A New Temporal CSP Framework Handling Composite Variables and Activity Constraints

Malek Mouhoub and Amrudee Sukpan

University of Regina

Dept of Computer Science

Wascana Parkway, Regina, SK, Canada, S4S 0A2

{mouhoubm,sukpan1a}@cs.uregina.ca

Abstract

A well known approach to managing the numeric and the symbolic aspects of time is to view them as Constraint Satisfaction Problems (CSPs). Our aim is to extend the temporal CSP formalism in order to include activity constraints and composite variables. Indeed, in many real life applications the set of variables involved by the temporal constraint problem to solve is not known in advance. More precisely, while some temporal variables (called events) are available in the initial problem, others are added dynamically to the problem during the resolution process via activity constraints and composite variables. Activity constraints allow some variables to be activated (added to the problem) when activity conditions are true. Composite variables are defined on finite domains of events. We propose in this paper two methods based respectively on constraint propagation and stochastic local search (SLS) for solving temporal constraint problems with activity constraints and composite variables. We call these problems Conditional and Composite Temporal Constraint Satisfaction Problems (CCTCSPs). Experimental study we conducted on randomly generated CCTCSPs demonstrates the efficiency of our exact method based on constraint propagation in the case of middle constrained and over constrained problems while the SLS based method is the technique of choice for under constrained problems and also in case we want to trade search time for the quality of the

solution returned (number of solved constraints).

1 Introduction

Representing and reasoning about numeric and/or symbolic aspects of time is crucial in many real world applications such as scheduling and planning [1, 2, 3, 4], natural language processing [5, 6], molecular biology [7] and temporal database [8]. A well-know approach to managing these two aspects of time is to view them as Constraint Satisfaction Problems (CSPs). We talk then about temporal constraint networks [9, 10, 11, 12]. Here, a CSP involves a list of variables defined on discrete domains of values and a list of relations constraining the values that the variables can simultaneously take [13, 14, 15].

In a temporal constraint network, variables, corresponding to temporal objects, are defined on a set of time points or time intervals while constraints can either restrict the domains of the variables and/or represent the relative position between variables. The relative position between variables can be expressed via qualitative or quantitative relations. Quantitative relations are temporal distances between temporal variables while qualitative relations represent incomplete and less specific symbolic information between variables. Constraint propagation techniques and backtrack search are then used to check the consistency of the temporal network and to infer new temporal information. While a considerable research work has been proposed to reasoning on the metric or the symbolic aspects of time (respectively through metric and qualitative networks), little work such as [16, 17, 2, 18] has been developed to manage both types of information. In [19, 20], we have developed a temporal model, TemPro, based on Allen's interval algebra [9] 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 numeric and symbolic temporal information into a binary CSP[1] called Temporal CSP (or TCSP[2]) where variables are temporal events defined on domains of numeric intervals and binary constraints between variables correspond to disjunctions of Allen primitives [9]. The resolution method for solving the TCSP is based on constraint propagation and requires two stages. In the first stage, local consistency is enforced by applying the arc consistency on variable domains and the path consistency on symbolic relations. A backtrack search algorithm is then performed in the second stage to check

[1] In a binary CSP constraints can only be unary or binary.

[2] Note that the acronym TCSP was used in [11]. The well known TCSP, as defined by Dechter et al, is a quantitative temporal network used to represent only numeric temporal information. Nodes represent time points while arcs are labeled by a set of disjoint intervals denoting a disjunction of bounded differences between each pair of time points.

the consistency of the TCSP by looking for a feasible solution. Note that for some TCSPs local consistency implies the consistency of the TCSP network [17]. The backtrack search phase can be avoided in this case.

In order to deal with a large variety of real world applications, we present in this paper an extension of the modeling framework TemPro including the following :

- Managing composite temporal variables. Composite temporal variables are variables whose values are temporal events. In other words, the temporal events are the possible values the composite variables can take.
- Handling activity constraints. This is the case where temporal variables (composite or events) can have either active or non active status. Only active variables require an assignment from their domain of values. Non active variables will not be considered during the resolution of the temporal network until they are activated. A variable can be activated by default (in the initial problem) or by an activity constraint. Given two variables X_i and X_j, an activity constraint has the following form $(X_i = a_{i1}) \vee \ldots (X_i = a_{ip}) \rightarrow X_j$. This activity constraint will activate X_j if the active variable X_i is assigned one of the values $a_{i1} \ldots a_{ip}$ from its domain.

