
Constraints: An International Journal, 2, 151–164 (1998)

Experimental Analysis of Numeric and Symbolic Constraint Satisfaction Techniques for Temporal Reasoning

MALEK MOUHOUB mouhoub@loria.fr
LORIA, BP 239, 54506 Vandœuvre-lès-Nancy, France

FRANÇOIS CHARPILLET charp@loria.fr
LORIA, BP 239, 54506 Vandœuvre-lès-Nancy, France

JEAN PAUL HATON jph@loria.fr
LORIA, BP 239, 54506 Vandœuvre-lès-Nancy, France

Abstract. Many temporal applications like planning and scheduling can be viewed as special cases of the numeric and symbolic temporal constraint satisfaction problem. Thus we have developed a temporal model, TemPro, based on the interval Algebra, to express such applications in term of qualitative and quantitative temporal constraints. TemPro extends the interval algebra relations of Allen to handle numeric information. To solve a constraint satisfaction problem, different approaches have been developed. These approaches generally use constraint propagation to simplify the original problem and backtracking to directly search for possible solutions. The constraint propagation can also be used during the backtracking to improve the performance of the search. The objective of this paper is to assess different policies for finding if a TemPro network is consistent. The main question we want to answer here is "how much constraint propagation is useful" for finding a single solution for a TemPro constraint graph. For this purpose, we have experimented by randomly generating large consistent networks for which either arc and/or path consistency algorithms (AC-3, AC-7 and PC-2) were applied. The main result of this study is an optimal policy combining these algorithms either at the symbolic (Allen relation propagation) or at the numerical level.

Keywords: temporal reasoning, constraint satisfaction, arc consistency, path consistency

1. Introduction

Time plays an important role in various applications 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 A.I. is Allen's interval algebra. A significant amount of work has been devoted to this approach, e.g. [10], [21], [22], [23], [24], [25], [26] and so on. Most work has been devoted to the complexity analysis and to the definition of sub-classes of the interval algebra that admit complete polynomial algorithms for constraints maintenance [4]. Some generalisation 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 representation such as the numerical representation used in the Dean's TMM [6], Meiri's approach [13] or in IxTeT [7]. As a drawback it is more complex and remains costly. Furthermore it does not take into account numerical constraints unless restrictions are accepted [10], [17].

In order to address these two drawbacks of interval algebra, we have defined in our

model TemPro a representation and algorithms that extends the interval algebra approach in order to take into account efficiently numerical constraints. In addition to interval algebra relations, TemPro model handles different kinds of entities such as windows, durations for intervals and dates defined over window's bounds (that can be defined in form of mathematical equations). The algorithmic approach is that of constraint satisfaction over discrete domains.

Since solving constraint satisfaction problems is a NP-hard problem, filtering techniques, such as arc and path consistency which simplify the task by eliminating some local inconsistencies, are particularly studied. We present in this paper the use of these techniques to solve constraint satisfaction temporal problems. The main question we want to answer is how much constraint propagation is useful for finding a single solution for a constraint satisfaction temporal problem. For this purpose, we have compared different policies where arc and/or path consistency algorithms were applied for finding if a constraint satisfaction temporal problem is consistent. The main result of this study is an optimal policy combining these algorithms either at the symbolic or at the numerical level.

This paper is organised as follows. In sections two and three, we present our model and propagation algorithms. In the fourth section, our model is compared to other approaches. Sections five and six present the experimental results.

2. Knowledge Representation

2.1. Events

In TemPro, temporal objects are called events. Events have a uniform reified representation made up of a proposition and its temporal qualification: $Evt = OCCUR(p, I)$ defined by Allen [1] and denoting the fact that the proposition p occurred over the interval I. For the sake of notation simplicity, an event is used to denote its temporal qualification in this article.

