Augmenting a CSP Portfolio with SAT Representations and Solvers*

Barry Hurley**, Lars Kotthoff, Yuri Malitsky, and Barry O'Sullivan

Cork Constraint Computation Centre
Department of Computer Science, University College Cork, Ireland
{b.hurley|l.kotthoff|y.malitsky|b.osullivan}@4c.ucc.ie

Abstract. In recent years, portfolio approaches to solving SAT problems and CSPs have become increasingly common. There are also a number of different techniques for converting SAT problems into CSPs. In this paper, we leverage advances in both areas and present a novel hierarchical portfolio-based approach to CSP solving that does not rely purely on CSP solvers, but may convert a problem to SAT choosing a conversion technique and the accommodating SAT solver. Our experimental evaluation relies on competition CSP instances and uses eight CSP solvers, three SAT encodings and eighteen SAT solvers. We demonstrate that significant performance improvements can be obtained by considering alternative view-points of a combinatorial problem.

1 Introduction

The pace of development in both CSP and SAT solver technology has been rapid. Combined with portfolio and algorithm selection technology, impressive performance improvements over systems that have been developed only a few years previously have been demonstrated. Constraint satisfaction problems and satisfiability problems are both NP-complete and, therefore, there exist polynomial-time transformations between them. We can leverage this fact to convert CSPs into SAT problems and solve them using SAT solvers.

In this paper we show that different SAT solvers have different performance on different encodings of the same CSP. In fact, the particular choice of encoding that will give good performance with a particular SAT solver is dependent on the problem instance to be solved. We show that, in addition to using dedicated CSP solvers, to achieve the best performance for solving a CSP, the best course of action might be to translate it to SAT and solve it using a SAT solver. We name our approach Proteus, after the Greek god Proteus, the shape-shifting water deity that can foretell the future.

Our approach offers a novel perspective on using SAT solvers for constraint solving. The idea itself is not new. The solvers `Sugar`, `Azucar` and `CSP2SAT4J` are

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

** Student.

three examples for SAT-based CSP solving. `Sugar` [4] has been very competitive in recent CSP solver competitions. It encodes the CSP to SAT using a specific encoding, known as the order encoding, which will be discussed in more detail later in this paper. `Azucar` [5] is a related SAT-based CSP solver that uses the compact order encoding. However, both `Sugar` and `Azucar` use a single predefined solver to solve the encoded CSP instances. Our work does not assume that conversion to SAT is the best way of solving a problem, but considers multiple candidate encodings and solvers.

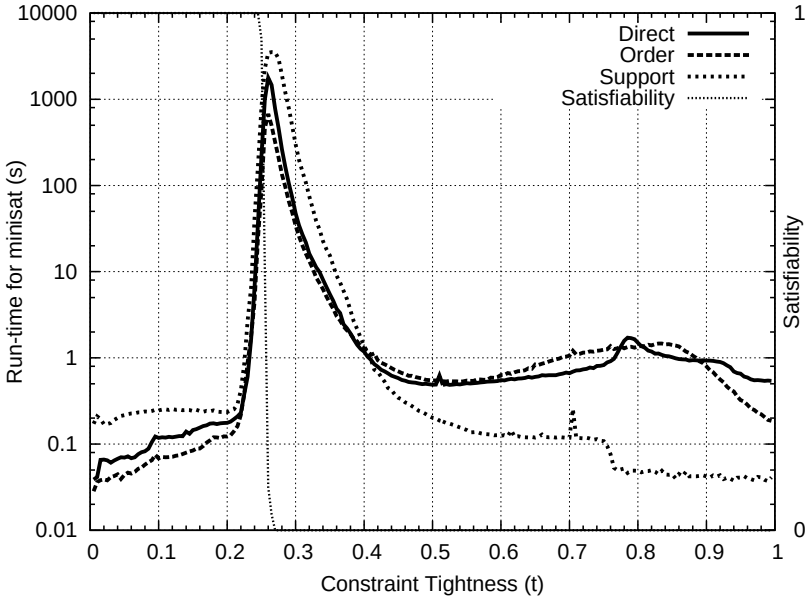
In contrast to our approach, `CSP2SAT4J` [2] uses the `SAT4J` library as its SAT back-end and a set of static rules to choose either the direct or the support encoding for each constraint. Our approach does not have predefined rules but instead chooses the encoding and solver dynamically based on features of the problem to solve.

Our approach employs algorithm selection techniques to dynamically choose whether to translate to SAT, and if so, which SAT encoding and the solver to use. We note three contrasting example approaches to algorithm selection for the constraint satisfaction and satisfiability problems: CPHydra (CSP), SATZilla (SAT), and ISAC (SAT). CPHydra [3] contains an algorithm portfolio of CSP solvers which partitions CPU-Time between components of the portfolio in order to maximize the expected number of solved problem instances within a fixed time limit. SATZilla [6], at its core, uses cost-sensitive decision forests that vote on the SAT solver to use for an instance. In addition to that, it contains a number of practical optimizations, for example running a pre-solver to quickly solve the easy instances. ISAC [1] is a cluster-based approach that groups instances based on their features and then finds the best solver for each cluster. A comparison to these systems is discussed in Section 3.1.