We call conditional TCSP (CTCSP) a TCSP augmented by activity constraints. Solving a CTCSP can be seen like solving a TCSP dynamically i.e when some of the variables and their corresponding constraints are added dynamically during the resolution of the TCSP. We call a composite CTCSP (CCTCSP) a CTCSP including composite temporal variables. A CCTCSP represents a finite set of possible CTCSPs where each CTCSP corresponds to a complete assignment of values (temporal events) to composite variables. Solving a CCTCSP consists of finding a feasible scenario for one of its possible CTCSPs. Solving a CTCSP requires a backtrack search algorithm with exponential complexity in time $O(D^N)$ where N is the total number of temporal events and D the domain size of each event. The possible number of CTCSPs the CCTCSP involves is d^M where M is the number of composite variables and d their domain size. Thus, solving a CCTCSP requires a backtrack search algorithm of complexity $O(D^N \text{ x } d^M)$. To overcome this difficulty in practice, we propose in this paper two methods respectively based on constraint propagation and stochastic local search (SLS) for solving efficiently CCTCSPs. Constraint propagation includes arc consistency [14] as well as forward check and full look ahead strategies [15]. On the other hand, the SLS method we use is based on the Min-Conflict-Random-Walk (MCRW) algorithm [21]. Experimental study on randomly generated CCTCSPs demonstrates the efficiency of our exact method based on constraint propagation in case we look for a

complete solution while the SLS based method is the technique of choice in case we want to trade search time for the quality of the solution.

The rest of the paper is structured as follows. In the next section we introduce the CCTCSP framework through an example. Sections 3 and 4 are respectively dedicated to the constraint propagation techniques and SLS method for solving CCTCSPs. Section 5 describes the experimental comparative tests we have conducted on random CCTCSPs. Finally, concluding remarks are covered in Section 6.

2 Conditional and Composite Temporal Constraint Satisfaction Problems (CCTCSPs)

Managing conditional, composite and dynamic CSPs has already been reported in the literature [22, 23, 24, 25, 26, 27, 28, 29]. [22] introduced the notion of *Dynamic Constraint Satisfaction Problems* for configuration problems (renamed *Conditional Constraint Satisfaction Problems (CCSPs)* later). In contrast with the standard CSP paradigm, in a CCSP the set of variables requiring assignment is not fixed by the problem definition. A variable has either *active* or *nonactive* status. An activity constraint enforces the change of the status of a given variable from $nonactive$ to $active$. In [23], Freuder and Sabin have extended the traditional CSP framework by including the combination of three new CSP paradigms : *Meta CSPs, Hierarchical Domain CSPs, and Dynamic CSPs*. This extension is called *composite CSP*. In a composite CSP, the variable values can be entire sub CSPs. A domain can be a set of variables instead of atomic values (as it is the case in the traditional CSP). The domains of variable values can be hierarchically organized. The participation of variables in a solution is dynamically controlled by activity constraints. Jónsson and Frank [27] proposed a general framework using procedural constraints for solving dynamic CSPs. This framework has been extended to a new paradigm called Constraint-Based Attribute and Interval Planning (CAIP) for representing and reasoning about plans [28]. CAIP and its implementation, the EUROPA system, enable the description of planning domains with time, resources, concurrent activities, disjunctive preconditions and conditional constraints. The main difference, comparing to the formalisms we described earlier, is that in this latter framework [27] the set of constraints, variables and their possible values do not need to be enumerated beforehand which gives a more general definition of dynamic CSPs. Note that the definition of dynamic CSPs in [27] is also more general than the one in [26] since in this latter work variable domains are predetermined. Finally, in [29], Tsamardinos et al propose the Conditional Temporal Problem (CTP) formalism for Conditional Planning under temporal constraints. This model extends the well known quali-

tative temporal network proposed in [11] by adding instantaneous events (called observation nodes) representing conditional constraints.

