Decomposition of the NVALUE constraint

Christian Bessiere[1], George Katsirelos[2], Nina Narodytska[2], Claude-Guy Quimper[3], and Toby Walsh[2]

[1] LIRMM, CNRS, Montpellier, email: bessiere@lirmm.fr
[2] NICTA and University of NSW, Sydney, Australia, email: {george.katsirelos,nina.narodytska,toby.walsh}@nicta.com.au
[3] École Polytechnique de Montréal, email: cquimper@gmail.com

Abstract. We study decompositions of NVALUE, a global constraint that can be used to model a wide range of problems where values need to be counted. Such decompositions are useful as few constraints toolkits provide propagators for this global constraint. One of our decompositions maintains a global view as enforcing bound consistency on the decomposition achieves bound consistency on the original global constraint. Our experiments demonstrate that this decomposition is efficient enough to be used in practice.

1 Introduction

Global constraints are an important feature of constraint toolkits. They capture common patterns in real world problems, and provide efficient propagators for pruning the search space. Whilst many propagators have been proposed in the literature, most toolkits only contain a few propagation algorithms. Implementing propagators is a difficult and time-consuming job, best undertaken by those familiar with the internals of the particular constraint toolkit. Consider, for example, the NVALUE constraint which counts the number of values used by a set of variables [1]. This is a common constraint that can be used to model many practical problems such as timetabling and frequency allocation. At least four different propagation algorithms for the NVALUE constraint have been proposed [2–4]. However, whilst the NVALUE constraint is supported by at least one constraint solver (namely SICStus Prolog), it is not supported in the latest release of BProlog, ECLiPSe, FaCiLe, Gecode, GNU Prolog, Minion or ILOG Solver.

An attractive alternative to implementing a specialized propagation algorithm is to develop a decomposition of the global constraint which efficiently simulates (some or all of) the pruning of the propagator. For instance, we have recently shown that bound consistency propagators for the ALL-DIFFERENT and GCC constraints can be efficiently simulated by some carefully designed decompositions [5]. We now turn to a global constraint that is less supported by constraint toolkits. We show that decompositions can simulate a complex algorithm for enforcing bound consistency on the NVALUE constraint. Decompositions of global constraints like these offer a number of advantages. For example, they can be easily added to a new solver. As a second example, decompositions can suggest small nogoods. They can work well therefore with learning based heuristics.

2 Background

We assume values are taken from the set 1 to d. We write $dom(X_i)$ for the domain of possible values for X_i, $min(X_i)$ for the smallest value in $dom(X_i)$, $max(X_i)$ for the greatest, and $range(X_i)$ for the interval $[min(X_i), max(X_i)]$. A *global constraint* is one in which the number of variables n is a parameter. For instance, the global NVALUE$([X_1, \ldots, X_n], N)$ constraint ensures that $N = |\{X_i \mid 1 \leq i \leq n\}|$ [1]. Constraint solvers typically use backtracking search to explore the space of partial assignments. After each assignment, propagation algorithms prune the search space by enforcing local consistency properties like domain, range 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 an assignment is called a *support*. A constraint is *disentailed* iff there is no possible support. A constraint is *range consistent* (*RC*) iff, when a variable is assigned any of the values in its domain, there exist compatible values between the minimum and maximum domain value for all the other variables of the constraint. Such an assignment is called a *bound support*. A constraint is *bound consistent* (*BC*) iff the minimum and maximum value of every variable of the constraint belong to a bound support. We will compare local consistency properties applied to sets of constraints, c_1 and c_2 which are logically equivalent. As in [6], a local consistency property Φ on c_1 is as strong as Ψ on c_2 iff, given any domains, if Φ holds on c_1 then Ψ holds on c_2; Φ on c_1 is stronger than Ψ on c_2 iff Φ on c_1 is as strong as Ψ on c_2 but not vice versa; Φ on c_1 is equivalent to Ψ on c_2 iff Φ on c_1 is as strong as Ψ on c_2 and vice versa. Finally, as constraint solvers usually enforce local consistency after each assignment down a branch in the search tree, we will compute the total amortised cost of enforcing a local consistency down an entire branch of the search tree. This captures the incremental cost of propagation.

3 NVALUE constraint

Pachet and Roy proposed the NVALUE constraint (called by them the "cardinality on attribute values" constraint) to model a combinatorial problem in selecting musical play-lists [1]. It can also be used to model the number of frequencies used in a frequency allocation problem or the number of rooms needed to timetable a set of exams. It generalizes several other global constraints including ALL-DIFFERENT (which ensures a set of variables take all different values) and NOT-ALL-EQUAL (which ensures a set of variables do not all take the same value). Enforcing domain consistency on the NVALUE constraint is NP-hard (Theorem 3 in [7]) even when N is fixed (Theorem 2 in [3]). In fact, computing the lower bound on N is NP-hard (Theorem 3 in [8]). In addition, enforcing domain consistency on the NVALUE constraint is not fixed parameter tractable since it is in the $W[2]$-complete complexity class along with problems like minimum hitting set (Theorem 2 in [9]). However, a number of polynomial propagation algorithms have been proposed that achieve bound consistency and some closely related levels of local consistency [2–4]. Unfortunately, as indicated earlier, very few constraint toolkits provide implementations of these propagators.

3.1 Simple decomposition

If a global constraint is not supported, one option is to decompose it into primitive constraints that are. For instance, we can decompose the NVALUE constraint by introducing 0/1 variables and posting the following constraints:

$$X_i = j \rightarrow B_j = 1 \qquad \forall 1 \leq i \leq n, 1 \leq j \leq d \tag{1}$$
$$B_j = 1 \rightarrow \bigvee_{i=1}^{n} X_i = j \quad \forall 1 \leq j \leq d \tag{2}$$
$$\sum_{j=1}^{m} B_j = N \tag{3}$$

Unfortunately, this simple decomposition hinders propagation. It can be BC whereas BC on the corresponding NVALUE constraint detects disentailment.

