is sufficient for bounding the treewidth as treewidth can be defined as the minimum width of all possible orderings. We refer the reader to the references for more details and mention here that this constraint requires $\mathcal{O}(N^2)$ variables and $\mathcal{O}(N^3)$ clauses.

All in all the size of an instance created by our encoding is polynomial in the input, i.e., the candidate parent sets. Depending on the input data, the size of the resulting instances is dominated by either the acyclicity or the treewidth constraint.

4.4 Available Benchmarks

We generated the BTW-BNSL benchmarks based 21 standard BNSL benchmark datasets, and used $W = 2, 3, 4$ as the treewidth bounds. This resulted in 63 instances. In more detail, the benchmark datasets and the used scoring functions were the following.

Asia 100, 1000 and 10000, sampled from a BN with 8 nodes, Insurance 100, 1000, (27 nodes), Water 100, 1000 (32 nodes), Alarm 100 (37 nodes), and Hailfinder 100, 1000, 10000, (56 nodes). The precomputed local scores for these datasets are available from `http://www.cs.york.ac.uk/aig/sw/gobnilp/`. The scores were computed using the BDeu scoring function with an equivalent sample size of 1.

Abalone (9 nodes), Wine (14 nodes), Zoo (17 nodes), Voting (17 nodes), Hepatitis (20 nodes), Heart (23 nodes), Horse (28 nodes), and Flag (29 nodes). These raw datasets are available from: `http://archive.ics.uci.edu/ml`. We computed the scores using the well-known MDL scoring function.

[3] We did not find a better performing (in terms of solver runtimes) cardinality encoding in preliminary experiments.

[4] The W term follows from the fact that the size of all candidate parent sets can be assumed to be at most W when learning a network with treewidth at most W. This is due to the fact that choosing a parent set of cardinality greater than W for some node would immediately imply that the treewidth of the learned network structure would be greated than W.

Table 2: Overview of the benchmarks encoding bounded treewidth Bayesing network structure learning

Filename:	$< r_ >$BTWBNSL_*dataset*_TWBound$b$.wcnf	
Parameter	**Description**	**Range**
r	Indicates that the weights of the soft clauses are rounded to integers	"Rounded" "
dataset	Name of the dataset the instance is based on	
b	The enforced treewidth bound on the network	$b \in \{2, 3, 4\}$

Housing (14 nodes) and Adult (15 nodes). Pre-computed local scores for these datasets are available from `http://www.cs.helsinki.fi/u/jazkorho/aistats-2013/`.

The parameter values and filename convention used for the benchmarks are detailed in Table 2.

4.5 Solver Performance

We empirically compared our MaxSAT-based approach for BTW-BNSL to a previously proposed dynamic programming algorithm (DP) [27] developed specifically for BTW-BNSL, as well as competing state-of-the-art approaches based on integer-linear programming: TwILP [36], based on integrating domain-specific cutting planes for iteratively ruling out cyclic structures found during search; and IP, a recently proposed approach based on directly encoding BTW-BNSL as an integer program [35]. Out of the 63 benchmarks, 56 were solved by MaxHS via our MaxSAT encoding using a timeout of 8 hours. The corresponding numbers of solved instances for DP, TwILP, and IP were 25, 45, and 41, respectively. MaxHS also turned our to be the best-performing MaxSAT solver, when compared to various other state-of-the-art MaxSAT solvers. For example, the Eva [34] solver was able to solve 9, and the ILP-2013 [3] approach (based on encoding MaxSAT instances to integer programs in a standard way and calling Cplex) was able to solve 13 instances.

