, , 1–20 ()

Temporal Constraints: A Survey *

EDDIE SCHWALB AND LLUíS VILA eschwalb@ics.uci.edu, vila@iiia.csic.es
Information and Computer Science, University of California, Irvine, USA and Institute for Research in Artificial Intelligence (IIIA)–CSIC, Catalonia, Spain

Editor:

Abstract. *Temporal Constraint Satisfaction* is an information technology useful for representing and answering queries about the times of events and the temporal relations between them. Information is represented as a *Constraint Satisfaction Problem* (CSP) where variables denote event times and constraints represent the possible temporal relations between them. The main tasks are two: (i) deciding consistency, and (ii) answering queries about scenarios that satisfy all constraints. This paper overviews results on several classes of Temporal CSPs: *qualitative interval*, *qualitative point*, *metric point*, and some of their combinations. Research has progressed along three lines: (i) identifying tractable subclasses, (ii) developing exact search algorithms, and (iii) developing polynomial-time approximation algorithms. Most available techniques are based on two principles: (i) enforcing local consistency (e.g. path-consistency), and (ii) enhancing naive backtracking search.

Keywords: Temporal Constraints, Temporal Reasoning, Constraint Processing.

1. Introduction

A *Constraint Satisfaction Problem* (CSP) is a set of *constraints* over a set of *variables*, where each variable has a set of possible values called its *domain*. Each constraint specifies the allowed assignments for a subset of variables. A *Temporal CSP* (TCSP) is a particular class of CSP where variables represent times and constraints represent sets of allowed temporal relations between them . Consider the following example:

> A patient requires three medical exams, each followed within 12 hours by a treatment session. Exams and treatments cannot overlap. Both are completed within 4 hours and must be at least 8 hours apart. The exams require resources available from the 8th to the 12ve and from the 20th to the 21st of this month.

We are interested in answering queries such as the following: find a feasible schedule (if any), find all feasible schedules, what are the feasible times for an exam or a treatment ?, what are the feasible relations between two exams or treatments ?, what are the feasible relations between all exams and treatments ?.

Different TCSPs are defined depending on the time entity that variables can represent, namely time points, time intervals, durations (i.e. distances between time points) and the class of constraints, namely qualitative, metric or both. For example, the constraint "exams and treatments cannot overlap" is a qualitative

* This work has been partially supported by the Spanish CICYT grant *REST* (TIC-96-0721-C02-02)

interval one. The constraint "from the 8th to the 12ve and from the 20th to the 21st" is a metric point one.

This paper overviews the results on deciding satisfiability of and answering queries on TCSPs. The classes of TCSPs surveyed are *qualitative point* (Vilainetal89), *qualitative interval* (Allen83), *metric point* (Dechteretal91) and some of their combinations (KautzLadkin91, Meiri96).

The paper is organised as follows. Section 2 presents general definitions and techniques for TCSPs. Section 3, 4 and 5 survey qualitative point, qualitative interval and metric TCSPs respectively. Section 6 surveys the most relevant of their combinations. We shall assume the reader is familiar with CSP notions and techniques (look at (Dechter92) for a survey).

2. Generalities

In this section we discuss the particularities of Temporal CSPs with respect to standard CSP.

Variables represent either time points, time intervals or durations. **Domains** are defined on a single set whose structure is usually isomorphic to either the set of integers or rationals. We shall refer to is as *time-structure*. The domain of both time point and duration variables is this set whereas time interval variables the domain is the set of *ordered pairs* of this set. Different classes of **constraints**, namely qualitative, metric,... are characterised by the underlying **set of basic temporal relations** (henceforth **BTR**). All **BTR** ssatisfy two conditions: (i) its elements are mutually exclusive, and (ii) the union of the elements is the universal constraint.

All TCSP constraints are binary and have the form $C_{ij} = \{r_1, \dots, r_k\}$ where X_i, X_j are variables, $k > 0$, and $r_1, \dots, r_k \in$ **BTR**. Their interpretation is

$$(X_i\ r_1\ X_j) \vee \cdots \vee (X_i\ r_k\ X_j)$$

It is possible to have **unary** metric point constraints but they can be regarded as binary constraints. To this purpose a special variable X_0 whose domain has a single element $D_0 = \{0\}$ is introduced. Now the unary constraint $C_i = \{r_1, \dots, r_k\}$ is then expressed as the binary constraint $C_{0i} = \{r_1, \dots, r_k\}$.

The form of temporal constraints is a fundamental particularity of TCSP. Whereas constraints in standard CSP are described by their extensions, TCSP constraints are intentionally described in terms of **BTR** elements. Now, solutions can be computed by processing **BTR** subsets instead of processing specific extensions.

2.1. Solutions

A constraint with a single disjunct is called **singleton**. A **singleton labelling** of a TCSP assigns to each pair of variables X_i, X_j a basic temporal relation r such that $r \subseteq C_{ij}$. The **solutions** of a TCSP are its *consistent* singleton labellings. Consistency of a singleton labelling is defined according to the specific semantics of each TCSP class.

Because the solutions of a TCSPs are singleton labellings instead of variable instantiations, the notion of feasible value is replaced by the notion of **feasible relation**. A relation $r \in \mathbf{BTR}$ is feasible for the pair (X_i, X_j) iff there exists one solution where r is assigned to this pair. Notice that in qualitative TCSP, r feasible for (X_i, X_j) implies that $r \in C_{ij}$ whereas in the metric case it is relaxed to $r \subseteq C_{ij}$. The *minimal constraint* C_{ij}^{min} is the set of feasible relations between X_i and X_j. As in CSP, a TCSP where all constraints are minimal is said to be **minimal** and, given a TCSP, it is always possible to find an *equivalent*, minimal TCSP.

2.2. Lines of Research

TCSP focussed on two problems: (i) deciding consistency, which is closely related to the task of finding one solution, and (ii) finding the minimal representation which, in some sense, represents all solutions. All other queries mentioned in section 1 are at least as difficult as deciding consistency. Computing the minimal representation is harder that deciding consistency but allows answering many queries at a low cost in general.

Research has progressed along three lines:

- *Identifying tractable subclasses and developing specialised algorithms for them.* These classes are defined by two sorts of parameters: (i) properties of the constraint graph (e.g. degree, width, ...), and (ii) the class of constraints.

- *Enhancing search algorithms.* There are two well-known methods: (i) backtracking search, and (ii) iterative refinement search such as GSAT The former is guaranteed to terminate with the correct answer but does not scale up due to its exponential complexity. The latter scales up well but is not guaranteed to terminate with a solution.