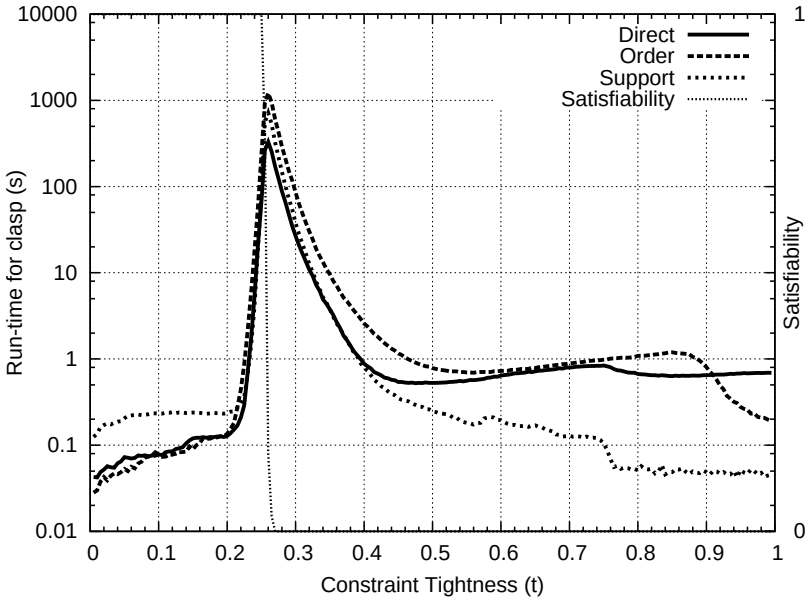
2 Multiple Encodings and Solvers

To motivate our work, we performed a detailed investigation for two solvers to assess the relationship between solver and problem encoding with features of the problem to be solved. For this experiment we considered uniform random binary CSPs with a fixed number of variables, domain size and number of constraints, and varied the constraint tightness. The constraint tightness t is a measure of the proportion of forbidden to allowed possible assignments to the variables in the scope of the constraint. We vary it from 0 to 1, where 0 means that all assignments are allowed and 1 that no assignments are part of a solution, in increments of 0.005. At each tightness the mean run-time of the solver on 100 random CSP instances is reported. Each instance contains 30 variables with domain size 20 and 300 constraints. This allowed us to study the performance of SAT encodings and solvers across the phase transition.

Figure 1 plots the run-time for `Minisat` and `Clasp` on uniformly random binary CSPs that have been translated to SAT using three different encodings. Observe that in Figure 1(a) there is a distinct difference in the performance of `Minisat` on each of the encodings, sometimes an order of magnitude difference.



(a) Performance of `Minisat` on SAT-encoded URB CSP.



(b) Performance of `Clasp` on SAT-encoded URB CSP.

Fig. 1. `Minisat` and `Clasp` on random binary CSPs.

Before the phase transition we see that the order encoding achieves the best performance on these instances and maintains this even at the phase transition. Beginning at constraint tightness 0.41, the order encoding gradually starts achieving poorer performance and the support encoding now achieves the best performance. Notably, if we rank the encodings based on their performance, the ranking changes after the phase transition. This illustrates that there is not just a single encoding that will perform best overall and that the choice of encoding matters, but also that this choice is dependent on problem characteristics such as constraint tightness.

Around the phase transition, we observe contrasting performance for `Clasp`, as illustrated in Figure 1(b). Using `Clasp`, the ranking of encodings around the phase transition is direct $\succ$ support $\succ$ order; whereas for `Minisat` the ranking is order $\succ$ direct $\succ$ support. Note also that the peaks at the phase transition differ in magnitude between the two solvers. These differences underline the importance of the choice of solver, in particular in conjunction with the choice of encoding – making the two choices in isolation does not consider the interdependencies that affect performance in practice.

3 Experimental Evaluation

In addition to the random CSP instances, our analysis also comprises benchmarks from the CSP solver competitions.[1] Of these, we consider the instances that contain either inequality or binary extensional constraints that our tool can translate to SAT. Altogether, we use 2207 instances from the Graph Colouring, Random, Quasi-random, Black Hole, Quasi-group Completion, Quasi-group with Holes, Langford, Towers of Hanoi and Pigeon Hole problem classes. While these benchmarks do not exhibit the full richness of what CP can offer, our results demonstrate significant gains and we are confident that these would also be achievable for a larger benchmark set. An extension of this work to global constraints and a wider set of benchmarks is currently underway.

Figure 2 illustrates the respective performance of CSP-based and SAT-based methods on these instances. Unsurprisingly the dedicated CSP methods often achieve the best performance. There are, however, cases where considering SAT-based methods has the potential to yield significant performance improvements. In particular, there are a number of instances that are unsolved by any CSP solver but can be solved using SAT-based methods. The Proteus approach aims to unify the best of both worlds and take advantage of the potential performance gains.

