TECHNICAL REPORT IC-PARC-04-02

Global Filtering for the Disjointness Constraint on Fixed Cardinality Sets

Andrew Sadler and Carmen Gervet

IC-PARC
Centre for Planning and Resource Control
William Penney Laboratory
Imperial College London
London
SW7 2AZ

Global Filtering for the Disjointness Constraint on Fixed Cardinality Sets

March 2004

Abstract

Finite set constraints represent a natural choice to model configuration design problems using set cardinality and disjointness, covering or partition constraints over (families of) set variables. Such constraints are available in most set-based constraint languages, often in the form of n-ary decomposable constraints. The corresponding filtering algorithms make use of local bound consistency techniques. In this paper we show that when the set cardinality constraints are handled together with the n-ary constraints, and set variable domains are specified by set intervals, efficient global filtering algorithms can be derived. We consider the particular case of the n-ary disjoint constraint `disjoint(`$[X_1,\ldots,X_n],[c_1,\ldots,c_n]$`)` for a family of pairwise disjoint sets X_i of fixed cardinality c_i. We present a set of conditions and inference rules to infer Bounds Consistency (BC), together with an efficient global filtering algorithm. We also explain why this level of pruning cannot be achieved with a common FD formulation based on the alldiff constraint, enriched with lexicographic ordering constraints; but is actually equivalent to a dual FD representation based on the Generalized Cardinality Constraint.

1 Introduction and previous work

Finite set constraints solvers have been embedded in a growing number of CP languages such as Conjunto [7], Ilog Solver [11], MOZART [14], FaCiLe [2], CHOCO [12] and shown their strengths in modelling configuration design problems such as matching problems, and handling symmetries with a natural mathematical formulation [6, 3]. Conjunto and its peers comprise the usual set operation symbols ($\cap, \cup, \setminus$), the set cardinality relation ($|\ \ |$) and the set inclusion relation ($\subseteq$). For practical modelling reasons most languages provide a set of *n*-ary constraints which are syntactic abstractions for a collection of respectively binary and ternary constraints. Set variables range over set domains (sets of sets) specified by intervals whose lower and upper bounds are known sets, ordered by set inclusion, e.g. $X \in [\{1\},\{1,2,3\}]$. The lower bound, denoted $glb(X)$, contains the definite elements of the set (1) while the upper bound $lub(X)$, contains in addition the potential elements $(2,3)$. The constraint reasoning is based on local bound consistency techniques extended to handle set constraints [8].

Previous studies in constraint programming have demonstrated the importance of designing global filtering algorithms for specific classes of constraints. It has essentially been the realm of Finite Domain (FD) constraints pioneered by [4] and [15]'s work on the `alldiff`. We are not aware of any global filtering method for some Finite Set constraints, except some preliminary results in [17]. In this paper, we show that when the set cardinality constraints are considered together with n-ary set constraints, a global filtering can be enforced efficiently.

In this paper, we consider the particular case of `disjoint(`$[X_1,\ldots,X_n],[c_1,\ldots,c_n]$`)`, an n-ary set constraint of pairwise disjoint sets of fixed cardinalities. Fixed cardinality sets occur in most configuration design problems and thus cover a large class of practical problems. However, when the set cardinality is known, the global disjoint constraint can be formulated by an equivalent FD model: i.e. where c_i FD variables are created to represent each set, with domain the l.u.b. of the corresponding set, and the `alldiff` constraint is applied to all variables. Symmetries may be removed by having a lexicographic ordering constraint between variables representing one set. For example the constraint system over sets:

```
X in [{},{1,2,3,4}], Y in [{}, {1,3,4}]
disjoint([X,Y], [2,2])
```

can be modeled as:

```
X1,X2 in [1,2,3,4],  Y1,Y2 in [1,3,4]
alldiff([X1,X2,Y1,Y2]), X1 < X2, Y1 < Y2
```

While both models are semantically equivalent, more pruning can be achieved with the set model than its FD counterpart. The key lies in an inference rule, which we define, that detects when an element *must* belong to a specific set (e.g. in the above example, 2 must belong to X). The FD model can not enforce this addition because there exist solutions with 2 assigned either to $X1$ (i.e. $X = \{2,3\}$ or $X = \{2,4\}$) or to $X2$ (i.e. $X = \{1,2\}$). The FD model introduces a disjunction which is not resolved by the ordering constraints (meant to remove symmetries). The strengths of the set model and global filtering we propose, come from the fact that 1) we can address the disjointness together with the set cardinality constraints in a deterministic manner to infer BC, 2) we do so at a computational cost similar to the `alldiff` filtering method [15].

