# Constraint-based Temporal Reasoning with Preferences

Lina Khatib [a] Paul Morris [a] Robert Morris [a] Francesca Rossi [b] Alessandro Sperduti [b]
K. Brent Venable [b]
[a] *NASA Ames Research Center, Moffett Field, CA 94035 USA*
[b] *University of Padova, Dept. of Pure and Applied Mathematics, Via G. B. Belzoni 7, 35131 Padova, Italy*

Often we need to work in scenarios where events happen over time and preferences are associated with event distances and durations. Soft temporal constraints allow one to describe in a natural way problems arising in such scenarios.

In general, solving soft temporal problems requires exponential time in the worst case, but there are interesting subclasses of problems which are polynomially solvable. In this paper we identify one of such subclasses giving tractability results. Moreover, we describe two solvers for this class of soft temporal problems, and we show some experimental results. The random generator used to build the problems on which tests are performed is also described. We also compare the two solvers highlighting the tradeoff between performance and robustness.

Sometimes, however, temporal local preferences are difficult to set, and it may be easier instead to associate preferences to some complete solutions of the problem. To model everything in a uniform way via local preferences only, and also to take advantage of the existing constraint solvers which exploit only local preferences, we show that machine learning techniques can be useful in this respect. In particular, we present a learning module based on a gradient descent technique which induces local temporal preferences from global ones. We also show the behavior of the learning module on randomly-generated examples.

Keywords: temporal constraints, preferences, scheduling, learning constraints.

## 1. Introduction and Motivation

Several real world problems involving the manipulation of temporal information can naturally be viewed as having preferences associated with local temporal decisions. By a local temporal decision we mean one associated with how long a single activity should last, when it should occur, or how it should be ordered with respect to other activities.

For example, an antenna on an earth orbiting satellite such as Landsat 7 must be slewed so that it is pointing at a ground station in order for recorded science to be downlinked to earth. Assume that as part of the daily Landsat 7 scheduling activity a window $W$ is identified within which a slewing activity to one of the ground stations for one of the antennae can begin, and thus there are choices for assigning the start time for this activity. Notice that the time window represents a hard constraint in the sense that no slewing can happen outside such a time interval. Antenna slewing on Landsat 7 has been shown to occasionally cause a slight vibration to the satellite. Consequently, it is preferable for the slewing activity not to overlap any scanning activity. Thus, if there are any start times $t$ within $W$ such that no scanning activity occurs during the slewing activity starting at $t$, then $t$ is to be preferred. Of course, the cascading effects of the decision to choose $t$ on the scheduling of other satellite activities must be taken into account as well. For example, the selection of $t$, rather than some earlier start time within $W$, might result in a smaller overall contact period between the ground station and satellite, which in turn might limit the amount of data that can be downlinked during this period. This may conflict with the preference for attaining maximal contact times with ground stations, if possible.

Reasoning simultaneously with hard temporal constraints and preferences, as illustrated in the

AI Communications
ISSN 0921-7126, IOS Press. 

example just given, is crucial in many situations. We tackle this problem by exploiting the expressive power of semi-ring based soft constraints [5,6], an approach which allows to handle hard requirements and preferences at the same time. In particular, we embed this method for handling preferences into an existing model for handling temporal hard constraints. The framework we obtain allows to model temporal preferences of different types. Problems specified in this framework are in generally difficult to solve. However, there are subclasses of such problems which are tractable. In this paper we consider one of such subclasses, which is identified by a specific underlying hard constraint structure (Simple Temporal Problems [11]), by a specific semi-ring (the Fuzzy semiring where the goal is to maximize the minimum of the local preferences), and by preference functions shaped in a certain way (semi-convex functions).

While it is easy to imagine that the general framework can be used in many scenarios, one may wonder whether the specific tractable subclass we consider is useful in practice. We will consider each restriction in turn:

*Simple temporal problems* [11] require that the allowed durations or distances between two events are contained in a single temporal interval. This is a reasonable restriction in many problems. For example, this approach has been used to model and solve scheduling problems in the space application domain [1]. In general, what simple temporal constraints do not allow are disjunctions of the form "I would like to go swimming either before or after dinner". When such disjunctions are needed, one can always decompose the problem into a set of simple temporal problems [46]. However, this causes the complexity of the problem to increase.

*Maximizing the minimum preference* can be regarded as implementing a cautious attitude. In fact, considering just the minimum preference as the assessment of a solution means that one focuses on the worst feature. Preferences higher than the worst one are completely ignored. The optimal solutions are those where the worst feature is as good as possible. This approach is appropriate in many critical applications where risks avoidance is the main goal. For example, this is the case of medical and space applications.

*Semi-convex preference functions* are, informally, functions with only one peak. Such functions can model a wide range of common temporal preference statements such as "This event should last as long (or as little) as possible", "I prefer this to happen around a given time", or "I prefer this to last around a given amount of time".

For the tractable subclass considered in this paper, we provide two solvers, we study their properties, and we compare them in terms of efficiency on randomly generated temporal problems. This experiments, together with the tractability results of the paper, show that solving such problems is feasible in practice. This is not so obvious, since it proves that adding the expressive power of preferences to simple temporal constraints does not make the problems more difficult.

In some scenarios, specifying completely the local preference functions can be difficult, while it can be easier to rate complete solutions. This is typical in many cases. For example, it occurs when we have an expert, whose knowledge is difficult to code as local preferences, but who can immediately recognize a good or a bad solution.

In the second part of this paper we will consider these scenarios and we will induce local preference functions, via machine learning techniques, from solution ratings provided by an external source. The machine learning approach is useful when it is not known or evident how to model such ratings as a combination of local preferences. This methodology allows us to induce tractable temporal problems with preferences which approximate as well as possible the given set solution ratings. Experimental results show that the learned problems generalize well the given global preferences.

We envision several fields of application for the results presented in this paper. However, planning and scheduling for space missions has directly inspired our work, so we will refer to two examples in this area.

NASA has a wealth of scheduling problems in which temporal constraints have shown to be useful in some respect but have also demonstrated some weaknesses, one of which is the lack of capability to deal with preferences. Remote Agent [31], [27], represents one of the most interesting examples. This experiment consisted of placing an artificial intelligence system on board to plan and execute spacecraft activities. Before this experiment, traditional spacecrafts were subject to a low level direct commanding with rigid time-stamps which left the spacecraft little flexibility to shift around the time of commanding or to change the hard-

ware used to achieve the commands. One of the main features of Remote Agent is to have a desired trajectory specified via high-level goals. For example, goals can specify the duration and the frequency of time windows within which the spacecraft must take asteroid images. This experiment proved the power of temporal constraint-based systems for modeling and reasoning in a space application. The benefit of adding preferences to this framework would be to allow the planner to maximize the mission manager's preferences. Reasoning on the feasibility of the goals while maximizing preferences can then be used to allow the plan execution to proceed while obtaining the best possible solution preference-wise. Notice that our cautious approach to preferences is appropriate in this context due to its intrinsic critical nature.

Our learning approach has a direct application in this field as well. Consider for example Mapgen, the mixed-initiative activity plan generator, developed to produce the Mars daily plans for the two exploration rovers Spirit and Opportunity [1]. The main task performed by such a system is to generate plans and schedules for science and engineering activities, allowing hypothesis testing and resource computation and analysis. Such system has been developed using a hard constraint approach and in particular Simple Temporal Problems are the main underlying reasoning engine. Given a complete plan generated by Mapgen for a rover, it is rated globally according to several criteria. For example, an internal tool of Mapgen allows a computation of the energy consumption of such a plan, from which a resource-related preference can be obtained. On the other side, the judgment of the scientist requesting the data is fundamental. Furthermore, the engineers, who are responsible for the status of the instruments, should be able to express their preferences on the length and modality of usage of each equipment on board. During the mission, all these preferences were collected and an optimal plan was generated through human-interaction by tweaking manually the initial proposed plan. Since most of such preferences are provided as global ratings by the experts and have no explicit encoding in local terms, we believe our learning and solving system could allow the human-interaction phase to start directly from highly ranked plans. The application we foresee would allow, as a first step, to induce local preferences on the hard temporal constraints used by Mapgen from the different sources. Then the second step, which solves the obtained problems, would provide useful guidance to judge unexplored plans in terms of the different criteria.

The paper is organized as follows: Section 2 gives an overview of the background underlying our work. In particular, fundamental definitions and main results are described for temporal constraints, soft constraints, and machine learning. In Section 3 Temporal Constraints with Preferences (TCSPPs) are formally defined and various properties are discussed. After showing that TCSPPs are NP-hard, Simple Temporal Problems with Preferences (STPPs), that is, TCSPPs with one interval on each constraint, are studied. In particular, a subclass of STPPs, characterized by assumptions on both the underlying semiring and the shape of the preference functions, is shown to be tractable. In Section 4 two different solvers for such STPPs are described. Experimental results on the performance of both solvers are supplied in Section 5. In Section 6 a learning module designed for tractable STPPs is described, and experimental results on randomly generated problems are given.

Earlier versions of parts of this paper have appeared in [17], [38], in [35] and in [37].

## 2. Background

In this section we give an overview of the background on which our work is based. First we will describe temporal constraint satisfaction problems [11], a well known framework for handling quantitative time constraints. Then we will define semiring-based soft constraints [6]. Finally, we will give some background on inductive learning techniques, which we will use in Section 6 for learning local temporal preferences from global ones.

### *2.1. Temporal constraints*

One of the requirements of a temporal reasoning system is its ability to deal with metric information. In other words, a well designed temporal reasoning system must be able to handle information on duration of events ("It will take from ten to twenty minutes to get home") and ordering of events ("Let's go to the cinema before dinner"). Quantitative temporal networks provide a convenient formalism to deal with such information be-

cause they consider time points as the variables of a problem. A *time point* may be a beginning or an ending point of some event, as well as a neutral point of time. An effective representation of quantitative temporal networks is based on constraints [11].

**Definition 1 (TCSP)** *A Temporal Constraint Satisfaction Problem (TCSP) consists of a set of variables* $\{X_1, \ldots, X_n\}$ *and a set of unary and binary constraints over pairs of such variables. The variables have continuous or discrete domains; each variable represents a time point. Each constraint is represented by a set of intervals* [1] $\{I_1, \ldots, I_k\} = \{[a_1, b_1], \ldots, [a_k, b_k]\}$. *A unary constraint* $T_i$ *restricts the domain of variable* $X_i$ *to the given set of intervals; that is, it represents the disjunction* $(a_1 \leq X_i \leq b_1) \vee \ldots \vee (a_k \leq X_i \leq b_k)$. *A binary constraint* $T_{ij}$ *over variables* $X_i$ *and* $X_j$ *constrains the permissible values for the distance* $X_j - X_i$*; it represents the disjunction* $(a_1 \leq X_j - X_i \leq b_1) \vee \ldots \vee (a_k \leq X_j - X_i \leq b_k)$. *Constraints are assumed to be given in the canonical form in which all intervals are pair-wise disjoint.*

A TCSP can be represented by a directed *constraint graph* where nodes represent variables and an edge $X_i \longrightarrow X_j$ indicates constraint $T_{ij}$ and it is labeled by the corresponding interval set. A special time point $X_0$ is introduced to represent the "beginning of the world". All times are relative to $X_0$; thus, we can treat each unary constraint $T_i$ as a binary constraint $T_{0i}$.

**Example 1** *Alice has lunch between noon and 1pm and she wants to go swimming for two hours. She can either go to the pool from 3 to 4 hours before lunch, since she must shower and drive home, or 3 to 4 hours after lunch since it is not safe to swim too soon after a meal. This scenario can be modeled as a TCSP, as shown in Figure 1. There are five variables:* $X_0$, $L_s$ *(starting time for lunch),* $L_e$ *(end time for lunch),* $S_S$ *(start swimming),* $S_e$ *(end swimming). For example, the constraint from* $X_0$ *to* $L_s$ *states that lunch must be between 12 and 1pm while, the constraint from* $L_s$ *to* $S_s$ *states that the distance between the start of the swimming activity and the start of lunch must be either between 3 and 4 hours, or between -4 and -3 hours. Similarly for the other constraints.*

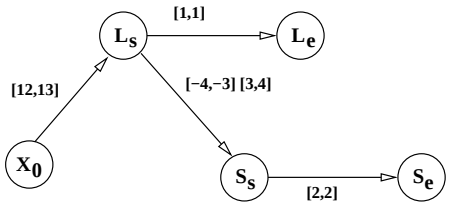


Fig. 1. A TCSP.

Given a TCSP, a tuple of values for its variables, say $\{v_1, \ldots, v_n\}$, is called a *solution* if the assignment $\{X_1 = v_1, \ldots, X_n = v_n\}$ does not violate any constraint. A TCSP is said to be *consistent* if it has a solution. Also, $v_i$ is a *feasible value* for variable $X_i$ if there exists a solution in which $X_i = v_i$. The set of all feasible values for a variable is called its *minimal domain*. A *minimal constraint* $T_{ij}$ between $X_i$ and $X_j$ is the set of values $v$ such that $v = v_j - v_i$, where $v_j$ is a feasible value for $X_j$ and $v_i$ is a feasible value for $X_i$. A TCSP is minimal if its domains and constraints are minimal. It is *decomposable* if every assignment of values to a set of its variables which does not violate the constraints among such variables can be extended to a solution.

Constraint propagation over TCSPs is defined using three binary operations on constraints: union, intersection and composition.

**Definition 2** *Let* $T = \{I_1, \ldots, I_l\}$ *and* $S = \{J_1, \ldots, J_m\}$ *be two temporal constraints defined on the pair of variables* $X_i$ *and* $X_j$*. Then:*

- *The Union of T and S, denoted* $T \cup S$*, is:* $T \cup S = \{I_1, \ldots, I_l, J_1, \ldots, J_m\}$.
- *The Intersection of T and S, denoted* $T \oplus S$*, is:* $T \oplus S = \{K_k = I_i \cap J_j | i \in \{1, \ldots, l\}, j \in \{1, \ldots, \}\}$.

**Definition 3** *Let* $T = \{I_1, \ldots, I_l\}$ *be a temporal constraint defined on variables* $X_i$ *and* $X_k$ *and* $S = \{J_1, \ldots, J_m\}$ *a temporal constraint defined on variables* $X_k$ *and* $X_j$*. Then the composition of T and S, denoted by* $T \otimes S$ *is a temporal constraint defined on* $X_i$ *and* $X_j$ *as follows:* $T \otimes S = \{K_1, \ldots, K_n\}$, $K_h = [a + c, b + d]$, $\exists I_i = [a, b]$, $J_j = [c, d]$.

Notice that the composition of two temporal constraints, say $S$ and $T$, defined respectively on

[1] For simplicity, we assume closed intervals; however the same applies to semi-open intervals.

the pairs of variables $(X_i, X_k)$ and $(X_k, X_j)$, is a constraint defined on the pair $(X_i, X_j)$ which allows only pairs of values, say $(v_i, v_j)$, for which there exists a value $v_k$, such that $(v_i, v_k)$ satisfies $S$ and $(v_k, v_j)$ satisfies $T$.

Given a TCSP, the first interesting problem is to determine its consistency. If the TCSP is consistent, we may wish to find some solutions, or to answer queries concerning the set of all solutions. All these problems are NP-hard [11].

Notions of local consistency may be interesting as well. For example, a TCSP is said to be *path consistent* iff, for each of its constraint, say $T_{ij}$, we have $T_{ij} \subseteq \oplus_{\forall k}(T_{ik} \otimes T_{kj})$.

A TCSP in which all constraints specify a single interval is called a *Simple Temporal Problem*. In such a problem, a constraint between $X_i$ and $X_j$ is represented in the *constraint graph* as an edge $X_i \longrightarrow X_j$ labeled by a single interval $[a_{ij}, b_{ij}]$ that represents the constraint $a_{ij} \leq X_j - X_i \leq b_{ij}$. An STP can also be associated with another directed weighted graph $G_d = (V, E_d)$, called the *distance graph*, which has the same set of nodes as the constraint graph but twice the number of edges: for each binary constraint over variables $X_i$ and $X_j$, the distance graph has an edge $X_i \longrightarrow X_j$ which is labeled by weight $b_{ij}$, representing the linear inequality $X_j - X_i \leq b_{ij}$, as well as an edge $X_j \longrightarrow X_i$ which is labeled by weight $-a_{ij}$, representing the linear inequality $X_i - X_j \leq -a_{ij}$.

Each path from $X_i$ to $X_j$ in the distance graph $G_d$, say through variables $X_{i_0} = X_i, X_{i_1}, X_{i_2}, \ldots, X_{i_k} = X_j$ induces the following *path constraint*: $X_j - X_i \leq \sum_{h=1}^{k} b_{i_{h-1} i_h}$. The intersection of all induced path constraints yields the inequality $X_j - X_i \leq d_{ij}$, where $d_{ij}$ is the length of the shortest path from $X_i$ to $X_j$, if such a length is defined, i.e., if there are no negative cycles in the distance graph. An STP is consistent if and only if its distance graph has no negative cycles [45,20]. This means that enforcing path consistency is sufficient for solving STPs [11]. It follows that a given STP can be effectively specified by another complete directed graph, called a *d-graph*, where each edge $X_i \longrightarrow X_j$ is labeled by the shortest path length $d_{ij}$ in the distance graph $G_d$.

