Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence

Kernels for Global Constraints*

Serge Gaspers and **Stefan Szeider**
Vienna University of Technology
Vienna, Austria
gaspers@kr.tuwien.ac.at, stefan@szeider.net

Abstract

Bessière *et al.* (AAAI'08) showed that several intractable global constraints can be efficiently propagated when certain natural problem parameters are small. In particular, the complete propagation of a global constraint is fixed-parameter tractable in k – the number of holes in domains – whenever bound consistency can be enforced in polynomial time; this applies to the global constraints ATMOST-NVALUE and EXTENDED GLOBAL CARDINALITY (EGC).

In this paper we extend this line of research and introduce the concept of *reduction to a problem kernel*, a key concept of parameterized complexity, to the field of global constraints. In particular, we show that the consistency problem for ATMOST-NVALUE constraints admits a linear time reduction to an equivalent instance on $O(k^2)$ variables and domain values. This small kernel can be used to speed up the complete propagation of NVALUE constraints. We contrast this result by showing that the consistency problem for EGC constraints does not admit a reduction to a polynomial problem kernel unless the polynomial hierarchy collapses.

1 Introduction

Constraint programming (CP) offers a powerful framework for efficient modeling and solving of a wide range of hard problems [Rossi *et al.*, 2006]. At the heart of efficient CP solvers are so-called *global constraints* that specify patterns that frequently occur in real-world problems. Efficient propagation algorithms for global constraints help speed up the solver significantly [van Hoeve and Katriel, 2006]. For instance, a frequently occurring pattern is that we require that certain variables must all take different values (e.g., activities requiring the same resource must all be assigned different times). Therefore most constraint solvers provide a global ALLDIFFERENT constraint and algorithms for its propagation. Unfortunately, for several important global constraints a complete propagation is NP-hard, and one switches therefore to incomplete propagation such as bound consistency [Bessière *et al.*, 2004]. In their AAAI'08 paper, Bessière *et al.* [2008] showed that a complete propagation of several intractable constraints can efficiently be done as long as certain natural problem parameters are small, i.e., the propagation is *fixed-parameter tractable* [Downey and Fellows, 1999]. Among others, they showed fixed-parameter tractability of the ATLEAST-NVALUE and EXTENDED GLOBAL CARDINALITY (EGC) constraints parameterized by the number of "holes" in the domains of the variables. If there are no holes, then all domains are intervals and complete propagation is polynomial by classical results; thus the number of holes provides a way of *scaling up* the nice properties of constraints with interval domains.

In this paper we bring this approach a significant step forward, picking up a long-term research objective suggested by Bessière *et al.* [2008]: whether intractable global constraints admit a *reduction to a problem kernel* or *kernelization*.

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., [Fellows, 2006; Guo and Niedermeier, 2007; Rosamond, 2010]). It can be seen as a *preprocessing with performance guarantee* that reduces an instance in polynomial time to an equivalent instance, the *kernel*, whose size is a function of the parameter [Fellows, 2006; Guo and Niedermeier, 2007; Fomin, 2010].

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. Consequently one aims at kernels that are as small as possible; the kernel size provides a performance guarantee for the preprocessing. Some NP-hard 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 [Bodlaender *et al.*, 2009a].

Kernelization fits perfectly into the context of CP where preprocessing and data reduction (e.g., in terms of local consistency algorithms, propagation, and domain filtering) are key methods [Bessière, 2006; van Hoeve and Katriel, 2006].

Results Do the global constraints ATMOST-NVALUE and EGC admit polynomial kernels? We show that the answer is *"yes"* for the former and *"no"* for the latter.

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' in time $O(1.6181^k)$. The combination of kernelization

*Research funded by the ERC (COMPLEX REASON, 239962).

and branching yields efficient algorithms for the consistency and propagation of (ATMOST-)NVALUE constraints.

On the other hand, we show that a similar result is unlikely for the EGC constraint: One cannot reduce an EGC constraint C with k holes in polynomial time to a consistency-equivalent EGC constraint C' of size polynomial in k. This result is subject to the complexity theoretic assumption that NP $\not\subseteq$ coNP/poly whose failure implies the collapse of the Polynomial Hierarchy to its third level, which is considered highly unlikely by complexity theorists.

2 Formal Background

Parameterized Complexity A *parameterized problem* 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. A parameterized problem P is *fixed-parameter tractable* (FPT) if a given instance (x, k) can be solved in time $O(f(k) \cdot p(|x|))$ where f is an arbitrary computable function of k and p is a polynomial in the input size $|x|$.

Kernels A *kernelization* for a parameterized problem $P \subseteq \Sigma^* \times \mathbb{N}$ 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) $(x, k) \in P$ iff $(x', k') \in P$ and (ii) $|x'| + k' \leq g(k)$, where g is an arbitrary computable function. The function g is referred to as the *size* of the kernel. If g is a polynomial then we say that P admits a *polynomial kernel*.

