MaxSAT Evaluation 2017

Solver and Benchmark Descriptions

Carlos Ansotegui, Fahiem Bacchus, Matti Järvisalo, and Ruben Martins *(editors)*

University of Helsinki
Department of Computer Science
Series of Publications B
Report B-2017-2

Helsinki 2017

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 heterogenous 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

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/lsimon/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 an 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

[1] Bacchus, F., Hyttinen, A., Järvisalo, M., Saikko, P.: Reduced cost fixing in maxsat. In: Proc. CP. p. in press (2017)

[2] Davies, J., Bacchus, F.: Solving MAXSAT by solving a sequence of simpler SAT instances. In: Proc. CP. Lecture Notes in Computer Science, vol. 6876, pp. 225–239. Springer (2011)

[3] Davies, J., Bacchus, F.: Exploiting the power of MIP solvers in MaxSAT. In: Proc. SAT. Lecture Notes in Computer Science, vol. 7962, pp. 166–181. Springer (2013)

[4] Davies, J.: Solving MAXSAT by Decoupling Optimization and Satisfaction. Ph.D. thesis, University of Toronto (2013), http://www.cs.toronto.edu/~jdavies/Davies_Jessica_E_201311_PhD_thesis.pdf

[5] Davies, J., Bacchus, F.: Postponing optimization to speed up MAXSAT solving. In: Proc. CP. Lecture Notes in Computer Science, vol. 8124, pp. 247–262. Springer (2013)

[6] Saikko, P., Berg, J., Järvisalo, M.: LMHS: A SAT-IP hybrid MaxSAT solver. In: Proc. SAT. Lecture Notes in Computer Science, vol. 9710, pp. 539–546. Springer (2016)

[7] Saikko, P.: Re-implementing and Extending a Hybrid SAT-IP Approach to Maximum Satisfiability. Master's thesis, University of Helsinki (2015), http://hdl.handle.net/10138/159186

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 ϕ is the clause $\phi \vee \neg x$, where x is a variable not occurring elsewhere; moreover, the weight associated with clause ϕ is associated with the soft literal x. 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, based on the 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:

$$x_0 + x_1 + x_2 + x_3 + \neg y_1 + \neg y_2 + \neg y_3 \geq 3$$
$$y_1 \rightarrow y_2 \quad y_2 \rightarrow y_3$$

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):

$$x_0 + x_1 + x_2 + x_3 + \neg y_1 \geq 3$$
$$x_0 + x_1 + x_2 + x_3 + \neg y_2 \geq 2$$
$$x_0 + x_1 + x_2 + x_3 + \neg y_3 \geq 1$$

Concerning PMRES, it introduces the following constraints:

$$x_0 \vee x_1 \vee \neg y_1 \quad z_1 \leftrightarrow x_0 \wedge x_1$$
$$z_1 \vee x_2 \vee \neg y_2 \quad z_2 \leftrightarrow z_1 \wedge x_2$$
$$z_2 \vee x_3 \vee \neg y_3$$

which are essentially equivalent to the following constraints:

$$x_0 + x_1 + \neg z_1 + \neg y_1 \geq 2 \quad z_1 \rightarrow y_1$$
$$z_1 + x_2 + \neg z_2 + \neg y_2 \geq 2 \quad z_2 \rightarrow y_2$$
$$z_2 + x_3 \quad + \neg y_3 \geq 1$$

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 $2 \cdot (k+1)$. ONE, which is essentially a smart encoding of OLL, is the special case for $k = \infty$, and PMRES is the special case for $k = 1$. For the example unsatisfiable core, another possibility is $k = 2$, which would results in the following constraints:

$$x_0 + x_1 + x_2 + \neg z_1 + \neg y_1 + \neg y_2 \geq 3 \quad z_1 \rightarrow y_1 \quad y_1 \rightarrow y_2$$
$$z_1 + x_3 \quad + \neg y_3 \quad \geq 1$$

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

The analysis of unsatisfiable core is preceded by a *shrink* procedure [1]. Specifically, a reiterated progression search is performed on the unsatisfiable core returned by the SAT solver. Such a procedure significantly reduce the size of the unsatisfiable core, even if it does 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

[1] Mario Alviano and Carmine Dodaro. Anytime answer set optimization via unsatisfiable core shrinking. *TPLP*, 16(5-6):533–551, 2016.

[2] Mario Alviano, Carmine Dodaro, and Francesco Ricca. A maxsat algorithm using cardinality constraints of bounded size. In Qiang Yang and Michael Wooldridge, editors, *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 2677–2683. AAAI Press, 2015.

[3] Benjamin Andres, Benjamin Kaufmann, Oliver Matheis, and Torsten Schaub. Unsatisfiability-based optimization in clasp. In *28th International Conference on Logic Programming*, pages 211–221, Budapest, Hungary, September 2012.

[4] Carlos Ansótegui, Maria Luisa Bonet, and Jordi Levy. Solving (weighted) partial maxsat through satisfiability testing. In *SAT 2009*, pages 427–440, Swansea, UK, June 2009. Springer.

[5] Carlos Ansótegui, Maria Luisa Bonet, and Jordi Levy. SAT-based MaxSAT algorithms. *Artificial Intelligence*, 196(0):77–105, March 2013.

[6] Josep Argelich, Inês Lynce, and João P. Marques Silva. On solving boolean multilevel optimization problems. In *21st International Joint Conference on Artificial Intelligence*, pages 393–398, Pasadena, California, July 2009. IJCAI Organization.

[7] Gilles Audemard and Laurent Simon. Predicting learnt clauses quality in modern SAT solvers. In *21st International Joint Conference on Artificial Intelligence*, pages 399–404, Pasadena, California, July 2009. IJCAI Organization.

[8] Nina Narodytska and Fahiem Bacchus. Maximum satisfiability using core-guided MaxSAT resolution. In *Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 2717–2723, Québec City, Canada, July 2014. AAAI Press.

MaxRoster:Solver Description

Takayuki Sugawara
Sugawara Systems
3-24-13 Kitanakayama Izumi-ku Sendai-City,Japan
nurse-support@sugawaras-systems.com

***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 engine,one is local search solver Ramp and another is MapleSAT with CHB. First, Ramp is used 6sec and then 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

[1] Yi Fan, Zongjie Ma, Kaile Su, Abdul Sattar,Chengqian Li, “Ramp: A Local Search Solver based on Make-positive Variables “ MaxSAT Evaluation 2016.

[2] Jia Hui Liang, Vijay Ganesh, Pascal Poupart, Krzysztof Czarnecki: Exponential Recency Weighted Average Branching Heuristic for SAT Solvers. AAAI 2016: 3434-3440

[3] A. Morgado, A. Ignatiev, J. Marques-Silva: MSCG: Robust Core-Guided MaxSAT Solving. Special Issue on SAT 2014 Competitions and Evaluations. JSAT Volume 9, 2014.

[4] Martins, R., Joshi, S., Manquinho, V.M., Lynce, I.: Incremental cardinality constraints for MaxSAT. In: CP (2014).

Loandra: PMRES Extended with Preprocessing Entering MaxSAT Evaluation 2017

Jeremias Berg, Tuukka Korhonen, and Matti Järvisalo
HIIT, Department of Computer Science, University of Helsinki, Finland

I. Preliminaries

We briefly overview the Loandra MaxSAT solver as it participated in the 2017 MaxSAT evaluation. We assume familiarity with conjunctive normal form (CNF) formulas and weighted partial maximum satisfiability (MaxSAT). Treating a CNF formula as a set of clauses a MaxSAT instance consists of two CNF formulas, the hard clauses F_h and the soft clauses F_s, as well a weight function $w\colon F_s \to \mathbb{N}$.

Loandra makes extensive use of SAT-based preprocessing using labels [3], [4]. In order to enable sound application of most SAT-based preprocessing techniques for MaxSAT, each soft clause C is first extended with a fresh label variable l_C. Afterwards the preprocessor is invoked on the clauses in $F_h \cup \{C \vee l_C \mid C \in F_s\}$. During execution the preprocessor is forbidden from resolving on the added labels. Afterwards, the preprocessed instance is converted back to standard MaxSAT by treating all clauses in the preprocessor as hard and introducing a soft clause $(\neg l_C)$ with weight $w(C)$ for each added label.

II. Structure and execution of Loandra

The architecture of Loandra consists of two closely interleaved parts; the *solver* and the *preprocessor*. The solver is a reimplementation of the PMRES MaxSAT algorithm [15] extended with weight-aware core extraction (WCE) as described in [6]. The preprocessor is the recently proposed tool MaxPre [11], modified to support addition of clauses.

In more detail, whenever invoked on an MaxSAT instance (F_h, F_s, w) Loandra first preprocesses the input instance as described in [11]. Afterwards the preprocessed instance is extracted from the preprocessor and given to the solver. When initializing the solver, we follow [5] and do not introduce the soft clauses of form $(\neg l_C)$, but instead reuse the literals as assumption variables to be used in core extraction. Then the solver is invoked on the preprocessed instance. Except for the base algorithm, Loandra also uses stratification and clause hardening [1] as well as clause cloning through assumptions and reusing assumption variables as relaxation variables [6]. This guarantees that the working formula is only modified by adding clauses to it, making it possible to keep the state of the internal SAT solver throughout the solving process. During execution, all cardinality constraints added due to core relaxation are also added to the preprocessor as well. When the working formula is sufficiently modified, the the execution is switched back to the preprocessor which attempts to further simplify the modified formula, i.e. the original clauses with some clauses hardened and the new cardinality constraints. If the preprocessor is successful, the solver is reinitialized on the modified formula. Loandra terminates whenever the solver terminates. At this point the optimal model for the original formula can be reconstructed from the preprocessor.

III. Details on the competition builds

There are three version of Loandra competing in the 2017 MaxSAT evaluation.

- LOANDRA$_I$, which follows the description given above.
- LOANDRA$_P$, which only invokes its preprocessor once and then runs the solver on the preprocessed instance.
- LOANDRA$_S$, which only uses its solver, not invoking the preprocessor at all.

These solvers are built on top of the open source Open-WBO system [13], [14] and use Glucose 3.0 [2] as the internal SAT solver. All preprocessing calls are done with label matching turned off, with the SKIPTECHNIQUE parameter set to 20, and using a technique loop with blocked clause elimination [10], unit propagation, bounded variable elimination [8], subsumption elimination, self-subsuming resolution [8], [9], [12] as well as group-subsumed label elimination [7], [11] and binary core removal [11]. See [11] for more details on the settings of MaxPre. The additional preprocessing step is attempted whenever more than 500 clauses have been hardened since the preprocessing attempt.

IV. Compilation and Usage

Building and using Loandra resembles building and using Open-WBO. A statically linked version of Loandra in release mode can be built from the code by first running MAKE LIB in the maxpre subfolder and then MAKE RS in the base folder. One significant difference to Open-WBO is the need of C++11 features for building Loandra.

After building, Loandra can be invoked from the terminal. Except for the formula file, Loandra accepts a number of command line arguments; the flag "-inpr" enables execution following LOANDRA$_I$, the flag "-pre" enables execution following LOANDRA$_P$ and the flag "-printM" prints out the optimal model of the instance, and not only its cost. The rest of the flags resemble the flags accepted by Open-WBO; invoke ./loandra_static –help-verb for more information.

References

[1] C. Ansótegui, M. L. Bonet, J. Gabàs, and J. Levy, "Improving SAT-based weighted MaxSAT solvers," in *Proc. CP*, ser. Lecture Notes in Computer Science, vol. 7514. Springer, 2012, pp. 86–101.

