A Generalized Arc-Consistency Algorithm for a Class of Counting Constraints: Revised Edition that Incorporates One Correction

Thierry Petit and Nicolas Beldiceanu and Xavier Lorca
Mines-Nantes, LINA UMR CNRS 6241,
4, rue Alfred Kastler, FR-44307 Nantes, France.
{Thierry.Petit, Nicolas.Beldiceanu, Xavier.Lorca}@mines-nantes.fr

Abstract

This paper introduces the seq_bin meta-constraint with a polytime algorithm achieving generalized arc-consistency according to some properties. seq_bin can be used for encoding counting constraints such as change, smooth or increasing_value. For some of these constraints and some of their variants GAC can be enforced with a time and space complexity linear in the sum of domain sizes, which improves or equals the best known results of the literature.

1 Introduction

Many constraints are such that a counting variable is equal to the number of times a given property is satisfied in a sequence of variables. To represent some of these constraints in a generic way, we introduce the seq_bin(N, X, C, B) meta-constraint, where N is an integer variable, X is a sequence of integer variables and C and B are two binary constraints.

Based on the notion C-stretch, a generalization of stretch [Pesant, 2001] where the equality constraint is made explicit and is replaced by C, seq_bin holds if and only if two conditions are both satisfied: (1) N is equal to the number of C-stretches in the sequence X, and (2) B holds on any pair of consecutive variables in X.

Among the constraints that can be expressed thanks to seq_bin, many were introduced for solving real-world problems, e.g., change [Cosytec, 1997] (time tabling problems), smooth [Beldiceanu et al., 2010a] (time tabling and scheduling), or increasing_value [Beldiceanu et al., 2010b] (symmetry breaking for resource allocation problems).

The main contribution of this paper is a generic polytime filtering algorithm for seq_bin, which achieves generalized arc-consistency (GAC) according to some conditions on B and C. This algorithm can be seen as a generalization of the increasing_value filtering algorithm [Beldiceanu et al., 2010b]. Given n the size of X, d the maximum domain size, and \(\Sigma_d\) the sum of domain sizes, we characterize properties on C and B which lead to a time and space complexity in \(O(\Sigma_d)\). These properties are satisfied when seq_bin represents increasing_value, and several variants of change (provided its parameter is a monotonic binary constraint, e.g., \(\leq\), \(<\), \(\geq\), \(>\)). For these constraints, our technique improves or equals the best known results.

Section 2 provides the definitions used in this paper. Section 3 defines seq_bin and shows how to express well-known constraints with seq_bin. Section 4 provides a necessary and sufficient condition for achieving GAC. Section 5 details the corresponding GAC filtering algorithm. Finally, Section 6 discusses about related works and Section 7 concludes.

2 Background

A Constraint Network is defined by a sequence of variables \(X = [x_0, x_1, \ldots, x_{n-1}]\), a sequence of domains \(D\), where each \(D(x_i) \in D\) is the finite set of values that variable \(x_i\) can take, and a set of constraints \(C\) that specifies the allowed combinations of values for given subsets of variables. \(\min(x)\) (resp. \(\max(x)\)) is the minimum (resp. maximum) value of \(D(x)\). A sequence of variables \(X' = [x_i, x_{i+1}, \ldots, x_j]\), \(0 \leq i \leq j \leq n-1\) (resp. \(i > 0\) or \(i < n-1\), is a subsequence (resp. a strict subsequence) of X and is denoted by \(X' \subseteq X\) (resp. \(X' \subset X\)). \(A[X]\) denotes an assignment of values to variables in X. Given \(x_i \in X\), \(A[x_i]\) is the value of \(x_i\) in \(A[X]\). \(A[X]\) is valid if and only if \(\forall x_i \in X, A[x_i] \in D(x_i)\). An instantiation \(I[X]\) is a valid assignment of \(X\). Given \(x_i \in X\), \(I[x_i]\) is the value of \(x_i\) in \(I[X]\). Given the sequence \(X\) and \(i, j\) two integers such that \(0 \leq i \leq j \leq n-1\), \(I[x_i, x_j]\) is the projection of \(I[X]\) on \([x_i, x_{j+1}, \ldots, x_j]\). A constraint \(C(X) \in C\) specifies the allowed combinations of values for \(X\). We also use the simple notation \(C(C(X))\) defines a subset \(R_C(D)\) of the cartesian product of the domains \(\prod_{x_i \in X} D(x_i)\). If \(X\) is a pair of variables, then \(C(X)\) is binary. We denote by \(vCw\) a pair of values \((v, w)\) that satisfies a binary constraint \(C\). \(\neg C\) is the opposite of \(C\), that is, \(\neg C\) defines the relation \(\neg R_C(D) = \prod_{x_i \in X} D(x_i) \setminus R_C(D)\). A feasible instantiation \(I[X]\) of \(C(X)\) is an instantiation which is in \(R_C(D)\). We say that \(I[X]\) satisfies \(C(X)\), or that \(I[X]\) is a support on \(C(X)\). Otherwise, \(I[X]\) violates \(C(X)\). If \(C\) is a binary constraint on \(X = [x_i, x_{i+1}]\) and \(v \in D(x_i)\), then the set of supports such that \(x_i = v\) can be considered as a set of values (a subset of \(D(x_{i+1})\)). A solution of a constraint network is an instantiation of all the variables satisfying all the constraints.

Value \(v \in D(x_i), x_i \in X\), is (generalized) arc-consistent (GAC) with respect to \(C(X)\) if and only if \(v\) belongs to a support of \(C(X)\). A domain \(D(x_i), x_i \in X\), is GAC with respect to \(C(X)\) if and only if \(\forall v \in D(x_i), v\) is GAC with
respect to $C(X)$. $C(X)$ is GAC if and only if $\forall x_i \in X$, $D(x_i)$ is GAC with respect to $C(X)$. A constraint network is GAC if and only if it is closed for GAC [Bessière, 2006]: $\forall x_i \in X$ all values in $D(x_i)$ that are not GAC with respect to a constraint in $C$ have been removed.

3 The seq_bin Meta-Constraint

We first generalize the notion of stretches [Pesant, 2001] to characterize a sequence of consecutive variables where the same binary constraint is satisfied.

