Jornada de Seguimiento de Proyectos, 2005
Programa Nacional de Tecnologías Informáticas

Resolución de Problemas Combinatorios con Restricciones Suaves y Fuzzy Utilizando Algoritmos de Satisfactibilidad TIC2003-00950

Teresa Alsinet *
Universidad de Lleida

Abstract

The main goal of our project is to solve, using SAT technology, combinatorial problems that have both soft and hard constraints in its definition. At the same time such constraints can be crisp or fuzzy. In our approach we will use the language (extended with weights and preference degrees when necessary) of Boolean and many-valued formulas to model the problems. Additionally, we will use (maximum) satisfiability algorithms to find the solutions. In other words, the goal of our project is to design and implement a generic problem solving method for OCSPs (over-constrained constraint satisfaction problem) using (maximum) Boolean and many-valued algorithms. To experimentally evaluate our algorithms we will build a benchmark repository with realistic OCSP instances and design and implement random instance generators.

Keywords: Satisfiability algorithms, soft and fuzzy constraints.

1 Context and objectives

A constraint satisfaction problem (CSP) is defined as a tuple (V, D, C), where $X = \{X_1, \ldots, X_n\}$ is a set of variables, $D = \{D_1, \ldots, D_n\}$ is a set of domains in which D_i denotes the finite set of possible values that can be assigned to the variable X_i and $C = \{C_1, \ldots, C_r\}$ is a set of constraints where each constraint denotes a subset of the Cartesian product of the domains of the variables associated with the constraint. A solution to the CSP is an assignment of values to the variables such that all the constraints are satisfied. We have different problems: the decision problem (check if at least one solution exists), finding all solutions and finding the best solution under some quality criteria. For finding solutions of a CSP, systematic and local search algorithms have been developed [17]. A particular class of CSPs, widely studied, is the Satisfiability problem of Boolean CNF formulas (SAT). In this problem variables are boolean, and the constraints are always disjunctions of boolean literals. Different algorithms have been developed for solving SAT: Chaff, RelSAT, WalkSAT and Satz [12, 13, 14] (among others).

*Email: tracy@eps.udl.es

These algorithms, except WalSAT that is a local search algorithm, are systematic algorithms based on the Davis-Putnam algorithm. The differences between both techniques are the data structures used for fast logical unit-propagation, the intelligent branching heuristics used and the kind of backtracking used (chronological versus conflict directed back-jumping). One problem that is between CSP and SAT is the Many-valued SAT problem. This problem is like SAT, but now variables are not boolean because we consider a truth-value set of any cardinality and we have many-valued literals. A many-valued literal is an expression of the form $S : p$ where p is a propositional variable and S denotes the subset of the truth-value set that satisfies the literal. This SAT problem for Many-valued formulas has been widely studied by our research group.

Solving combinatorial problems using CSPs as a constraint language, has one main drawback: a solution of a CSP has to satisfy all the constraints. However, in may real situations, we cannot find such a solution, but there may be some assignments that satisfy *almost* all the constraints and can be considered as acceptable solutions because the constraints not satisfied may not be as *important* as the satisfied by the assignment. These unsolvable CSPs are known as over-constrained CSPs (OCSPs). A solution for a OCSP is an assignment that satisfies in the *best* possible way the constraints of the problem. In this context, we have to consider different kinds of constraints: (i) crisp: constraints involving precise knowledge or information, (ii) fuzzy: constraints involving vague knowledge, (iii) hard: constraints that have to be satisfied in every solution of the OCSP and (iv) soft: constraints that are not needed to be satisfied in every solution of the CSP. In the context of OCSP, there are two relevant problems related to the classical SAT problem: Max-SAT and weighted Max-SAT [15, 16]. Max-SAT is the problem of finding a boolean assignment to the variables of a CNF formula such that the maximum possible number of clauses is satisfied. The clauses are equally important, so we do not have here hard or soft constraints. In contrast, in the weighted Max-SAT, we associate to every clause a "weight" that in some way defines the importance of the clause. Now the goal is to find an assignment with the maximum value of the sum of the weight of the satisfied clauses. The current existing implemented algorithms for Max-SAT and Weighted Max-SAT are not as efficient as the ones for SAT. One reason could be that they are harder to solve than SAT. For example, 2SAT is polynomially solvable, but the corresponding Max-2SAT and Weighted Max-2SAT are NP-hard problems.

