Available at
www.ElsevierComputerScience.com
POWERED BY SCIENCE DIRECT®
Journal of Discrete Algorithms 1 (2003) 303–312
JOURNAL OF DISCRETE ALGORITHMS
www.elsevier.com/locate/jda

On-line algorithms for networks of temporal constraints

Fabrizio d'Amore [a,*], Fabio Iacobini [b]

[a] *Dipartimento di Informatica e Sistemistica, Università di Roma "La Sapienza", Via Salaria 113, 00198 Roma, Italy*
[b] *Oracle Italia, Via Bombay 1, 00144 Roma, Italy*

Abstract

We consider a semi-dynamic setting for the Temporal Constraint Satisfaction Problem (TCSP), where we are requested to maintain the path-consistency of a network under a sequence of insertions of new (further) constraints between pairs of variables. We show how to maintain the path-consistency in $O(nR^3)$ amortized time on a sequence of $\Theta(n^2)$ insertions, where n is the number of vertices of the network and R is its range, defined as the maximum size of the minimum interval containing all the intervals of a single constraint.

Furthermore we extend our algorithms to deal with more general temporal networks where variables can be points and/or intervals and constraints can also be defined on pairs of different kinds of variables. For such cases our algorithms maintain their performance. Finally, we adapt our algorithms to also maintain the arc-consistency of such generalized networks in $O(R)$ amortized time for $\Theta(n^2)$ insertions.

Keywords: Network constraints; Temporal constraints; On-line algorithms

1. The problem

Given a totally ordered universe U, a *network of temporal constraints* [7] is a pair (G, ℓ), where $G = (V, E)$ is an undirected graph and ℓ is a function mapping its edges to sets of (closed) intervals of U. Each vertex of $V = \{v_1, \ldots, v_n\}$ is a *variable* varying in U and each edge e is a *constraint* on the difference between its incident variables, namely, if $\ell(e) = \{I_1, \ldots, I_k\}$, I_j being a closed interval of U, and $e = \{v_i, v_j\}$, then the difference

* Corresponding author.
E-mail addresses: damore@dis.uniroma1.it (F. d'Amore), fabio.iacobini@oracle.com (F. Iacobini).

1570-8667/$ – see front matter
doi:10.1016/S1570-8667(03)00031-5

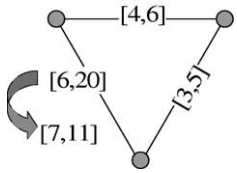

Fig. 1. Constraint [6, 20] can be restricted to [7, 11].

$v_{\max\{i,j\}} - v_{\min\{i,j\}}$ must be in $\bigcup_{I\in\ell(e)} I$. A typical choice for the universe U is the set $\mathbb{Z}$ of the signed integers. The *Temporal Constraint Satisfaction Problem* (TCSP) is the problem of finding an n-ple of U^n satisfying the constraints, namely, for each $\{v_i, v_j\} \in E$ it results $v_{\max\{i,j\}} - v_{\min\{i,j\}} \in \bigcup_{I\in\ell(v_i,v_j)} I$.

Problems involving temporal constraints arise in several areas of computer science such as scheduling [2,10], program verification [3,13,14], real time systems [18], temporal databases [21,22] and artificial intelligence (in [7] there is an extensive bibliography). See the survey paper in [20] for further details.

In [6] it is proved that the problem of deciding whether a given network admits a solution is NP-hard, even if we restrict each constraint to consist of no more than two intervals. Conversely, the particular problem occurring when each constraint is defined through a simple interval, known as the *Simple Temporal Problem* (STP), is polynomial.

Typically, the TCSP is solved through the use of backtracking and this causes exponential running time. In order to improve the overall performance, a preliminary step is normally carried out, consisting of making the network path-consistent. In a network it frequently happens that the constraint defined on a pair of variables can be made more restrictive without altering the set of the solutions because of the constraints between pairs of vertices in a path linking the initial pair. An example is illustrated in Fig. 1. Informally, the process of making a given network path-consistent consists of restricting its constraints to keep into account the effects of the propagation of the other constraints. This pre-processing is useful for lowering the running time of the successive backtracking phase and in some cases it is sufficient for deciding the whole problem (see Section 4 and [4,7]). It should be clear that in general the path-consistency of a network does not provide us with significant information about the existence of a solution: a path-consistent network can admit 0 or more solutions, although it is true that the process of restricting constraints could lead to empty ones (i.e., $\ell(e) = \emptyset$). In [19] it is proved that even local consistency algorithms can be exponential due to *fragmentation* problems, i.e., the excessive growth of the number of intervals in a constraint. The authors provide polynomial approximation algorithms, which are efficient yet effective in detecting inconsistencies and reducing fragmentation.

