Dynamic Path Consistency for Interval-based Temporal Reasoning

Malek Mouhoub
Department of Computer Science
University of Regina
3737 Waskana Parkway,
Regina SK, Canada, S4S 0A2
email : mouhoubm@cs.uregina.ca

ABSTRACT

Path consistency is an important component of the resolution method needed to check for the consistency of an Allen's Interval-based temporal network. While this local consistency algorithm reduces, in general, the size of the search space before applying the backtrack search, it implies the consistency of the problem for some specific temporal networks. Our goal in this paper is to maintain the path consistency of a temporal network in a dynamic environment i.e anytime a temporal relation is added or removed. For this purpose we propose a dynamic path consistency algorithm based on van Beek's path consistency technique. Experimental tests on randomly generated dynamic interval-based temporal networks demonstrate the efficiency of our algorithm to deal with large size temporal problems in a dynamic environement.

KEY WORDS
Temporal Reasoning, Constraint Satisfaction, Path Consistency, Planning and Scheduling.

1 Introduction

Many real world applications including scheduling, planning, molecular biology and natural language processing rely mainly on temporal reasoning. When using a temporal reasoning system based on Allen's interval-based framework, a problem involving temporal information is first transformed into a temporal network which is in fact a particular case of a constraint satisfaction problem (CSP) involving interval constraints. Propagation algorithms are then used to check for the consistency of the network and, if so, find one or more scenarios that are consistent with the temporal information. More precisely the resolution method used for checking the consistency of the network is divided in two phases : a local consistency algorithm is first used to reduce the size of the search space by removing from the temporal relations some Allen primitives that do not belong to any solution. The backtrack search algorithm is then performed to look for a possible solution. Since we are dealing here with relations on infinite domains, path consistency is the pruning technique of choice performed in the form of symbolic computations with the constraint relations themselves rather than on tuples of values[1, 2, 3]. Ladkin[2] proposed a solving algorithm where path consistency is used as a preprocessing phase and also during the backtrack search. A path consistency algorithm can also be used as a heuristic to test whether a temporal network is consistent.

One main concern when solving CSPs in general and temporal networks in particular, is the ability to deal with constraints in a dynamic environment. Our goal in this paper is to maintain the path consistency in a dynamic environment i.e anytime a constraint is added or removed. This is of interest in many real world applications such as reactive scheduling and planning where the system has to react to new external information corresponding to constraint restriction or relaxation. In scheduling problems, for example, a solution corresponding to an ordering of tasks to be processed can no longer be consistent if a given machine becomes unavailable. We have then to look for another solution (ordering of tasks) satisfying the old constraints and taking into account the new information. For this purpose we have modified the path consistency algorithm proposed by van Beek[4] in order to deal with the addition and the relaxation of constraints in an efficient way. Experimental tests performed on randomly generated dynamic temporal constrain networks show the efficiency of our algorithm to deal with large size problems.

In the following section we will present some background for symbolic temporal constraints. We will then present our dynamic path consistency algorithm in chapter 3. Chapter 4 is dedicated to the experiments we have performed to evaluate our algorithm. Finally, concluding remarks and possible perspectives of our work are presented in section 5.

2 Representing Temporal Information

As we stated earlier Allen's approach for reasoning about time is based on the notion of time intervals and binary relations on them. A time interval I is an ordered pair (I^-, I^+) such that $I^- < I^+$, where X^- and X^+ are points on the real line. There are thirteen basic relations that can hold between intervals (see table 1 for the definition of the thirteen Allen primitives). A symbolic relation between two intervals is represented by the disjunction of some Allen prim-

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

Table 1. Allen primitives

itives. For example if I and J represent the process time corresponding to two tasks that should be processed in mutual exclusion (no overlapping between process times) then the corresponding relation will be $I \ P \vee P^{\smile} \ J$.

