Handling Temporal Constraints in a Dynamic Environment

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

Abstract

Managing symbolic and metric temporal information is fundamental for many real world applications such as scheduling, planning, data base design, computational linguistics and computational models for molecular biology. This motivates us to develop a temporal constraint solving system based on CSPs for handling the two types of temporal information. A main challenge when designing such systems is the ability to deal with temporal constraints in a dynamic and evolutive environment. That is to check, anytime a new constraint is added, whether a solution to the problem (consistent scenario) continues to be a solution when a new constraint is added and if not, whether a new solution satisfying the old and new constraints can be found. We talk then about on line temporal CSP-based systems capable of reacting, in an efficient way, to any new external information during the constraint resolution process. In this paper we will present three different techniques we use to tackle dynamic temporal constraint satisfaction problems. These three techniques are respectively based on constraint propagation, randomized local search and genetic algorithms.

Keywords: Temporal Reasoning, Constraint Satisfaction, Planning and Scheduling.

1 Introduction

In any constraint satisfaction problem (CSP) there is a collection of variables which all have to be assigned values from their discrete domains, subject to specified constraints. Because of the importance of these problems in so many different fields, a wide variety of techniques and programming languages from artificial intelligence, operations research and discrete mathematics are being developed to tackle problems of this kind. An important issue when dealing with a constraint satisfaction problem in the real world is the ability of maintaining the consistency of the problem anytime a new constraint is added. Indeed this change may affect the solution already obtained and respecting the old constraints. Our goal in this paper is to maintain the global consistency, in a dynamic environment in the case of constraint restriction, of a constraint satisfaction problem involving qualitative and quantitative temporal constraints. This is of practical relevance since it is often required to check whether a solution to a CSP involving temporal constraints continues to be a solution when a new constraint is added and if not, whether a new solution satisfying the old and new constraints can be found. 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.

In a previous work [Mouhoub *et al.*, 1998], we have developed a temporal model, TemPro, based on Allen's interval algebra and a discrete representation of time, to express numeric and symbolic time information in terms of qualitative and quantitative temporal constraints. More precisely, TemPro translates an application involving temporal information into a binary Constraint Satisfaction Problem[1] where constraints are temporal relations between variables defined on domains of numeric intervals. We call it Temporal Constraint Satisfaction Problem (TCSP)[2]. Managing temporal information consists then of maintaining the consistency of the related TCSP using constraint satisfaction techniques. Local consistency is enforced by applying the arc consistency for numeric constraints and the path consistency for symbolic relations. Global consistency is then obtained by using a backtrack search algorithm to look for a possible solution. Note that for some TCSPs local consistency implies global consistency [Meiri, 1996].

In order to check for the global consistency of a TCSP in a dynamic environment, we will propose three different resolution techniques. The first one is an exact method based on the above constraint satisfaction techniques that we have adapted in order to handle the addition of constraints in an efficient way. The second technique is based on stochastic local

[1] A binary CSP involves a list of variables defined on finite domains of values and a list of binary relations between variables.

[2] Note that this name and the corresponding acronym was used in [Dechter *et al.*, 1991]. A comparison of the approach proposed in this later paper and our model TemPro is described in [Mouhoub *et al.*, 1998].

search. Indeed the underlying local search paradigm is well suited for recovering solutions after local changes (addition of constraints) of the problem occur. The third method based on genetic algorithms is similar to the second one except that the search is multi-directional and maintains a list of potential solutions (population of individuals) instead of a single one. This has the advantage to allow the competition between solutions of the same population which simulates the natural process of evolution. The main difference between the three methods is that the first one is a systematic search technique that guarantees the completeness of the solution provided which is not the case of the other two approximation methods.

In order to evaluate the performance of the three methods to deal with TCSPs in a dynamic environment, experimental tests on randomly generated TCSPs have been performed.

