Filtering Algorithms for the NValue Constraint

Christian Bessiere[1], Emmanuel Hebrard[2], Brahim Hnich[3], Zeynep Kiziltan[4], and Toby Walsh[2]

[1]LIRMM-CNRS
[2]NICTA and UNSW
[3]4C and UCC
[4]University of Bologna
bessiere@lirmm.fr, {ehebrard, tw}@cse.unsw.edu.au, brahim@4c.ucc.ie, zkiziltan@deis.unibo.it

Abstract. The constraint NValue counts the number of different values assigned to a vector of variables. Propagating generalized arc consistency on this constraint is NP-hard. We show that computing even the lower bound on the number of values is NP-hard. We therefore study different approximation heuristics for this problem. We introduce three new methods for computing a lower bound on the number of values. The first two are based on the *maximum independent set problem* and are incomparable to a previous approach based on intervals. The last method is a linear relaxation of the problem. This gives a tighter lower bound than all other methods, but at a greater asymptotic cost.

1 Introduction

The NValue constraint counts the number of distinct values used by a vector of variables. It is a generalization of the widely used AllDifferent constraint [12]. It was introduced in [4] to model a musical play-list configuration problem so that play-lists were either homogeneous (used few values) or diverse (used many). There are many other situations where the number of values (e.g., resources) used at the same time are limited. In such cases, a NValue constraint can aid both modelling and solving.

Enforcing generalized arc consistency (GAC) on the NValue constraint is NP-hard [3]. One way to deal with this intractability is to identify a tractable decomposition or approximation method. The NValue constraint can be decomposed into two other global constraints: the AtMostNValue and the AtleastNValue constraints. Unfortunately, while enforcing GAC on the AtLeastNValue constraint is polynomial, we show that enforcing GAC on the AtMostNValue constraint is also NP-hard. We will therefore focus on various approximation methods for propagating the AtMostNValue constraint.

We introduce three new approximations. Two are based on graph theory while the third exploit a linear relaxation encoding. We compare the level of filtering achieved with a previous approximation method due to Beldiceanu based on intervals that runs in $O(n\log(n))$ [1] for finding a lower bound on N, and linear for pruning values. We show that the two new algorithms based on graph theory are incomparable with Beldiceanu's, though one is strictly tighter than the other. Both algorithms, however, have a $O(n^2)$ time complexity. We also show that the linear relaxation method dominates all other approaches in terms of the filtering, but with a higher computational cost. Finally, we demonstrate how all of these methods can be used in a filtering algorithm for the NValue constraint.

2 Formal Background

2.1 Constraint satisfaction problems

A constraint satisfaction problem (CSP) consists of a set of variables, each with a finite domain of values, and a set of constraints that specify allowed combinations of values for subsets of variables. We use upper case for variables, X_i, or vectors of variables, $\bar{X}$, and lower case for values, v, or assignments, $\bar{v}$. The domain of a variable X_i, $D(X_i)$ is a set of values. A full or partial assignment $\bar{v} = \langle v_1, \ldots, v_m \rangle$ of $\bar{X} = \langle X_1, \ldots, X_m \rangle$ is a vector of values such that $v_i \in D(X_i)$. A solution to a CSP is a full assignment of values to the variables satisfying the constraints. The minimum (resp. maximum) value in the domain of a variable X_i is $min(X_i)$ (resp. $max(X_i)$). The cardinality of an assignment $\bar{v}$ is $card(\bar{v})$, the number of distinct values used. For instance if $\bar{v} = \langle a, b, a, b, c \rangle$, $card(\bar{v}) = 3$. The maximum (resp. minimum) cardinality of a vector of variables $\bar{X}$, $card{\uparrow}(\bar{X})$ (resp. $card{\downarrow}(\bar{X})$) is the largest (resp. smallest) cardinality among all possible assignments.

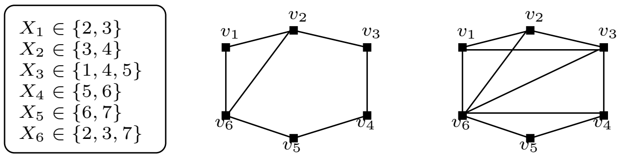
Constraint solvers typically explore partial assignments enforcing a local consistency property using either specialized or general purpose propagation algorithms. Given a constraint C on the variables $\bar{X}$, a *support* for $X_i = v_j$ on C is a partial assignment $\bar{v}$ of $\bar{X}$ containing $X_i = v_j$ that satisfies C. A value $v_j \in D(X_i)$ without support on a constraint is *arc inconsistent*. A variable X_i is *generalized arc consistent* (GAC) on C iff every value in $D(X_i)$ has support on C. A constraint C is GAC iff each constrained variable is GAC on C. A *bound support* on C is a support where the interval $[min(X_i), max(X_i)]$ is substituted for the domain of each constrained variable X_i. A variable X_i is *bound consistent* (BC) on C if $min(X_i)$ and $max(X_i)$ have bound support on C. A constraint is BC iff all constrained variables are BC on C.

In line with [11], we say that a local consistency property Φ on C is as strong as Ψ (written $\Phi \succeq \Psi$) iff, given any domains, if Φ holds then Ψ holds; Φ is stronger than Ψ (written $\Phi \succ \Psi$) iff $\Phi \succeq \Psi$ but not $\Psi \succeq \Phi$; Φ is equivalent to Ψ (written $\Phi \equiv \Psi$) iff $\Phi \succeq \Psi$ and $\Psi \succeq \Phi$; and that they are incomparable otherwise (written $\Phi \bowtie \Psi$).

2.2 Graph theoretic concepts

Given a family of sets $\mathcal{F} = \{S_1, \ldots, S_n\}$ and a graph $G = (V, E)$ with the set of vertices $V = \{v_1, \ldots, v_n\}$ and set of edges E, G is the *intersection graph of* $\mathcal{F}$ iff

$$\forall i, j \ \langle v_i, v_j \rangle \in E \leftrightarrow S_i \cap S_j \neq \emptyset$$



(a) domains (b) $G_{\bar{X}}$ (c) $G_{\bar{I}}$

Fig. 1. Domains, intersection graph and interval graph.