Our contribution lies in the definition of a global satisfiability condition and two inference rules, and the proof that they are necessary and sufficient to ensure satisfiability and BC for the `disjoint(`$[X_1,\ldots,X_n],[c_1,\ldots,c_n]$`)` constraint over sets of fixed cardinalities. While implementing the inference rules as such would lead to an exponential time complexity, we show how the proof procedure can be turned into an efficient global filtering algorithm which detects satisfiability and infers BC in polynomial $O(ncv\sqrt{nc})$ time, where v is the size of the union of the set domains upper bounds and c is the largest cardinality. The proof exploits Hall's theorem and the application of packing numbers to partially known sets (specified by set intervals).

We finally show how our algorithm compares with the GCC GAC algorithm[16], when considering a dual FD representation of the disjoint constraint based on the Generalized Cardinality Constraint. This dual model holds because of the injective mapping between the elements and the sets (each element belongs to atmost 1 set).

2 Background

A fair amount of research has been devoted in configuration design to derive counting functions that determine the maximum number of sets, with a given cardinality, allowed in a known superset under some constraints [5]. The configurations are referred to as *t-designs* where instances are solutions to `partition` and `atmost` constraints "any two sets have atmost one element in common" (e.g. Steiner triple systems), or *t-packings* where instances are solutions to `disjoint` and `distinct` constraints over a family of unknown or partially known sets. The counting functions can be used *successfully* to perform some global reasoning if they can be applied to partially known sets. We use a counting function that derives the *packing number* which (we show) can be applied when using set intervals to specify partially known sets.

To simplify the proof and discussion we assume from here on that all sets have the same fixed cardinality c which we denote `disjoint(`$[X_1,\ldots,X_n],c$`)` . This greatly simplifies the reasoning of the proof without affecting the ability of the algorithm to work on the original constraint. Indeed any arbitrary `disjoint(`$[X_1,\ldots,X_n],[c_1,\ldots,c_n]$`)` constraint can be transformed into a `disjoint(`$[X_1,\ldots,X_n],c$`)` constraint. We do so by simply padding the g.l.b.s of smaller set variables with new and distinct elements until all sets have the same cardinality c. The two constraints clearly have exactly the same solutions modulo the added elements.

2.1 Global disjointness and packing number

The `disjoint(`$[X_1,\ldots,X_n],c$`)` constraint requires any two of the set variables to be pairwise disjoint and of cardinality c. A solution to this constraint can be expressed as a t-packing.

Definition 1 *A $t-(v,c,\lambda)$ packing, or t-packing, is a collection of distinct subsets (blocks) of a ground set A, where:*

- *Each block has c elements*
- *A has v elements, and is called the* base set
- *Any set of t elements of A appears in atmost λ blocks*
- *we have $0 < t \leq c < v$, and $\lambda > 0$.*

A family of *disjoint* sets of same cardinality c, subsets of a known set A of size v, forms a $1-(v,c,1)$ packing, since every single element in A should appear in atmost 1 set. This also holds for partially defined sets, where A is defined as the union of the set domains l.u.b.'s. Thus we define all solutions to the `disjoint(`$[X_1,\ldots,X_n],c$`)` constraint to be $1-(v,c,1)$ packings where:

$$\begin{array}{ll} 1)\ A = \bigcup_{i:1\ldots n} lub(X_i) & 2)\ |A| = v \\ 3)\ \forall_{i:1\ldots n} |X_i| = c & 4)\ 1 \leq c < v \end{array}$$

For a given $t-(v,c,\lambda)$ packing, the *packing number* determines the maximum number of sets that can exist in the packing. This number is not known or easily

derivable for *all* t-packings. However, it has been derived for some combinations of parameters and also many bounds are known. For the disjointness constraint, the packing number is known and defines the maximum number of disjoint sets of size c that we can build from a v-element superset. We have:

Definition 2 *The packing number of a* $1-(v,c,1)$ *packing is* $\lfloor \frac{v}{c} \rfloor$.

It is worth noting that when the packing number (for disjoint sets) is exactly $\frac{v}{c}$ (ie when $n = \frac{v}{c}$), we have a partition since all sets are disjoint and each element of the base set will belong to exactly one set. This remark is core to our inference rules.

2.2 Hall's theorem [9]

Let us now recall Hall's theorem since we will make use of it in our proof of satisfiability and BC. Given a collection of subsets $X_1,\ldots,X_n$ of a v-set A, a System of Distinct Representatives (SDR) for $X_1,\ldots,X_n$ is a family $\{e_1,\ldots,e_n\}$ of elements of A satisfying the conditions

$$1)\ \ e_i \in X_i \text{ for } i = 1\ldots n \qquad 2)\ \ e_i \neq e_j \text{ for } i \neq j$$

The first condition asserts that the elements are representatives of the sets, and the second that they are distinct.