5 Causal Structure Discovery

Discovering causal relations between quantities of interest is an essential part of many fields of science. Information on causal relations allows us to understand and predict system behavior not only when a system is in its natural (passively observed) state (e.g., patient without drugs), but also when the system is intervened on (e.g., when a doctor gives a certain drug to the patient) [38]. Although randomized controlled trials are the most reliable way of obtaining causal information, recent advances in causal inference have made it possible to formally gain causal information also from passively observed data [38, 43]. In the simplest scenario we consider here, we have passively observed measurement data from the system under investigation (Figure 3, left), and the aim is to find the graph[5] describing the causal relations working in the data generating system (Figure 3, right). In this task, the following MaxSAT-based approach

[5]Even with infinite amount of samples, we can only identify the true causal graph up to an equivalence class of graphs. Here we aim at finding a representative graph from that equivalence class.

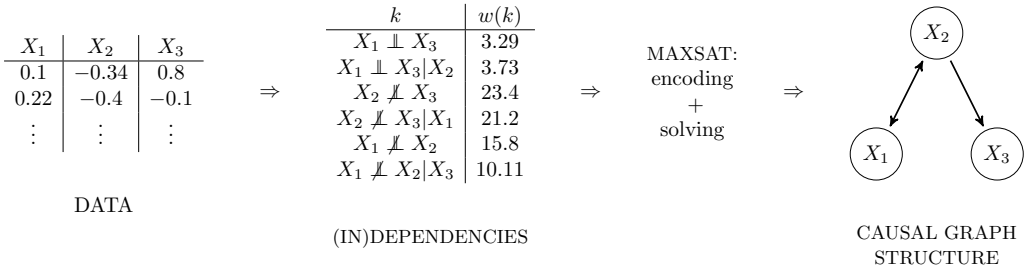

Figure 3: The causal structure discovery problem by example

currently allows for most general graph space (cycles and latent variables) and offers also better accuracy than previous approach [25]. As a trade-off for generality and accuracy, the approach currently has limited scalability, and is thus open for improvements.

A causal structure (see an example in Figure 3, right) is a mixed graph $G = (X, E)$ over a set of nodes $X = \{X_1, \ldots, X_N\}$ that represents measured aspects of a system (e.g., smoking habits, age, height, gender). The set of edges $E = E_{\rightarrow} \cup E_{\leftrightarrow}$ consists of directed edges $E_{\rightarrow} = \{(\ X_i, X_j) \mid X_i \in X, X_j \in X, X_i \neq X_j\}$ and (symmetric) bidirected edges $E_{\leftrightarrow} = \{\{X_i, X_j\} \mid X_i \in X, X_j \in X, X_i \neq X_j\}$. Directed edges ($\rightarrow$) in the graph represent causal relations (e.g., smoking causes cancer). Note that causal graphs are allowed to include directed cycles [26], and so, between any two nodes there can be up to three edges ($\rightarrow$,$\leftarrow$,$\leftrightarrow$).

Bi-directed edges are used for representing the presence of exogenous or outside influence on the measured variables. More formally, a bi-directed edge $X_i \leftrightarrow X_j$ denotes the presence of a 'latent confounder' (e.g., particular but unidentified gene), that has a causal effect on both X_i and X_j, i.e., a structure of the form $X_i \leftarrow X_k \rightarrow X_j$, with X_k being unmeasured. Instead of including potentially many nodes whose values are not measured, this inclusion of bi-directed edges in the graph allows for a type of a canonical representation of causal structures, with a graph over just the measured nodes (see [38, 43] for details).

Due to the more general types of graphs and models, the approach for finding optimal graph structures taken in Section 3 is not applicable in this context. Instead, intuitively, we search for a causal graph whose reachability properties match the statistical dependence (e.g. correlation) properties of the data. So first, for each pair of variables $\{X_i, X_j\}$ and each *conditioning set* $C \subseteq X \setminus \{X_i, X_j\}$ we test whether the variables are statistically dependent[6] ($X_i \not\perp\!\!\!\perp X_j|C$) or independent ($X_i \perp\!\!\!\perp X_j|C$) in the observed data (Figure 3, middle). Furthermore, we also obtain a weight describing the reliability of the decision (see Section 4.2 for details).

