Disjoint, Partition and Intersection Constraints for Set and Multiset Variables *

Christian Bessiere[1], Emmanuel Hebrard[2], Brahim Hnich[2], and Toby Walsh[2]

[1] LIRMM, Montpelier, France. bessiere@lirmm.fr
[2] Cork Constraint Computation Centre, University College Cork, Ireland.
{e.hebrard, b.hnich, tw}@4c.ucc.ie

Abstract. We have started a systematic study of global constraints on set and multiset variables. We consider here disjoint, partition, and intersection constraints in conjunction with cardinality constraints. These global constraints fall into one of three classes. In the first class, we show that we can decompose the constraint without hindering bound consistency. No new algorithms therefore need be developed for such constraints. In the second class, we show that decomposition hinders bound consistency but we can present efficient polynomial algorithms for enforcing bound consistency. Many of these algorithms exploit a dual viewpoint, and call upon existing global constraints for finite-domain variables like the global cardinality constraint. In the third class, we show that enforcing bound consistency is NP-hard. We have little choice therefore but to enforce a lesser level of local consistency when the size of such constraints grows.

1 Introduction

Global (or non-binary) constraints are one of the factors central to the success of constraint programming [7, 8, 1]. Global constraints permit the user to model a problem easily (by compactly specifying common patterns that occur in many problems) and solve it efficiently (by calling fast and effective constraint propagation algorithms). Many problems naturally involve sets and multisets. For example, the social golfers problem (prob010 at CSPLib.org) partitions a set of golfers into foursomes. Set or multiset variables have therefore been incorporated into most of the major constraint solvers (see, for example, [3, 6, 5, 11] for sets, [4] for multisets —under the name *bags*). In a recent report, Sadler and Gervet describe a propagator for a global disjoint constraint on set variables with a fixed cardinality [10]. The aim of this paper is to study other such global constraints on set and multiset variables. Using the techniques proposed in [2], we have proved that some of these global constraints are NP-hard to propagate. For example, both the `atmost1-incommon` and `distinct` constraints on sets of fixed cardinality proposed in [9] are NP-hard to propagate. We prove that others are polynomial but not decomposable without hindering propagation. We therefore give efficient algorithms for enforcing bound consistency on such constraints.

* The last three authors are supported by Science Foundation Ireland and an ILOG software grant. We wish to thank Zeynep Kiziltan for useful comments.

2 Formal background

A multiset is an unordered list of elements in which repetition is allowed. We assume that the elements of sets and multisets are integers. Basic operations on sets generalize to multisets. We let $occ(m, X)$ be the number of occurrences of m in the multiset X. Multiset union and intersection are defined by the identities $occ(m, X \cup Y) = max(occ(m, X), occ(m, Y))$ and $occ(m, X \cap Y) = min(occ(m, X), occ(m, Y))$. Finally, we write $|X|$ for the cardinality of the set or multiset X, and use lower case to denote constants and upper case to denote variables.

An integer variable N is a variable whose domain is a set of integers, $dom(N)$. The minimum (maximum) element of N is denoted by $min(N)$ ($max(N)$). A set (resp. multiset) variable X is a variable whose domain is a set of sets (resp. multisets) of integers, given by an upper bound $ub(X)$ and a lower bound $lb(X)$ (i.e., $lb(X) \subseteq X \subseteq ub(X)$). We define bound consistency for integer, set and multiset variables. We can therefore reason about constraints which simultaneously involve integer, set and multiset variables. An assignment is bound valid if the value given to each set or multiset variable is within these bounds, and the value given to each integer variable is between the minimum and maximum integers in its domain. A constraint is bound consistent (denoted by $BC(C)$) iff for each set or multiset variable X, $ub(X)$ (resp. $lb(X)$) is the union (resp. intersection) of all the values for X that belong to a bound valid assignment satisfying the constraint, and for each integer variable N, there is a bound valid assignment that satisfies the constraint for the maximum and minimum values in the domain of X. An alternative definition of BC for set and multiset variables is that the characteristic function (a vector of 0/1 variables) for each set variable, or the occurrence representation (a vector of integer variables) for each multiset variable is bound consistent [12]. We say that a constraint is "decomposable" if there exists a decomposition into a polynomial number of bounded arity constraints, and this decomposition does not hinder bound consistency. We will also use generalized arc consistency (GAC). A constraint is GAC iff every value for every variable can be extended to a solution of the constraint.

3 Taxonomy of global constraints

Global constraints over set and multiset variables can be composed from the following (more primitive) constraints:

Cardinality constraints: Many problems involve constraints on the cardinality of a set or multiset. For example, each shift must contain at least five nurses.

Intersection constraints: Many problems involve constraints on the intersection between any pair of sets or multisets. For example, shifts must have at least one person in common.

Partition constraints: Many problems involve partitioning a set or multiset. For example, orders must be partitioned to slabs in the steel mill slab design problem.

Ordering constraints: Many problems involve sets or multisets which are indistinguishable. For example, if each group in the social golfers problem is represented by a set then, as groups are symmetric, these sets can be permuted. We can break this symmetry by ordering the set variables.

Counting constraints: We often wish to model situations when there are constraints on the number of resources (values) used in a solution. For example, we might have set variables for the nurses on each shift and want to count the number of times each nurse has a shift during the monthly roster.

Weight and colour constraints: Many problems involve sets in which there is a weight or colour associated with each element of a set and there are constraints on the weights or colours in each set. For example, the weight of the set of orders assigned to a slab should be less than the slab capacity.

Tables 1 and 2 summarize some of the results presented in this paper. Given a collection of set or multiset variables, Table 1 shows different combinations of restrictions on the cardinality of the intersection of any pair of set or multiset variables (rows) with constraints restricting the cardinality of each set or multiset variable (columns). For instance, the top left corner is the `Disjoint` constraint, in which where all pairs of set or multiset variables are disjoint (i.e., their intersection is empty) and there is no restriction on the cardinality of the individual sets or multisets. On the other hand, the `NEDisjoint` also ensures that each set or multiset is non-empty. Table 2 is similar to Table 1, except that we also ensure that the set or multiset variables form a partition. Constraints like `Disjoint`, `Partition`, and `FCDisjoint` on set variables have already appeared in the literature [3, 5, 9, 10, 4].

All results apply to set or multiset variables unless otherwise indicated. In each entry, we name the resulting global constraint, state whether it is tractable to enforce BC on it and whether it is decomposable. For example, the `FCPartition` constraint over set variables (see Table 2) is not decomposable but we can maintain BC on it in polynomial time. Over multiset variables, the constraint becomes intractable.

