wjert, 2022, Vol. 8, Issue 1, 172-179. *Review Article* **ISSN 2454-695X**

World Journal of Engineering Research and Technology

WJERT

www.wjert.org **SJIF Impact Factor: 5.924**

GRAPH REPRESENTAION FOR THE VARIABLE BASED SOFT SAME CONSTRAINT

[*1]M. Devika and [2]Dr. G. Shobha Latha

[1]Research Scholar, Department of Mathematics, Sri Krishna Devaraya University, Anantapuramu, A. P. India.

[2]Professor, Department of Mathematics, Sri Krishna Devaraya University, Anantapuramu, A. P. India.

Article Received on 29/10/2021 **Article Revised on 19/11/2021** **Article Accepted on 09/12/2021**

***Corresponding Author**
M. Devika
Research Scholar, Department of Mathematics, Sri Krishna Devaraya University, Anantapuramu, A. P. India.

ABSTRACT

In our propagation algorithms we have applied techniques from flow theory to compute an initial solution and to make the soft same constraint hyper-arc consistent. The same constraint is defined on two sequences of variables and states that the variables in one sequence use the same values as the variables in the other sequence. The same constraint can be applied to rostering problems where we need to assign two types of personnel to each other.

KEYWORDS: Graph representation, Soft same constraint.

1. INTRODUCTION

Many real-life problems are over-constrained. In personnel reostering problems for example, people often have conflicting preferences. To such problems there does not exist a feasible solution that respects all preferences. However, we still want to find some solution, preferably one that minimizes the total number of conflicts. In case of the personnel rostering example, we may want to construct a roster in which the number of respected preferences is equally spread among the employees.

In constraint programming, we seek for an (optimal) feasible solution to a given problem. Hence, we cannot apply constraint programming directly to over-constrained problems, because it finds no solutions. As a remedy there have been proposed several methods. Most of these methods introduce so called soft constraints that are allowed to be violated. Constraints that are not allowed to be violated are called hard constraints. Most methods then try to find a solution that minimizes the number of violated constraints, or some other measure of constraint violation.

Global constraints are often key elements in successfully modeling and solving real-life applications with constraint programming. For many soft global constraints, however, no efficient propagation algorithms were available, up to very recently. Moreover, it was stated that

"Active solving of global constraints is only applicable on the assumption that the constraint must be satisfied in any solution. Soft constraints may be violated in an admissible solution, and cannot therefore be handled by the usual techniques for global constraints". (quoted from Wallace, (Caseau and Puget 2003). In this chapter we show that "usual techniques for global constraints" can be used to handle soft constraints. In particular, we apply techniques that were used earlier to handle the weighted all different constraint.

We distinguish two main objectives with respect to soft global constraints; useful violation measures and efficient propagation algorithms. Both issues will be addressed in this chapter, depending heavily on a technique from operations research: flow theory.

In many cases we can represent a solution to a global constraint as a property in some graph representation of the constraint. For example, a solution to the all different constraint corresponds to a matching in the associated value graph. There exists a large class of such global constraints, see for example (Beldiceanu 2000) for a collection. In this chapter, we focus on global constraints for which a solution can be representing by a flow in a graph.

Our method adds violation arcs to the graph representation of a global constraint. To these arcs we assign a cost, corresponding to some violation measure of the constraint. Each tuple in the constraint, the corresponding flow does not use any violation arc, and the cost is 0. If the tuple does not satisfy the constraint, the corresponding flow must use violation arcs, whose costs represent the cost of violation of this tuple.

This approach allows us to define and implement useful violation measures for soft global constraints. Moreover, we present an efficient generic propagation algorithm for soft global constraints, making use of flow theory. We apply our method to several global constraints that are well-known to the constraint programming community: the all different constraint, the global cardinality constraint, the regular constraint and the same constraint, which will be defined when considered in this chapter. To each of these global constraints we apply several violation measures, some of which are newly introduced.

We give an overview of related literature. Then our method to soften global constraints is presented. We first discuss the general concepts of constraint softening and violation measures. Then we describe the addition of violation arcs to the graph representation and present the generic hyper-arc consistency propagation algorithm.

The four global constraints mentioned above: the all different constraint, the global cardinality constraint, the same constraint and the regular constraint. For each constraint we introduce useful violation measures and the corresponding graph representations. We also analyze the corresponding propagation algorithms to achieve hyper arc consistency.

We propose to use soft global constraints to aggregate the costs of violation of different soft constraints.

2. GRAPH REPRESENTATION

A solution to the all different constraint corresponds to a matching in the corresponding value graph. We can give an equivalent representation in terms of flows as follows see also (Regin 1994,1996,1999a,1999b).