- *Developing polynomial-time approximation algorithms that are sound although not complete.*

2.3. Techniques

The building blocks for temporal constraint processing techniques are the complement, converse, intersection and composition **constraint operators**. They are set-theoretically defined in terms of their definition over **BTR** which is specific to each TCSP class.

Enforcing Local Consistency. The idea is to enforce some degree of local consistency i.e. k-consistency for some $k \leq n$, to eliminate non-feasible labels from the problem constraints. In most cases, enforcing local consistency can be done in polynomial time. The most simple and popular local-consistency notions are *arc-consistency* and *path-consistency* (see (Dechter92)). **Arc-consistency**, or 2-consistency, is only applicable on constraints that involve the X_0. **Path-consistency** is less local since involves all paths between any two variables. Nev-

Algorithm PC
1. $Q \leftarrow \{(i,k,j) | (i<j) \text{ and } (k \neq i,j)\}$
2. **while** $Q \neq \{\}$ **do**
3. select and delete a path (i,k,j) from Q
4. **if** $C_{ij} \neq C_{ik} \circ C_{kj}$ **then**
5. $C_{ij} \leftarrow C_{ij} \cap (C_{ik} \circ C_{kj})$
6. **if** $C_{ij} = \{\}$ **then** exit (inconsistency)
7. $Q \leftarrow Q \cup \{(i,j,k),(k,i,j) \mid 1 \leq k \leq n, i \neq k \neq j\}$
8. **end-if**
9. **end-while**

end-algorithm

Figure 1. Algorithm PC for enforcing path-consistency.

ertheless it is well-know that it is enough to enforce 3-consistency to guarantee it.

Path-consistency can be highly effective and sometimes turns out be enough to decide consistency, e.g. in the case of a singleton labelling. The classical algorithm PC shown in figure 2.3 enforces path-consistency in $O(v^3)$ steps.

Search Methods. Since TCSPs are in general intractable, complete algorithms must perform some sort of search. The notion of *partial singleton labelling* is defined as an assignment where some constraints are labelled by a singleton and some are not. The search space of a TCSP is defined over all possible partial singleton labellings. Practical backtracking algorithms for TCSP proceed by forward checking and variable assignment as we show in figure . Forward checking is implemented by enforcing local consistency. Variable assignment consists of a mere selection of a **BTR** in the constraint at hand. Intuitively, a backtracking search algorithm successively labels each (disjunctive) constraint with one of its **BTR**s as long as the resulting partial labelling is consistent. Once inconsistency is detected, the algorithm backtracks. The number of dead-ends encountered strongly depends on strategy for deciding on the ordering. For most tractable subclasses, however, enforcing path-consistency at step 2 is sufficient to guarantee that a solution can be found in a backtrack free manner.

3. Qualitative Point Constraints

In qualitative point TCSP variables represent time points and $\mathbf{BTR} = \{<,=,>\}$. Three algebras have been studied:

name	abbrv	relations
basic point algebra	**BPA**	$<,=,>,?$
convex point algebra	$\mathbf{PA}_c$	$\emptyset,<,=,>,\leq,\geq,?$
point algebra	**PA**	$\emptyset,<,=,>,\leq,\geq,?,\neq$

Algorithm Backtracking
1. *Depth* $\leftarrow 0$;
2. Apply PC; this removes some redundant **BTR**s.
3. **if** inconsistency was detected **then**
4. **if** *Depth* $= 0$ **then** *exit* with failure.
5. Undo the last **BTR**labelling.
6. *Depth* $\leftarrow$ *Depth* $- 1$; Go to step 8.
7. **if** all constraints are **BTR**s **then** *exit* with the solution.
8. Replace (non-deterministically) a disjunctive constraint by a single **BTR**.
9. *Depth* $\leftarrow$ *Depth* $+ 1$; Go back to step 2.

end-algorithm

Figure 2. An Scheme for Practical Backtracking Algorithms for solving TCSPs.

The most relevant contributions on **PA** constraints are *IxTeT*, vanBeek's and *TimeGraph-II*.

*3.1. Basic Point Algebra (***BPA***)*

A **BPA** TCSP is either inconsistent or represents a strict partial order. If it is consistent then the non-universal input constraints are *minimal*. Thus, finding a solution is equivalent to finding a total order. This can be done applying *topological sort* in $O(v+e)$ steps. Enforcing path-consistency correctly decides consistency and computes the minimal constraints but requires $O(v^3)$ steps.

3.2. Convex Point Algebra ($\mathbf{PA_c}$*)*

The set of constraints can be represented as a weighted, directed graph using the following translation:

$$\begin{array}{|llll|} \hline x_i = x_j & \textit{translates to} & x_i \leq x_j, & x_j \leq x_i \\ x_i \leq x_j & \textit{translates to} & x_i \xrightarrow{+\infty} x_j, & x_j \xrightarrow{0} x_i \\ x_i < x_j & \textit{translates to} & x_i \xrightarrow{+\infty} x_j, & x_j \xrightarrow{-\epsilon} x_i \\ \hline \end{array}$$

Consequently, for the restricted case in which the relations $<, >$ are not allowed, finding a solution accounts for finding the *shortest-path* using Dijkstra's algorithm in $O(v^2)$ steps. Otherwise, we need to use Floyd-Warshall *all-pairs shortest-paths* algorithm which is equivalent to enforcing path-consistency in $O(v^3)$ steps (LadkinMaddux88, Cormenetal90).

3.3. *Point Algebra (**PA**)*

3.3.1. Deciding Consistency In *IxTeT* (GhallabMounir89), a **PA** TCSP is translated into a graph with $\leq$ and $\neq$ edges only. The translation is as follows:

$\emptyset$	$\mapsto$	"the problem is inconsistent"	?	$\mapsto$	no edge
$<$	$\mapsto$	$\xrightarrow{\leq}, \xrightarrow{\neq}$	$\leq$	$\mapsto$	$\xrightarrow{\leq}$
$>$	$\mapsto$	$\xleftarrow{\leq}, \xleftarrow{\neq}$	$\geq$	$\mapsto$	$\xleftarrow{\leq}$
$=$	$\mapsto$	the two vertices are "collapsed" into a single vertex	$\neq$	$\mapsto$	$\xrightarrow{\neq}$

The resulting $\leq$-$\neq$-graph, has the following property (GhallabMounir89):

> "... A $\leq$-$\neq$-graph is consistent iff no pair of vertices connected by a $\neq$ edge are involved in a loop through $\leq$ edges."

It is checked by collapsing every $\leq$-loop into a single vertex. If two collapsed vertices are connected by a $\neq$ edge then the TCSP is inconsistent.

