Limits of Preprocessing*

Stefan Szeider
Vienna University of Technology, Vienna, Austria

Abstract

We present a first theoretical analysis of the power of polynomial-time preprocessing for important combinatorial problems from various areas in AI. We consider problems from Constraint Satisfaction, Global Constraints, Satisfiability, Nonmonotonic and Bayesian Reasoning. We show that, subject to a complexity theoretic assumption, none of the considered problems can be reduced by polynomial-time preprocessing to a problem kernel whose size is polynomial in a structural problem parameter of the input, such as induced width or backdoor size. Our results provide a firm theoretical boundary for the performance of polynomial-time preprocessing algorithms for the considered problems.

Introduction

Many important computational problems that arise in various areas of AI are intractable. Nevertheless, AI research was very successful in developing and implementing heuristic solvers that work well on real-world instances. An important component of virtually every solver is a powerful polynomial-time preprocessing procedure that reduces the problem input. For instance, preprocessing techniques for the propositional satisfiability problem are based on Boolean Constraint Propagation (see, e.g., Eén and Biere, 2005), CSP solvers make use of various local consistency algorithms that filter the domains of variables (see, e.g., Bessiere, 2006); similar preprocessing methods are used by solvers for Nonmonotonic and Bayesian reasoning problems (see, e.g., Gebser et al., 2008, Bolt and van der Gaag, 2006, respectively).

Until recently, no provable performance guarantees for polynomial-time preprocessing methods have been obtained, and so preprocessing was only subject of empirical studies. A possible reason for the lack of theoretical results is a certain inadequacy of the P vs NP framework for such an analysis: if we could reduce in polynomial time an instance of an NP-hard problem just by one bit, then we can solve the entire problem in polynomial time by repeating the reduction step a polynomial number of times, and P $=$ NP follows.

With the advent of *parameterized complexity* (Downey, Fellows, and Stege 1999), a new theoretical framework became available that provides suitable tools to analyze the power of preprocessing. Parameterized complexity considers a problem in a two-dimensional setting, where in addition to the input size n, a *problem parameter* k is taken into consideration. This parameter can encode a structural aspect of the problem instance. A problem is called *fixed-parameter tractable* (FPT) if it can be solved in time $f(k)p(n)$ where f is a function of the parameter k and p is a polynomial of the input size n. Thus, for FPT problems, the combinatorial explosion can be confined to the parameter and is independent of the input size. It is known that a problem is fixed-parameter tractable if and only if every problem input can be reduced by polynomial-time preprocessing to an equivalent input to the same problem whose size is bounded by a function of the parameter (Downey, Fellows, and Stege 1999). The reduced instance is called the *problem kernel*, the preprocessing is called *kernelization*. The power of polynomial-time preprocessing can now be benchmarked in terms of the size of the kernel. Once a small kernel is obtained, we can apply any method of choice to solve the kernel: brute-force search, heuristics, approximation, etc. (Guo and Niedermeier 2007). Because of this flexibility a small kernel is generally preferable to a less flexible branching-based fixed-parameter algorithm. Thus, small kernels provide an additional value that goes beyond bare fixed-parameter tractability.

In general the size of the kernel is exponential in the parameter, but many important NP-hard optimization problems such as Minimum Vertex Cover, parameterized by solution size, admit *polynomial kernels*, see, e.g., (Bodlaender et al. 2009) for references.

In previous research several NP-hard AI problems have been shown to be fixed-parameter tractable. We list some important examples from various areas:

- Constraint satisfaction problems (CSP) over a fixed universe of values, parameterized by the induced width (Dechter and Pearl 1989; Gottlob, Scarcello, and Sideri 2002).
- Consistency and generalized arc consistency for intractable global constraints, parameterized by the cardinalities of certain sets of values (Bessière et al. 2008).
- Propositional satisfiability (SAT), parameterized by the size of backdoors (Nishimura, Ragde, and Szeider 2004).
- Positive inference in Bayesian networks with variables of bounded domain size, parameterized by size of loop cutsets (Pearl 1988; Bidyuk and Dechter 2007).
- Nonmonotonic reasoning with normal logic programs, parameterized by feedback width (Gottlob, Scarcello, and Sideri 2002).

*Research funded by the ERC (COMPLEX REASON, Grand Reference 239962).
Copyright © 2011, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

However, only exponential kernels are known for these fundamental AI problems. Can we hope for polynomial kernels?

Results Our results are throughout negative. We provide strong theoretical evidence that none of the above fixed-parameter tractable AI problems admits a polynomial kernel. More specifically, we show that a polynomial kernel for any of these problems causes a collapse of the Polynomial Hierarchy to its third level, which is considered highly unlikely by complexity theorists.

Our results are general: The kernel lower bounds are not limited to a particular preprocessing technique but apply to any clever technique that could be conceived in future research. Hence the results contribute to the foundations of AI.

