INEQUALITY-SUM: A GLOBAL CONSTRAINT CAPTURING THE OBJECTIVE FUNCTION

JEAN-CHARLES RÉGIN¹ AND MICHEL RUEHER²

Abstract. This paper introduces a new method to prune the domains of the variables in constrained optimization problems where the objective function is defined by a sum $y = \Sigma x_i$, and where the integer variables $x_i$ are subject to difference constraints of the form $x_j - x_i \leq c$. An important application area where such problems occur is deterministic scheduling with the mean flow time as optimality criteria. This new constraint is also more general than a sum constraint defined on a set of ordered variables. Classical approaches perform a local consistency filtering after each reduction of the bound of $y$. The drawback of these approaches comes from the fact that the constraints are handled independently. We introduce here a global constraint that enables to tackle simultaneously the whole constraint system, and thus, yields a more effective pruning of the domains of the $x_i$ when the bounds of $y$ are reduced. An efficient algorithm, derived from Dijkstra’s shortest path algorithm, is introduced to achieve interval consistency on this global constraint.

Mathematics Subject Classification. 65K05, 90C35.

1. INTRODUCTION

A great part of the success of constraint programming techniques in solving combinatorial problems is due to the capabilities of filtering algorithms to prune...
the search space. Roughly speaking, a filtering algorithm attempts to remove
values from the domains of all variables occurring in a constraint whenever the
domain of one of these variables is modified.

Arc consistency filtering algorithms on binary constraints are very popular but
significant gains in performance have also been obtained during recent years with
filtering algorithms associated with more complex constraints [13]. These new
filtering algorithms work on so-called “global constraints”, e.g., cumulative con-
straint [2], edge-finder algorithm [4], all-diff constraint [11], cardinality constraint
[7, 12]. They take into account the relations between the different occurrences of
the same variable in a given set of constraints.

In this paper, we introduce a new global constraint that can achieve significant
domain pruning in constrained optimization problems where the objective function
is defined by a sum $y = \Sigma x_i$, and where the integer variables $x_i$ are subject to
difference constraints of the form $x_j - x_i \leq c$. Two important applications where
such constraint systems occur are minimizing mean flow time and minimizing
tardiness in deterministic scheduling problems. The following presentation of these
applications is adapted from [3].

The mean flow time is defined by $F = \frac{1}{n} \sum_{j=1}^{n} (C_j - r_j)$ where $C_j$ and $r_j$ are
respectively the completion time and the ready time of task $J_j$. Difference con-
straints are due to the precedence constraints and the distances between the tasks
(and therefore between their completion times). The mean flow time criterion is
important from the user’s point of view since its minimization yields a minimiza-
tion of the mean response time and the mean in-process time of the scheduled
tasks set.

The mean tardiness is defined by $D = \frac{1}{n} \sum_{j=1}^{n} (D_j)$ where $D_j = \max(C_j - d_j, 0)$,
and where $d_j$ is the due date of task $J_j$. Minimizing this criteria is useful when
penalty functions are defined in accordance with due dates.

Both problems are $NP$-hard in most interesting cases [3, 6].

Another useful application of this constraint is a sum constraint defined on an
ordered set of variables.

Currently, in the constraint programming framework, such optimization prob-
lems are tackled by solving a sequence of decision problems: the solution of each
decision problem must not only satisfy the initial constraint system but it must
also provide a better bound than the best-known solution. In other words, each
new decision problem must satisfy an additional constraint specifying that the
value of $y$ is better than the current bound. To take advantage of this additional
constraint to cut the search space, and thus to avoid redoing almost always the
same work for each decision problem, we introduce here a new global constraint.

In the remainder of this section we first detail the motivation of our approach
before showing how it works on a short example.
1.1. Motivation

We consider the constrained optimization problem:

\[ \text{Minimize } f(x) \]
\[ \text{subject to } p_i(x) \leq 0 \quad (i = 1, \ldots, m) \]
\[ q_i(x) = 0 \quad (i = 1, \ldots, r) \]

where \( f \) is a scalar function of a vector \( x \) of \( n \) components, \( p_i(x) \) and \( q_i(x) \) are functions which may be non-linear. We assume that an initial box \( D \) is given (i.e., the domains of \( x \) are bounded) and we seek the global minimum of \( f(x) \) in \( D \). For the sake of simplicity, we also assume in the rest of the paper that \( f \) is a sum of the form \( y = \sum_{i=1}^{n} x_i \) (in Sect. 6, we discuss a sum of the form \( y = \sum_{i=1}^{n} a_i x_i \)). Unless otherwise mentioned, we assume in the rest of this paper that all the variables take integer values. Efficient filtering algorithms are available for sum constraints but in our case these algorithms are weakened by the fact that variables \( x_i \) involved in the objective function also occur in many other constraints. Among all these constraints, there is a subset of binary inequalities that only involve variables occurring in the objective function. Such inequalities may correspond to distance constraints as well as to constraints which have been introduced to break down symmetries of the problem to solve.

Note that the binary inequalities and the sum only model a sub-problem of a real application. Additional constraints are required to capture all the restrictions and features. So, what is needed is an efficient filtering algorithm for the conjunction of binary inequalities and the sum constraint.

Dechter et al. [5] have shown that shortest path algorithms can efficiently tackle such systems of inequalities in temporal constraint networks problems. The purpose of this paper is to introduce a new global constraint, named \( \text{IS} \), which handles as a single global constraint the sum constraint and a system of binary inequalities. An efficient algorithm – using a shortest path algorithm on a graph of reduced costs – is introduced to achieve interval consistency (see Def. 1) on this global constraint. Before going into the details, let us outline the advantages of this approach on a short example.