The main goal of our project is to solve, using SAT technology, combinatorial problems that have both soft and hard constraints in its definition. At the same time such constraints can be crisp or fuzzy. In our approach we will use the language (extended with weights and preference degrees when necessary) of Boolean and many-valued formulas to model the problems. Additionally, we will use (maximum) satisfiability algorithms to find the solutions. In other words, the goal of our project is to design and implement a generic problem solving method for OCSPs using (maximum) Boolean and many-valued algorithms. To experimentally evaluate our algorithms we will build a benchmark repository with realistic OCSP instances and design and implement random instance generators. Finally, in order to model constraints dealing with fuzzy information and uncertainty (preferences) we have to formalize a language being able to deal with both kinds of human indeterminism; define realistic problems for which such approach will be a suitable representation model; and develop exact satisfiability branch-and-bound algorithms.

To be precise the objectives of the project can be summarized a follows:

- The design and implementation of efficient Max-SAT algorithms for many-valued clausal formulas, and the improvement of exact algorithms for Boolean formulas by means of the use of some new data structures for modeling formulas, new search strategies, non chronological backtracking, and some learning techniques for clauses. The aim is to reduce the search space that should be compute in order to find the optimal solution.

- The design and implementation of a generic method for solving OCSP, based on satisfiability algorithms for Boolean and many-valued formulas. At the same time such constraints can be both hard or soft, and crisp or fuzzy.

- The development of a benchmark repository build from realistic combinatorial problems. The benchmark repository will allow us to evaluate the generic method for solving OCSP.

- The analysis, from logical and computational points of view, of different propositional languages for representing hard, soft, crisp and fuzzy constraints.

2 Contributions of the project

2.1 Max-SAT solvers

When we started the project, there were a wide variety of extremely efficient SAT solvers, but very few exact Max-SAT solvers implementing the Davis-Putnam procedure with a branch and bound scheme [3, 8, 10]. Nowadays, there are a considerable number of fast Max-SAT solvers [5, 6, 9, 19, 20] and, roughly speaking, a Max-2SAT instance with 100 variables is currently solved 1000 times faster than five years ago. Basically, the best existing solvers improve previous solvers in the following aspects: (i) efficient data structures for representing and manipulating formulas, (ii) lower bound computation methods that provide lower bounds of better quality, (iii) clever variable selection heuristics, and (iv) powerful inference techniques.

Our first contribution was to show that Borchers and Furman algorithm (BF) could be improved by incorporating a new version of the Jeroslow-Wang rule [11], and considering inconsistency counts in the computation of the lower bound [4, 5]. We refer to that solver as AMP.

We then focused our research on developing a faster solver than AMP. Our first steps was to define lazy data structures inspired by the two-watched literals data structures of zChaff [21]. We defined very simple data structures that allow one to efficiently traverse the search space and, at the same time, to apply inference techniques and compute lower bounds very quickly. The only constraint of our data structures is that we have to apply a static variable selection heuristic. We refer to this solver as Lazy (Publications [1] and [2]). Besides the data structures, Lazy implements an original lower bound of better quality and applies neighborhood resolution as a preprocessing technique. We have developed both a weighted (Publication [1]) and an unweighted version (Publication [2]).

In our last contribution (Publication [18]), we have designed and implemented a Max-SAT solver that makes a novel use of unit propagation: it is used to compute lower bounds. Our approach subsumes existing lower bound computation methods like those based on inconsistency counts. It is worth to point out that our solver is the best performing state-of-the-art solver, at least, on randomly generated Max-2SAT and Max-3SAT instances.

We are currently working in two directions: On the one hand, we are incorporating more powerful inference techniques into our solvers. In particular, we have defined a unit resolution rule that ensures optimality (the standard unit resolution rule discards optimal solutions). On the other hand, we are designing and implementing Max-SAT solvers for multiple-valued clausal forms.

2.2 Solving over-constrained solvers with SAT technology

We developed a new generic problem solving approach for over-constrained problems based on Max-SAT. To this end, we defined a Boolean clausal form formalism, called *soft CNF formulas*, that deals with blocks of clauses instead of individual clauses, and that allows one to declare each block either as *hard* (i.e., must be satisfied by any solution) or *soft* (i.e., can be violated by some solution). An optimal solution for a given soft CNF formula consists of finding a truth assignment that satisfies all the hard blocks and the maximum number of soft blocks.

