# Series-Parallel and Tree-Decomposition Approaches for Fuzzy Constraint Networks

Alfonso Bosch[1], Francisco Guil[1], Carmen Martinez[1], and Roque Marin[2]

[1] Dpto. de Lenguajes y Computacion
Universidad de Almeria
04120 Almeria (Spain)
{abosch, fguil, cmcruz}@ual.es

[2] Dpto. de Ingenieria de la Informacion y las Comunicaciones
Universidad de Murcia
Campus de Espinardo
30071 Espinardo (Murcia, Spain)
roque@dif.um.es

**Abstract.** In this work, we present a Disjunctive Fuzzy Constraint Networks model for continuous domains, which generalizes the Disjunctive Fuzzy Temporal Constraint Networks model for temporal reasoning, and we propose the use of the series-parallel and tree-decomposition approaches for simplifying its processing. After a separate empirical evaluation process of both techniques, a combined evaluation process over the same problem repository has been carried out, finding that series-parallel problems practically subsume tree-decomposable problems.

## 1 Introduction

Fuzzy Constraint Networks (FCN) model, introduced in [14,16], allows expressing simple constraints, representing them by means of a convex and normalized possibility distribution over real numbers. Fuzzy constraints allow combining precise and imprecise information, which can be also qualitative and quantitative. This model is suitable for temporal reasoning and other continuous domains where the combination of such constraint types is required. A fuzzy model allows intermediate consistency degrees, and to quantify the possibility and necessity of a relationship or query.

Fuzzy constraints are used in several contexts, such as medical systems, phytosanitary control, and other domains [1,15,25].

In certain tasks, such as planning, a more general model is needed, where constraints can be convex or not. Then, the FCN model is enhanced, allowing the definition of a constraint with a finite set of possibility distributions, normalized and convex, obtaining the Disjunctive Fuzzy Constraint Networks (DFCN) model. For temporal reasoning, this model extends the TCSP (Temporal Constraint Satisfaction Problems) framework proposed by Dechter [8], and it allows constraints such as "Irrigation *is much before* or *a little after* than Treatment", and subsumes the Vilain & Kautz point algebra (PA) [22]. This framework allows representing all the possible

F.J. Garijo, J.C. Riquelme, and M. Toro (Eds.): IBERAMIA 2002, LNAI 2527, pp. 275-284, 2002.

relationships between time points, between intervals and between time points and intervals, and their disjunctions (without managing repetitive patterns).

The aim of this framework is to contribute to constraint-based reasoning under uncertainty for continuous domains, using fuzzy CSPs (Constraint Satisfaction Problems) for search and querying, mixing different filtering techniques and backtrack search.

The main drawback of DFCN is its computational inefficiency, because generally these networks are non-decomposable networks [7,24], needing backtracking to find a solution [11,12,23]. Determining the consistency and computing the minimal network are also exponential. With small problems, this is not a drawback, but in order to generalize the use of the model in a general scope, it would be interesting to simplify its processing, if possible. The idea is to explore different approaches to be used before applying backtracking.

One approach is to try avoiding backtracking, using the topology of the problem graph [8]. Another one is decomposing the network into subproblems that can be solved separately. A third approach is to apply preprocessing, reducing the original network and testing the problem consistency [20,21].

The remainder of this paper is organized as follows. Section 2 presents the DFCN model; Section 3 presents two approaches for managing constraint networks: series-parallel networks and tree decomposition; section 4 presents the empirical evaluation and the analysis of the results; and section 5 summarizes the conclusions and presents the future work.

## 2 The Disjunctive Fuzzy Constraint Networks Model

A disjunctive fuzzy constraint network (DFCN) $L^d$ consists of a finite set of $n$+1 variables $X_0, \ldots, X_n$ ($X_0$ as origin for problem variables), whose domain is the set of real numbers $\boldsymbol{R}$, and a finite set of disjunctive binary constraints $L_{ij}^d$ among these variables. $X_0$ is a variable added to use only binary constraints, and it can be assigned to an arbitrary value (for simplicity's sake, this value is usually 0).