Now, under some common theoretical assumptions (see [43] for details), there exists a conditional dependence $X_i \not\perp\!\!\!\perp X_j|C$ in the observed data if and only if there is a so-called *d-connecting path given* C between X_i and X_j in the causal graph structure of the true data generating system. A d-connecting path given set of nodes C is a path (repeated edges are allowed) such that every 'collider node' connected with two incoming edges on the path is in C and other nodes on the path are not in C [44, 38]. For example, path $X_1 \leftarrow X_2 \leftrightarrow X_3 \leftarrow X_4$ is a d-connecting path between X_1 and X_4 for $C = \{X_3\}$, but not for $C = \emptyset$ or $C = \{X_2, X_3\}$. Thus in the data generated by a system with causal structure $X_1 \leftarrow X_2 \leftrightarrow X_3 \leftarrow X_4$, we would observe dependence $X_1 \not\perp\!\!\!\perp X_4|X_3$ and independencies $X_1 \perp\!\!\!\perp X_4$ and $X_1 \perp\!\!\!\perp X_4|X_2, X_3$ (theoretically).

Thus, according to this theory, the statistical (in)dependence (Figure 3, middle) relations

[6]Intuitively, X_i is statistically dependent on X_j given C iff the value of X_i helps to predict X_j when we already know the values of variables in C.

directly translate to reachability and separability constraints on the paths of the causal graph, and hence provide the input to a constraint solver. However, the statistical independence tests run on limited sample sizes produce errors relatively often, and thus the obtained constraints are unsatisfiable simultaneously in any realistic scenario. This gives rise to an optimization problem, which we address via MaxSAT.

5.1 Problem Definition

The input to the causal structure discovery optimization problem is a set K of reachability and separability constraints. In more detail, K includes a constraint for each pair of nodes in the graph and for each conditioning set C, stating whether the variables should be reachable or separable by d-connecting paths (for an example input, see Figure 3 middle). A weight function $w(k)$ gives a non-negative cost for not satisfying each reachability/separability constraint $k \in K$. The task is to find a causal graph G^* (Figure 3, right) that minimizes the sum of costs of reachability/separability constraints that are not satisfied:

$$G^* \in \operatorname*{arg\,min}_{G \in CG(n)} \sum_{k \in K \,:\, G \not\models k} w(k), \tag{2}$$

where the class of causal graphs with n nodes is denoted by $CG(n)$, and $G \not\models k$ denotes that a causal graph G does not satisfy a reachability/separability constraint $k \in K$.

5.2 Computing Weights

We consider three different weighting schemes: 1) all the constraints have unit weights; 2) the independence constraints have unit weights while the dependence constraints have infinite weights; and 3) log-weights. Particular types of Bayesian tests also produce a probability of independence that directly describes the reliability of the test result [26]. We input the more reliable constraint of the two options: $X_i \perp\!\!\!\perp X_j | C$ vs. $X_i \not\perp\!\!\!\perp X_j | C$, with the difference of the log-probabilities as its non-negative real-valued weight.

5.3 MaxSAT Encoding

Finally, we give an overview of two alternative encodings for the problem of causal structure discovery, fully detailed in [26, 25]. As the central definition of d-connection is rather involved, we aim here only at giving some intuition on the encodings without full formal details. In both of the encodings, we represent each of the bi-directed and directed edges for pairs of distinct nodes with Boolean variables. The input reachability and separability constraints are represented by individual Boolean variables, and included in both of the encodings as unit soft clauses with weights assigned according to the input. Note that already the number of input constraints is exponential w.r.t. to the number of nodes in the graph.

The particular type of reachability used (d-connections), establishing the connection between the causal graph and the input constraints, needs to be encoded as hard constraints. The first option is to build up all d-connecting paths with recursion over path length [26]. This amounts to $\mathcal{O}(n^3)$ for each constraint, where n is the number of nodes in the graph (although many of the clauses and variables can be shared between the path-encodings for the different reachability/separability constraints). Alternatively, we can encode all independence/dependence relations by applying particular graph operations consecutively starting from the causal graph [25]. This also gives complexity of $\mathcal{O}(n^3)$ per constraint, but allows for more sharing

between the constraints. We have found that this also translates into improved solver runtimes in practice.