Example 1 : Consider the following typical temporal reasoning problem[1] :

1. *John, Mary and Wendy* **separately** *rode to the soccer game.*
2. *John either* **started or arrived** *just as Mary* **started**.
3. *John's trip* **overlapped** *the soccer game.*
4. *Mary's trip took place* **during** *the game or else the game took place* **during** *her trip.*

The above story includes qualitative information (words in boldface). There are four main events : John, Mary and Wendy are going to the soccer game respectively and the soccer game itself. The graph of figure 1 represents the temporal network corresponding to the above example. We use a graphical notation where nodes represent the different events and arcs are labeled with the temporal relations. For example the relation $D \vee D^{\smile}$ represents the temporal information listed in the forth item above.

[1]This problem is basically taken from an example presented by Ligozat, Guesgen and Anger at the tutorial : Tractability in Qualitative Spatial and Temporal Reasoning, IJCAI'01. We have modified the example for the purpose of our work.

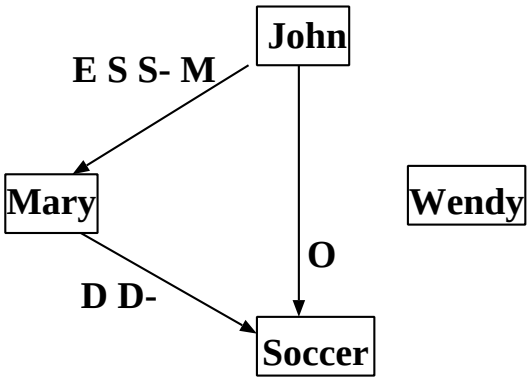

Figure 1. A Temporal Network.

3 Dynamic Maintenance of Path Consistency

Before we present the maintenance of path consistency of a temporal network in a dynamic environment let us introduce the notions of dynamic constraint satisfaction in general and in the case of temporal networks.

3.1 Dynamic Constraint Satisfaction Problem

A dynamic constraint satisfaction problem (DCSP) P is a sequence of static CSPs $P_0, \ldots, P_i, P_{i+1}, \ldots, P_n$ each resulting from a change in the preceding one imposed by the "outside world". This change can either be a restric-

	E	P	$P^{\smile}$	D	$D^{\smile}$	O	$O^{\smile}$	M	$M^{\smile}$	S	$S^{\smile}$	F	$F^{\smile}$
E	E	P	$P^{\smile}$	D	$D^{\smile}$	O	$O^{\smile}$	M	m	S	s	F	$F^{\smile}$
P	P	P	I	u	P	P	u	P	u	P	P	u	P
$P^{\smile}$	p	I	$P^{\smile}$	$v^{\smile}$	$P^{\smile}$	$v^{\smile}$	$P^{\smile}$	$v^{\smile}$	$P^{\smile}$	$v^{\smile}$	$P^{\smile}$	$P^{\smile}$	$P^{\smile}$
D	D	P	$P^{\smile}$	D	I	u	$v^{\smile}$	P	$P^{\smile}$	D	$v^{\smile}$	D	u
$D^{\smile}$	$D^{\smile}$	v	$u^{\smile}$	n	$D^{\smile}$	$z^{\smile}$	$y^{\smile}$	$z^{\smile}$	$y^{\smile}$	$z^{\smile}$	$D^{\smile}$	$y^{\smile}$	$D^{\smile}$
O	O	P	$u^{\smile}$	y	v	x	n	P	$y^{\smile}$	O	$z^{\smile}$	y	x
$O^{\smile}$	$O^{\smile}$	v	$P^{\smile}$	z	$u^{\smile}$	n	$x^{\smile}$	$z^{\smile}$	$P^{\smile}$	z	$x^{\smile}$	$O^{\smile}$	$y^{\smile}$
M	M	P	$u^{\smile}$	y	P	P	y	P	a	M	M	y	P
$M^{\smile}$	$M^{\smile}$	v	$P^{\smile}$	z	$P^{\smile}$	z	$P^{\smile}$	b	$P^{\smile}$	z	$P^{\smile}$	$M^{\smile}$	$M^{\smile}$
S	S	P	$P^{\smile}$	D	v	x	z	P	$M^{\smile}$	S	b	D	x
$S^{\smile}$	s	v	$P^{\smile}$	z	$D^{\smile}$	$z^{\smile}$	$O^{\smile}$	$z^{\smile}$	m	b	s	$O^{\smile}$	$D^{\smile}$
F	F	P	$P^{\smile}$	D	$u^{\smile}$	y	$x^{\smile}$	M	$P^{\smile}$	D	$x^{\smile}$	F	a
$F^{\smile}$	$F^{\smile}$	P	$u^{\smile}$	y	$D^{\smile}$	O	$y^{\smile}$	M	$y^{\smile}$	O	$D^{\smile}$	a	$F^{\smile}$