1.2. An illustrative example

Consider the constraint network \( \mathcal{C} = \{ C_1 : x_1 + x_2 = y, \quad C_2 : (x_1 \leq x_2 - 1) \} \) where \( D(x_1) = [0, 6], D(x_2) = [1, 7] \) and \( D(y) = [1, 13] \). Interval \([a, b]\) denotes the set of integer \( S = \{ k : a \leq k \wedge k \leq b \} \).

Constraint network \( \mathcal{C} \) is arc consistent. Now, suppose that \( \min(y) \) is set to 6. Arc consistency is unable to achieve any domain pruning. This is due to the fact that arc consistent filtering handles the constraints one by one. Now, let us examine what happens when constraints \( C_1 \) and \( C_2 \) are handled as a single constraint. To satisfy constraint \( C_2 \), the value of \( x_1 \) must be strictly less than the value of \( x_2 \), and thus, constraint \( C_1 \) cannot be satisfied when \( x_2 \) takes its values in \([1, 3] \). So, values \([1, 3] \) in \( D(x_2) \) can be deleted. On this example, a global handling of \( C_1 \) and \( C_2 \) drastically reduces the search space.
1.3. A brief summary of our framework

The filtering process on $C_1$ and $C_2$ is exactly what will be performed on global constraint $IS$. More precisely, let:

- a sum constraint $S_{um}$ defined by $y = \sum_{i=1}^{n} x_i$;
- a set of binary inequalities $I_{neq} = \{x_i - x_j \leq c_{ji}, \ (i, j \in [1, n])\}$;
- a set of domain constraints $D_{om} = \{l_i \leq x_i \leq u_i, \ (i \in [1, n])\}$.

The global constraint $IS$ is defined by $I_{neq} \cup D_{om} \cup \{S_{um}\}$. Each time a bound of $y$ is modified, the filtering on $IS$ starts by performing the following operations:

1. filtering $\{I_{neq} \cup D_{om}\}$ by interval consistency;
2. filtering $S_{um}$ by interval consistency;
3. updating the bounds of every $x_i$ with respect to constraint set $I_{neq} \cup D_{om} \cup \{S_{um}\}$.

Step 1 can be achieved with a shortest path algorithm on the graph associated with $\{I_{neq} \cup D_{om}\}$. (See Sect. 3.2.) Since the graph is likely to contain a negative cycle, this step can be achieved in $O(mn)$ running time.

It is also easy to show that step 2 can be performed with a simple algorithm that runs in $O(n)$ where $n$ is the number of variables. (See Sect. 3.1.)

The contribution of this paper is an efficient filtering algorithm for enforcing interval consistency on the global constraint $IS$.

Outline of the paper. Section 2 introduces the notation and recalls the basics of CSP and of shortest paths that are needed in the rest of the paper. Section 3 successively shows how interval consistency can be achieved on a sum constraint and on binary inequalities. Section 4 defines interval consistency on the global constraint $IS$ while Section 5 details the algorithm for finding a minimum value of $x_i$ with respect to constraints $I_{neq} \cup D_{om} \cup \{S_{um}\}$.

2. Background

In order to make this paper self-contained, we now introduce the required background of CSP and of shortest paths.

2.1. Basics of CSP

A finite constraint network $P = (X, D, C)$ is defined by:

- a set of variables $X = \{x_1, ..., x_n\}$;
- a set $D = \{D(x_1), ..., D(x_n)\}$ of current domains where $D(x_i)$ is a finite set of possible values for variable $x_i$;
- a set $C$ of constraints between the variables.

A total ordering $\prec$ can be defined on the domains without loss of generality. We will denote by $\min(x_i)$ and $\max(x_i)$ the minimal and the maximal value of $D(x_i)$ w.r.t. to $\prec$. 

A constraint $C$ on the ordered set of variables $X(C) = (x_1, ..., x_r)$ is a subset $T(C)$ of the Cartesian product $D(x_1) \times \cdots \times D(x_r)$ that specifies the allowed combinations of values for the variables $(x_1, ..., x_r)$. An element of $D(x_1) \times \cdots \times D(x_r)$ is called a tuple on $X(C)$ and is noted $\tau$. $\tau[k]$ is the $k$th value of $\tau$. $|X(C)|$ is the arity of $C$.

A value $a$ for $x$ is often denoted by $(x, a)$ while $\text{index}(C, x)$ is the position of $x$ in $X(C)$.

Let $P = (X, D, C)$ be a constraint network. The tuple $\tau = (v_1, \ldots, v_n)$ is a solution of $P$ if the assignment $((x_1, v_1), \ldots, (x_n, v_n))$ satisfies all the constraints of $C$. A value $v$ is a feasible value for $x$ if there exists a solution in which $x = v$.

Let $C$ be a constraint of $\mathcal{C}$. A tuple $\tau$ of $X(C)$ is valid if $\forall (x, a) \in \tau, a \in D(x)$. A value $a \in D(x)$ is consistent with $C$, either if $x \notin X(C)$, or if there exists a valid tuple $\tau \subset T(C)$ such that $a = \tau[\text{index}(C, x)]$. A constraint is arc consistent if $\forall x_i \in X(C), D(x_i) \neq \emptyset$ and $\forall a \in D(x_i)$, $a$ is consistent with $C$.

Filtering by arc consistency is often too costly for non-binary constraints and global constraints. Interval consistency [8] can be achieved more efficiently. In particular, this is the case for constraints interval-consistent but not arc-consistent since $(x, 0)$ is not consistent with this constraint.

2.2. Basics of shortest paths

We briefly recall here a few ideas about shortest paths that are needed in the rest of the paper. Most of the definitions are due to Tarjan [14].