Hall's theorem says that a family of finite sets has an SDR **iff** the union of any k of the sets contains at least k distinct elements. Hall proved that this condition is also sufficient for the existence of an SDR.

For example, the sets $\{1,3\},\{2,4\},\{1,4\},\{2,5\}$ possess an SDR since we can select the family $\{3,2,1,5\}$ to represent respectively each of the four sets. However, the sets $\{1,3\},\{2,4\},\{1,4\},\{1,2\},\{2,3\}$ do not have an SDR since the five sets have only 4 distinct elements in their union, which is not enough to represent 5 sets.

Notations To simplify the coming discussion we introduce the following conventions. We use S to denote $\{X_1,\ldots,X_n\}$ (all the variables involved in the constraint). Subsets of the problem variables (when required) will be referred to as U or R. A solution to a constraint is an assignment of values to variables such that the constraint is satisfied. We denote a solution by the mapping $\texttt{sol} : S \mapsto \mathscr{P}(A)$ where $A = \bigcup_{X \in S} lub(X)$.

3 Satisfiability

To achieve BC for the `disjoint(`S,c`)` constraint, there must be at least one solution to the constraint (both theoretically and algorithmically). This section gives necessary and sufficient conditions to determine satisfiability for the `disjoint(`S,c`)` constraint.

The `disjoint(`S,c`)` constraint is set within a finite set constraint system, and thus prior to satisfiability, the set domain constraints and associated set cardinality constraints must be locally consistent. We recall such notions, together with the local satisfiability condition for `disjoint(`S,c`)` , denoted `SAT-0`.

3.1 Local consistency[8]

The local consistency notions for the basic set domain and cardinality constraints are:

$$\begin{array}{ll} a. & X \in [glb(X), lub(X)] \Leftrightarrow glb(X) \subseteq X \subseteq lub(X) \\ b. & |glb(X)| \leq |X| \leq |lub(X)| \end{array}$$

`SAT-0` applies to a family S of pairwise disjoint sets. It ensures that definite elements of a set (in the g.l.b.) are not available to any other set (not in the l.u.b.).

$$\forall X, Y \in S, \left\{ \begin{array}{l} X \subseteq (lub(X) \setminus glb(Y)) \\ Y \subseteq (lub(Y) \setminus glb(X)) \end{array} \right.$$

The constraint `disjoint(`S, c`)` is *locally consistent* **iff** the local consistency notions *a.* and *b.* hold together with `SAT-0`.

Consider the system of constraints:

```
X in [{1},{1,2,3,4,5}], Y in [{2},{1,2,3}],
Z in [{3},{3,4,5}], disjoint({X,Y,Z},2).
```

Applying `SAT-0` prunes 1 and 3 from $lub(Y)$. This leads to a failure since `Y` is forced to contain only the element 2 (the local consistency notion b. is not satisfied).

3.2 SDR and `disjoint(`$S, 1$`)`

It is hopefully clear that from an SDR $\{e_1 \cdots e_n\}$ for the family of sets $lub(X_1) \cdots lub(X_n)$ we can construct a solution to a `disjoint(`$\{X_1, \ldots, X_n\}, 1$`)` constraint provided the g.l.b. for all variables is empty. And vice-versa.

This statement does not generalize to sets of arbitrary cardinality c, nor to set variables with non-empty g.l.b.s. The following global satisfiability condition, `SAT-1`, addresses these points.

3.3 SAT-1

`SAT-1` exploits the packing number and the fact that any subset of a family of disjoint sets is also a family of disjoint sets. Given a family S of pairwise disjoint sets of size c, `SAT-1` states that for *any* collection of sets $U : U \subseteq S$, the packing number associated with the base set A (union of the upper bounds) must be greater than the size of U:

$$\forall U \subseteq S,\ A = \bigcup_{X \in U} lub(X),\ \frac{|A|}{c} \geq |U|$$

Consider the system of constraints:

```
X in [{},{1,2,3,4}], Y in [{},{1,2,4,5}],
Z in [{},{1,3,5}], disjoint({X,Y,Z},2).
```

`SAT-1` holds for all families of sets except one. For the family $\{X, Y, Z\}$ we have: $A = \{1,2,3,4,5\}, |A| = 5$ and $\frac{5}{2} < 3$, so the system is unsatisfiable.