We designed and implemented two solvers for Soft CNF formulas (Publications [5] and [6]). Our solvers are branch and bound algorithms equipped with original lazy data structures, powerful inference techniques, lower bounds of good quality, and original variable selection heuristics. We tested our solvers on a representative sample of instances (random 2-SAT, Max-CSP, graph coloring, quasigroup completion, and pigeon hole). Since we exploit more the structure of the problem than Max-SAT approaches, both our inference techniques and our lower bound computation method give raise to better performance profiles. The gains, compared with Max-SAT, are up to two orders of magnitude. Besides, we compared our solvers with Max-CSP solvers, and observed that our solvers are very competitive (an in many cases superior) to Max-CSP solvers.

We are currently working on the extension of the formalism of Soft CNF formulas in order to include fuzzy constraint; only crisp constraints are treated in the existing solvers of Soft CNF formulas.

2.3 SAT and CSP problems typical-case complexity

Usually, one defines the complexity of a Satisfiability problem over clausal forms (CNFs) with Many-valued (signed) literals in which all the literals have the same syntactic form [7]. However, when using these signed CNF formulas as a constraint language to encode some real-world combinatorial problem (frequency assignment, scheduling, timetabling, ...) not all the clauses of the resulting CNF have literals with the same form.

So, beyond understanding the worst-case complexity of signed CNF formulas with a rigid syntactic structure, in our work (Publication [4]) we start the study of the effect of the presence of different fractions of clauses in a Signed CNF with a particular structure in their signed literals.

We consider two different kinds of signed literals: signed literals whose classical Many-valued satisfiability problem (the problem in which all the literals have the same form) is tractable (polynomially solvable) and signed literals for which the same problem is NP-complete. Then, we consider the Satisfiability problem for CNFs in which a fraction p of the clauses contain "NP-complete literals" (from the second kind) and a fraction (1-p) of the clauses contain "tractable literals" (from the first kind). In the two combinations we have tested in the paper, what we find is that even if the problems are NP-complete for any $p > 0$, the typical-case

complexity (average complexity) of solving a test-set of instances is not the same for any value of p. For one of the problems, there is a critical value of p that separates two very different regions: one in which all the problems are solved with polynomial average time and the other in which all the problems are solved with exponential average time.

In conclusion, when facing the question of which constraint language one should use to encode a combinatorial problem one needs to solve, looking only at worst-case complexity results is not a sufficient information to take into account.

2.4 A benchmark for distributed CSPs

A distributed CSP problem is a Constraint satisfaction problem which is divided in different subproblems. Every such subproblem belongs to an unique "agent" that is the only one allowed to change the values of the variables of the subproblem. We have two different kinds of constraints: intra-agent constraints (constraints between variables of a same subproblem) and inter-agent constraints (constraints between variables of different subproblems).

In our work (Publication [8]) we consider the worst-case and typical-case complexity of solving the distributed constraint satisfaction problem (DisCSP). We perform the study by first introducing two realistic benchmark problems: SensorDCSP and GSensorDCSP, and studying their worst-case complexity. These problems are based on a real-world application that arises in the context of networked distributed resource allocation systems for the tracking of mobile objects. To perform a realistic analysis of the typical-complexity, we have used a discrete-event network simulator, which allows us to model the impact of different network traffic conditions of the performance of the (distributed) algorithms. We use two well known DisCSP algorithms, ABT and AWC, although as the result of one of our empirical results we propose an improvement for the ABT algorithm.

One important result obtained in the typical-case complexity analysis we have performed is that random delays (due to network traffic or in some cases actively introduced by the agents) can affect the performance of the algorithms in very different ways, because we have found that AWC is considerably more robust to the variance of the delays than ABT.

As a consequence, when designing DisCSP algorithms one wants to use in real communication networks, one should take into account the sensibility of the algorithms to the networks conditions that the algorithm is going to find.