In this paper we consider a semi-dynamic setting for the TCSP, where we are requested to maintain the path-consistency of a network under a sequence of insertions of a new (further) constraint between pairs of variables; in terms of graphs this corresponds to restricting, for a given edge e, the set of intervals $\ell(e)$. The operation we consider is very general, allowing also to model the case where we add a constraint between two non-adjacent variables: this is achieved by creating an edge e between the pair of variables and defining $\ell(e)$ to be equal to the inserting constraint. We show how to maintain path-

consistent a network in the defined setting in $O(nR^3)$ amortized time on a sequence of $\Theta(n^2)$ insertions, where n is the number of vertices and R is the range of the network, defined as the maximum size of the minimum interval containing all the intervals of a single constraint. This is the first paper explicitly dealing with a non-static setting of the problem, although some dynamic techniques have been suggested in [20] and a preliminary version of our results have appeared in [5]. The best off-line algorithm known for the problem is PC-2 [7] which runs in $O(n^3R^3)$ worst case time. Off-line algorithms exist in the literature for the non-temporal case, where constraints are described by explicitly listing pairs of allowed values, thus giving raise to quadratic space per constraint. For this case the best algorithm is PC-4 [9] which runs in $O(n^3a^3)$ where a denotes the size of the largest involved domain.

Furthermore we extend our algorithms to deal with more general temporal networks [15], where variables can be points and/or intervals and constraints can be defined on pairs of different kinds of variables [1,25]. For such cases our algorithms maintain their performance. Finally we adapt our algorithms to also maintain the arc-consistency of such generalized networks, which is a particular kind of path-consistency limited to paths of length 1. The property is maintained in $O(R)$ amortized time for $\Theta(n^2)$ insertions. This result should be compared to that in [4], where a fully dynamic algorithm is presented for the STP only, capable of performing insertions/deletions of constraints in $O(mR)$ worst case time per operation, where m is the number of edges.

2. Preliminaries

Networks of temporal constraints have been defined in Section 1 by introducing only binary constraints. Often it is convenient to consider unary constraints as well, to the purpose of handling cases where a variable v_i is constrained to vary in a *domain* $D_i \subseteq U$. Unary constraints can be easily introduced inside a network $\mathcal{N}$ through the introduction of a new vertex v_0 (associated with a fixed value, e.g., 0) and an edge $\{v_i, v_0\}$ with appropriate label, for each vertex v_i needing a unary constraint. The resulting network is said to be an *augmented* network and is denoted by $\mathcal{N}^+$. Premising that two networks $\mathcal{M}_1$ and $\mathcal{M}_2$ with the same set of solutions are said to be *equivalent* ($\mathcal{M}_1 \equiv \mathcal{M}_2$), it is clear that $\mathcal{N}$ and $\mathcal{N}^+$ are not equivalent, but a solution for $\mathcal{N}$ also satisfying the k unary constraints is directly obtained by projecting a solution for $\mathcal{N}^+$ along its first component. If $\mathcal{N}^+$ admits no solutions then there are no solution to $\mathcal{N}$ also satisfying the unary constraints.

For simplifying what follows, and without loss of generality, we refer our algorithms to complete graphs. If a graph is not complete we assume the existence of all the complementary edges with label $\{U\}$ (or U), if U is (is not) an interval; we call this a *trivial constraint*. The assumption does not require extra storage. In addition, we assume that the intervals of $\ell(e)$ are disjoint (WLOG). We call *length* of a constraint the difference between the maximum and the minimum values it allows and *range* R of a network the maximum length of its non-trivial constraints; clearly, $|\ell(e)| \in O(R)$.