5.4 Available Benchmarks

Causal discovery instances were generated from random linear Gaussian causal models, with 5-7 observed variables each. The average degree of the causal graphs was 2. For the log-weights, both the floating-point and the rounded integer weighted version of the instances are included. The weighting schemes, including whether integer or floating-pointing representation of weights were used, have considerably different running times. The parameter values and filename convention used for the benchmarks are detailed in Table 3.

5.5 Solver Performance

We shortly report on a previously unpublished solver comparison on the causal structure discovery benchmarks. We used the following MaxSAT solvers: Clasp [21], Eva [34], MaxHS [17], MSCG [33], OpenWBO [32], Primal-Dual [9], and QMaxSAT [28], and enforced a per-instance timeout of 900 seconds. Considering all the 116 benchmarks (including all three weighting schemes), we observed that the two best-performing solver were QMaxSAT (a model-based solver, solving 114 instances) and Clasp (solving 109 instances using its default branch-and-bound scheme). All other solvers solved at most 85 instances. On the perhaps most interesting log-weighted benchmarks—based on the weighting scheme that was shown in [25] to be the most accurate one in terms of the discovered causal structures—Clasp performed best, solving all 35 instances each under 250 seconds. QMaxSAT was second-best, solving the 35 instances each under 750 seconds. MaxHS came in third, with 26 solved instances, while the other solvers did not manage to solve more than 11 instances.

Table 3: Overview of the benchmarks encoding causal structure discovery

Filename:	causal_nn_ii_NN_enc_$weight$_$type$.wcnf	
Parameter	**Description**	**Range**
n	Number of observed variables in the causal graph	$n \in \{5, 6, 7\}$
i	Instance (seed) of the problem	$i \in [0, 1, \ldots, 10]$
N	The number of samples	$N \in \{500, 1000, 10000\}$
enc	The encoding used	"uai13" (see [26]) "uai14" (see [25])
weight	The type of weight (see [25])	"log" "const" "hardeps"
type	Indicates if the weights of the soft clauses are rounded to integers	"dec" (not rounded) "int" (rounded)

6 Conclusions

The main motivation of this work is to provide the MaxSAT community novel weighted partial MaxSAT benchmarks, thus extending the currently somewhat limited range of benchmarks from different problem domains. To this end, we outlined three types of important data analysis problems, shortly explained how we have recently succesfully addressed these problems with MaxSAT technology, and provided details on a large benchmark set containing instances generated under various parameter value combinations from each of the described problem domains. To motivate MaxSAT solver developers to extend their solvers to accept floating-point representations of weights on soft clauses, we also provide the benchmarks in their original real-valued weights in addition to rounded integer weighted versions.

The data analysis problems described represent somewhat untraditional problems in terms of typical problem domains for which SAT-based technology has proven an effective solving approach. Nevertheless, MaxSAT has proven to provide interesting alternative approaches to solving each of the problem domains. In light of this, we hope this work motivates further research on using SAT-based techniques for data analysis. From the experiences we have gained when working on using MaxSAT in these problem domains, we also see potential for motivating the development of MaxSAT techniques for more efficiently solving hard combinatorial optimization problems arising from the data analysis domain. For example, for scaling up MaxSAT for data analysis, there is an evident need for coping with large input data (perhaps e.g. by adopting a higher level representation language and applying partial grounding techniques within the MaxSAT solving step), and a need for incremental interfaces in the context of MaxSAT. On the other hand, such problem domains involve constraints which are naturally represented on the Boolean level, which gives promise for developing further SAT-based approaches to data analysis problems. Finally, we find it interesting that best-performing MaxSAT solvers are MaxHS (a SAT-MIP hybrid; for the correlation clustering and the BTW-BNSL problems) and Clasp (using a branch-and-bound algorithm; for the causal structure discovery problem). Moreover, MaxHS shows noticeably better runtime performance on BTW-BNSL than e.g. core/model-based MaxSAT solvers that are based purely on a SAT solver, as well as competing state-of-the-art exact approaches (both IP-based and specialized algorithms). This also seems to suggest that these benchmarks can increase heterogeneity among the current standard benchmark sets used for evaluating (weighted partial) MaxSAT solvers.