Our results suggest the investigation of alternative approaches to polynomial-time preprocessing; for instance, preprocessing that produces in polynomial time a Boolean combination of polynomially sized kernels instead of one single kernel.

Formal Background

A *parameterized problem* $\mathbf{P}$ is a subset of $\Sigma^* \times \mathbb{N}$ for some finite alphabet Σ. For a problem instance $(x, k) \in \Sigma^* \times \mathbb{N}$ we call x the main part and k the parameter. We assume the parameter is represented in unary. For the parameterized problems considered in this paper, the parameter is a function of the main part, i.e., $k = \pi(x)$ for a function π. We then denote the problem as $\mathbf{P}(\pi)$, e.g., U-**CSP**(width) denotes the problem U-**CSP** parameterized by the width of the given tree decomposition.

A parameterized problem $\mathbf{P}$ is *fixed-parameter tractable* if there exists an algorithm that solves any input $(x, k) \in \Sigma^* \times \mathbb{N}$ in time $O(f(k) \cdot p(|x|)$ where f is an arbitrary computable function of k and p is a polynomial in n.

A *kernelization* for a parameterized problem $\mathbf{P} \subseteq \Sigma^* \times \mathbb{N}$ is an algorithm that, given $(x, k) \in \Sigma^* \times \mathbb{N}$, outputs in time polynomial in $|x| + k$ a pair $(x', k') \in \Sigma^* \times \mathbb{N}$ such that (i) $(x, k) \in \mathbf{P}$ if and only if $(x', k') \in \mathbf{P}$ and (ii) $|x'| + k' \leq g(k)$, where g is an arbitrary computable function, called the *size* of the kernel. In particular, for constant k the kernel has constant size $g(k)$. If g is a polynomial then we say that $\mathbf{P}$ admits a *polynomial kernel*.

Every fixed-parameter tractable problem admits a kernel. This can be seen by the following argument due to Downey, Fellows, and Stege (1999). Assume we can decide instances (x, k) of problem $\mathbf{P}$ in time $f(k)|n|^{O(1)}$. We kernelize an instance (x, k) as follows. If $|x| \leq f(k)$ then we already have a kernel of size $f(k)$. Otherwise, if $|x| > f(k)$, then $f(k)|x|^{O(1)} \leq |x|^{O(1)}$ is a polynomial; hence we can decide the instance in polynomial time and replace it with a small decision-equivalent instance (x', k'). Thus we always have a kernel of size at most $f(k)$. However, $f(k)$ is superpolynomial for NP-hard problems (unless P $=$ NP), hence this generic construction is not providing polynomial kernels.

We understand *preprocessing* for an NP-hard problem as a *polynomial-time* procedure that transforms an instance of the problem to a (possible smaller) solution-equivalent instance of the same problem. Kernelization is such a preprocessing with a *performance guarantee*, i.e., we are guaranteed that the preprocessing yields a kernel whose size is bounded in terms of the parameter of the given problem instance. In the literature also different forms of preprocessing have been considered. An important one is *knowledge compilation*, a two-phases approach to reasoning problems where in a first phase a given knowledge base is (possibly in exponential time) preprocessed ("compiled"), such that in a second phase various queries can be answered in polynomial time (Cadoli et al. 2002).

Tools for Kernel Lower Bounds

In the sequel we will use recently developed tools to obtain kernel lower bounds. Our kernel lower bounds are subject to the widely believed complexity theoretic assumption NP $\not\subseteq$ co-NP/poly (or equivalently, PH $\neq \Sigma_p^3$). In other words, the tools allow us to show that a parameterized problem does not admit a polynomial kernel unless the Polynomial Hierarchy collapses to its third level (see, e.g., Papadimitriou, 1994).

A *composition algorithm* for a parameterized problem $\mathbf{P} \subseteq \Sigma^* \times \mathbb{N}$ is an algorithm that receives as input a sequence $(x_1, k), \ldots, (x_t, k) \in \Sigma^* \times \mathbb{N}$, uses time polynomial in $\sum_{i=1}^{t} |x_i| + k$, and outputs $(y, k') \in \Sigma^* \times \mathbb{N}$ with (i) $(y, k') \in \mathbf{P}$ if and only if $(x_i, k) \in \mathbf{P}$ for some $1 \leq i \leq t$, and (ii) k' is polynomial in k. A parameterized problem is *compositional* if it has a composition algorithm. With each parameterized problem $\mathbf{P} \subseteq \Sigma^* \times \mathbb{N}$ we associate a classical problem UP[$\mathbf{P}$] $= \{\, x\#1^k : (x, k) \in P \,\}$ where 1 denotes an arbitrary symbol from Σ and $\#$ is a new symbol not in Σ. We call UP[$\mathbf{P}$] the *unparameterized version* of $\mathbf{P}$.

The following result is the basis for our kernel lower bounds.