Global Constraints An instance of the 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 denote by $dom(x)$ the domain of a variable x and by $scope(C)$ the subset of variables involved in a constraint C. An *instantiation* is an assignment α of values to variables such that $\alpha(x) \in dom(x)$ for each variable $x \in scope(C)$. A constraint can be specified extensionally by listing all legal instantiations of its variables or intensionally, by giving an expression involving the variables in the constraint scope [Smith, 2006]. *Global constraints* are certain extensionally described constraints involving an arbitrary number of variables [van Hoeve and Katriel, 2006]. For example, an instantiation is legal for an ALLDIFFERENT global constraint C if it assigns pairwise different values to the variables in $scope(C)$.

Consistency A global constraint C is *consistent* if there is a legal instantiation of its variables. C is *hyper arc consistent* (*HAC*) if for each variable $x \in scope(C)$ and each value $v \in dom(x)$, there is a legal instantiation α such that $\alpha(x) = v$ (in that case we say that C supports v for x). In the literature, HAC is also called *domain consistent* or *generalized arc consistent*. C is *bound consistent* if when a variable $x \in scope(C)$ is assigned the minimum or maximum value of its domain, there are compatible values between the minimum and maximum domain value for all other variables in $scope(C)$. The main algorithmic problems for a global constraint C are the following: *Consistency*, to decide whether C is consistent, and *Enforcing HAC*, to remove from all domains the values that are not supported by the respective variable.

It is clear that if HAC can be enforced in polynomial time for a constraint C, then the consistency of C can also be decided in polynomial time (we just need to see if any domain became empty). The reverse is true for constraints that satisfy a certain closure property (see [van Hoeve and Katriel, 2006]), which is the case for most constraints of practical use, and in particular for all constraints considered below. The same correspondence holds with respect to fixed-parameter tractability. Hence, we will focus mainly on Consistency.

3 NValue Constraints

The NVALUE constraint was introduced by Pachet and Roy [1999]. For a set of variables X and a variable N, NVALUE(X, N) is consistent if there is an assignment α such that exactly $\alpha(N)$ different values are used for the variables in X. ALLDIFFERENT is the special case where $dom(N) = \{|X|\}$. Beldiceanu [2001] and Bessière *et al.* [2006] decompose NVALUE constraints into two other global constraints: ATMOST-NVALUE and ATLEAST-NVALUE, which require that at most N or at least N values are used for the variables in X, respectively. The Consistency problem is NP-complete for NVALUE and ATMOST-NVALUE constraints, and polynomial time solvable for ATLEAST-NVALUE constraints.

For checking the consistency of an ATMOST-NVALUE constraint C, we are given an instance $\mathcal{I}$ consisting of a set of variables $X = \{x_1, \ldots, x_n\}$, a totally ordered set of values D, a map $dom : X \to 2^D$ assigning a non-empty domain $dom(x) \subseteq D$ to each variable $x \in X$, and an integer N.[1] A *hole* in a subset $D' \subseteq D$ is a couple $(u, w) \in D' \times D'$, such that there is a $v \in D \setminus D'$ with $u < v < w$ and there is no $v' \in D'$ with $u < v' < w$. We denote the number of holes in the domain of a variable $x \in X$ by #holes(x). The parameter of the consistency problem for ATMOST-NVALUE constraints is $k = \sum_{x \in X}$ #holes(x). An *interval* $I = [v_1, v_2]$ of a variable x is an inclusion-wise maximal hole-free subset of its domain. Its *left endpoint* $\mathsf{l}(I)$ and *right endpoint* $\mathsf{r}(I)$ are the values v_1 and v_2, respectively. Fig. 1 gives an example of an instance and its interval representation. We assume that instances are given by a succinct description, in which the domain of a variable is given by the left and right endpoint of each of its intervals. As the number of intervals of the instance $\mathcal{I} = (X, D, dom, N)$ is $n + k$, its size is $|\mathcal{I}| = O(n + |D| + k)$. In case dom is given by an extensive list of the values in the domain of each variable, a succinct representation can be computed in linear time.

A greedy algorithm by Beldiceanu [2001] checks the consistency of an ATMOST-NVALUE constraint in linear time when all domains are intervals (i.e., $k = 0$). Further, Bessière *et al.* [2008] have shown that Consistency (and Enforcing HAC) is FPT, parameterized by the number of holes, for all constraints for which bound consistency can be enforced in polynomial time. A simple algorithm for checking the consistency of ATMOST-NVALUE goes over all instances obtained from restricting the domain of each variable to one of its intervals, and executes the algorithm of [Beldiceanu, 2001] for each of these 2^k instances. The running time of this FPT algorithm is clearly bounded by $O(2^k \cdot |\mathcal{I}|)$.

[1]If D is not part of the input (or is very large), we may construct D by sorting the set of all endpoints of intervals in time $O((n + k) \log(n + k))$. Since, w.l.o.g., a solution contains only endpoints of intervals, this step does not compromise the correctness.

