# Solving and Learning Soft Temporal Constraints: Experimental Scenario and Examples

F. Rossi, K.B. Venable
Department of Pure and Applied Mathematics, University of Padova, Italy
frossi@math.unipd.it, kvenable@studenti.math.unipd.it

A. Sperduti
Department of Computer Science, University of Pisa, Italy
perso@di.unipi.it

L. Khatib(*), P. Morris, R. Morris
(*) Kestrel Technology
NASA Ames Research Center, Moffett Field, CA, USA
{lina,pmorris,morris}@ptolemy.arc.nasa.gov

October 1, 2001

**Abstract**

Soft temporal constraint problems allow to describe in a natural way scenarios where events happen over time and preferences are associated to event distances and durations. However, sometimes such 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 use machine learning techniques which learn the local preferences from the global ones.

In this paper we describe the existing framework for both solving and learning preferences in temporal constraint problems, the implemented modules, the experimental scenario, and preliminary results on some examples.

## 1 Introduction and motivation

Several real world problems involving the manipulation of temporal information in order to find an assignment of times to a set of activities or events can naturally be viewed as having preferences associated with local temporal decisions, where 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.

However, in many temporal reasoning problems it is difficult or impossible to specify a local preference on durations. In real world scheduling problems, for example, it is sometimes easier to see how preferrable is a solution, but it may be virtually impossible to specify how specific ordering choices between pairs of events contribute to such global preference value. If such knowledge were at hand, it could be used as heuristics to guide the scheduler to prefer local assignments that were found to yield better solutions.

We solve this problem by automatically generating local temporal preference information via a machine learning approach using a representation of local preferences in terms of soft constraints [7, 1, 8].

This paper presents the current formalism and results for soft temporal constraint problems, and describes the machinery we have developed for both solving and learning such problems. The implemented modules rely on the theoretical results (such as those on tractability of some classes of problems) and make some assumptions for both tractability and efficiency. In particular, our solver is able to deal with soft temporal constraints with one interval per constraint, and with a particular shape of the preference functions, which assure tractability (like for Simple Temporal Constraints in the case of hard constraints); our random problem generator is based on some parameters to generate a soft temporal problem, which suitably extend the usual ones for hard CSPs (density, tightness, ...); our learning module learns preference functions which have the shape of convex quadratic functions, to assure a trade-off between efficiency and generality, and assumes that preferences are dealt with via a fuzzy (max-min) framework. We also show the results of some tests of our modules on randomly generated problems and also toy problems.

## 2 Temporal constraint problems with preferences

The Temporal CSP framework (TCSP) [3] has been used widely to solve temporal reasoning problems. This framework is based on knowledge represented as constraints on distances or durations of events. More precisely, variables represent events happening over time, and each constraint gives an allowed range for the distances or durations, expressed as a set of intervals over the time line. For example, a constraint over $X$ and $Y$ could say that $5 \leq Y - X \leq 10$ or $15 \leq Y - X \leq 30$, which is formally represented by the two intervals $[5, 10]$ and $[15, 30]$. Informally, this constraint states that the distance between the two events should be either between 5 and 10, or between 15 and 30. Satisfying such a constraint means choosing any of the allowed distances. A solution for a TCSP consisting of a set of temporal constraints is an assignment of values to its variables such that all constraints are satisfied.

As expected, general TCSPs are NP-hard. However, TCSPs with just one interval for each constraint, called STPs, are polynomially solvable. In fact, one can transform the given STP into a distance graph, apply to this graph a shortest

path algorithm, and then assign to each variable the value corresponding to the shortest distance thus found (see [3] for details).

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, recently [5] a new framework has been proposed, where each temporal constraint is associated with a preference function, which specifies the preference for each distance. This framework is based on a simple merger of TCSPs and soft constraints, where for soft constraints we have taken a general framework based on semirings [2]. The result is a class of problems called Temporal Constraint Satisfaction problems with preferences (TCSPPs).

A *soft temporal constraint* in a TCSPP is represented by a pair consisting of a set of disjoint intervals and a preference function: $\langle I = \{[a_1, b_1], \ldots, [a_n, b_n]\}, f\rangle$, where $f : I^1 \rightarrow A$, is a mapping of the elements of $I$ into preference values, taken from a set $A$.

A *solution* to a TCSPP is a complete assignment to all the variables that satisfies the distance constraints. Each solution has a *global preference value*, obtained by combining the local preference values found in the constraints. To formalize the process of combining local preferences into a global preference, and comparing solutions, we impose a semiring structure onto the TCSPP framework.

