Reasoning about Numeric and Symbolic Time Information

Malek Mouhoub
Department of Math & Computer Science, University of Lethbridge
4401 University Drive,
Lethbridge, Alberta, Canada, T1K 3M4
mouhoub@cs.uleth.ca

Abstract

Many applications such as planning, scheduling and natural language processing involve managing both symbolic and numeric aspect of time. We have developed a temporal model, TemPro, based on the interval algebra, to express such applications in terms of qualitative and quantitative temporal constraints. TemPro extends the interval algebra relations of Allen to handle numeric information. To solve a temporal constraint problem represented by TemPro, we have developed a method using constraint propagation, at the numeric and symbolic levels. In order to deal with real time applications or those applications where a complete solution cannot be obtained, we have modified the propagation techniques so that they will be able to solve temporal problems by giving a solution with a quality depending on the time allocated for computation.

1. Introduction

Time plays an important role in various activities such as planning, scheduling, process control, etc. This has led researchers to propose numerous approaches for representing and reasoning on time. One of the most attractive work in Artificial Intelligence is Allen's interval algebra[1]. A significant amount of work has been devoted to this approach, e.g. [9] [24][25, 26][27]. Most work has been devoted to complexity analysis and to the definition of sub-classes of the interval algebra that admit complete polynomial algorithms for constraints maintenance [19][2]. Some generalization were also studied [12] [11], mostly at the representation level. The expressiveness of representation used in interval algebra is more general than those relying on point based representations such as the numerical representation used in the Dean's TMM [4], Meiri's approach [16], or in IxTeT [7] . As a drawback, interval algebra is more complex and costly. Furthermore, it does not take into account numerical constraints, unless restrictions are accepted [9][20].

In order to address these two drawbacks, we have defined a temporal model, TemPro[18], that extends the interval algebra in order to efficiently take into account numerical constraints. In addition to interval algebra relations, the TemPro model handles different kinds of entities such as windows, durations for intervals and dates defined over a window's limits (that can be defined in form of mathematical equations). To solve a temporal problem represented by TemPro, we have developed a method based on constraint satisfaction techniques, at the numeric and the symbolic levels. Indeed, since we use a discrete representation of time, a temporal problem including both numeric and symbolic information will be converted, with TemPro, to a constraint satisfaction problem.

Since constraint satisfaction is an NP-hard problem (a problem that cannot be solved in polynomial time), filtering techniques, such as arc and path consistency, which simplify the task by eliminating some local inconsistencies, have been largely studied. We present in this paper the use of such techniques to solve temporal constraint satisfaction problems. In order to deal with real time applications or those applications where a complete solution cannot be obtained, we have modified the propagation techniques so that they will be able to solve temporal problems by giving a solution with a quality depending on the time allocated for computation.

In the following section, we will present through an example, the different components of the model TemPro. The corresponding resolution techniques for solving temporal constraint problems are then presented in section 3. In section 4 we will show how are scheduling problems solved using our model. Section 5 concerns propagation techniques that deal with unsolvable and real time problems. Some experiments evaluating these techniques are reported in the same section. Concluding remarks and possible extensions of the model TemPro are given in section 6.

2. Knowledge Representation

Consider the following temporal reasoning problem :

John, Mike and Lisa work for a company in Calgary. It takes John 20 minutes, Mike 25 minutes and Lisa 30 minutes to get to work. Every day, John left home between 7:20 and 7:26. Mike arrives at work between 7:55 and 8:00 and Lisa arrives at work between 7:50 and 8:00. We also know that John and Mike meet at a traffic light on their way to work, that Mike arrives to work before Lisa and that Lisa and John go to work at the same time.

