Detecting and Exploiting Subproblem Tractability

Christian Bessiere U. Montpellier Montpellier, France bessiere@lirmm.fr

Clement Carbonnel Université de Grenoble Grenoble France clement.carbonnel@ensimag.imag.fr

Emmanuel Hebrard LAAS-CNRS Toulouse, France hebrard@laas.fr

George Katsirelos MIAT, INRA Toulouse France george.katsirelos@toulouse.inra.fr

Toby Walsh NICTA & UNSW Sydney, Australia toby.walsh@nicta.com.au

Abstract

Constraint satisfaction problems may be *nearly* tractable. For instance, most of the relations in a problem might belong to a tractable language. We introduce a method to take advantage of this fact by computing a *backdoor* to this tractable language. The method can be applied to many tractable classes for which the membership test is itself tractable. We introduce therefore two polynomial membership testing algorithms, to check if a language is closed under a *majority* or conservative *Mal'tsev* polymorphism, respectively. Then we show that computing a minimal backdoor for such classes is fixed parameter tractable (FPT) if the tractable subset of relations is given, and W[2]-complete otherwise. Finally, we report experimental results on the XCSP benchmark set. We identified a few promising problem classes where problems were nearly closed under a majority polymorphism and small backdoors could be computed.

1 Introduction

The characterisation of tractable classes of constraint satisfaction problems is an active and important research area. However, most constraint toolkits do not exploit tractable classes in any specific way. Indeed, solvers often struggle to solve tractable problems [Petke and Jeavons, 2009]. What prevents tractability results from being more widely used in practice? Part of the answer comes from the problem of *testing membership*. The question of membership of a given problem instance to a polynomial class is not always easy to answer. Moreover, since many tractable classes exist, many membership tests may be needed. Unfortunately, membership testing has not received the same attention as the problem of identifying tractable classes in the first place. Additionally, problem instances are often messy and do not belong to any known tractable class. For instance, many tractable classes are defined by a restricted constraint language and even a single extra constraint excludes an instance from this tractable class. In this case, even though the instance is "close" to being a member of the class, we cannot directly exploit tractability.

We make a significant step toward filling the gap between theory and practice in this domain. We introduce a method to take advantage of instances "almost" fitting a tractable class. Proximity to a tractable class is a powerful tool in solving a problem instance. The basic idea is to branch first on the part of the problem that makes it intractable in order to obtain a tractable subproblem. We propose a scheme that performs such reasoning automatically. We quantify the tractability of a CSP instance with respect to a tractable class Γ by the size of the smallest *backdoor* [Williams *et al.*, 2003], i.e., the smallest set of variables that yields a member of Γ when instantiated. Once a backdoor is found, we can solve an instance by branching first on the backdoor variables, and then using a polynomial algorithm for Γ. Instances of this type are fixed parameter tractable (FPT) in the parameter k equal to the size of their backdoor.

We consider first the tractable class defined by languages admitting a *majority polymorphism*. Unfortunately, we prove it is W[2]-hard to compute a minimal backdoor with respect to this class. However, if we fix the target tractable language (i.e., we fix a subset of the relations of the problem admitting a majority polymorphism), then finding a minimal backdoor is fixed parameter tractable. This tractable subset can be common to families of instances. In this case, even if obtaining such a subset is computationally expensive, the cost can be amortized over many instances. For each instance, we can compute a backdoor *relative* to the tractable subset computed offline, and solve the instance using it, both with a complexity exponential only in the size of the backdoor. This scheme thus constitutes a way to take advantage automatically of an instance being "almost" tractable. We report experimental results assessing the practicality of this approach.[1]

Since our method is highly dependent on the ability to test membership to a tractable class efficiently, we introduce new polynomial membership testing algorithms for two important tractable classes. A future challenge is to reduce the large polynomial complexity of these algorithms and to devise algorithms for other classes. Nevertheless, we were able to use our algorithms in experiments on the XCSP [Roussel and Lecoutre, 2009] benchmark library. Some problem classes in this library are indeed "almost" tractable as a large subset of their language is closed under a majority polymorphism.

The paper is organised as follows: In Section 3, we in-

[1] The source code developed for this evaluation is available at http://sourceforge.net/projects/kpoly

troduce two polynomial membership testing algorithms for tractable classes where the membership test was not previously known to be tractable. Then in Sections 4 and 5, we describe our general FPT scheme, and analyse its complexity depending on if we know which subset of the language is tractable. Finally, in Section 6 we report experimental results with majority polymorphisms on the XCSP repository.

2 Formal Background

Constraint Satisfaction Problems. A constraint satisfaction problem (CSP) consists of deciding if a constraint network has solutions. A constraint network $\mathcal{P}$ is defined by the triple $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$, where $\mathcal{X} = \{x_1, \ldots, x_n\}$ is a set of n variables, $\mathcal{D}$ is a set of values, $\mathcal{D}(x) \subseteq \mathcal{D}$ is the finite set of possible values for x (its domain), and $\mathcal{C}$ is set of constraints. A constraint $C \in \mathcal{C}$ consists of a scope $X(C)$ and a relation $R(C)$ over the domains of the variables in $X(C)$. $V(C)$, the set of available values for variables constrained by C is defined by $V(C) = \bigcup_{x \in X(C)} \mathcal{D}(x)$. Given a network $\mathcal{P} = \langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$, we let $n = |\mathcal{X}|$ be the number of variables, $d = |\mathcal{D}|$ be the number of values, $e = |\mathcal{C}|$ be the number of constraints, and $t = \max_{C \in \mathcal{C}}(|R(C)|)$ be the maximum number of tuples in a relation. Given a tuple $\tau \in R(C)$, we denote $\tau[x]$ the element of τ corresponding to $x \in X(C)$.