Theorem 1 (Bodlaender et al., 2009, Fortnow and Santhanam, 2008). *Let* $\mathbf{P}$ *be a parameterized problem whose unparameterized version is* NP*-complete. If* $\mathbf{P}$ *is compositional, then it does not admit a polynomial kernel unless* NP $\subseteq$ co-NP*/poly, i.e., the Polynomial Hierarchy collapses.*

Let $\mathbf{P}, \mathbf{Q} \subseteq \Sigma^* \times \mathbb{N}$ be parameterized problems. We say that $\mathbf{P}$ is *polynomial parameter reducible* to $\mathbf{Q}$ if there exists a polynomial time computable function $K : \Sigma^* \times \mathbb{N} \to \Sigma^* \times \mathbb{N}$ and a polynomial p, such that for all $(x, k) \in \Sigma^* \times \mathbb{N}$ we have (i) $(x, k) \in \mathbf{P}$ if and only if $K(x, k) = (x', k') \in \mathbf{Q}$, and (ii) $k' \leq p(k)$. The function K is called a *polynomial parameter transformation*.

The following theorem allows us to transform kernel lower bounds from one problem to another.

Theorem 2 (Bodlaender, Thomassé, and Yeo, 2009). *Let* $\mathbf{P}$ *and* $\mathbf{Q}$ *be parameterized problems such that* UP[$\mathbf{P}$] *is* NP*-complete,* UP[$\mathbf{Q}$] *is in* NP*, and there is a polynomial parameter transformation from* $\mathbf{P}$ *to* $\mathbf{Q}$*. If* $\mathbf{Q}$ *has a polynomial kernel, then* $\mathbf{P}$ *has a polynomial kernel.*

Constraint Networks

Constraint networks have proven successful in modeling everyday cognitive tasks such as vision, language comprehension, default reasoning, and abduction, as well as in applications such as scheduling, design, diagnosis, and temporal and spatial reasoning (Dechter 2010). A *constraint network* is a triple $I = (V, U, \mathcal{C})$ where V is a finite set of variables, U is a finite universe of values, and $\mathcal{C} = \{C_1, \ldots, C_m\}$ is set of constraints. Each constraint C_i is a pair (S_i, R_i) where S_i is a list of variables of length r_i called the *constraint scope*, and R_i is an r_i-ary relation over U, called the *constraint relation*. The tuples of R_i indicate the allowed combinations of simultaneous values for the variables S_i. A *solution* is a mapping $\tau : V \to U$ such that for each $1 \leq i \leq m$ and $S_i = (x_1, \ldots, x_{r_i})$, we have $(\tau(x_1), \ldots, \tau(x_{r_i})) \in R_i$. A constraint network is *satisfiable* if it has a solution.

With a constraint network $I = (V, U, \mathcal{C})$ we associate its *constraint graph* $G = (V, E)$ where E contains an edge between two variables if and only if they occur together in the scope of a constraint. A *width w tree decomposition* of a graph G is a pair (T, λ) where T is a tree and λ is a labeling of the nodes of T with sets of vertices of G such that the following properties are satisfied: (i) every vertex of G belongs to $\lambda(p)$ for some node p of T; (ii) every edge of G is is contained in $\lambda(p)$ for some node p of T; (iii) For each vertex v of G the set of all tree nodes p with $v \in \lambda(p)$ induces a connected subtree of T; (iv) $|\lambda(p)| - 1 \leq w$ holds for all tree nodes p. The *treewidth* of G is the smallest w such that G has a width w tree decomposition. The *induced width* of a constraint network is the treewidth of its constraint graph (Dechter and Pearl 1989). We note in passing that the problem of finding a tree decomposition of width w is NP-hard but fixed-parameter tractable in w.

Let U be a fixed universe containing at least two elements. We consider the following parameterized version of the constraint satisfaction problem (CSP).

U-**CSP**(width): the instance is a constraint network $I = (V, U, \mathcal{C})$ and a width w tree decomposition of the constraint graph of I. w is the parameter. The question is whether is I satisfiable.

It is well known that U-**CSP**(width) is fixed-parameter tractable over any fixed universe U (Dechter and Pearl 1989; Gottlob, Scarcello, and Sideri 2002), for generalizations see (Samer and Szeider 2010). We contrast this classical result and show that it is unlikely that U-**CSP**(width) admits a polynomial kernel, even in the simplest case where $U = \{0, 1\}$.

Theorem 3. $\{0,1\}$-**CSP**(width) *does not admit a polynomial kernel unless the Polynomial Hierarchy collapses.*

Proof. We show that $\{0,1\}$-**CSP**(width) is compositional. Let (I_i, T_i), $1 \leq i \leq t$, be a given sequence of instances of $\{0,1\}$-**CSP**(width) where $I_i = (V_i, U_i, \mathcal{C}_i)$ is a constraint network and T_i is a width w tree decomposition of the constraint graph of I_i. We may assume, w.l.o.g., that $V_i \cap V_j = \emptyset$ for $1 \leq i < j \leq t$ (otherwise we can simply change the names of variables). We form a new constraint network $I = (V, \{0,1\}, \mathcal{C})$ as follows. We put $V = \bigcup_{i=1}^{t} V_i \cup \{a_1, \ldots, a_t, b_0, \ldots, b_t\}$ where a_i, b_i are new variables. We define the set $\mathcal{C}$ of constraints in three groups.