Theorem 1 *The constraint* `disjoint(`S,c`)` *, locally consistent in a finite set constraint system, is satisfiable* ***iff*** `SAT-1` *holds.*

Proof $\Rightarrow$ Clearly if the constraint is satisfiable then we can build n disjoint sets of size c. Thus $\forall U \subseteq S$ we have $|A = \bigcup_{X \in U} lub(X)| \geq |\bigcup_{X \in U} \texttt{sol}(X)|$, and also $|\bigcup_{X \in U} \texttt{sol}(X)| = |U| * c$ since the sets $\texttt{sol}(X)$ are all disjoint of size c. By transitivity we have $|A| \geq |U| * c$, thus `SAT-1` holds.

$\Leftarrow$ Now let us assume that `SAT-1` holds. We will show that it is always possible to construct a solution satisfying the constraint. We do so by reducing the initial constraint to an equivalent disjoint constraint over S' composed of new set variables of cardinality 1 with empty g.l.b.s and showing that Hall's theorem can be applied to these l.u.b.s, to give a solution.

We generate for each set $X_i \in S$, c set variables $Y_{ij} \in [\emptyset, l_{ij}]$ of cardinality 1, such that there is one new set for each element in $glb(X_i)$ and the remaining new sets have the same l.u.b. as X_i. We have:

$$l_{ij} = \begin{cases} \{x_j\},\ j^{th} \text{ element of } glb(X_i) \\ lub(X_i) \text{ otherwise} \end{cases}$$

We will now show that $\forall U' \subseteq S' \left(|A'| \geq |U'|\right)$ where $A' = \bigcup_{Y_{ij} \in U'} lub(Y_{ij})$, and hence by Hall's theorem that a solution exists. To count the elements in A' for an arbitrary subset of variables U', we partition U' into two disjoint sets U'' and R'', and count the elements in each set separately.

We define U'' as the set of all $Y_{ij} \in U'$ created from $glb(X_i)$ for which there is no other $Y_{ij'} \in U'$ created from $lub(X_i)$.

From `SAT-0`, the absence of any corresponding upperbound sets and the fact that the sets are all singletons we get the following equality. $|A'| = |U''| + |\bigcup_{Y_{ij} \in R''} lub(Y_{ij})|$

Combine this with the following equality based on the definition of U'' and R''. $|U''| = |U'| - |R''|$ to get $|A'| = |U'| + \left(|\bigcup_{Y_{ij} \in R''} lub(Y_{ij})| - |R''|\right)$

Now if we can show $|\bigcup_{Y_{ij} \in R''} lub(Y_{ij})| \geq |R''|$ then we can conclude that $|A'| \geq |U'|$.

Define $R' = \{X_i \,|\, Y_{ij} \in R''\}$ to be the projection of the variables R'' back to the original problem variables. By definition of R'', we have that $\bigcup_{Y_{ij} \in R''} lub(Y_{ij}) = \bigcup_{X_i \in R'} lub(X_i)$. Hence `SAT-1` gives us that $|\bigcup_{Y_{ij} \in R''} lub(Y_{ij})| \geq c * |R'|$. Finally, the construction of S' (and consequently R'') ensures that $c * |R'| \geq |R''|$. By transitivity of $\geq$ we have shown $|\bigcup_{Y_{ij} \in R''} lub(Y_{ij})| \geq |R''|$. ■

4 BC rules

We now define two global inference rules allowing for the necessary pruning to infer BC for the `disjoint`(S,c) constraint.

Let us recall the definition of BC for our constraint domain[20]. In general terms, a n-ary FD constraint is BC, if and only if any assignment from a variable's domain can be extended to a solution [13]. For finite set constraint systems this corresponds to the following definition:

Definition 3 *An n-ary set constraint is BC **iff** :*

1. *any element in the l.u.b. of a set variable appears in that set in at least one solution,*
2. *any element which appears in the same set in all solutions must be part of the g.l.b. of that set.*

4.1 IR-2

`IR-2` allows us to prune elements from the l.u.b. The inference rule says that when a family of sets, U subset of S forms a partition of a base set A, all the elements in the base set A become unavailable to all sets Y outside U (since they must be covered by one set in U). We define the partition condition for any $U \subseteq S$ as follows:

$$\text{partition}(U,A,c) \Leftrightarrow A = \bigcup_{X \in U} lub(U), |A| = |U| * c$$

and state `IR-2` to be

$$\forall\, U \subseteq S, \text{partition}(U,A,c) \Rightarrow \forall Y \in S \setminus U,\ X \cap A = \emptyset$$

Consider the system of constraints:

```
[X,Y] in [{},{1,2,3,4}],
Z in [{},{3,4,5,6}], disjoint({X,Y,Z},2).
```

Applying `IR-2` for the different families of sets we see that $U = \{X,Y\}$ forms a partition of the base set $A = \{1,2,3,4\}$ (since $|A| = |U| * 2$). Thus all the elements must be covered by X and Y, implying that they should be removed from Z. This leads to the final system of constraints:

```
[X,Y] in [{},{1,2,3,4}], Z = {5,6},
disjoint({X,Y,Z},2).
```

4.2 IR-3

`IR-3` also makes use of the partition condition together with an additional condition to add elements to g.l.b., that is elements that should be part of all solutions.

