Temporal Reasoning with Layered Preferences

Luca Anselma[1], Alessandro Mazzei[1], Luca Piovesan[2] and Paolo Terenziani[2]

[1] Dipartimento di Informatica, Università di Torino, corso Svizzera 185, 10149 Torino, Italy
[2] DISIT, Università del Piemonte Orientale "A. Avogadro", Alessandria, Italy
{anselma,mazzei}@di.unito.it,
{luca.piovesan,paolo.terenziani}@uniupo.it

Abstract. Temporal representation and temporal reasoning is a central in Artificial Intelligence. The literature is moving to the treatment of "non-crisp" temporal constraints, in which also preferences or probabilities are considered. However, most approaches only support numeric preferences, while, in many domain applications, users naturally operate on "*layered*" scales of values (e.g., Low, Medium, High), which are domain- and task-dependent. For many tasks, including decision support, the evaluation of the *minimal network* of the constraints (i.e., the tightest constraints) is of primary importance. We propose the first approach in the literature coping with *layered preferences on quantitative temporal constraints*. We extend the widely used *simple temporal problem* (STP) framework to consider layered user-defined preferences, proposing (i) a formal representation of quantitative constraints with layered preferences, and (ii) a temporal reasoning algorithm, based on the general algorithm *Compute-Summaries*, for the propagation of such temporal constraints. We also prove that our temporal reasoning algorithm evaluates the minimal network.

Keywords: Temporal constraints with preferences, Temporal reasoning, Temporal constraint propagation.

1 Introduction

Representing and reasoning about time is fundamental in many "intelligent" tasks and activities, such as planning, scheduling, human–machine interaction, natural language understanding, diagnosis, robotics and data management. Since the Eighties, many different approaches to *quantitative* (i.e., considering the metric of time; e.g., *action B must be started at least 1 hour after the end of action A*) and\or *qualitative* (i.e., considering only the relative position of actions\events; e.g., *A before B*) temporal constraints have been developed (see, e.g., the surveys in [1]). A milestone approach, regarding quantitative constraints, is STP [2], in which constraints of the form $P_1[c,d]P_2$ model the minimal (c) and maximal (d) distance between a pair of time points (notably, the above constraints are equivalent to *Bounds on Differences* (BoD) constraints of the form $c \leq P_2 - P_1 \leq d$, and [c,d] is usually called ***admissibility interval***). Different AI approaches have been provided which, given a set of STP constraints, propose ***reasoning*** algorithms that propagate such constraints, to check their *consistency*, and\or to find a *solution* (i.e., an instantiation of all the variables such that all constraints are satisfied), or to make explicit the *minimal network* of constraints (i.e., a set of constraints which has exactly the same solutions of the original one, and in which the minimum and maximum implied distances between each pair variables are made explicit – see [2]).

Example 1. Let t_1, t_2 and t_3 be time points, and let KB be the following set of STP constraints: KB={t_1[10,15] t_2, t_2[20,30]t_3, t_1[25,40]t_3}. KB is consistent, and {t_1=0, t_2=10, t_3=30} is a *solution* (also termed *scenario*) of KB. The *tightest* constraints implied by KB (the so called *minimal network* [2]) are KB'={t_1[10,15] t_2, t_2[20,30]t_3, t_1[30,40]t_3} (in particular, the minimum distance between t_1 and t_3 is 30). ■

While in several tasks the goal is to find a scenario, in others, such as in decision support or mixed-initiative approaches, the minimal network of the constraints must be determined, to provide users with a compact representation of all the possible solutions (since the choice of a specific solution has to be left to the users).

So far, we have only considered "crisp" constraints, in the sense that they represent a set of "equally possible" distances between variables. All of these approaches rely on the framework of classical Constraint Satisfaction Problem (CSP), inheriting from it a number of fundamental limitations, mainly related to a lack of flexibility and a limited representation of uncertainty [3]. A paradigmatic example is the execution of clinical treatments with temporal constraints. Usually, expressing "crisp" temporal constraints is not possible/useful in the medical context: constraints are expressed in the form of recommendations, to be followed as much as possible.