A network is said to be *consistent* if the corresponding TCSP admits a solution. An edge $\{v_i, v_j\}$ is *arc-consistent* [12] iff for each $x_i \in D_i$ there is $x_j \in D_j$ such that the pair (x_i, x_j) satisfies the constraint $\ell(v_i, v_j)$. A path $(v_{i_1}, \ldots, v_{i_p})$, $p \geqslant 3$, is *path-consistent*

[16] iff for each $x_{i_1} \in D_{i_1}$ and $x_{i_p} \in D_{i_p}$ satisfying $\ell(v_{i_1}, v_{i_p})$ there are $p-2$ values x_{i_j} such that $(x_{i_j}, x_{i_{j+1}})$ satisfies $\ell(v_{i_j}, v_{i_{j+1}})$, for $j = 2, \ldots, p-1$.

A network is said to be *arc-consistent* iff all its edges are arc-consistent; analogously, it is said to be *path-consistent* iff all its paths are path-consistent. The set of networks that are path-consistent is denoted by $\mathcal{PC}$. Moreover, a network of n vertices is *i-consistent* [8], with $1 < i \leqslant n$, iff for any subset $V' \subseteq V$ such that $|V'| = i-1$ there exists a $(i-1)$-ple $\in U^{i-1}$ satisfying the constraints defined by the edges of the subgraph induced by V' and for each variable $v \in V \setminus V'$ the $(i-1)$-ple can be extended into a i-ple $\in U^i$ satisfying all constraints defined by the edges of the subgraph induced by $V' \cup \{v\}$.

We define the following operations on constraints. Let $A = \{I_1, \ldots, I_{|A|}\}$ and $B = \{J_1, \ldots, J_{|B|}\}$ be two constraints.

- $A \oplus B = \{K_i \mid K_i = I_j \cap J_k, \text{ for some } j \text{ and } k\}$.
  Intersection: only values allowed by both the constraints are admitted. The operation is commutative and associative.
- $A \otimes B = \{K_i \mid K_i = [a+c, b+d]$ for some $I_j = [a, b]$ and some $J_k = [c, d]\}$.
  Composition: values r are admitted such that $r = s + t$, where s is allowed by A and t is allowed by B. The operation is commutative and associative.

Also, we introduce a (reflexive) order in the set of all constraints $\mathcal{C}$. Given two constraints $C_1, C_2 \in \mathcal{C}$, we write $C_1 \sqsubseteq C_2$ if each pair allowed by C_1 is also allowed by C_2.

Lemma 2.1. *A network is path-consistent iff for each ordered m-ple $(v_{i_1}, \ldots, v_{i_m})$, $m \geqslant 3$, it results $\ell(v_{i_1}, v_{i_m}) \sqsubseteq \ell(v_{i_1}, v_{i_2}) \otimes \cdots \otimes \ell(v_{i_{m-1}}, v_{i_m})$.*

Proof. It easily follows from the definition of operation $\otimes$. □

Lemma 2.2. *A network is arc-consistent iff it is 2-consistent.*

Proof. It immediately follows from the definition of i-consistency for $i = 2$. □

Lemma 2.3. *A network is path-consistent iff it is 3-consistent.*

Proof. ($\Rightarrow$) Immediate.

($\Leftarrow$) By induction on the length of the path. By hypothesis the network is 3-consistent: it follows that each path of length 2 is path-consistent. Suppose now that each path of length m is path-consistent and consider a path $(v_{i_1}, \ldots, v_{i_{m+2}})$ of length $m+1$. By inductive hypothesis the path $(v_{i_1}, \ldots, v_{i_{m+1}})$ is path-consistent and by Lemma 2.1 it holds $\ell(v_{i_1}, v_{i_{m+1}}) \sqsubseteq \ell(v_{i_1}, v_{i_2}) \otimes \cdots \otimes \ell(v_{i_m}, v_{i_{m+1}})$. On the other hand by the base of the induction we know that $\ell(v_{i_1}, v_{i_{m+2}}) \sqsubseteq \ell(v_{i_1}, v_{i_{m+1}}) \otimes \ell(v_{i_{m+1}}, v_{i_{m+2}})$. So we obtain

$$\ell(v_{i_1}, v_{i_{m+2}}) \sqsubseteq \ell(v_{i_1}, v_{i_2}) \otimes \cdots \otimes \ell(v_{i_{m+1}}, v_{i_{m+2}})$$

which by Lemma 2.1 proves the thesis. □

An immediate consequence of the above lemma is the following.