The basic idea is that if a subset U of S forms a partition of a base set A, and there are elements e of A that occur in only one l.u.b. then such elements should be added to the corresponding g.l.b., since they must belong to the partition and can not belong to any other set. This is independent of whether such elements belong to l.u.b. outside the partition or not.

$$\forall\, U \subseteq S,\ \text{partition}(U,A,c),$$

$$\exists e \in A, \exists! X \in U | e \in lub(X) \Rightarrow e \in X$$

Consider a slight change to the previous example:

```
X in [{},{1,2,3,4}], Y in [{},{1,2,3}],
Z in [{},{3,4,5,6,7}], disjoint({X,Y,Z},2).
```

The set $U = \{X, Y\}$ still forms a partition of the base set $A = \{1,2,3,4\}$ (since $|A| = |U| * 2$), but 4 occurs only in $lub(X)$. Thus applying `IR-3` leads to adding 4 to X: $X \in [\{4\}, \{1,2,3,4\}]$. The system of constraints at fixed point is:

```
X in [{4},{1,2,3,4}], Y in [{},{1,2,3}],
Z in [{},{5,6,7}], disjoint({X,Y,Z},2).
```

5 Prelude to BC

In the following section we prove that `IR-2` and `IR-3` are necessary and sufficient to ensure BC for the global constraint `disjoint(`S,c`)` . In order to prove BC we make use of a *swap graph* structure. Despite being defined in terms of the constraint and a *single* solution, *all* solutions are contained within and can be generated from it.

Definition 4 *Given a constraint* $C =$ `disjoint(`S,c`)` *and a solution* `sol` *to C. We define* $A = \bigcup_{X \in S} lub(X)$, $B = A \setminus \bigcup_{X \in S} glb(X)$ *and* $D = A \setminus \bigcup_{X \in S}$ `sol`(X). *Let* α *be a new set variable in* $[\emptyset, B]$. *We extend* `sol` *for* α *with* `sol`$(\alpha) = D$. *The* swap graph $SG(C,$ `sol`$)$ *is the directed labeled graph*

- *whose node set is* $S \cup \{\alpha\}$
- *with edges* $X \overset{e}{\mapsto} Y$ *where* $e =$ `sol`$(X) \cap lub(Y)$ *and* $e \neq \emptyset$

It follows that, for a given swap graph $SG(C,$ `sol`$)$ we have:

$glb(X) =$ elements which appear on the self-edge only (not on any other outgoing edge)

$lub(X) =$ union of all labels in incoming edges (which terminate at X)

`sol`$(X) =$ label on the self-edge.

Definition 5 *An edge* $X \overset{e}{\mapsto} Y$ *is valid* ***iff*** *there is a solution with at least one element of e appearing in* `sol`(Y).

Theorem 2 *If an edge* $X \overset{e}{\mapsto} Y$ *is in a cycle, there exists a solution with at least one element from e appearing in* `sol`(Y) *(equiv. the edge is* valid*).*
Proof Follows directly from the fact that a cycle can be seen as a permutation acting on the current solution to produce a new solution. An edge in a cycle allows for the rearrangement of the elements from X to Y where none are lost, added or changed (i.e. permutation). ■

Theorem 3 *An edge is valid* ***iff*** *there is a cycle involving the edge.*
Proof $\Rightarrow$ Suppose the edge is valid (i.e. there is a solution `sol`$'$ such that an element $e_1 \in e$ appears in `sol`$'(Y)$). `sol`$'$ can be described as a permutation of `sol` (since every solution can be described as a permutation of every other solution). Furthermore since any permutation can be expressed as a number of simple cycle permutations (see [5])