We adopt both the CCSP [22] and the composite CSP [23] paradigms and extend the modeling framework TemPro [20] by including conditional temporal constraints and composite temporal events as shown in introduction. TemPro will then have the ability to transform constraint problems involving numeric information, symbolic information, conditional constraints and composite variables into the CCTCSP we have described in introduction. Comparing to the formalisms we mentioned above, ours has the following specificities.

1. Our work focuses on temporal constraints while the previous literature is on general constraints, if we exclude the work in [29] and [28]. Both these latter formalisms handle only quantitative time information while ours combines both quantitative and qualitative temporal constraints.

2. Our model is domain independent and is not tight to a particular area such as planning or scheduling. It can however be used in a large variety of applications involving symbolic and|or numeric temporal constraints. Moreover, the qualitative constraints are based on the whole Allen Algebra [9] which offers more expressiveness. Altough this will lead to NP-hard problems, the solving techniques that we will present in the next 2 Sections overcome this difficulty, in practice, as we will see in Section 5.

3. Our model is based on a discrete representation of time. Thus, events are defined on discrete values (numeric intervals). This offers an easier way to handle numeric temporal information with different granularities. It will also enable the constraint propagation techniques and approximation methods to be applied in a straight forward manner.

4. Numeric and symbolic temporal constraints as well as conditional constraints and composite variables, are managed within the same constraint graph.

In the following we will define the CCTCSP model and its corresponding network (graph representation) through an example.

Definition

A Conditional and Composite Temporal Constraint Satisfaction Problem (CCTCSP) is a tuple $\langle E, D_E, X, D_X, IV, C, A\rangle$, where

E=$\{e_1, \ldots e_n\}$ is a finite set of temporal variables that we call events. Events have a uniform reified representation made up of a proposition and its temporal qualification : $Evt = OCCUR(p, I)$ defined by Allen [9] and denoting the fact that the proposition p occurred over the interval I. For the sake of notation simplicity, an event is used in this paper to denote its temporal qualification.

D_E= $\{D_{e_1}, \ldots D_{e_n}\}$ is the set of domains of the events. Each domain D_{e_i} is the finite and discrete set of numeric intervals the event e_i can take. D_{e_i} is expressed by the fourfold $[begintime_{e_i}$, $endtime_{e_i}$, $duration_{e_i}$, $step_{e_i}]$ where $begintime_{e_i}$ and $endtime_{e_i}$ are respectively the earliest start time and the latest end time of the corresponding event, $duration_{e_i}$ is the duration of the event and $step_{e_i}$ defines the distance between the starting time of two adjacent intervals within the event domain. The discretization step $step_{e_i}$ allows us to handle temporal information with different granularities.

X= $\{x_1, \ldots x_m\}$ is the finite set of composite variables.

D_X=$\{D_{x_1}, \ldots D_{x_m}\}$ is the set of domains of the composite variables. Each domain D_{x_i} is the set of possible events the composite variable x_i can take.

IV is the set of initial variables. An initial variable can be a composite variable or an event. $IV \subseteq E \bigcup X$.

C = $\{C_1, \ldots C_p\}$ is the set of *compatibility constraints*. Each compatibility constraint is a qualitative temporal relation between two variables in case the two variables are events or a set of qualitative relations if at least one of the two variables involved is composite. A qualitative temporal relation is a disjunction of Allen primitives [9] (see table 1 for the definition of the Allen primitives).

A is the set of *activity constraints*. Each activity constraint has the following form : $(X_i = a_{i1}) \vee \ldots (X_i = a_{ip}) \rightarrow X_j$ where X_i and X_j are events or composite variables. This activity constraint is fired if X_i is active and is assigned one of the values $a_{i1} \ldots a_{ip}$ from its domain. The variable X_j will then be activated.