The rest of the paper is organized as follows : in the next section, we will present through an example, the different components of our model TemPro. The three methods for maintaining the global consistency of TCSPs in a dynamic environment are then presented respectively in sections 3, 4 and 5. Section 6 is dedicated to the experimental evaluation on randomly generated TCSPs of the methods we propose. Concluding remarks and possible perspectives of our work are then presented in section 7.

2 CSP-based Representation of Numeric and Symbolic Constraints : the model TemPro

TemPro transforms any problem under qualitative and quantitative constraints into a binary CSP where constraints are disjunctions of Allen primitives [Allen, 1983] (see table 1 for the definition of the 13 Allen primitives) and variables, representing temporal events, are defined on domains of time intervals. We call this later a Temporal Constraint Satisfaction Problem (TCSP)[3]. Each event domain (called also temporal window) contains the Set of Possible Occurrences (SOPO) of numeric intervals the corresponding event can take. The SOPO is the numeric constraint of the event. It is expressed by the fourfold $[earliest\_start,\ latest\_end,\ duration,\ step]$ where : $earliest\_start$ is the earliest start time of the event, $latest\_end$ is the latest end time of the event, $duration$ is the duration of the event and $step$ is the discretization step corresponding to the number of time units between the start time of two adjacent intervals belonging to the event domain.

To illustrate the different components of the model TemPro let us consider the following scheduling problem[4].

Example 1

The production of two items A and B requires three mono processor machines M_1, M_2 and M_3. Each of the two items can be produced using two different ways depending on the order in which the machines are used. The process time of each machine is variable and depends on the task to be processed. The following lists the different ways to produce each of the two items (the process time for each machine is mentioned in brackets) :

item A: $M_2(3), M_1(3), M_3(6)$ *or*
$M_2(3), M_3(6), M_1(3)$
item B: $M_2(2), M_1(5), M_2(2), M_3(7)$ *or*
$M_2(2), M_3(7), M_2(2), M_1(5)$

Relation	Symbol	Inverse	Meaning
X precedes Y	P	P-	X Y
X equals Y	E	E	X / Y
X meets Y	M	M-	X Y
X overlaps Y	O	O-	X Y
X during Y	D	D-	X Y
X starts Y	S	S-	X Y
X finishes Y	F	F-	Y X

Table 1: Allen primitives

The goal here is to find a possible schedule of the different machines to produce the two items and respecting all the constraints of the problem. We also assume that items A and B should be produced within 25 and 30 units of time respectively.

In the following we will describe how is the above problem transformed into a TCSP using our model TemPro. Figure 1 illustrates the graph representation of the TCSP corresponding the the scheduling problem. A temporal event corresponds here to the contribution of a given machine to produce a certain item. For example, the event AM_1 corresponds to the use of machine M_1 to produce the item A, ..., etc. Seven events are needed in total to produce the two items as follows :

item A: $AM_2(3), AM_1(3), AM_3(6)$ or
$AM_2(3), AM_3(6), AM_1(3)$
item B: $BM_{21}(2), BM_1(5), BM_{22}(2), BM_3(7)$ or
$BM_{21}(2), BM_3(7), BM_{22}(2), BM_1(5)$

The translation to Allen primitives of the disjunction of the two sequences required to produce item B needs a 3-ary relation involving BM_1, BM_{22} and BM_3. This relation states that BM_{22} should occur between BM_1 and BM_3. Since

[3]Note that this name and the corresponding acronym was used in [Dechter *et al.*, 1991]. A comparison of the approach proposed in this later paper and our model TemPro is described in [Mouhoub *et al.*, 1998].

[4]This problem is taken from [Laborie, 1995].