[2] G. Audemard, J.-M. Lagniez, and L. Simon, "Improving Glucose for incremental SAT solving with assumptions: Application to MUS extraction," in *Proc. SAT*, ser. Lecture Notes in Computer Science, vol. 7962. Springer, 2013, pp. 309–317.
[3] A. Belov, M. Järvisalo, and J. Marques-Silva, "Formula preprocessing in MUS extraction," in *Proc. TACAS*, ser. Lecture Notes in Computer Science, vol. 7795. Springer, 2013, pp. 108–123.
[4] A. Belov, A. Morgado, and J. Marques-Silva, "SAT-based preprocessing for MaxSAT," in *Proc. LPAR-19*, ser. Lecture Notes in Computer Science, vol. 8312. Springer, 2013, pp. 96–111.
[5] J. Berg, P. Saikko, and M. Järvisalo, "Improving the effectiveness of SAT-based preprocessing for MaxSAT," in *Proc. IJCAI*. AAAI Press, 2015, pp. 239–245.
[6] J. Berg and M. Järvisalo, "Weight-aware core extraction in SAT-based MaxSAT solving," in *Proc. CP*, ser. Lecture Notes in Computer Science, 2017, to appear.
[7] J. Berg, P. Saikko, and M. Järvisalo, "Subsumed label elimination for maximum satisfiability," in *Proc. ECAI*, ser. Frontiers in Artificial Intelligence and Applications, vol. 285. IOS Press, 2016, pp. 630–638.
[8] N. Eén and A. Biere, "Effective preprocessing in SAT through variable and clause elimination," in *Proc. SAT*, ser. Lecture Notes in Computer Science, vol. 3569. Springer, 2005, pp. 61–75.
[9] J. Groote and J. Warners, "The propositional formula checker Heer-Hugo," *Journal of Automated Reasoning*, vol. 24, no. 1/2, pp. 101–125, 2000.
[10] M. Järvisalo, A. Biere, and M. Heule, "Blocked clause elimination," in *Proc. TACAS*, ser. Lecture Notes in Computer Science, vol. 6015. Springer, 2010, pp. 129–144.
[11] T. Korhonen, J. Berg, P. Saikko, and M. Järvisalo, "MaxPre: An extended MaxSAT preprocessor," in *Proc. SAT*, ser. Lecture Notes in Computer Science, S. Gaspers and T. Walsh, Eds., 2017, to appear.
[12] K. Korovin, "iProver – an instantiation-based theorem prover for first-order logic," in *Proc. IJCAR*, ser. Lecture Notes in Computer Science, vol. 5195. Springer, 2008, pp. 292–298.
[13] R. Martins, S. Joshi, V. Manquinho, and I. Lynce, "Incremental cardinality constraints for MaxSAT," in *Proc. CP*, ser. Lecture Notes in Computer Science, vol. 8656. Springer, 2014, pp. 531–548.
[14] R. Martins, V. Manquinho, and I. Lynce, "Open-WBO: A modular MaxSAT solver," in *Proc. SAT*, ser. Lecture Notes in Computer Science, vol. 8561. Springer, 2014, pp. 438–445.
[15] N. Narodytska and F. Bacchus, "Maximum satisfiability using core-guided MaxSAT resolution," in *Proc. AAAI*. AAAI Press, 2014, pp. 2717–2723.

The MSUSorting MaxSAT solver

Eivind Jahren, Roberto Asín Achá

1 Solver Description

The MSUSorting solver builds on the work by Martins et al. [1] of leveraging incremental SAT solvers for the MSU3 algorithm [2], and the Totalizer encoding. The MSUSorting solver extends this work to the the mixed encoding by Abio et al. [3] and the MSU4 algorithm [4], using the glucose-syrup SAT solver [5].

2 Incremental MSU3 and MSU4 algorithms

The MSU3 algorithm is an unsatisfiable core based algorithm, akin to the Fu-Malik algorithm [6], [7], which uses a cardinality encoding to bound the number of unsatisfied clauses. The main difference for the incremental version of the algorithm is enabling the cardinality encoding to be updated. Martins et al. [1] uses the totalizer encoding [8] for this purpose.

We extend this work with new updatable cardinality encodings based on cardinality networks [9] and parametric cardinality networks [3]. We also use these updatable cardinality encodings to make the MSU4 algorithm [4] incremental. See [10] for details.

3 Updatable Cardinality Encodings

We use a generic framework for making updatable cardinality encodings which we call *delayed variables* [10]. This framework enables us to make updatable versions of the totalizer, cardinality network, and mixed encoding. A delayed variable is one which is not yet introduced to the SAT solver, so any clause it occurs in is not given to the SAT solver until the variable is undelayed. This allows delayed variables to be substituted without changing the formula given to the SAT solver.

4 Selecting strategies

The solver has two tweakable parameters: whether to choose MSU3 or MSU4, and whether to choose the cardinality networks or the totalizer encoding in the mixed encoding [3]. We found that many benchmarks can quickly be solved with either MSU3 or MSU4 but not by both. Since MSU3 and MSU4 share internal state, we simply switch to MSU4 once a time limit (500s) has been reached. This ensures that some time is spent solving the problem with both algorithms, and progress made with MSU3 is reused for MSU4.

The mixed encoding combines the totalizer and cardinality network encoding. Using the totalizer encoding encoding means fewer variables, while the cardinality network encoding has fewer clauses. We found that when using the mixed encoding with the MSU3 & MSU4 algorithms, the encoding should favor the totalizer encoding heavily. The solver is given a limit on the of number of extra clauses beyond the minimal amount, and uses totalizer as long as the budget is not exceeded. If the limit is exceeded, cardinality network is used where it saves the most clauses per additional variable until the limit is satisfied. The limit is quite generous: eight times the number of clauses in the input formula.

References

[1] R. Martins, S. Joshi, V. Manquinho, and I. Lynce, "Incremental cardinality constraints for maxsat," in *Principles and Practice of Constraint Programming*, B. O'Sullivan, Ed. Springer, 2014, pp. 531–548.

[2] J. Marques-Silva and J. Planes, "On using unsatisfiability for solving maximum satisfiability," *arXiv preprint arXiv:0712.1097*, 2007.

[3] I. Abío, R. Nieuwenhuis, A. Oliveras, and E. Rodríguez-Carbonell, "A parametric approach for smaller and better encodings of cardinality constraints," in *Principles and Practice of Constraint Programming*, C. Schulte, Ed. Springer, 2013, pp. 80–96.

[4] J. Marques-Sila and J. Planes, "Algorithms for maximum satisfiability using unsatisfiable cores," in *Advanced Techniques in Logic Synthesis, Optimizations and Applications*, K. Gulati, Ed. Springer, 2011, pp. 171–182.

[5] G. Audemard and L. Simon, "Predicting learnt clauses quality in modern sat solvers." in *Proceedings of the Twenty-First International Joint Conference On Artificial Intelligence*, vol. 3. IJCAI, 2009, pp. 399–404.

[6] Z. Fu and S. Malik, "On solving the partial max-sat problem," in *International Conference on Theory and Applications of Satisfiability Testing*. Springer, 2006, pp. 252–265.

[7] Z. Fu, *Extending the power of Boolean satisfiability solvers: Techniques and applications*. Princeton University, 2007.

[8] O. Bailleux and Y. Boufkhad, "Efficient cnf encoding of boolean cardinality constraints," in *Principles and Practice of Constraint Programming*, F. Rossi, Ed. Springer, 2003, pp. 108–122.

[9] R. Asín, R. Nieuwenhuis, A. Oliveras, and E. Rodríguez-Carbonell, "Cardinality networks: a theoretical and empirical study," *Constraints*, vol. 16, no. 2, pp. 195–221, 2011.

[10] E. Jahren and R. Asín Achá, "Resizing cardinality constraints for maxsat," *Manuscript submitted for publication*, 2017.

LMHS in MaxSAT Evaluation 2017

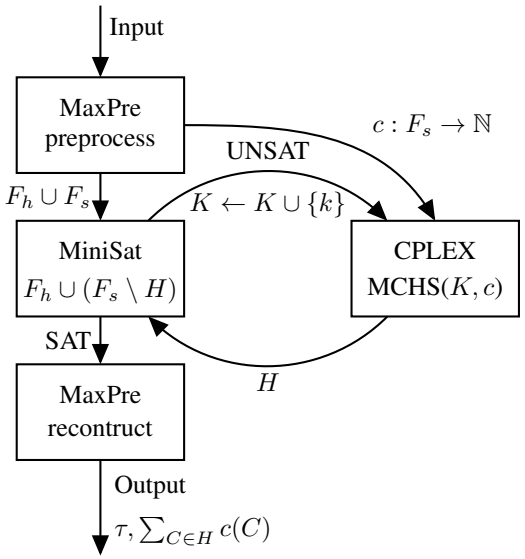
Paul Saikko and Tuukka Korhonen and Jeremias Berg and Matti Järvisalo
HIIT, Department of Computer Science
University of Helsinki, Finland

***Abstract*—We describe recent updates to the LMHS MaxSAT solver, submitted to the 2017 MaxSAT Evaluation.**

I. Introduction

An updated version of the LMHS MaxSAT solver [1] is submitted to the 2017 MaxSAT evaluation. This version includes many incremental updates and bugfixes. Major improvements include the addition of a purpose–built MaxSAT preprocessor MaxPre, and LP-based reduced–cost fixing for forcing soft clauses during search.

II. Implicit Hitting Set Algorithm



LMHS implements the implicit hitting set algorithm [2] for MaxSAT [3], [4]. We apply MaxSAT preprocessing to simplify the problem before solving. After preprocessing, the MaxSAT cost function c is input to the optimizer and the CNF formula (hard clauses F_h and soft clauses F_s) is given to the satisfiability checker. MiniSat 2.2 [5] is used as the satisfiability checker, and CPLEX 12.7 [6] as the optimizer.

In short, the implicit hitting set loop alternates between checking the satisfiability of the formula (excluding a hitting set H) to find an unsatisfiable core k. Unsatisfiable cores are accumulated in a set K, for which the optimizer finds a minimum–cost hitting set wrt. the cost function c.

Upper bounds on the optimal solution cost (feasible solutions) are found during search LMHS's core minimization proceduce and non–optimal hitting set phase (not pictured). Lower bounds are proved by the optimizer.

III. LCNF Preprocessing

LMHS has been updated with a new MaxSAT preprocessor, MaxPre [7]. MaxPre implements a range of well-known and recent SAT-based preprocessing techniques as well as MaxSAT-specific techniques that make use of weights of soft clauses. MaxSAT specific techniques include group detection, label matching, group-subsumed label elimination, and binary core removal. Tight integration with MaxPre's C++ API eliminates unnecessary I/O overhead. LMHS solves the preprocessed instance directly as a labelled CNF formula [8], which avoids the addition of new auxiliary variables to soft clauses.

IV. Reduced-cost Fixing

We implement recent reduced–cost fixing techniques for MaxSAT [9]. LP-based reduced-cost fixing together with bounds allow for some soft clauses to be hardened or relaxed during search, simplifying the problem. This inexpensive technique requires only that the LP relaxation of the hitting set IP is solved once per iteration.

V. Incomplete Track

New for 2017 we also submit LMHS to the incomplete track. The large number of feasible solutions found during search means that LMHS can provide a solution at any point during the search, after verifying that one exists.

VI. Availability

LMHS is open source and available at https://www.cs.helsinki.fi/group/coreo/lmhs/. MaxPre is available as a standalone preprocessor at https://www.cs.helsinki.fi/group/coreo/maxpre/.

References

[1] P. Saikko, J. Berg, and M. Järvisalo, "LMHS: A SAT-IP hybrid MaxSAT solver," in *Proc. SAT*, ser. LNCS, vol. 9710. Springer, 2016, pp. 539–546.
[2] R. M. Karp, "Implicit hitting set problems and multi-genome alignment," in *Proc. CPM*, ser. LNCS, vol. 6129. Springer, 2010, p. 151.
[3] J. Davies and F. Bacchus, "Solving MAXSAT by solving a sequence of simpler SAT instances," in *Proc. CP*, ser. LNCS, vol. 6876. Springer, 2011, pp. 225–239.
[4] ——, "Postponing optimization to speed up MAXSAT solving," in *Proc. CP*, ser. LNCS, vol. 8124. Springer, 2013, pp. 247–262.
[5] N. Eén and N. Sörensson, "An extensible SAT-solver," in *Proc. SAT*, ser. LNCS, vol. 2919. Springer, 2003, pp. 502–518.
[6] IBM, "CPLEX Optimizer," 2017, http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/.
[7] T. Korhonen, J. Berg, P. Saikko, and M. Järvisalo, "MaxPre: An extended MaxSAT preprocessor," in *Proc. SAT*, ser. LNCS. Springer, 2017, To appear.
[8] J. Berg, P. Saikko, and M. Järvisalo, "Improving the effectiveness of sat-based preprocessing for MaxSAT," in *Proc. IJCAI*, 2015, pp. 239–245.
[9] F. Bacchus, M. Järvisalo, P. Saikko, and A. Hyttinen, "Reduced cost fixing in MaxSAT," in *Proc. CP*, ser. LNCS. Springer, 2017, To appear.

Open-WBO in MaxSAT Evaluation 2017

Ruben Martins[†], Miguel Terra-Neves[⋆], Saurabh Joshi[‡], Mikoláš Janota[⋆], Vasco Manquinho[⋆], Inês Lynce[⋆]
[†]University of Texas at Austin / Carnegie Mellon University, USA
[⋆]INESC-ID / Instituto Superior Técnico, Universidade de Lisboa, Portugal
[†]Indian Institute of Technology, Hyderabad, India

I. Introduction

Open-WBO is an open source MaxSAT solver that started as a spin-off of WBO [1]. Open-WBO implements a variety of algorithms for solving Maximum Satisfiability (MaxSAT) and Pseudo-Boolean (PB) formulas. The algorithms used in 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 4.1 [3]. The key novelties of Open-WBO are: (i) incremental MaxSAT solving [4] and (ii) partitioning-based MaxSAT solving [5], [6], [7]. Open-WBO is particularly efficient for partial MaxSAT and has been one of the best solvers in the MaxSAT Evaluations of 2014, 2015 and 2016. Two versions of Open-WBO were submitted to the MaxSAT Evaluation 2017: LSU and RES. The remainder of this document describes the algorithms and encodings used in each version.

