A Global Constraint Combining a Sum Constraint and Difference Constraints

Jean-Charles Régin[1] and Michel Rueher[2]

[1] ILOG Les Taissounieres HB2
1681, route des Dolines Sophia Antipolis 06560 Valbonne
regin@ilog.fr
[2] Université de Nice–Sophia-Antipolis, I3S, ESSI
930, route des Colles - B.P. 145 06903 Sophia-Antipolis, France
rueher@essi.fr

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 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. 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 Dikjstra's shortest path algorithm, is introduced to achieve interval consistency on this global constraint.

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 [Sim96]. These new filtering algorithms work on so–called "global constraints", e.g., cumulative constraint [BC94], edge-finder algorithm [CP94,Nui94], all–diff constraint [Rég94], cardinality constraint [HD91,Rég96]. 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

R. Dechter (Ed.): CP 2000, LNCS 1894, pp. 384–395, 2000.

function is defined by a sum $y = \Sigma x_i$, and where the 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 [BESW93].

The mean flow time is defined by $\overline{F} = \frac{1}{n}\sum_{j=1}^{n}(C_j - r_j)$ where C_j and r_j are respectively the completion time and the ready time of task T_j. Difference constraints 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 minimization of the mean response time and the mean in-process time of the scheduled tasks set.
The mean tardiness is defined by $\overline{D} = \frac{1}{n}\sum_{j=1}^{n}(D_j)$ where $D_j = \max(C_j - d_j, 0)$, and where d_j is the due date of task T_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 [BESW93,DKD97].

Currently, in the constraint programming framework, such optimization problems 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:

$$\begin{array}{ll} \text{Minimize } f(x) & \\ \text{subject to} & p_i(x) \leq 0 \quad (i = 1, \ldots, m) \\ & q_i(x) = 0 \quad (i = 1, \ldots, r) \end{array}$$

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 $y = \Sigma_{i=1}^{i=n} x_i$. Handling a sum of the form $y = \Sigma_{i=1}^{i=n} a_i x_i$ is straightforward as shown in Section 6.

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 [DMP91] 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 `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 definition 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 $y = \Sigma_{i=1}^{i=n} x_i$,
- a set of binary inequalities $\mathcal{I}_{neq} = \{x_i - x_j \leq c_{ji}, \quad (i, j \in [1, n])\}$,
- a set of domain constraints $\mathcal{D}_{om} = \{l_i \leq x_i \leq u_i, \quad (i \in [1, n])\}$.

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

1. Filtering $\{\mathcal{I}_{neq} \cup \mathcal{D}_{om}\}$ by interval consistency;
2. Filtering S_{um} by interval consistency;
3. Updating the bounds of every x_i with respect to constraint set $\mathcal{I}_{neq} \cup \mathcal{D}_{om} \cup \{S_{um}\}$.

Step 1 can be achieved with a shortest path algorithm on the graph associated with $\{\mathcal{I}_{neq} \cup \mathcal{D}_{om}\}$. (See Section 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 Section 3.1.)
The contribution of this paper is an efficient *filtering algorithm* for step 3.

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 $\mathcal{I}_{neq} \cup \mathcal{D}_{om} \cup \{S_{um}\}$.

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** $P = (\mathcal{X}, \mathcal{D}, \mathcal{C})$ is defined by :

- a set of *variables* $\mathcal{X} = \{x_1, ..., x_n\}$;
- a set $\mathcal{D} = \{D(x_1), ..., D(x_n)\}$ of current *domains* where $D(x_i)$ is a finite set of possible values for variable x_i;
- a set $\mathcal{C}$ 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 $\prec$.
$|\mathcal{C}|$ denotes the number of constraints while $|X|$ denotes the number of variables.
A **constraint** C on the ordered set of variables $X(C) = (x_1, ..., x_r)$ is a subset $T(C)$ of the Cartesian product $D(x_1) \times ... \times D(x_r)$ that specifies the *allowed* combinations of values for the variables $(x_1, ..., x_r)$. An element of $D(x_1) \times ... \times D(x_r)$ is called a *tuple* on $X(C)$ and is noted τ. $\tau[k]$ is the k^{th} value of τ. $|X(C)|$ is the *arity* of C.