$x = P \vee O \vee M$
$y = D \vee O \vee S$
$z = D \vee O^{\smile} \vee F$
$a = E \vee F \vee F^{\smile}$
$b = E \vee S \vee S^{\smile}$
$u = P \vee O \vee M \vee D \vee S$
$v = P \vee O \vee M \vee D^{\smile} \vee F^{\smile}$
$n = E \vee F \vee D \vee O \vee S \vee F^{\smile} \vee D^{\smile} \vee O^{\smile} \vee S^{\smile}$

Table 2. Allen's composition table

tion (adding a new constraint) or a relaxation (removing a constraint because it is no longer interesting or because the current CSP has no solution). More precisely, P_{i+1} is obtained by performing a restriction (addition of a constraint) or a relaxation (suppression of a constraint) on P_i. We consider that P_0 (initial CSP) has an empty set of constraints.

3.2 Dynamic Temporal Constraint Satisfaction Problem

Since a temporal network is a CSP where constraints are disjunctions of Allen primitives, the definition of a dynamic temporal constraint satisfaction problem (DTCSP) is slightly different from the definition of a DCSP. Indeed in the case of a DTCSP, a restriction can be obtained by removing one or more Allen primitive from a given constraint. A particular case is when the initial constraint is equal to the disjunction of the 13 primitives (we call it the universal relation I) which means that the constraint does not exist (there is no information about the relation between the two involved events). In this particular case, removing one or more Allen primitives from the universal relation is equivalent to adding a new constraint. Using the same way, a relaxation can be obtained by adding one or more Allen primitives to a given constraint. A particular case is when the new constraint has 13 Allen primitives which is equivalent to the suppression of the constraint.

3.3 Dynamic Path Consistency

3.3.1 Path Consistency Algorithm

The path consistency algorithm (called also transitive closure algorithm) works as follows :

> Choose any three nodes I, J and K of the temporal network and checks whether $R_{IK} = R_{IK} \cap (R_{IJ} \otimes R_{JK})$. If R_{IK} is updated then this update should be propagated to the rest of the network. The algorithm iterates until no more such changes are possible.

R_{IK} (respectively R_{IJ} and R_{JK}) is the binary relation between node I and node K (respectively between nodes I and J; and between nodes J and K). $\cap$ is the intersection operator between two relations (the result of the intersection between two relations is the common Allen primitves between the two relations). $\otimes$ is the composition operator between two relations. Indeed, Allen[1] has defined a 13x13 composition table between Allen primitives (see table 2). The path consistency algorithm assumes that the constraint graph is complete. If the initial graph is not complete then it is transformed to a complete one by adding arcs labeled with the universal relation I which corresponds to the disjunction of the thirteen Allen primitives.