2.2. Qualitative Constraints

Qualitative constraints specify the relative temporal position of an event with respect to other events. The qualitative constraint between two events ev_1 and ev_2 can take the following forms : $ev_1 = OCCUR(p, I_1)$, $ev_2 = OCCUR(p, I_2)$; $I_1\ r_1\ r_2 \ldots r_n\ I_2$, where each of the r_i's is one of the thirteen Allen primitives: *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.

2.3. Quantitative Constraints

Additional information about an event can be stated, thus restricting its temporal qualification to belong to a given SOPO i.e. the Sets Of Possible Occurrences where the given event

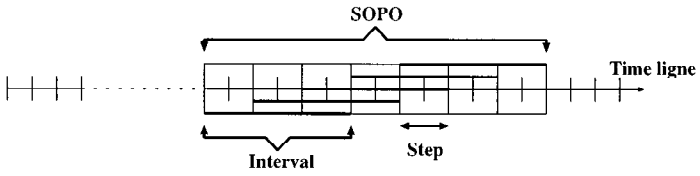

Figure 1. The SOPO of a given event.

can take place. In our case, a SOPO is restricted to be a finite set of intervals with constant duration. We represent a SOPO 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, e.g, $event_1 = (P, I) \wedge I \in [12, 88, 4, 4]$. Thus, if e_i is an event numerically constrained by the following SOPO $[inf_i, sup_i, d_i, p_i]$, then the set of possible occurrences of e_i is defined as:

$$I = \{occ_j \mid begin(occ_j) = inf_i + k * p_i, end(occ_j) = begin(occ_j) + d_i * p_i, end(occ_j) <= sup_i,\ k \in \left[0, \frac{sup_i - inf_i}{p_i} - d_i\right] \cap \aleph\}.$$

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

2.4. Related Work

Some research work addressing the problem of handling both metric and symbolic constraints has already been reported. Meiri [13] 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 [9] have proposed a model allowing 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. Indeed, quantitative constraints are represented by linear inequalities in the Kautz and Ladkin approach while they are modeled by SOPOs in our representation. Thus we have 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 vise 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 minimal Allen and metric constraints network is performed in one pass. A backtracking algorithm is then used to search for a possible numeric solution.

3. Solving Qualitative and Metric Constraint Temporal Problems

3.1. Constraint Satisfaction Problems

Let N be the set of variables x_i, each defined on a discrete domain $\{a_{i1}, \ldots, a_{in}\}$ and R a set of constraining relations on a subset of these variables. A constraint satisfaction problem (CSP) consists of finding all sets of values $\{a_{1j1}, \ldots, a_{njn}\}$ for $(x_1, \ldots, x_n\}$ satisfying all relations belonging to R. The network $G = (N, R)$ characterising the CSP is usually a graph in which the vertices represent variables, and edges represent relations. Since CSP is a NP-complete problem, algorithms assuming only local consistency have been developed. Such algorithms transform the network of constraints G into an equivalent and simpler network G' by removing from the domain of each variable some values that cannot 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.

3.2. TemPro based Temporal Constraint Satisfaction

In the previous section, we have demonstrated how both qualitative and metric constraints could be mapped into a network-based representation. The network involves: a set of variables $\{EV_1, \ldots, EV_n\}$, each defined on a discrete domain D_i standing for the set of possible occurrences in which the corresponding event can hold, and a set of binary constraints, each representing a qualitative disjunctive relation between a pair of events. An example of such a constraint network is shown in figure 2. The example illustrates the problem of scheduling a set of non preemptive tasks over a single processor. We are given the following information. 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 characterised by a processing time, an earliest begin time and a latest end time (which are respectively 3, 10 and 15 for T_1).

Before giving the consistency checking algorithms and their experimental evaluation, let us present the notions of arc-consistency and path-consistency in TemPro.