	$\forall i < j \ \ldots$			
$\forall k \ \ldots$	$\|X_i \cap X_j\| = 0$	$\|X_i \cap X_j\| \leq k$	$\|X_i \cap X_j\| \geq k$	$\|X_i \cap X_j\| = k$
-	`Disjoint` polynomial *decomposable*	`Intersect`$_\leq$ polynomial *decomposable*	`Intersect`$_\geq$ polynomial *decomposable*	`Intersect`$_=$ NP-hard *not decomposable*
$\|X_k\| > 0$	`NEDisjoint` polynomial *not decomposable*	`NEIntersect`$_\leq$ polynomial *decomposable*	`NEIntersect`$_\geq$ polynomial *decomposable*	`FCIntersect`$_=$ NP-hard *not decomposable*
$\|X_k\| = m_k$	`FCDisjoint` poly on sets, NP-hard on multisets *not decomposable*	`FCIntersect`$_\leq$ NP-hard *not decomposable*	`FCIntersect`$_\geq$ NP-hard *not decomposable*	`NEIntersect`$_=$ NP-hard *not decomposable*

Table 1. Intersection $\times$ Cardinality

	$\bigcup_i X_i = X \ \wedge \ \forall i < j \ \ldots$			
$\forall k \ \ldots$	$\|X_i \cap X_j\| = 0$	$\|X_i \cap X_j\| \leq k$	$\|X_i \cap X_j\| \geq k$	$\|X_i \cap X_j\| = k$
-	`Partition`: polynomial *decomposable*	?	?	?
$\|X_k\| > 0$	`NEPartition`: polynomial *not decomposable*	?	?	?
$\|X_k\| = m_k$	`FCPartition` polynomial on sets, NP-hard on multisets *not decomposable*	?	?	?

Table 2. Partition + Intersection $\times$ Cardinality

4 Disjoint constraints

The `Disjoint` constraint on set or multiset variables is decomposable into binary empty intersection constraints without hindering bound consistency [12]. When it is over sets, it appears in a number of constraint solvers such as ILOG Solver (under the name `IlcAllNullIntersect`) and ECLiPSe. On multisets, the binary version of `Disjoint` appears in ILOG Configurator [4].

We now study bound consistency on the `NEDisjoint` and `FCDisjoint` constraints over set and multiset variables. These constraints are not decomposable so we present algorithms for enforcing BC on them or we prove intractability.

4.1 FCDisjoint

A filtering algorithm for `FCDisjoint` over set variables was independently proposed in [10]. We give here an alternative polynomial algorithm that uses a dual encoding with integer variables (also briefly described at the end of [10]). J.F. Puget has pointed out to us that this algorithm is very similar to the propagation algorithm used in ILOG Solver for the `IlcAllNullIntersect` constraint when cardinalities are specified for the set variables involved, and when the "extended" propagation mode is activated. We further show that bound consistency on `FCDisjoint` is NP-hard on multisets.

When $X_1, \ldots, X_n$ are set variables and $k_1, \ldots, k_n$ are given constants, we can achieve BC on a $\texttt{FCDisjoint}(X_1, \ldots, X_n, k_1, \ldots, k_n)$ constraint as follows:

Algorithm *`BC-FCD-Sets`*

1. For all $v \in \bigcup ub(X_i)$, introduce an integer variable Y_v with $dom(Y_v) = \{\}$
2. Initialize the domain of each Y_v as follows:
   (a) $dom(Y_v) \leftarrow \{i \mid v \in lb(X_i)\}$
   (b) if $|dom(Y_v)| > 1$ then **fail**
   (c) if $|dom(Y_v)| = 0$ then $dom(Y_v) \leftarrow \{i \mid v \in ub(X_i)\} \cup \{n+1\}$ /* n+1 is a dummy */
3. Maintain GAC on $\texttt{gcc}(Y, \{1..n+1\}, B)$ where Y is the array of Y_v's, and B is the array of the corresponding bounds of the i's where for all $i \leq n$ we have $B[i] = k_i..k_i$ and $B[n+1] = 0..\infty$
4. Maintain the following channelling constraints, for all $i \leq n$ and for all v:
   (a) $i \in dom(Y_v) \leftrightarrow v \in ub(X_i)$
   (b) $dom(Y_v) = \{i\} \leftrightarrow v \in lb(X_i)$

Remark. $\texttt{gcc}(Y, \{1..n+1\}, B)$ is the global cardinality constraint that imposes that in any assignment S of the variables Y, the value i from $\{1..n+1\}$ appears a number of times in the range $B[i]$. The dummy value $n+1$ is necessary to prevent a failure of the `gcc` when an Y_v cannot take any value in $1..n$ (i.e., value v cannot be used by any X_i).

We first prove the following lemma.

Lemma 1. *Define the one-to-one mapping between assignments S of the dual variables Y and assignments S' of the original set variables X_i by: $v \in S'[X_i]$ iff $S[Y_v] = i$. Then S is consistent with `gcc` in step (3) of `BC-FCD-Sets` iff S' is consistent for `FCDisjoint`.*

Proof. ($\Rightarrow$) We prove that S' is:

Disjoint: Each dual variable Y_v has a unique value, say i. Therefore in S' a value v cannot appear in more than one of the variables $X_1 \dots X_n$. In the case where $Y_v = n+1$, v does not belong to any set variable assignment.

Fixed Cardinality: `gcc` ensures that the values i are used by exactly k_i dual variables Y_{v_j}. Hence, $|S'[X_i]| = k_i$.

($\Leftarrow$) We prove that S is:

Consistent with gcc: By construction of Y, if $|S'[X_i]| = k_i$ for each $i \in 1..n$, each i will appear exactly k_i times in S, thus satisfying the `gcc`. (The dummy value $n+1$ has no restriction on its number of occurrences in Y.)

Consistent with Y domains: By construction. □

In the algorithm `BC-FCD-Sets`, let d be the number of Y_v variables introduced, where each Y_v has domain of size at most $n+1$.

Theorem 1. *`BC-FCD-Sets` is a sound and complete algorithm for enforcing bound consistency on `FCDisjoint` with set variables, that runs in $O(nd^2)$ time.*

Proof. Soundness. A value v is pruned from $ub(X_i)$ in step (4) of `BC-FCD-Sets` either because i was not put in $dom(Y_v)$ in step (2) or because the `gcc` has removed i from $dom(Y_v)$ in step (3). Lemma 1 tells us that both cases imply that v cannot belong to X_i in a satisfying tuple for `FCDisjoint`. A value v is added to $lb(X_i)$ in step (4) if $dom(Y_v) = \{i\}$ after applying GAC on the `gcc`. From Lemma 1 we deduce that any satisfying tuple for `FCDisjoint` necessarily contains v in X_i. We must also show that the algorithm does not fail if `FCDisjoint` can be made bound consistent. `BC-FCD-Sets` can fail in only two different ways. First, it fails in step (2) if a value belongs to two different lower bounds. Clearly, `FCDisjoint` cannot then be made bound consistent. Second, it fails in step (3) if the `gcc` cannot be made GAC. In this case, we know by Lemma 1 that `FCDisjoint` cannot then be made bound consistent.