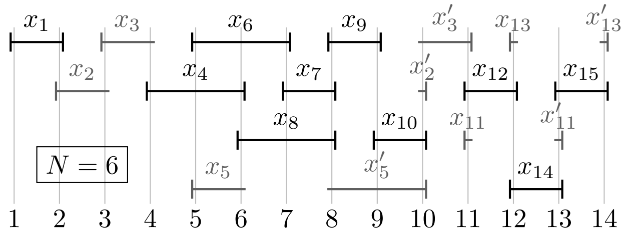

Fig. 1: Interval representation of an ATMOST-NVALUE instance $\mathcal{I} = (X, D, dom, N)$, with $X = \{x_1, \dots, x_{15}\}$, $N = 6$, $D = \{1, \dots, 14\}$, and $dom(x_1) = \{1, 2\}$, $dom(x_2) = \{2, 3, 10\}$, etc.

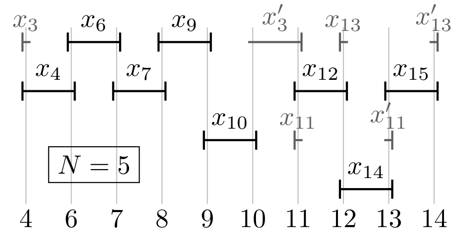

Fig. 2: Instance obtained from the instance of Fig. 1 by exhaustively applying rules **Red-$\subseteq$**, **Red-Dom**, and **Red-Unit**.

3.1 Kernelization Algorithm

Let $\mathcal{I} = (X, D, dom, N)$ be an instance for the consistency problem for ATMOST-NVALUE constraints. The *friends* of an interval I are the other intervals of I's variable. An interval is *optional* if it has at least one friend, and *required* otherwise. For a value $v \in D$, let $\mathsf{ivl}(v)$ denote the set of intervals containing v.

A *solution* for $\mathcal{I}$ is a subset $S \subseteq D$ of at most N values such that there exists an instantiation assigning the values in S to the variables in X. The algorithm may detect for some value $v \in D$, that, if the problem has a solution, then it has a solution containing v. In this case, the algorithm *selects* v, i.e., it removes all variables whose domain contains v, it removes v from D, and it decrements N by one. The algorithm may detect for some value $v \in D$, that, if the problem has a solution, then it has a solution not containing v. In this case, the algorithm *discards* v, i.e., it removes v from every domain and from D. (Note that no new holes are created with respect to $D \setminus \{v\}$.) The algorithm may detect for some variable x, that every solution for $(X \setminus \{x\}, D, dom|_{X \setminus \{x\}}, N)$ contains a value from $dom(x)$. In that case, it *removes* x.

The algorithm sorts the intervals by increasing right endpoint (ties are broken arbitrarily). Then, it exhaustively applies the following three reduction rules.

Red-$\subseteq$: If there are two intervals I, I' such that $I' \subseteq I$ and I' is required, then remove the variable of I.

Red-Dom: If there are two values $v, v' \in D$ such that $\mathsf{ivl}(v') \subseteq \mathsf{ivl}(v)$, then discard v'.

Red-Unit: If $|dom(x)| = 1$ for some variable x, then select the value in $dom(x)$.

In the example from Fig. 1, **Red-$\subseteq$** removes the variables x_5 and x_8 because $x_{10} \subseteq x_5'$ and $x_7 \subseteq x_8$, **Red-Dom** removes the values 1 and 5, **Red-Unit** selects 2, which deletes variables x_1 and x_2, and **Red-Dom** removes 3 from D. The resulting instance is depicted in Fig. 2.

After none of these rules apply, the algorithm scans the remaining intervals from left to right. An interval that has already been scanned is either a *leader* or a *follower* of a subset of leaders. Informally, for a leader L, if a solution contains $\mathsf{r}(L)$, then there is a solution containing $\mathsf{r}(L)$ and the right endpoint of each of its followers.

The algorithm scans the first intervals up to, and including, the first required interval. All these intervals become leaders.

The algorithm then continues scanning intervals one by one. Let I be the interval that is currently scanned and I_p be the last interval that was scanned. The *active* intervals are those that have already been scanned and intersect I_p. A *popular* leader is a leader that is either active or has at least one active follower.

- If I is optional, then I becomes a leader, the algorithm continues scanning intervals until scanning a required interval; all these intervals become leaders.
- If I is required, then it becomes a follower of all popular leaders that do not intersect I and that have no follower intersecting I. If all popular leaders have at least two followers, then set $N := N - 1$ and **merge** the second-last follower of each popular leader with the last follower of the corresponding leader; i.e., for every popular leader, the right endpoint of its second-last follower is set to the right endpoint of its last follower, and then the last follower of every popular leader is removed.

After having scanned all the intervals, the algorithm exhaustively applies Rules **Red-$\subseteq$**, **Red-Dom**, and **Red-Unit** again.