Let $G = (X, U)$ be a directed graph, where $X$ is a set of nodes and $U$ a set of arcs; $m$ denotes $|U|$ whereas $n$ denotes $|X|$. Each arc $(i, j)$ is associated with an integer called the cost of the arc and denoted $c_{ij}$. A path from node $v_i$ to node $v_k$ in $G$ is a list of nodes $[v_i, \ldots, v_k]$ such that $(v_i, v_{i+1})$ is an arc for $i \in [1..k-1]$. A path is simple if all its nodes are distinct. A path is a cycle if $k > 1$ and $v_1 = v_k$. 

$|\mathcal{C}|$ denotes the number of constraints while $|X|$ denotes the number of variables.
The length of a path \( p \), denoted by \( \text{length}(p) \), is the sum of the costs of the arcs contained in \( p \). A shortest path from a node \( s \) to a node \( t \) is a path from \( s \) to \( t \) whose length is minimum. A cycle of negative length is called a negative cycle. There is a shortest path from \( s \) to \( t \) if and only if no path from \( s \) to \( t \) contains a negative cycle. \( d(u, v) \) denotes the shortest path distance from node \( u \) to node \( v \) in \( G \) while \( s \) denotes the source node.

\( G_{rc} \) is the graph derived from \( G \) by replacing, for each arc \((u, v), c_{uv}\) with its reduced cost \( rc_{uv} = c_{uv} + d(s, u) - d(s, v) \). The shortest path distance from node \( a \) to node \( b \) in \( G_{rc} \) is denoted by \( d^0(a, b) \). The following properties \([1]\) hold in \( G_{rc} \):

1. \( \forall(u, v) \in G : rc_{uv} \geq 0; \)
2. \( d(a, b) = d^0(a, b) - d(s, a) + d(s, b) \).

3. INTERVAL CONSISTENCY FILTERING

This section successively shows how interval consistency can be achieved on a sum constraint and on binary inequalities.

3.1. SUM CONSTRAINT

We will consider the following definition of a sum constraint:

**Definition 1.** Let \( X = \{x_1, ..., x_r\} \) be a set of variables. \( S_{um} = SUM(X, y) \) is a sum constraint defined by the set of tuples \( T(S_{um}) \):

\[
T(S_{um}) = \left\{ \tau : \tau \text{ is a tuple of } X \cup \{y\} \land \sum_{i=1}^{r} \tau[i] - \tau[\text{index}(S_{um}, y)] = 0 \right\}.
\]

**Proposition 1.** Let \( X = \{x_1, ..., x_r\} \) be a set of variables whose domain are interval of integers. Then, for every value \( v \) such that \( \sum_{x_i \in X} \min(D(x_i)) \leq v \leq \sum_{x_i \in X} \max(D(x_i)) \) there exists an instantiation of the variables of \( X \) such that \( \sum_{x_i \in X} x_i = v \).

**Proof.** Let \( S \min = \sum_{x_i \in X} \min(D(x_i)) \). We have \( S \min \leq v \leq \sum_{x_i \in X} \max(D(x_i)) \). Consider any ordering of the variables of \( X \): \( x_1, ..., x_r \). There exists an index \( i \) such that \( v = S \min + \sum_{j=1}^{i-1} (\max(D(x_j)) - \min(D(x_j))) + p \) with \( p \leq \max(D(x_i)) - \min(D(x_i)) \). Since \( D^*(x_i) \) is the interval of integers \([\min(D(x_i)), \max(D(x_i))]\) then \( p \in D^*(x_i) \) and the instantiation of \( X \) defined by \( x_j = \max(D(x_j)) \) if \( j < i \), \( x_i = p + \min(D(x_i)) \), and \( x_j = \min(D(x_j)) \) if \( j > i \) satisfies \( \sum_{x_i \in X} x_i = v \). \( \square \)

From this proposition we have:

**Corollary 1.** Let \( X = \{x_1, ..., x_r\} \) be a set of variables whose domain are interval of integers. Then establishing the interval consistency of the constraints \( \sum_{x_i \in X} x_i \leq v \) and \( \sum_{x_i \in X} x_i \geq v \) is equivalent to establish the interval consistency of the constraint \( \sum_{x_i \in X} x_i = v \).
From Corollary 1 we immediately have:

**Proposition 2.** Let $X \cup \{y\}$ be a set of variables and let $S_{um} = SUM(X,y)$ be a sum constraint. $S_{um}$ is interval-consistent if and only if the following four conditions hold:

1. $\min(y) \geq \sum_{x_i \in X} \min(x_i)$;
2. $\max(y) \leq \sum_{x_i \in X} \max(x_i)$;
3. $\forall x_i \in X : \min(x_i) \geq \min(y) - \sum_{x_j \in X - \{x_i\}} \max(x_j)$;
4. $\forall x_i \in X : \max(x_i) \leq \max(y) - \sum_{x_j \in X - \{x_i\}} \min(x_j)$.

Interval consistency filtering of $SUM(X,y)$ can be achieved efficiently in an incremental way. The essential observation is that $\sum_{x_j \in X - \{x_i\}} \max(x_j)$ is equal to $\sum_{x_j \in X} \max(x_j) - \max(x_i)$ and $\sum_{x_j \in X - \{x_i\}} \min(x_j)$ is equal to $\sum_{x_j \in X} \min(x_j) - \min(x_i)$. Since the sum over $X$ can be computed only once, the above conditions can be checked in $O(n)$. Thus, the cost of updating the intervals after a modification of bounds of several variables is in $O(n)$. What is instructive with this complexity is the fact that it does not depend on the size of the domains of the variables.