Theorem 1 *BC on* NVALUE *is stronger than BC on its decomposition into (1) to (3).*

Proof: Clearly BC on NVALUE is at least as strong as BC on the decomposition. To show strictness, consider $X_1 \in \{1, 2\}$, $X_2 \in \{3, 4\}$, $B_j \in \{0, 1\}$ for $1 \leq j \leq 4$, and $N = 1$. Constraints (1) to (3) are BC. However, the corresponding NVALUE constraint has no bound support and thus enforcing BC on it detects disentailment. □

We observe that enforcing DC instead of BC on constraints (1) to (3) in the example of the proof above still does not prune any value. To decompose NVALUE without hindering propagation, we must look to more complex decompositions.

3.2 Decomposition into ATMOSTNVALUE and ATLEASTNVALUE

Our first step in decomposing the NVALUE constraint is to split it into two parts: an ATMOSTNVALUE and an ATLEASTNVALUE constraint. ATLEASTNVALUE$([X_1, \ldots, X_n], N)$ holds iff $N \leq |\{X_i | 1 \leq i \leq n\}|$ whilst ATMOSTNVALUE$([X_1, \ldots, X_n], N)$ holds iff $|\{X_i | 1 \leq i \leq n\}| \leq N$.

Running Example. *Consider a* NVALUE *constraint over the following variables and values:*

	1	2	3	4	5
X_1	*	*	*		*
X_2		*			
X_3		*	*	*	
X_4				*	
X_5			*	*	
N	*	*			*

Suppose we decompose this into an ATMOSTNVALUE *and an* ATLEASTNVALUE *constraint. Consider the* ATLEASTNVALUE *constraint. The 5 variables can take at most 4 different values because X_2, X_3, X_4, and X_5 can only take values $2, 3$ and 4. Hence, there is no bound support for $N = 5$. Enforcing BC on the* ATLEASTNVALUE *constraint therefore prunes $N = 5$. Consider now the* ATMOSTNVALUE *constraint. Since X_2 and X_4 guarantee that we take at least 2 different values, there is no bound support for $N = 1$. Hence enforcing BC on an* ATMOSTNVALUE *constraint prunes $N = 1$. If*

$X_1 = 1$, 3 *or* 5*, or* $X_5 = 3$ *then any complete assignment uses at least 3 different values. Hence there is also no bound support for these assignments. Pruning these values gives bound consistent domains for the original* NVALUE *constraint:*

	1	2	3	4	5
X_1	*				
X_2	*				
X_3	*	*	*		
X_4				*	
X_5				*	
N	*				

To show that decomposing the NVALUE constraint into these two parts does not hinder propagation in general, we will use the following lemma. Given an assignment S of values, $card(S)$ denotes the number of distinct values in S. Given a vector of variables $X = X_1 \dots X_n$, $card_{\uparrow}(X) = max\{card(S) \mid S \in \Pi_{X_i \in X} range(X_i)\}$ and $card_{\downarrow}(X) = min\{card(S) \mid S \in \Pi_{X_i \in X} range(X_i)\}$.

Lemma 1 (adapted from [4]) *Consider* NVALUE$([X_1, \dots, X_n], N)$. *If* $D(N) \subseteq [card_{\downarrow}(X), card_{\uparrow}(X)]$*, then* N *is BC.*

Proof: Let S_{min} be an assignment of X in $\Pi_{X_i \in X} range(X_i)$ with $card(S_{min}) = card_{\downarrow}(X)$ and S_{max} be an assignment of X in $\Pi_{X_i \in X} range(X_i)$ with $card(S_{max}) = card_{\uparrow}(X)$. Consider the sequence $S_{min} = S_0, S_1, \dots, S_n = S_{max}$ where S_{k+1} is the same as S_k except that X_{k+1} has been assigned its value in S_{max} instead of its value in S_{min}. $|card(S_{k+1}) - card(S_k)| \leq 1$ because they only differ on X_{k+1}. Hence, for any $p \in [card_{\downarrow}(X), card_{\uparrow}(X)]$, there exists $k \in 1..n$ with $card(S_k) = p$. Thus, (S_k, p) is a bound support for p on NVALUE$([X_1, \dots, X_n], N)$. Therefore, $min(N)$ and $max(N)$ have a bound support. □

We now prove that decomposing the NVALUE constraint into ATMOSTNVALUE and ATLEASTNVALUE constraints does not hinder pruning when enforcing BC.

Theorem 2 *BC on* NVALUE$([X_1, \dots, X_n], N)$ *is equivalent to BC on* ATMOSTNVALUE$([X_1, \dots, X_n], N)$ *and on* ATLEASTNVALUE$([X_1, \dots, X_n], N)$.

Proof: Suppose the ATMOSTNVALUE and ATLEASTNVALUE constraints are BC. The ATMOSTNVALUE constraint guarantees that $card_{\downarrow}(X) \leq min(N)$ and the ATLEASTNVALUE constraint guarantees that $card_{\uparrow}(X) \geq max(N)$. Therefore, $D(N) \in [card_{\downarrow}(X), card_{\uparrow}(X)]$. By Lemma 1, the variable N is bound consistent.

Consider a variable/bound value pair $X_i = b$. Let (S^b_{least}, p_1) be a bound support of $X_i = b$ in the ATLEASTNVALUE constraint and (S^b_{most}, p_2) be a bound support of $X_i = b$ in the ATMOSTNVALUE constraint. We have $card(S^b_{least}) \geq p_1$ and $card(S^b_{most}) \leq p_2$ by definition of ATLEASTNVALUE and ATMOSTNVALUE. Consider the sequence $S^b_{least} = S^b_0, S^b_1, \dots, S^b_n = S^b_{most}$ where S^b_{k+1} is the same as S^b_k except that X_{k+1} has been assigned its value in S^b_{most} instead of its value in S^b_{least}. $|card(S^b_{k+1}) - card(S^b_k)| \leq 1$ because they only differ on X_{k+1}. Hence, there exists $k \in 1..n$ with $min(p_1, p_2) \leq card(S^b_k) \leq max(p_1, p_2)$. We know that p_1 and p_2 belong to $range(N)$ because they belong to bound supports. Thus, $card(S^b_k) \in range(N)$ and $(S^b_k, card(S^b_k))$ is a bound support for $X_i = b$ on NVALUE$([X_1, \dots, X_n], N)$. □

When enforcing domain consistency, Bessiere *et al.* [4] noted that decomposing the NVALUE constraint into ATMOSTNVALUE and ATLEASTNVALUE constraints does hinder propagation, but only when $dom(N)$ contains just $card_{\downarrow}(X)$ and $card_{\uparrow}(X)$ and there is a gap in the domain in-between (see Theorem 1 in [4] and the discussion that follows). When enforcing BC, any such gap in the domain for N is ignored.