Definition 1 (C-stretch). Let $I[X]$ be an instantiation of the variable sequence $X = [x_0, x_1, \ldots, x_{n-1}]$ and $C$ a binary constraint. The $C$-sequence constraint $C(I[X], C)$ holds if and only if:

- Either $n = 1$,
- or $n > 1$ and $\forall k \in [0, n - 2] C(I[x_k], I[x_{k+1}])$ holds.

A C-stretch of $I[X]$ is a subsequence $X' \subseteq X$ such that the two following conditions are both satisfied:

1. The C-sequence $C(I[X'], C)$ holds,
2. $\forall X''$ such that $X' \subseteq X'' \subseteq X$ the C-sequence $C(I[X''], C)$ does not hold.

The intuition behind Definition 1 is to consider the maximum length subsequences where the binary constraint $C$ is satisfied between consecutive variables. Thanks to this generalized definition of stretches we can now introduce seq_bin.

Definition 2. The meta-constraint seq_bin$(N, X, C, B)$ is defined by a variable $N$, a sequence of $n$ variables $X = [x_0, x_1, \ldots, x_{n-1}]$ and two binary constraints $C$ and $B$. Given an instantiation $I[N, x_0, x_1, \ldots, x_{n-1}]$, seq_bin$(N, X, C, B)$ is satisfied if and only if for any $i \in [0, n - 2]$, $I[x_i] B I[x_{i+1}]$ holds, and $I[N]$ is equal to the number of C-stretches in $I[X]$.

The constraint CHANGE was introduced in the context of timetabling problems [Cossette, 1997], in order to put an upper limit on the number of changes of job types during a given period. The relation between classical stretches and CHANGE was initially stressed in [Hellsten, 2004, page 64]. CHANGE is defined on a variable $N$, a sequence of variables $X = [x_0, x_1, \ldots, x_{n-1}]$, and a binary constraint $C \in \{=, \neq, <, >, \leq, \geq\}$. It is satisfied if and only if $N$ is equal to the number of times the constraint $C$ holds on consecutive variables of $X$. Without hindering propagation (the constraint network is Berge-acyclic), CHANGE can be reformulated as seq_bin$(N', X, C, \text{true}) \land [N = N' - 1]$, where true is the universal constraint.

Smooth$(N, X)$ is a variant of CHANGE$(N, X, C)$, where $x_i C x_{i+1}$ is defined by $|x_i - x_{i+1}| > \text{csl}$, csl $\in \mathbb{N}$. It is useful to limit the number of drastic variations on a cumulative profile [Beldiceanu et al., 2010a; De Clercq, 2010].

As a last example, consider the Increasing_NVALUE constraint, which is a specialized version of NVALUE [Pachet and Roy, 1999]. It was introduced for breaking variable symmetry in the context of resource allocation problems [Beldiceanu et al., 2010b]. Increasing_NVALUE is defined on a variable $N$ and on a sequence of variables $X = [x_0, x_1, \ldots, x_{n-1}]$. Given an instantiation, Increasing_NVALUE$(N, X)$ is satisfied if and only if $N$ is equal to the number of distinct values assigned to variables in $X$, and for any $i \in [0, n - 2]$, $x_i \leq x_{i+1}$. We reformulate Increasing_NVALUE$(N, X)$ as seq_bin$(N, X, =, \leq)$.

4 Consistency of seq_bin

We first present how to compute, for any value in a given domain of a variable $x_i \in X$, the minimum and maximum number of C-stretches within the suffix of $X$ starting at $x_i$ (resp. the prefix of $X$ ending at $x_i$) satisfying a chain of binary constraints of type $B$. Then, we introduce several properties useful to obtain a feasibility condition for seq_bin, and a necessary and sufficient condition for filtering which leads to the GAC filtering algorithm presented in Section 5.

4.1 Computing of the Number of C-stretches

According to Definition 2, we have to ensure that the chain of $B$ constraints are satisfied along the sequence of variables $X = [x_0, x_1, \ldots, x_{n-1}]$. An instantiation $I[X]$ is said B-coherent if and only if $\forall n \in [0, n - 2]$, $I[x_i] B I[x_{i+1}]$. A value $v \in D(x_i)$ is said to be B-coherent with respect to $x_i$ if and only if it can be part of at least one B-coherent instantiation. Then, given an integer $i \in [0, n - 2]$, if $v \in D(x_i)$ is B-coherent with respect to $x_i$ there exists $w \in D(x_{i+1})$ such that $v B w$.

Consequently, within a given domain $D(x_i)$, values that are not B-coherent can be removed since they cannot be part of any solution of seq_bin. Our aim is now to compute for each B-coherent value $v$ in the domain of any variable $x_i$ the minimum and maximum number of C-stretches on $X$.

Notation 1. $\underline{g}(x_i, v)$ (resp. $\overline{g}(x_i, v)$) is the minimum (resp. maximum) number of C-stretches within the sequence of variables $[x_i, x_{i+1}, \ldots, x_{n-1}]$ under the hypothesis that $x_i = v$. $p(x_i, v)$ (resp. $\overline{p}(x_i, v)$) is the minimum (resp. maximum) number of C-stretches within the sequence $[x_0, x_1, \ldots, x_i]$ under the hypothesis that $x_i = v$. Given $X = [x_0, x_1, \ldots, x_{n-1}]$, $\underline{g}(X)$ (resp. $\overline{g}(X)$) denotes the minimum (resp. maximum) value of $\underline{g}(x_0, v)$ (resp. $\overline{g}(x_0, v)$).

Lemma 1. Given seq_bin$(N, X, C, B)$ with $X = [x_0, x_1, \ldots, x_{n-1}]$, assume the domains in $X$ contain only B-coherent values. Given $i \in [0, n - 1]$ and $v \in D(x_i)$,

- If $i = n - 1$: $\underline{g}(x_n, v) = 1$.
- Else:

$$\underline{g}(x_i, v) = \min_{w \in D(x_{i+1})} \left( \min_{[v B w] \land [w C w]} \underline{g}(x_{i+1}, w) \right)$$

