Classifying Constraint Satisfaction Problems

A Proposal for a Bachelor's or a Master's Thesis

Victor Lagerkvist
victor.lagerkvist@liu.se

George Osipov
george.osipov@liu.se

TCSLAB, IDA, Linköping University

Background

The question of P vs NP is one of the most well-studied problems in theoretical computer science and mathematics. First posed by Kurt Gödel in 1956 in personal correspondence with John von Neumann [4], this problem has seen very little progress since then. Not only is this question notoriously difficult, but it is also one of the remaining Millennium problems, and has a monetary reward of one million dollars [2]. Thus, solving P vs NP not only implies infinite fame and glory, but allows one to live a luxurious life while tackling the five remaining Millennium problems.

In this thesis proposal we do not aim to solve P vs NP (although this would be most welcome), but to explore some related questions in the computational complexity of the constraint satisfaction problem (CSP). One example of such problem is the Sudoku puzzle. You are given a 9×9 board with some places containing numbers from 1 to 9. The goal is to fill out the remaining squares with the numbers of 1 to 9 so that the following constraints are satisfied: the numbers in each row, each column, and each 3×3 box with bold edges should be different.

5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

More generally, one defines a constraint satisfaction problem by a set of *variables* (empty squares in our example), the *domain*, i.e. the set of values that the variables can take (integers from 1 to 9 for Sudoku), and the *constraint language*, which specifies the kinds of constraints one can express. The domain and the constraint language determine computational complexity of the CSP.

In the past 15-20 years, the *algebraic approach* has proven to be very successful in drawing the line between the easy (in P) and the hard (NP-complete) CSPs [3]. Informally, this approach makes it possible to relate particular algebraic properties of a constraint language to its computational complexity. One remarkable success of the algebraic approach is the dichotomy theorem proved independently in 2017 by Bulatov [1] and Zhuk [5]. In general, checking whether a certain computational problem is in P or NP-complete requires either designing an efficient algorithm or providing a hardness reduction from another NP-complete problem. Both these tasks require experience, patience and care, and quite often creativity. The dichotomy theorem of Bulatov and Zhuk allows to automate this process for all finite-domain CSPs: if the constraint language of interest has a particular (computable) algebraic property, then the corresponding CSP is in P, otherwise it is NP-complete.

Goals

We propose to take a closer look at the criterion for deciding the computational complexity of finite-domain CSPs. The criterion can be thought of as an algorithm that, given the description of the constraint language, outputs whether the CSP defined by this language is in P or NP-complete. For a *bachelor's thesis*, we ask, how can a tool be implemented to decide whether a finite-domain CSP is in P or NP-complete? A desirable outcome would be a reasonably efficient implementation that can be used to generate a database of easy and hard CSPs (for smaller domains). For a *master's thesis*, we further ask about the complexity of this criterion. There are several algebraic properties that can be used for the task. Which ones yield the best performance in practice? Are there better ways (both theoretically and practically) to determine the complexity of finite-domain CSPs in some special cases?

Contact

Sounds interesting? Do not hesitate to contact us!

References

[1] Andrei A Bulatov. A dichotomy theorem for nonuniform csps. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 319–330. IEEE, 2017.

[2] Clay Math Insitite. Millenium problems. `https://www.claymath.org/millennium-problems`.

[3] Peter Jeavons. On the algebraic structure of combinatorial problems. *Theoretical Computer Science*, 200(1-2):185–204, 1998.

[4] Richard J. Lipton. Gödel's lost letter. `http://rjlipton.wordpress.com/the-gdel-letter/`.

[5] Dmitriy Zhuk. A proof of the csp dichotomy conjecture. *Journal of the ACM (JACM)*, 67(5):1–78, 2020.