Softening Gcc and Regular with preferences

Jean-Philippe Metivier, Patrice Boizumault, Samir Loudni

Softening Gcc and Regular with preferences

Jean-Philippe Métivier Patrice Boizumault Samir Loudni
GREYC (UMR 6072) — University of Caen
Campus Côte de Nacre
Bd du Maréchal juin - BP 5186
14032 Caen CEDEX — France
{jmetivie,patrice.boizumault}@info.unicaen.fr loudni@iutc3.unicaen.fr

ABSTRACT

In this paper, we present the soft global constraints Σ-gcc and Σ-regular which are the soft versions with preferences of well-known global constraints Gcc and Regular. For each of them, we introduce a new violation based semantic which takes into account preferences and propose algorithms to enforce hyperarc consistency in polynomial time, making use of flow theory.

1. INTRODUCTION

Many real-life problems are over-constrained since there exists no solution satisfying all the constraints. In this situation, it is natural to allow certain constraints, the soft ones, to be violated.

Soft versions of some well-known global constraints, such AllDifferent, Gcc and Regular have been recently introduced [8, 9]. But these soft global constraints do not take into account preferences.

When expressing preferences, each constraint has a weight reflecting its importance (amount of violation or "cost to pay" if the constraint is not satisfied). For an instantiation, the amount of violation is the sum of the weights of all unsatisfied constraints. A first softening of a global constraint with preferences has been proposed for AllDifferent in [3].

In this paper, we present the soft global constraints Σ-gcc and Σ-regular which are the soft versions, with preferences, of the global constraints Gcc and Regular. For each of these two constraints, we introduce a new violation based semantic which takes into account preferences between violations and propose algorithms to enforce hyperarc consistency in polynomial time, making use of flow theory.

In Section 2, we introduce our *decomposition based* semantic of violation for Σ-gcc and show how hyperarc consistency can be enforced in polynomial time. In Section 3, we present our *value based* semantic of violation for Σ-regular, and show how hyperarc consistency can be enforced in polynomial time thanks to path computations in directed graph. Finally, we conclude and draw some future works.

SAC'09 March 8-12, 2009, Honolulu, Hawaii,U.S.A.

2. SOFT GLOBAL CONSTRAINT Σ-GCC

2.1 The Global Cardinality Constraint

JC. Régin has introduced in [5] the Global Cardinality Constraint (Gcc). This constraint defined on a set of n variables specifies, for each value in the union of theirs domains ($Doms$), bounds on the maximal (u_j) and minimal (l_j) number of times values v_j can occur in a solution.

DEFINITION 1. *(Gcc, [5]) Let $\mathcal{X}$ be a set of n variables and D_i be the domain of the variable X_i, let $Doms = \cup_{i \in \mathcal{X}} D_i$. Let $l_j, u_j \in \mathbb{N}$ with $l_j \leqslant u_j$ for all $v_j \in Doms$.*

$$Gcc(\mathcal{X}, l, u) = \{(v_1,\ldots,v_n) \in D_1 \times \ldots \times D_n,\ s.t.\ \forall v_j \in Doms,\ l_j \leqslant |\ \{X_i \in \mathcal{X} \mid X_i = v_j\}\ | \leqslant u_j\}$$

THEOREM 1 ([5]). *Gcc$(\mathcal{X}, l, u)$ is hyperarc consistent iff there exits a feasible s–t flow of value n (where $n = |\ \mathcal{X}\ |$) in the network $\mathcal{N} = \{\mathcal{V}, \mathcal{A}\}$ with:*

$$\begin{array}{l}
\mathcal{V} = \mathcal{X} \cup Doms \cup \{s, t\} \\
\mathcal{A} = \mathcal{A}_s \cup \mathcal{A}_{\mathcal{X}} \cup \mathcal{A}_t \\
\mathcal{A}_s = \{(s, X_i) \mid i \in \{1, \ldots, n\}\}, \\
\mathcal{A}_{\mathcal{X}} = \{(X_i, v_j) \mid v_j \in D_i, i \in \{1, \ldots, n\}\}, \\
\mathcal{A}_t = \{(v_j, t) \mid v_j \in Doms\}, \\
\textit{demand and capacity functions:} \\
\forall a \in \mathcal{A}_s,\ d(a) = 1 \textit{ and } c(a) = 1 \\
\forall a \in \mathcal{A}_{\mathcal{X}},\ d(a) = 0 \textit{ and } c(a) = 1 \\
\forall a \in \mathcal{A}_t,\ d(a) = l_j \textit{ and } c(a) = u_j
\end{array}$$

