PREFACE

The MaxSAT Evaluations are a series of events focusing on the evaluation of current state-of-the-art systems for solving optimization problems via the Boolean optimization paradigm of maximum satisfiability (MaxSAT). Organized yearly starting from 2006, the year 2017 brought on the 12th edition of the MaxSAT Evaluations. Some of the central motivations for the MaxSAT Evaluation series are to provide further incentives for further improving the empirical performance of the current state of the art in MaxSAT solving, to promote MaxSAT as a serious alternative approach to solving NP-hard optimization problems from the real world, and to provide the community at large heterogeneous benchmark sets for solver development and research purposes. In the spirit of a true evaluation—rather than a competition, unlike e.g. the SAT Competition series—no winners are declared, and no awards or medals are handed out to overall best-performing solvers.

In 2017, a new team stepped in to organize the MaxSAT Evaluation. Several changes to the evaluation arrangements were introduced with this change.

The 2017 evaluation consisted of two main tracks, one for solvers focusing on unweighted and one for solvers focusing on weighted MaxSAT instances. In contrast to the previous instantiations of MaxSAT Evaluations, no distinction was made between “industrial” and “crafted” benchmarks. Furthermore, no track for purely randomly generated MaxSAT instances was organized this year. In addition to the main tracks, a special track for incomplete MaxSAT solvers was organized, using two short per-instance time limits (60 and 300 seconds), differentiating from the per-instance time limit of 1 hour imposed in the main tracks.

In terms of rules, solvers were now required to be open-source, and the source codes of all participating solvers were made available online on the evaluation webpages after the results from the evaluation were presented. This new requirement was introduced to promote easier entrance to the world of MaxSAT solver development and was also motivated by the success of open-source SAT solvers. A special “no-restrictions” track was arranged to accommodate developers unable to adhere to the open-source requirements—however, no solvers were submitted to this special track.

Following the SAT Competitions, a 1-2 page solver description was required, to provide some details on the search techniques implemented in the solvers. The solvers descriptions together with descriptions of new benchmarks for 2017 are collected together in this compilation.

Benchmark selection for the 2017 evaluation was refined with the aim of making the 2017 benchmark sets balanced in terms of the number of representative instances included from different benchmark problem domains.

We would like to thank the previous MaxSAT Evaluation organizers for their noticeably efforts and hard work on organizing the MaxSAT Evaluations for several consecutive years. The evaluations have played an important role in bringing MaxSAT to its current position as a competitive approach to tackling NP-hard optimization problems. We hope that the success of MaxSAT Evaluations continues also in the forthcoming years.

Finally, we would like to thank everyone who contributed to MaxSAT Evaluation 2017 by submitting their solvers or new benchmarks. We are also grateful for the computational resources provided by the StarExec initiative which enabled running the 2017 evaluation smoothly.

MaxSAT Evaluation 2017 Organizers
Contents

Preface ................................................................. 3

Solver Descriptions

MaxHS v3.0 in the 2017 MaxSat Evaluation
  Pahsem Bacchus .................................................. 8
Maxino
  Mario Alviano .................................................... 10
MaxRoster: Solver Description
  Takayuki Sugawara .............................................. 12
Loandra: PMRES Extended with Preprocessing Entering MaxSAT Evaluation 2017
  Jeremias Berg, Tuukka Korhonen, and Matti Järvisalo ............... 13
The MSUSorting MaxSAT solver
  Eivind Jahren and Roberto As ÍĄí Achaá ................................ 15
LMHS in MaxSAT Evaluation 2017
  Paul Saikko, Tuukka Korhonen, Jeremias Berg, and Matti Järvisalo .... 16
Open-WBO in MaxSAT Evaluation 2017
  Ruben Martins, Miguel Terra-Neves, Saurabh Joshi, Mikolas Janota, Vasco Manquinho, and Ines Lynce ................................. 17
QMaxSAT1702 and QMaxSATuc
  Naoki Uemura, Aolong Zha, and Miyuki Koshimura .................... 18

Benchmark Descriptions

MaxSAT Benchmarks: CSS Refactoring
  Matthew Hague and Anthony Widjaja Lin .......................... 20
MaxSAT Benchmarks based on Determining Generalized Hypertree-width
  Jeremias Berg, Neha Lodha, Matti Järvisalo, and Stefan Szeider .......... 22
Discrete Optimization Problems in Dynamics of Abstract Argumentation: MaxSAT Benchmarks
  Andreas Niskanen, Johannes P. Wallner, and Matti Järvisalo ............ 23
Lisbon Wedding: Seating arrangements using MaxSAT
  Ruben Martins and Justine Sherry .................................. 25
ASP to MaxSAT: Metro, ShiftDesign, TimeTabling and BioRepair
  Ruben Martins .................................................. 27
SOLVER DESCRIPTIONS
MaxHS v3.0 in the 2017 MaxSat Evaluation

Fahiem Bacchus
Department of Computer Science
University of Toronto
Ontario, Canada
Email: fbacchus@cs.toronto.edu