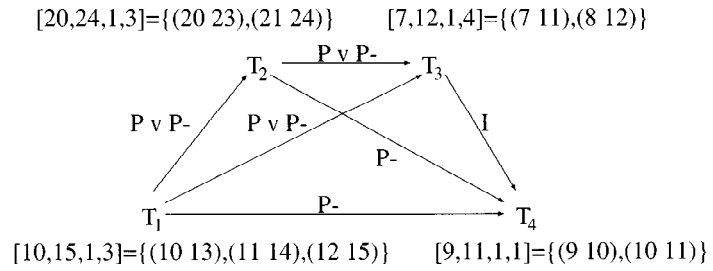

Figure 2. TemPro representation of a scheduling problem.

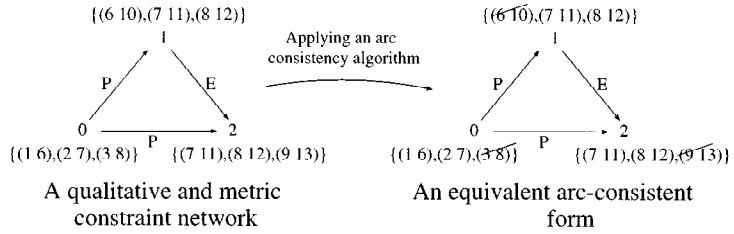

Figure 3. Arc-consistency in TemPro.

3.3. 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 occ_i belonging to D_i, there exists a value occ_j belonging to D_j such that the qualitative disjunctive relation R_{ij} of the pair of variables (EV_i, EV_j) is satisfied. We say then that occ_i belonging to D_i is said to be supported by occ_j belonging to D_j. The network can be converted into an equivalent arc-consistent form by applying an arc-consistency algorithm (cf. figure 3).

Arc consistency techniques have been widely studied. In a previous work [19], we have applied AC-4 [15] for metric constraint propagation in TemPro. But its space complexity and average time complexity suggest returning to AC-3, since the latter has been reported [27] to be more efficient than AC-4, despite its non-optimal time complexity. To improve the average time complexity of AC-4 while keeping its worst case complexity, Bessière [2] proposed AC-6. AC-6 eliminates the problem of space complexity of AC-4 (AC-6 space complexity is $O(ed)$ while it is $O(ed^2)$ for AC-4) and checks just enough amount of constraints to compute the arc-consistent domain. More recently, Bessière and its co-authors [3] proposed an enhancement of AC-6 called AC-7. AC-7 uses a meta-level knowledge to infer support in order to reduce constraint checking traditionally used by the other arc con-

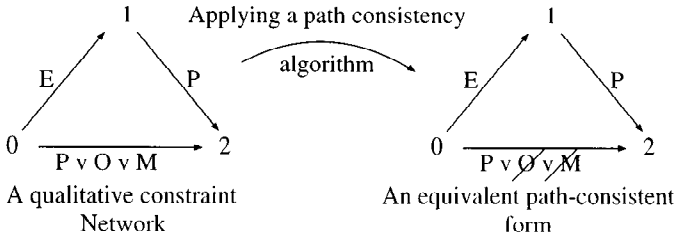

Figure 4. Path consistency.

sistency algorithms to establish support. In fact, using a general property of constraints, i.e bidirectionality, AC-7 eliminates constraint checking that other arc consistency algorithms perform (AC-3, AC-4 and AC-6). In this paper, we will focus on the assessment of AC-3 and AC-7 when applied and customized to TemPro networks.

To improve the efficiency of AC-3 and AC-7, we have modified these algorithms by considering specific properties of TemPro networks :

- the lists of all pairs to be checked are maintained in a decreasing order during the process. Thus, the time requirement to check whether the pair of nodes to be tested already exists in the list, is reduced.

