PhD thesis proposal : New width parameters to explain and improve modern solvers performances on satisfaction problems

— **Supervisors :** Benjamin Bergougnoux and Cyril Terrioux, Aix Marseille Université, LIS.
— **Contact :** benjamin.bergougnoux@lis-lab.fr.
— **Keywords :** Parameterized complexity, width parameters, SAT, CSP, solvers.

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. These problems are notoriously famous for their theoretical hardness. Nonetheless, significant results have been obtained to solve these problems both in theory and in practice.

In practice, the performances of software—called "solvers"—to solve SAT and CSP have drastically improved in the last decades. These solvers are able to solve 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. These performances do not contradict the theoretical hardness of SAT and CSP, as P$\neq$NP does not mean that all the instances of these problems are hard to solve. It is believed that the instances on which modern solvers excel must have hidden structures that boost the effectiveness of the algorithmic techniques and heuristics used by the solvers. However, the current state of knowledge does not provide any theoretical reasons behind these good performances.

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 to define more general tractable classes is the parameterized complexity theory [3]. This framework 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. 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 most general tractable classes we know for 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. 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 ([4])**.** *CSP can be solved in time* $O(\mathsf{dom}^{k+1} \cdot n)$ *where* dom *is the maximum domain size,* k *the treewidth of a given tree decomposition of the primal graph and* n *the number of variables (in particular, CSP is FPT parameterized by* dom *and* k*).*

Many similar results have been proved with generalization of treewidth such as fractional hypertree width [6]. 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.

While results like Theorem 1 are theoretically impressive for their level of generality, the motivations of these works are often purely theoretical and far from practical considerations. In fact, none of the known tractable classes for satisfaction problems are used in practice and none of them are able to explain the impressive performances of modern solvers.

The goal of this thesis is to provide theoretical explanations on the impressive performances of modern solvers for satisfaction problems. For doing so, we are going to look for tractable classes interesting in practice by considering aspects that are most of the time overlooked in theory but essential in practice such as space complexity, global constraints and the impact of heuristics.

One objective of this thesis is to re-explore the parameterized complexity of satisfaction problems by taking into consideration the space complexity. Most algorithms for satisfaction problems based on width parameters are pure dynamic programming algorithms with a space usage 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. Hence, it is quite natural to look for efficient FPT algorithms that use polynomial space. This is possible when we consider *treedepth* (a shallow restriction of treewidth). For example, SAT, Max-SAT and #SAT can be solved in time $2^{O(\mathsf{td})}n^{\bar{c}O(1)}$ and space $O(\mathsf{td} \cdot \log n)$ with td the treedepth of a given decomposition [1]. In this thesis, we will look for more general parameters allowing such time and space efficient algorithms for these problems.

Another objective of this thesis is to find width parameters for satisfaction problems that are meaningful in practice. For instance, there exists efficient solver based on tree decompositions such as the ones based on the *backtracking on tree decomposition (BTD)* algorithm introduced by Jégou and Terrioux [8]. While BTD clearly highlights the practical interest of tree-decompositions for solving CSP, it is not really the case for treewidth. In practice, the performance of BTD are significantly better when we consider tree decompositions where each bag induced a connected graph [7, 9] even if we end up with much bigger bags. Hence, we should try to find a parameter different from treewidth that allows to measure more precisely the performances of BTD.

A huge limitation of almost all studied width parameters for CSP is that they are defined on the structures induced by the scopes of the constraints : the *syntax* of the constraints. As such, they are completely oblivious to the structures induced by the *semantics* of the constraints, i.e., their list of allowed combinations of values. This is a huge limitation in practice, since the performances of solvers may differ on instances with the same constraint hypergraph. While there exists some parameters—called *hybrid*—that takes into account the syntax and the semantics of constraints [5], they are far from practical considerations. One way of proceeding would be to consider parameter that measure the "width" of tree decompositions with respect to the semantics of the constraints.

In order to be interesting in practice, our results on CSP should consider global constraints. Almost all known tractable classes for CSP rely on an extensive representation of constraints in the form a list of allowed tuples. In practice, the huge modeling power of CSP and the performances of its solvers are tied to global constraints that captures a relation between a non-fixed number of variables. The archetypal example of global constraint is the $\mathsf{all\text{-}different}(x_1, \ldots, x_n)$ constraint requiring that $x_1, \ldots, x_n$ take pairwise distinct values [10]. Few results are known on the complexity of CSP with global constraints (e.g. [2]). Considering the importance of global constraints in practice and the state of the art in theory, it is quite natural to study the questions raised in this thesis through the lens of global constraints.

Bibliography

[1] B. Bergougnoux, V. Chekan, and G. Stamoulis. A logic-based algorithmic meta-theorem for treedepth : Single exponential FPT time and polynomial space. *Proceedings of the 2026 Annual ACM-SIAM Symposium on Discrete Algorithms, SODA*, 2026.

[2] D. A. Cohen and P. G. Jeavons. The power of propagation : when GAC is enough. *Constraints An Int. J.*, 2017.

[3] M. Cygan, F. V. Fomin, L. Kowalik, D. Lokshtanov, D. Marx, M. Pilipczuk, M. Pilipczuk, and S. Saurabh. *Parameterized Algorithms*. Springer, 2015.

[4] R. Dechter and J. Pearl. Tree clustering for constraint networks. *Artif. Intell.*, 1989.

[5] R. Ganian, S. Ordyniak, and S. Szeider. A join-based hybrid parameter for constraint satisfaction. *Principles and Practice of Constraint Programming - 25th International Conference, CP*, 2019.

[6] M. Grohe and D. Marx. Constraint solving via fractional edge covers. *ACM Trans. Algorithms*, 2014.

[7] P. Jégou, H. Kanso, and C. Terrioux. On the relevance of optimal tree decompositions for constraint networks. *IEEE 30th International Conference on Tools with Artificial Intelligence, ICTAI*, 2018.

[8] P. Jégou and C. Terrioux. Hybrid backtracking bounded by tree-decomposition of constraint networks. *Artif. Intell.*, 146(1) :43–75, 2003.

[9] P. Jégou and C. Terrioux. Combining restarts, nogoods and bag-connected decompositions for solving csps. *Constraints An Int. J.*, 2017.

[10] J. Régin. A filtering algorithm for constraints of difference in csps. *Proceedings of the 12th National Conference on Artificial Intelligence*, 1994.