1. MaxHS

MaxHS is a MaxSat solver that originated in the PhD work of Davies [4]. It was the first MaxSat solver to utilize the Implicit Hitting Set (IHS) approach, and its core components are described in [4], [2], [3], [5]. Other useful insights into IHS are provided in [6], [7]. IHS solvers utilize both an integer programming (IP) solver and a SAT solver in a hybrid approach to MaxSat solving. MaxHS utilizes minisat v2.2 as its SAT solver and IBM’s CPLEX v12.7 as its IP solver. Interestingly experiments with more sophisticated SAT solvers like Glucose http://www.labri.fr/~perso/lismon/glucose/ and Lingeling http://fmv.jku.at/lingeling/ yielded inferior performance. This indicates that the SAT problems being solved are quite simple, too simple for the more sophisticated techniques used in these SAT solvers to pay off. Simpler SAT problems are one of the original motivations behind MaxHS [2].

The MaxHS v3.0 is essentially the same as the version that was entered in the 2016 MaxSat evaluation, but with some clean up of the code, some extensions to the techniques used, and some previously undetected bugs fixed. These bugs were mainly impediments to performance, but one bug was found that had not appeared in prior testing on over 6000 instances!

The main features of v3.0, as compared to the prior published descriptions of MaxHS are as follows (familiarity with the basics of the IHS approach is assumed).

1.0.1. Termination based on Bounding. MaxHS v3.0 maintains an upper bound (and best model found so far) and a lower bound on the cost of an optimal solution (the IP solver computes valid lower bounds). MaxHS terminates when the gap between the lower bound and upper bound is low enough (with integer weights when this gap is less than 1, the upper bound model is optimal). This means that MaxHS no longer needs to wait until the IP solver returns a hitting set whose removal from the set of soft clauses yields SAT; it can return when the IP solver’s best lower bound is close enough to show that the best model is optimal.

1.0.2. Early Termination of Cplex. In previous versions of MaxHS, the IP solver was run to completion forcing it to find an optimal solution every time it is called. However, with bounding, optimal solutions are not always needed. In particular, if the IP solver finds a feasible solution whose cost is better than the current best model it can return that: either the IP solution is feasible for the MaxSat problem, in which case we can lower the upper bound, or it is infeasible in which case we can obtain additional cores to augment the IP model (and thus increase the lower bound). Terminating the IP solver before optimization is complete can yield significant time savings.

1.0.3. Reduced Cost fixing via the LP-Relaxation. Using an LP relaxation and the reduced costs associated with the optimal LP solution, some soft clauses can be hardened or immediately falsified. See [1] for more details.

1.0.4. Mutually Exclusive Soft Clauses. Sets of soft clauses of which at most one can be falsified or at most one can be satisfied are detected. When all of these soft clauses have the same weight they can all be more compactly encoded with a single soft clause. This encoding does not always yield better performance due to some subtle effects. However, techniques were developed to better exploit such information, and a fuller description of these techniques is in preparation. With these techniques performance gains were achieved.

1.0.5. Other clauses to the IP Solver. Problems with a small number of variables are given entirely to the IP solver, so that it directly solves the MaxSat problem. In this case the SAT solver is used to first compute some additional clauses and cores, and to find a better initial model for the IP solver. This additional information from the SAT solver often makes the IP solver much faster than just running the IP solver and represents an alternate way of hybridizing SAT and IP solvers.

1.0.6. Other techniques for finding Cores. MaxHS iteratively calls the IP solver to obtain a hitting set of the cores computed so far. If that hitting set does not yield an optimal MaxSat solution then more cores must be added to the IP solver. In some of these iterations very few cores can be found causing only a slight improvement to the IP solver’s model. This results in a large number of time consuming calls to the IP solver. Two method were developed to aid
this situation (a) we ask the IP solver for more solutions and generate cores from these as hitting sets as well and (b) if we have a new upper bound model we try to improve this model by converting it to a minimal correction set (MCS). In converting the upper bound model to an MCS we either find a better model (lowering the upper bound) or we compute additional conflicts that can be added to the IP solver.

1.0.7. Incomplete MaxSat Solving. The solver maintains upper bounding models as described above, and in its normal operation it terminates only when it is able to prove that its best model is in fact optimal. However, often it is able to find very good upper bounding models or even optimal models long before termination (proving a model to be optimal is generally as hard or even harder than finding it). For the incomplete track we simply output the best model found so far at timeout.

References

Maxino

Mario Alviano
Department of Mathematics and Computer Science
University of Calabria
87036 Rende (CS), Italy
Email: alviano@mat.unical.it

Abstract—Maxino is based on the \(k\)-ProcessCore algorithm, a parametric algorithm generalizing OLL, ONE and PMRES. Parameter \(k\) is dynamically determined for each processed unsatisfiable core by a function taking into account the size of the core. Roughly, \(k\) is in \(O(\log n)\), where \(n\) is the size of the core. Satisfiability of propositional theories is checked by means of a pseudo-boolean solver extending Glucose 4.1 (single thread).