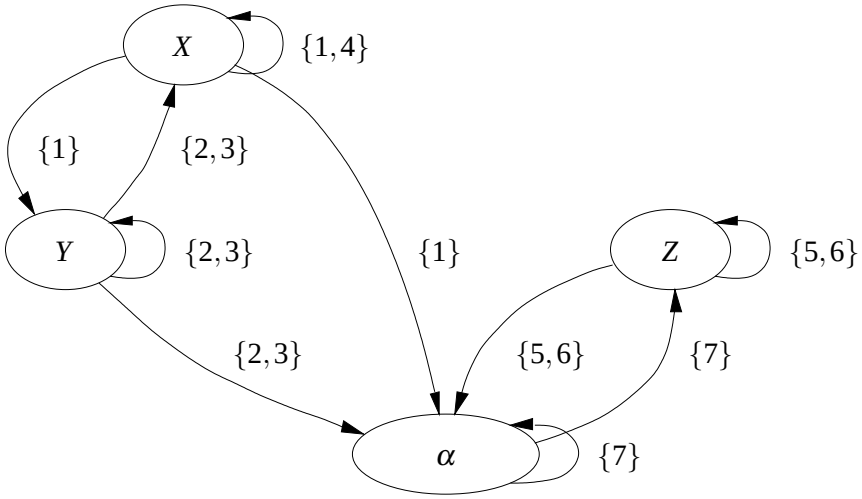

Figure 1: The *swap graph* of the previous example constraint for the solution $\texttt{sol}(X) = \{1,4\}$, $\texttt{sol}(Y) = \{2,3\}$, $\texttt{sol}(Z) = \{5,6\}$.

of the form mentioned above, the permutation mapping `sol` to $\texttt{sol}'$ must contain a cycle with an edge from X to Y labeled with e_1.

$\Leftarrow$ Theorem2 ■

Theorem 4 *A collection of nodes $U \subseteq S$ is unreachable from the rest of the graph **iff** U forms a partition of the base set $A = \bigcup_{X \in U} lub(X)$.*

Proof Split the nodes S into two sets U and U' where U is unreachable from U'. There can be no edges originating in U' and terminating in U, hence all edges which terminate in U must also originate in U. This tells us that A, the union of all upperbounds ($A = \bigcup_{X \in U} lub(X)$) is equal to the union of the solution values appearing in the sets ($A = \bigcup_{X \in U} \texttt{sol}(X)$). By definition $|\bigcup_{X \in U} \texttt{sol}(X)| = |U| * c$, and so we have `partition(`$U,A,c$`)` . Similarly any solution to a problem containing a partition must assign all upperbound elements of the partition U to the sets in U and hence the graph would have no incoming arcs from other nodes. ■

Corollary 1 *The collection of all nodes which are unreachable from α represents a partition. This follows as a special case of theorem 4.*

Theorem 5 *Within a minimal partition (one containing no sub-partitions) there is always a path from any node to any other node.*

Proof Consider a minimal partition P of the nodes S. Now split P into two non-empty, mutually disconnected subsets U and U'. By theorem 4 both U and U' must be partitions, but since P is a minimal partition this cannot happen. Thus we can conclude that there can be no mutually disconnected subsets of P, equivalently there is always a path between any two nodes. ■

Corollary 2 *Minimal partitions are strongly connected components (SCCs) which are unreachable from α. This follows directly from corollary 1 and theorem 5.*

6 Key theorem and proof of BC

Theorem 6 *A satisfiable* `disjoint(`S,c`)` *constraint is BC* ***iff*** `IR-2` *and* `IR-3` *hold.*

The proof is in two parts following our definition of BC for finite set constraint systems.

The first condition of BC says that for all sets, "any element in the l.u.b. of a set variable appears in that set in at least one solution". This is equivalent to saying that all edges in $SG(C,\texttt{sol})$ are valid. We now prove that the elements excluded by `IR-2` correspond exactly to the invalid edges in the swap graph.

Proof part 1

1. `IR-2` seeks all partitions U and removes elements, covered by U, from l.u.b's of sets outside U. This corresponds to all the elements on edges leaving a partition, $X \overset{e}{\mapsto} Y, X \in U, Y \notin U$.

2. By theorem 4, for all sets of nodes U in SG such that U forms a partition, there can be no edges entering U. Thus leaving edges from U can not be part of any cycle and are thus invalid (theorem 3).

3. Edges within a partition are necessarily within a minimal partition and by corollary 2 within an SCC, thus in a cycle. By corollary 3 those edges are valid. Consequently `IR-2` removes *exactly* all the invalid edges.

The second condition of BC says that for all sets, "any element which appears in the same set in all solutions must be part of the g.l.b. of that set".

part 2 We prove that the preconditions of `IR-3` (X belongs to a partition and e occurs uniquely in $lub(X)$ within the partition) hold **iff** the element e appears in the set X in all solutions (and hence must be added to the g.l.b.).

We do this by splitting the graph into the sub-graph U containing nodes reachable from α and the sub-graph of nodes which are unreachable.

If $X \in U$, theorem 4 tells us that X is not in a partition (since U contains no partitions) and theorem 2 tells us that there is a solution where e does not appear in any set in U.