A disjunctive binary constraint $L_{ij}^d$ among variables $X_i$, $X_j$ is defined with a finite set of possibility distributions, $\{\pi_{ij}^1, \pi_{ij}^2, \ldots, \pi_{ij}^k\}$ normalized and convex [9], defined over the set of real numbers $\boldsymbol{R}$; for $x \in \boldsymbol{R}$, $\pi_m(x) \in [0,1]$ represents the possibility that a quantity $m$ can be precisely $x$.

A value assignation for variables $X_i$, $X_j$, $X_i=a$; $X_j=b$, $a, b \in \boldsymbol{R}$, satisfies the constraint $L_{ij}^d$ iff it satisfies one of its individual constraints:

$$\exists \pi_{ij}^p \in L_{ij}^d / \pi_{ij}^p(b-a) > 0 \tag{1}$$

The maximum possibility degree of satisfaction of a constraint $L_{ij}^d$ for an assignment $X_i = a$, $X_j = b$ is

$$\sigma_{ij}^{\max}(a,b) = \max_{1 \le p \le k} \pi_{ij}^{p}(b-a) \tag{2}$$

A constraint $L_{ij}^{d}$ among variables $X_i$, $X_j$ defines a symmetric constraint $L_{ji}^{d}$ among $X_j$, $X_i$, and the lack of a constraint is equivalent to the universal constraint $\pi_U$. A DFCN can be represented with a directed graph, where each node corresponds to a variable and each arc corresponds to a constraint between the connected variables, omitting symmetric and universal constraints. The set of possible solutions of a DFCN $L^d$ is defined as the fuzzy subset from $\boldsymbol{R}^n$ associated to the possibility distribution given as:

$$\pi_S(v_1,...,v_n) = \min_{\substack{0 \le i \le n \\ 0 \le j \le n}} (\sigma_{ij}^{\max}(v_i, v_j) \tag{3}$$

An $n$-tuple $V = (v_1, ...v_n) \in \boldsymbol{R}^n$ of precise values is an $\sigma$-possible solution of a DFCN $L^d$ if $\pi_S(V) = \sigma$. We say that a DFCN $L^d$ is consistent if it is 1-consistent, and it is inconsistent if it does not have any solution.

Given a DFCN $L^d$, it is possible to find out several networks which are equivalent to $L^d$. We can obtain this networks using the composition and intersection operations, defined in [3] for temporal reasoning. Among all the equivalent networks, there is always a network $M^d$ DFCN that is minimal. This network contains the minimal constraints. If $M^d$ contains an empty constraint, $L^d$ is inconsistent. If $p$ if the maximum of possibility distributions in each constraint, and the network has $q$ disjunctive constrains and $n$ variables, then the minimal network $M^d$ of a DFCN $L^d$ can be obtained with a complexity O($pqn3$), where $n3$ is the cost of solving each case non disjunctive FCN [16]. Due to this exponential complexity, we need to find a more practical approach.

## 3 Series-Parallel Networks and Tree Decomposition

It is well known that topological characteristics of constraint networks can help to select more effective methods to solve them, and there are previous studies about this topic [6,8]. These characteristics have been examined for both FCN and DFCN models; in this work, we will focus only in topics involved with disjunctive problems, because they are exponential. The selected approaches are series-parallel networks and tree-decomposition.

### 3.1 Series-Parallel Networks

A network is series-parallel [18] in respect to a pair of nodes $i$,$j$ if it can be reduced to arc ($i$,$j$) applying iteratively this reduction operation: a) select a node with a degree of two or less; b) remove it from the network; c) connect its neighbours. A network is series-parallel if it is series-parallel in respect to every pair of nodes. The basic algorithm for checking if a network is series-parallel has an O($n^3$) complexity, and there

is a more efficient algorithm that checks this property with an O($n$) complexity [26], applied to fault-tolerant networks (IFI networks).

If a DFCN is series-parallel, the path consistent network is the minimal network, although the intersection and composition operations are non-distributive [26]. As a subproduct of checking if a network is series-parallel, a variable ordering is obtained, when deleting the nodes. Applying directional path-consistency (DPC) algorithm [8] in the reverse order, a backtrack-free network is obtained and the minimal constraint between the first two variables of the ordering too. This can be interesting when we need only to compute a minimal constraint for two variables, and not the minimal network, as in LaTeR [5]. In addition, if the network is series-parallel, we can decide absolutely whether the network is consistent, by applying DPC algorithm in the reverse order.