4 ATMOSTNVALUE constraint

We now give a decomposition for the ATMOSTNVALUE constraint which does not hinder bound consistency propagation. To decompose the ATMOSTNVALUE constraint, we introduce 0/1 variables, A_{ilu} to represent whether X_i uses a value in the interval $[l, u]$, and "pyramid" variables, M_{lu} with domains $[0, \min(u - l + 1, n)]$ which count the number of values taken inside the interval $[l, u]$. To constrain these introduced variables, we post the following constraints:

$$A_{ilu} = 1 \iff X_i \in [l, u] \quad \forall\, 1 \leq i \leq n, 1 \leq l \leq u \leq d \tag{4}$$

$$A_{ilu} \leq M_{lu} \quad \forall\, 1 \leq i \leq n, 1 \leq l \leq u \leq d \tag{5}$$

$$M_{1u} = M_{1k} + M_{(k+1)u} \quad \forall\, 1 \leq k < u \leq d \tag{6}$$

$$M_{1d} \leq N \tag{7}$$

Running Example. *Consider the decomposition of an* ATMOSTNVALUE *constraint over the following variables and values:*

	1	2	3	4	5
X_1	*	*	*		*
X_2		*			
X_3		*	*	*	
X_4				*	
X_5			*	*	
N	*	*			

Observe that we consider that value 5 for N has already been pruned by ATLEASTNVALUE*, as will be shown in next sections. Bound consistency reasoning on the decomposition will make the following inferences. As $X_2 = 2$, from* (4) *we get $A_{222} = 1$. Hence by* (5)*, $M_{22} = 1$. Similarly, as $X_4 = 4$, we get $A_{444} = 1$ and $M_{44} = 1$. Now $N \in \{1, 2\}$. By* (7) *and* (6)*, $M_{15} \leq N$, $M_{15} = M_{14} + M_{55}$, $M_{14} = M_{13} + M_{44}$, $M_{13} = M_{12} + M_{33}$, $M_{12} = M_{11} + M_{22}$. Since $M_{22} = M_{44} = 1$, we deduce that $N > 1$ and hence $N = 2$. This gives $M_{11} = M_{33} = M_{55} = 0$. By* (5)*, $A_{111} = A_{133} = A_{155} = A_{533} = 0$. Finally, from* (4)*, we get $X_1 = 2$ and $X_5 = 3$. This gives us bound consistent domains for the* ATMOSTNVALUE *constraint.*

We now prove that this decomposition does not hinder propagation in general.

Theorem 3 *BC on constraints (4) to (7) is equivalent to BC on* ATMOSTNVALUE *$([X_1, \ldots, X_n], N)$, and takes $O(nd^3)$ time to enforce down the branch of the search tree.*

Proof: First note that changing the domains of the X variables cannot affect the upper bound of N by the ATMOSTNVALUE constraint and, conversely, changing the lower bound of N cannot affect the domains of the X variables.

Let $Y = \{X_{p_1}, \ldots, X_{p_k}\}$ be a maximum cardinality subset of variables of X whose ranges are pairwise disjoint (i.e., $range(X_{p_i}) \cap range(X_{p_j}) = \emptyset, \forall i, j \in 1..k, i \neq j$). Let $I_Y = \{[b_i, c_i] \mid b_i = min(X_{p_i}),\ c_i = max(X_{p_i}), X_{p_i} \in Y\}$ be the corresponding ordered set of disjoint ranges of the variables in Y. It has been shown in [3] that $|Y| = card_{\downarrow}(X)$.

Consider the interval $[b_i, c_i] \in I_Y$. Constraints (5) ensure that the variables $M_{b_i c_i}$ $i = [1, \ldots, k]$ are greater than or equal to 1 and constraints (6) ensure that the variable M_{1d} is greater than or equal to the sum of lower bounds of variables $M_{b_i c_i}$, $i = [1, \ldots, k]$, because intervals $[b_i, c_i]$ are disjoint. Therefore, the variable N is greater than or equal to $card_{\downarrow}(X)$ and it is bound consistent.

We show that when N is BC and $dom(N) \neq \{card_{\downarrow}(X)\}$, all X variables are BC. Take any assignment $S \in \Pi_{X_i \in X} range(X_i)$ such that $card(S) = card_{\downarrow}(X)$. Let $S[X_i \leftarrow b]$ be the assignment S where the value of X_i in S has been replaced by b, one of the bounds of X_i. We know that $card(S[X_i \leftarrow b]) \in [card(S) - 1, card(S) + 1] = [card_{\downarrow}(X) - 1, card_{\downarrow}(X) + 1]$ because only one variable has been flipped. Hence, any assignment (S, p) with $p \geq card_{\downarrow}(X) + 1$ is a bound support. $dom(N)$ necessarily contains such a value p by assumption.

The only case when pruning might occur is if the variable N is ground and $card_{\downarrow}(X) = N$. Constraints (6) imply that M_{1d} equals the sum of variables $M_{1,b_1-1} + M_{b_1,c_1} + M_{c_1+1,b_2-1} \ldots + M_{b_N,c_N} + M_{c_N+1,d}$. The lower bound of the variable M_{c_i,b_i} is greater than one and there are $|Y| = card_{\downarrow}(X) = N$ of these intervals. Therefore, by constraint (7), the upper bound of variables $M_{c_{i-1}+1,b_i-1}$ that correspond to intervals outside the set I_Y are forced to zero.

There are $O(nd^2)$ constraints (4) and constraints (5) that can be woken $O(d)$ times down the branch of the search tree. Each requires $O(1)$ time for a total of $O(nd^3)$ down the branch. There are $O(d^2)$ constraints (6) which can be woken $O(n)$ times down the branch and each invocation takes $O(1)$ time. This gives a total of $O(nd^2)$. The final complexity down the branch of the search tree is therefore $O(nd^3)$. □

5 Faster decompositions

We can improve how our solver handles this decomposition of the ATMOSTNVALUE constraint by adding implied constraints and by implementing specialized propagators. Our first improvement is to add an implied constraint and enforce BC on it:

$$M_{1d} = \sum_{i=1}^{d} M_{ii} \tag{8}$$

This does not change the asymptotic complexity of reasoning with the decomposition, nor does it improve the level of propagation achieved. However, we have found that the fixed point of propagation is reached quicker in practice with such an implied constraint.