A Very Short Description of the Solver

The solver MAXINO is build on top of the SAT solver GLUCOSE [7] (version 4.1). MaxSAT instances are normalized by replacing non-unary soft clauses with fresh variables, a process known as relaxation. Specifically, the relaxation of a soft clause \(\phi\) is the clause \(\phi \lor \neg \varphi\), where \(\varphi\) is a variable not occurring elsewhere; moreover, the weight associated with clause \(\phi\) is associated with the soft literal \(\varphi\). Hence, the normalized input processed by MAXINO comprises hard clauses and soft literals, so that the computational problem amounts to maximize a linear function, which is defined by the soft literals, subject to a set of constraints, which is the set of hard clauses.

The algorithm implemented by MAXINO to address such a computational problem is based on unsatisfiable core analysis, and in particular takes advantage of the following invariant: A model of the constraints that satisfies all soft literals is an optimum model. The algorithm then starts by searching such a model. On the other hand, if an inconsistency arises, the unsatisfiable core returned by the SAT solver is analyzed. The analysis of an unsatisfiable core results into new constraints and new soft literals, which replace the soft literals involved in the unsatisfiable core. The new constraints are essentially such that models satisfying all new soft literals actually satisfy all but one of the replaced soft literals. Since there is no model that satisfies all replaced soft literals, it turns out that the invariant is preserved, and the process can be iterated.

Specifically, the algorithm implemented by MAXINO is \(k\)-ProcessCore procedure introduced by Alviano et al. [2]. It is a parametric algorithm generalizing OLL [3], ONE [2] and PMRES [8]. Intuitively, for an unsatisfiable core \(\{x_0, x_1, x_2, x_3\}\), ONE introduces the following constraint:

\[
\begin{align*}
x_0 \land x_1 \land x_2 \land x_3 \land \neg y_1 \land \neg y_2 \land \neg y_3 \geq \nabla y_1 \rightarrow y_2 \land y_2 \rightarrow y_3
\end{align*}
\]

where \(y_1, y_2, y_3\) are fresh variables (the new soft literals that replace \(x_0, x_1, x_2, x_3\)). OLL introduces the following constraints (the first immediately, the second if a core containing \(y_1\) is subsequently found, and the third if a core containing \(y_2\) is subsequently found):

\[
\begin{align*}
x_0 \land x_1 \land x_2 \land x_3 \land \neg y_1 \geq \nabla y_0 \land x_1 \land x_2 \land x_3 \land \neg y_2 \geq \nabla y_0 \land x_1 \land x_2 \land x_3 \land \neg y_3 \geq \nabla y_0
\end{align*}
\]

Concerning PMRES, it introduces the following constraints:

\[
\begin{align*}
x_0 \lor x_1 \lor \neg y_1 \land \neg y_2 \land \neg y_3 \geq \nabla y_0 \land x_1 \land x_2 \land x_3 \land \neg y_2 \land \neg y_3 \geq \nabla y_0 \land x_1 \land x_2 \land x_3 \land \neg y_3 \geq \nabla y_0
\end{align*}
\]

which are essentially equivalent to the following constraints:

\[
\begin{align*}
x_0 \land x_1 \land \neg z_1 \land \neg y_1 \geq \nabla z_1 \land x_0 \land x_1 \land z_1 \land x_2 \land x_3 \land \neg y_2 \land \neg y_3 \geq \nabla z_2 \land x_2 \land x_3 \land \neg y_3 \geq \nabla z_2
\end{align*}
\]

where \(y_1, y_2, y_3\) are fresh variables (the new soft literals that replace \(x_0, x_1, x_2, x_3\)), and \(z_1, z_2\) are fresh auxiliary variables.

Algorithm K, instead, introduces a set of constraints of bounded size, where the bound is given by the chosen parameter \(k\), and is specifically \(\cdot k \cdot\) ONE, which is essentially a smart encoding of OLL, is the special case for \(k = \infty\). For example unsatisfiable core, another possibility is \(k = 0\), which would result in the following constraints:

\[
\begin{align*}
x_0 \land x_1 \land \neg z_1 \land \neg y_1 \land \neg y_2 \land \neg y_3 \geq \nabla z_1 \land x_0 \land x_1 \land y_1 \land y_2 \land y_3 \geq \nabla z_1
\end{align*}
\]

In this version of MAXINO, the parameter \(k\) is dynamically determined based on the size of the analyzed unsatisfiable core: \(k \in O(n)\), where \(n\) is the size of the core.