(1) For each $1 \leq i \leq t$ and each constraint $C = ((x_1, \ldots, x_r), R) \in \mathcal{C}_i$ we add to $\mathcal{C}$ a new constraint $C' = ((x_1, \ldots, x_r, a_i), R'))$ where $R' = \{\, (u_1, \ldots, u_r, 0) : (u_1, \ldots, u_r) \in R \,\} \cup \{(1, \ldots, 1)\}$.

(2) We add t ternary constraints $C_1^*, \ldots, C_t^*$ where $C_i^* = ((b_{i-1}, b_i, a_i), R^*)$ and $R^* = \{(0,0,1), (0,1,0), (1,1,1)\}$.

(3) Finally, we add two unary constraints $C^0 = ((b_0), (0))$ and $C^1 = ((b_t), (1))$ which force the values of b_0 and b_t to 0 and 1, respectively.

Let G, G_i be the constraint graphs of I and I_i, respectively. We observe that $a_1, \ldots, a_t$ are cut vertices of G. Removing these vertices separates G into independent parts $P, G_1', \ldots, G_t'$ where P is the path $b_0, b_1, \ldots, b_t$, and G_i' is isomorphic to G_i. By standard techniques (see, e.g., Kloks, 1994), we can put the given width w tree decompositions $T_1, \ldots, T_t$ of $G_1', \ldots, G_t'$ and the trivial width 1 tree decomposition of P together to a width $w+1$ tree decomposition T of G. Clearly (I, T) can be obtained from (I_i, T_i), $1 \leq i \leq t$, in polynomial time.

It is not difficult to see that I is satisfiable if and only if at least one of the I_i is satisfiable.

In order to apply Theorem 1, it remains to observe that UP[$\{0,1\}$-**CSP**(width)] is NP-complete. ☐

Satisfiability

The *propositional satisfiability problem* (SAT) was the first problem shown to be NP-hard (Cook 1971). Despite its hardness, SAT solvers are increasingly leaving their mark as a general-purpose tool in areas as diverse as software and hardware verification, automatic test pattern generation, planning, scheduling, and even challenging problems from algebra (Gomes et al. 2008). SAT solvers are capable of exploiting the hidden structure present in real-world problem instances. The concept of *backdoors*, introduced by Williams, Gomes, and Selman (2003) provides a means for making the vague notion of a hidden structure explicit. Backdoors are defined with respect to a "sub-solver" which is a polynomial-time algorithm that correctly decides the satisfiability for a class $\mathcal{C}$ of CNF formulas. More specifically, Gomes et al. (2008) define a *sub-solver* to be an algorithm $\mathcal{A}$ that takes as input a CNF formula F and has the following properties: (i) *Trichotomy*: $\mathcal{A}$ either rejects the input F, or determines F correctly as unsatisfiable or satisfiable; (ii) *Efficiency*: $\mathcal{A}$ runs in polynomial time; (iii) *Trivial Solvability*: $\mathcal{A}$ can determine if F is trivially satisfiable (has no clauses) or trivially unsatisfiable (contains the empty clause); (iv.) *Self-Reducibility*: if $\mathcal{A}$ determines F, then for any variable x and value $\varepsilon \in \{0,1\}$, $\mathcal{A}$ determines $F[x = \varepsilon]$. $F[\tau]$ denotes the formula obtained from F by applying the partial assignment τ, i.e., satisfied clauses are removed and false literals are removed from the remaining clauses.

We identify a sub-solver $\mathcal{A}$ with the class $\mathcal{C}_{\mathcal{A}}$ of CNF formulas whose satisfiability can be determined by $\mathcal{A}$. A *strong $\mathcal{A}$-backdoor set* (or *$\mathcal{A}$-backdoor*, for short) of a CNF formula F is a set B of variables such that for each possible truth assignment τ to the variables in B, the satisfiability of $F[\tau]$ can be determined by sub-solver $\mathcal{A}$ in time $O(n^c)$. Hence, if we know an $\mathcal{A}$-backdoor of size k, we can decide the satisfiability of F by running $\mathcal{A}$ on 2^k instances $F[\tau]$, yielding a time bound of $O(2^k n^c)$. Hence SAT decision is fixed-parameter tractable in the backdoor size k for any sub-solver $\mathcal{A}$. Hence the following problem is clearly fixed-parameter tractable for any sub-solver $\mathcal{A}$.

SAT($\mathcal{A}$-backdoor): the instance is a CNF formula F, and an $\mathcal{A}$-backdoor B of F of size k. The parameter is k, the question is whether F is satisfiable.

