Solving Fuzzy Temporal Problems Without Backtracking

Alfonso Bosch[1], Francisco Guil[1], Carmen Martinez[1], Roque Marin[2]
1 Dpto. Lenguajes y Computación. Universidad de Almeria.(Spain)
2 Dpto. Ingeniería de la Información y Comunicaciones. Univ. de Murcia (Spain)
{abosch, fguil, cmcruz}@ual.es, roque@dif.um.es

Abstract

The Disjunctive Fuzzy Temporal Constraint Network model allows to express complex qualitative and quantitative temporal constraints, but its complexity is exponential. There are several subsets of these networks that can be solved in polynomial time. Another subset can be reduced to equivalent networks with a tree structure. In order to study the usefulness of these approaches, an evaluation has been carried out, generating sets of random test problems. The goal is to obtain a set of methods for managing Disjunctive Fuzzy Temporal Constraint Networks, integrating them in a general-purpose module, and to obtain, if possible, heuristic knowledge for selecting the most suitable method for each concrete problem.

1. Introduction

Fuzzy Temporal Constraint Network (FTCN) model, introduced in [1,9], allows to express temporal constraints with convex and normalized possibility distributions. Fuzzy temporal constraints allow combining precise, imprecise, qualitative, and quantitative information. Then, this model is suitable for temporal reasoning in 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.

In certain tasks, such as planning, a more general model is needed, where temporal constraints can be convex or not. Then, the FTCN model is enhanced, allowing the definition of a constraint with a finite set of possibility distributions, normalized and convex, obtaining the Disjunctive Fuzzy Temporal Constraint Networks (DFTCN) model. This model extends the TSCP framework proposed by Dechter [5], and it allows constraints such as "Irrigation *is much before* or *a little after* than Treatment", subsuming the Vilain & Kautz point algebra (PA) [14]. This framework allows representing all the possible relationships between time points, intervals, and their disjunctions. The aim of this framework is to contribute to constraint-based temporal reasoning under uncertainty, using fuzzy TCSPs for search and querying.

The main drawback of DFTCN is its computational inefficiency, because generally these networks are non-decomposable networks [5], needing backtracking to find a solution. 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 [6]. 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.

This model is being integrated with an advisor system for the control of plagues in greenhouses. The advisor system (SAEPI) makes a decision about the need of a chemical treatment, where several crops and/or plagues can be involved. The next step is the selection of the best treatment plan, based on a multicriteria decision model. This treatment consists usually of multiple actions (application of chemical products, release of natural enemies), and there are complementary measures too (take out auxiliary elements, like beehives). The different actions are affected by temporal constraints among them and with general cultivation scheduling. The last step is to use the DFTCN model to assess the suitability of the treatment plan, checking its consistency, and generating a schedule, or checking if a schedule proposed by the user (a solution) is possible or not, and its degree of possibility.

The remainder of this paper is organized as follows. Section 2 presents the DFTCN model; Section 3 presents approaches for constraint networks: series-parallel networks and tree decomposition; Sections 4 and 5 present an empirical evaluation and an analysis of the results; and Section 6 summarizes the conclusions and presents the future work.

2. The Disjunctive Fuzzy Temporal Constraint Networks Model

A disjunctive fuzzy temporal constraint network (DFCTN) L^d consists of a finite set of n+1 temporal variables $X_0, \ldots ,X_n$ (X_0 as time origin for problem variables), whose domain is a full ordered set of equidistant precise instants $\tau = \{t_0, t_1,..., t_i, ...\}$, and a finite set of disjunctive binary constraints L_{ij}^{u} among these variables. The separation between two consecutive instants, t_{i+1}-t_i, will be the maximum precision factor f_p, which can be selected depending on the nature of the problem. 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} among temporal variables X_i, X_j is defined with a finite set of possibility distributions, $\{\pi_{ij},\pi_{ij},...,\pi_{ij}\}$ normalized and convex [7], defined over a set ***I***, isomorphic with integers ***Z***; for $n \in$ ***I***, π_m (n) $\in$ [0,1] represents the possibility that a time quantity m can be precisely n time units.