For any graph G, there exists a family of sets $\mathcal{F}$ such that the intersection graph of $\mathcal{F}$ is G. Thus, the class of intersection graphs is simply the class of all undirected graphs [8]. The class of graphs obtained by the intersection of *intervals*, instead of sets, is known as *interval graphs*.

Given a vector of variables $\bar{X} = \langle X_1, \ldots, X_m \rangle$, we use $G_{\bar{X}} = (V, E)$ for the induced intersection graph, i.e the graph where $V = \{v_1, \ldots, v_m\}$ and $\forall i, j \cdot \langle v_i, v_j \rangle \in E \leftrightarrow D(X_i) \cap D(X_j) \neq \emptyset$. Similarly, we use $\bar{I}$ for the same vector of variables, where all domains are seen as intervals instead, i.e., for each i, $D(X_i) = [min(X_i), max(X_i)]$. $G_{\bar{I}}$ is the induced interval graph, defined like $G_{\bar{X}}$, but on the intervals instead. For instance, the domains in Figure (1,a) induce the intersection graph in (1,b) and the interval graph in (1,c)

Finally, we recall that an *independent set* is a set of vertices with no edge in common. The *independence number* $\alpha(G)$ of a graph G, is the number of vertices in an independent set of maximum cardinality. A *clique* is the dual concept: a set of vertices such that any pair has an edge between. A *clique cover* of G is a partition of the vertices into cliques. The cardinality of the *minimum clique cover* is $\theta(G)$. For instance, the interval graph of Figure (1,c) has $\{\{v_1, v_2, v_3\}\{v_4, v_5, v_6\}\}$ as a minimal clique cover, hence $\theta(G_{\bar{I}}) = 2$. Similarly, the intersection graph of Figure (1,b) has $\{v_1, v_3, v_5\}$ as a maximal independent set, hence $\alpha(G_{\bar{X}}) = 3$.

3 The NValue constraint

In this section we define the NValue constraint and we show that it can be decomposed into two simpler constraints. Whereas one of these constraints is polynomial to propagate using a maximum matching algorithm, the second is NP-hard so we look at approximate methods.

Definition 1. NValue$(N, \langle X_1, \ldots, X_m \rangle)$ *holds iff* $N = |\{X_i|\ 1 \leq i \leq m\}|$

Enforcing GAC on the NValue constraint is NP-hard in general [3]. We can, however, decompose it into two simpler constraints: the AtLeastNValue and the AtMostNValue constraints.

Definition 2. AtLeastNValue$(N, \langle X_1, \ldots, X_m \rangle)$ *holds iff* $N \leq |\{X_i|\ 1 \leq i \leq m\}|$. AtMostNValue$(N, \langle X_1, \ldots, X_m \rangle)$ *holds iff* $N \geq |\{X_i|\ 1 \leq i \leq m\}|$.

We can identify precisely when the decomposition of a NValue constraint does not hinder propagation.

Theorem 1. *If* AtLeastNValue *and* AtMostNValue *are GAC and* $|D(N)| \neq 2$ *or* $min(N) + 1 = max(N)$*, then* NValue *is GAC.*

Proof. Suppose that the decomposition is GAC. Then we have $card{\downarrow}(\bar{X}) \leq min(N)$ and $card{\uparrow}(\bar{X}) \geq max(N)$. Thus, by Lemma 1 (see Appendix), N is GAC for NValue. Furthermore, we know that if $|D(N)| > 2$, or $D(N)$ contains a value v such that $card{\downarrow}(\bar{X}) < v < card{\uparrow}(\bar{X})$ then all variables in $\bar{X}$ are GAC (see Lemma 2 in Appendix). Therefore we only need to cover three cases:

- $D(N) = \{card{\uparrow}(\bar{X})\}$. Let v be an arc inconsistent value in $\bar{X}$. There is no assignment whose cardinality is greater than $card{\uparrow}(\bar{X})$, therefore v is arc inconsistent because it participates only in assignments of cardinality below N. Hence v is arc inconsistent for AtLeastNValue, which contradicts the hypothesis.
- $D(N) = \{card{\downarrow}(\bar{X})\}$. Analogous to the last case.
- $D(N) = \{card{\downarrow}(\bar{X}), card{\uparrow}(\bar{X})\}$: If $card{\downarrow}(\bar{X}) + 1 = card{\uparrow}(\bar{X})$ then NValue is GAC (see Lemma 2 in Appendix). Otherwise, there is a gap between the bounds. This is the only case where the decomposition is GAC but NValue may not be. For instance, consider the domains: $X_1 \in \{1, 2, 3\}, X_2 \in \{1, 2\}, X_3 \in \{1\}, N \in \{1, 3\}$. Whilst enforcing GAC on NValue$(N, \langle X_1, X_2, X_3\rangle)$ will prune $X_1 = 2$, these domains are GAC for the decomposition. □

If the domain of N contains only $card{\downarrow}(\bar{X})$ and $card{\uparrow}(\bar{X})$, and these two values are not consecutive, then NValue may not be GAC even though AtMostNValue and AtLeastNValue are GAC. However, as we show in section 7, we can make GAC on the decomposition equivalent, by performing an extra pruning in this situation.