Acknowledgements

Work funded by Academy of Finland (grants 251170 Centre of Excellence in Computational Inference Research, 276412, and 284591), Finnish Foundation for Technology Promotion (TES), and Research Funds of the University of Helsinki.

References

[1] Nir Ailon, Moses Charikar, and Alantha Newman. Aggregating inconsistent information: Ranking and clustering. *J. ACM*, 55(5), 2008.

[2] Carlos Ansótegui, Maria Luisa Bonet, and Jordi Levy. Solving (weighted) partial MaxSAT through satisfiability solving. In *Proc. SAT*, volume 5584 of *LNCS*, pages 427–440. Springer, 2009.

[3] Carlos Ansótegui and Joel Gabàs. Solving (weighted) partial maxsat with ILP. In *Proc CPAIOR*, volume 7874 of *LNCS*, pages 403–409. Springer, 2013.

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

[5] Amir Ben-Dor, Ron Shamir, and Zohar Yakhini. Clustering gene expression patterns. *Journal of Computational Biology*, 6(3/4):281–297, 1999.

[6] Jeremias Berg and Matti Järvisalo. Optimal correlation clustering via MaxSAT. In *Proceedings of the 2013 IEEE 13th International Conference on Data Mining Workshops (ICDMW 2013)*, pages 750–757. IEEE Press, 2013.

[7] Jeremias Berg and Matti Järvisalo. Cost-optimal constrained correlation clustering via weighted partial maximum satisfiability. *Artificial Intelligence*, to appear (2015).

[8] Jeremias Berg, Matti Järvisalo, and Brandon Malone. Learning optimal bounded treewidth bayesian networks via maximum satisfiability. In *Proc. AISTATS*, volume 33 of *JMLR Workshop and Conference Proceedings*, pages 86–95. JMLR, 2014.

[9] Nikolaj Bjørner and Nina Narodytska. Maximum satisfiability using cores and correction sets. In *Proc. IJCAI*, pages 246–252. AAAI Press, 2015.

[10] Hans L. Bodlaender. Discovering treewidth. In *Proc. SOFSEM*, volume 3381 of *LNCS*, pages 1–16. Springer, 2005.

[11] Francesco Bonchi, Aristides Gionis, Francesco Gullo, and Antti Ukkonen. Chromatic correlation clustering. In *Proc. KDD*, pages 1321–1329. ACM, 2012.

[12] Francesco Bonchi, Aristides Gionis, and Antti Ukkonen. Overlapping correlation clustering. In *Proc. ICDM*, pages 51–60. IEEE, 2011.

[13] Nicolò Cesa-Bianchi, Claudio Gentile, Fabio Vitale, and Giovanni Zappella. A correlation clustering approach to link classification in signed networks. In *Proc. COLT*, volume 23 of *JMLR Procedings*, pages 34.1–34.20. JMLR.org, 2012.

[14] Moses Charikar, Venkatesan Guruswami, and Anthony Wirth. Clustering with qualitative information. *J. Comput. Syst. Sci.*, 71(3):360–383, 2005.

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

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

[17] Jessica Davies and Fahiem Bacchus. Exploiting the power of mip solvers in maxsat. In *Proc - SAT*, volume 7962 of *LNCS*, pages 166–181. Springer, 2013.

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

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

[20] Zhaohui Fu and Sharad Malik. On solving the partial MAX-SAT problem. In *Proc. SAT*, volume 4121 of *LNCS*, pages 252–265. Springer, 2006.