We would like to emphasize that if the domains of the variables are not considered as interval of integers it becomes difficult to establish the consistency of the constraint as shown by the following proposition:

**Proposition 3.** Finding a tuple on the variables of $X$ such that $\sum_{x_i \in X} x_i = v$ is an $NP$-Complete problem in general.

**Proof.** This problem is obviously in $NP$ (easy polynomial certificate). We transform SUMSET-SUM to this problem. SUMSET-SUM is: *Instance*: finite set $A$, size $s(a_i)$ in $\mathbb{Z}^+$ for each $a_i \in A$, positive integer $k$. *Question*: is there a subset $A'$ of $A$ such that the sum of the sizes of the elements in $A'$ is exactly $k$?

For each $a_i \in A$ we define a variable $x_i$ whose domain is $\{0, s(a_i)\}$. Then $\sum_{x_i \in X} x_i = k$ exactly solves SUBSET-SUM. □

### 3.2. Binary inequalities

Arc consistency can be achieved on binary inequalities like $I_{neq}$ by using specific filtering algorithms such as AC-5 [15]. However, the complexity of such algorithms depends on the size of domains of the variables. Thus, they are rather ineffective for detecting inconsistencies. Interval consistency can be achieved in $O(mn)$ where $n$ is the number of variables and $m = |I_{neq}| + 2n$. This is due to a result of Dechter et al. [5] on the “Simple Temporal Constraint Satisfaction Problem” (STCSP). Roughly speaking, interval consistency can be achieved by searching for shortest paths in a particular graph $G = (N,E)$, called the distance graph, where node set $N$ represents the variables and arc set $E$ stands for the inequality constraints.
More formally, let $P = (\mathcal{X}, \mathcal{D}_c, \mathcal{C}_{eq})$ be a CSP where $\mathcal{D}_c$ denotes a set of continuous domains. The distance graph $G = (N, E)$ associated with $P$ is defined in the following way (see Fig. 1):

- The node set $N$ contains:
  - A special node $s$, named source, with a domain $D(s)$ that is reduced to a single value $\{0\}$.
  - One node for each variable $x_i$ in $\mathcal{X}$.

- The arc set $E$ contains:
  - An arc $(x_j, x_i)$ with cost $c_{ji}$ for each inequality $x_i \leq x_j + c_{ji}$.
  - An arc $(x_i, s)$ with cost $-\min(x_i)$ for each variable $x_i$ in $\mathcal{X}$.
  - An arc $(s, x_i)$ with cost $\max(x_i)$ for each variable $x_i$ in $\mathcal{X}$.

Arcs $(x_i, s)$ and arcs $(s, x_i)$ result from the definition of domain $D^*_c(x_i) = [\min(x_i), \max(x_i)]$ by the inequalities: $0 \leq x_i - \min(x_i)$ (so $s \leq x_i - \min(x_i)$) and $x_i \leq \max(x_i)$ (so $x_i \leq s + \max(x_i)$).

This problem statement results from the following optimality condition of shortest paths: 
$$d(s, x_j) \leq d(s, x_i) + c_{ij} \quad \text{for all } (x_i, x_j) \in N.$$ This inequality states that for every arc $(x_i, x_j)$ in the network the length of the shortest path to node $x_j$ is not greater than the length of the shortest path to node $x_i$ plus the length of the arc $(x_i, x_j)$. Dechter et al. have shown [5] that:

**Theorem 1.** A STCSP is consistent iff its distance graph has no negative directed cycles.

**Theorem 2.** Let $G$ be the directed graph representation of a consistent STCSP $P = (\mathcal{X}, \mathcal{D}, \mathcal{C})$ where $\mathcal{C}$ is a set of binary inequalities. The set of feasible values for $x_i$ is $[-d(x_i, s), +d(s, x_i)]$, where $d(x_i, x_j)$ denotes the shortest path from node $x_i$ to node $x_j$. 
Dechter et al. have extended network based methods for solving mixed-integer linear problems [10]. So, it is trivial to show that their results hold when the domains are restricted to intervals of integers.

Theorem 1 states that the problem has no solution if $G$ contains a negative cycle. Indeed, a negative cycle indicates that some of the inequalities are contradictory. The following property results from Theorem 2:

**Proposition 4.** Let $P = (X, D, C)$ be a STCSP and let $G = (N, E)$ be the distance graph associated with $P^* = (X, D^*, C)$. If $G$ contains no negative cycle, then

$$\forall x_i \in X : (D^*(x_i) = [-d(x_i, s), d(s, x_i)]) \Rightarrow P \text{ is interval-consistent.}$$

**Proof.** Assume that $D^*(x_i) = [-d(x_i, s), d(s, x_i)]$. Since $G$ contains no negative cycles, it results from Theorem 2 that $-d(x_i, s)$ and $d(s, x_i)$ are feasible values. Thus, $P$ is interval-consistent. □

According to Theorem 2, interval consistency can be achieved by computing the shortest paths between $s$ and the $x_i$, when $G$ does not contain any negative cycle. Computing shortest paths when the problem graph is likely to contain a negative cycle can be achieved in $O(mn)$ running time [14]. When the graph contains no negative arcs, Dijkstra’s algorithm computes shortest paths in $O(m + n \log n)$. Of course, Dijkstra’s algorithm can always be used on the graph of reduced costs. A nice property of the distance graph $G = (N, E)$ associated with $P^* = (X, D^*, C)$ is that the reduced costs can be derived from the minimal and maximal values of the domains.

