Problem Transformations and Algorithm Selection for CSPs

Barry Hurley and **Barry O'Sullivan***
Cork Constraint Computation Centre
Department of Computer Science, University College Cork, Ireland
b.hurley@4c.ucc.ie

1 Introduction

A constraint satisfaction problem (CSP) provides a powerful paradigm to model a large number of practical problems such as scheduling, planning, vehicle routing, configuration, network design, routing and wavelength assignment. An instance of a CSP is represented by a set of variables, each of which can be assigned a value from its domain. The assignments to the variables must be consistent with a set of constraints, where each constraint limits the values that can be assigned to variables. Finding a solution to a CSP is typically done using systematic search, based on backtracking. Because the general problem is NP-Complete, systematic search algorithms have exponential worst-case run times, which has the effect of limiting the scalability of these methods. However, the development of effective heuristics and a wide variety of solvers means than many problems become tractable. Further progress has been made on algorithms that are tailored to a particular problem domain. These algorithms have been specifically designed and tuned to perform well on a single class of instances but may perform poorly on many other problem domains. This has led to the development of solver portfolios that exploit the variation among solvers to identify the best solver for a given instance. An overview of our current contributions in this field is given in Section 3.

Another option is to translate the problem to an alternative representation, for example the satisfiability problem (SAT). Several polynomial-time transformations from CSP to SAT are known [Prestwich, 2009]. The practicality of using such transformations to solve CSPs using SAT has been demonstrated by the development of a number of SAT-based CSP solvers. `Sugar`, `Azucar` and `CSP2SAT4J` are three examples of such solvers which have proven to be highly competitive in the most recent CSP solver competition. `Sugar` [Tamura *et al.*, 2009] encodes the CSP to SAT using a specific encoding, known as the order encoding. `Azucar` [Tanjo *et al.*, 2012] is a related SAT-based CSP solver that uses the compact order encoding. `CSP2SAT4J` [Le Berre and Lynce, 2008] uses static rules to choose either the direct or the support encoding for each constraint. However, each of these use a single predefined SAT solver to solve the encoded CSP instances.

*This work is supported by Science Foundation Ireland Grant 10/IN.1/I3032 and FP7 FET-Open Grant 284715.

2 On the Best Encoding-Solver Combination

Little is known about how to choose the right SAT encoding/solver combination when solving CSPs, it is this gap that this research attempts to address. We examined the utility of multiple encodings and multiples solvers, which exhibits superiority to a single encoding and solver. Specifically, we evaluated a variety of encoding/solver combinations for solving CSPs and show that significant performance benefits can be obtained if the correct pairing is made at an instance level.

Extensive empirical investigations show that the choice of encoding and solver should be made in combination and not independently when solving SAT-encoded CSPs. When analyzing the performance for a particular SAT solver for CSP solving, the conclusions drawn as to which encoding delivers the best performance cannot be applied to a different solver. Similarly, investigating the choice of solver for an encoding does not yield reliable information for other encodings.

No single solver performs best across all encodings. Similarly, no single encoding is the best choice for all solvers. By changing the encoding or solver used, many problem instances can be solved several orders-of-magnitude faster in comparison to existing SAT-based CSP solvers. When choosing a SAT encoding to convert CSPs, the solver that should be used to solve the SAT problem needs to be taken into account to achieve good performance. Similarly, when choosing a solver the encoding needs to be taken into account. This offers a novel perspective on using SAT for constraint solving.

3 A Hierarchical Portfolio of Solvers and Transformations

There has been a significant increase in the number of solving systems that have been developed in recent years. Some aim to be applicable to a diverse set of problems, others have been developed for specific problem classes. Solver competitions attempt to evaluate these systems from a high level on a broad range of benchmark problems. However a system that performs very well on a single class of problems, may perform poorly when evaluated over the entire range of benchmarks [Gomes and Selman, 2001]. This has led to the development of solver portfolios which have consistently dominated the recent competitions in the SAT and CSP domains. CPHYDRA [O'Mahony *et al.*, 2008], SATZILLA [Xu *et al.*, 2008] and ISAC [Kadioglu *et al.*, 2010] are three examples of

such portfolios.

Portfolios such as these operate on the basis that the best on-average solver can be out-performed by a portfolio of possibly slower on-average solvers [Gomes and Selman, 2001]. Our initial venture [Hurley and O'Sullivan, 2012] into this field studied a variety of adaptation schemes for a family of case-based reasoning algorithm portfolios for the SAT problem. Our results demonstrate that the choice of adaptation scheme is important for performance with schemes that consider run time rather that relative ranking gives superior performance. We demonstrated that a case-based reasoning approach to this task is competitive, and out-performs individual high-performing SAT solvers in a wide variety of problem domains.