II. Open-WBO 2017: LSU Version

The LSU version is based on a linear search algorithm SAT-UNSAT [8] with lexicographical optimization for weighted problems [9]. This algorithm works by performing a sequence of calls to a SAT solver and refining an upper bound μ on the number of unsatisfied soft clauses. To restrict μ at each iteration, we need to encode a cardinality constraint (pseudo-Boolean constraint) for unweighted (weighted) problems into CNF. The LSU version versions uses the Modulo Totalizer encoding [10] for cardinality constraints and the Generalized Totalizer encoding (GTE) [11] for pseudo-Boolean constraints.

III. Open-WBO 2017: RES Version

The RES version is based on the unsatisfiability-based algorithms MSU3 [12] and OLL [13]. These algorithms work by iteratively refining a lower bound λ on the number of unsatisfied soft clauses until an optimum solution is found. Both MSU3 and OLL use the Totalizer encoding for incremental MaxSAT solving [4]. For unweighted MaxSAT, we extended the incremental MSU3 algorithm [4] with resolution-based partitioning techniques [7]. We represent a MaxSAT formula using a resolution-based graph representation and iteratively join partitions by using a proximity measure extracted from the graph representation of the formula. The algorithm ends when only one partition remains and the optimal solution is found. Since the partitioning of some MaxSAT formulas may be unfeasible or not significant, we heuristically choose to run MSU3 with or without partitions. In particular, we do not use partition-based techniques when one of the following criteria is met: (i) the formula is too large ($>$ 1,000,000 clauses), (ii) the ratio between the number of partitions and soft clauses is too high (> 0.8), or (iii) the sparsity of the graph is too small (< 0.04). Currently, Open-WBO only supports partition-based techniques for unweighted problems. For weighted MaxSAT, we use the OLL MaxSAT algorithm [13].

IV. Availability

The first release of Open-WBO is available under a MIT license at *http://sat.inesc-id.pt/open-wbo/*. The second release of Open-WBO is available under a MIT license in Github at *https://github.com/sat-group/open-wbo*. This version includes the partitioning techniques that made Open-WBO one of the best solvers for partial MaxSAT in the MaxSAT Evaluations of 2015 and 2016. To contact the authors please send an email to: `open-wbo@sat.inesc-id.pt`.

Acknowledgments

We would like to thank Laurent Simon and Gilles Audemard for allowing us to use Glucose in the MaxSAT Evaluation.

References

[1] V. Manquinho, J. Marques-Silva, and J. Planes, "Algorithms for Weighted Boolean Optimization," in *SAT*. Springer, 2009, pp. 495–508.
[2] N. Eén and N. Sörensson, "An Extensible SAT-solver," in *SAT*. Springer, 2003, pp. 502–518.
[3] G. Audemard and L. Simon, "Predicting Learnt Clauses Quality in Modern SAT Solvers," in *IJCAI*, 2009, pp. 399–404.
[4] R. Martins, S. Joshi, V. Manquinho, and I. Lynce, "Incremental Cardinality Constraints for MaxSAT," in *CP*. Springer, 2014, pp. 531–548.
[5] R. Martins, V. Manquinho, and I. Lynce, "On Partitioning for Maximum Satisfiability," in *ECAI*. IOS Press, 2012, pp. 913–914.
[6] R. Martins, V. M. Manquinho, and I. Lynce, "Community-based partitioning for maxsat solving," in *SAT*. Springer, 2013, pp. 182–191.
[7] M. Neves, R. Martins, M. Janota, I. Lynce, and V. M. Manquinho, "Exploiting Resolution-Based Representations for MaxSAT Solving," in *SAT*. Springer, 2015, pp. 272–286.
[8] D. Le Berre and A. Parrain, "The Sat4j library, release 2.2," *Journal on Satisfiability, Boolean Modeling and Computation*, vol. 7, no. 2-3, pp. 59–6, 2010.
[9] J. Marques-Silva, J. Argelich, A. Graça, and I. Lynce, "Boolean lexicographic optimization: algorithms & applications," *Annals of Mathematics and Artificial Intelligence*, vol. 62, no. 3-4, pp. 317–343, 2011.
[10] T. Ogawa, Y. Liu, R. Hasegawa, M. Koshimura, and H. Fujita, "Modulo Based CNF Encoding of Cardinality Constraints and Its Application to MaxSAT Solvers," in *ICTAI*. IEEE, 2013, pp. 9 – 17.
[11] S. Joshi, R. Martins, and V. M. Manquinho, "Generalized Totalizer Encoding for Pseudo-Boolean Constraints," in *CP*. Springer, 2015, pp. 200–209.
[12] J. Marques-Silva and J. Planes, "On Using Unsatisfiability for Solving Maximum Satisfiability," *CoRR*, 2007.
[13] A. Morgado, C. Dodaro, and J. Marques-Silva, "Core-Guided MaxSAT with Soft Cardinality Constraints," in *CP*. Springer, 2014, pp. 564–573.

QMaxSAT1702 and QMaxSATuc

Naoki Uemura, Aolong Zha, and Miyuki Koshimura
Graduate School/Faculty of Information Science and Electrical Engineering, Kyushu University
744 Motooka, Nishi-ku, Fukuoka, Japan

QMaxSAT is a SAT-based MaxSAT solver which uses CNF encoding of Pseudo-Boolean (PB) constraints [1]. The current version is obtained by adapting a CDCL based SAT solver Glucose 3.0 [2], [3]. There are two main types among SAT-based MaxSAT algorithms: core-guided and model-guided. QMaxSAT follows the model-guided approach.

Let $\phi = \{(C_1, w_1), \ldots, (C_m, w_m), C_{m+1}, \ldots, C_{m+m'}\}$ be a MaxSAT instance where C_i is a soft clause having a weight w_i $(i = 1, \ldots, m)$ and C_{m+j} is a hard clause $(j = 1, \ldots, m')$. A new blocking variable b_i is added to each soft clause $C_i (i = 1, \ldots, m)$. Solving the MaxSAT problem for ϕ is reduced to find a SAT model of $\phi' = \{C_1 \vee b_1, \ldots, C_m \vee b_m, C_{m+1}, \ldots, C_{m+m'}\}$ which minimizes $\sum_{i=1}^{m} w_i \cdot b_i$.

QMaxSAT leaves the manipulation of PB constraints $\sum_{i=1}^{m} w_i \cdot b_i < k$ to Glucose by encoding them into SAT. Several encodings have been proposed so far. We adopt Totalizer [4], Binary Adder [5], Modulo Totalizer [6], and Weighted Totalizer [7] for encodings PB constraints. The last one is essentially the same as Generalized Totalizer [8]. Which encoding is used depends on the total $\sum_{i=1}^{m} w_i$ of weights of all soft clauses and k.

We introduce a new SAT encoding for PB constrains, called Mixed Radix Weighted Totalizer [9] into QMaxSAT1702. This encoding is an extension of Weighted Totalizer, incorporating the idea of mixed radix base [10].

QMaxSATuc is a hybrid solver between core-guided and model-guided while it mainly follows model-guided approach. QMaxSATuc runs in two modes: core-guided and model-guided. QMaxSATuc alternates these modes. QMaxSATuc performs core-guided mode with a set B of blocking variables. B is initialized to $\{b_1, \ldots, b_m\}$, i.e. the set of all blocking variables.

In core-guided mode, all blocking variables in B are negated. These negated variables are passed to Glucose as assumptions. Glucose treats each literal in assumptions as an unit clause. Glucose returns a subset of assumptions used in the UNSAT proof. Each soft clause corresponding to a blocking variable in the subset can be regarded as an element in the unsat-core of ϕ'. We make a clause having all blocking variables in the subet as literals, and add it to the clause database in order to eliminate the core. Thus, we minic the core-guided approach. We also subtract all the blocking variables in the subet from B. In model-guided mode, nothing is passed to Glucose as assumptions. This is the normal mode of QMaxSAT .

References

[1] M. Koshimura, T. Zhang, H. Fujita, and R. Hasegawa, "Qmaxsat: A partial max-sat solver," *JSAT*, vol. 8, no. 1/2, pp. 95–100, 2012.

[2] G. Audemard and L. Simon, "Predicting learnt clauses quality in modern SAT solvers," in *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*, C. Boutilier, Ed., 2009, pp. 399–404.

[3] N. Eén and N. Sörensson, "An extensible sat-solver," in *Theory and Applications of Satisfiability Testing, 6th International Conference, SAT 2003. Santa Margherita Ligure, Italy, May 5-8, 2003 Selected Revised Papers*, ser. Lecture Notes in Computer Science, E. Giunchiglia and A. Tacchella, Eds., vol. 2919. Springer, 2003, pp. 502–518.

[4] O. Bailleux and Y. Boufkhad, "Efficient CNF encoding of boolean cardinality constraints," in *Principles and Practice of Constraint Programming - CP 2003, 9th International Conference, CP 2003, Kinsale, Ireland, September 29 - October 3, 2003, Proceedings*, ser. Lecture Notes in Computer Science, F. Rossi, Ed., vol. 2833. Springer, 2003, pp. 108–122.

[5] J. P. Warners, "A linear-time transformation of linear inequalities into conjunctive normal form," *Inf. Process. Lett.*, vol. 68, no. 2, pp. 63–69, 1998.

[6] T. Ogawa, Y. Liu, R. Hasegawa, M. Koshimura, and H. Fujita, "Modulo based CNF encoding of cardinality constraints and its application to maxsat solvers," in *2013 IEEE 25th International Conference on Tools with Artificial Intelligence, Herndon, VA, USA, November 4-6, 2013*. IEEE Computer Society, 2013, pp. 9–17.

[7] S. Hayata and R. Hasegawa, "Improvement in CNF encoding of cardinal constraints for weighted partial maxsat," in *SIG-FPAI-B404*. Japan Society for Artificial Intelligence, March 2015, pp. 80–84, in Japanese.

[8] S. Joshi, R. Martins, and V. M. Manquinho, "Generalized totalizer encoding for pseudo-boolean constraints," in *Principles and Practice of Constraint Programming - 21st International Conference, CP 2015, Cork, Ireland, August 31 - September 4, 2015, Proceedings*, ser. Lecture Notes in Computer Science, G. Pesant, Ed., vol. 9255. Springer, 2015, pp. 200–209.

[9] N. Uemura, H. Fujita, M. Koshimura, and A. Zha, "A SAT encoding of pseudo-Boolean constraints based on mixed radix," in *SIG-FPAI-B506*. Japan Society for Artificial Intelligence, March 2017, pp. 12–17, in Japanese.

[10] M. Codish, Y. Fekete, C. Fuhs, and P. Schneider-Kamp, "Optimal base encodings for pseudo-boolean constraints," in *Tools and Algorithms for the Construction and Analysis of Systems - 17th International Conference, TACAS 2011, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2011, Saarbrücken, Germany, March 26-April 3, 2011. Proceedings*, ser. Lecture Notes in Computer Science, P. A. Abdulla and K. R. M. Leino, Eds., vol. 6605. Springer, 2011, pp. 189–204.

BENCHMARK DESCRIPTIONS

MaxSAT Benchmarks: CSS Refactoring

Matthew Hague
Royal Holloway University of London
Egham, UK
Email: matthew.hague@rhul.ac.uk

Anthony Widjaja Lin
University of Oxford
Oxford, UK
Email: Anthony.Lin@cs.ox.ac.uk

Abstract—We identify CSS refactorings that can minimize the size of a given CSS file by inserting new rules that override parts of existing rules in the file. These overridden parts can then be removed, leading to an overall reduction in the size. Care must be taken when rules are combined to avoid altering the semantics of the styling document. Identifying a refactoring which leads to the greatest reduction is a MaxSAT problem. The submitted benchmarks are generated from CSS files from several popular websites.

I. Description

CSS files are routinely minimized before deployment, using simple minimization techniques such as removing comments and spaces and replacing strings with shorter equivalents (e.g. #ff0000 vs. red) [1], [2], [3], [4]. . Recent work has attempted to provide more complex minimizations that act on the file globally rather than locally [5], [6], [7], [8], [9].