Our second improvement decreases the asymptotic complexity of enforcing BC on the decomposition of Section 4. The complexity is dominated by reasoning with constraints (4) which channel from X_i to A_{ilu} and thence onto M_{lu} (through constraints (5)). If constraints (4) were not woken uselessly, enforcing BC should cost $O(1)$ per constraint down the branch. Unfortunately, existing solvers wake up such constraints as soon as a bound is modified, thus a cost in $O(d)$. We therefore implemented a specialized propagator to channel between X_i and M_{lu} efficiently. To be more precise, we remove the $O(nd^2)$ variables A_{ilu} and replace them with $O(nd)$ Boolean variables Z_{ij}. We then add the following constraints

$$Z_{ij} = 1 \iff X_i \leq j \qquad 1 \leq j \leq d \tag{9}$$

$$Z_{i(l-1)} = 1 \vee Z_{iu} = 0 \vee M_{lu} > 0 \qquad 1 \leq l \leq u \leq d, 1 \leq i \leq n \tag{10}$$

These constraints are enough to channel changes in the bounds of the X variables to M_{lu}. There are $O(nd)$ constraints (9), each of which can be propagated in time $O(d)$ over a branch, for a total of $O(nd^2)$. There are $O(nd^2)$ clausal constraints (10) and each of them can be made BC in time $O(1)$ down a branch of the search tree, for a total cost of $O(nd^2)$. Since channeling dominates the asymptotic complexity of the entire decomposition of Section 4, this improves the complexity of this decomposition to $O(nd^2)$. This is similar to the technique used in [5] to improve the asymptotic complexity of the decomposition of the ALL-DIFFERENT constraint.

Our third improvement is to enforce stronger pruning by observing that when $M_{lu} = 0$, we can remove the interval $[l, u]$ from all variables, regardless of whether this modifies their bounds. This corresponds to enforcing RC on constraints (4). Interestingly, this is sufficient to achieve RC on the ATMOSTNVALUE constraint. Unfortunately, constraints (10) cannot achieve this pruning and using constraints (4) increases the complexity of the decomposition back to $O(nd^3)$. We do it by extending the decomposition with $O(d \log d)$ Boolean variables $B_{il(l+2^k)} \in [0, 1], 1 \leq i \leq n, 1 \leq l \leq d, 0 \leq k \leq \lfloor \log d \rfloor$. The following constraint ensures that $B_{ijj} = 1 \iff X_i = j$.

$$\text{DOMAINBITMAP}(X_i, [B_{i11}, \ldots, B_{idd}]) \tag{11}$$

Clearly we can enforce RC on this constraint in time $O(d)$ over a branch, and $O(nd)$ for all variables X_i. We can then use the following clausal constraints to channel from variables M_{lu} to these variables and on to the X variables. These constraints are posted for every $1 \leq i \leq n, 1 \leq l \leq u \leq d, 1 \leq j \leq d$ and integers k such that $0 \leq k \leq \lfloor \log d \rfloor$:

$$B_{ij(j+2^{k+1}-1)} = 1 \vee B_{ij(j+2^k-1)} = 0 \tag{12}$$

$$B_{ij(j+2^{k+1}-1)} = 1 \vee B_{i(j+2^k)(j+2^{k+1}-1)} = 0 \tag{13}$$

$$M_{lu} \neq 0 \vee B_{il(l+2^k-1)} = 0 \qquad 2^k \leq u - l + 1 < 2^{k+1} \tag{14}$$

$$M_{lu} \neq 0 \vee B_{i(u-2^k+1)u} = 0 \qquad 2^k \leq u - l + 1 < 2^{k+1} \tag{15}$$

The variable $B_{il(l+2^k-1)}$, similarly to the variables A_{lu}, is true when $X_i \in [l, l + 2^k - 1]$, but instead of having one such variable for every interval, we only have them for

intervals whose length is a power of two. When $M_{lu} = 0$, with $2^k \leq u - l + 1 < 2^{k+1}$, the constraints (14)–(15) set to 0 the B variables that correspond to the two intervals of length 2^k that start at l and finish at u, respectively. In turn, the constraints (12)–(13) set to 0 the B variables that correspond to intervals of length 2^{k-1}, all the way down to intervals of size 1. These trigger the constraints (11), so all values in the interval $[l, u]$ are removed from the domains of all variables.

Example. *Suppose* $X_1 \in [5, 9]$. *Then, by* (9)*,* $Z_{14} = 0$*,* $Z_{19} = 1$ *and by* (10)*,* $M_{59} > 0$*. Conversely, suppose* $M_{59} = 0$ *and* $X_1 \in [1, 10]$*. Then, by (14)–(15), we get* $B_{158} = 0$ *and* $B_{169} = 0$*. From* $B_{158} = 0$ *and (12)–(13) we get* $B_{156} = 0$*,* $B_{178} = 0$*,* $B_{155} = B_{166} = B_{177} = B_{188} = 0$*, and by* (11)*, the interval* $[5, 8]$ *is pruned from* X_1*. Similarly,* $B_{169} = 0$ *causes the interval* $[6, 9]$ *to be removed from* X_1*, so* $X_1 \in [1, 4] \cup \{10\}$.

Note that RC can be enforced on each of these constraints in constant time over a branch. There exist $O(nd \log d)$ of the constraints (12)–(13) and $O(nd^2)$ of the constraints (14)–(15), so the total time to propagate them all down a branch is $O(nd^2)$.