In [11] it is shown that any consistent STP is backtrack-free (that is, decomposable) relative to the constraints in its *d-graph*. Moreover, the set of temporal constraints of the form $[-d_{ji}, d_{ij}]$ is the *minimal STP* corresponding to the original STP and it is possible to find one of its solutions using a backtrack-free search that simply assigns to each variable any value that satisfies the minimal network constraints compatibly with previous assignments. Two specific solutions (usually called the *latest* and the *earliest* one) are given by $S_L = \{d_{01}, \ldots, d_{0n}\}$ and $S_E = \{d_{10}, \ldots, d_{n0}\}$, which assign to each variable respectively its latest and earliest possible time [11].

The *d-graph* (and thus the *minimal network*) of an STP can be found by applying Floyd-Warshall's *All-Pairs-Shortest-Path* algorithm [13] to the distance graph with a complexity of $O(n^3)$ where $n$ is the number of variables. Since, given the *d-graph*, a solution can be found in linear time, the overall complexity of solving an STP is polynomial.

### *2.2. Soft constraints*

In the literature there are many formalizations of the concept of *soft constraints* [43,40,32]. Here we refer to the one described in [6,5], which however can be shown to generalize and express many of the others [6,4].

In a few words, a soft constraint is just a classical constraint where each instantiation of its variables has an associated element (also called a preference) from a partially ordered set. Combining constraints will then have to take into account such additional elements, and thus the formalism has also to provide suitable operations for combination ($\times$) and comparison ($+$) of tuples of preferences and constraints. This is why this formalization is based on the concept of semiring, which is just a set plus two operations.

**Definition 4 (semirings and c-semirings)** *A semiring is a tuple $\langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$ such that:*

- *$A$ is a set and $\mathbf{0}, \mathbf{1} \in A$;*
- *$+$ is commutative, associative and $\mathbf{0}$ is its unit element;*
- *$\times$ is associative, distributes over $+$, $\mathbf{1}$ is its unit element and $\mathbf{0}$ is its absorbing element.*

*A c-semiring is a semiring $\langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$ such that:*

- *$+$ is defined over possibly infinite sets of elements of $A$ in the following way:*
  - *$\forall a \in A$, $\sum(\{a\}) = a$;*

  - $\sum(\emptyset) = \mathbf{0}$ *and* $\sum(A) = \mathbf{1}$*;*
  - $\sum(\bigcup A_i, i \in S) = \sum(\{\sum(A_i), i \in S\})$ *for all sets of indexes* $S$*, that is, for al sets of subsets of* $A$ *(flattening property);*
- $\times$ *is commutative.*

Let us consider the relation $\leq_S$ over A such that $a \leq_S b$ iff $a + b = b$. Then it is possible to prove that (see [5]):

- $\leq_S$ is a partial order;
- $+$ and $\times$ are monotone on $\leq_S$;
- $\mathbf{0}$ is its minimum and $\mathbf{1}$ its maximum;
- $\langle A, \leq_S \rangle$ is a complete lattice and, for all $a, b \in A$, $a + b = lub(a, b)$.

Moreover, if $\times$ is idempotent, then $\langle A, \leq_S \rangle$ is a complete distributive lattice and $\times$ is its glb. Informally, the relation $\leq_S$ gives us a way to compare (some of the) tuples of preferences and constraints. In fact, when we have $a \leq_S b$, we will say that $b$ *is better than (or preferred to)* $a$.

**Definition 5 (constraints)** *Given a c-semiring* $S = \langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$*, a finite set* $D$ *(the domain of the variables), and an ordered set of variables* $V$*, a constraint is a pair* $\langle def, con \rangle$ *where* $con \subseteq V$ *and* $def : D^{|con|} \rightarrow A$.

Therefore, a constraint specifies a set of variables (the ones in *con*), and assigns to each tuple of values in $D$ of these variables an element of the semiring set $A$. This element can be interpreted in many ways: as a level of preference, or as a cost, or as a probability, etc. The correct way to interpret such elements determines the choice of the semiring operations.

**Definition 6 (SCSP)** *A soft constraint satisfaction problem is a pair* $\langle C, con \rangle$ *where* $con \subseteq V$ *and* $C$ *is a set of constraints over* $V$.

Note that classical CSPs are isomorphic to SCSPs where the chosen c-semiring is: $S_{CSP} = \langle \{false, true\}, \vee, \wedge, false, true \rangle$.

Fuzzy CSPs [40,42] extend the notion of classical CSPs by allowing non crisp constraints, that is, constraints which associate a preference level with each tuple of values. Such level is always between 0 and 1, where 1 represents the best value and 0 the worst one. The solution of a fuzzy CSP is then defined as the set of tuples of values (for all the variables) which have the maximal value. The way they associate a preference value with an $n$-tuple is by minimizing the preferences of all its subtuples. The motivation for such a max-min framework relies on the attempt to maximize the value of the least preferred tuple. It is easy to see that Fuzzy CSPs can be modeled in the SCSP framework by choosing the c-semiring: $S_{FCSP} = \langle [0, 1], max, min, 0, 1 \rangle$.

**Definition 7 (combination)** *Given two constraints* $c_1 = \langle def_1, con_1 \rangle$ *and* $c_2 = \langle def_2, con_2 \rangle$*, their* combination $c_1 \otimes c_2$ *is the constraint* $\langle def, con \rangle$*, where* $con = con_1 \cup con_2$ *and* $def(t) = def_1(t \downarrow^{con}_{con_1}) \times def_2(t \downarrow^{con}_{con_2})$[2].

The combination operator $\otimes$ can be straightforwardly extended also to finite sets of constraints: when applied to a finite set of constraints $C$, we will write $\bigotimes C$. In words, combining constraints means building a new constraint involving all the variables of the original ones, and which associates to each tuple of domain values for such variables a semiring element which is obtained by multiplying the elements associated by the original constraints with the appropriate subtuples.

**Definition 8 (projection)** *Given a constraint* $c = \langle def, con \rangle$ *and a subset* $I$ *of* $V$*, the* projection *of* $c$ *over* $I$*, written* $c \Downarrow_I$*, is the constraint* $\langle def', con' \rangle$ *where* $con' = con \cap I$ *and* $def'(t') = \sum_{t/t\downarrow^{con}_{I \cap con} = t'} def(t)$.

Informally, projecting means eliminating some variables. This is done by associating to each tuple over the remaining variables a semiring element which is the sum of the elements associated by the original constraint with all the extensions of this tuple over the eliminated variables.

**Definition 9 (solution constraint)** *The* solution constraint *of an SCSP problem* $P = \langle C, con \rangle$ *is the constraint* $Sol(P) = (\bigotimes C) \Downarrow_{con}$.

That is, to obtain the solution constraint of an SCSP, we combine all constraints, and then project over the variables in *con*. In this way we get the constraint over *con* which is "induced" by the entire SCSP.

[2] By $t \downarrow^X_Y$ we mean the projection of tuple $t$, which is defined over the set of variables $X$, over the set of variables $Y \subseteq X$.

**Definition 10 (solution)** *Given an SCSP problem $P$, consider $Sol(P) = \langle def, con \rangle$. A solution of $P$ is a pair $\langle t, v \rangle$ where $t$ is an assignment to all the variables in con and $def(t) = v$.*

**Definition 11 (optimal solution)** *Given an SCSP problem $P$, consider $Sol(P) = \langle def, con \rangle$. An optimal solution of $P$ is a pair $\langle t, v \rangle$ such that $t$ is an assignment to all the variables in con, $def(t) = v$, and there is no $t'$, assignment to con, such that $v <_S def(t')$.*

Therefore optimal solutions are solutions which are not dominated by any other solution in terms of preferences. The set of optimal solutions of an SCSP $P$ will be written as $Opt(P)$.

**Example 2** *Figure 2 shows an example of a fuzzy CSP. Variables are within circles, and constraints are undirected links among the variables. Each constraint is defined by associating a preference level (in this case between 0 and 1) to each assignment of its variables to values in their domains. Figure 2 shows also two solutions, one of which ($S_2$) is optimal.*

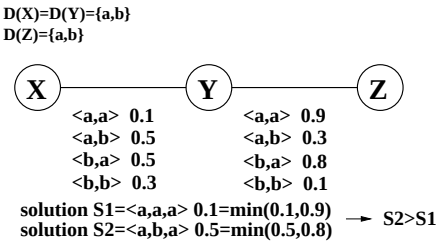


Fig. 2. A Fuzzy CSP and two of its solutions.

SCSPs can be solved by extending and adapting the techniques usually used for classical CSPs. For example, to find the best solution, we could employ a branch-and-bound search algorithm (instead of the classical backtracking). Also the so-called constraint propagation techniques, like arc-consistency [21] and path-consistency, can be generalized to SCSPs [5,6].

The detailed formal definition of *constraint propagation* (sometimes called also *local consistency*) for SCSPs can be found in [5,6]. For the purpose of this paper, what is important to say is that a *propagation rule* is a function which, given an SCSP, generates the solution constraint of a subproblem of it. It is possible to show that propagation rules are idempotent, monotone, and intensive functions (over the partial order of problems) which do not change the solution constraint. Given a set of propagation rules, a constraint propagation algorithm applies them in any order until stability. It is possible to prove that constraint propagation algorithms defined in this way have the following properties if the multiplicative operation of the semiring is idempotent: equivalence, termination, and uniqueness of the result.

Thus we can notice that the generalization of local consistency from classical CSPs to SCSPs concerns the fact that, instead of deleting values or tuples of values, obtaining local consistency in SCSPs means changing the semiring value associated with some tuples or domain elements. The change always brings these values towards the worst value of the semiring, that is, the **0**.

### *2.3. Inductive learning*

The problem of learning temporal preferences from examples of solutions ratings can be formally described as an inductive learning problem [39,23]. Inductive learning can be defined as the ability of a system to induce the correct structure of a map $t(\cdot)$ which is known only for particular inputs. More formally, defining an example as a pair $(x, t(x))$, the computational task is as follows: given a collection of examples of $t(\cdot)$, i.e., the *training set*, return a function $h(\cdot)$ that approximates $t(\cdot)$. Function $h(\cdot)$ is called a hypothesis.

A common approach to inductive learning is to evaluate the quality of a hypothesis $h$ on the training set through an *error function* [15]. An example of a popular error function, that can be used over the reals, is the sum of squares error [15]: $SSE = \frac{1}{2}\sum_{i=1}^{n}(t(x_i) - h(x_i))^2$, where $(x_i, t(x_i))$ is the $i$-th example of the training set. Other error functions that can be used to evaluate the quality of a hypothesis are the maximum absolute error and mean absolute error, respectively defined as: $E_{max} = max_{1,\ldots n}|t(x_i) - h(x_i)|$, and $E_{med} = (\sum_{i=1\ldots n}|t(x_i) - h(x_1)|)/n$. Given a starting hypothesis $h_0$, the goal of learning is to minimize the chosen error function by modifying $h_0$. This can be done by using a definition of $h$ which depends on a set of internal parameters $W$, i.e., $h \equiv h_W$, and then adjusting these parameters. This adjustment can be formulated in different

ways, depending on whether their domain is isomorphic to the reals or not. The usual way to be used over the reals, and if $h_W$ is continuous and differentiable, is to follow the negative of the *gradient* of the error function with respect to $W$. This technique is called *gradient descent* [15]. Specifically, the set of parameters $W$ is initialized to small random values at time $\tau = 0$ and updated at time $\tau + 1$ according to the following equation, known as $\Delta$-*rule*: $W(\tau+1) = W(\tau) + \Delta W(\tau)$, where $\Delta W(\tau) = -\eta \frac{\partial E}{\partial W(\tau)}$ and $\eta$ is the step size used for the gradient descent, called the *learning rate*. Learning is usually stopped when a minimum of the error function is reached. Note that, in general, there is no guarantee that the minimum found this way is a global minimum for the function to be learned.

Once the learning phase is finished, the resulting function $h$ is evaluated over a set of examples, called the *test set*, which is disjoint from the training set. The evaluation is done by computing the error, with the same options as for the error computation on the training set.

As far as how the examples are used, learning techniques can be divided in two categories: stochastic (also called online) and batch (also called offline) learning. Batch supervised learning is the classical approach in machine learning: a set of examples is obtained and used in order to learn a good approximating function (i.e. train the system), before the system is used. On the other hand, in online learning, data gathered during the normal operation of the system are used to continuously adapt the learned function. For example, in batch learning, when minimizing the sum of squares error, the sum would be computed as in $SSE = \frac{1}{2}\sum_{i=1}^{n}(t(x_i) - h(x_i))^2$, where $x_1, \ldots, x_n$ are *all* the examples of the training set. On the other hand, in stochastic learning, the weights are updated after the presentation of each training example, which may be sampled with or without repetition. This corresponds to the minimization of the instantaneous error which, in the case of sum of squares error, would be $SSE = \frac{1}{2}(t(x_i) - h(x_i))^2$ when computed on the $i$-th example. It can be shown that, for sufficiently small values of the learning rate $\eta$, stochastic gradient descent converges to the minimum of a batch error function [15].

Although batch learning seems faster for small training sets and systems, stochastic learning is faster for large training sets, it helps escaping local minima and provides a more natural approach for learning nonstationary tasks [23,2,41,47]. Moreover, stochastic methods seem more robust to errors, omissions or redundant data in the training set can be corrected or ejected during the training phase. Additionally, training data can often be generated easily and in great quantities when the system is in operation, whereas it is usually scarce and precious before. In a broad sense, stochastic learning is essential if the goal is to obtain a learning system as opposed to a merely learned one, as pointed out in [48].

The learning module we will present in Section 6 performs a stochastic gradient descent on $SSE$.

## 3. Temporal CSPs with Preferences

Although very expressive, TCSPs are able to model just *hard* temporal constraints. This means that all constraints have to be satisfied, and that the solutions of a constraint are all equally satisfying. However, in many real-life scenarios these two assumptions do not hold. In particular, sometimes some solutions are preferred with respect to others. Therefore the global problem is not to find a way to satisfy all constraints, but to find a way to satisfy them optimally, according to the preferences specified.

To address such problems we propose a framework, where each temporal constraint is associated with a preference function, which specifies the preference for each distance or duration. This framework is based on both TCSPs and semiring-based soft constraints. The result is a class of problems which we will call Temporal Constraint Satisfaction Problems with Preferences (TCSPPs).

**Definition 12 (soft temporal constraint)** *A soft temporal constraint is a 4-tuple* $\langle (X,Y), I, A, f \rangle$ *consisting of*

- *an ordered pair of variables* $(X,Y)$ *over the integers, called the scope of the constraint;*
- *a set of disjoint intervals* $I = \{[a_1, b_1], \ldots, [a_n, b_n]\}$*, where all* $a_i$*'s and* $b_i$*'s are integers, and* $a_i \leq b_i$ *for all* $i = 1, \ldots, n$*;*
- *a set of preferences* $A$*;*
- *a preference function* $f$*, where* $f : \bigcup_{i=1}^{n}[a_i, b_i] \rightarrow A$*, which is a mapping of the elements belonging to an interval of* $I$ *into preference values, taken from set* $A$*.*

*Given an assignment of the variables $X$ and $Y$, say $v_x$ and $v_y$, we say that this assignment satisfies the constraint $\langle (X,Y), I, A, f\rangle$ iff there is $[a_i, b_i] \in I$ such that $a_i \leq v_y - v_x \leq b_i$. In such a case, the preference associated with the assignment by the constraint is $f(v_y - v_x)$.*

**Definition 13 (TCSPP)** *Given a semiring $S = \langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$, a Temporal Constraint Satisfaction Problems with Preferences over $S$ is a pair $\langle V, C\rangle$, where $V$ is a set of variables and $C$ is a set of soft temporal constraints over pairs of variables in $V$ and with preferences in $A$.*

**Definition 14 (solution)** *Given a TCSPP $\langle V, C\rangle$ over a semiring $S$, a solution is an assignment to all the variables in $V$, say $t$, that satisfies all the constraints in $C$. An assignment $t$ is said to satisfy a constraint $c$ in $C$ with preference $p$ if the projection of $t$ over the pair of variables of $c$ satisfies $c$ with an associated preference equal to $p$. We will write $pref(t, c) = p$.*

Each solution has a *global preference value*, obtained by combining, via the $\times$ operator of the semiring, the preference levels at which the solution satisfies the constraints in $C$.

**Definition 15 (solution's preference)** *Given a TCSPP $\langle V, C\rangle$ over a semiring $S$ and one of its solutions $t = \langle v_1, \ldots, v_n\rangle$, its preference, denoted by val(t), is computed by $\Pi_{c\in C} pref(s, c)$, where the product here is performed by using the multiplicative operation of semiring $S$.*

The optimal solutions of a TCSPP are those solutions which are not dominated by any other solution in preference terms.

**Definition 16 (optimal solutions)** *Given a TCSPP $P = \langle V, C\rangle$ over the semiring $S$, a solution $t$ of $P$ is optimal if for every other solution $t'$ of $P$, $t' \not\geq_S t$.*

To see an instance of TCSPPs, consider TCSPPs over the semiring $S_{fuzzy} = \langle [0, 1], max, min, 0, 1\rangle$, used for fuzzy constraint solving [42]. In this case, the global preference value of a solution is the minimum of all the preference values associated with the distances selected by this solution in all constraints, and the optimal solutions are those with the maximal value. We will use this class of TCSPPs, also called fuzzy TCSPPs, extensively in this paper.