Figure 1 shows a series-parallel network. It can be reduced to any of its arcs applying the reduction process.

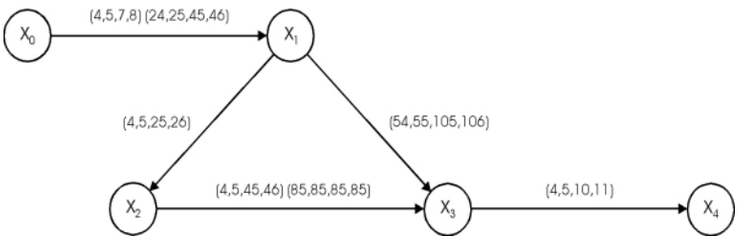


**Fig. 1.** Example of series-parallel network

However, the network shown in Figure 2 is not series-parallel, because there is not any admissible reduction sequence for any arc. We can see easily that the only node with grade less or equal to two is $X_0$.

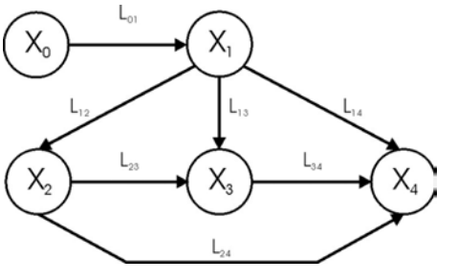


**Fig. 2.** Example of non series-parallel network

The proposed algorithm for checking if a network is series-parallel (a variant of the algorithm proposed in [26]) is:

**SP (Series-Parallel) Algorithm**

```
  Input:  A Fuzzy Constraint Network.
  Output: A node removal sequence.


begin
  for each i=0..n Calculate-degree (i)
  NodeQueue = {nodes with degree 1 and 2}
  While (NodeQueue <> ∅ and |V| > 3)
    begin
     node = Extract(NodeQueue)
     V <- V - {node}
     if Degree(node) = 1
       then Degree(Neighbour(node)) --
     if Degree(Neighbour(node)) = 2
       then Introduce(NodeQueue, Neighbour(node))
       else if Connected(Neighbours(node))
              then Degree(Neighbours(node)) --
     if Degree(Neighbours(node)) = 2
       then Introduce(NodeQueue, Neighbours(node))
       else E <- E + {NeighboursArc(node)}
    end
  if (NodeQueue = ∅ and |V| > 3)
    then exit ("The network is not series-parallel")
end
```

**Fig. 3.** Series-parallel algorithm

The algorithm ends when the queue is empty or there are three nodes left in the network. If the queue is empty, the reduction process has finished. In such case, if there are more than three nodes, the network is not series-parallel, because there are at least four nodes with a degree greater than two (there is a graph that is homomorphic to $K_4$ [26]).

### 3.2 Tree Decomposition

We stated that general DFCN are not tractable when searching the minimal network and finding a solution, but both problems are tractable when a DFCN has a tree structure. Then, it seems adequate to study the possibility of removing redundancies from a DFCN to extract (if it is possible) a tree representing a relative DFCTN equivalent to the original one.

Tree-decomposition is proposed by Meiri *et al.* [17] for discrete CSPs, and it can be extended to DFCN. If a tree $T^d$ can be extracted from a path-consistent network by means of arc removal, the tree $T^d$ represents exactly the original network. Otherwise, if a tree representation cannot be extracted, the algorithm stops and notifies this fact. The tree extraction using arc removal will be possible only when the path-consistent

network is also minimal. In addition, if the path-consistent network is minimal, we can state that if the algorithm cannot find the tree decomposition, then there is no tree representation.