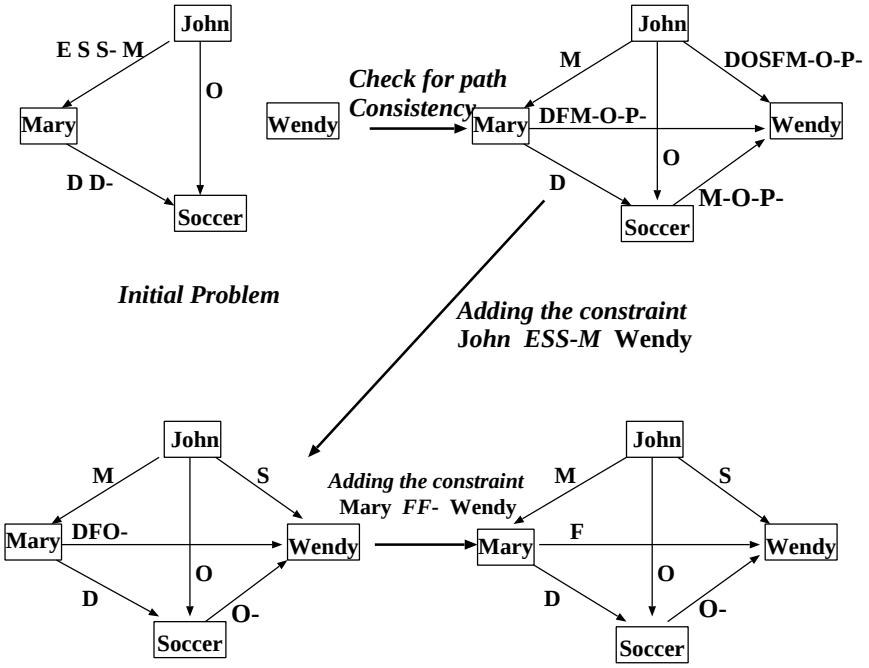

Figure 2. Restriction of a temporal network

3.3.2 Dynamic Path Consistency Algorithm

The goal of the resolution method we present here consists of maintaining the path consistency anytime a constraint is added (constraint restriction) or removed (constraint relaxation). In both cases (constraint restriction or relaxation) the new information corresponding to the suppression of some Allen primitives (in the case of constraint restriction) or to the addition of some Allen primitives (in the case of constraint relaxation) is propagated to the entire network using transitive closure based on the composition between operators. In the case of constraint restriction if a given relation becomes empty then the path consistency cannot be maintained when considering the restriction. The details of the constraint restriction and relaxation methods are presented as follows.

Constraint Restriction

The pseudo code of the resolution method for constraint restriction is presented below. Anytime a new constraint is added (disjunction of some Allen primitives), the method works as follows :

1. Compute the intersection of the new constraint with the corresponding constraint in the arc and path consistent graph.

   If the result of the intersection is an empty relation then the new constraint cannot be added otherwise it will violate the path consistency of the graph.

   Else (a) Replace the current constraint of the graph by the result of the intersection.

   (b) Perform the dynamic path consistency (DPC) in order to propagate the update of the constraint to the rest of the graph. If the resulting graph is not path consistent then the new constraint cannot be added.

Constraint Relaxation

Let us assume we want to relax a given constraint by adding to it some Allen primitives. The resolution method works then as follows :

1. Run the dynamic path consistency algorithm starting from the triangle containing the relaxed constraint. If at least one of the other two relations belonging to the triangle is updated, propagate this change to the other constraints.

Function Restrict(i,j)
1. $C_{ij} \leftarrow new\_constraint \bigcap C_{ij}$
2. **if** $(C_{ij} = \emptyset)$ **then**
3. return "Constraint cannot be added"
4. **else**
5. $updated\_list \leftarrow \{C_{ij}\}$
6. **if** $\neg DPC(updated\_list)$ **then**
7. return "Constraint cannot be added"
8. **endif**
9. **endif**

Function $DPC(updated\_list)$

1. $L \leftarrow updated\_list$
2. **while** $(L \neq \emptyset)$ **do**
3. select and delete an (x,y) from L
4. **for** $k \leftarrow 1$ to n, $k \neq x$ and $k \neq y$ **do**
5. $t \leftarrow C_{xk} \bigcap C_{xy}.C_{yk}$
6. **if** $(t \neq C_{xk})$ **then**
7. **if** $(t = \emptyset)$ **then return** false
8. $C_{xk} \leftarrow t$
9. $C_{kx} \leftarrow INVERSE(t)$
10. $L \leftarrow L \cup \{(x,k)\}$
11. **endif**
12. $t \leftarrow C_{ky} \bigcap C_{kx}.C_{xy}$
13. **if** $(t \neq C_{ky})$ **then**
14. **if** $(t = \emptyset)$ **then return** false
15. $C_{ky} \leftarrow t$
16. $C_{yk} \leftarrow INVERSE(t)$
17. $L \leftarrow L \cup \{(k,y)\}$
18. **endif**
19. **endfor**
20. **endwhile**
21. **return** true