Using TemPro, the above example can be mapped into a network-based representation involving 3 temporal events, J, L and M representing the fact that John, Lisa and Mike are going to work respectively, and a set of binary constraints, each representing a qualitative relation between a pair of events (see figure 2). Each temporal event is defined on a discrete domain standing for the set of possible occurrences (SOPO) in which the event can hold. The SOPO represents the quantitative constraint of the event and is defined by the fourfold $[begintime, endtime, duration, step]$ where $begintime$ and $endtime$ are respectively the earliest begin time and the latest end time of the corresponding event, $duration$ is the duration of the event and $step$ defines the distance between the starting time of two adjacent intervals within the SOPO. Thus, if e_i is an event numerically constrained by the fourfold $[inf_i, sup_i, d_i, p_i]$, then the SOPO of e_i is defined as :
$SOPO_i = \{intv_j \mid begin(intv_j) = inf_i + k * p_i, end(intv_j) = begin(intv_j) + d_i * p_i, end(intv_j) <= sup_i,\ k \in [0, \frac{sup_i - inf_i}{p_i} - d_i] \cap \aleph\}$.

$begin$ and end are functions on intervals that return the begin and the end points of a given interval, respectively (see figure 1).

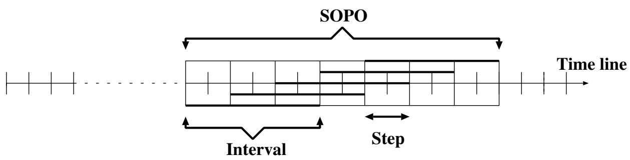


Figure 1. The SOPO of a given event

The qualitative relation between two events I and J can take the following form : $I_1\ r_1\ r_2 \ldots r_n\ I_2$ where each of the r_i 's is one of the thirteen Allen primitives[1] : $Precedes$, $during$, $overlaps$, $meets$, $starts$, $finishes$ noted respectively P, D, O, M, S and F; their converses $P^{\smile}$, $D^{\smile}$, $O^{\smile}$, $M^{\smile}$, $S^{\smile}$ and $F^{\smile}$; and the equality relation E (see table 1).

Relation	Symbol	Inverse	Meaning
X precedes Y	P	$P^{\smile}$	XXX YYY
X equals Y	E	E	XXX YYY
X meets Y	M	$M^{\smile}$	XXXYYY
X overlaps Y	O	$O^{\smile}$	XXXX YYYY
X during y	D	$D^{\smile}$	XXX YYYYYY
X starts Y	S	$S^{\smile}$	XXX YYYYY
X finishes Y	F	$F^{\smile}$	XXX YYYYY

Table 1. Allen primitives

Related work

Some research works addressing the problem of handling both metric and symbolic constraints have already been reported. Meiri [16] has proposed a model based on a single network (time map) managing both constraints : metric constraints that restrict the distance between time points, and symbolic constraints that specify the relative position between temporal objects (either points or intervals). Kautz and Ladkin [8] have proposed a model allowing for the representation and processing of metric temporal information in the form of a system of simple linear inequalities to encode metric relations between time points, and systems of binary constraints in Allen's qualitative temporal calculus to encode qualitative relations between time points. In the Kautz and Ladkin approach, both kind of constraints are independently processed in separate networks. At the representation level, our approach is close to that of Kautz and Ladkin, but it differs in the way metric constraints are defined. Quantitative constraints are represented by linear inequalities in the Kautz and Ladkin approach, while they are modeled by SOPOs in our representation. Thus, we come out with a uniform language at the numerical and symbolic level . In the approach of Kautz and Ladkin, the propagation is performed separately in both metric and qualitative networks. Translation procedures are then used to update the metric constraints from the new Allen constraints and vice versa. This process is repeated until no new statements can be derived. As we will see in the next sections, in our approach the process to compute Allen and metric constraints network is performed in one pass. A backtracking algorithm is then used to search for a possible numerical solution.

3. Finding a single solution

Given a TemPro constraint-based network, one important task of reasoning is to determine the consistency of the network, and therefore to find a solution of this network if the latter is consistent. Since we are dealing with a constraint satisfaction problem, deciding consistency is NP-hard. In order to overcome these difficulties, we propose an algorithm based on local consistency algorithms. Indeed, these algorithms transform the network of constraints into an equivalent and simpler one by removing from the domain of each variable some values that do not belong to any global solution. A k-consistency algorithm removes all inconsistencies involving all subsets of k variables belonging to N. For $k = 2$ and $k = 3$, the solutions are called arc and path consistent respectively. The k-consistency problem is polynomial in time $O(n^k)$, where n is the cardinal of N. A k-consistency algorithm does not solve the constraint satisfaction problem, but simplifies it. Due to the incompleteness of constraint propagation, in the general case, search is necessary to solve a CSP problem, even to check if a single solution exists.