A value a for x is often denoted by (x, a) while $index(C, x)$ is the position of x in $X(C)$.
Let $P = (\mathcal{X}, \mathcal{D}, \mathcal{C})$ 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 $\mathcal{C}$. 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 τ 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** iff $\forall x_i \in X(C), D(x_i) \neq \emptyset$ 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** [HSD98] can be achieved more efficiently. Interval consistency is derived from an approximation of arc consistency for continuous domains. It is based on an approximation of finite domains by finite sets of successive integers. More precisely, if D is a domain, interval consistency works on D^* defined by the set $\{\min(D), \ldots, \max(D)\}$ where $\min(D)$ and $\max(D)$ denote respectively the minimum and maximum values in D. A constraint C is interval–consistent [1] if for all x_i in $X(C)$, $\min(D(x_i)) \leq \max(D(x_i))$ and if both $\min(D(x_i))$ and $\max(D(x_i))$ are consistent with C.

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 [Tar83].
Let $G = (X, U)$ be a directed graph, where X is a set of nodes and U a set of arcs. Each arc (i, j) is associated with an integer called the cost of the arc and denoted c_{ij}. A *path* from node v_1 to node v_k in G is a list of nodes $[v_1, ..., v_k]$ such that (v_i, v_{i+1}) is an arc for $i \in [1..k-1]$. A path is *simple* if all its nodes are distinct. A path is a *cycle* if $k > 1$ and $v_1 = v_k$. The *length* of a path p, denoted by $length(p)$, 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 iff 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 [AMO93] 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.

[1] For specific constraint systems, interval consistency and arc consistency are equivalent. In particular, this is the case for constraints $\mathcal{I}_{neq} \cup \mathcal{D}_{om} \cup S_{um}$ if the initial domains are finite sets of successive of integers (i.e., if $D_0^*(x_i) = D_0(x_i)$ for all variables). However, for more complex constraints this property does not hold. Consider for instance constraint $x^2 = 4$ and $D(x) = [-2, 2]$. This constraint is interval–consistent but not arc–consistent since $(x, 0)$ is not consistent with this constraint.

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 $T(S_{um})$:*
$T(S_{um}) = \{\ \tau$ *such that* τ *is a tuple of* $X \cup \{y\}$*, and*
$(\sum_{i=1}^{r} \tau[i]) - \tau[index(S_{um}, y)] = 0\}$

Proposition 1 *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) $\forall x_i \in X : \max(x_i) \leq \max(y) - \sum_{x_j \in X - \{x_i\}} \min(x_j)$

These conditions directly result from the definition of interval consistency. 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 $\sum_{x_j \in X} \max(x_j) - \max(x_i)$. 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.

3.2 Binary Inequalities

Arc consistency can be achieved on binary inequalities like $\mathcal{I}_{neq}$ by using specific filtering algorithms such as AC-5 [VDT92]. 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 $m = |\mathcal{I}_{neq}| + 2n$. This is due to a result of Dechter et al. [DMP91] 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 $P = (\mathcal{X}, \mathcal{D}^*, \mathcal{C})$ be a CSP such that $\mathcal{I}_{neq} \subset \mathcal{C}$ and where $\mathcal{D}^*$ denotes a set of continuous domains. The distance graph $G = (N, E)$ associated with P is defined in the following way:

- 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 $D^*(x_i) = \{\min(x_i), ..., \max(x_i)\}$ 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: $d(s, x_j) \leq d(s, x_i) + c_{ij}$ 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 [DMP91] that :

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

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 2 *Let $P = (\mathcal{X}, \mathcal{D}, \mathcal{C})$ be a STCSP and let $G = (N, E)$ be the distance graph associated with $P^* = (\mathcal{X}, \mathcal{D}^*, \mathcal{C})$.*
$\forall x_i \in \mathcal{X} : (D^*(x_i) = \{-d(x_i, s), ..., +d(s, x_i)\}) \Rightarrow P$ *is interval–consistent*

Proof:
Assume that $D^*(x_i) = \{-d(x_i, s), ..., +d(s, x_i)\}$ and that G contains no negative cycles. From Theorem 2 it results 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 [Tar83]. 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 $P^* = (\mathcal{X}, \mathcal{D}^*, \mathcal{C})$ is that the reduced costs can be derived from the minimal and maximal values of the domains.