- As the occurrences (intervals) of the domains of each variable are ordered in increasing order (see the definition of a SOPO in subsection 2.2.3), we use the following property for improving the efficiency of the arc consistency algorithm AC-3 : $[P^{\smile} \in R_{ij} \wedge occ_{ik}\ P^{\smile}\ occ_{jl}] \Rightarrow \forall m > k\ \ occ_{im}\ P^{\smile}\ occ_{jl}$, where occ_{ik}, $occ_{im} \in D_i$ and $occ_{jl} \in D_j$. When the AC-3 process is applied to the edge (i, j), the above property enables us to avoid checking all occurrences $occ_{im} \in D_i$ if a certain occurrence $occ_{ik} \in D_i$ (where $m > k$) is supported by $occ_{jl} \in D_j$ (within the relation $P^{\smile}$).

3.4. Symbolic Path-Consistency of a TemPro Network

The 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) $(r \in R_{ik} \ \Rightarrow\ r \in R_{ij} \circ R_{jk})$ 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. Montanari [16] showed that 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 (cf. figure 4). We could as well apply path consistency reduction at the numerical level, but preliminary experiments led us to give up this approach because the additional cost of path consistency compared to arc-consistency is not out weighted by the speed up of the search phase.

Among path consistency algorithms, we will focus on the assessment of PC-2 when applied 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 stabilised 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 stabilises. This is the principle of PC-1 [11].

In fact, only the triangle of edges (M_{ij} entries) whose labels have changed in the previous iteration need to be recomputed. Thus, we use (as proposed by Mackworth [14] 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 [25]. To improve the efficiency of the algorithm PC-2, we changed as reported by [5] the way composition-intersection of relations are achieved during the path consistency process (following the idea "one support is sufficient").

4. Finding a Single Solution

Given a TemPro constraint-based network, one of the interesting reasoning tasks is to determine its consistency and therefore to find a solution of the network. Deciding consistency is in this case NP-hard. In order to find how much constraint propagation is useful for finding a single solution to a TemPro constraint graph we have defined a TemPro Consistency Algorithm (TCA), given in table 1, based on the two following steps:

1. *Perform some consistency inferences in a single preprocessing pass. Let CI1 be this algorithm.*

2. *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 algorithm in order to restore consistency. Let CI2 be this algorithm. If the network succeeds, fix an occurrence on another variable and run CI2 again until each occurrence is fixed on the domain of each variable of the network. We then 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 variable selected.*

TCA is equivalent to the algorithm proposed by Ladkin and Reinefeld [11] for networks in Allen temporal calculus if CI1 and CI2 are set to the path-consistency algorithm PC-1. TCA is equivalent to MAC [18] if both CI1 and CI2 are set to AC-4. Arc (AC-3, AC-7) and symbolic path (PC-2) consistency algorithms defined in the previous section can be applied both at pre-processing and search phases.

The next section reports experimental results obtained by randomly generating large consistent networks for which either arc or path consistency algorithms (AC-3, AC-7 and PC-2) were applied.

Table 1. TemPro Consistency Algorithm (TCA).

```
1.  Main
2.      Begin
3.          if PC() then
4.              if AC(nr_ac, etiq_node) then
5.                  for each e_i ∈ etiq_node[1] do
6.                      If Search_AC(nr_ac, etiq_node, 1, e_i) then
7.                          return "Consistent"
8.                      end
9.                  end
10.                 return "Path and Arc consistent but not Consistent"
11.             else
12.                 return "Path Consistent but not Arc Consistent"
13.             end
14.         else
15.             return "Not Path Consistent"
16.         end
17.     End

1.      Function Search_AC (nr_ac, etiq_node, i, e)
2.              /* nr_ac: number of the AC algorithm to apply, i: index of the current node
3.                 etiq_node: array containing the domain of the variables, e: an interval */
4.      Begin
5.          etiq_node_save ← etiq_node                          /* save etiq_node */
6.          if length(etiq_node[i] = 1) or AC (nr_ac, etiq_node) then
7.              if i = n then
8.                  result_num[i] ← e
9.                  return "true"
10.             end
11.         else
12.             for each e_j ∈ etiq_node[i + 1] do
13.                 if Search_AC(nr_ac, etiq_node, i + 1, e_j) then
14.                     result_num[i] ← e
15.                     return "true"
16.                 end
17.             end
18.         end
19.         etiq_node ← etiq_node_save                          /* restore etiq_node */
20.         return "false"
21.     End
```

