The Parameterized Complexity of Global Constraints*

C. Bessiere
LIRMM
Montpellier,
France
bessiere@lirmm.fr

E. Hebrard
4C
UCC, Ireland
ehebrard@4c.ucc.ie

B. Hnich
Izmir Uni. of Economics,
Izmir,
Turkey
brahim.hnich@ieu.edu.tr

Z. Kiziltan
CS Department
Uni. of Bologna, Italy
zeynep@cs.unibo.it

C.-G. Quimper
École Polytechnique
de Montréal,
Canada
cquimper@alumni.uwaterloo.ca

T. Walsh
NICTA and UNSW
Sydney, Australia
toby.walsh@nicta.com.au

Abstract

We argue that parameterized complexity is a useful tool with which to study global constraints. In particular, we show that many global constraints which are intractable to propagate completely have natural parameters which make them fixed-parameter tractable and which are easy to compute. This tractability tends either to be the result of a simple dynamic program or of a decomposition which has a strong backdoor of bounded size. This strong backdoor is often a cycle cutset. We also show that parameterized complexity can be used to study other aspects of constraint programming like symmetry breaking. For instance, we prove that value symmetry is fixed-parameter tractable to break in the number of symmetries. Finally, we argue that parameterized complexity can be used to derive results about the approximability of constraint propagation.

Introduction

One of the jewels of constraint programming is the notion of a global constraint (Régin 1994; 1996; Bessiere and Régin 1997; Régin and Rueher 2000; Beldiceanu and Contegean 1994; Frisch et al. 2002). Global constraints specify patterns that occur in many real-world problems, and come with efficient and effective propagation algorithms for pruning the search space. For instance, we often have sets of variables which must take different values (e.g. activities in a scheduling problem requiring the same resource must all be assigned different times). Most constraint solvers therefore provide a global ALLDIFFERENT constraint which is propagated efficiently and effectively (Régin 1994). Unfortunately many common and useful global constraints proposed by researchers in the past have turned out to be intractable to propagate completely (e.g. (Quimper 2003; Bessiere et al. 2004; Bessière et al. 2005; 2006b; Samer and Szeider 2008; Artiouchine, Baptiste, and Durr 2008)). In this paper, we argue that we can understand more about the source of this intractability by using tools from parameterized complexity. The insights gained from this analysis may lead to better search methods as well as new propagation algorithms.

*Brahim Hnich is supported by the Scientific and Technological Research Council of Turkey (TUBITAK) under Grant No. SOBAG-108K027. Toby Walsh is funded by the Australian Government's Department of Broadband, Communications and the Digital Economy and the Australian Research Council.

Formal background

A constraint satisfaction problem (CSP) consists of a set of variables, each with a finite domain of values, and a set of constraints specifying allowed combinations of values for some subset of variables. We presume any constraint can be checked in polynomial time. We use capital letters for variables (e.g. X, Y), and lower case for values (e.g. d and d_i). We consider both finite domain and set variables. A variable X takes one value from a finite domain of possible values $dom(X)$. A set variable S takes a set of values from a domain of possible sets. We view set variables as a vector of 0/1 finite domain variables representing the characteristic function of the set. This representation is equivalent to that of maintaining upper bound ($ub(S)$) and lower bound ($lb(S)$) on the potential and definite elements in the set. However, it permits us to simplify our analysis.

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 local consistency properties like domain or bound consistency. A constraint is *domain consistent* (*DC*) iff when a variable is assigned any of the values in its domain, there exist compatible values in the domains of all the other variables of the constraint. Such values are called a *support*. A CSP is domain consistent iff every constraint is domain consistent. A constraint is *bound consistent* (*BC*) iff when a variable is assigned the minimum (or maximum) value in its domain, there exist compatible values between the minimum and maximum domain value for all the other variables. Such values are called a *bound support*. A CSP is bound consistent iff every constraint is bound consistent. A *global constraint* is one in which the number of variables is not fixed. For instance, the NVALUE$([X_1, \ldots, X_n], N)$ constraint ensures that n variables, X_1 to X_n, take N different values (Pachet and Roy 1999). The ALLDIFFERENT constraint (Régin 1994) is a special case of the NVALUE constraint in which $N = n$.