Proposition 3 *Let $P = (\mathcal{X}, \mathcal{D}, \mathcal{C})$ be a CSP and let $G = (N, E)$ be the distance graph associated with $P^* = (\mathcal{X}, \mathcal{D}^*, \mathcal{C})$.*
$\forall x_i, x_j \in \mathcal{X} : \quad rc_{ij} = c_{ij} + \max(x_i) - \max(x_j)$
$\forall x_i, x_j \in \mathcal{X} : \quad d(x_i, x_j) = d^0(x_i, x_j) - \max(x_i) + \max(x_j)$

These properties trivially result from the definition of the reduced costs.

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 $\mathcal{I}_{neq}$ be a set of binary inequalities defined on $X = (x_1, \ldots, x_r)$. Global constraint* $\texttt{IS}(X, y, \mathcal{I}_{neq})$ *is defined by the set of tuples $T(IS)$:*
$T(IS) = \{\ \tau$ *such that* τ *is a tuple of* $X(IS)$*, and*
$(\sum_{i=1}^{r} \tau[i]) - \tau[index(IS, y)] = 0$*, and*
the values of τ *satisfy* $\mathcal{I}_{neq}$ $\}$

Interval consistency for IS can be defined by extending inequalities of Proposition 1 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:
$\texttt{IS}(\{x_1, x_2\}, y, \{(x_1 \leq x_2 - 1)\})$ where $D(x_1) = [0, 6]$, $D(x_2) = [1, 7]$ and $D(y) = [1, 13]$. Suppose that $\min(y)$ is set to 6. Let us recall inequality (3) of Proposition 1: $\forall x_i \in X : \min(x_i) \geq \min(y) - \sum_{x_j \in X - \{x_i\}} \max(x_j)$. For x_2, this inequality states that $\min(x_2) \geq 6 - \max(x_1)$. 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, $\max(x_1)$ must be compatible with the smallest value of $D^*(x_2)$ which is consistent with the constraint $(x_1 \leq x_2 - 1)$.

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 $\mathcal{I}_{neq}$ when x_i is instantiated to a. Using this notation, inequality (3) can be rewritten in the following form:

$$\forall a \in D^*(x_i) : a \geq \min(y) - \textstyle\sum_{j \neq i} \max_{(x_i \leftarrow a)}(x_j)$$

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 4 *Let $X \cup \{y\}$ be a set variables and let* $\texttt{IS}(X, y, \mathcal{I}_{neq})$ *be a global sum constraint. IS is interval-consistent iff the following conditions hold:*

(1*b*) $\min(y) \geq \sum_{x_i \in X} \min(x_i)$
(2*b*) $\max(y) \leq \sum_{x_i \in X} \max(x_i)$

$$(3b)\ \forall x_i \in X : \min(x_i) \geq \min(y) - \sum_{x_j \in X - \{x_i\}} \max_{(x_i \leftarrow \min(x_i))}(x_j)$$

$$(4b)\ \forall x_i \in X : \max(x_i) \leq \max(y) - \sum_{x_j \in X - \{x_i\}} \min_{(x_i \leftarrow \max(x_i))}(x_j)$$

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 $\mathcal{I}_{neq}$ 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 general interval consistency filtering algorithm of constraint `IS` is given in Figure 1. This algorithm is started whenever bounds of variables in $\mathcal{X} \cup \{y\}$ are modified. Note that steps 1 and 2 are systematically performed when interval consistency on `IS` is achieved for the first time.
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. If a bounds of y have been modified, then interval consistency will be achieved on $SUM(X, y)$ with an algorithm derived from Proposition 1.
2. If a bound of some variable of X is modified, then interval consistency will be achieved on the conjunction of binary inequalities $\mathcal{I}_{neq} \cup \mathcal{D}_{om}$ with a shortest path algorithm.
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 $\underline{x_i} \in [\min(x_i), \max(x_i)]$ such that inequality (3b) of Proposition 4 holds. It follows from inequality (3b) that

$$\underline{x_i} = \min(y) - \sum_{x_j \in X - \{x_i\}} \max_{(x_i \leftarrow \underline{x_i})}(x_j) \quad (1)$$

The essential observation is that the value of $max(x_j)$ depends only on the upper–bounds of the variables x_k that belong to a shortest path from s to x_j.

