# The Conjunction of Interval AMONG Constraints

Gilles Chabert, Sophie Demassey

# The Conjunction of Interval AMONG Constraints

Gilles Chabert and Sophie Demassey

TASC, Mines-Nantes, INRIA, LINA CNRS
4, rue Alfred Kastler 44300 Nantes, France
(gilles.chabert|sophie.demassey)@mines-nantes.fr

**Abstract.** An AMONG constraint holds if the number of variables that belong to a given value domain is between given bounds. This paper focuses on the case where the variable and value domains are intervals. We investigate the conjunction of AMONG constraints of this type. We prove that checking for satisfiability – and thus, enforcing bound consistency – can be done in polynomial time. The proof is based on a specific decomposition that can be used as such to filter inconsistent bounds from the variable domains. We show that this decomposition is incomparable with the natural conjunction of AMONG constraints, and that both decompositions do not ensure bound consistency. Still, experiments on randomly generated instances reveal the benefits of this new decomposition in practice. This paper also introduces a generalization of this problem to several dimensions and shows that satisfiability is $\mathcal{NP}$-complete in the multi-dimensional case.

## 1 Introduction

The problem addressed in this paper can be formally stated as a Constraint Satisfaction Problem composed of a conjunction of AMONG constraints. An AMONG constraint [1] restricts the number of variables that take their values in a given set, called the *value domain*.

Enforcing bound consistency on a general conjunction of AMONG constraints is $\mathcal{NP}$-hard [12], but some tractable cases have been investigated: when the value domains are *all disjoint* [12], or when the value domains are *all equal*, like in the SEQUENCE constraint [15, 4] and its generalizations [10]. In this paper, we consider an open case where the value domains are *arbitrary intervals*. We also examine this problem in higher dimensions, when variables come as vectors and intervals as boxes. This problem has applications in various contexts, such as logistics or sensor networks.

We start by illustrating the one-dimensional case on an event scheduling problem. The computational complexity is analyzed in Section 2, where the corresponding satisfiability problem is proven to be tractable. As in previous works [12, 4] on conjunctions of AMONG constraints, the proof of tractability stems from the reformulation into a dual model, based on value domain indicator variables. However, in contrast with these works, the possible overlapping of the value domains in our case results in a non-direct relation between the primal and

dual models. We then investigate in Section 3 an algorithm for enforcing *bound consistency* and two relaxations by decomposition. Section 4 presents computational experiments of these algorithms on randomly generated instances. The multi-dimensional variant of the problem is investigated in Section 5 where an illustration is given as well as the proof of intractability. Finally, Section 6 explains how our reformulation contrasts with previous works on other conjunctions of AMONG constraints.

### 1.1 A scheduling example

Assume $n$ events have to be scheduled inside a period of time represented by consecutive slots. Each event lasts one slot and requires resources (rooms, transport, commodities, press coverage, etc.). On one hand, resources have temporary capacities so that the number of events occurring during a time window should not exceed some value. On the other hand, resources also require a minimum number of events to happen in a time interval in order to be profit-making.

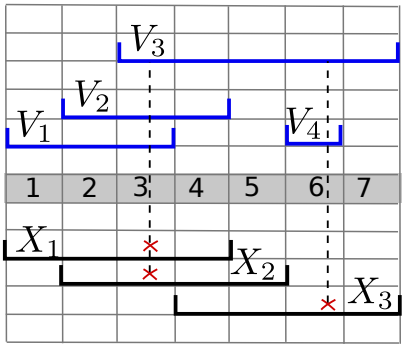


**Fig. 1.** An instance with 3 variables and 4 constraints.

In Figure 1, we consider 3 events that have to be scheduled inside the time intervals $X_1 = [1, 4]$, $X_2 = [2, 5]$ and $X_3 = [4, 7]$ respectively. We also consider 4 resource constraints. The first one requires the number of events occurring inside $V_1 = [1, 3]$ to be more than $\underline{k}_1 = 1$ and less than $\overline{k}_1 = 2$. The second requires at most $\overline{k}_2 = 2$ events inside $V_2 = [2, 4]$. For the third resource, we have $V_3 = [3, 7]$, $\underline{k}_3 = 2$, $\overline{k}_3 = 3$ and for the last one $V_4 = [6, 6]$, $\underline{k}_4 = 0$, $\overline{k}_4 = 1$.

A possible solution to the problem consists in scheduling the two first events at time 3 and the third one at time 6.

### 1.2 Problem statement

In the previous example, we want the number of elements (the events) that belong to a given set $V$ to be bounded below and above by two integers $\underline{k}$ and

$\overline{k}$, respectively. Such a condition is called an AMONG constraint [1, 2, 12]. Set $V$ is called *value domain* and interval $[\underline{k}, \overline{k}]$ its *capacity interval*.

We give now a definition of the AMONG constraint with interval value domains. The set of integer intervals is denoted by $\mathbb{IZ}$ (and by $\mathbb{IZ}_+$ for non-negative integers). The lower and upper bounds of an interval $X \in \mathbb{IZ}$ are denoted by $\underline{x}$ and $\overline{x}$.

**Definition 1.** INTERVAL-AMONG. *Given a value domain $V \in \mathbb{IZ}$ and a capacity interval $K = [\underline{k}, \overline{k}] \in \mathbb{IZ}_+$, then the constraint* AMONG$(x, V, K)$ *holds for a tuple $x = (x_j)_{j \in J} \in \mathbb{Z}^J$ iff*

$$\underline{k} \leq \text{card}\{j \in J \mid x_j \in V\} \leq \overline{k}.$$

We call a conjunction of such constraints, an INTERVAL-AMONGS constraint:

**Definition 2.** INTERVAL-AMONGS. *Given a family of intervals $V = (V_i)_{i \in I} \in \mathbb{IZ}^I$ with respective capacity intervals $K = (K_i)_{i \in I} \in \mathbb{IZ}_+^I$, then the constraint* INTERVAL-AMONGS$(x, V, K)$ *holds for a tuple $x = (x_j)_{j \in J} \in \mathbb{Z}^J$ iff*

$$\text{AMONG}(x, V_i, K_i), \quad \forall i \in I. \tag{1}$$

The satisfiability of INTERVAL-AMONGS is the problem of deciding, given a family of intervals $X = (X_j)_{j \in J} \in \mathbb{IZ}^J$, called *variable domains*, whether the constraint has a solution in $X$, that is whether there exists a tuple $x \in X$ such that INTERVAL-AMONGS$(x, V, K)$ holds.