3.1 The AtLeastNValue constraint

We first have a brief look at the AtLeastNValue constraint. It is known [1] that $card{\uparrow}(\bar{X})$ is the cardinality of the maximal matching of the bipartite graph with a class of vertices representing the variables, another the values, and where an edge links two vertices if and only if it corresponds to a valid assignment. Indeed, this is the basic idea behind Régin's algorithm for enforcing GAC on the AllDifferent constraint [12]. We can easily derive a propagation procedure for AtLeastNValue using the polynomial algorithm for the SoftAllDiff constraint [10] that counts the number of variables that need to be reassigned to satisfy the constraint. We can use this algorithm to compute $card{\uparrow}(\bar{X})$. Moreover, we can use this same algorithm to prune the values in $\bar{X}$ that do not belong to a maximal matching. This nearly provides us with an algorithm for enforcing GAC on AtLeastNValue. One difference is that we do not always want to prune the values that do not participate in a maximal matching. We shall see how this algorithm can be used when pruning the variables in $\bar{X}$ in section 7. We refer the reader to [10] for more details about this algorithm, and we focus on the constraint AtMostNValue for the rest of the paper.

3.2 The AtMostNValue constraint

We adapt the proof of NP-hardness for NValue [3] to also show that enforcing GAC on an AtMostNValue constraint alone is intractable.

Theorem 2. *Enforcing GAC on a* AtMostNValue$(N, \langle X_1, \ldots, X_m\rangle)$ *constraint is NP-hard, and remains so even if* N *is ground.*

Proof. We use a reduction from 3SAT. Given a formula in k variables and n clauses, we construct the AtMost$N$Value$(X_1, \ldots, X_{k+n}, N)$ constraint in which $D(X_i) = \{i, \neg i\}$ for all $i \in [1, k]$, and each X_i for $i > k$ represents one of the n clauses. If the jth clause is $x \vee \neg y \vee z$ then $D(X_{k+j}) = \{x, \neg y, z\}$. By construction, the variables will consume k distinct values, hence if $N = k$, the constructed AtMost$N$Value constraint has a solution iff the original 3SAT problem has a satisfying assignment. The completeness is easy to see as the support is a polynomial witness. Hence testing a value for support is NP-complete, and enforcing GAC is NP-hard. □

Note that this proof is a reduction of 3SAT into the problem of propagating GAC on $\bar{X}$ when N is ground. This means that pruning $\bar{X}$ alone is NP-hard. Indeed, even computing just the lower bound on N, given $\bar{X}$ is no easier.

Theorem 3. *Computing the value of* $card{\downarrow}(\bar{X})$ *is NP-hard.*

Proof. Computing $card{\downarrow}(\bar{X})$ is equivalent to finding the cardinality of a *minimum hitting set* of $\bar{X}$ seen as a family of sets. A *hitting set* of a family of sets $\mathcal{F}$, is a set that intersects each member of $\mathcal{F}$. Computing the cardinality of the smallest possible hitting set is NP-hard [9]. If we have one variable X_i in $\bar{X}$ for each set $S_i \in \mathcal{F}$, and $D(X_i) = S_i$, then $card{\downarrow}(\bar{X})$ is equal to the cardinality of a minimum hitting set of $\mathcal{F}$. □

4 Existing algorithm for the AtMostNValue constraint

We first recall Beldiceanu's algorithm, then we introduce a graph theoretic view of his method. We shall refer to Beldiceanu's algorithm as `OI`, for *ordered intervals*. The first step is to order the domains by increasing lower bound. Then the following procedure (algorithm 1) can be applied, the value returned ($N_{distinct}$) is a lower bound on $card{\downarrow}(\bar{X})$.

The intervals are explored one at a time, and a new group, i.e. a clique of the interval graph, is completed when an interval is found that does not overlap with all previous ones in the group. The time complexity is $O(nlog(n))$ for sorting, and then the algorithm itself is linear, the loop visits each domain at most twice (when this domain is distinct from the previous). Hence, the worst case time complexity is dominated by $O(nlog(n))$. This algorithm is proved correct, that is, it returns a valid lower bound, by noticing that the intervals with smallest maximum value for each group are pairwise disjoint. Consequently, at least as many values as groups, that is, $N_{distinct}$, have to be used. As there was no proof given in [1], we present one here:

Algorithm 1: `OI`: The interval-based algorithm introduced in [1]

Data : $\bar{X} = [X_1, \dots X_m]$
Result : $N_{distinct}$
$N_{distinct} \leftarrow 1$; $reinit \leftarrow 1$; $i \leftarrow 1$; $low \leftarrow -\infty$; $up \leftarrow \infty$;
while $i < m$ **do**
  $i \leftarrow i + 1 - reinit$;
  if $reinit$ *or* $(low < min(X_i))$ **then** $low \leftarrow min(X_i)$;
  if $reinit$ *or* $(up > max(X_i))$ **then** $up \leftarrow max(X_i)$;
  $reinit \leftarrow (low > up)$;
  $N_{distinct} \leftarrow N_{distinct} + reinit$;
return $N_{distinct}$;

Proposition 1 (given in [1] without proof). *Let $\{C_1, \dots, C_k\}$ be a partition of the intervals, output of* `OI`. *If $\bar{I} = \langle I_1, \dots, I_k \rangle$ is the vector of intervals where I_i is the element of C_i with least maximum value, then all elements of $\bar{I}$ have empty pairwise intersections.*

Proof. `OI` scans all intervals by increasing lower bound, partitioning into groups on the way. When the algorithm ends, we have k groups $C_1, \dots, C_k$. For any group C_i, consider the interval I_1 with least upper bound. This interval does not intersect any interval in any group C_j such that $j > i$. Suppose it was the case, i.e, there exists $I_2 \in C_j$ which intersects with I_1, since the intervals are ordered by increasing lower bound, I_2 cannot be completely *below* any interval in C_i. It must then be either completely *above* or overlapping. However, since I_1 has the least upper bound and intersects I_2, all intervals in C_i must also intersect I_2. It follows that I_2 should belong to C_i hence the contradiction. The set containing the interval with least upper bound of every group is then pairwise disjoint, and is of cardinality k. □

Moreover, it is easy to see that, when the domains are indeed intervals, this bound can be achieved. If, for each group, we assign all the variables of this group to one of the common values, then we obtain an assignment of cardinality $N_{distinct}$. This argument is used in [1] to show that `OI` achieves BC on N.