Parameterized complexity

Recently, Bessiere *et al.* have shown that a number of common global constraints are intractable to propagate (Bessière et al. 2006b). For instance, whilst enforcing bound consistency on the NVALUE constraint is polynomial, domain consistency is NP-hard. We show here that the tools of parameterized complexity can provide a more fine-grained view of such complexity results. These complexity tools help us to identify more precisely what makes a global constraint (in)tractable. The insights gained may guide search – for example, we shall see that they can identify small backdoors on which to branch – as well as suggesting new propagation algorithms.

We introduce the necessary tools from parameterized complexity theory. 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, k is some parameter, c is a constant, and n is the size of the input. For example, vertex cover ("Given a graph with n vertices, is there a subset of vertices of size k or less that cover each edge in the graph") is NP-hard in general, but fixed-parameter tractable with respect to k since it can be solved in $O(1.31951^k k^2 + kn)$ time. Hence, provided k is small, vertex cover can be solved effectively.

Downey *et al.* argue (Downey, Fellows, and Stege 1999) that about half the naturally parameterized NP-hard problems in (Garey and Johnson 1979) are fixed-parameter tractable including 3 out of the 6 basic problems. 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. $W[t]$ is characterized by the maximum number t of unbounded fan-in gates on the input-output path of a Boolean circuit specifying the problem. There is considerable evidence to suggest that $W[1]$-hardness implies parametric intractability. In particular, the halting problem for non-deterministic Turing machines is $W[1]$-complete with respect to the length of the accepting computation.

An example

Parameterized complexity gives us a more fine-grained view of the complexity of propagating global constraints. Consider again the NVALUE constraint. Whilst enforcing domain consistency on the NVALUE constraint is NP-hard in general, it is fixed-parameter tractable to enforce in the total number of values in the domains. In hindsight, it is perhaps obvious that propagating NVALUE is easy when the number of values is fixed. However, many different propagation algorithms for NVALUE have been proposed, and none of them prune all possible values and are polynomial when the number of values is fixed.

Theorem 1 *Enforcing domain consistency on* NVALUE$([X_1, \ldots, X_n], N)$ *is fixed-parameter tractable in* $k = |\bigcup_{i \in 1..n} dom(X_i)|$.

Proof: We give an automaton for accepting solutions to this constraint that scans through X_1 to X_n and then N. The states of this automaton are all the possible sets of values that can be used by the X_i, plus one accepting state F. As there are k possible values in the domains of the X_i, there are $O(2^k)$ states. The transition on seeing X_i from state q goes to state $q \cup \{X_i\}$. Finally, we only accept a transition on seeing N from state q if $|q| = N$. Quimper and Walsh have shown that we can efficiently enforce domain consistency wrt the assignments accepted by such an automaton in $O(2^k nd)$ time where $d = \max(\{|dom(X_i)|\}_{i \in 1..n})$ using a simple decomposition (Quimper and Walsh 2007). □

Thus, if the total number of values in the domains is small, NVALUE is tractable to propagate completely since propagation takes polynomial time in the number of variables. For such complexity results to be useful, the identified parameter needs to be "natural" and potentially small. It helps also if the parameter is easy to compute. We can then build a propagator which only decides to propagate the constraint completely when it is cheap to do so. As we shall see, several global constraints are fixed-parameter tractable with respect to the total number of values in the domains. This parameter is easy to compute, and is often small in practice. For instance, many radio frequency link assignment problems only require half a dozen values. We shall also see that dynamic programming is often a means (as here) to show that a global constraint is fixed-parameter tractable.

Other parameters can help identify features that make a global constraint intractable. For instance, domain consistency on the NVALUE constraint is intractable to enforce when the fixed parameter is the maximum number of values that can be used by the X_i.

Theorem 2 *Enforcing domain consistency on* NVALUE$([X_1, \ldots, X_n], N)$ *is* $W[2]$*-hard in* $k = \max(dom(N))$.

Proof: Hitting set is $W[2]$-complete in the size of the hitting set. Given a collection of sets $\{S_1, \ldots, S_n\}$ and an integer k, consider the set of variables $\{X_1, \ldots, X_n, N\}$ such that $dom(N) = \{0..k\}$ and $\forall i \in 1..n,\ dom(X_i) = S_i$. It is easy to see that a solution of NVALUE$([X_1, \ldots, X_n], N)$ corresponds to a hitting set of cardinality k or less. Thus, we can reduce hitting set to NVALUE immediately. □

Hence, we see that the complexity of propagating the NVALUE constraint comes from the potentially large number of values in the domains of the X_i.