A justification of the the max-min framework adopted in fuzzy TCSPPs is to formalize the criterion of maximizing the value of the least preferred tuple. This can be interpreted as a having a conservative attitude which identifies a solution with its weakest part.

Of course other approaches to preferences can be adopted. For example, the weighted semiring can be used, in which preferences are costs, ranging in $[0, +\infty]$, and the goal is to minimize the sum of such costs. Such temporal preferences have been considered in [29].

**Example 3** *Consider again the scenario described in Example 1, where Alice can go swimming either before or after lunch. Alice might, for example, prefer to have lunch as early as possible. Moreover, if she goes swimming in the morning, she might want to go as late as possible so she can sleep longer, while, if she goes in the afternoon, she might prefer to go as early as possible so she will have more time to get ready for the evening. These preferences can be represented using a weighted TCSPP or a fuzzy TCSPP as shown, respectively, in Figure 3 part (a) and (b). It is easy to see that the assignment $\langle L_s = 12, S_s = 15\rangle$ (we omitted $X_0$ since we assume its value always to be 0) is an optimal solution, since it has the minimum cost, that is, 0, in the weighted TCSPP and maximum preference, that is, 1, in the fuzzy TCSPP.*

Notice that our framework is a generalization of TCSPs, since TCSPs are just TCSPPs over the the semiring $S_{csp} = \langle \{false, true\}, \vee, \wedge, false, true\rangle$, which allows to describe hard constraint problems [22].

As for TCSPs, a special instance of TCSPPs is characterized by a single interval in each constraint. We call such problems *Simple Temporal Problems with Preferences* (STPPs), since they generalize Simple Temporal Problems (STPs) [11]. This case is interesting because, as noted above, STPs are polynomially solvable, while general TCSPs are NP-hard, and the computational effect of adding preferences to STPs is not immediately obvious. STPPs are also expressive enough to represent many real life scenarios.

**Example 4** *Consider the Landsat 7 example given in the introduction. In Figure 4 we show an STPP*

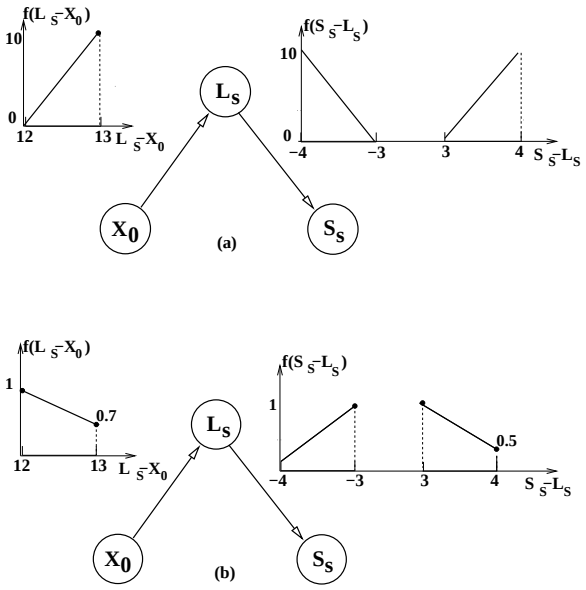


Fig. 3. The constraint graphs of a Weighted TCSPP (a) and of a Fuzzy TCSPP (b).

*that models it. There are 3 events to be scheduled: the start time ($S_s$) and ending time ($S_e$) of a slewing activity, and the start time of an image retrieval activity ($I_s$). Here the beginning of time is represented by variable $S_s$. The slewing activity in this example can take from 3 to 10 units of time, but it is preferred that it takes the shortest time possible. This is modeled by the constraint from $S_s$ to $S_e$. The image taking can start any time between 3 and 20 units of time after the slewing has been initiated. This is described by the constraint from $S_s$ to $I_s$. The third constraint, from $S_e$ to $I_s$, models the fact that it is better for the image taking to start as soon as the slewing has stopped.*

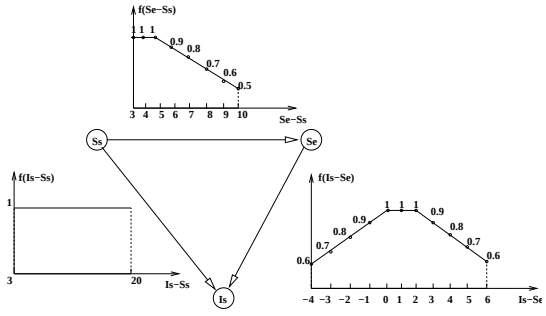


Fig. 4. The STPP for the Landsat 7 example.

### *3.1. Complexity of solving TCSPPs and STPPs*

As noted in Section 2, TCSPs are NP-hard problems. Since the addition of preference functions can only make the problem of finding the optimal solutions more complex, it is obvious that TCSPPs are NP-hard problems as well. In fact, TCSPs are just TCSPPs over the $S_{CSP}$ semiring.

We now turn our attention to the complexity of general STPPs. We recall that STPs are polynomially solvable, thus one might speculate that the same is true for STPPs. However, it is possible to show that, in general, STPPs fall into the class of NP-hard problems.

**Theorem 1 (Complexity of STPPs)** *Solving STPPs is NP-hard.*

**Proof:** We prove this result by reducing an arbitrary TCSP to an STPP. Consider a TCSP, and take any of its constraints, say $I = \{[a_1, b_1], \ldots [a_n, b_n]\}$. We will now obtain a corresponding soft temporal constraint containing just one interval (thus belonging to an STPP). The semiring that we will use for the resulting STPP is the classical one: $S_{csp} = \langle \{false, true\}, \vee, \wedge, false, true \rangle$. Thus the only allowed preference values are $false$ and $true$. Assuming that the intervals in $I$ are ordered such that $a_i \leq a_{i+1}$ for $i \in \{1, \ldots, n-1\}$, the interval of the soft constraint is just $[a_1, b_n]$. The preference function will give value $true$ to all elements belonging to an interval in $I$ and $false$ to the others. Thus we have obtained an STPP whose set of solutions with value 1 (which are the optimal solutions, since $false \leq_S true$ in the chosen semiring) coincides with the set of solutions of the given TCSP. Since finding the set of solutions of a TCSP is NP-hard, it follows that the problem of finding the set of optimal solutions to an STPP is NP-hard. □

However, in the following of the paper we will show there are classes of STPPs which are polynomially solvable: a sufficient condition is having semi-convex preference functions and a semiring with a total order of preference values and an idempotent multiplicative operation. In [12] it has been shown that the only aggregation operator on a totally ordered set that is idempotent is $min$, i.e. the multiplicative operator of the $S_{FCSP}$ semiring.

### 3.2. Path consistency for TCSPPs

Given a constraint network, it is often useful to find the corresponding minimal network in which the constraints are as explicit as possible. This task is normally performed by enforcing various levels of local consistency. For TCSPPs, in particular, we can define a notion of *path consistency* by just extending the notion of path consistency for TCSPs [11]. Given two soft constraints, and a semiring $S$, we define:

- the *intersection* of two soft constraints $T_{ij}$ and $T'_{ij}$, defined on the same pair of variables, written $T_{ij} \oplus_S T'_{ij}$, as the soft temporal constraint $T''_{ij} = \langle I_{ij} \oplus I'_{ij}, f \rangle$, where:
  - $I_{ij} \oplus I'_{ij}$ is the pairwise intersection of intervals in $I_{ij}$ and $I'_{ij}$, and
  - $f(a) = f_{ij}(a) \times_S f'_{ij}(a)$ for all $a \in I_{ij} \oplus I'_{ij}$.
- the *composition* of two soft constraints $T_{ik}$ and $T_{kj}$, with variable $X_k$ in common, written $T_{ik} \otimes_S T_{kj}$, as the soft constraint $T_{ij} = \langle I_{ik} \otimes I_{kj}, f \rangle$, defined on variables $X_i$ and $X_j$, where:
  - $a \in I_{ik} \otimes I_{kj}$ iff there exists a value $a_1 \in I_{ik}$ and $a_2 \in I_{kj}$ such that $a = a_1 + a_2$, and
  - $f(a) = \sum\{f_{ik}(a_1) \times_S f_{kj}(a_2) | a = a_1 + a_2, a_1 \in I_{ik}, a_2 \in I_{kj}\}$.

The *path-induced* constraint on variables $X_i$ and $X_j$ is $R^{path}_{ij} = \oplus_S \forall k (T_{ik} \otimes_S T_{kj})$, i.e., the result of performing $\oplus_S$ on each way of generating paths of length two from $X_i$ to $X_j$. A constraint $T_{ij}$ is *path-consistent* iff $T_{ij} \subseteq R^{path}_{ij}$, i.e., $T_{ij}$ is at least as strict as $R^{path}_{ij}$. A TCSPP is path-consistent iff all its constraints are path-consistent.

It is interesting to study under which assumptions, by applying the path consistency operation $T_{ij} := T_{ij} \oplus_S (T_{ik} \otimes_S T_{kj})$ to any constraint of a given TCSPP, the resulting TCSPP is equivalent to the given one, that is, it has the same set of solutions with the same preferences. The assumptions can be derived directly from those which are sufficient in generic SCSPs, as stated by the following theorem.

**Theorem 2** *Consider an TCSPP $P$ defined on a semiring which has an idempotent multiplicative operator. Then, applying operation $T_{ij} := T_{ij} \oplus_S (T_{ik} \otimes_S T_{kj})$ for any $k$ to any constraint $T_{ij}$ of $P$ returns an equivalent TCSPP.*

**Proof:** Consider TCSPPs $P_1$ and $P_2$ on the same set of variables $\{X_1, \ldots, X_n\}$ and defined over a semiring with an idempotent multiplicative operator. Assume also that the set of constraints of $P_2$ consists of the constraints of $P_1$ minus $\{T_{ij}\}$ plus $\{T_{ij} \oplus_S (T_{ik} \otimes_S T_{kj})\}$.

To show that $P_1$ is equivalent to $P_2$ we must show that every solution $t$ of $P_1$ with preference $val(t)$ is a solution of $P_2$ with the same preference. Notice that $P_1$ and $P_2$ differ only for the constraint defined on variables $X_i$ and $X_j$, which is $T_{ij} = \langle I, f \rangle$ in $P_1$ and $T'_{ij} = T_{ij} \oplus_S (T_{ik} \otimes_S T_{kj}) = \langle I', f' \rangle$ in $P_2$, where $T_{ik} = \langle I_{ik}, f_{ik} \rangle$ and $T_{kj} = \langle I_{kj}, f_{kj} \rangle$ are the same in $P_1$ and $P_2$.

Now, since $T'_{ij} = T_{ij} \oplus_S (T_{ik} \otimes_S T_{kj}) = \langle I', f' \rangle$, then $I' = I \oplus (I_1 \otimes I_2)$. This means that $I' \subseteq I$. Assuming $I' \subset I$, we will now show that no element $a \in I\text{-}I'$ can be a projection of a solution $s$ of $P_1$. Assume to the contrary that $s$ is a solution of $P_1$ such that $s \downarrow_{X_i, X_j} = (s_i, s_j)$ and $s_j - s_i = a \in I\text{-}I'$. Then, since $a \notin I'$ means that there is no $a_1 \in I_{ik}$ nor $a_2 \in I_{kj}$ such that $a = a_1 + a_2$, then either $s_k - s_i = a_1 \notin I_{ik}$ or $s_j - s_k = a_2 \notin I_{kj}$. But this cannot be the case, since $s$ is assumed to be a solution of $P_1$.

From the above argument we can conclude that, for any solution $t$ of $P_1$, we have $t \downarrow_{X_i, X_j} \in I'$. Thus $P_1$ and $P_2$ have the same set of solutions.

Consider solution $t$ in $P_1$. Then, as stated in the previous section, the global preference associated with $t$ in $P_1$ is $val(t) = \times\{f_{pq}(v_q - v_p) | (v_p, v_q) = t \downarrow_{X_p, X_q}\}$, which can be rewritten, highlighting the preferences obtained on constraints $T_{ij}$, $T_{ik}$ and $T_{kj}$, as: $val(t) = f(v_j - v_i) \times f(v_k - v_i) \times f(v_j - v_k) \times B$, where $B = \times\{f_{pq}(v_q - v_p) | (v_p, v_q) = t \downarrow_{X_p, X_q}, (p, q) \notin \{(i, j), (k, i), (j, k)\}\}$.

Similarly, in $P_2$ the global preference of $t$ is $val'(t) = f'(v_j - v_i) \times f(v_k - v_i) \times f(v_j - v_k) \times B$. We want to prove that $val(t) = val'(t)$. Notice that B appears in $val(t)$ and in $val'(t)$, hence we can ignore it.

By definition: $f'(v_j - v_i) = f(v_j - v_i) \times Q$ where $Q = \sum_{v'_k | (v'_k - v_i) \in I_{ik}, (v_j - v'_k) \in I_{kj}} [f(v'_k - v_i) \times f(v_j - v'_k)]$.

Now, among the possible assignments to variable $X_k$, say $v'_k$, such that $(v'_k - v_i) \in I_{ik}$ and $(v_j - v'_k) \in I_{kj}$, there is the assignment given to $X_k$ in solution $t$, say $v_k$. Thus we rewrite $f'(v_j - v_i)$ in the following way: $f'(v_j - v_i) = f(v_j - v_i) \times \{[f(v_k - v_i) \times f(v_j - v_k)] +$

$\sum_{v'_k|(v'_k-v_i)\in I_{ik},(v_j-v'_k)\in I_{kj},v'_k\neq v_k}[f(v'_k-v_i)\times f(v_j-v'_k)]\}$.

At this point, the preference of solution $t$ in $P_2$ is

$val'(t) = f(v_j-v_i)\times\{[f(v_k-v_i)\times f(v_j-v_k)]+\sum_{v'_k|(v'_k-v_i)\in I_{ik},(v_j-v'_k)\in I_{kj},v'_k\neq v_k}[f(v'_k-v_i)\times f(v_j-v'_k)]\}\times f(v_k-v_i)\times f(v_j-v_k)\times B$.

We will now show a property that holds for any two elements $a, b \in A$ of a semiring with an idempotent multiplicative operator: $a \times (a + b) = a$. In [6] it is shown that $\times$ is *intensive* with respect to the ordering of the semiring, that is, for any $a, c \in A$ we have $a \times c \leq_S a$. In particular this holds for $c = (a+b)$ and thus $a \times (a+b) \leq_S a$. On the other hand, since $a + b$ is the lub of $a$ and $b$, $(a+b) \geq_S a$, and by monotonicity of $\times$ we get that $a\times(a+b) \geq a\times a$. At this point we use the idempotency assumption on $\times$ and obtain $a \times a = a$ and, thus, $a \times (a + b) \geq_S a$. Therefore $a \times (a + b) = a$.

We now use this result in the formula describing $val'(t)$, setting $a = (f(v_k - v_i) \times f(v_j - v_k))$, and $b = \sum_{v'_k|(v'_k-v_i)\in I_{ik},(v_j-v'_k)\in I_{kj},v'_k\neq v_k}[f(v'_k - v_i) \times f(v_j-v'_k)]$. We obtain: $val'(t) = f(v_j-v_i)\times f(v_k-v_i) \times f(v_j - v_k) \times B$, which is exactly $val(t)$. □

Under the same condition, applying this operation to a set of constraints (rather than just one) returns a final TCSPP which is always the same independently of the order of application. Again, this result can be derived from the more general results that hold for SCSPs [6], as shown by following theorem.