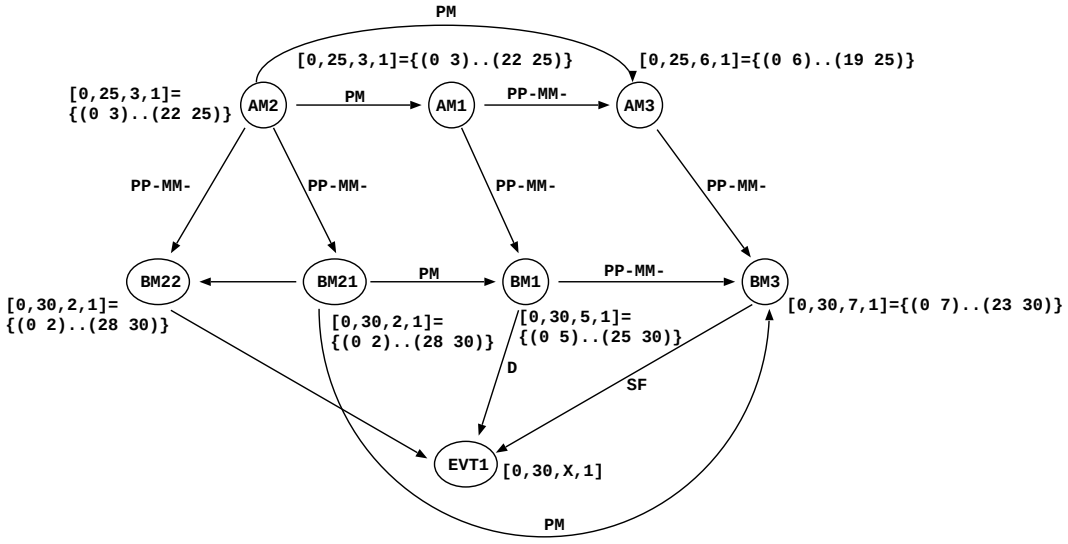

Figure 1: TCSP corresponding to the problem presented in example 1.

our temporal network handles only binary relations, the way we use to represent this kind of 3-ary relations is as follows : we create an additional event (EVT_1) and represent the constraints for producing item B as shown in figure 1. The duration X of EVT_1 is greater (or equal) than the sum of the durations of BM_1, BM_{22} and BM_3. Figure 2 illustrates the solution to the above problem provided by the constraint propagation based method we have described in introduction. Note that this solution is optimal[5] but not unique.

3 Dynamic Maintenance of Global Consistency using CSP techniques

Before we present the resolution method for maintaining the global consistency of temporal constraints in a dynamic environment, let us introduce the notion of dynamic temporal constraint satisfaction.

3.1 Dynamic Temporal Constraint Satisfaction Problem (DTCSP)

A dynamic temporal constraint satisfaction problem (DTCSP) is a sequence of static TCSPs : $TCSP_0$, ..., $TCSP_i$, $TCSP_{i+1}$, ..., $TCSP_n$ each resulting from a change in the preceding one imposed by the "outside world". This change corresponds to a constraint restriction or relaxation. In this paper we will focus only on constraint restriction. More precisely, $TCSP_{i+1}$ is obtained by performing a restriction on $TCSP_i$. We consider that $TCSP_0$ (initial TCSP) has an empty set of constraints. A restriction can be obtained by removing one or more Allen primitives 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.

[5] The total processing time of all machines needed to produce the five items, 26 seconds, is minimal

3.2 The Resolution Method

Given that we start from a consistent TCSP, the goal of the resolution method we present here consists of maintaining the global consistency (existence of a solution) anytime a new constraint is added. The method works as follows :

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

   If the result of the intersection is not an empty relation **then**

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

   (b) **If** the new constraint is inconsistent with the current solution **then**

   i. Perform the numeric $\rightarrow$ symbolic conversion for the updated constraint. If the symbolic relation becomes empty then the new constraint cannot be added. The numeric $\rightarrow$ symbolic conversion works as follows : from the numeric information, we can extract the corresponding symbolic relation. An intersection of this relation with the given qualitative information will reduce the size of the latter which simplifies the size of the original problem.
   Although the exact algorithm that converts numeric to symbolic time information requires $O(e(Max(\frac{sup_i - inf_i - d_i}{s_i}))^2)$ in time where e is the number of qualitative constraints, we have defined a method that extracts most of the primitives within a relation between each pair of events in constant time reducing the complexity to $O(e)$. The method consists of using the

