Managing Temporal Constraints with Preferences: Representation, Reasoning, and Querying

Paolo Terenziani, Andolina Antonella, Luca Piovesan

Abstract— Representing and managing temporal knowledge, in the form of temporal constraints, is a crucial task in many areas, including knowledge representation, planning, and scheduling. The current literature in the area is moving from the treatment of "crisp" temporal constraints to fuzzy or probabilistic constraints, to account for preferences and\or uncertainty. Given a set of temporal constraints, the evaluation of the *tightest* implied constraints is a fundamental task, which is essential also to provide *reliable query-answering facilities*. However, while such tasks have been widely addressed for "crisp" temporal constraints, they have not attracted enough attention in the "non-crisp" context yet. We overcome such a limitation, by (i) extending quantitative temporal constraints to cope with *preferences*, (ii) defining a *temporal reasoning algorithm* which evaluates the *tightest* temporal constraints, and (iii) providing suitable *query-answering facilities* based on it.

Keywords — I.2 Artificial Intelligence, I.2.4 Knowledge Representation Formalisms and Methods, I.2.4.d Knowledge base management, I.2.8.b Constraint satisfaction

1 INTRODUCTION

Representing and managing temporal knowledge is an essential task in many areas, including planning, scheduling, human–machine interaction, natural language understanding, diagnosis, and robotics. Different approaches have been developed, ranging from *general-purpose* approaches, like Hidden Markov Models, Bayesian Networks, or logical approaches, to more *specific approaches* focusing on constraint satisfaction problems, and *on temporal constraints* (see, e.g., the surveys in [1]–[3]). The latter approaches (*which are the focus of this paper*) can be distinguished on the basis of whether they focus on the *qualitative* or *quantitative temporal constraints*. Among the approaches of the first kind, Allen's Interval Algebra [4] and the Point Algebra [5] deal with a *qualitative* representation of temporal knowledge relative to intervals and points respectively (e.g., "A is ***before*** B"). Quantitative approaches, such as [6], [7], deal with metric temporal statements (e.g., "the starting time of B is ***between 10 and 20 minutes after*** the ending point of A"). Also, hybrid approaches have been proposed, integrating qualitative and metric information in a single model [8], [9]. Such approaches also propose ***temporal reasoning*** algorithms that propagate such constraints, to check their *consistency*, and\or to find a *scenario* (i.e., a solution: an instantiation of all events such that all constraints are satisfied), or to make explicit the *tightest* implied constraints.

- *Paolo Terenziani is with DISIT – Università del Piemonte Orientale "Amedeo Avogadro", Viale Teresa Michel 11, 15121 Alessandria, Italy, Email: paolo.terenziani@uniupo.it;*
- *Luca Piovesan is with DISIT – Università del Piemonte Orientale Amedeo Avogadro", Viale Teresa Michel 11, 15121 Alessandria, Italy, Email: luca.piovesan@uniupo.it;*
- *Antonella Andolina is with ITCS Sommeiller, C.so Duca degli Abruzzi, 20 - 10129 Torino, Italy, Email: antoando@libero.it.*

Example 1. To simplify the discussion, we consider the Simple Temporal Problem (STP) quantitative temporal constraints [6]. Let t_1, t_2 and t_3 be time points, and let KB be the following set of constraints, representing the minimum and maximum distances between pairs of points: KB={t_1[8,10]t_2, t_2[10,11]t_3, t_1[15,20]t_3}. KB is consistent, and {t_1=0, t_2=8, t_3=18} is a scenario (solution) of KB. The *tightest* constraints implied by KB (the so called *minimal network* [6]) are KB'={t_1[8,10]t_2, t_2[10,11]t_3, t_1[18,20]t_3} (in particular, the minimum distance between t_1 and t_3 must be 18). ■

While in several task (e.g., in scheduling) the goal is to find a *scenario*, in others, such as decision support, the *tightest* constraints must be determined, to provide users with a compact representation of *all* the possible solutions they can adopt (since the choice of a specific solution has to be left to the users). This is the case, e.g., when supporting physicians in the execution of clinical guidelines, such as the one in Figure 1. In such contexts, also ***query answering*** is important, to give users a way to explore the space of solutions.

Example 2. Given KB, the user might ask:
(Q1) *May I execute t_3 16 after t_1?*
(Q2) *If I have performed t_1 at 20, when (in which range of time) should I perform t_2 and t_3?* ■

The literature shows that temporal reasoning and query answering are closely related tasks: *correct query answering* can be provided only if temporal reasoning evaluates the *tightest* temporal constraints. Indeed, computing the *tightest* constraints is a fundamental task, to which a lot of efforts have been devoted in the literature [1]–[3], [5], [6].