Identifying $\leq$-loops is equivalent to identifying *strongly connected components* (SCC) as defined in graph theory (vanBeek89). Efficient algorithms for computing SCCs are based on *two-way topological sort* and take $O(v+e)$ steps (Tarjan72). *TimeGraph-II* follows the same approach ((GereviniSchubert95a) theorems 2.8, subsection 3.1 and theorem 3.2).

3.3.2. Finding a Solution Once a **PA**-TCSP is free of $\leq$-loops, a solution can be easily computed using topological sort in $O(v+e)$ steps.

3.3.3. Answering Queries on Feasible Relations All feasible relations can be determined by computing the minimal representation. Path-consistency is proven to find the feasible relations for **BPA** and **PA**$_c$ but it is not complete for **PA**. Figure 3.3.3 shows a counter-example, commonly known as the *forbidden subgraph* (vanBeekCohen90).

The minimal representation can be obtained by enforcing *4-consistency*, however van Beek proposed a more practical approach based on the following observation: the forbidden subgraph must be included in every **PA** TCSP which is path-consistent but not minimal (vanBeek92)[1]. This property leads to the following two step algorithm:

1. Enforce path-consistency. It requires $O(v^3)$ steps.
2. Search systematically for the *forbidden subgraphs* and update the labels. It requires $O(e_{\neq} v^2)$ steps, where $e_{\neq}$ is the number of $\neq$ constraints.

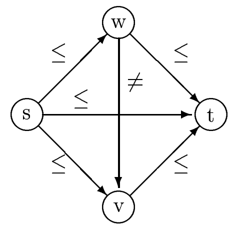

Figure 3. The unique non-minimal path-consistent **PA** TCSP.

Although the worst case complexity of this algorithm is $O(v^4)$, it has been empirically observed that the path-consistency step dominates the computation (vanBeek90c).

This algorithm can be adapted to process *dynamic problems*, i.e. problems where the variables and constraints added and/or removed while feasible relation queries are posed. The idea is maintaining an internal representation that approximates a complete graph, allows efficient query answering and supports incremental update of temporal constraints. Systems such as *IxTeT* or *TimeGraph-II* use internal representations that take advantage of the inherent structure of temporal information.

Using an indexed spanning tree *IxTeT*'s internal representation is built by (i) computing the *maximal weight spanning tree*, (ii) adding some residual edges between different branches of the tree, and (iii) labelling the nodes with an index.

The *indexed spanning tree* is computed in $O(v+e)$ steps and experimental results show that both retrieval and update take linear time (GhallabMounir89). Although *IxTeT* has a clear practical interest, it is not complete: it fails to compute the correct answer when the input TCSP includes the *forbidden subgraph*.

Arranging Time Points into Chains *TimeGraph-II* (GereviniSchubert95a) is specially tailored to domains where chain-like aggregates are dominant such as natural language systems. The internal data structure, called *time-graph*, is organised in chains and the algorithms for building and maintaining the time-graph are designed to maximise the length of these chains. Building a *time-graph* involves three steps: (i) ranking the vertices, (ii) computing next-greater links, and (iii) propagating $<$ through *forbidden graphs*.

In *TimeGraph-II* the feasible relation between two events can be computed in $O(e+v)$ However, if the events involved in the query belong to either the same chain or are related by a $\neq$, the query can be answered in constant time.

3.4. Summary

Worst-case bounds have been established for the tasks of deciding consistency, finding a solution and generating the minimal TCSP (vanBeek90c). These results

	Task	Time Cost worst-case	Time Cost average-case
van Beek	Deciding Consistency by collapsing SCCs.	$O(v+e)$	
	All feasible relations by PC+*forbidden graphs.*	$O(\max(v^3, e_{\neq}v^2))$	$O(v^3)$
IxTeT	Building *IxTree*	$O(v+e)$	
	Feasible relation		$O(v)$
	Adding new relation		$O(v)$
TimeGraph-II	Building a *time-graph*	$O(e+v)$	$O(e)$
	Feasible relation	$O(e+v)$	$O(v)$

Figure 4. Summary of qualitative point TCSP results.

are difficult to contrast with the empirical evaluation of algorithms optimised to answer feasible relation queries for a restricted domain. *IxTeT* experiments show that a structure based on an indexed maximal spanning tree efficiently supports both feasible relation queries and dynamic updating of constraints. *TimeGraph*'s major improvement upon *IxTeT* was in providing correct answers when the TCSP includes $<$-paths and *forbidden graphs.* Table 3.4 summarises these results.

4. Qualitative Interval Constraints

In *qualitative interval TCSP* variables represent time intervals and

$$BTR = \left\{ \begin{array}{c} \textbf{before}, \textbf{after}, \textbf{meets}, \textbf{met\_by}, \\ \textbf{overlaps}, \textbf{overlaps\_by}, \textbf{during}, \textbf{contains}, \textbf{equals}, \\ \textbf{starts}, \textbf{started\_by}, \textbf{finishes}, \textbf{finished\_by} \end{array} \right\}$$

IA **BTR**s can be expressed by conjunctions of **PA** relations as described in figure 4, where X^-,X^+ are the beginning and end points of the interval X respectively.

Indefinite information is expressed as disjunctions of **BTR** elements. In the initial example, to express the statement “Exams and treatments cannot overlap” we need to specify that for each exam and treatment, the time interval of the exam is either `Before` or `Meets` after the interval of the treatment. This is denoted by $I_{exam_i}\{\texttt{Before}, \texttt{After}\}I_{treatment_j}$. The total number of possible indefinite relations is $2^{13} = 8192$.

Operators are defined as usual (see section 2). The composition of pairs of **BTR** elements is given by a 13×13 table in (Allen83).

Relation	PA representation	Inverse	PA representation
X `Before` Y	$X^+ < Y^-$	X `After` Y	$Y^+ < X^-$
X `Equal` Y	$X^- = Y^- \wedge X^+ = Y^+$	X `Equal` Y	$X^- = Y^- \wedge X^+ = Y^+$
X `Meets` Y	$X^+ = Y^-$	X `Met_by` Y	$X^- = Y^+$
X `Overlaps` Y	$X^- < Y^- \wedge X^+ < Y^+$ $\wedge\, Y^- < X^+$	X `Overlapped_by` Y	$X^- > Y^- \wedge X^+ > Y^+$ $\wedge\, X^- < Y^+$
X `During` Y	$Y^- < X^- \wedge X^+ < Y^+$	X `Contains` Y	$X^- < Y^- \wedge Y^+ < X^+$
X `Starts` Y	$X^- = Y^- \wedge X^+ < Y^+$	X `Started_by` Y	$Y^- = X^- \wedge Y^+ < X^+$
X `Finishes` Y	$X^- > Y^- \wedge X^+ = Y^+$	X `Finished_by` Y	$Y^- > X^- \wedge Y^+ = X^+$

Figure 5. The PA representation of the 13 IA relations.