Constraint solvers typically use backtracking search to explore the space of partial assignments. After each assignment, constraint propagation algorithms prune the search space by enforcing a local consistency. Given a variable x, a value $v \in \mathcal{D}(x)$ and a constraint C such that $x \in X(C)$, the *assignment* $\langle x, v \rangle$ is *arc consistent* (AC) with respect to C iff there exists a tuple $\tau \in R(C)$ such that $\tau[x] = v$ and $\forall y \in X(C)$, $\tau[y] \in \mathcal{D}(y)$. Such a tuple is called a *support* for $\langle x, v \rangle$. A constraint network $\mathcal{P} = \langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$ is arc consistent iff, for all variables $x \in \mathcal{X}$, for all values $v \in \mathcal{D}(x)$, $\langle x, v \rangle$ is arc consistent with respect to all constraints $C \in \mathcal{C}$. We write $AC(\mathcal{P})$ for the constraint network obtained by removing all values from $\mathcal{P}$ that are not arc consistent. If the resulting constraint network is empty, $\mathcal{P}$ is *arc inconsistent*. An assignment $\langle x, v \rangle$ is *singleton arc consistent* (SAC) iff, the constraint network obtained by setting x to v is not arc inconsistent. For a network $\mathcal{P}$, we define a *constraint hypergraph* $G(\mathcal{P})$ with a vertex for each variable and a hyperedge equal to the scope of each constraint. The *primal graph* $G_c(\mathcal{P})$ is obtained from $G(\mathcal{P})$ by replacing each hyperedge by a clique of edges.

Polymorphisms. Throughout the paper we will consider classes of CSPs whose languages admit a *polymorphism* of a certain kind. Constraint problems whose language is closed under certain kinds of polymorphisms (*majority, affine, constant, or binary idempotent commutative and associative*) are tractable [Jeavons *et al.*, 1997]. It was later shown [Bulatov and Dalmau, 2006] that the *Mal'tsev* polymorphism also yields a tractable language.

Let C be a constraint, $r = |X(C)|$ its arity. The mapping $f : V(C)^m \mapsto V(C)$, of arity[2] m, is a polymorphism of $R(C)$ (or $R(C)$ is closed under f) iff for every m-tuple $(\tau_1, \ldots, \tau_m)$ of elements of $R(C)$, we have:

$$\langle f(\tau_1[1], \ldots, \tau_m[1]), \ldots, f(\tau_1[r], \ldots, \tau_m[r]) \rangle \in R(C)$$

Consider the relations $R_1 = \{\langle 1, 0 \rangle, \langle 1, 1 \rangle\}$ and $R_2 = \{\langle 0, 1, 0 \rangle, \langle 0, 0, 0 \rangle, \langle 1, 0, 1 \rangle, \langle 1, 1, 1 \rangle\}$. The function $f(x, y) = (x + y \bmod 2)$ is not a polymorphism of R_1 because $\langle f(1, 1), f(0, 1) \rangle = \langle 0, 1 \rangle \notin R_1$. However, it is a polymorphism of R_2. To see this we consider the operator f on all possible pairs of tuples (since f is commutative we considerer unordered pairs). Pairs involving the tuple $\langle 0, 0, 0 \rangle$ are mapped to the other element of the pair. Repetitions of a tuple t are mapped to $\langle 0, 0, 0 \rangle$. We illustrate the three remaining cases below. For each pair of tuples τ_1, τ_2 and each variable x_i we give the value of the function $f(\tau_1[x_i], \tau_2[x_i])$:

f	f	f	f	f	f	f	f	f
0	1	0	0	1	0	1	0	1
1	0	1	1	1	1	1	0	1
$\|\|$	$\|\|$	$\|\|$	$\|\|$	$\|\|$	$\|\|$	$\|\|$	$\|\|$	$\|\|$
1	1	1	1	0	1	0	1	0

A ternary polymorphism f has the *majority* property iff for all $v, w \in V(C)$, it holds that $f(v, v, w) = f(v, w, v) = f(w, v, v) = v$. It is *Mal'tsev* if $v, w \in V(C)$, it holds that $f(v, v, w) = f(w, v, v) = w$. Note a polymorphism cannot be both Mal'tsev and majority. A polymorphism f of arity m is *idempotent* iff for all $v \in V(C)$, $f(v, \ldots, v) = v$. Moreover, it is *conservative* iff for all $\langle v_1, \ldots, v_m \rangle \in \mathcal{D}^m$: $f(v_1, \ldots, v_m) \in \{v_1, \ldots, v_m\}$.