Example 2. As a running example, we consider a case of a comorbid patient suffering from both urinary tract infection and gastroesophageal reflux. Nalidixic acid (NA) is an antibiotic used for the treatment of urinary tract infections. As maintenance therapy, it should be administered two times a day, with an interval of 12 hours between each pair of consecutive administrations. However, such a recommendation with a "high" preference is not strict since the interval can also be of 11 or 13 hours with a "medium" preference, or of 10, 14 or 15 hours with a "low" preference. On the other hand, calcium carbonate (CC) is used to treat gastroesophageal reflux and it should be taken with "high" preference after lunch and dinner, but it can also be taken with a "medium" preference during meals. Since the concurrent administration of the two drugs can lead to a decrease of the effect of NA, to avoid interactions the first administration of NA (henceforth NA1), in the morning, must be administered at least 1 hour before the first administration of CC (CC1). Analogously, the delay between the second administration of CC (CC2) and the second administration of NA (NA2), in the evening, should be at least 3 hours with "low" preference or 4 hours with "high" preference. Besides the above constraints, our approach can also consider constraints that may arise from patient's lifestyle preferences. In particular, in this example we suppose that the patient may want to take the first NA dose in the morning at 5am with "low" preference, or after 6am with "high" preference. Moreover, we hypothesize that the patient has lunch time preferences (which are 12pm or 1pm with "high" preference, 11am or 2pm with "medium" preference, 3pm with "low" preference), and dinner time preferences (which are 6pm, 7pm or 8pm pm with "high" preference, 9 pm with "medium" preference, 5pm or 10pm with "low" preference). ■

To deal with issues related to preferences in constraints, 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.

Concerning ***qualitative temporal constraints***, in their seminal work Badaloni and Giacomin [4] have defined a new formalism in which the "crisp" qualitative temporal in Allen's Interval Algebra are associated with a degree of *plausibility,* and have proposed temporal reasoning algorithms to propagate such constraints. In [5] Dubois et al. have proposed the calculus of fuzzy Allen relations (including the composition table) and the patterns for propagating uncertainty about (fuzzy) Allen relations in a possibilistic way. Ryabov et al. [6] attach a *probability* to each of Allen's basic interval relations. A similar probabilistic approach has been proposed more recently by Mouhoub and Liu [7], as an adaptation of the general probabilistic CSP framework. Finally, in [8] Dubois et al. have shown how possibilistic temporal uncertainty can be handled in the setting of point algebra. ***"Non-crisp" quantitative temporal constraints*** have been considered by Khatib et al. [9], that extended the STP and the TCSP framework [2] to consider temporal *preferences*. An analogous approach has been recently proposed in [10]. However, such approaches only consider numeric preferences while, in many application domains, experts express preferences in term of a "layered" scale of "qualitative" preferences (e.g., <Low,Medium,High>). The approach in this paper overcomes such a limitation, being parametric with respect to the scale (see Section 4 for detailed comparisons with the literature).

2 Representing STP constraints with layered preferences

In current approaches in the literature augmenting STP with preferences, given a constraint $P_1[c,d]P_2$, a numeric value of preference is associated with each possible distance in the admissibility interval [c,d]. However, in many areas and applications, preferences distribute in a "regular" way over the intervals, forming a sort of "*pyramid*" of ***nested*** admissibility intervals, in which the top interval has the highest preference and the bottom the lowest one. Example 2 is just one paradigmatic example, but this is the case in all situations in which preferences are "centered" on a given set of temporal values, and decrease while getting far from this center. In this paper, we focus on such nested distributions of preferences (that we will call "***pyramid***" for short), showing that considering such distributions provide several advantages with respect to the association of preferences with each possible distance in the admissibility intervals. First, we introduce the notion of "layered" scale of preferences.

Definition 1. Scale of Qualitative Preferences (SQP). An SQP (or "scale", for short) S_r of ***cardinality*** *r* is composed by an enumerative set $\{p_1, \dots, p_r\}$ of r labels (r>0), and a strict and total ordering relation < over the set. For simplicity, we denote an SQP by an ordered list $\langle p_1, \dots, p_r \rangle$, such that $\forall i, 1 \leq i < r, p_i < p_{i+1}$. ■

Terminology. Given an SQP S_r of ***cardinality*** *r,* we indicate by $S_r(i)$ the ith value in the scale S_r $(1 \leq i \leq r)$ and we denote by $\mathbb{S}$ the domain of SQPs. ■

For example, an SQP coping with Example 2 is S_3^{ex}:<low, medium, high>.