The analysis of unsatisfiable core is preceded by a shrink procedure [1]. Specifically, a reiterared progression search is performed on the unsatisfiable core returned by the SAT solver. Such a procedure significantly reduces the size of the unsatisfiable core, even if it not necessarily returns an unsatisfiable core of minimal size. Since minimality of the unsatisfiable cores is not a requirement for the Additionally, satisfiability checks performed during the shrinking process are subject to a budget on the number of conflicts, so that the overhead due to hard checks is limited. Specifically, the budget is set to the number of conflicts arose in the satisfiability check that lead to detecting the unsatisfiable core; if such a number is less than 1000 (one thousand), the budget is raised to 1000. The budget is divided by 2 every time the progression is reiterated.
Weighted instances are handled by stratification and introducing remainders [4]–[6]. Specifically, soft literals are partitioned in strata depending on the associated weight. Initially, only soft literals of greatest weight are considered, and soft literals in the next stratum are added only after a model satisfying all considered soft literals is found. When an unsatisfiable core is found, the weight of all soft literals in the core is decreased by the weight associated with last added stratum. Soft literals whose weight become zero are not considered soft literals anymore.

Finally, a preprocessing step is performed on unweighted instances, which essentially iterates on all hard clauses of the input theory, sorted by length, and checks whether they already witness some unsatisfiable core. Specifically, an hard clause witnesses an unsatisfiable core if all literals in the clause are the complement of a soft literal. If this is the case, the unsatisfiable core is analyzed immediately. The rationale for such a preprocessing step is that hard clauses in the input theory are often small, and the smaller the better for the unsatisfiable core based algorithms.

REFERENCES

Abstract

In this document, we briefly describe the techniques employed by the MaxRoster solver participating in MaxSAT competition 2017.

I. Introduction

MaxRoster participates in Incomplete Track. MaxRoster has two engines, one is local search solver Ramp and another is MapleSAT with CHB. First, Ramp is used for 6 seconds and then the complete maxsat algorithm starts using MapleSAT. Our aim is to make feasible solution better, though it has ability of getting optimum solution.

II. Implementation

Weighted Instances:

For weighted instances, either incremental version of OLL algorithm or model-based algorithm is used. Initially, MaxRoster makes a call to the SAT solver using solely the hard clauses. If SAT, the cost of this model represents an initial upper bound on the MaxSAT solution. The ratio of the cost mainly determines which algorithm should be invoked later. In model-based algorithm, we implemented special clause counting the inputs with same weight in MapleSAT to address large and different weights for the instance.

Unweighted Instances:

For unweighted instances, either incremental version of MCU3 algorithm or model-based algorithm is used. Initially, MCU3 algorithm is invoked. If predefined timeout occurs in the process, then MaxRoster switches to model-based algorithm dynamically.

References


Loandra: PMRES Extended with Preprocessing

Entering MaxSAT Evaluation 2017

Jeremias Berg, Tuukka Korhonen, and Matti Järvisalo

Loandra makes extensive use of SAT-based preprocessing

1.1 Overview

The solver is first extended with a fresh label variable

2 RELIMINARIES

Loandra is a technique for solving weighted MaxSAT problems

2.1 STRUCTURE AND EXECUTION OF

2.2 OANDRA

Weighted partial maximum satisfiability (MaxSAT). Treating a

3 DETAILS ON THE COMPETITION BUILDS

3.1 OANDRA

This guarantees that the working formula is only modified

4 COMPILATION AND

4.1 SAGE

When the preprocessor is successful, the solver is reinitialized on

5 SAGE

3.2 OANDRA

If the preprocessor is successful, the solver is reinitialized on

5.1 SAGE

In more detail, whenever invoked on an MaxSAT instance

5.2 SAGE

This guarantees that the working formula is only modified

5.3 SAGE

Whenever more than 500 clauses have been hardened since

5.4 SAGE

MaxPre [11], modified to support addition of clauses.

5.5 SAGE

It is possible to keep the state

5.6 SAGE

Extracting a soft clause

5.7 SAGE

...


the number of unsatisfied soft clauses. To restrict algorithms MSU3 [12] and OLL [13]. These algorithms work be unfeasible or not significant, we heuristically choose to run found. Since the partitioning of some MaxSAT formulas may when only one partition remains and the optimal solution is partitioning techniques [7]. We represent a MaxSAT formula MaxSAT solving [4]. For unweighted MaxSAT, we extended MSU3 and OLL use the Totalizer encoding for incremental encoding [10] for cardinality constraints and the Generalized Boolean constraint) for unweighted (weighted) problems into iteration, we need to encode a cardinality constraint (pseudo-LSU and RES. The remainder of this document describes the Open-WBO were submitted to the MaxSAT Evaluation 2017: MaxSAT Evaluations of 2014, 2015 and 2016. Two versions of partial MaxSAT and has been one of the best solvers in the Open-WBO are based on a sequence of calls to a SAT solver. Even though Open-WBO can use any MiniSAT-like solver [2], for the purpose of this evaluation we are currently using Glucose A. Morgado, C. Dodaro, and J. Marques-Silva, “Core-Guided MaxSAT,” in CP. Springer, 2014, pp. 564–573.