**Proposition 5.** Let $P = (X, D, C)$ be a CSP and let $G = (N, E)$ be the distance graph associated with $P^* = (X, D^*, C)$. If $P$ is interval-consistent, then the following relations hold:

$$\forall x_i, x_j \in X : rc_{ij} = c_{ij} + \max(x_i) - \max(x_j)$$

$$\forall x_i, x_j \in X : d(x_i, x_j) = d^0(x_i, x_j) - \max(x_i) + \max(x_j).$$

These properties trivially result from the definition of the reduced costs and Theorem 2.

4. **Global IS constraint**

Now, let us show how interval consistency of the global constraint $IS$ can be achieved. A global constraint $IS$ represents the conjunction of a sum constraint and a set of binary inequalities defined on variables involved in the sum constraint. More formally, we have:

**Definition 2.** Let $SUM(X, y)$ be a sum constraint, and $I_{neq}$ be a set of binary inequalities defined on $X = (x_1, \ldots, x_r)$. Global constraint $IS(X, y, I_{neq})$ is defined by the set of tuples $T(IS)$:

$$T(IS) = \{ \tau : \tau \text{ is a tuple of } X(IS) \land \sum_{i=1}^{r} \tau[i] - \tau[\text{index(IS, y)}] = 0 \land$$

$$\forall (x_i \leq x_j + c_{ji}) \in I_{neq} : \tau[i] \leq \tau[j] + c_{ji} \}. $$
To define interval consistency for IS, we have to extend inequalities of Proposition 2 in order to take into account the binary inequalities between the variables involved in the sum constraint.

Let us highlight this point by considering again the initial example. Now, the constraint network is expressed with one global constraint IS:

\[
IS(\{x_1, x_2\}, y, \{(x_1 \leq x_2 - 1)\})
\]

where \(D(x_1) = [0, 6]\), \(D(x_2) = [1, 7]\) and \(D(y) = [1, 13]\).

Suppose that \(\min(y)\) is set to 6. Inequality (3) of Proposition 2 states that:

\[
\forall x_i \in X : \min(x_i) \geq \min(y) - \sum_{x_j \in X - \{x_i\}} \max(x_j).
\]

So, for \(x_2\), we have: \(\min(x_2) \geq 6 - \max(x_1)\). Since \(\max(x_1) = 6\). This inequality holds when \(\min(x_2)\) is equal to 1 although the constraint \((x_1 \leq x_2 - 1)\) is violated for \(x_2 = 1\) and \(x_1 = 6\).

Thus, inequality (3) must be modified in order to take into account the constraint \((x_1 \leq x_2 - 1)\). More precisely, the value of \(x_i\) and the upper bounds of \(x_j\) considered in inequality (3) should satisfy the constraints \((x_i \leq x_j - c_{ji})\) of \(I_{neq}\).

Let \(\min_{(x_i \leftarrow a)}(x_j)\) and \(\max_{(x_i \leftarrow a)}(x_j)\) respectively be the minimum and the maximum values of \(D^*(x_j)\) which satisfy the binary inequalities \(I_{neq}\) when \(x_i\) is instantiated to \(a\). Using this notation, inequality (3) can be rewritten in the following form:

\[
\forall x_i \in X : (x_i \leftarrow a) \Rightarrow a \geq \min(y) - \sum_{j \neq i} \max_{(x_i \leftarrow a)}(x_j).
\]

This new inequality does not hold for \(x_2 = 1\) and \(x_1 = 6\). The smallest value of \(D^*(x_2)\) that satisfies this inequality is 4.

So, interval-consistency of IS can be defined in the following way:

**Proposition 6.** Let \(X \cup \{y\}\) be a set variables and let IS\((X, y, I_{neq})\) be a global sum constraint. IS is interval-consistent iff the following conditions hold:

1. \(\min(y) \geq \sum_{x_i \in X} \min(x_i)\)
2. \(\max(y) \leq \sum_{x_i \in X} \max(x_i)\)
3. \(\forall x_i \in X : \min(x_i) \geq \min(y) - \sum_{x_j \in X - \{x_i\}} \max_{(x_i \leftarrow \min(x_i))}(x_j)\)
4. \(\forall x_i \in X : \max(x_i) \leq \max(y) - \sum_{x_j \in X - \{x_i\}} \min_{(x_i \leftarrow \max(x_i))}(x_j)\)

**Proof.** From inequalities (1b) and (2b) it results that constraint \(SUM(X, y)\) holds when \(y\) is set to \(\min(y)\) and when \(y\) is set to \(\max(y)\). Since \(y\) occurs only in \(SUM(X, y)\), it follows that IS is interval-consistent for \(y\). Inequality (3b) ensures that both constraint \(SUM(X, y)\) and the inequalities of \(I_{neq}\) hold when \(x_i\) is set to \(\min(x_i)\). This reasoning remains valid for inequality (4b).
Conversely, it is trivial to show that inequalities (1b), (2b), (3b), and (4b) hold if IS is interval-consistent.

The scheme of the interval consistency filtering algorithm of constraint IS is given in Algorithm 1. This algorithm is started whenever one bound of a variable in X ∪ {y} are modified. Note that steps 1 and 2 are systematically performed when interval consistency on IS is achieved for the first time. Negative cycles are detected in step 1 of Algorithm 1.

In the rest of this paper, we will only detail the search process of the minimum value of a variable (step 3 in Algorithm 1) since the same kind of reasoning holds for searching maximum values (step 4 in Algorithm 1).

Algorithm 1: Filtering IS by interval consistency

1. Interval consistency is achieved on SUM(X, y) with an algorithm derived from Proposition 2 whenever a bound of y is modified.
2. Interval consistency is achieved on the conjunction of binary inequalities \( I_{\text{neq}} \cup \mathcal{D}_0 \) with a shortest path algorithm whenever a bound of some variable of X is modified.
3. For every variable \( x \in X \), the minimum value of x satisfying inequality (3b) will be computed.
4. For every variable \( x \in X \), the maximum value of x satisfying inequality (4b) will be computed.

