Combination of Among and Cardinality Constraints

Jean-Charles Régin

Computing and Information Science, Cornell University, Ithaca NY 14850 USA.
jcregin@cs.cornell.edu

Abstract. A cardinality constraint imposes that each value of a set V must be taken a certain number of times by a set of variables X, whereas an among constraint imposes that a certain number of variables of a set X must take a value in the set V.
This paper studies several combinations of among constraints and several conjunctions of among constraints and cardinality constraints. Some filtering algorithms are proposed and they are characterized when it is possible. Moreover, a weak form of Singleton arc consistency is considered. At last, it is shown how the global sequencing constraint and the global minimum distance constraint can be easily modeled by some conjunctions of cardinality and among constraints. Some results are also given for the global minimum distance constraint. They show that our study outperforms the existing constraints in ILOG Solver.

1 Introduction

Cardinality and among constraints are common to almost real-life problems. For instance, they are present in car sequencing (only some cars of a sequence can take a given option), radio frequency allocation problems (only of node of a pair of adjacent nodes can take a frequency in a set), rostering problems... The resolution of these applications can be improved if we are able to better combine these constraints.
This paper proposes to study in detail the among constraint and the combination of among constraints. We prove that the general problem of the combination of among constraints is NP-Complete. Thus, we propose to study some specific combinations which are tractable and for which we give a filtering algorithm establishing arc consistency.
Then, we consider the conjunction of cardinality constraints and among constraints and give arc consistency algorithms for two types of conjunction that are useful in practice. We also propose an original algorithm which can be viewed as a weak form of Singleton arc consistency.
At last, we will show how to model the global sequencing constraint and the global minimum distance constraint by some conjunctions of cardinality and among constraints. We also give some results for the global minimum distance constraint that outperform the existing constraint in ILOG Solver.

2 Preliminaries

2.1 Graph Theory

These definitions are based on books of [1], [2], and [3].
A **directed graph** or **digraph** $G = (X, U)$ consists of a **node set** X and an **arc set** U, where every arc (u, v) is an ordered pair of distinct nodes. We will denote by $X(G)$ the node set of G and by $U(G)$ the arc set of G. A **path** from node v_1 to node v_k in G is a list of nodes $[v_1, ..., v_k]$ such that (v_i, v_{i+1}) is an arc for $i \in [1..k-1]$. An undirected graph is **connected** if there is a path between every pair of nodes. The maximal connected subgraphs of G are its **connected components**. A directed graph is **strongly connected** if there is a path between every pair of nodes. The maximal strongly connected subgraphs of G are its strongly connected components.
Let G be a graph for which each arc (i, j) is associated with two integers l_{ij} and u_{ij}, respectively called the **lower bound capacity** and the **upper bound capacity** of the arc. A **flow** in G is a function f satisfying the following two conditions[1] :
• For any arc (i, j), f_{ij} represents the amount of some commodity that can "flow" through the arc. Such a flow is permitted only in the indicated direction of the arc, i.e., from i to j. For convenience, we assume $f_{ij} = 0$ if $(i, j) \notin U(G)$.
• A **conservation law** is observed at each node: $\forall j \in X(G) : \sum_i f_{ij} = \sum_k f_{jk}$.
A **feasible flow** is a flow in G that satisfies the **capacity constraint**, that is, such that $\forall (i, j) \in U(G)\ l_{ij} \leq f_{ij} \leq u_{ij}$.

Definition 1. *The* **residual graph** *for a given flow f, denoted by $R(f)$, is the digraph with the same node set as in G. The arc set of $R(f)$ is defined as follows: $\forall (i, j) \in U(G)$:*
• *$f_{ij} < u_{ij} \Leftrightarrow (i, j) \in U(R(f))$ and upper bound capacity $r_{ij} = u_{ij} - f_{ij}$.*
• *$f_{ij} > l_{ij} \Leftrightarrow (j, i) \in U(R(f))$ and upper bound capacity $r_{ji} = f_{ij} - l_{ij}$.*
All the lower bound capacities are equal to 0.

2.2 Constraint Programming

A finite **constraint network** $\mathcal{N}$ is defined as a set of n **variables** $X = \{x_1, \ldots, x_n\}$, a set of current **domains** $\mathcal{D} = \{D(x_1), \ldots, D(x_n)\}$ where $D(x_i)$ is the finite set of possible **values** for variable x_i, and a set $\mathcal{C}$ of **constraints** between variables. $\mathcal{D}_0 = \{D_0(x_1), \ldots, D_0(x_n)\}$ to represent the set of initial domains of $\mathcal{N}$. Indeed, we consider that any constraint network $\mathcal{N}$ can be associated with an initial domain $\mathcal{D}_0$ (containing $\mathcal{D}$), on which constraint definitions were stated.
A **constraint** C on the ordered set of variables $X(C) = (x_{i_1}, \ldots, x_{i_r})$

[1] Without loss of generality (see p.45 and p.297 in [3]), and to overcome notation difficulties, we will consider that if (i, j) is an arc of G then (j, i) is not an arc of G, and that all boundaries of capacities are nonnegative integers.