We have focussed on CSS refactoring, which, as well as minimizing a CSS file, can be used to aid website development. The goal is to identify a new CSS rule that can be inserted into a CSS file. This new rule will combine the effect of parts of several other rules in the file. After the new rule has been inserted, the remaining file can be trimmed, leading to a reduction in size. As a simple example, consider the following CSS file.

```
.a { color: red }
.b { color: red }
```

We can refactor this file by introducing a new rule at the end of the file.

```
.a { color: red }
.b { color: red }
.a, .b { color: red }
```

This new rule overrides the behaviour of the previous two rules, which can then be removed. This leads to the smaller and more maintainable file shown below.

```
.a, .b { color: red }
```

Identifying refactorings which provide the maximum file size reduction is a NP-complete problem, and can be reduced to an instance of MaxSAT. We are currently developing a tool which minimizes CSS files based upon this reduction. We have included a number of WCNF benchmarks derived from our experiments with encodings of the refactoring problem into MaxSAT. These experiments have used CSS files from a number of popular websites.

II. Included Benchmarks

The included benchmarks are derived from CSS files used on a number of popular websites. These are briefly described below.

- amazon.dimacs – a refactoring problem derived from a CSS file taken from the Amazon website.
- archlinux.dimacs – a refactoring problem derived from the CSS file used on the Arch Linux homepage.
- arxiv.dimacs – a refactoring problem derived from the CSS file used on arXiv.org.
- dblp.dimacs – a refactoring problem derived from the CSS file used on the DBLP website.
- ebay.dimacs – a refactoring problem derived from a CSS file used on the eBay website.
- facebook.dimacs – a refactoring problem derived from a CSS file used on Facebook's website.
- github.dimacs – a refactoring problem derived from a CSS file used on the Github website.
- guardian.dimacs – a refactoring problem derived from the Guardian news website CSS file.
- openstreetmap.dimacs – a refactoring problem derived from a CSS file used on the Open Street Map website.
- wikipedia.dimacs – a refactoring problem derived from the CSS file used on Wikipedia.
- w3schools.dimacs – a refactoring problem derived from a CSS file used on the W3 Schools website.

References

[1] N. C. Zakas and N. Sullivan, "Csslint," http://csslint.net/, 2011, referred in April 2017.
[2] F. Schmitz and Contributors, "Csstidy," http://csstidy.sourceforge.net/, 2005, referred in April 2017.
[3] G. Martino and Contributors, "Uncss," https://github.com/giakki/uncss, 2013, referred April 2017.
[4] B. Briggs and Contributors, "cssnano," http://cssnano.co, 2015, referred in January 2017.
[5] A. Mesbah and S. Mirshokraie, "Automated analysis of CSS rules to support style maintenance," in 34th International Conference on Software Engineering, ICSE 2012, June 2-9, 2012, Zurich, Switzerland, 2012, pp. 408–418. [Online]. Available: http://dx.doi.org/10.1109/ICSE.2012.6227174

[6] D. Mazinanian, N. Tsantalis, and A. Mesbah, "Discovering refactoring opportunities in cascading style sheets," in Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering, (FSE-22), Hong Kong, China, November 16 - 22, 2014, 2014, pp. 496–506. [Online]. Available: http://doi.acm.org/10.1145/2635868.2635879

[7] M. Hague, A. W. Lin, and C. L. Ong, "Detecting redundant CSS rules in HTML5 applications: a tree rewriting approach," in Proceedings of the 2015 ACM SIGPLAN International Conference on Object-Oriented Programming, Systems, Languages, and Applications, OOPSLA 2015, part of SPLASH 2015, Pittsburgh, PA, USA, October 25-30, 2015, 2015, pp. 1–19. [Online]. Available: http://doi.acm.org/10.1145/2814270.2814288

[8] P. Genevès, N. Layaïda, and V. Quint, "On the analysis of cascading style sheets," in Proceedings of the 21st World Wide Web Conference 2012, WWW 2012, Lyon, France, April 16-20, 2012, 2012, pp. 809–818. [Online]. Available: http://doi.acm.org/10.1145/2187836.2187946

[9] M. Bosch, P. Genevès, and N. Layaïda, "Reasoning with style," in Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015, 2015, pp. 2227–2233. [Online]. Available: http://ijcai.org/Abstract/15/315

MaxSAT Benchmarks based on Determining Generalized Hypertree-width

Jeremias Berg*, Neha Lodha†, Matti Järvisalo*
and, Stefan Szeider †
*HIIT, Department of Computer Science, University of Helsinki, Finland
†Institute of Information Systems, Vienna University of Technology, Austria

I. Problem Overview

This benchmark set contains MaxSAT instances for determining the generalized hypertree-width [2], [3] (GHTW) of specific undirected graphs. GHTW is an important measure within graph theory: similarly as Treewidth, GHTW can be used to identify tractable instances of several different NP-hard problems. Specifically, whenever an NP-hard problem can be modeled using a hypergraph, the instances of that problem for which the underlying hypergraph has bounded GHTW, can be solved in polynomial time. As such computing GHTW has received interest in domains in which instances can be easily modeled using hypergraphs, for example, in the analysis of finite-domain constraint satisfaction problems [3], [2].

Following [2], given a hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$, a generalized hypertree-decomposition for $\mathcal{H}$ is a triple (T, χ, λ) s.t. $T = (N, E)$ is a tree, and χ and λ are two labeling functions, associating a set of nodes $\chi(p)$ and edges $\lambda(p)$ of $\mathcal{H}$ to each node p (bag) of T. Furthermore, we require that λ and χ satisfy the following conditions.

1) For each node b of $\mathcal{H}$, there is a node p of T s.t. $b \in \chi(p)$.
2) For each pair of nodes a and b included in some edge h of $\mathcal{H}$, there exists a node p s.t. $\{a, b\} \subset \chi(p)$.
3) For each node b of $\mathcal{H}$ the set $\{p \in N \mid b \in \chi(p)\}$ induces a connected subtree of T.
4) For each node p of T $\chi(p) \subseteq \cup\lambda(p)$.

Notice that the first three requirements imply that (T, χ) form a tree decomposition of the *primal graph* of $\mathcal{H}$ [2]. The width of (T, χ, λ) is the size of the largest edge-labeling set: $\max_{p \in N}\{\lambda(p)\}$. The GHTW of $\mathcal{H}$ is the minimum width of all generalized hypertree-decompositions of $\mathcal{H}$. Computing GHTW of a graph is known to be NP-hard, and even determining if if a graph as GHTW less than k is NP-complete for any fixed $k > 2$ [4].

II. MaxSAT Encoding

The MaxSAT encoding for GHTW used in these benchmarks is extended from the MaxSAT encoding for computing the treewidth of a graph first proposed in [6] and further developed in [1]. Given a graph G as input, the treewidth encoding includes hard clauses that describe a perfect elimination ordering of G and soft clauses that enforce minimization of the maximum clique size. The encoding for GHTW is extended by including extra variables to capture λ and soft clauses that minimize the number of edges assigned to any one bag,

III. Datasets in the benchmark set

The benchmark set consists of 42 MaxSAT instances generated based on standard graph benchmarks from [5]; The filename convention of the WCNF files in the benchmark set is

GenHyperTW_graphname.wcnf

where "graphname" gives the name of the graph. For each instance, optimal cost equals the generalized hypertree-width of the underlying graph.

References

[1] J. Berg and M. Järvisalo, "SAT-based approaches to treewidth computation: An evaluation," in *Proc. ICTAI*. IEEE Computer Society, 2014, pp. 328–335.
[2] G. Gottlob, G. Greco, and F. Scarcello, "Treewidth and hypertree width," in *Tractability: Practical Approaches to Hard Problems*, L. Bordeaux, Y. Hamadi, and P. Kohli, Eds. Cambridge University Press, 2014, pp. 3–38.
[3] G. Gottlob, N. Leone, and F. Scarcello, "Hypertree decompositions: A survey," in *Proc. MFCS*, ser. Lecture Notes in Computer Science, vol. 2136. Springer, 2001, pp. 37–57.
[4] G. Gottlob, Z. Miklós, and T. Schwentick, "Generalized hypertree decompositions: NP-hardness and tractable variants," in *Proc. PODS*. ACM, 2007, pp. 13–22.
[5] G. Gottlob and M. Samer, "A backtracking-based algorithm for hypertree decomposition," *ACM Journal of Experimental Algorithmics*, vol. 13, 2008.
[6] M. Samer and H. Veith, "Encoding treewidth into SAT," in *Proc. SAT*, ser. Lecture Notes in Computer Science, vol. 5584. Springer, 2009, pp. 45–50.

Discrete Optimization Problems in Dynamics of Abstract Argumentation: MaxSAT Benchmarks

Andreas Niskanen
Helsinki Institute for Information Technology HIIT, Department of Computer Science, University of Helsinki, Finland
Email: andreas.niskanen@helsinki.fi

Johannes P. Wallner
Institute of Information Systems
TU Vienna, Austria
Email: wallner@dbai.tuwien.ac.at

Matti Järvisalo
Helsinki Institute for Information Technology HIIT, Department of Computer Science, University of Helsinki, Finland
Email: matti.jarvisalo@helsinki.fi

***Abstract*—Argumentation is an active area of modern artifical intelligence (AI) research. Abstract argumentation, with argumentation frameworks (AFs) modelling conflicts between arguments, is the core knowledge representation formalism of argumentation in AI. Different argumentation semantics provide a way to determine sets of non-conflicting arguments, i.e., extensions, from the AF, represented as a directed graph. Recently, there has been a strong focus on the dynamic aspects of AFs, and on computational problems which arise from them, which are often NP-hard optimization problems. These have been successfully tackled via constraint-based declarative approaches, most notably encodings in maximum satisfiability (MaxSAT). Here we present a benchmark description for two such optimization problems, namely, extension enforcement and AF synthesis, including preliminaries on abstract argumentation, problem definitions, instance generation and naming conventions.**

I. Preliminaries

We begin by formally defining argumentation frameworks [1] (see also [2]) and the argumentation semantics considered in this work.

Definition 1. *An* argumentation framework (AF) *is a pair* $F = (A, R)$*, where* A *is a finite non-empty set of arguments and* $R \subseteq A \times A$ *is the attack relation. The pair* $(a, b) \in R$ *indicates that* a *attacks* b*, i.e.,* a *is a counterargument for* b*. An argument* $a \in A$ *is* defended *(in* F*) by a set* $S \subseteq A$ *if, for each* $b \in A$ *such that* $(b, a) \in R$*, there is a* $c \in S$ *such that* $(c, b) \in R$*.*

Semantics for AFs are defined through functions σ which assign to each AF $F = (A, R)$ a set $\sigma(F) \subseteq 2^A$ of extensions. We consider for σ the functions adm, com, and stb, which stand for admissible, complete, and stable, respectively.

Definition 2. *Given an AF* $F = (A, R)$*, the* characteristic function $\mathcal{F}_F : 2^A \to 2^A$ *of* F *is* $\mathcal{F}_F(S) = \{x \in A \mid x \text{ is defended by } S\}$*. Moreover, for a set* $S \subseteq A$*, the* range of S *is* $S_R^+ = S \cup \{x \in A \mid (y, x) \in R, y \in S\}$.

Definition 3. *Let* $F = (A, R)$ *be an AF. A set* $S \subseteq A$ *is* conflict-free (in F) *if there are no* $a, b \in S$ *such that* $(a, b) \in R$*. We denote the collection of conflict-free sets of* F *by* $cf(F)$*. For a conflict-free set* $S \in cf(F)$ *it holds that*

- $S \in stb(F)$ *iff* $S_R^+ = A$*;*
- $S \in adm(F)$ *iff* $S \subseteq \mathcal{F}_F(S)$*;*
- $S \in com(F)$ *iff* $S = \mathcal{F}_F(S)$.

If $E \in \sigma(F)$ *for semantics* σ*, we call* E *a* σ*-extension, or an extension under semantics* σ*.*

II. Problem Definitions

A. Extension Enforcement

The task of argument-fixed extension enforcement [3], [4] is to modify the attack structure R of an AF $F = (A, R)$ in a way that a given set T becomes (a subset of) an extension under a given semantics σ. *Strict* enforcement requires that the given set of arguments has to be exactly a σ-extension, while in *non-strict* enforcement it is required to be a subset of a σ-extension. We denote strict by s and non-strict by ns.

Formally, denote by

$$enf(F, T, s, \sigma) = \{R' \mid F' = (A, R'),\ T \in \sigma(F')\},$$

the set of attack structures that strictly enforce T under σ for an AF F, and by

$$enf(F, T, ns, \sigma) = \{R' \mid F' = (A, R'),\ \exists T' \in \sigma(F') : T' \supseteq T\}$$

for non-strict enforcement.

The Hamming distance between two attack structures R and R' is $|R \Delta R'| = |R \setminus R'| + |R' \setminus R|$, i.e., the number of changes (additions or removals of attacks) of an enforcement. We consider extension enforcement as an optimization problem, where the number of changes is minimized.