6 ATLEASTNVALUE constraint

There is a similar decomposition for the ATLEASTNVALUE constraint. We introduce 0/1 variables, A_{ilu} to represent whether X_i uses a value in the interval $[l, u]$, and integer variables, E_{lu} with domains $[0, n]$ to count the number of times values in $[l, u]$ are *re*-used, that is, how much the number of variables taking values in $[l, u]$ exceeds the number $u - l + 1$ of values in $[l, u]$. To constrain these introduced variables, we post the following constraints:

$$A_{ilu} = 1 \iff X_i \in [l, u] \qquad \forall\, 1 \leq i \leq n, 1 \leq l \leq u \leq d \tag{16}$$

$$E_{lu} \geq \textstyle\sum_{i=1}^{n} A_{ilu} - (u - l + 1) \qquad \forall\, 1 \leq l \leq u \leq d \tag{17}$$

$$E_{1u} = E_{1k} + E_{(k+1)u} \qquad \forall\, 1 \leq k < u \leq d \tag{18}$$

$$N \leq n - E_{1d} \tag{19}$$

Running Example. *Consider the decomposition of an* ATLEASTNVALUE *constraint over the following variables and values:*

	1	2	3	4	5
X_1	*	*	*		*
X_2		*			
X_3		*	*	*	
X_4				*	
X_5			*	*	
N	*	*			*

Bound consistency reasoning on the decomposition will make the following inferences. As $dom(X_i) \subseteq [2, 4]$ *for* $i \in 2..5$*, from* (16) *we get* $A_{i24} = 1$ *for* $i \in 2..5$*. Hence, by* (17)*,* $E_{24} \geq 1$*. By* (18)*,* $E_{15} = E_{14} + E_{55}$*,* $E_{14} = E_{11} + E_{24}$*. Since* $E_{24} \geq 1$ *we deduce that* $E_{15} \geq 1$*. Finally, from* (19) *and the fact that* $n = 5$*, we get* $N \leq 4$*. This gives us bound consistent domains for the* ATLEASTNVALUE *constraint.*

We now prove that this decomposition does not hinder propagation in general.

Theorem 4 *BC on the constraints* (16) *to* (19) *is equivalent to BC on* ATLEASTNVALUE $([X_1, \ldots, X_n], N)$*, and takes* $O(nd^3)$ *time to enforce down the branch of the search tree.*

Proof: First note that changing the domains of the X variables cannot affect the lower bound of N by the ATLEASTNVALUE constraint and, conversely, changing the upper bound of N cannot affect the domains of the X variables.

It is known [2] that $card_{\uparrow}(X)$ is equal to the size of a maximum matching M in the value graph of the constraint. Since $N \leq n - E_{1d}$, we show that the lower bound of E_{1d} is equal to $n - |M|$.[4] We first show that we can construct a matching $M(E)$ of size $n - min(E_{1d})$, then show that it is a maximum matching. The proof uses a partition of the interval $[1, d]$ into a set of maximal saturated intervals $I = \{[b_j, c_j]\}$, $j = 1, \ldots, k$ such that $min(E_{b_j,c_j}) = \sum_{i=1}^{n} min(A_{ib_jc_j}) - (c_j - b_j + 1)$ and a set of unsaturated intervals $\{[b_j, c_j]\}$such that $min(E_{b_j,c_j}) = 0$.

Let $I = \{[b_j, c_j] \mid j \in [1 \ldots k]\}$ be the ordered set of maximal intervals such that $min(E_{b_j,c_j}) = \sum_{i=1}^{n} min(A_{ib_jc_j}) - (c_j - b_j + 1)$. Note that the intervals in I are disjoint otherwise intervals are not maximal. An interval $[b_i, c_i]$ is smaller than $[b_j, c_j]$ iff $c_i < b_j$. We denote the union of the first j intervals $D_I^j = \bigcup_{i=1}^{j} [b_i, c_i]$, $j = [1, \ldots, k]$, $p = |D_I^k|$ and the variables whose domain is inside one of intervals I $X_I = \{X_{p_i} | D(X_{p_i}) \subseteq D_I^k\}$.

Our construction of a matching uses two sets of variables, X_I and $X \setminus X_I$. First, we identify the cardinality of these two sets. Namely, we show that the size of the set X_I is $p + min(E_{1,d})$ and the size of the set $X \setminus X_I$ is $n - (p + min(E_{1,d}))$.

Intervals I are saturated therefore each value from these intervals are taken by a variable in X_I. Therefore, X_I has size at least p. Moreover, there exist $min(E_{1d})$ additional variables that take values from D_I^k, because values from intervals between two consecutive intervals in I do not contribute to the lower bound of the variable E by construction of I. Therefore, the number of variables in D_I^k is at least $p + min(E_{1,d})$. Note that constraints (18) imply that E_{1d} equals the sum of variables $E_{1,b_1-1} + E_{b_1,c_1} + E_{c_1+1,b_2-1} \ldots + E_{b_k,c_k} + E_{c_k+1,d}$. As intervals in I are disjoint then $\sum_{i=1}^{k} min(E_{b_i,c_i}) = |X_I| - p$. If $|X_I| > p + min(E_{1,d})$ then $\sum_{i=1}^{k} min(E_{b_i,c_i}) > min(E_{1,d})$ and the lower bound of the variable E_{1d} will be increased. Hence, $|X_I| = p + min(E_{1,d})$.

Since all these intervals are saturated, we can construct a matching M_I of size p using the variables in X_I. The size of $X \setminus X_I$ is $n - p - min(E_{1d})$. We show by contradiction that we can construct a matching $M_{D-D_I^k}$ of size $n - p - min(E_{1d})$ using the variables in $X \setminus X_I$ and the values $D - D_I^k$.

Suppose such a matching does not exist. Then, there exists an interval $[b, c]$ such that $|(D \setminus D_I^k) \cap [b, c]| < \sum_{i \in X \setminus X_I} min(A_{ibc})$, i.e., after consuming the values in I with variables in X_I, we are left with fewer values in $[b, c]$ than variables whose domain is contained in $[b, c]$. We denote $p' = |[b, c] \cap D_I^k|$, so that p' is the number of values inside the interval $[b, c]$ that are taken by variables in X_I. The total number of variables inside the interval $[b, c]$ is greater than or equal to $\sum_{i=1}^{n} min(A_{ibc})$. The total number of variables X_I inside the interval $[b, c]$ equals to $p' + min(E_{b,c})$. Therefore,

[4] We assume that E_{1d} is not pruned by other constraints.

