Constraint Satisfaction Techniques for Combinatorial Problems

David E. Narváez
Golisano College of Computing and Information Sciences
Rochester Institute of Technology
Rochester, NY, USA 14623
den9562@rit.edu

Abstract

The last two decades have seen extraordinary advances in industrial applications of constraint satisfaction techniques, while combinatorial problems have been pushed to the sidelines. We propose a comprehensive analysis of the state of the art in constraint satisfaction problems when applied to combinatorial problems in areas such as graph theory, set theory, algebra, among others. We believe such a study will provide us with a deeper understanding about the limitations we still face in constraint satisfaction problems.

Introduction

In recent years, constraint satisfaction problems (CSPs) have drawn much attention due to their applications to several areas of industrial research. This research focus has brought a torrent of positive results in areas like SAT solvers, satisfiability modulo theories, answer set programming, etc. These results often rely on the fact that even though determining the satisfiability of a constraint program is NP-hard, many industrial applications exhibit constraints that computers are able to deal with easily. Benchmarks stemming from these applications often showcase the advantages of the different techniques presented, and seldom are there references to where these techniques perform poorly. Often times their worst-case scenarios can be found in combinatorial problems which are more structured, have more symmetries and encode deeper questions in discrete mathematics. These problems fall conveniently outside the common definition of what an industrial application is, and thus new techniques are seldom analyzed for these problems. In this sense, the fact that we have come a long way from determining the satisfiability problem is NP-complete to solving answer set programs with thousands of variables can be reinterpreted as a change of the yardstick used to measure advances.

Neglecting these combinatorial problems has a negative effect not only in the study of discrete mathematics but in our true understanding of constraint satisfaction. The lack of a formal definition of what an industrial application of CSP is makes it hard to understand exactly what distribution of CSPs arises from these. In the case of combinatorial problems, their definition and structure is much better understood, giving us a better benchmark to understand the behavior of algorithms on a class of problems. This is supported by a number of results that have used combinatorial problems to exemplify the limitations of well-known techniques in constraint satisfaction and theorem proving in general: Tseitin's expander graphs (Urquhart 1987), Ramsey formulas (Krishnamurthy and Moll 1981), and pigeon hole formulas (Krajíček 2001) were used to provide hard cases for resolution; while pebble games are used to study proof complexity (Järvisalo et al. 2012) given the well-understood structure of pebbling formulas.

Background and Definitions

In its more general form, constraint satisfaction problems (CSPs) consist of a set of variables $\mathbf{X}$ each taking values in a domain D and a set of constraints C involving variables in $\mathbf{X}$ and operations over these variables. For instance, in Boolean satisfiability problems the domain D takes the form of $\{\bot, \top\}$ and the constraints are expressed over the operations $\wedge$, $\vee$, $\neg$. In the case of integer linear programs (ILP), the domain of the variables is the set of integers, and the constraints are inequalities over the operations of addition and multiplication.

Many problems can be modeled as CSPs, where different encodings offer different advantages and nuisances. The types of problems for which CSPs have been applicable can be roughly divided into two categories: industrial applications and abstract applications. While no formal definition of these has been given in the literature, one can informally define industrial applications as those arising from practical applications like circuit design and hardware checking, whereas abstract applications correspond to CSPs arising from problems in discrete mathematics. The work by Ansótegui et. al (Ansótegui, Bonet, and Levy 2009) shows a structural characterization of industrial SAT instances by looking at the bi-partite graph representing the relationships between clauses and variables.

Preliminary Result on Boolean Formulas

Since my goal is to study the applications of state-of-the-art techniques in CSPs to combinatorial problems, it is natural to start by looking at the SAT problem and its variations. Within this domain, my strategy is to understand what are the key concepts and techniques and how they can be applied to combinatorial problems. The next two subsections summarize preliminary work that follows these two lines of research.

Backbones and Backdoors in Satisfiability

Backbones and backdoors have been identified as hidden structures that are related to the hardness of SAT instances (Williams, Gomes, and Selman 2003). Given that our hypothesis is that combinatorial problems provide hard instances of SAT problems, we wish to study these structures closely and determine the role they play in the instances we are interested in. In (Hemaspaandra and Narváez 2017a) we looked at a separation in the complexity of finding a backbone versus that of finding the value to which that backbone must be set in order to find a satisfying assignment. This separation is true under the widely-believed assumption that $\mathrm{P} \neq \mathrm{NP} \cap \mathrm{coNP}$. As a continuation of this work (Hemaspaandra and Narváez 2017b), we prove, under the assumption that $\mathrm{P} \neq \mathrm{NP}$, the existence of easily-recognizable sets of Boolean formulas for which it is hard to tell if they have a backbone, and easily-recognizable families of formulas for which it is easy to find a backdoor but the problem of determining the satisfiability of a formula in these sets is still NP-complete.