We can now formally define the notion of preference function, focusing specifically on "pyramid" preferences. Intuitively speaking, a pyramid preference function over an admissibility interval D is a function such that the preference values (expressed as a SQP) (non-strictly) increase until the maximum preference value is reached at a certain value v∈D, and then (non-strictly) decrease. Formally:

Definition 2. Pyramid preference function (PPF). A pyramid preference function $Pref_{S_r,D}$ with scale $S_r \in \mathbb{S}$ over an admissibility interval D is a total function $Pref_{S_r,D}: D \rightarrow S_r$ such that:

$$\exists v \in D, \forall v_1, v_2 \in D,$$
$$\Big((v_1 \leq v \wedge v_2 \leq v \wedge v_1 \leq v_2) \Rightarrow Pref_{S_r,D}(v_1) \leq \; Pref_{S_r,D}(v_2)\Big) \wedge$$
$$\Big((v \geq v_1 \wedge v \geq \; v_2 \wedge v_1 \leq v_2) \Rightarrow Pref_{S_r,D}(v_1) \geq \; Pref_{S_r,D}(v_2)\Big)$$

If p_s is the maximum value of a PPF (over a scale $S_r \in \mathbb{S}$), we say the PPF has height s. If we term $\mathbb{PP}$ the domain of PPFs, STP constraints with pyramid preferences can be abstractly defined as follows.

Definition 3. STP constraint with Pyramid preferences (PyP_STP). Given a scale $S_r \in \mathbb{S}$, a PyP_STP is a constraint $\langle x, y, [c,d], Pref_{S,[c,d]}\rangle$ where $Pref_{S,[c,d]}$ is a PPF in $\mathbb{PP}$. ■

Terminology. We say that the *height* of PyP_STP is the *height* of its preference function.

We can exploit the fact that preferences form a pyramid of height s of nested ***admissibility intervals*** by proposing a compact and "user-friendly" representation of PyP_STP constraints (of height s).

Definition 4. Compact representation of STP constraints with Pyramid preferences. Given a scale $S_r \in \mathbb{S}$ of cardinality r, a PyP_STP of height s (s≤r) can be compactly represented by a constraint $\langle x, y, \langle\langle [c_1, d_1], p_1\rangle, \ldots, \langle [c_s, d_s], p_s\rangle\rangle\rangle$ where $[c_i, d_i]$, $i = 1, \ldots, s$ are admissibility intervals such that $c_i \leq c_{i+1} \wedge d_i \geq d_{i+1}$, $1 \leq i < s$ and $p_1, \ldots, p_s \in S_r$ denote the s lowest qualitative values in the scale S_r (i.e., $p_i = S_r(i)$, $1 \leq i \leq s$). ■

For instance, the constraint between two consecutive administrations of NA (NA1 and NA2) in Example 2 can be represented through the "compact" PyP_STP constraint $\langle NA1, NA2, \langle\langle [10,15], low\rangle, \langle [11,13], medium\rangle, \langle [12,12], high\rangle\rangle\rangle$.

Let $\mathbb{P}\text{y}\mathbb{P}\_\mathbb{STP}$ denote the domain of the "compact" PyP_STPs as defined above.

The ***semantics*** of a "compact" PyP_STP of the form $\langle x, y, \langle\langle [c_1, d_1], p_1\rangle, \ldots, \langle [c_s, d_s], p_s\rangle\rangle\rangle$ over a scale $S_r \in \mathbb{S}$ of cardinality r (s≤r) is that it compactly represent a constraint $\langle x, y, [c_1, d_1], Pref_{S,[c,d]}\rangle$ (see Definition 3) where $Pref_{S,[c,d]}$ is such that:

(i) $\forall v \in [c_s, d_s] \; Pref_{S,D}(v) = S_r(s)$
(ii) $\forall i \; 1 \leq i < s \; \forall v \in ([c_i, d_i] - [c_{i+1}, d_{i+1}]) \; Pref_{S,D}(v) = S_r(i)$

The intuitive meaning of a "compact" PyP_STP $\langle x, y, \langle\langle [c_1, d_1], p_1\rangle, \ldots, \langle [c_s, d_s], p_s\rangle\rangle\rangle$ over a scale S_r (s≤r) is that the *difference y-x*