Extension Enforcement ($M \in \{s, ns\}$)
Input: AF $F = (A, R)$, $T \subseteq A$, and semantics σ.
Task: Find an AF $F^* = (A, R^*)$ with

$$R^* \in \operatorname*{arg\,min}_{R' \in enf(F,T,M,\sigma)} |R \Delta R'|.$$

B. AF Synthesis

Let A be a given non-empty finite set of arguments. In AF synthesis [5], we are given two sets of weighted examples of subset of A, representing semantical information with weights intuitively expressing the relative trust. The computational task

is to synthesize, or construct, an AF that optimally represents the examples as extensions and non-extensions. That is, an example $e = (S, w)$ is a pair with $S \subseteq A$ a subset of the set of arguments, and a positive integer $w > 0$ representing the example's weight. Denote the set of arguments of an example $e = (S, w)$ by $S_e = S$ and the weight by $w_e = w$.

An instance of the AF synthesis problem is a quadruple $P = (A, E^+, E^-, \sigma)$, with a non-empty set A of arguments, two sets of examples, E^+ and E^-, that we call positive and negative examples, respectively, and semantics σ. An AF F satisfies a positive example e if $S_e \in \sigma(F)$; similarly, F satisfies a negative example if $S_e \notin \sigma(F)$. For a given AF F, the associated cost w.r.t. P, denoted by $cost(P, F)$, is the sum of weights of examples not satisfied by F. Formally, $cost(P, F)$ is

$$\sum_{e \in E^+} w_e \cdot I(S_e \notin \sigma(F)) + \sum_{e \in E^-} w_e \cdot I(S_e \in \sigma(F)),$$

where $I(\cdot)$ is the indicator function. The task in AF synthesis is to find an AF of minimum cost over all AFs.

AF Synthesis
INPUT: $P = (A, E^+, E^-, \sigma)$
TASK: Find an AF F^* with

$$F^* \in \arg\min_{F=(A,R)}(cost(P, F)).$$

III. Instance Generation

For both optimization problems, we first generated a large set of instances using the random models described in the following subsections. From this set, we picked a representative set of benchmarks using the results of the corresponding MaxSAT solver comparisons. This resulted in 20 partial MaxSAT instances for strict extension enforcement under the complete semantics, 20 partial MaxSAT instances for non-strict extension enforcement under the stable semantics, and 40 weighted partial MaxSAT instances for AF synthesis under the stable semantics.

A. Extension Enforcement

The (partial) MaxSAT encodings for NP-fragments of extension enforcement are presented in [4]. To generate the benchmark instances, given a number of arguments and an edge probability p, we formed an AF based on the Erdős-Rényi random digraph model, where each attack is included independently with probability p. Given an AF and a number of enforced arguments, we constructed a corresponding enforcement instance by sampling the enforced arguments uniformly at random from the set of arguments, without replacement. For each number of arguments $|A| \in \{25, 50, \dots\}$ and each edge probability $p \in \{0.05, 0.1, 0.2, 0.3\}$, we generated five AFs. For each AF, we generated five enforcement instances with $|T|$ enforced arguments, for each $|T|/|A| \in \{0.05, 0.1, 0.2, 0.3\}$.

B. AF Synthesis

The (weighted partial) MaxSAT encodings for NP-fragments of AF synthesis are presented in [5]. We picked $5, 10, \dots, 80$ positive examples from a fixed set of 100 arguments uniformly at random with probability $p^+_{\text{arg}} = 0.25$. Then $|E^-| = 20, 40, \dots, 200$ negative examples were sampled from the set $A = \bigcup \mathbb{S}_{E^+}$, and each argument was included with probability $p^-_{\text{arg}} = \frac{\sum_{e \in E^+} |S_e|/|E^+|}{|\bigcup \mathbb{S}_{E^+}|}$. Again, each example was assigned as weight a random integer from the interval $[1, 10]$. For each choice of parameters, this procedure was repeated 10 times to obtain a representative set of benchmarks.

IV. Naming Conventions

A. Extension Enforcement

The instances for extension enforcement are named by

`extension-enforcement_<mode>_<sem>_<args>_<prob>_<af_id>_<enfs>_<enf_id>.wcnf`

where `<mode>` is the type (non-strict or strict) of enforcement, `<sem>` is the AF semantics, `<args>` is the number of arguments, `<prob>` is the attack probability, `<enfs>` is the number of enforced arguments, and `<af_id>` and `<enf_id>` are IDs assigned to each instance.

B. AF Synthesis

The instances for AF synthesis are named by

`af-synthesis_<sem>_<n_pos>_<n_neg>_<id>.wcnf`

where `<mode>` is the AF semantics, `<n_pos>` is the number of positive examples, `<n_neg>` is the number of negative examples, and `<id>` is an ID assigned to each instance.

V. Benchmark Instances

All instances used in the MaxSAT evaluation 2017 are found online at https://www.cs.helsinki.fi/group/coreo/benchmarks/.

Acknowledgments

This work is supported by Academy of Finland, grants #251170 (COIN Centre of Excellence in Computational Inference Research), #276412, and #284591; Doctoral School in Computer Science DOCS and Research Funds of the University of Helsinki; and Austrian Science Fund (FWF): I2854 and P30168-N31.

References

[1] P. M. Dung, "On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games," *Artificial Intelligence*, vol. 77, no. 2, pp. 321–358, 1995.

[2] P. Baroni, M. Caminada, and M. Giacomin, "An introduction to argumentation semantics," *Knowledge Engineering Review*, vol. 26, no. 4, pp. 365–410, 2011.

[3] S. Coste-Marquis, S. Konieczny, J. Mailly, and P. Marquis, "Extension enforcement in abstract argumentation as an optimization problem," in *Proc. IJCAI*. AAAI Press, 2015, pp. 2876–2882.

[4] J. P. Wallner, A. Niskanen, and M. Järvisalo, "Complexity results and algorithms for extension enforcement in abstract argumentation," in *Proc. AAAI*. AAAI Press, 2016, pp. 1088–1094.

[5] A. Niskanen, J. P. Wallner, and M. Järvisalo, "Synthesizing argumentation frameworks from examples," in *Proc. ECAI*, ser. Frontiers in Artificial Intelligence and Applications, vol. 285. IOS Press, 2016, pp. 551–559.

Lisbon Wedding: Seating arrangements using MaxSAT

Ruben Martins
Carnegie Mellon University
rubenm@cs.cmu.edu

Justine Sherry
Carnegie Mellon University
sherry@cs.cmu.edu

***Abstract*—Having a perfect seating arrangement for weddings is not an easy task. Can Alice sit next to Bob? Can we ensure that Charles and his ex-girlfriend Eve not be seated together? Meeting such constraints is classically one of the most difficult tasks in planning a wedding – and guests will not accept 'it's NP-complete!' as an excuse for poor seating arrangements. We discuss how MaxSAT can provide the optimal seating arrangement for a perfect wedding, saving brides and grooms (including the authors) from hours of struggle.**

I. Introduction

This benchmark description describes the encoding used for the wedding seating arrangement for our wedding in Lisbon. We needed to seat our guests according to a long list of constraints. For example, members of the same family should sit together; friends who went to school together should sit together; individuals with a history of conflict should be seated apart; etc. We wanted to maximize the happiness of our guests and what better way to do that than to encode the problem into MaxSAT! MaxSAT was an ideal solution for our own wedding: i) it saved us tens of hours, ii) it was stress free, and iii) in the rare case that a guest complained about their seating arrangement, we just blamed the algorithm![1]

II. MaxSAT Encoding

When making a seating arrangement, we first need to define the size of each table and how many guest we have. Assume that our guests are defined by the set P and the tables are defined by the set T. Each table has at least l guests and at most u guests.

Variables. We define our variables as being p_t, meaning that guest p is seated at table t. For simplicity, we do not consider where each person is seated at each table but only if a given person p is seated or not at table t. To characterize our guests, we use a set of auxiliary variables S that denotes characteristics of each person, namely s_t^p denotes the characteristics of person p, seated at table t.

Hard constraints. The hard constraints define the shape of each table and guarantee that each guest will be seated in exactly one place.

- Each guest will be seated at exactly one table:

$$\forall_{p \in P} \sum_{t \in T} p_t = 1$$

- Each table will have at most u guests:

$$\forall_{t \in T} \sum_{p \in P} p_t \leq u$$

- Each table will have at least l guests:

$$\forall_{t \in T} \sum_{p \in P} p_t \geq l$$

Since some guests may have disagreements with each other, we also included some exclusion constraints that guarantee that guests which have conflicts with each other are not seated in the same table. For every pair of guests p and p' that have a conflict with each other we include the following constraints that guarantee that they will not seat together:

$$\forall_{t \in T} (p_t + p'_t \leq 1)$$

To enforce that if a person p is seated at table t then t will contain all labels belonging to p we add additional hard constraints that enforce that table t will contain all the labels from guests that are seated there:

$$\forall_{t \in T} \forall_{p \in P} \forall_{s \in S_p} (p_t \implies s_t^p)$$

Soft constraints. The soft constraints describe the commonalities between guests that share a table. We attach a set of labels to each person that describes her. Example of labels are: spoken languages, university they attended or family last name. Our goal is to minimize the number of labels in each table, i.e. we want to maximize what guests have in common at each table. Let S_t be the set of labels that can occur in table t.

- Minimize the number of labels in each table:

$$min : \sum\nolimits_{t \in T} \sum_{s \in S_t} s$$

Since some labels may be more important than other (e.g. spoken language), we may associate a different weight to each label.

[1]While we were convinced that the algorithm's output was optimal, our guests were not all so enlightened.

III. GENERATOR

We iteratively generated our constraints, adding additional labels or marking guests as in conflict and feeding them to the MaxSAT solver until we arrived at a solution we were happy with. We generated 30 versions of our seating arrangements based on these iterative versions. The generator takes as input: i) the number of tables, ii) the minimum number of guests per table, iii) the maximum number of guests per table, iv) a .csv file with the list of guests and the labels associated with each guest, v) a .txt file with weights for each label, and vi) a .txt file with a set of conflicting labels so that those guests are not seated together.

The problem was encoded using a pseudo-Boolean formalism and translated to MaxSAT using the Open-WBO framework [1]. The following encodings are used by Open-WBO to convert a pseudo-Boolean formula to MaxSAT: i) Ladder encoding [2], [3] (at-most-one constraints), ii) Modulo Totalizer encoding [4] (cardinality constraints) and iii) Generalized Totalizer encoding [5] (pseudo-Boolean constraints).

REFERENCES

[1] Ruben Martins, Vasco Manquinho, Ines Lynce: Open-WBO: A Modular MaxSAT Solver. SAT 2014: 438-445

[2] Carlos Ansotegui, Felip Manya: Mapping problems with finite-domain variables into problems with boolean variables. SAT 2004: 115

[3] Ian Gent, Peter Nightingale: A new encoding of All Different into SAT. ModRef 2004

[4] Toru Ogawa, Yangyang Liu, Ryuzo Hasegawa, Miyuki Koshimura, Hiroshi Fujita: Modulo Based CNF Encoding of Cardinality Constraints and Its Application to MaxSAT Solvers. ICTAI 2013: 9-17

[5] Saurabh Joshi, Ruben Martins, Vasco Manquinho: Generalized Totalizer Encoding for Pseudo-Boolean Constraints. CP 2015: 200-209

ASP to MaxSAT: Metro, ShiftDesign, TimeTabling and BioRepair

Ruben Martins
Carnegie Mellon University
rubenm@cs.cmu.edu

I. Introduction

This benchmark description describes the origin of the `Metro`, `ShiftDesign`, `TimeTabling` and `BioRepair` benchmarks which can be tracked back to their first encoding in Answer Set Programming (ASP). ASP is a form of declarative programming and has been successfully applied to many practical applications. These benchmarks are a few examples of real-world instances where ASP has been used. `Metro` benchmarks describe problems related to transport systems (e.g. [1]). `ShiftDesign` [2] targets a scheduling problem where the goal is to minimize the number of shifts such that it reduces understaffing. `TimeTabling` [3] describes scheduling problems related to educational timetabling and `BioRepair` [4] addresses the problem of repairing large-scale biological networks.

These benchmarks have also been recently translated to pseudo-Boolean (PB) with the tool ACYC2SOLVER [7], [8] and submitted to the pseudo-Boolean Evaluation 2015 [5]. The benchmarks in PB format are available at [6].

II. MaxSAT Evaluation 2017

Each benchmark set (`Metro`, `ShiftDesign`, `TimeTabling`, `BioRepair`) consists of 30 instances and these were translated from pseudo-Boolean to MaxSAT using the Open-WBO framework [9]. The following encodings are used by Open-WBO to convert a pseudo-Boolean formula to MaxSAT: i) Ladder encoding [10], [11] (at-most-one constraints), ii) Modulo Totalizer encoding [12] (cardinality constraints) and iii) Generalized Totalizer encoding [13] (pseudo-Boolean constraints).