5. COMPUTING A NEW MINIMUM

The goal is to find the smallest value \( x_i \in [\min(x_i), \max(x_i)] \) such that inequality (3b) of Proposition 6 holds. It follows from inequality (3b) that

\[
\hat{x}_i = \min(y) - \sum_{x_j \in X - \{x_i\}} \max_{x_i \leftarrow x_i} (x_j).
\]

So, we have to determine the largest value of \( x_j \) which is consistent with \( x_i \).

The algorithm is based on the following properties:

**Proposition 7.** Let \( d_{G_{x_i \leftarrow x_i}}(s, x_j) \) denote the shortest path from s to \( x_j \) in the graph derived from G by instantiating \( x_i \) with \( x_i \). Then,

\[
\max_{(x_i \leftarrow x_i)} (x_j) = d_{G_{x_i \leftarrow x_i}}(s, x_j).
\]

**Proof.** This property results from Theorem 2 when \( G_{x_i \leftarrow x_i} \) does not contain any negative cycle. Negative cycles are detected in step 1 of Algorithm 1. So, if there exists a negative cycle, it has been introduced by the instantiation of \( x_i \) with \( x_i \) in \( G_{x_i \leftarrow x_i} \).
Suppose that the instantiation of $x_i$ with $x_i$ introduces a negative cycle in $G_{x_i \leftarrow x_i}$. This cycle would contain $x_i$, so we would have $d(s, x_i) + d(x_i, s) < 0$ and thus $2d_{\min}(x_i) < 0$ which would be in contradiction with $x_i \in [\min(x_i), \max(x_i)]$. 

To compute $d_{G_{x_i \leftarrow x_i}}(s, x_j)$ efficiently, we have to establish a link between the shortest paths in $G$ and the ones in $G_{x_i \leftarrow x_i}$; otherwise we would have to compute the shortest paths at each step on a new graph. The following propositions help us to establish such a connection.

**Proposition 8.** Let $G = (X, U)$.

$$\forall s, x_i, x_j \in X : d_G(s, x_j) = \min (d_G(s, x_i) + d_{G - \{x_i\}}(x_i, x_j), d_{G_{x_i \leftarrow x_i}}(s, x_j)).$$

**Proof.** First, let us recall that in a graph without negative cycles, it exists a simple shortest path whenever a shortest path exists. So, we have to examine two cases:

1. No simple shortest path from $s$ to $x_j$ contains $x_i$. Then, we have $d_G(s, x_j) = d_{G - \{x_i\}}(s, x_j)$ and the proposition holds.

2. Some simple shortest path $P$ from $s$ to $x_j$ contains $x_i$. That’s to say $P = d_G(s, x_i) + d_G(x_i, s)$. Since $d_G(s, x_i)$ contains $x_i$, $s$ cannot belong to $d_G(x_i, s)$ and we have $d_G(s, x_j) = d_G(s, x_i) + d_{G - \{x_i\}}(x_i, x_j)$.

Next property states that if the shortest path from $s$ to $x_j$ goes through $x_i$ when $x_i$ is instantiated to $x_i$, then $\max(x_i, x_j) = x_i + d_{G - \{x_i\}}(x_i, x_j)$, otherwise $\max(x_i, x_j) = \max(x_j)$.

**Proposition 9.** After the achievement of steps 1 and 2 of Algorithm 1 we have:

$$\max_{\langle x_i \leftarrow x_i \rangle} (x_j) = \min (x_i + d_{G - \{x_i\}}(x_i, x_j), \max(x_j)).$$

**Proof.** Proposition 8 states for the graph $G_{x_i \leftarrow x_i}$:

$$d_{G_{x_i \leftarrow x_i}}(s, x_j) = \min (d_{G_{x_i \leftarrow x_i}}(s, x_i) + d_{G_{x_i \leftarrow x_i} - \{x_i\}}(x_i, x_j), d_{G_{x_i \leftarrow x_i} - \{x_i\}}(s, x_j)).$$

Since the instantiation of $x_i$ to $x_i$ only modifies the values of $c_{x_i}$ and of $c_{x_i, s}$, the shortest paths in $G - \{x_i\}$ are equal to the shortest paths in $G_{x_i \leftarrow x_i} - \{x_i\}$. So, we have:

$$d_{G_{x_i \leftarrow x_i}}(s, x_j) = \min (d_{G_{x_i \leftarrow x_i}}(s, x_i) + d_{G - \{x_i\}}(x_i, x_j), d_{G - \{x_i\}}(s, x_j)).$$

On the other hand, after the achievement of step 2 of Algorithm 1, Theorem 2 ensures that the set of feasible values for $x_i$ is $[-d(x_i, s), +d(s, x_i)]$. The graph $G_{x_i \leftarrow x_i}$ is the graph $G$ where the values of $c_{x_i}$ and of $c_{x_i, s}$ have respectively been set to $x_i$ and to $-x_i$. Since $x_i \in [-d(x_i, s), +d(s, x_i)]$, $x_i$ is a valid value, and thus, Theorem 2 on $G_{x_i \leftarrow x_i}$ states that the set of feasible values of $x_i$ is $[-d_{G_{x_i \leftarrow x_i}}(x_i, s), +d_{G_{x_i \leftarrow x_i}}(s, x_i)] = [x_i, x_i]$. So, $d_{G_{x_i \leftarrow x_i}}(s, x_i) = x_i$. 

Moreover, from Theorem 2 we have \( d(s, x_j) = \max(x_j) \). Therefore the proposition holds.