A value assignation for variables X_i, X_j, X_i=a; X_j=b, a, b $\in$ τ, satisfies the constraint L_{ij} 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} 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} among X_i, X_j defines a symmetric constraint L_{ji} among X_j, X_i, and the lack of a constraint is equivalent to the universal constraint π_U . A DFTCN can be represented with a directed graph, where each node maps to a variable and each arc maps to a constraint between the connected variables, omitting symmetric and universal constraints. The set of possible solutions of an DFTCN L_d is defined as the fuzzy subset from τ^n associated to the possibility distribution given as:

$$\pi_S(t_1,...,t_n) = \min_{\substack{0\le i\le n \\ 0\le j\le n}} (\sigma_{ij}^{\max}(t_i,t_j) \tag{3}$$

An n-tuple $T = (t_1, \ldots t_n) \in \tau^n$ of precise instants is an σ-possible solution of a DFTCN L^d if $\pi_S(T) = \sigma$. We say that a DFTCN L^d is consistent if it is 1-consistent, and it is inconsistent if it does not have any solution.

Given a DFTCN 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]. Among all the equivalent networks, there is always a network M^d DFTCN that is minimal. This network contains the minimal constraints. If M^d contains a 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 DFTCN L^d can be obtained with a complexity $O(p^q n^3)$, where n^3 is the cost of resolving each case non disjunctive FTCN[10]. Due to this exponential complexity, we need to find a more practical approach.

3. Series-Parallel Networks and Tree Decomposition

Topological characteristics of constraint networks can help us to select more effective methods to solve them, and there are previous studies about this topic [6]. In this work, we will focus on series-parallel networks and tree-decomposition.

3.1. Series-Parallel Networks

A network is series-parallel [12] 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. There is an algorithm that checks this property with an O(n) complexity [15].

If a DFTCN is series-parallel, the path consistent network is the minimal network [15]. 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 [6] 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, as in LaTeR [4]. 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.

3.2. Tree Decomposition

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

Tree-decomposition is proposed by Meiri *et al.* [11] for discrete CSPs, and it can be extended to DFTCN. 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. The tree extraction using arc removal will be possible only when the path-consistent network is also minimal. If the path-consistent network is minimal, and 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 [2,11,13]. The algorithm works as follows: Given a path-consistent DFTCN, it examines each triplet of variables, identifying the redundancies of each triplet, assigning weights to the arcs depending on the founded 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.

4. Empirical Evaluation

We have conducted an empirical evaluation process, generating sets of random DFTCN with different characteristics, preprocessing them with PC-2 algorithm from Mackworth [8], and applying Tree-Decomposition and Series-Parallel algorithms.

The parameters used in our problem generator are n (variables), R (constraint range), p (constraint possibility distributions), q (connectivity), T (constraint tightness) and F (fuzziness). The values selected for the first test battery have been $n= 4 – 40$; $R= 600$; $p= 1,2,4,8,16,32$; $q= 0.1,0.3,0.5$; $T= 0.1,0.5,0.9$; $F=0.1$.
n ranges from 4 to 40 variables, in order to see the evolution from very small to large problems. R has a fixed value of 600, enough to fit the maximum number of possibility distributions, and due to the continuous nature of the model (versus the discrete CSP models), it seems not necessary to test different values. p (non-disjunctive problems) to 32 (highly disjunctive problems), with a exponential progression, because this is intuitively one of the factors with higher influence on the problem complexity. Another factor with high influence is q, representing the number of problem constraints; we use three levels, from 0.1 to 0.5, enough to create highly interconnected graphs. The tightness of the problem is the fraction of values allowed by a constraint, and there are three cases: few values allowed (0.1), average (0.5) and many values (0.9). F indicates the fraction of values allowed with a possibility degree lower than 1, with a fixed value of 0.1.

The number of generated problem sets is $npqT$ = 37x6x3x3 = 1998. For each set of problems, we have generated and solved 20 individual problems. All the generated and processed problems have been stored in a problem repository.

In every evaluation process, it is necessary to verify the results. We have repeated the process with new sets of problems for 4, 6 ... 40 variables, comparing the obtained results. The difference on the percentage of tree-decomposable problems was 0.3 %, and the difference on the execution time was 2.6%. For series-parallel, the difference was 0.2%.