Lemma 2.4. *A network is path-consistent iff for each triple* (v_i, v_j, v_k)*, with* $i \neq j$*,* $i \neq k$ *and* $j \neq k$*, it results* $\ell(v_i, v_k) \sqsubseteq \ell(v_i, v_j) \otimes \ell(v_j, v_k)$.

The following results characterizes the arc-consistency of augmented networks, defined at the beginning of this section.

Lemma 2.5. *An augmented network is arc-consistent iff all paths of length* 2 *originating from* v_0 *are path-consistent.*

Proof. ($\Rightarrow$) As the network is arc-consistent any edge (v_i, v_j) is such that for each $z_i \in D_i$ there is $z_j \in D_j$ such that (z_i, z_j) is allowed by $\ell(v_i, v_j)$. After the insertion of v_0 the above property can be expressed as $\ell(v_0, v_i) \sqsubseteq \ell(v_0, v_j) \otimes \ell(v_j, v_i)$.

($\Leftarrow$) Symmetric reasoning. □

3. Incremental path-consistency

We denote by $C(i, j)$ the constraint between variables v_i and v_j of the temporal network made path-consistent. Note that $C(i, j) \sqsubseteq \ell(v_i, v_j)$.

Given a network $\mathcal{N} = ((V, E), C) \in \mathcal{PC}$, algorithm `IPC` (Incremental Path-Consistency, Fig. 2) inserts a new constraint A on edge (v_i, v_j) and maintains the path-consistency of the network. This is obtained through suitable restrictions on the constraints of the network (we will use the term "edge restriction" as a synonymous of "constraint restriction"). Basically, the algorithm stores edges that have been restricted into a set Q and, for each restricted edge (v_k, v_l), restores the local consistency of the $2(n-2)$ paths (v_k, v_l, v_m) and (v_l, v_k, v_m), for $v_m \in V \setminus \{v_k, v_l\}$, by possible restrictions on edges (v_k, v_m) and (v_l, v_m) (function `LPC`, Fig. 3). Edge (v_k, v_m) is restricted if $C(k,m) \oplus (C(k,l) \otimes C(l,m)) \sqsubseteq C(k,m)$ (function `Revise`, Fig. 4, inspired to an analogous function in [12]). Newly restricted edges are added to Q.

```
1.  Algorithm IPC(N, i, j, A)
2.  input: a path-consistent network N, an edge (i, j), a constraint A
3.  side effect: inserts A on (i, j) and maintains the path-consistency of N
    by restricting its constraints
4.  begin
5.     Z := A ⊕ C(i, j)                    computes the effect of A
6.     if Z ≠ C(i, j) then
7.        C(i, j) := Z                     restricts (i, j)
8.        Q := {(i, j)}                    inserts restricted edge into Q
9.        while Q ≠ ∅ do                   unprocessed restricted edges?
10.          (k, l) := an element of Q     chooses an edge in Q
11.          Q := Q \ {(k, l)}             removes it from Q
12.          Q := Q ∪ LPC(k, l)            restores local consistency and increments Q
13.       endwhile
14.    endif
15. end
```

Fig. 2. Algorithm `IPC` maintains the path-consistency of $\mathcal{N}$ under the insertion of a new constraint A on (i, j).

```
1.  function LPC(i, j)
2.  input: edge (i, j) that has been restricted
3.  output: set S of restricted edges
4.  side effect: restores local consistency on each triangle on (i, j)
5.  begin
6.     S := ∅
7.     for k ∈ {1, ..., n} \ {i, j} do          for each non-incident vertex
8.        if Revise(i, j, k) then               (i, k) restricted?
9.           S := S ∪ {(i, k)}
10.       endif
11.       if Revise(j, i, k) then               (j, k) restricted?
12.          S := S ∪ {(j, k)}
13.       endif
14.    endfor
15.    return S
16. end
```

Fig. 3. Function LPC (Local Path-Consistency) propagates the effect of the constraint restriction on (i, j) to the other variables. It returns the set S of the newly restricted edges.

```
1.  function Revise(i, j, k)
2.  input: subgraph (triangle) where to re-enforce consistency
3.  output: true iff (i, k) is restricted
4.  begin
5.     Z := C(i, k) ⊕ (C(i, j) ⊗ C(j, k))     new constraint for (i, k)
6.     if Z = C(i, k) then                    no restriction?
7.        return false                        (i, k) not restricted
8.     else
9.        C(i, k) := Z
10.       return true                         (i, k) has been restricted
11.    endif
12. end
```

Fig. 4. Function Revise takes in input three indices i, j and k and makes the path-consistency of the path (i, j, k). The function returns true iff it restricts (i, k).