The tree decomposition method consists of removing redundant constraints from the original network, until a tree that exactly represents the network without information loss is found [2,17,19]. The algorithm works as follows: Given a path-consistent DFCN, it examines each triplet of variables, identifying the redundancies of each triplet, assigning weights to the arcs depending on the found redundancies. The generated tree, $T^d$, is a maximum weight spanning tree (MWST) respect to these weights. The last step is to verify that $T^d$ represents truly the original network. If $T^d$ does not represent truly the original network, removed arcs can be added again, until both networks become equivalent. This algorithm has a polynomial cost, and when applied to a minimal disjunctive network, it determines whether the network is decomposable or not.

We proposed an algorithm [2] that generates a tree $T^d$ with these characteristics. If a tree can be extracted from a path-consistent network using arc removal, the tree $T^d$ represents exactly the original network. Otherwise, if tree representation cannot be extracted, the algorithm stops and notifies this fact. The tree extraction using arc removal will be possible only when the path-consistent network is also minimal. In addition, if the path-consistent network is minimal, we can state that if the algorithm cannot find the tree decomposition, then there is no tree representation. The algorithm extends the proposal of Meiri *et al.* [17] for discrete CSPs, and it is depicted in [19].

## 4 Empirical Evaluation and Results

We have conducted an empirical evaluation process, generating sets of random DFCN with different characteristics, preprocessing them with PC-2 (Path-Consistency) algorithm from Mackworth [13], and applying Tree Decomposition (TD) and Series-Parallel (SP) algorithms.

The parameters used in our problem generator are $n$ (the number of variables), $R$ (the range of the constraints), $p$ (the number of possibility distributions in each constraint), $q$ (the connectivity of the graph), $T$ (the tightness of the constraints) and $F$ (the fuzziness of the constraints). All the constraints generated in each problem have the same number of possibility distributions. The values selected for the first test battery were $n$= 4 – 40; $R$= 600; $p$= 1,2,4,8,16,32; $q$= 0.1,0.3,0.5; $T$= 0.1,0.5,0.9.

The first analysis of the results obtained in the TD evaluation process is presented in [2]. In [4], we analyzed deeper the behaviour of TD, and presented a first analysis of SP, over a new battery of problems generated with the same parameter set. When analyzing the results, we observed that the overall trend of SP was similar to the corresponding of TD: when increasing the problem size, the number of problems where these heuristics succeed diminished. The main found difference was that, in every category, the number of SP problems was always greater than TD problems.

These results lead us to make a combined evaluation of both approaches, testing the SP algorithm over the same battery of path-consistent problems used as input for TD evaluation. Figure 4 shows the overall trend of both algorithms. Note that SP curve is

always over TD curve. The bars show the number of path-consistent problems for each number of variables.

The next question is to check the relationship between the two approaches, that is, to know whether a TD problem is also SP, and vice versa. Figure 5 shows the fraction of the four possible cases: 0-0 represents the problems that are neither TD nor SP. 1-1 represents the problems that are both TD and SP. 0-1 represents the problems that are SP but not TD. In addition, 1-0 represents the problems that are TD but not SP. The last case is the most interesting one, because practically there are not problems with this pattern. Looking in depth the result database, there are only five isolated problems TD that are not SP into a population of 25068 problems. In addition, 3551 problems that are SP but not TD, and 5005 problems are SP and TD. Then, the overall fraction of SP problems represents a 34.13 %, versus a 19.98 % for TD.

From this analysis, we propose using SP as standard approach, because it runs with a lower time and memory requirements, and practically subsumes the TD approach.

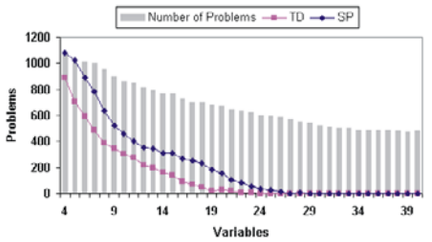


**Fig. 4.** Number of PC, TD, and SP problems vs. Variable Number.

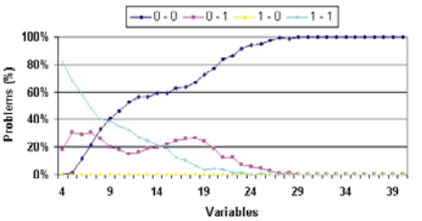


**Fig. 5.** Comparative analysis of TD and SP problems v. Variable Number. (TD-SP)

The interest of determining whether a FCN is SP consists on avoiding the need of backtracking. First, the consistency of the problem can be determined with DPC (Directional Path-Consistency) algorithm [8], using the inverse node removal sequence. The output network from DPC can be used to obtain a solution without backtracking, also using the inverse removal sequence. In addition, if the minimal network is needed, it can be computed applying PC-2.