Now, recall that $G_{\bar{X}}$ is the intersection graph of the variables in $\bar{X}$, whereas $G_{\bar{I}}$ is the interval graph of the same variables. It is easy to see that `OI` computes at once a clique cover and an independent set of $G_{\bar{I}}$. Moreover, since for any graph $\alpha(G) \leq \theta(G)$, if a graph G contains an independent set *and* a clique cover of cardinality n, we must conclude that $n = \alpha(G) = \theta(G)$. Indeed, interval graphs belong to the class of perfect graphs, for which, by definition, the independence number is equal to the size of the minimum clique cover. Therefore, we know that the output of `OI`, i.e., $N_{distinct}$ is equal to $\alpha(G_{\bar{I}})$ and also to $\theta(G_{\bar{I}})$. It can be shown that, in this case, the cardinality of the minimum clique cover on the interval graph is equal to the cardinality of the minimum hitting set on $\bar{I}$ itself. This is due to the fact that a set of intervals that pairwise intersect always share a common interval, any element of this interval hitting all of them. To summarize, in the special case where the domains of all variables in $\bar{X}$ are intervals (denoted $\bar{I}$), the following equality holds: $\alpha(G_{\bar{I}}) = \theta(G_{\bar{I}}) = card{\downarrow}(\bar{I})$

As a consequence, the value $N_{distinct}$ is exact, hence `OI` achieves bound consistency on N for the constraint AtMostNValue. However, considering domains as intervals may be in some case a very crude approximation. If we consider the intersection graph $G_{\bar{X}}$ instead of the interval graph $G_{\bar{I}}$, the relation becomes: $\alpha(G_{\bar{X}}) \leq \theta(G_{\bar{X}}) \leq card\downarrow(\bar{X})$

Any of those three quantities is a valid lower bound, though they are NP-hard to compute. They are, on the other hand, tighter approximations that do not consider domains as intervals. Indeed, since $G_{\bar{X}}$ has more edges than $G_{\bar{I}}$, it follows immediately that: $\alpha(G_{\bar{X}}) \geq \alpha(G_{\bar{I}})$ $(= \theta(G_{\bar{I}}) = card\downarrow(\bar{I}))$

5 Three new approaches

We present two algorithms approximating $\alpha(G_{\bar{X}})$ and a linear relaxation approximating directly the minimum hitting set problem, and hence $card\downarrow(\bar{X})$.

5.1 A greedy approach

We have seen that `OI` approximates the lower bound on N by computing the exact value of $\alpha(G_{\bar{I}})$, the independence number of the interval graph induced by $\bar{X}$. Here the idea is to compute the independence number of $G_{\bar{X}}$, $\alpha(G_{\bar{X}})$.

Whilst computing the exact value of $\alpha(G_{\bar{X}})$ is intractable for unrestricted graphs, some efficient approximation schemes exist for that problem. We use here a very simple heuristic algorithm for computing the independence number of a graph, referred to as "the natural greedy algorithm". We denote it `MD`, for *minimum degree* from now on. It consists in removing the vertices of minimum degree as well as their neighborhood in turn. The number of iterations i is such that $i \leq \alpha(G)$. This algorithm is studied in detail in [6]. If we suppose that the intersection graph is constructed once and maintained during search, then a careful implementation can run in $O(n+m)$ where n is the number of vertices and m is the number of edges (linear in the size of the graph). However, computing the intersection graph requires $n(n+1)/2$ tests of intersection. Each of those may require at most d equality checks, where d is the size of the domains in $\bar{X}$. Notice that efficient data structures, such as bit vectors, are often used to represent domains and thus allow intersection checks in almost constant time in practice. This suggests an implementation where the graph is never actually computed, but an intersection check is done each time we need to know if an edge links two nodes. The worst case time complexity is then $O(dn^2)$ if intersection is linear in the size of the sets or $O(n^2)$ if it is constant. We denote $\Gamma(v)$ the neighborhood of v, $\Gamma(v) = \{w | vw \in E\}$.

5.2 Turán's approximation

Alternatively, we can use an even simpler approximation. Turán proposed a lower bound of $\frac{n^2}{2m+n}$ for $\alpha(G)$ in [13], where n is the number of vertices and m the number of edges. Therefore assuming that m is computed once, and revised whenever a domain changes or whenever the constraint is called again, this formula gives a lower bound in constant time. The worst case time complexity

Algorithm 2: `MD`: A greedy algorithm approximating the maximum independent set of a graph

Data : $G = (V, E)$
Result : $N_{distinct}$
if $G = \emptyset$ **then** return 0;
choose $v \in V$ such that $d(v)$ is minimum;
return 1+MD($G(V \setminus (\Gamma(v) \cup \{v\}))$);

is the same as `MD`'s (because of the initialization). However, this heuristic can be much more efficient in practice. We refer to this method as `Turan`.

5.3 A linear relaxation approach

We have shown the following inequalities: $\alpha(G_{\bar{I}}) \leq \alpha(G_{\bar{X}}) \leq card\!\downarrow\!(\bar{X})$.

We have seen that the cardinality of the minimum hitting set problem where the family of sets is formed by the domains of the variables in $\bar{X}$ is equal to the lower bound on N, that is, $card\!\downarrow\!(\bar{X})$. One difficulty is that approximation algorithms proposed in the literature for minimum hitting set return a set which may be too large, and so do not provide a valid lower bound. However, we consider here a linear relaxation that can be solved in polynomial time that gives a lower bound on the minimum hitting set cardinality, and thus, of N. We refer to this method as LP.