3.1. Analysis

We first notice that Revise runs in $O(R^2)$ time (the operations carried out at line 5 may take $O(R \log R)$ time for the intersection and $O(R^2)$ time for the composition). Also, each constraint can be restricted at most $O(R)$ times. As there are at most $O(n^2)$ constraints, the while cycle at lines 9–13 of IPC is executed at most $O(n^2 R)$ times. Thus the total running time of IPC is $O(n^2 R) \cdot \Theta(n) \cdot O(R^2) = O(n^3 R^3)$.

Theorem 3.1. *Given a network $\mathcal{N}$ with n vertices, algorithm* IPC *runs in time $O(n^3 R^3)$, where R is the maximum between the range of $\mathcal{N}$ and the length of the new constraint.*

Let $\mathcal{N}$ and $\mathcal{N}'$ be two networks such that $\mathcal{N} \equiv \mathcal{N}'$ and $\mathcal{N}' \in \mathcal{PC}$. Let us denote by $\mathcal{M} + A_{ij}$ the network obtained through the addition of the constraint A_{ij} between v_i and v_j in $\mathcal{M}$ and by $\texttt{IPC}(\mathcal{M}, i, j, A_{ij})$ the network built by applying $\texttt{IPC}(i, j, A_{ij})$ to $\mathcal{M}$. In

order to prove the correctness of IPC we first observe that $\texttt{IPC}(\mathcal{N}', i, j, A_{ij}) \in \mathcal{PC}$. This can be easily proved by contradiction, exploiting Lemma 2.4.

Theorem 3.2. *Given a path-consistent network $\mathcal{N}'$ and a new constraint A_{ij},* $\texttt{IPC}(\mathcal{N}', i, j, A_{ij}) \in \mathcal{PC}$.

For proving that $\texttt{IPC}(\mathcal{N}', i, j, A_{ij}) \equiv \mathcal{N} + A_{ij}$, we first give the following theorem, that can be easily proved by induction on the number of restrictions carried out by Revise.

Theorem 3.3. *Given a path-consistent network $\mathcal{N}'$ and a new constraint A_{ij},* $\texttt{IPC}(\mathcal{N}', i, j, A_{ij}) \equiv \mathcal{N}' + A_{ij}$.

The theorem on the equivalence now follows.

Theorem 3.4. *If $\mathcal{N} \equiv \mathcal{N}' \in \mathcal{PC}$, then for any constraint A_{ij},* $\texttt{IPC}(\mathcal{N}', i, j, A_{ij}) \equiv \mathcal{N} + A_{ij}$.

Proof. By Theorem 3.3 we know that $\texttt{IPC}(\mathcal{N}', i, j, A_{ij}) \equiv \mathcal{N}' + A_{ij}$. Let S be the set of solutions of $\mathcal{N}$ and of $\mathcal{N}'$. If all n-ples in S satisfy the new constraint A_{ij} the theorem holds. Otherwise, let $S' \subset S$ be the set of n-ples in S satisfying A_{ij}. Note that the insertion of a new constraint cannot introduce new solutions in a network, hence the sets of solutions of $\mathcal{N}' + A_{ij}$ and of $\mathcal{N} + A_{ij}$ are both contained in S. But, being S' the set of solutions satisfying all the constraints, it follows that it is the set of solutions of $\mathcal{N}' + A_{ij}$ and of $\mathcal{N} + A_{ij}$, which are therefore equivalent. □

Passing to consider the running time of IPC over a sequence of constraint insertions, we (surprisingly) get the result that $\Theta(n^2)$ executions of IPC globally take $O(n^3R^3)$ time. This can be proved through *amortized analysis* [23]. Let us denote by q_i the number of pairs inserted in Q during the ith execution of IPC and by $\bar{q}_i = \sum_{j=1}^{i} q_i$ the total number of pairs inserted in Q during the first i executions of IPC. We choose the following potential:

$$\Phi_i = 2(n-2)(Ri - \bar{q}_i).$$

Notice that $\Phi_0 = 0$; moreover $\Phi_{n^2} = 2(n-2)(Rn^2 - \bar{q}_{n^2}) > 0$ because $\bar{q}_{n^2} < Rn^2$.

