Harnessing Tractability in Constraint Satisfaction Problems
(Extended abstract)

Clément Carbonnel

1 Introduction

The Constraint Satisfaction Problem (CSP) is a fundamental problem in computer science that takes as input a finite set of variables, a finite set of values and a set of constraints, which are arbitrary relations imposed on certain subsets of variables. The goal is to find an assignment of values to variables that satisfies every constraint.

The generality of CSP and its connections to logic, universal algebra and graph theory have drawn considerable attention from the complexity theory community. In particular, a rich theory has been developed to study how restricting the catalog of available relations (the “constraint language”) affects the complexity of CSP. The ultimate goal is to prove the Feder-Vardi conjecture, which states that such restrictions are always either in P or NP-complete for finite constraint languages. This intense research has produced a series of increasingly sophisticated polynomial-time algorithms that solve CSPs over certain constraint languages. Unfortunately, this wealth of results on tractable subproblems of CSP has seen little to no practical use so far due to the numerous obstacles that must be overcome first.

First, many of these tractable classes are defined by elusive algebraic properties and more often than not the complexity of testing for them is unknown - in some cases, even decidability is an issue. Second, even if we knew how to test efficiently membership in those classes, real-world CSP instances are unlikely to exhibit readily the desired properties so mechanisms must be designed in order to make the framework more adaptable. Third, some of those polynomial-time algorithms were not designed with performance in mind and have critical flaws, such as an exponential dependency on the domain size. Fourth and last, the definition of CSP used in most theoretical works does not leave room for one of the most widely used feature of constraint solvers, global constraints.

This dissertation presents significant progress on these issues by devising novel polynomial-time membership tests for important tractable classes of CSP, studying in a systematic fashion the parameterized complexity of extending known tractable classes through the framework of strong backdoors, and showing how the notion of kernelization can be adapted to make efficient use of (near-)tractability within NP-hard global constraints.

2 Contributions

2.1 The complexity landscape of strong backdoor detection

We first address adaptability. Real-world CSP instances may not often belong to a well-studied tractable classe of CSP, but some instances may be “nearly tractable” in an algorithmically
exploitable sense. This notion is commonly captured by strong backdoors, which are subsets of variables whose complete instantiation is guaranteed to leave residual instances that belong to a fixed “target” tractable class. If a strong backdoor $B_T$ to a nice tractable class $T$ is known, this information is easy to use within a general-purpose CSP solver. For instance, during search the solver can switch to a polynomial-time algorithm as soon as every variable in $B$ has been assigned. The central issue, however, is to actually compute $B_T$. The main question we ask in this chapter is the following: for which choices of target tractable class $T$ is a minimum-size backdoor $B_T$ efficiently computable? We study this question for language-based classes $T$, which are predominant in the CSP literature but for which it remained largely unanswered.

First, we prove that for any “reasonable” choice of $T$ this problem is NP-hard. However, in practice we are not interested in large backdoors; an algorithm that scales exponentially in the backdoor size only would be acceptable. We can stop it if it runs for too long, as it implies that no small backdoor exists. We therefore refine the analysis using parameterized complexity, a framework specialized in proving or disproving the existence of such algorithms.

We show that when the parameter is the backdoor size alone, the problem of computing $B_T$ for any “reasonable” class $T$ is $W[1]$-hard, and hence unlikely to be FPT. If we increase slightly the parameter by including the maximum constraint arity as well, we obtain a much more interesting picture. In particular, we show that under reasonable assumptions computing $B_T$ is FPT if and only if $T$ is “h-Helly” for a fixed integer $h$ - a locality property that holds in particular for all finite unions of tractable classes defined by predicates on individual constraints, such as max-closed or connected row-convex constraints. This almost-dichotomy allows us to classify the parameterized complexity of strong backdoor detection for every tractable class studied in the literature except one, constant-closed languages, for which we exhibit a specialized polynomial-time algorithm.

Our analysis is so far only concerned with the exact computation of a minimum-size strong backdoor, which may be a little too ambitious for practical purposes. Thus, we define partition backdoors, a relaxation of strong backdoors that can always be computed in FPT time with respect to the backdoor size. We report preliminary experiments on the XCSP instance repository, with the target class $T$ being the (rather small) class of conservative majority languages. The results are promising, as small partition backdoors were identified for several instances in reasonable time. However the inability to handle constraints represented in intention (as opposed to lists of tuples) as well as the lack of efficient membership tests for larger tractable classes proved to be serious limitations to the framework, which are partly addressed in the other chapters.

These results were published in [1] and [2].

### 2.2 Detection algorithms for conservative tractable classes

A tractable class of CSP is conservative if membership in that class is never lost after reducing a variable domain. Those classes are of prime interest since domain manipulation is common in constraint programming, and for technical reasons they are the preferred targets for the backdoor approach of the previous chapter. Language-based conservative tractable classes are very well understood: fifteen years ago Bulatov described a necessary and sufficient condition for a conservative constraint language to be tractable, therefore establishing a dichotomy between tractable and intractable conservative languages. Bulatov’s formulation implies an algorithm for deciding if this condition holds, but this algorithm is super-exponential in the
domain size and is therefore unsuitable for most practical purposes.