## 2 Complexity

Régin [12] proved that the satisfiability of a conjunction of AMONG constraints on arbitrary variable $X_j \subseteq \mathbb{Z}$ and value $V_i \subseteq \mathbb{Z}$ domains is $\mathcal{NP}$-complete, even if the $X_j$'s are intervals. He also studied the case where the AMONG constraints relate the same set $X$ of variables, like in INTERVAL-AMONGS, and then proved that the problem becomes tractable when the value domains $V_i$'s are pairwise disjoint. In this section, we relax this latter condition and prove that the problem remains tractable when the value domains $V_i$'s are intervals.

**Theorem 1.** *The satisfiability of* INTERVAL-AMONGS *is in $\mathcal{P}$.*

The proof of this theorem is split in two parts. Lemma 1 shows that the problem is equivalent to the satisfiability of a system of linear inequalities $(P_L)$. Lemma 2 shows that this system can be solved in polynomial time. To introduce $(P_L)$, we first build an intermediate system $(P)$. The construction of both is also considerably lightened by making some prior assumptions that do not lose generality. We start by presenting them.

### 2.1 Preliminary assumptions

Let $m = \text{card}(I)$ be the number of value domains, i.e. the number of AMONG constraints, and let $\Sigma = \bigcup_{i \in I} V_i \subseteq \mathbb{Z}_+$ denote the union of all the value domains. First, one can assume w.l.o.g. that $\Sigma$ has at most $2m$ elements. Indeed, for any value $s \in \Sigma$, let $\mathcal{V}(s)$ denote the intersection of all intervals $V_i$ that contains $s$, with $i \in I$. For any variable $x_j$ such that $s \in X_j$, INTERVAL-AMONGS is satisfiable with $x_j \in \mathcal{V}(s)$ iff it is with $x_j = s$. As a consequence, we can merge together all the contiguous values $s$ and $s+1$ such that $\mathcal{V}(s) = \mathcal{V}(s+1)$. This leads to at most $2m$ groups of values. Further, we assume for simplicity and w.l.o.g. that $\Sigma$ is a discrete interval $[1, p] \in \mathbb{IZ}_+$ with $p = O(m)$. Notice that the size of variable domains, the size of value domains and the number of constraints can then be considered to be all of the same order. This remark will play a role in the experiments of Section 4.

### 2.2 The cardinality decomposition ($P$)

We introduce now the following Constraint Satisfaction Problem $(P)$ as an intermediate step of our transformation. It is equivalent to INTERVAL-AMONGS in the sense that $x$ is a solution of INTERVAL-AMONGS if and only if $(x, y)$ is a solution of $(P)$ for some vector $y$.

$$
\begin{array}{lrll}
(P): & \underline{k}_i \leq \sum_{s \in V_i} y_s \leq \overline{k}_i, & \forall i \in I, & (2) \\
& \text{AMONG}(\ (x_j)_{j \in J}, \{s\}, y_s\ ), & \forall s \in \Sigma, & (3) \\
& x_j \in X_j, & \forall j \in J, & \\
& y_s \in \mathbb{Z}_+, & \forall s \in \Sigma. &
\end{array}
$$

For each value $s \in \Sigma$, $y_s$ represents the number of variables $x$ assigned to $s$. In the example of §1.1, $\Sigma = [1, 7]$, and for the solution proposed, we have $y_1 = y_2 = y_4 = y_5 = y_7 = 0$, $y_6 = 1$ and $y_3 = 2$.

Constraints (3) make use of the variant of the AMONG predicate with variable capacity. They can easily be linearized in the $x$ and $y$ variables, however the reformulation of $(P)$ resulting from this linearization does not have the integrality property. Our key idea is then to drop variables $x$ and to reinject constraints (3) in the system under the form of additional linear inequalities on $y$. This way, we come up with a system with only $p$ variables whose satisfiability is still equivalent to INTERVAL-AMONGS but which has, this time, the integrality property.

Note that $(P)$ and INTERVAL-AMONGS remain equivalent regardless of whether domains are intervals or not. However, as it will be emphasized later, the following reformulation $(P_L)$ holds only if variable domains are intervals, and the resulting system may not be tractable if value domains are not intervals.

### 2.3 Equivalence between $(P)$ and $(P_L)$

**Lemma 1.** INTERVAL-AMONGS$(x, V, K)$ *is satisfiable if and only if the following system of linear inequalities has at least one integer solution* $y = (y_s)_{s\in\Sigma} \in \mathbb{Z}_+^{\Sigma}$*:*

$$(P_L): \qquad \underline{k}_i \leq \sum_{s\in V_i} y_s \leq \overline{k}_i, \qquad \forall i \in I, \tag{2}$$

$$L_{[a,b]} \leq \sum_{s\in[a,b]} y_s, \qquad \forall a \leq b \in \Sigma, \tag{3'}$$

$$\sum_{s\in\Sigma} y_s \leq n, \tag{3"}$$

*where, for each non-empty interval* $[a, b]$ *of* $\Sigma$*,* $L_{[a,b]}$ *denotes the number of variable domains included in* $[a, b]$*:* $L_{[a,b]} = \text{card}\{j \in J \mid X_j \subseteq [a, b]\}$*.*

*Proof.* We shall prove that there is a mapping between the feasible solutions $y$ of $(P_L)$ and the feasible solutions $x$ of INTERVAL-AMONGS. Assume there exists $x \in \Pi_{j\in J} X_j \subseteq \Sigma^n$ satisfying (1) and let $y_s$ denote, for each value $s \in \Sigma$, the number of entries in $x$ which are equal to $s$:

$$y_s = \text{card}\{j \in J \mid x_j = s\}, \quad \forall s \in \Sigma.$$

Then, $y$ is a feasible solution of $(P_L)$, as the satisfaction of constraints (2) directly holds from (1), constraints (3') from $X_j \subseteq [a, b] \implies x_j \in [a, b]$ and (3") from $\sum_{s\in\Sigma} y_s = n$.