So, if the fact of assigning a to x_i—i.e., reducing the upper–bound of x_i to a—introduces x_i in a shortest path from s to x_j, then the length of this shortest path (i.e., $d(s, x_j)$) is the greatest value that is consistent with $\underline{x_i}$ for the computation of the sum y. So,

$$\max_{(x_i \leftarrow \underline{x_i})}(x_j) = \min(d(s, x_j), \underline{x_i} + d(x_i, x_j)) \tag{2}$$

We are now in a position to prove the following nice property:

$$\max_{(x_i \leftarrow \underline{x_i})}(x_j) = \underline{x_i} + d(x_i, x_j) \tag{3}$$

Proof:
First note that $d(x_i, x_j)$ is always finite since there exists at least one path from x_i to x_j which goes through s. So we can distinguish two cases :

1. x_i will belong to a shortest path from s to x_j when x_i is set to the value of $\underline{x_i}$ we are searching for. Then, $\max_{(x_i \leftarrow \underline{x_i})}(x_j) = \underline{x_i} + d(x_i, x_j)$;
2. x_i will never belong to the shortest path from s to x_j, even when x_i is set to $\underline{x_i}$. That is to say, $\max(x_j)$ does not depend on x_i. So $\underline{x_i} = \min(x_i) = -d(x_i, s)$.
   Since $\underline{x_i} + d(x_i, s) = 0$ we can rewrite $\max_{(x_i \leftarrow \underline{x_i})}(x_j)$ in the following way : $\max_{(x_i \leftarrow \underline{x_i})}(x_j) = d(s, x_j) = \underline{x_i} + d(x_i, s) + d(s, x_j)$. It remains to show that $d(x_i, s) + d(s, x_j) = d(x_i, x_j)$, i.e., that s belongs to a shortest path from x_i to x_j.
   Since x_i does not belong to a shortest path from s to x_j, we have $d(s, x_j) < d(s, x_i) + d(x_i, x_j)$. Suppose now that there is no shortest from x_i to x_j which contains s. Then $d(x_i, x_j) < d(x_i, s) + d(s, x_j)$ and thus, $d(s, x_j) < d(s, x_i) + d(x_i, s) + d(s, x_j)$. However, when x_i is set to $\underline{x_i}$, $d(s, x_i) = -d(x_i, s)$ yielding the following inconsistency $d(s, x_j) < d(s, x_j)$.

⊙

Substituting equivalence 3 in Equation 1, yields the following linear equation that gives the value of $\underline{x_i}$, i.e., the new lower bound of x_i:

$$|X| * \underline{x_i} = \min(y) - \sum_{x_j \in X - \{x_i\}} d(x_i, x_j) \tag{4}$$

The point is that the greatest value of x_j which is consistent with $\underline{x_i}$ can be determined by computing $d(x_i, x_j)$ on the graph of reduced costs. No propagation 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 Figure 1.) This results from Theorem 2 which states that $-d(x_i, s)$ is the lower bound of $D^*(x_i)$.

The shortest distance between x_i to all x_j can be computed with Dijkstra's shortest path algorithm on the graph of reduced costs in $O(m + n \log n)$. Then, the computation of $\underline{x_i}$ can be achieved in $O(n)$ time. So, the total complexity of the algorithm is $O(m + n \log n) * n$.