Proof. By induction. From Definition 1, for any $v \in D(x_{n-1})$, we have $\underline{g}(x_{n-1}, v) = 1$ (i.e., a C-stretch of length 1). Consider now $x_i \in X$ with $i < n - 1$, and a value $v \in D(x_i)$. Consider the set of instantiations $I[x_{i+1}, x_{i+2}, \ldots, x_{n-1}]$ that are B-coherent, and that minimize the number of C-stretches in $[x_{i+1}, x_{i+2}, \ldots, x_{n-1}]$. We denote this minimum number of C-stretches by $\text{mins}$. At least one B-coherent instantiation exists since all values in the domains of $[x_{i+1}, x_{i+2}, \ldots, x_{n-1}]$ are B-coherent. For
each such instantiation, let us denote by $w$ the value associated with $I[x_{i+1}]$. Either there exists such an instantiation with $\text{mins}$ $C$-stretches with the conjunction $B \land C$ satisfied by $(I[x_i], I[x_{i+1}])$. Then, $s(x_i, v) = s(x_{i+1}, w)$ since the first $C$-stretch of $I[x_{i+1}, x_{i+2}, \ldots, x_{n-1}]$ is extended when augmenting $I[x_{i+1}, x_{i+2}, \ldots, x_{n-1}]$ with value $v$ for $x_i$. Or all instantiations $I[x_{i+1}, x_{i+2}, \ldots, x_{n-1}]$ with $\text{mins}$ $C$-stretches are such that $C$ is violated by $(I[x_i], I[x_{i+1}])$: $(I[x_i], I[x_{i+1}])$ satisfies $B \land \neg C$. By construction, any instantiation $I[x_{i+1}, x_{i+2}, \ldots, x_{n-1}]$ with $\text{mins}$ $C$-stretches has a number of $C$-stretches strictly greater than $\text{mins}$. Consequently, given $I[x_{i+1}, x_{i+2}, \ldots, x_{n-1}]$ with $\text{mins}$ $C$-stretches, the number of $C$-stretches obtained by augmenting this instantiation with value $v$ for $x_i$ is exactly $\text{mins} + 1$.

Lemma 2. Given $\text{SEQ\_BIN}(N, X, C, B)$ with $X = \{x_0, x_1, \ldots, x_n\}$, assume the domains in $X$ contain only $B$-coherent values. Given $i \in [0, n-1]$ and $v \in D(x_i)$:

- If $i = n - 1$:
  
  $$\pi(x_i, v) = \pi(x_{i+1}, v) = 1.$$ 

- Else:
  
  $$\pi(x_i, v) = \max_{w \in D(x_i)} \left( \frac{\max_{v \in B \cap C} \left( \pi(x_{i+1}, w) \right)}{\max_{v \in B \cap \neg C} \left( \pi(x_{i+1}, w) \right) + 1} \right)$$

Given a sequence of variables $[x_0, x_1, \ldots, x_{n-1}]$ such that their domains contain only $B$-coherent values, for any $x_i$ in the sequence and any $v \in D(x_i)$, computing $\pi(x_i, v)$ (resp. $\pi(x_i, v)$) is symmetrical to $s(x_i, v)$ (resp. $\pi(x_i, v)$). We substitute $w$ with max (resp. max) $x_{i+1}$ by $x_{i-1}$, and $v \in B \cap C$ for feasibility.

4.2 Properties on the Number of $C$-stretches

This section provides the properties linking the values in a domain $D(x_i)$ with the minimum and maximum number of $C$-stretches in $X$. We consider only $B$-coherent values, which may be part of a feasible instantiation of $\text{SEQ\_BIN}$. Next property is a direct consequence of Lemmas 1 and 2.

Property 1. For any $B$-coherent value $v$ in $D(x_i)$, with respect to $x_i$, $s(x_i, v) \leq \pi(x_{i+1}, v)$.

Property 2. Consider $\text{SEQ\_BIN}(N, X, C, B)$, a variable $x_i \in X$ ($0 \leq i \leq n - 1$), and two $B$-coherent values $v_1, v_2 \in D(x_i)$. If $i = n - 1$ or if there exists a $B$-coherent $w \in D(x_{i+1})$ such that $v_1 B w$ and $v_2 B w$, then $\pi(x_i, v_1) + 1 \geq \pi(x_i, v_2)$.

Proof. Obviously, if $i = n - 1$. If $v_1 = v_2$, by Property 1 the property holds. Otherwise, there exist two values $v_1$ and $v_2$ such that $3wD(x_{i+1})$ for which $v_1 B w$ and $v_2 B w$, and $\pi(x_i, v_1) + 1 < \pi(x_i, v_2)$. By Lemma 2, $\pi(x_i, v_1) \geq \pi(x_{i+1}, w)$. By Lemma 1, $s(x_i, v_2) \leq s(x_{i+1}, w) + 1$. From hypothesis $H$, this entails $\pi(x_i, v_2) \leq \pi(x_{i+1}, w) + 1$, which leads to $\pi(x_{i+1}, w) < \pi(x_i, v_1)$, which is, by Property 1, not possible.

Property 3. Consider $\text{SEQ\_BIN}(N, X, C, B)$, a variable $x_i \in X$ ($0 \leq i \leq n - 1$), and two $B$-coherent values $v_1, v_2 \in D(x_i)$. If either $i = n - 1$ or there exists $B$-coherent $w \in D(x_{i+1})$ such that $v_1 B w$ and $v_2 B w$, then for any $k \in [\min(s(x_i, v_1), s(x_i, v_2)), \max(s(x_i, v_1), s(x_i, v_2))]$, there is $k \in s(x_i, v_1), \pi(x_i, v_1)$ or $k \in s(x_i, v_2), \pi(x_i, v_2)$.

4.3 Properties on Binary Constraints

Property 3 is central for providing a GAC filtering algorithm based on the count, for each $B$-coherent value in a domain, of the minimum and maximum number of $C$-stretches in complete instantiations. Given $\text{SEQ\_BIN}(N, X, C, B)$, we focus on binary constraints $B$ which guarantee that Property 3 holds.

Definition 3. [Van Hentenryck et al., 1992] A binary constraint $F$ is monotonic if and only if there exists a total ordering $< \prec$ of values in domains such that for any value $v$ and any value $w$, $v F w$ holds implies $v F w'$ holds for all valid tuple such that $v' \prec v$ and $w' \prec w'$.

Binary constraints $<, >, \leq$ and $\geq$ are monotonic, as well as the universal constraint $\text{true}$.

Property 4. Consider $\text{SEQ\_BIN}(N, X, C, B)$ such that all non $B$-coherent values have been removed from domains of variables in $X$. $B$ is monotonic if and only if for any variable $x_i \in X$, $0 \leq i < n - 1$, for any values $v_1, v_2 \in D(x_i)$, there exists $w \in D(x_{i+1})$ such that $v_1 B w$ and $v_2 B w$.