Completeness. Let $v \in ub(X_i)$ after step (4). Then, $i \in dom(Y_v)$ after step (3). The `gcc` being GAC, there exists an assignment S satisfying `gcc`, with $S[Y_v] = i$. Lemma 1 guarantees there exists an assignment S' with $\{v\} \subseteq S'[X_i]$. In addition, let $v \notin lb(X_i)$ after step (4). Then, there exists $j \in dom(Y_v), j \neq i$, after step (3). Thus, there is an assignment S satisfying `gcc` with $S[Y_v] = j$. Lemma 1 tells us that there is a satisfying assignment S' of `FCDisjoint` with v not in $S'[X_i]$.

Complexity. Step (1) is in $O(d)$, and step (2) in $O(nd)$. Step (3) has the complexity of the `gcc`, namely $O(nd^2)$ since we have d variables with domains of size at most $n+1$. Step (4) is in $O(nd)$. Thus, `BC-FCD-Sets` is in $O(nd^2)$. □

Theorem 2. *Enforcing bound consistency on `FCDisjoint` with multiset variables is NP-hard.*

Proof. We transform 3SAT into the problem of the existence of a satisfying assignment for `FCDisjoint`. Let $F = \{c_1, \dots, c_m\}$ be a 3CNF on the Boolean variables $x_1, \dots, x_n$. We build the constraint $\texttt{FCDisjoint}(X_1, \dots, X_{3n+m}, k_1, \dots, k_{3n+m})$ as follows. Each time a Boolean variable x_i appears positively (resp. negatively) in a clause c_j, we create a value v_i^j (resp. w_i^j). For each Boolean variable x_i, we create two values p_i and n_i. Then, we build the $3n+m$ multiset variables as follows.

1. $\forall i \in 1..n$, /* X_i will take the p_i's iff $x_i = 1$ */
   (a) k_i = number of occurrences of x_i in a clause
   (b) $\{\} \subseteq X_i \subseteq \{v_i^j \mid x_i \in c_j\} \cup \{p_i, \ldots, p_i\}$ /*k_i copies of p_i*/
2. $\forall i \in n+1..2n$, /* X_i will take the n_i's iff $x_i = 0$ */
   (a) k_i = number of occurrences of $\neg x_i$ in a clause
   (b) $\{\} \subseteq X_i \subseteq \{w_i^j \mid \neg x_i \in c_j\} \cup \{n_i, \ldots, n_i\}$ /*k_i copies of n_i*/
3. $\forall i \in 2n+1..3n$, /* X_i forces X_{i-n} and X_{i-2n} to be consistent */
   (a) k_i = 1
   (b) $\{\} \subseteq X_i \subseteq \{n_i, p_i\}$
4. $\forall j \in 1..m$, /* X_{3n+j} represents the clause c_j */
   (a) k_{3n+j} = 1
   (b) $\{\} \subseteq X_{3n+j} \subseteq \{v_{i_1}^j, w_{i_2}^j, v_{i_3}^j\}$ if $c_j = x_{i_1} \vee \neg x_{i_2} \vee x_{i_3}$

Let M be a model of F. We build the assignment S on the X_i's such that $\forall i \in 1..n$, if $M[x_i] = 1$ then $S[X_i] = \{p_i, \ldots, p_i\}$, $S[X_{i+n}] = \{w_i^j \in ub(X_{i+n})\}$, $S[X_{i+2n}] = \{n_i\}$, else $S[X_i] = \{v_i^j \in ub(X_i)\}$, $S[X_{i+n}] = \{n_i, \ldots, n_i\}$, $S[X_{i+2n}] = \{p_i\}$.

By construction, the cardinalities k_i are satisfied and the disjointness are satisfied on $X_1 \ldots, X_{3n}$. In addition, the construction ensures that if a Boolean variable x_i is true in M (resp. false in M) none of the v_i^j (resp. w_i^j) are used and all the w_i^j (resp. v_i^j) are already taken by $X_1 \ldots, X_{3n}$. Thus, $\forall j \in 1..m$, $S[X_{3n+j}]$ is assigned one of the values v_i^j or w_i^j representing a true literal x_i or $\neg x_i$ in M. And M being a 3SAT model, we are sure that there exists such values not already taken by $X_1 \ldots, X_{3n}$. Therefore, S satisfies `FCDisjoint`.

Consider now an assignment S of the X_i's consistent with `FCDisjoint`. Build the interpretation M such that $M[x_i] = 1$ iff $S[X_{i+2n}] = \{n_i\}$. Thanks to the disjointness and cardinalities among $X_1 \ldots, X_{3n}$, we guarantee that if $S[X_{i+2n}] = \{n_i\}$ all the w_i^j are already taken by X_{i+n}, and if $S[X_{i+2n}] = \{p_i\}$ all the v_i^j are already taken by X_i, so that they cannot belong to any X_{3n+j}. But S satisfying `FCDisjoint`, we know that for each $j \in 1..m$, X_{3n+j} is assigned a value consistent with $X_1 \ldots, X_{3n}$. Therefore, M is a model of F.

As a result, deciding the existence of a satisfying assignment for `FCDisjoint` with multiset variables is NP-complete. Then, deciding whether GAC finds a wipe out on the occurrence representation is coNP-complete. In addition, on the transformation we use, if GAC detects a wipe then BC does[3] (because of the way p_i and n_i values are set). So, deciding whether BC detects a wipe out is coNP-complete, and enforcing bound consistency on `FCDisjoint` with multiset variables is NP-hard. □

4.2 NEDisjoint

The constraint `NEDisjoint`$(X_1, \ldots, X_n)$ on set variables can be seen as a particular case of constraint `FCDisjoint` in which the cardinality of the variables X_i can vary

[3] GAC on the occurrence representation of multisets is in general not equivalent to BC (whilst on sets it is). If $ub(X_1) = ub(X_2) = \{1, 1, 2, 2\}$, and $k_1 = k_2 = 2$, GAC on the occurrence representation of `FCDisjoint` removes the possibility for X_1 to have 1 occurrence of 1. BC does not remove anything since the bounds 0 and 2 for $occ(1, X_1)$ are consistent.

from 1 to ∞ instead of being fixed to k_i. Since the way the algorithm BC−FCD−Sets is written permits to express such an interval of values for the cardinality of the set variables X_i, the algorithm BC−NED−Sets is a very simple modification of it. In step (3) of BC−FCD−Sets it is indeed sufficient to assign $B[i]$ to $1..\infty$ instead of $k_i..k_i$, for $1 \leq i \leq n$. J.F. Puget has pointed out to us that the IlcAllNullIntersect constraint in "extended" mode will also achieve BC on non-empty set variables.