In the example from Fig. 2, variable x_6 is merged with x_9, and x_7 with x_{10}. **Red-Dom** then removes the values 7 and 8, resulting in the instance depicted in Fig. 3.

3.2 Correctness and Kernel Size

Let $\mathcal{I}' = (X', D', dom', N')$ be the instance resulting from applying one operation of the kernelization algorithm to an instance $\mathcal{I} = (X, D, dom, N)$. An operation is an instruction which modifies the instance: **Red-$\subseteq$**, **Red-Dom**, **Red-Unit**, and **merge**. We show that there exists a solution S for $\mathcal{I}$ iff there exists a solution S' for $\mathcal{I}'$. A solution is *nice* if each of its elements is the right endpoint of some interval. Clearly, for every solution, a nice solution of the same size can be obtained by shifting each value to the next right endpoint of an interval. Thus, when we construct S' from S (or vice-versa), we may assume that S is nice.

Rule **Red-$\subseteq$** is sound because a solution for $\mathcal{I}$ is a solution for $\mathcal{I}'$ and vice-versa, because any solution $\mathcal{I}'$ contains a value v of $I \subseteq I'$, as I is required. **Red-Dom** is correct because if $v' \in S$, then $S' := (S \setminus \{v'\}) \cup \{v\}$ is a solution for $\mathcal{I}'$ and for $\mathcal{I}$. **Red-Unit** is obviously correct ($S = S' \cup dom(x)$).

After having applied these 3 reduction rules, observe that the first interval is optional and contains only one value. Suppose the algorithm has started scanning intervals. By construction, the following properties apply to $\mathcal{I}'$.

Property 1. *A follower does not intersect any of its leaders.*

Property 2. *If I, I' are two (distinct) followers of the same leader, then I and I' do not intersect.*

Before proving the correctness of the **merge** operation, let us first show that the subset of leaders of a follower is not empty.

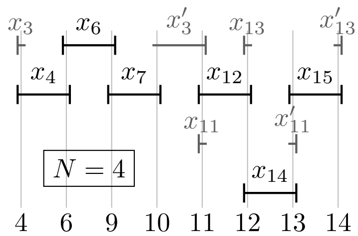

Fig. 3: Kernelized instance.

Claim 1. *Every interval that has been scanned is either a leader or a follower of at least one leader.*

Proof. First, note that **Red-Dom** ensures that each value in D is the left endpoint of some interval and the right endpoint of some interval. Let I be the interval that is currently scanned and I_p be the previously scanned interval. If I_p or I is optional, then I becomes a leader. Suppose I and I_p are required. We have $\mathsf{l}(I) > \mathsf{l}(I_p)$, otherwise I would have been removed by **Red-**$\subseteq$. By **Red-Dom**, there is some interval I_ℓ with $\mathsf{r}(I_\ell) = \mathsf{l}(I_p)$. If I_ℓ is a leader, I becomes a follower of I_ℓ; otherwise I becomes a follower of I_ℓ's leader. $\square$

The following two lemmas prove the correctness of the **merge** operation. Recall that $\mathcal{I}'$ is an instance obtained from $\mathcal{I}$ by one application of the **merge** operation.

Lemma 1. *If S is a nice solution for $\mathcal{I}$, then there exists a solution S' for $\mathcal{I}'$ with $S' \subseteq S$.*

Proof. Consider the step where the kernelization algorithm applies the **merge** operation. At that step, each popular leader has at least two followers and the algorithm merges the last two followers of each popular leader and decrements N by one. The currently scanned interval is I. Let F_2 denote the set of intervals that are the second-last follower of a popular leader, and F_1 the set of intervals that are the last follower of a popular leader before merging. Let M denote the set of merged intervals. Clearly, every interval of $F_1 \cup F_2 \cup M$ is required as all followers are required.

Claim 2. *Every interval in F_1 intersects $\mathsf{l}(I)$.*

Proof. Let $I_1 \in F_1$. By construction, $\mathsf{r}(I_1) \in I$, as I becomes a follower of every popular leader that has no follower intersecting I, and no follower has a right endpoint larger than $\mathsf{r}(I)$. Moreover, $\mathsf{l}(I_1) \leq \mathsf{l}(I)$ as no follower is a strict subset of I by **Red-**$\subseteq$ and the fact that all followers are required. $\square$

Let I^- be the interval of F_2 with the largest right endpoint. Let L be a leader of I^-. By construction and **Red-**$\subseteq$, L is a leader of I and is thus popular. Let $t_1 \in S \cap I$ be the smallest value of S that intersects I and let $t_2 \in S \cap I^-$ be the largest value of S that intersects I^-. By Property 2, $t_2 < t_1$.