Before presenting our resolution algorithm, let us present the notions of arc-consistency and path-consistency in TemPro.

3.1. Arc consistency of a TemPro network

A TemPro network is arc-consistent if all its arcs are arc consistent. An arc (i, j) is arc consistent if and only if, for any value a belonging to $SOPO_i$, there exists a value b belonging to $SOPO_j$ such that the qualitative disjunctive relation R_{ij} of the pair of variables (EV_i, EV_j) is satisfied. a belonging to $SOPO_i$ is said to be supported by b belonging to $SOPO_j$. The network can be converted into an equivalent arc-consistent form by applying an arc-consistency algorithm (cf. figure 3).

In a previous work [18], we have compared different arc consistency algorithms and concluded that AC-3[14] is the algorithm of choice in general. In this paper we will use AC-3 (see table 2) to perform arc-consistency on a TemPro network.

3.2. Path consistency of a TemPro network

A network is symbolic path consistent if all its paths are consistent. When the path length is 2, path consistency is called 3-consistency. The network is symbolic 3-consistent if, for each triplet (i, j, k) $\quad (r \in R_{ik} \ \Rightarrow \ r \in R_{ij} \circ R_{jk})$ where r is a basic Allen relation, R_{ij}, R_{ik} and R_{jk} are qualitative disjunctive relations, and $R_{ij} \circ R_{jk}$ is calculated using the relation composition table for the interval algebra defined by Allen [1]. Montanari [17] showed that 3-

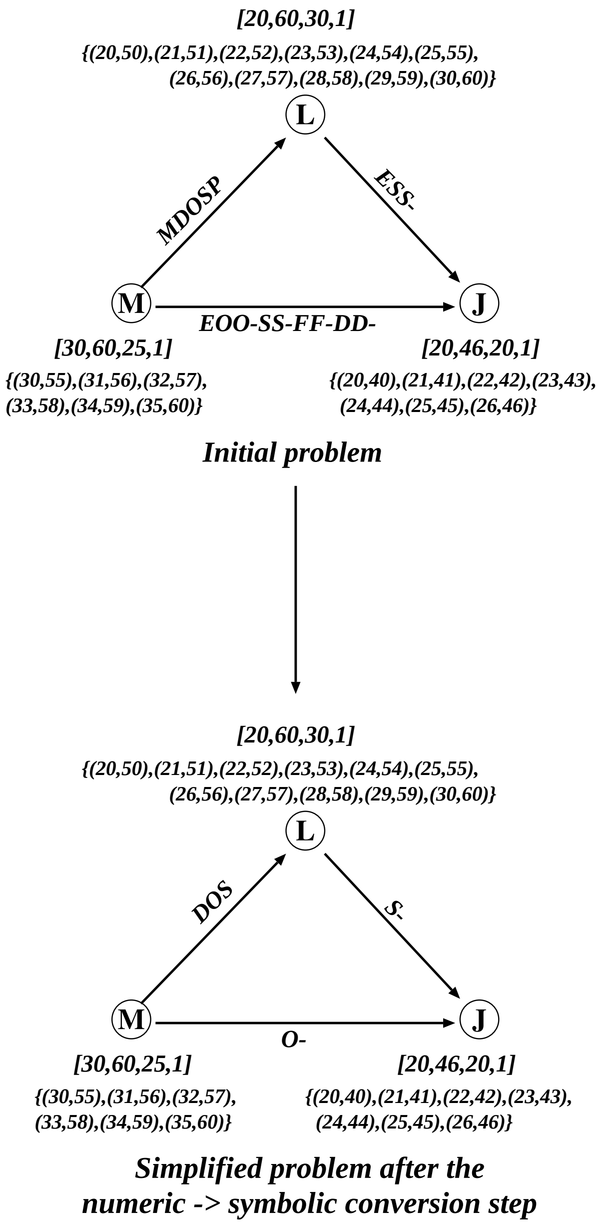


Figure 2. Numeric $\rightarrow$ symbolic conversion step.