Backdoors

As we shall see in the next section, dynamic programming is a frequent way to obtain fixed-parameter tractability results for constraint propagators. Another method is to identify a decomposition of the global constraint in which there is a strong backdoor containing a bounded number of variables. This backdoor is often a cycle cutset into an acyclic (and thus polynomial) subproblem. A *strong backdoor* is a subset of variables which give a polynomial subproblem however they are instantiated (Williams, Gomes, and Selman 2003). A *cycle cutset* is a subset of variables which break all cycles in the problem (Dechter and Pearl 1987). Once the cycle cutset is instantiated, the problem can be solved in a backtrack free manner using a domain consistency algorithm.

Consider the global constraint, DISJOINT$([X_1, \ldots, X_n], [Y_1, \ldots, Y_m])$ (Beldiceanu 2000).

This ensures that $X_i \neq Y_j$ for any i and j. Such a global constraint is useful in scheduling and time-tabling problems. Enforcing domain consistency on DISJOINT is NP-hard (Bessiere et al. 2006a). However, it is fixed-parameter tractable in the total number of values in the domains. To show this, we give a simple decomposition which has a strong backdoor of bounded size that is a cycle cutset. In fact, we give a slightly stronger result. Enforcing domain consistency on DISJOINT is fixed-parameter tractable in the size of the intersection of the domains of X_i and Y_j. Clearly if the total number of values is bounded then the intersection is too.

Theorem 3 *Enforcing domain consistency on* DISJOINT$([X_1, \ldots, X_n], [Y_1, \ldots, Y_m])$ *is fixed-parameter tractable in* $k = |\bigcup_{i \in 1..n} dom(X_i) \cap \bigcup_{j \in 1..m} dom(Y_j)|$.

Proof: Without loss of generality, we assume $n \geq m$. Consider a set variable S and the decomposition: $X_i \in S$ and $Y_j \notin S$. Recall that a set variable can be viewed as a vector of 0/1 variables representing the characteristic function. Let $S_v = 1$ iff $v \in S$. Then, the set $\{S_v \mid v \in \bigcup_{i \in 1..n} dom(X_i) \cap \bigcup_{j \in 1..m} dom(Y_j)\}$ is a strong backdoor because once these variables are set, the X_i and Y_j are disconnected and domain consistency on the decomposition prunes the same values as on the original constraint. Detecting the supports can therefore be done in $O(nd)$ time where $d = \max(\{|dom(X_i)|\}_{i \in 1..n} \cup \{|dom(Y_j)|\}_{j \in 1..m})$. Since there are at most $O(2^k)$ possible instantiations for the strong backdoor, enforcing domain consistency on the DISJOINT constraint can be achieved by calling $O(2^k)$ times domain consistency on the decomposition and taking the union of the 2^k domain consistent domains obtained. This takes $O(2^k nd)$ time. □

Other examples

We give some other examples of global constraints which are fixed-parameter tractable to propagate.

Uses

The global constraint USES$([X_1, \ldots, X_n], [Y_1, \ldots, Y_m])$ holds iff $\{X_i \mid 1 \leq i \leq n\} \subseteq \{Y_j \mid 1 \leq j \leq m\}$. That is, the X_i use only values used by the Y_j. Enforcing domain consistency on such a constraint is NP-hard (Bessière et al. 2005). However, it is fixed-parameter tractable to propagate in the total number of values in the domain of Y_j. We let $k = |\bigcup_{j \in 1..m} dom(Y_j)|$. We give an automaton for accepting solutions to the USES constraint that scans through Y_1 to Y_m and then X_1 to X_n. The states of this automaton are all the possible sets of values that can be used by Y_j. As there are k possible values in the domains of the Y_j, there are $O(2^k)$ possible states. The transition on seeing Y_j from state q goes to $q \cup \{Y_j\}$. We also only accept a transition on seeing X_i from a state q if $X_i \in q$. This transition goes to state q itself. We can therefore enforce domain consistency using the simple decomposition in (Quimper and Walsh 2007) in $O(2^k(n+m)d)$ time where $d = \max(\{|dom(X_i)|\}_{i \in 1..n} \cup \{|dom(Y_j)|\}_{j \in 1..m})$.