Figure 6 shows the minimal network for the sample FCN shown in Figure 1, obtained with PC-2. Using this information, we can obtain a solution for this network, using the removal sequence {0,2,4,3,1}. We instantiate the variables in this order: $X_1$, $X_3$, $X_4$, $X_2$, and $X_0$. A 1-possible solution is:

$X_1 = 5$
$X_3 = 60$
$X_4 = 65$
$X_2 = 15$
$X_0 = 0$

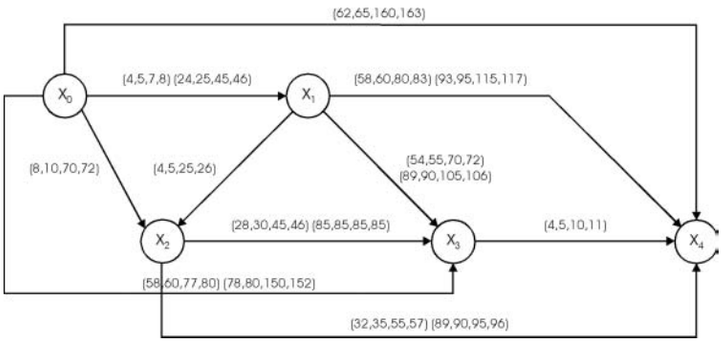


**Fig. 6.** Minimal network obtained with PC-2

## 5 Conclusions and Future Work

In this work, we have proposed the DFCN model for constraint networks in continuous domains. Among the candidate techniques for managing these networks, we have selected tree-decomposition and series-parallel networks, carrying out two evaluation processes. First, we made an independent evaluation of both approaches. The analysis of this process leads us to carry out a combined evaluation process. After a detailed study of the results of the last one, we can say that SP has a greater success than TD, and SP practically subsumes TD. In addition, SP presents a lower complexity, and it offers all the features of TD: obtaining solutions without backtracking and computing the minimal network with path-consistency algorithms. Moreover, problem consistency and solutions can be determined and obtained using directional path-consistency, with a complexity lower than general path-consistency.

As future work, we propose to study the application of SP for decomposing the networks onto two types of subnetworks: SP subproblems, which could be solved with the techniques applied for SP, and non-SP subproblems, which could be solved with backtracking. This could be an alternative to other decomposition approaches proposed by our group [3] and other standard techniques, as nonseparable components [10].

Another proposal could be a deeper study of cases of problems that are TD and not SP, trying to find a pattern of this type of problems, where TD can be useful.

All this information could be used to select the better approach for each particular network.

**Acknowledgements.** The authors would like to thank the anonymous reviewers for their helpful comments on preliminary versions of this paper.

This work is partially supported by an EC FEDER Program grant (1FD97-0255-C03-03) and a Spanish MCYT Program grant ((TIC2000-0873-C02-02).