Given a vector of variables $\bar{X} = \langle X_1, \ldots X_m\rangle$, let $V = \bigcup_{v \in \bar{X}} D(x)$ be the total set of values. Then let $\{y_v | \ v \in V\}$ be a set of linear variables, and LP is as follows:

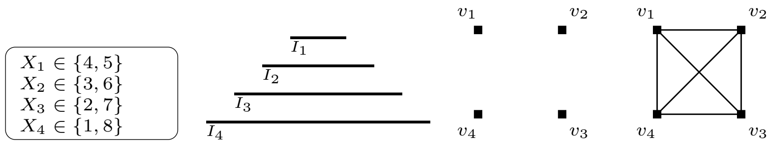
$$min \sum_{v \in V} y_v \quad subject \ \ to \quad \sum_{v \in D(X_i)} y_v \geq 1 \quad \forall X_i \in \bar{X}$$

where $y_v \geq 0$ forall $v \in V$.

The best polynomial linear program solvers based on the interior point methods run in $O(v^3 L)$ where v is the number of variables and L is the number of bits in the input. The number of variables in our linear program is nd ($d = |D(X_i)|$) and we have $n = |\bar{X}|$ inequalities of size d. Therefore, the worst case time complexity is $O(n^4 d^4)$. In practice, the simplex method may behave better even though it has an exponential worst case time complexity.

6 Theoretical Analysis

We will compare local consistency properties applied to the AtMostNValue constraint. In our case, the AtMostNValue constraint holds iff the lower bound returned by the propagation algorithm (consistency) does not exceed $max(N)$. Thus, given consistency properties Φ and Ψ, $\Phi \succeq \Psi$ means that the lower bound on N returned by the algorithm enforcing Φ is greater than or equal to the lower



(a) domains (b) intervals (c) $G_{\bar{X}}$ (d) $G_{\bar{I}}$

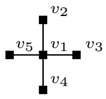
Fig. 2. Example for `OI` $\not\succeq$ `MD`$(G_{\bar{X}})$ and for `OI` $\not\succeq$ `Turan`

bound returned by the algorithm enforcing Ψ.[1] We consider comparing the level of consistency achieved by the following algorithms: `OI`, `MD`, `Turan`, and `LP`.

Note that, since only the lower bound on N is considered in this comparison, `OI` is then equivalent to BC. We do not compare with generalized arc consistency either, as this is NP-hard to enforce and all our algorithms are polynomial and strictly weaker.

Theorem 4. `MD` $\succ$ `Turan`

Proof. For a proof that `MD` is as strong as `Turan` see [6]. Moreover, it is easy to find an example showing that `MD` is strictly stronger. For instance consider the following domains: $X_1 \in \{1,2,3,4,5,6,7,8\},\ X_2 \in \{1,2\},\ X_3 \in \{3,4\},\ X_4 \in \{5,6\}$, *and* $X_5 \in \{7,8\}$.

The induced intersection graph is as follows: 

When applying `MD`, we obtain an independent set of size 4. However, `Turan` returns: $\lceil \frac{n^2}{2m+n} \rceil = \lceil \frac{25}{13} \rceil = 2$. and we deduce a lower bound of 2 for N. □

Theorem 5. `Turan` $\bowtie$ `OI`

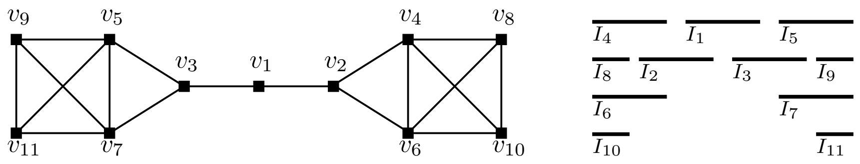
Proof. To see that `Turan` is not as strong as `OI`, consider the example used in the proof of Theorem 4. The domains being intervals, we know that `OI` computes the exact lower bound, 4. However the Turán heuristics gives us 2.

To see that `OI` is not as strong as Turán, consider the domains in Figure 2. The induced intersection graph $G_{\bar{X}}$ has 4 vertices ($n = 4$) and no edges ($m = 0$), thus `Turan` returns 4. However, the interval graph $G_{\bar{I}}$ induced by the same domains is a clique and then `OI` returns 1. □

Theorem 6. `MD` $\bowtie$ `OI`

Proof. To see that `MD` $\not\succeq$ `OI`, consider the interval graph in Figure (3,a) induced by the intervals of Figure (3,b). The exact independence number is 4 (for instance $\{v_2, v_3, v_8, v_9\}$ is an independent set of cardinality 4), and thus `OI` returns 4. However, the vertex with minimal degree is v_1, and no independent set of cardinality 4 involves v_1, therefore `MD` is not as strong as `OI`.

[1] We refer to the level of local consistency achieved by an algorithm A as A as well.



(a) graph ($G_{\bar{I}}$) (b) intervals ($\bar{I}$)

Fig. 3. Example for `MD` $\not\succeq$ `OI`

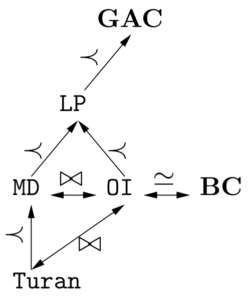


Fig. 4. Relations between algorithms and consistencies on the AtMostNValue constraint

To see that `OI` $\not\succeq$ `MD`($G_{\bar{X}}$), see Figure 2. It is easy to construct domains where the interval graph can have arbitrarily more edges than the intersection graph. For instance the domains in Figure 2,a induce a complete interval graph, or an unconnected intersection graph. Therefore `OI` is not as strong as `MD`. □

Theorem 7. *`LP` $\succ$ `MD`, `LP` $\succ$ `OI` and `LP` $\succ$ `Turan`.*

Proof. We first show that the value returned by `LP` is greater or equal to $\alpha(G_{\bar{X}})$. Consider a maximum independent set A of the intersection graph. We know that any two variables in A have no value in common. However for each variable $X_i \in A$ we have: $\sum_{v \in D(X_i)} y_v \geq 1$. Since the domains of those variables are disjoint, we have:

$$\sum_{v \in \bigcup_{X_i \in A} D(X_i)} y_v \geq |A| = \alpha(G_{\bar{X}})$$

And thus the total sum to minimize is greater than or equal to $\alpha(G_{\bar{X}})$. However, recall that `OI`, `MD` and `Turan` all approximate $\alpha(G_{\bar{X}})$ by giving a lower bound. Therefore `LP` is as strong as `OI`, `MD` and `Turan`. Moreover, the variables $X_1 \in \{1, 2\}, X_2 \in \{2, 3\}, X_3 \in \{1, 3\}$ constitute an example showing that `LP` is strictly stronger, as the optimal sum for `LP` is 1.5, whilst $\alpha(G_{\bar{X}}) = 1$.