2.5 On the formalization of a propositional language for representing fuzziness and uncertainty

In the last years defeasible argumentation frameworks have proven to be a successful approach to formalizing qualitative, commonsense reasoning from incomplete and potentially inconsistent knowledge. Defeasible Logic Programming (or DeLP) [18] is one of such formalisms, combining results from defeasible argumentation theory and logic programming. Although DeLP has proven to be a suitable framework for building real-world applications that deal with incomplete and contradictory information in dynamic domains, it cannot deal with explicit uncertainty, nor with vague knowledge, as defeasible information is encoded in the object language using "defeasible rules".

In the framework of the project we have formalized P-DeLP (Publications [14] and [16]), a new logic programming language that extends original DeLP capabilities for qualitative

reasoning by incorporating the treatment of possibilistic uncertainty and fuzzy knowledge. Such features are formalized on the basis of PGL [1, 2], a possibilistic logic based on the Horn-rule fragment of Gödel fuzzy logic. In PGL formulas are built over fuzzy propositional variables and the certainty degree of formulas is expressed by means of a necessity measure. In a logic programming setting, the proof method for PGL is based on a complete calculus for determining the maximum degree of possibilistic entailment of a fuzzy goal.

In the context of complex logic-programming frameworks (like the one provided by extended logic programming), PGL lacks of an adequate mechanism to handle contradictory information, as conflicting derivations can be found. In P-DeLP such conflicts are solved using an argument-based inference engine. Formulas are supported by arguments, which have an attached necessity measure associated with the supported conclusion. The ultimate answer to queries is given in terms of warranted arguments, computed through a dialectical analysis. One particularly interesting feature of P-DeLP is the possibility of defining aggregated preference criteria by combining the necessity measures associated with arguments with other syntax-based criteria.

Finally, Defeasible argumentation in general and P-DeLP in particular provide a way of modeling non-monotonic inference. From a logical viewpoint, capturing defeasible inference relationships for modeling argument and warrant is particularly important, as well as the study of their logical properties. In our work (Publication [15]) we have analyzed two non-monotonic operators for P-DeLP which model the expansion of a knowledge base by adding new weighed facts associated with argument conclusions and warranted literals, resp. Different logical properties for the proposed expansion operators have been studied and contrasted with a traditional SLD-based Horn logic. One particularly interesting feature is that this analysis provides useful comparison criteria that can be extended and applied to other argumentation frameworks.

3 Publications and collaborations

The results obtained in the framework of the project have already been published as journal articles or in conference proceedings:

[1] T. Alsinet, F. Manyà and J. Planes. *Improved Exact Solver for Weighted Max-SAT.* Proceedings of the Eighth International Conference on the Theory and Applications of Satisfiability Testing, SAT-2005, St. Andrews, Scotland, LNCS 3569, pp. 371-378, 2005.

[2] T. Alsinet, F. Manyà and J. Planes. *A Max-SAT Solver with Lazy Data Structures.* Proceedings of the IX Ibero-American Conference on Artificial Intelligence, IBERAMIA-2004, Puebla, México, Springer LNCS 3315, pp. 334-342, 2004.

[3] C. Ansótegui and F. Manyà. *Mapping Problems with Finite-Domain Variables to Problems with Bolean Variables.* Proceedings of the Seventh International Conference on the Theory and Applications of Satisfiability Testing, SAT-2004, Vancouver, Canada, Springer LNCS 3542, pp. 111-119, 2004.

[4] C. Ansótegui, R. Béjar, A. Cabiscol and F. Manyà. *The Interface between P and NP in Signed CNF Formulas.* Proceedings of the 34th International Symposium on Multiple-Valued Logics (ISMVL), Toronto, Canada. IEEE CS Press, pp. 251-256, 2004.

[5] J. Argelich and F. Manyà. *Solving Over-constrained Problems with SAT Technology.* Proceedings of the Eighth International Conference on the Theory and Applications of satisfiability Testing, SAT-2005, St. Andrews, Scotland, LNCS 3569, pp.1-15, 2005.

[6] J. Argelich and F. Manyà. *Solving Over-Constrained Problems with Max-SAT Algorithms.* Proceedings of the Workshop on Modelling and Solving Problems with Constraints, 16th European Conference on Artificial Intelligence, ECAI-2004, Valencia, Spain, pp. 116-124, 2004.