Acknowledgments

We thank the original creators of these benchmarks that encoded them into ASP and the organizers of the PB Evaluation 2015 for translating them to PB format.

References

[1] Gerhard Brewka, Martin Diller, Georg Heissenberger, Thomas Linsbichler, Stefan Woltran: Solving Advanced Argumentation Problems with Answer-Set Programming. AAAI 2017: 1077-1083
[2] Michael Abseher, Martin Gebser, Nysret Musliu, Torsten Schaub, Stefan Woltran: Shift Design with Answer Set Programming. LPNMR 2015: 32-39
[3] Mutsunori Banbara, Takehide Soh, Naoyuki Tamura, Katsumi Inoue, Torsten Schaub: Answer set programming as a modeling language for course timetabling. TPLP 13(4-5): 783-798 (2013)
[4] Martin Gebser, Carito Guziolowski, Mihail Ivanchev, Torsten Schaub, Anne Siegel, Sven Thiele, Philippe Veber: Repair and Prediction (under Inconsistency) in Large Biological Networks with Answer Set Programming. KR 2010
[5] Pseudo-Boolean Evaluation 2015. http://pbeva.computational-logic.org/
[6] ASP Instances from the Application Track. http://pbeva.computational-logic.org/benchmarks/ASP.tar.gz
[7] Martin Gebser, Tomi Janhunen, Jussi Rintanen: Answer Set Programming as SAT modulo Acyclicity. ECAI 2014: 351-356
[8] Martin Gebser, Tomi Janhunen, Jussi Rintanen: SAT Modulo Graphs: Acyclicity. JELIA 2014: 137-151
[9] Ruben Martins, Vasco Manquinho, Ines Lynce: Open-WBO: A Modular MaxSAT Solver. SAT 2014: 438-445
[10] Carlos Ansotegui, Felip Manya: Mapping problems with finite-domain variables into problems with boolean variables. SAT 2004: 115
[11] Ian Gent, Peter Nightingale: A new encoding of All Different into SAT. ModRef 2004
[12] Toru Ogawa, Yangyang Liu, Ryuzo Hasegawa, Miyuki Koshimura, Hiroshi Fujita: Modulo Based CNF Encoding of Cardinality Constraints and Its Application to MaxSAT Solvers. ICTAI 2013: 9-17
[13] Saurabh Joshi, Ruben Martins, Vasco Manquinho: Generalized Totalizer Encoding for Pseudo-Boolean Constraints. CP 2015: 200-209

MSE17 Benchmarks: DALculus

Ruben Martins
Carnegie Mellon University
rubenm@cs.cmu.edu

I. Introduction

The Development Assurance Level (DAL) indicates the level of rigor of the development of a software or hardware function of an aircraft. This problem can be encoded into a multi-objective pseudo-Boolean (PB) formula [1] and solved using a Boolean optimizer. The origin of these benchmarks can be tracked back to the optimization challenge at the LION 9 conference [3], [4]. Since most pseudo-Boolean solvers do not support multi-objective optimization, only SAT4J [2] and OPEN-WBO [6] participated in this challenge.

II. MaxSAT Evaluation 2017

A multi-objective function $f_1, \dots f_n$ with lexicographical ordering can be encoded into MaxSAT by assigning weights to each objective function f_i such that unsatisfying any soft clause in f_i will have a higher cost than unsatisfying any soft clause in $f_{i+1}, \dots, f_n$. The multi-objective function was encoded into MaxSAT using the Boolean Lexicographic Optimization scheme described in [5]. The remainder PB formula was translated into MaxSAT using the OPEN-WBO framework [6]. The following encodings are used by OPEN-WBO to convert a pseudo-Boolean formula to MaxSAT: i) Ladder encoding [7], [8] (at-most-one constraints), ii) Modulo Totalizer encoding [9] (cardinality constraints) and iii) Generalized Totalizer encoding [10] (pseudo-Boolean constraints). The benchmark set submitted to the MaxSAT Evaluation 2017 consists of 96 benchmarks (48 "easy" and 48 "hard"). The original benchmarks are available at [3].

Acknowledgments

We thank Pierre Bieber, Rémi Delmas and Christel Seguin from the French Aerospace Lab ONERA for creating the original multi-objective pseudo-Boolean encoding for the "Dalculus" benchmarks [1] and for submitting them to the optimization challenge at the LION 9 conference [3].

References

[1] Pierre Bieber, Remi Delmas, Christel Seguin: DALculus - Theory and Tool for Development Assurance Level Allocation. SAFECOMP 2011: 43-56
[2] Daniel Le Berre, Anne Parrain: The Sat4j library, release 2.2. JSAT 7(2-3): 59-6 (2010)
[3] Challenge LION9. http://www.lifl.fr/LION9/challenge.php
[4] Challenge LION9: Results. http://www.cril.univ-artois.fr/ChallengeLion9/results/results.php?idev=75
[5] Joao Marques-Silva, Josep Argelich, Ana Graça, Inês Lynce: Boolean lexicographic optimization: algorithms & applications. Ann. Math. Artif. Intell. 62(3-4): 317-343 (2011)
[6] Ruben Martins, Vasco Manquinho, Ines Lynce: Open-WBO: A Modular MaxSAT Solver. SAT 2014: 438-445
[7] Carlos Ansotegui, Felip Manya: Mapping problems with finite-domain variables into problems with boolean variables. SAT 2004: 115
[8] Ian Gent, Peter Nightingale: A new encoding of All Different into SAT. ModRef 2004
[9] Toru Ogawa, Yangyang Liu, Ryuzo Hasegawa, Miyuki Koshimura, Hiroshi Fujita: Modulo Based CNF Encoding of Cardinality Constraints and Its Application to MaxSAT Solvers. ICTAI 2013: 9-17
[10] Saurabh Joshi, Ruben Martins, Vasco Manquinho: Generalized Totalizer Encoding for Pseudo-Boolean Constraints. CP 2015: 200-209

Solving RNA Alignment with MaxSAT

Ruben Martins
Carnegie Mellon University
rubenm@cs.cmu.edu

I. Introduction

The similarity between RNA sequences can be used to study the evolutionary or functional similarity between two sequences. One way to measure this similarity is to compute a pairwise sequence alignment based on the longest common subsequence (LCS) algorithm [1]. RNA sequences can be represented by arc-annotated sequences where an arc correspond to a bond. Sequence alignment based on the LCS algorithm do not consider *pseudoknots* which correspond to crossing arcs. Indeed, when considering the problem of finding the maximal common subsequence with arcs and pseudoknots, the alignment problem for RNA becomes NP-Complete [2].

II. Finding Common Subsequences with Arcs and Pseudoknots

Consider the RNA sequences s_1 (above) and s_2 (below) shown in Figure 1. An alignment between s_1 and s_2 obeys the following properties: i) it is an one-to-one mapping that preserves the order of the subsequence, ii) the arcs induced by the mapping are preserved, and iii) the mapping produces a common subsequence. We refer the reader to [2] for a formal definition of the *arc-preserving longest common subsequence problem*.

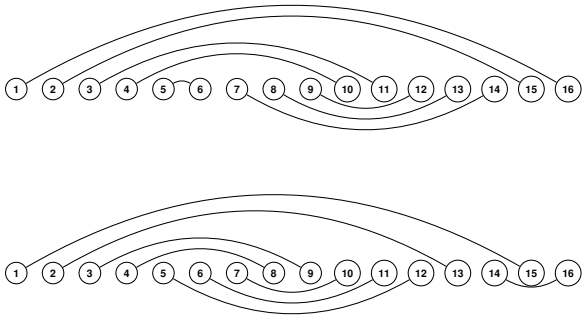


Fig. 1. Example of RNA sequences s_1 (above) and s_2 (below)

Table I describes the maximal alignment $\mathcal{A}$ between s_1 and s_2. An alignment is maximal if it maps the maximal number of arcs between s_1 and s_2. Each arc $c \in s_1$ in $\mathcal{A}$ is aligned to an arc $c \in s_2$. An arc c is denoted by a pair of nodes (n_i, n_j) where n_i and n_j corresponds to the nodes that form c.

This problem can be encoded into Maximum Satisfiability (MaxSAT) or pseudo-Boolean (PB) formulas and solved using a Boolean optimizer. However, this encoding leads to challenging benchmarks that are beyond the reach of Boolean optimizers. [1]

TABLE I
Maximal alignment between s_1 and s_2

s_1		s_2
(4,10)	$\Leftrightarrow$	(4,8)
(3,11)	$\Leftrightarrow$	(3,9)
(9,12)	$\Leftrightarrow$	(7,10)
(8,13)	$\Leftrightarrow$	(6,11)
(7,14)	$\Leftrightarrow$	(5,12)
(2,15)	$\Leftrightarrow$	(2,13)
(1,16)	$\Leftrightarrow$	(1,15)

III. Preprocessing using Super-Arcs

To simplify the alignment problem, we consider a preprocessing step using the notion of super-arcs. We call s a super-arc if multiple arcs $c_1, \ldots, c_k$ can be merged into a new arc s with weight k. This merge operation is only possible if $c_1, \ldots, c_k$ are not pseudoknots (i.e., there are no crossing arcs between $c_1, \ldots, c_k$). For example, the arcs (1,16) and (2,15) in s_1 can be merged into a new super-arc with weight 2. Figure 2 shows the preprocessed structure of s_1 and s_2 after applying the super-arc preprocessing, respectively denoted by s'_1 and s'_2. As can be seen in Figure 2, the preprocessing significantly reduces the size of the graph representation.

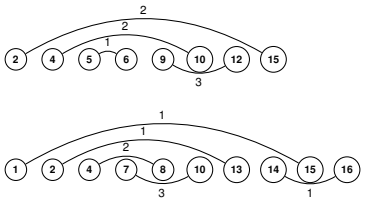


Fig. 2. Preprocessed RNA sequences s'_1 (above) and s'_2 (below)

The encoding of a maximal alignment using super-arcs differs mainly from the original encoding in the following way. A super-arc can be mapped to multiple arcs as long as the sum of the weights of their mappings is smaller or equal to the weight of the super-arc. Table II shows the maximal alignment between s'_1 and s'_2 using the notion of super-arcs. For each arc,

[1] The last attempt at solving the arc-preserving longest common subsequence problem with MaxSAT/PB solvers was done in 2010 and since then MaxSAT solvers have been significantly improved. It may be that the encoding for this problem can be now solved by state-of-the-art MaxSAT solvers. In 2010 (when this encoding was created), MINISAT+ [4] was the most efficient solver for this kind of benchmarks.

TABLE II
MAXIMAL ALIGNMENT BETWEEN s'_1 AND s'_2

s_1		s_2
(4,10)-2	$\Leftrightarrow$	(4,8)-2
(9,12)-3	$\Leftrightarrow$	(7,10)-3
(2,15)-1	$\Leftrightarrow$	(2,13)-1
(2,15)-1	$\Leftrightarrow$	(1,15)-1

we include: i) the starting node, ii) the ending node, and iii) the weight that was mapped. As mentioned before, an arc with weight k can be mapped to more than one arc if the sum of the mappings is smaller or equal to k. For example, node (2,15) is mapped to two different nodes with weight 1. This is allowed since node (2,15) has weight 2 in the super-arc representation.

For this example, the optimal alignment with and without super-arc preprocessing is equivalent, i.e. both approaches lead to an alignment that maps 7 arcs from s_1 to s_2. However, in general, the maximal alignment when using super-arcs preprocessing is not always equivalent to the alignment without preprocessing. Even though this technique does not always preserves optimal solutions, it can be used to study different alignment properties between RNA sequences and has been further explored in [3].

IV. MAXSAT EVALUATION 2017

The benchmark set `rna-alignment` consists of 103 instances that were translated from pseudo-Boolean to MaxSAT using the OPEN-WBO framework [6]. The following encodings are used by OPEN-WBO to convert a pseudo-Boolean formula to MaxSAT: i) Ladder encoding [7], [8] (at-most-one constraints), ii) Modulo Totalizer encoding [9] (cardinality constraints) and iii) Generalized Totalizer encoding [10] (pseudo-Boolean constraints). From these 103 instances, 3 have origin from real RNA sequences (`tmosaic-tob-chim`, `tmosaic-tob-yel`, `tmosaic-yel-chim`), whereas the remaining 100 were randomly generated by the author. The random generator takes into account the pseudoknots structure that appears in real RNA sequences [5] and is available upon email request to the author.