A *semiring* is a tuple $\langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$ such that $A$ is a set and $\mathbf{0}, \mathbf{1} \in A$; $+$, the additive operation, is commutative, associative and $\mathbf{0}$ is its unit element; $\times$, the multiplicative operation, is associative, distributes over $+$, $\mathbf{1}$ is its unit element and $\mathbf{0}$ is its absorbing element. A *c-semiring* is a semiring in which $+$ is idempotent (i.e., $a + a = a, a \in A$), $\mathbf{1}$ is its absorbing element, and $\times$ is commutative. These additional properties (w.r.t. usual semirings) are required to cope with the usual nature of constraints.

C-semirings allow for a partial order relation $\leq_S$ over $A$ to be defined as $a \leq_S b$ iff $a + b = b$. Informally, $\leq_S$ gives us a way to compare tuples of values and constraints, and $a \leq_S b$ can be read *b is better than a*. Moreover, one can prove that $+$ 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 where, for all $a, b \in A$, $a + b = lub(a, b)$ (where *lub*=least upper bound); if $\times$ is idempotent, then $\langle A, \leq_S\rangle$ is a complete distributive lattice and $\times$ is its greatest lower bound (*glb*).

Given a semiring[2] with a set of values $A$, each preference function $f$ associated with a soft constraint $\langle I, f\rangle$ of a TCSPP takes an element from $I$ and returns an element of $A$, where $A$ is the carrier of a semiring. This allows us to associate a preference with a duration or a distance.

[1]Here by I we mean the set of all elements appearing in the intervals of I.

[2]For simplicity, from now on we will write *semiring* meaning *c-semiring*.

The two semiring operations allow for complete solutions to be evaluated in terms of the preference values assigned locally. More precisely, given a solution $t$ in a TCSPP with associated semiring $\langle A, +, \times, \mathbf{0}, \mathbf{1}\rangle$, let $T_{ij} = \langle I_{i,j}, f_{i,j}\rangle$ be a soft constraint over variables $X_i, X_j$ and $(v_i, v_j)$ be the projection of $t$ over the values assigned to variables $X_i$ and $X_j$ (abbreviated as $(v_i, v_j) = t_{\downarrow X_i, X_j}$). Then, the corresponding preference value given by $f_{ij}$ is $f_{ij}(v_j - v_i)$, where $v_j - v_i \in I_{i,j}$. Finally, where $F = \{x_1, \ldots, x_k\}$ is a set, and $\times$ is the multiplicative operator on the semiring, let $\times F$ abbreviate $x_1 \times \ldots \times x_k$. Then the global preference value of $t$, $val(t)$, is defined to be $val(t) = \times\{f_{ij}(v_j - v_i) \mid (v_i, v_j) = t_{\downarrow X_i, X_j}\}$. The optimal solutions of a TCSPP are those solutions which have the best global preference value, where "best" is determined by the ordering $\leq_S$ of the values in the semiring.

For example, consider the semiring $S_{fuzzy} = \langle [0, 1], max, min, 0, 1\rangle$, used for fuzzy constraint solving [9]. The global preference value of a solution will be the minimum of all the preference values associated with the distances selected by this solution in all constraints, and the best solutions will be those with the maximal value. Another example is the semiring $S_{csp} = \langle \{false, true\}, \vee, \wedge, false, true\rangle$, which allows to describe hard constraint problems [6].

A special case occurs when each constraint of a TCSPP contains a single interval. We call such problems *Simple Temporal Problems with Preferences* (STPPs), due to the fact that they generalize Simple Temporal Problems (STPs) [3]. 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. In [5] it has been shown that, while in general TCSPPs are NP-hard, under certain restrictions on the "shape" of the preference functions and on the semiring, STPPs are tractable.

Semi-convex functions are such that, if one draws a horizontal line anywhere in the Cartesian plane defined by the function, the set of $X$ such that $f(X)$ is not below the line forms an interval. More formally, a *semi-convex* function $f$ is one such that, for all $Y$, the set $\{X$ such that $f(X) \geq Y\}$ forms an interval. It is easy to see that semi-convex functions include linear ones, as well as convex and some step functions. 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 1 shows some examples of semi-convex and non-semi-convex functions.

It is proven in [5] that STPPs with semi-convex preference functions and a semiring with a total order of preference values and an idempotent multiplicative operation can be solved in polynomial time.