Let \( S \) be the set of variables \( x_j \) for which \( x_j + d_{G-\{s\}}(x_i, x_j) > \max(x_j) \), that’s to say \( d_{G-\{s\}}(x_i, x_j) > \max(x_j) - x_i = c_{x_i, s} + d(s, x_j) = d(x_i, x_j) \). In other words, \( S \) is the set of variables for which the shortest path \( d_{G-\{s\}}(x_i, x_j) \) in \( G - \{s\} \) is greater than the shortest path \( d(x_i, x_j) \) in \( G \).

Then, by Proposition 9, equation (1) is equivalent to:

\[
x_i = \min(y) - \sum_{y \in S} \max(x_k) - \sum_{y \in X - (S \cup \{x_i\})} x_i + d_{G-\{s\}}(x_i, x_j)
\]

which is equivalent to

\[
x_i = \min\left(\frac{1}{|X| - |S|} \left( \min(y) - \sum_{y \in S} \max(x_k) - \sum_{y \in X - (S \cup \{x_i\})} d_{G-\{s\}}(x_i, x_j) \right) \right).
\]

We are now in position to define an algorithm for the computation of \( x_i \) in an iterative way. The key idea of the algorithm is that we can find the value of \( x_i \) by seeking for the minimal set \( S \) consistent with equation (3). Let \( T \) be a set of variables and

\[
\alpha(T) = \min\left(\frac{1}{|X| - |T|} \left( \min(y) - \sum_{x \in T} \max(x_k) - \sum_{y \in X - (T \cup \{x_i\})} d_{G-\{s\}}(x_i, x_j) \right) \right).
\]

If \( T \) is minimal and if \( T \) is the set of variables \( x_j \) for which \( \alpha(T) + d_{G-\{s\}}(x_i, x_j) > \max(x_j) \) then \( x_i = \max(\min(x_i), \alpha(T)) \). A initial set \( T \) can be defined by the variables \( x_j \) for which \( \min(x_i) + d_{G-\{s\}}(x_i, x_j) > \max(x_j) \). Then, the following procedure is repeated: compute \( \alpha(T) \) and compute the new set \( T \) according to the value of \( \alpha(T) \). This procedure is repeated until a fix point is reached, that is \( T \) is no longer modified by the new procedure. Algorithm 2 implements this mechanism. The point is that new variables may be added to \( T \) when the value of \( \alpha(T) \) is shifted up. Indeed, \( x_i = \alpha(T) \) belongs to less and less shortest paths, as the value of \( x_i \) increases. Note that the value of \( \alpha(T) \) computed by the function \textbf{Lower-bound} may be smaller than \( \min(x_i) \) when \( \min(y) \) does not introduce any constraint. That’s why this function returns \( \max(\alpha(T), \min(x_i)) \).

Algorithm 2 computes \( x_i \) in an at most \( n \) iterations since at least one variable is put in \( T \) at each step. The two sums can be updated in \( O(1) \) when a new element is added to \( T \).

Before starting this algorithm we have to compute the shortest paths \( d_{G-\{s\}}(x_i, x_j) \). The point is that the \( d_{G-\{s\}}(x_i, x_j) \) can be computed on the graph of reduced costs. (See Property 5.) Moreover, these shortest path distances have only to be computed once since they do not depend on the values of the domains. So, the greatest value of \( x_j \) which is consistent with \( x_i \) can be determined by computing \( d_{G-\{s\}}(x_i, x_j) \) on the graph of reduced costs. No propagation
Algorithm 2: Computing $x_i$

Function(Lower-bound(IN: $\Delta$, $x_i$, OUT :$x_i$))

% $\Delta$ : sorted list of $\Delta_j = \max(x_j) - d_{G-\{s\}}(x_i, x_j)$

$T \leftarrow \emptyset$

$\alpha(T) \leftarrow \min(x_i)$

repeat

$T \leftarrow T \cup \{x_j : \Delta_j < \alpha(T)\}$

$\alpha(x_i) \leftarrow \left[ \frac{1}{|X| - |T|} \left( \min(y) - \sum_{x_k \in T} \max(x_k) - \sum_{x_j \in X - (T \cup \{x_i\})} d_{G-\{s\}}(x_i, x_j) \right) \right]$

until $T$ does no more change

return $\max(\alpha(T), \min(x_i))$

---

**Figure 2.** Distance graph associated to the CSP $P_1$.

step is required: when $\min(x_i)$ is increased it is useless to reconsider $\min(x_j)$ if $x_j$ has been updated before $x_i$ during step 3 of the interval consistency filtering algorithm. (See Algorithm 1.) This results from Proposition 4 which states that $-d(x_i, s)$ is the lower bound of $D^*(x_i)$.

Here is a short example that illustrates the process performed by algorithm 2. Consider the CSP $P_1$ defined by the following distance constraints:

$$
D^*_{(x_1)} = [3, 10] \quad D^*_{(x_2)} = [1, 3] \quad D^*_{(x_3)} = [3, 5] \quad D^*_{(x_4)} = [5, 8] \quad D^*_{(x_5)} = [8, 10]
$$

$\begin{align*}
  x_2 &\leq x_3 - 1 \\
  x_3 &\leq x_4 - 2 \\
  x_4 &\leq x_5 - 2 \\
  x_1 &\leq x_4 + 2.
\end{align*}$

The distance graph associated to the CSP $P_1$ is given by Figure 2.