If $X \notin U$ then by theorem 4, X is in a partition and hence e must appear in one of the sets in the partition. Now consider that e occurs uniquely in $lub(X)$ within the partition, clearly e must appear in X in all solutions, hence `IR-3` is necessary. To show that it is complete, consider the case where there is another set Y in the partition s.t. $e \in lub(Y)$. By theorem 5 there is a cycle involving the edge, hence by theorem 2 there is another solution which assigns e to Y, hence `IR-3` is sufficient. ■

7 Algorithm

The algorithm works by first creating and then maintaining a *swap graph* structure. In order to create a swap graph we require an initial solution. To ensure satisfiability (which is equivalent to applying `SAT-1`) we explicitly maintain and update a solution.

To find an initial solution, the problem is decomposed, as per the proof of satisfiability into a search for an SDR cast as a bipartite matching problem. The bipartite graph will have $n * c$ variable nodes, $O(v)$ value nodes and $O(n * c * v)$ edges. We use the maximal matching algorithm of Hopcroft and Karp [10] which runs in $O(|\text{edges}|\sqrt{|\text{variable nodes}|}) \equiv O(ncv\sqrt{nc})$ time.

Once we have a solution we build a swap graph from the bipartite graph (containing the matching) in time $O(n^2c)$ by the algorithm below. The swap graph is stored as an adjacency matrix.

funct *MakeSwapGraph*(*bipartitegraph*, *matching*)
  for $i = 1$ **to** n **do**
    for $l = 1$ **to** c **do**
      val ⟵ value assigned to variable $i + l$
      for $j = 1$ **to** n **do**
        if variable j is reachable from *val*
          then add *val* to the edge from i to j

To achieve BC we compute all SCCs which do not contain α (equiv. minimal partitions) using Tarjan's[19] algorithm which runs in $O(|edges| + |nodes|) \equiv O(n^2 + n)$ time and a breadth first search from α which runs in $O(n^2)$ time. Any edges leaving an SCC are removed (in constant time each) and the l.u.b. of the corresponding variable pruned, as per `IR-2`. Any singleton elements in an SCC are added to the associated g.l.b. as per `IR-3` again in $O(n^2c)$. This procedure we call *SCCPruning*.

begin
  bi ⟵ *MakeBipartiteGraph*(*Sets*)
  matching ⟵ *MaximalMatching*(*bi*, *Sets*)
  sg ⟵ *MakeSwapGraph*(*bi*, *matching*)
  SCCPruning(*sg*)
end

The overall time complexity of this algorithm is then $O(ncv + ncv\sqrt{nc} + n^2c + n^2c)$. With the simplifying assumption that the number of elements is always larger than the number of sets (ie. $v \geq n$ and hence $O(n^2c) \equiv O(ncv)$), we have a complexity of $O(ncv\sqrt{nc})$.

7.1 Incremental updates

Having achieved a state of BC, we can do better than repeat the above algorithm should the domains of our variables change. We can incrementally update our swap graph structure in $O(n^2v)$ time for single element domain changes (thus ensuring satisfiability) and re-run the *SCCPruning* procedure to re-establish BC. The updates are achieved

by finding cycles like those used in the proof of BC, and modifying the graph according to the permutation of the solution which the cycle represents.

8 Models and complexity comparison

Though the discussion focuses on all sets having the same fixed cardinality c, our algorithm works unaltered when each set has different (but still fixed) cardinality.

As mentioned earlier in the paper, it is possible to model the `disjoint(`S,c`)` constraint using FD variables constrained to be different. Assuming such an `alldiff` constraint is solved using the global filtering algorithm [15] and any intra-set ordering constraints are solved using standard arc or bound consistency.

1) Both algorithms detect failure in the initial problem with the same time complexity since they use the same bipartite graph modelling.

2) While the two algorithms are similar in time complexity, it is impossible with the FD model to guarantee BC, as we illustrated in the introduction, since addition of an element to the g.l.b. of a set variable as part of pruning, can only be approximated in the FD model.

However, we can model the problem in another way, by having a FD variable corresponding to every element of the base set where the value assigned to the variable indicates the set in which this variable occurs. Using such a model, we can constrain the number of times a given value (set identifier) appears in the list of variables (set elements) to be the cardinality of the set. The Global Cardinality Constraint (GCC) of [16] can be used to enforce GAC on the FD model which corresponds to BC on the set model. Consider a disjointness constraint on the family of sets $\{S_1,..,S_n\}$ of fixed cardinality c. Let $A = \bigcup_i S_i$ and $|A| = m$. A dual finite domain representation is such that:

