Master internship proposal:
Space and time efficient algorithms for Satisfaction problems

Advisor: Benjamin Bergougnoux

COALA, LIS, Aix Marseille Université, CNRS
Contact: *benjamin.bergougnoux@lis-lab.fr*
Website: https://benjaminbergougnoux.github.io/

1 Context

The computational problems SAT (the satisfiability problem in propositional logic) and CSP (the Constraint Satisfaction Problem) are at the heart of many domains in computer science including symbolic artificial intelligence and complexity theory. These two generic problems and their optimization and counting variants are used to model and solve a wide variety of academic and industrial problems. SAT and CSP are notoriously famous for their theoretical hardness: not only they are NP-complete, but no algorithm is known to have a better worst-case time complexity than the brute force algorithm who tries every possible assignment of values for the variables [4]. Nonetheless, significant results have been obtained to solve these problems both in theory and in practice.

In theory, the efforts on solving exactly NP-hard problems focus on identifying classes of instances that are tractable, e.g., that can be solved in polynomial time. A successful approach in this line of research is the theory of parameterized complexity. It uses a parameter to measure the amount of structure present in data—generally, the lower the parameter, the more structured it is. The main goal is to design algorithms whose runtime have a low dependency on the input size and to confine the combinatorial explosion to the parameter as much as possible. This makes this framework particularly well-suited to deal with various problems on big data sets. Typically, one aim to find algorithms—called FPT—running in time $f(k)n^c$ where k is the parameter, c a constant, f a computable function and n the input size. The goal being to find such algorithms where the $f(k)$ factor and constant c are as small as possible. When FPT is not possible, one can look for algorithms—called XP—that run in time $f(k)n^{O(g(k))}$ where g is computable function. However, FPT algorithms are significantly more efficient than XP algorithms, and they prove that the parameter captures entirely the hardness of the problem.

The most general tractable classes of NP-hard problems are generally obtained via a family of parameters called *width parameters*. These parameters measure the amount of structure on combinatorial objects (e.g., graph, hypergraph) via some notion of recursive decompositions. The archetypal, and possibly most celebrated width parameters is treewidth. Roughly, a (hyper)graph has treewidth at most k if it can be represented by a structural decomposition—called *tree decomposition*—into vertex sets of size $k+1$ (called *bags*) that are connected in a tree-like fashion. A huge variety of problems admit efficient parameterized algorithms with these width parameters (as illustrated by Courcelle's theorem). This is the case for SAT and CSP via the width parameters

of the graphs and hypergraphs associated with the instances of these problems. For example, the following is a well-known FPT result on CSP.

Theorem 1.1. *[2] CSP can be solved in time* $\mathsf{dom}^{O(\mathsf{tw})}|\mathbf{I}|^{O(1)}$ *where* dom *is the maximum domain size,* tw *the treewidth of the primal graph and* $|\mathbf{I}|$ *the input size (in particular, CSP is FPT parameterized by* dom *and* tw*).*

Many FPT and XP results have been proved with generalization of treewidth such as fractional hypertree width. Moreover, several of these results are probably optimal, in the sense that obtaining a faster algorithm, or proving the same result with a more general parameter is not possible unless some reasonable complexity conjecture fails [5]. However, the motivations of these works were often purely theoretical and far from practical considerations.

In practice, the efficiency of software—called "solvers"—to solve SAT and CSP have drastically improved in the last decades. These solvers are able to solve very large real-world instances (with millions of variables) within reasonable time. Modern solvers are generally based on algorithmic techniques such as constraint propagation and constraint learning, enhanced by the use of extremely effective heuristics. Their good performances on real world instances imply that these instances have a lot of hidden structures that boost the effectiveness of these techniques and heuristics. However, the current state of knowledge does not provide any theoretical reasons behind these good performances. The only understanding we have—such as the presence of *communities* in SAT instances [6]—are based on experimental observations. Moreover, we appear to have reached a level at which significant improvement would surely require fundamentally different techniques. Indeed, while solvers perform well on some big instances, there are some on which they really struggle. Solving such challenging instances might require strong theoretical analysis of their structures and new solving paradigms.

2 Goal of the internship

Most algorithms on SAT and CSP based on width parameters are pure dynamic programming algorithms. However, the space usage of these algorithms is exponential in the decomposition's width. In practical applications, this is often a prohibitive factor, because the machine is likely to simply run out of space much before the elapsed time exceeds tolerable limits. Therefore, a natural question arises: Can we reduce the space complexity of these algorithms? Unfortunately, some complexity-theoretical evidences suggest that this reduction of space complexity is not possible for algorithms based on treewidth without worsening the time complexity, e.g., 3-Coloring is probably not solvable by a $2^{O(\mathsf{tw})}n^{O(1)}$ time algorithm using $2^{o(\mathsf{tw})}n^{O(1)}$ space [7].