Parameterized complexity. A problem is *fixed-parameter tractable* (*FPT*) if it can be solved in $O(f(k)n^c)$ time where f is *any* computable function, n is the size of the input, k is some parameter, and c is a constant. For example, vertex cover is fixed-parameter tractable with respect to the size of the cover k since it can be solved in $O(1.31951^k k^2 + kn)$ time [Downey *et al.*, 1999]. Above FPT, Downey and Fellows have proposed a hierarchy of fixed-parameter *intractable* problem classes:

$$FPT \subseteq W[1] \subseteq W[2] \subseteq \ldots \subseteq XP$$

For instance, the clique problem is $W[1]$-complete with respect to the size of the clique, whilst the dominating set problem is $W[2]$-complete with respect to the size of the dominating set. There is considerable evidence to suggest that $W[1]$-hardness implies parametric intractability and that $W[t]$ is strictly easier than $W[t+1]$. The class XP contains problems that can be solved in time $O(n^{f(k)})$.

3 Detecting Tractable Problems

We first give two new polynomial membership testing algorithms: one for testing whether a set of relations is closed under a majority polymorphism, and a second for testing whether a set of binary relations is closed under a conservative Mal'tsev polymorphism. We shall use the notion of *indicator problem* introduced in [Jeavons *et al.*, 1997].

[2]Observe that the arity of a polymorphism is orthogonal to the arity of a constraint.

Definition 1 (Indicator problem)**.** *Let Γ be a set of relations over a domain $\mathcal{D}$. The indicator problem of order m of Γ, denoted $\mathcal{P}_m(\Gamma, \mathcal{D})$, is defined as follows:*

- *A set of variables $\mathcal{X} = \{x_{\bar{v}} \mid \bar{v} \in \mathcal{D}^m\}$;*
- *A domain (common to all variables) $\mathcal{D}$;*
- *Each relation $R \in \Gamma$ of arity r, and each tuple sequence $\langle \tau_1, \ldots, \tau_m \rangle \in R^m$, yields the constraint $(\langle x_{\bar{v}_1}, \ldots, x_{\bar{v}_r} \rangle, R)$ where $\bar{v}_i = \langle \tau_1[i], \ldots, \tau_m[i] \rangle$.*

The solutions of an indicator problem $\mathcal{P}_m(\Gamma, \mathcal{D})$ are polymorphisms of a language Γ over a domain $\mathcal{D}$. There is one variable per m-tuple of values in $\mathcal{D}$, standing for the image of this tuple under the polymorphism. For a relation R of arity r, and for each combination of m tuples of R, there is a constraint on the r variables obtained by traversing these m tuples component-wise whose relation is precisely R. This constraint ensures that applying the polymorphism component-wise on these tuples yields a tuple that belongs to R.

3.1 Majority

It has been recently shown [Chen *et al.*, 2013] that we can solve a problem backtrack-free by applying SAC at each node when the set of relations admits a majority polymorphism. We shall use this property to detect a majority polymorphism. We use a specific indicator problem $\mathcal{P}_{maj}(\Gamma, \mathcal{D})$ defined as $\mathcal{P}_3(\Gamma, \mathcal{D})$, with the following extra constraints:

$$\forall \langle v, w \rangle \in \mathcal{D}^2,\ x_{\langle v,v,w \rangle} = x_{\langle v,w,v \rangle} = x_{\langle w,v,v \rangle} = v$$

Theorem 1. *The existence of a majority polymorphism of a language Γ over a domain $\mathcal{D}$ can be decided in $O(rd^7lt^3(t + d))$. where $l = |\Gamma|$, $d = |\mathcal{D}|$, t and r are, respectively, the maximum size and arity of a relation in Γ.*

Proof: There is a bijection between solutions of $\mathcal{P}_{maj}(\Gamma, \mathcal{D})$ and majority polymorphisms of Γ over $\mathcal{D}$. Notice that the language of $\mathcal{P}_{maj}(\Gamma, \mathcal{D})$ is precisely Γ plus extra unary constraints that are by definition closed under the majority operation. This ensures that Γ admits a majority polymorphim over $\mathcal{D}$ iff $\mathcal{P}_{maj}(\Gamma, \mathcal{D})$ can be solved backtrack-free using SAC.

The following backtrack-free procedure detects a majority polymorphism of a set of relations Γ over a domain $\mathcal{D}$:

- Build the indicator problem $\mathcal{P}_{maj}(\{R(C) \mid C \in \mathcal{C}\}, \mathcal{D})$.
- Iteratively enforce SAC on this problem and assign a variable.
- If SAC fails return `False`, otherwise, if all variables are assigned return `True`.

SAC can be enforced in $O(ndT)$ on a constraint network involving n variables of domain size d, where T is the time complexity of achieving AC [Bessiere and Debruyne, 2008]. The network $\mathcal{P}_{maj}(\Gamma, \mathcal{D})$ has d^3 variables of domain size d, and lt^3 constraints. Since AC can be achieved in $O(er(t+d))$ and SAC can be enforced up to d^3 times on this network, the total time complexity of the procedure is $O(rd^7lt^3(t + d))$. In addition, building the indicator problem takes lt^3 time. □