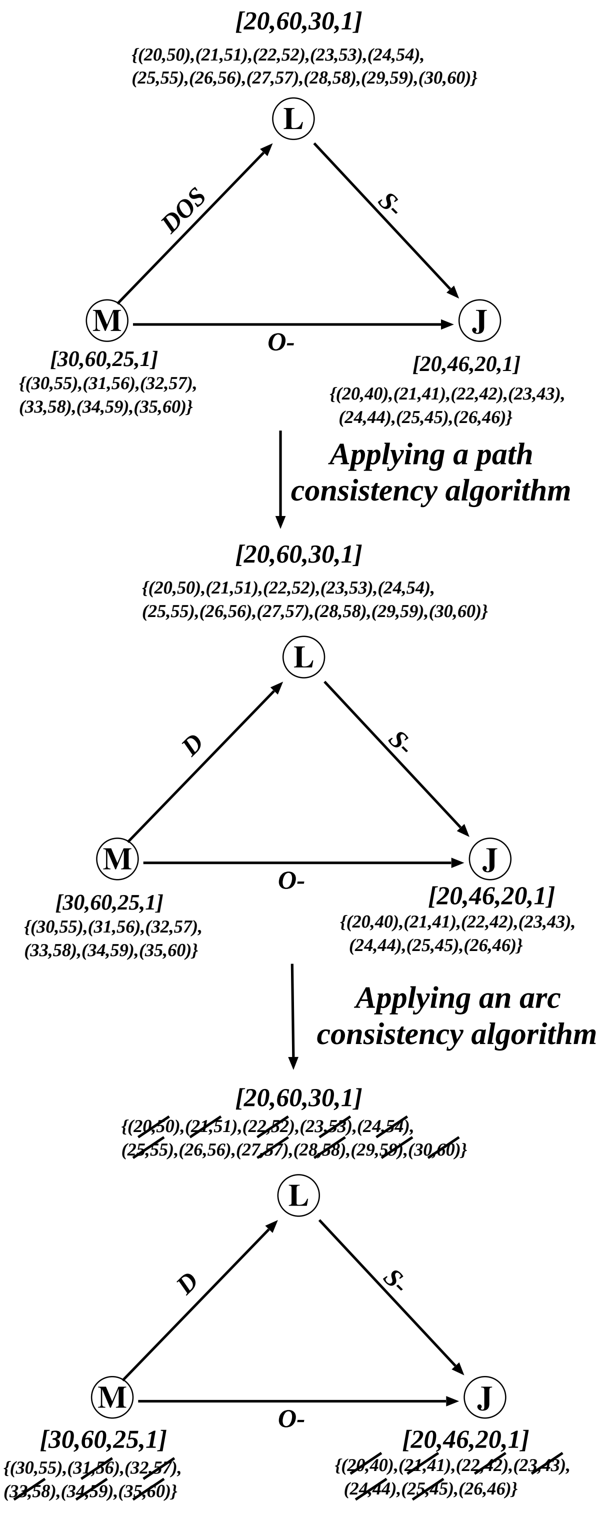


Figure 3. Applying path and arc consistency on a Temporal Network

Function $REVISE(i,j)$
  $REVISE \leftarrow false$
  For each interval $a \in SOPO_i$ **Do**
    If $\neg compatible(a,b)$ for each interval $b \in SOPO_j$ **Then**
      remove a from $SOPO_i$
      $REVISE \leftarrow true$
    End-If
  End-For

Algorithm AC-3
1. Given a TemPro network $TN = (E,R)$
   (*E: set of events*,
   R: set of disjunctive relations between events)
2. $Q \leftarrow \{(i,j) \mid (i,j) \in R\}$
   (list initialized to all relations of TN)
3. **While** $Q \neq Nil$ **Do**
4. $Q \leftarrow Q - \{(i,j)\}$
5. **If** $REVISE(i,j)$ **Then**
6. $Q \leftarrow Q \sqcup \{(k,i) \mid (k,i) \in R \wedge k \neq j\}$
7. **End-If**
8. **End-While**

Table 2. Pseudo-code of the algorithm AC-3

consistency implies general path consistency. A TemPro network can be converted into an equivalent path consistent form by applying a path consistency algorithm to the qualitative disjunctive relations of the underlying qualitative network (see figure 3).