In many cases, it is useless to compute all $d(x_i, x_j)$ for checking property (3b). Indeed, the process can be speeded up by checking whether $\min(x_i)$ is greater than the difference

$$\Delta = \frac{\min(y) - \sum_{x_j \in X - \{x_i\}} d(x_i, x_j)}{|X|} \tag{5}$$

"while" computing the $d(x_i, x_j)$. To achieve such a check the essential point is the approximation of the $d(x_i, x_j)$ by their lower–bound. Since $d(s, x_j) \leq d(s, x_i) + d(x_i, x_j)$ we have $d(s, x_j) - d(s, x_i) \leq d(x_i, x_j)$ and thus $\max(x_j) - \max(x_i) \leq d(x_i, x_j)$. So, an upper-bound of Δ can be computed by replacing $d(x_i, x_j)$ with $\max(x_j) - \max(x_i)$ in Equation 5. Then, each time a node is scanned in Dijkstra's shortest path algorithm, we have just to subtract

$$\frac{1}{|X|} \left(d(x_i, x_j) - (\max(x_j) - \max(x_i)) \right)$$

from this upper bound. The shortest path algorithm can thus be stopped as soon as inequality (3b) of Proposition 4 holds.

6 Discussion

It is also instructive to remark that our algorithm still works when the function to be optimized is of the form $y = \Sigma_{i=1}^{i=n} \alpha_i x_i$ where the α_i are a non-negative real number. To capture the exact contribution of each x_i in the sum when the α_i are different from the value 1, we need only introduce the coefficient of x_i in Equation 1:

$$\underline{x_i} = \frac{1}{\alpha_i} \left(\min(y) - \sum_{x_j \in X - \{x_i\}} \alpha_j \max_{(x_i \leftarrow \underline{x_i})} (x_j) \right) \tag{6}$$

In Section 1, we defined $\mathcal{I}_{neq}$ as the subset of binary inequalities that involve only variables occurring in the objective function $f(x)$. However, $\mathcal{I}_{neq}$ 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 3. 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

AMO93. Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows*. Prentice Hall, 1993. 388

BC94. N. Beldiceanu and E. Contejean. Introducing global constraints in chip. *Journal of Mathematical and Computer Modelling*, 20(12):97–123, 1994. 384

BESW93. J. Blazewicz, K. Ecker, G. Schmidt, and J. Weglarz. *Scheduling in Computer and Manufacturing Systems*. Springer Verlag, 1993. 385

CP94. J. Carlier and E. Pinson. A practical use of jackson's preemptive schedule for solving the job-shop problem. *Annals of Operations Research*, 26(12):269–287, 1994. 384

DKD97. Moshe Dror, Wieslaw Kubiak, and Paolo Dell'Olmo. Scheduling chains to minimize mean flow time. *Information Processing Letters*, 61(6):297–301, 28 March 1997. 385

DMP91. R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artificial Intelligence*, 49(1-3):61–95, January 1991. 386, 389, 390

HD91. Pascal Van Hentenryck and Yves Deville. The cardinality operator: A new logical connective for constraint logic programming. In *Proc. of ICLP'91*, pages 745–759, 1991. 384

HSD98. Pascal Van Hentenryck, Vijay Saraswat, and Yves Deville. Design, implementation, and evaluation of the constraint language cc(FD). *Journal of Logic Programming*, 37(1-3):139–164, October 1998. 388

Nui94. W. P. Nuijten. *Time and Resource Constrained Scheduling: A Constraint Satisfaction Approach*. PhD thesis, Eindhoven University of Technology, 1994. 384

Rég94. J-C. Régin. A filtering algorithm for constraints of difference in CSPs. In *Proc. of AAAI94*, pages 362–367, Seattle, Washington, 1994. 384

Rég96. J-C. Régin. Generalized arc consistency for global cardinality constraint. In *Proc. of AAAI-96*, pages 209–215, Portland, Oregon, 1996. 384

Sim96. H. Simonis. Problem classification scheme for finite domain constraint solving. In *CP96, Workshop on Constraint Programming Applications: An Inventory and Taxonomy*, pages 1–26, Cambridge, MA, USA, 1996. 384

Tar83. Robert E. Tarjan. *Data Structures and Network Algorithms*. CBMS 44 SIAM, 1983. 388, 390

VDT92. P. Van Hentenryck, Yves Deville, and Choh-Man Teng. A generic arc-consistency algorithm and its specializations. *Artificial Intelligence*, 57(2–3):291–321, October 1992. 389