Proof. ($\Rightarrow$) From Definition 3 and since we consider only $B$-coherent values, each value has at least one support on $B$. Moreover, from Definition 3, $\{w \mid v_1 B C w\} \subseteq \{w \mid v_1 C w\}$ or $\{w \mid v_1 C w\} \subseteq \{w \mid v_2 C w\}$. The property holds. ($\Leftarrow$) Suppose that the second proposition is true (hypothesis $H$) and $B$ is not monotonic. From Definition 3, if $B$ is not monotonic then $\exists v_1$ and $v_2$ in the domain of a variable $x_i \in X$ such that, by considering the constraint $B$ on the pair of variables $(x_i, x_{i+1})$, neither $\{w \mid v_1 B C w\} \subseteq \{w \mid v_1 C w\}$ nor $\{w \mid v_1 C w\} \subseteq \{w \mid v_2 C w\}$. Thus, there exists a support $v_1 B w$ such that $(v_2, w)$ is not a support on $B$, and a support $v_2 B w'$ such that $(v_1, w')$ is not a support on $B$. We can have $D(x_{i+1}) = \{w, w'\}$, which leads to a contradiction with $H$. The property holds.

4.4 Feasibility

From Property 4, this section provides an equivalence relation between the existence of a solution for $\text{SEQ\_BIN}$ and the current variable domains of $X$ and $N$. Without loss of generality, in this section we consider that all non $B$-coherent values have been removed from domains of variables in $X$. First, Definition 2 entails the following necessary condition for feasibility.

Proposition 1. Given $\text{SEQ\_BIN}(N, X, C, B)$, if $s(X) > \max(D(N))$ or $\pi(X) < \min(D(N))$ then $\text{SEQ\_BIN}$ fails.

$D(N)$ can be restricted to $[s(X), \pi(X)]$, but $D(N)$ may have holes or may be strictly included in $[s(X), \pi(X)]$. We have the following proposition.
Proposition 2. Consider $\text{SEQ}_\text{BIN}(N, X, C, B)$ such that $B$ is monotonically, with $X = \{x_0, x_1, \ldots, x_{n-1}\}$. For any integer $k$ in $[s(x_0, v), \pi(x_0, v)]$ there exists $v$ in $D(x_0)$ such that $k \in [s(x_0, v), \pi(x_0, v)]$. 

Proof. Let $v_1 \in D(x_0)$ a value such that $s(x_0, v_1) = s(X)$. Let $v_2 \in D(x_0)$ a value such that $\pi(x_0, v_2) = \pi(X)$. By Property 4, either $n = 1$ or $3w = D(x_1)$ such that $v_1 B v_2$ and $v_3 B w$. Thus, from Property 3, $\forall k \in [s(x_0, v_1), \pi(x_0, v_1)]$, either $k \in [s(x_0, v_2), \pi(x_0, v_2)]$ or $k \in [s(x_0, v_3), \pi(x_0, v_3)]$. }

By Proposition 2, any value for $N$ in $D(N) \cap [s(X), \pi(X)]$ is generalized arc-consistent provided a property is satisfied on the instance of $\text{SEQ}_\text{BIN}$ we consider: given a variable $x_i$, for any value $v$ in $D(x_i)$ and for all $k \in [s(x_i, v), \pi(x_i, v)]$, there exists a solution of $\text{SEQ}_\text{BIN}(N, X, C, B)$ with exactly $k$ $C$-stretches.

Definition 4. The constraint $\text{SEQ}_\text{BIN}(N, X, C, B)$ is counting-continuous if and only if for any instantiation $I[X]$ with $k$ $C$-stretches, for any variable $x_i \in X$, changing the value of $x_i$ in $I[X]$ leads to a number of $C$-stretches equal to $k$, or $k + 1$, or to $k - 1$.

Property 5. Consider $\text{SEQ}_\text{BIN}(N, X, C, B)$ such that $B$ is monotonically, with $X = \{x_0, x_1, \ldots, x_{n-1}\}$, $i$ a variable and $v \in D(x_i)$. If $\text{SEQ}_\text{BIN}(N, X, C, B)$ is counting-continuous, then there exists for any integer $k \in [s(x_i, v), \pi(x_i, v)]$ an instantiation $I[\{x_1, \ldots, x_{n-1}\}]$ with exactly $k$ $C$-stretches.

Proof. By recurrence, we assume that the property is true for all instantiations of $\{x_j, \ldots, x_{n-1}\}$ such that $j > i$ (the property is obviously true if $j = n - 1$). At step $i$, we assume that there exists $k \in [s(x_i, v), \pi(x_i, v)]$, and as such that there is no instantiation $I[\{x_1, \ldots, x_{n-1}\}]$ with $k$ $C$-stretches, while $\text{SEQ}_\text{BIN}(N, X, C, B)$ is counting-continuous (hypothesis). We prove that this assumption leads to a contradiction. By Lemmas 1 and 2, there exists an instantiation $I'[\{x_1, \ldots, x_{n-1}\}]$ with $s(x_i, v)$ $C$-stretches and $I'[x_i] = v$, and there exists an instantiation $I''[\{x_1, \ldots, x_{n-1}\}]$ with $\pi(x_i, v)$ $C$-stretches and $I''[x_i] = v$. Thus, by hypothesis, $k > s(x_i, v)$ and $k < \pi(x_i, v)$. We have $\pi(x_i, v) \geq s(x_i, v) + 2$. By Property 3 and since the property is assumed true for all instantiations of $\{x_1, \ldots, x_{n-1}\}$, there exists at least one pair of values $w_1, w_2$ in $D(x_{i+1})$ such that $s(x_{i+1}, w_1) = \pi(x_{i+1}, w_1) + 2$, $(v, w_1)$ satisfies $C$ and $(v, w_2)$ violates $C$ (this is the only possible configuration leading to the hypothesis). In this case $\text{SEQ}_\text{BIN}(N, X, C, B)$ is not counting-continuous: Given an instantiation $I[x_1, \ldots, x_{n-1}]$ with $I[x_i] = v$ and $I[x_{i+1}] = w_1$, changing $w_1$ by $w_2$ for $x_{i+1}$ increases the number of $C$-stretches by 2.