**Theorem 3** *Consider an TCSPP $P$ defined on a semiring with an idempotent multiplicative operator. Then, applying operation $T_{ij} := T_{ij} \oplus_S (T_{ik} \otimes_S T_{kj})$ to a set of constraints of $P$ returns the same final TCSPP regardless of the order of application.*

**Proof:** Applying operation $T_{ij} := T_{ij} \oplus_S (T_{ik} \otimes_S T_{kj})$ can be seen as applying a function $f$ to an TCSPP P that returns another TCSPP $f(P) = P'$. The local consistency algorithm that applies this operation until quiescence can thus be seen as the repetitive application of function $f$. As in the case of generic SCSPs, it is possible to use a result of classical chaotic theory [9] that ensures the independence of the result on the order of application of closure operators, that is, functions that are idempotent, monotone and intensive. We will now prove that function $f$ satisfies such requirements if the multiplicative operator is idempotent. In particular, under such an assumption, function $f$ is:
*Idempotent*: In fact, applying twice operation $T_{ij} := T_{ij} \oplus_S (T_{ik} \otimes_S T_{kj})$ to constraint $T_{ij}$, when constraints $T_{ik}$ and $T_{kj}$ have not changed, gives the same result as applying it only once;
*Monotone*: Consider the ordering on SCSPs defined in [6]. Such ordering can be redefined here as follows: given TCSPP $P_1$ and TCSPP $P_2$ we say that $P_1 \leq_P P_2$ iff for every constraint $T^1_{pq} = \langle I^1_{pq}, f^1_{pq}\rangle$ in $P_1$ and corresponding constraint $T^2_{pq} = \langle I^2_{pq}, f^2_{pq}\rangle$ in $P_2$, then $I^1_{pq} \subseteq I^2_{pq}$ and $\forall w \in I^1_{pq}$ we have $f^1_{pq}(w) \leq f^2_{pq}(w)$. If now we apply operation $f$ to $P_1$ and $P_2$ we get two new TCSPPs, $f(P_1)$ and $f(P_2)$, that differ respectively from $P_1$ and $P_2$ only on constraint $T^1_{ij}$ and $T^2_{ij}$. By intensivity of $\times$, applying $T_{ij} := T_{ij} \oplus_S (T_{ik} \otimes_S T_{kj})$ to a constraint can only shrink its interval and lower the preferences corresponding to the remaining elements. Since this change depends only on the preferences on constraints $T_{ik}$ and $T_{kj}$, and by assumption we have that $T^1_{ik} \leq_P T^2_{ik}$ and $T^1_{kj} \leq_P T^2_{kj}$, by monotonicity of $\times$ the new preferences in constraint $T^1_{ij}$ in $f(P_1)$ are smaller than or equal to those on constraint $T^2_{ij}$ in problem $f(P_2)$.
*Intensive*: That is, $f(P_1) \leq_P P_1$ for any TCSPP $P_1$. In fact, as mentioned in the previous point, $f(P_1)$ differs from $P_1$ only by constraint $T_{ij}$. However, $f$ ensures that constraint $T_{ij}$ in $f(P_1)$ can only have a smaller or equal interval with respect to that in $P_1$ and that remaining elements can have preferences smaller than or equal to the ones in $P_1$. □

Thus any TCSPP can be transformed into an equivalent path-consistent TCSPP by applying the operation $T_{ij} := T_{ij} \oplus_S (T_{ik} \otimes_S T_{kj})$ to all constraints $T_{ij}$ until no change occurs on any constraint. We will call this path consistency enforcing algorithm TCSPP_PC-2 when applied to an TCSPP and STPP_PC-2 when applied to an STPP.

Figure 5 shows the TCSPP obtained by applying path consistency to the TCSPP in Figure 3.

Path consistency is proven to be polynomial for TCSPs, with complexity $O(n^3\ R^3)$, where $n$ is the number of variables and $R$ is the range of the constraints [11]. However, applying it is, in general, not sufficient to find a solution. Again, since a TCSP is a special instance of TCSPP over the $S_{CSP}$ semiring, applying path consistency is not sufficient to find an optimal solution of an TCSPP either. On the other hand, with STPPs over the same semiring, that is STPs, applying STPP_PC-2

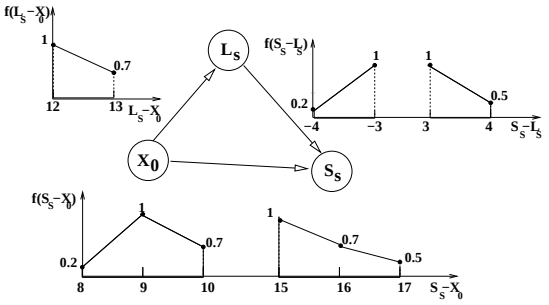


Fig. 5. The path consistent constraint graph of the TCSPP in Figure 3.

is sufficient [11]. It is easy to infer that the hardness result for STPPs, given in Section 3.1, derives either from the nature of the semiring or from the shape of the preference functions.

### *3.3. Semi-convexity and path consistency*

When the preference functions are linear, and the semiring chosen is such that its two operations maintain such linearity when applied to the initial preference function, it can be seen that the initial STPP can be written as a linear programming problem, solving which is tractable [8]. In fact, consider any given TCSPP. For any pair of variables $X$ and $Y$, take each interval for the constraint over $X$ and $Y$, say $[a, b]$, with associated linear preference function $f$. The information given by each of such intervals can be represented by the following inequalities and equation: $X - Y \leq b$, $Y - X \leq -a$ and $f_{X,Y} = c_1(X - Y) + c_2$. Then if we choose the fuzzy semiring $S_{FCSP} = \langle [0, 1], max, min, 0, 1\rangle$, the global preference value $V$ will satisfy the inequality $V \leq f_{X,Y}$ for each preference function $f_{X,Y}$ defined in the problem, and the objective function is $max(V)$. If instead we choose the semiring $\langle R, min, +, \infty, 0\rangle$, where the objective is to minimize the sum of the preference level, we have $V = f_1 + \ldots + f_n$[3] and the objective function is $min(V)$ . In both cases, the resulting set of formulas constitutes a linear programming problem.

Linear preference functions are expressive enough in many cases, but there are also several situations in which we need preference functions which are not linear. A typical example arises when we want to state that the distance between two events must be as close as possible to a single value. Then, unless this value is one of the extremes of the interval, the preference function is convex, but not linear. Another case is one in which preferred values are as close as possible to a single distance value, but in which there are some subintervals where all values have the same preference. In this case, the preference criteria define a *step function*, which is not even convex.

We consider the class of semi-convex functions which includes linear, convex, and also some step functions. More formally, a *semi-convex* function $f$ is one such that, for all $y \in \Re^+$, the set $\{x \in X$ such that $f(x) \geq y\}$ forms an interval. For example, the *close to k* criteria cannot be coded into a linear preference function, but it can be specified by a semi-convex preference function, which could be $f(x) = x$ for $x \leq k$ and $f(x) = 2k - x$ for $x > k$. Figure 6 shows some examples of semi-convex and non-semi-convex functions.

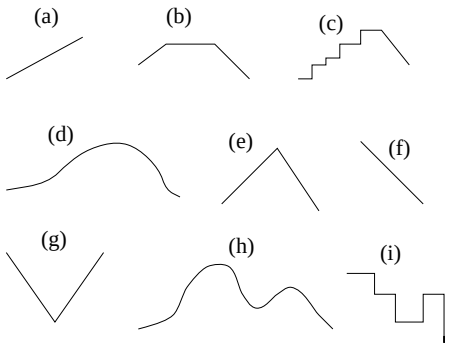


Fig. 6. Examples of semi-convex functions [(a)-(f)] and non-semi-convex functions [(g)-(i)]

Semi-convex functions are closed under the operations of intersection and composition, when certain semirings are chosen. For example, this happens with the fuzzy semiring, where the intersection performs the *min* and composition performs the *max* operation.

**Theorem 4 (closure under intersection)** *Given two semi-convex preference functions $f_1$ and $f_2$ which return values over a totally-ordered semiring $S = \langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$ with an idempotent multiplicative operator $\times$, let $f$ be defined as $f(a) = f_1(a) \times f_2(a)$. Then, $f$ is a semi-convex function as well.*

**Proof:** Given any $y$, consider the set $\{x : f(x) \geq y\}$, which by definition coincides with $= \{x :$

[3] In this context, the "+" is to be interpreted as arithmetic "+".

$f_1(x) \times f_2(x) \geq y\}$. Since $\times$ is idempotent then we also have $f_1(x) \times f_2(x) = glb(f_1(x), f_2(x))$. Moreover, since $S$ is totally ordered, we have $glb(f_1(x), f_2(x)) = min(f_1(x), f_2(x))$, that is the glb coincides with one of the two elements, that is the minimum of the two [6]. Thus, we have $\{x : f_1(x) \times f_2(x) \geq y\} = \{x : min(f_1(x), f_2(x)) \geq y\}$. Of course, $\{x : min(f_1(x), f_2(x)) \geq y\} = \{x : f_1(x) \geq y \text{ and } f_2(x) \geq y\} = \{x : x \in [a_1, b_1] \text{ and } x \in [a_2, b_2]\}$ since each of $f_1$ and $f_2$ is semi-convex. Now, by definition, $\{x : x \in [a_1, b_1] \text{ and } x \in [a_2, b_2]\} = [a_1, b_1] \cap [a_2, b_2] = [max(a_1, a_2), min(b_1, b_2)]$. We have thus proven the semi-convexity of $f$, since $\{x : f(x) \geq y\}$ is a unique interval. □

A similar result holds for the composition of semi-convex functions:

**Theorem 5 (closure under composition)** *Given a totally ordered semiring with an idempotent multiplicative operation $\times$, let $f_1$ and $f_2$ be semi-convex functions which return values over the semiring. Define $f$ as $f(a) = \sum_{b+c=a}(f_1(b) \times f_2(c))$, where $b + c$ is the sum of two integers. Then $f$ is semi-convex.*

**Proof:** From the definition of semi-convex functions, it suffices to prove that, for any given $y$, the set $S = \{x : f(x) \geq y\}$ identifies a unique interval. If $S$ is empty, then it identifies the empty interval. In the following we assume $S$ to be not empty.

By definition of $f$: $\{x : f(x) \geq y\} = \{x : \sum_{u+v=x}(f_1(u) \times f_2(v)) \geq y\}$, where $u + v$ is the sum of two integers and $\sum$ generalizes the additive operator of the semiring. In any semiring, the additive operator is the lub operator. Moreover, if the semiring has an idempotent $\times$ operator and is totally ordered, the lub of a finite set is the maximum element of the set. Thus, $\{x : \sum_{u+v=x}(f_1(u) \times f_2(v)) \geq y\} = \{x : max_{u+v=x}(f_1(u) \times f_2(v)) \geq y\}$ which coincides with $\{x : \exists u, v \mid x = u + v \text{ and } (f_1(u) \times f_2(v)) \geq y\}$.

Using the same steps as in the proof of Theorem 4, we have: $\{x : \exists u, v \mid x = u + v \text{ and } (f_1(u) \times f_2(v)) \geq y\} = \{x : \exists u, v \mid x = u + v \text{ and } min(f_1(u), f_2(v)) \geq y\}$. But this set coincides with $\{x : \exists u, v \mid x = u + v \text{ and } f_1(x) \geq y \text{ and } f_2(x) \geq y\} = \{x : \exists u, v \mid x = u + v \text{ and } \exists a_1, b_1, a_2, b_2 \mid u \in [a_1, b_1] \text{ and } v \in [a_2, b_2]\}$ since each of $f_1$ and $f_2$ is semi-convex. This last set coincides with $\{x : x \in [a_1 + a_2, b_1 + b_2]\}$. We have thus shown the semi-convexity of a function obtained by combining semi-convex preference functions. □

These results imply that applying the STPP_PC-2 algorithm to an STPP with only semi-convex preference functions, and whose underlying semiring contains a multiplicative operation that is idempotent and a totally ordered preference set, will result in an STPP whose induced soft constraints have only semi-convex preference functions.

Consider now an STPP with semi-convex preference functions and defined on a semiring with an idempotent multiplicative operator and a totally ordered preference set, like $S_{FCSP} = \langle [0, 1]\ max, min, 0, 1\rangle$. In the following theorem we will prove that, if such an STPP is also path consistent, then all its preference functions must have the same maximum preference value.

**Theorem 6** *Consider a path consistent STPP $P$ with semi-convex functions defined on a totally-ordered semiring with an idempotent multiplicative operator. Then all its preference functions have the same maximum preference.*

**Proof:** All the preference functions of an STPP have the same maximum iff any pair of soft temporal constraints of the STPP is such that their preference function have the same maximum. Notice that the theorem trivially holds if there are only two variables. In fact, in this case, path consistency is reduced to performing the intersection on all the constraints defined on the two variables and a single constraint is obtained.

For the following of the proof we will denote with $T_{IJ} = \langle I_{IJ}, f_{IJ}\rangle$ the soft temporal constraint defined on variables $I$ and $J$.

Let us now assume that there is a pair of constraints, say $T_{AB}$ and $T_{CD}$, such that $\forall h \in I_{AB}, f_{AB}(h) \leq M$, and $\exists r \in I_{AB}, f_{AB}(r) = M$, and $\forall g \in I_{CD}, f_{CD}(g) \leq m$, and $\exists s \in I_{CD}, f_{CD}(s) = m$ and $M > m$.

Let us first note that any soft temporal constraint defined on the pair of variables $(I, J)$ induces a constraint on the pair $(J, I)$, such that $max_{h \in I_{IJ}} f_{IJ}(h) = max_{g \in I_{JI}}\ f_{JI}(g)$. In fact, assume that $I_{IJ} = [l, u]$. This constraint is satisfied by all pairs of assignments to $I$ and $J$, say $v_I$ and $v_J$, such that $l \leq v_J - v_I \leq u$. These inequalities hold iff $-u \leq v_I - v_J \leq -l$ hold. Thus the interval of constraint $T_{JI}$ must be $[-u, -l]$. Since each assignment $v_J$ to $J$ and $v_I$ to $I$ which identifies

element $h \in [l,u]$ with a preference $f_{IJ}(x) = p$ identifies element $-x \in [-u,-l]$, it must be that $f_{IJ}(x) = f_{JI}(-x)$. Thus $f_{IJ}$ and $f_{JI}$ have the same maximum on the respective intervals.

Consider now the triangle of constraints $T_{AC}$, $T_{AD}$ and $T_{DC}$. To do this, we assume that, for any three variables, there are constraints connecting every pair of them. This is without loss of generality because we assume to work with a path-consistent STPP.

Given any element $a$ of $I_{AC}$, since the STPP is path consistent it must be that: $f_{AC}(a) \leq \sum_{a_1+a_2=a}(f_{AD}(a_1) \times f_{DC}(a_2))$, where $a_1$ is in $I_{AD}$ and $a_2$ is in $I_{DC}$.

Let us denote with $max(f_{IJ})$ the maximum of the preference function of the constraint defined on variables $I$ and $J$ on interval $I_{IJ}$. Then, since $\times$ and $+$ are monotone, the following must hold: $f_{AC}(a) \leq \sum_{a_1+a_2=a}((f_{AD}(a_1) \times f_{DC}(a_2))) \leq max(f_{AD}) \times max(f_{DC})$. Notice that this must hold for every element $a$ of $I_{AC}$, thus also for those with maximum preference, thus: $max(f_{AC}) \leq max(f_{AD}) \times max(f_{DC})$. Now, since $\times$ is intensive, we have $max(f_{AD}) \times max(f_{DC}) \leq max(f_{DC})$. Therefore, $max(f_{AC}) \leq max(f_{DC}) = m$.

Similarly, if we consider the triangle of constraints $T_{CB}$, $T_{CD}$, and $T_{DB}$ we can conclude that $max(f_{CB}) \leq max(f_{CD}) = m$.

We now consider the triangle of constraints $T_{AB}$, $T_{AC}$ and $T_{CB}$. Since the STPP is path consistent, then $max(f_{AB}) \leq max(f_{AC})$ and $max(f_{AB}) \leq max(f_{CB})$. But this implies that $max(f_{AB}) \leq m$, which contradicts the hypothesis that $max(f_{AB}) = M > m$. □

Consider an STPP $P$ that satisfies the hypothesis of Theorem 6 having, hence, the same maximum preference $M$ on every preference function. Consider the STP $P'$ obtained by P taking the subintervals of elements mapped on each constraint into preference $M$. In the following theorem we will show that, if P is a path consistent STPP, then $P'$ is a path consistent STP.

**Theorem 7** *Consider a path consistent STPP $P$ with semi-convex functions defined on a totally ordered semiring with idempotent multiplicative operator. Consider now the STP $P'$ obtained from $P$ by considering on each constraint only the subinterval mapped by the preference function into its maximal value for that interval. Then, $P'$ is a path consistent STP.*

**Proof:** An STP is path consistent iff all its constraints are path consistent, that is, for every constraint $T_{AB}$, we have $T_{AB} \subseteq \oplus_K(T_{AK} \otimes T_{KB})$, where $K$ varies over the set of variables of the STP [11]. Assume that $P'$ is not path consistent. Then there must be at least a hard temporal constraint of $P'$, say $T_{AB}$, defined on variables $A$ and $B$, such that there is at least a variable $C$, with $C \neq A$ and $C \neq B$, such that $T_{AB} \not\subset T_{AC} \otimes T_{CB}$. Let $[l_1,u_1]$ be the interval of constraint $T_{AB}$, $[l_2,u_2]$ the interval of constraint $T_{AC}$ and $[l_3,u_3]$ the interval of constraint $T_{CB}$. The interval of constraint $T_{AC} \otimes T_{CB}$ is, by definition, $[l_2+l_3, u_2+u_3]$. Since we are assuming that $T_{AB} \not\subset T_{AC} \otimes T_{CB}$, it must be that $l_1 < l_2 + l_3$, or $u_2 + u_3 < u_1$, or both.

Let us first assume that $l_1 < l_2 + l_3$ holds. Now, since $l_1$ is an element of an interval of $P'$, by Theorem 6 it must be that $f_{AB}(l_1) = M$, where $f_{AB}$ is the preference function of the constraint defined on $A$ and $B$ in STPP P and $M$ is the highest preference in all the constraints of $P$. Since $P$ is path consistent, then there must be at least a pair of elements, say $a_1$ and $a_2$, such that $a_1 \in I_{AC}$ (where $I_{AC}$ is the interval of the constraint defined on $A$ and $C$ in $P$), $a_2 \in I_{CB}$ (where $I_{CB}$ is the interval of the constraint defined on $C$ and $B$ in $P$), $l_1 = a_1 + a_2$, and $f_{AC}(a_1) \times f_{CB}(a_2)) = M$. Since $\times$ is idempotent and $M$ is the maximum preference on any constraint of $P$, it must be that $f_{AC}(a_1) = M$ and $f_{CB}(a_2) = M$. Thus, $a_1 \in [l_2,u_2]$ and $a_2 \in [l_3,u_3]$. Therefore, it must be that $l_2 + l_3 \leq a_1 + a_2 \leq u_2 + u_3$. But this is in contradiction with the fact that $l_1 = a_1 + a_2$ and $l_1 < l_2 + l_3$.