Conversely, let $y$ be a feasible solution of $(P_L)$. Consider the capacitated directed bipartite graph $G = (J \cup \Sigma, E, c)$ on the arc set $E = \{(j, s) \in J \times \Sigma \mid s \in X_j\}$ with capacity $c_e = 1$ on each arc $e \in E$. We add to $G$ a source $u$ and an arc $(u, j)$ of capacity 1, for all $j \in J$, a sink $v$ and an arc $(s, v)$ of capacity $y_s$ for all $s \in \Sigma$ (see Figure 2). Every feasible $(u, v)$-flow of value $n$ defines a feasible solution $x$ of INTERVAL-AMONGS, by setting $x_j$ the flow on arc $(j, s)$, for all $j \in J$. To prove there exists such a flow, we use Hoffman's theorem (see e.g. [9]) and show that the capacity of any $(u, v)$-cutset $(U, V)$ of $G$ is greater than or equal to $n$. Since (3") imposes the flow to be less than or equal to $n$, then the maximal flow will be exactly $n$.

Let $(U, V)$ be a cutset of $G$, $\Sigma_U = \Sigma \cap U$, $J_U = J \cap U$, and $J'_U = \{j \in J_U \mid X_j \subseteq \Sigma_U\}$. By definition of $G$, the arcs in the cutset $(U, V)$ are of the form, either $(u, j)$ with $j \in J \setminus J_U$ and capacity 1, or $(j, s) \in E$ with $j \in J_U$, $s \in \Sigma \setminus \Sigma_U$ and capacity 1, or $(s, v)$ with $s \in \Sigma_U$ and capacity $y_s$. The total capacity of the first set of arcs is $\text{card}(J \setminus J_U)$. The capacity of the second set is $\text{card}(J_U \setminus J'_U)$ since, for all $j \in J_U \setminus J'_U$, $X_j \not\subseteq \Sigma_U$, then there exists at least one arc $(j, s) \in E$ in the cutset. Last, to bound the capacity $\sum_{s\in\Sigma_U} y_s$ of the third set, we first write $\Sigma_U$ as the union of $r$ disjoint intervals: $\Sigma_U = [a_1, b_1] \cup \ldots \cup [a_r, b_r]$. Now, by definition: $\text{card}(J'_U) = \text{card}\{j \in J \mid X_j \subseteq \Sigma_U\}$. Since the $X_j$ are all intervals, the condition $X_j \subseteq \Sigma_U$ implies $X_j$ is included in exactly one interval $[a_l, b_l]$ with $1 \leq l \leq r$. Therefore:

$$\text{card}(J'_U) = \sum_{l=1}^{r} \text{card}\{j \in J \mid X_j \subseteq [a_l, b_l]\} = \sum_{l=1}^{r} L_{[a_l,b_l]}$$

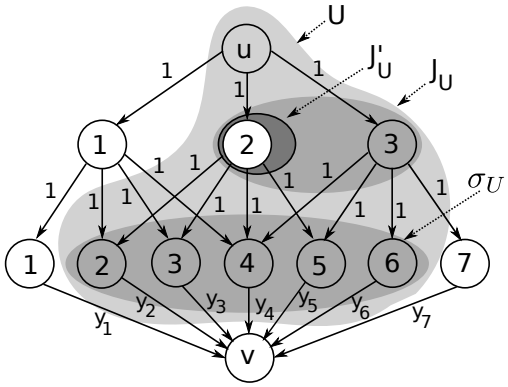


**Fig. 2. The network flow model of constraints** (3) corresponding to the example of §1.1, and an example of a cutsest used in the proof of Lemma 1. The cutset $U$ is painted in light gray. The subset of nodes $J_U$ and $\Sigma_U$ are in medium gray and $J'_U$ in dark gray.

which implies, according to (3'):

$$\mathrm{card}(J'_U) \leq \sum_{l=1}^{r} \sum_{s\in[a_l,b_l]} y_s = \sum_{s\in\Sigma_U} y_s.$$

So, the capacity of the third set is at least $\mathrm{card}(J'_U)$. Hence, the total capacity of the cutset is at least $n$ and the result follows. ■

### 2.4 Tractability of $(P_L)$

Remark that the proof in the previous paragraph remains true when relaxing in $(P_L)$ every constraint in (3') corresponding to some interval $[a,b]$ that does not include any variable domain $X_j$. We can still decrease the number of constraints in $(P_L)$ by merging every constraint in (2) to the constraint in (3') corresponding to the same interval. More precisely, $(P_L)$ can be rewritten as:

$$(P_L): \qquad L'_{[a,b]} \leq \sum_{s\in[a,b]} y_s \leq U'_{[a,b]}, \quad \forall [a,b] \subseteq \Sigma,$$

where, for any interval $[a,b] \subseteq \Sigma$:

$$L'_{[a,b]} := \begin{cases} \max(L_{[a,b]}, \underline{k}_i) & \text{if } [a,b] \text{ coincides with } V_i \text{ for some } i \in I, \\ L_{[a,b]} & \text{otherwise.} \end{cases}$$

$$U'_{[a,b]} := \begin{cases} \min(n, \overline{k}_i) & \text{if } [a,b] = \Sigma = V_i \text{ for some } i \in I, \\ n & \text{else if } [a,b] = \Sigma, \\ \overline{k}_i & \text{else if } [a,b] = V_i \text{ for some } i, \\ +\infty & \text{otherwise.} \end{cases}$$