Among

The AMONG constraint was introduced in CHIP to model resource allocation problems like car sequencing (Beldiceanu and Contegean 1994). It counts the number of variables using values from a given set. AMONG$([X_1, \ldots, X_n], [d_1, \ldots, d_m], N)$ holds iff $N = |\{i \mid X_i = d_j, 1 \leq i \leq n, 1 \leq j \leq m\}|$. We here consider a generalisation of AMONG where instead of the fixed values d_j we have a set variable S. That is, AMONG$([X_1, \ldots, X_n], S, N)$ holds iff $N = |\{i \mid X_i \in S, 1 \leq i \leq n\}|$. Enforcing domain consistency on this extended version of AMONG is NP-hard (Bessiere et al. 2006a). However, it is fixed-parameter tractable to propagate in $k = |ub(S) \setminus lb(S)|$. AMONG can be decomposed into $(X_i \in S) \leftrightarrow (B_i = 1), \forall i \in 1..n$, and $\sum_{i \in 1..n} B_i = N$, where B_i are additional Boolean variables. (Note that the sum constraint is polynomial to propagate when it sums Boolean variables.) S is a cycle cutset of the decomposition. Thus, once S is set, domain consistency on the decomposition is equivalent to domain consistency on the original AMONG constraint. Since there are at most $O(2^k)$ possible instantiations for S, enforcing domain consistency on the AMONG constraint can be achieved by calling $O(2^k)$ times domain consistency on the decomposition and making the union of the 2^k domain consistent domains obtained. This takes $O(2^k nd)$ time.

Roots

Many counting and occurrence constraints can be specified using the global ROOTS constraint (Bessière et al. 2005). ROOTS$([X_1, \ldots, X_n], S, T)$ holds iff $S = \{i \mid X_i \in T\}$. As before, we consider S and T as shorthand for the vector of 0/1 variables representing the associated characteristic function. ROOTS can specify a wide range of other global constraints including the AMONG, ATMOST, ATLEAST, USES, DOMAIN and CONTIGUITY constraints. Enforcing domain consistency on ROOTS is NP-hard (Bessière et al. 2006c). However, it is fixed-parameter tractable to propagate in $k = |ub(T) \setminus lb(T)|$. ROOTS can be decomposed into $(i \in S) \leftrightarrow (X_i \in T), \forall i \in 1..n$. T is a cycle cutset of the decomposition. Thus, once T is set, domain consistency on the decomposition is equivalent to domain consistency on the original ROOTS constraint. Since there are at most $O(2^k)$ possible instantiations for T, enforcing domain consistency on the ROOTS constraint can be achieved by calling $O(2^k)$ times domain consistency on the decomposition and making the union of the 2^k domain consistent domains obtained. This takes $O(2^k nd)$ time.

Bound consistency

Often bound consistency on a global constraint is tractable but domain consistency is NP-hard to enforce. For instance, as we observed before, bound consistency on the NVALUE constraint is polynomial, but domain consistency is NP-hard (Bessière et al. 2006b). As a second example, bound consistency on the INTERDISTANCE constraint is polynomial, but domain consistency is NP-hard (Régin 1997; Artiouchine, Baptiste, and Durr 2008).

$\text{INTERDISTANCE}([X_1, \ldots, X_n], p)$ holds iff $|X_i - X_j| \geq p$ for $i \neq j$. This global constraint is useful in scheduling problems like runway sequencing. As a third example, the extended global cardinality constraint, $\text{EGCC}([X_1, \ldots, X_n], [O_1, \ldots, O_m])$ ensures $O_j = |\{i \mid X_i = j\}|$ for all j. Enforcing bound consistency on the O_j and domain consistency on the X_i is polynomial, but enforcing domain consistency on all variables is NP-hard (Quimper 2003). As a fourth and final example, bound consistency on a linear equation with coefficients set to one is polynomial, but domain consistency is NP-hard.

We can give a general fixed-parameter tractability result for such global constraints. The parameter is the sum of the number of non-interval domains and of the maximum number of "holes" in a domain. This measures how close the domains are to intervals. If there are many holes in the domains, then we are far from intervals and the problem is intractable. We define $\#intervals(S) = |\{v \in S \mid v+1 \notin S\}|$. If S contains no holes then $\#intervals(S) = 1$. If S contains p holes, then $\#intervals(S) = p+1$.