# 3 A solving module for STPPs

The tractability results for STTPs can be translated in practice as follows: to find an optimal solution for an STPP, we can first apply path-consistency (suitably adapted to STPPs, see [5]) and then use a search procedure to find a solution without the need to backtrack. More in details, it is possible to show

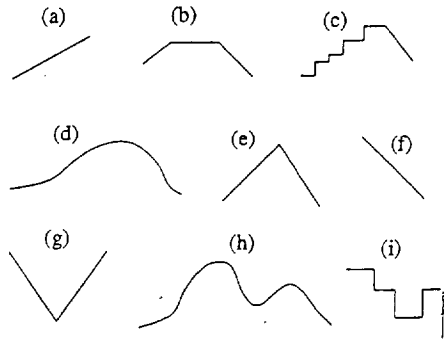


Figure 1: Examples of semi-convex functions (a)-(f) and non-semi-convex functions (g)-(i)

that:

- 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 (see [5]). The only difference in the two problems is that the new one can have worse preference values in the preference functions.
- After applying path-consistency, all preference functions in $P'$ have the same best preference level.
- 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 $P$.

Our solving module relies on these results. In fact, the STPP solver takes as input an STPP with semi-convex preference functions, and returns an optimal solution of the given problem, working as follows:

- First, path-consistency is applied to the given problem, producing a new problem $P'$.
- Then, an STP corresponding to $P'$ is constructed 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.

The STPP solver has been tested both on toy problems and on randomly-generated problems. The random generator we have developed focusses 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 facility of expressing functions in this class (just three parameters). 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 of variables;
- range $r$ for the initial solution: to assure that the generated problem has at least a solution, we first generate such a solution, by giving to each variable a random value within the range $[0, r]$ ;
- density: percentage of constraints that are not universal (that is, with the biggest interval and preference 1 for all interval values);
- maximum expansion from initial solution (max): 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*): we recall that each preference function 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 passes through the end points of the interval, with value 0, and the middlepoint, with value 0.5, and then modifies it according to the percentages specified for $a$, $b$, and $c$.

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 $[t - t1, t + t2]$, where $t1$ and $t2$ 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 explicitely 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 [10].

In Table 1 we show some results for finding an optimal solution for STPPs generated by our generator, which has been developed in C++ and tested on a Pentium III 1GHz. As it can be seen, our current solver is very slow. The main reason is that it uses a pointwise representation of the constraint intervals and

| | n=10 | n=20 | n=30 | n= 40 | n=50 |
|---|---|---|---|---|---|
| d=20% | 1'22" | 802" | 13'34" | 47'13" | >1h |
| d=40% | 28" | 3'53" | 6'30" | 14'37" | 25'5" |
| d=60% | 16" | 2'37" | 4'24" | 9'28" | 18' |
| d=80% | 8" | 1'11" | 2'49" | 4'34" | 11'18" |

Table 1: Time needed to find an optimal solution, as a function of the number of variables ($n$) and the density ($d$). The other parameters are: range for the initial solution = 100, maximum expansion for all constraints = 100, perturbation for preference functions: 20, 30, 30.

the 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.

A new solver, which uses a different solving algorithm, not based on path-consistency, is under development. This solver will be able to avoid the pointwise representation and will perform a binary search over the preference value range to find the optimal preference level (the same one that is now found by path-consistency). We are confident that the new solver will be much faster than the current one. However, it will be less general because it will work with specific classes of preference functions with specific analytical representations.

# 4 Learning soft temporal constraints

We now describe our methodology to learn preferences in STPPs from examples of solution ratings. Notice that for now we focus on STPPs rather than considering general TCSPPs, since STPPs are, with some restrictions, tractable problems, and since the whole point of this paper is to make soft temporal reasoning more practical.

The problem to be considered here can be formally described as an inductive learning problem [11]. Inductive learning can be defined as the ability of a system to induce the correct structure of a map $d$ which is known only for particular inputs. More formally, defining an example as a pair $(x, d(x))$, the computational task is as follows: given a collection of examples of $d$, i.e., the *training set*, return a function $h$ that approximates $d$. Function $h$ is called a hypothesis.

A common approach to inductive learning, especially in the context of neural networks, is to evaluate the quality of a hypothesis $h$ (on the training set) through an *error function* [4]. An example of popular error function, that can be

used over the reals, is the sum of squares error [4]: $E = \frac{1}{2}\sum_{i=1}^{n}(d(x_i) - h(x_i))^2$, where $(x_i, d(x_i))$ is the $i$-th example of the training set.

