Constraints, 2, 151–164 (1998)

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

MALEK MOUHOUB, FRANÇOIS CHARPILLET AND JEAN PAUL HATON

mouhoub@loria.fr charp@loria.fr 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 numeric and symbolic temporal constraint satisfaction problem. 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 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. Constraint propagation can also be used during backtracking to improve the performance of the search. The objective of this paper is to assess different policies for checking if a TemPro network is consistent. The main question we want to answer is how much constraint propagation is useful for finding a single solution for a TemPro constraint graph. For this purpose, we have randomly generated large consistent networks for which 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 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 A.I. is Allen's interval algebra. A significant amount of work has been devoted to this approach, e.g. (Koomen, 87), (Valdéz-Peréz, 87), (van Beek, 89,90,92,96), (Vilain and Kautz, 86). 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 (Nebel and Burckert 95), (Bessière et al, 96). Some generalisation were also studied (Ligozat, 91), (Ladkin, 87), 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 (Dean and McDermott, 87), Meiri's approach (Meiri, 96), or in IxTeT (Ghallab, 89) . As a drawback, interval algebra is more complex and costly. Furthermore, it does not take into account numerical constraints, unless restrictions are accepted (Koomen, 87), (Rit, 88).

In order to address these two drawbacks, we have defined in our model TemPro a representation and the corresponding algorithms that extends 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). The algorithmic approach is the one of constraint satisfaction over discrete domains.

Since constraint satisfaction is a NP-hard problem, 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. The main question we want to answer is how much constraint propagation is useful for finding a single solution for a given problem. For this purpose, we have compared different policies where arc and/or path consistency algorithms are 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 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 (Allen, 83) and denoting the fact that the proposition p occured over the interval I. For the sake of notation simplicity, an event is used in this paper to denote its temporal qualification.

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 expressed, thus restricting its temporal qualification to belong to a given SOPO, i.e., the Sets Of Possible Occurrences where the 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 SOPO $[inf_i, sup_i, d_i, p_i]$, then the set of possible occurrences of e_i is defined as :

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

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

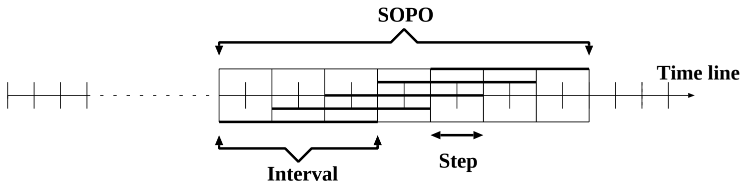

Figure 1. The SOPO of a given event

2.4. Related work

Some research works addressing the problem of handling both metric and symbolic constraints have already been reported. Meiri (Meiri, 96) 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 (Kautz and Ladkin, 91) 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 minimal Allen and metric constraints network is performed in one pass. A backtracking algorithm is then used to search for a possible numerical 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. This example illustrates the problem of scheduling a set of non preemptive tasks over a single processor. More precisely, 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).

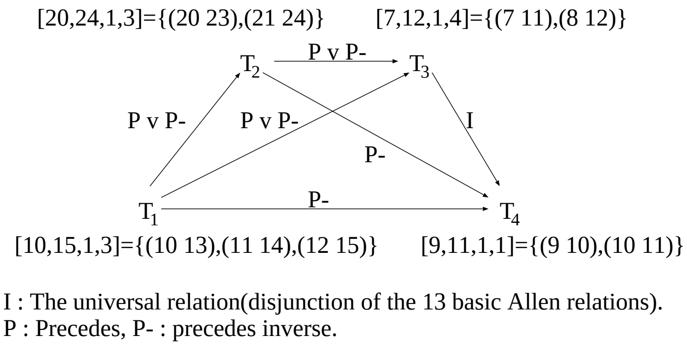

Figure 2. TemPro representation of a scheduling problem.

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

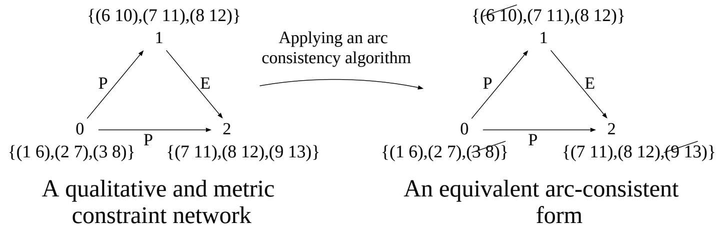

Figure 3. Arc-consistency in TemPro

Arc consistency techniques have been widely studied. In a previous work, we have applied AC-4 (Mohr and Henderson, 86) for metric constraint propagation in TemPro. But the space and average time complexity of this algorithm suggest returning to AC-3, since the latter has been reported (Wallace, 93) 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 (Bessière, 94) 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 constraints to compute the arc-consistent domain. More recently, Bessière and his co-authors (Bessière et al, 95) proposed an enhancement of AC-6 called AC-7. AC-7 uses meta-level knowledge to infer support in order to reduce constraint checking traditionally used by the other arc consistency 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 focus on the assessment of AC-3 and AC-7 applied and customised 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 : $[P^{\smile} \in R_{ij} \wedge occ_{ik}\ P^{\smile}\ occ_{jl}] \Rightarrow \forall m > k \quad occ_{im}\ P^{\smile}\ occ_{jl}$, where $occ_{ik},\ occ_{im} \in D_i$ and $occ_{jl} \in D_j$. When the AC-3 process is applied to 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*

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. Montanari (Montanari, 74) 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 since the additional cost of path consistency compared to arc-consistency is not out weighted by the speed up of the search phase.