Among path consistency algorithms, we applied PC-2[13] to TemPro networks. The composition $R_{ij} \circ R_{jk}$ is calculated and then intersected with R_{ik} to form the new potentially smaller relation $R_{ik} \cap (R_{ij} \circ R_{jk})$. We call this a triangle operation. A triangle operation is stabilized if the new relation $R_{ik} \cap (R_{ij} \circ R_{jk})$ is identical to R_{ik}. A qualitative network is 3-consistent (and therefore path consistent) if every triangle operation stabilizes. This is the principle of PC-1 [10].
In fact, only the triangle of edges (M_{ij} entries) whose labels have changed during the previous iteration need to be recomputed. Thus, we use (as proposed by Mackworth [13] and Allen [1]) a queue data structure for maintaining the triangles that must be recomputed. The computation simply proceeds until the queue is empty. This is the principle of PC-2 that we have implemented in the same way as reported by van Beek and Manchak [26]. To improve the efficiency of PC-2, we changed, as reported by [2], the way compositions-intersections of relations are achieved during the path consistency process (following the idea “one support is sufficient”).

3.3. Resolution algorithm

Our resolution algorithm is described by the following steps :

Numeric → Symbolic conversion : *From the numeric information, we can extract the corresponding symbolic relations. An intersection of these relations with the given qualitative information will reduce the size of the latter which simplifies the size of the original problem. When applying this step to our example we will obtain the following temporal network (see figure 2).*

Pre-processing : *Perform some local consistency inferences (arc and path consistency algorithms) on the numeric and symbolic levels in order to reduce the size of the problem.*
Figure 3 presents the temporal network of our example after applying the pre-processing step i.e path consistency on the symbolic relations and arc consistency on the numeric constraints (SOPOs).

Backtrack search : *Perform a backtrack search algorithm in order to look for a single solution. This can be performed as follows :*

Choose a node and instantiate the corresponding variable to an occurrence belonging to its domain. Discard from its domain the remaining values and run some consistency algorithms in order to restore consistency. If the network succeeds, fix an occurrence on another variable and run again the local consistency algorithms until each occurrence is fixed on the domain of each variable of the network. We obtain a solution corresponding to the set of the occurrences fixed on the domain of each variable. If the network does not succeed at some point, backtrack and choose another occurrence on the domain of the last selected variable.

4. Solving scheduling problems with non preemptive tasks

Consider the following scheduling problem :

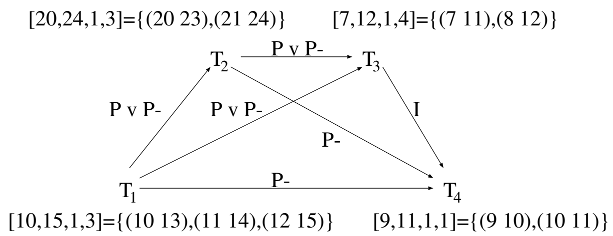
Three tasks T_1, T_2 and T_3 are processed by a mono processor machine. A task T_4 must be processed before T_1 and T_2. Each task is characterized by a processing time, an earliest begin time and a latest end time (see table 3). We assume that the tasks are non pre-emptives.

The above example can easily be translated into a TemPro network. Indeed, the different tasks can be represented by temporal events where the SOPOs define the numeric

Tasks	Earliest begin time	Latest end time	duration
T_1	10	15	3
T_2	20	24	3
T_3	7	12	4
T_4	9	11	1

Table 3. Numeric information of the different tasks

temporal information of the tasks and the qualitative constraints indicate the relative temporal position of a task compared to another. For example, $T_1 \quad P \vee P^{\smile} \quad T_2$ represent the fact that the tasks T_1 and T_2 are executed in mutual exclusion. Figure 5 describes the representation of the above example by the model TemPro.