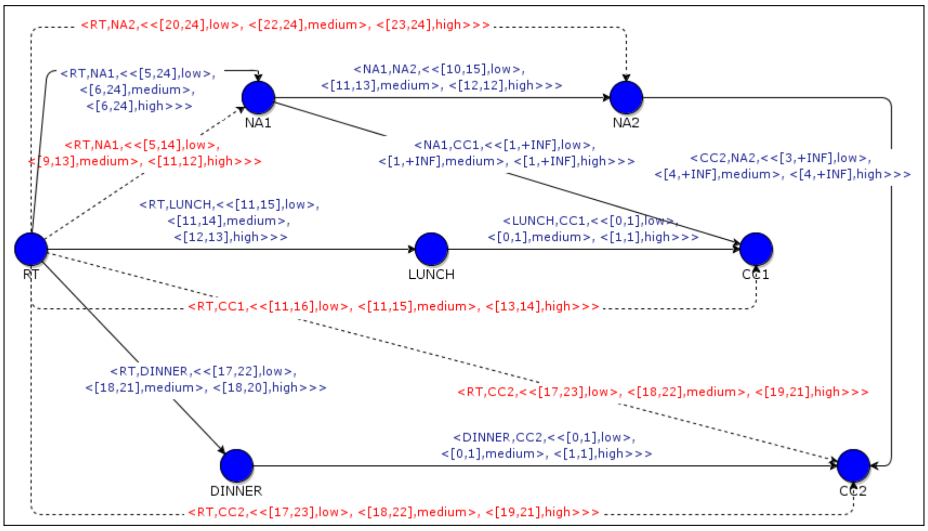
<RT,NA2,<<[20,24],low>, <[22,24],medium>, <[23,24],high>>>
<RT,NA1,<<[5,24],low>, <[6,24],medium>, <[6,24],high>>>
<NA1,NA2,<<[10,15],low>, <[11,13],medium>, <[12,12],high>>>
NA1
NA2
<RT,NA1,<<[5,14],low>, <[9,13],medium>, <[11,12],high>>>
<NA1,CC1,<<[1,+INF],low>, <[1,+INF],medium>, <[1,+INF],high>>>
<CC2,NA2,<<[3,+INF],low>, <[4,+INF],medium>, <[4,+INF],high>>>
<RT,LUNCH,<<[11,15],low>, <[11,14],medium>, <[12,13],high>>>
<LUNCH,CC1,<<[0,1],low>, <[0,1],medium>, <[1,1],high>>>
RT
LUNCH
CC1
<RT,CC1,<<[11,16],low>, <[11,15],medium>, <[13,14],high>>>
<RT,DINNER,<<[17,22],low>, <[18,21],medium>, <[18,20],high>>>
<RT,CC2,<<[17,23],low>, <[18,22],medium>, <[19,21],high>>>
<DINNER,CC2,<<[0,1],low>, <[0,1],medium>, <[1,1],high>>>
DINNER
CC2
<RT,CC2,<<[17,23],low>, <[18,22],medium>, <[19,21],high>>>

Figure 1. Graphical representation of the constraints in Example 2. The solid edges represent the input constraints. The dashed edges represent part of the output of temporal reasoning.

between y and x is in $[c_s, d_s]$ with preference $S_r(s)$, or in $[c_{s-1}, d_{s-1}] - [c_s, d_s]$ with preference S_r(s-1), or …. or in $[c_1, d_1] - [c_2, d_2]$ with preference $S_r(1)$

For example, the "compact" PyP_STP $\langle NA1, NA2, \langle\langle[10,15], low\rangle, \langle[11,13], medium\rangle, \langle[12,12], high\rangle\rangle\rangle$ represents the fact that the difference between NA2 and NA1 has preference "high" if it is exactly 12, "medium" if it is 11 or 13, and "low" if it is 10, or between 14 and 15.

As in the general case, a graph representation can be provided for a set of "compact" PyP_STP constraints (as defined in Definition 4).

Definition 5. Graph of PyP_STP (PyP_STP_G). Given an SQP S_r of cardinality r, a set B of "compact" PyP_STP $\langle x, y, \langle\langle[c_1, d_1], p_1\rangle, \dots, \langle[c_s, d_s], p_s\rangle\rangle\rangle$ can be represented by an oriented graph $G = \langle V, E\rangle$ with a labelling function λ, where V (the set of nodes) represents the set of variables in B, E ⊆ V×V, and $\lambda: E \rightarrow \mathbb{PyP\_STP}$. ■

In Figure 1, we graphically represent the PyP_STP_G arising from Example 2, at a granularity of hours. We represent the constraints in Example 2 with solid edges labelled with blue lettering. Notice that, in the figure, the time points RT, LUNCH and DINNER represent respectively the reference time (which, in our example, is 12am of the current day), and lunch and dinner times.

3 Temporal reasoning

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, ⊕, ⊙, **0**, **1**)* in [11], which is shown in Figure 2 below. *Compute-Summaries* is indeed a highly parametric *all-to-all shortest paths* algorithm to solve different problems concerning oriented paths in a graph. Such an algorithm takes as input a graph *(V,E)*, a labelling function $\lambda: E \rightarrow L$ 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 $\langle L,\oplus,\odot,\mathbf{0},\mathbf{1}\rangle$ is a *closed semiring* (see Section 3.4), evaluates the *all-to-all shortest paths* (i.e., the *minimal network*) [11]. Thus, our idea is to define the *extension* $\odot^P$ and *resume* $\oplus^P$ operators (and their *identities*) operating on "compact" PyP_STP constraints (i.e., on both *admissibility intervals* and *preferences*) in such a way that the structure $\langle L,\oplus^P,\odot^P,\mathbf{0},\mathbf{1}\rangle$ is a *closed 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 *minimal network*, as desired.