Given a starting hypothesis $h_0$, the goal of learning is to minimize the error function $E$ 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 the domain is isomorphic to the reals or not. The usual way to be used over the reals, and if $h_W$ is continuous and derivable, is to follow the negative of the *gradient* of $E$ with respect to $W$. This technique is called *gradient descent* [4]. 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: $W(\tau + 1) = W(\tau) + \Delta W(\tau)$, where $\Delta W(\tau) = -\eta \frac{\partial E}{\partial W(\tau)}$, and $\eta$ is the step size used for the gradient descent. Learning is stopped when a minimum of $E$ is reached. Note that, in general, there is no guarantee that the found minimum is global.

Learning in our context can be used to find suitable preference functions to be associated to the constraints of a given STP. More precisely, let $P = (V, C)$ be an STP where $V$ is a set of variables with domains consisting of time instants, and $C$ is a set of distance constraints of the form $l \leq X - Y \leq u$, where $X, Y \in V$ and $l, u$ are time points. Let $f$ be a function $f : S \to U$, where $S$ is the set of solutions to $P$ and $U$ 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 $(s, f(s))$, where $s$ is a solution to the original STP and $f(s)$ is its "score". In the following, we use $P$ to denote an STP and $P'$ to denote a corresponding STPP. Also, $val_{P'}(t)$ is used to indicate the value of a solution $t$ over $P'$.

**Semiring choice.** Let $P$ and $f$ be as defined above, and suppose a set of examples $TR = \{(t_1, r(t_1)), \ldots, (t_m, r(t_m))\}$ is given. To infer the local preferences, we must also be given the following: a semiring whose element set $A$ contains the values $r(t_i)$ in the examples; and a distance function over such a semiring. For example, if the score values are positive real numbers, we could choose the semiring $\langle \Re^+, min, +, +\infty, 0 \rangle$ and, as distance function, the usual one over reals: $dist(val_{P'}(t), r(t)) = \mid val_{P'}(t) - r(t) \mid$. Given all the above, the goal is to define a corresponding STPP $P'$ over the same semiring such that $P$ and $P'$ have the same set of variables, domains and interval constraints, and for each $t$ such that $(t, r(t))$ is an example, $dist(val_{P'}(t), r(t)) < \epsilon$, where $\epsilon > 0$ and small.

**Parameters.** If the first condition is true, the only free parameters that can be arbitrarily chosen in order to satisfy the other conditions are the values to be associated to each distance. For each constraint $c_{ij} = \{[a_1, b_1], \ldots, [a_n, b_n]\}$ in $P$, the idea is to associate, in $P'$, a free parameter $w_d$, where $d = X_j - X_i$ (note

that such a parameter must belong to the set of the chosen semiring), to each element $d$ in $I = \bigcup_i [a_i, b_i]$. This parameter will represent the preference over that specific distance. With the other distances, those outside $I$, we associate the constant $\mathbf{0}$, (the lowest value of the semiring (w.r.t. $\leq_S$)).

If $I$ contains an infinite number of distances, we would need an infinite number of parameters, which would make learning impossible. To avoid this problem, we can restrict the class of preference functions to a subset which can be described via a finite number of parameters. For example, if we use only linear functions, we just need two parameters $a$ and $b$, since every linear function 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 $f_W : (W \times I) \rightarrow A$.

The value assigned to each solution $t$ in $P'$ is

$$val_{P'}(t) = \prod_{c_{ij} \in P'} [\sum_{d \in \bigcup_{D \in I_{ij}} D} \text{check}(d, t, i, j) \times f_W(d)] \quad (1)$$

where $\prod$ generalizes the $\times$ operation, $\sum$ generalizes $+$, $I_{ij}$ is the set of intervals associated to constraint $c_{ij}$, and $\text{check}(d, t, i, j) = \mathbf{1}$ if $d = t \downarrow_{X_j} - t \downarrow_{X_i}$ and $\mathbf{0}$ otherwise. Note that, for each constraint $c_{ij}$, there is exactly one distance $d$ such that $\text{check}(d, t, i, j) = \mathbf{1}$, namely $d = t \downarrow_{X_j} - t \downarrow_{X_i}$. Thus, $val_{P'}(t) = \prod_{c_{ij} \in P'} f_W(t \downarrow_{X_j} - t \downarrow_{X_i})$. 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.