□

7 A propagation algorithm for the NValue constraint

A template for an approximate propagation algorithm for NValue is given in Algorithm 3. In this template, one may use any of the methods described in the previous sections. The pruning on N is straightforward (Line 1 and 2). When we have $max(N) < min(N)$ (Line 3), there is clearly an inconsistency, and the algorithm fails. In the following subsections, we consider the cases of Lines 4, 5 and 6, where some filtering may be achieved. All other cases (Line 7) satisfy the preconditions of Lemma 2 (see Appendix) and the constraint is GAC. Therefore, either the constraint is GAC, or we are unable to deduce any inconsistency because the lower bound lb for $card{\downarrow}(\bar{X})$ is not tight enough.

Algorithm 3: Algorithm for propagating the NValue constraint

Data : $\bar{X}$, N
Result :
$ub \leftarrow card{\uparrow}(\bar{X}))$;
$lb \leftarrow approx(card{\downarrow}(\bar{X})))$;
1 $max(N) \leftarrow min(max(N), ub)$;
2 $min(N) \leftarrow max(min(N), lb)$;
3 **if** $(max(N) < min(N))$ **then** fail;
4 **case** $(ub = min(N) = max(N) \neq lb)$: pruning from below;
5 **case** $(lb = min(N) = max(N) \neq ub)$: pruning from above;
6 **case** $(|D(N)| = 2$ *and* $min(N) + 1 < max(N))$: pruning from within;
7 **otherwise** return;

7.1 Pruning from below

This pruning is triggered when $card{\uparrow}(\bar{X}) = min(N)$ and $card{\downarrow}(\bar{X}) < min(N)$. In this situation, we know that some assignments may have too small cardinality, and therefore some values may not participate in assignments of cardinality $card{\uparrow}(\bar{X}) = min(N)$, which is the only cardinality satisfying the constraint. Making AtLeastNValue GAC is then sufficient to make the whole constraint GAC as this corresponds to the first of the three possible cases discussed in the proof of Theorem 1. In this situation, we can use a polynomial procedure for enforcing GAC on the SoftAllDiff constraint which counts the number of variables that have to be reassigned in order to be all different [10].

7.2 Pruning from above

This is the dual case, we know that some assignments may have too large cardinality, and therefore some values may participate only in assignments of cardinality above $max(N)$ (and we assume $max(N) = card{\downarrow}(\bar{X})$). This corresponds to the second case of the proof of Theorem 1. Making AtMostNValue GAC is then sufficient to make the whole constraint GAC. Note that here we are not sure to achieve GAC.

In [1] (p. 6), two observations are made in order to prune $\bar{X}$ which are relevant here when using `MD` to compute $min(N)$. We reformulate these observations consistently to the graph notations we used. First, let A be a set of variables that form an independent set of the intersection graph, and let $X_i \in (\bar{X} \setminus A)$ be assigned to a value v which does not belong to any domain in A. It follows that the minimum number of values required will be at least $\alpha(G_{\bar{X}}) + 1$. Hence we can prune the value v from the domain of X_i when N is equal to $\alpha(G_{\bar{X}})$. This way of pruning the variables can be used with `MD` as well as with `OI`. There are no further difficulties when going from interval graphs to intersection graphs. Consequently, given an independent set A, we can propagate the following constraint: $\forall X_i \in \bar{X},\ \exists X_j \in A\ s.t.\ X_i = X_j$. Second, suppose that A' is another distinct independent set. Thus, we have: $\forall X_i \in \bar{X},\ \exists X_{j_1} \in A,\ \exists X_{j_2} \in A'\ s.t.\ (X_i = X_{j_1} \wedge X_i = X_{j_2})$. Therefore, one can prune values in $\bar{X}$ by finding a set of independent sets $\mathcal{A} = \{A_1, \dots A_k\}$. The set of consistent value $\mathcal{V}$ is defined as follows: $\forall A \in \mathcal{A}, U_A = \bigcup_{X_i \in A} D(X_i), \quad \mathcal{V} = \bigcap_{A \in \mathcal{A}} U_A$. It may be difficult to compute *all* independent sets of cardinality equal to N. One must therefore find a set which is as large as possible. In [1] from the first one found with `OI`, each independent set that differs by only one vertex is deduced. This can be computed in linear time, without increasing the algorithm's complexity. As a result this way $\bar{X}$ is pruned, the algorithm described in [1] does not enforce BC on AtMostNValue. The following domains are a counter example: $X_1 \in \{1,2\},\ X_2 \in \{2,3\},\ X_3 \in \{3,4\},\ X_4 \in \{4,5\},\ and\ N \in \{2\}$. Only the values 2 for X_1, X_2 and 4 for X_3, X_4 are bound consistent. However, the independent sets considered will be $\{X_1, X_3\}$ and $\{X_1, X_4\}$. Therefore, the values that are consistent are $\{1, 2, 4\}$. This way of pruning can make holes in domains. Therefore the level of consistency achieved on AtMostNValue is incomparable with bound consistency. Although they are not equivalent, one can easily derive a procedure to enforce BC from `OI`. To check the (say lower) bound of a variable X_i, we assign this bound to X_i and compute N again. If $card{\downarrow}(\bar{X})$ after this assignment is greater than N, this bound is not BC.