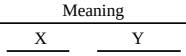
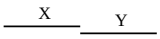
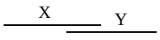
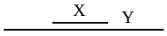
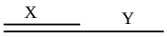
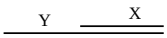
Relation	Symbol	Inverse	Meaning
X Before Y	B	Bi	
X Equals Y	E	E	
X Meets Y	M	Mi	
X Overlaps Y	O	Oi	
X During Y	D	Di	
X Starts Y	S	Si	
X Finishes Y	F	Fi	

Table 1. Allen primitives.

Example 1

Consider the following temporal problem:

> John, Mike and Lisa are going to see a movie on Friday. John will pick Lisa up and Mike will meet them at the theater. If John arrives at Lisa before 7:30, then they will stop at a convenient store to get some snacks and pops. It will take them 30 minutes to reach the theater if they stop at the store and 15 minutes otherwise. There are three different shows playing: $movie_1, movie_2$ and $movie_3$. If they finish the movie by 9:15, they will stop at a Pizza place 10 minutes after the end of the movie and will stay there for 30 minutes. John leaves home between 7:00 and 7:20. Lisa lives far from John (15 minutes driving). Mike leaves home between 7:15 and 7:20 and it takes him 20 minutes to go to the theater. $movie_1, movie_2$ and $movie_3$ start at 7:30, 7:45 and 7:55 and finish at 9:00, 9:10 and 9:20 respectively.

The goal here is to check if this story is consistent (has a feasible scenario). The story can be represented by the CCTCSP of figure 1. There are 6 events JPL, JL, JLS, M, $P1$ and $P2$ and 1 composite variable WM representing the following information :

- JPL : *John will pick Lisa up.*
- JL : *John and Lisa are going to see a movie.*
- JLS : *John and Lisa will stop at a convenient store.*

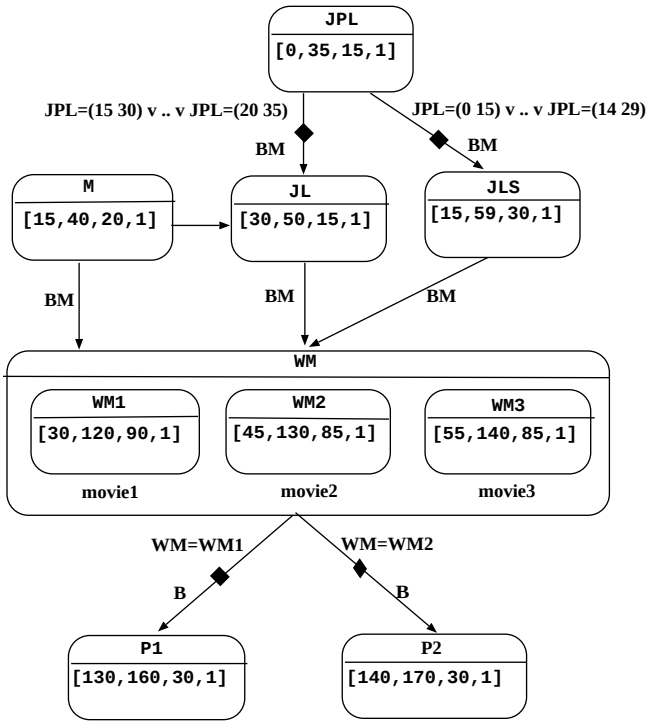


Figure 1. CCTCSP of example 1.

- M : *Mike is going to see a movie.*
- $P1$: *John, Mike and Lisa will stop at a Pizza place after watching* $movie_1$.
- $P2$: *John, Mike and Lisa will stop at a Pizza place after watching* $movie_2$.
- WM : *John, Mike and Lisa are watching a movie.* WM can take one of the following three values from its domain : WM_1, WM_2 and WM_3 corresponding to $movie_1, movie_2$ and $movie_3$ respectively.