**Learning semi-convex preference functions.** Suppose we are given a class of STPs to be "softened" via the learning approach defined above. As we know, STPs are tractable [3]. 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. Moreover, one needs to choose a semiring which preserves semi-convexivity.

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 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. Therefore, there will be cases in which the desired solution ratings, as specified in the training set, cannot be matched. For example, the user could have specified a set of examples which is not consistent with any soft temporal constraint problem using that class of semi-convex preference functions. 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

| Max | Mean error (min,max) | Number of examples |
|---|---|---|
| 20 | 0.03 (0.02,0.04) | 500 |
| 30 | 0.03 (0.02,0.04) | 600 |
| 40 | 0.0333 (0.02,0.05) | 700 |

Table 2: Mean error and number of examples for learning preferences in some STPPs.

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.

# 5 The learning module

We have developed a learning module which can learn fuzzy STPPs where the preference functions are quadratic functions of the form $ax^2 + bx + c$ with $a \leq 0$. Notice that this class includes both constant, linear, and semi-convex quadratic functions.

The input is a set of pairs consisting of a solution and its preference. Part of this set will be used as the training set, and the rest as the test set. Learning is performed via a gradient descent technique using an approximated version of the min operation which is continuous and derivable. Learning stops when the global error is under a certain threshold. At this point, we have an STPP with preference functions in the shape of convex quadratic functions, whose solution are ranked very similarly to the original examples in the input.

The learning module has been tested on some randomly generated problems: every test involves the generation of an STPP via our generator, and then the generation of some examples of solutions and their rating. Then the STPP, without its preference functions, is given to the learning module, which, starting from the examples, learns new prefernce functions over the constraints, until the error (that is, the difference between the solution ratings in the test set and in the new problem) is small enough.

Figure 2 shows the number of examples in the training (and also in the test) set, and the mean error (computed as the average of the mean error for three problems), for learning preferences in STPPs with 20 variables, range = 40, density = 40%, and function perturbation parameters 10, 10, and 5. The maximum expansion, which, we recall, is related to the tightness notion, is 20, 30, and 40.

What can be noticed, even with such a small number of tests, is that the mean error increases as the parameter max increases, even if one uses more examples (which should ease the learning process). This is due to the fact that a larger value for max may yields larger intervals, and thus preference functions with larger domains and a larger number of solutions, which require more work from the learning algorithm. This trend seems to be confirmed also by other experimental results not reported here.

## 6 Conclusions and future work

This paper has presented the experimental scenario and first examples for both solving and learning soft temporal constraints. We plan to further test the overall system, composed of the solver and the learning module, using other classes of randomly generated STPPs and also real-life problem instances such as satellite event scheduling.

We also plan to extend our solver to deal with soft temporal problems which are not simple, or which have preference functions which are not semi-convex. In this line, we already have some theoretical results which suggest us to decompose such problems into STPPs, find their solutions, and then suitably combine the solution sets to generate the best solutions of the original problem. We plan to follow such results in the development of such solver.

## References


[1] A. Biso, F. Rossi, and A. Sperduti. Experimental Results on Learning Soft Constraints. *Proc. KR 2000*, Morgan Kaufmann, 2000.

[2] S. Bistarelli, U. Montanari, and F. Rossi. Semiring-based Constraint Solving and Optimization. *Journal of the ACM*, 44(2):201–236, March 1997.

[3] R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. Artificial Intelligence, Vol. 49, 1991, pp. 61-95.

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

[5] L. Khatib, P. Morris, R. Morris, F. Rossi. Temporal Constraint Reasoning With Preferences. *Proc. IJCAI 2001.*

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

[7] F. Rossi and A. Sperduti. Learning solution preferences in constraint problems. *Journal of Experimental and Theoretical Computer Science*, 1998. Vol 10.

[8] L. Khatib, P. Morris, R. Morris, F. Rossi, A. Sperduti. Learning Preferences on Temporal Constraints: A Preliminary Report. *Proc. TIME 2001*, IEEE Computer Society Press, 2001.

[9] T. Schiex. Possibilistic constraint satisfaction problems, or "how to handle soft constraints?". In *Proc. 8th Conf. of Uncertainty in AI*, pages 269–275, 1992.

[10] E. Schwalb, R. Dechter. Coping with disjunctions in temporal constraint satisfaction problems. In *Proc. AAAI-93*, 1993.

[11] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, 1995.