Theorem.1: A solution to all different $(x_1,\ldots,x_n)$ corresponds to an integer feasible s-t flow of value n in the. digraph $A = (V, A)$ with vertex set

$V = X \cup D_X \cup \{s, t\}$

and arc set $A = A_s \cup A_x \cup A_t$

Where $A_s = \{(s, x_i) \mid i \in \{1,\ldots,n\}\}$,

$A_x = \{(x_i, d) \mid d \in D_i, i \in \{1,\ldots,n\}\}$,

$A_t = \{(d, t) \setminus d \in D_x\}$,

with capacity function $c(a) = 1$ *for all* $a \in A$.

Proof: With an integer feasible s-t flow f of value n in A we associate the assignment $x_i = d$ for all arcs $a = (x_i, d) \in A_X$ with $f(a) = 1$. Because $c(a) = 1$ for all $a \in A_s \cup A_t$, this is indeed a solution to the all different constraint. As value (*f*) = *n*, all variables have been assigned a value. Similarly, each solution to the all different gives rise to a corresponding appropriate flow in A.

Example.1: Consider again the CSP. In Figure represents the corresponding graph representation of the all different constraint is presented.

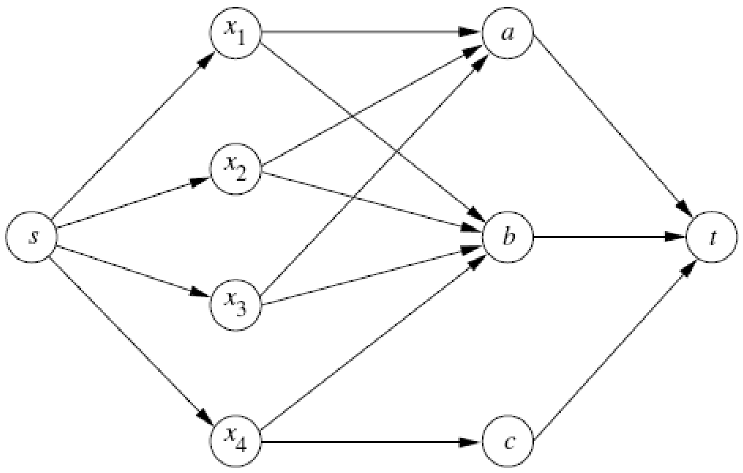

Figure 1: Graph representation for the all different constraint. For all arcs the capacity is 1.

We can recognize the value graph of *X* in *A*, being the subgraph on *X* and Dx. An integer feasible flow *f* of value *n* in *A* corresponds to a matching *M* in the value graph as follows:

$f(a) = 1 \Leftrightarrow a \in M$ for all $a \in Ax$

Corresponds to the subgraph on *X* and Dx of the residual graph A_f.

3. SOFT SAME CONSTRAINT

Definition 1

The same constraint is defined on two sequences of variables and states that the variables in one sequence use the same values as the variables in the other sequence. The constraint was introduced by (Beldiceanu 2000). One can also view the same constraint as demanding that one sequence is a permutation of the other. A hyper-arc consistency algorithm for the same constraint was presented by (Beldiceanu, Katriel and Thiel 2004b), making use of flow theory. The same constraint can be applied to rostering problems where we need to assign two types of personnel to each other. An example is the assignment of the same number of doctors and nurses on a particular date.

Definition 2 (Same constraint). Let X=$x_1,\ldots,x_n$ and Y=$y_1,\ldots,y_n$ and be sequences of variables with respective finite domains $D_1,\ldots\ldots,D_n$ and $D'_{1},\ldots, D'_{n}$. Then

$$\mathtt{same}(X,Y) = \left\{ (d_1,\ldots,d_n,d'_1,\ldots,d'_n) \mid d_i \in D_i, d'_i \in D'_i, \bigcup_{i=1}^{n}\{d_i\} = \bigcup_{i=1}^{n}\{d'_i\} \right\}.$$

Note that in the above definition

$\bigcup_{i=1}^{n}\{d_i\}$ and $\bigcup_{i=1}^{n}\{d'_i\}$

are families, in which elements may occur more than once.

To the same constraint we apply the variable based violation measure μ_{var}. Denote the symmetric difference of two sets S and T by SΔT, i.e. SΔT = (S\T) $\cup$ (T\S). For same (X,Y) we have.

$$\mu_{\text{var}}(X,Y) = \left| \left(\bigcup_{i=1}^{n}\{x_i\} \right) \Delta \left(\bigcup_{i=1}^{n}\{y_i\} \right) \right| /2.$$

Example. 2: Consider the following over constrained CSP.

$x_1 \in \{a,b,c\}$, $x_2 \in \{c,d,e\}$, $x_3 \in \{c,d,e\}$,

$y_1 \in \{a,b\}$, $y_2 \in \{a,b\}$, $y_3 \in \{c,d\}$,

same (x_1, x_2, x_3, y_1, y_2, y_3)

we have μ_{var} (a,c,c,a,b,c) = 1 because {a,c,c} Δ {a,b,c} = {b,c} and $|\{b,c\}|/2 = 1$.