When NEDisjoint involves multiset variables, BC remains polynomial. In fact, it is sufficient to transform the multisets in sets and to use BC−NED−Sets on the obtained sets. Once BC achieved on these sets, we just have to restore the initial number of occurrences, noted init-occ, for each remaining value. The cardinality of the multisets are not bounded above, so that if one value has support, any number of occurrences of the same value have support also.

Algorithm BC−NED−Msets

1. **for each** $i \in 1..n$, v occurring in $ub(X_i)$ **do**
   init-occ$_{ub}(X_i, v) \leftarrow \texttt{occ}(v, ub(X_i))$; $\texttt{occ}(v, ub(X_i)) \leftarrow 1$
   init-occ$_{lb}(X_i, v) \leftarrow \texttt{occ}(v, lb(X_i))$; $\texttt{occ}(v, lb(X_i)) \leftarrow min(1, \text{init-occ}_{lb}(X_i, v))$
2. BC−NED−Sets$(X_1, \ldots, X_n)$
3. **for each** $i \in 1..n$, $v \in ub(X_i)$ **do**
   $\texttt{occ}(v, ub(X_i)) \leftarrow$ init-occ$_{ub}(X_i, v)$
   if $v \in lb(X_i)$ **then** $\texttt{occ}(v, lb(X_i)) \leftarrow max(1, \text{init-occ}_{lb}(X_i, v))$

5 Partition constraints

The Partition constraint is decomposable into binary empty intersection constraints and ternary union constraints involving n additional variables without hindering bound consistency [12]. It appears in a number of constraint solvers such as ILOG Solver (under the name IlcPartition) and ECLiPSe when it is over sets. On the other hand, the non-empty and fixed cardinality partition constraints are not decomposable. We therefore present algorithms for enforcing BC on them or we prove intractability.

5.1 FCPartition

It is polynomial to enforce BC on the FCPartition constraint on set variables, but NP-hard on multisets. On set variables, enforcing BC on FCPartition is very similar to enforcing BC on FCDisjoint. Indeed, if the set X being partitioned is fixed, then we can simply decompose a fixed cardinality partition constraint into a fixed cardinality disjoint, union and cardinality constraints without hindering bound consistency.[4] If X is not fixed, we need to do slightly more reasoning to ensure that the X_i's are a partition of X. We present here the additional lines necessary to deal with this.

Line numbers with a prime represent lines modified wrt BC−FCD−Sets. The others are additional lines.

[4] As in the FCDisjoint case, J.F. Puget tells us that the filtering algorithm of the IlcPartition constraint in [5] uses a similar approach when the "extended" mode is set.

Algorithm `BC-FCP-Sets`

1'. For all $v \in ub(X)$, introduce an integer variable Y_v with $dom(Y_v) = \{\}$
2. Initialize the domain of each Y_v as follows:
  ...
  (c') if $|dom(Y_v)| = 0$ then $dom(Y_v) \leftarrow \{i \mid v \in ub(X_i)\}$
  (d) if $v \notin lb(X)$ then $dom(Y_v) \leftarrow dom(Y_v) \cup \{n+1\}$
  (e) if $|dom(Y_v)| = 0$ then **fail**
...
4. Maintain the following channelling constraints, for all $i \leq n$ and for all v:
  ...
  (c) $n+1 \notin dom(Y_v) \leftrightarrow v \in lb(X)$
  (d) $ub(X) \subseteq \bigcup ub(X_i)$

Lemma 2. *Define the one-to-one mapping between assignments S of the dual variables Y and assignments S' of the original set variables X_i and X by: $S'[X] = \bigcup S'[X_i]$ and $v \in S'[X_i]$ iff $S[Y_v] = i$. Then S is consistent with* `gcc` *in step (3) of* `BC-FCP-Sets` *iff S' is consistent for* `FCPartition`*.*

Proof. ($\Rightarrow$) We prove that S' is:
Disjoint and Fixed Cardinality: See Lemma 1.
Partition: Lines (2.c'-d) guarantee that for a value $v \in lb(X)$, Y_v cannot be assigned the dummy value $n+1$ in S. Hence, S' necessarily has an X_i with $v \in S'[X_i]$. Because of line (1'), none of the Y_v represent a value $v \notin ub(X)$. Hence, for all i, $S'[X_i] \subseteq ub(X)$, then $S'[X] \subseteq ub(X)$.
($\Leftarrow$) We prove that S is:
Consistent with `gcc`*:* See Lemma 1.
Consistent with Y: If S' is a satisfying assignment for `FCPartition`, $S'[X_i] \subseteq S'[X], \forall i$. Since $S'[X] \subseteq ub(X)$, we know that any value v appearing in S' has a corresponding variable Y_v. And by construction (lines 2.a, 2.c, 2.d, we know that S is consistent with Y domains. □

Theorem 3. `BC-FCP-Sets` *is a sound and complete algorithm for enforcing bound consistency on* `FCPartition` *with set variables that runs in $O(nd^2)$ time.*

Proof. Soundness. A value v is pruned from $ub(X_i)$ in step (4) of `BC-FCP-Sets` for one of the reasons that already held in `FCDisjoint` or because Y_v has not been created in line (1'). Lemma 2 tells us that all cases imply that v cannot belong to X_i in a satisfying tuple for `FCPartition`. Soundness of $lb(X_i)$ comes from Lemma 2 as it came from Lemma 1 on `FCDisjoint`. We must also show that the algorithm does not fail if `FCPartition` can be made bound consistent. `BC-FCP-Sets` can fail in line (2.e) if a value v that belongs to $lb(X)$ cannot belong to any X_i. Clearly, `FCPartition` cannot then be made bound consistent. The other cases of failure are the same as for `FCDisjoint`. A value v is pruned from $ub(X)$ in step (4.d) because none of the X_i contains v in its upper bound. This means that this value cannot belong to any satisfying assignment S' (Lemma 2). A value v is added to $lb(X)$ in line (4.c) if no assignment S satisfying the `gcc` verifies $S[Y_v] = n+1$. This means that v is assigned to a variable X_i in all assignments satisfying `FCPartition`.

Completeness. Let $v \in ub(X)$ after step (4). Then, there exists X_i with $v \in ub(X_i)$, and so $i \in dom(Y_v)$ after step (3). `gcc` being GAC, there exists an assignment S satisfying `gcc`, with $S[Y_v] = i$. Lemma 2 guarantees there exists an assignment S' with $\{v\} \subseteq S'[X]$. Thus, v is in $ub(X)$. In addition, let $v \notin lb(X)$ after step (4). Then, $n+1 \in dom(Y_v)$ after step (3). Thus, there is an assignment S satisfying `gcc` with $S[Y_v] = n+1$. Lemma 2 tells us that there is a satisfying assignment S' of `FCPartition` with v not in $S'[X]$.