is a subset $T(C)$ of the Cartesian product $D_0(x_{i_1}) \times \cdots \times D_0(x_{i_r})$ that specifies the **allowed** combinations of values for the variables $x_1, \ldots, x_r$. An element of $D_0(x_1) \times \cdots \times D_0(x_r)$ is called a **tuple on** $X(C)$. $\tau[x]$ denotes the value of x in the tuple τ.
Let C be a constraint. A tuple τ on $X(C)$ is **valid** if $\forall x \in X(C), \tau[x] \in D(x)$. C is **consistent** iff there exists a tuple τ of $T(C)$ which is valid. A value $a \in D(x)$ is **consistent with** C iff $x \notin X(C)$ or there exists a valid tuple τ of $T(C)$ with $a = \tau[x]$. A constraint is **arc consistent** iff $\forall x_i \in X(C), D(x_i) \neq \varnothing$ and $\forall a \in D(x_i)$, a is consistent with C.
An instantiation of all variables that satisfies all the constraints is called a solution of a CN. Constraint Programming (CP) proposes to search for a solution by associating with each constraint a filtering algorithm that removes some values of variables that cannot belong to any solution. These filtering algorithms are repeatedly called until no new deduction can be made. Then, CP uses a search procedure (like a backtracking algorithm) where filtering algorithms are systematically applied when the domain of a variable is modified.
We will use the following notations:

- $\overline{x}$ (resp. $\underline{x}$) denotes the maximum (resp. minimum) value of $D(x)$.
- $D(X)$ denotes the union of domains of variables of X (i.e. $D(X) = \cup_{x_i \in X} D(x_i)$).
- $\#(a, \tau)$ is the number of occurrences of the value a in the tuple τ.
- $\#(a, X)$ is the number of variables of X such that $a \in D(x)$.

2.3 Element Constraint

The element constraint has been introduced in [4]. It defines a functional link between two variables. We propose a definition which is convenient for our purpose.

Definition 2. *Let f be a function*[2] *from a set S_1 to a set S_2. An* **element constraint** *C is a binary constraint defined on two variables x and y and associated with f and such that*
$T(C) = \{\tau$ *s.t.* τ *is a tuple on* $\{x, y\}$ *and* $\tau[y] = f(\tau[x])\}$
It is denoted by element(y, f, x).
We will say that a variable y is **created by an element constraint** *element(y, f, x) if y is defined with a domain equal to* $\cup_{a \in D(x)} f(a)$[3] *and if the element constraint is added to the problem.*

Note that it is easy to maintain arc consistency for an element constraint because it is a functional constraint. This operation can be done in $O(d)$, where d is the size of the largest domain of x and y [5].

[2] In mathematics, a function is a relation, such that each element of a set is associated with a unique element of another set (possibly the same).
[3] This means that the domain of x is not altered by the propagation after the definition of y and the addition of the element constraint.

2.4 Cardinality Constraints

The Global Cardinality Constraints (GCC) has been proposed by [6]. It constraints the number of times every value is taken to be in an interval. In this initial definition, the intervals are statically given by their lower and upper bounds. Then, it has been proposed by [7] and [8] to deal with variables instead of interval. This version is more convenient for our purpose:

Definition 3. *A* **global cardinality constraint involving cardinality variables** *defined on a set of variables X and a set of variables K and associated with a set of values V is a constraint C in which each value $a \in V$ is associated with a cardinality variable $K[a]$ and*
$T(C) = \{\tau$ s.t. τ is a tuple on $X(C)$ and $\forall a \in V : K[a] = \#(a, \tau)\}$
It is denoted by $gcc(X, V, K)$.

This constraint has been called cardVar-GCC, but we think that there is no reason to differentiate it from a GCC because there is no ambiguity to differentiate the parameters, this is why we will use the same name. A GCC C is consistent iff there is a flow in an directed graph $N(C)$ called the value network of C [6]:

Definition 4. *Given $C = gcc(X, V, K)$ a GCC; the* **value network** *of C is the directed bipartite graph $N(C)$ in which each arc is associated with a lower and an upper bound. The node set of $N(C)$ is defined by:*
• *the set of variables X called the variable set of $N(C)$;*
• *the set $D(X) \cup V$ called the value set of $N(C)$;*
• *a node s called the source and a node t called the sink.*
The arc set of $N(C)$ is defined as follows:
• *there is an arc from a variable x to a value a of $(D(X) \cup V)$ if and only if $a \in D(x)$. For every arc (x, a) we have $l_{xa} = 0$ and $u_{xa} = 1$;*
• *there is an arc from s to every variable $x \in X$. For every arc (s, x) we have $l_{sx} = u_{sx} = 1$.*
• *there is an arc from each value $a \in D(X) \cup V$ to the sink t. If $a \in V$ then $l_{at} = \underline{K}[a]$ and $u_{at} = \overline{K}[a]$ else $l_{at} = 0$ and $u_{at} = |X|$.*
• *there is an arc from t to s with $l_{ts} = u_{ts} = |X|$.*

Proposition 1. *[6] Let C be a GCC. Then,*
• *C is consistent if and only if there is a feasible flow in $N(C)$.*
• *Let f be a feasible flow in $N(C)$. A value a of a variable $x \in X$ is not consistent with C if and only if $f_{xa} = 0$ and a and x do not belong to the same strongly connected component in $R(f)$.*