Similarly for the case in which $u_2 + u_3 < u_1$. □

Notice the the above theorem, and the fact that an STP is path consistent iff it is consistent (i.e., it has at least a solution) [11] allows us to conclude that if an STPP is path consistent, then there is at least a solution with preference $M$. In the theorem that follows we will claim that $M$ is also the highest preference assigned to any solution of the STPP.

**Theorem 8** *Consider a path consistent STPP $P$ with semi-convex functions defined on a totally-ordered semiring with idempotent multiplicative operator and with maximum preference $M$ on each function. Consider now the STP $P'$ obtained from $P$ by considering on each constraint only the subinterval mapped by the preference function into $M$. Then, the set of optimal solutions of $P$ is the set of solutions of $P'$.*

**Proof:** Let us call $Opt(P)$ the set of optimal solutions of $P$, and assume they all have preference $opt$. Let us also call $Sol(P')$ the set of solutions of $P'$. By Theorem 7, since $P$ is path consistent and hence globally consistent, we have $Sol(P') \neq \emptyset$.

First we show that $Opt(P) \subseteq Sol(P')$. Assume that there is an optimal solution $s \in Opt(P)$ which is not a solution of $P'$. Since $s$ is not a solution of $P'$, there must be at least a hard constraint of $P'$, say $I'_{ij}$ on variables $X_i$ and $X_j$, which is violated by $s$. This means that the values $v_i$ and $v_j$ which are assigned to variables $X_i$ and $X_j$ by $s$ are such that $v_j - v_i \notin I'_{ij}$. We can deduce from how $P'$ is defined, that $v_j - v_i$ cannot be mapped into the optimal preference in the corresponding soft constraint in $P$, $T_{ij} = \langle I_{ij}, f_{ij} \rangle$. This implies that the global preference assigned to $s$, say $f(s)$, is such that $f(s) \leq f_{ij}(v_j - v_i) < opt$. Hence $s \notin Opt(P)$ which contradicts out hypothesis.

We now show $Sol(P') \subseteq Opt(P)$. Take any solution $t$ of $P'$. Since all the intervals of $P'$ are subintervals of those of $P$, $t$ is a solution of $P$ as well. In fact, $t$ assigns to all variables values that belong to intervals in $P'$ and are hence mapped into the optimal preference in $P$. This allows us to conclude that $t \in Opt(P)$. □

## 4. Solving Simple Temporal Problems with Preferences

In this section we will describe two solvers for STPPs. Both find an STP such that all its solutions are optimal solutions of the STPP given in input. Both solvers require the tractability assumptions on the shape of the preference functions and on the underlying semiring, to hold. In particular, we will consider semi-convex preferences based on the fuzzy semiring.

### *4.1. Path-solver: a solver based on path consistency*

The theoretical results of the previous section can be translated in practice as follows: to find an optimal solution for an STPP, we can first apply path-consistency and then use a search procedure to find a solution without the need to backtrack. Summarizing, we have shown that: (1) Semi-convex functions are closed w.r.t. path-consistency: if we start from an STPP $P$ with semi-convex functions, and we apply path-consistency, we get a new STPP $P'$ with semi-convex functions (by Theorems 4 and 5); the only difference in the two problems is that the new one can have smaller intervals and worse preference values in the preference functions.
(2) After applying path-consistency, all preference functions in $P'$ have the same best preference level (by Theorem 6).
(3) Consider the STP obtained from the STPP $P'$ by taking, for each constraint, the sub-interval corresponding to the best preference level; then, the solutions of such an STP coincide with the best solutions of the original $P$ (and also of $P'$). Therefore, finding a solution of this STP means finding an optimal solution of $P$.

| **Pseudocode for path-solver** |
|---|
| 1. **input** STPP P; |
| 2. STPP P'=STPP_PC-2(P); |
| 3. **if** P' inconsistent **then return** $\emptyset$; |
| 4. STP P"=REDUCE_TO_BEST(P'); |
| 5. **return** EARLIEST_BEST(P"). |

Fig. 7. Path-solver.

Our first solving module, which we call path-solver, relies on these results. In fact, the solver takes as input an STPP with semi-convex preference functions, and returns an optimal solution of the given problem, working as follows and as shown in Figure 7: First, in line 2 path-consistency is applied to the given problem, by function STPP_PC-2, producing a new problem $P'$. Then, in line 4 an STP corresponding to $P'$ is constructed, applying REDUCE_TO_BEST to $P'$, by taking the subintervals corresponding to the best preference level and forgetting about the preference functions. Finally, a backtrack-free search is performed to find a solution of the STP, specifically the earliest one is returned by function EARLIEST_BEST in line 5. All these steps are polynomial, so the overall complexity of solving an STPP with the above assumptions is polynomial, as stated by the following theorem.

**Theorem 9** *Given an STPP with semi-convex preference functions, defined on a semiring with an idempotent multiplicative operator, with $n$ variables, maximum size of intervals $r$, and $l$ distinct totally ordered preference levels, the complexity of path-solver is $O(n^3rl)$.*

**Proof:** Let us follow the steps performed by path-solver to solve an STPP. First we apply STPP_PC-2. This algorithm must consider $n^2$ triangles. For each of them, in the worst case, only one of the preference values assigned to the $r$ different elements is moved down of a single preference level. This means that we can have $O(rl)$ steps for each triangle. After each triangle is visited, at most $n$ new triangles are added to the queue. If we assume that each step which updates $T_{ij}$ needs constant time, we can conclude that the complexity of STPP_PC-2 is $O(n^3rl)$.

After STPP_PC-2 has finished, the optimal solution must be found. Path-solver achieves this by finding a solution of the STPP $P''$, obtained from $P'$ by considering only intervals mapped into maximum preference on each constraint. In [11] it has been shown that this can be done without backtracking in $n$ steps (see also Section 2.1). At each step, a value is assigned to a new variable while not violating the constraints that relate this variable with the previously assigned variables. Assume there are $d$ possible values left in the domain of the variable, then the compatibility of each value must be checked on at most $n-1$ constraints. Since for each variable the cost of finding a consistent assignment is $O(rd)$, the total cost of finding a complete solution of $P'$ is $O(n^2d)$. The complexity of this phase is clearly dominated by that of STPP_PC-2. This allows us to conclude that the total complexity of finding an optimal solution of $P$ is $O(n^3rl)$. □

In the above proof we have assumed that each step of STPP_PC-2 $T_{ij} := T_{ij} \oplus_S (T_{ik} \otimes_S T_{kj})$ is performed in constant time. If we count the arithmetic operations performed during this step, we can see that there are $O(r^3)$ of them. In fact, each constraint of the triangle has at most $r$ elements in the interval, and, for each element of the interval in constraint $T_{ij}$, the preference of $O(r^2)$ decompositions must be checked. With this measure, the new complexity of finding an optimal solution is $O(r^4n^3l)$.

**Example 5** *In Figure 8 we show the effect of applying path-solver on the example in Figure 4. As it can be seen, the interval on the constraint on variables* Ss *and* Is *has been reduced from [3,20] to [3,14] and some preferences on all the constraints have been lowered. It is easy to see that the optimal preference of the STPP is 1 and the minimal STP containing all optimal solutions restricts the duration of the slewing to interval [4,5], the interleaving time between the slewing start and the image start to [3,5] and the interleaving time between the slewing stop and the image start to [0,1].*

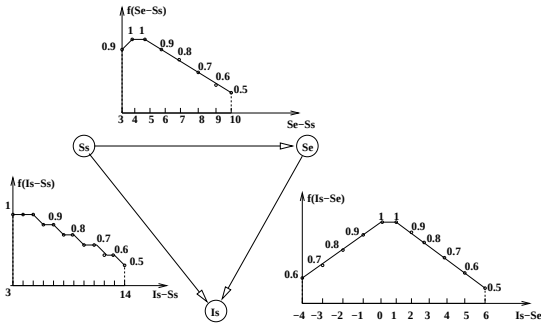


Fig. 8. The STPP representing the Landsat 7 scenario depicted in Figure 4 after applying STPP_PC-2.

An approach similar to that in path-solver is proposed in [49], in which *Fuzzy Temporal Constraint Networks* are defined. In such networks, fuzzy intervals are used to model a *possibility* distribution (see [50]) over intervals. Although our work and the work in [49] have two completely different goals, since ours is to model temporal flexibility in terms of preferences, while in [49] they consider vagueness of temporal relations, there are many points in common. In fact, despite the different semantics, the framework is very close to ours since they give results for Fuzzy Simple Temporal Problems with unimodal possibility distribution, where unimodal is a synonym of semi-convexity. Thus, they combine the possibilities with *min* and compare them with *max*, as we do with preferences. To test whether a fuzzy temporal network is consistent, they propose a generalization of the classical path consistency algorithm (PC-1)[24], while we extend the more efficient version for TCSPs (PC-2) proposed in [11]. They compute the minimal network and detect inconsistency if an interval becomes empty. However, they do not prove that after path consistency the possibilities are all lowered to the same maximum and that the STP obtained considering only intervals mapped into that maximum is also minimal. This is what allows us to find an optimal solution in polynomial time. Another difference is that our approach is more general, since TCSPPs can model other classes of preferences than just fuzzy ones.

We will show in Section 5.4 that, also because of its generality, the path consistency approach is substantially slower in practice than the chopping solver that we will describe in the next section.

### *4.2. Chop-solver*

Given an STPP and an underlying semiring with set of preference values $A$, let $y \in A$ and $\langle I, f \rangle$ be a soft constraint defined on variables $X_i$ and $X_j$ in the STPP, where $f$ is semi-convex. Consider the interval defined by $\{x : x \in I \text{ and } f(x) \geq y\}$. Since $f$ is semi-convex, this set defines a single interval for any choice of $y$. Let this interval define a constraint on the same pair $X_i$ and $X_j$. Performing this transformation on each soft constraint in the original STPP results in an STP, which we refer to as $STP_y$. Notice that this procedure is related to what are know as $\alpha$-cuts in fuzzy set theory [19].

Not every choice of $y$ will yield an STP that is solvable. Let *opt* be the highest preference value (in the ordering induced by the semiring) such that $STP_{opt}$ has a solution. We will now prove that the solutions of $STP_{opt}$ are the optimal solutions of the given STPP.

**Theorem 10** *Consider any STPP $P$ with semi-convex preference functions over a totally ordered semiring with $\times$ idempotent. Take opt as the highest $y$ such that $STP_y$ has a solution. Then the solutions of $STP_{opt}$ are the optimal solutions of $P$.*

**Proof:** First we prove that every solution of $STP_{opt}$ is an optimal solution of $P$. Take any solution of $STP_{opt}$, say $t$. This instantiation $t$, in $P$, has global preference $val(t) = f_1(t_1) \times \ldots \times f_n(t_n)$, where any $t_k$ is the distance $v_j - v_i$ for an assignment to variables $X_i$ and $X_j$, that is, $(v_i, v_j) = t \downarrow_{X_i,X_j}$, and $f_i$ is the preference function associated with the soft constraint $\langle I_i, f_i \rangle$, with $v_j - v_i \in I_i$. Now assume that $t$ is not optimal in $P$. That is, there is another instantiation $t'$ such that $val(t') > val(t)$. Since $val(t') = f_1(t'_1) \times \ldots \times f_n(t'_n)$, by monotonicity of $\times$, $val(t') > val(t)$ implies that there is at least one $i$ such that $f_i(t'_i) > f_i(t_i)$. Let us take the smallest of the $f_i(t'_i)$, call it $w'$, and construct $STP_{w'}$. It is easy to see that $STP_{w'}$ has at least $t'$ as a solution, therefore *opt* is not the highest value of $y$ such that $STP_y$ has a solution, contradicting our assumption.

Next we prove that every optimal solution of $P$ is a solution of $STP_{opt}$. Take any $t$ optimal for $P$, and assume it is not a solution of $STP_{opt}$. This means that, for some constraint $i$, $f(t_i) < opt$. Therefore if we compute $val(t)$ in $P$, we have that $val(t) < opt$. Then take any solution $t'$ of $STP_{opt}$. Since $\times$ is idempotent, we have that $val(t') \leq opt$, thus $t$ was not optimal for $P$ as initially assumed. □

This result suggests a way to find an optimal solution of an STPP with semi-convex functions: we can iteratively choose a $w \in A$ and then solve $STP_w$, until $STP_{opt}$ is found. Both phases can be performed in polynomial time, and hence the entire process is polynomial. The second solver for STPPs that we have implemented [35], and that we will call 'chop-solver', follows this approach.

The solver finds an optimal solution of the STPP identifying first $STP_{opt}$ and then returning its earliest. Preference level *opt* is found by performing a binary search in $[0, 1]$. In Figure 9 we show the pseudocode for this solver.

**Pseudocode for Chop-solver**

```
1. input STPP P;
2. input precision;
3. integer n ← 0;
4. real lb← 0, ub← 1, y← 0, STP STP_0 ← Chop(P,y);
5. if(Consistency(STP_0))
6.    y← 1, STP STP_1 ← Chop(P,y);
7.   if (STP_0 = STP_1) return solution of STP_0;
8.    if (Consistency(STP_1)) return solution;
9.    else
10.     y← 0.5, n ← n+1;
11.     while (n<=precision)
12.      if(Consistency(Chop(P,y)))
13.        lb ← y, y ← y+(ub-lb)/2, n← n+1;
14.      else
15.        ub← y, y←y-(ub-lb)/2, n←n+1;
16.     end of while;
17.     return solution;
18. else exit.
```

Fig. 9. Chop-solver.

Three variables are maintained during the search: *ub* containing the lowest level at which an inconsistent STP was found, *lb* containing the highest level at which a consistent STP was found, and $y$ for the current level at which we need to perform the "chopping". It is easy to see that *ub* and *lb* are the upper and lower bound of the portion of the [0,1] interval to which we can restrict our search.

The algorithm takes in input an STPP $P$ and the desired precision (lines 1 and 2). Then a counter, which will be used to store the current

level of precision achieved in each step of the algorithm, is initialized to 0 (line 3). In line 4 variable $ub$ is initialized to 1, variable $lb$ to 0 and the search for the optimal preference level starts with $y = 0$. Function *Chop* applied to an STPP $P$ and a preference level $w$ returns the STP $STP_w$ obtained by chopping $P$ at $w$. $STP_0$ is the STP we would obtain by considering all the soft constraints as hard constraints, that is, with preference function equal to 1 on the elements of the interval and to 0 everywhere else. If it is found not to be consistent the algorithm stops, informing the user that the whole problem is inconsistent (line 5). Otherwise $STP_1$, obtained chopping the STPP at level 1, is considered (line 6). If $STP_1$ is the same (in the sense it has exactly the same constraints) as $STP_0$ then a solution of $STP_0$ is returned (line 7). This happens when the algorithm is given in input a consistent hard constraint problem. If, instead, $STP_0$ and $STP_1$ are different but $STP_1$ is consistent, then, since 1 is the highest preference value in the preference set, a solution of $STP_1$ is returned (line 8). Otherwise the search proceeds updating the three values, $ub$, $lb$ and $y$, depending on the outcome of the consistency test (lines 9-17). Function *Consistency* receives, as input, an STP, and it checks if it is consistent. Such test is performed using *All-Pairs-Shortest-Path* (see Section 2.1). If the number of decimal digits given in input is reached, then Chop-solver returns a solution (either the earliest or the latest solution, respectively corresponding to the assignments $x_i = -d_{i0}$ and $x_i = d_{0i}$). If instead the last STP considered is inconsistent a solution of the last consistent one is returned.

For example, Chop-solver when applied to the triangular STPP shown in Figure 4 will stop the search when it reaches preference level 1 (line 8) and finds that the STP obtained chopping the STPP at 1 is consistent The corresponding minimal network has the following constraints: [4,5] on the constraint on *Ss* and *Se*, [3,5] on the constraint on *Ss* and *Is*, and [0,1] on the constraint *Is* and *Se*.

The following theorem shows that chop-solver finds an optimal solution of an STPP respecting the tractability assumptions in polynomial time.

**Theorem 11** *The complexity of chop-solver is* $O(k \times n^3)$ *where* $n$ *is the number of variables and* $k$ *is the number of decimal digits of the preferences.*

**Proof:** At each preference level considered within the binary search, the chopping of the preference functions is linear in the number of constraints and hence takes $O(n^2)$ where $n$ is the number of variables. At each step, Floyd-Warshall's algorithm is applied to solve $STP_y$ with complexity $O(n^3)$ (see Section 2.1). Thus, we can conclude that the overall complexity of Chop-solver is $O(k \times n^3)$ where $k$ is the maximum number of steps allowed in the binary search, i.e. the number of decimal digits corresponding to the precision of the preference representation. □

## 5. Experimental Results for the Solvers

In this section we will describe our random generator of STPPs, and we will give experimental results on solvers, comparing their performance on classes of randomly generated STPPs.

### *5.1. Generating random STPPs*

Both STPP solvers described in the previous sections have been tested on randomly-generated problems. The random generator we have developed focuses on a particular subclass of semi-convex preference functions: convex quadratic functions of the form $ax^2 + bx + c$, with $a \leq 0$. The choice has been suggested both by the expressiveness of such a class of functions and also by the ease of expressing functions in this class by just three parameters: $a$, $b$, and $c$. Moreover, it generates fuzzy STPPs, thus preference values are between 0 and 1.