With algorithms that do not compute independent sets in order to get a lower bound on N, like the linear relaxation method or the Turán heuristic, we are in a different situation. However, we can simply wait until $min(N) > max(N)$ and fail in this case, without pruning any variable in $\bar{X}$. Alternatively we could compute a new lower bound for each value v of each X_i, that is, $O(nd)$ times. We set $y_v = 1$. and if the objective function fails to be lower than or equal to N, then v is arc inconsistent. Since the pruning on $\bar{X}$ happens in a limited number of situations, it may be cost effective to use this complete method.

7.3 Pruning from within

This pruning is triggered when $card{\downarrow}(\bar{X}) = min(N)$, $card{\uparrow}(\bar{X}) = max(N)$ and $card{\downarrow}(\bar{X}) + 1 < card{\uparrow}(\bar{X})$. This is the last of the three cases in the proof of Theorem 1. In this case AtMostNValue and AtLeastNValue can be GAC whilst NValue is not. However, we can use a conditional constraint to do some pruning in this particular case. The idea is, when these conditions are met, to trigger the following constraint to perform this extra filtering:

$$Min = card{\downarrow}(\bar{X}) \wedge Max = card{\uparrow}(\bar{X}) \wedge$$
$$(\textsc{AtMostNValue}(Min, \bar{X}) \vee \textsc{AtLeastNValue}(Max, \bar{X}))$$

Min and Max are two extra variables. We have the following theorem:

Theorem 8. *If $D(N) = \{card{\downarrow}(\bar{X}), card{\uparrow}(\bar{X})\}$ and $card{\downarrow}(\bar{X}) + 1 < card{\uparrow}(\bar{X})$ then* NValue$(N, \bar{X})$ *is GAC iff the decomposition and (when the conditions are met) the conditional constraint are GAC.*

Proof. ($\Rightarrow$) The case where $D(N) = \{card{\downarrow}(\bar{X}), card{\uparrow}(\bar{X})\}$ or $card{\downarrow}(\bar{X}) + 1 < card{\uparrow}(\bar{X})$ does not hold is covered by Theorem 1. Now suppose this condition holds, and there is a value $v_i \in D(X_i)$ which is not GAC for NValue. By definition, this implies that any assignment such that the i^{th} element is v_i has a cardinality different from $card{\downarrow}(\bar{X})$ and from $card{\uparrow}(\bar{X})$, since these values are in $D(N)$. Moreover, there is no assignment with cardinality above $card{\uparrow}(\bar{X})$ or below $card{\downarrow}(\bar{X})$. Therefore we deduce that any assignment $\bar{v}$ involving v_i is such that $card{\downarrow}(\bar{X}) < card(\bar{v}) < card{\uparrow}(\bar{X})$. Hence, if $Min = card{\downarrow}(\bar{X}) \wedge Max = card{\uparrow}(\bar{X})$ holds, then v_i would be inconsistent for both AtMostNValue$(Min, \bar{X})$ and AtLeastNValue$(Max, \bar{X})$).

($\Leftarrow$) If a value v_i belongs to a support, i.e., an assignment whose cardinality is either $card{\downarrow}(\bar{X})$ or $card{\uparrow}(\bar{X})$, then either AtMostNValue$(Min, \bar{X})$ or AtLeastNValue$(Max, \bar{X})$) or both are GAC. □

Hence, we simply assign $card{\downarrow}(\bar{X})$ to N, then we compute B_1, the set of values inconsistent for AtMostNValue. Similarly, we assign $card{\uparrow}(\bar{X})$ to N and compute B_2, the set of values inconsistent for AtLeastNValue, In both cases, we use the methods described in section 7.1 and 7.2. Once this is done, we restore the domain of N, and prune all values in $B_1 \cap B_2$. Notice that B_1 may be underestimated, hence we do not achieve GAC.

8 Related work

Two algorithms, on the same line as `OI`, yet achieving BC, have been introduced in [2]. In this technical report, the authors also extend the constraint to deal with weights on values. Observe that filtering on the weighted version of the constraint can easily be done with the linear relaxation method. Indeed, the weights on values can be represented as coefficients in the linear equations.

The maximum independent set is a well known problem in graph theory and a number of approximation algorithms have been proposed. We used two simple and intuitive algorithms for the sake of simplicity and because `MD` is successful in practice. However, algorithms with better approximation ratio exist, for instance see [7]. Any such algorithm may replace `MD` into the propagation algorithm.

We have seen that the linear programming approach is always stronger, even than a complete method for finding a maximum independent set. It is difficult to identify where the linear relaxation for the minimum hitting was first introduced, as it is such a simple model. It is certainly given in [5]. One weakness of the linear programming approach is that it is difficult to deduce which values to prune when $min(N) = max(N)$.

9 Conclusion

Propagating generalized arc consistency on the NValue constraint is NP-hard. In order to filter inconsistent values, one has to obtain tight bounds on the number of distinct values used in assignments. Whilst the upper bound can be obtained in polynomial time with a maximal matching procedure, the lower bound alone is NP-hard to compute. Therefore, our focus is on methods which achieve lesser levels of consistency. A procedure proposed by Beldiceanu considers domains as intervals, which allows the independence number of the induced interval graph to be computed in polynomial time. The independence number of this graph is a valid lower bound on the number of distinct values. We introduce three new methods for approximating this lower bound. The first two approximate the independence number of the intersection graph. However, these algorithms have a quadratic worst case time complexity, and do not guarantee a tighter lower bound. The last and most promising approach is to use a linear relaxation of the minimum hitting set problem. The cardinality of the minimum hitting set is a tight lower bound on the number of distinct values. This always finds a tighter lower bound than the approaches based on the maximum independent set problem. In our future work, we will compare these methods experimentally.

ACKNOWLEDGEMENTS

Brahim Hnich is currently supported by Science Foundation Ireland under Grant No. 00/PI.1/C075.