Consistency-check can be performed in $O(n \times m)$ (where $m = |\ \mathcal{A}\ |$). To maintain hyperarc consistency, all arcs (and then values) which do not belong to any feasible flow are removed thanks to computation of strongly connected components in $O(n+m)$ (see [7] for more details).

EXAMPLE 1. *Consider the following CSP:*

$D_1 = D_2 = \{1, 2\}, D_3 = \{1\}, D_4 = \{2\},$ *and* $D_5 = \{2, 3\}$
$Gcc([X_1, X_2, X_3, X_4, X_5], [1, 1, 0], [2, 2, 1])$.

The network associated to this Gcc constraint is described in Figure 1. $(1, 2, 1, 2, 3)$ *and* $(2, 1, 1, 2, 3)$ *are the two solutions of this CSP.*

2.2 Decomposition Based Semantic for Σ-GCC

2.2.1 Gcc as a meta-constraint

Gcc can be considered as a meta-constraint and decomposed into a set of atleast constraints enforcing the lower bound for each $v_j \in Doms$, and a set of atmost constraints enforcing the upper bound for each $v_j \in Doms$.

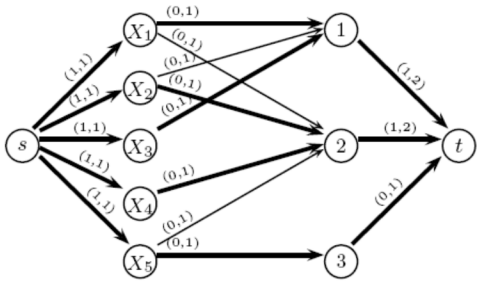

Figure 1: Network representation for the Gcc constraint presented in Example 1, bold arcs represent a feasible flow of value 5.

DEFINITION 2 ([2]). *Let $v_j \in Doms$, let l_j and u_j their lower and upper bounds,*

$$\texttt{atleast}(\mathcal{X}, v_j, l_j) \equiv |\{X_i \in \mathcal{X} \mid X_i = v_j\}| \geqslant l_j$$
$$\texttt{atmost}(\mathcal{X}, v_j, u_j) \equiv |\{X_i \in \mathcal{X} \mid X_i = v_j\}| \leqslant u_j$$

Then, Gcc can be defined as follows:

$$\texttt{Gcc}(\mathcal{X}, l, u) \equiv \bigwedge_{v_j \in Doms} (\texttt{atleast}(\mathcal{X}, v_j, l_j) \wedge \texttt{atmost}(\mathcal{X}, v_j, u_j))$$

2.2.2 Decomposition Based Semantic

To each value $v_j \in Doms$, we associate a *shortage function* ($s(\mathcal{X}, v_j)$) measuring the number of missing assignments of v_j to satisfy $\texttt{atleast}(\mathcal{X}, v_j, l_j)$, and an *excess function* ($e(\mathcal{X}, v_j)$) measuring the number of assignments of v_j in excess to satisfy $\texttt{atmost}(\mathcal{X}, v_j, u_j)$ [9]:

$$s(\mathcal{X}, v_j) = max(0, l_j - |\{X_i \mid X_i = v_j\}|)$$
$$e(\mathcal{X}, v_j) = max(0, |\{X_i \mid X_i = v_j\}| - u_j)$$

To express preferences on the minimal/maximal requirements on each value v_j, we associate to each l_j (resp. u_j) a weight $\varphi_j^{atleast}$ (resp. φ_j^{atmost}) denoting the amount of violation to pay if the requirement is not satisfied. We can define the violation cost of the constraint $\texttt{atleast}(\mathcal{X}, v_j, l_j)$ as follows:

$$weight(\texttt{atleast}(\mathcal{X}, v_j, l_j)) = \varphi_j^{atleast} \times s(\mathcal{X}, v_j)$$