[7] J. Argelich and F. Manyà. *An Exact Max-SAT Solver for Over-Constrained Problems.* Proceedings of the Workshop on Preferences and Soft Constraints, 10th International Conference on Principles and Practice of Constraint Programming, CP-2004, Toronto, Canada, 2004

[8] R. Béjar, C. Domslak, C. Fernández, C. Gomes, B. Krishnamachari, B. Selman and M. Valls. *Sensor Networks and Distributed CSP: Communication, Computation and Complexity.* Artificial Intelligence, 161: 117-147, 2005.

[9] M. Capobianco, C.I. Chesñevar and G. Simari. *An Argument-based Framework to Model an Agent's Beliefs in a Dynamic Environment.* First Internacional Workshop on Argumentation in Multiagent Systems. Springer LNCS, Vol. 3366, pp.95-110, 2005.

[10] C. Chesñevar, R. Brena and J. Aguirre. *Knowledge Distribution in Large Organizations Using Defeasible Logic Programming.* 18th Canadian Conference in Artificial Intelligence, Victoria, British Columbia, Canada. Springer LNAI 3501, pp. 244-256, 2005.

[11] C.I. Chesñevar and A.G. Maguitman. *An Argumentative Approach to Assesing Natural Language Usage based on the Web Corpus.* European Conference on Artificial Intelligence (ECAI 2004), pp. 581-585. Valencia, Spain, 2004.

[12] C.I. Chesñevar and A.G. Maguitman. *Combining Argumentation and Web Search Technology: Towards a Qualitative Approach for Ranking Results.* Intl. Journal of Advanced Computational Intelligence & Intelligent Informatics, Vol. 9, No. 1, pp. 53-60, 2005.

[13] C.I. Chesñevar, A.G. Maguitman and G. Simari. *A first Approach to Argument-Based Recommender Systems based on Defeasible Logic Programing.* International Workshop on Non Monotonic Reasoning (NMR 2004), pp. 109-117 Whistler, Canada, 2004.

[14] C. Chesñevar, G. Simari, T. Alsinet and L. Godo. *A Logic Programming Framework for Possibilistic Argumentation with Vague Knowledge.* Uncertainty in Artificial Intelligence (UAI-2004), pp. 76-84, ISBN 0-9749039-0-6. Banff, Canada, 2004.

[15] C. Chesñevar, G. Simari, L. Godo and T. Alsinet. *Argument-based Expansion Operators in Possibilistic Defeasible Logic Programming: Characterization and Logical Properties.* 8th European Conference on Symbolic and .Qualitative Approaches to Reasoning with Uncertainty (ECSQARU-2005), Barcelona, Spain. LNAI 3571, pp. 353-365, 2005.

[16] C. Chesñevar, G. Simari, T. Alsinet and L. Godo. *Modelling Agent Reasoning in a Logic Programming Framework for Possibilistic Argumentation.* 2nd European Workshop on Multiagent Systems (EUMAS 2004), pp. 135-142. Barcelona, Spain, 2004.

[17] S.A. Gómez and C.I. Chesñevar. *Integrating Defeasible Argumentation with Fuzzy ART Neural Networks for Pattern Classification.* Journal of Computer Science and Technology, Vol. 4, No. 1, pp.45-57, 2004.

[18] C. M. Li, F. Manyà, and J. Planes. *Exploiting unit propagation to compute lower bounds in branch and bound max-sat solvers.* In 11th International Conference on Principles and Practice of Constraint Programming, CP-2005, Sitges, Spain. Springer LNCS, 2005.

Some of the above publications are the result of collaborations with national and international researchers. To design and implement Max-SAT algorithms we have collaborated with Chu Min Li of the University of Picardie. To study the heavy-tails phenomena and distributed CSPs we have collaborated with Carla Gomes and Bart Selman of the University of Cornell. Finally, to formalize a language for representing fuzzy and uncertain information in the framework of argumentative systems we have collaborated with Lluís Godo of the IIIA-CSIC and Guillermo Simari of the University Nacional del Sur.

Finally, we would remark that Jordi Planes and Josep Argelich are developing their PhD. Thesis in the framework of the project.

References

[1] Alsinet, T. and Godo, L. A complete calculus for possibilistic logic programming with fuzzy propositional variables. In *Proc. of the UAI-2000 Conference*, pages 1–10, 2000.