Theorem 4 *Suppose enforcing bound consistency on a global constraint over X_1 to X_n is polynomial. Then enforcing domain consistency is fixed-parameter tractable in $k = p + q$ where $p = \max(\{\#intervals(dom(X_i))\}_{i \in 1..n})$ and q is the number of non-interval variables.*

Proof: We give a decomposition with a strong backdoor. Consider X_i. Suppose the jth interval in $dom(X_i)$ runs from l_j to u_j (that is, $l_j - 1 \notin dom(X_i)$, $u_j + 1 \notin dom(X_i)$, and $[l_j, u_j] \subseteq dom(X_i)$). We introduce a variable Z_i. When $Z_i = j$, X_i will be restricted to the jth interval. To ensure this, we post $(Z_i = j) \leftrightarrow (l_j \leq X_i \leq u_j)$. By definition of bound consistency, when domains are all intervals, a constraint is bound consistent iff it contains at least a satisfying tuple. Thus, the Z_i are a strong backdoor into a subproblem which bound consistency can solve because when all Z_i are set, all X_i become intervals. Checking if a value v for X_i is domain consistent on the global constraint is done by instantiating X_i to v (which is an interval), and by trying all the possible instantiations of the backdoor, except Z_i, until bound consistency does not fail. This means that a support contains $X_i = v$. The total cost is bounded by $nd \cdot \prod_i \#intervals(dom(X_i))$ times the cost of bound consistency on this constraint, with $\prod_i \#intervals(dom(X_i)) \leq p^q$. $\square$

Meta-constraints

Parameterized complexity also provides insight into propagators for meta-constraints. A *meta-constraint* is a constraint that applies other constraints. For instance, given a constraint C of arity p, the meta-constraint $\text{CARDPATH}(N, [X_1, \ldots, X_n], C)$ holds iff N of the constraints, $C(X_1, \ldots, X_p)$ to $C(X_{n-p+1}, \ldots, X_n)$ hold (Beldiceanu and Carlsson 2001). This permits us to specify, say, that we want at least 2 days "off" in every 7 along a sequence of shifts. CARDPATH can encode a range of Boolean connectives since $N \geq 1$ gives disjunction, $N = 1$ gives exclusive or, and $N = 0$ gives negation. It therefore has numerous applications in domains including car-sequencing and rostering.

Enforcing domain consistency on CARDPATH is NP-hard even when enforcing domain consistency on each C is polynomial (Bessière et al. 2007). However, domain consistency is fixed-parameter tractable to enforce with respect to the sum of the arity of C and the maximum domain size.

Theorem 5 *Enforcing domain consistency on $\text{CARDPATH}(N, [X_1, \ldots, X_n], C)$ is fixed-parameter tractable in $k = p + d$ where p is the arity of C and $d = \max(\{|dom(X_i)|\}_{i \in 1..n})$.*

Proof: We give an automaton for accepting solutions to this constraint that scans through X_1 to X_n and then read N. The states of this automaton are the possible sequences of values of length smaller than $p-1$ labelled by the value 0, plus $n+1$ copies of the sequences of values of length $p-1$ labelled with the integers from 0 to n, and a final accepting state F. The integer labels count the number of times C has been satisfied so far. For $l < p$, the transition on reading v_l from the state $[v_1, \ldots, v_{l-1}]$ (labelled 0) goes to the state $[v_1, \ldots, v_{l-1}, v_l]$ with label 0. The transition on reading v_p from the state $[v_1, \ldots, v_{p-1}]$ with label r goes to the state $[v_2, \ldots, v_p]$ with label r', where $r' = r + 1$ if $C(v_1, \ldots, v_p)$ is satisfied and $r' = r$ otherwise. Finally, there is a transition reading character r from any state labelled with r to the final state F. The state F is the unique final state and the initial state is the empty sequence ϵ (labelled 0). There are $O(nd^{p-1})$ distinct states. We can therefore enforce domain consistency using the simple decomposition in (Quimper and Walsh 2007) in $O(d^p n^2)$ time. $\square$

This is another example of a fixed-parameter tractability result with respect to two parameters, p and d. However, CARDPATH is not fixed-parameter tractable with respect to d. In fact it is NP-hard when d is fixed.