And respectively for the constraint $\texttt{atmost}(\mathcal{X}, v_j, u_j)$:

$$weight(\texttt{atmost}(\mathcal{X}, v_j, u_j)) = \varphi_j^{atmost} \times e(\mathcal{X}, v_j)$$

Let C_{dec} be the decomposition of a Gcc constraint into a set of atleast and atmost constraints. The decomposition based semantic μ_{dec} computes the weighted sum of $\texttt{atleast}(\mathcal{X}, v_j, l_j)$ and $\texttt{atmost}(\mathcal{X}, v_j, u_j)$ constraints in C_{dec} that are violated.

DEFINITION 3. *(Decomposition Based Violation Measure)*

$$\mu_{dec}(\mathcal{X}) = \sum_{v_j \in Doms} \varphi_j^{atleast} \times s(\mathcal{X}, v_j) \; + \; \varphi_j^{atmost} \times e(\mathcal{X}, v_j)$$

DEFINITION 4. *(Consistency of Σ-gcc) Let z be a cost variable with domain D_z, $\Sigma\texttt{-gcc}(\mathcal{X}, l, u, z, \mu_{dec})$ is hyper-arc consistent iff there exists a complete instanciation $\mathcal{A}$ s.t. $\mu_{dec}(\mathcal{A}) \leqslant max(D_z)$.*

Employee	Shifts
Mick	*M*
John	*M*
James	*M, A*
Patty	*M, A*
Julia	*M*
Helen	*A, N*

Table 1: Availabilities

Shifts	Bounds		Costs	
	l	u	$\varphi^{atleast}$	φ^{atmost}
M	1	2	4	2
A	3	4	1	2
N	2	2	3	4

Table 2: Bounds and costs

2.2.3 Example

Consider the following planning problem that consists in building timetables for a group of six employees over a period of a week (more precisely from Monday to Thursday). There are three basic shifts in a day, namely Morning (M), Afternoon (A) and Night (N).

Each shift requires a minimal number of employees to be operational, and a maximal number of employees to be profitable (see Table 2). Each employee provides the set of its availabilities (see Table 1).

This problem can be easily modeled as a CSP: to each employee i, we associate a variable X_i whose domain D_i contains the shift requests of employee i; bounds on shift requirements can be expressed using a Gcc constraint.

$D_1 = D_2 = \{M\}, D_3 = D_4 = \{M, A\}, D_5 = \{M\}, D_6 = \{A, N\}$
$\texttt{Gcc}([X_1, X_2, X_3, X_4, X_5, X_6], [1, 3, 2], [2, 4, 2])$.

But, shift requests of employees are frequently too restrictive in order to get a solution. The violation of the minimal/maximal requirements for a shift may not have the same importance w.r.t. to the profits. In fact, for a Morning shift, it is more important to respect its lower bound (l_i) rather than its upper bound (u_i), since any violation of l_i may have a great impact on the other shifts (i.e., the production may be delayed). In contrast, for a Night shift, one prefers to respect its upper bound (u_i) rather than its lower bound because having one more worker could be considered as too expensive. Table 2 summarizes, for each shift, the values of l_i and u_i and their violation costs.

This over-constrained problem can be now expressed as a CSP with preferences (on shift requirements) using a Σ-gcc constraint.

$D_1 = D_2 = \{M\}, D_3 = D_4 = \{M, A\}, D_5 = \{M\}, D_6 = \{A, N\}$
$D_z = [0..6] \Sigma\texttt{-gcc}([X_1, X_2, X_3, X_4, X_5, X_6], [1, 3, 2], [2, 4, 2], z, \mu_{dec})$.

The solution (M,M,A,A,M,N), whose violation cost is equal to $1{\times}\varphi_M^{atmost}{+}1{\times}\varphi_A^{atleast}{+}1{\times}\varphi_N^{atleast}{=}2{+}4{=}6$, satisfies the Σ-gcc as its cost is less or equal to $max(D_z) = 6$. Figure 2 shows the network associated to this example.