3.3.3 Example

The top right graph of figure 2 is the temporal network obtained after applying the path consistency. Note that some constraints such as the relation ($M^{\smile} \vee O^{\smile} \vee P^{\smile}$) between the soccer game and Wendy is an implicit constraint deduced after the performing the path consistency.

Let us assume now that we have the following constraint restrictions :

1. John either **started or arrived** just as Wendy **started**.
2. Mary and Wendy **arrived together but started at different times**.
3. Wendy **arrived** just as the soccer game **started**.

The first operation corresponds to the addition of the relation $S \vee S^{\smile} \vee M \vee E$ between John and Wendy. The intersection of this relation with the current constraint between the two events will lead to the relation S. The second operation corresponds to the addition of the relation $F \vee F^{\smile}$ between Mary and Wendy. The intersection of this relation with the current constraint between the two events will lead to the relation F. The relation F does not conflict with the global solution obtained so far. Thus the consistency of the graph is maintained.

The third operation corresponds to the addition of the relation P between Wendy and soccer game. The intersection of this relation with the current constraint between the two events will lead to an empty relation. Thus this third constraint cannot be added.

Nb events	incremental method	static method
20	0.05	0.47
40	0.27	12.12
60	0.91	85.78
80	2.39	350.13
100	5.06	1259.01

Table 3. Comparative tests on randomly generated dynamic TCSPs.

4 Experimentation

In this section, we present experimental tests evaluating the performance in time of the method we propose for maintaining the path consistency of temporal constraints in a dynamic environment. The tests are performed on randomly generated dynamic TCSPs. Each problem instance is defined by a set of actions corresponding to N(N-1)/2 additions and N(N-1)/4 retractions of constraints (N is the number of events). Constraints are added and removed in a random order, the final TCSP will have $N(N-1)/4$ constraints. The experiments are performed on a SUN SPARC Ultra 5 station. All the procedures are coded in C|C++.

Table 3 describes the results of the tests. The first column corresponds to the number of events of the generated problem. The second and third columns indicate respectively the running time in seconds needed by the new resolution method we propose (that we call incremental method) and the method for static TCSPs (described in subsection 2.2), to maintain the local consistency of the generated problem. As we can easily see the new method we propose in much faster because anytime a constraint is added or removed, it does not perform the arc and path consistency on the entire graph (as it is the case of the static method) but only on those constraints affected by the change.

5 Conclusion and Future Work

In this paper we have presented a method for maintaining, in a dynamic environment, the local consistency of a temporal constraint satisfaction problem (TCSP). Since we are dealing with numeric and symbolic constraints, local consistency concerns the path consistency. The method is of interest for any application where qualitative temporal information should be managed in an evolutive environment. This can be the case of real world applications such as reactive scheduling and planning where any new information corresponding to a constraint restriction or relaxation should be handled in an efficient way. The method can also be used for over constrained problems where constraints have to be removed until the consistency is reestablished.

One perspective of our work is to handle the addition and relaxation of constraints during the backtrack search phase (needed to look for a possible solution to the prob-

lem). For example, suppose when processing the backtrack search a constraint is added during the instantiation of the current variable. In this case, the assignment of the variables already instantiated should be reconsidered and the domains of the current and future variables (no assigned variables) should be updated. The other perspective of our work is to maintain in a dynamic environment the global consistency of a TCSP. Let us assume that after a solution is found for a given problem, a new constraint is added. We have then to check if there still exist a solution to the problem after the addition of the constraint.

References

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

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

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

[4] 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.