We would like to thank Laurent Simon and Gilles Audemard as a spin-off of WBO [1]. Open-WBO implements a variety of algorithms for solving Maximum Satisfiability (MaxSAT) and

Since the partitioning of some MaxSAT formulas may

When only one partition remains and the optimal solution is found. Since the partitioning of some MaxSAT formulas may

We would like to thank Laurent Simon and Gilles Audemard as a spin-off of WBO [1]. Open-WBO implements a variety of algorithms for solving Maximum Satisfiability (MaxSAT) and...
In core-guided mode, all blocking variables in the subset from the clause database in order to eliminate the core. Thus, we add all blocking variables in the subset as literals, and add it to blocking variables.

QMaxSATuc performs core-guided mode with a set of all blocking variables. Glucose returns a subset of assumptions as assumptions. Glucose treats each literal in assumptions as a blocking variable in the subset from the clause database in order to eliminate the core. Thus, we add all blocking variables in the subset as literals, and add it to blocking variables.

QMaxSATuc runs in two modes: core-guided and model-guided while it mainly follows model-guided approach. QMaxSATuc alternates these modes.
MaxSAT Benchmarks based on Determining Generalized Hypertree-width
Jeremias Berg, Neha Lodha y, Matti Järvisalo ...
The MaxSAT encoding for computing maximum clique size. The encoding for GHTW is extended by ordering of $G$ coding includes hard clauses that describe a perfect elimination...domain constraint satisfaction problems. Modeled using hypergraphs, for example, in the analysis of received interest in domains in which instances can be easily solved in polynomial time. As such computing GHTW has which the underlying hypergraph has bounded GHTW, can be modeled using a hypergraph, the instances of that problem for problems. Specifically, whenever an NP-hard problem can be used to identify tractable instances of several different NP-hard problems. Similarly as Treewidth, GHTW can be specific undirected graphs. GHTW is an important measure mining the generalized hypertree-width (GHTW) of $k \geq 0$...minimize the number of edges assigned to any one bag, \( \beta \) is the minimum width of \( T \) induces a connected subtree of \( H \) of all generalized hypertree-decompositions of \( H \).

Notice that the first three requirements imply that $\beta$ is NP-complete for $GHTW$ of a graph is known to be NP-hard, and even determining if a graph has $GHTW$ less than $k$ is NP-complete. The $GHTW$ of a graph is known to be NP-hard, and even determining if a graph has $GHTW$ less than $k$ is NP-complete.
MaxSAT Benchmarks Encoding Optimal Causal Graphs

Antti Hyttinen and Matti Raisalo
HIIT, Department of Computer Science, University of Helsinki, Finland

Abstract—We shortly describe the problem of causal structure discovery as a combinatorial optimization problem and how a set of MaxSAT instances submitted to MaxSAT Evaluation 2017 were generated based on real-world datasets in this problem domain.

I. CAUSAL STRUCTURE DISCOVERY

Discovering causal relations between quantities of interest is an essential part of many fields of science. Information about 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) [5]. 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 [5], [6]. In the simplest scenario we consider here, we have passively observed measurement data from the system under investigation (Figure 1, left), and the aim is to learn the graph describing the causal relations working in the data generating system (Figure 1, right). In this task, the following MaxSAT-based approach currently allows for most general graph space (cycles and latent variables) and offers also high accuracy than previous approaches [2]. As a trade-off between generality and accuracy, the approach currently has a bias towards scalability, and is thus open for improvements.

A causal structure (see an example in Figure 1, left) is here a mixed graph over a set of nodes that represents measured aspects of a system (e.g., smoking habits, age, height, gender). Directed edges (symmetric) bidirected edges . Directed edges () in the graph represent causal relations (e.g., smoking causes cancer). Note that causal graphs are here allowed to include directed cycles (e.g., supply and demand), and so, between any two nodes there can be up to three edges ( ).

Bi-directed edges are used for representing the presence of exogenous or outside influence on the measured variables. More formally, a bi-directed edge denotes the presence of a “latent confounder” (e.g., particular but unidentified gene), that has a causal effect on both nodes, i.e., a structure of the form , with 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 [5], [6] for details).

Intuitively, we nd a causal graph whose reachability properties match the statistical dependence (e.g., correlation) properties of the data. So rst, for each pair of variables and each we test whether the variables are statistically dependent ( ). If not, is statistically independent and we drop the edge by removing the value helps to predict when we already know the values of variables in—or independent ( )—in the observed data (Figure 1, middle). Furthermore, we also obtain a weight describing the reliability of the decision. Finally, under some common theoretical assumptions (see [6] for details), there exists a conditional dependence in the observed data if and only if there is a so-called path between and in the causal graph structure of the true data generating system. A d-connected path given set of nodes path (repeated edges

Fig. 1: The causal structure discovery problem by example [1]
are allowed) such that every ‘collider node’ connected with two incoming edges on the path is and other nodes on the path are not in[5], [7]. For example, path

is a d-connecting path between and for

Thus in the data generated by a system with causal structure

we would observe dependence and independencies and (theoretically).

Thus according to this theory, the statistical (in)dependence (Figure 1, middle) relations 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 data 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.