Example 3. Given KB, suppose that the reasoning process is not complete, so that not all the tightest constraints

are obtained. E.g., suppose that KB" ={t_1[8,10]t_2, t_2[10,11]t_3, t_1[16,20]t_3} is obtained. Given KB", the answer to a user's query Q1 is YES, which is a *wrong* answer. Thus, only the availability of the *tightest* constraints can grant the *correctness* (and, thus, the *reliability*) of the answers to the user. ■

In all the approaches to temporal constraints mentioned so far, temporal constraints are "*crisp*", in the sense that they represent a set of "*equally possible*" temporal relations\constraints between two time units. All of these proposals rely on the framework of Constraint Satisfaction Problem (CSP), in that they face the relevant reasoning tasks by representing the temporal objects as variables with temporal domains, and the available temporal knowledge as a set of constraints between these variables. Unfortunately, these temporal constraint-based reasoning approaches inherit from CSP a number of fundamental limitations, mainly related to a lack of flexibility and a limited representation of uncertainty [10]. Specifically, many problems lead to a large set of possible solutions, but often there may be ***preferences*** among them, while standard CSP approaches would consider each of them as "equally possible".

A paradigmatic example are temporal constraints in medical treatments. In the Clinical Guideline in Figure 1 (which is a simplified version of part of the guideline provided by the British National Institute for Health and Care Excellence – NICE), the execution times t1, t2, t3, t4 of the clinical actions are constrained as described in Example 4.

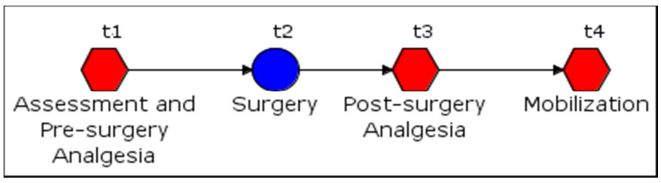

Figure 1 Simplified Clinical Guideline for the treatment of hip fracture.

Example 4. t2 must be executed within 12 (with preference 2), 24 (with preference 4), 36 (with preference 2) or 48 (with preference 1) hours after t1; t3 must be executed within 12 (with preference 4) and 24 (with preference 1) hours after t2; t4 must be executed within 24, 48 or 72 hours (with preferences 4, 2, 1 respectively).

To deal with such issues, a huge stream of research has extended the CSP formalism in a fuzzy direction, by replacing classical ***"crisp"*** constraints with soft ***"not-crisp"*** constraints modeled by fuzzy relations. Restricting the attention to CSP approaches only, a number of temporal reasoning approaches based on the Fuzzy Constraint Satisfaction Problem [10] has been devised. Concerning *qualitative* constraints, Ryabov et al. [11] attach a *probability* to each of Allen's basic interval relations. Using the operations of inversion, composition, and addition, defined for this probabilistic representation, they present a path consistency algorithm for temporal reasoning. A similar probabilistic approach has been proposed more recently by Mouhoub and Liu [12], as an adaptation of the general probabilistic CSP framework. On the same line of research, Badaloni and Giacomin [13] extend Allen's interval-based framework to associate a *preference degree* to relations between intervals. Other approaches based on the Fuzzy Constraint Satisfaction Problem have been devised, in order to handle *quantitative* temporal information in terms of points (or times of events) and in terms of fuzzy metric constraints between them. Barro et al. [14] introduce a model for the representation and handling of fuzzy temporal references. Khatib et al. [15] extend constraint-based temporal reasoning (and, in particular, the STP and the TCSP framework [6]) to allow for reasoning about temporal *preferences*, basing their approach on *C-semiring* properties. Mouhoub and Sukpan [16] have devised an hybrid framework managing both quantitative and qualitative temporal constraints with preferences, in which temporal intervals are considered, and temporal reasoning is used to find a *solution* (scenario) optimizing preferences.

Despite most of the above "non-crisp" approaches focus on *temporal reasoning*, in the form of propagation of temporal constraints, none of them grants that the output is the *tightest* set of constraints (given the input ones). This is a major limitation: as discussed in example 3 above, it implies that such approaches cannot grant the correctness when dealing with queries like the ones in Example 2. Indeed, quite surprisingly, such a type of query answering is not explicitly addressed by any approach to "non-crisp" temporal constraints in the literature.

In this work, we aim at providing a *non-crisp* extension to *quantitative* constraints, supporting the possibility of expressing alternative distances between time points, and of associating a *preference* to each alternative, to grant flexibility and expressiveness in the representation of quantitative temporal constraints. Second, and more important, we aim at supporting *temporal reasoning* and *query answering* facilities on KBs of such constraints. To achieve such a goal, and to grant for the absolute reliability of query answering, we propose a temporal reasoning algorithm which we prove that computes the *tightest* constraints (i.e., the *all-to-all shortest paths* between time points).