An STPP is generated according to the value of the following parameters:

- number $n$ of variables;
- range $r$ for the initial solution: to assure that the generated problem has at least one solution, we first generate such a solution, by giving to each variable a random value within the range $[0, r]$;
- density $d$: percentage of constraints that are not universal (that is, with the maximum range and preference 1 for all interval values);
- maximum expansion $max$ from initial solution: for each constraint, the bounds of its interval are set by using a random value between 0 and $max$, to be added to and subtracted from the timepoint identified for this constraint by the initial solution;

– perturbation of preference functions ($pa$, $pb$, $pc$): each preference function, being of the form $ax^2 + bx + c$, can be described by three values ($a$, $b$, and $c$); to set such values for each constraint, the generator starts from a standard quadratic function which has value 0 at the end points of the interval and value 0.5 at the middlepoint, and then modifies its three parameters according to the percentages $pa$, $pb$, and $pc$. In more detail, for each parameter $q = a, b, c$, a value in the interval $[-|q * pq/100|, |q * pq/100|]$ is selected and added to the corresponding parameter. If the the new parabola does not satisfy the requirements it is discarded and a new perturbation is generated and tested.

For example, if we call the generator with the parameters $\langle 10, 20, 30, 40, 20, 25, 30\rangle$, it will generate a fuzzy STPP with 10 variables. Moreover, the initial solution will be chosen by giving to each variable a value between 0 and 20. Among all the constraints, 70% of them will be universal, while the other 30% will be specified as follows: for each constraint, consider the timepoint specified by the initial solution, say $t$; then the interval will be $[t-t_1, t+t_2]$, where $t_1$ and $t_2$ are random numbers between 0 and 40. Finally, the preference function in each constraint is specified by taking the default one and changing its three parameters $a$, $b$, and $c$, by, respectively, 20%, 25%, and 30%.

To compare our generator with the usual one for classical CSPs, we notice that the maximum expansion ($max$) for the constraint intervals roughly corresponds to the tightness. However, we do not have the same tightness for all constraints, because we just set an upper bound to the number of values allowed in a constraint. Also, we do not explicitly set the domain of the variables, but we just set the constraints. This is in line with other temporal CSP generators, like the one in [44].

### *5.2. Experimental results for path-solver*

In Figures 10 and 11 we show some results for finding an optimal solution for STPPs generated by our generator using path-solver, which has been developed in C++ and tested on a Pentium III at 1GHz.

It must be noted that the implementation of path-solver uses a point-wise representation of the constraint intervals and of their preference functions. This makes the solver more general, since it can represent any kind of preference functions, even those that don't have an analytical representation via a small set of parameters. In fact, even starting from convex quadratic functions, which need just three parameters, the first solving phase, which applies path-consistency, can yield new preference functions which are not representable via three parameters only. For example, we could get semi-convex functions which are generic step functions, and thus not representable by giving new values to the initial three parameters.

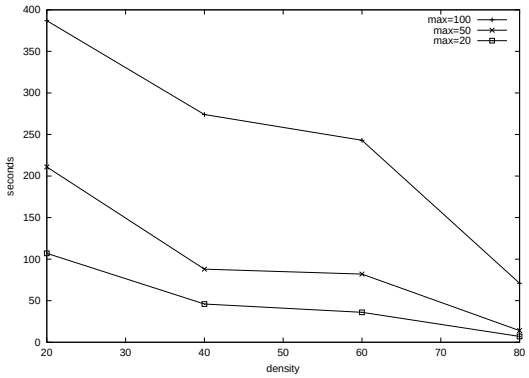


Fig. 10. Time needed by path-solver to find an optimal solution (in seconds), as a function of the density. The other parameters are: $n$=20, $r$=100, $pa$=20, $pb$=20, and $pc$=30. Mean on 30 examples.

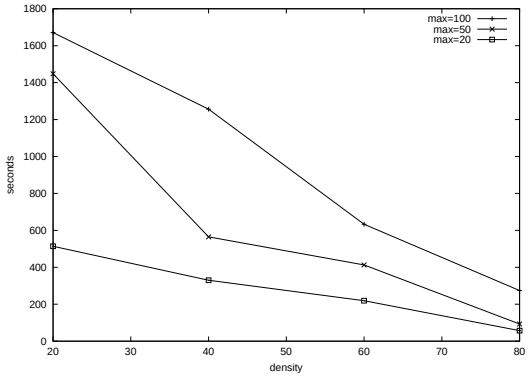


Fig. 11. Time needed by path-solver to find an optimal solution (in seconds), as a function of density ($d$). The other parameters are: $n$=50, $r$=100, $pa$=20, $pb$=20, and $pc$=30. Mean on 30 examples.

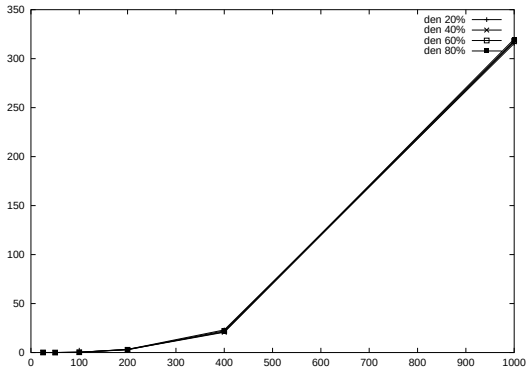


Fig. 12. Time, in seconds, required by chop-solver to solve an STPP. The number of variables (x-axis) and the density vary. The values of the other parameters are: r=100000 max=50000, pa=5, pb=5, and pc=5. Mean on 10 examples.

Among the input parameters of the random generator, we have chosen to vary the number of variables $n$ ($n = 20$ in Figure 10 and $n = 50$ in Figure 11), the density (from 20% to 80% on the $x$-axis), and the maximum range of intervals (20, 50 and 100 elements).

We have chosen to leave parameter $r$, representing the range of the first solution, fixed to 100, since it clearly does not effect the speed of execution. In fact, changing $r$ and maintaining all other parameters with the same value, it corresponds to a "translation" of the problem. The other parameters which are fixed in our experiments are the distortion parameters of the parabolas: $pa$, $pb$ and $pc$. Such values are related to the shape of the parabolas and, indirectly, determine the maximum number of distinct preference levels. The percentages we have chosen, that is, $pa = 20$, $pb = 20$ and $pc = 30$, have proven to allow a wide variety of preference levels between 0 and 1.

The curves depicted in Figures 10 and 11 represent the mean on 30 generated problems. By looking at these experimental results, we can see that:

- The time needed to solve the problem is directly proportional to the number of variables. This is an obvious consequence of the complexity of the algorithm which is polynomial in $n$;
- The time is also directly proportional to the size of the intervals ($max$). This is due to the point-wise representation of constraints used by the algorithm;
- It is instead inverse proportional to the density of the constraints. The reason for this is that the constraints generated have smaller intervals than the universal constraints.
- Moreover, the smaller the maximum range $max$, the weaker the impact on performance of varying the density. This is another clear consequence of the relevance of the size of intervals w. r. t. performance.

As it can be seen, this solver is very slow. For example, it takes 200 seconds to solve a problem with 20 variables, maximum size 50 of the intervals, and density 20%. The main reason is that it uses a pointwise representation of the constraint intervals and of their preference functions.

### *5.3. Experimental results for chop-solver*

Figure 12 shows some experimental results for chop-solver. We have used basically the same random generator used to test the solver described in Section 3, although it has been slightly modified since the two solvers use two different representations of a constraint. In fact, while path-solver has a point-wise representation of the constraints and their preference functions, chop-solver represents each preference function by the values of its three parameters. More precisely, each constraint is represented by only two integers for the left and right ends of the interval and three doubles as parameters of the preference function.

We have tested chop-solver by varying the number of variables, from a minimum of 25 up to a maximum of 1000, and the density from 20% to 80%, keeping precision fixed to 8 digits.

From Figure 12 we can see that chop-solver is not sensitive to variations in the density. The tested implementation of Chop-solver uses Floyd-Warshall's algorithm which is the best choice for problems with high density, as the ones considered here. It should be noticed, however, that for problems with a sparse graph, as can be the case in some scheduling problems, Bellman-Ford's algorithm should be used instead.

Chop-solver is sensitive to the number of variables since it yields an increase of the number of constraints on which the intersection procedure must be performed.

The choice of maintaining a fixed maximum enlargement of the intervals, that can be interpreted

as a fixed tightness, is justified by the continuous representation of the constraint this solver uses. Increasing *max* affects this representation of a constraint only by making these values bigger. However, this change does not affect the complexity of any of the operations performed by chop-solver.

### *5.4. Path-solver vs. chop-solver*

In Table 1, 2 and 3 we can see a comparison between chop-solver and path-solver.

| | D=20 | D=40 | D=60 | D=80 |
|---|---|---|---|---|
| path-solver | 515.95 | 235.57 | 170.18 | 113.58 |
| chop-solver | 0.01 | 0.01 | 0.02 | 0.02 |

Table 1

Time, in seconds, needed by path-solver and chop-solver to solve STPPs with $n = 30$, $r = 100$, $max = 50$, $pa = 10$, $pb = 10$, $pc = 5$, and varying density $D$. Results are mean on 30 examples.

| | D=20 | D=40 | D=60 | D=80 |
|---|---|---|---|---|
| path-solver | 1019.44 | 516.24 | 356.71 | 320.28 |
| chop-solver | 0.03 | 0.03 | 0.03 | 0.03 |

Table 2

Time, in seconds, needed by path-solver and chop-solver to solve STPPs with $n = 40$, $r = 100$, $max = 50$, $pa = 10$, $pb = 10$, $pc = 5$, and varying density $D$. Results are mean on 30 examples.

| | D=20 | D=40 | D=60 | D=80 |
|---|---|---|---|---|
| path-solver | 2077.59 | 1101.43 | 720.79 | 569.47 |
| chop-solver | 0.05 | 0.05 | 0.06 | 0.07 |

Table 3

Time, in seconds, used by path-solver and chop-solver to solve STPPs with $n = 50$, $r = 100$, $max = 50$, $pa = 10$, $pb = 10$, $pc = 5$, and varying density $D$. Results are mean on 30 examples.

It appears clear that chop-solver is much faster than path-solver. It is also true that, in a sense, it's also more precise since it can find an optimal solution with a higher precision. It must be kept in mind, though, that path-solver is more general. In fact, the point-wise representation of the constraints, to be blamed for its poor performance, allows one to use any kind of semi-convex function, e.g. step functions, that cannot be easily compactly parametrized. We recall that such a point-wise representation is required in order to apply path consistency. It is also true that, in general, time is dealt with as a discretized quantity, which means that, once the measuring unit that is most significant for the involved events is fixed, the problem can be automatically cast in the point-wise representation. Moreover, even wanting to extend the types of parametrized functions in the continuous representation for chop-solver, we must remember that the system deriving from intersecting the constant at chopping level and the function must be solvable in order to find the possible intersections. However, the continuous representation used by chop-solver is, undoubtedly, more natural because it reflects the most obvious idea of temporal preferences that is, an interval plus a preference function over it.

## 6. Inductive Learning of Local Temporal Preferences

It is not always easy to specify the preference functions in each temporal constraint in a way that the real-life problem at hand is faithfully modeled. This happens because sometimes it is easier to specify only global preference functions, to be associated with entire solutions, rather than local preference functions to be attached to the constraints. For this reason, and since the whole TCSPP machinery is based on local preference functions, we propose here a method to induce local preferences from global ones.

We now describe our methodology to learn preferences in STPPs from examples of solution ratings. As we did for the two solvers described above, also in this part of the paper we focus on classes of STPPs which are tractable, rather than general TCSPPs.

### *6.1. A general strategy for learning soft temporal constraints*

Learning in our context can be used to find suitable preference functions to be associated with the constraints of a given STP. More precisely, let $P = (V, C)$ be an STP where $V$ is a set of variables and $C$ is a set of distance constraints of the form $l \leq X - Y \leq u$. Let also $t$ be a function $t : S \rightarrow A$,

where $S$ is the set of solutions of $P$ and $A$ is a set of values indicating the "quality" of the solution.

The learning task consists of transforming the STP into an STPP, with each constraint $c_{i,j} \in C$ replaced by a soft constraint $\langle c_{i,j}, f_{i,j} \rangle$, where $f_{i,j}$ is the local preference function for $c_{i,j}$.

The examples to be used in the learning task consist of pairs $(s, t(s))$, where $s$ is a solution to the original STP and $t(s)$ is its "score". In the following, we use $P$ to denote an STP and $P'$ to denote a corresponding STPP. Te goal of the learning procedure is to define $P'$ in a way such that the preference associated with a solution $s$ in $P'$, that is $val_{P'}(s)$, approximates well the one assigned by the target function $t$.

Let $P$ and $f$ be as defined above, and suppose a set of examples $TR = \{(s_1, t(s_1)) \ , \ldots , (s_m, t(s_m))\}$ is given. To infer the local preferences, we must also be given the following: a semiring whose element set $A$ contains the values $t(s_i)$ in the examples; a distance function over such a semiring. Given all of the above, the goal of the learning procedure is to define a corresponding STPP $P'$ over the same semiring such that $P$ and $P'$ have the same set of variables, variable domains and interval constraints, and for each $t$ such that $(s, t(s))$ is an example, $dist(val_{P'}(s), t(s)) < \epsilon$, where $\epsilon > 0$ and small.

Once the semiring is decided, the only free parameters that can be arbitrarily chosen are the values to be associated with each distance. For each constraint, $c_{ij} = I_{ij} = [l_{ij}, u_{ij}]$ in an STP $P$, the idea is to associate, in $P'$, a free parameter $w_d$, where $d = X_j - X_i$, to each element $d$ in $I_{ij}$. This parameter will represent the preference over that specific distance. With the other distances, those outside $I_{ij}$, we associate the constant $\mathbf{0}$, (the lowest value of the semiring (w.r.t. $\leq_S$)).

If $I_{ij}$ contains many time points, we would need a great number of parameters. To avoid this problem, we can restrict the class of preference functions to a subset which can be described via a small number of parameters. For example, linear functions just need two parameters $a$ and $b$, since they can be expressed as $a \cdot (X_j - X_i) + b$. In general, we will have a function which depends on a set of parameters $W$, thus we will denote it as $f_W : (W \times I_{ij}) \rightarrow A$.

The value assigned to each solution $s$ in $P'$ is

$$val_{P'}(s) = \prod_{c_{ij} \in P'} [\sum_{d \in I_{ij}} \text{check}(d, s, i, j) \times f_W(d)]$$

where $\prod$ generalizes the $\times$ operation, $\sum$ generalizes $+$, $I_{ij}$ is the set of intervals associated with constraint $c_{ij}$, and $\text{check}(d, s, i, j) = \mathbf{1}$ if $d = s \downarrow_{X_j} - s \downarrow_{X_i}$ and $\mathbf{0}$ otherwise. Note that, for each constraint $c_{ij}$, there is exactly one distance $d$ such that $\text{check}(d, s, i, j) = \mathbf{1}$, namely $d = t \downarrow_{X_j} - t \downarrow_{X_i}$. Thus, $val_{P'}(s) = \prod_{c_{ij} \in P'} f_W(s \downarrow_{X_j} - s \downarrow_{X_i})$. The values of the free parameters in $W$ may be obtained via a minimization of the error function, which will be defined according to the distance function of the semiring.

Suppose we are given a class of STPs to be "softened" via the learning approach defined above. As we know, STPs are tractable [11]. However, in general we may end up with STPPs which are not tractable, since there is no guarantee that our learning approach returns preference functions which are semi-convex. For this reason the technique described in [33,3] cannot be used directly and a new learning algorithm is introduced, in order to guarantee the tractability of the STPP produced in output. Moreover, one needs to choose a semiring which preserves semi-convexity. To force the learning framework to produce semi-convex functions, we can specialize it for a specific class of functions with this property. For example, we could choose convex quadratic functions (parabolas) of the form $f(d) = a \cdot d^2 + b \cdot d + c$, where $a \leq 0$. In this case we just have three parameters to consider: $W = \{a, b, c\}$.

Of course, by choosing a specific class of semi-convex functions $f_W$, not all local preference shapes will be representable. Even if one chooses $f_W$ to cover any semi-convex function, there is no guarantee that the desired solution ratings will be matched. In general, the learning process will return a soft temporal problem which will approximate the desired rating as much as possible, considering the chosen class of functions and the error function. But we will gain tractability for the solution process of the resulting problems: starting from the class of STPs, via the learning approach we will obtain a class of STPPs which is tractable as well.

Our learning module uses the gradient descent approach, as described in Section 2.3. Since the problem we tackle is numerical, no symbolic approach is suitable. Moreover, the function we want to learn involves the min function, which is not differentiable, and thus it difficult to minimize the error function. To solve this problem, we use a

differentiable approximation of the min function. The approximation we use is a good one because it has a parameter which allows us to control the degree of approximation. However, it introduces a non-linearity. In this case, gradient descent is the simplest technique to use.