3.1 Greater than the Sum of its Parts

Given the possibilities of Proteus, to consider multiple representations and solvers, the question remains as to whether a different portfolio approach that considers

[1] CSP solver competition instances
`http://www.cril.univ-artois.fr/~lecoutre/benchmarks.html`

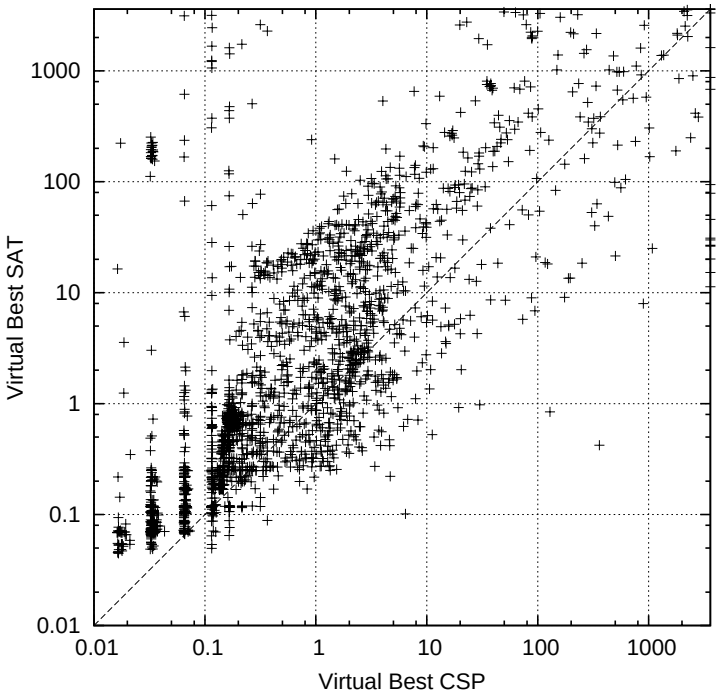


Fig. 2. Performance of virtual best CSP solver portfolio against the virtual best SAT-based portfolio that selects the best encoding/solver combination.

just CSP or just SAT solvers could do better. Table 1 summarizes the virtual best performance that such portfolios could achieve. Our CSP and SAT models are able to choose from 8 and 18 complete solvers respectively. The VB CSP is the approach that always chooses the best CSP solver for the current instance, while the VB SAT chooses the best SAT encoding/solver combination. VB Proteus is the portfolio that chooses the best overall approach/encoding.

It is interesting to note that Proteus has the potential to outperform the other approaches we present in this table. Specifically, the VB CPHYDRA, which considers three of the CSP solvers, is the best possible performance that could be obtained from that portfolio if a perfect choice of solver was made. Neither SATZILLA nor ISAC consider different SAT encodings. Therefore, the best possible performance either of them could achieve for a specific encoding is represented in the last three lines of Table 1.

These results do not only demonstrate the benefit of considering the different ways of solving CSPs, but also eliminate the need to compare with existing portfolio systems since we are computing the best possible performance that any of those systems could theoretically achieve. Therefore, the strength of the Proteus approach is very convincing.

Table 1. Virtual best performances ranked by PAR10 score.

Method	Mean PAR10	Number Solved
VB Proteus	54	2207
VB SAT	111	2207
VB CSP	224	2197
VB CPHydra	326	2191
VB Order Encoding	969	2156
VB Direct Encoding	1450	2125
VB Support Encoding	2333	2070

4 Conclusions and Future Work

In this paper we have presented 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 solving approaches. To the best of our knowledge, this is the first time a portfolio approach like this has been proposed. We have 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. The overall performance can be improved significantly compared to restricting the portfolio to a single problem representation.

References

1. Kadioglu, S., Malitsky, Y., Sellmann, M., Tierney, K.: ISAC - Instance-Specific Algorithm Configuration. In: Coelho, H., Studer, R., Wooldridge, M. (eds.) ECAI. Frontiers in Artificial Intelligence and Applications, vol. 215, pp. 751–756. IOS Press (2010)
2. Le Berre, D., Lynce, I.: CSP2SAT4J: A Simple CSP to SAT Translator. In: Proceedings of the 2nd International CSP Solver Competition (2008)
3. O'Mahony, E., Hebrard, E., Holland, A., Nugent, C., O'Sullivan, B.: Using Case-based Reasoning in an Algorithm Portfolio for Constraint Solving. Proceeding of the 19th Irish Conference on Artificial Intelligence and Cognitive Science (2008)
4. Tamura, N., Tanjo, T., Banbara, M.: System Description of a SAT-based CSP Solver Sugar. In: Proceedings of the 3rd International CSP Solver Competition. pp. 71–75 (2009)
5. Tanjo, T., Tamura, N., Banbara, M.: Azucar: A SAT-Based CSP Solver Using Compact Order Encoding — (Tool Presentation). In: SAT 2012. pp. 456–462. Springer (2012)
6. Xu, L., Hutter, F., Hoos, H.H., Leyton-Brown, K.: SATzilla: Portfolio-based Algorithm Selection for SAT. Journal of Artificial Intelligence Research pp. 565–606 (2008)