2.3 Network Representation for Σ-GCC

As for Gcc, Σ-gcc can be modeled by a network. From the initial network $\mathcal{N}$ associated to Gcc (see Section 2.1), WJ. van Hoeve and *al.* have defined the network $\mathcal{N}_{soft}$ associated to soft-gcc [9]. *Violation arcs* are added in order to model shortage and excess functions:

- Shortage violation arcs, $\mathcal{A}_{shortage}{=}\{(s, v_j) \mid v_j{\in}Doms\}$, are used to model $\texttt{atleast}(\mathcal{X}, v_j, l_j)$ constraints. For each shortage violation arc $a{=}(s, v_j)$, its demand $d(a){=}0$, its capacity $c(a){=}l_j$, and its weight $w(a){=}1$.
- Excess violations arcs, $\mathcal{A}_{excess}{=}\{(v_j, t) \mid v_j{\in}Doms\}$, are used to model $\texttt{atmost}(\mathcal{X}, v_j, u_j)$ constraints. For

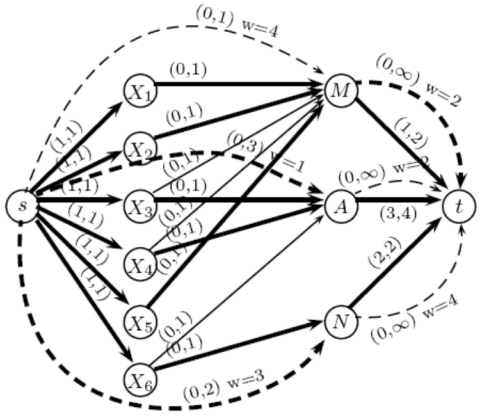

Figure 2: Network representation for the Σ-gcc presented in section 2.2.3, bold arcs represent a maximal flow of weight 6.

each excess violation arc $a = (v_j, t)$, its demand $d(a)$=0, its capacity $c(a) = \infty$, and its weight $w(a) = 1$.

Now, we can define the network $\mathcal{N}_\Sigma$ associated to a Σ-gcc constraint. For the decomposition based semantic introduced in Section 2.2.2, the cost associated to the violation of a lower bound (l_i) is not necessarily the same to the one associated to the violation of an upper bound (u_i):

- Let $v_j \in Doms$, and $\varphi_j^{atleast}$ be the weight associated to the violation of the lower bound l_j for v_j. For each shortage violation arc $a = (s, v_j)$, its demand $d(a) = 0$, its capacity $c(a) = l_j$, and its weight $w(a) = \varphi_j^{atleast}$.
- Let $v_j \in Doms$, and φ_j^{atmost} be the weight associated to the violation of the upper bound u_j for v_j. For each excess violation arc $a = (v_j, t)$, its demand $d(a) = 0$, its capacity $c(a) = \infty$, and its weight $w(a) = \varphi_j^{atmost}$.

2.4 Consistency-Check

Let z be a cost variable that represents the allowed amount of violation of a Σ-gcc constraint. To check the consistency of a Σ-gcc constraint we use the following corollary:

COROLLARY 1. *The constraint Σ-gcc($\mathcal{X}, l, u, z, \mu_{dec}$) is hyperarc consistent if and only if there exists an integer s–t flow f of value n in $\mathcal{N}_\Sigma$ with $weight(f) \leqslant max(D_z)$.*

PROOF. To an integer s–t flow f of value n in $\mathcal{N}_\Sigma$ we associate the assignment $X_i = v_j$ for all arcs $a = (X_i, v_j) \in \mathcal{A}_\mathcal{X}$ with $f(a) = 1$. By construction, the cost function measures the decomposition based violation cost. □

Consistency-Check can be performed thanks to a Ford & Fulkerson algorithm in $O(max(n, \sum_{v_j \in Doms} l_j) \times (m + n \times log(n)))$ [1].