[21] Martin Gebser, Benjamin Kaufmann, and Torsten Schaub. Conflict-driven answer set solving: From theory to practice. *Artif. Intell.*, 187:52–89, 2012.

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

[23] David Heckerman. A tutorial on learning with Bayesian networks. In *Learning in Graphical Models*, volume 89 of *NATO ASI Series*, pages 301–354. Springer, 1998.

[24] David Heckerman, Dan Geiger, and David M. Chickering. Learning Bayesian networks: The combination of knowledge and statistical data. *Machine Learning*, 20:197–243, 1995.

[25] Antti Hyttinen, Frederick Eberhardt, and Matti Järvisalo. Constraint-based causal discovery: Conflict resolution with answer set programming. In *Proc. UAI*, pages 340–349. AUAI Press, 2014.

[26] Antti Hyttinen, Patrik Hoyer, Frederick Eberhardt, and Matti Järvisalo. Discovering cyclic causal models with latent variables: A general SAT-based procedure. In *Proc. UAI*, pages 301–310. AUAI Press, 2013.

[27] Janne H. Korhonen and Pekka Parviainen. Exact learning of bounded tree-width Bayesian networks. In *Proc. AISTATS*, pages 370–378, 2013.

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

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

[30] Steffen L. Lauritzen and David J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society. Series B (Methodological)*, 50(2):157–224, 1988.

[31] Ruben Martins, Saurabh Joshi, Vasco Manquinho, and Ines Lynce. Incremental cardinality constraints for MaxSAT. In *Proc. CP*, volume 8656 of *LNCS*, pages 531–548. Springer, 2014.

[32] Ruben Martins, Vasco M. Manquinho, and Inês Lynce. Open-WBO: A modular MaxSAT solver. In *Proc. SAT*, volume 8561 of *LNCS*, pages 438–445. Springer, 2014.

[33] António Morgado, Carmine Dodaro, and Joao Marques-Silva. Core-guided MaxSAT with soft cardinality constraints. In *Proc. CP*, volume 8656 of *LNCS*, pages 564–573. Springer, 2014.

[34] Nina Narodytska and Fahiem Bacchus. Maximum satisfiability using core-guided MaxSAT resolution. In *Proc. AAAI*, pages 2717–2723. AAAI Press, 2014.

[35] Siqi Nie, Denis Deratani Mauá, Cassio Polpo de Campos, and Qiang Ji. Advances in learning bayesian networks of bounded treewidth. In *Proc, NIPS*, pages 2285–2293, 2014.

[36] Pekka Parviainen, Hossein Shahrabi Farahani, and Jens Lagergren. Learning bounded tree-width bayesian networks using integer linear programming. In *Proc. AISTATS*, volume 33 of *JMLR Workshop and Conference Proceedings*, pages 751–759. JMLR, 2014.

[37] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., 1988.

[38] Judea Pearl. *Causality: Models, Reasoning, and Inference*. Cambridge University Press, 2000.

[39] Marko Samer and Helmut Veith. Encoding treewidth into SAT. In *Proc. SAT*, volume 5584 of *LNCS*, pages 45–50. Springer, 2009.

[40] Gideon Schwarz. Estimating the dimension of a model. *The Annals of Statistics*, 6(2):461–464, 1978.

[41] Ron Shamir, Roded Sharan, and Dekel Tsur. Cluster graph modification problems. *Discr. Appl. Math.*, 144(1-2):173–182, 2004.

[42] Tomi Silander, Teemu Roos, Petri Kontkanen, and Petri Myllymäki. Factorized normalized maximum likelihood criterion for learning Bayesian network structures. In *Proc. PGM*, pages 257–272, 2008.

[43] Peter Spirtes, Clark Glymour, and Richard Scheines. *Causation, Prediction, and Search*. Springer-Verlag, 1993.

[44] Milan Studený. Bayesian networks from the point of view of chain graphs. In *Proceedings of UAI*, pages 496–503. Morgan Kaufmann, 1998.