Remember now that $\Sigma = [1, p]$. To further simplify, we reformulate $(P_L)$ as the following system of linear inequalities:

$$(P_T): \qquad z_b - z_a \leq d_{ab}, \qquad \forall a, b \in \{0\} \cup \Sigma = [0, p] \tag{4}$$

using a new change of variables: $z_0 = 0$, $z_b = \sum_{s=1}^{b} y_s$ $(\forall b \in \Sigma)$, and defining matrix $d = (d_{ab}) \in Z^{(p+1)\times(p+1)}$ as: $d_{ab} = \begin{cases} U'_{[a+1,b]} & \text{if } a < b \\ -L'_{[b+1,a]} & \text{if } a > b \\ 0 & \text{if } a = b, \end{cases} \quad \forall a, b \in [0, p].$

System $(P_T)$ is a *Temporal Constraint Network*, so-called by Dechter et al [7], as such inequalities are frequently encountered as precedence and temporal constraints in planning and scheduling problems. The satisfiability of such systems can be checked in polynomial time.

**Lemma 2.** *An integer solution of $(P_L)$ can be searched in polynomial time.*

*Proof.* Let $G_d$ be a complete directed graph with $p+1$ vertices numbered from 0 to $p$, and with weight $d_{ab}$ on each arc $(a, b)$ of $G_d$. Shostak's theorem [13] states that $(P_T)$ is feasible if and only if graph $G_d$ has no negative cycle. Building the weighted graph $G_d$ can be done in $O(p^2)$ time, and checking that it has no negative cycle can be done in $O(p^3)$ time using Floyd-Warshall's algorithm (see e.g. [9]). ■

## 3 Bound consistency

We focus now on filtering algorithms for the Interval-Amongs constraint. Since variable domains are intervals, we are only interested in bound consistency (BC). In particular, we do not consider *generalized arc consistency*.

Remember first that bound consistency can be achieved in polynomial time if satisfiability in any given domain can be checked in polynomial time. It suffices to embed a satisfiability check inside a *shaving* loop, where each variable is instantiated in turn to its bounds until a fixpoint is reached. Hence, as corollary of Theorem 1, the bound consistency for Interval-Amongs can be achieved in polynomial time (while in the general – non-interval – case, it is $\mathcal{NP}$-hard). However, the complexity of the shaving algorithm is in $O(n^2m^4)$, as detailed in §3.2. This complexity is too high for practical purposes. So, we first study different decompositions of Interval-Amongs, from which faster algorithms will be derived afterwards.

### 3.1 Consistency strength

When a constraint is semantically equivalent to the conjunction of two constraint systems $c_1$ and $c_2$, we note this decomposition $(c_1, c_2)$ and call *BC on* $(c_1, c_2)$ the fixpoint of BC filtering on the two constraint systems $c_1$ and $c_2$, taken separately.

**AMONG-Based Decomposition.** By definition, INTERVAL-AMONGS is a conjunction of AMONG constraints. Let us call this decomposition the AMONG-based decomposition. We have the following lemma:

**Lemma 3.** *BC on* INTERVAL-AMONGS *is strictly stronger than BC on the* AMONG*-based decomposition.*

*Proof.* Consider value domains $V_1 = [1,1]$, $V_2 = [2,2]$ with $K_1 = K_2 = [1,1]$ and two variables $x_1, x_2$. The domain $X_1 = X_2 = [0,2]$ is BC with respect to (w.r.t.) both AMONG while the bound $x_1 = 0$ cannot satisfy INTERVAL-AMONGS. ■

**CARDINALITY-Based Decomposition.** Another decomposition grows out naturally from our complexity study and the reformulation $(P)$, at the price of introducing dual cardinality variables $y$ (whose initial domains are $\mathbb{Z}_+$). $(P)$ is the conjunction of two sub-systems of constraints (2) and (3), each being considered as one global constraint (algorithms achieving BC for these two constraints are introduced in §3.2). We note $((2),(3))$ this decomposition. It also hinders bound consistency, as the following counter-example shows.

**Lemma 4.** *BC on* INTERVAL-AMONGS *is strictly stronger than BC on* $((2),(3))$.

*Proof.* Consider two variables with domains $X_1 = [1,3]$ and $X_2 = [1,3]$, one value domain $V = [1,3]$ with cardinality $K_1 = [1,1]$ and $Y_1 = Y_2 = Y_3 = [0,1]$. It is BC w.r.t (2) and w.r.t (3). However, INTERVAL-AMONGS has no solution since both variables take their values in $[1,3]$ while the number of variables in this interval is bounded by $\overline{k}_1 = 1$. ■

We can also propose a decomposition $(P_L,(3))$ that we will call the CARDINALITY-based decomposition. Next lemmas shows that this decomposition is stronger than $((2),(3))$ but still weaker than INTERVAL-AMONGS.