Proposition 3. Given an instance of $\text{SEQ}_\text{BIN}(N, X, C, B)$ which is counting-continuous and such that $B$ is monotonically, $\text{SEQ}_\text{BIN}(N, X, C, B)$ has a solution if and only if $[s(X), \pi(X)] \cap D(N) \neq \emptyset$.

Proof. ($\Rightarrow$) Assume $\text{SEQ}_\text{BIN}(N, X, C, B)$ has a solution. Let $I[\{N\} \cup X]$ be such a solution. By Lemmas 1 and 2, the number of $C$-stretches $I[N]$ belongs to $[s(X), \pi(X)]$. ($\Leftarrow$)

Let $k \in [s(X), \pi(X)] \cap D(N)$ (not empty). From Proposition 2, for any value $k$ in $[s(X), \pi(X)]$, $\exists v \in D(x_0)$ such that $k \in [s(x_0, v), \pi(x_0, v)]$. Since $\text{SEQ}_\text{BIN}(N, X, C, B)$ is counting-continuous, there exists an instantiation of $X$ with $k$ $C$-stretches. By Definition 2 and since Lemmas 1 and 2 consider only $B$-coherent values, there is a solution of $\text{SEQ}_\text{BIN}(N, X, C, B)$ with $k$ $C$-stretches.

4.5 Necessary and Sufficient Filtering Condition

Given $\text{SEQ}_\text{BIN}(N, X, C, B)$, Proposition 3 can be used to filter the variable $N$ from variables in $X$. Propositions 1 and 2 ensure that every remaining value in $[s(X), \pi(X)] \cap D(N)$ is involved in at least one solution satisfying $\text{SEQ}_\text{BIN}$. We consider now the filtering of variables in $X$.

Proposition 4. Given an instance of $\text{SEQ}_\text{BIN}(N, X, C, B)$ which is counting-continuous and such that $B$ is monotonically, let $v$ be a value in $D(x_i)$, $i \in [0, n - 1]$. The two following propositions are equivalent:

1. $v$ is $B$-coherent and $v$ is GAC with respect to $\text{SEQ}_\text{BIN}$

2. \[
D(v) = \left\{ p(x_i, v) + s(x_i, v) - 1, \pi(x_i, v) + s(x_i, v) - 1 \right\} \cap D(N) \neq \emptyset
\]

Proof. If $v$ is not $B$-coherent then, by Definition 2, $v$ is not GAC. Otherwise, $p(x_i, v)$ (resp. $s(x_i, v)$) is the exact minimum number of $C$-stretches among $B$-coherent instantiations $I[x_0, x_1, \ldots, x_r]$ (resp. $I[x_1, x_2, \ldots, x_{n-1}]$) such that $I[x_i] = v$. Thus, by Lemma 1 (and its symmetrical for prefixes), the exact minimum number of $C$-stretches among $B$-coherent instantiations $I[x_0, x_1, \ldots, x_r]$ such that $I[x_i] = v = p(x_i, v) + s(x_i, v) - 1$. Let $D(v) \subseteq D$ such that all domains in $D(v)$ are equal to domains in $D$ except $D(x_i)$ which is reduced to $\{v\}$. We call $X_{(v)}$ the sequence of variables associated with domains in $D(v)$. By construction $p(x_i, v) + s(x_i, v) - 1 = s(X_{(v)})$. By a symmetrical reasoning, $\pi(x_i, v) + s(x_i, v) - 1 = \pi(X_{(v)})$. By Proposition 3, the proposition holds.

The “− 1” in expressions $p(x_i, v) + s(x_i, v) - 1$ and $\pi(x_i, v) + s(x_i, v) - 1$ prevents us from counting twice a $C$-stretch at an extremity $x_i$ of the two sequences $\{x_0, x_1, \ldots, x_r\}$ and $\{x_1, x_2, \ldots, x_{n-1}\}$.

5 GAC Filtering Algorithm

Based on the necessary and sufficient filtering condition of Proposition 4, this section provides an implementation of the GAC filtering algorithm for a counting-continuous instance of $\text{SEQ}_\text{BIN}(N, X, C, B)$ with a monotonic constraint $B$.

If $B \notin \{\leq, \geq, <, >, \text{true}\}$ then the total ordering entails monotonicity of $B$ is not the natural order of integers. In this case, if $<$ is not known, it is necessary to compute such an ordering with respect to all values in $\cup_{i=0}^{n-1}(D(x_i))$, once before the first propagation of $\text{SEQ}_\text{BIN}$. Consider that the two variables of $B$ can take any value in $\cup_{i=0}^{n-1}(D(x_i))$: Due to the inclusion of sets of supports of values (see Definition 3), the order remains the same when the domains of the variables constrained by $B$ do not contain all values in $\cup_{i=0}^{n-1}(D(x_i))$.

To compute $<$, the following procedure can be used: Count the number of supports of each value, in $O(d^2)$ time
coherent values can be removed from domains in \(O\) the values in domains the quantities are stored, phase domain. Since all the variables need to be scanned and for all of \(w\), according to \(D\) decomposed into four phases:

1. Remove all non-\(B\)-coherent values in the domains of \(X\).
2. For all values in the domains of \(X\), compute the minimum and maximum number of \(C\)-stretches of prefixes and suffixes.
3. Adjust the minimum and maximum value of \(N\) with respect to the minimum and maximum number of \(C\)-stretches of \(X\).
4. Using the result phase 2 and Proposition 4, prune the remaining \(B\)-coherent values.

With respect to phase 1, recall that \(B\) is monotonic: According to \(<\), for any pair of variables \((x_i, x_{i+1})\), \(\forall v_0\) in \(D(x_i)\) such that \(\forall v_j \in D(x_i), v_j \neq v_0, v_j\) has a set of supports on \(B(x_i, x_{i+1})\) included in the supports of \(v_0\) on \(B(x_i, x_{i+1})\). By removing from \(D(x_i)\) non supports of \(v_0\) on \(B(x_i, x_{i+1})\) in \(O(\|D(x_i)\|)\), all non-\(B\)-coherent values of \(D(x_i)\) with respect to \(B(x_i, x_{i+1})\) are removed. By repeating such a process in the two directions (starting from the pair \((x_{n-2}, x_{n-1})\) and from the pair \((x_0, x_1)\)), all non-\(B\)-coherent values can be removed from domains in \(O(\Sigma_{D})\) time complexity.