$\sum_{i \in X \setminus X_I} min(A_{ibc}) \leq \sum_{i=1}^{n} min(A_{ibc}) - p' - min(E_{b,c})$. On the other hand, the number of values that are not taken by the variables X_I in the interval $[b, c]$ is $c - b + 1 - p'$. Therefore, we obtain the inequality $c - b + 1 - p' < \sum_{i=1}^{n} min(A_{ibc}) - p' - min(E_{b,c})$ or $min(E_{bc}) < \sum_{i=1}^{n} min(A_{ibc}) - (c - b + 1)$. By construction of I, $\sum_{i=1}^{n} min(A_{ibc}) - (c - b + 1) < min(E_{bc})$, otherwise the intervals in I that are subsets of $[b, c]$ are not maximal. This leads to a contradiction, so we can construct a matching $M(E)$ of size $n - min(E_{1d})$.

Now suppose that $M(E)$ is not a maximum matching. This means that $min(E_{1d})$ is overestimated by propagation on (16) and (19). Since $M(E)$ is not a maximum matching, there exists an augmenting path of $M(E)$, that produces M', such that $|M'| = |M(E)| + 1$. This new matching covers all the values that $M(E)$ covers and one additional value q. We show that q cannot belong to the interval $[1, d]$.

The value q cannot be in any interval in I, because all values in $[b_i, c_i] \in I$ are used by variables whose domain is contained in $[b_i, c_i]$. In addition, q cannot be in an interval $[b, c]$ between two consecutive intervals in I, because those intervals do not contribute to the lower bound of E_{1d}. Thus, M' cannot cover more values than $M(E)$ and they must have the same size, a contradiction.

We show that when N is BC and $dom(N) \neq \{card_{\uparrow}(X)\}$, all X variables are BC. Take any assignment $S \in \Pi_{X_i \in X} range(X_i)$ such that $card(S) = card_{\uparrow}(X)$. Let $S[X_i \leftarrow b]$ be the assignment S where the value of X_i in S has been replaced by b, one of the bounds of X_i. We know that $card(S[X_i \leftarrow b]) \in [card(S) - 1, card(S) + 1] = [card_{\uparrow}(X) - 1, card_{\uparrow}(X) + 1]$ because only one variable has been flipped. Hence, any assignment (S, p) with $p \leq card_{\uparrow}(X) - 1$ is a bound support. $dom(N)$ necessarily contains such a value p by assumption.

We now show that if $N = card_{\uparrow}(X)$, enforcing BC on the constraints (16)–(19) makes the variables X BC with respect to the ATLEASTNVALUE constraint. We first observe that in a bound support, variables X must take the maximum number of different values because $N = card_{\uparrow}(X)$. Hence, in a bound support, variables X that are not included in a saturated interval will take values outside any saturated interval they overlap and they all take different values. We recall that $min(E_{1d}) = n - |M| = n - card_{\uparrow}(X)$. Hence, by constraint (19), $E_{1d} = n - N$. We recall the the size of set X_I equals $p + E_{1d}$. Constraints (18) imply that E_{1d} equals the sum of variables $E_{1,b_1-1} + E_{b_1,c_1} + E_{c_1+1,b_2-1} \ldots + E_{b_k,c_k} + E_{c_k+1,d}$ and $\sum_{i=1}^{k} min(E_{b_i,c_i}) = |X_I| - p = min(E_{1d}) = max(E_{1d})$. Hence, by constraints (18), the upper bounds of all variables E_{b_i,c_i} that correspond to the saturated intervals are forced to $min(E_{b_i,c_i})$. Thus, by constraints (16) and (17), all variables in $X \setminus X_I$ have their bounds pruned if they belong to D_I^k. By constraints (18) again, the upper bounds of all variables E_{lu} that correspond to the unsaturated intervals are forced to take value 0, and all variables $E_{l'u'}$ with $[l', u'] \subseteq [l, u]$ are forced to 0 as well. Thus, by constraints (16) and (17), all variables in $X \setminus X_I$ have their bounds pruned if they belong to a Hall interval of other variables in $X \setminus X_I$. This is what BC on the ALL-DIFFERENT constraint does [5].

There are $O(nd^2)$ constraints (16) that can be woken $O(d)$ times down the branch of the search tree in $O(1)$, so a total of $O(nd^3)$ down the branch. There are $O(d^2)$ constraints (17) which can be propagated in time $O(n)$ down the branch for a $O(nd^2)$. There are $O(d^2)$ constraints (18) which can be woken $O(n)$ times each down the branch

for a total cost in $O(n)$ time down the branch. Thus a total of $O(nd^2)$. The final complexity down the branch of the search tree is therefore $O(nd^3)$. □

The complexity of enforcing BC on the ATLEASTNVALUE constraint can be improved to $O(nd^2)$ in way similar to that described in Section 5 and in [5].

7 Experimental results

To evaluate these decompositions, we performed experiments on two problem domains. We used the same problems as in a previous experimental comparison of propagators for the ATMOSTNVALUE constraint [4]. We ran experiments with Ilog Solver 6.2 on an Intel Xeon 4 CPU, 2.0 Ghz, 4Gb RAM.

7.1 Dominating set of the Queen's graph

The problem is to put the minimum number of queens on a $n \times n$ chessboard, so that each square either contains a queen or is attacked by one. This is equivalent to the dominating set problem of the Queen's graph. Each vertex in the Queen's graph corresponds to a square of the chessboard and there exists an edge between two vertices iff a queen from one square can attack a queen from the other square. To model the problem, we use a variable X_i for each square, and values from 1 to n^2 and post a single ATMOSTNVALUE$([X_1, \ldots, X_{n^2}], N)$ constraint. The value j belongs to $D(X_i)$ iff there exists an edge (i, j) in the Queen's graph or $j = i$. We use minimum domain variable ordering and a lexicographical value ordering. For $n \leq 120$, all minimum dominating sets for the Queen's problem are either of size $\lceil n/2 \rceil$ or $\lceil n/2 + 1 \rceil$ [10]. We therefore only solved instances for these two values of N.