Each event domain is represented by the fourfold $[begintime$, $endtime$, $duration$, $step]$. In the case of JPL, the domain is $[0, 35, 15, 1]$ where 0 (the time origin corresponding to 7:00) is the earliest start time, 35 is the latest end time, 15 is the duration, and 1 (corresponding to 1 min) is the discretization step. For the sake of simplicity all the events in this story have the same step. Arcs represent either a compatibility constraint or an activity constraint (case of arcs with diamond) between variables. The compatibility constraint is denoted by one or more qualitative relations. The activity constraint shows the condition to be satisfied and the qualitative relation between the two variables in case the condition is true. Each qualitative relation is a disjunction of some Allen primitives [9]. For example, the relation BM between JPL and JL denotes the disjunction $Before \vee Meets$.

3 Constraint Propagation for Solving CCTCSPs

Different methods for solving conditional CSPs have been reported in the literature [25, 22, 24, 30]. In [25], all possible CSPs are first generated from the CCSP to solve. CSP techniques are then used on the generated CSPs in order to look for a possible solution. Dependencies between the activity constraints are considered in order to generate a directed a-cyclic graph (DAG), where the root node corresponds to the set of initially active variables. Activity constraints are applied during the derivation of one total order from the partial order given by the resulting DAG. In [22, 24] resolution methods have been proposed and are directly applied on CCSPs. Maintaining arc consistency (MAC) is used to prune inconsistent branches by removing inconsistent values during the search [24]. The solving method starts by instantiating the active variables. For each active variable instantiation, the algorithm first checks the compatibility constraints and then activates the activity constraints. The method will then enforce look-ahead consistency (through arc consistency) along the compatibility constraints and prunes inconsistent values from the domains of future variables. When activity constraints come into play, newly activated variables are added to the set of future variables. MAC is then applied to the set of all active variables. In [30, 24], a CCSP is reformulated into an equivalent standard CSP. A special value "null" is added to the domains of all the variables which are not initially active. A variable instantiation with "null" indicates that the variable does not participate in the problem resolution. The CCSP is transformed into a CSP by including the "null" values. The disadvantage is that, in a large constraint problem, all variables and all constraints are taken into account simultaneously even if some are not relevant to the problem at hand. In the above methods, backtrack search is used for both the generation of possible CSPs and the search for a solution in each of the generated CSPs. Thus, these methods require an exponential time for generating the different CSPs and an exponential time for searching a solution in each generated CSP. Moreover these methods are limited to handle only activity constraints. The other problem of the above methods is the redundant work done when checking at each time the consistency of the same set of variables (subset of a given generated CSP).

The goal of the constraint propagation method we propose for solving CCTCSPs is to overcome, in practice, the difficulty due to the exponential search space of the possible TCSPs generated by the CCTCSP to solve and also the search space we consider when solving each TCSP. In the same way as reported in [22, 24], we use constraint propagation in order to detect earlier later failure. This will allow us to discard at the early stage any subset containing conflicting variables. The description

of the method we propose is as follows :

1. The method starts with an initial problem containing a list of initially activated temporal events and composite variables. Arc consistency is applied on the initial temporal events and composite variables in order to reduce some inconsistent values which will reduce the size of the search space. If the temporal events are not consistent (in the case of an empty domain) then the method will stop. The CCTCSP is inconsistent in this case.

2. Following the forward check principle [15], pick an active variable v, assign a value to it and perform arc consistency between this variable and the non assigned active variables. If one domain of the non assigned variables becomes empty then assign another value to v or backtrack to the previously assigned variable if there are no more values to assign to v. Activate any variable v' resulting from this assignment and perform arc consistency between v' and all the active variables. If arc inconsistency is detected then deactivate v' and choose another value for v (since the current assignment of v leads to an inconsistent CCTCSP). If v is a composite variable then assign an event to it (from its domain). Basically, this consists of replacing the composite variable with one event evt of its domain. We then assign a value to evt and proceed as shown before except that we do not backtrack in case all values of evt are explored. Instead, we will choose another event from the domain of the composite variable v or backtrack to the previously assigned variable if all values (events) of v have been explored. This process will continue until all the variables are assigned in which case we obtain a solution to the CCTCSP.

The arc consistency in the above two steps is enforced as follows.