Complexity. See proof of `BC-FCD-Sets`. □

Theorem 4. *Enforcing BC on* `FCPartition`$(X_1, \ldots, X_n, X, k_1, \ldots, k_n)$ *with multiset variables is NP-hard.*

Proof. We know that deciding the existence of a satisfying assignment is NP-complete for `FCDisjoint`$(X_1, \ldots, X_n, k_1, \ldots, k_n)$ with multiset variables. If we build a multiset variable X with $lb(X) = \emptyset$ and $ub(X) = \bigcup_i ub(X_i)$, then `FCPartition`$(X_1, \ldots, X_n, X, k_1, \ldots, k_n)$ has a satisfying assignment if and only if `FCDisjoint`$(X_1, \ldots, X_n, k_1, \ldots, k_n)$ has one. Thus, enforcing bound consistency on `FCPartition` is NP-hard. □

5.2 NEPartition

The constraint `NEDisjoint`$(X_1, \ldots, X_n)$ on set variables was a particular case of `FCDisjoint` in which the cardinality of the variables X_i can vary from 1 to ∞ instead of being fixed to k_i. This is exactly the same for `NEPartition` on set variables, which is a particular case of `FCPartition`. Replacing "$B[i] \leftarrow k_i..k_i$" by "$B[i] \leftarrow 1..\infty$" in `BC-FCP-Sets`, we obtain `BC-NEP-Sets`.

When `NEPartition` involves multiset variables, BC remains polynomial. As for `BC-NED-Msets`, the trick is to transform multisets in sets and to use `BC-NEP-Sets` on the obtained sets. We just need to be careful with the compatibility of the occurrences of values in X_i variables and the X being partitioned. Once BC is achieved on these sets, we have to restore the initial number of occurrences and check again compatibility with X.

Algorithm `BC-NEP-Msets`

1. **if** $\bigcup_i lb(X_i) \not\subseteq ub(X)$ **or** $lb(X) \not\subseteq \bigcup_i ub(X_i)$ **then** failure
2. **for each** $i \in 1..n$, v occurring in $ub(X_i)$ **do**
   - 2.1. **if** $\texttt{occ}(v, ub(X_i)) < \texttt{occ}(v, lb(X))$ **then** $\texttt{occ}(v, ub(X_i)) \leftarrow 0$
   - 2.2. **if** $\texttt{occ}(v, ub(X_i)) > \texttt{occ}(v, ub(X))$ **then** $\texttt{occ}(v, ub(X_i)) \leftarrow \texttt{occ}(v, ub(X))$
   - 2.3. $\text{init-occ}_{ub}(X_i, v) \leftarrow \texttt{occ}(v, ub(X_i))$; $\texttt{occ}(v, ub(X_i)) \leftarrow 1$
   - 2.4. $\text{init-occ}_{lb}(X_i, v) \leftarrow \texttt{occ}(v, lb(X_i))$; $\texttt{occ}(v, lb(X_i)) \leftarrow min(1, \text{init-occ}_{lb}(X_i, v))$
3. store $lb(X)$; $lb(X) \leftarrow \texttt{set-of}(lb(X))$; $ub(X) \leftarrow \texttt{set-of}(ub(X))$
4. `BC-NEP-Sets`$(X_1, \ldots, X_n, X)$
5. restore $lb(X)$
6. **for each** $i \in 1..n$, $v \in ub(X_i)$ **do**
   - 6.1. $\texttt{occ}(v, ub(X_i)) \leftarrow \text{init-occ}_{ub}(X_i, v)$
   - 6.2. **if** $v \in lb(X_i)$ **then** $\texttt{occ}(v, lb(X_i)) \leftarrow max(1, \text{init-occ}_{lb}(X_i, v), \texttt{occ}(v, lb(X)))$
7. $lb(X) \leftarrow lb(X) \cup \bigcup_i lb(X_i)$; $ub(X) \leftarrow \bigcup_i ub(X_i)$

Theorem 5. *`BC-NEP-Msets` is a sound and complete algorithm for enforcing bound consistency on `NEPartition` with multiset variables, that runs in $O(nd^2)$ time.*

Proof. (Sketch.) As for `NEDisjoint` on multiset variables, enforcing bound consistency on `NEPartition` after having transformed the multisets in sets (i.e., we keep only one occurrence of each value in the lower and upper bounds), the removal of a value v from an upper bound by `BC-NEP-Sets` is a sufficient condition for the removal of all occurrences of v in the original multiset upper bound. The addition v to a lower bound is a sufficient condition for the addition of some occurrences of v in the lower bound (the right number depends on the number of occurrences of v in $lb(X)$ and in the lower bound of the X_i holding v. It is then sufficient to ensure consistency between the number of occurrences in the X_i and X (lines 1, 2.1, 2.2, and 7), to transform multisets in sets (lines 2.3, 2.4, and 3), to call `BC-NEP-Sets` (line 4), and to restore appropriate numbers of occurrences (lines 5 and 6). Line 1 guarantees that $ub(X)$ can cover all the X_i's lower bounds and that $lb(X)$ can be covered by the X_i's upper bounds. A value v can be assigned in X_i if and only if it can cover the occurrences of v in $lb(X)$ (line 2.1), and it cannot occur more than in $ub(X)$ (line 2.2). Finally, a value v occurs in $lb(X)$ at least as many times as it occurs in some $lb(X_i)$, and occurs in $ub(X)$ exactly as many times as in the $ub(X_i)$ having its greatest number of occurrences (line 7). The complexity is dominated by line 4, with the call to `BC-NEP-Sets` which is $O(nd^2)$. □

6 Intersection constraints

The `Disjoint` constraint restricts the pair-wise intersection of any two set or multiset variables to the empty set. We now consider the cases where the cardinality of the pair-wise intersection is either bounded or equal to a given constant or integer variable (lower case characters denote constants while upper case denote variables):

$\texttt{Intersect}_{\leq}(X_1, \ldots, X_n, K)$ iff $|X_i \cap X_j| \leq K$ for any $i \neq j$.
$\texttt{Intersect}_{\geq}(X_1, \ldots, X_n, K)$ iff $|X_i \cap X_j| \geq K$ for any $i \neq j$.
$\texttt{Intersect}_{=}(X_1, \ldots, X_n, K)$ iff $|X_i \cap X_j| = K$ for any $i \neq j$.

As usual, we can also add non-emptiness and fixed cardinality constraints to the set or multiset variables. For example, $\texttt{FCIntersect}_{\leq}(X_1, \ldots, X_n, K, C)$ iff $|X_i \cap X_j| \leq K$ for any $i \neq j$ and $|X_i| = C$ for all i. If $K = 0$, $\texttt{Intersect}_{\leq}$ and $\texttt{Intersect}_{=}$ are equivalent to `Disjoint`.