5. Experimental Design

Our experiments concern problems 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 the tasks 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 $[0, Nr]$, *of chosen basic Interval-Interval relations.*

CSP problems can be characterised by their tightness, which could be measured, as shown in [18], 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 of a problem is proportional to the number of calls of the arc consistency algorithm during the backtrack search phase of the TCA algorithm. Indeed, for the problems having large values of tightness, called *tighter problems* [20], the search space is reduced to the numerical solution only after a few calls of the arc consistency algorithm. For the problems having small values of tightness, called *looser problems*, the solution is reached after many calls of the arc consistency algorithm.

All the tests presented in figures 5 and 6 have been performed on a Sun SparcStation 20 with 32 MBytes of Memory. We generated for each experiment 1000 complete graphs. The tests presented in figure 5a and 5d are executed on consistent problems randomly generated as shown before. The average value of the tightness characterising the problem generated is obtained, for each experiment, given a fixed value of *Horizon*, *step* and the number *Nr*. The tests presented in figure 5c are carried out on inconsistent problems generated as shown before but omitting the first step. Figure 5a and 5c show the average time in seconds required respectively to obtain a single solution and to detect inconsistencies of networks of size 100 and 200, respectively. Figure 5b presents the variance corresponding to the tests presented in figure 5a. The tests shown in figure 5d present the average time needed to obtain a single solution by each of the following algorithms: backtracking (BT), forward checking (FL), full lookahead (FL) and really full lookahead (RFL). These algorithms differ in the degrees of the arc consistency algorithm AC-3 performed at the nodes of the temporal constraint graph in the second phase of the TCA algorithm.