5. Results

Figure 1 shows that the fraction of series-parallel problems, decreases with the problem size. Figure 2 shows decreases with the problem size. Above 24 variables, there is not any path-consistent problem that could be tree-decomposable. And the number of path-consistent problems decreases also with the problem size.

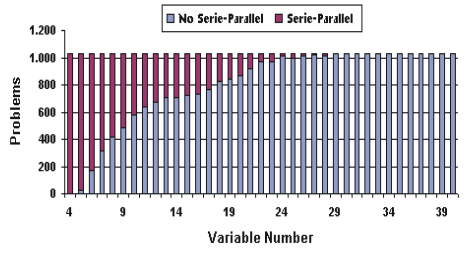

Fig. 1. Series-Parallel problems versus Variable Number

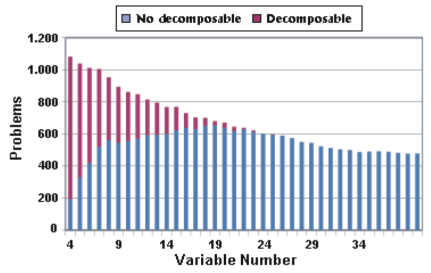

Fig. 2. Tree-decomposable and Path-consistent problems versus Variable Number

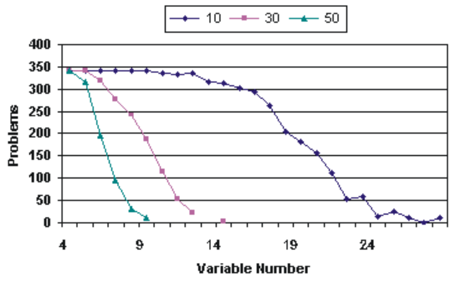

Fig. 3. SP problems vs. Variable Number for each connectivity

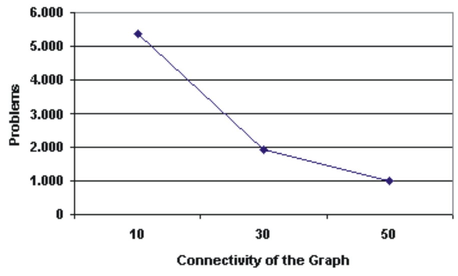

Fig. 4. SP problems vs. connectivity

Figure 3 shows that connectivity has a strong impact on the series-parallel property. For sparse problems (10% connectivity) the curve holds up to approximately 17 variables, and drops after this point.

Figure 4 shows the overall influence of connectivity, presenting a inverse exponential trend.

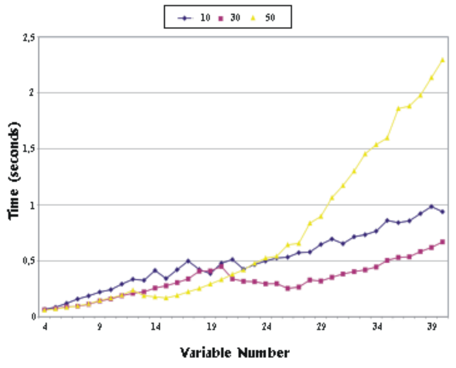

Fig. 5. TD execution time for each connectivity vs. Variable Number.

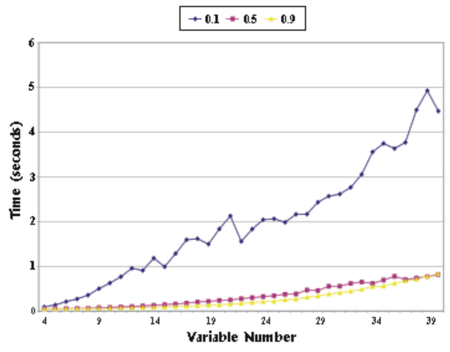

Fig. 6. TD execution time for each tightness vs. Variable Number.

Figure 5 shows that a connectivity of 10% (which maps with problems with less constraints) has a higher cost than the connectivity of 30%, while a connectivity of 50% presents a lower cost until 20 variables, and starting from 25 variables the cost increases dramatically. Then, there is not a proportional response between the number of problem constraints and the elapsed time on this parameter range. As we know that above 25 variables there are not tree-decomposable problems, the lower cost for few variables when complexity is high could be a consequence of the earlier detection of that the problem is non tree-decomposable. We can check it looking at the number and fraction of tree-decomposable problems for the different parameter combinations.