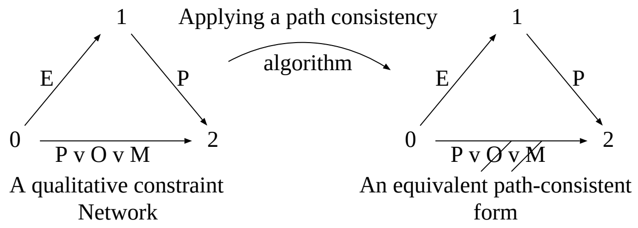

Figure 4. Path consistency

Among path consistency algorithms, we will focus on the assessment of PC-2 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 (Ladkin and Reinefeld, 92).
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 (Mackworth, 77) and Allen (Allen, 83)) 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 (vanBeek and Manchak, 96). To improve the efficiency of PC-2, we changed, as reported by (Bessière, 96), the way compositions-intersections 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 important task of reasoning is to determine the consistency of the network, and therefore to find a solution of this 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. This algorithm is 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*

```
Main
    Begin
        if PC() then
            if AC(nr_ac, etiq_node) then
                for each e_i ∈ etiq_node[1] do
                    If Search_AC(nr_ac, etiq_node, 1, e_i) then
                        return "Consistent"
                    end
                end
                return "Path and Arc Consistent but not Consistent"
            else
                return "Path Consistent but not Arc Consistent"
            end
        else
            return "Not Path Consistent"
        end
    End
```

```
    Function Search_AC (nr_ac, etiq_node, i, e)
    /* nr_ac : order number of the AC algorithm to apply, i : index of the current
node
        etiq_node : array containing the domain of the variables, e : an interval */
    Begin
        etiq_node_save ← etiq_node                          /* save etiq_node */
        if length(etiq_node[i] = 1) or AC (nr_ac, etiq_node) then
            if i = n then
                result_num[i] ← e
                return "true"
            end
        else
            for each e_j ∈ etiq_node[i + 1] do
                if Search_AC(nr_ac, etiq_node, i + 1, e_j) then
                    result_num[i] ← e
                    return "true"
                end
            end
        end
        etiq_node ← etiq_node_save                          /* restore etiq_node */
        return "false"
    End
```

Table 1. The TemPro Consistency Algorithm (TCA).

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 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.

TCA is equivalent to the algorithm proposed by Ladkin and Reinefeld (Ladkin and Reinefeld, 92) for networks in Allen temporal calculus, if CI1 and CI2 are set to the path-consistency algorithm PC-1. TCA is equivalent to MAC (Sabin and Freuder, 94) 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.

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 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 can be measured, as shown in (Sabin and Freuder, 94), 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* (Tsang, 93), the search space is reduced to the numerical solution only after a few calls of the arc consistency algorithm. For 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. For each experiment 1000 complete graphs were generated. The tests presented in figure 5a and 5d were executed on consistent problems randomly generated as shown above. 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 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.

6. Discussion

While the path consistency technique is effective in reducing the number of atoms (basic relations) per qualitative constraint in the pre-processing phase of the algorithm, this technique 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 pre-processing phase. However, the path consistency technique is useful to detect inconsistencies. The time performance of 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.

By 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 (Haralick and Elliot, 80), 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 pre-processing and AC-3 in the search phase with the forward-check