### *6.2. The learning module*

We will now describe in detail how the gradient descent technique can be implemented when all the preference functions are convex quadratics and the underlying semiring is the fuzzy semiring $S_{FCSP} = \{[0,1], max, min, 1, 0\}$. If the problem we are considering has $n$ variables, only $\nu = (\frac{(n-1)n}{2}) - n$ of the $n^2$ constraints need to be considered, since the remaining are just reciprocal. Given solution $s$, let $s_i$ be its projection on the $i$-th constraint. The global preference assigned to $s$ is then:

$$h(s) = val_{P'}(s) = \prod_{i=1}^{\nu} f_i(s_i), \tag{1}$$

where $f_i$ is the preference function of the $i$-th constraint. Since the multiplicative operator of the fuzzy semiring is $min$ and the function $s$ are all semi-convex quadratics, we can rewrite the above as:

$$h(s) = min_{i=1,\cdots,\nu}\{a_i s_i^2 + b_i s_i + c_i\}. \tag{2}$$

We can now substitute what we have obtained in the sum of squares error:

$$SSE = \frac{1}{2}\sum_{s \in Tr}(t(s) - min_{i=1,\cdots,\nu}\{a_i s_i^2 + b_i s_i + c_i\})^2.$$

The learning module performs a *stochastic* gradient descent (see Section 2), which means that the error and its update are computed after considering each example of the training set. We must therefore ignore the summation on all the examples, obtaining:

$$SSE(s) = \frac{1}{2}(t(s) - [min_{i=1,\cdots,\nu}\{a_i s_i^2 + b_i s_i + c_i\}])^2.$$

For the sake of notations we will, from now on, indicate $SSE(s)$ simply with $E$. Our aim is to derive the error function with respect to the parameters of the problem, $\{a_1, b_1, c_1, \cdots, a_\nu, b_\nu, c_\nu\}$, and to modify the parameters in a way such that the error decreases. In other words, following the opposite direction at which the gradient points. In order to be able to compute $\frac{\partial E}{\partial a_i}$, $\frac{\partial E}{\partial b_i}$, and $\frac{\partial E}{\partial c_i}$, we must replace $min$ with a continuous approximation [14]:

$$min_{i=1,\cdots,\nu}(\alpha_i) \simeq min_\nu^\beta(\alpha_i) = -\frac{1}{\beta}\ln(\frac{1}{\nu}\sum_{i=1}^{\nu} \mathrm{e}^{-\beta\alpha_i})$$

where parameter $\beta \geq 0$ determines the goodness of the approximation. Recalling that in our context $\alpha_i = a_i s_i^2 + b_i s_i + c_i$, we obtain the final formulation of the error:

$$E = \frac{1}{2}(t(s) + \frac{1}{\beta}\ln(\frac{1}{\nu}\sum_{i=1}^{\nu} \mathrm{e}^{-\beta(a_i s_i^2 + b_i s_i + c_i)})])^2.$$

From this expression we can obtain the updated values for the parameters of the parabolas on all the constraints following the $\Delta$-*rule*:

$$\tilde{a_i} = a_i + \Delta a_i \text{ with } \Delta a_i = -\eta\frac{\partial E}{\partial a_i}$$

$$\tilde{b_i} = a_i + \Delta b_i \text{ with } \Delta b_i = -\eta\frac{\partial E}{\partial b_i}$$

$$\tilde{c_i} = c_i + \Delta c_i \text{ with } \Delta c_i = -\eta\frac{\partial E}{\partial c_i}.$$

In these formulas, parameter $\eta$ is the learning rate which controls the magnitude of the update. More precisely, $\frac{\partial E}{\partial a_i}$ is:

$$\frac{\partial E}{\partial a_i} = (t(s) - h(s))(-(\frac{\partial h(s)}{\partial a_i}))$$

in which only $h(s)$ depends on $\{a_1, b_1, c_1, \cdots, a_\nu, b_\nu, c_\nu\}$, while $t(s)$ is the target preference of $s$ in the training set and it is totally independent from the above parameters. Consider $h(s)$ rewritten using the continuous approximation of $min$, that is:

$$h(s) = -\frac{1}{\beta}\ln(\frac{1}{\nu}\sum_{j=1}^{\nu} \mathrm{e}^{-\beta(a_j s_j^2 + b_j s_j + c_j)})$$

and the expression for $\frac{\partial h(s)}{\partial a_i}$, which can be obtained through a few easy steps:

$$\frac{\partial h(s)}{\partial a_i} = \frac{1}{(\sum_{j=1}^{\nu} \mathrm{e}^{-\beta(a_j s_j^2 + b_j s_j + c_j)})}(s_i^2 \mathrm{e}^{-\beta(a_i s_i^2 + b_i s_i + c_i)}).$$

In what follows let us set

$$Q_1 = \frac{1}{(\sum_{j=1}^{\nu} \mathrm{e}^{-\beta(a_j s_j^2 + b_j s_j + c_j)})}$$

and

$$Q_2 = e^{-\beta(a_i s_i^2 + b_i s_i + c_i)}$$

.

$$\frac{\partial E}{\partial a_i} = (t(s) - h(s))Q_1(-s_i^2 Q_2).$$

Similarly, we can compute the derivative of the error w.r.t. $b_i$ and $c_i$:

$$\frac{\partial E}{\partial b_i} = (t(s) - h(s))Q_1(-s_i Q_2)$$

$$\frac{\partial E}{\partial c_i} = (t(s) - h(s))Q_1(-Q_2)$$

Finally, we can write the complete expression for the update:

$$\tilde{a_i} = a_i - \eta[(t(s) - h(s))Q_1(-s_i^2 Q_2)] \quad (3)$$

$$\tilde{b_i} = b_i - \eta[(t(s) - h(s))Q_1(-s_i Q_2)] \quad (4)$$

$$\tilde{c_i} = c_i - \eta[(t(s) - h(s))Q_1(-Q_2)] \quad (5)$$

For each solution $s$, following the formulas illustrated above, we compute the error $E$ and the update $\tilde{a_i}$, $\tilde{b_i}$ and $\tilde{c_i}$. In changing the parameters of the preference functions, two conditions must hold at all time: (1) all of the functions must be semi-convex and (2) the image of the function (that is, $f_i(I)$ if $I$ is the interval of the temporal constraint), must be contained in $[0, 1]$, due to the fuzzy semiring.

A parabola is semi-convex iff $a \leq 0$. The algorithm maintains this property for all the parabolas, simply replacing an updated parameter $a$ that is strictly positive with 0.

This method is, in some way, similar to projective methods [30], often used in practice. Let us consider in detail how it affects the update of a parameter $a_i$: $\tilde{a_i} = a_i - \eta[(t(s) - h(s))Q_1(-s_i^2 Q_2)]$ Suppose this is the first update of $a_i$ to a strictly positive $\tilde{a_i}$. We can then assume that $a_i < 0$. In order for $\tilde{a_i}$ to be positive it must be that: $-\eta[(t(s) - h(s))Q_1(-s_i^2 Q_2)] \geq 0$ or equivalently, $\eta[(t(s) - h(s))Q_1(s_i^2 Q_2)] \geq 0$.
This can only happen if all the factors are positive. But this implies that the error $(t(s) - h(s))$ must be positive, which means that the hypothesized preference is smaller than the true one, that is, $t(s) \geq h(s)$. Moreover, the difference between $t(s)$ and $h(s)$ must be big enough to allow for the second addend to be bigger in modulo than $|a_i|$. Forcing $\tilde{a_i} = 0$ means behaving as if the error would be smaller, that is, only sufficient to make the second addend equal to $a_i$.

As for the second condition, we introduce the notion of *truncated*, or $fuzzy$, parabola. For any semi-convex parabola, the corresponding fuzzy parabola is the semi-convex function that coincides with the original function on all the elements that are mapped into values between 0 and 1, and that maps all the elements originally mapped to negative values to 0, and all the elements mapped to values greater than 1 to 1.

Fuzzy parabolas fit well into the fuzzy schema but, on the other side, their first derivatives are discontinuous. For the implementation of the algorithm, three different possibilities have been considered: (a) perform learning using only fuzzy parabolas and their derivatives; (b) use fuzzy parabolas for the update keeping the derivatives of the continuous parabolas; (c) use the parabolas and let them free to evolve during the learning phase even out of the [0,1] interval, and consider fuzzy parabolas only after the completion of the learning phase. Hypothesis (a) has been discarded due to the difficulty of dealing with the discontinuity of the first derivatives of fuzzy parabolas. Hypothesis (b) has been implemented but a detailed study of the consequences of the distortion of the gradient descent it creates has convinced us to drop it. The final learning module has been implemented according to hypothesis (c).

We compute the following errors:

- the error used by the learning module: sum of squares error with parabolas, $E$, and with fuzzy parabolas, $E^f$.
- The maximum absolute error and mean absolute error with both parabolas, $E_{max}$ and $E_{med}$, and fuzzy parabolas, $E^f_{max}$ ed $E^f_{med}$. The maximum error tells us how much the hypothesized preference value of a solution is far from the target in the worse case. It can thus be defined in general as $max_s|t(s) - h(s)|$, where, in $E_{max}$, $h(s)$ is computed using parabolas, while in $E^f_{max}$ it is computed using fuzzy parabolas. In the case of parabolas this allows us to see immediately if functions outside the $[0, 1]$ interval are learned, while in the case of fuzzy parabolas it gives a strict

upper bound to the worst performance on any solution. The mean absolute error, instead, gives an overall measure of the quality of the learned functions. It gives the average over all solutions of the distance of the hypothesized preference and the true one. In general it can be defined as $\frac{\sum_{s\in T}|t(s)-h(s)|}{|T|}$, where $|T|$ is the number of solutions considered (e.g., those in the training set or those in the test set). We have chosen the improvement in terms of the mean absolute error as the criterion to stop the learning phase.

Once the learning phase has ended, the STPP returned by the algorithm is with fuzzy parabolas. It is safe to do so since the errors computed with parabolas dominate those with fuzzy parabolas as stated in the following theorem.

**Theorem 12** *Consider an STPP P with preference functions described as convex parabolas with positive values, and the corresponding STPP F with the same parabolas as P, where all values greater than* 1 *have been lowered to* 1 *(i.e., with fuzzy parabolas). Then:*

1. *given any solution of P (and F′) s and corresponding target $t(s)$, with $0 \leq t(s) \leq 1$, the sum of squares error of s in P is greater than or equal to that in F, that is, $SSE^P(s) \geq SSE^F(s)$;*
2. *given a set of solutions of P with their corresponding target, say $T = \{(s, t(s))|$ s solution of $P, 0 \leq t(s) \leq 1\}$, the maximum error on T computed in P, that is, $E^P_{max} = max_{s\in T}|t(s) - val_P(s)|$, is greater than or equal to that computed in F, that is, $E^P_{max} \geq E^F_{max}$;*
3. *given a set of solutions of P with their corresponding target, say $T = \{(s, t(s))|$ s solution of $P, 0 \leq t(s) \leq 1\}$, then the mean error on T computed in P, $E^P_{med} = \frac{\sum_{s\in T}|t(s)-val_P(s)|}{|T|}$ is greater than or equal to that computed in F, that is, $E^P_{med} \geq E^F_{med}$.*

**Proof:** (1) Consider any constraint $C$ in P and let $p(x) = ax^2 + bx + c$ be its preference function and $fp(x)$ the corresponding fuzzy parabola in F. Consider solution $s$ and its projection on $C$, $s_C$. Now $p(s_C) = fp(s_C)$ if $p(s_C) \leq 1$, otherwise $p(s_C) > 1$ and $fp(s_C) = 1$ (by definition of fuzzy parabola). By definition we have that: $SSE^P(s) = \frac{1}{2}(t(s)-[min_{i=1,\cdots,\nu}\{p_i(s_i)\}])^2$, while: $SSE^F(s) = \frac{1}{2}(t(s) - [min_{i=1,\cdots,\nu}\{fp_i(s_i)\}])^2$.

The only case in which $val_P(s) = min_{i=1,\cdots,\nu}\{p_i(s_i)\}$ and $val_F(s)$ which is $min_{i=1,\cdots,\nu}\ \{fp_i(s_i)\}$ differ is if $p_i(s_i) > 1$, $\forall i$. In such case we will have $val_P(s) > 1$ and $val_F(s) = 1$. This means that $|t(s) - val_P(s)| > |t(s) - val_F(s)|$, since $0 \leq t(s) \leq 1$, which implies that $(t(s) - val_P(s))^2 > (t(s) - val_F(s))^2$. This leads us to $SSE^P(s) \geq SSE^F(s)$.

(2) By what is stated above, we have that for any $s \in T$ either $|t(s) - val_P(s)| = |t(s) - val_F(s)|$ or $|t(s) - val_P(s)| > |t(s) - val_F(s)$. This means that $E^P_{max} = max_{s\in T}|t(s) - val_P(s)| \geq E^F_{max} = max_{s\in T}|t(s) - val_F(s)|$.

(3) Again, since for any $s \in T$ either $|t(s) - val_P(s)| = |t(s) - val_F(s)|$ or $|t(s) - val_P(s)| > |t(s) - val_F(s)|$, when we sum the distances on all the solutions it must be that $\sum_{s\in T}|t(s) - val_P(s)| \geq \sum_{s\in T}|t(s) - val_F(s)|$. □

Notice that, while in the derivatives the approximation of *min* is used, everywhere else when $h(s)$ must be computed the actual *min* function is used. The reason for this is that it's better to use approximations only when it is necessary.

The pseudocode of the learning module is shown in Figure 13. In line 1 the algorithm takes as input STPP P. P contains all the intervals, in some sense the "hard" constraints of the problems. Its preference functions are initialized according to some criterion that decides the $a$, $b$ and $c$ values for each of them. For example, the preference functions might be set to the constant function $f(x) = 1$ by setting $a = 0$, $b = 0$, and $c = 1$. Once the initialization is completed, the algorithm must start scanning the solutions and corresponding global preferences contained in the training set (lines 2,3).

For each solution the algorithm updates the parameters of all the preference functions following the $\Delta$-*rule*, that is, the procedure described earlier in this paragraph, lines(4-6). Notice that function UPDATE$(P, E, \eta, \beta)$ updates the parameters on all the parabolas given the current problem P, the sum of squares error $E$, the learning rate $\eta$ and the *min* approximation parameter $\beta$ according to Equations 3, 4, and 5.

Once all the examples in the training set have been examined the errors $E$, $E_{max}$, $E_{med}$, $E^f$, $E^f_{max}$, and $E^f_{med}$ are computed. The percentage of improvements of $E^f_{med}$ w. r. t. its last value, $PLI^f_{E_{med}}$, is computed as well (line 7). This value is

**Algorithm STPP_LEARNING_MODULE**
1. **initialize** STPP P ;
2. **do**
3. **for each** $s \in Tr$;
4. **compute** $E$;
5. UPDATE$(P, E, \eta, \beta)$;
6. **end of for;**
7. **compute** $E$, $E_{max}$, $E_{med}$, $E^f$, $E^f_{max}$, $E^f_{med}$, $PLI^f_{E_{med}}$, on training set;
8. **if** $(PLI^f_{E_{med}} < Thres)$ stopcount++;
9. **if** $(PLI^f_{E_{med}} \geq Thres)$ stopcount=0;
10. **while**(stopcount<maxit);
11. **compute** $E^{ts}_c$, $E^{ts}_{max}$, $E^{ts}_{med}$ on test set;
12. **output**;

Fig. 13. Pseudocode of STPP_LEARNING_MODULE.

used as the stopping criterion for the learning module. In detail, if the values of $E^f_{med}$ at iterations $j$, $j+1$, and $j+2$ are respectively, $e_j$, $e_{j+1}$ and $e_{j+2}$ then $PLI^f_{E_{med}}$ at iteration $j+2$ is $(e_{j+2} - e_{j+1})$, the value of threshold $Thres$ that appears in lines 8 and 9, is $\alpha \times (e_{j+1} - e_j)$, where $\alpha$ is a parameter chosen in $[0.5, 1[$. If for a certain number of iterations, $maxit$, $PLI^f_{E_{med}}$ fails to exceed threshold $Thres$, the algorithm stops (line 10).

At this point the parabolas are reduced to fuzzy parabolas and the testing phase starts (line 11). A set of examples, i.e. pairs (solution, target), none of which appears in the training set, namely the test set, is evaluated using the STPP produced by the module. By comparing the preferences assigned to the solutions by the STPP with their targets, errors $E^{ts}$, $E^{ts}_{max}$, and $E^{ts}_{med}$ are computed. They are, obviously, the sum of squares error, the maximum absolute error and the mean absolute error on the test set. In line 12 the output is given. More precisely, the output will consist of the learned STPP and some data collected during the learning and the testing phase. Although the data to be collected can be specified at experimental setup time, usually it will contain all the errors computed at each iteration and the errors computed in the testing phase.

Using machine learning techniques to induce local preferences from global ones is not novel. In [7] the author considers the problem of eliciting from a user a utility function which represents her preferences over a set of alternatives. Each alternative, or outcome, is decomposed into a set of attributes and the utility function is assumed to be a linear combination of the utilities over the attributes.

Comparing this approach to ours, the outcomes in [7] are solutions of the STPP and the attributes are constraints. Given an outcome, the global utility of that outcome is a weighted sum of the local utilities of the attributes while, for us, is the minimum preference on any constraint. The data from which they perform the updating is a set of ordered pairs of outcomes, while we have a set of solutions with a rankings attached to them. Their learning technique is based on attaching weights to linear constraints and altering the weights until the LP problem is satisfiable. In our context, the global preference is not linear, and the learning technique is different since it is based on gradient descent and the update of the parameters is not based on the satisfaction of a set of constraints.