3.2 Conservative Mal'tsev on binary relations

As previously, we shall use a specific form of indicator problem to detect conservative Mal'tsev polymorphisms of binary relations. We call $\mathcal{P}_{mal}(\Gamma, \mathcal{D})$ the constraint network equal to $\mathcal{P}_3(\Gamma, \mathcal{D})$ with the extra following constraints:

$$\forall \langle v, w \rangle \in \mathcal{D}^2,\ x_{\langle v,w,w \rangle} = x_{\langle w,w,v \rangle} = v$$

$$\forall \langle u, v, w \rangle \in \mathcal{D}^3,\ x_{\langle u,v,w \rangle} \in \{u, v, w\}$$

Moreover, for the purpose of our demonstration, we shall reformulate the indicator problem $\mathcal{P}_{mal}(\Gamma, \mathcal{D})$ as follows. For every triple $\langle u, v, w \rangle$ we replace the domain of $x_{\langle u,v,w \rangle}$ by $\{1, 2, 3\}$, and for each constraint $C = (S, R)$ of $\mathcal{P}_{mal}(\Gamma, \mathcal{D})$, where $S = \langle x_{\langle v_1,v_2,v_3 \rangle}, x_{\langle w_1,w_2,w_3 \rangle} \rangle$, we replace R by $R_{|S} = \{\langle i, j \rangle \mid \langle v_i, w_j \rangle \in (\{v_1, v_2, v_3\} \times \{w_1, w_2, w_3\} \cap R)\}$. Notice that values may be repeated. For instance the domain of $x_{\langle v,w,v \rangle}$ is $\{1, 2, 3\}$, however the values 1 and 3 are symmetric and both share the same supports corresponding to those of v. Clearly, the reformulation preserves the satisfiability of $\mathcal{P}_{mal}(\Gamma, \mathcal{D})$, though it may introduce symmetric solutions.

Lemma 1 (derived from [Bulatov, 2002])**.** *Let R be some binary relation that has a Mal'tsev polymorphism. Then, R is rectangular, i.e., for any values a, b, c, d, if $< a, c >$, $< a, d >$ and $< b, c > \in R$, then $< b, d > \in R$.*

It follows that a binary constraint C admits a Mal'tsev polymorphism only if the digraph whose edges are given by the tuples of $R(C)$ is a set of disjoint bicliques.

Lemma 2. *Let Γ be a language of binary relations, $\mathcal{D}$ be a set of values, and $R \in \Gamma$. Let $S = \langle x_{\langle u,v,w \rangle}, x_{\langle a,b,c \rangle} \rangle$ and $C = (S, R_{|S})$ be a constraint of $AC(\mathcal{P}_{mal}(\Gamma, \mathcal{D}))$. If Γ is closed under a conservative Mal'tsev polymorphism, then $R_{|S}$ is either:*

- *The equality relation, or*
- *The universal relation, or*
- *The union of $\{\langle 2, 2 \rangle\}$ and $\{1, 3\} \times \{1, 3\}$.*

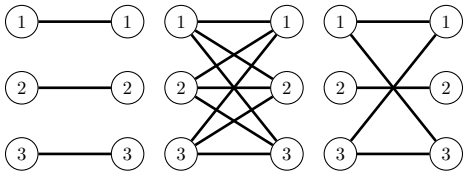
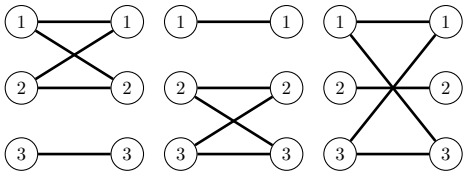


Figure 1: The 3 possible structures for $R_{|S}$ after AC.

Proof: Observe that C can only be part of the indicator problem if $\{\langle u, a \rangle, \langle v, b \rangle, \langle w, c \rangle\} \subseteq R$, hence $\{\langle 1, 1 \rangle, \langle 2, 2 \rangle, \langle 3, 3 \rangle\} \subseteq R_{|S}$.

Then, by Lemma 1, $R_{|S}$ is either :

- A single biclique (universal relation) ;
- 3 bicliques, (equality relation) ;



(a) Case 1 (b) Case 2 (c) Case 3

- 2 bicliques (one has a single edge).

The latter case yields 3 possibilities :

In case 1, the existence of the edges $\{\langle 1,1\rangle, \langle 2,1\rangle, \langle 3,3\rangle\}$ implies that $\{\langle u,a\rangle, \langle v,a\rangle, \langle w,c\rangle\} \subseteq R$. Consequently, the constraint $(\langle x_{\langle u,v,w\rangle}, x_{\langle a,a,c\rangle}\rangle, R_{|\langle x_{\langle u,v,w\rangle}, x_{\langle a,a,c\rangle}\rangle})$ belong to $\mathcal{P}_{mal}(\Gamma, \mathcal{D})$ with $R_{|\langle x_{\langle u,v,w\rangle}, x_{\langle a,a,c\rangle}\rangle} = R_{|S}$. However, by construction of the indicator problem $\mathcal{D}(x_{\langle a,a,c\rangle}) = \{3\}$. Therefore, enforcing AC on this constraint will reduce the domain of $x_{\langle u,v,w\rangle}$ to the singleton $\{3\}$ since only the tuple $\langle 3,3\rangle$ is a support. In other words, this constraint will be entailed, hence can be removed after enforcing AC.