Assume that we want to compute $x_4$ after a modification of $\min(y)$. The data compute before the algorithm starts are reported in Table 1.
Table 1. Data computed for CSP $P_1$.
\[
\begin{array}{|c|c|c|}
\hline
i & \max(i) & d_{G-{s}}(x_4,i), \Delta_i = \max(i) - d_{G-{s}}(x_4,i) \\
\hline
1 & 10 & 2 & 8 \\
2 & 3 & -3 & 6 \\
3 & 5 & -2 & 7 \\
5 & 10 & \infty & -\infty \\
\hline
\end{array}
\]

Suppose that $\min(y)$ is set to 26. Since $\min(x_4) = 5$, variables $x_5$ is added to $T$. Thus, $\alpha(T) \leftarrow \lceil \frac{1}{4}(26 - (10) - (2 - 3 - 2)) \rceil = 5$. No variable is added to $T$ and the algorithm returns $\min(x_4)$.

Now suppose that $\min(y)$ is shifted up to 35. The set $T$ still contains $x_5$ at the beginning but now $\alpha(T)$ is equal to $\lceil \frac{1}{4}(35 - (10) - (2 - 3 - 2)) \rceil = 7$. Thus, $x_2$ is added to $T$ and $\alpha(T)$ is set to $\lceil \frac{1}{4}(35 - (10 + 3) - (2 - 2)) \rceil = 8$. Then, $x_3$ is added to $T$ and $\alpha(T)$ is set to $\lceil \frac{1}{4}(35 - (10 + 3 + 5) - (2 - 2)) \rceil = 8$. The fixed point is reached and the algorithm returns 8 that actually corresponds to the smallest value of $x_4$ which is interval consistent in the constraint system $P_1 \cup \{\min(y) = 35\}$.

5.1. Complexity issues

The shortest distance between every $x_i$ and all other nodes can be computed with Dijkstra’s shortest path algorithm on the graph of reduced costs in $n \times O(m + n \log n)$. Of course, since we have to check whether the graph contains negative cycles the cost of the shortest paths between the first considered $x_i$ and all other nodes will be in $O(nm)$. The shortest paths have only to be recomputed when the constraints changes$^1$.

$\Delta_i$ can be computed in $O(n)$ time. Sorting $\Delta$ costs $O(n \log n)$ whereas $\alpha(x_i)$ can be computed in constant time at each iteration step. Then, the computation of $x_i$ can be achieved in $O(n)$ time.

So, the cost enforcing interval consistency on the $IS$ constraint is $n \times O(m + n \log n)$ time.

Maintaining interval consistency on $IS$ after the modification of some bound can be done in $n \log(n)$ time.

6. Discussion

It is also instructive to remark that our algorithm still works when the function to be optimized is of the form $y = \sum_{i=1}^{n} \alpha_i x_i$ where the $\alpha_i$ are non-negative real

$^1$ Constraints may change at the branching step in optimization problems; for instance, a constraint $|x - y| \geq 5$ yields two constraint systems: one with the constraint $x - y \geq 5$, and one with the constraint $y - x \geq 5$. 
numbers. However, interval consistency of IS can no longer be established in polynomial time since in this case the sum constraint becomes NP-Complete:

**Proposition 10.** Finding a tuple on the variables of $X$ such that $\sum_{x_i \in X} a_i x_i = v$ is an NP-Complete problem even if the domain of the variables of $X$ are interval of integers.

**Proof.** This problem is obviously in NP (easy polynomial certificate). We transform SUMSET-SUM to this problem. SUMSET-SUM is:

**Instance:** finite set $A$, size $s(a_i)$ in $\mathbb{Z}^+$ for each $a_i \in A$, positive integer $k$. **Question:** is there a subset $A'$ of $A$ such that the sum of the sizes of the elements in $A'$ is exactly $k$?

For each $a_i \in A$ we define a variable $x_i$ whose domain is the interval of integers $[0, 1]$. Then $\sum_{x_i \in X} (s(a_i)x_i) = k$ exactly solves SUBSET-SUM. □

Thus Corollary 1 cannot be extended for the case where the function to be optimized is of the form $y = \sum_{i=1}^{n} \alpha_i x_i$. Nevertheless, we can modify the previous algorithm in order to obtain a weaker filtering algorithm.

To capture the exact contribution of each $x_i$ in the sum when the $\alpha_i$ are different from the value 1, we need only introduce the coefficient of $x_i$ in Equation 1:

$$x_i = \frac{1}{\alpha_i} \left( \min(y) - \sum_{x_j \in X - \{x_i\}} \alpha_j \max_{x_i = x_j} (x_j) \right).$$

(4)

In Section 1, we defined $I_{\text{neq}}$ as the subset of binary inequalities that involve only variables occurring in the objective function $f(x)$. However, $I_{\text{neq}}$ could be extended to the subset of binary inequalities that involve either variables occurring in $f(x)$ or variables connected to variables occurring in $f(x)$. For instance, assume that $x_1$ and $x_2$ occur in the objective and let $\{x_1 \leq y_1 + c; y_1 \leq y_2 + c'; z_1 \leq z_2 + c''\}$ be the set of binary inequalities. Then, this extended set of inequalities would contain $\{x_1 \leq y_1 + c; y_1 \leq y_2 + c'\}$.

Let $S_x$ be the set of variables occurring in $f(x)$. To capture this extension, we need only to replace $X$ by $S_x$ in $\sum_{x_j \in X - \{x_i\}}$ of equations (1) and (5). Considering this extended set of inequalities may entail a better pruning of the domains.

### 7. Conclusion

This paper has introduced a new global constraint which handles as a single constraint a sum constraint and a system of binary linear inequalities. An efficient algorithm has been proposed to achieve an interval-consistent filtering of this new global constraint. The cost of this algorithm is not higher than the cost of a filtering algorithm which handles only the inequalities. A direct application of this constraint concerns optimization problems where it introduces a kind of “back” propagation process.
References