We are concerned with the question of whether instead of trying all 2^k possible partial assignments we can reduce the instance to a polynomial kernel. We will establish a very general result that applies to all possible sub-solvers.

Theorem 4. **SAT**($\mathcal{A}$-backdoor) *does not admit a polynomial kernel for any sub-solver $\mathcal{A}$ unless the Polynomial Hierarchy collapses.*

Proof. We devise polynomial parameter transformations from the following parameterized problem which is known to be compositional (Fortnow and Santhanam 2008) and therefore unlikely to admit a polynomial kernel.

SAT(vars): the instance is a CNF formula F on n variables. The parameter is n, the question is whether F is satisfiable.

Let F be a CNF formula and V the set of all variables of F. Due to property (ii) of a sub-solver, V is an $\mathcal{A}$-backdoor set for any $\mathcal{A}$. Hence, by mapping (F, n) (as an instance of **SAT**(vars)) to (F, V) (as an instance of **SAT**($\mathcal{A}$-backdoor)) provides a (trivial) polynomial parameter transformation from **SAT**(vars) to **SAT**($\mathcal{A}$-backdoor). Since the unparameterized versions of both problems are clearly NP-complete, the result follows by Theorem 2. ☐

Let **3SAT**(π) (where π is an arbitrary parameterization) denote the problem **SAT**(π) restricted to 3CNF formula, i.e., to CNF formulas where each clause contains at most three literals. In contrast to **SAT**(vars), the parameterized problem **3SAT**(vars) has a trivial polynomial kernel: if we remove duplicate clauses, then any 3CNF formula on n variables contains at most $O(n^3)$ clauses, and so is a polynomial kernel. Hence the easy proof of Theorem 4 does not carry over to **3SAT**($\mathcal{A}$-backdoor). We therefore consider the cases **3SAT**(HORN-backdoor) and **3SAT**(2CNF-backdoor) separately, these cases are important since the detection of HORN and 2CNF-backdoors is fixed-parameter tractable (Nishimura, Ragde, and Szeider 2004).

Theorem 5. *Neither* **3SAT**(HORN-backdoor) *nor* **3SAT**(2CNF-backdoor) *admits a polynomial kernel unless the Polynomial Hierarchy collapses.*

Proof. (Sketch; for more details see http://arxiv.org/abs.1104.5566.) Let $\mathcal{C} \in \{\text{HORN}, \text{2CNF}\}$. We show that **3SAT**($\mathcal{C}$-backdoor) is compositional. Let (F_i, B_i), $1 \leq i \leq t$, be a given sequence of instances of **3SAT**($\mathcal{C}$-backdoor) where F_i is a 3CNF formula and B_i is a $\mathcal{C}$-backdoor set of F_i of size k.

If $t > 2^k$ then we can determine whether some F_i is satisfiable in time $O(t2^k n) \leq O(t^2 n)$ which is polynomial in $t + n$. If the answer is yes, then we output (F_i, B_i), otherwise we output (F_1, B_1). It remains to consider the case where $t \leq 2^k$. For simplicity, assume $t = 2^k$. Let V_i denote the set of variables of F_i. We may assume, w.l.o.g., that $B_1 = \cdots = B_t$ and that $V_i \cap V_j = B_1$ for all $1 \leq i < j \leq t$ since otherwise we can change names of variable accordingly. We take a set $Y = \{y_1, \dots, y_s\}$ of new variables. Let $\tau_1, \dots, \tau_t$ denote all possible truth assignments to Y. From each F_i we construct a formula F'_i such that $F'_i[\tau_i]$ and F_i are decision-equivalent, and $F'_i[\tau_j]$ is trivially satisfiable for $j \neq i$. This can be done such that (i) F is satisfiable if and only if at least one of the formulas F_i is satisfiable and (ii) $B = Y \cup B_1$ is a $\mathcal{C}$-backdoor of F. Hence we have a composition algorithm for **3SAT**($\mathcal{C}$-backdoor). Since UP[**3SAT**($\mathcal{C}$-backdoor)] is clearly NP-complete, the result follows from Theorem 1. ☐

Global Constraints

The success of today's constraint solvers relies heavily on efficient algorithms for special purpose *global constraints* (van Hoeve and Katriel 2006). A global constraint specifies a pattern that frequently occurs in real-world problems, for instance, it is often required that variables must all take different values (e.g., activities requiring the same resource must all be assigned different times). The ALLDIFFERENT global constraint efficiently encodes this requirement.

More formally, a global constraint is defined for a set S of variables, each variable $x \in S$ ranges over a finite domain $dom(x)$ of values. An *instantiation* is an assignment α such that $\alpha(x) \in dom(x)$ for each $x \in S$. A global constraint defines which instantiations are legal and which are not. A global constraint is *consistent* if it has at least one legal instantiation, and it is *domain consistent* (or hyper arc consistent) if for each variable $x \in S$ and each value $d \in dom(x)$ there is a legal instantiation α with $\alpha(x) = d$. For all types of global constraints considered in this paper, domain consistency can be reduced to a quadratic number of consistency checks, hence we will focus on consistency. We assume that the size of a representation of a global constraint is polynomial in $\sum_{x \in S} |dom(x)|$.