The strongly connected component can be identified in $O(n + m)$ for a graph having n nodes and m arcs [9], thus arc consistency for the variables of X and for a GCC can be established with the same complexity. With the previous definition of $N(C)$ we have $n = |X| + |D(X) \cup V|$ and $m = (\sum_{x \in X} |D(x)|) + n + 1$. Note that we can merge all values of $D(X)$ that does not belong to V into a single value representing the fact that a variable can be assigned to a value which is not in V. This information can be easily maintained and in this case we have $m = (\sum_{x \in X} |D(x) \cap V| + 1) + n + 1$ which less than the previous value. Arc consistency for the variables of K can be much more difficult to compute as shown by [8]:

Proposition 2. *[8] If the domain of each variable of K is a range of integers then arc consistency for the variables of K can be established in $O(nm + n^{2.66})$, else the problem is NP-Complete.*

However, a simple filtering algorithm based on constraints addition can be associated with the variables of K [7]:

Proposition 3. *Let $C = gcc(X, V, K)$ be a GCC, and f be any feasible flow in $N(C)$. Then, we have:*
- $\forall a_i \in V\ K[i] \leq \#(a_i, X)$
- $\sum_{a_i \in V} K[i] \leq |X|$ *and if* $D(X) \subseteq V$ *then* $\sum_{a_i \in V} K[i] = |X|$
- *for every connected component CC of $GV(X)$ we have:*

if $vals(CC) \subseteq V$ *then* $\sum_{a_i \in vals(CC))} K[i] = |vars(CC)|$,
else $\sum_{a_i \in vals(CC))} K[i] \leq |vars(CC)|$,
where $vals(CC)$ denotes the values of V belonging to CC and $vars$ denotes the variables of X belonging to CC and $GV(X)$ is the value graph of X that is $GV(X) = (X, \cup_{x_i \in X} D(x_i), E)$ where $(x, a) \in E$ iff $a \in D(x)$.

Bound consistency of a sum constraint involving p variables can be established in $O(p)$. The strongly connected components of the residual graph are computed to establish arc consistency of the variables of X, thus bound consistency of the constraints of Proposition 3 can be implemented in $O(|K|)$.

3 The Among Constraint

Definition 5. *An* **among constraint** *defined on a set of variables X and a cardinality variable k and associated with a set of value V is a constraint C such that*
$T(C) = \{\tau$ *s.t.* τ *is a tuple on* $X(C)$ *and* $k = \sum_{a \in V} \#(a, \tau)\}$
It is denoted by $among(X, V, k)$.

It is straightforward to design a filtering algorithm establishing arc consistency for this constraint. For instance, we can associate with each variable x of X a (0,1) variable x_V defined as follows: $x_V = 1$ if and only if $x = a$ with $a \in V$. Then the constraint can be rewritten $\sum x_V = k$.

Definition 6. *Let $C_1 = among(X_1, V_1, k_1)$ and $C_2 = among(X_2, V_2, k_2)$ be two among constraints. If $X_1 \cap X_2 = \varnothing$ we will say that C_1 and C_2 are* **variable disjoint**. *If $V_1 \cap V_2 = \varnothing$ we will say that C_1 and C_2 are* **value disjoint**.

We propose to study whether it is possible to design some efficient filtering algorithms associated with a conjunction of among constraints. Three possible relations between among constraints:
1. the among constraints are variable disjoint.
2. the among constraints are value disjoint.
3. none of the previous property is satisfied.

Variable disjoint among constraints are totally independent and therefore it is trivial to study their conjunction.

3.1 Value disjoint among constraints

Consider $\mathcal{A} = \{A_1, A_2, ..., A_n\}$ a set of n among constraints that are pairwise value disjoint where every A_i is equal to $among(X_i, V_i, k_i)$. This set of constraints can be efficiently combined by transforming the conjunction into another conjunction for which arc consistency can be efficiently established. This transformation requires to define new variables from the initial variables involved in the among constraints.
First, since all the V_i sets are disjoint, we can define the following function ndx:

Definition 7. *For any value a if there exists v_i such that $a \in V_i$, then $ndx(a) = i$, else $ndx(a) = -1$.*

Then, we associate every variable x_i involved in an among constraint with a more complex function denoted by f_i^{Ind}:

Definition 8. *Let Ind be the triplet $(\mathcal{A}, U, \alpha)$ where $\mathcal{A}$ is a set of value disjoint among constraints, each of them defined on a subset of X and associated with a subset of V; $U = \{u_1, u_2, ..., u_n\}$ is a set of pairwise distinct values with $U \cap (V \cup D(X)) = \varnothing$, and α is a value s.t. $\alpha \notin (U \cup V \cup D(X))$. For each variable $x_i \in X$ we define function f_i^{Ind} as follows: $\forall a \in D(x)$ with $k = ndx(a)$ if $k \neq -1$ and $x \in X_k$ then $f_i^{Ind}(a) = u_k$ else $f_i^{Ind}(a) = \alpha$.*

Now, for each variable $x_i \in X$ a variable y_i is created by the element constraint $element(y_i, f_i^{Ind}, x_i)$. Let Y be the set of these newly created variables.