We adopt the notation in [11]: (i) 1,2,…,n indicate the vertices in V, where n=|V|; (ii) we extend the notion of labelling function to paths: the label of a path p=<v_1,v_2, …, v_t> is $\lambda(p)=\lambda(v_1,v_2)\odot\lambda(v_2,v_3)\odot\ldots\odot\lambda(v_{t-1},v_t)$; (iii) 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., $L_{ij} = \oplus_{p=i\rightarrow j}\lambda(p)$); (iv) 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., $L_{ij}^k = \oplus_{p\in Q}$ $\lambda(p)$, where Q is the set of all paths in (V,E) connecting i to j and traversing only the nodes in {1,…,k}). We assume that $\lambda(i,j)=\mathbf{0}$ if $(i,j)\notin E$.

```
Compute-Summaries(λ,V,E,⊕,⊙,0,1) algorithm
1. n ← |V|
2. for i←1 to n do
3.  for j←1 to n do
4.         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 do
7.   for i←1 to n do
8.         for j←1 to n do
9.            $L_{ij}^k$ ← $L_{ij}^{k-1}$ ⊕ ($L_{ik}^{k-1}$ ⊙ $L_{kj}^{k-1}$)
```

Figure 2. Compute-Summaries algorithm

Whenever the application of $\oplus$ and $\odot$ operators in line 9 obtains an inconsistent constraint (see below), the algorithm stops and signals the inconsistency.

Complexity. The complexity of Compute-Summaries is $\Theta\left(n^3 \cdot \left(T_\oplus + T_\odot\right)\right)$, where $T_\oplus$ and $T_\odot$ denote the time required to evaluate $\oplus$ and $\odot$ respectively [11].

We can still adopt the above algorithm to perform correct and complete propagation, but we have to extend it with a proper definition of the $\oplus$ and $\odot$ operators (that we term $\oplus^P$ and $\odot^P$), to deal with "compact" PyP_STP constraints. As regards the operations on the admissibility intervals, we adopt the "standard" ones adopted in the STP framework [2]: the resume of two admissibility intervals [c_1,d_1] and [c_2,d_2] is their intersection [max(c_1,c_2), min(d_1,d_2)], while their extension is the sum of their starting and the ending points [c_1+c_2, d_1+d_2]. However, in the case of "compact"

PyP_STP constraints, such operations have to be iterated on the admissibility intervals at each level of the pyramids. As regards preferences, they can be easily evaluated by stating that, at each layer i, the preference p_i=S(i) is retained.

In the constraint propagation algorithm, in case a degenerate interval is computed at a level *i* (i.e., the intersection is empty), all admissibility intervals higher or equal than *i* can be dropped from the set of "compact" PyP_STPs since this corresponds to an inconsistency at level *i*. If the inconsistency is at the first level of the PyP_STPs, then the entire set of PyP_STPs is inconsistent.

Our formal definition of the resume operator $\oplus^P$ is reported below. At each level *i* of the PyP_STPs C' and C'', the intersection between the admissibility intervals at level *i* in C' and C'' is computed, and its result is paired with the preference associated with that level, until the highest level common to both constraints is reached or an inconsistency at a given level is detected.

Definition. Resume ($\oplus^P$). Given a scale S_r, and given two "compact" PyP_STPs $C' = \langle x, y, \langle\langle [c'_1, d'_1], p_1\rangle, \ldots, \langle [c'_k, d'_k], p_k\rangle\rangle\rangle$ and $C'' = \langle x, y, \langle\langle [c''_1, d''_1], p_1\rangle, \ldots, \langle [c''_l, d''_l], p_l\rangle\rangle\rangle$ (where $1 \le k \le r$ and $1 \le l \le r$), their resume is obtained as follows:

$$C' \oplus^P C'' = \langle x, y, \langle\langle [\max(c'_i, c''_i), \min(d'_i, d''_i)], p_i\rangle, i = 1, \ldots, \min(k, l)\rangle\rangle$$

