Artificial Intelligence 216 (2014) 1–19

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Guarantees and limits of preprocessing in constraint satisfaction and reasoning ☆

Serge Gaspers [a,*], Stefan Szeider [b]

[a] UNSW Australia and NICTA, Sydney, Australia
[b] Institute of Information Systems, Vienna University of Technology, Austria

ARTICLE INFO

Article history:
Received 24 July 2013
Received in revised form 12 June 2014
Accepted 29 June 2014
Available online 11 July 2014

Keywords:
Fixed-parameter tractability
Kernelization
Constraint satisfaction
Reasoning
Computational complexity

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 under structural restrictions. All these problems involve two tasks: (i) identifying the structure in the input as required by the restriction, and (ii) using the identified structure to solve the reasoning task efficiently. We show that for most of the considered problems, task (i) admits a polynomial-time preprocessing to a problem kernel whose size is polynomial in a structural problem parameter of the input, in contrast to task (ii) which does not admit such a reduction to a problem kernel of polynomial size, subject to a complexity theoretic assumption. As a notable exception we show that the consistency problem for the AtMost-NValue constraint admits a polynomial kernel consisting of a quadratic number of variables and domain values. Our results provide a firm worst-case guarantees and theoretical boundaries for the performance of polynomial-time preprocessing algorithms for the considered problems.

1. Introduction

Many important computational problems that arise in various areas of AI are intractable. Nevertheless, AI research has been 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., [27]), CSP solvers make use of various local consistency algorithms that filter the domains of variables (see, e.g., [4]); similar preprocessing methods are used by solvers for Nonmonotonic and Bayesian reasoning problems (see, e.g., [38,13], respectively). The history of preprocessing, like applying reduction rules to simplify truth functions, can be traced back to the 1950s [56]. A natural question in this regard is how to measure the quality of preprocessing rules proposed for a specific problem.

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 could solve the entire problem in polynomial time by repeating the reduction step a polynomial number of times, and P = NP would follow.

☆ Preliminary and shorter versions of this paper appeared in the proceedings of IJCAI 2011 [36] and AAAI 2011 [63].

* Corresponding author.

http://dx.doi.org/10.1016/j.artint.2014.06.006
0004-3702/

With the advent of *parameterized complexity* [24], 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 whose size is bounded by a function of the parameter [25]. 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. [42]. 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.

Kernelization is an important algorithmic technique that has become the subject of a very active field in state-of-the-art combinatorial optimization (see, e.g., the references in [28,42,46,58]). Kernelization can be seen as a *preprocessing with performance guarantee* that reduces a problem instance in polynomial time to an equivalent instance, the *kernel*, whose size is a function of the parameter [28,32,42,46].

Once a kernel is obtained, the time required to solve the instance is a function of the parameter only and therefore independent of the input size. While, in general, the time needed to solve an instance does not necessarily depend on the size of the instance alone, the kernelization view is that it preprocesses the easy parts of an instance, leaving a core instance encoding the hard parts of the problem instance. Naturally one aims at kernels that are as small as possible, in order to guarantee good worst-case running times as a function of the parameter, and the kernel size provides a performance guarantee for the preprocessing. Some NP-hard combinatorial problems such as k-Vertex Cover admit polynomially sized kernels, for others such as k-Path an exponential kernel is the best one can hope for [11].

As an example of a polynomial kernel, consider the k-Vertex Cover problem, which, for a graph $G = (V, E)$ and an integer parameter k, is to decide whether there is a set S of at most k vertices such that each edge from E has at least one endpoint in S. Buss' kernelization algorithm for k-Vertex Cover (see [14]) computes the set U of vertices with degree at least $k+1$ in G. If $|U| > k$, then reject the instance, i.e., output a trivial No-instance (e.g., the graph K_2 consisting of one edge and the parameter 0), since every vertex cover of size at most k contains each vertex from U. Otherwise, if $G \setminus U$ has more than $k(k-|U|)$ edges, then reject the instance, since each vertex from $G \setminus U$ covers at most k edges. Otherwise, output the instance $(G \setminus (U \cup L), k-|U|)$, where L is the set of degree-0 vertices in $G \setminus U$. This instance has $O(k^2)$ vertices and edges. Thus, Buss' kernelization algorithm gives a quadratic kernel for k-Vertex Cover.

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

1. Constraint satisfaction problems (CSP) over a fixed universe of values, parameterized by the induced width [41].
2. Consistency and generalized arc consistency for intractable global constraints, parameterized by the cardinalities of certain sets of values [5].
3. Propositional satisfiability (SAT), parameterized by the size of backdoors [51].
4. Positive inference in Bayesian networks with variables of bounded domain size, parameterized by size of loop cutsets [53,9].
5. Nonmonotonic reasoning with normal logic programs, parameterized by feedback width [41].

All these problems involve the following two tasks.

(i) *Structure Recognition Task*: identify the structure in the input as required by the considered parameter.
(ii) *Reasoning Task*: use the identified structure to solve a reasoning task efficiently.

For most of the considered problems we observe the following pattern: the Structure Recognition Task admits a polynomial kernel, in contrast to the Reasoning Task, which does not admit a polynomial kernel, unless the Polynomial Hierarchy collapses to its third level.

A negative exception to this pattern is the recognition problem for CSPs of small induced width, which most likely does not admit a polynomial kernel.

A positive exception to this pattern is the AtMost-NValue global constraint, for which we obtain a polynomial kernel. As in [5], the parameter is the number of holes in the domains of the variables, measuring how close the domains are to being intervals. More specifically, we present a *linear time* preprocessing algorithm that reduces an AtMost-NValue constraint C with k holes to a consistency-equivalent AtMost-NValue constraint C' of size polynomial in k. In fact, C' has at most $O(k^2)$ variables and $O(k^2)$ domain values. We also give an improved branching algorithm checking the consistency of C'