ACKNOWLEDGMENTS

We thank Guillaume Blin and Florian Sikora for fruitful discussions during the Summer of 2010 at the University of Marne-la-Valle, Paris, France that led to the preprocessing technique based on super-arcs. The initial work on super-arcs was further explored by Blin et al. [3] where they extended it using the notion of *common arc-annotated supersequence*.

REFERENCES

[1] T. Smith and M. Waterman. Identification of common molecular subsequences. Journal of Molecular Biology 147 (1981), 195-197

[2] Patricia A. Evans: Finding Common Subsequences with Arcs and Pseudoknots. CPM 1999: 270-280

[3] Guillaume Blin, Alain Denise, Serge Dulucq, Claire Herrbach, Hélène Touzet: Alignments of RNA Structures. IEEE/ACM Trans. Comput. Biology Bioinform. 7(2): 309-322 (2010)

[4] Niklas En, Niklas Srensson: Translating Pseudo-Boolean Constraints into SAT. JSAT 2(1-4): 1-26 (2006)

[5] PseudoBase++: Database for pseudoknots. http://pseudobaseplusplus.utep.edu/

[6] Ruben Martins, Vasco Manquinho, Ines Lynce: Open-WBO: A Modular MaxSAT Solver. SAT 2014: 438-445

[7] Carlos Ansotegui, Felip Manya: Mapping problems with finite-domain variables into problems with boolean variables. SAT 2004: 115

[8] Ian Gent, Peter Nightingale: A new encoding of All Different into SAT. ModRef 2004

[9] Toru Ogawa, Yangyang Liu, Ryuzo Hasegawa, Miyuki Koshimura, Hiroshi Fujita: Modulo Based CNF Encoding of Cardinality Constraints and Its Application to MaxSAT Solvers. ICTAI 2013: 9-17

[10] Saurabh Joshi, Ruben Martins, Vasco Manquinho: Generalized Totalizer Encoding for Pseudo-Boolean Constraints. CP 2015: 200-209

MaxSAT Benchmarks Encoding Optimal Causal Graphs

Antti Hyttinen and Matti Järvisalo
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 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) [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 find the graph[1] 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 better accuracy than previous approaches [2]. 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 1, right) is here 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 here allowed to include directed cycles [3] (e.g., supply and demand), 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 [5], [6] for details).

Intuitively, we find 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 ($X_i \not\perp\!\!\!\perp X_j | C$)—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—or independent ($X_i \perp\!\!\!\perp X_j | C$) in the observed data (Figure 1, middle). Furthermore, we also obtain a weight describing the reliability of the decision.

Now, under some common theoretical assumptions (see [6] 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

[1]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.

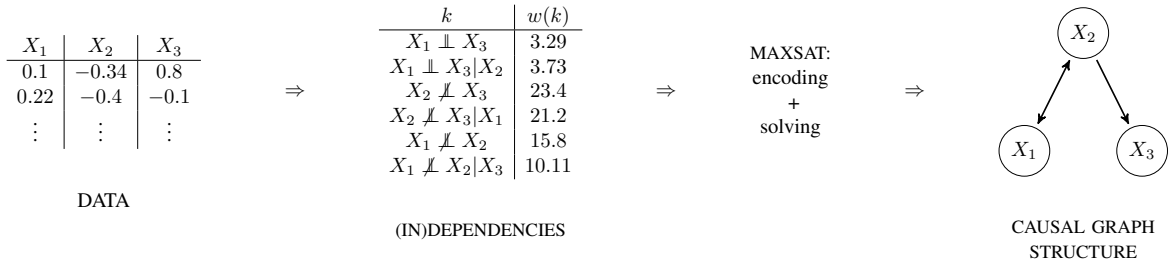


Fig. 1: The causal structure discovery problem by example [1]

are allowed) such that every 'collider node' connected with two incoming edges on the path is in C and other nodes on the path are not in C [5], [7]. 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 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 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.

A. 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 1 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 1, right) that minimizes the sum of costs of reachability/separability constraints that are not satisfied:

$$G^* \in \operatorname*{arg\,min}_{G \in CG(n)} \sum_{k \in K \,:\, G \not\models k} w(k), \tag{1}$$

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 $k \in K$.

II. MaxSAT Encoding

The optimization problem is computationally challenging. For obtaining good accuracy, a large number of (in)dependence constraints K are needed; we use *all* testable (in)dependence constraints ($\binom{n}{2}2^{n-2}$ for n 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 $k \in K$ is encoded as a unit soft clause over a distinct Boolean variable representing k with weight $w(k)$. 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 definition supports latent variables). We employed the BDEU score with equivalent sample size 10 to obtain independence constraint weights for this discrete data. The were turned to intergers by multiplying by 1000 and rounding. All files use the encoding over conditioning and marginalization operations [2]. The number of variables was selected for each data such that we would get a sensible comparison among different MaxSAT solvers.

IV. File Name Convention

The instances in this benchmark set are named using following convention:

`causal_<dataset>_<n>_<N>.wcnf`

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

References

[1] J. Berg, A. Hyttinen, and M. Järvisalo, "Applications of MaxSAT in data analysis," in *6th Pragmatics of SAT Workshop (PoS 2015)*, 2015.

[2] A. Hyttinen, F. Eberhardt, and M. Järvisalo, "Constraint-based causal discovery: Conflict resolution with answer set programming," in *Proc. UAI*. AUAI Press, 2014, pp. 340–349.

[3] A. Hyttinen, P. O. Hoyer, F. Eberhardt, and M. Järvisalo, "Discovering cyclic causal models with latent variables: A general SAT-based procedure," in *Proc. UAI*. AUAI Press, 2013, pp. 301–310.

[4] A. Hyttinen, P. Saikko, and M. Järvisalo, "A core-guided approach to learning optimal causal graphs," in *Proc. IJCAI*. IJCAI, 2017, pp. 645–651.

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

[6] P. Spirtes, C. Glymour, and R. Scheines, *Causation, Prediction, and Search*. MIT Press, 2000.

[7] M. Studený, "Bayesian networks from the point of view of chain graphs," in *Proc. UAI*. Morgan Kaufmann, 1998, pp. 496–503.

Generalized Ising Model (Cluster Expansion) Benchmark

Wenxuan Huang[1]
[1]Department of Materials Science and Engineering
Massachusetts Institute of Technology
Cambridge, MA 02139, USA
Key01027@mit.edu

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

Keywords*— *Cluster Expansion, Ising Model, Computational Material Science

I. INTRODUCTION

Lattice models have wide applicability in science [1-10], and have been used in a wide range of applications, such as magnetism [11], alloy thermodynamics [12], fluid dynamics [13], phase transitions in oxides [14], and thermal conductivity [15]. A lattice model, also referred to as generalized Ising model [16] or cluster expansion [12], is the discrete representation of materials properties, e.g., formation energies, in terms of lattice sites and site interactions. In first-principles thermodynamics, lattice models take on a particularly important role as they appear naturally through a coarse graining of the partition function [17] of systems with substitutional degrees of freedom. As such, they are invaluable tools for predicting the structure and phase diagrams of crystalline solids based on a limited set of ab-initio calculations [18-22]. In particular, the ground states of a lattice model determine the 0K phase diagram of the system. However, the procedure to find and prove the exact ground state of a lattice model, defined on an arbitrary lattice with any interaction range and number of species remains an unsolved problem, with only a limited number of special-case solutions known in the literature [23-29].

In general systems, an approximation of the ground state is typically obtained from Monte Carlo simulations, which by their stochastic nature can prove neither convergence nor optimality. Thus, in light of the wide applicability of the generalized Ising model, an efficient approach to finding and proving its true ground states would not only resolve long-standing uncertainties in the field and give significant insight into the behavior of lattice models, but would also facilitate their use in ab-initio thermodynamics.

Until recently, we develop the strong links between ground state solving of cluster expansion with MAXSAT [30]. In this benchmark, we generated a Cluster expansion systems with by fitting Density Functional Theory (DFT) energies of $Li_xFe_{1-x}O_1$ systems with grid size of 5 by 5 by 5 with roughly about 100 types of interactions and try to test what is the best possible solution to the cluster expansion problem.

The general formulation of ground state problem of cluster expansion is

A lattice model is a set of fixed sites on which objects (spins, atoms of different types, atoms and vacancies, etc.) are to be distributed. Its Hamiltonian consists of coupling terms between pairs, triplets, and other groups of sites, which we refer to as "clusters". A formal definition of effective cluster interactions can be found in [12]. Before discussing the algorithmic details of our method, it is essential to establish a precise mathematical definition of a general lattice model Hamiltonian and the task of determining its ground states. The ground state problem can formally be stated as follows: Given a set of effective cluster interactions (ECI's) $J \in \mathbf{R}^{\mathbf{C}}$, where $\mathbf{C}$ is the set of interacting clusters and $\mathbf{R}$ is the set of real numbers, what is the configuration $s : \mathbf{D} \to \{0,1\}$, where $\mathbf{D}$ is the domain of configuration space, such that the global Hamiltonian H is minimized:

$$\min_s H = \min_s \lim_{N\to\infty} \frac{1}{(2N+1)^3} \sum_{(i,j,k)\in\{-N,\ldots,N\}^3} \sum_{\alpha\in\mathbf{C}} J_\alpha \prod_{(x,y,z,p,t)\in\alpha} s_{i+x,j+y,k+z,p,t} \quad (1)$$

In the Hamiltonian of Eq. (1), each $\alpha \in \mathbf{C}$ is an individual interacting cluster of sites. In turn, each site within α is defined by a tuple (x,y,z,p,t), wherein (x,y,z) is the index of the primitive cell containing the interacting site, p denotes the index of the sub-site to distinguish between multiple sub-lattices in that cell, and t is the species occupying the site. To discretize the interactions, we introduce the ''spin'' variables $s_{x,y,z,p,t}$, where $s_{x,y,z,p,t} = 1$ indicates that the pth sub-site of the (x,y,z) primitive cell is occupied by species t, and otherwise $s_{x,y,z,p,t} = 0$. The energy can be represented in terms of spin products, where each cluster α is associated with an ECI J_α denoting the energy associated with this particular cluster. To obtain the energy of the entire system, each cluster needs to be translated over all possible periodic images of the primitive cell, i.e., we have to consider all possible translations of the interacting cluster α, defined as a set of (x,y,z,p,t), by (i,j,k) lattice primitive cells

translations, yielding the spin product $\prod_{(x,y,z,p,t)\in\alpha} s_{i+x,j+y,k+z,p,t}$.

Finally, the prefactor $\frac{1}{(2N+1)^3}$ normalizes the energy to one lattice primitive cell, and the limit of N approaching infinity emphasizes our objective of minimizing the average energy over the entire infinitely large lattice. One remaining detail is that the Hamiltonian given in Eq. (1) is constrained such that that each site in the lattice must be occupied. For the sake of simplicity, lattice vacancies are included as explicit species in the Hamiltonian, so that all spin variables associated with the same site sum up to one:

$$\sum_{t\in\mathbf{c}(p)} s_{x,y,z,p,t} = 1 \;\forall (x,y,z,p)\in\mathbf{F} \qquad (2)$$

In Eq. (2), $\mathbf{F}$ is the set of all sites in the form of (x,y,z,p), and $\mathbf{c}(p)$ denotes the set of species that can occupy sub-site p. The domain of configuration space $\mathbf{D}$ can be formally defined as the set of all (x,y,z,p,t), with $t\in\mathbf{c}(p)$.

To further illustrate the notation introduced above, Figure 1 depicts an example of a two-dimensional lattice Hamiltonian for a square lattice with two sub-sites in each lattice primitive cell, i.e., $p\in\{0,1\}$. Each sub-site may be occupied by 3 types of species, so that $t\in\{0,1,2\}$, where $t=0$ shall be the reference (for example, vacancy) species. Hence, the energy of the system relative to the reference can be encoded into $t\in\{1,2\}$. Furthermore, the Hamiltonian shall be defined by only 2 different pairwise interaction types with the associated clusters $\alpha=\{(0,0,0,1,2),(1,2,0,0,1)\}$ and $\beta=\{(0,1,0,0,2),(0,0,0,1,2)\}$, and thus the set of all clusters is $\mathbf{C}=\{\alpha,\beta\}$. The first three of the five indices between "( )" brackets indicate the initial unit cell position, the forth index corresponds to the position in the unit cell (sub-site index), and the last index gives the species. The third component of the cell index (x,y,z) was retained for generality but set to 0 for this two-dimensional example. The example configuration shown in Figure 1 depicts three specific interactions: The interaction represented on the bottom left in in the figure is of type α with $(i,j,k)=(0,0,0)$, corresponding to the spin product $J_\alpha s_{0,0,0,1,2}\cdot s_{1,2,0,0,1}$. The interaction in the center of the figure also belongs to type α but with $(i,j,k)=(1,1,0)$, corresponding to the spin product $J_\alpha s_{0+1,0+1,0,1,2}\cdot s_{1+1,2+1,0,0,1} = J_\alpha s_{1,1,0,1,2}\cdot s_{2,3,0,0,1}$. Lastly, the interaction on the right represents an interacting β cluster, with $(i,j,k)=(3,0,0)$, yielding a spin product of $J_\beta s_{0+3,1,0,0,2} s_{0+3,0,0,1,2} = J_\beta s_{3,1,0,0,2} s_{3,0,0,1,2}$.