The ALLSAT Problem for Ramsey Colorings

We say that a graph F arrows the pair of graphs (G, H) (written $F \rightarrow (G, H)$) when every 2-coloring of the edges of F contains a monochromatic G in the first color or a monochromatic H in the second color. This can be modeled as a Boolean formula where the variables correspond to the edges of F and the clauses are of the form $e_1 \vee e_2 \vee \ldots e_k$ for every subgraph isomorphism from G to F and $\overline{e_1} \vee \overline{e_2} \vee \ldots \overline{e_k}$ for every subgraph isomorphism from H to F. If the resulting formula is not satisfiable, then $F \rightarrow (G, H)$. If F does not arrow the pair (G, H), then every model of the formula represents a 2-coloring of the edges of F witnessing the fact that $F \not\rightarrow (G, H)$. Generating complete sets of witnesses for the arrowing property is an important building block in computational Ramsey theory.

In (Narváez 2018) I study the effect of symmetry breaking techniques in ALLSAT problems encoding the negation of the arrowing property. Specifically, I look at the use of the popular `Shatter` (Aloul, Sakallah, and Markov 2006) tool for symmetry breaking for formulas generated from this encoding. Through this study, I was able to pinpoint two issues of using `Shatter` for ALLSAT. The first one is a blow-up in the number of models of the resulting formula after symmetry breaking. The second one is related to properties that exist in the combinatorial domain that are not carried over by the encoding and are thus not accessible to `Shatter`, resulting in incomplete sets of colorings generated from the formula with symmetry breaking.

Research Plan

In the immediate future, I expect to shift my attention to using QBF solvers to model the problem of equivalence of nondeterministic finite automata (NFAs), for educational purposes. While it is known that it is possible to encode NFA equivalence as a QBF problem, we are not aware of any implementation that links this to teaching automata theory. *Before AAAI-18* I expect to fill this gap by prototyping an integration between QBF solvers and some of the most common tools used to teach computing theory.

Moving forward, I expect to connect the theoretical work made on backbones and backdoors with specific combinatorial problems, e.g., by showing that, under that measure of hardness, Boolean formulas arising from encodings of certain combinatorial problems are provably hard.

I am interested in exploring other paradigms of constraint satisfaction, namely Answer Set Programming and Satisfiability Modulo Theories, which are two paradigms that have shown great progress in later years. Nevertheless, the wide array of constraint satisfaction paradigms and the inclusions between them (for example, Boolean formulas can be considered a special case of ASP which in turn is a special case of SMT) pose an interesting question for combinatorial problems: given a combinatorial problem, at which point does the power of a generalization of the current paradigm provides no advantage to solving this problem? This is an interesting question to address since it will shed light on the research done on these combinatorial problems as well as help understand the powers and limitations of the different paradigms in constraint satisfaction.

Acknowledgments

The author would like to thank the anonymous reviewers of AAAI-18 for their valuable comments.

References

Aloul, F. A.; Sakallah, K. A.; and Markov, I. L. 2006. Efficient symmetry breaking for boolean satisfiability. *IEEE Transactions on Computers* 55(5):549–558.

Ansótegui, C.; Bonet, M. L.; and Levy, J. 2009. On the structure of industrial SAT instances. In *Principles and Practice of Constraint Programming*, 127–141.

Hemaspaandra, L., and Narváez, D. 2017a. The opacity of backbones. In *AAAI*, 3900–3906. AAAI Press.

Hemaspaandra, L., and Narváez, D. 2017b. The opacity of backbones and backdoors under a weak assumption. Technical Report arXiv:1706.04582 [cs.AI] Revised, October 2017.

Järvisalo, M.; Matsliah, A.; Nordström, J.; and Živný, S. 2012. Relating proof complexity measures and practical hardness of SAT. In *Principles and Practice of Constraint Programming*, 316–331.

Krajíček, J. 2001. On the weak pigeonhole principle. *Fundamenta Mathematicae* 170:123–140.

Krishnamurthy, B., and Moll, R. N. 1981. Examples of hard tautologies in the propositional calculus. In *STOC*, 28–37. ACM.

Narváez, D. 2018. Exploring the use of Shatter for ALLSAT through Ramsey-type problems. In *AAAI (Student Abstract)*. To appear.

Urquhart, A. 1987. Hard examples for resolution. *Journal of the ACM* 34(1):209–219.

Williams, R.; Gomes, C. P.; and Selman, B. 2003. Backdoors to typical case complexity. In *IJCAI*, 1173–1178.