The same reasoning applies to case 2, and the only possibility left is case 3. □

Corollary 1. *If we:*

- *Enforce arc consistency on the indicator problem,*
- *Remove all constraints involving assigned variable(s),*
- *Merge all variables constrained together by an equality,*
- *Remove all universal relations,*

Then the resulting network only has constrained variables with domain size ≥ 2, and every relation will be equal to:

$$\{\langle 1,1\rangle, \langle 1,3\rangle, \langle 2,2\rangle, \langle 3,1\rangle, \langle 3,3\rangle\}$$

Therefore, assigning every non-assigned variable x to $\{1,3\} \cap \mathcal{D}(x)$ is always a solution. Indeed, every variable's domain contains at least two values so it contains either 1 or 3, and they are always compatible. Now we can easily prove that this tractable class can be detected in polynomial time.

Theorem 2. *The existence of a conservative Mal'tsev polymorphism of a language Γ over a domain $\mathcal{D}$ can be decided in $O(ld^6)$ where $l = |\Gamma|$ and $d = |\mathcal{D}|$.*

Proof: By Corollary 1, we know that if Γ is closed under a conservative Mal'tsev polymorphism, after a sequence of transformations (including AC), we can solve the indicator problem by assigning any value in $\{1,3\} \cap \mathcal{D}(x)$ to each variable x. Therefore, if enforcing AC returns a failure, or if no such such assignment can be built, then we know that Γ is not closed under any Mal'tsev polymorphism. Moreover, each step in this procedure takes linear time in the number of constraints of the indicator problem (which equals d^6 where $d = |\mathcal{D}|$ in the worst case). Creating the indicator problem takes $O(ld^6)$ time. □

4 Exploiting Tractable Subproblems

Our approach to exploiting tractable subproblems is by identifying a small backdoor. We are not limited to a specific tractable class, but can use any class characterized by a polymorphism that satisfies either of two properties: idempotency, or, more strongly, being conservative. The stronger property gives an algorithm that is FPT in a strictly smaller parameter. Both majority and Mal'tsev imply idempotency. The exact problem is stated as follows.

NAME: ALMOST-TRACTABLE-CSP(P)
INPUT: A CSP $\mathcal{P} = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}_1 \cup \mathcal{C}_2\rangle$ and a property P such that $\mathcal{P}_2 = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}_2\rangle$ belongs to a tractable class characterised by a polymorphism with property P.
PARAMETERS: d: domain size, m: the number of variables in $\mathcal{C}_1$, k: the minimum vertex cover of the primal graph of $\mathcal{C}_1$, k_h: the minimum hyper vertex cover of the hyper graph of $\mathcal{C}_1$, and r: the maximum arity of constraints in $\mathcal{C}_1$.
PROBLEM: Is $\mathcal{P}$ satisfiable?

Our approach to exploit almost tractable subproblems is inspired by the *cycle-cutset* method [Dechter and Pearl, 1987]. This instantiates variables of the constraint network until the remaining subproblem is acyclic, where directed arc consistency will solve the problem in polynomial time [Freuder, 1982]. The cycle cutset method can be seen as one of the first attempts to solve almost tractable problems. However, the tractable class was based on the structure of the network rather than the language.

4.1 Idempotent classes

An assignment is a unary relation with a single tuple, which is closed under any idempotent polymorphism. We use this in our first FPT algorithm.

Theorem 3. ALMOST-TRACTABLE-CSP(IDEMPOTENT) *is FPT with parameter $d + m$.*

Proof: We show that the set of all m variables that participate in constraints in $\mathcal{C}_1$ form a backdoor with respect to the algorithm for the tractable class. Indeed, let $\mathcal{C}_2$ be closed under the idempotent polymorphism f. Since $f(v,\ldots,v) = v$, any relation with a single tuple is closed under f, so the union of $\mathcal{C}_2$ with the assignment relations is also closed under f. Therefore, instantiating the variables that appear in constraints in $\mathcal{C}_1$ leaves a tractable instance. The search tree entailed by the backdoor has size d^m. □

4.2 Conservative classes

A set of prunings can be expressed by the addition of unary constraints to a CSP. Moreover, all unary relations are closed under any conservative polymorphism. In this case there exists an algorithm which is exponential in the strictly smaller parameter k. We note that $k \leq (r-1)k_h < m$.

Theorem 4. ALMOST-TRACTABLE-CSP(CONSERVATIVE) *is FPT with parameter $d + k$.*

Proof: We use this vertex cover of the primal graph of $\mathcal{P}_1$ as a backdoor. That is, we branch on the variables of the vertex cover, yielding a tree of size d^k. At each leaf of this tree, all constraints of $\mathcal{C}_1$ are reduced to unary constraints, so the induced CSP $\mathcal{P}'$ has all the constraints of $\mathcal{P}_2$ and additionally some unary constraints, therefore $\mathcal{P}'$ belongs to the same

conservative tractable class as $\mathcal{P}_2$ and its satisfiability can be determined in polynomial time. □

5 Identifying Tractable Subproblems

When we do not know explicitly which set of relations belongs to a tractable language, we face a harder problem than just finding the backdoor. We must also guess the subset of relations that gives a small backdoor. We will demonstrate this difficulty on languages that are tractable under a majority polymorphism. We conjecture that similar results hold for other tractable languages.