2.5 Maintaining hyperarc consistency

To maintain hyperarc consistency, we use the following corollary:

COROLLARY 2. *The constraint Σ-gcc is hyperarc consistent if and only if for every arc $a \in \mathcal{A}_\mathcal{X}$ there exists an integer s–t flow f of value $max(n, \sum_{v_j \in Doms} l_j)$ in $\mathcal{N}_\Sigma$ with $f(a) = 1$ and $weight(f) \leqslant max(D_z)$.*

Algorithm 1: Filtering algorithm for Σ-GCC.

```
Compute f a minimal weight flow of value max(n, Σ_{v_j∈Doms} l_j)
min(D_z) ← weight(f)
if D_z = ∅ then
    return inconsistent
else
    foreach vertex v∈N^f the residual graph do
        weight(s→v) ← mininal distance from s to v
        weight(v→t) ← mininal distance from v to t
    foreach X_i∈X do
        foreach v_j∈D_i do
            if (X_i,v_j) ∈ A then
                wp ← weight(t → X_i) + weight(v_j → t)
                if weight(f)+wp > max(D_z) then
                    Remove v_j from D_i
        if D_i = ∅ then return inconsistent
    return consistent
```

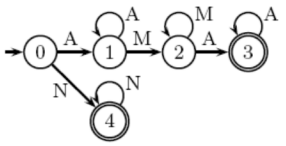

Figure 3: DFA associated to the timetabling

The Σ-gcc of Example 2 can be made hyperarc consistent by removing value M from D_3 and D_4 and value A from D_6.

We can use the same technique as cost-gcc [6] to compute the weight of a flow using a specific arc.

For each arc (X_i, v_j), we compute the weight of rerouting the flow in order to use this arc thanks to the search of the minimal weighted cycle ($v_j \to t \to s \to X_i \to v_j$). Then, we add this weight to the initial flow. If this sum is greater than $max(D_z)$, we have to remove the value v_j from D_i (see Algorithm 1).

Complexity of the filtering is $O((n+d) \times (m + n \times log(n)))$.

3. A SOFT REGULAR: Σ-REGULAR

3.1 The Regular Constraint

3.1.1 Definition and Example

Let $\mathcal{M} = (Q, \Sigma, \delta, q_0, F)$ be a deterministic finite automaton (DFA) where Q a finite set of states, Σ an alphabet, δ a transition function defined on $Q \times \Sigma \to Q$, q_0 the initial state and $F \subseteq Q$ the set of final states. Given an input word, the automaton starts in the initial state q_0 and processes the word one symbol at a time, applying the transition function δ at each step. The word is accepted if and only if the last state reached belongs to F. All words accepted by a DFA belong to the language recognized by the DFA noted $L(\mathcal{M})$.

For the automaton described in Figure 3, words $AAMA$ and NNN belong to $L(\mathcal{M})$ but it is not the case for $NNAN$.

The Regular constraint states that a word, represented by a sequence of n variables $X_1, X_2, \ldots, X_n$, has to be accepted by a given DFA.

DEFINITION 5. *(Regular [4]) Let $\mathcal{M} = (Q, \Sigma, \delta, q_0, F)$ be*

a DFA and $\mathcal{X} = X_1, X_2, \ldots, X_n$ *be a sequence of* n *variables with respective domains* $D_1, D_2, \ldots, D_n \subset \Sigma$. *Then*

$$\texttt{Regular}(\mathcal{X},\mathcal{M}) = \{(v_1, \ldots, v_n) \in D_1 \times \ldots \times D_n \mid (v_1, \ldots, v_n) \in L(\mathcal{M})\}.$$

3.1.2 Graph associated to a `Regular` constraint

`Regular` can be modeled with a directed graph.

THEOREM 2. *([4]) Let* $\mathcal{M} = (Q, \Sigma, \delta, q_0, F)$ *be a DFA. A constraint* $\texttt{Regular}(\mathcal{X}, \mathcal{M})$ *is hyperarc consistent iff there exists an* s–t *path in the graph* $\mathcal{G} = \{\mathcal{V}, \mathcal{A}\}$ *defined as follows:*