6.1 At most intersection constraints

We show that $\texttt{Intersect}_{\leq}$ and $\texttt{NEIntersect}_{\leq}$ can be decomposed without hindering bound consistency, but that it is NP-hard to enforce BC on $\texttt{FCIntersect}_{\leq}$.

Theorem 6. *$BC(\texttt{Intersect}_{\leq}(X_1, \ldots, X_n, K))$ is equivalent to $BC(|X_i \cap X_j| \leq K)$ for all $i < j$.*

Proof. Suppose $BC(|X_i \cap X_j| \leq K)$ for all $i < j$. We will show that $BC(\forall i < j. |X_i \cap X_j| \leq K)$. Consider the occurrence representation of the set or multiset variables. Let

X_{il} be the number of occurrences of the value l in X_i. Consider the upper bound on X_{il}. We will construct a support for this value for X_{il} that simultaneously satisfies $|X_i \cap X_j| \leq K$ for all $i < j$. The same support will work for the lower bound on X_{il}. First, we assign K with its upper bound. Then we pick any j with $i \neq j$. As $BC(|X_i \cap X_j| \leq K)$, there is some assignment for X_{jn} and X_{im} ($l \neq m$) within their bounds that satisfies $|X_i \cap X_j| \leq K$. We now extend these assignments to get a complete assignment for every other set or multiset variable as follows. Every other X_{pq} ($p \neq i$ and $p \neq j$) is assigned its lower bound. This can only help satisfy $|X_i \cap X_j| \leq K$ for all $i < j$. This assignment is therefore support for X_{il}. We can also construct support for the upper of lower bound of K in a similar way. Maximality of the bound consistent domains is easy. Consider any value for X_{il} smaller than the lower bound or larger than the upper bound. As this cannot be extended to satisfy $|X_i \cap X_j| \leq K$ for some j, it clearly cannot be extended to satisfy $|X_i \cap X_j| \leq K$ for all $i < j$. A similar argument holds for any value for K smaller than the lower bound or larger than the upper bound. Hence, $BC(\forall i < j . |X_i \cap X_j| \leq K)$. □

Given a set of set or multiset variables, the non-empty intersection constraint `NEIntersect`$_{\leq}(X_1, \ldots, X_n, K)$ ensures that $|X_i \cap X_j| \leq K$ for $i \neq j$ and $|X_i| > 0$ for all i. If $K = 0$, this is the `NEDisjoint` constraint which is not decomposable. If $K > 0$, the constraint is decomposable.

Theorem 7. *If $K > 0$ then $BC($`NEIntersect`$_{\leq}(X_1, \ldots, X_n, K))$ is equivalent to $BC(|X_i \cap X_j| \leq K)$ for all $i < j$ and $BC(|X_i| > 0)$ for all i.*

Proof. Suppose $BC(|X_i \cap X_j| \leq K)$ for all $i < j$ and $BC(|X_i| > 0)$ for all i. Then $|lb(X_i) \cap lb(X_j)| \leq max(K)$ for all $i < j$. And if $|ub(X_i)| = 1$ for any i then $lb(X_i) = ub(X_i)$. Consider some variable X_i and any value $a \in ub(X_i) - lb(X_i)$. We will find support in the global constraint for X_i to take the value $\{a\} \cup lb(X_i)$. Consider any other variable X_j. If $|lb(X_j)| = 0$ then we pick any value $b \in ub(X_j)$ and set X_j to $\{b\}$. This will ensure we satisfy the non-emptiness constraint on X_j. As $k > 0$ and $|X_j| = 1$, we will satisfy the intersection constraint between X_j and any other variable. If $|lb(X_j)| > 0$ then we set X_j to $lb(X_j)$. This again satisfy the non-emptiness constraint on X_j. Since $|lb(X_i) \cap lb(X_j)| \leq max(K)$ for all $i < j$, we will also satisfy the intersection constraints. Support can be found in a similar way for X_i to take the value $lb(X_i)$ if this is non-empty. Finally, $min(K)$ has support since $BC(|X_i \cap X_j| \leq K)$ for all $i < j$. Hence `NEIntersect`$_{\leq}(X_1, \ldots, X_n, K)$ is BC. □

Enforcing BC on `FCIntersect`$_{\leq}$ is intractable.

Theorem 8. *Enforcing BC on `FCIntersect`$_{\leq}(X_1, \ldots, X_n, k, c)$ for $c > k > 0$ is NP-hard.*

Proof. Immediate from Theorem 5 in [2]. □

Sadler and Gervet introduce the `atmost1-incommon` and `distinct` constraints for set variables with a fixed cardinality [9]. The `atmost1-incommon` constraint is `FCIntersect`$_{\leq}(X_1, \ldots, X_n, 1, c)$. Similarly, the `distinct` constraint on sets of fixed cardinality is is `FCIntersect`$_{\leq}(X_1, \ldots, X_n, c-1, c)$. The reduction used in Theorem 5 in [2] works with all these parameters. Hence, all are NP-hard to propagate.

6.2 At least intersection constraints

Similar to the at most intersection constraint, $\texttt{Intersect}_{\geq}$ and $\texttt{NEIntersect}_{\geq}$ can be decomposed without hindering bound consistency. However, it is NP-hard to enforce BC on $\texttt{FCIntersect}_{\geq}$.

Theorem 9. *$BC(\texttt{Intersect}_{\geq}(X_1, \ldots, X_n, K))$ is equivalent to $BC(|X_i \cap X_j| \geq K)$ for all $i < j$.*

Proof. The proof is analogous to that of Theorem 6 except we extend a partial assignment to a complete assignment that is interval support by assigning each of the additional X_{pq} with the upper bound and (where appropriate) K with its lower bound. □

Two sets cannot have an intersection unless they are non-empty. Hence this result also shows that $BC(\texttt{NEIntersect}_{\geq}(X_1, \ldots, X_n, K))$ for $K > 0$ is equivalent to BC on the decomposition. By comparison, enforcing BC on $\texttt{FCIntersect}_{\geq}$ is intractable.