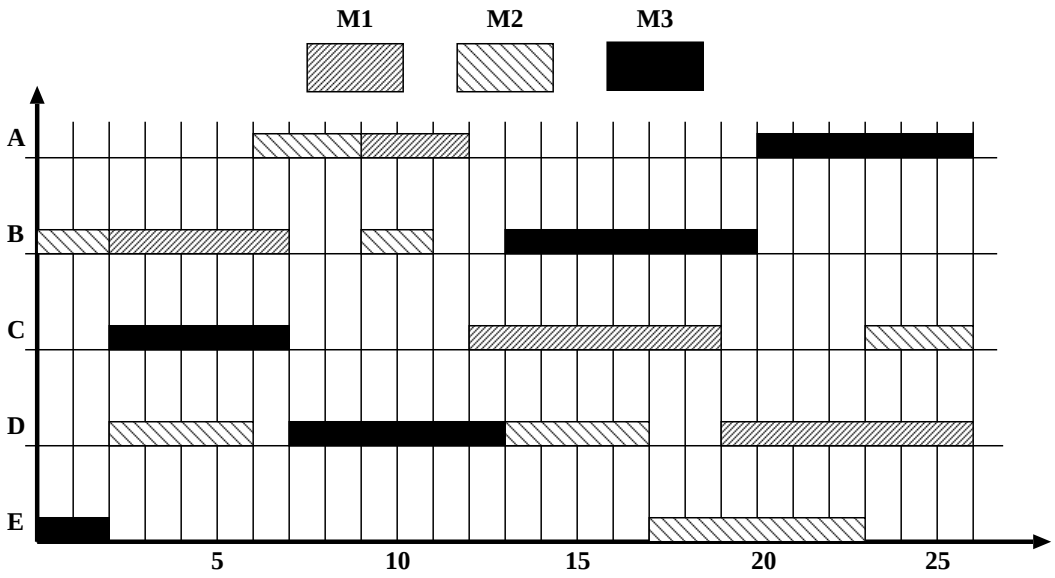

Figure 2: Optimal solution provided by the CSP based method.

information concerning the lower bound, upper bound and duration of the event temporal window instead of its occurrences. For example :

- if $inf_i > sup_j$ then $e_i \ P^{\smile} \ e_j$,
- if $d_i > d_j$ then E, S, F, D cannot belong to the relation between e_i and e_j,
- ...etc.

e_i and e_j are two events and inf_i, sup_j, d_i and d_j are respectively the earliest start time of e_i, latest end time of e_j, duration of e_i and duration

ii. Perform 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.

iii. Perform dynamic arc consistency (DAC) starting with the updated constraints. If the new graph is not arc consistent then the new constraint cannot be added.

iv. Perform the backtrack search algorithm in order to look for a new solution to the problem. The backtrack search will start here from the point (resume point) it stopped in the previous search when it succeeded to find a complete assignment satisfying all the constraints. This way the part of the search space already explored in the previous searches will be avoided. The search will explore the rest of the search space. If a solution is found then the point where the backtrack search stopped is saved as new resume point and the new solution is returned. Otherwise the graph is inconsistent (when adding the new constraint). The new constraint cannot be added.

Else the new constraint cannot be added otherwise it will violate the consistency of the graph.

DPC it the path consistency algorithm PC-2 [van Beek and Manchak, 1996] we have adapted to handle constraint additions in an incremental way. DAC is the new arc consistency algorithm AC-3 [Zhang and Yap, 2001; Bessière and Régin, 2001] we have adapted for temporal constraints in a dynamic environment. A detailed description of DAC can be found in [Mouhoub and Yip, 2002].

4 Dynamic Maintenance of Global Consistency using Stochastic Local Search

In this section we present the way to solve dynamic TCSPs using stochastic local search method. The algorithm that we will consider in the following is based on a common idea known under the notion of local search. In local search, an initial configuration (potential solution) is generated randomly and the algorithm moves from the current configuration to a neighborhood configurations until a complete solution (solution satisfying all the constraints) or a good one has been found. When a new constraint is added, the algorithm restarts the search from the point corresponding to the last solution obtained, and iterates until a new solution respecting the old constraints and the new one is found.