In Section 2, we introduce our representation of quantitative temporal constraints with preferences. In Section 3, which is the core of the paper, we describe our temporal reasoning algorithm, which is an instantiation of Cormen et al.'s *Compute-Summary* general algorithm [17] to compute the *tightest* constraints between each pair of time points. Section 4 briefly mention some query answering facilities provided by our approach. Finally, Section 5 contains conclusions and future work.

2 Quantitative Temporal Networks with Preferences

In this work, we extend *quantitative* temporal constraints to support the possibility of associating preferences to alternative constraints. As most approaches focusing on quantitative constraints (see [1]–[3]), we base our approach on the notion of *distance* between *time points*. A preference is associated to each distance.

Definition. Quantitative Temporal Label with Preferences (QTLP) Let

- let $t_i,t_j \in \mathbb{R}$ be time points
- let $p_1, \ldots, p_n \in \mathbb{R}$ be preferences
- let $d_1, \ldots, d_n \in \mathbb{Z}$ be distances (between points)

A Quantitative Temporal Label with Preferences (QTLP) is a list $<(d_1,p_1),\ldots,(d_n,p_n)>$, or the distinguished label $\perp$, denoting the non-existent constraint (in which no distance is possible, and the preference is zero). Henceforth $\mathbb{L}$ denotes the domain of quantitative temporal labels with preferences.

Definition. Quantitative Temporal Constraint with Preferences (QTCP) and Quantitative Temporal Network with Preferences (QTNP). A QTCP is a constraint of the form t_i C t_j where $C \in \mathbb{L}$, and $t_i,t_j \in \mathbb{R}$ are time points. A QTNP is an oriented graph G=<V,E> with a labelling function λ, where V is a set P={$t_1,\ldots,t_n$} of time points, $E \subseteq P \times P$, and $\lambda: E \rightarrow \mathbb{L}$. ■

The meaning of a QTCP constraint t_i $<(d_1,p_1),\ldots,(d_n,p_n)>$ t_j is that the *distance* t_j-t_i between t_j and t_i is d_1 with preference p_1, or … or d_n with preference p_n. A Quantitative Temporal Network with Preferences (QTNP) is a set of QTCPs.

The QTNP modelling Example 4 in the introduction can be graphically modelled as shown in Figure 2, where we use *12-hour units* as basic *granularity*. For simplicity, in the paper we use preferences which are natural numbers, but notice that our approach also supports real number preferences.

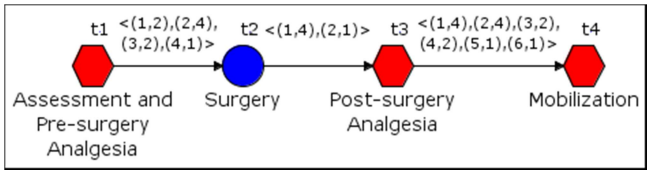

Figure 2. QTPN for Example 4: graphical representation.

3 An algorithm to evaluate the tightest constrains on TQNPs

3.1 General strategy

Temporal reasoning on a QTNP can be performed by an algorithm propagating the constraints to obtain the *tightest* equivalent QTNP, i.e., a QTNP which has exactly the same solutions of the original one, and in which (i) the minimum and maximum implied distances between each pair of points (as in the "minimal network" for the "crisp" case [6]) and (ii) the preferences for each distance are made explicit. Instead of inventing a new algorithm and then proving that it computes the *tightest* constraints, we adopt a different strategy. We exploit the general algorithm *Compute-Summaries*(λ,V,E,$\oplus$,$\odot$,**0**,**1**) in [17]. *Compute-Summaries* is indeed a highly parametric *all-to-all shortest path* algorithm to solve different problems concerning oriented paths in a graph. Such an algorithm takes in input a graph *(V,E)*, a labelling function λ:$E \rightarrow \mathbb{L}$ operating on the edges of the graph, an "*extension*" operator $\odot$, a "*resume*" operator $\oplus$, the *identity* for $\odot$ (indicated by **1**), and the *identity* for $\oplus$ (indicated by **0**). It is a dynamic algorithm, which, in case $\odot$,$\oplus$,**1**, and **0** are defined in such a way that ($\mathbb{L}$,$\oplus$,$\odot$,**0**,**1**) is a *C-semiring* (see Section 3.4), evaluates the *all-to-all shortest paths* (i.e., the *tightest* labels for each path) [17]. Thus, our idea is to define the *extension* $\odot^P$ and *resume* $\oplus^P$ operators (and their *identities*) for our problem in such a way that the structure ($\mathbb{L}$,$\oplus^P$,$\odot^P$,**0**,**1**) is a *C-semiring*. Then, we adopt a specific instance of *Compute-Summaries*, by instantiating its parameters with our operators. In such a way, we achieve an algorithm that computes the *shortest paths* (i.e., the *tightest* labels), as desired.