4. GRAPH REPRESENTAION

A graph representation for the same constraint was given by (Beldiceanu, Ka- triel and Thiel 2004b).

Theorem.2: (Beldiceanu et al. 2004b) A solution to same (*X*, *Y*) corresponds to an integer feasible s-t flow of value n in the digraph S = (V, A) with vertex set

$V = X \cup (D_x \cap D'_y) \cup Y \cup \{s, t\}$

and arc set $A = A_s \cup A_x \cup A_y \cup A_t$,

Where

$$\begin{aligned} A_s &= \{(s, x_i) \mid i \in \{1, \ldots, n\}\}, \\ A_X &= \{(x_i, d) \mid d \in D_i \cap D_Y, i \in \{1, \ldots, n\}\}, \\ A_Y &= \{(d, y_i) \mid d \in D_i' \cap D_X, i \in \{1, \ldots, n\}\}, \\ A_t &= \{(y_i, t) \mid i \in \{1, \ldots, n\}\}, \end{aligned}$$

with capacity function c(a) = 1 for all a $\in$ *A.*

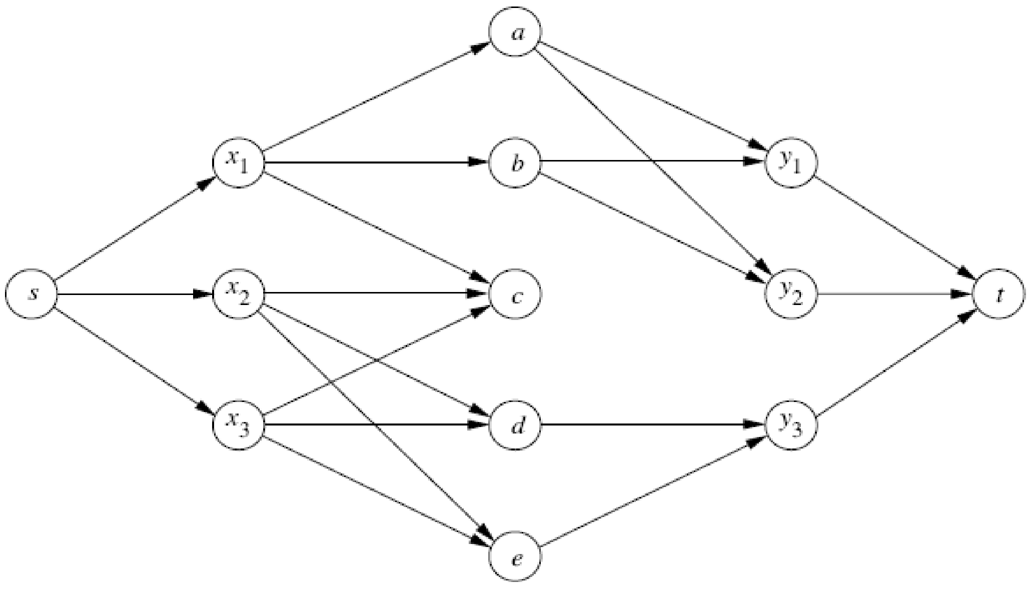

Figure 1: Graph representation for the same constraint. For all arcs the capacity is 1.

5. VARIABLE-BASED VIOLATION MEASURE

To the graph S of we add the arc sets

$$\begin{aligned} \tilde{A}_X &= \{(x_i, d) \mid d \in D_Y \setminus D_i, i = 1, \ldots, n\}, \text{ and} \\ \tilde{A}_Y &= \{(d, y_i) \mid d \in D_X \setminus D_i', i = 1, \ldots, n\} \end{aligned}$$

with capacity c(a) = n for all arcs $a \in \tilde{A}_X \cup \tilde{A}_Y$. As before, we apply a cost function $w : A \cup \tilde{A}_X \cup \tilde{A}_Y \rightarrow \{0,1\}$, where

$$w(a) = \begin{cases} 1 \text{ if } a \in \tilde{A}_X \cup \tilde{A}_Y, \\ 0 \text{ otherwise.} \end{cases}$$

Let the resulting digraph be denoted by S_{var}.

Example: The corresponding graph representation of the variable-based soft same constraint is presented.

Corollary : The constraint soft same (X,Y,z,μ_{var}) is hyper-arc consistent if and only if.

i) for every arc $a \in Ax \cup Ay$ there exists a feasible s-t flow f of value n in S_{var} with f(a) = 1 and weight (f) $\leq$ max Dz, and ii)min Dz $\geq$ weight (f) for a minimum-weight s-t flow f of value n in S_{var}.

Proof: An assignment x_i = d corresponds to the arc a = (x_i, d) with f(a) = 1. By construction, all variables need to be assigned to a value and the cost function exactly measures the variable-based cost of violation.