The pseudo-code in figure 3 illustrates the local search strategy using the Min-Conflict-Random-Walk technique (MCRW). This technique starts by randomly generating an initial configuration (corresponding to a potential solution). It then chooses randomly any conflicting event in the configuration, i.e., the event that is involved in any unsatisfied constraint, and picks a value (numeric interval) which minimizes the number of violated constraints (break ties randomly). If no such value exists, it picks randomly one value that does not increase the number of violated constraints (the

```
procedure MCRW(Max_Moves,p)
Begin
  s ← random valuation of events;
  while there is a new constraint to be processed do
    nb_moves ← 0;
    while eval(s) > 0 & nb_moves < Max_Moves do
      if probability p verified then
        choose randomly an event evt in conflict;
        choose randomly an interval intv for evt;
      else
        choose randomly an event evt in conflict;
        choose an interval intv that minimizes
        the number of conflicts for evt;
      endif
      if intv ≠ current value of evt then
        assign intv to evt;
        nb_moves ← nb_moves+1;
      endif
    endwhile
  return the solution s
  endwhile
End
```

Figure 3: Pseudo-code of the MCRW method.

current value of the event is picked only if all the other values increase the number of violated constraints). In order to avoid being trapped in a local minimum, a random-walk strategy used, works as follows : for a given conflicting event, this strategy picks randomly a value with probability p, and apply the Min Conflict heuristic with probability $1 - p$. In the worst case, the time cost required in each move corresponds to the time needed to determine the value that minimizes the number of violated constraints. This time is of order $O(N\ Max_{1 \leq i \leq N}(\frac{sup_i - inf_i - d_i}{s_i}))$ where N is the number of variables and sup_i, inf_i, s_i and d_i are respectively the latest end time, earliest start time, duration and step of a given event evt_i. Max is the function that returns the maximum of a list of numbers.

5 Dynamic Maintenance of Global Consistency using Genetic Algorithms

Genetic algorithms (GAs) perform multi-directional searches by maintaining potential solutions or scenarios (called also population of individuals) and encouraging information formation and exchange between these directions. It is an iterative procedure that maintains a constant size population of candidate solutions. Each iteration is called a generation and it undergoes some changes. *Crossover* and *mutation* are the two primary genetic operators that generate or exchange information in GAs. Under each generation, *good solutions* are expected to be produced and *bad solutions* die. It is the role of the objective (evaluation or fitness) function to distinguish the goodness of the solution. In the case of TCSPs we define the following concepts.

Individual (potential solution) : one possible assignment of numeric intervals to all events i.e set of couples (ev_i, occ_j), where ev_i is an event and occ_j is a possible interval belonging to the domain of ev_i. In other words the individual represents a potential solution to the problem.

Random individual : random assignment of intervals to all events.

Population : a set of individuals (potential solutions).

Mutation : unary operator that returns a new individual (child) by assigning new values (numeric intervals) to some events of a given individual (parent).

Crossover : n-ary operator that takes as arguments two or more individuals and returns a new individual with assignments belonging to parent individuals.

Fitness (evaluation) function : returns a measure of an individual. The measure corresponds here to the quality of the solution. The quality is defined by the number of satisfied constraints.

```
1.  Begin
2.    t ← 1
3.    // P(t) denotes a population at iteration t
4.    P(t) ← n randomly generated individuals
5.    eval ← evaluate P(t)
6.    while there is a new constraint to process do
6.      while termination condition is not satisfied do
7.        begin
8.          t ← t + 1
9.          select P(t) from P(t − 1)
10.         alter P(t)
11.         evaluate P(t)
12.       end
13.   end
```

Figure 4: Genetic Algorithm.