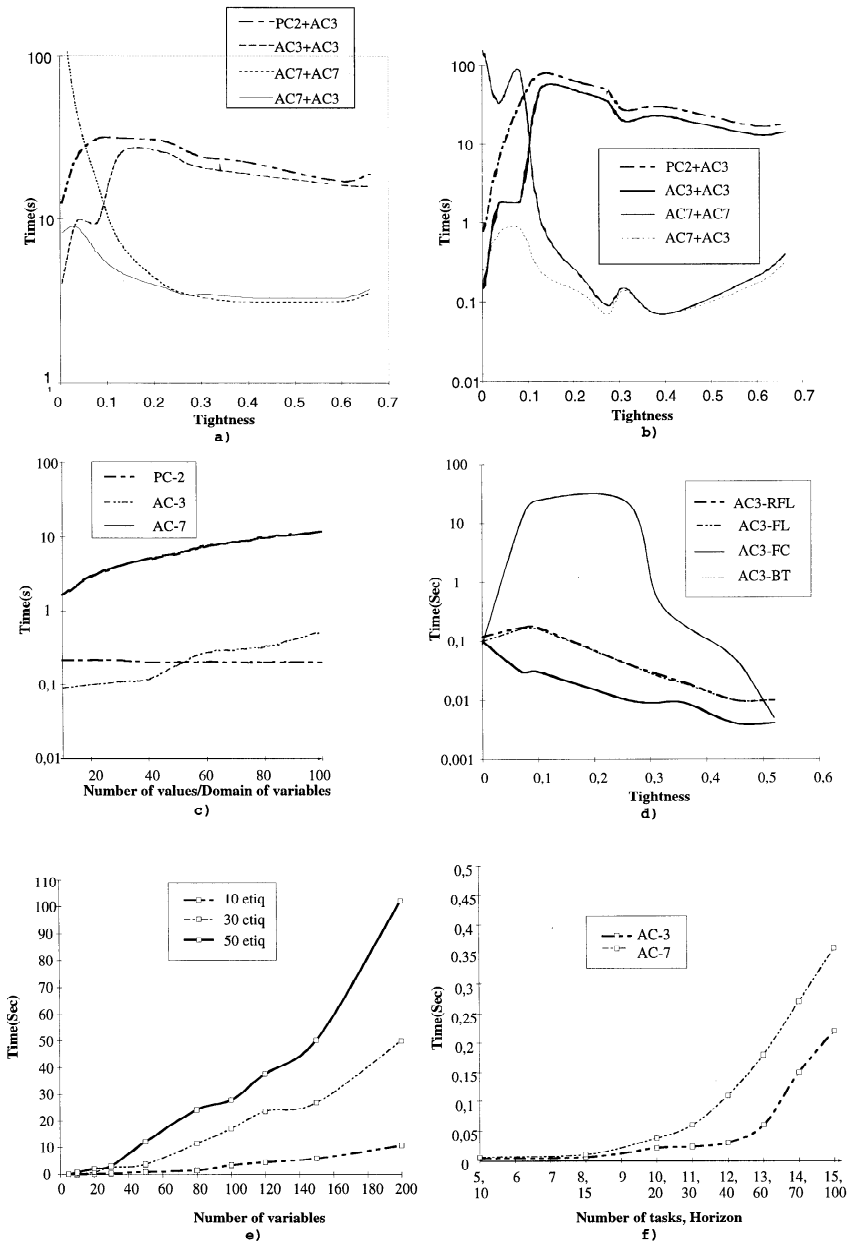

Figure 5. Experimental tests on numeric and symbolic temporal problems.

6. Discussion

While the path consistency technique is effective in reducing the number of atoms (basic relations) per qualitative constraint in the preprocessing phase of the algorithm, this tech-

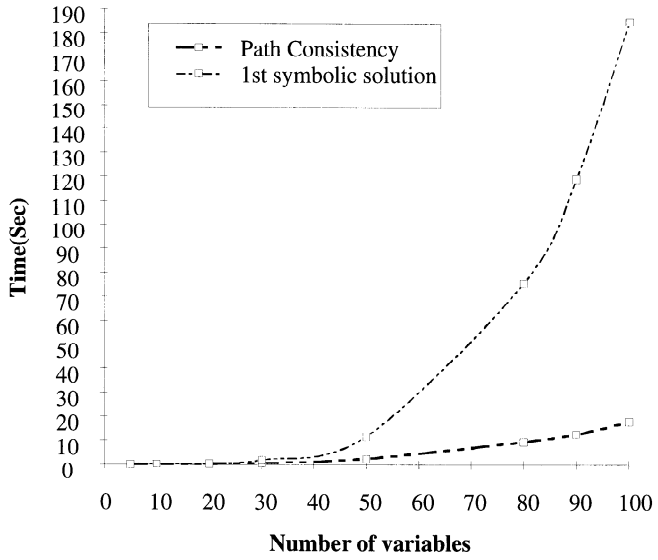

Figure 6. Experimental tests on symbolic temporal problems.

nique does not improve the global time needed to find a solution. The time performance of the (PC-2 + AC-3) + AC-3 method (see figure 5a) is affected by path consistency calculations, and, therefore, this method presents the worst results (in general) when compared to methods using only an arc consistency algorithm, on quantitative constraints in the preprocessing phase. However, the path consistency technique is useful to detect inconsistencies. The time performance of the algorithm PC-2 to detect inconsistencies (see figure 5c) are better than for the arc consistency algorithms AC-3 and AC-7 for large number of values per domain of variable.