4.1. Complexity of Tasks and Tractable Classes

Deciding consistency (and computing a solution) of Interval TCSPs is in NP-complete (Vilainetal89).

The first and most simple tractable class identified was the subclass which can be represented by **PA** TCSPs, called **Pointisable TCSPs** (Vilainetal89). Linear time algorithms for processing this class were developed (GereviniSchubert93, DrakengrenJonsson96). For this subclass, enforcing path-consistency correctly decides consistency and enforcing 4-consistency computes the minimal constraints (vanBeek92).

Macro Relations can be used to describe tractable classes. By shifting one of the four interval endpoints leaving the other three fixed, a partial order on the 13 relations is obtained (Nokel89a). This partial order was used to represent coarse temporal information through the notion of neighbourhood (Freksa92). Two relations are *conceptual neighbours* if they can be derived from each other by shifting to the right one of the four interval endpoints leaving the other three fixed. A set of relations forms a *conceptual neighbourhood* if each relation is a conceptual neighbour of at least one other relation in the set. It is convenient to consider the following macro relations:

$\cap = \{ m, mi, o, oi, s, si, f, fi, d, di, =\}$	
$\alpha = \{m, o\}$,	$\alpha^{-1} = \{mi, oi\}$
$\subset = \{s, f, d\}$,	$\subset^{-1} = \{si, fi, di\}$
$\prec = \{b\}$	$\succ = \{bi\}$
$\prec\!\cap = \{b, \cap\}$,	$\cap\!\succ = \{\cap, bi\}$
$? = \{b, \cap, bi\}$,	$\prec\!\!\succ = \{b, bi\}$

Δ-classes The Δ-*notation* is used to describe subclasses of the Interval Algebra which are based on the macro relations described above. $\Delta = \{m_1, \ldots, m_k\}$ denotes the set of allowed *macro relations* that can be used to label constraints. Note that $\{m_1, \ldots, m_k\}$ may not describe an algebra nor a sub-algebra, and they need not be

Class Name	Relations Used (Δ-class)	Reference
Interval Orders	$\{\prec,\ \succ,\ \cap\}$	(Fishburn70)
Interval Graphs	$\{\prec\!\!\succ,\ \cap\}$	(GilmoreHoffman64, FulkersonGross65, BoothLeuker76, KorteMohring89)
Circle (overlap) Graphs	$\{\ \{\alpha, \alpha^{-1}, =\},\ \{\prec, \succ, \subset, \subset^{-1}\}\ \}$	(Gaboretal89, Bouchet87)
Interval Containment Graphs	$\{\ \{\subset, \subset^{-1}\},\ \{\prec, \succ, \alpha, \alpha^{-1}, =\}\ \}$	(GolumbicScheinerman89)
Po-sets of dimension 2	$\{\ \{\subset\},\ \{\subset^{-1}\},\ \{\prec, \succ, \alpha, \alpha^{-1}, =\}\ \}$	(DushnikMiller41, Bakeretal72, GolumbicScheinerman89)

Figure 6. Tractable classes.

closed under converse, composition and intersection. An interesting result regarding the complexity of a very simple restricted subclass of the Interval TCSP is as follows:

THEOREM 1 *(GolumbicShamir91) Deciding consistency of an Interval TCSP in which the relations are* $\Delta = \{\prec,\ \succ,\ \cap, ?\}$ *and* $\Delta = \{\prec\cap,\ \cap\succ,\ \prec\!\!\succ,\ ?\}$ *is in NP-complete.*

Despite the fact that the restricted class described above is intractable, several Δ-based tractable subclasses were identified. Table 4.1, originally given in (GolumbicShamir91), describes a number of well-known recognition problems in graph theory and partially ordered sets which can be viewed as restricted subclass of the Interval Algebra. A linear time algorithm for deciding consistency of $\Delta = \{\prec, \succ, \cap, \prec\cap, \cap\succ, ?\}$ is given in (GolumbicShamir91). A cubic time algorithm for deciding consistency of $\Delta = \{\prec, \succ, \cap, \prec\!\!\succ\}$ is given in (GolumbicShamir91). Efficient algorithms for $\Delta = \{\prec, \succ, \cap\}$ and $\Delta = \{\prec\!\!\succ, \cap\}$ can be found in (BelferGolumbic90, BelferGolumbic91).

The unique **maximal tractable subclass** that includes *all* 13 relations was identified (NebelBuerckert95) . Define three *atomic formulas*: $(X_i \leq X_j)$, $(X_i = X_j)$ and $(X_i \neq X_j)$, where X_i, X_j are point variables and $i < j$. An *ORD-Horn* clause is a disjunction of these *atomic formulas*, and an *ORD-Horn* formula is a conjunction of *ORD-Horn* clauses. The class of relations which can be described by *ORD-Horn* formulas, denoted $\mathcal{H}$, is closed under converse, intersection and composition (NebelBuerckert95).

EXAMPLE: The ORD-Horn representation of the (pointisable) relation $X\{d, o, s\}Y$ is the formula

$$\begin{aligned}&\{(X^- \leq X^+),\ (X^- \neq X^+),\ (Y^- \leq Y^+),\ (Y^- \neq Y^+),\\ &\ (X^- \leq Y^+),\ (X^- \neq Y^+),\ (Y^- \leq X^+),\ (Y^- \neq X^+),\ (X^+ \leq Y^+),\ (X^+ \neq Y^+)\}\end{aligned}$$

where X^-, X^+ and Y^-, Y^+ denote the end points of the intervals X and Y respectively. The relation $(X^- \neq Y^- \ \vee\ X^+ \neq Y^+)$, the complement of $X\{=\}Y$, is in $\mathcal{H}$ but is not pointisable. □

THEOREM 2 *(NebelBuerckert95, Ligozat96)*

- *A sub-algebra S is tractable iff the closure of S under the converse, intersection and composition operators is tractable.*
- *$\mathcal{H}$ is the unique maximal tractable subclass and*
- *enforcing path-consistency decides consistency for $\mathcal{H}$.*

A simplified proof for this theorem is presented in (Ligozat96). In addition a backtrack free algorithm for computing scenarios and a more general theory of relations between linear orders were developed (Ligozat96).

Twelve maximal tractable subclasses that do not use all 13 basic relations were characterised (DrakengrenJonsson96). Four of these can express *sequentiability* of intervals, which cannot be described in the ORD-Horn subclass. The satisfiability algorithm, which is common to all these algebras, was shown to be linear. The definition of the classes and the algorithm rely on the notion of *maximal acyclic* relations.