However, this reduction of space complexity is possible when we consider *treedepth*: a shallow restriction of treewidth. Several NP-hard problems such as Independent Set or Hamiltonian Cycle can be solved in time $2^{O(\mathsf{td})}n$ and polynomial space (i.e., $n^{O(1)}$) where td is the treedepth of a given a decomposition. For some problems, the space complexity can even be reduced to $O(\mathsf{td}\log n)$. Typically, the main idea is to reformulate the classic bottom-up dynamic programming approach so that it can be replaced by a simple top-down recursion. This reformulation is by no means easy—it often involves a highly non-trivial use of algebraic transforms or other tools of algebraic flavor.

All these space efficient algorithms based on treedepth have been unified and generalized via a recent algorithmic meta theorem based on some logic called $\mathsf{NEO}_2[\mathsf{FRec}]{+}\mathsf{ACK}$ [1]. In particular, this logic captures SAT and Max-SAT via the incidence graphs associated to CNF formulas. The model checking algorithm provided with this logic solves SAT, Max-SAT and also #SAT in time $2^{O(\mathsf{td})}n$ and space $O(\mathsf{td}\log n)$. However, $\mathsf{NEO}_2[\mathsf{FRec}]{+}\mathsf{ACK}$ is not able to capture CSP via its

primal graph or incidence graph. It is known that CSP can be solved in time $O(\mathsf{dom}^{\mathsf{td}} n^{O(1)})$ and polynomial space where dom is the maximum domain size, td the treedepth of the primal graph, and n the input size [3]. Since, the treedepth of the primal graph is a quite restrictive parameter, it is natural to wonder whether we can generalize this result.

The goal of this internship is to find more general parameters allowing efficient parameterized algorithms for SAT and CSP using polynomial space.

In particular, we could try to the best algorithm in polynomial space for CSP parameterized by the treedepth of its incidence graph. Moreover, it is also possible to look for negative results such as the above-mentioned one for 3-Coloring [7]. Such negative results would be highly valuable to not get stuck looking for inexistent positive results.

In the classical formalization of CSP, the constraints are given as a list of allowed combinations of values. Hence, unlike SAT, the constraints of CSP can be highly asymmetrical w.r.t. the variables in their scopes. This is why $\mathsf{NEO}_2[\mathsf{FRec}]{+}\mathsf{ACK}$ can express SAT but not CSP on their incidence graphs. Hence, it makes sense to look at restrictions/variant of the classical formalization with limitations on the set of constraints. Forcing symmetries on the constraints of a CSP instance could allow us to use the known algebraic tools for obtaining polynomial space algorithm with treedepth.

In fact, most theoretical results on CSP completely ignore global constraints, i.e., constraints of arbitrary arity where the set of allowed tuple is implicit, for example: $\texttt{alldifferent}(x_1, \ldots, x_n)$ specify that the variables $x_1, \ldots, x_n$ must take pairwise different values. In practice, global constraints are essential to improve the modeling power of CSP and the performances of solvers with dedicated subroutines to propagate each kind of constraint. It would be interesting to study the time and space complexity of solving CSP instances with only some kind of global constraints (e.g., only `alldifferent` constraints).

3 Possibility of extension into a Ph.D.

If the internship is successful, the advisor is available to supervise a subsequent Ph.D. with a senior member of its team but there is no secure funding for it at the moment.

4 Prerequisites

No prerequisites other than an interest for graphs, complexity and algorithms.

References

[1] B. Bergougnoux, V. Chekan, and G. Stamoulis. A logic-based algorithmic meta-theorem for treedepth: Single exponential fpt time and polynomial space, 2025.

[2] R. Dechter and J. Pearl. Tree clustering for constraint networks. *Artif. Intell.*, 38(3):353–366, 1989.

[3] E. C. Freuder and M. J. Quinn. Taking advantage of stable sets of variables in constraint satisfaction problems. In A. K. Joshi, editor, *Proceedings of the 9th International Joint Conference on Artificial Intelligence. Los Angeles, CA, USA, August 1985*, pages 1076–1078. Morgan Kaufmann, 1985.

[4] R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? *J. Comput. Syst. Sci.*, 63(4):512–530, 2001.

[5] D. Marx. Tractable hypergraph properties for constraint satisfaction and conjunctive queries. *J. ACM*, 60(6):42:1–42:51, 2013.

[6] Z. Newsham, V. Ganesh, S. Fischmeister, G. Audemard, and L. Simon. Impact of community structure on SAT solver performance. In C. Sinz and U. Egly, editors, *Theory and Applications of Satisfiability Testing - SAT 2014 - 17th International Conference, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 14-17, 2014. Proceedings*, volume 8561 of *Lecture Notes in Computer Science*, pages 252–268. Springer, 2014.

[7] M. Pilipczuk and M. Wrochna. On Space Efficiency of Algorithms Working on Structural Decompositions of Graphs. *ACM Trans. Comput. Theory*, 9(4):18:1–18:36, 2018.