[2] Alsinet, T. and Godo, L. A proof procedure for possibilistic logic programming with fuzzy constants. In *Proc. of the ECSQARU-2001 Conference*, pages 760–771, 2001.

[3] J. Alber, J. Gramm, and R. Niedermeier. Faster exact algorithms for hard problems: A parameterized point of view. In *25th Conf. on Current Trends in Theory and Practice of Informatics*, LNCS, pages 168–185. Springer-Verlag, November 1998.

[4] T. Alsinet, F. Manyà, and J. Planes. Improved branch and bound algorithms for Max-2-SAT and weighted Max-2-SAT. In *Proceedings of the Catalonian Conference on Artificial Intelligence*, 2003.

[5] T. Alsinet, F. Manyà, and J. Planes. Improved branch and bound algorithms for Max-SAT. In *Proceedings of the 6th International Conference on the Theory and Applications of Satisfiability Testing, SAT-2003, Portofino, Italy*, 2003.

[6] T. Alsinet, F. Manyà, and J. Planes. Improved exact solvers for weighted Max-SAT. In *Proceedings of the 8th International Conference on the Theory and Applications of Satisfiability Testing, SAT-2005, St. Andrews, Scotland, UK*, pages 371–367. Springer LNCS 3569, 2005.

[7] R. Bejar, A. Cabiscol, C. Fernandez, F. Manya and C. Gomes. Capturing Structure with Satisfiability. *7th International Conference of Constaint Programming, CP-2001.* Paphos. Cyprus. LNCS 2239 pp. 137-152, 2001.

[8] B. Borchers and J. Furman. A two-phase exact algorithm for MAX-SAT and weighted MAX-SAT problems. *Journal of Combinatorial Optimization*, 2:299–306, 1999.

[9] S. de Givry, J. Larrosa, P. Meseguer, and T. Schiex. Solving Max-SAT as weighted CSP. In *9th International Conference on Principles and Practice of Constraint Programming, CP-2003, Kinsale, Ireland*, pages 363–376. Springer LNCS 2833, 2003.

[10] E. Freuder and R. Wallace. Partial constraint satisfaction. *Artificial Intelligence*, 58:21–71, 1992.

[11] R. G. Jeroslow and J. Wang. Solving propositional satisfiability problems. *Annals of Mathematics and Artificial Intelligence*, 1:167–187, 1990.

[12] C.M. Li and Anbulagan. Look-ahead versus Look-back for Satisfiability Problems. *International Conference on Principles and Practice of Constraints Programming, CP-1997*, 1997.

[13] M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang and S. Malik. Chaff: Engineering an Efficient SAT Solver. 39th Design Automation Conference, 2001.

[14] B. Selman, H. Kautz and B. Cohen. Noise Strategies for Improving Local Search. *National Conference on Artificial Intelligence, AAAI-96*, 1996.

[15] S. Joy, J. Mitchell and B. Borchers Solving MAX-SAT and Weighted MAX-SAT problems using Branch-and-Cut. *Journal of Combinatorial Optimization*, 2003.

[16] B. Borchers and J. Furman. A two-phase exact algorithm for MAX-SAT and weighted MAX-SAT problems. *journal of Combinatorial Optimization*, 2:299–306, 1999.

[17] I. Miguel and Q. Shen (2001). *Solution Techniques for Constraint Satisfaction Problems: Foundations.* Artificial Intelligence Review 15, pp. 243-267.

[18] García, A. and Simari, G. Defeasible Logic Programming: An Argumentative Approach. *Theory and Practice of Logic Programming*, 4(1):95–138, 2004.

[19] Z. Xing and W. Zhang. Efficient strategies for (weighted) maximum satisfiability. In *Proceedings of CP-2004*, pages 690–705, 2004.

[20] H. Zhang, H. Shen, and F. Manya. Exact algorithms for MAX-SAT. In *4th Int. Workshop on First order Theorem Proving*, June 2003.

[21] L. Zhang, C. Madigan, M. Moskewicz, and S. Malik. Efficient conflict driven learning in a Boolean satisfiability solver. In *International Conference on Computer Aided Design, ICCAD-2001, San Jose/CA, USA*, pages 279–285, 2001.