I : The universal relation(disjunction of the 13 basic Allen relations).
P : Precedes, P- : precedes inverse.

Figure 4. TemPro representation of a scheduling problem.

Since the Allen primitives used in the case of scheduling problems are mainly P and $P^{\smile}$, and giving the fact that the domain intervals of each event are sorted in increasing order, we use the following property for improving the efficiency of the arc consistency algorithm AC-3 :

$$[P^{\smile} \in R_{ij} \wedge intv_{ik}\; P^{\smile}\; intv_{jl}] \Rightarrow$$
$$[\forall m > k \;:\; intv_{im}\; P^{\smile}\; intv_{jl}]$$

where $intv_{ik}$, $intv_{im} \in SOPO_i$ and $intv_{jl} \in SOPO_j$. When the AC-3 process is applied to the edge (i, j), the above property enables us to avoid checking all intervals $intv_{im} \in SOPO_i$(where $m > k$) if a certain interval $intv_{ik} \in SOPO_i$ is supported by $intv_{jl} \in SOPO_j$(within the relation $P^{\smile}$).
Thus, we have implemented in the case of scheduling problems the function $revise$ in the following way :

```
Function   revise(i, j)
Begin
 revise ← false
 For each interval intv_ik ∈ SOPO_i Do
  If ¬compatible(R_ij, intv_ik, intv_jl)
   for all intv_jl ∈ SOPO_j Then
   remove intv_ik from SOPO_i
   revise ← true
  Else
   If P⌣ ∈ R_ij
        ∧ compatible(P⌣, intv_ik, intv_jl) Then
    return revise
  (do not check the rest of the
      intervals in SOPO_i)
  End If
 End Do
End
```

5. Solving temporal constraint problems in real time

In the real world, a main challenge for temporal constraint problems like scheduling and planning applications is the availability to run on-line, that is to give a solution quickly enough so that the system can use it without missing the deadline of the first events. However, most of the temporal problems are NP-Hard. Heuristic and approximation algorithms are in general used for large problems[21][23] and, obviously, are of interest when they provide near optimal solutions with a polynomial computational complexity. In our work, we have developed a complete method, which, in contrast to heuristic methods, can provide solutions to a given problem that are guaranteed to be optimal. In our work, optimality refers to the number of satisfied constraints. The method is based on partial constraint satisfaction techniques[6][28], capable of solving temporal constraint problems by giving a solution with a quality depending on the time allocated for computation. It consists in using a branch and bound variant of the search algorithm in order to satisfy a maximum number of constraints. We have then transformed the propagation techniques we have seen in the previous section, as follows:

1. In the pre-processing phase of the resolution method, either than performing arc consistency algorithms to remove some values that do not belong to any solution, we use direct arc consistency algorithms[5][28]to count the number of inconsistencies counts associated with each value (number of domains that offer no support for that values). Note that, with direct arc consistency algorithms, checking is unidirectional so inconsistency counts are non-redundant.
   In order to perform direct consistency checking, a variable ordering should be first established. The ordering heuristic we use consists in sorting variables by decreasing number of constraints shared with those already instantiated.
2. In the search phase, a cost function corresponding to the number of violated constraints is associated with each path of the search tree. The branch and bound algorithm starts by setting an upper bound value on the cost function. Search is then performed down on each path until all variables are instantiated and in this case a new lower bound is found, or when a partial assignment of values to variables is at least as great as the actual lowest cost.

In order to evaluate our algorithm, we have performed some experiments on temporal constraint problems randomly generated with complete solutions. The experiments were performed on a SUN SPARC Ultra 5 station. All the procedures are coded in C/C++. The temporal problems are randomly generated as follows:

Randomly pick n (n is the number of variables of the problem to generate) pairs (x, y) of integers such that $x < y$ and $x, y \in [0, \ldots, Horizon]$ ($Horizon$ is the parameter before which all events must be processed). This set of n pairs forms the initial solution where each pair corresponds to a time interval.

Metric constraints are generated as follows:

For each interval (x, y) randomly pick an interval contained within $[0..Horizon]$ and containing the interval (x, y). These newly generated intervals define the SOPO of the corresponding variable.