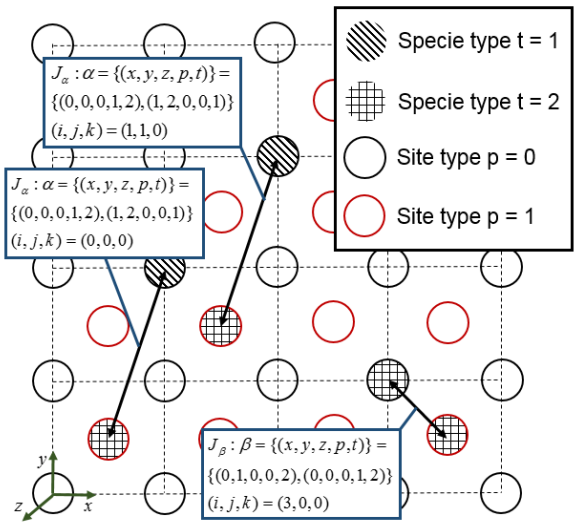


Figure 1: Illustration of a lattice Hamiltonian and examples of cluster interactions. The primitive unit of the lattice is indicated by a thin dashed line, and sites are represented by circles. Two different site types are distinguished by black and red borders, respectively. The non-vacancy species that can occupy the sites are indicated by two different hatchings.

II. MAXSAT ENCODING

To illustrate this approach, we consider the example of a binary 1D system with a positive point term J_0 and a negative nearest-neighbor interaction J_{NN}, on a 2-site unit cell. For this system, the transformation is:

$$\begin{aligned}
E &= \min_{\hat{s}_0,\hat{s}_1}\left(J_0\hat{s}_0 + J_0\hat{s}_1 + J_{NN}\hat{s}_0\hat{s}_1\right)\\
&= -\max\left(J_0(1-\hat{s}_0) - J_0 + J_0(1-\hat{s}_1) - J_0 - J_{NN}(\hat{s}_0\hat{s}_1)\right)\\
&= -\max\left(J_0(\neg\hat{s}_0) - J_0 + J_0(\neg\hat{s}_1) - J_0 + (-J_{NN})\left((1-\neg\hat{s}_0)\hat{s}_1\right)\right)\\
&= -\max\left(J_0(\neg\hat{s}_0) - J_0 + J_0(\neg\hat{s}_1) - J_0 + (-J_{NN})\hat{s}_1 + (-J_{NN})(1-\neg\hat{s}_0\hat{s}_1) - (-J_{NN})\right)\\
&= \left(2J_0 - J_{NN}\right) - \mathrm{MAXSAT}\left(J_0(\neg\hat{s}_0)\wedge J_0(\neg\hat{s}_1)\wedge(-J_{NN})(\hat{s}_1)\wedge(-J_{NN})(\hat{s}_0\vee\neg\hat{s}_1)\right)
\end{aligned} \qquad (5)$$

where the indicator variable $\hat{s}_i$ is now also a Boolean variable in the MAX-SAT setting, and the $\wedge$, $\vee$ and $\neg$ operators correspond to logical "and", "or" and "not" respectively. Note that, although in a MAX-SAT problem the coefficient of each clause needs to be positive, it is still possible to transform an arbitrary set of cluster interactions J_i into a proper MAX-SAT input, as in the example above.

The above encoding is the much much simpler version of our benchmark system. In our benchmark problems, we have many more types of interactions, for example, triplet, $J_{t1}s_0s_1s_2$ and quadruplets $J_{q1}s_0s_1s_2s_3$ etc.

III. REFERENCE

1. Li, X., et al., *Direct visualization of the Jahn–Teller effect coupled to Na ordering in Na5/8MnO2.* Nature materials, 2014.
2. Garbulsky, G.D. and G. Ceder, *Linear-programming method for obtaining effective cluster interactions in alloys from total-energy calculations: Application to the fcc Pd-V system.* Physical Review B, 1995. **51**(1): p. 67.
3. Struck, J., et al., *Engineering Ising-XY spin-models in a triangular lattice using tunable artificial gauge fields.* Nature Physics, 2013. **9**(11): p. 738-743.
4. Aidun, C.K. and J.R. Clausen, *Lattice-Boltzmann method for complex flows.* Annual review of fluid mechanics, 2010. **42**: p. 439-472.
5. Mueller, T. and G. Ceder, *Effective interactions between the N-H bond orientations in lithium imide and a proposed ground-state structure.* Physical Review B, 2006. **74**(13): p. 134104.
6. Kremer, K. and K. Binder, *Monte Carlo simulation of lattice models for macromolecules.* Computer Physics Reports, 1988. **7**(6): p. 259-310.
7. Seko, A., et al., *Prediction of ground-state structures and order-disorder phase transitions in II-III spinel oxides: A combined cluster-expansion method and first-principles study.* Physical Review B, 2006. **73**(18): p. 184117.
8. Rothman, D.H. and S. Zaleski, *Lattice-gas cellular automata: simple models of complex hydrodynamics*. Vol. 5. 2004: Cambridge University Press.
9. van de Walle, A., *A complete representation of structure-property relationships in crystals.* Nat Mater, 2008. **7**(6): p. 455-458.
10. Van der Ven, A. and G. Ceder, *Vacancies in ordered and disordered binary alloys treated with the cluster expansion.* Physical Review B, 2005. **71**(5): p. 054102.
11. Casola, F., et al., *Direct Observation of Impurity-Induced Magnetism in a Spin-1 2 Antiferromagnetic Heisenberg Two-Leg Spin Ladder.* Physical review letters, 2010. **105**(6): p. 067203.
12. Sanchez, J.M., F. Ducastelle, and D. Gratias, *Generalized cluster description of multicomponent systems.* Physica A: Statistical Mechanics and its Applications, 1984. **128**(1): p. 334-350.
13. Frisch, U., B. Hasslacher, and Y. Pomeau, *Lattice-Gas Automata for the Navier-Stokes Equation.* Physical Review Letters, 1986. **56**(14): p. 1505-1508.
14. Li, W., J.N. Reimers, and J.R. Dahn, *Crystal structure of Li x Ni 2-x O 2 and a lattice-gas model for the order-disorder transition.* Physical Review B, 1992. **46**(6): p. 3236.
15. Chan, M.K.Y., et al., *Cluster expansion and optimization of thermal conductivity in SiGe nanowires.* Physical Review B, 2010. **81**(17): p. 174303.
16. Ising, E., *Beitrag zur Theorie des Ferromagnetismus.* Zeitschrift für Physik, 1925. **31**(1): p. 253-258.
17. Ceder, G., *A derivation of the Ising model for the computation of phase diagrams.* Computational Materials Science, 1993. **1**(2): p. 144-150.
18. Hinuma, Y., Y.S. Meng, and G. Ceder, *Temperature-concentration phase diagram of P 2-Na x CoO 2 from first-principles calculations.* Physical Review B, 2008. **77**(22): p. 224111.
19. Ozoliņš, V., C. Wolverton, and A. Zunger, *Cu-Au, Ag-Au, Cu-Ag, and Ni-Au intermetallics: First-principles study of temperature-composition phase diagrams and structures.* Physical Review B, 1998. **57**(11): p. 6427.
20. Asta, M. and V. Ozoliņš, *Structural, vibrational, and thermodynamic properties of Al-Sc alloys and intermetallic compounds.* Physical Review B, 2001. **64**(9): p. 094104.
21. Burton, B.P. and A. van de Walle, *First principles phase diagram calculations for the octahedral-interstitial system.* Calphad, 2012. **37**(0): p. 151-157.
22. Zhou, F., T. Maxisch, and G. Ceder, *Configurational electronic entropy and the phase diagram of mixed-valence oxides: The case of Li x FePO 4.* Physical review letters, 2006. **97**(15): p. 155704.
23. Dublenych, Y.I., *Ground states of the Ising model on the Shastry-Sutherland lattice and the origin of the fractional magnetization plateaus in rare-earth-metal tetraborides.* Phys Rev Lett, 2012. **109**(16): p. 167202.
24. Dublenych, Y.I., *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.* Physical Review E, 2011. **84**(1).
25. Dublenych, Y.I., *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.* Physical Review E, 2011. **84**(6): p. 061102.
26. Teubner, M., *Ground states of classical one-dimensional lattice models.* Physica A: Statistical Mechanics and its Applications, 1990. **169**(3): p. 407-420.
27. Kanamori, J. and M. Kaburagi, *Exact Ground States of the Lattice Gas and the Ising Model on the Square Lattice.* Journal of the Physical Society of Japan, 1983. **52**(12): p. 4184-4191.
28. Kaburagi, M. and J. Kanamori, *Ground State Structure of Triangular Lattice Gas Model with up to 3rd Neighbor Interactions.* Journal of the Physical Society of Japan, 1978. **44**(3): p. 718-727.
29. Finel, A. and F. Ducastelle, *On the phase diagram of the FCC Ising model with antiferromagnetic first-*

neighbour interactions. EPL (Europhysics Letters), 1986. **1**(3): p. 135.
30. Huang, W., et al., *Finding and proving the exact ground state of a generalized Ising model by convex optimization and MAX-SAT.* Physical Review B, 2016. **94**(13): p. 134424.

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[1].

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[2] 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

where x is the sum of the minimum fill-ins of each ignored cycle. Hence for each of the MaxSAT instances, the minimum fill-in of its underlying graph is equal to the sum of the optimal cost of the MaxSAT instance and the value x.

[1] https://pacechallenge.wordpress.com/pace-2017/track-b-minimum-fill-in/

[2] Obtained from http://www.staff.science.uu.nl/~bodla101/treewidthlib/.

References

[1] J. Berg and M. Järvisalo, "SAT-based approaches to treewidth computation: An evaluation," in *Proc. ICTAI*. IEEE Computer Society, 2014, pp. 328–335.

[2] H. L. Bodlaender and A. M. C. A. Koster, "Safe separators for treewidth," *Discrete Mathematics*, vol. 306, no. 3, pp. 337–350, 2006.

[3] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[4] M. Samer and H. Veith, "Encoding treewidth into SAT," in *Proc. SAT*, ser. Lecture Notes in Computer Science, vol. 5584. Springer, 2009, pp. 45–50.

[5] M. Yannakakis, "Computing the minimum fill-in is NP-complete," *SIAM Journal on Algebraic Discrete Methods*, vol. 2, no. 1, pp. 77–79, 1981.

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 $\text{SIZE}(CB) = \sum_{j=1}^{m}(u_j - l_j)$, 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, and t, the minimum-width confidence band problem, 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 $CB^* \in \arg\min \text{SIZE}(CB)$ over those $CB = (l, u)$ for which (i) $\sum_{i=1}^{n} I[\text{ERROR}(x^i, CB) > s] \leq k$ and (ii) $\sum_{i=1}^{n} I[x_j^i < l_j \vee x_j^i > u_j] \leq t$ for all $1 \leq j \leq m$, is a solution to MWCB(k, s, t).

III. MaxSAT Encoding

For an exact description of the MaxSAT encoding for 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 maxtemp-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 data[1], and is obtained from 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

MinWidthCB_dataset_n_m_Kk_Ss_Tt.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, and t are the values of the input parameters to 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

[1] R. Asín, R. Nieuwenhuis, A. Oliveras, and E. Rodríguez-Carbonell, "Cardinality networks: a theoretical and empirical study," *Constraints*, vol. 16, no. 2, pp. 195–221, 2011.

[2] J. Berg, E. Oikarinen, M. Järvisalo, and K. Puolamäki, "Minimum-width confidence bands via constraint optimization," in *Proc. CP*, ser. Lecture Notes in Computer Science, 2017, to appear.

[3] J. Korpela, K. Puolamäki, and A. Gionis, "Confidence bands for time series data," *Data Mining and Knowledge Discovery*, vol. 28, no. 5–6, pp. 1530–1553, 2014.

[4] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[5] G. B. Moody and R. G. Mark, "The impact of the MIT-BIH arrhythmia database," *IEEE Engineering in Medicine and Biology Magazine*, vol. 20, no. 3, pp. 45–50, 2001.

[1] http://archive.ics.uci.edu/ml/datasets/Individual+household+electric+power+consumption

Solver Index

Benchmark Index

Author Index