Another related work is presented in [16], which is in the context of search engines. In particular, it proposes to use a learning technique that will change the ranking function of the engine according to the clickthrough data retrieved from the user. In other words, an agent asks a search engine a query $q$, and the engine propose a list of links which is produced by its ranking function. Such links are presented in order of importance (or relevance). If however the order in which the user clicks on such links is not the one proposed by the engine, then such information (called clickthrough data) can be used to change the engine's ranking function.

This work is related to ours since the concern is again to modify the modeling of the user's preferences by looking at his preferences over solutions (links in this case). The context is however different, since we deal with temporal problems and we do not consider the possibility of a repeated interaction between the system and the user. Another work which is based on our learning approach but considers an online environment and a repeated interaction is [34].

### *6.3. Experimental results*

The learning module has been tested on some randomly generated problems: every test involves the generation of an STPP via the generator described in Section 5.1, and then the random selection of some examples of solutions and their rating.

Among the input parameters of the generator, we have chosen to maintain fixed the range of the

first solution. In fact, the learning module is not sensitive to the position in the time-line of the timepoints around which the problem is generated. This is true even if a translation of the STPP does change the parameters of the parabolas on the constraints. The learning is however not sensitive to the magnitude of such parameters. This is true for learning systems in general, since the regularity and shape of functions have a much bigger impact on the difficulty of learning than the size of the parameters, assuming an appropriated learning rate $\eta$. In all the experiments we have kept such a domain at $r = 40$, so all the problems have a solution within the first 40 units of time.

We have also maintained fixed the distortion parameters to the values $pa = 10$, $pb = 10$ and $pc = 5$. These values have been chosen to allow for a wide range of different preference values.

We have instead decided to vary the number of variables $n$. In particular, Table 4 shows results for problems with $n = 25$, Table 5 for problems with $n = 20$ and Table 6 for problems with $n = 15$.

We have also considered three different sizes of the intervals: $max = 20$, $max = 30$ and $max = 40$. According to the maximum size of the intervals, we have also changed the size of the training and the test set. In particular, we have given 500 examples to the training set and 500 to the test set for problems with $max = 15$; while 600 examples have been used in each set for problems with $max = 20$ and 700 examples for problems with $max = 25$. Notice that in all cases the size of the training and test set was less than 1% of the total number of solutions.

Another parameter which we have varied is the density of non-universal constraints. In particular, we have considered densities $D = 40\%$, $D = 60\%$, and $D = 80\%$. These are fairly high values, but we have chosen to consider highly constrained networks since they have more non-trivial preference functions and are, hence, more interesting from a learning point of view.

As for the parameters of the learning module, we have set them as follows: $\beta = 8$ and $\eta = 10^{-7}$. The learning process starts with all the preference functions set to the constant $y = 1$. The stop criterion has been set to 100 consecutive steps with an improvement of the average absolute error on the training set, $E_{med}$, smaller than the 70% of the previous one.

In Tables 4, 5 and 6 we show results on $E^{ts}_{med}$, that is, the mean absolute error on the test set. The first value is the mean on 30 examples, while the values between brackets are respectively the minimum and the maximum mean absolute error obtained.

| max | D=40 | D=60 | D=80 |
|---|---|---|---|
| 20 | 0.017 (0.013,0.022) | 0.007 (0.006,0.008) | 0.0077 (0.0075,0.0081) |
| 30 | 0.022 (0.017,0.025) | 0.013 (0.01,0.017) | 0.015 (0.005,0.028) |
| 40 | 0.016 (0.011,0.019) | 0.012 (0.012,0.013) | 0.0071 (0.006,0.0079) |

Table 4

$E^{ts}_{med}$ on STPPs with $n = 25$, $r = 40$, $pa = 10$, $pb = 10$, and $pc = 5$.

| max | D=40 | D=60 | D=80 |
|---|---|---|---|
| 20 | 0.032 (0.022,0.043) | 0.012 (0.01,0.016) | 0.005 (0.004,0.006) |
| 30 | 0.032 (0.021,0.040) | 0.018 (0.016,0.021) | 0.0074 (0.0073,0.0077) |
| 40 | 0.033 (0.05,0.021) | 0.023 (0.025,0.022) | 0.016 (0.011,0.021) |

Table 5

$E^{ts}_{med}$ on STPPs with $n = 20$, $r = 40$, $pa = 10$, $pb = 10$, and $pc = 5$.

| max | D=40 | D=60 | D=80 |
|---|---|---|---|
| 20 | 0.018 (0.01,0.028) | 0.009 (0.007,0.012) | 0.009 (0.007,0.012) |
| 30 | 0.021 (0.018,0.027) | 0.014 (0.011,0.017) | 0.016 (0.010,0.021) |
| 40 | 0.024 (0.023,0.026) | 0.019 (0.019,0.021) | 0.0086 (0.007,0.01) |

Table 6

$E^{ts}_{med}$ on STPPs with $n = 15$, $r = 40$, $pa = 10$, $pb = 10$, and $pc = 5$.

From the experimental results we can see that the error is reasonable, since it ranges from 0.004 to 0.05, while the preference values are in $[0, 1]$. Moreover, it seems not to be affected by the number of variables. This means that there can be larger problems that are less difficult to learn. Finally, there is a loose connection between the density and the error. A reason for this could be that

a greater density ensures a wider variety of preference values, which in turn translates into a training set with many different preference values, helping the module in the inference process.

We conclude by giving some information on the number of iterations and the time used by the algorithm. All the tests have been performed on a machine with a Pentium III 1GHz processor and 512 Mb of RAM. The minimum number of iterations has been 357 while the maximum number has been 3812. The shortest time used has been of 2 minutes and 31 seconds while the longest 8 minutes and 18 seconds. Note that these results were obtained on over 4 different problems since the time needed for a single iteration is not constant.

## 7. Conclusions and Future Work

Summarizing, the main results of this paper are:

- the definition of a framework, based on temporal constraints [11] and soft constraints [5,6], capable of modeling temporal preferences;
- theoretical complexity results for solving temporal problems with preferences as well as the identification of a tractable sub-class;
- the design and implementation of two solvers for the tractable class (that is, for simple temporal problems with fuzzy semi-convex functions): the first one based on a local consistency rule known as path consistency [10], and the second one based on a decomposition approach using $\alpha$-cuts [19]; the one defined by fuzzy semi-convex temporal preferences);
- the design and implementation of a learning module capable of eliciting local preferences from global ones;
- a complete experimental scenario to assess the value of the solvers and the learning module on randomly generated temporal problems with fuzzy preferences.

The results presented in this paper have inspired many new lines of research. For example, different optimization criteria, such as Pareto optimality and utilitarian max-plus, have been considered in [18] and in [25]. The STPP framework has also recently been extended to deal with disjunctive temporal constraints [28]. Another line of research combines preferences and uncertainty in temporal problems. For example, in [36] the notion of optimality w.r.t. preferences is paired with that of robustness to uncontrollable events, while in [26] the authors provide a system which combines probabilistic information on uncertain events with preferences.

Many issues remain open. An interesting line of work is to consider conditional preferences, that is, preferences that change depending on when other events occur. We also plan to test further our solvers and to try applying different learning techniques for inducing local preferences. We are also considering other optimization criteria and developing specific solvers that follow them, possibly using search.

## References


[1] M. Ai-Chang, J. L. Bresina, L. Charest, A. Chase, J. Cheng jung Hsu, A. K. Jónsson, B. Kanefsky, P. H. Morris, K. Rajan, J. Yglesias, B. G. Chafin, William C. Dias, and P. F. Maldague. Mapgen: Mixed-initiative planning and scheduling for the mars exploration rover mission. *IEEE Intelligent Systems*, 19(1):8–12, 2004.

[2] L.B. Almeida, T. Langlois, J.D. Amaral, and A. Plankhov. Parameter adaptation in stochastic optimization. In D. Saad, editor, *Online Learning in Neural Networks*, pages 111–134. Cambridge University Press, 1998.

[3] A. Biso, F. Rossi, and A. Sperduti. Experimental results on learning soft constraints. In *Proc. Seventh International Conference on Principles of Knowledge Representation and Reasoning (KR 2000)*, pages 435–444. Morgan Kaufmann, 2000.

[4] S. Bistarelli, H. Fargier, U. Montanari, F. Rossi, T. Schiex, and G. Verfaillie. Semiring-based CSPs and valued CSPs: Basic properties and comparison. *Over-Constrained Systems*, 1106:111–150, 1996.

[5] S. Bistarelli, U. Montanari, and F. Rossi. Constraint solving over semirings. In *Proc. Fourteenth International Joint Conference on Artificial Intelligence (IJCAI 95)*, pages 624–630. Morgan Kaufmann, 1995.

[6] S. Bistarelli, U. Montanari, and F. Rossi. Semiring-based constraint solving and optimization. *Journal of the ACM*, 44(2):201–236, mar 1997.

[7] J. Blythe. Visual exploration and incremental utility elicitation. In *Proc. Eighteenth National Conference on Artificial Intelligence and Fourteenth Conference on Innovative Applications of Artificial Intelligenc (AAAI/IAAI 2002)*, pages 526–532. AAAI Press/MIT Press, 2002.

[8] T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms*. MIT press, Cambridge, MA, 1990.

[9] P. Cousot. Asynchronous iterative methods for solving a fixed point system of monotone equations in a complete lattice. Technical Report R. R. 88, Institut National Polytechnique de Grenoble, 1977.

[10] R. Dechter. *Constraint processing*. Morgan Kaufmann, 2003.

[11] R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artif. Intell*, 49(1-3):61–95, 1991.

[12] D. Dubois and H. Prade. A review of fuzzy set aggregation connectives. *Inf. Sci.*, 36(1-2):85–121, 1985.

[13] Robert W. Floyd. Algorithm 97: Shortest path. volume 36, page 345, 1962.

[14] C. Goller. A connectionist approach for learning search-control heuristics for automated deduction. Technical report, Technical University Munich,Computer Science, 1997.

[15] S. Haykin. *Neural Networks: a comprehensive Foundation*. IEEE Press, 1994.

[16] T. Joachims. Optimizing search engines using clickthrough data. In *Proc. Eighth ACM International Conference on Knowledge Discovery and Data Mining (KDD 2002)*. ACM, 2002.

[17] L. Khatib, P. Morris, R. A. Morris, and F. Rossi. Temporal constraint reasoning with preferences. In *Proc. Seventeenth International Joint Conference on Artificial Intelligence (IJCAI 2001)*, pages 322–327. Morgan Kaufmann, 2001.

[18] L. Khatib, P. H. Morris, R. A. Morris, and K. B. Venable. Tractable Pareto optimization of temporal preferences. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI 2003)*, pages 1289–1294. Morgan Kaufmann, 2003.

[19] G. J. Klir and B. Yuan. *Fuzzy Sets and Fuzzy Logic: Theory and Applications*. Prentice Hall, NJ, 1995.

[20] C. E. Leiserson and J. B. Saxe. A mixed-integer linear programming problem which is efficiently solvable. *J. Algorithms*, 9(1):114–128, 1988.

[21] A. K. Mackworth. Consistency in networks of relations. *Artif. Intell.*, 8(1):99–118, 1977.

[22] A.K. Mackworth. Constraint satisfaction. In S. C. Shapiro, editor, *Encyclopedia of AI (second edition)*, volume 1, pages 285–293. John Wiley & Sons, 1992.

[23] T.M. Mitchell. *Machine Learning*. WCB/McGraw-Hill, 1997.

[24] U. Montanari. Networks of constraints: Fundamental properties and applications to picture processing. *Inf. Sci.*, 7:95–132, 1974.

[25] P. Morris, R. Morris, L. Khatib, S. Ramakrishnan, and A. Bachmann. Strategies for global optimization of temporal preferences. In *Proceeding of the 10th International Conference on Principles and Practice of Constraint Programming (CP-04)*, volume 3258 of *Lecture Notes in Computer Science*, pages 588–603. Springer, 2004.

[26] R. A. Morris, P. H. Morris, L. Khatib, and N. Yorke-Smith. Temporal planning with preferences and probabilities. In *ICAPS'05 Workshop on Constraint Programming for Planning and Scheduling*, 2005.

[27] N. Muscettola, P. H. Morris, B. Pell, and B. D. Smith. Issues in temporal reasoning for autonomous control systems. In *Agents*, pages 362–368, 1998.

[28] B. Peintner and M. E. Pollack. Low-cost addition of preferences to DTPs and TCSPs. In *Proceedings of the Nineteenth National Conference on Artificial Intelligence (AAAI 2004)*, pages 723–728. AAAI Press / The MIT Press, 2004.

[29] Bart Peintner and Martha E. Pollack. Anytime, complete algorithm for finding utilitarian optimal solutions to stpps. In *Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference (AAAI 2005)*, pages 443–448. AAAI Press AAAI Press / The MIT Press, 2005.

[30] M. Pelillo and M. Refice. Learning compatibility coefficients for relaxation labeling processes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 16(9):933–945, 1994.

[31] K Rajan, D. E. Bernard, G. Dorais, E. B. Gamble, B. Kanefsky, J. Kurien, W. Millar, N. Muscettola, P. Pandurang Nayak, N. F. Rouquette, B. D. Smith, W. Taylor, and Y. W. Tung. Remote Agent: An autonomous control system for the new millennium. In *Proceedings of the 14th European Conference on Artificial Intelligence (ECAI 2000)*, pages 726–730. IOS Press, 2000.

[32] J-C. Regin, J.F. Puget, and T.Petit. Representation of soft constraints by hard constraints. In *Proc. of JFPLC'2002*. Université de Nice Sophia-Antipolis, 2002.

[33] F. Rossi and A. Sperduti. Learning solution preferences in constraint problems. *J. Exp. Theor. Artif. Intell.*, 10(1):103–116, 1998.

[34] F. Rossi and A. Sperduti. Acquiring both constraint and solution preferences in interactive constraint systems. *Constraints*, 9(4), 2004.

[35] F. Rossi, A. Sperduti, K. B. Venable, L. Khatib, P. H. Morris, and R. A. Morris. Learning and solving soft temporal constraints: An experimental study. In *Proc. Eighth International Conference on Principles and Practice of Constraint Programming (CP 2002)*, number 2470 in LNCS, pages 249–263. Springer, 2002.

[36] F. Rossi, K. B. Venable, and N. Yorke-Smith. Controllability of soft temporal constraint problems. In *Proc. Tenth International Conference on Principles and Practice of Constraint Programming (CP 2004)*, number 3258 in LNCS, pages 588–603. Springer, 2004.

[37] F. Rossi, K.B. Venable, A. Sperduti, L. Khatib, P. Morris, and R. Morris. Solving and learning soft temporal constraints. *AI*IA Notizie*, 4:22–26, 2002.

[38] F. Rossi, K.B. Venable, A. Sperduti, L. Khatib, P. Morris, and R. Morris. Two solvers for tractable constraints with preferences. In *Proc. AAAI 2002 Workshop on Preferences in AI and CP: Symbolic Approaches*, 2002.

[39] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 2003.

[40] Z. Ruttkay. Fuzzy constraint satisfaction. In *Proc. First IEEE Conference on Evolutionary Computing*, pages 542–547. IEEE, 1994.

[41] D. Saad. *Online Learning in Neural Networks*. Cambridge University Press, 1998.

[42] T. Schiex. Possibilistic Constraint Satisfaction problems or "How to handle soft constraints?". In *Proc. Eighth Annual Conference on Uncertainty in Artificial Intelligence (UAI 1992)*, pages 268–275. Morgan Kaufmann, 1992.

[43] T. Schiex, H. Fargier, and G. Verfaillie. Valued Constraint Satisfaction Problems: Hard and easy problems. In *Proc. Fourteenth International Joint Conference on Artificial Intelligence (IJCAI 95)*, pages 631–639. Morgan Kaufmann, 1995.

[44] E. Schwalb and R. Dechter. Coping with disjunctions in Temporal Constraint Satisfaction Problems. In *Proc. Eleventh National Conference on Artificial Intelligence (AAAI 93)*, pages 127–132. AAAI Press/MIT Press, 1993.

[45] R. E. Shostak. Deciding linear inequalities by computing loop residues. *J. ACM*, 28(4):769–779, 1981.

[46] K. Stergiou and M. Koubarakis. Backtracking algorithms for disjunctions of temporal constraints. *Artif. Intell.*, 120(1):81–117, 2000.

[47] R. S. Sutton. Adapting bias by gradient descent: An incremental version of delta-bar-delta. In *Proc. Tenth National Conference on Artificial Intelligence (AAAI 92)*, pages 171–176. MIT Press, 1992.

[48] R. S. Sutton and S. D. Whitehead. Online learning with random representations. In *Proc. Tenth International Conference on Machine Learning (ICML 1993)*, pages 314–321. Morgan Kaufmann, 1993.

[49] L. Vila and L. Godo. On fuzzy temporal constraint networks. *Mathware and Soft computing*, 3:315–334, 1994.

[50] L. Zadeh. Calculus of fuzzy restrictions. In *Fuzzy Sets and their Applications*, pages 1–39. Academic Press, 1975.