For several important types $\mathcal{T}$ of global constraints, the problem of deciding whether a constraint of type $\mathcal{T}$ is consistent (in symbols $\mathcal{T}$-**Cons**) is NP-hard. Examples for such intractable global constraints are NVALUE, DISJOINT, and USES (Bessière et al. 2004). An NVALUE constraint over a set X of variables requires from a legal instantiation α that $|\{\,\alpha(x) : x \in X\,\}| = N$; ALLDIFFERENT is the special case where $N = |X|$. The global constraints DISJOINT and USES are specified by two sets of variables X, Y; DISJOINT requires that $\alpha(x) \neq \alpha(y)$ for each pair $x \in X$ and $y \in Y$; USES requires that for each $x \in X$ there is some $y \in Y$ such that $\alpha(x) = \alpha(y)$. For a set X of variables we write $dom(X) = \bigcup_{x \in X} dom(x)$.

Bessière et al. (2008) considered $dx = |dom(X)|$ as parameter for NVALUE, $dxy = |dom(X) \cap dom(Y)|$ as parameter for DISJOINT, and $dy = |dom(Y)|$ as parameter for USES. They showed that consistency checking is fixed-parameter tractable for the constraints under the respective parameterizations, i.e., the problems NVALUE-**CONS**(dx), DISJOINT-**CONS**(dxy), and USES-**CONS**(dy) are fixed-parameter tractable. We show that it is unlikely that their results can be improved in terms of polynomial kernels.

Theorem 6. *The problems* NVALUE-**CONS**(dx), DISJOINT-**CONS**(dxy), USES-**CONS**(dy) *do not admit polynomial kernels unless the Polynomial Hierarchy collapses.*

Proof. We devise a polynomial parameter reduction from **SAT**(vars) using a construction of Bessière et al. (2004). Let $F = \{C_1, \ldots, C_m\}$ be a CNF formula over variables $x_1, \ldots, x_n$. We consider the clauses and variables of F as the variables of a global constraint with domains $dom(x_i) = \{-i, i\}$, and $dom(C_j) = \{\, i : x_i \in C_j \,\} \cup \{\, -i : \neg x_i \in C_j \,\}$. Now F can be encoded as an NVALUE constraint with $X = \{x_1, \ldots, x_n, C_1, \ldots, C_m\}$ and $N = n$ (clearly F is satisfiable if and only if the constraint is consistent). Since $dx = 2n$ we have a polynomial parameter reduction from **SAT**(vars) to NVALUE-**CONS**(dx). Similarly, as observed by Bessière et al. (2009), F can be encoded as a DISJOINT constraint with $X = \{x_1, \ldots, x_n\}$ and $Y = \{C_1, \ldots, C_m\}$ ($dxy \leq 2n$), or as a USES constraint with $X = \{C_1, \ldots, C_m\}$ and $Y = \{x_1, \ldots, x_n\}$ ($dy = 2n$). Since the unparameterized problems are clearly NP-complete, the result follows by Theorem 2. □

Further results on kernels for global constraints have been obtained by Gaspers and Szeider (2011).

Bayesian Reasoning

Bayesian networks (BNs) have emerged as a general representation scheme for uncertain knowledge (Pearl 2010). A BN models a set of stochastic variables, the independencies among these variables, and a joint probability distribution over these variables. For simplicity we consider the important special case where the stochastic variables are Boolean. The variables and independencies are modelled in the BN by a directed acyclic graph $G = (V, A)$, the joint probability distribution is given by a table T_v for each node $v \in V$ which defines a probability $T_{v|U}$ for each possible instantiation $U = (d_1, \ldots, d_s) \in \{\mathrm{true}, \mathrm{false}\}^s$ of the parents $v_1, \ldots, v_s$ of v in G. The probability Pr(U) of a complete instantiation U of the variables of G is given by the product of $T_{v|U}$ over all variables v. We consider the problem **Positive-BN-Inference** which takes as input a Boolean BN (G, T) and a variable v, and asks whether Pr($v = \mathrm{true}) > 0$. The problem is NP-complete (Cooper 1990). The problem can be solved in polynomial time if the BN is *singly connected*, i.e, if there is at most one undirected path between any two variables (Pearl 1988). It is natural to parametrize the problem by the number of variables one must delete in order to make the BN singly connected (the deleted variables form a *loop cutset*). In fact, **POSITIVE-BN-INFERENCE**(loop cutset size) is easily seen to be fixed-parameter tractable as we can determine whether Pr($v = \mathrm{true}) > 0$ by taking the maximum of Pr($v = \mathrm{true} \mid U$) over all 2^k possible instantiations of the k cutset variables, each of which requires processing of a singly connected network. However, although fixed-parameter tractable, it is unlikely that the problem admits a polynomial kernel.