3.2 The Compute-Summaries algorithm

We adopt the notation in [17]: (i) 1,2,…,n indicate the vertices in V, where n=|V|; (ii) L_{ij} denotes the application of the resume operator $\oplus$ to all the paths from i to j in the graph (V,E) (i.e., $L_{ij} = \bigoplus_{i\text{-}p\rightarrow j}\lambda(p)$); (iii) L_{ij}^k denotes the application of the resume operator $\oplus$ to all the paths from i to j in the graph (V,E) traversing the nodes 1,…,k only (i.e., $L_{ij}^k = \bigoplus_{p\in Q}\lambda(p)$, where Q is the set of all paths in (V,E) connecting i to j and traversing only nodes in {1,…,k}).

Compute-Summaries(λ,V,E,$\oplus$,$\odot$,0,1) algorithm

```
1. n ← |V|
2. for i←1 to n
3.  do for j←1 to n
4.          do if j=i then L_ij^0 ← 1 ⊕ λ(i,j)
5.              else L_ij^0 ← λ(i,j)
6. for k←1 to n
7.   do for i←1 to n
8.          do for j←1 to n do
9.             L_ij^k ← L_ij^(k-1) ⊕ (L_ik^(k-1) ⊙ L_kj^(k-1))
```

Figure 3. Compute-Summaries algorithm

Complexity. The complexity of Compute-Summaries is $\Theta(T_{\oplus},T_{\odot})^3$, where denote the time required to evaluate $T_{\oplus}$, and $T_{\odot}$ respectively [17].

3.3 Definition of the extension and resume operators

Now, we define our resume and extension operators. For the sake of simplicity, we adopt the following notation.

Notation. Given two QTLPs *c1* and *c2*, we indicate with $p1^d$ and $p2^d$ the preference of the distance *d* in the first and in the second QTLP respectively.

In our approach, the operator *resume* $\oplus^P$ is used to "merge" two constraints $<(d1_1,p1_1),\ldots,(d1_n,p1_n)>$ and $<(d2_1,p2_1),\ldots,(d2_m,p2_m)>$ *concerning the same pair of time points*. The set union between the two input sets of distances is computed, and, for each distance in the union, its preference is:

(i) its input preference, if it only appears in one of the two constraints, or

(ii) the sum of its preferences in the first and in the second constraint, if it appears in both constraints.

Definition. Resume($\oplus^P$). Given two PTQLs $<(d1_1,p1_1),\ldots,(d1_n,p1_n)>$ and $<(d2_1,p2_1),\ldots,(d2_m,p2_m)>$ their resume is defined as follows:

- let D1={$d1_1,\ldots,d1_n$}, D2={$d2_1,\ldots,d2_m$}, and D'= D1 ∪ D2={$d'_1,\ldots,d'_k$},
- let $p^{d'}$ be defined as follows:
  (i) $p^{d'} = (p1^{d1_u}+p2^{d2_v})$ if $\exists d1_u \in$ D1, $\exists d2_v \in$ D2 such that $d'=d1_u=d2_v$,

(ii) $p^{d'} = p1^{d1_u}$ if $\exists d1_u \in D1$ such that $d' = d1_u \wedge \neg\exists d2_v \in D2$ such that $d' = d2_v$

(iii) $p^{d'} = p1^{d2_v}$ if $\exists d2_v \in D2$ such that $d' = d2_v \wedge \neg\exists d1_u \in D1$ such that $d' = d1_u$

$<(d1_1,p1_1),\ldots,(d1_n,p1_n)> \oplus^P <(d2_1,p2_1),\ldots,(d2_m,p2_m)> = <(d'_1, p^{d'_1}),\ldots, (d'_k, p^{d'_k})>$ ■

Example 5. As an example, let us consider the resume of the constraint between t_1 and t_2 of Example 4 and the new constraint $t_1<(2,2),(3,1),(4,1),(5,1)>t_2$. $<(1,2),(2,4),(3,2),(4,1)> \oplus^P <(2,2),(3,1),(4,1),(5,1)> = <(1,2),(2,6),(3,3),(4,2),(5,1)>$ ■

The extension operator $\odot^P$ is used to infer the constraint between two time points t_i and t_j, given the constraint between t_i and t_k and the constraint between t_k and t_j. The output distances are evaluated as the pairwise sum of the input distances. For each output distance *d'*, its preference $p^{d'}$ is obtained by summing up the product $p1^{d1_u} * p2^{d2_v}$ of the preferences of each pair of distances $d1_u$ (from the first input constraint) $d2_v$ (from the second constraint) such that $d1_u + d2_v = d'$. More formally:

Definition. Extension ($\odot^P$). Given two QTLPs $<(d1_1,p1_1),\ldots,(d1_n,p1_n)>$ and $<(d2_1,p2_1),\ldots,(d2_m,p2_m)>$, their extension is defined as follows:

- let $\{d'_1,\ldots,d'_r\} = \{d' \setminus d' = d1_s + d2_t \wedge d1_s \in \{d1_1,\ldots,d1_n\} \wedge d2_t \in \{d2_1,\ldots,d2_m\}\}$, and
- let $p^{d'} = \Sigma_{d1_u + d2_v = d'} (p1^{d1_u} * p2^{d2_v})$, then

$<(d1_1,p1_1),\ldots,(d1_n,p1_n)> \odot^P <(d2_1,p2_1),\ldots,(d2_m,p2_m)> =$

$<(d'_1, p^{d'_1}), \ldots, (d'_r, p^{d'_r})>$ ■

Example 6. As an example, let us consider the composition of the constraints between t_1 and t_2 and between t_2 and t_3 in Example 4: $<(1,2),(2,4),(3,2),(4,1)> \odot^P <(1,4), (2,1)> = <(2,8), (3,18), (4,12), (5,6), (6,1)>$ ■

Definition. Identities for $\odot^P$ and $\oplus^P$. The identity ***1*** for $\odot^P$ is $<(0,1)>$ since, given any $c \in \mathbb{L}$, $c \odot^P <(0,1)> = <(0,1)> \odot^P c = c$. The identity ***0*** for $\oplus^P$ is $\perp$ since, given any $c \in \mathbb{L}$, $c \oplus^P \perp = \perp \oplus^P c = c$. By definition, $\perp$ is also the annihilator for $\odot^P$: since it represent the non-existing constraint, for all $c \in \mathbb{L}$, $c \odot^P \perp = \perp \odot^P c = \perp$.

3.4 C-semiring properties

$(\mathbb{L}, \oplus^P, \odot^P, \perp, <(0,1)>)$ is a *C-semiring*.

Definition. Monoid. A monoid is a triple $(A, \otimes, \mathbf{0})$ where

1) $\otimes$ is a ***closed*** binary operator on the set A: for each $a,b \in A$, $a \otimes b \in A$
2) $\otimes$ is ***associative***: for each $a,b,c \in A$, $a \otimes (b \otimes c) = (a \otimes b) \otimes c$.
3) **0** is an ***identity*** element for $\otimes$: for each $a \in A$, $a \otimes \mathbf{0} = \mathbf{0} \otimes a = a$. ■

A monoid is ***commutative*** if $\otimes$is commutative: for each $a,b \in A$, $a \otimes b = b \otimes a$.

Definition. C-semiring. A C-semiring is a 5-tuple $R = (A, \oplus, \otimes, 0, 1)$ such that

1) $(A, \oplus, 0)$ is a ***commutative monoid***.
2) $(A, \otimes, 1)$ is a ***monoid***.
3) $\otimes$distributes over $\oplus$: for all $a,b,c \in A$, $(a \oplus b) \otimes c = (a \otimes c) \oplus (b \otimes c)$.
4) 0 is anannihilator for$\otimes$: for all $a \in A$, $0 \otimes a = a \otimes 0 = 0$■

Property. $(\mathbb{L}, \oplus^P, \odot^P, \perp, <(0,1)>)$ is a *C-semiring*.

Proof (sketch). By definition, both $\oplus^P$ and $\odot^P$ are ***closed*** over $\mathbb{L}$. $\oplus^P$ is ***associative***, since it performs the union (which is associative) of the distances and, as regards the preferences, it maintains the original preferences or sum up them (which is also an associative operation) in case a distance appears in both constraints. Such operations are also ***commutative***. By definition, $\perp$ is the identity for $\oplus^P$. Thus, $(\mathbb{L}, \oplus^P, \perp)$ is a ***commutative monoid***. $\odot^P$ is ***associative***, since it performs the pairwise sum (which is associative) of the distances and, as regards the preferences, is sum up their products (which are also associative operations). $<(0,1)>$ is the identity for $\odot^P$. Thus, $(\mathbb{L}, \odot^P, <(0,1)>)$ is a ***monoid***. $\odot^P$ ***distributes*** over $\oplus^P$: concerning the distances, this is due to the fact that pairwise sum distributes over union. Concerning the preferences, preferences in the union are the input ones, except in case a distance appears in both labels. In such cases, the preferences are summed. However, product (performed by the extension operator) distributes over sum. ■

If $(\mathbb{L}, \odot^P, <(0,1)>)$ is a *C-semiring*, Cormen et al.'s algorithm evaluates the *all-to-all shortest paths* on a graph, by evaluating new labels (λ function) for the edges trough a resume ($\oplus$) and an extension ($\odot$) operators [17]. *Intuitively speaking*, the fact that resume and extension have the properties of *C-semirings* grants that *the result is independent of the ordering in which such operations are applied*:

(1) *associativity* grants independence of the application ordering considering a single operator, while

(2) the *distributivity* of the extension operator with respect to the resume one grants that, given two constraints c_1 and c_2 between two points t_1 and t_2, and a constraint c_3 between t_2 and t_3, one gets the same results (2.1) by first resuming c_1 and c_2, and then by composing through the extension operator the result with c_3 to evaluate the constraint between t_1 and t_3, or (2.2) by first composing c_1 and c_3 and c_2 and c_3 through the extension operator, to determine two new constraints between c_1 and c_3, and then resuming the results.

Notably, the identity ***0*** for $\oplus$ intuitively represents the non-existing constraint, while the identity ***1*** for $\odot$ intuitively correspond to the distance between a point and itself.

4 Query Answering Facilities

Given a KB of QTCPs, temporal reasoning evaluates the *tightest* temporal constraints between each pair of time points. This is like an implicit representation of the "possible solutions" to the problem. Thus, it is very important to provide users with supports for querying the resulting set of constraints (see, e.g., Example 2). Part of the BNF grammar of our query language is reported in Figure 4.

<Query> ::= <HypQ> | <StandardQ>

<HypQ> ::= <StandardQ> IF <QTNP>
<StandardQ> ::= {<BaseQList>} | IP <Op> <Pref> |
GP <Op> <Pref> | {<BoolQ>}
<BaseQList> ::= <Point> ? <Point> |
<Point> ? <Point> , <BaseQList>
<BoolQ> ::= <SBTQ> | <CBTQ>

Fig 4. Query language (BNF grammar).

- Hypothetical queries (<HypQ>) are standard queries <StandardQ> that have to be answered with the assumption that some additional temporal constraints (such a set of QTCPs is a Quantitative Temporal Network with Preferences, indicated by <QTNP>) hold.

Example 7. "<StandardQ> IF $\{t_1<(1,2),(2,4)>t_2\}$" asks the query <StandardQ> in the hypothesis that t_2 is executed within 12 (1 unit) or 24 hours (2 units) after t_1. ■

Such queries are answered by first (provisionally) adding the new QTCPs to the set of constraints, and then applying our instantiation of Cormen et al.'s algorithm to obtain the new tightest constraints. A warning is given in case the new constraints are not consistent with the previous ones. Then, <StandardQ> is answered (as detailed below) in the new set of constraints. Notice also that Resume **($\oplus^P$)** must be used in order to add the new hypothetical constraints.

We distinguish among four types of standard queries.

- Basic extraction queries (<BaseQList>) ask for the temporal distances (and their preferences) between a list of pairs of time points. Such queries are trivially answered by reading the temporal constraints from the minimal network.

- Individual preference (IP) queries provide as output the constraints in a QTNP obtained by removing (from each constraint) all those pairs (d,p) such that p <Op> <Pref> does not hold, where <Op> is a comparison operator (i.e., one of <, ≤, =, ≥, >), and <Pref> is a preference value. Empty QTCPs are removed from the output.

Example 8. In Example 4, one may want to obtain only the constraints with preference greater or equal to 4. The corresponding IP query is "IP ≥ 4". ■

- Global preference (GP) queries are similar to individual preference queries, but, after the removal, the resulting constraints are propagated, using our algorithm.

- Boolean (<BoolQ>) queries can be simple (SBTQ) or composed (CBTQ). Such queries ask about the validity of one (SBTQ) or more (CBTQ) constraints between pair of points, and return a boolean value.

An SBTQ $t_i<(d_1$ Op $p_1)>t_j$ (e.g., "$t_1<(3 \geq 2)>t_2$") is simply answered by considering in the tightest constraints relating t_i and t_j and checking whether the distance between t_i and t_j may be d_1, and whether its preference satisfies the condition imposed by the Op operator.

Answering a CBTQ query (e.g., "$\{t_1<(3, \geq 1)>t_2$, $t_1<(5 \geq 2)>t_3\}$) requires four steps.

(1) each SBTQ is checked independently of the others. If at least one of them is not satisfied, a negative answer is provided. Otherwise, steps 2,3, and 4 are performed.
(2) For each SBTQ $t_i <(d_1$ Op $p_1)> t_j$ the corresponding QTCP relating t_i and t_j is modified by removing all the distances except d_1.
(3) The resulting constraints are then propagated via (our instantiation of) the Cormen et al.'s algorithm.
(4) The answer is YES if the resulting set of constraint is consistent, NO otherwise.