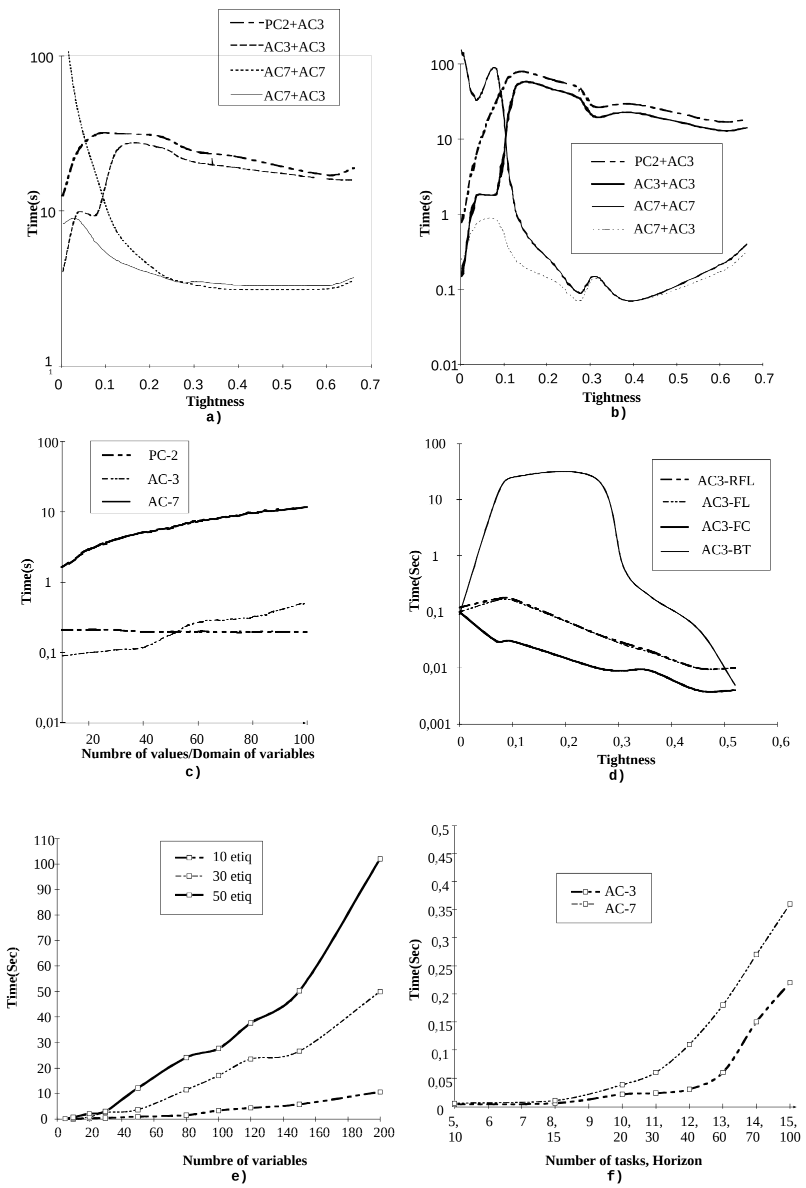

Figure 5. Experimental tests on numeric and symbolic temporal problems.

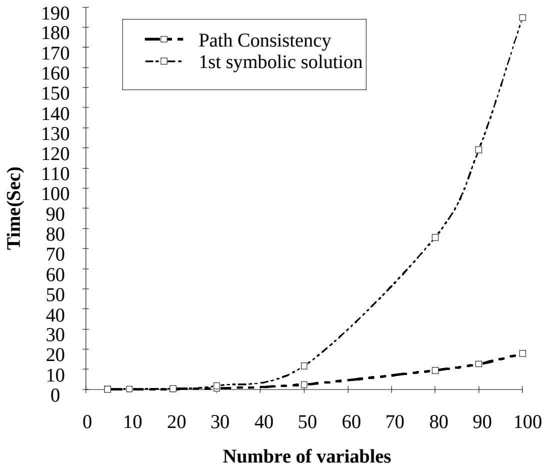

Figure 6. Experimental tests on symbolic temporal problems.

strategy) to solve temporal randomly generated constraint problems. The results demonstrate the applicability of the TCA algorithm to problems with a number of variables less then 200.

Figure 5f concerns experiments on randomly generated problems consisting 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 this figure, AC3 is better than AC7. This is due to the property defined in 3.3 to improve the efficiency of the algorithm AC3 and permits the reduction of the number of consistency check when the P Allen primitive 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. 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 attach 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 attached to 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 obviously much larger than the number of nodes.

7. Conclusion

We have presented in this paper our TemPro model 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 defined on average an optimal 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 :

- 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, as indicated in the following table :

	under constrained problems	middle and over constrained problems	scheduling problems
AC-3	pre-processing and backtrack search	backtrack search phase	pre-processing and backtrack search
AC-7	–	pre-processing phase	–

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

References

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

C. Bessière. Arc-consistency and arc-consistency again. *Artificial Intelligence*, 65:179–190, 1994.

C. Bessière. A simple way to improve path consistency processing in interval algebra networks. In *AAAI'96*, pages 375–380, Portland, 1996.

C. Bessière, E. Freuder, and J.C. Regin. Using inference to reduce arc consistency computation. In *IJCAI'95*, pages 592–598, Montréal, Canada, 1995.

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

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

M. Ghallab and A. Mounir-Alaoui. Managing efficiently temporal relations through indexed spanning trees. In *In Proc. IJCAI'89*, pages 1297–1303, Menlo Park, Calif, USA, 1989.

R.M. Haralick and G.L. Elliott. Increasing tree search efficiency for Constraint Satisfaction Problems. *Artificial Intelligence*, 14:263–313, 1980.

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

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

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

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

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

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

R. Mohr and T. Henderson. Arc and path consistency revisited. *Artificial Intelligence*, 28:225–233, 1986.

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

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

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

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

E. Tsang. *Foundation of Constraint Satisfaction*. Academic Press, 1994.

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

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

P. van Beek. Reasoning about qualitative temporal information. In *AAAI'90*, pages 297–326, Boston, MA, 1990.

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

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.

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

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