4.2. Techniques

The original constraint propagation algorithm Allen provides in (Allen83) enforces path-consistency. This algorithm hasn't changed much over the years, and today it is still used as the major constraint propagation algorithm (for Interval TCSPs). A more sophisticated algorithm, which enforces 4-consistency, can be found in (vanBeek90c). These algorithms are sound but incomplete for deciding consistency and approximate the minimal constraints.

4.2.1. Hierarchical IA TCSPs Reference intervals can be used to form clusters to reduce the space requirements and time complexity of enforcing path-consistency (Allen83). Clusters are formed by associating a set of intervals with one a reference interval that subsumes them. Efficiency of constraint propagation is improved by enhancing path-consistency as follows: Constraint propagation takes place within each cluster separately. Inter-cluster constraints, between a pair of variables X_i, X_j from different clusters, are computed by processing triangles in which X_k specifies a reference interval only. If the reference intervals are disjoint, then enforcing path-consistency within the clusters is sufficient to enforce path-consistency for the whole TCSP.

To improve efficiency of enforcing path-consistency on general TCSPs where there are no reference intervals (or they are not disjoint), reference intervals can be generated on-the-fly (Koomen87). This reduces the number of triangles processed yet, if done correctly, computes a path-consistent TCSP.

4.2.2. Empirical Evaluation Next, we survey results of three experiments reported in (LadkinReinefeld92, SchwalbDechter97), aimed at evaluating the effectiveness of path-consistency for: (i) removing disjunctions, (ii) detecting inconsistencies, and (iii) pruning dead-ends in backtrack search.

The ability of path-consistency in **removing redundant disjunctions** was evaluated on randomly generated problems by Ladkin and Reinefeld (LadkinReinefeld92). Initially, the average number of disjunctions generated was 7.5 (i.e. 50% of 13). For consistent TCSPs, after enforcing path-consistency the average number of disjunctions did not drop under 5.5. For inconsistent TCSPs, the average number of disjuncts after the first iteration of PC did not go above 4.5. Most inconsistencies were found in the first 3 iterations.

As a measure of effectiveness of path-consistency in **detecting inconsistencies**, it was suggested to use the fraction of problems for which path-consistency can correctly decide consistency (SchwalbDechter97). Since enforcing path-consistency is sound, the only type of incorrect answers are those cases where path-consistency did not detect inconsistency of a consistent TCSP. For most of the problems path-consistency was accurate. However, for problems where about 8 relations out of 13 were allowed, path-consistency was useless. This is one of the properties of the *transition region* (CheesmanKanefsky91, Mitchelletal92, WilliamsHogg93).

The effectiveness for **pruning dead-ends in backtrack search** has been evaluated on the algorithm in figure . For most problems, path-consistency was very effective (LadkinReinefeld92). However, when about 8 relations out of 13 were allowed, the problems encountered were the most difficult and path-consistency was not as effective. As a result, there is an exponential increase in the number of dead-ends (SchwalbDechter97). This is one of the properties of the *transition region* (CheesmanKanefsky91, Mitchelletal92, WilliamsHogg93).

4.3. Summary

For qualitative Interval TCSPs, also called the Interval Algebra (IA), answering queries is intractable. Nevertheless, many relation-based tractable classes exist and the unique maximal tractable class using *all* 13 relations was identified. The most common technique used for deciding consistency and computing feasible relations of the IA is enforcing path-consistency. For all the tractable classes surveyed, it correctly decides consistency. To compute a solution, backtrack search is used. Incorporating path-consistency as a forward checking procedure within backtrack search was shown to be very effective in pruning dead-ends.

5. Metric Point Constraints

In metric point TCSP variables specify time points and **BTR** is the set of intervals of *time-structure*. Therefore, a metric constraint has the form $C_{ij} = \{[a_1, b_1], \ldots, [a_k, b_k]\}$ where the intervals are pairwise disjoint, which is interpreted as

$$(a_1 \leq X_j - X_i \leq b_1) \wedge \ldots \wedge (a_k \leq X_j - X_i \leq b_k)$$

Accordingly, a unary constraint $C_i = \{[a_1, b_1], \ldots, [a_k, b_k]\}$ is interpreted as $(a_1 \leq X_i \leq b_1) \wedge \ldots \wedge (a_k \leq X_i \leq b_k)$.

Qualitative Point TCSPs can be described using Metric Point TCSPs by mapping the qualitative point-point constraints into metric constraints (Ligozat91, KautzLadkin91, Meiri96). Similarly, metric TCSPs can be translated, with loss of information, into Qualitative TCSPs (KautzLadkin91).

Given two metric constraints T and S, the basic **Operators** are defined as follows:

1. The *inverse* of $T = \{[a_1, b_1], \ldots, [a_k, b_k]\}$ is $\breve{T} = \{[-b_k, -a_k], \ldots, [-b_1, -a_1]\}$.
2. The *intersection* of T and S, denoted by $T \cap S$, admits only values that are allowed by both of them.
3. The *composition* of T and S, denoted by $T \circ S$, admits only values r for which there exists $t \in T$ and $s \in S$ such that $r = t + s$

A **solution** is a consistent singleton labelling. A singleton labelling of a Metric TCSP is a selection of a single interval from each constraint. Consistency of a labelling can be decided by enforcing path-consistency in $O(v^3)$ where v is the number of variables. Note that when a constraint C_{ij} is not specified in the input, it is assumed to specify the single interval $[-\infty, \infty]$.

THEOREM 3 *(Dechteretal91) Deciding consistency (and computing a solution) of a Metric Point TCSP is in NP-complete.*

5.1. Tractable Classes

There are three known relation based tractable classes: Simple Temporal Problems (STP), STP with inequation constraints (for continuous domains only) and Star TCSPs. There is also a graph-based tractable class called series-parallel TCSPs.

5.1.1. Simple Temporal Problems (STP) *Simple Temporal Problems* (STP) specify a single interval per constraint. An STP can be associated with a directed edge-weighted graph, G_d, called a *distance graph* (d-graph), having the same vertices as the constraint graph G; each edge $i \to j$ is labelled by a weight w_{ij} representing the constraint $X_j - X_i \leq w_{ij}$. An STP is consistent iff the corresponding d-graph G_d has no negative cycles and the minimal network of the STP corresponds to the *minimal distances* in G_d. Therefore, Floyd-Warshall's all-pairs shortest-path algorithm enforces path-consistency and is complete for STPs (Dechteretal91).

5.1.2. Single Intervals with Inequation constraints The class of Simple Temporal Networks was further extended to include *disjunctions of inequations* (i.e. $x \neq y$). This extension is tractable if the domains are dense (i.e. rationals or reals) (Koubarakis92). This class of constraints may be encountered when resolution is combined with variable elimination.