Claim 3. *S contains no value t_0 such that $t_2 < t_0 < t_1$.*

Proof. Suppose S contained such a value t_0. As S is nice, t_0 is the right endpoint of some interval I_0. As t_2 is the rightmost value intersecting S and any interval in F_2, I_0 is not in F_2. As I_0 has already been scanned, and was scanned after every interval in F_2, I_0 is in F_1. However, by Claim 2, I_0 intersects $\mathsf{l}(I)$. As no scanned interval has a larger right endpoint than I, $t_0 \in S \cap I$, which contradicts the fact that t_1 is the smallest value in $S \cap I$ and that $t_0 < t_1$. $\square$

Claim 4. *Suppose $I_1 \in F_1$ and $I_2 \in F_2$ are the last and second-last follower of a popular leader L', respectively. Let $M_{12} \in M$ denote the interval obtained from merging I_2 with I_1. If $t_2 \in I_2$, then $t_1 \in M_{12}$.*

Proof. Assume otherwise that $t_2 \in I_2$, but $t_1 \notin M_{12}$. As $t_2 < t_1$, we have $t_1 > \mathsf{r}(M_{12}) = \mathsf{r}(I_1)$. But then S is not a solution as $t_2 < \mathsf{l}(I_1)$ and $S \cap I_1 = \emptyset$ by Claim 3. $\square$

Claim 5. *If I' is an interval with $t_2 \in I'$, then $I' \in F_2 \cup F_1$.*

Proof. First, suppose I' is a leader. As every leader has at least two followers when I is scanned, I' has two followers whose left endpoint is larger than $\mathsf{r}(I') \geq t_2$ (by Property 1) and smaller than $\mathsf{l}(I) \leq t_1$ (by **Red-**$\subseteq$). Thus, at least one of them is included in the interval (t_2, t_1) by Property 2, which contradicts S being a solution by Claim 3.

Similarly, if I' is a follower of a popular leader, but not among the last two followers of any popular leader, Claim 3 leads to a contradiction as well.

Finally, if I' is a follower, but has no popular leader, then it is to the left of a popular leader, and thus to the left of t_2. $\square$

Consider the set T_2 of intervals that intersect t_2. By Claim 5, $T_2 \subseteq F_2 \cup F_1$. For every interval $I' \in T_2 \cap F_2$, the corresponding merged interval of $\mathcal{I}'$ intersects t_1 by Claim 4. For every interval $I' \in T_2 \cap F_1$, and every interval $I'' \in F_2$ with which I' is merged, S contains some value $x \in I''$ with $x < t_2$. Thus, $S' := S \setminus \{t_2\}$ is a solution for $\mathcal{I}'$. $\square$

Lemma 2. *If S' is a nice solution for $\mathcal{I}'$, then there exists a solution S for $\mathcal{I}$ with $S' \subseteq S$.*

Proof. As in the previous proof, consider the step where the kernelization algorithm applies the **merge** operation. The currently scanned interval is I. Let F_2 and F_1 denote the set of intervals that are the second-last and last follower of a popular leader before merging, respectively. Let M denote the set of merged intervals.

By Claim 2, every interval of M intersects $\mathsf{l}(I)$. On the other hand, every interval of $\mathcal{I}'$ whose right endpoint intersects I is in M, by construction. Thus, S' contains the right endpoint of some interval of M. Let t_1 denote the smallest such value, and let I_1 denote the interval of $\mathcal{I}$ with $\mathsf{r}(I_1) = t_1$ (due to **Red-**$\subseteq$, there is a unique such interval). Let I_2 denote the interval of $\mathcal{I}$ with the smallest right endpoint such that there is a leader L whose second-last follower is I_2 and whose last follower is I_1, and let $t_2 := \mathsf{r}(I_2)$.

Claim 6. *Let $I'_1 \in F_1$ and $I'_2 \in F_2$ be two intervals that are merged into one interval M'_{12}. If $t_1 \in M'_{12}$, then $t_2 \in I'_2$.*

Proof. Suppose $t_1 \in M'_{12}$ but $t_2 \notin I'_2$. If $I'_2 \subseteq (t_2, \mathsf{l}(I'_1))$, then I'_2 would have become a follower of L, which contradicts that I_1 is the last follower of L. Otherwise, $\mathsf{r}(I'_2) < t_2$. But then, I_1 is a follower of the same leader as I'_1, as $\mathsf{l}(I_1) \leq \mathsf{l}(I'_1)$, and thus $I_1 = I'_1$. By definition of I_2, however, $t_2 = \mathsf{r}(I_2) \leq \mathsf{r}(I'_2)$, a contradiction. $\square$

By the previous claim, a solution S for $\mathcal{I}$ is obtained from a solution S' for $\mathcal{I}'$ by setting $S := S' \cup \{t_2\}$. $\square$

Thus, the kernelization algorithm returns an equivalent instance. To bound the kernel size by a polynomial in k, let $\mathcal{I}^* = (V^*, D^*, dom^*, N^*)$ be the instance resulting from applying the algorithm to an instance $\mathcal{I} = (V, D, dom, N)$.