Theorem 6 *Enforcing domain consistency on $\text{CARDPATH}(N, [X_1, \ldots, X_n], C)$ is NP-hard even if $|dom(X_i)| \leq 2 \ \forall i \in 1..n$.*

Proof: The reduction used in the proof of NP-hardness of CARDPATH in Theorem 12 (Bessière et al. 2007) uses just 1 or 2 domain values for each variable. $\square$

Symmetry breaking

Parameterized complexity also provides insight into symmetry breaking. Symmetry is a common feature of many real-world problems that dramatically increases the size of the search space if it is not taken into consideration. Consider, for instance, value symmetry. A *value symmetry* is a bijection σ on values that preserves solutions. That is, if $X_i = a_i$ for $1 \leq i \leq n$ is a solution then $X_i = \sigma(a_i)$ is also. For example, if two values are interchangeable, then any possible permutation of these values is a symmetry. A simple and effective mechanism to deal with symmetry is to add constraints to eliminate symmetric solutions (Puget 1993; Crawford et al. 1996). For example, given a set of value symmetries Σ, we can eliminate all symmetric solutions by posting the global constraint

VALSYMBREAK($\Sigma, [X_1, \ldots, X_n]$). This ensures that, for each $\sigma \in \Sigma$:

$$\langle X_1, \ldots, X_n \rangle \leq_{\text{lex}} \langle \sigma(X_1), \ldots, \sigma(X_n) \rangle$$

Enforcing domain consistency on such a global symmetry breaking constraint is NP-hard (Walsh 2007). However, this complexity depends on the number of symmetries. Breaking all value symmetry is fixed-parameter tractable in the number of symmetries.

Theorem 7 *Enforcing domain consistency on* VALSYMBREAK($\Sigma, [X_1, \ldots, X_n]$) *is fixed-parameter tractable in* $k = |\Sigma|$.

Proof: We give an automaton for accepting solutions to this constraint that scans through X_1 to X_n. The states of the automaton are the set of value symmetries which have been broken up to this point in the vector. For instance, at the ith state, σ is a symmetry in the state iff $\langle X_1, \ldots, X_i \rangle <_{\text{lex}} \langle \sigma(X_1), \ldots, \sigma(X_i) \rangle$. If we are in the state q, we accept X_i if $X_i \leq \sigma(X_i)$ or ($X_i > \sigma(X_i)$ and $\sigma \in q$). ¿From the state q, on seeing X_i, we move to $q \cup \{\sigma \mid \sigma \in \Sigma, X_i < \sigma(X_i)\}$. There are $O(2^k)$ possible states. We can enforce domain consistency using the simple decomposition in (Quimper and Walsh 2007) in $O(2^k nd)$ time where $d = \max(\{|dom(X_i)|\}_{i \in 1..n})$. $\Box$

Approximate consistency

Parameterized complexity also provides insight into the approximability of constraint propagation. For optimization problems, global constraints can incorporate a variable taking the objective value. We can use approximation algorithms to filter partially such global constraints (Sellmann 2003). For example, consider the knapsack constraint, KNAPSACK($[X_1, \ldots, X_n], [w_1, \ldots, w_n], C, [p_1, \ldots, p_n], P$) which holds iff:

$$\sum_{i=1}^{n} w_i X_i \leq C \quad \text{and} \quad \sum_{i=1}^{n} p_i X_i > P$$

C is the capacity and P is the profit. Based on a fully polynomial time approximation scheme, Sellmann gives a dynamic programming propagator for filtering the domains which guarantees an *approximate consistency* that filter values which only have supports that are a factor ϵ outside the optimal profit. A *fully polynomial time approximation scheme* (*FPTAS*) is an algorithm that computes an answer with relative error ϵ in time polynomial in the input length and in $1/\epsilon$. A weaker notion is an *efficient* polynomial time approximation scheme (*efficient PTAS*) which is an algorithm that computes an answer with relative error ϵ in time polynomial in the input length and in some function of ϵ.

We can use parameterized complexity results to show that such approximate consistency is intractable to achieve. In particular, we can exploit a theorem first proved by Bazgan that if a problem has an efficient *PTAS* then it is in FPT (Downey, Fellows, and Stege 1999). Consider again the constraint NVALUE($[X_1, \ldots, X_n], N$). We might ask if we can approximately filter domains.

Theorem 8 *There is no polynomial algorithm for enforcing approximate consistency on* NVALUE($[X_1, \ldots, X_n], N$) *unless* $FPT = W[2]$.