$\mathcal{V} = \mathcal{V}_1 \cup \mathcal{V}_2 \cup \cdots \cup \mathcal{V}_n \cup \{s, t\}$
where $\mathcal{V}_i = \{q_k^i \mid q_k \in Q\}$ *for* $i = 1, \ldots, n+1$
and $\mathcal{A} = \mathcal{A}_s \cup \mathcal{A}_1 \cup \mathcal{A}_2 \cup \cdots \cup \mathcal{A}_n \cup \mathcal{A}_t$
where
$\mathcal{A}_s = \{(s, q_0)\}$
$\mathcal{A}_i = \{(q_k^i, q_l^{i+1}, v_j) \mid v_j \in D_i, \delta(q_k, v_j) = q_l\}$ *for* $i = 1, \ldots, n$
$\mathcal{A}_t = \{(q_k^{n+1}, t) \mid q_k \in F)\}$

3.1.3 Consistency-Check

The Consistency-Check can be efficiently performed by computing an s–t path thanks to a breadth-first search in $O(m)$, where m is the number of arcs in $\mathcal{G}$ (with $m \leqslant n \times |Q| \times |\Sigma|$).

3.1.4 Filtering

Filtering can be performed in two steps [4]:

1. For each vertex (different from the target t) with no outgoing arc, all incoming arcs are removed. For each vertex (different from the sink s) with no incoming arc, all outgoing arcs are removed.
2. If there exists no arc (q_k^i, q_l^{i+1}, v_j) between q_k^i and q_l^{i+1}, then value v_j can be removed from domain D_i.

So, filtering consists in checking each arc and it can be performed in $O(m)$.

3.2 Distance Based Semantic for Σ-REGULAR

In this section, we introduce a new value based semantic taking into account the distance between expected and unexpected values for a transition.

DEFINITION 6. *(Expected value) Value* $v_j \in \Sigma$ *is said to be expected for a couple of states* (q_k, q_l) *if and only if there exists a transition from state* q_k *to state* q_l *using symbol* v_j. $EV(q_k, q_l) = \{v \in \Sigma \mid \delta(q_k, v) = q_l\}$ *will denote the set of expected values for* (q_k, q_l) .

DEFINITION 7. *(Distance) Let* (q_k, q_l) *be a couple of states, value* $v_j \in EV(q_k, q_l)$ *and value* $v_i \in \Sigma$.

- $distance(v_j, v_i) = 0$ *if* $v_i \in EV(q_k, q_l)$
- $distance(v_j, v_i) = \varphi_{i,j}$ *if* $v_i \in \Sigma \setminus EV(q_k, q_l)$

where each $\varphi_{i,j}$ *corresponds to the cost for having performed a transition from* q_k *to* q_l *using an unexpected value* v_i *instead of expected value* v_j.

Between two words, the value based distance, $V(W_a, W_b)$, is the sum of the distances between their corresponding symbols. The value based distance between a word and a language is the minimal value based distance between the word and any word of the language.

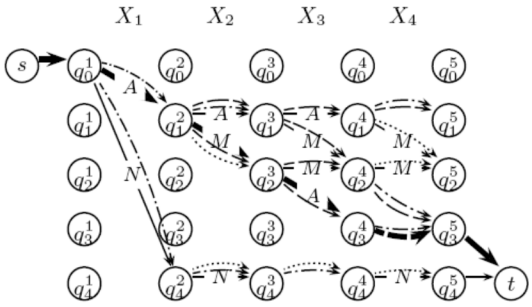

Figure 4: Graph for Σ-`regular`$(\mathcal{X}, \mathcal{M}, \mu_{val}, z)$ **with** $\mathcal{M}$ **the DFA presented in Figure 3. Dashed arcs represent a** $M \leftrightarrow A$ **move, Dotted one a** $M \leftrightarrow N$ **move and mixed ones a** $A \leftrightarrow N$**move. Bold arcs represent the solution** (A, M, A, M) **of cost 1.**

DEFINITION 8. *(Value Based Semantic Measure) For a* Σ-`regular`$(\mathcal{X}, \mathcal{M}, z, \mu_{val})$ *constraint, the value based violation measure is defined as:*

$$\mu_{val}(\mathcal{X}) = min_{W \in L(\mathcal{M})} V(\mathcal{X}, W)$$

DEFINITION 9. *(Consistency of* Σ-`regular`*) Let* z *be a cost variable with domain* D_z, Σ-`regular`$(\mathcal{X}, \mathcal{M}, z, \mu_{val})$ *is consistent iff there exists a complete instiation* $\mathcal{A}$ *such that* $\mu_{val}(\mathcal{A}) \leqslant max(D_z)$.