5 Comparisons and Conclusions

The literature about temporal constraint is moving from the treatment of "crisp" temporal constraint to fuzzy or probabilistic constraints, to account for preferences and\or uncertainty. However, until now, no approach to "non-crisp" temporal constraints has focused on the evaluation of the tightest temporal constraints, and on query answering.

In the wide literature in the area (see the introductory section), some other approaches are based on *C-semirings* (e.g., [15], [18]). The approach by Kathib et al. [15] is the closest to our one. Khatib et al. consider *quantitative* temporal constraints (and, specifically, STP and TCSP constraints [6]) and extend them by associating *preferences* to distances. Indeed, Kathib et al. define their *resume* and *extension* (using a different terminology) operations ***between preferences only*** in such a way that they form a C-semiring. On the other hand, they adopt standard STP/TCSP [15] composition and intersection operators to propagate *distances*. As a consequence, their *overall* operations for determining the labels (λ function) of the graph's edges (i.e., the *whole* operations which provide as output the labels, i.e., both their *preferences* ***and*** *distances*) ***do not*** form a C-semiring. For instance, it is easy to show that in Katib et al.'s approach, in the evaluation of edges's labels, the *distributivity* of the *extension* operator with respect to the *resume* operator ***does not*** hold. Thus, Katib et al.'s constraint propagation approach is *not ordering-independent* (see point (2) at the end of Section 3.4), so that it ***does not*** always provide the *tightest* constraints between each pair of time points.

Our approach provides a main advance with respect to the state of the art: it is the first approach to the propagation of "not-crisp" temporal constraints that aims at

(i) giving to users reliable *query-answering* facilities
(ii) providing the evaluation of the *tightest* constraints between temporal entities.

To achieve (ii), we defined our resume and extension operator in such a way that they form a C-semiring structure, so that we can exploit the properties of Cormen et al.'s all-to-all shortest path algorithm [17]. Notably, if one neglects the initialization phase (lines 1–5), the Compute-Summaries algorithm is a *generalization* of Floyd-Warshall's algorithm (in which *min* and *addition* are generalized by the *resume* and the *extension* operators respectively), which is widely used to compute the *minimal network* (i.e., the *tightest* constraints) for STP temporal constraints. In this sense, we can

say that we extend classical approaches to "crisp" quantitative (STP) temporal constraints to consider also preferences. This consideration suggests a possible evolution. Recently, several algorithms have been devised in order to optimize Floyd-Warshall's algorithm in the evaluation of STP minimal networks (consider, for instance, delta-STP [19] or P3C [20]). In our future work, we aim at investigating whether such optimizations can be used also in our context, in which also preferences have to be taken into account. Moreover, the work in [16] presents several interesting advances with respect to the state of the art, including the capability of coping with four different types of preferences: numeric and symbolic temporal preferences, composite preferences and conditional preferences. In our future work, we aim at investigating whether our approach could be extended to provide such an additional expressiveness.

Finally, we want to conclude by highlighting that, although the approach we propose is totally domain and task independent, we aim at applying it mainly within our GLARE project [21], a long-term project (started in 1997) with one of the major hospitals in Italy, aiming at providing physicians with decision support through the management of clinical guidelines.

ACKNOWLEDGMENTS

The author is very grateful to Fabio Rapallo for useful insights concerning C-Semirings. The research in this paper has been partially supported by INdAM and by UPO, Ricerca Locale.