Example Consider seven variables $x_1, x_2, ..., x_7$, each having a domain equal to [0..7] and 3 among constraints A_1=among($\{x_1, x_2, x_3, x_4\}$,{0,1},k_1), A_2=among($\{x_2, x_4, x_5, x_6\}$,{2,3},k_2), A_3=among($\{x_3, x_4, x_6, x_7\}$,{4,5},k_3). We have $\mathcal{A} = \{A_1, A_2, A_3\}$ and we define $Ind = (\mathcal{A}, \{u_1, u_2, u_3\}, \alpha)$. Then, we obtain $D(y_1) = \{u_1, \alpha\}$, $D(y_2) = \{u_1, u_2, \alpha\}$, $D(y_3) = \{u_1, u_3, \alpha\}$, $D(y_4) = \{u_1, u_2, u_3, \alpha\}$, $D(y_5) = \{u_2, \alpha\}$, $D(y_6) = \{u_2, u_3, \alpha\}$, $D(y_7) = \{u_3, \alpha\}$. For instance, we have $D(y_2) = \{u_1, u_2, \alpha\}$ because x_2 belongs to $X(A_1)$ and $X(A_2)$ and so the values of $\{0, 1\}$ of x_2 correspond to the value u_1 of y_2 and the values of $\{2, 3\}$ of x_2 correspond to the value u_2 of y_2 and the other values of x_2 are associated with the value α of y_2.
All the variables created by element constraints take their values from the set $\{u_1, u_2, ..., u_n\} \cup \{\alpha\}$, then by constraining the number of times these values are taken, we constrain at the same time the number of times any value of a set V_i is taken, and due to the definition of the variables created by element constraints we count only the variables of X_i that take a value in V_i.
The following proposition formally shows the link between a conjunction of among constraints and only one GCC:

Proposition 4. *The establishment of the arc consistency for the conjunction of value disjoints among constraints constraints $\{A_1, A_2, ..., A_n\}$ is equivalent to establishing arc consistency for the constraint network containing the element constraints $\{element(y_i, f_i^{Ind}, x_i), x_i \in X\}$ and the GCC: $gcc(Y, U, \{k_1, k_2, ..., k_n\})$.*

Proof. We can establish arc consistency for the conjunction of the constraints $\{element(y_i, f_i^{Ind}, x_i), x_i \in X\}$ and $gcc(Y, U, \{k_1, k_2, ..., k_n\})$ by establishing arc consistency of the constraint network (CN) consisting of these constraints, because the constraint graph associated with this constraint network is an hyper-graph without any cycle and whose every pair of edges have at most one node (i.e. variable) in common. Thus, arc consistency for this CN is equivalent to arc consistency for the constraint equals to the conjunction of all the constraints in the network. Moreover, from any solution of the CN defined by $\{A_1, A_2, ..., A_n\}$ we can build a solution of the CN define by the element constraints and the GCC, and conversely. Therefore the proposition holds. □

3.2 General conjunction

Proposition 5. *Finding a tuple on the variables of X involved in among constraints is an NP-Complete problem in general.*

Proof. This problem is obviously in NP (easy polynomial certificate). We transform the NP-Complete problem Tripartite Matching (see [10]) to this problem. Tripartite Matching is:
Instance: Three sets B, G and H each containing n elements and a ternary relation $T \subseteq B \times G \times H$. *Question*: find a set of n triples in T, no two of which have a component in common.
We define a set X of n variables, each having a domain equal to $[1..|T|]$. For every pair $\{t_i, t_j\}$ of elements of T having a component in common we define the among constraint: $among(X, \{i, j\}, \{0, 1\})$. This constraint ensures that at most one of the element of $\{t_i, t_j\}$ can be assigned to a variable of X. This model exactly solves Tripartite Matching. □

However, it is possible to define some links between the cardinality variables of two among constraints.

Proposition 6. *Let $A_1 = among(X_1, V_1, k_1)$ and $A_2 = among(X_2, V_2, k_2)$ be two among constraints such that $X_1 \cap X_2 \neq \varnothing$ and $V_1 \cap V_2 \neq \varnothing$. Then, we have:*
$k_1 = k_{(X_1 \cap X_2) \rightarrow (V_1 \cap V_2)} + k_{(X_1 \cap X_2) \rightarrow (V_1 - V_2)} + k_{(X_1 - X_2) \rightarrow V_1}$
$k_2 = k_{(X_1 \cap X_2) \rightarrow (V_1 \cap V_2)} + k_{(X_1 \cap X_2) \rightarrow (V_2 - V_1)} + k_{(X_2 - X_1) \rightarrow V_2}$
where: $k_{Y \rightarrow W}$ is the number of times the values of W are taken by the variables of Y.

The proof of this proposition is straightforward. The sum constraints introduced by this proposition can be easily added to the constraint network and then the filtering algorithms associated them reduce the domain of the cardinality variables. This idea is more general and easier to understand than the algorithm proposed by [11] to combine sequences.

4 Integration of some Among Constraints into Cardinality Constraints

We have seen that under some conditions it is possible to establish arc consistency for some conjunctions of among constraints. In this section

we show that the same kind of result can be obtained by adding a GCC to some conjunctions of among constraints.

Definition 9. *Let X be a set of variables and V be a set of values.*
- *an X-among constraint is an among constraint defined on the set X of variables and on another variable q, that is of the form $among(X, W, q)$.*
- *a V-among constraint is an among constraint associated with the set of value V.*

4.1 Cardinality Constraint and value disjoint X-among constraints

We propose a filtering algorithm establishing arc consistency for the variables of X for a conjunction of a GCC and a set of value disjoint among constraints defined on the same set of variables X.
The efficient algorithm of the GCC is based on the flow theory and uses a specific network. The X-among constraints only introduce new constraints on the cardinality variables of the GCC. Since the among constraints are pairwise value disjoint there is no problem to take into account these new constraints: a slight modification of the value network is sufficient.