**Lemma 5.** *BC on the* CARDINALITY*-based decomposition is strictly stronger than BC on* $((2),(3))$.

*Proof.* Constraint (2) is implied by $(P_L)$, so BC on $(P_L,(3))$ is stronger than BC on $((2),(3))$. It is actually strictly stronger: the example in the proof of Lemma 4 is not BC w.r.t. $(P_L)$ since $X_1 \subset [1,3]$ and $X_2 \subset [1,3]$ imposes $2 \leq y_1 + y_2 + y_3$ while $y_1 + y_2 + y_3 \leq 1$, an inconsistent system. ■

**Lemma 6.** *BC on* INTERVAL-AMONGS *is strictly stronger than BC on the* CARDINALITY*-based decomposition.*

*Proof.* Consider two variables $x_1, x_2$ with domains $X_1 = [1,2]$ and $X_2 = [1,4]$ and value domains $V_1 = [1,2]$, $V_2 = [2,3]$ and $V_3 = [3,4]$ with cardinalities $K_1 = [1,1]$, $K_2 = [1,1]$ and $K_3 = [1,1]$. Note that (3') does not introduce additional constraint to (2). The initial domains $Y_1{=}\cdots{=}Y_4{=}\mathbb{Z}_+$ are reduced to $Y_1{=}\cdots{=}Y_4{=}[0,1]$ by (2). BC is then achieved w.r.t. $(P_L,(3))$

– The domain $Y_1 \times \cdots \times Y_4$ is BC w.r.t (2) since each value $1, \ldots, 4$ can be either discarded or taken once.
– The domain $X_1 \times X_2 \times Y_1 \times \cdots \times Y_4$ is BC w.r.t. (3). First, every value in $\{1,\ldots,4\}$ can be taken either by 0 or 1 variable among $x_1$ and $x_2$. Second, all the values of $X_1$ and $X_2$ can be taken.

However, the domain is not BC w.r.t. INTERVAL-AMONGS. Indeed, the bound $x_2 = 1$ cannot satisfy the constraint. Either, $x_1 < 2$ and the number of variables inside $V_2$ is $0 \notin K_2$. Either $x_1 = 2$ and the number of variables inside $V_1$ is $2 \notin K_1$. ■

**Using both decompositions.** We have first the following result:

**Lemma 7.** *BC on the* CARDINALITY*-based decomposition and on the* AMONG*-based decomposition are not comparable.*

*Proof.* The example in the proof of Lemma 3 is BC w.r.t. the AMONG-based decomposition, but not w.r.t. the CARDINALITY-based decomposition where $y_0 + y_1 + y_2 \leq 2$, $1 \leq y_1$ and $1 \leq y_2$ imply $y_0 = 0$ which forces $x_1 > 0$. Conversely, the example in the proof of Lemma 6 is BC w.r.t the CARDINALITY-based but not the AMONG-based decomposition. Indeed, $X_1 \subseteq V_1$ and $K_1 = [1, 1]$, so $x_2 = 1$ is filtered out by $\text{AMONG}(X, V_1, K_1)$. ■

Merging the two decompositions does still not reach the BC of the constraint:

**Lemma 8.** *BC on* INTERVAL-AMONGS *is strictly stronger than BC on the conjunction of the* CARDINALITY*-based and the* AMONG*-based decomposition.*

*Proof.* We just have to slightly modify the example in the proof of Lemma 6. Set $X_1$ to $[0, 2]$ instead of $[1, 2]$. The bound $x_2 = 1$ is still BC w.r.t. CARDINALITY-based decomposition and it is now also BC w.r.t. the AMONG-based decomposition (since $X_1$ is not included in $V_1$ anymore). ■

### 3.2 Filtering algorithms

This section presents some algorithms and complexities. The complexities will be given with respect to $n$ and $m$ only because $m$ is also, within a constant factor, the maximal width for both variable domains and value domains (see §2.1). In particular, if we call an $x$-value a pair $(x_j, v)$ such that $v \in X_j$, the total number of $x$-values is bounded by $n \times \max_j |X_j| = nm$. Similarly, $n$ is also an upper bound for the capacities so that the number of $y$-values is bounded by $m \times \max_i |Y_i| = mn$.

**INTERVAL-AMONGS.** First, as said in §2.4, system $(P_L) : L'_{[a,b]} \leq \sum_{s \in [a,b]} y_s \leq U'_{[a,b]}, \forall [a, b] \subseteq \Sigma$, can be cast into a temporal constraint network $(P_T)$, providing a change of variables. The satisfiability can then be checked with Floyd-Warhsall algorithm (FW). This leads to the following complexity:

**Lemma 9.** *BC on* Interval-Amongs *can be enforced in* $O(n^2m^4)$.

*Proof.* $O(n^2m^4)$ is the product of the time required to check the satisfiability of $(P_T)$ using Fw, which is $O(p^3) = O(m^3)$, by the number of iterations in the outer shaving loop, which is $n^2m$ in the worst case (where $2n$ bounds are checked each time one of the $nm$ $x$-value is removed). ■

This complexity cannot be easily improved as all shortest paths algorithms share the same complexity on dense graphs like here. Fw algorithm also works incrementally in $O(p^2)$ if the distance of one edge is modified between two runs. However, instantiating one variable to its bound in the shaving loop can potentially modify $L_{[a,b]}$ for up to $p$ intervals $[a, b]$, i.e., impact $p$ distances at the same time. Furthermore, the satisfiability check does not exhibit the support for the current tested bound in terms of the $x$ variables (the $x$ variables disappear in the check) so that the shaving loop has to naively sweep across all the values.

**Among-based decomposition.** Propagating the conjunction of Among gives the following complexity:

**Lemma 10.** *BC on the* Among *decomposition can be enforced in* $O(n^2m^2)$.

*Proof.* The cost of applying BC on a single Among constraint is $O(n)$ (see e.g., [12, 15]). In the worst case, there is $O(m)$ calls of no effect between two removals, and all the $x$-values are eventually removed so that the total number of calls is $O(nm^2)$. ■

**Cardinality-based decomposition.** Let us move to the Cardinality-based decomposition, i.e., $(P_L, (3))$.

Let us first focus on $(P_L)$. It turns out that Fw run on $(P_T)$ does not only check satisfiability of $(P_L)$ but also provides all the information necessary to enforce BC on the $y$ variables. Indeed, taking the notations of §2.4, any path $(a = a_0, a_1, \ldots, a_{k+1} = b)$ in $G_d$ induces by transitivity from $(P_T)$ the relation $z_b - z_a \leq \sum_{i=0}^{k} d_{a_i a_{i+1}}$. Hence, each value $d_{ab}$ in $(P_T)$ can be replaced by the distance (i.e. the length of a shortest path) between $a$ and $b$ in $G_d$. This is precisely what Fw does. Now, $d_{(s-1)s}$ is nothing but the upper bound of the feasible domain for $z_s - z_{s-1} = y_s$ while $-d_{s(s-1)}$ is the lower bound of the feasible domain for $-z_{s-1} + z_s = y_s$. So, after the execution of Fw, the BC of $(P_L)$ is nothing but $[-d_{s(s-1)}, d_{(s-1)s}]$ for every $y_s$, $s \in \Sigma$.