REFERENCES

[1] L. Vila, "A Survey on Temporal Reasoning in Artificial Intelligence," *AI Commun*, vol. 7, no. 1, pp. 4–28, 1994.
[2] E. Schwalb and L. Vila, "Temporal Constraints: A Survey," *Constraints*, vol. 3, no. 2–3, pp. 129–149, Jun. 1998.
[3] P. Terenziani, "Reasoning about Time," in *Encyclopedia of Cognitive Science*, John Wiley & Sons, Ltd, 2006, pp. 869–874.
[4] J. F. Allen, "Maintaining knowledge about temporal intervals," *Commun. ACM*, vol. 26, no. 11, pp. 832–843, Nov. 1983.
[5] M. B. Vilain and H. A. Kautz, "Constraint Propagation Algorithms for Temporal Reasoning," in *Proceedings of the 5th National Conference on Artificial Intelligence. Philadelphia, PA, August 11-15, 1986. Volume 1: Science*, 1986, pp. 377–382.
[6] R. Dechter, I. Meiri, and J. Pearl, "Temporal Constraint Networks," *Artif Intell*, vol. 49, no. 1–3, pp. 61–95, May 1991.
[7] M. Koubarakis, "From local to global consistency in temporal constraint networks," *Theor. Comput. Sci.*, vol. 173, no. 1, pp. 89–112, Feb. 1997.
[8] I. Meiri, "Combining Qualitative and Quantitative Constraints in Temporal Reasoning," *Artif Intell*, 87, no. 1–2, pp. 343–385, 1996.
[9] H. A. Kautz and P. B. Ladkin, "Integrating Metric and Qualitative Temporal Reasoning," in *Proceedings of the Ninth National Conference on Artificial Intelligence - Volume 1*, Anaheim, California, 1991, pp. 241–246.
[10] D. Dubois, H. Fargier, and H. Prade, "Possibility theory in constraint satisfaction problems: Handling priority, preference and uncertainty," *Appl. Intell.*, vol. 6, no. 4, pp. 287–309, Oct. 1996.
[11] V. Ryabov and A. Trudel, "Probabilistic temporal interval networks," in *Proc. 11th International Symposium on Temporal Representation and Reasoning, 2004. TIME 2004.*, 2004, pp. 64–67.
[12] M. Mouhoub and J. Liu, "Managing uncertain temporal relations using a probabilistic Interval Algebra," in *IEEE International Conference on Systems, Man and Cybernetics*, 2008, pp. 3399–3404.
[13] S. Badaloni and M. Giacomin, "The algebra IAfuz: a framework for qualitative fuzzy temporal reasoning," *Artif. Intell.*, vol. 170, no. 10, pp. 872–908, Jul. 2006.
[14] S. Barro, R. Marín, J. Mira, and A. R. Patón, "A model and a language for the fuzzy representation and handling of time," *Fuzzy Sets Syst.*, vol. 61, no. 2, pp. 153–175, Jan. 1994.
[15] L. Khatib, P. Morris, R. Morris, and F. Rossi, "Temporal Constraint Reasoning with Preferences," in *Proceedings of the 17th International Joint Conference on Artificial Intelligence - Volume 1*, San Francisco, CA, USA, 2001, pp. 322–327.
[16] M. Mouhoub and A. Sukpan, "Managing Temporal Constraints with Preferences," *Spat. Cogn. Comput.*, vol. 8, no. 1–2, pp. 131–149, May 2008.
[17] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*. MIT Press and McGraw-Hill Book Company, 1989.
[18] S. Bistarelli, U. Montanari, F. Rossi, and F. Santini, "Unicast and Multicast QoS Routing with Soft-constraint Logic Programming," *ACM Trans Comput Log.*, vol. 12, no. 1, p. 5:1–5:48, Nov. 2010.
[19] L. Xu and B. Y. Choueiry, "A new effcient algorithm for solving the simple temporal problem," , in Proc. *10th Int'l Symposium on Temporal Representation and Reasoning, 2003*, pp. 210–220.
[20] L. Planken, M. de Weerdt, and R. van der Krogt, "P3C: A New Algorithm for the Simple Temporal Problem," in *Proc. of the 18th International Conference on International Conference on Automated Planning and Scheduling*, Sydney, Australia, 2008, pp. 256–263.
[21] A. Bottrighi and P. Terenziani, "META-GLARE: A meta-system for defining your own computer interpretable guideline system—Architecture and acquisition," *Artif. Intell. Med.*, vol. 72, pp. 22–41, Sep. 2016.

Paolo Terenziani received the Laurea in Computer Science at the University of Turin (Italy) with 110/110 and laude. On 1993 he got the PhD in Computer Science at the Universities of Turin and Milan (Italy). In 1993 he became a Researcher; in 1998 he became Assistant Professor. Since 2000 he is Full Professor at DISIT, University of Eastern Piedmont, Alessandria, Italy. Paolo Terenziani's research activity covers Artificial Intelligence, Databases, and Medical Informatics. He published more than 150 papers about these topics in refereed international journals and conference proceedings. As a recognition of his research merits, in 1998 he got the annual "Artificial Intelligence Prize" from the Italian Association for Artificial Intelligence.

Antonella Andolina was born in Siracuse (Italy) on May 20th, 1964. In 1990 she got the Laurea in Computer Science at the University of Turin (Italy) with 105/110 and laude. On 1991 she got the Master in Informatics and Automation at Consorzio per la Ricerca e l'Educazione Permanente (COREP) at the Politecnico, Turin, Italy.

From 20\10\1991 to 26\10\1992 she worked as Software Engineer and Product Manager at Mesar Team, Turin. Since 27\10\1992 she is Professor in Computer Science in the High Schools. She currently works at the ITCS Sommeiller, Turin. Her research activity concerns mainly the field of Artificial Intelligence, and specifically the areas of knowledge representation and temporal reasoning.

Luca Piovesan is a Postdoctoral researcher at DISIT, University of Eastern Piedmont, Alessandria, Italy. He received the master and PhD degrees in computer science from Università degli Studi di Torino, in 2012 and 2016, respectively. His research interests include artificial intelligence, medical informatics, and temporal databases.