Definition 10. *Given $G = gcc(X, V, K)$ and $\mathcal{A} = \{A_1, A_2, ..., A_n\}$ a set of value disjoint among constraint such that $A_i = among(X, V_i, k_i)$; the* **value network** *of $C = (G \cup \mathcal{A})$ is the directed bipartite graph $N(C)$ obtained from $N(G)$ (the bipartite network associated with G), as follows: For each set of values V_i of an among constraint:*
- *a new node w_i is defined*
- *for each value a in V_i, the arc (a, t) is replaced by the arc (a, w_i) which has the same lower and upper bounds as (a, t)*
- *an arc (w_i, t) with $l_{w_i t} = \underline{k}_i$ and $u_{w_i t} = \overline{k}_i$ is added.*

Then we immediately have a proposition similar to Prop.1:

Proposition 7. *Given $G = gcc(X, V, K)$, $\mathcal{A} = \{A_1, A_2, ..., A_n\}$ a set of value disjoint X-among constraints, $C = G \cup \mathcal{A}$ the conjunction of G and $\mathcal{A}$, and $N(C)$ be the bipartite value network associated with this conjunction. Then,*
- *C is consistent if and only if there is a feasible flow in $N(C)$.*
- *let f be a feasible flow in $N(C)$. A value a of a variable $x \in X$ is not consistent with C if and only if $f_{xa} = 0$ and a and x do not belong to the same strongly connected component in $R(f)$.*

The previous proposition is dedicated to the variables of X. We can obtain a filtering algorithm for the cardinality variables by adding for each among constraint $A_i = among(X, V_i, k_i)$ the constraint $\sum_{a \in V_i} K[a] = k_i$.
Note that it is possible to take into account some among constraints defined on a superset Y of X. In this case, we can transform the problem into an equivalent one for which all constraints are defined on the very same set. This transformation uses function f_i^{Ind} (See Def.8.) For every variable y_i of $Y - X$, a variable z_i is created by the element constraint $element(z_i, f_i^{Ind}, y_i)$. Let Z be the set of newly created variables. Then, the initial GCC is replaced by the $gcc(X \cup Z, V, K)$ and each among constraint A_i is replaced by the constraint $among(X \cup Z, (V_i \cap V) \cup \{u_i\}, k_i)$.

4.2 Cardinality Constraint and variable disjoint V-among constraints

We propose a filtering algorithm establishing arc consistency for the variables of X for a conjunction of a GCC and a set of variable disjoint among constraints associated with the same set of values V. We will assume that the GCC is also associated with V and that the V-among constraints are defined on subset of variables of X.
This conjunction of constraints can be efficiently taken into account by transforming it into another conjunction for which arc consistency can be efficiently established. This transformation has been proposed by [11] and requires the definition of new variables called abstract variables. In this section, we give a more more simple version of this transformation.

Definition 11. *Let Red be the pair $(\mathcal{A}, U)$ where $\mathcal{A}$ is a set of n variable disjoint among constraints, each of them defined on a subset of X and associated with V; $U = \{u_1, u_2, ..., u_n\}$ is a set of pairwise distinct values with $U \cap (V \cup D(X)) = \varnothing$. For each variable $x_i \in X$ we define function f_i^{Red} as follows: $\forall a \in D(x_i)$ if $a \in V$ then $f_i^{Red}(a) = a$ else $f_i^{Red}(a) = u_k$, where k is the index of the among constraint involving x_i.*

Now, for each variable $x_i \in X$ a variable y_i is created by the element constraint $element(y_i, f_i^{Red}, x_i)$. Let Y be the set of these newly created variables. The following proposition is a reformulation of the proposition given in [11]:

Proposition 8. *Given $G = gcc(X, V, K)$ and $\mathcal{A} = \{A_1, A_2, ..., A_n\}$ a set of n V-among constraints that are pairwise variable disjoint. We denote by $\overline{X}$ the set of variables of X that do not belong to any X_i. For each among constraint $A_i = among(X_i, V, k_i)$, we define the variable $K_A[u_i] = |X_i| - k_i$. Then, we have:*
The establishment of the arc consistency for the conjunction of variable disjoint V-among constraints $\{A_1, A_2, ..., A_n\}$ and a $gcc(X, V, K)$ is equivalent to establishing arc consistency for the constraint network containing the element constraints $\{element(y_i, f_i^{Red}, x_i),\ x_i \in X\}$ and the GCC: $gcc(\overline{X} \cup Y, V \cup U, K \cup K_A)$.

It is possible to take into account some among constraints defined on a superset Y of X by applying the transformation proposed in the previous section.

5 Stronger Filtering Algorithm

In this section we propose an efficient algorithm to study some of the consequences of the instantiation of a variable for a GCC combined with some among constraints. In general, the conjunction of some among constraint can be an NP-Complete problem. However, we have shown in the previous sections that under some conditions, we can efficiently establish arc consistency for the conjunction of a GCC and some among constraints.