The pseudo code of the GA based method is illustrated in figure 4. The method starts from a population of p random individuals and iterates until the termination condition is satisfied. The method maintains a population of n individuals, $P(1) = \{ind_1^1, \ldots, ind_n^1\}$ for iteration 1, ... $P(t) = \{ind_1^t, \ldots, ind_n^t\}$ for iteration t, ... etc. Each individual (potential solution) ind_i^t is evaluated using the fitness function. A new population at iteration $t+1$ is then formed by selecting the more fit individuals (*select* step in line 9) from the population of iteration t. Some of the selected individuals will be transformed (*alter* step in line 10) by the mutation and crossover operators. The algorithm is executed until it is running out of time or a solution with the best (or acceptable) quality is found. In the same manner as for the local search methods, when a new constraint is added, the GA algorithm restarts from the state where it found the last solution and iterates until a new solution, respecting the old and new constraints, is obtained.

N	C	Dynamic CSPs	MCRW	GAs
20	95	0.10	0.11	0.27
40	390	0.35	0.22	0.34
60	885	1.02	0.79	0.92
80	1580	2.58	1.24	1.27
100	2475	6.10	1.89	2.01
200	9950	28	2.23	2.89

Table 2: Comparative tests on randomly generated DTCSPs.

6 Experimentation

In order to evaluate and compare the performance of the three methods we propose we have performed experimental tests on randomly generated DTCSPs. The criteria used to evaluate the three different methods is the running time needed to maintain the global consistency of the DTCSP. The experiments are performed on a SUN SPARC Ultra 5 station. All the procedures are coded in C/C++. Each DTCSP is randomly generated as follows :

- Randomly generate N temporal windows (SOPOs) corresponding to N variables,
- and a list of C temporal relations (disjunction of Allen primitives).

The resolution algorithm will then process the list of temporal relations in an incremental way. For each constraint the algorithm will check for the global consistency of the resulting temporal problem. If the problem is still consistent then the constraint is added otherwise the constraint is avoided.

Table 2 presents the results of tests performed on DTCSPs defined by the number of variables N and the number of constraints C. As we can easily see, the approximation methods are faster than the exact one however as we said in introduction these methods do not guarantee in general the completeness of the solution provided at each time.

7 Conclusion and Future Work

In this paper we have presented three different ways for maintaining the global consistency of a temporal constraint satisfaction problem in an incremental way. The methods are of interest for any application where qualitative and numeric 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 should be handled in an efficient way.

One perspective of our work is to handle the relaxation of constraints during the resolution process. For example, suppose that during the search a given constraint is removed. Would it be worthwhile to find those values removed previously because of this constraint and to put them back in the search space or would it be more costly than just continuing on with search.

References

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

[Bessière and Régin, 2001] C. Bessière and J. C. Régin. Refining the basic constraint propagation algorithm. In *Seventeenth International Joint Conference on Artificial Intelligence (IJCAI'01)*, pages 309–315, Seattle, WA, 2001.

[Dechter *et al.*, 1991] R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artificial Intelligence*, 49:61–95, 1991.

[Laborie, 1995] P. Laborie. *Une approche intégrée pour la gestion de ressources et la synthèse de plans*. PhD thesis, École Nationale Supérieure des Télécommunications, 1995.

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

[Mouhoub and Yip, 2002] M. Mouhoub and J. Yip. Dynamic CSPs for Interval-based Temporal Reasoning. In *Fifteenth International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems (IEA/AIE-2002), Published in Lectures Notes in Computer Science(LNCS) 2358.*, pages 575–585, Cairns, Australia, 2002.

[Mouhoub *et al.*, 1998] M. Mouhoub, F. Charpillet, and J.P. Haton. Experimental Analysis of Numeric and Symbolic Constraint Satisfaction Techniques for Temporal Reasoning. *Constraints: An International Journal*, 2:151–164, Kluwer Academic Publishers, 1998.

[van Beek and Manchak, 1996] 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.

[Zhang and Yap, 2001] Yuanlin Zhang and Roland H. C. Yap. Making ac-3 an optimal algorithm. In *Seventeenth International Joint Conference on Artificial Intelligence (IJCAI'01)*, pages 316–321, Seattle, WA, 2001.