For instance, the resume operation applied to the two following "compact" PyP_STP constraints between the points RT and NA2 produces a new restricted PyP_STP constraint as result $\langle RT, NA2, \langle\langle [0,24], low\rangle, \langle [0,24], medium\rangle, \langle [0,24], high\rangle\rangle\rangle \oplus^P \langle RT, NA2, \langle\langle [15,39], low\rangle, \langle [17,37], medium\rangle, \langle [18,36], high\rangle\rangle\rangle = \langle RT, NA2, \langle\langle [15,24], low\rangle, \langle [17,24], medium\rangle, \langle [18,24], high\rangle\rangle\rangle$

In the extension operator $\odot^P$, at each level *i* of the pyramids, the new admissibility interval is computed by summing pairwise the starting and ending points of the input admissibility intervals at level *i*. The resulting admissibility interval is paired with the preference associated with that level, until the highest level, i.e., the highest level common to both constraints, is reached.

Definition. Extension ($\odot^P$). Given a scale S_r, and given two PyP_STPs $C' = \langle x, y, \langle\langle [c'_1, d'_1], p_1\rangle, \ldots, \langle [c'_k, d'_k], p_k\rangle\rangle\rangle$ and $C'' = \langle y, z, \langle\langle [c''_1, d''_1], p_1\rangle, \ldots, \langle [c''_l, d''_l], p_l\rangle\rangle\rangle$ (where $1 \le k \le r$ and $1 \le l \le r$), their extension is obtained as follows:

$$C' \odot^P C'' = \langle x, z, \langle\langle [c'_i + c''_i, d'_i + d''_i], p_i\rangle, i = 1, \ldots, \min(k, l)\rangle\rangle$$

For instance, the resume operation applied to the two following "compact" PyP_STP constraints, concerning RT and NA1 and concerning NA1 and NA2 respectively, produces as a result a new PyP_STP constraint concerning RT and NA2 $\langle RT, NA1, \langle\langle [5,24], low\rangle, \langle [6,24], medium\rangle, \langle [6,24], high\rangle\rangle\rangle \odot^P \langle NA1, NA2, \langle\langle [10,15], low\rangle, \langle [11,13], medium\rangle, \langle [12,12], high\rangle\rangle\rangle\rangle = \langle RT, NA2, \langle\langle [15,39], low\rangle, \langle [17,37], medium\rangle, \langle [18,36], high\rangle\rangle\rangle$

Complexity of the $\oplus^P$ and $\odot^P$ operators. Since the intersection and the sum of two intervals can be computed in constant time and they are computed for each preference

value, both the $\oplus^P$ and $\odot^P$ operators can operate in time $\Theta(r)$, where r is the number of preference values in S_r, thus $T_{\oplus} = T_{\odot} = \Theta(r)$.
Thus, the complexity of the Compute-Summaries algorithm is $\Theta(n^3 \cdot r)$.

Identities for $\oplus^P$ and $\odot^P$. Given a Scale S_r with cardinality r, the identity ***1*** for $\odot^P$ is $\perp = \langle\langle[0,0], p_1\rangle, \ldots, \langle[0,0], p_r\rangle\rangle$ since, given any PyP_STP $C = \langle\langle[c_1, d_1], p_1\rangle, \ldots, \langle[c_r, d_r], p_r\rangle\rangle$, $C \odot^P \langle\langle[0,0], p_1\rangle, \ldots, \langle[0,0], p_r\rangle\rangle = \langle\langle[0,0], p_1\rangle, \ldots, \langle[0,0], p_r\rangle\rangle \odot^P C = C$.

The identity ***0*** for $\oplus^P$ is $\top = \langle\langle[-\infty, +\infty], p_1\rangle, \ldots, \langle[-\infty, +\infty], p_r\rangle\rangle$ since, given any PyP_STP $C = \langle\langle[c_1, d_1], p_1\rangle, \ldots, \langle[c_r, d_r], p_r\rangle\rangle$, $C \oplus^P \langle\langle[-\infty, +\infty], p_1\rangle, \ldots, \langle[-\infty, +\infty], p_r\rangle\rangle = \langle\langle[-\infty, +\infty], p_1\rangle, \ldots, \langle[-\infty, +\infty], p_r\rangle\rangle \oplus^P C = C$.
$\langle\langle[-\infty, +\infty], p_1\rangle, \ldots, \langle[-\infty, +\infty], p_r\rangle\rangle$ is also the annihilator for $\odot^P$. In fact, given any PyP_STP $C = \langle\langle[c_1, d_1], p_1\rangle, \ldots, \langle[c_r, d_r], p_r\rangle\rangle$, $C \odot^P \langle\langle[-\infty, +\infty], p_1\rangle, \ldots, \langle[-\infty, +\infty], p_r\rangle\rangle = \langle\langle[-\infty, +\infty], p_1\rangle, \ldots, \langle[-\infty, +\infty], p_r\rangle\rangle \odot^P C = C$.