NAME: PARTITION-MAJORITY-CSP
INPUT: A CSP $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$, integer k
PROBLEM: Does there exist a partition of C into $\mathcal{C}_1, \mathcal{C}_2$ such that $\mathcal{C}_2$ admits a conservative majority polymorphism and $\mathcal{C}_1$ has a vertex cover of size at most k?

Theorem 5. PARTITION-MAJORITY-CSP *is NP-complete even for binary CSPs.*

Proof: A partition along with the vertex cover provides a witness, because of Theorem 1, so the problem is in NP. For hardness, we reduce from Vertex Cover on a graph $G = \{V, E\}$. We construct a CSP $P = \langle X, C \rangle$ with a variable for each vertex and a constraint between two variables iff there exists an edge between the corresponding vertices. All constraints are the same relation $R = \{(1,1), (1,2), (2,3), (4,1), (5,2), (4,3), (7,1), (6,2), (6,3)\}$. This relation has no majority polymorphism, as the first three tuples require that $f(1,2,3) \in \{1,2\}$, the next three require that $f(1,2,3) \in \{2,3\}$ and the last three require that $f(1,2,3) \in \{1,3\}$. No subset of constraints admits a majority polymorphism, so C_2 has to be empty. Hence the problem is equivalent to finding the minimum vertex cover. □

When the parameter is the size of the vertex cover $G_c(\mathcal{P})$, it is easy to see that PARTITION-MAJORITY-CSP is in XP. Let S be a subset of k variables and $\mathcal{C}_1$ be the constraints which are covered by S in $G_c(\mathcal{P})$. We can test in polynomial time whether $\mathcal{C}_2$ has a conservative majority polymorphism. There are $O(n^k)$ such sets, which gives an XP algorithm.

Lemma 3. *For any $k \geq 3$, there exists a set of k relations such that every subset of $k-1$ relations admits a conservative majority polymorphism but the entire set does not.*

Proof: Let $R_1 = \{(1,3), (1,4), (2,5)\}$, $R_2 = \{(1,3), (2,4), (1,5)\}$, $R_3 = \{(2, 3(k-2)), (1, 3(k-2)+1), (2, 3(k-2)+2)\}$ and $R_i = \{((3(i-3), 3(i-2)), (3(i-3)+1, 3(i-2)+1), (3(i-3)+2, 3(i-2)+2)\}$ for $4 \leq i \leq k$. For example, for $k = 4$ this is instantiated to

R_1		R_2		R_3		R_4	
1	3	1	3	2	6	3	6
1	4	2	4	1	7	4	7
2	5	1	5	1	8	5	8

We first show that the complete set of relations admits no conservative majority polymorphism f. From R_1 we get $f(3,4,5) \in \{3,4\}$ and from R_2 we get $f(3,4,5) \in \{3,5\}$, so $f(3,4,5) = 3$. From R_3 we get $f(3(k-2), 3(k-2)+1, 3(k-2)+2) \in \{3(k-2)+1, 3(k-2)+2\}$. Finally, from $R_i, 4 \leq i \leq k$, we get $f(3(i-2), 3(i-2)+1, 3(i-2)+2) = j \iff f(3(i-1), 3(i-1)+1, 3(i-1)+2) = j+3$. This chain gives us that $f(3,4,5) \in \{4,5\} \not\ni 3$.

On the other hand, consider any subset of at most $k-1$ relations. If we omit either R_1 or R_2, we only require $f(3,4,5) \in \{3,4\}$ (respectively, $\{3,5\}$) which has a non-empty intersection with $\{4,5\}$. If we omit R_3, we place no restriction on $f(3(k-2), 3(k-2)+1, 3(k-2)+2)$, so it does not conflict with $f(3,4,5) = 3$. Finally, if we omit any of the relations $R_4, \ldots, R_k$, we break the chain that connects $f(3,4,5)$ to $f(3(k-2), 3(k-2)+1, 3(k-2)+2)$, so they can be chosen independently. Therefore, any subset of $k-1$ relations has a conservative majority polymorphism. □

Theorem 6. PARTITION-MAJORITY-CSP *is $W[2]$-hard when the parameter is k, the size of the vertex cover of the primal graph $G_c(\mathcal{P})$.*

Proof: We reduce from an instance of Hitting Set over the universe U, with sets $S_1, \ldots, S_m$, each of size p, and with minimum hitting set size s to an instance of PARTITION-MAJORITY-CSP with $k = s$ with maximum arity 2. Indeed, Hitting Set is $W[2]$-hard for the parameter s. Note that we assume all sets have the same size. If that is not the case, we pad each set using new unique elements.

We create a variable X_e for each element e in the Hitting Set problem and a variable X_S for each set S. For each set S, we create a set R_S of p relations using the method of lemma 3 and using distinct values for each set. We then create p constraints $C_{X_S X_e}$ for each $e \in S$ using the relations of R_S. This CSP has a partition into $\mathcal{C}_1$ and $\mathcal{C}_2$ s.t. the vertex cover of $\mathcal{C}_1$ is $k = s$ iff the minimum hitting set has size s.