Let us focus now on (3). The system is nothing but a Global-Cardinality (Gcc) constraint. In its original form [11], the $Y$ are considered as constant intervals. Fortunately, a BC filtering algorithm has also been devised in [8] with the cardinalities being variables, with asymptotic running time in $O(n + m)$.

There is now a tricky detail. We do not get the BC on the Cardinality-based decomposition simply by plugging both algorithms together in a fixpoint loop. Indeed, one also has to increment $d_{[a,b]}$ in $(P_T)$ each time a domain $X_j = [a, b + 1]$ or $X_j = [a - 1, b]$ is filtered to $[a, b]$ by (3). In other words, Fw can

be awoken either by the removal of a value from the $y$ or the $x$ variables. That precaution said, putting both algorithms together in a fixpoint loop gives the BC on $(P_L, (3))$ with the following worst-case complexity.

**Lemma 11.** *BC on the* Cardinality*-based decomposition can be enforced in* $O(nm^3 + n^2m)$.

*Proof.* There is only two constraints, linked by the $x$ and $y$ variables. Fw can now be called incrementally and takes $O(m^2)$ time. Gcc takes $O(n+m)$. In the worst case, each time a constraint is called, one value must be removed (otherwise the fixpoint is reached, the other constraint being already consistent). Hence, each algorithm is called for half of the total number of $x$-values and $y$-values, i.e., $O(nm)$ times. This gives $O(nm \times (m^2 + (n+m))) = O(nm^3 + n^2m)$. ■

## 4 Computational evaluation

We have proposed in the previous section a filtering algorithm for Interval-Among derived from the Cardinality-based decomposition, as an alternative to the (natural) Among-based decomposition.

The consistencies they enforce are not formally comparable, and neither their time complexities are. However, we can say that our decomposition better captures the globality of the constraint in the sense that it is only made of 2 constraints, $(P_L)$ and (3), instead of $m$. We present in this section some experiments we have made to support this claim.

First of all, both decompositions have been implemented in the Choco 2.1.2 platform [14]. The Interval-Among package, including the following benchmark, is freely available on the authors' web sites.

We have decided to base the comparison on a sequence of randomly generated instances of Interval-Among. Let us briefly explain how an instance is generated. First, we fix $p = n$, that is, the instance has a set of $n$ variables and $n$ value domains, variable and value domains being random subintervals of $[0, n]$. This limits the number of parameters to consider and allows to compare the theoretical worst running times required for enforcing BC on the two decompositions: $O(n^4)$ in both cases. To set capacity bounds, we start from an *a priori* solution and fix capacities accordingly. More precisely, we create a tuple $(\tau_1, \ldots, \tau_n)$ by randomly picking a value $\tau_j$ inside each variable domain $X_j$. Then, for each value interval $V_i$, we count the number $n_i$ of $\tau_j$'s that belongs to $V_i$ and set $K_i = [n_i - 1, n_i + 1]$. A single Interval-Among constraint, especially with relaxed capacities, usually induces a huge number of symmetries. For this reason, we only look for one solution.

Now, for each value of $n$ from 10 to 32, we have generated 100 instances as explained above and run two solvers, one for each decomposition. Solvers are stopped as soon as a first solution is found. The default variable/value choice heuristic `DomOverWDeg` of Choco has been used.

We have compared running times using the following logarithmic scale. For each instance, if we denote by $t_1$ the time required for the CARDINALITY decomposition and $t_2$ the time required for the AMONG decomposition, the outcome is one of the 5 following answers:

(1) $t_2 \geq 10^2 t_1$ (2) $10^2 t_1 > t_2 \geq 10 t_1$ (3) $10 t_1 > t_2$ and $t_1 < 10 t_2$
(4) $10 t_2 \leq t_1 < 10^2 t_2$ (5) $10^2 t_2 \leq t_1$

We have then counted the number of instances that yields answer (1) and so on. Instances for which $t_1 < 0.1s$ and $t_2 < 0.1s$, i.e., the "easiest" ones are discarded to avoid spurious results with instances where the running time is dominated by the initialization of Java structures.

Figure 3 reports the results we have gotten. It clearly shows that the CARDINALITY decomposition is the most efficient one: The solid curve shows the

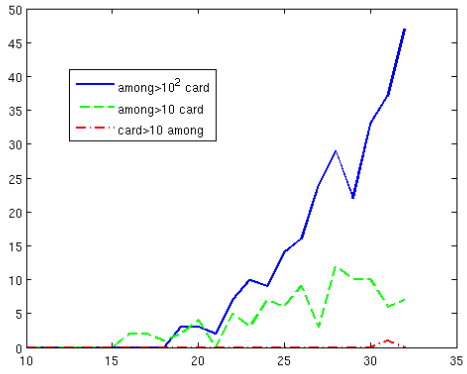


**Fig. 3.** Solving time comparison between the CARDINALITY-based and the AMONG-based decompositions for instance size varying from 10 to 32. The solid and dashed curves depict the number of instances on which the CARDINALITY decomposition improves upon the AMONG decomposition. The dash-dot curve depicts the opposite case.

number of instances where the running time is at least 100 times faster with the CARDINALITY decomposition. We see that this number grows quickly with $n$. For $n = 32$, we get almost half of the instances. For the other instances, either the cardinality decomposition is 10 times faster (dashed curve) or the first solution was very easy to find for both algorithms (not depicted here). There was no instance where the AMONG decomposition was 100 times faster and only a single one (for $n = 31$) where it was 10 times faster (dash-dot curve).