## References


1. S. Barro, R. Marín, and A.R. Patón, "A model and a language for the fuzzy representation and handling of time", *Fuzzy Sets and Systems*, 61, 1994, pp. 153-175.
2. A. Bosch, M. Torres, I. Navarrete, and R. Marín, "Tree Decomposition of Disjunctive Fuzzy Temporal Constraint Networks". *Proc. of Computational* Intelligence*: Methods and Applications CIMA'2001*, ICSC-NAISO, Bangor (UK), 2001, #1714-066, 7 pages.
3. A. Bosch, M. Torres, R. Marín. Reasoning with Disjunctive Fuzzy Temporal Constraint Networks. TIME-2002, Manchester (UK), 8 pages., 2002 (accepted).
4. A. Bosch, C. Martínez, F. Guil, R. Marín. Solving Fuzzy Temporal Problems Without Backtracking.. Eurasian-2002, Teherán (Irán), 10 pages., 2002 (accepted).
5. V. Brusoni, L. Console, B. Pernici, and P. Terenziani, "LaTeR: a general purpose manager of temporal information", *Methodologies for intelligent systems 8*, LNCS 869, Springer, 1994, pp. 255-264.
6. R. Dechter, "Enhancement Schemes for Constraint Processing: Backjumping, Learning and Cutset Decomposition." *Artificial Intelligence* 41, Elsevier, 1990, pp. 273-312.
7. R. Dechter, I. Meiri, and J. Pearl, "Temporal constraint networks", *Artificial Intelligence* 49, Elsevier, 1991, pp. 61-95.
8. R. Dechter, and J. Pearl, "Network-based heuristics for constraint-satisfaction problems", *Artificial Intelligence*, 34, Elsevier, 1987, pp. 1-38
9. D. Dubois, H. Prade, *Possibility Theory: An approach to computerized processing of uncertainty*, Plenum Press, New York, 1988.
10. S. Even, *Graph Algorithms*. Computer Science Press, Rockville, MD, 1979.
11. E. Freuder, "A sufficient condition for backtrack-free search", *Journal of the ACM* 29, 1, ACM Press, 1982, pp. 24-32.
12. G. Kondrak, and P. van Beek, "A Theoretical Evaluation of Selected Backtracking Algorithms", *Artificial Intelligence* 89, Elsevier, 1997, pp. 365-387.
13. A. Mackworth, "Consistency in networks of relations", *Artificial Intelligence* 8, Elsevier, 1977, pp. 99-118.
14. R. Marín, S. Barro, A. Bosch, and J. Mira, "Modelling the representation of time from a fuzzy perspective", *Cybernetics and Systems*, 25, 2, Taylor&Francis, 1994, pp. 207-215.
15. R. Marín, S. Barro, F. Palacios, R. Ruiz, and F. Martin, "An Approach to Fuzzy Temporal Reasoning in Medicine", *Mathware & Soft Computing*, 3, 1994, pp. 265-276.
16. R. Marín, M. Cardenas, M. Balsa, and J. Sanchez, "Obtaining solutions in fuzzy constraint networks", *Int. Journal of Approximate Reasoning*, 16, Elsevier, 1997, pp. 261-288.
17. I. Meiri, R. Dechter, and J. Pearl, "Uncovering trees in constraint networks", *Artificial Intelligence*, 86, Elsevier, 1996, 245-267.
18. U. Montanari, "Networks of constraints: fundamental properties and applications to picture processing", *Information Science*, 7, 1974, pp. 95-132.
19. I. Navarrete, R. Marín, and M. Balsa, "Redes de Restricciones Temporales Disyuntivas Borrosas", *Proceedings of ESTYLF'95*, Murcia, European Society for Fuzzy Logic and Technology, (Spain), 1995, pp. 57-63
20. E. Schwalb, and R. Dechter, "Coping With Disjunctions on Temporal Constraint Networks", *Proc. American Association Artificial Intelligence'93*, AAAI, Washington, 1993, pp. 127-132.
21. E. Schwalb, and R. Dechter, "Processing Disjunctions in Temporal Constraint Networks", *Artificial Intelligence* 93, Elsevier, 1997, pp. 29-61.
22. E. Schwalb, and L. Vila, "Temporal Constraints: A Survey", *Constraints* 3 (2/3), 1998, pp. 129-149.
23. K. Stergiou and M. Koubarakis, "Backtracking Algorithms for Disjunctions of Temporal Constraints", *Artificial Intelligence* 120, Elsevier, 2000, pp. 81-117.
24. E. Tsang, *Foundations of Constraint Satisfaction*, Academic Press, London, 1993.

25. Túnez, S.; Del Aguila, I.; Bienvenido, F.; Bosch, A. y Marín, R.(1996). Integrating decision support and knowledge-based system: application to pest control in greenhouses. Procedings 6th International Congress for Computer Technology in Agriculture (ICCTA'96), pp. 417-422. Wageningen.
26. J.A. Wald, and C.J. Colburn, "Steiner Trees, Partial 2-Trees and Minimum IFI Networks", *Networks,* 13, 1983, pp. 159-167.