($\Rightarrow$) Suppose H is a hitting set. For each element $e \in H$ and for every set S such that $e \in S$, we place $C_{X_e X_S}$ in $\mathcal{C}_1$. Since H is a hitting set, $\mathcal{C}_2$ contains at most $p-1$ relations from each set of p relations that involve variable X_S. Therefore, the relations involving X_S that are in $\mathcal{C}_2$ admit a conservative majority polymorphism. Since relations involving X_S are over disjoint domains with relations involving $X_{S'}$ for any set S' different from S, they do not interfere with the parts of the polymorphism affecting each other. Thus $\mathcal{C}_2$ has a conservative majority polymorphism. Moreover, the set $\{X_e | e \in H\}$ is a vertex cover of $\mathcal{C}_1$ by the fact that we only include constraints that involve one of these variables in $\mathcal{C}_1$.

($\Leftarrow$) We first show that $\mathcal{C}_1$ includes exactly one constraint of the form $C_{X_e X_S}$ for each set S. It includes at least one such constraint, otherwise $\mathcal{C}_2$ does not admit a majority polymorphism. To show it includes at most one such constraint, observe first that vertex cover is monotone, i.e., adding edges cannot decrease the size of the minimum vertex cover. Second, in order to ensure that $\mathcal{C}_2$ admits a majority polymorphism, we only need one out of every p relations that involve a variable X_S to be in $\mathcal{C}_1$ and by monotonicity of vertex cover we can assume that the other $p-1$ are relations are in $\mathcal{C}_2$.

Since each X_S appears in exactly one relation in $\mathcal{C}_1$, if it is in the vertex cover to cover the constraint $C_{X_S X_e}$, we can replace it by X_e and get a vertex cover that is no larger. So there exists a minimum size vertex cover with variables X_e only. This set of variables corresponds to a set of elements covering the sets $S_1, \ldots, S_m$, because each covers at least

one relation $C_{X_e X_S}$, these constraints encode set inclusion and all sets are covered by the chosen relations. □

This result also implies $W[2]$-hardness for partitioning when the tractable subproblem is closed under a non-conservative majority polymorphism. On the other hand, if the parameter also includes the domain size then we lose hardness. However, the complexity of the best algorithm we have in this case is rather impractical.

Theorem 7. PARTITION-MAJORITY-CSP *is FPT when the parameter is* $d + k + r$.

Proof: Given domain size d and maximum constraint arity r, there are 2^{d^r} possible relations. Suppose C_1 and C_2 are the partitions. If there exist two constraints c_1 and c_2 with the same relation R such that $c_1 \in \mathcal{C}_1$ and $c_2 \in \mathcal{C}_2$, then the partition $\mathcal{C}'_1 = \mathcal{C}_1 \setminus \{c_1\}, \mathcal{C}'_2 = \mathcal{C}_2 \cup \{c_1\}$ is such that $\mathcal{C}'_2$ still admits a conservative majority polymorphism, while the minimum vertex cover of $\mathcal{C}'_1$ cannot be larger than the minimum vertex cover of $\mathcal{C}_1$. So it is sufficient to consider only partitions of the relations rather than of the constraints. There are $2^{2^{d^r}}$ possible partitions of the relations into $\mathcal{C}_1$ and $\mathcal{C}_2$ and for each of these we can discover the minimum vertex cover of the primal graph of $\mathcal{C}_2$ in time 2^k, and verify in polynomial time that $\mathcal{C}_2$ admits a conservative majority polymorphism. The total complexity of this algorithm is $O(2^{2^{d^r}} d^k p(n))$. □

2^{d^r} is much larger than the number of constraints in any practical problem, even for very small values of d and r. However, when the number of different relations in an instance is q, it is always the case that $q < 2^{d^r}$.

Corollary 2. PARTITION-MAJORITY-CSP *is FPT when the parameter is* $k + q$*, where* k *is the size of the vertex cover and* q *is the number of distinct relations in* $\mathcal{P}$.

Proof: The proof of theorem 7 goes through when we only consider the q different relations present in $\mathcal{P}$. The complexity of this algorithm is $O(d^k 2^q p(n))$. □

6 Implementation and Evaluation

In order to assess the practical value of this approach, and to find almost-polynomial series of instances, we have implemented this procedure using conservative majority polymorphisms as the target polynomial class. We implemented the algorithm of Corollary 2 in a branch-and-bound fashion, rather than generate-and-test. We made several improvements which reduce runtime in practice. First, we performed preprocessing of an instance with SAC to reduce the size of the indicator problem. Second, we implemented a nogood database of sublanguages which admit no majority polymorphism in order to avoid testing any of their supersets. We also perform pre-emptive detection of trivially blocking and majority-friendly relations. For testing existence of a majority polymorphism, we enforce SAC on the indicator problem using SAC3-SDS [Bessiere *et al.*, 2011]. We compute vertex covers using the basic $O(2^k p(n))$ algorithm combined with a few heuristics inspired from [Balasubramanian *et al.*, 1998].

We implemented the algorithm in C++ and ran experiments on a 2.2 Ghz Intel Core i7 with 8 Gb of memory on 191 series of instances made available from the Fourth International Constraint Solver Competition. We used only instances without global constraints and converted all constraints to extensional form. For each family of instances, we tested the algorithm on a sample of representative instances. We placed an upper bound of 40 on the computed backdoor size. If the algorithm reported no backdoor for any instance in the sample, we pessimistically concluded that no instance in that family has a small backdoor, otherwise we ran the algorithm on the remaining instances.