EXAMPLE: (Koubarakis92) Consider the following set of constraints: $X_3 \leq X_1$, $X_5 < X_1$, $X_1 \leq X_2$ and $X_4 \neq X_1$. Eliminating X_1 results in $X_3 \leq X_2$, $X_5 < X_2$ with the addition of disjunction $X_4 \neq X_3 \ \vee \ X_4 \neq X_2$. □

In this case **deciding consistency** can be done in $O(v^3 e)$(Koubarakis92) and **minimal constraints** can be computed in $O(v^5)$ by enforcing 5-consistency (Koubarakis95).

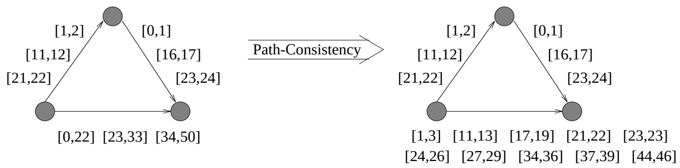

Figure 7. The fragmentation problem.

5.1.3. Star metric TCSPs A metric TCSPs is a Star if its binary constraints C_{ij} specify single intervals and their unary constraints C_{0i} specify an arbitrary number of intervals. Deciding their consistency requires $O(v^3ek+e^2k^2)$ steps where v is the number of variables, e is the number of constraints and k is the maximum number of intervals per unary constraint (SchwalbDechter97).

This class of problems is commonly encountered when tasks are to be scheduled within a set of available time windows. For example, to represent the introductory treatment plan scheduling problem, we could use a disjunctive unary constraint to specify the times the equipment and the therapist are available.

5.1.4. Series Parallel A TCSP is said to be *series-parallel* with respect to a pair of nodes, i and j, if it can be reduced to the edge (i,j) by repeated applications of the following *reduction* operation: select a node of degree 2 or less, remove it from the network, and connect its neighbours. Deciding whether a TCSP is series-parallel requires $O(v)$ steps where v is the number of variables. If the TCSP is series-parallel, deciding consistency can be done using the **directed path-consistency** algorithm (Dechteretal91) in $O(nk)$ where k is the maximal number of intervals per constraint.

5.2. Techniques

The path-consistency algorithm for metric constraints was introduced as a parallel to the path-consistency algorithms used to process CSPs and Qualitative TCSPs.

5.2.1. Complexity of Path-Consistency When time is described by integer or rational numbers, then algorithm PC terminates in $O(v^3R^3)$ and $O(v^3R^2)$ steps respectively (Dechteretal91). However, when the range R is very large or the domains are continuous enforcing path-consistency is problematic and becomes impractical (exponential) (PoesioBrachman91, SchwalbDechter97). Consider the network presented in figure 5.2.1, having 3 variables, 3 constraints and 3 intervals per constraint. After enforcing path-consistency, two constraints remain unchanged in the path-consistent network while the third is broken into 10 subintervals. As this behaviour is repeated over numerous triangles in the network, the number of intervals may become exponential.

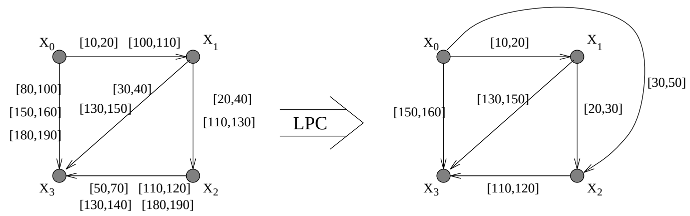

Figure 8. The power of removing disjunctions.

5.2.2. Coping with Fragmentation Since enforcing path-consistency is exponential, it was suggested to approximate path-consistency with two polynomial time algorithms called Upper Lower Tightening (ULT) and Loose Path-Consistency (LPC) (SchwalbDechter93). **ULT** Algorithm ULT terminates in $O(v^3ek + e^2k^2)$ steps

where k is the maximum number of intervals per constraint. After it terminates, if C_{ij} specifies a single interval $(i \neq 0, j \neq 0)$ then the assignment of X_i to the lower bound of C_{0i} is a solution.

LPC Algorithm LPC terminates in $O(n^3k^3e)$ steps but is stronger than ULT, namely it computes tighter constraints and is capable of detecting inconsistencies that ULT cannot detect.

EXAMPLE: To illustrate the effectiveness of LPC in removing disjunctions, consider the sample TCSP in figure 5.2.2. Applying ULT on this TCSP does not have any effect. After processing with LPC, a total of 8 redundant disjunctions were removed. Consequently, the search space was reduced from 96 possible labelling to 1. □

5.2.3. Empirical Evaluation of Techniques The experiments are surveyed here are aimed at (SchwalbDechter97): (i) evaluating the efficiency and effectiveness tradeoff between enforcing path-consistency and applying ULT, and (ii) the ability of ULT and LPC to remove disjunctions and effectively prune dead-ends.

Comparing Path-Consistency vs. ULT It was reported that path-consistency is exponential in the fragmentation and thus may be impractical even for small problems of 10 variables. Despite the fact that ULT is orders of magnitude more efficient than PC, it is able to detect inconsistency in about 70% of the cases that path-consistency does (SchwalbDechter97).

Backtracking The efficiency of backtracking search was improved by a factor of 10^6 for tiny problems of 12 variables and 66 constraints. This was achieved by using the polynomial time approximation algorithms in three ways (SchwalbDechter97): (i) as a preprocessing phase before initiating search, to reduce the fragmentation, (ii) to perform forward checking procedure (within backtracking) for early detection of dead-ends, and (iii) as an advice generator for dynamic ordering of the constraints to be labelled. A phase transition (CheesmanKanefsky91, Mitchelletal92) was also observed (SchwalbDechter97).

5.3. Summary

Metric TCSPs provide a framework for describing *disjunctive* linear difference constraints. In general, answering queries is intractable. Four tractable classes were surveyed: Simple Temporal Problems (STP), STP with disjunctions of inequation constraints, the Star and series-parallel TCSP.

Enforcing path-consistency is exponential (in contrast to other TCSPs) due to the fragmentation problem. There are two algorithms for controlling fragmentation and removing redundant disjunctions: Upper lower Tightening (ULT) and Loose Path-Consistency (LPC). When incorporated within backtracking search, these algorithms can improve the search performance by orders of magnitude.

6. Combining Temporal Constraints

The qualitative and metric point and interval TCSPs were combined into a unified model proposed by Meiri (Meiri96) which follows the general definition given in section 2. In this section we summarise some results on various specific combinations.