Theorem 7. **POSITIVE-BN-INFERENCE**(loop cutset size) *does not admit a polynomial kernel unless the Polynomial Hierarchy collapses.*

Proof. (Sketch.) We give a polynomial parameter transformation from **SAT**(vars) and apply Theorem 2. The reduction is based on the reduction from 3SAT given by Cooper (1990). However, we need to allow clauses with an arbitrary number of literals since, as observed above, **3SAT**(vars) has a polynomial kernel. Let F be a CNF formula on n variables. We construct a BN (G, T) such that for a variable v we have Pr($v = \mathrm{true}) > 0$ if and only if F is satisfiable. Cooper uses *input nodes* u_i for representing variables of F, *clause nodes* c_i for representing the clauses of F, and *conjunction nodes* d_i for representing the conjunction of the clauses. We proceed similarly, however, we cannot represent a clause of large size with a single clause node c_i, as the required table T_{c_i} would be of exponential size. Therefore we split clauses containing more than 3 literals into several clause nodes. For instance, a clause node c_1 with parents u_1, u_2, u_3 is split into clause nodes c_1, c_2 where c_1 has parents u_1, u_2 and c_2 has parents c_1, u_3. It remains to observe that the set of input nodes $E = \{u_1, \ldots, u_n\}$ is a loop cutset of the constructed BN, hence we have indeed a polynomial parameter transformation from **SAT**(vars) to **POSITIVE-BN-INFERENCE**(loop cutset size). The result follows by Theorem 2. □

Nonmonotonic Reasoning

Logic programming with negation under the stable model semantics is a well-studied form of nonmonotonic reasoning (Gelfond and Lifschitz 1988; Marek and Truszczyński 1999). A (normal) *logic program* P is a finite set of rules r of the form $(h \leftarrow a_1 \wedge \cdots \wedge a_m \wedge \neg b_1 \wedge \cdots \wedge \neg b_n)$ where h, a_i, b_i are *atoms*, where h forms the head and the a_i, b_i from the body of r. We write $H(r) = h$, $B^+(r) = \{a_1, \ldots, a_m\}$, and $B^-(r) = \{b_1, \ldots, b_n\}$. Let I be a finite set of atoms. The *GF reduct* P^I of a logic program P under I is the program obtained from P by removing all rules r with $B^-(r) \cap I \neq \emptyset$, and removing from the body of each remaining rule r' all literals $\neg b$ with $b \in I$. I is a *stable model* of P if I is a minimal model of P^I, i.e., if (i) for each rule $r \in P^I$ with $B^+(r) \subseteq I$ we have $H(r) \in I$, and (ii) there is no proper subset of I with this property. The *undirected dependency graph* $U(P)$ of P is formed as follows. We take the atoms of P as vertices and add an edge $x - y$ between two atoms x, y if there is a rule $r \in P$ with $H(r) = x$ and $y \in B^+(r)$, and we add a path $x - u - y$ if $H(r) = x$ and $y \in B^-(r)$ (u is a new vertex of degree 2). The *feedback width* of P is the size of a smallest set V of atoms such that every cycle of $U(P)$ runs through an atom in V.

A fundamental computational problems is **Stable Model Existence (SME)**, which asks whether a given normal logic program has a stable model. The problem is well-known to be NP-complete (Marek and Truszczyński 1991). Gottlob, Scarcello, and Sideri (2002) showed that **SME**(feedback width) is fixed-parameter tractable, for generalizations see (Fichte and Szeider 2011). We show that this result cannot be strengthened with respect to a polynomial kernel.

Theorem 8. **SME**(feedback width) *does not admit a polynomial kernel unless the Polynomial Hierarchy collapses.*

Proof. (Sketch.) Niemelä (1999) describes a polynomial-time transformation that maps a CNF formula F to a logic program P such that F is satisfiable if and only if P

has a stable model. From the details of the construction it is easy to observe that the feedback width of P is at most twice the number of variables in F, hence we have a polynomial parameter transformation from **SAT**(vars) to **SME**(feedback width). The result follows by Theorem 2. $\square$

Conclusion

We have established super-polynomial kernel lower bounds for a wide range of important AI problems, providing firm limitations for the power of polynomial-time preprocessing for these problems. We conclude from these results that in contrast to many optimization problems (see Section 1), typical AI problems do not admit polynomial kernels. Our results suggest the consideration of alternative approaches. For example, it might still be possible that some of the considered problems admit polynomially sized *Turing kernels*, i.e., a polynomial-time preprocessing to a Boolean combination of a polynomial number of polynomial kernels. In the area of optimization, parameterized problems are known that do not admit polynomial kernels but admit polynomial Turing kernels (Fernau et al. 2009). This suggests a theoretical and empirical study of Turing kernels for the AI problems considered.