Intuitively, the identity ***0*** for $\oplus^P$ represents the non-existing constraint, while the identity ***1*** for $\odot^P$ corresponds to the distance between a point and itself.

Property. $\langle \mathbb{P}\mathrm{y}\mathbb{P}\_\mathbb{S}\mathbb{T}\mathbb{P}, \oplus^P, \odot^P, \perp, \top\rangle$ is a *closed semiring.*

Proof (sketch). By definition, both $\oplus^P$ and $\odot^P$ are ***closed*** over $\mathbb{P}\mathrm{y}\mathbb{P}\_\mathbb{S}\mathbb{T}\mathbb{P}$. $\oplus^P$ is ***associative***, since, it operates considering each preference level, and, at each level, it computes the maximum and the minimum of the endpoints of the admissibility intervals, and the maximum and the minimum operators are associative. For the same reasons, $\oplus^P$ is also ***commutative*** and ***idempotent***. Thus, since $\perp$ is the identity for $\oplus^P$, $\langle \mathbb{P}\mathrm{y}\mathbb{P}\_\mathbb{S}\mathbb{T}\mathbb{P}, \oplus^P, \perp\rangle$ is a ***commutative and idempotent monoid***. $\odot^P$ is ***associative***, since at each preference level it performs the pairwise sum of the endpoints of the admissibility intervals, and the sum is associative. Thus, since $\top$ is the identity for $\odot^P$, $\langle \mathbb{P}\mathrm{y}\mathbb{P}\_\mathbb{S}\mathbb{T}\mathbb{P}, \odot^P, \top\rangle$ is a ***monoid***. Moreover, $\top$ is an annihilator for $\odot^P$ and $\odot^P$ ***distribute*** over finite and countably infinite $\oplus^P$: this is due to the fact that pairwise sum distributes over the minimum and the maximum operators. Thus, $\langle \mathbb{P}\mathrm{y}\mathbb{P}\_\mathbb{S}\mathbb{T}\mathbb{P}, \oplus^P, \odot^P, \perp, \top\rangle$ is a *closed semiring.* ■

As an example of application of our version of the Compute-Summaries algorithm, in Figure 1 we report with dashed edges labelled with red lettering some of the "compact" PyP_STP constraints obtained from the application of the algorithm to Example 2 (note that, due to graphic reasons, we cannot represent all the PyP_STP constraints of the minimal network). For example, the Compute-Summaries algorithm adopting our $\oplus^P$ and $\odot^P$ operators has restricted the original constraint between RT and NA1 to the constraint $\langle RT, NA1, \langle\langle[6,14], low\rangle, \langle[9,13], medium\rangle, \langle[11,12], high\rangle\rangle\rangle$, which limits the first administration of NA between 6am and 2pm with a "low" preference, between 9am and 1pm with a "medium" preference, and between 11am and 12pm with an "high" preference; moreover, a new constraint $\langle RT, NA2, \langle\langle[20,24], low\rangle, \langle[22,24], medium\rangle, \langle[23,24], high\rangle\rangle\rangle$ has been derived between RT and NA2, which limits the second administration of NA between 8pm and 12am with a “low” preference, between 10pm and 12am with a “medium” preference, and between 11pm and 12am with a “high” preference.

We have implemented our approach in Java (JDK 8), and run experimental tests on an Intel Core i7-6700HQ CPU, considering different numbers of nodes (from 200 to 1000), and scales of different cardinalities (4, 8, 12, 16, and 20). Table 1 shows the computation times. Results show that the execution time, starting from 600 nodes, is cubic in the number of the nodes and grows linearly in the cardinality of the scales.

Table 1. Experimental results (computation times in ms).

# levels \ # nodes	**200**	**400**	**600**	**800**	**1000**
4	91	508	864	2102	3965
8	47	524	1602	3869	7976
12	82	717	2402	5722	11133
16	113	985	3300	7468	15511
20	127	1192	3849	9132	18262

4 Comparisons and conclusions

Until now, only few AI approaches have considered quantitative temporal constraints with preferences. Among them, the approach in this paper is the first one that has focused on user-defined layered preferences and the evaluation of the tightest constraints. The approach in the literature which is the closest one to the one in this paper is the approach by Terenziani et al. [10]. In [10], the authors have extended STP by associating a quantitative (numeric) preference with each possible distance between time points. As in this paper, they have exploited *closed semirings* to grant that the tightest constraints are evaluated by the reasoning algorithm. The approach in this paper generalizes and extends the approach in [10] to two main respects: (i) we generalize it to operate also on continuous domains for the variables (while [10] only copes with discrete domains), (ii) we support user-defined layered preferences (while only numeric preferences were considered in [10]).