When comparing the methods AC-7 + AC-3 and AC-3 + AC-3 (fig 5a), we conclude that AC-7 is better than AC-3 in the preprocessing phase of the algorithm (except for looser problems where a method which uses a problem reduction algorithm like AC-7 before search is likely to spend effort unnecessarily in attempting to reduce the problem. Indeed, for looser problems many leaves of the search space represent solutions). However, while AC-7 is better than AC-3 in the preprocessing phase (taking advantage of the bidirectionality), time performance of the method using AC-7 in the backtrack search phase, namely AC-7 + AC-7, decreases for looser problems. We observe that methods using AC-3, namely AC-7 + AC-3 and AC-3 + AC-3, present better performance in this case. Indeed, for looser problems, the solution is obtained after many calls of the arc consistency algorithm in the backtrack search phase, and the overhead of memory during the backtrack search is a disadvantage of AC-7.

In figure 5d, the strategy FC presents the best performance when compared to the other strategies. We conclude, as reported by [8], that it is better to apply constraint propagation only in a limited form.

Figure 5e shows the performance of the TCA algorithm using the optimal method (AC-7 in preprocessing and AC-3 in the search phase with the forward-check strategie) to solve temporal constraints problems randomly generated. The results demonstrate the applicability of the TCA algorithm to problems with a number of variables less then 200.

The figure 5f concerns experiments on randomly generated problems of scheduling a set of non pre-emptive tasks over a single processor. For every problem, each task is characterised by a processing time, an earliest begin time (the time before which the task cannot start) and a latest end time (the time before which the task must complete). As shown before (see 3.2), this kind of problems can be easily translated into TemPro networks. Indeed, the tasks are represented by the nodes of the constraint network, qualitative constraints are then the disjunctive relations: $P \vee P^{\smile} \vee M \vee M^{\smile}$ while quantitative constraints are the SOPOs representing the characteristics of each task. The figure presents the time needed by each of the two methods AC7 + AC3 and AC3 + AC3 to obtain a single solution (consistent scheduling). As shown in the figure, AC3 is better than AC7. This is because of the property we have defined in subsection 3.3 to improve the efficiency of the algorithm AC3 and which permit the reduction of the number of consistency check when the Allen primitive P is present.

Figure 6 presents the CPU time needed to obtain the path consistency and the first symbolic solution on randomly generated symbolic constraint temporal problems. As we remark, the time required to get the first solution is much greater than the time needed to perform the path consistency. This is in contrast to the numeric solution search where the time needed in the backtracking phase is smaller than the time required to perform the local consistency in the first phase of the algorithm. This is because in the case of symbolic solution search, we fix in each step of the search an atom on each arc of the graph while in the case of numeric solution search an interval is fixed on each node of the graph. It should be noted that the number of arcs in a complete graph (after applying path consistency, the graph becomes complete) of nodes is $\frac{n*n-1}{2}$, which is evidently much large than the number of nodes.

7. Conclusion

We have presented our model TemPro designed for handling a representation and algorithms that extend the interval algebra approach in order to efficiently take into account numerical constraints. The expressiveness of 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, Meiri's approach or in IxTeT. The additional expressiveness of TemPro enables us to deal with application problems that can be addressed by point based approaches. As a drawback, our algorithmic approach belongs to a class of problem that is more complex in term of computation.

Our approach is based on constraint satisfaction over discrete domains. Since solving constraint satisfaction problems is a NP-hard problem, we have defined an optimal policy on average 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:

- path consistency is very expensive to find a single numeric solution but is useful to detect inconsistencies of TemPro networks,
- arc consistency, in a limited form (following the principle of the forward-check) during the numeric solution search, outperforms other approaches,
- AC-3 and AC-7 have complementary application domain depending on the complexity of the problem.

	under constrained problems	middle and over constrained problems	scheduling problems
AC-3	preprocessing and backtrack search	backtrack search phase	preprocessing and backtrack search
AC-7	—	preprocessing phase	—