Polymorphisms are operations from $D^k$ to $D$ that preserve relations under componentwise application. The polymorphisms of a language have been long known to determine its complexity, so it is not surprising that Bulatov’s conservative tractability criterion requires the existence of a collection of polymorphisms satisfying favourable identities. There has been remarkably little work on polynomial-time polymorphism detection algorithms prior to this dissertation, and the only known method fails to apply to Bulatov’s specific polymorphisms.

The most technical result of this thesis, and probably the strongest, is a proof that Bulatov’s criterion can be decided in polynomial time.

**Theorem 1.** The dichotomy for conservative constraint satisfaction is polynomially decidable.

The algorithm exploits the properties of conservative algebras to reduce the problem to a set of polynomially many highly-structured CSP instances, and then proceeds to solve them using a novel algorithmic technique which we call treasure hunt. The outline of treasure hunt seems classical - from a partial solution $\phi_k$ over $k$ well-chosen variables, it computes in polynomial time a partial solution $\phi_{k+c}$ over $k+c$ variables for a small constant $c$. The key here is that $\phi_{k+c}$ is not obtained by extending $\phi_k$, but is computed from scratch by exploring carefully a search space whose algebraic structure is partly described by $\phi_k$. We show that this method is applicable whenever we are looking for conservative polymorphisms satisfying identities which can be described by an idempotent linear Mal’tsev condition and whose existence imply solvability by a semiuniform algorithm. New polynomial-time detection algorithms for interesting polymorphisms are obtained as corollaries, essentially settling every important open problem of this kind in the conservative setting. These algorithms are then used to show that CSPs over languages with conservative Mal’tsev or conservative k-edge polymorphisms can be solved in “true” polynomial time, improving upon the best known algorithms which had an exponential dependency in the domain size.

Theorem 1 does not mean that testing for this criterion in real-world instances will be feasible in the near future - the number of operations needed to run our procedure is a polynomial whose degree is well above a hundred. The core issue is that our methods are coarse-grained, in the sense that we ignore most of the highly intricate local structure of the problem and focus on a few general properties instead. To address this problem we present an orthogonal, fine-grained analysis that allows us to prove the existence of simpler and extremely efficient detection algorithms for two special cases of conservative tractable languages, conservative majority and conservative Mal’tsev languages. The downside of this approach is that it does not scale well to more general settings as the technical details become rapidly overwhelming.

These results were published in [3] and [4].

### 2.3 Kernel-powered propagators for global constraints

At the heart of the success of constraint programming lies the concept of global constraints, which are families of relations represented algorithmically (through a propagator) rather than extensionally. These constraints scale very well to large arities without being overly memory-intensive, and the extensive literature on propagators for common types of global constraints makes up for the loss of a direct access to the list of tuples. Several of the most widely used global constraints, such as AtMostNValue, do not have polynomial-time exact propagators as their underlying predicates are NP-complete.
The goal of propagation is to compute all variable-value pairs \((x,v)\) such that setting \(x\) to \(v\) extends to a tuple of the constraint (those pairs are consistent). This process can be reduced to polynomially many decision problems, and most tractability notions for decision problems can be transferred to propagation problems this way (membership in P, fixed-parameter tractability...). However, kernelization, one of the most useful notion in parameterized complexity theory, does not translate easily.

For a parameterized decision problem, a kernelization is a polynomial-time computable function that maps the input instance to an equivalent one whose size is bounded by the parameter. Kernelizations are highly desirable, as for relevant parameters they may shrink considerably the problem size at polynomial cost. They often proceed by narrowing down the source of hardness to a few variables; the others can be eliminated in polynomial time. Kernelization has been used in the context of constraint propagation in the past, but always in tandem with probing - pick a variable-value pair \((x,v)\), set \(x \leftarrow v\), kernelize the obtained decision problem, solve and repeat. This use of kernelization is not quite satisfactory because probing is not a very efficient approach, as algorithms that perform the propagation in one sweep are generally much faster. Furthermore, the whole propagation process cannot be associated with a single, clear-cut kernel that highlights the most relevant variables and values for propagation purposes.

In this chapter, we develop a full theory of kernelization tailored for constraint propagation. Our loss-less kernelizations come in three distinct flavours, which correspond to different tradeoffs between efficiency and generality. In spirit, those kernelizations allow us to reduce the problem of propagating a global constraint to that of propagating a smaller one, in which both the number of values and the number of variables are bounded by the parameter. The most favourable of these kernelizations is extremely flexible and allows us to propagate partially the original constraint even if only partial information on the kernel is known; this process provably converges towards complete propagation as more information on the kernel becomes available.

To showcase this new theory of “constraint kernelization”, we use the case study of the VERTEX COVER constraint parameterized by cover size. Working with specialized variants of crown decompositions, we show that this constraint admits a full hierarchy of loss-less kernels of the strongest kind, with almost no increase in size compared to the best known kernel for the decision problem (which we prove not to be loss-less unless \(P=NP\)). We report experiments on random instances of the BALANCED VERTEX COVER problem, which we model as a CSP involving a VERTEX COVER constraint. We compare our kernel-based propagator (implemented in the MISTRAL constraint programming library) with other approaches and show that computing loss-less kernels significantly strengthens propagation, especially on relatively sparse graphs, and generally improves solving times.

These results were published in [5] and [6].

Publications arising from the thesis