Therefore, in practice, a set of GCCs and a set of among constraints will be modeled by a set of such conjunctions of constraints in addition to some among constraints and some GCCs. The conjunction of all constraints is managed by the propagation mechanism. It is sometimes worthwhile to try to deduce more information by using techniques like Singleton Arc Consistency, shaving or probing. The common idea of these methods is to instantiate some variables and to trigger the propagation mechanism after such an instantiation while expecting that a failure will occur. In this case, indeed, we know that the instantiation does not lead to any solution and so we can remove the value that was assigned from the domain of the selected variable. Unfortunately, these methods have a cost which is often too high in practice and prevent us from using them, at least if we consider all the possible instantiations. In fact, if we have n variables and d values in the domains of the variables, then nd instantiations will have to be considered and possibly several times because after a modification the constraint network has changed. Thus, some methods propose to consider only a subset of the variables and/or a subset of values and/or a subset of constraints.

In this section we will consider a problem containing $C = gcc(X, V, K)$ a GCC and some other constraints mainly dealing with the cardinality variables of the GCC. Our goal is to perform a stronger level of consistency, that is to prune more the domains of the variables of X, but we would like to avoid to be too much systematic. We aim to study the consequences for the cardinality variables of all possible instantiations of the variables of X. We focus our attention on the cardinality variables because these are these variables that can modify the most the problem.

A possible algorithm consists of successively trying all the possible instantiations of the variables of X and then to use the most powerful filtering algorithm associated with a GCC involving cardinality variables. However, this method requires to call nd times an algorithm in $O(nm + n^{2.66})$ (see Prop. 2) which certainly prevent us from using it.

Another possibility is to use a weaker filtering algorithm for the cardinality variables. For instance, the algorithm based on Proposition 3. The advantage of this algorithm is that it has the same complexity as the establishment of the arc consistency for the GCC (i.e. $O(m)$). Thus we will have an new algorithm in $O(ndm)$ if we try each possible instantiation once.

In this section we present another algorithm with an $O(dm)$ time complexity, which is much more acceptable in practice.

The filtering algorithm of a GCC is based on the concepts of Hall variable set and Hall value set:

Definition 12. *Let $gcc(X, V, K)$ be a GCC.*

- *An **Hall variable set** is a set of variables $A \subseteq X$ such that* $|A| = \sum_{a \in D(A)} \overline{K}[a]$
- *An **Hall value set** is a set of values $V \subseteq D(X)$ such that* $\sum_{v \in V} \underline{K}[v] = |vars(V)|$*, where $vars(V)$ is the set of variables having a value of V in their domain.*

Proposition 9. *Let $C = gcc(X, V, K)$ be a GCC.*
• *if A is an Hall variable set then every value (x, a) with $x \in (X - A)$ and $a \in D(A)$ is not consistent with C.*
• *if V is an Hall value set then every value (x, a) with $x \in vars(V)$ and $a \notin V$ is not consistent with C.*
• *if a value (x, a) is not consistent with C then one of the previous property can prove that (x, a) is not consistent with C.*

The last property of the previous proposition proved that the application of the two first properties is sufficient to remove all the values that are not consistent with C. We have the straightforward proposition:

Proposition 10. *Let $C = gcc(X, V, K)$ be a GCC, f be a feasible flow in $N(C)$, A be an Hall variable set and V be an Hall value set.*
(1.a) $\forall a \in D(A)\ K[a] = \overline{K}[a]$.
(1.b) the variables of A belong to strongly connected components of $R(f)$ that do not contain t.
(2.a) $\forall v \in V\ K[v] = \underline{K}[v]$.
(2.b) the variables of $vars(V)$ belong to strongly connected components of $R(f)$ that do not contain t.

We propose to add to the problem some new among constraints, which in some cases lead to direct filtering algorithm. For this purpose we introduce the concept of pseudo Hall set:

Definition 13. • *A **pseudo Hall variable set** is a set of variables $A \subseteq X$ such that $|A| = (\sum_{a \in D(A)} \overline{K}[a]) - 1$*
• *A **pseudo Hall value set** is a set of values $V \subseteq D(X)$ such that $\sum_{v \in V} \underline{K}[v] = |vars(V)| - 1$*

Then, suppose that A is a pseudo Hall variable set. If a variable of $X - A$ is instantiated with a value of $D(A)$ then the set A will become an Hall variable set and by Prop.10 the cardinality variables associated with the values of $D(A)$ can be set to their maximum value. So, the instantiation of one variable may imply the instantiation of some other variables that can have a huge impact on the problem. Similarly, suppose that V is a pseudo Hall value set. If a variable of $vars(V)$ is instantiated to a value which is not in V then V will become an Hall value set and by Prop.10 the cardinality variables associated with the values of V can be set to their minimum value. Once again, the instantiation of one variable may imply the instantiation of some other variables.
Pseudo Hall sets can be identified by removing values from the domains of some variables. Once a pseudo Hall set is identified we know that by instantiating some variables we can also instantiated some cardinality variables. Then, we propose to instantiate these cardinality variables and to trigger the propagation. If a failure occurs then we know that the creation of this Hall set is not possible and we introduce a constraint preventing its creation. More precisely, if we identify a pseudo Hall variable set A whose a variable contains a value b in its domain and if the problem has no solution when b is taken by $y \notin A$ then we can introduce the

constraint ensuring that at least one variable of A will take the value b. Similarly, if we identify V a pseudo Hall value set with a set of variables $Y \in vars(V)$ and if the problem has no solution when the variables of Y are instantiated to a value $b \notin V$ then we can introduce the constraint ensuring that at least one variable of Y must take a value of V.