We compare our decomposition with two simple decompositions of the ATMOSTNVALUE constraint. The first decomposition is the one described in Section 3.1 except that in constraint (3), we replace "$=$" by "$\leq$". We denote this decomposition $Occs$. The second decomposition is similar to the first one, but we use the cardinality variables of a GCC constraint to keep track of the used values. We call this decomposition $Occs_{gcc}$. The final two decompositions are variants of the decomposition described in Section 4, which we call $Pyramid_{BC}$ or $Pyramid_{RC}$ depending whether we enforce BC or RC on our decomposition. As explained in Section 5, we channel the variables X_i directly to the pyramid variables M_{lu} to avoid introducing many auxiliary variables A_{ilu} and we add the redundant constraint $\sum_{i=1}^{n^2} M_{ii} = M_{1,n^2}$ to the decomposition to speed up the propagation across the pyramid. For the decomposition that enforces RC, we did not fully implement the $O(nd^2)$ decomposition of Section 5, but rather a simple channeling propagator that achieves RC in $O(nd^3)$ on (4), but with better asymptotic constants than constraints (4). Finally, we re-implemented the ternary sum constraint $Z = X + Y$ in Ilog. This gave us about 30% speed up.

Results are presented in Table 1. Our decomposition performs better than the other two decompositions, both in runtime and in number of backtracks. We observe that BC and RC prune the same (i.e., same number of backtracks) on our decomposition but BC is faster on larger problems. It should be pointed out that our results are comparable with the results for the ATMOSTNVALUE bounds consistency propagator from [4].

Table 1. Backtracks and rumtime (in seconds) to solve the dominating set problem for the Queen's graph.

n	N	$Occs$		$Occs_{gcc}$		$Pyramid_{BC}$		$Pyramid_{RC}$	
		backtracks	time	backtracks	time	backtracks	time	backtracks	time
5	3	34	0.01	34	0.06	7	**0.00**	7	**0.00**
6	3	540	0.16	540	2.56	118	**0.03**	118	**0.03**
7	4	195,212	84.50	195,212	1681,21	83,731	**15.49**	83,731	21.21
8	5	390,717	255.64	390,717	8,568.35	256,582	**58.42**	256,582	89.30

Whilst our decomposition is not as efficient as the best results presented in that paper, our decomposition was easier to implement.

7.2 Random binary CSP problems

We also reproduced the set of experiments on random binary CSP problems from [4]. These problems can be described by four parameters. The number of variables n, the domain size d, the number of binary constraints m and the number of forbidden tuples in each binary constraint. The first three classes are hard problems at the phase transition in satisfiability. The last two classes are under-constrained problems. We add a single AtMostNValue constraint over all variables to bound the number of values N that can be used in a solution.

As in [4], we generated 500 instances for each of the following 5 classes:

- class A : $n = 100, d = 10, m = 250, t = 52, N = 8$
- class B : $n = 50, d = 15, m = 120, t = 116, N = 6$
- class C : $n = 40, d = 20, m = 80, t = 240, N = 6$
- class D : $n = 200, d = 15, m = 600, t = 85, N = 8$
- class E : $n = 60, d = 30, m = 150, t = 350, N = 6$

All instances are solved using the minimum domain variable ordering heuristic, a lexicographical value ordering and a timeout of 600 seconds. We use the same decompositions of the AtMostNValue constraint as in the experiments with the dominating set of the Queen's graph. Results are given in Table 2. On classes A, B, C (hard problems), our decomposition is faster than the other two decompositions and solves more instances whatever we use BC or RC. On classes D, E (under-constrained problems), enforcing BC on our decomposition does not prune the search space enough. This leads to a high number of backtracks and a significant slow down. Our decomposition with RC is again better than the other decompositions.

These experiments demonstrate that this new decomposition is efficient to use in practice. Of course, if the toolkit contains a specialized BC propagator for the NValue constraint, we will probably do best to use this. However, when the toolkit lacks such a propagator (as is often the case), it is reasonable to try out our decomposition.

Table 2. Randomly generated binary CSPs with an ATMOSTNVALUE constraint. For each class we give two lines of results. Line 1: number of instances solved in 600 sec (#solved), average backtracks on solved instances (#bt), average time on solved instances (time). Line 2: number of instances solved by all methods, average backtracks and time on these instances.

	$Occs$			$Occs_{gcc}$			$Pyramid_{BC}$			$Pyramid_{RC}$		
$Class$	#solved	#bt	time	#solved	#bt	time	#solved	#bt	time	#solved	#bt	time
A total solved	453	139,120	111.2	79	8,960	302.8	**463**	**168,929**	**101.8**	462	148,673	105.7
solved by all	79	8,960	7.1	79	8,960	302.8	79	**9,104**	**5.7**	79	8,739	6.3
B total solved	473	228,757	113.5	125	37,377	292.9	**492**	**224,862**	**89.0**	491	235,715	94.9
solved by all	125	7,377	17.6	125	37,377	292.9	125	**32,810**	**10.9**	125	32,110	12.2
C total solved	479	233,341	110.3	156	37,242	290.3	**492**	**234,915**	**79.5**	490	224,802	84.2
solved by all	156	37242	16.4	156	37,242	290.3	156	**32,184**	**9.7**	156	31,715	11.1
D total solved	482	8,306	6.0	456	207	14.9	416	168,021	24.2	**489**	**13,776**	**9.0**
solved by all	391	**195**	**0.2**	391	195	13.1	391	145,534	14.9	391	690	0.4
E total solved	**500**	331	0.3	**500**	331	5.1	**500**	4,252	0.4	**500**	**174**	**0.1**
solved by all	500	331	0.3	500	331	5.1	500	4,252	0.4	500	**174**	**0.1**
TOTALS												
Total solved/tried	2,387/2,500			1,316/2,500			2,363 /2,500			**2,432**/2,500		
Avg time for solved	67.0			87.5			59.364			**58.0**		
Avg bts for solved	120,303			8,700			163,473			**123,931**		

8 Other related work

Decompositions have been given for a number of other global constraints. For example, Beldiceanu *et al.* identify conditions under which global constraints specified as automata can be decomposed into signature and transition constraints without hindering propagation [12]. As a second example, many global constraints can be decomposed using ROOTS and RANGE which can themselves often be propagated effectively using simple decompositions [13]. As a third example, the REGULAR and CFG constraints can be decomposed without hindering propagation [14, 15]. As a fourth example, decompositions of the SEQUENCE constraint have been shown to be effective [16]. Most recently, we demonstrated that the ALL-DIFFERENT and GCC constraint can be decomposed into simple primitive constraints without hindering bound consistency propagation [5]. These decompositions also introduced variables to count variables using values in an interval. For example, the decomposition of ALL-DIFFERENT ensures that no interval has more variables taking values in the interval than the number of values in the interval. Using a circuit complexity lower bound, we also proved that there is no polynomial sized SAT decomposition of the ALL-DIFFERENT constraint (and therefore of its generalizations like NVALUE) on which unit propagation achieves domain consistency [17].