Theorem 10. *Enforcing BC on $\texttt{FCIntersect}_{\geq}(X_1, \ldots, X_n, k, c)$ for $c > k > 0$ is NP-hard.*

Proof. We let $k = 1$. We can reduce the $k = 1$ case to the $k > 1$ case by adding $k - 1$ additional common values to each set variable. The proof again uses a reduction of a 3SAT problem in n variables. The same reduction is used for set or multiset variables. We let $c = n$ and introduce a set variable, S with domain $\{\} \subseteq S \subseteq \{1, \neg 1, \ldots, n, \neg n\}$. This will be set of literals assigned true in a satisfying assignment. For each clause, φ we introduce a set variable, X_φ. Suppose $\varphi = x_i \vee \neg x_j \vee x_k$, then X_φ has domain $\{d_1^\varphi, \ldots, d_{n-1}^\varphi\} \subseteq X_\varphi \subseteq \{i, \neg j, k, d_1^\varphi, \ldots, d_{n-1}^\varphi\}$, where $d_1^\varphi, \ldots, d_{n-1}^\varphi$ are dummy values. To satisfy the intersection and cardinality constraint, S must take at least one of the literals which satisfy φ. Finally, we introduce n set variables, X_i to ensure that one and only one of i and $\neg i$ is in S. Each X_i has domain $\{f_1^i, \ldots, f_{n-1}^i\} \subseteq X_i \subseteq \{f_1^i, \ldots, f_{n-1}^i, i, \neg i\}$. The constructed set variables then have a solution which satisfies the intersection and cardinality constraints iff the original 3SAT problem is satisfiable. Hence enforcing bound consistency is NP-hard. □

6.3 Equal intersection constraints

Unlike the at most or at least intersection constraints, enforcing BC on $\texttt{Intersect}_{=}$ is intractable even without cardinality constraints on the set or multiset variables.

Theorem 11. *Enforcing BC on $\texttt{Intersect}_{=}(X_1, \ldots, X_n, k)$ is NP-hard for $k > 0$.*

Proof. Immediate from Theorem 6 in [2]. □

The same reduction can also be used with the constraint that each set or multiset has a non-empty or fixed cardinality.

Lemma 3. *Enforcing BC on $\texttt{FCIntersect}_{=}(X_1, \ldots, X_n, k)$ is NP-hard for $k > 0$.*

Lemma 4. *Enforcing BC on $\texttt{NEIntersect}_{=}(X_1, \ldots, X_n, k, c)$ is NP-hard for $k > 0$.*

7 Experimental results

To show the benefits of these global constraints, we ran some experiments using ILOG's Solver toolkit with a popular benchmark involving set variables. The social golfers problem $\langle p, m, n, t \rangle$ is to schedule t golfers into m groups of size n for p weeks, such that no golfer plays in the same group as any other golfer twice. To model this problem, we introduce a set variable of fixed cardinality to represent every group in each week. Each week is then a partition of the set of golfers. Between any two groups, their intersection must contain at most one golfer. We also consider a generalization of the problem in which there is an excess of golfers and some golfers rest each week. When there is no excess of golfers, `FCPartition` shows no improvement upon its decomposition into ILOG's `IlcPartition` and cardinality constraints. When there is an excess of golfers, the partitioning constraint is replaced by a disjointness constraint.

We compare the same model using the `FCDisjoint` constraint and its decomposition into ILOG's `IlcAllNullIntersect` constraint and cardinality constraints on groups. In the latter case, the filtering level is fixed either to "Default" or "Extended". We understand from conversations with Ilog that "Default" implements the decomposition whilst "Extended" enforces BC on the global constraint. We ran experiments with a time limit of 10 minutes, and five settings for m and n. For each, we present the results for all numbers p of weeks such that at least one strategy needs at least one fail, and at least one strategy can solve the problem within the time limit. We solved each problem using five different variable ordering strategies:

- **static golfer:** picks each golfer in turn, and assigns him to the first possible group of every week.
- **static week:** picks each golfer in turn, and assigns him to one group in the first incomplete week.
- **min domain:** picks a pair (golfer, week) such that the total number of groups in which the golfer can participate in during the given week is minimum, then assigns this golfer to one group.
- **default (group):** ILOG Solver's default strategy for set variables ordered by groups; this picks an element $v \in ub(S)$ and adds it to the lower bound ($v \in S$).
- **default (week):** ILOG Solver's default strategy for set variables ordered by weeks.

We observe that, in terms of fails, `FCDisjoint` and `IlcAllNullIntersect`-Extended are equivalent, with two exceptions[5]. Both outperform the decomposition model or are the same. The runtimes follow a similar behaviour, although the decomposition model can be faster when the number of fails are equal. The speed-up obtained by reasoning on the global constraint rather than on disjointness and cardinality separately can be of several orders of magnitude in some cases. With the two default heuristics (last two columns in the table), we notice no difference between our global constraint and the decomposition. These heuristics are not, however, always the best. The min domain heuristic can be superior, but sometimes needs the pruning provided by the global constraint to prevent poor performance.

[5] We do not understand these two exceptions but suspect there may be some complex interaction with the dynamic branching heuristic.