In order to identify some pseudo Hall sets we propose to remove in turn each value. When a value is removed the variables instantiated to it are also removed. Therefore the current flow of the GCC is still a feasible flow in the new residual graph and we can establish arc consistency for the GCC in $O(m)$. This procedure will compute new strongly connected component in $R(f)$ and from Prop.10 we can identify some Hall sets from them. Then, we need to identify among these Hall sets which ones are pseudo Hall sets in the original GCC. This step is necessary because a newly created Hall sets can be independent of any pseudo Hall set. Note that it is useless to consider values belonging to the domain of variables of an Hall set.

Algorithm 1:

```
StrongerFilteringAlgorithm(C, f)
    Let Scc(t) be the strongly connected component containing t in R(f)
    for each value a ∈ Scc(t) do
        Q(f) ← R(f)
        remove from Q(f) the value a and the set of variables Y = {y s.t. f_ya = 1}
        compute the strongly connected components of Q(f)
        for each strongly connected component S with t ∉ S do
            if vars(S) is a pseudo Hall variable set of C then
                instantiate the cardinality variable of S to their maximal value
                trigger the propagation
                if a failure occurs then
                    add the constraint among(vars(S), {a}, [1..|vars(S)|])
            if vals(S) is a pseudo Hall value set of C then
                instantiate the cardinality variable of S to their minimal value
                trigger the propagation
                if a failure occurs then
                    add the constraint among(Y, vals(S), [1..|Y|])
StrongerConsistencyWithAlldiff(C, f)
    Let Scc(t) be the strongly connected component containing t in R(f)
    for each value a ∈ Scc(t) do
        Q(f) ← R(f)
        remove from Q(f) the value a and the variable y with f_ya = 1
        compute the strongly connected components of Q(f)
        for each strongly connected component S with t ∉ S do
            if vars(S) is a pseudo Hall variable set of C then
                instantiate the cardinality variable of S to 1
                trigger the propagation
                if a failure occurs then
                    remove b from the domain of the variables of X − vars(S)
```

Algorithm 1 contains an implementation of the procedure we have described and a specific procedure when the GCC is an alldiff constraint. In fact, this algorithm is complex in general, but it can be simplified when the GCC is an alldiff constraint (like for the allMinDistance constraint), because there is no Hall value set when all the lower bound capacities are equal to 0. In addition, instead of adding an among constraint we can directly remove the value a from the domain of the variables that are not in the pseudo Hall variable set.

An example of this algorithm is presented in next section.

6 Application to the Global Minimum Distance Constraint

This constraint has been proposed by [12] and is mentioned in [13–15]. A global minimum distance constraint defined on X, a set of variables, states that for any pair of variable x and y of X the constraint $|x-y| \geq k$ must be satisfied.

Definition 14. *A* **global minimum distance constraint** *is a constraint C associated with an integer k such that*
$T(C) = \{\tau \text{ s.t. } \tau \text{ is a tuple of } X(C) \text{ and } \forall a_i, aj \in \tau : |a_i - a_j| \geq k\}$

This constraint is present in frequency allocation problems.
A filtering algorithm has been proposed for this constraint [12]. Note that there is a strong relation between this constraint and the sequence constraint. A $1/q$ sequence constraint constrained two variables assigned to the same value to be separated by at least $q-1$ variables, in regard to the variable ordering. Here we want to select the values taken by a set of variables such that are all pairs of values are at least k units apart.
This constraint is simply a conjunction of X-among constraints. For each value $a \in D(X)$ we define the among constraint $among(X, [a..a+k], [0,1])$. Then the global minimum distance constraint is equivalent to the conjunction of these X-among constraints. If we define a GCC C stating that each value of $D(X)$ has to be taken at most once by a variable of X (in other words an alldiff constraint) then we can model the global minimum constraint by using any several conjunctions of C and a set of value disjoint X-among constraints. Of course the model also uses the general conjunction of among constraints that we have proposed. The model will be equivalent to the global minimum distance constraint provided that each X-among constraint belongs to at least one conjunction. Note that it is possible to use in the conjunction less constrained among constraints for instance the constraint $among(X, [a..a+k], [0,1])$ can be replaced by any among constraint $among(X, V, [0,1])$ where $V \subseteq [a..a+k]$. This can be useful to ensure that every value of $D(X)$ is covered by an among constraint in the conjunction with C.
This model is powerful. For instance consider the following problem involving 3 variables x, y and z with $D(x) = D(y) = \{1,2,3\}$, $D(z) = \{0,1,2,3,4,5\}$ and a minimal distance equals to 2. The constructive disjunction will obtain the new domains: $D(x) = D(y) = \{1,3\}$ and $D(z) = \{0,1,3,4,5\}$ whereas the conjunction we propose of an alldiff constraint and the among constraints: $among(X, [0,1], [0,1])$, $among(X, [2,3], [0,1])$, and $among(X, [4,5], [0,1])$ will deduce that z cannot be assigned neither to $[0,1]$ nor to $[2,3]$. In addition, the propagation between the among constraints leads to $D(x) = D(y) = \{1,3\}$, and $D(y) = \{5\}$.
Moreover, consider now that the initial domains are $D(x) = \{0,4\}$, $D(y) = \{1,3\}$ and $D(z) = \{2,3,4,5\}$. For this problem only the stronger filtering algorithm that we have presented is able to deduce that the only one solution is $x = 0$, $y = 3$ and $z = 5$. This algorithm outperforms the allMinDistance constraint of ILOG Solver.