## 5 The multi-dimensional INTERVAL-AMONGS

An other application of INTERVAL-AMONGS arises in the context of sensor networks: the problem is to localize geographically, in 2D or 3D, a number of targets

with a number of radars. In the $d$-dimensional case, variable and value domains become $d$-vectors of discrete intervals, $\vec{X}$ and $\vec{V}$. For convenience, a vector of intervals is identified to a *box*, that is an element of $\mathbb{IZ}^d$, the cross product of its components. Unfortunately the multi-dimensional variant becomes untractable as stated in this section.

### 5.1 A target localization example

Assume the coordinates of $m$ targets in the plane have to be determined from the intensity of the signal measured by $n$ antennas. An antenna only detects objects in a given area and the intensity of the measured signal gives bounds on the number of detected targets. Basically, the higher the signal, the more targets in the area covered by the antenna.

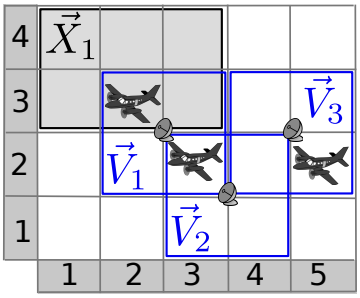


**Fig. 4.** An instance in two dimensions with 3 (vector) variables and 3 constraints.

In Figure 4, we look for the positions of 3 aircrafts, each vector having an a priori domain, like $\vec{X}_1$ for the first aircraft (the other domains are omitted for clarity). We also have 3 detection areas, $\vec{V}_1$, $\vec{V}_2$ and $\vec{V}_3$, each centered on a different antenna. We know from the signal of the first antenna that between 2 and 3 targets are in $\vec{V}_1$ (high signal). Similarly, the number of targets in $\vec{V}_2$ is between 1 and 2 (medium signal) and in $\vec{V}_3$ between 0 and 1 (low signal). A possible solution is depicted.

### 5.2 Complexity

**Proposition 1.** *The satisfiability of* Interval-Amongs *in* 2*-dimension is* $\mathcal{NP}$*-complete.*

*Proof.* A tuple is a certificate so the problem is in $\mathcal{NP}$. We transform now the *rectangle clique partition problem*, which was proven to be $\mathcal{NP}$-complete (see Section 4 in [5]). More precisely, let us consider problem $P$ defined as follows:
*Input*: $m$ boxes $\vec{Y}_1, \ldots, \vec{Y}_m$ in $\mathbb{IZ}^2$ and an integer $k \in \mathbb{Z}_+$.

*Question*: Is there $m$ vectors $\vec{y}_1, \ldots, \vec{y}_m$ in $\mathbb{Z}^2$ such that $\vec{y}_i \in \vec{Y}_i$ for all $i$, $1 \leq i \leq m$, and $\text{card}\{\vec{y}_1, \ldots, \vec{y}_m\} \leq k$ ?

We apply now the following transformation. We build, in linear time, an instance $P'$ of INTERVAL-AMONGS with $n = k$ variable domains $\vec{X}_j = \bigcup_{i=1}^{m} \vec{Y}_i$, $\forall 1 \leq j \leq k$, and with $m$ value domains $\vec{V}_i = \vec{Y}_i$ and $K_i = [1, k]$, $\forall 1 \leq i \leq m$.

Assume $P$ is satisfiable and consider a solution tuple $\vec{y} = (\vec{y}_1, \ldots, \vec{y}_m)$. Since $\text{card}\{\vec{y}_1, \ldots, \vec{y}_m\} = k$, there exists a tuple of $k$ distinct vectors $\vec{x} = (\vec{x}_1, \ldots, \vec{x}_k)$ of $\mathbb{Z}^2$ such that $\{\vec{y}_1, \ldots, \vec{y}_m\} = \{\vec{x}_1, \ldots, \vec{x}_k\}$. For all $j$, $\vec{x}_j \in \vec{X}_j$. Next, for all $i$, there exists at least one vector in $\vec{x}$ and at most $k$ that coincide with $\vec{y}_i$. Hence, the number of $\vec{x}_j$'s that belong to $\vec{Y}_i$ is in $[1, k]$. So $P'$ is satisfiable.

Conversely, consider a solution tuple $(\vec{x}_1, \ldots, \vec{x}_k)$ to $P'$. For all $i$, there exists at least one $\vec{x}_j$ such that $\vec{x}_j \in \vec{Y}_i$. Put $\vec{y}_i = \vec{x}_j$. We have $\vec{y}_i \in \vec{Y}_i$ and, by construction, the tuple $(\vec{y}_1, \ldots, \vec{y}_m)$ has at most $k$ distinct vectors. So the answer to $P$ is "yes". ■

## 6 Related Works

Our approach shares some similarities with preceding works on other conjunctions of AMONG constraints. Note first that reformulation $(P)$ is an extension of the one proposed in [3] for one AMONG constraint. $(P)$ is composed of a sub-system of *capacity constraints* (2) on the dual variables $(y_s)_{s \in \Sigma}$, and a sub-system of *channelling constraints* (3) between the $x$ and $y$ variables.

For the conjunction of AMONG constraints on disjoint value domains, Régin [12] encodes the capacity constraints as one GCC on value domain indicator variables $y$ which are channelled to the $x$ variables by the relation $y_j = i \iff x_j \in V_i$. For SEQUENCE, the channelling is even simpler since all value domains are equal and thus can be assimilated to $\{0, 1\}$: $y_j = 1 \iff x_j \in V$. Brand et al. [4] encode the capacity constraints as a temporal constraint network $\underline{k} \leq \sum_{s=j}^{j+l} y_s \leq \overline{k}$ $(\forall j \in J)$. Maher et al. [10] transform it thereafter into a linear program, and then into a flow network model on which they apply an incremental filtering algorithm similar to GCC [11].