To achieve phase 2 we use Lemmas 1 and 2 and their symmetrical formulations for prefixes. Without loss of generality, we focus on the minimum number of \(C\)-stretches of a value \(v_j\) in the domain of a variable \(x_i, i < n - 1\), thanks to Lemma 1. Assume that for all \(w \in D(x_{i+1}), s(x_{i+1}, w)\) has been computed. If there is no particular property on \(C\), the supports \(S_j \in D(x_{i+1})\) of \(v_j\) on \(C(x_i, x_{i+1})\) and the subset \(\neg S_j \in D(x_{i+1})\) of non-supports of \(v_j\) on \(C(x_i, x_{i+1})\) which satisfy \(B\) have to be scanned, in order to determine for each set a value \(w \in S_j\) minimizing \(s(x_{i+1}, w)\) and a value \(w' \in \neg S_j\), minimizing \(s(x_{i+1}, w')\) + 1. This process takes \(O(\|D(x_{i+1})\|)\) for each value, leading to \(O(d^2)\) for the whole domain. Since all the variables need to be scanned and for all the values in domains the quantities are stored, phase 2 takes \(O(n d^2)\) in time, and \(O(\Sigma_{D})\) in space.

Phases 3 and 4 take \(O(\Sigma_{D})\) time each since all the domains have to be scanned. By Proposition 4, all the non-GAC values have been removed after this last phase.

If \(B \in \{\leq, \geq, <, >, \text{true}\}\), \(<\) is known. The worst-case time and space results come from Phase 2. The bottleneck stems from the fact that, when a domain \(D(x_i)\) is scanned, the minimum and maximum number of \(C\)-stretches of each value are computed from scratch, while an incremental computation would avoid to scan \(D(x_{i+1})\) for each value in \(D(x_i)\).

This observation leads to Property 6. Again, we focus on the minimum number of \(C\)-stretches on suffixes. Other cases are symmetrical.

**Notation 2.** Given \(s_{\text{seq.bin}}(N, X, C, B)\), \(x_i \in X, 0 \leq i < n\) and a value \(v_j \in D(x_i)\), if \(i < n - 1\), let \(V_j\) denote the set of integer values such that a value \(s(v_j, w) \in V_j\) corresponds to each \(w \in D(x_{i+1})\) and is equal to:

- \(s(x_{i+1}, w)\) if and only if \(w \in S_j\)
- \(s(x_{i+1}, w) + 1\) if and only if \(w \in \neg S_j\)

Within notation 2, the set \(V_j\) corresponds to the minimum number of stretches of values in \(D(x_{i+1})\) increased by one if they are non supports of value \(v_j\) with respect to \(C\).

**Property 6.** Given a counting-continuous instance of \(s_{\text{seq.bin}}(N, X, C, B)\) such that \(B \in \{\leq, \geq, <, >, \text{true}\}\) and \(x_i \in X, 0 \leq i < n - 1\), if the computation of \(\min_{w \in D(x_i)}(s(v_j, w))\) for all \(v_j \in D(x_i)\) can be performed in \(O(\|D(x_{i+1})\|)\) then \(\text{GAC can be achieved on } s_{\text{seq.bin}}\) in \(O(\Sigma_{D})\) time and space complexity.

**Proof.** Applying Lemma 1 to the whole domain \(D(x_i)\) takes \(O(\|D(x_{i+1})\|)\) time. Storing the minimum number of stretches for each value in \(D(x_i)\) requires \(O(\|D(x_i)\|)\) space. Phase 2 takes \(O(\Sigma_{D})\) space and \(O(\Sigma_{D})\) time.

When they are represented by a counting-continuous instance of \(s_{\text{seq.bin}}\), the practical constraints mentioned in the introduction satisfy a condition that entails Property 6: Given \(x_i\), it is possible to compute in \(O(\|D(x_{i+1})\|)\) the quantity \(\min_{w \in D(x_{i+1})}(s(v_0, w))\) for the first value \(v_0 \in D(x_i)\) and then, following the natural order of integers, to derive with a constant or amortized time complexity the quantity for the next value \(v_1\), and then the quantity for the next value \(v_2\), and so on. Thus, to obtain \(\text{GAC in } O(\Sigma_{D})\) for all these constraints, we specialize Phase 2 in order to exploit such a property. We now detail how to proceed.

With respect to the constraints mentioned in the introduction corresponding to instances of \(s_{\text{seq.bin}}\) which are not counting-continuous, the same time complexity can be reached but the algorithm does not enforce \(\text{GAC}\).

Thus, when \(s_{\text{seq.bin}}\) represents \(\text{CHANGE, SMOOTH or INCREASING}_N^{\text{VALUE}}\), computing \(\min_{w \in D(x)}(s(v_0, w))\) for the minimum value \(v_0 = \min(D(x_i))\) (respectively the maximum value) can be performed by scanning the minimum number of \(C\)-stretches of values in \(D(x_{i+1})\).

We now study for \(\text{CHANGE, SMOOTH and INCREASING}_N^{\text{VALUE}}\) how to efficiently compute the value \(\min_{w \in D(x)}(s(v_0, w))\) of \(v_k \in D(x_i)\), either directly or from the previous value \(\min_{w \in D(x)}(s(v_{k-1}, w))\), in order to compute \(\min_{w \in D}(s(v_j, w))\) for all \(v_j \in D(x_i)\) in \(O(\|D(x_i)\|)\) time and therefore achieve Phase 2 in \(O(\Sigma_{D})\).

### The Change Constraint

Section 3 showed a reformulation of \(\text{CHANGE}(N, X, C, \text{CTR})\) as \(s_{\text{seq.bin}}(N', X, C, \text{true}) \land [N = N' - 1]\), where \(C\) is the opposite of \(\text{CTR}\).

If \(C\) is ‘\(\text{true}\)’ (the principle is similar for ‘\(<\)’ and ‘\(\leq\)’), the instance of \(s_{\text{seq.bin}}\) is counting-continuous, because \(B\) is \(\text{true}\) and \(C\) is monotonic. The monotonicity of \(C\), with its corresponding total ordering ‘\(<\)’, guarantees that given three consecutive variables \(x_{i-1}, x_i, x_{i+1}\) and \(v_1 \in D(x_{i-1}), v_2 \in D(x_i), v_3 \in D(x_{i+1})\), if \((v_1, v_2)\) and \((v_2, v_3)\) both violate \(C\), then we necessarily have \(v_1 \succ v_2 \succ v_3\). Therefore, changing
value $v_2$ by a new value $v'_2$ such that $v_1 < v'_2$ (to satisfy $C$) entails $v'_2 > v_2$, and thus still $v'_2 > v_3$ (which violates $C$). It is not possible to remove (or, symmetrically, to add) two violations of $C$ within an instantiation only by changing the value of one variable. The instance of seqbin is counting-continuous and thus the algorithm enforces GAC (by Proposition 4).