Qualitative constraints are obtained as follows :

Compute the basic Interval-Interval relations that can hold between each interval pair of the initial solution. Add to each relation a random number in the interval $[0, Nr]$ $(1 < Nr \leq 13)$ of chosen basic Interval-Interval relations.

The generated problems can be characterized by their tightness, which can be measured, as shown in [22], using the following definition :

> *The tightness of a CSP problem is the fraction of all possible pairs of values from the domain of two variables that are not allowed by the constraint.*

The tightness depends in our case on the parameters $Horizon, Nr$ and the density of the problem ($\frac{2 \times \text{Number of constraints}}{N \times N-1}$).

The “anytime curve” (see figure 5) is based on the average number of constraint violations found after each period of time. The curve reports tests performed on 100 problems, of size 100 each, with complete solutions.

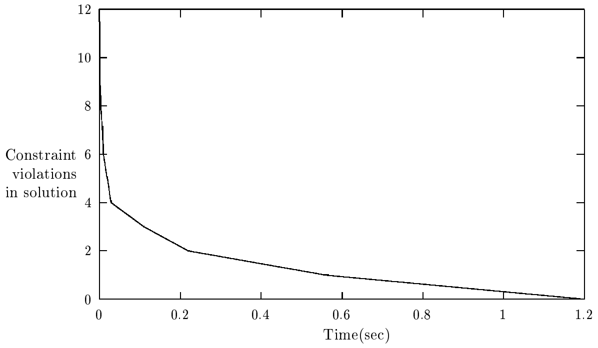


Figure 5. Averaged anytime curve. 100-variable problems, tightness=0.10

For small size problems, our method provides interesting results. However for large problems, the algorithm becomes impracticable and heuristic methods are then the procedures of choice. In this case, our method can be used to evaluate the goodness of results returned by heuristic methods.

6. Conclusion and Future Work

We have presented in this paper our model TemPro that extends the interval algebra approach in order to efficiently take into account numerical constraints. The expressiveness of the representation used in TemPro is more general than those relying on point based representation, such as the numerical representation used in the Dean's TMM, in Meiri's approach, or in IxTeT. The additional expressiveness of TemPro makes it possible to deal with application problems that can be addressed by point based approaches. As a drawback, our approach belongs to a class of problem that is more complex in terms of computation.
Our approach is based on constraint satisfaction over discrete domains. Since solving constraint satisfaction problems is a NP-hard task, we have presented in this paper a policy for finding a single solution in a TemPro network. This policy is a combination of filtering and search algorithms running at the symbolic and/or at the numerical level and translations methods from the symbolic to the numeric levels. TemPro has also the ability to run in real time, that is to provide a solution of a temporal constraint problem with a certain quality and without missing a given deadline.

TemPro is currently used in our Group to address applications such as situation interpretation in the domain of robotics and scheduling.

In the near future, we intend to integrate our model TemPro into a planning system. Indeed, Planning systems based on the state space approach, for example, use a sequential notion of time (systems like TWEAK[3] and SNLP[15] are based on a ordering relation between operators). Our goal is to enrich the time representation of the planning system especially by handling numeric and symbolic constraints. A problem to plan will include, in addition to the implicit temporal relations (orderings generated during the planning process to solve some resource conflicts), a set of explicit temporal constraints. One of the main roles of the planner is to maintain, during the construction of the plan, the consistency of the latter given the temporal constraints. The resolution techniques we have developed will then be used to perform the temporal constraint propagation.

Another perspective consists in solving temporal constraint problems in a dynamic environment. We may, for example, add new information or relax some constraints when there are no more solutions. In those cases we need to check if there still exist solutions to the problem every time a constraint has been added or removed. Adding temporal constraints can easily be handled by TemPro, we have just to put in this case the new constraint in the lists of constraints to be checked. However, constraint relaxation can not be handled by our techniques. Indeed, when we remove a constraint, our algorithms cannot find which value, that has been already removed, must be put back and which one must not. We must then transform our satisfaction techniques so that they will be able to save the reason why a certain value has been removed.