6.1. Interval-Point Qualitative Constraints

In the interval-point algebra, abbreviated **IPA** variables represent either time points or time intervals. A new class of constraints between a point and an interval and vice-versa is defined based on a **BTR** composed of the relations `Before`, `Starts`, `During`, `Finishes`, `After`, and their inverses (Vilain82, Ligozat91, Meiri96). Thus **IPA** is has 2^5 relations. Since interval-interval constraints are not included, it is strictly less expressive than **IA**, however the resulting problem is still intractable.

THEOREM 4 *(Meiri96) Deciding consistency of* **IPA** *TCSP (which excludes* **IA** *constraints) is NP-complete.*

IPA relations are a subset of the more general class of relations called (p, q)-relations defined between a pair of linear ordered sets having p and q elements (Ligozat91).

6.2. Point Algebra + Metric Domain Constraints

TCSPs resulting from augmenting Point TCSPs with unary metric constraints are also intractable. The subclasses investigated were either $\mathbf{PA}_c$ or **PA** augmented with one of the following unary metric constraints (Meiri96):

	Discrete		Single interval		Multiple interval	
Deciding consistency						
PA$_c$	AC	$O(ek)$	AC+PC	$O(n^2)$	AC+PC	$O(n^2k)$
PA	NP-Complete		AC+PC	$O(en)$	NP-Complete	
Computing minimal constraints						
PA$_c$	AC+PC	$O(n^2k)$	AC+PC	$O(n^2)$	AC+PC	$O(n^2k)$
PA			AC+PC	$O(en^2)$		

Figure 9. Results on Combined TCSPs.

- **Discrete**: Specified by a finite set of values.
- **Single interval**: Specified by a single metric interval.
- **Multiple intervals**: Specified by a set of disjoint metric intervals.

Complexity results and techniques for these classes are summarised in figure 6.2 where AC denotes *Arc-Consistency*, PC denotes *Path-Consistency* and k is the maximum number of intervals defining the domain (Meiri96).

6.3. Interval Algebra + Metric Constraints

TCSPs resulting from augmenting qualitative interval TCSPs with metric point constraints are intractable. Apart from backtracking search, there are two methods for deciding consistency and approximating the minimal constraints:

1. Enforcing path-consistency on a combined TCSP in $O(n^3R^3)$ steps where R is the range of the metric constraints (see Section 5) (Meiri96).

2. Iteratively enforcing path-consistency on the point metric and qualitative interval sub-TCSPs independently, and translating and propagating information between them in $O(n^5R^3)$ (MATS system (KautzLadkin91)).

The advantage of combining the three kinds of TCSPs into a unified framework is two fold: (i) it provides a simple uniform knowledge representation model, and (ii) PC algorithm can be directly used to enforce path-consistency of the combined TCSP (Meiri96).

To obtain the unified framework, a new qualitative interval-point constraint was introduced and the composition operator $\circ$ was extended to accommodate the new constraint (Meiri96). The IA transitivity tables were augmented with tables composing interval-point constraints with point-point and interval-interval constraints.

6.4. Summary

The different kinds of TCSPs surveyed in this paper were combined into a single unified framework. As a result, the significant body of knowledge accumulated for each of these classes separately is applicable to the unified framework.

7. Concluding Remarks

There has been an increasing interest on temporal constraint processing in the AI community since Allen's seminal work in 1983 (Allen83). The intensive research on the subject, specially during the last decade, has produced a significant body of knowledge that influenced research on other sorts of constraint-based problems, such as spatial reasoning, as well as on related reasoning tasks such as scheduling and planning. TCSP results are relevant to a variety of applications in computer science including temporal databases, medical informatics, computer-aided design, computer-aided manufacturing. Our aim in this paper has been providing an organised inventory of these results that makes them more accessible.

Certainly, TCSP research benefited from the CSP background, however it is worth realizing the singularities of temporal CSPs that motivated the formation of this dynamic sub-area. Despite the intensive analysis devoted to this class of problems during last years, it is somewhat surprising to see how new open issues keep arising. The relatively recent, conclusive results on characterising tractable classes for the interval algebra, started by Nebel and Bückert's work (NebelBuerckert95), has been a landmark that determined current research directions. On the one hand, it led to studies related to the Interval algebra such as alternative proofs (Ligozat96), an exhaustive characterisation of those tractable classes (DrakengrenJonsson96, DrakengrenJonsson97) and techniques for related problems such as finding a solution (GereviniCristani97) and finding the minimal network (Bessiereetal96). On the other hand attention is being paid to more sophisticated temporal constraint problems: combinations of interval constraints with other sorts (Jonssonetal96), duration constraints (NavarreteMarin97), fuzzy constraints (VilaGodo94), periodic constraints (Morrisetal95), disjunctive constraints (Koubarakis96). We will probably see more on these classes of temporal constraints as well as on their application in task-oriented reasoners in the near future.

Notes

1. A slight mistake in the proof is corrected in (GereviniSchubert94).

References

James Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26:832–843, 1983.

K.R. Baker, P.C. Fishburn, and F.S. Roberts. Partial orders of dimension 2. *Networks*, 2:11–28, 1972.

A. Belfer and M. Golumbic. A combinatorical approach to temporal reasoning. In *Proc. Information Technology IEEE*, pages 774–780, 1990.

A. Belfer and M.C. Golumbic. The role of combinatorical structures in temporal reasoning. Technical report, IBM Research report, 1981.

Christian Bessiere, Amar Isli, and Gerard Ligozat. Global consistency in interval algebra networks: Tractable subclasses. In *Proc. ECAI'96*, pages 3–7, 1996.

K.S. Booth and G.S. Leuker. Testing for the consecutive ones property, interval graphs, and planarity using pq-tree algorithms. *J. Comput. Sys. Sci.*, 13:335–379, 1976.

A. Bouchet. Reducing prime graphs and recognizing circle graphs. *Combinatorics*, 7:243–254, 1987.

P. Cheesman and B. Kanefsky. Where the really hard problems are. In *Proc. IJCAI'91*, pages 163–169, 1991.

Thomas Cormen, Charles Leiserson, and Ronald Rivest. *Introduction to Algorithms*. McGraw Hill, 1990.

Rina Dechter. *Encyclopedia of Artificial Intelligence*, chapter Constraint Networks. John Wiley&Sons Inc., 2nd edition edition, 1992.

Rina Dechter, Itay Meiri, and Judea Pearl. Temporal constraint networks. *Artificial Intelligence*, 49:61–95, 1991.

Thomas Drakengren and Peter Jonsson. Maximal tractable subclassse of Allen's interval algebra: Preliminary report. In *Proc. AAAI'96*, pages 389–394, 1996.