In Figure 6, the highest cost maps with a tightness of 0.1 (tighter problems), with multiple peaks, while the 0.5 and 0.9 values have a similar cost and lower.

Table 1 shows the number of decomposable problems versus the number of possibility distributions for each constraint.

Table 1. Number of tree-decomposable problems vs. the number of possibility distributions

	Possibility distributions for each constraint					
Variables	**1**	**2**	**4**	**8**	**16**	**32**
4	156	148	144	147	145	146
5	114	114	133	107	118	122
6	100	102	100	94	94	101
7	95	87	73	64	85	80
8	80	62	70	55	59	62
9	57	61	60	47	57	65
10	47	51	51	43	54	57
11	50	50	42	40	51	42
12	33	32	38	40	37	40
13	38	44	28	29	30	29
14	14	25	33	34	32	26
15	33	29	25	19	21	17
16	15	17	15	7	15	21
17	12	9	13	12	12	14
18	8	13	6	8	6	6
19	3		4	5	5	7

20	3	2	9	5	3	8
21	7	6		4	2	4
22		2		1	2	2
23	3			3	1	
24						
25					1	1
Total	868	854	844	764	830	850

We can see that the number of possibility distributions has not a direct influence on the number of tree-decomposable problems. A behaviour pattern cannot be derived from Table 1. For each number of variables, the number of tree-decomposable problems versus the number of possibility distributions has variations, but between a range (sometimes they are significant, but without a pattern). In fact, when grouping the number of tree-decomposable problems for each number of possibility distributions, the only total that presents a meaningful deviation corresponds to 8 possibility distributions for each constraint (which is the lowest). Then, we can state that the level of disjunctions in a DFTCN has not influence on the possibility that the network could be tree-decomposable.

6. Conclusions and Future Work

This paper presents the application of tree decomposition and series-parallel networks for an efficient management of DFTCN problems, which nature is exponential. We have carried out an exhaustive evaluation test, verifying its accuracy. The cost of tree decomposition is polynomial, and its success is mainly limited by the problem size and connectivity. The cost of series-parallel test is linear, and its usefulness seems to be higher.

A natural continuation of this work is to study deeper the obtained results, in order to search a cost model for TD or an estimation about whether a DFTCN is tree-decomposable (or series-parallel) or not, based on its initial features.

There are also other topological properties of DFTCN that are being exploited and studied. Preprocessing methods are also interesting, because they can reduce the backtracking search effort. The general goal is to develop a prototype for the representation and management of temporal information, with a selection mechanism that chooses the most suitable technique for each task.

The selection mechanism could use initially heuristics based on direct problem features (topology, disjunction ratio, number of disjunctions, range). In a next stage, we can study if it is possible to propose a cost model to bound "tractable" problems, and the necessary effort to solve a concrete problem.

Acknowledgements. 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), pp. .36-43, 2002.

[4] 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.

[5] R. Dechter, I. Meiri, and J. Pearl, "Temporal constraint networks", *Artificial Intelligence* 49, Elsevier, 1991, pp. 61-95.

[6] R. Dechter, and J. Pearl, "Network-based heuristics for constraint-satisfaction problems", *Artificial Intelligence*, 34, Elsevier, 1987, pp. 1-38

[7] D. Dubois, H. Prade, *Possibility Theory: An approach to computerized processing of uncertainty*, Plenum Press, New York, 1988.

[8] A. Mackworth, "Consistency in networks of relations", *Artificial Intelligence* 8, Elsevier, 1977, pp. 99-118.

[9] 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.

[10] 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.

[11] I. Meiri, R. Dechter, and J. Pearl, "Uncovering trees in constraint networks", *Artificial Intelligence*, 86, Elsevier, 1996, 245-267.

[12] U. Montanari, "Networks of constraints: fundamental properties and applications to picture processing", *Information Science*, 7, 1974, pp. 95-132.

[13] 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

[14] E. Schwalb, and L. Vila, "Temporal Constraints: A Survey", *Constraints* 3 (2/3), 1998, pp. 129-149.

[15] J.A. Wald, and C.J. Colburn, "Steiner Trees, Partial 2-Trees and Minimum IFI Networks", *Networks,* 13, 1983, pp. 159-167.