The input to the causal structure discovery optimization problem is a set of reachability and separability constraints. In more detail, includes a constraint for each pair of nodes in the graph and for each conditioning setting whether the variables should be reachable or separable by d-connecting paths (for an example input, see Figure 1 middle). A weight function gives a non-negative cost for not satisfying each reachability/separability constraint. The task is to nd a causal graph (Figure 1, right) that minimizes the sum of costs of reachability/separability constraints that are unsatised:

\[
\text{(1)}
\]

where the class of causal graphs with is denoted by , and denotes that a causal graph does not satisfy a reachability/separability constraint

II. MAXSAT ENCODING

The optimization problem is computationally challenging. For obtaining good accuracy, a large number of (in)dependence constraints are needed; we use testable (in)dependence constraints ( for nodes). The d-separation condition for a solution satisfying a particular (in)dependence constraint is also quite intricate. On the other hand, this separation condition can be relatively naturally encoded declaratively as Boolean constraints. We give here an intuitive overview of the encoding of [2] in terms of MaxSAT. Each (in)dependence constraint is encoded as a unit soft clause over a distinct Boolean variable representing with weight . Additional Boolean variables are used for representing the solutions searched over, i.e., the edge relation of causal graphs. The d-connecting walks are encoded as hard clauses, linking the edge relation with the (in)dependence constraints.

III. DATASETS AND WEIGHTS

The benchmarks are based on real-world datasets often used for benchmarking exact Bayesian network structure learning algorithms. The datasets were also used recently in [4]. We considered suitable-sized subsets of the variables in the datasets, the remaining variables becoming thus latent (causal graph denition supports latent variables). We employed the BDEU score with equivalent sample size 10 to obtain independence constraint weights for this discrete data. The were

converted to integers by multiplying by 1000 and rounding. All datasets use the encoding over conditioning and marginalization of variables [2]. The number of variables was selected for each dataset such that we would get a sensible comparison among different MaxSAT solvers.

IV. FILE NAME CONVENTION

where is the name of the dataset from which the instance was generated from, is the number of observed variables (i.e., the number of nodes) in the causal graph, and is the number of samples used for generating the instance from the dataset.

REFERENCES


Abstract— We constructed the benchmark set of generalized Ising model for MAXSAT competition.

Keywords— Cluster Expansion, Ising Model, Computational Material Science

\[
\begin{align*}
H &= \sum_{\alpha} \sum_{\alpha \in \mathcal{C}} \prod_{t} \\
&= \sum_{\alpha} \sum_{\alpha} \prod_{t} \\
&= \sum_{\alpha} \sum_{\alpha} \prod_{t} \\
J_{\alpha} &= \begin{cases} 
1 & (x,y,z) \\
0 & (x,y,z,p,t) \\
\alpha & (x,y,z,p,t) \\
\end{cases}
\end{align*}
\]
\[
\prod_{(x,y,z,p) \in \alpha} s_{(x,y,z,p)} = \frac{1}{(2N+1)^3} \]

\[
\sum_{(x,y,z,p) \in F} s_{x,y,z,p} = 1 \forall (x,y,z,p) \in F
\]

\[
(x,y,z,p) \in \mathbf{F} \\
p \in \mathbf{c}(p) \\
t \in \mathbf{D}
\]

\[
t \in \{0,1,2\} \\
t = 0
\]

\[
E = \min \{J_s \hat{i}_i + J_n \hat{i}_i + J_m \hat{i}_i, J_s \hat{i}_i + J_n \hat{i}_i \}
\]

\[
\beta = \{(0,0,0,2),(0,0,0,1,2)\} \\
C = \{\alpha, \beta\}
\]

\[
(i,j,k) = (0,0,0) \\
\alpha
\]

\[
(i,j,k) = (1,1,0) \\
\alpha
\]

\[
(i,j,k) = (3,0,0) \\
\alpha
\]

\[
J_s s_{0,0,1,2} s_{1,2,0,0,1} \\
J_n s_{1,1,0,1,2} s_{1,2,0,0,1} = J_o s_{1,1,0,1,2} s_{2,5,0,0,1} \\
J_q s_{0,1,0,1,2} s_{3,0,0,1,2}
\]

\[
J_i s_{0,0,0,0,1} s_{1,2,0,0,0,1}
\]

\[
J_n s_{3,3,0,0,2} s_{3,0,0,1,2}
\]

34

Linear-programming method for obtaining effective cluster interactions in alloys from total-energy calculations: Application to the fcc Pd-V system.

Engineering Ising-XY spin-models in a triangular lattice using tunable artificial gauge fields.

Lattice-Boltzmann method for complex flows.

Effective interactions between the N-H bond orientations in lithium imide and a proposed ground-state structure.

Monte Carlo simulation of lattice models for macromolecules.

Prediction of ground-state structures and order-disorder phase transitions in II-III spinel oxides: A combined cluster-expansion method and first-principles study.

Lattice-gas cellular automata: simple models of complex hydrodynamics

A complete representation of structure-property relationships in crystals.