Thomas Drakengren and Peter Jonsson. Towards a comploete classification of tractability in Allen's algebra. In *Proc. IJCAI'97*, pages 389–394, 1997.

B. Dushnik and E.W. Miller. Partially ordered sets. *Amer. J. Math*, 63:600–610, 1941.

Christian Freksa. Temporal reasoning based on semi-intervals. *Artificial Intelligence*, 54:199–227, 1992.

D.R. Fulkerson and O.A. Gross. Incidence matrices and interval graphs. *Pacific J. Math.*, 15:835–855, 1965.

C.P. Gabor, K.J. Supowit, and W.L. Hsu. Recognizing circle graphs in polynomial time. *J. ACM*, 36:435–473, 1989.

Alfonso Gerevini and Matteo Cristani. On finding a solution in temporal constraint satsifaction problems. In *Proc. IJCAI'97*, pages 1460–1465, 1997.

Alfonso Gerevini and Lenhart Schubert. Efficient temporal reasoning through timegraphs. In *Proc. IJCAI'93*, pages 648–654, 1993.

Alfonso Gerevini and Lenhart Schubert. On computing the minimal labels in time point algebra networks. Technical Report 9408-10, Insituto per la Ricerca Scientifica e Tecnologica, 1994.

Alfonso Gerevini and Lenhart Schubert. Efficient algorithms for qualitative reasoning about time. *Artificial Intelligence*, 74(3):207–248, 1995.

Malik Ghallab and Amine Mounir Alaoui. Managing efficiently temporal relations through indexed spanning trees. In *Proc. IJCAI'89*, pages 1297–1303, 1989.

P.C. Gilmore and A.J. Hoffman. A characterization of comparability graphs and interval graphs. *Canadian J. Math*, 16:539–548, 1964.

M. Golumbic and E. Scheinerman. Containment graphs, posets and related classes of graphs. *Ann. N.Y. Acad. Sci*, 555:192–204, 1989.

Martin Golumbic and Ron Shamir. Algorithms and complexity for reasoning about time. Technical Report 22-91, Rutgers, NewJersey, 1992.

Peter Jonsson, Thomas Drakengren, and Christer Backstrom. Tractable subsclasses of the poinot-interval algebra: A complete classification. In *Proc. KR'96*, pages 352–363. AAAI Press/The MIT Press, 1996.

Henry Kautz and Peter Ladkin. Integrating metric and qualitative temporal reasoning. In *Proc. AAAI'91*, pages 241–246, 1991.

Johannes Koomen. The TIMELOGIC temporal reasoning system. Technical Report 231, Univ. of Rochester, Computer Science Dept., November 1987. (revised March 1989).

N. Korte and R.H. Möhring. An incremental linear time algorithm for recognizing interval graphs. *SIAM J. Comput.*, 18:68–81, 1989.

M. Koubarakis. Dense time and temporal constraints with $\neq$. In *Proc. KR'92*, pages 24–35, 1992.

M. Koubarakis. From local to global consistency in temporal constraint networks. In *Proc. CP'95*, 1995.

Manolis Koubarakis. Tractable disjunctions of linear constraints. In *Proc. CP'96*, pages 297–307, 1996.

P. Ladkin and R.D. Maddux. The algebra of constraint satisfaction problems and temporal reasoning. Technical report, Krestel Institute, 1988.

Peter Ladkin and A. Reinefeld. Effective solution of qualitative interval constraint problems. *Artificial Intelligence*, 57:105–124, 1992.

Gérard Ligozat. On generalized interval calculi. In *Proc. AAAI'91*, 1991.

Gérard Ligozat. A new proof of tractability for ord-horn relations. In *Proc. AAAI'96*, 1996.

Itay Meiri. Combining qualitative and quantitative constraints in temporal reasoning. *Artificial Intelligence*, 87:343–385, 1996.

David Mitchell, Bart Selman, and Hector Levesque. Hard and easy distributions of sat problems. In *Proc. AAAI'92*, 1992.

Robert Morris, Lina Khatib, and Gerard Ligozat. Generating scenarios from specifications of repeating events. In *Proc. TIME'95*, 1995.

Isabel Navarrete and Roque Marin. Qualitative temporal reasoning with points and durations. In *Proc. IJCAI'97*, pages 1454–1459, 1997.

B. Nebel and H. J. Bürckert. Reasoning about temporal relations: A maximal tractable subclass of Allen's interval algebra. *Journal of the Association for Computing Machinery (ACM)*, 42(1):43–66, 1995.

K. Nökel. Convex relations between time intervals. In J. Retti and K. Leidlmair, editors, *5 Osterreichische Artificial-Intelligence-Tagung*, pages 298–302, 1989.

P.Fishburn. Intransitive indifference with unequal indifference intervals. *J Math Psych.*, 7:144–149, 1970.

M. Poesio and R.J. Brachman. Metric constraints for maintaining appointments: Dates and repeated activities. In *Proc. AAAI'91*, pages 253–255, 1991.

Eddie Schwalb and Rina Dechter. Coping with disjunctions in temporal constraint satisfaction problems. In *Proc. AAAI'93*, pages 127–132, 1993.

Eddie Schwalb and Rina Dechter. Processing temporal constraint networks. *Artificial Intelligence*, in press, 1997. Also available as UCI technical report (1995).

R.E. Tarjan. Depth-first search and linear graph algorithms. *SIAM J. Comput.*, 1:146–160, 1973.

Peter van Beek. Approximation algorithms for temporal reasoning. In *Proc. IJCAI'89*, pages 1291–1296, 1989.

Peter van Beek. *Exact and Approximate Reasoning about Qualitative Temporal Relations*. PhD thesis, Dept. of Computer Science, University of Alberta, August 1990.

Peter van Beek. Reasoning about qualitative temporal information. *Artificial Intelligence*, 58:297–326, 1992.

Peter van Beek and Robin Cohen. Exact and approximate reasoning about temporal relations. *Computational Intelligence*, 6:132–144, 1990.

Lluís Vila and Lluís Godo. Query-answering in fuzzy temporal metric constraints. Report de Recerca 94/13, IIIA, 1994.

Marc Vilain. A system for reasoning about time. In *Proc. AAAI'82*, pages 197–201, 1982.

Marc Vilain, Henry Kautz, and Peter van Beek. Constraint propagation algorithms for temporal reasoning: A revised report. In Daniel S. Weld and Johan de Kleer, editors, *Readings on Qualitative Reasoning about Physical Systems*, pages 373–381. Morgan Kauffman, 1989.

C.P. Williams and T. Hogg. A typicality of phase transition search. *Computational Intelligence*, 9(3):211–238, 1993.