Property 3. *$\mathcal{I}$ and $\mathcal{I}^*$ have at most $2k$ optional intervals.*

Property 3 holds for $\mathcal{I}$ as every optional interval is adjacent to at least one hole and each hole is adjacent to two optional intervals. It holds for $\mathcal{I}^*$ as the algorithm introduces no holes.

Lemma 3. *$\mathcal{I}^*$ has at most $4k$ leaders.*

Proof. For every required interval that becomes a leader, an optional interval also becomes a leader. As every interval is scanned only once, the number of leaders is at most $4k$ by Property 3. □

The following lemma can be proved by analyzing how many new followers a leader can get in a period where no optional interval is scanned. The analysis considers different cases based on the popular leader with the rightmost right endpoint, and is omitted here due to space constraints. See the ArXiv Report abs/1104.2541 for the proof.

Lemma 4. *Every leader has at most $4k$ followers.*

By Claim 1 and Lemmas 3 and 4, $\mathcal{I}^*$ has $O(k^2)$ intervals. By **Red-Dom**, every value in D^* is the right endpoint and the left endpoint of some interval. Thus, $|X^*| + |D^*| = O(k^2)$. We omit the running time analysis and arrive at our main theorem.

Theorem 1. *The Consistency problem for* ATMOST-NVALUE *constraints, parameterized by the number k of holes, admits a linear time reduction to a problem kernel with $O(k^2)$ variables and $O(k^2)$ domain values.*

Using the succinct description of the domains, the size of the kernel can be bounded by $O(k^2)$.

3.3 Improved FPT Algorithm and HAC

Using the kernel from Theorem 1 and the simple algorithm described in the beginning of this section, one arrives at a $O(2^k k^2 + |\mathcal{I}|)$ time algorithm for checking the consistency of an ATMOST-NVALUE constraint. Borrowing ideas from the kernelization algorithm, we now reduce the exponential dependency on k in the running time.

Theorem 2. *The Consistency problem for* ATMOST-NVALUE *constraints admits a $O(\rho^k k^2 + |\mathcal{I}|)$ time algorithm, where k is the number of holes in the domains of the input instance $\mathcal{I}$, and $\rho = \frac{1+\sqrt{5}}{2} < 1.6181$.*

Proof. The first step of the algorithm invokes the kernelization algorithm and obtains an equivalent instance $\mathcal{I}'$ with $O(k^2)$ intervals in time $O(|\mathcal{I}|)$.

Now, we describe a branching algorithm checking the consistency of $\mathcal{I}'$. Let I_1 denote the first interval of $\mathcal{I}'$ (in the ordering by increasing right endpoint). I_1 is optional. Let $\mathcal{I}_1$ denote the instance obtained from $\mathcal{I}'$ by selecting $\mathsf{r}(I_1)$ and exhaustively applying Rules **Red-Dom** and **Red-Unit**. Let $\mathcal{I}_2$ denote the instance obtained from $\mathcal{I}'$ by removing I_1 (if I_1 had exactly one friend, this friend becomes required) and exhaustively applying **Red-Dom** and **Red-Unit**. Clearly, $\mathcal{I}'$ is consistent iff $\mathcal{I}_1$ or $\mathcal{I}_2$ is consistent.

Note that both $\mathcal{I}_1$ and $\mathcal{I}_2$ have at most $k-1$ holes. If either $\mathcal{I}_1$ or $\mathcal{I}_2$ has at most $k-2$ holes, the algorithm recursively checks whether at least one of $\mathcal{I}_1$ and $\mathcal{I}_2$ is consistent. If both $\mathcal{I}_1$ and $\mathcal{I}_2$ have exactly $k-1$ holes, we note that in $\mathcal{I}'$,

(1) I_1 has one friend,

(2) no other optional interval intersects I_1, and

(3) the first interval of both $\mathcal{I}_1$ and $\mathcal{I}_2$ is I_f, which is the third optional interval in $\mathcal{I}'$ if the second optional interval is the friend of I_1, and the second optional interval otherwise.

Thus, the instance obtained from $\mathcal{I}_1$ by removing I_1's friend and applying **Red-Dom** and **Red-Unit** may differ from $\mathcal{I}_2$ only in N. Let s_1 and s_2 denote the number of values smaller than $\mathsf{r}(I_f)$ that have been selected to obtain $\mathcal{I}_1$ and $\mathcal{I}_2$ from $\mathcal{I}'$, respectively. If $s_1 \leq s_2$, then the non-consistency of $\mathcal{I}_1$ implies the non-consistency of $\mathcal{I}_2$. Thus, the algorithm need only recursively check whether $\mathcal{I}_1$ is consistent. On the other hand, if $s_1 > s_2$, then the non-consistency of $\mathcal{I}_2$ implies the non-consistency of $\mathcal{I}_1$. Thus, the algorithm need only recursively check whether $\mathcal{I}_2$ is consistent.