In both cases, as the capacity+channelling constraint system is Berge-acyclic, then the flow-based filtering on the dual model achieves AC on the original model. This is not our case, as our channelling (3) is itself a conjunction of AMONG constraints where the $y$ variables play the role of the variable capacities. As a consequence, this sub-system can also be reformulated as a flow network (see Figure 2) but where the $y$ represent the arc capacities instead of the flow values. We employ this flow model to prove the polynomial reduction from $(P_L)$ and, in part, to filter our channelling sub-system as we encode it as a GCC, but we cannot use it to filter the dual system, in contrast to [12, 10].

Our dual system is actually encoded as a temporal constraint network $(P_T)$, like in [4, 10]. However, because our network is a complete graph, we use the standard Floyd-Warshall algorithm to filter values. Reducing it to a flow problem as in [10] would require a specific structure and using Johnson's algorithm or

the incremental variant of [6] as in [4] would be a better option only if the graph was sparse.

## 7 Conclusion

Providing that domains are intervals, we have shown that a conjunction of AMONG constraints, named INTERVAL-AMONGS, becomes a tractable constraint. We have also introduced different decompositions of the constraint and compared them on the basis of filtering power. The first is basically an "horizontal" decomposition (where we consider all the values of a single interval) and comes from the very definition of the constraint. The second is a "vertical" one (where we consider a single value shared by all the intervals) and turns to be the right formulation to prove our main theorem. Decomposition is a remarkable aspect of constraint programming as it automatically yields, through the process of constraint propagation, a composition of existing algorithms, each initially designed for a different purpose. We have illustrated this well by plugging together the Floyd-Warshall algorithm (for the temporal constraint network) and a flow-based filtering (for the GLOBAL-CARDINALITY constraint). Both decomposition have been implemented and compared on random instances. Results reinforce the superiority of the second decomposition for tackling our problem.

In this paper, we also investigated the multi-dimensional variant of INTERVAL-AMONGS, motivated by a target localization problem. We have shown that achieving BC in this case remains $\mathcal{NP}$-hard. Note that this constraint is naturally decomposable into its $d$ projections, which brings us back to the one-dimensional case. However this additional decomposition hinders filtering a lot, as the upper capacities are canceled by the decomposition. On the contrary, the AMONG constraint has a straightforward extension to the multi-dimensional case. Hence, the situation is now more favorable to an approach based on the (*vector*-)AMONG decomposition. In future works, we aim at finding a tighter decomposition that exploits INTERVAL-AMONGS to solve this $\mathcal{NP}$-hard problem.

Also, the algorithm presented in this paper to enforce bound consistency of INTERVAL-AMONGS simply embeds a satisfiability check inside a heavy shaving loop. The existence of a more elegant algorithm is still an open question. We conjecture that it is a challenging question as such result would subsequently prove Theorem 1, which was precisely the delicate part of the present work.

## References


1. N. Beldiceanu and E. Contejean. Introducing Global Constraints in CHIP. *Journal of Mathematical and Computer Moddeling*, 20(12):97–123, 1994.
2. C. Bessière, E. Hebrard, B. Hnich, Z. Kiziltan, and T. Walsh. Among, Common and Disjoint Constraints. In *CSCLP: Recent Advances in Constraints*, volume 3978 of *Lecture Notes in Computer Science*, pages 29–43, 2005.
3. C. Bessière, E. Hebrard, B. Hnich, Z. Kiziltan, and T. Walsh. The Range and Roots Constraints: Specifying Counting and Occurrence Problems. In *IJCAI*, pages 60–65, 2005.

4. S. Brand, N. Narodytska, C.G. Quimper, P. Stuckey, and T. Walsh. Encodings of the sequence constraint. In *Principles and Practice of Constraint Programming (CP'08)*, volume 4741 of *Lecture Notes in Computer Science*, pages 210–224, 2007.
5. G. Chabert, L. Jaulin, and X. Lorca. A Constraint on the Number of Distinct Vectors with Application to Localization. In *15th International Conference on Principles and Practice of Constraint Programming (CP'09)*, volume 5732 of *Lecture Notes in Computer Science*, pages 196–210, 2009.
6. S. Cotton and O. Maler. Fast and flexible difference constraint propagation for DPLL(T). In *SAT'06*, pages 170–183, 2006.
7. R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artificial Intelligence*, 49(1-3):61–95, 1991.
8. I. Katriel and S. Thiel. Complete Bound Consistency for the Global Cardinality Constraint. *Constraints*, 10(3):191–217, 2005.
9. E. Lawler. *Combinatorial Optimization: Networks and Matroids*. Saunders College Publishing, 1976.
10. M.J. Maher, N. Narodytska, C-G. Quimper, and T. Walsh. Flow-Based Propagators for the SEQUENCE and Related Global Constraints. In *Principles and Practice of Constraint Programming (CP'08)*, volume 5202 of *Lecture Notes in Computer Science*, pages 159–174, 2008.
11. J-C. Régin. Generalized Arc Consistency for Global Cardinality Constraint. In *13th conference on Artificial intelligence*, AAAI'96, pages 209–215, 1996.
12. J-C. Régin. Combination of Among and Cardinality Constraints. In *Integration of AI and OR Techniques in Constraint Programming (CPAIOR'05)*, volume 3524 of *Lecture Notes in Computer Science*, pages 288–303, 2005.
13. R. Shostak. Deciding linear inequalities by computing loop residues. *Journal of the ACM*, 28(4):769–779, 1981.
14. CHOCO Team. choco: an open source java constraint programming library. Research report 10-02-INFO, Ecole des Mines de Nantes, 2010.
15. W-J. van Hoeve, G. Pesant, L-M. Rousseau, and A. Sabharwal. New filtering algorithms for combinations of among constraints. *Constraints*, 14:273–292, 2009.