References

[1] J. Allen. Maintaining knowledge about temporal intervals. *CACM*, 26(11):832–843, 1983.

[2] C. Bessière, A. Isli, and G. Ligozat. Global consistency in interval algebra networks : tractable subclasses. In *ECAI'96*, Budapest, Hongrie, 1996.

[3] D. Chapman. Planning for conjunctive goals. *Artificial Intelligence.*, 32:333–377, 1987.

[4] T. L. Dean and D. V. McDermott. Temporal database management. *Artificial Intelligence*, 32:1–55, 1987.

[5] R. Dechter and J. Peal. Network-based heuristics for constraint satisfaction problems. *Artificial Intelligence*, 34:1–38, 1988.

[6] E. C. Freuder and R. J. Wallace. Partial constraint satisfaction. *Artificial Intelligence*, 58:21–70, 1992.

[7] M. Ghallab and T. Vidal. Focusing on a Sub-graph for Managing Efficiently Numerical Temporal Constraints. In *In Proc. FLAIRS'95*, 1995.

[8] H. Kautz and P. Ladkin. Integrating metric and qualitative temporal reasoning. In *AAAI'91*, pages 241–246, Anaheim, CA, 1991.

[9] A. Koomen. The timelogic temporal reasoning system in common Lisp. *Technical Report TR231, Universit de Rochester*, pages 241–246, 1987.

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

[11] P. B. Ladkin. Models of axioms for time intervals. In *AAAI'87*, pages 234–239, 1987.

[12] G. Ligozat. On generalized interval calculi. In *AAAI'91*, pages 234–240, Anaheim, CA, 1991.

[13] A. K. Mackworth. Consistency in networks of relations. *Artificial Intelligence*, 8:99–118, 1977.

[14] A. K. Mackworth and E. Freuder. The complexity of some polynomial network-consistency algorithms for constraint satisfaction problems. *Artificial Intelligence*, 25:65–74, 1985.

[15] D. McAllester and D. Rosenblitt. Systematic nonlinear planning. In *AAAI-91*, pages 634–639, 1991.

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

[17] U. Montanari. Fundamental properties and applications to picture processing. *Information Sciences*, 7:95–132, 1974.

[18] M. Mouhoub, F. Charpillet, and J. Haton. Experimental analysis of numeric and symbolic constraint satisfaction techniques for temporal reasoning. *Constraints: An International Journal*, 2:151–164, June 1998.

[19] B. Nebel and H. Burckert. Reasoning about temporal relations: A maximal tractable subclass of Allen's Interval Algebra. *J. ACM*, 42:43–66, 1995.

[20] J. F. Rit. Modélisation et propagation de contraintes temporelles pour la planification. *Thèse de Doctorat, INPG*, 1988.

[21] A. B. P. S. Minton, M. D. Johnston and P. Laird. Minimizing conflicts: a heuristic repair method for constraint satisfaction and scheduling problems. *Artificial Intelligence*, 58:161–205, 1992.

[22] D. Sabin and E. C. Freuder. Contradicting conventional wisdom in constraint satisfaction. In *Proc. 11th ECAI*, pages 125–129, Amsterdam, Hollande, 1994.

[23] B. Selman and H. A. Kautz. An empirical study of greedy local search for satisfiability testing. In *AAAI'93*, pages 46–51, 1993.

[24] R. E. Valdéz-Pérez. The satisfiability of temporal constraint networks. In *IJCAI'89*, pages 1291–1296, Menlo Park, Calif, USA, 1989.

[25] P. van Beek. Reasoning about qualitative temporal information. *Artificial Intelligence*, 58:297–326, 1992.

[26] P. van Beek and D. W. Manchak. The design and experimental analysis of algorithms for temporal reasoning. *Journal of Artificial Intelligence Research*, 4:1–18, 1996.

[27] M. Vilain and H. Kautz. Constraint propagation algorithms for temporal reasoning. In *AAAI'86*, pages 377–382, Philadelphia, PA, 1986.

[28] R. J. Wallace. Partial constraint satisfaction. *Lecture Notes in Computer Science*, 923:121–138, 1995.