Appendix: conditions when NValue is GAC

In order to show when enforcing GAC on the decomposition of the NValue constraint enforces GAC on NValue, we used two lemmas. These identify when the variables in the NValue constraint are GAC. First, N is GAC iff its bounds are between $card{\downarrow}(\bar{X})$ and $card{\uparrow}(\bar{X})$.

Lemma 1. *Any value in $D(N)$ is GAC for* NValue *as long as it is lower than or equal to $card{\uparrow}(\bar{X})$ and greater than or equal to $card{\downarrow}(\bar{X})$.*

Proof. Let S be any assignment of $\bar{X}$. Consider assigning $\bar{X}$ as in S, one variable at a time. Let $\bar{X}_k$ be $\bar{X}$ at step k, that is, with k ground variables. Hence, since $\bar{X}$ involves m values, $\bar{X}_m$ corresponds to Sol At a step k, the value of $card{\downarrow}(\bar{X}_k)$ (resp. $card{\uparrow}(\bar{X}_k)$) increases (resp. decreases) by at most one with respect to step $k-1$. Moreover, when every variable is assigned, $card{\downarrow}(\bar{X}_m)$ $=card{\uparrow}(\bar{X}_m) = card(S)$. Therefore, for any value p between $card{\downarrow}(\bar{X}_0)$ and $card{\uparrow}(\bar{X}_0)$, there exists k such that either $card{\downarrow}(\bar{X}_k) = p$ or $card{\uparrow}(\bar{X}_k) = p$. Consequently p has a support for a sub-domain $\bar{X}_k$ and is thus GAC. □

Second, the variables in $\bar{X}$ are GAC if either $D(N) = [card{\downarrow}(\bar{X}), card{\uparrow}(\bar{X})]$ or there exists at least one value lower than $card{\uparrow}(\bar{X})$ and greater than $card{\uparrow}(\bar{X})$.

Lemma 2. *If either* $D(N) = [card{\downarrow}(\bar{X}), card{\uparrow}(\bar{X})]$ *or* $card{\downarrow}(\bar{X})+1 < card{\uparrow}(\bar{X})$ *and* $[card{\downarrow}(\bar{X}) + 1, card{\uparrow}(\bar{X}) - 1] \cap D(N) \neq \emptyset$ *then* $\bar{X}$ *is GAC.*

Proof. We first show the first part of the disjunction. Recall that $card{\downarrow}(\bar{X})$ (resp. $card{\uparrow}(\bar{X})$) is the cardinality of the smallest (resp. largest) possible assignment. Therefore, if the domain of N is equal to the interval $[card{\downarrow}(\bar{X}), card{\uparrow}(\bar{X})]$ it means that all assignments of $\bar{X}$ have a cardinality in $D(N)$.

For the second part, we use again the argument that assigning a single variable can affect the bounds by at most one. In other words, for all $X_i \in \bar{X}$, a value $v \in D(X_i)$ (without loss of generality) belongs to an assignment of cardinality either $card{\downarrow}(\bar{X})$, $card{\downarrow}(\bar{X}) + 1$, $card{\uparrow}(\bar{X})$ or $card{\uparrow}(\bar{X}) - 1$. Moreover, let $\bar{X}_{X_i=v}$ be $\bar{X}$ where the domain of X_i is reduced to $\{v\}$. We have $card{\downarrow}(\bar{X}_{X_i=v}) \leq card{\downarrow}(\bar{X}) + 1$ and $card{\uparrow}(\bar{X}_{X_i=v}) \geq card{\uparrow}(\bar{X}) - 1$. Hence, by assumption $D(N) \cap [card{\downarrow}(\bar{X}_{X_i=v}), card{\uparrow}(\bar{X}_{X_i=v})] \neq \emptyset$, and by applying Lemma 1, we know that there exists a tuple satisfying NValue with $X_i = v$. $\square$

References

1. N. Beldiceanu. Pruning for the *minimum* constraint family and for the *Number of Distinct Values* constraint family. In *Proceedings CP-01*, 2001.
2. N. Beldiceanu, M. Carlsson, and S. Thiel. Cost-Filtering Algorithms for the two sides of the *Sum of Weights of Distinct Values* Constraint. SICS technical report, 2002.
3. C. Bessiere, E. Hebrard, B. Hnich, and T. Walsh. The complexity of global constraints. In *Proceedings AAAI-04*, 2004.
4. P. Roy F. Pachet. Automatic generation of music programs. In *Proceedings CP-99*, 1999.
5. S. Shahar G. Even, D. Rawitz. Hitting sets when the vc-dimension is small, (submitted to a journal publication) 2004.
6. M. Halldórsson and J. Radhakrishnan. Greed is good: Approximating independent sets in sparse and bounded-degree graphs. In *Proceedings STOC-94*, pages 439–448, 1994.
7. V. Th. Paschos M. Demange. Improved approximations for maximum independent set via approximation chains. *Appl. Math. Lett.*, 10:105–110, 1997.
8. E. Marzewski. Sur deux propriétés des classes s'ensembles. Fund. Math., 33:303–307, 1945.
9. D.S. Johnson M.R. Garey. *Computers and Intractability: A Guide to the Theory of NP-completeness*. W.H. Freeman and Company, 1979.
10. T. Petit, J.C. Regin, and C. Bessiere. Specific filtering algorithms for over-constrained problems. In *Proceedings CP-01*, 451-463.
11. R. Debruyne C. Bessiere. Some practicable filtering techniques for the constraint satisfaction problem. In *Proceedings IJCAI-97*, 1997.
12. J.C. Régin. A filtering algorithm for constraints of difference in CSPs. In *Proceedings AAAI-94*, pages 362–367, 1994.
13. P. Turán. On an extremal problem in graph theory. In *(in Hungarian),* Mat. Fiz. Lapok, pages 48:436–452, 1941.