3.3 Graph Representation for Σ-`regular`

A Σ-`regular` constraint can be modeled as a graph $\mathcal{G}_\Sigma$. Let $\mathcal{G}$ be the valued graph associated to a `Regular` constraint (see 3.1.2). For each layer i, and each couple of states (q_k, q_l) linked by at least one transition, $\mathcal{A}_{i,k,l} = \{(q_k^i, q_l^{i+1}, v) \mid v \in D_i \setminus EV(q_k^i, q_l^{i+1})\}$ denotes the set of violation arcs associated to unexpected values for (q_k, q_l). Each violation arc (q_k^i, q_l^{i+1}, v) of each $\mathcal{A}_{i,k,l}$ is added to $\mathcal{G}$. Let v_j in $EV(q_k^i, q_l^{i+1})$; the valuation of (q_k^i, q_l^{i+1}, v) is $distance(v_j, v)$, in order that transitions performed using unexpected values induce extra-cost.

COROLLARY 3. *Let* z *be a cost variable with domain* D_z, *a constraint* Σ-`regular`$(\mathcal{X}, \mathcal{M}, z, \mu_{val})$ *is hyperarc consistent if and only if there exists an* s–t *path* p *in* $\mathcal{G}_\Sigma$ *such that* $weight(p) \leqslant max(D_z)$.

PROOF. To an s–t path in $\mathcal{G}_\Sigma$, we associate, to each variable X_i a value v_j for all arc (q_k^i, q_l^{i+1}) using the value v_j such that (q_k^i, q_l^{i+1}, v_j) belongs to p. By construction, the weight of p corresponds exactly to the value based measure of violation. □

Consider Figure 4 where $\mathcal{M}$ is the DFA depicted in Figure 3, $\mathcal{X} = \{X_1, X_2, X_3, X_4\}$, $D_1 = \{A, N\}$, $D_2 = \{M, A, N\}$, $D_3 = \{M, A\}$, $D_4 = \{M, N\}$. Violation arcs are added in dashed, dotted and mixed.

3.4 A timetabling example

Consider again the previous timetabling problem. The consecutive shifts of an employee have to respect the following rules:

- **Rule 1**: If an N shift is assigned to an employee for a particular day, this shift must be assigned to this employee for all the week.

Cost	M	A	N
M	-	1	4
A	1	-	2
N	4	2	-

Table 3: Shift moves and their costs.

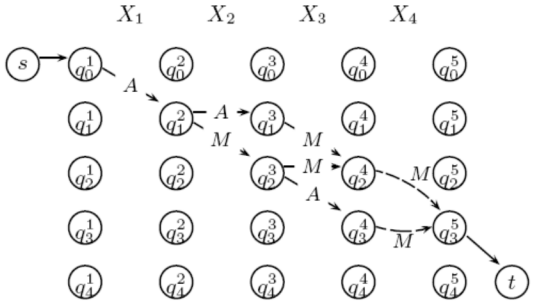

Figure 5: Filtered Network for Σ-regular($\mathcal{X}, \mathcal{M}, \mu_{val}$,z) in Figure 4 with the filtered domains $D_1 = \{A\}$, $D_2 = D_3 = \{A, M\}$ and $D_4 = \{M\}$.

- **Rule 2**: An employee having an A shift on Monday must have at least an M shift over the week. Moreover, an A shift must be assigned to this employee on Thursday.

These planning rules can be modeled as a DFA (see Figure 3) and can be expressed using a `Regular` constraint. The language recognized by the DFA is $NN^\star$ for **Rule 1** and $AA^\star MM^\star AA^\star$ for **Rule 2**.

As shift requests make the problem over-constrained, we have to soften the `Regular` constraint with preferences (i.e. Σ-`regular`) in order to get an acceptable solution. For an employee and a particular day, moving from one shift to another one will be allowed, but it will introduce a cost/weight relative to the non-respect of the two planning rules. For instance, moving an employee from M to N has a high cost compared to moving an employee from A to N. Moving an employee from M to A induces a low cost. Table 3 summarizes the cost of each move.