To achieve step 3, in $O(D(x_i))$, we introduce two quantities $lt(v_j, x_{i+1})$ and $geq(v_j, x_{i+1})$ respectively equal to $\min_{w \in [\min(D(x_i)), v_j]}(\lambda(x_{i+1}, w))$ and $\min_{w \in [v_j, \max(D(x_i))]}(\lambda(x_{i+1}, w))$. The computation is performed in three steps:

1. Starting from $v_0 = \min(D(x_i))$, that is, the value having the smallest number of supports for $C$ on $x_{i+1}$, compute $lt(v_j, x_{i+1})$ in increasing order of $v_j$. Taking advantage that, given a value $v_{j-1} \in D(x_i)$ and the next value $v_j \in D(x_i)$, $\min(D(x_i))$, $v_{j-1}$ is included in $\min(lt(v_j, x_{i+1}))$. Therefore, the computation of all $\min_{w \in [\min(D(x_i)), v_j]}(\lambda(x_{i+1}, w))$ can be amortized over $D(x_{i+1})$. The time complexity for computing $lt(v_j, x_{i+1})$ for all $v_j \in D(x_i)$ is in $O(|D(x_i)| + |D(x_{i+1})|).

2. Similarly starting from $v_0 = \max(D(x_i))$, compute incrementally $geq(v_j, x_{i+1})$ in decreasing order of $v_j$, in $O(|D(x_i)| + |D(x_{i+1})|).

3. Finally, for each $v_j \in D(x_i)$, $\min_{w \in D(x_{i+1})}(s(v_j, w))$ is equal to $\min(lt(v_j, x_{i+1}), geq(v_j, x_{i+1}) + 1)$.

Since step 3. takes $O(D(x_i))$, we get an overall time complexity for Phase 2 in $O(\Sigma D_i)$.

- If $C$ is ‘=’, ‘≠’, or $|x_i - x_{i+1}| \leq cst$ (the latter case corresponds to the Smooth constraint), the filtering algorithm does not guarantees GAC because the corresponding instances of seqbin are not counting-continuous. Step 3. can also be performed in $O(D(x_i))$, leading to an overall time complexity for Phase 2 in $O(\Sigma D_i)$:
  - If $C$ is ‘=’ then for each $v_j \in D(x_i)$ there is a unique potential support for $C$ on $x_{i+1}$, the value $v_j$. Therefore, by memorizing once the value $\text{vmin}_1$ in $D(x_i)$ which corresponds to the smallest minimum numbers of $C$-stretches on the suffix starting at $x_{i+1}$. \( \forall v_j, \min_{w \in D(x_{i+1})}(s(v_j, w)) = \min(\lambda(x_{i+1}, v_j), \lambda(x_{i+1}, \text{vmin}_1) + 1) \), assuming $\lambda(x_{i+1}, v_j) = +\infty$ when $v_j \notin D(x_{i+1})$.
  - If $C$ is ‘≠’ then for each $v_j \in D(x_i)$ there is a single non support. By memorizing the two values $\text{vmin}_1$ and $\text{vmin}_2$ which minimize the minimum numbers of $C$-stretches on the suffix starting at $x_{i+1}$, for any value $v_{j-1} \in D(x_i)$, $\min_{w \in D(x_{i+1})}(s(v_{j-1}, w))$ is equal to: $\min(\lambda(x_{i+1}, \text{vmin}_1) + 1, \lambda(x_{i+1}, \text{vmin}_2))$ when $\text{vmin}_1 = v_j$, and $\lambda(x_{i+1}, v_j)$ otherwise.
- Smooth is a variant of Change(N, X, CTR), where $x_i$ CTR $x_{i+1}$ is $|x_i - x_{i+1}| > cst$, $cst \in \mathbb{N}$, that can be reformulated as seqbin(N, X, C, true) $\wedge$ [N $\neq N - 1$, where $C$ is $|x_i - x_{i+1}| \leq cst$. Assume $v_0 = \min(D(x_i))$ and we scan values in increasing order. Supports of values in $D(x_i)$ for $|x_i - x_{i+1}| \leq cst$ define a set of sliding windows for which both the starts and the ends are increasing sequences (not necessarily strictly). Thus, $\min_{w \in D(x_{i+1})}(s(v_j, w))$ can be computed for all $v_j \in D(x_i)$ in $O(|D(x_i)|)$ thanks to the ascending minima algorithm.\(^1\) Given a value $v_j \in D(x_i)$ the set $\sim S_j$ of non supports of $v_j$ on $|x_i - x_{i+1}| \leq cst$ is partitioned in two sequences of values: a first sequence before the smallest support and a second sequence after the largest support. While scanning values in $D(x_i)$ these two sequences correspond also to sliding windows on which the ascending minima algorithm can also be used.

The IncreasingValue constraint

It is represented by $\text{seq-bin}(N, X, =, \leq)$, which is counting-continuous (see [Beldiceanu et al., 2010b] for more details). The algorithm enforces GAC. Since $B$ is not true, we have to take into account $B$ when evaluating $\min_{w \in D(x_{i+1})}(s(j, w))$ for each $v_j \in D(x_i)$. Fortunately, we can start from $v_0 = \max(D(x_i))$ and consider the decreasing order since $B$ is ‘<’. In this case the set of supports on $B$ can only increase as we scan $D(x_i)$. $C$ is ‘=’, then for each $v_j \in D(x_i)$ there is a unique potential support for $C$ on $x_{i+1}$, the value $v_j$. We memorize once the value $\text{vmin}_1$ in $D(x_{i+1})$ which corresponds to the smallest minimum numbers of $C$-stretches on the suffix starting at $x_{i+1}$, only on supports of the current value $v_j \in D(x_i)$ on $B$. \( \forall v_j, \min_{w \in D(x_{i+1})}(s(v_j, w)) = \min(\lambda(x_{i+1}, v_j), \lambda(x_{i+1}, \text{vmin}_1) + 1) \), assuming $\lambda(x_{i+1}, v_j) = +\infty$ when $v_j \notin D(x_{i+1})$. Since the set of supports on $B$ only increases, $\text{vmin}_1$ can be updated for each new value in $D(x_i)$ in $O(1)$.