The recursive calls of the algorithm may be represented by a search tree labeled with the number of holes of the instance. As the algorithm either branches into only one subproblem with at most $k-1$ holes, or two subproblems with at most $k-1$ and at most $k-2$ holes, respectively, the number of leaves of this search tree is $T(k) \leq T(k-1) + T(k-2)$, with $T(0) = T(1) = 1$. Using standard techniques in the analysis of exponential time algorithms (see, e.g., [Fomin and Kratsch, 2010]), and by noticing that the number of operations executed at each node of the search tree is $O(k^2)$, the running time of the algorithm can be upper bounded by $O(\rho^k k^2)$. □

For the example of Fig. 3, the instances $\mathcal{I}_1$ and $\mathcal{I}_2$ are computed by selecting the value 4, and removing the interval x_3, respectively. The reduction rules select the value 9 for $\mathcal{I}_1$ and the values 6 and 10 for $\mathcal{I}_2$. Both instances start with the interval x_{11}, and the algorithm recursively solves $\mathcal{I}_1$ only, where the solution $\{4, 9, 12, 13\}$ is obtained for the kernelized instance, which corresponds to the solution $\{2, 4, 7, 9, 12, 13\}$ for the instance of Fig. 1.

Corollary 1. *HAC for an* ATMOST-NVALUE *constraint can be enforced in time $O(\rho^k \cdot k^2 \cdot |D| + |\mathcal{I}| \cdot |D|)$, where k is the number of holes in the domains of the input instance $\mathcal{I} = (X, D, dom, N)$, and $\rho = \frac{1+\sqrt{5}}{2} < 1.6181$.*

Proof. We first remark that if a value v can be filtered from the domain of a variable x (i.e., v has no support for x), then v can be filtered from the domain of all variables, as for any legal instantiation α with $\alpha(x') = v$, $x' \in X \setminus \{x\}$, the assignment obtained from α by setting $\alpha(x) := v$ is a legal instantiation as well. Also, filtering the value v creates no new holes as the set of values can be set to $D \setminus \{v\}$.

Now we enforce HAC by applying $O(|D|)$ times the algorithm from Theorem 2. Assume the instance $\mathcal{I} = (X, D, dom, N)$ is consistent. If $(X, D, dom, N-1)$ is consistent, then no value can be filtered. Otherwise, check, for each $v \in D$, whether the instance obtained from selecting v is consistent and filter v if this is not the case. □

4 Extended Global Cardinality Constraints

An EGC constraint C is specified by a set of variables $\{x_1, \dots, x_n\}$ and for each value $v \in \bigcup_{i=1}^{n} dom(x_i)$ a set $D(v)$ of non-negative integers. C is consistent if each variable can take a value from its domain such that the number of variables taking a value v belongs to the set $D(v)$.

The Consistency problem for EGC constraints is NP-hard [Quimper *et al.*, 2004] in general and polynomial time solvable if all sets $D(\cdot)$ are intervals [Régin, 1996]. By the result of Bessière *et al.* [2008], the Consistency problem for EGC constraints is FPT, parameterized by the number of holes in the sets $D(\cdot)$. Thus Régin's result generalizes to instances that are close to the interval case. However, it is unlikely that EGC constraints admit a polynomial kernel.

Theorem 3. *The Consistency problem for* EGC *constraints, parameterized by the number of holes in the sets $D(\cdot)$, does not admit a polynomial kernel unless* NP $\subseteq$ coNP/poly.

Proof. The *unparameterized version* of a parameterized problem $P \subseteq \Sigma^* \times \mathbb{N}$ is UP(P) $= \{ x\#1^k : (x,k) \in P \} \subseteq (\Sigma \cup \{\#\})^*$ where 1 is an arbitrary symbol from Σ and $\#$ is a new symbol not in Σ. Let $P, Q \subseteq \Sigma^* \times \mathbb{N}$ be parameterized problems. P is *polynomial parameter reducible* to Q if there is a polynomial time computable function $f : \Sigma^* \times \mathbb{N} \to \Sigma^* \times \mathbb{N}$ and a polynomial p, such that for all $(x,k) \in \Sigma^* \times \mathbb{N}$, we have $(x,k) \in P$ iff $(x',k') = f(x,k) \in Q$, and $k' \le p(k)$.

We prove the theorem by combining three known results.

(1) [Bodlaender *et al.*, 2009b] Let P and Q be parameterized problems such that UP(P) is NP-complete, UP(Q) is in NP, and P is polynomial parameter reducible to Q. If Q has a polynomial kernel, then P has a polynomial kernel.

(2) [Fortnow and Santhanam, 2008] The problem of deciding the satisfiability of a CNF formula (SAT), parameterized by the number of variables, does not admit a polynomial kernel, unless NP $\subseteq$ coNP/poly.