We continued this line of research in combination with work on problem transformations that demonstrated the potential gains from taking multiple transformations and solvers into account when solving CSPs using SAT. We have developed a portfolio approach that does not rely on a single problem representation or set of solvers, but leverages our ability to convert between problem representations to increase the space of possible solvers. To the best of our knowledge, this is the first time a portfolio approach like this has been proposed.

We demonstrated the significant performance improvements that can be achieved empirically on a large set of diverse benchmarks with a portfolio based on a range of different state-of-the-art solvers. We have investigated a range of different CSP to SAT encodings and evaluated the performance of a large number of machine learning approaches and algorithms.

4 Future Work

Our initial line of research has shown that, to achieve the best performance on a constraint satisfaction problem, it may be beneficial to translate it to a satisfiability problem. For this translation, it is important to choose both the encoding and satisfiability solver in combination. By doing so, the contrasting performance among solvers on different representations of the same problem can be exploited. In taking all these considerations into account, performance can be improved significantly compared to restricting the portfolio to a single problem representation.

We have translated constraint satisfaction problems to SAT, conversely there exist translations from SAT to CSP. An obvious direction would be to look at these translations in combination with multiple CSP solvers. These concepts could also generalise to many other problem domains where transformations, like CSP to SAT, exist.

This work is inspired by the Numberjack platform [Hebrard *et al.*, 2010][1]. Numberjack provides a flexible interface to several frameworks for combinatorial optimization. It offers a common interface for problem specification and the ability to solve such a problem using mxed integer programming, satisfiability, or constraint programming techniques. We aim to integrate our work on problem transformation and algorithm selection into Numberjack with a view to providing a powerful and flexible framework for combinatorial optimization problems.

[1]The Numberjack website: `http://numberjack.ucc.ie/`

References

[Gomes and Selman, 2001] Carla P. Gomes and Bart Selman. Algorithm Portfolios. *Artificial Intelligence*, 126(1-2):43–62, 2001.

[Hebrard *et al.*, 2010] Emmanuel Hebrard, Eoin O'Mahony, and Barry O'Sullivan. Constraint Programming and Combinatorial Optimisation in Numberjack. In *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, 7th International Conference, CPAIOR 2010*, pages 181–185, 2010.

[Hurley and O'Sullivan, 2012] Barry Hurley and Barry O'Sullivan. Adaptation in a CBR-Based Solver Portfolio for the Satisfiability Problem. In *Case-Based Reasoning Research and Development - 20th International Conference*, Lecture Notes in Computer Science, pages 152–166. Springer, 2012.

[Kadioglu *et al.*, 2010] Serdar Kadioglu, Yuri Malitsky, Meinolf Sellmann, and Kevin Tierney. ISAC - Instance-Specific Algorithm Configuration. In *Proceedings of the 14th European Conference on Artificial Intelligence, ECAI*, 2010.

[Le Berre and Lynce, 2008] Daniel Le Berre and Inês Lynce. CSP2SAT4J: A Simple CSP to SAT Translator. In *Proceedings of the 2nd International CSP Solver Competition*, 2008.

[O'Mahony *et al.*, 2008] Eoin O'Mahony, Emmanuel Hebrard, Alan Holland, Conor Nugent, and Barry O'Sullivan. Using Case-based Reasoning in an Algorithm Portfolio for Constraint Solving. *Proceeding of the 19th Irish Conference on Artificial Intelligence and Cognitive Science*, 2008.

[Prestwich, 2009] Steven David Prestwich. CNF Encodings. In *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, pages 75–97. IOS Press, 2009.

[Tamura *et al.*, 2009] Naoyuki Tamura, Tomoya Tanjo, and Mutsunori Banbara. System Description of a SAT-based CSP Solver Sugar. In *Proceedings of the 3rd International CSP Solver Competition*, pages 71–75, 2009.

[Tanjo *et al.*, 2012] Tomoya Tanjo, Naoyuki Tamura, and Mutsunori Banbara. Azucar: A SAT-Based CSP Solver Using Compact Order Encoding — (Tool Presentation). In *The 15th International Conference on Theory and Applications of Satisfiability Testing — SAT*, pages 456–462, 2012.

[Xu *et al.*, 2008] Lin Xu, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. SATzilla: Portfolio-based Algorithm Selection for SAT. *Journal of Artificial Intelligence Research*, pages 565–606, 2008.