6 Related Work

Using automata, Change can be represented either by Regular [Pesant, 2004] or by Cost-Regular [Demassey et al., 2006]. In the first case this leads to a GAC algorithm in $O(n^2 \cdot d^2)$ time [Beldiceanu et al., 2010a, pages 584–585, 1544–1545] (where $d$ denotes the maximum domain size). In the second case the filtering algorithm of Cost-Regular does not achieve GAC.

Bessière et al. [Bessière et al., 2008] presented an encoding of the CardPath constraint with SLIDE2. A similar reformulation can be used for encoding seqbin(N, X, C, B). Recall that SLIDE2(C, [x_0, x_1, . . . , x_{n-1}]) holds if and only if $C(x_{ij}, . . . , x_{ij+k-1})$ holds for $0 \leq i \leq \frac{n}{k}$. Following a schema similar to the one proposed in Section 4 of Bessière et al. paper, seqbin(N, X, C, B) can be represented by adding a variable $N'$ and $n$ variables $[M_0, . . . , M_{n-1}]$, with $M_0 = 0$ and $M_{n-1} = N'$. seqbin(N, X, C, B) is then reformulated by SLIDE2($C'(M_0, x_0, M_1, x_1, . . . , M_{n-1}, x_{n-1})$ $\wedge$ $N' = N - 1$), where $C' = [-C(x_{i+1}, x_{i+1}] \wedge B(x_i, x_{i+1}] \wedge M_{i+1} = M_i + 1] \vee [C(x_i, x_{i+1}] \wedge B(x_i, x_{i+1}] \wedge M_{i+1} = M_i$. According to Section 6 of Bessière et al. paper, GAC can be achieved thanks to a reformulation of SLIDE2, provided a complete propagation is performed on $C'$, which is the case because $B(x_i, x_{i+1})$ and $C(x_i, x_{i+1})$ involve the same variables. The reformulation requires $n$ additional intersection

\(^1\)See http://home.tiac.net/~crt/2001/slidingmin.html
variables (one by sub-sequence \([M_i, x_i]\)), on which \(O(n)\) compatibility constraints between pairs of intersection variables and \(O(n)\) functional channelling constraints should hold. Arity of \(C'\) is \(k = 4\) and \(j = 2\): the domain of an intersection variable contains \(O(d^{k-j}) = O(d^2)\) values (corresponding to binary tuples), where \(d\) is the maximum size of a domain. Enforcing GAC on a compatibility constraint takes \(O(d^3)\) time, while functional channelling constraint take \(O(d^2)\), leading to an overall time complexity \(O(nd^3)\) for enforcing arc-consistency on the reformulation, corresponding to GAC for \(\text{seq-bin}\). To compare such a time complexity \(O(nd^3)\) with our algorithm, note that \(O(\sum D_i)\) is upper-bounded by \(O(nd)\).

At last, some ad hoc techniques can be compared to our generic GAC algorithm, e.g., a GAC algorithm in \(O(n^3m)\) for \text{change} [Hellsten, 2004, page 57], where \(m\) is the total number of values in the domains of \(X\). Moreover, the GAC algorithm for \(\text{seq-bin}\) generalizes to a class of counting constraints the ad-hoc GAC algorithm for \text{increasing}\_\text{value} [Beldiceanu et al., 2010b] without degrading time and space complexity in the case where \(\text{seq-bin}\) represents \text{increasing}\_\text{value}.

7 Conclusion

Our contribution is a structural characterization of a class of counting constraints for which we come up with a general polytime filtering algorithm achieving GAC under some conditions, and a characterization of the property which makes such an algorithm linear in the sum of domain sizes. A still open question is whether it would be possible or not to extend this class (e.g., considering \(n\)-ary constraints for \(B\) and \(C\)) without degrading complexity or giving up on GAC, in order to capture more constraints.

References

[Beldiceanu et al., 2010a] N. Beldiceanu, M. Carlsson, and J.-X. Rampon. Global Constraint Catalog, 2nd Ed. Technical Report T2010-07, SICS, 2010.

[Beldiceanu et al., 2010b] N. Beldiceanu, F. Hermenier, X. Lorca, and T. Petit. The \text{increasing value} Constraint. In CPAIOR, volume 6140 of LNCS, pages 25–39, 2010.

[Bessi`ere et al., 2008] C. Bessi`ere, E. Hebrard, B. Hnich, Z. Kızıltan, and T. Walsh. SLIDE: A useful special case of the CARDPATH constraint. In ECAI, 2008.

[Bessi`ere, 2006] C. Bessi`ere. Constraint Propagation. In F. Rossi, P. van Beek, and T. Walsh, editors, Handbook of Constraint Programming, chapter 3. Elsevier, 2006.

[Cosytec, 1997] Cosytec. CHIP Reference Manual, release 5.1 edition, 1997.

[De Clercq, 2010] A. De Clercq. A soft constraint for cumulative problems with over-loads of resource. In CP Doctoral Programme, pages 49–55, 2010.

[Demassey et al., 2006] S. Demassey, G. Pesant, and L.-M. Rousseau. A cost-regular based hybrid column generation approach. Constraints, 11(4):315–333, 2006.

[Hellsten, 2004] L. Hellsten. Consistency propagation for \text{stretch} constraints. Master’s thesis, Waterloo, 2004.

[Pachet and Roy, 1999] F. Pachet and P. Roy. Automatic generation of music programs. In CP, volume 1713 of LNCS, pages 331–345, 1999.

[Pesant, 2001] G. Pesant. A Filtering Algorithm for the \text{Stretch} Constraint. In CP, volume 2239 of LNCS, pages 183–195, 2001.

[Pesant, 2004] G. Pesant. A Regular Language Membership Constraint for Finite Sequences of Variables. In CP, volume 3258 of LNCS, pages 482–495, 2004.

[Van Hentenryck et al., 1992] P. Van Hentenryck, Y. Deville, and C.-M. Teng. A generic arc-consistency algorithm and its specializations. Artificial Intelligence, 57(2-3):291–321, 1992.