Direct Observation of Impurity-Induced Magnetism in a Spin-1/2 Antiferromagnetic Heisenberg Two-Leg Spin Ladder.

Generalized cluster description of multicomponent systems.

Lattice-Gas Automata for the Navier-Stokes Equation.

Crystal structure of Li x Ni 2-x O 2 and a lattice-gas model for the order-disorder transition.

Cluster expansion and optimization of thermal conductivity in SiGe nanowires.

Beitrag zur Theorie des Ferromagnetismus.

A derivation of the Ising model for the computation of phase diagrams.

Temperature-concentration phase diagram of P 2-Na x CoO 2 from first-principles calculations.

Cu-Au, Ag-Au, Cu-Ag, and Ni-Au intermetallics: First-principles study of temperature-composition phase diagrams and structures.

Structural, vibrational, and thermodynamic properties of Al-Sc alloys and intermetallic compounds.

First principles phase diagram calculations for the octahedral-interstitial system.

Configurational electronic entropy and the phase diagram of mixed-valence oxides: The case of Li x FePO 4.

Ground states of the Ising model on the Shastry-Sutherland lattice and the origin of the fractional magnetization plateaus in rare-earth-metal tetraborides.

Ground states of the lattice-gas model on the triangular lattice with nearest- and next-nearest-neighbor pairwise interactions and with three-particle interaction: Full-dimensional ground states.

Ground states of the lattice-gas model on the triangular lattice with nearest- and next-nearest-neighbor pairwise interactions and with three-particle interaction: Ground states at boundaries of full-dimensional regions.

Ground states of classical one-dimensional lattice models.

Exact Ground States of the Lattice Gas and the Ising Model on the Square Lattice.

Ground State Structure of Triangular Lattice Gas Model with up to 3rd Neighbor Interactions.

On the phase diagram of the FCC Ising model with antiferromagnetic first-
neighbour interactions.

Finding and proving the exact ground state of a generalized Ising model by convex optimization and MAX-SAT.


MaxSAT Benchmarks from the Minimum Fill-in Problem

Jeremias Berg*, Tuukka Korhonen*, and Matti Järvisalo*

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

I. PROBLEM OVERVIEW

This benchmark set consists of MaxSAT instances encoding the problem of determining the minimum fill-in for specific undirected graphs.

A cycle in an undirected graph $G = (V, E)$ is a sequence of nodes $v_1, \ldots, v_n$ such that $G$ has an edge between any $v_i$ and $v_{i+1}$ and $v_1 = v_n$. A cycle has a chord if there are 2 nodes $v_i$ and $v_j$ s.t. $j > i + 1$ and $G$ includes an edge between $v_i$ and $v_j$. The graph $G$ is chordal if any cycle of length 4 or greater has a chord.

Given a (possibly non-chordal) graph $G$, the NP-hard [5] minimum fill-in problem asks to determine the minimum number of edges that need to be added to $G$ in order to make $G$ chordal. The problem has applications in several different domains and was one of the tracks at the 2017 PACE challenge.

II. MAXSAT ENCODING

The MaxSAT encoding for minimum fill-in is adapted from the MaxSAT encoding for computing the treewidth of a graph, first proposed in [4] and further developed in [1]. Given a graph $G$ as input, the treewidth encoding includes hard clauses that describe a so-called perfect elimination ordering of $G$ and soft clauses that enforce minimization of the maximum clique size. The minimum fill-in encoding is obtained from this by instead including soft clauses that minimize the total number of added edges.

III. DATASETS IN THE BENCHMARK SET

The benchmark set consists of 28 MaxSAT instances, created from standard graph benchmarks, including coloring instances as well as Bayesian network structures from the UCI machine learning repository [3].

Before generating each MaxSAT instance, the input graph was preprocessed using standard techniques proposed for treewidth in [2]. Afterwards each separate connected component can be treated separately as the minimum fill-in of the whole graph is equal to the sum of the minimum fill-ins of the separate components. Furthermore, each component consisting only of a cycle of length $n$ can be ignored, as the minimum fill-in of such a cycle contains $n - 3$ edges. The filename convention used for the instances in the benchmark set is

```
MinFill_Rx_graphname.wcnf
```


REFERENCES

MaxSAT Benchmarks from the Minimum-Width Confidence Band Problem

Jeremias Berg∗, Emilia Oikarinen†, Matti Järvisalo∗, and Kai Puolamäki†
∗HIIT, Department of Computer Science, University of Helsinki, Finland
†Finnish Institute of Occupational Health, Finland

I. Overview

Confidence intervals are commonly used to summarize distributions over reals, to denote ranges of data, to specify accuracies of estimates of parameters, or in Bayesian settings to describe the posterior distribution. Represented with an upper and a lower bound, confidence intervals are also easy to interpret together with the data. This benchmark set contains MaxSAT instances for the NP-hard optimization task of minimizing the width of multivariate confidence intervals, i.e., the minimum-width confidence band problem. The problem as well as the MaxSAT encoding for it were originally proposed in [2].