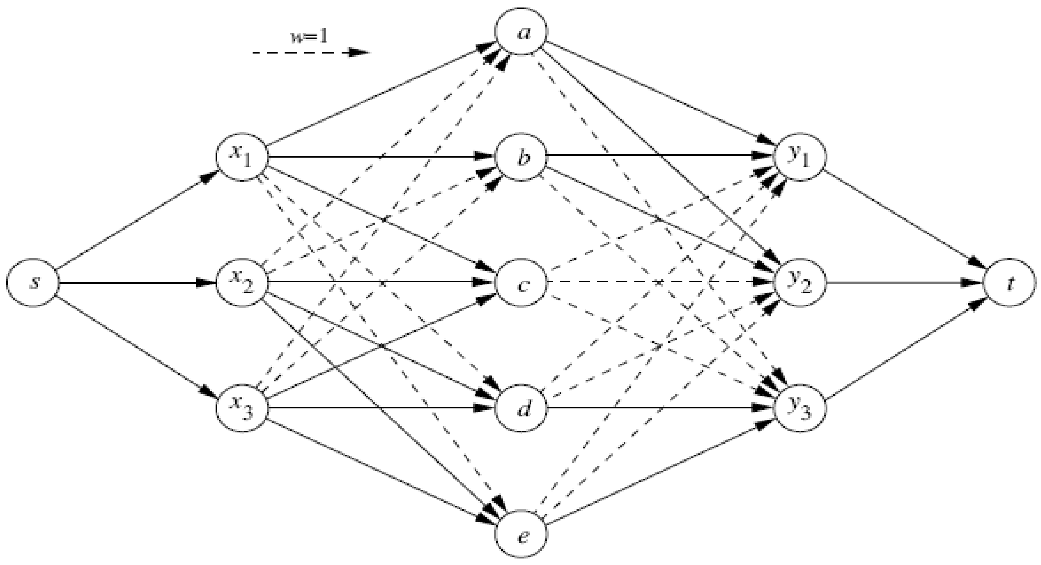

Figure 2: Graph representation for the variable-based soft same constraint. For all arcs the capacity is 1. Dashed arcs indicate the inserted weighted arcs with weight 1.

The constraint soft same (X,Y,z,μ_{var}) can be made hyper-arc consistent by applying Algorithm 2. Consistency can again be checked by computing an initial flow in $O(n(m + n \log n))$ time, and hyper-arc consistency can be achieved in $O(m(m + n \log n))$ time by applying Theorem 2. Here m denotes the number of arcs in S_{var}.

6. CONCLUSION

- ❖ For all these soft global constraints efficient hyper-arc consistency algorithms have been presented, inferred from our generic algorithm.
- ❖ In our propagation algorithms we have applied techniques from flow theory are consistent.
- ❖ The application of these techniques make our algorithms very efficient. This shows that the application of operations research techniques in constraint programming is also beneficial for propagation algorithms for soft global constraints.

REFERENCES

1. N. Beldiceanu., 2000 Global Constraints as Graph Properties on Structured Network of Elementary Constraints of the Same Type. Technical Report T 2000/01, SICS, 44: 54-78.
2. N. Beldiceanu, I. Katriel, and S. Thiel., 2004b Filtering Algorithms for the Same Constraint. In J.-C. Regin and M. Rueher, editors, Proceedings of the First International Conference on the Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, 3011: 65-79.
3. N. Beldiceanu and T. Petit., 2004. Cost Evaluation of Soft Global Constraints. In J.-C. Regin and M. Rueher, editors, Proceedings of the First International Conference on the Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems, 3011: 80-95.
4. Y. Caseau and F. Laburthe., 1997a. Solving Small TSPs with Constraints. In L. Naish, editor, Proceedings of the Fourteenth International Conference on Logic Programming (ICLP'97), 316-330.
5. J.-C. Regin., 1994, A Filtering Algorithm for Constraints of Dierence in CSPs. In Proceedings of the Twelfth National Conference on Arti¯cial Intelligence (AAAI), 1: 362-367.
6. J.-C. Regin., 1996 Generalized Arc Consistency for Global Cardinality Constraint. In Proceedings of the Thirteenth National Conference on Arti¯cial Intelligence and Eighth Innovative Applications of Arti¯cial Intelligence Conference(AAAI / IAAI), 1: 209-215.
7. J.-C. Regin., 1999a Arc Consistency for Global Cardinality Constraints with Costs. In J. Jaar, editor, Proceedings of the Fifth International Conference on Principles and Practice of Constraint Programming (CP'99), 1713: 390-404.
8. J.-C. Regin., 1999b, The symmetric alldi constraint. In Proceedings of the 16th International Joint Conference on Arti¯cial Intelligence, (IJCAI-99): 420-425.