(3) [Quimper *et al.*, 2004] Given a CNF formula F on k variables, one can construct in polynomial time an EGC constraint C_F such that (i) for each value v of C_F, $D(v) = \{0, i_v\}$ for an integer $i_v > 0$, (ii) $i_v > 1$ for at most $2k$ values v, and (iii) F is satisfiable iff C_F is consistent. Thus, the number of holes in C_F is at most twice the number of variables of F.

We observe that (3) is a polynomial parameter reduction from SAT, parameterized by the number of variables, to the Consistency problem for EGC, parameterized by the number of holes. Hence the theorem follows from (1) and (2). ☐

In recent research, Szeider [2011] showed superpolynomial kernel lower bounds for other parameteriztions of global constraints as considered by Bessière *et al.* [2008].

5 Conclusion

We have introduced the concept of kernelization to the field of constraint processing, providing both positive and negative results for the important global constraints NVALUE and EGC, respectively. On the positive side, we have developed an efficient linear-time kernelization algorithm for the consistency problem for ATMOST-NVALUE constraints, and have shown how it can be used to speed up the complete propagation of NVALUE and related constraints. On the negative side, we have established a theoretical result which indicates that EGC constraints do not admit polynomial kernels.

Our algorithms are efficient and the theoretical worst-case time bounds do not include large hidden constants. We therefore believe that the algorithms are practical, but we must leave an empirical evaluation for future research. We hope that our results stimulate further research on kernelization algorithms for constraint processing.

References

[Beldiceanu, 2001] N. Beldiceanu. Pruning for the minimum constraint family and for the number of distinct values constraint family. In *CP 01*, pp. 211–224, 2001.

[Bessière *et al.*, 2004] C. Bessière, E. Hebrard, B. Hnich, and T. Walsh. The complexity of global constraints. In *IAAI 04*, pp. 112–117, 2004.

[Bessière *et al.*, 2006] C. Bessière, E. Hebrard, B. Hnich, Z. Kiziltan, and T. Walsh. Filtering algorithms for the NValue constraint. *Constraints*, 11(4):271–293, 2006.

[Bessière *et al.*, 2008] C. Bessière, E. Hebrard, B. Hnich, Z. Kiziltan, C.-G. Quimper, and T. Walsh. The parameterized complexity of global constraints. In *AAAI 08*, pp. 235–240, 2008.

[Bessière, 2006] C. Bessière. Constraint propagation. In *Handbook of Constraint Programming*, chapter 3. Elsevier, 2006.

[Bodlaender *et al.*, 2009a] H. L. Bodlaender, R. G. Downey, M. R. Fellows, and D. Hermelin. On problems without polynomial kernels. *J. Comput. Syst. Sci.*, 75(8):423–434, 2009.

[Bodlaender *et al.*, 2009b] H. L. Bodlaender, S. Thomassé, and A. Yeo. Kernel bounds for disjoint cycles and disjoint paths. In *ESA 09*, Springer *LNCS* 5757, pp. 635–646, 2009.

[Downey and Fellows, 1999] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1999.

[Fellows, 2006] M. R. Fellows. The lost continent of polynomial time: Preprocessing and kernelization. In *IWPEC 06*, Springer *LNCS* 4169, pp. 276–277, 2006.

[Fomin and Kratsch, 2010] F. V. Fomin and D. Kratsch. *Exact Exponential Algorithms*. Springer, 2010.

[Fomin, 2010] F. V. Fomin. Kernelization. In *CSR 10*, Springer *LNCS* 6072, pp. 107–108, 2010.

[Fortnow and Santhanam, 2008] L. Fortnow and R. Santhanam. Infeasibility of instance compression and succinct PCPs for NP. In *STOC 08*, pp. 133–142, 2008.

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

[Pachet and Roy, 1999] F. Pachet and P. Roy. Automatic generation of music programs. In *CP 99*, pp. 331–345. Springer, 1999.

[Quimper *et al.*, 2004] C.-G. Quimper, A. López-Ortiz, P. van Beek, and A. Golynski. Improved algorithms for the global cardinality constraint. In *CP 04*, pp. 542–556, 2004.

[Régin, 1996] J.-C. Régin. Generalized arc consistency for global cardinality constraint. In *AAAI 96*, vol. 1, pp. 209–215, 1996.

[Rosamond, 2010] F. Rosamond. Table of races. In *Parameterized Complexity Newsletter*, pp. 4–5. 2010. http://fpt.wikidot.com/.

[Rossi *et al.*, 2006] F. Rossi, P. van Beek, and T. Walsh, editors. *Handbook of Constraint Programming*. Elsevier, 2006.

[Smith, 2006] B. M. Smith. Modelling. In *Handbook of Constraint Programming*, chapter 11. Elsevier, 2006.

[Szeider, 2011] S. Szeider. Limits of Preprocessing. In *AAAI 11*, 2011.

[van Hoeve and Katriel, 2006] W.-J. van Hoeve and I. Katriel. Global constraints. In *Handbook of Constraint Programming*, chapter 6. Elsevier, 2006.