Proof: By Theorem 2, enforcing domain consistency on NVALUE($[X_1, \ldots, X_n], N$) is $W[2]$-hard in $k = \max(dom(N))$. By Bazgan's theorem, we cannot have an efficient *PTAS* (and thus *FPTAS*) for this problem unless $FPT = W[2]$. $\Box$

Other related work

In addition to the related work already mentioned, there are a number of other related studies. The analysis of (in)tractability has a long history in constraint programming. Such work has tended to focus on the structure of the constraint graph (e.g. (Freuder 1982; Dechter and Pearl 1989)) or on the semantics of the constraints (e.g. (Cooper, Cohen, and Jeavons 1994)). However, these lines of research are concerned with a constraint satisfaction problem as a whole, and do not say much about global constraints.

For global constraints of bounded arity, asymptotic analysis has been used to study propagation both in general and for constraints with a particular semantics. For example, the GAC-Schema algorithm of (Bessiere and Régin 1997) has an $O(d^n)$ time complexity on constraints of arity n and domains of size d, whilst the GAC algorithm of (Régin 1994) for the n-ary ALLDIFFERENT constraint has $O(n^{\frac{3}{2}}d)$ time complexity. For global constraints like the CUMULATIVE and CYCLE constraints, there are very immediate reductions from bin packing and Hamiltonian circuit which demonstrate that these constraints are intractable to propagate in general. Bessiere *et al.* showed that many other global constraints like NVALUE are also intractable to propagate (Bessiere et al. 2004). More recently, Samer and Szeider have studied the parameterized complexity of the EGCC constraint (Samer and Szeider 2008). They show it is fixed-parameter tractable to enforce domain consistency in the tree-width of the value graph and the largest possible number of occurrences, but is $W[1]$-hard in just the tree-width. Note that tree-width itself is NP-hard to compute.

Conclusions

We have argued that parameterized complexity is a useful tool with which to study global constraints. In particular, we have shown that many global constraints like NVALUE, DISJOINT, and ROOTS, which are intractable to propagate completely have natural parameters which make them fixed-parameter tractable. This tractability tends either to be the result of a simple dynamic program or of a decomposition which has a strong backdoor of bounded size. This strong backdoor is often a cycle cutset. We also showed that parameterized complexity can be used to study other aspects of constraint programming like symmetry breaking. For instance, we proved that value symmetry is fixed-parameter tractable to break in the number of symmetries. Finally, we argued that parameterized complexity can be used to derive results about the approximability of constraint propagation. For example, we cannot enforce an approximate consistency within a guaranteed factor for the NVALUE constraint.

The insights provided by this work can help design new search methods. For example, NVALUE, DISJOINT, USES, AMONG and ROOTS all have efficient propagators when the total number of value is small with respect to the number of variables. We might therefore build a propagator that propagates partially using a decomposition if the total number of values is large, and calls a complete method otherwise. We might also exploit their decompositions by branching on the backdoor variables. Finally, when we have an efficient bound consistency propagator, it may be worthwhile "eliminating" holes in the domains by branching on those variables whose domains have holes, or by introducing variables to represent the intervals without holes and branching on these introduced variables. In the longer term, we hope to apply other ideas about tractability from parameterized complexity like reduction to a problem kernel.

References

Artiouchine, K.; Baptiste, P.; and Durr, C. 2008. Runway sequencing with holding patterns. *European Journal of OR*. In press.

Beldiceanu, N., and Carlsson, M. 2001. Revisiting the cardinality operator and introducing cardinality-path constraint family. In *Proc. of the Int. Conf. on Logic Programming (ICLP 2001)*.

Beldiceanu, N., and Contegean, E. 1994. Introducing global constraints in CHIP. *Mathematical Computer Modelling* 20(12):97–123.

Beldiceanu, N. 2000. Global constraints as graph properties on a structured network of elementary constraints of the same type. SICS Technical Report T2000/01.

Bessiere, C., and Régin, J. 1997. Arc consistency for general constraint networks: Preliminary results. In *Proc. of the 15th IJCAI*.

Bessiere, C.; Hebrard, E.; Hnich, B.; and Walsh, T. 2004. The complexity of global constraints. In *Proc. of the 19th National Conf. on AI*. AAAI.

Bessière, C.; Hebrard, E.; Hnich, B.; Kiziltan, Z.; and Walsh, T. 2005. The range and roots constraints: Specifying counting and occurrence problems. In *Proc. of 19th IJCAI*, 60–65.