II. Original Problem

The following definition is adapted from [2]. A confidence band is a pair of vectors \((l, u)\) s.t. \(l \leq u\) holds componentwise. The size of \(CB = (l, u)\) is \(\text{size}(CB) = \sum_{i=1}^{m} (u_i - l_i)\), i.e., the sum of the componentwise differences of \(l\) and \(u\). Given a vector \(x\) with \(m\) components and a confidence band \((l, u)\), the error of \(x\) is the number of components \(x_j\) of \(x\) that lie outside the confidence band, i.e., for which \(x_j < l_j\) or \(u_j < x_j\).

Given \(n\) vectors \(x^1, \ldots, x^n\) and integers \(k, s, t\), the minimum-width confidence band, \(\text{MWCB}(k, s, t)\), asks to find a confidence band of minimum size for which (i) the number of vectors \(x^i\) with error larger than \(s\) is at most \(k\), and (ii) at most \(t\) vectors lie outside the confidence band at any fixed component. More formally, any \(CB^* \in \text{arg min}_i \text{size}(CB)\) over those \(CB = (l, u)\) for which (i) \(\sum_{i=1}^{n} I[\text{ERROR}(x^i, CB) > s] \leq k\) and (ii) \(\sum_{i=1}^{n} I[x^i_j < l_j \lor x^i_j > u_j] \leq t\) for all \(1 \leq j \leq m\), is a solution to \(\text{MWCB}(k, s, t)\).

III. MaxSAT Encoding

For an exact description of the MaxSAT encoding for \(\text{MWCB}(k, s, t)\), we refer the reader to [2]. From the perspective of the MaxSAT evaluation, an interesting characteristic of the benchmarks in the set is that all instances consist only of binary clauses and cardinality constraints encoded using cardinality networks [1].

IV. Datasets in the Benchmark Set

The benchmark set consists of 222 benchmarks used in [2] and originate from 3 different datasets:

- Milan temperature data (milan), in the form of the max-temp-milan dataset from [3], contains average monthly maximum temperatures for a station located in Milan for the years 1763–2007. The full dataset contains 245 vectors, each with 12 components.
- UCI-Power data (power) consists individual household electric power consumption data1, and is obtained from the UCI machine learning repository [4]. The whole dataset contains 1417 vectors, each with 24 components.
- Heartbeat data (mitdb), in form of the preprocessed datasets heartbeat-normal and heartbeat-pvc from [3], contain annotated 30-minute records of normal and abnormal heartbeats [5]. There are in total 1507 vectors in heartbeat-normal and 520 vectors in heartbeat-pvc both with 253 components.

As in [2], the MaxSAT benchmarks are based on data sampled from these datasets. The naming convention of the WCNF benchmark instance files is

\[\text{MinWidthCB\_dataset\_n\_m\_k\_s\_t\_w.wcnf}\]

where “dataset” is the name of the underlying dataset, \(n\) is the number of vectors and \(m\) the number of components in the sampled datasets, and \(k, s, t\) are the values of the input parameters to \(\text{MWCB}(k, s, t)\). For each of the MaxSAT benchmark instances, optimal cost corresponds to the size of the minimum-width confidence bands. See [2] for more details.

REFERENCES

Solver Index

LMHS, 16
Loandra, 13

MaxHS, 8
Maxino, 10
MaxRoster, 12
MSUSorting, 15

Open-WBO, 17

QMaxSAT1702, 18
QMaxSATuc, 18
Benchmark Index

Answer set programming, 27
Argumentation dynamics, 23

Causal structure discovery, 31
Cluster expansion, 33
CSS refactoring, 20

Development assurance level, 28
Generalized hypertreewidth, 22

Minimum fill-in, 37
Minimum-width confidence bands, 38

RNA alignment, 29
Seating Arrangement, 25
Author Index

Alviano, Mario, 10
Asín Achá, Roberto, 15

Bacchus, Fahiem, 8
Berg, Jeremias, 13, 16, 22, 37, 38

Hague, Matthew, 20
Huang, Wenzuan, 33
Hyttinen, Antti, 31

Järvisalo, Matti, 13, 16, 22, 23, 31, 37, 38
Jahren, Eivind, 15
Janota, Mikolas, 17
Joshi, Saurabb, 17

Korhonen, Tuukka, 13, 16, 37
Koshimura, Miyuki, 18

Lin, Anthony Widjaja, 20
Lodha, Neha, 22
Lynce, Ines, 17

Manquinho, Vasco, 17
Martins, Ruben, 17, 25, 27–29

Niskanen, Andreas, 23
Oikarinen, Emilia, 38
Puolamäki, Kai, 38

Saikko, Paul, 16
Sherry, Justine, 25
Sugawara, Takayuki, 12
Szeider, Stefan, 22

Terra-Neves, Miguel, 17
Uemura, Naoki, 18

Wallner, Johannes P., 23
Zha, Aolong, 18