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

References

1. J. F. Allen. (1983). Maintaining knowledge about temporal intervals. *CACM* 26: 832–843.
2. C. Bessière. (1994). Arc-consistency and arc-consistency again. *Artificial Intelligence* 65: 179–190.
3. C. Bessière, E. Freuder, & J. C. Regin. (1995). Using inference to reduce arc consistency computation. *IJCAI'95*, Montréal, Canada, pages 592–598.
4. C. Bessière, A. Isli, & G. Ligozat. (1996). Global consistency in interval algebra networks: Tractable subclasses. *ECAI'96*, Budapest, Hongrie.
5. C. Bessière. (1996). A simple way to improve path consistency processing in interval algebra networks, *AAAI'96*, Portland, 375–380.
6. T. L. Dean & D. V. McDermott. (1987). Temporal database management. *Artificial Intelligence* 32: 1–55.
7. M. Ghallab & A. Mounir-Alaoui. (1989). Managing efficiently temporal relations through indexed spanning trees. *Proc. IJCAI'89*, Menlo Park, Calif, USA, pages 1297–1303.
8. R. M. Haralick & G. L. Elliott. (1980). Increasing tree search efficiency for constraint satisfaction problems. *Artificial Intelligence* 14: 263–313.
9. H. A. Kautz & P. B. Ladkin. (1991). Integrating metric and qualitative temporal reasoning. *AAAI-91*, Anaheim, CA, pages 241–246.
10. A. Koomen. (1987). The timelogic temporal reasoning system in common lisp. Technical Report TR231, Universit de Rochester, pages 241–246.
11. P. B. Ladkin & A. Reinefeld. (1992). Effective solution of qualitative interval constraint problems. *Artificial Intelligence* 57: 105–124.
12. G. Ligozat. (1991). On generalized interval calculi. *AAAI91*, Anaheim, CA, pages 234–240.
13. I. Meiri. (1996). Combining qualitative and quantitative constraints in temporal reasoning. *Artificial Intelligence* 87: 343–385.

14. A. K. Mackworth. (1977). Consistency in networks of relations. *Artificial Intelligence* 8: 99–118.

15. Mohr & Henderson. (1986). Arc and path consistency revisited. *Artificial Intelligence* 28: 225–233.

16. U. Montanari. (1974). Fundamental properties and applications to picture processing. *Information Sciences* 7: 95–132.

17. J. F. Rit. (1988). Modélisation et propagation de contraintes temporelles pour la planification. *Thèse de Doctorat à l'INPG*.

18. D. Sabin & E. C. Freuder. (1994). Contradicting conventional wisdom in constraint satisfaction. *Proc. 11th European Conference on Artificial Intelligence*, Amsterdam, Hollande, pages 125–129.

19. H. Tolba, F. Charpillet & J. P. Haton. (1991). Representing and propagating constraints in temporal reasoning. *AI Communications* 4: 145–151.

20. E. Tsang. (1994). *Foundation of Constraint Satisfaction*. Academic Press.

21. R. E. Valdéz-Pérez. (1989). The satisfiability of temporal constraint networks. *IJCAI'89*, Menlo Park, Calif, USA, pages 1291–1296.

22. P. van Beek. (1989). Approximation algorithms for temporal reasoning. *IJCAI'89*, Menlo Park, Calif, USA, pages 1291–1296.

23. P. van Beek. (1990) Reasoning about qualitative temporal information. *AAAI-90*, Boston, MA, pages 297–326.

24. P. van Beek. (1992). Reasoning about qualitative temporal information. *Artificial Intelligence* 58: 297–326.

25. P. van Beek & D. W. Manchak. (1996). The design and experimental analysis of algorithms for temporal reasoning. *Journal of Artificial Intelligence Research* 4: 1–18.

26. M. Vilain & H. Kautz. (1986). Constraint propagation algorithms for temporal reasoning. *AAAI-86*, Philadelphia, PA, pages 377–382.

27. R. J. Wallace. (1993). Why AC-3 is almost always better than AC-4 for establishing arc consistency in CSPs. *IJCAI'93*, Chambery, France, pages 239–245.