Σ-`regular` enables to express and quantify the violation of planning rules. Figure 4 shows its associated network.

3.5 Maintaining Hyperarc consistency

Filtering can be performed in three steps (see Figure 5):

- For each vertex with no outgoing arc, all incoming arcs are removed. For each vertex with no incoming arc, all outgoing arcs are removed.
- For each remaining arc a in the network, if there does not exist an s–t path using a with a weight lower or equal to $max(D_z)$, then arc a is removed.
- As for the second step of `Regular`'s filtering algorithm, for all k and l if a value v_j does not appear in at least one arc between q_k^i and q_l^{i+1}, then v_j can be removed from domain D_i.

For an acyclic graph, computing a shortest path having a particular arc can be performed in $O(m)$. If we use the same path computation as `cost-gcc` [6], filtering can be performed in O(m).

Algorithm 2: Filtering algorithm of Σ-REGULAR

```
foreach vertex v∈G_Σ do
    weight(s→v) ← minimal distance from s to v.
    weight(t→v) ← minimal distance from t to v (by
    reversing arcs).
min(D_z) ← weight(s→t)
if D_z = ∅ then
    return inconsistent
else
    foreach X_i ∈ X do
        foreach a ∈ A in layer A_i with a = (u, v) do
            wp←weight(s→u)+weight(a)+weight(v→t)
            if wp>max(D_z) then remove a
        update D_{X_i}
        if D_{X_i} = ∅ then return inconsistent
    return consistent
```

4. CONCLUSIONS

In this paper we have presented the soft global constraints Σ-`gcc` and Σ-`regular` which are the soft versions with preferences of the global constraints `Gcc` and `Regular`. For each of these two constraints, we have introduced a new violation semantic and proposed algorithms to enforce hyper-arc consistency in polynomial time, thanks to a network modelisation and flow algorithms.

In future works we want to improved Σ-`gcc` by using the matching based modeling proposed by A. Zanarini and *al.* in [11]. Many problems involve several global constraints which share a subset of variables. In this case, we want to study whether these interactions between global constraints can be used to compute better lower bounds.

5. REFERENCES

[1] L.R. Ford and D.R. Fulkerson. *Flow in Networks*. Princeton University Press, 1962.

[2] P. Van Hentenryck, H. Simonis, and M. Dincbas. Constraint satisfaction using constraint logic programming. *Artif. Intell.*, 58(1-3):113–159, 1992.

[3] J.P. Métivier, P. Boizumault, and S. Loudni. Σ-AllDifferent: Softening alldifferent in weighted csps. In *ICTAI (1)*, pages 223–230. IEEE Computer Society, 2007.

[4] G. Pesant. A regular language membership constraint for finite sequences of variables. In Wallace [10], pages 482–495.

[5] J.C. Régin. Generalized arc consistency for global cardinality constraint. In *AAAI/IAAI, Vol. 1*, pages 209–215, 1996.

[6] J.C. Régin. Arc consistency for global cardinality constraints with costs. In J. Jaffar, editor, *CP*, volume LNCS 1713, pages 390–404. Springer, 1999.

[7] R.E. Tarjan. Depth-first search and linear graph algorithms. *SIAM J. Comput.*, 1(2):146–160, 1972.

[8] W.J. van Hoeve. A hyper-arc consistency algorithm for the soft alldifferent constraint. In Wallace [10], pages 679–689.

[9] W.J. van Hoeve, G. Pesant, and L.M. Rousseau. On global warming: Flow-based soft global constraints. *J. Heuristics*, 12(4-5):347–373, 2006.

[10] M. Wallace, editor. *CP 2004*, volume LNCS 3258. Springer, 2004.

[11] A. Zanarini, M. Milano, and G. Pesant. Improved algorithm for the soft global cardinality constraint. In J. Christopher Beck and Barbara M. Smith, editors, *CPAIOR*, volume 3990 of *LNCS*, pages 288–299. Springer, 2006.