The results are mostly (but not wholly) negative, as we failed to find non-trivial tractable subproblems on most instances of the 191 tested families. The most frequent cause of failure is the indicator problem being too large to fit in memory (primary cause for 135 families). Other causes include being solved by preprocessing (16), or having a large backdoor size with respect to the problem size (40).

However, some families do show promising results. The 5 "`prime`" families, created by van Dongen, and in which each constraint is a linear equation with prime numbers as coefficients produced the most interesting tractable subproblems. For each of these series, the relations are intentional and conversion to extensional form yields very large relations. However, when the conversion is possible (9 instances out of 36 for the first family), a subproblem closed under conservative majority polymorphism was found with a very small backdoor (see figure 2). We were also able to find a backdoor of size 22 on the instance "`driverlogw-01c-sat_ext`", involving 71 variables. However this family contains only seven instances, and the other six either had no small backdoor or too large an indicator problem.

Instance	#Var.	#Act.	#Fix.	#Rel.	time (s)
driverlogw-01c	71	53	22	2/14	0.40
primes-10-20-2-1	100	79	3	2/20	439.92
primes-10-40-2-1	100	56	6	3/40	344.92
primes-10-40-3-1	100	43	3	3/40	7.80
primes-10-60-2-1	100	28	0	0/59	0.01
primes-10-60-2-3	100	14	0	0/59	0.09
primes-10-60-3-1	100	17	0	0/59	0.00
primes-10-80-2-1	100	11	0	0/79	0.00
primes-10-80-2-3	100	4	0	0/79	0.08
primes-10-80-3-1	100	8	0	0/79	0.01

Figure 2: Results for almost tractable instances. #Var is the number of variables, #Act is the number of non-assigned variables after SAC, #Fix is the size of the backdoor and #Rel is the ratio of relations that need to be removed.

7 Conclusions

We have argued that we can exploit constraint satisfaction problems which are *nearly* tractable. The basic idea is to compute a *backdoor* into a tractable language. Our method requires a tractable membership test. We therefore introduced two new polynomial time membership testing algorithms which check if a language admits a *majority* or a conservative *Mal'tsev* polymorphism. We proved that computing a minimal backdoor for such classes is fixed parameter tractable when the tractable subset of relations is given, and W[2]-complete otherwise.

References

[Balasubramanian *et al.*, 1998] R. Balasubramanian, Michael R. Fellows, and Venkatesh Raman. An Improved Fixed-Parameter Algorithm for Vertex Cover. *Inf. Process. Lett.*, 65(3):163–168, 1998.

[Bessiere and Debruyne, 2008] Christian Bessiere and Romuald Debruyne. Theoretical analysis of singleton arc consistency and its extensions. *Artif. Intell.*, 172(1):29–41, 2008.

[Bessiere *et al.*, 2011] Christian Bessiere, Stephane Cardon, Romuald Debruyne, and Christophe Lecoutre. Efficient Algorithms for Singleton Arc Consistency. *Constraints*, 16:25–53, 2011.

[Bulatov and Dalmau, 2006] Andrei A. Bulatov and Víctor Dalmau. A simple algorithm for mal'tsev constraints. *SIAM J. Comput.*, 36(1):16–27, 2006.

[Bulatov, 2002] Andrei A. Bulatov. Mal'tsev constraints are tractable. *Electronic Colloquium on Computational Complexity (ECCC)*, (034), 2002.

[Chen *et al.*, 2013] Hubie Chen, Victor Dalmau, and Berit Grußien. Arc Consistency and Friends. *J Logic Computation*, 23(1):87–108, 2013.

[Dechter and Pearl, 1987] R. Dechter and J. Pearl. The cycle-cutset method for improving search performance in ai applications. In *Proceedings 3rd IEEE Conference on AI Applications*, pages 224–230, Orlando FL, 1987.

[Downey *et al.*, 1999] R. G. Downey, M. R. Fellows, and U. Stege. Parameterized complexity: A framework for systematically confronting computational intractability. In *Contemporary Trends in Discrete Mathematics: From DIMACS and DIMATIA to the Future*, volume 49 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 49–99. 1999.

[Freuder, 1982] Eugene C. Freuder. A sufficient condition for backtrack-free search. *J. ACM*, 29(1):24–32, 1982.

[Jeavons *et al.*, 1997] Peter Jeavons, David A. Cohen, and Marc Gyssens. Closure properties of constraints. *J. ACM*, 44(4):527–548, 1997.

[Petke and Jeavons, 2009] Justyna Petke and Peter Jeavons. Tractable benchmarks for constraint programming. Technical Report RR-09-07, OUCL, 2009.

[Roussel and Lecoutre, 2009] Olivier Roussel and Christophe Lecoutre. Xml representation of constraint networks: Format xcsp 2.1. Technical report, CoRR abs/0902.2362, feb 2009.

[Williams *et al.*, 2003] Ryan Williams, Carla P. Gomes, and Bart Selman. Backdoors to typical case complexity. In *Proceedings of the 18th international joint conference on Artificial intelligence*, IJCAI'03, pages 1173–1178, San Francisco, CA, USA, 2003. Morgan Kaufmann Publishers Inc.