Bessiere, C.; Hebrard, E.; Hnich, B.; Kiziltan, Z.; and Walsh, T. 2006a. Among, common and disjoint constraints. In *Recent Advances in Constraints: Joint ERCIM/CoLogNET Int. Workshop on Constraint Solving and Constraint Logic Programming*.

Bessière, C.; Hebrard, E.; Hnich, B.; Kiziltan, Z.; and Walsh, T. 2006b. Filtering algorithms for the NVALUE constraint. *Constraints* 11(4):271–293.

Bessière, C.; Hebrard, E.; Hnich, B.; Kiziltan, Z.; and Walsh, T. 2006c. The ROOTS constraint. In *Proc. of 12th Int. Conf. on Principles and Practices of Constraint Programming (CP-2006)*.

Bessière, C.; Hebrard, E.; Hnich, B.; and Walsh, T. 2007. The complexity of global constraints. *Constraints* 12(2):239–259.

Cooper, M.; Cohen, D.; and Jeavons, P. 1994. Characterizing tractable constraints. *Artificial Intelligence* 65:347–361.

Crawford, J.; Luks, G.; Ginsberg, M.; and Roy, A. 1996. Symmetry breaking predicates for search problems. In *Proc. of the 5th Int. Conf. on Knowledge Representation and Reasoning, (KR '96)*.

Dechter, R., and Pearl, J. 1987. Network-based heuristics for constraint-satisfaction problems. *Artificial Intelligence* 34(1):1–38.

Dechter, R., and Pearl, J. 1989. Tree clustering for constraint networks. *Artificial Intelligence* 38:353–366.

Downey, R. G.; Fellows, M. R.; and Stege, U. 1999. Parameterized complexity: A framework for systematically confronting computational intractability. In *Contemporary Trends in Discrete Mathematics: From DIMACS and DIMATIA to the Future*. 49–99.

Freuder, E. 1982. A sufficient condition for backtrack-free search. *Journal of the ACM* 29(1):24–32.

Frisch, A.; Hnich, B.; Kiziltan, Z.; Miguel, I.; and Walsh, T. 2002. Global constraints for lexicographic orderings. In *Proc. of 8th Int. Conf. on Principles and Practices of Constraint Programming (CP-2002)*.

Garey, M., and Johnson, D. 1979. *Computers and intractability*. W.H. Freeman.

Pachet, F., and Roy, P. 1999. Automatic generation of music programs. In *Proc. of 5th Int. Conf. on Principles and Practice of Constraint Programming (CP99)*.

Puget, J.-F. 1993. On the satisfiability of symmetrical constrained satisfaction problems. In *Proc. of ISMIS'93*.

Quimper, C.-G., and Walsh, T. 2007. Decomposing global grammar constraints. In *13th Int. Conf. on Principles and Practices of Constraint Programming (CP-2007)*.

Quimper, C. 2003. Enforcing domain consistency on the extended global cardinality constraint is NP-hard. Technical Report CS-2003-39, University of Waterloo.

Régin, J.-C., and Rueher, M. 2000. A global constraint combining a sum constraint and difference constraints. In *Proc. of 6th Int. Conf. on Principles and Practice of Constraint Programming (CP2000)*.

Régin, J.-C. 1994. A filtering algorithm for constraints of difference in CSPs. In *Proc. of the 12th National Conf. on AI*. AAAI.

Régin, J.-C. 1996. Generalized arc consistency for global cardinality constraints. In *Proc. of the 13th National Conf. on AI*. AAAI.

Régin, J.-C. 1997. The global minimum distance constraint. Technical report, ILOG Inc.

Samer, M., and Szeider, S. 2008. Tractable cases of the extended global cardinality constraint. In *Proc. of CATS 2008, Computing: The Australasian Theory Symposium*.

Sellmann, M. 2003. Approximated consistency for knapsack constraints. In *Proc. of 9th Int. Conf. on Principles and Practice of Constraint Programming (CP2003)*.

Walsh, T. 2007. Breaking value symmetry. In *Proc. of 13th Int. Conf. on Principles and Practices of Constraint Programming (CP-2007)*.

Williams, R.; Gomes, C.; and Selman, B. 2003. Backdoors to typical case complexity. In *Proc. of the 18th IJCAI*.