		Number of Fails / CPU Time (s)				
problem	model	static golfer	static week	min domain	group (set)	week (set)
$\langle 6, 8, 4, 36 \rangle$	`FCDisjoint`	**10** / 0.15	-	52 / 0.14	183 / 0.11	-
	`IlcAllNullIntersect` (Extended)	**10** / 0.13	-	52 / 0.11	183 / 0.11	-
	`IlcAllNullIntersect` (Default)	-	-	190 / 0.13	183 / **0.08**	-
$\langle 3, 6, 6, 37 \rangle$	`FCDisjoint`	-	548 / 0.21	**0** / 0.02	27 / 0.02	22232 / 2.36
	`IlcAllNullIntersect` (Extended)	-	548 / 0.2	**0** / 0.02	27 / 0.03	22232 / 1.6
	`IlcAllNullIntersect` (Default)	-	-	**0** / **0.01**	27 / 0.02	22232 / 1.3
$\langle 3, 6, 6, 38 \rangle$	`FCDisjoint`	-	67 / 0.03	**0** / 0.03	4 / 0.02	3446 / 0.39
	`IlcAllNullIntersect` (Extended)	-	67 / 0.04	**0** / 0.02	4 / 0.03	3446 / 0.26
	`IlcAllNullIntersect` (Default)	-	-	**0** / **0.01**	4 / 0.02	3446 / 0.2
$\langle 3, 6, 6, 39 \rangle$	`FCDisjoint`	-	1261 / 0.3	**0** / **0.02**	7 / 0.03	171574 / 16.52
	`IlcAllNullIntersect` (Extended)	-	1261 / 0.27	**0** / **0.02**	7 / 0.02	171574 / 11.39
	`IlcAllNullIntersect` (Default)	-	-	**0** / **0.02**	7 / **0.02**	171574 / 8.85
$\langle 3, 6, 6, 40 \rangle$	`FCDisjoint`	12 / **0.02**	48 / 0.03	**0** / **0.02**	**0** / **0.02**	8767 / 0.79
	`IlcAllNullIntersect` (Extended)	12 / 0.03	48 / 0.03	**0** / **0.02**	**0** / 0.03	8767 / 0.6
	`IlcAllNullIntersect` (Default)	-	-	**0** / **0.02**	**0** / **0.02**	8767 / 0.46
$\langle 3, 5, 5, 26 \rangle$	`FCDisjoint`	-	44 / 0.03	**0** / 0.01	2 / **0**	813 / 0.08
	`IlcAllNullIntersect` (Extended)	-	44 / 0.02	**0** / 0.01	2 / 0.01	813 / 0.07
	`IlcAllNullIntersect` (Default)	-	177880 / 9.62	**0** / 0.01	2 / **0**	813 / 0.05
$\langle 3, 5, 5, 27 \rangle$	`FCDisjoint`	967161 / 160.92	5 / 0.01	**0** / 0.01	1 / 0.01	62 / 0.01
	`IlcAllNullIntersect` (Extended)	967161 / 96.94	5 / 0.01	**0** / 0.02	1 / 0.01	62 / 0.01
	`IlcAllNullIntersect` (Default)	-	1106 / 0.09	**0** / **0**	1 / 0.01	62 / 0.02
$\langle 3, 5, 5, 28 \rangle$	`FCDisjoint`	9 / **0.01**	32 / 0.03	**0** / **0.01**	19 / **0.01**	661 / 0.08
	`IlcAllNullIntersect` (Extended)	9 / **0.01**	32 / **0.01**	**0** / **0.01**	19 / **0.01**	661 / 0.06
	`IlcAllNullIntersect` (Default)	58218 / 3.65	22860 / 1.23	**0** / **0.01**	19 / **0.01**	661 / 0.05
$\langle 3, 5, 5, 29 \rangle$	`FCDisjoint`	6 / 0.02	2 / 0.01	**0** / 0.01	**0** / 0,01	18 / 0.01
	`IlcAllNullIntersect` (Extended)	6 / 0.01	2 / 0.01	**0** / 0.01	**0** / 0,01	18 / 0.01
	`IlcAllNullIntersect` (Default)	37208 / 2.25	209 / 0.02	**0** / 0.01	**0** / **0**	18 / 0.01
$\langle 3, 9, 9, 83 \rangle$	`FCDisjoint`	-	-	**0** / 0.12	453 / 0.17	-
	`IlcAllNullIntersect` (Extended)	-	-	-	453 / 0.13	-
	`IlcAllNullIntersect` (Default)	-	-	-	453 / **0.11**	-
$\langle 3, 9, 9, 84 \rangle$	`FCDisjoint`	-	-	**0** / 0.12	5 / 0.09	-
	`IlcAllNullIntersect` (Extended)	-	-	5 / 0.13	5 / 0.1	-
	`IlcAllNullIntersect` (Default)	-	-	-	5 / **0.08**	-
$\langle 3, 9, 9, 85 \rangle$	`FCDisjoint`	-	-	**0** / 0.13	30 / 0.09	-
	`IlcAllNullIntersect` (Extended)	-	-	**0** / 0.13	30 / 0.1	-
	`IlcAllNullIntersect` (Default)	-	-	1442064 / 159.75	30 / **0.08**	-
$\langle 10, 9, 3, 30 \rangle$	`FCDisjoint`	464 / 0.84	264 / 0.45	-	16055 / 3.75	**15** / 0.26
	`IlcAllNullIntersect` (Extended)	464 / 0.56	264 / 0.32	-	16055 / 2.2	**15** / 0.23
	`IlcAllNullIntersect` (Default)	-	-	-	16055 / 1.99	**15** / **0.22**
$\langle 10, 9, 3, 31 \rangle$	`FCDisjoint`	37 / 0.46	1 / 0.25	**0** / 0.39	2 / 0.28	113 / 0.29
	`IlcAllNullIntersect` (Extended)	37 / 0.41	1 / 0.24	**0** / 0.32	2 / 0.26	113 / 0.24
	`IlcAllNullIntersect` (Default)	-	51223 / 10.45	**0** / 0.32	2 / 0.25	113 / **0.23**

8 Conclusions

We have begun a systematic study of global constraints on set and multiset variables. We have studied here a wide range of disjoint, partition, and intersection constraints. The disjoint constraint on set or multiset variables is decomposable (and hence polynomial). On the other hand, the non-empty and fixed cardinality disjoint constraints are not decomposable without hindering bound consistency. We therefore present polynomial algorithms for enforcing bound consistency on the non-empty disjoint constraints for set or multiset variables, for enforcing BC on the fixed cardinality disjoint constraint for set variables, and prove that enforcing BC on the fixed cardinality disjoint constraint on multiset variables is NP-hard. We give very similar results for the partition, non-empty and fixed cardinality partition constraints. We also identify those non-empty intersection constraints which are decomposable, those which are not decomposable but polynomial, and those that are NP-hard. Many of the propagation algorithms we propose here exploit a dual viewpoint, and call upon existing global constraints for finite-domain variables like the global cardinality constraint. We are currently extending this study to counting constraints on set and multiset variables. Propagation algorithms for such constraints also appear to exploit dual viewpoints extensively.

References

1. N. Beldiceanu. Global constraints as graph properties on a structured network of elementary constraints of the same type. In *Proc. CP'00*, pages 52–66. Springer, 2000.
2. C. Bessiere, E. Hebrard, B. Hnich, and T. Walsh. The complexity of global constraints. In *Proc. AAAI'04*, 2004.
3. C. Gervet. Conjunto: constraint logic programming with finite set domains. *Proc. of the 1994 Int. Symp. on Logic Programming*, pages 339–358. MIT Press, 1994.
4. Ilog. User's manual. ILOG Configurator 2.3. 2004.
5. Ilog. User's manual. ILOG Solver 6.0. Sept. 2003.
6. T. Müller and M. Müller. Finite set constraints in Oz. *13th Logic Programming Workshop*, pages 104–115, Technische Universität München, 1997.
7. J-C. Régin. A filtering algorithm for constraints of difference in CSPs. In *Proc. AAAI'94*, pages 362–367. AAAI, 1994.
8. J-C. Régin. Generalized arc consistency for global cardinality constraints. In *Proc. AAAI'96*, pages 209–215. AAAI Press/The MIT Press, 1996.
9. A. Sadler and C. Gervet. Global reasoning on sets. In *Proc. of Workshop on Modelling and Problem Formulation (FORMUL'01)*, 2001. Held alongside CP-01.
10. A. Sadler and C. Gervet. Global Filtering for the Disjointness Constraint on Fixed Cardinality Sets. In Technical report IC-PARC-04-02, Imperial College London, March 2001.
11. J. Schimpf, A. Cheadle, W. Harvey, A. Sadler, K. Shen, and M. Wallace. ECLiPSe. In Technical report IC-PARC-03-01, Imperial College London, 2003.
12. T. Walsh. Consistency and propagation with multiset constraints: A formal viewpoint. In *Proc. CP'03*. Springer, 2003.