Theorem 3.5. *Any sequence of $\Theta(n^2)$ executions of* IPC *runs in $O(n^3R^3)$ worst case time.*

Proof. Let us first compute the (relative) cost of the sequence in terms of number of executions of Revise. The amortized relative complexity of the ith execution is $a'_i = t'_i + \Phi_i - \Phi_{i-1} = t'_i + 2(n-2)R - 2(n-2)q_i$. Now note that t'_i is equal to $2(n-2)q_i$. From this it follows that the amortized relative cost is $a'_i = 2(n-2)R \in O(nR)$. Thus, a sequence of $\Theta(n^2)$ executions has amortized relative complexity $O(n^3R)$. The thesis follows by recalling that Revise runs in $O(R^2)$ worst case time. □

Notice that, as IPC does not care about unconstrained variables, we can insert new ones into a path-consistent network in $O(1)$ time per insertion.

4. Extensions

We briefly show how algorithm IPC can be adapted to maintain the path-consistency of more general networks of temporal constraints, and how it can be specialized for maintaining the arc-consistency of such networks.

In more general networks, their variables of can independently be points or intervals. In the previous sections we considered networks with point variables. Interval variables are defined on domains whose values are intervals, e.g., $\{(a, b) \in \mathbb{Z}^2 \mid a \leq b\}$. In this case constraints are *qualitative*, e.g., an interval is before another interval, or a point is to the right of an interval etc. All networks with qualitative constraints are trivially arc-consistent. Also, large and significant classes of networks with qualitative constraints can be directly solved through the path-consistency (see [24,26] for the point algebra and [17] for a significant subclass of the interval algebra).

The intersection operation is immediately extended to the new case (only constraints of the same kind can be intersected), while the extension of the composition operation is more involved, needing to use the so-called "composition tables" [1,25]. Algorithm IPC maintains the path-consistency of networks of general temporal constraints, as long as the composition and the intersection needed in function Revise are correspondingly generalized. Thus all results for the TCSP directly extend to networks of general temporal constraints without changes. However, in the particular case of qualitative constraints only, the time complexity of Revise becomes constant because qualitative constraints have fixed size, and this cuts off the R^3 factor in the time costs obtained by Theorems 3.1 and 3.5.

In the case of the incremental arc-consistency, Lemma 2.5 suggests how to change algorithm IPC for maintaining the arc-consistency of a network of general temporal constraints: just maintain the path-consistency of the paths originating from v_0. The new algorithm, named IAC, can be obtained from IPC by appropriately changing function LPC, so that it will propagate along paths originating from 0 the effect of the constraint restriction on (i, j).

The analysis of algorithm IPC can be repeated for IAC using the potential function

$$\Phi_i = 2(i - \bar{r}_i) + (n - 2)\left(i\frac{R}{n} - \bar{s}_i\right),$$

where:

- $\bar{r}_i = \sum_{h=1}^{i} r_h$, r_h being the number of pairs (h, k), $h, k \neq 0$, inserted into Q during the hth execution of IAC;
- $\bar{s}_i = \sum_{h=1}^{i} s_h$, s_h being the number of pairs $(0, k)$, $k \neq 0$, inserted into Q during the hth execution of IAC.

The worst case running time of IAC can be obtained by the same technique used in the proof of Theorem 3.1. Note however that since IAC only restricts constraints involving v_0, and there are at most $O(n)$ such constraints, the cycle at line 9 of IPC is executed at most $O(nR)$ times.

Theorem 4.1. *Given a temporal network with n vertices, algorithm* IAC *maintains its arc-consistency running in* $O(R^3)$ *amortized time and in* $O(n^2R^3)$ *worst case time.*

In [11] it is shown that the networks computed by our algorithms are maxima over the order introduced by Montanari [16]. Thus, it is not possible to improve the performance of IPC and of IAC by reducing the number of restrictions on the constraints.