We have also tested our algorithm on some problems for instance the Radio Link Frequency Allocation Problem. In order to build some global minimum distance constraints some cliques with a good distance value have been identified. Then, with the filtering algorithm that we have proposed we have seen dramatic improvement for some instances and mainly without specific strategies for selecting the next variable and the next value. For instance, Problem 11 is solved with the new constraint in 1s instead of 150s without it.

7 Application to the Global Sequencing Constraint

These constraints arise in many real-life problems such as car sequencing and rostering problems where a lot of min/max constraints have to be verified for each period of q consecutive time units. In a rostering problem, one has to chose a type of work for each day, satisfying various constraints. Sequencing constraints are useful for expressing regulations such as:

- each sequence of 7 days must contain at least 2 days off.
- A worker cannot work more than 3 night shifts every 8 days.

A global sequencing constraint is a gcc for which for each sequence S_i of q consecutive variables of X, the number of variables of S_i instantiated to any value $v_i \in V \subseteq D(C)$ must be in an interval $[min, max]$.

Definition 15. *[11] A* **global sequencing constraint** *is a constraint C associated with three positive integers min, max, q and a subset of values $V \subseteq D(C)$ in which each value $a_i \in D(C)$ is associated with two positive integers l_i and u_i and*
$T(C) = \{$ *t such that t is a tuple of $X(C)$*
and $\forall a_i \in D(C) : l_i \leq \#(a_i, t) \leq u_i$
and for each sequence S of q consecutive
variables: $min \leq \sum_{v_i \in V} \#(v_i, t, S) \leq max\}$
It is noted $gsc(X(C), V, min, max, q, l, u)$, where $l = \{l_i\}$ and $u = \{u_i\}$.

This constraint is simply a conjunction of V-among constraints.
For each variable $x_i \in X$ we define the among constraint $among(\{x_i, ..., x_{i+q}\}, V, [min, max])$. Then the global sequencing constraint is equivalent to the conjunction of these X-among constraints and the GCC $C = gcc(X, V, K)$, where $K[i] = [l_i, u_i]$. We can model the global sequencing constraint by using several conjunctions of C with a set of variable disjoint V-among constraints. Of course the model also uses the general conjunction of among constraints that we have proposed. The model will be equivalent to the global sequencing constraint provided that each V-among constraint belongs to at least one conjunction. Note that it is possible to use in the conjunction less constrained among constraints for instance the constraint $among(\{x_i, ..., x_{i+q}\}, V, [min, max])$ can be replaced by any among constraint $among(Y, V, [min, max])$ where $Y \subseteq \{x_i, ..., x_{i+q}\}$. This can be useful to ensure that every variable of X is covered by an among constraint in the conjunction with C.

8 Conclusion

In this paper we have studied several combinations of among constraints and several conjunctions of among constraints and cardinality constraints. For each considered combination we have proposed an efficient filtering algorithm establishing arc consistency when it was possible. We have also shown that in general the combination of among constraints is an NP-Complete problem. In addition, we have proposed an original algorithm which can be viewed as a weak form of Singleton arc consistency. At last, we have proposed to model the global sequencing constraint and the global minimum distance constraint by conjunctions of cardinality and among constraints. We have also given some results for the global minimum distance constraint that outperform the existing constraint in ILOG Solver.

References

1. Berge, C.: Graphe et Hypergraphes. Dunod, Paris (1970)
2. Tarjan, R.: Data Structures and Network Algorithms. CBMS-NSF Regional Conference Series in Applied Mathematics (1983)
3. Ahuja, R., Magnanti, T., Orlin, J.: Network Flows. Prentice Hall (1993)
4. Van Hentenryck, P., Carillon, J.P.: Generality Versus Specificity: An Experience with AI and OR Techniques. In: Proceedings of AAAI-88. (1988)
5. Van Hentenryck, P., Deville, Y., Teng, C.: A generic arc-consistency algorithm and its specializations. Artificial Intelligence **57** (1992) 291–321
6. Régin, J.C.: Generalized arc consistency for global cardinality constraint. In: Proceedings AAAI-96, Portland, Oregon (1996) 209–215
7. Régin, J.C., Gomes, C.: The cardinality matrix constraint. In: CP'04, Toronto , Canada (2004) 572–587
8. Quimper, C.G., López-Ortiz, A., van Beek, P., Golynski, A.: Improved algorithms for the global cardinality constraint. In: Proceedings CP'04, Toronto, Canada (2004) 542–556
9. Tarjan, R.: Depth-first search and linear graph algorithms. SIAM Journal of Computing **1** (1972) 146–160
10. Papadimitriou, C.H.: Computational complexity. Addison Wesley (1994)
11. Régin, J.C., Puget, J.F.: A filtering algorithm for global sequencing constraints. In: CP97, proceedings Third International Conference on Principles and Practice of Constraint Programming. (1997) 32–46
12. Régin, J.C.: The global minimum distance constraint. Technical report, ILOG (1997)
13. ILOG: ILOG Solver 4.4 User's manual. ILOG S.A. (1999)
14. Régin, J.C.: Global Constraints and Filtering Algorithms. In: Constraints and Integer Programming combined. M. Milano ed, Kluwer (2003)
15. Régin, J.C.: Modélisation et Contraintes globales en programmation par contraintes. Habilitation à diriger des Recherches, Université de Nice-Sophia Antipolis (2004)