1. We have m FD variables: $\forall j \in A, X_j \in D = \{1,..,n\}$.
2. For each $i \in D$ the number of times i can be assigned to different X_j is $c = |S_i|$.
3. The solution equivalence between the two models is:

$$X_j = i \text{ iff } j \in S_i$$

The network flow model which forms the basis of the GCC constraint works because there is an injective mapping between the elements and sets (as the `alldiff` constraint has an injective mapping between variables and values). In the more general case of $t-(v,k,\lambda)$ packing problems there are no injective mappings, instead there are *sur*jective mappings. In our work, were unable to construct network flow models for these surjective problems and so derived our combinatorial counting arguments. We believe that recent results in combinatorial theory, in particular the generalisation of Hall's theorem to arbitrary hypergraphs[1] may lead to proof schemes for these more general cases which can in turn be converted to effiecient algorithms.

9 Acknowledgements

A shorter version of this paper was submitted to AAAI-04. The authors would like to thank one reviewer in particular for his thorough review and comparison with the GCC algorithm.

10 Conclusion and future work

This paper showed that it is possible to efficiently enforce global consistency on n-ary set constraints. While Hall's theorem forms the theoretical basis for important global FD constraints, we showed that it can be extended to n-ary set constraints using packing numbers. We described a global filtering approach to detect satisfiability and ensure BC for the `disjoint(`$[X_1,\ldots,X_n],[c_1,\ldots,c_n]$`)` constraint. From our global satisfiability condition, inference rules and proof, we gave an efficient polynomial time algorithm to achieve and incrementally maintain BC, which is not be possible using the most obvious FD formulation, though is possible for the dual model using the GCC constraint.

We implemented the constraint in the ECL^{i}PS^e system, building on the `ic_sets` library [18]. Preliminary experimental results indicate that the algorithm performs well compared to local propagation and demonstrates the expected benefits of maintaining a level of consistency which is independent of the search strategy. Future work comprises further experimental studies including a combination of global constraints with symmetry breaking for problems modelled with sets. We are also interested in identifying which level of consistency can be achieved when dealing with bounded cardinalities and a generalization of the inference rules to any $t-(v,c,\lambda)$ packing problem.

References

[1] Ron Aharoni and Penny Haxell. Hall's theorem for hypergraphs. *Journal of Graph Theory*, 35(2):83–85, 2000.

[2] N. Barnier and P. Brisset. Facile : A functional constraint library. In *CICLOPS'01*, 2001.

[3] N. Barnier and P. Brisset. Solving the Kirkman's Schoolgirl Problem in a Few Seconds. In *Proc. CP-02*, 2002.

[4] N. Beldiceanu and E. Contejean. Introducing Global Constraints in CHIP. In Elsevier Science, editor, *Mathematical Computation Modelling*, volume 20(12). 1994.

[5] C. Berge. *Principle of combinatorics*. Academic Press, 1971.

[6] Torsten Fahle, Stefan Schamberger, and Meinholf Sellman. Symmetry breaking. In *Proc. CP-01*, pages 93–107, 2001.

[7] Carmen Gervet. Conjunto: constraint logic programming with finite set domains. In *Proc. ILPS-94*, 1994.

[8] Carmen Gervet. Interval Propagation to Reason about Sets: Definition and Implementation of a Practical Language. *CONSTRAINTS journal*, 1(3):191–244, 1997.

[9] P. Hall. On Representatives of Subsets. *J. of London Math. Soc.*, 10:26–30, 1935.

[10] J E Hopcroft and R M Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 2(4), 1973.

[11] ILOG Inc. *ILOG Solver, User Manual*, 1997.

[12] Francois Laburthe. Choco: implementing a cp kernel. http://www.choco-constraints.net/, September 2000. In TRICS CP-00.

[13] A. Mackworth. On reading sketch maps. In *IJCAI'77*, pages 598–606, 1977.

[14] Tobias Müller. Solving set partitioning problems with constraint programming. In *Proc. PAPPACT'98*, March 1998.

[15] Jean-Charles Regin. A filtering algorithm for constraints of difference in csps. In *Proc. AAAI-94*, 1994.

[16] Jean-Charles Regin. Generalized arc consistency for global cardinality constraint. In *Proc. AAAI-96*, 1996.

[17] Andrew Sadler and Carmen Gervet. Global reasoning on sets. In *FORMUL'01 worshop in conjunction with CP-01*, 2001.

[18] Joachim Schimpf, Andrew M Cheadle, Warwick Harvey, Andrew Sadler, Kish Shen, and Mark Wallace. ECL^{i}PS^e. Technical Report 03-1, IC-Parc, Imperial College London, 2003.

[19] R E Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.

[20] Toby Walsh. Consistency and propagation with multiset constraints: A formal viewpoint. In *Proc. CP-2003*, 2003.