9 Conclusions

We have shown that NVALUE, a global constraint which is not supported in many constraint toolkits, can be decomposed into simple arithmetic constraints. This decomposition permits a global view to be maintained that achieves bound consistency. Our experiments demonstrate that this decomposition is an efficient and effective means to propagate the global NVALUE constraint. This decomposition would therefore be useful when a constraint toolkit lacks a specialized propagator. Decompositions of global constraints like this are interesting for a number of other reasons. First they provide fresh insight into the workings of specialized propagation algorithms. In this case, it is surprising that interval graph reasoning used to enforce bound consistency on the NVALUE can be simulated with simple arithmetic constraints. Second, such decompositions may make nogood learning easier to implement. We can identify compact nogoods with different parts of the decomposition. Third, the variables introduced in such decompositions give ths solver access to the state of the propagator. This may be useful when making branching decisions. Fourth, such decompositions can often be encoded effectively into SAT. We can thereby provide the power of global propagation algorithms to SAT solvers. Finally, we expect many other global constraints that count variables and values to be decomposable in similar ways. For instance, it is known that BC on the SOFTALLDIFFERENT constraint is equivalent to BC on the ATLEASTNVALUE constraint [4]. We are currently studying decompositions of other global constraints, such as the USED-BY [11] and other soft global constraints such as SOFTGCC [18].

Acknowledgements

NICTA is funded by the Australian Government's Department of Broadband, Communications and the Digital Economy, and the Australian Research Council through "Backing Australia's Ability" and the ICT Centre of Excellence programmes. Christian Bessiere is supported by the ANR project ANR-06-BLAN-0383-02.

References

1. Pachet, F., Roy, P.: Automatic generation of music programs. In Jaffar, J., ed.: Proceedings of Fifth International Conference on Principles and Practice of Constraint Programming (CP99), Springer (1999) 331–345
2. Beldiceanu, N.: Pruning for the minimum constraint family and for the number of distinct values constraint family. In Walsh, T., ed.: Proceedings of Seventh International Conference on Principles and Practice of Constraint Programming (CP2001), Springer (2001) 211–224
3. Bessiere, C., Hebrard, E., Hnich, B., Kiziltan, Z., Walsh, T.: Filtering algorithms for the NVALUE constraint. In: Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, 2nd International Conference (CPAIOR-2005). (2005)
4. Bessiere, C., Hebrard, E., Hnich, B., Kiziltan, Z., Walsh, T.: Filtering algorithms for the NVALUE constraint. Constraints **11** (2006) 271–293
5. Bessiere, C., Katsirelos, G., Narodytska, N., Quimper, C.G., Walsh, T.: Decompositions of all different, global cardinality and related constraints. In: Proceedings of 21st IJCAI, International Joint Conference on Artificial Intelligence (2009)

6. Debruyne, R., Bessiere, C.: Some practicable filtering techniques for the constraint satisfaction problem. In: Proceedings of the 15th IJCAI, International Joint Conference on Artificial Intelligence (1997) 412–417
7. Bessiere, C., Hebrard, E., Hnich, B., Walsh, T.: The complexity of global constraints. In: Proceedings of the 19th National Conference on AI, Association for Advancement of Artificial Intelligence (2004)
8. Bessiere, C., Hebrard, E., Hnich, B., Walsh, T.: The complexity of global constraints. Constraints **12** (2007) 239–259
9. Bessiere, C., Hebrard, E., Hnich, B., Kiziltan, Z., Quimper, C.G., Walsh, T.: The parameterized complexity of global constraints. In: Proceedings of the 23rd National Conference on AI, Association for Advancement of Artificial Intelligence (2008)
10. Östergård, P., Weakley, W.: Values of domination numbers of the queen's graph. The Electronic Journal of Combinatorics **8** (2001)
11. Beldiceanu, N., Katriel, I., Thiel, S.: Filtering algorithms for the same constraint. In: Proceedings of the First International Conference on Integration of AI and OR Techniques in Constraint Programming (CP-AI-OR), Springer-Verlag (2004) 65–79 LNCS 3011.
12. Beldiceanu, N., Carlsson, M., Debruyne, R., Petit, T.: Reformulation of Global Constraints Based on Constraints Checkers. Constraints **10** (2005) 339–362
13. Bessiere, C., Hebrard, E., Hnich, B., Kiziltan, Z., Walsh, T.: The Range and Roots constraints: Specifying counting and occurrence problems. In: Proceedings of 19th IJCAI, International Joint Conference on Artificial Intelligence (2005) 60–65
14. Quimper, C.G., Walsh, T.: Global grammar constraints. In: 12th International Conference on Principles and Practices of Constraint Programming (CP-2006), Springer-Verlag (2006)
15. Quimper, C.G., Walsh, T.: Decomposing global grammar constraints. In: 13th International Conference on Principles and Practices of Constraint Programming (CP-2007), Springer-Verlag (2007)
16. Brand, S., Narodytska, N., Quimper, C.G., Stuckey, P., Walsh, T.: Encodings of the sequence constraint. In: 13th International Conference on Principles and Practices of Constraint Programming (CP-2007), Springer-Verlag (2007)
17. Bessiere, C., Katsirelos, G., Narodytska, N., Walsh, T.: Circuit complexity and decompositions of global constraints. In: Proceedings of 21st IJCAI, International Joint Conference on Artificial Intelligence (2009)
18. van Hoeve, W.J., Pesant, G. and Rousseau, L. M.: On global warming: Flow-based soft global constraints. Journal of Heuristics **12** (2006), 4-5, p. 347–373.