Notably, the treatment of layered preferences provides several practical and theoretical advantages. First of all, it facilitates users, that are not required to assign a specific numeric value of preference to each distance. Second, it reduces the space complexity of the representation of constraints (in [10], each possible value in each constraint distance must be explicitly stored, together with its preference). Third, it reduces the temporal complexity of the algorithm computing the tightest constraints (which in [10] is $\Theta(n^3 \cdot |D|^2)$, where D is the domain of the distance values).

Although our approach is domain- and task-independent, we aim at applying it mainly within our GLARE project [12], to support physicians in the treatment of comorbid patients [13], by integrating medical preferences in the reasoning process. Moreover, we also wish to investigate its integration in temporal relational databases for dealing with temporal indeterminacy [14–18].

References

1. Vila, L.: A Survey on Temporal Reasoning in Artificial Intelligence. AI Commun. 7, 4–28 (1994).
2. Dechter, R., Meiri, I., Pearl, J.: Temporal Constraint Networks. Artif. Intell. 49, 61–95 (1991).
3. Dubois, D., Fargier, H., Prade, H.: Possibility theory in constraint satisfaction problems: Handling priority, preference and uncertainty. Applied Intelligence. 6, 287–309 (1996).
4. Badaloni, S., Giacomin, M.: The algebra IAfuz: a framework for qualitative fuzzy temporal reasoning. Artificial Intelligence. 170, 872–908 (2006).
5. Dubois, D., HadjAli, A., Prade, H.: Fuzziness and Uncertainty in Temporal Reasoning. J. UCS. 9, 1168 (2003).
6. Ryabov, V., Trudel, A.: Probabilistic temporal interval networks. In: Proc. TIME 2004. 11th International Symposium on. pp. 64–67. IEEE (2004).
7. Mouhoub, M., Liu, J.: Managing uncertain temporal relations using a probabilistic interval algebra. In: Systems, Man and Cybernetics, 2008. SMC 2008. IEEE Int Conf on. pp. 3399–3404. IEEE (2008).
8. Dubois, D., HadjAli, A., Prade, H.: A possibility theory-based approach to the handling of uncertain relations between temporal points. Int. J. Intell. Syst. 22, 157–179 (2007).
9. Khatib, L., Morris, P., Morris, R., Rossi, F.: Temporal constraint reasoning with preferences. In: Proc. IJCAI. pp. 322–327. Morgan Kaufmann (2001).
10. Terenziani, P., Andolina, A., Piovesan, L.: Managing Temporal Constraints with Preferences: Representation, Reasoning, and Querying. IEEE Trans. Knowl. Data Eng. 29, 2067–2071 (2017).
11. Cormen, T.H., Leiserson, C.E., Rivest, R.L.: Introduction to Algorithms. The MIT Press and McGraw-Hill Book Company (1989).
12. Bottrighi, A., Terenziani, P.: META-GLARE: A meta-system for defining your own computer interpretable guideline system—Architecture and acquisition. Artificial Intelligence in Medicine. 72, 22–41 (2016).
13. Anselma, L., Piovesan, L., Terenziani, P.: Temporal detection and analysis of guideline interactions. Artif. Intell. Med. 76, 40–62 (2017).
14. Anselma, L., Bottrighi, A., Montani, S., Terenziani, P.: Extending BCDM to Cope with Proposals and Evaluations of Updates. IEEE Trans. Knowl. Data Eng. 25, 556–570 (2013).
15. Anselma, L., Stantic, B., Terenziani, P., Sattar, A.: Querying now-relative data. Journal of Intelligent Information Systems. 41, 285–311 (2013).
16. Anselma, L., Bottrighi, A., Montani, S., Terenziani, P.: Managing proposals and evaluations of updates to medical knowledge: theory and applications. J Biomed Inform. 46, 363–376 (2013).
17. Anselma, L., Piovesan, L., Terenziani, P.: A 1NF temporal relational model and algebra coping with valid-time temporal indeterminacy. Journal of Intelligent Information Systems. 47, 345–374 (2016).
18. Anselma, L., Piovesan, L., Sattar, A., Stantic, B., Terenziani, P.: A Comprehensive Approach to "Now" in Temporal Relational Databases: Semantics and Representation. IEEE Trans. Knowl. Data Eng. 28, 2538–2551 (2016).