References

[1] J.F. Allen, Maintaining knowledge about temporal intervals, Comm. ACM 26 (11) (1983).
[2] F.D. Anger, R.V. Rodriguez, Effective Scheduling of Tasks Under Weak Temporal Interval Constraints, in: Lecture Notes in Comput. Sci., Vol. 945, Springer, Berlin, 1995.
[3] J. Carmo, A. Sernadas, A temporal logic framework for a layered approach to systems specification and verification, in: Proceedings of the Conference on Temporal Aspects in Information Systems, France, AFCET, 1987, pp. 31–47.
[4] R. Cervoni, A. Cesta, A. Oddi, Managing dynamic temporal constraint networks, in: Proceedings of AIPS '94, 1994.
[5] F. d'Amore, F. Iacobini, On-line algorithms for networks of temporal constraints, in: R.H. Möhring (Ed.), Proc. of the 23rd Int. Work. on Graph-Theoretic Concepts in Computer Science, WG '97, in: Lecture Notes in Comput. Sci., Vol. 1335, Springer, Berlin, 1997, pp. 144–156.
[6] E. Davis, Private communication reported in [7], 1989.
[7] R. Dechter, I. Meiri, J. Pearl, Temporal constraint networks, Artificial Intelligence 49 (1991).
[8] E.C. Freuder, A sufficient condition for backtrack-free search, J. ACM 29 (1982).
[9] C.-C. Han, C.H. Lee, Comments on Mohr and Henderson's path consistency algorithms, Artificial Intelligence 36 (1988).
[10] C.-C. Han, K.-J. Lin, J.W.-S. Liu, Scheduling jobs with temporal distance constraints, SIAM J. Comput. 24 (5) (1995) 1104–1121.
[11] F. Iacobini, On-line algorithms for maintaining the consistency in network of temporal constraints, Master's Thesis, School of Electrical Engineering, Università di Roma "La Sapienza", 1996 (in Italian).
[12] A.K. Mackworth, Consistency in networks of relations, Artificial Intelligence 8 (1977).
[13] Z. Manna, A. Pnueli, Verification of concurrent programs: Temporal proof principle, in: D. Kozen (Ed.), Logics of Programs, Proceedings 1981, in: Lecture Notes in Comput. Sci., Vol. 131, Springer, Berlin, 1981, pp. 200–252.
[14] Z. Manna, A. Pnueli, Verification of concurrent programs: the temporal framework, in: R.S. Boyer, J.S. Moore (Eds.), The Correctness Problem in Computer Science, Academic Press, New York, 1981, pp. 215–273.
[15] I. Meiri, Combining qualitative and quantitative constraints in temporal reasoning, in: Proc. of the 10th National Conference of the American Association for Artificial Intelligence, AAAI '91, 1991.
[16] U. Montanari, Networks of constraints: Fundamental properties and applications to picture processing, Inform. Sci. 7 (1974).
[17] B. Nebel, H.J. Bürckert, Reasoning about temporal relations: a maximal tractable subclass of Allen's interval algebra, J. ACM 42 (1995).
[18] J.S. Ostroff, Temporal Logic of Real-Time Systems, Research Studies Press, 1990.
[19] E. Schwalb, R. Dechter, Processing disjunctions in temporal constraint networks, Artificial Intelligence 93 (1997) 29–61.
[20] E. Schwalb, Ll. Vila, Temporal constraints: A survey, Constraints Internat. J. 3 (2/3) (1998) 129–149.
[21] A.U. Tansel, J. Clifford, S. Gadia, S. Jajodia, A. Segev, R. Snodgrass (Eds.), Temporal Databases: Theory, Design, and Implementation, Benjamin/Cummings, Redwood City, CA, 1993.
[22] A.U. Tansel, J. Clifford, S. Gadia, S. Jajopia, A. Segev, D.R. Snodgrass, Temporal Databases, Series on Database Systems and Applications, Benjamin/Cummings, Redwood City, CA, 1993.

[23] R.E. Tarjan, Amortized computational complexity, SIAM J. Algebraic Discrete Methods 6 (2) (1985) 306–318.
[24] P. van Beek, Reasoning about qualitative temporal information, Artificial Intelligence 58 (1992).
[25] M. Vilain, H.A. Kautz, Constraint propagation algorithms for temporal reasoning, in: Proc. of the 5th National Conference of the American Association for Artificial Intelligence, AAAI '86, 1986.
[26] M. Vilain, H.A. Kautz, P. van Beek, Constraint propagation algorithms for temporal reasoning: A revised report, in: D.S. Weld, J. de Kleer (Eds.), Readings in Qualitative Reasoning About Physical Systems, Morgan Kaufman, San Mateo, CA, 1989.