References

Bessière, C.; Hebrard, E.; Hnich, B.; and Walsh, T. 2004. The complexity of global constraints. *AAAI'04*.

Bessière, C.; Hebrard, E.; Hnich, B.; Kiziltan, Z.; Quimper, C.-G.; and Walsh, T. 2008. The parameterized complexity of global constraints. *AAAI'08*.

Bessière, C.; Hebrard, E.; Hnich, B.; Kiziltan, Z.; and Walsh, T. 2009. Range and roots: Two common patterns for specifying and propagating counting and occurrence constraints. *AIJ* 173(11):1054–1078.

Bessière, C. 2006. Constraint propagation. *Handbook of Constraint Programming*.

Bidyuk, B.; and Dechter, R. 2007. Cutset sampling for Bayesian networks. *JAIR* 28:1–48.

Bodlaender, H. L.; Downey, R. G.; Fellows, M. R.; and Hermelin, D. 2009. On problems without polynomial kernels. *JCSS* 75(8):423–434.

Bodlaender, H. L.; Thomassé, S.; and Yeo, A. 2009. Kernel bounds for disjoint cycles and disjoint paths. *ESA'09*.

Bolt, J.; and van der Gaag, L. 2006. Preprocessing the MAP problem. In *PGM'06*.

Cadoli, M.; Donini, F. M.; Liberatore, P.; and Schaerf, M. 2002. Preprocessing of intractable problems. *Inf. and Comp.* 176(2):89–120.

Cook, S. A. 1971. The complexity of theorem-proving procedures. *STOC'71*.

Cooper, G. F. 1990. The computational complexity of probabilistic inference using Bayesian belief networks. *AIJ* 42(2-3):393–405.

Dechter, R.; and Pearl, J. 1989. Tree clustering for constraint networks. *AIJ* 38(3):353–366.

Dechter, R. 2010. Constraint satisfaction. *The CogNet Library: References Collection*.

Downey, R.; Fellows, M. R.; and Stege, U. 1999. Parameterized complexity: A framework for systematically confronting computational intractability. *AMS-DIMACS 49*.

Eén, N.; and Biere, A. 2005. Effective preprocessing in SAT through variable and clause elimination. *SAT'05*, 2005.

Fernau, H.; Fomin, F. V.; Lokshtanov, D.; Raible, D.; Saurabh, S.; and Villanger, Y. 2009. Kernel(s) for problems with no kernel: On out-trees with many leaves. *STACS'09*.

Fichte, J. K.; and Szeider, S. 2011. Backdoors to tractable answer-set programming. *IJCAI'11*.

2008. Fortnow, L.; and Santhanam, R. Infeasibility of instance compression and succinct PCPs for NP. *STOC'08*.

Gaspers, S.; and Szeider, S. 2011. Kernels for global constraints. *IJCAI'11*.

Gebser, M.; Kaufmann, B.; Neumann, A.; and Schaub, T. 2008. Advanced preprocessing for answer set solving. *ECAI'08*.

Gelfond, M.; and Lifschitz, V. 1988. The stable model semantics for logic programming. *LP'88*.

Gomes, C. P.; Kautz, H.; Sabharwal, A.; and Selman, B. 2008. Satisfiability solvers. *Handbook of Knowledge Representation*.

Gottlob, G.; Scarcello, F.; and Sideri, M. 2002. Fixed-parameter complexity in AI and nonmonotonic reasoning. *AIJ* 138(1-2):55–86.

Guo, J.; and Niedermeier, R. 2007. Invitation to data reduction and problem kernelization. *ACM SIGACT News* 38(2):31–45.

Kloks, T. 1994. *Treewidth: Computations and Approximations*. Springer.

Marek, V. W.; and Truszczyński, M. 1991. Autoepistemic logic. *J. ACM* 38(3):588–619.

Marek, V. W.; and Truszczyński, M. 1999. Stable models and an alternative logic programming paradigm. *The Logic Programming Paradigm: a 25-Year Perspective*.

Niemelä, I. 1999. Logic programs with stable model semantics as a constraint programming paradigm. *AMAI* 25(3-4):241–273.

Nishimura, N.; Ragde, P.; and Szeider, S. 2004. Detecting backdoor sets with respect to Horn and binary clauses. *SAT'04*.

Papadimitriou, C. H. 1994. *Computational Complexity*. Addison-Wesley.

Pearl, J. 1988. *Probabilistic reasoning in intelligent systems.* Morgan Kaufmann.

Pearl, J. 2010. Bayesian networks. *The CogNet Library: References Collection*.

van Hoeve, W.-J.; and Katriel, I. 2006. Global constraints. *Handbook of Constraint Programming*.

Samer, M.; and Szeider, S.. 2010. Constraint satisfaction with bounded treewidth revisited. *JCSS*, 76(2):103–114.

Williams, R.; Gomes, C.; and Selman, B. 2003. On the connections between backdoors, restarts, and heavy-tailedness in combinatorial search. *SAT'03*.