- **Case 1 : the temporal constraint is** (Evt_1, Evt_2) **where** Evt_1 **and** Evt_2 **are two events**
  - The traditional arc consistency [14] is applied here i.e. each value a of Evt_1 should have a support in the domain of Evt_2.
- **Case 2 : the temporal constraint is** (X, Evt) **where** X **is a composite variable and** Evt **is an event**
  - Each value a, from the domain of a given event Evt_{X_k} within X, should have a support in the domain of Evt.
- **Case 3 : the temporal constraint is** (Evt, X)
  - Each value a, from the domain of Evt, should have a support in at least one domain of the events within X.

- **Case 4 : the temporal constraint is (X, Y) where X and Y are two composite variables**
  - Apply case 2 between X and each event Evt_{Y_k} within Y.

Using the above rules, we have implemented a new arc consistency algorithm for CCTCSPs as shown in Figure 2. This algorithm is an extension of the well known AC-3 procedure [14].

$REVISE(D_i, D_j)$
  $REVISE \leftarrow false$
  For each value $a \in D_i$ do
    if not compatible(a, b) for any value $b \in D_j$ then
      remove a from D_i
      $REVISE \leftarrow true$
    end if
  end for

$REVISE\_COMP\ (D_i, D_j)$
  $REVISE\_COMP \leftarrow false$
  if i is a single variable and j is a composite variable
    $D_{tmp} \leftarrow \emptyset$
    For each event $k \in D_j$ do
      $D \leftarrow\ D_i\ -\ D_{tmp}$
      $REVISE\_COMP \leftarrow REVISE\_COMP\ OR\ REVISE(D, D_k)$
      $D_{tmp} \leftarrow D_{tmp} \cup D$
    end for
    $D_i \leftarrow D_{tmp}$
  end if
  if i is a composite variable and j is a single variable
    For each event $k \in D_i$ do
      $REVISE\_COMP \leftarrow REVISE\_COMP\ OR\ REVISE(D_k, D_j)$
    end for
  end if
  if i and j are composite variables
    For each event $k \in D_i$ do
      $REVISE\_COMP(D_k, D_j)$
    end for
  end if

$AC-3-CCTCSP$
  Given a graph $G = (X, U)$
  $Q \leftarrow \{(i, j) | i, j \in U\}$
  while $Q \neq\ Nil$ do
    $Q \leftarrow Q - \{(i, j)\}$
    if i or j is composite variable
      if $REVISE\_COMP(D_i, D_j)$ then
        $Q \leftarrow \cup\{(k, i) | k, i \in U$ and $k \neq j\}$
      end if
    else if $REVISE(D_i, D_j)$ then
        $Q \leftarrow \cup\{(k, i) | k, i \in U$ and $k \neq j\}$
      end if
    end if
  end while

Figure 2. AC-3 for CCTCSPs.

4 Approximation methods for CCTCSPs

The method we presented in the previous Section is an exact technique that guarantees a complete solution. The method suffers however from its exponential time cost as we will see in the next Section. In many real-life applications where the execution time is an issue, an alternative will be to trade the execution time for the quality of the solution returned (number of solved constraints). This can be done by applying approximation methods such as local search and where the quality of the solution returned is proportional to the running time. In this Section we will study the applicability of a local search technique based on the Min-Conflict-Random-Walk (MCRW) [21] algorithm for solving CCTCSPs. MCRW has already been applied to solve TCSPs [20]. Basically, the method consists of starting from a complete assignment of temporal intervals to events and iterates by improving at each step the quality of the assignment (number of solved constraints) until a complete solution is found or a maximum number of iterations is reached. Given the dynamic aspect of CCTCSPs (some variables are added|removed dynamically during the resolution process) we propose the following algorithm based on MCRW for solving CCTCSPs.

MCRW-CCTCSP

1. The algorithm starts with a random assignment of values to the initial variables. If the initial variable is an event then it will be randomly assigned a value (temporal interval) from its domain. In the case where the initial variable is composite then it will be replaced by one variable selected randomly from its domain. This latter variable will then be randomly assigned a value from its domain.

2. Activate any variable where the activating condition is true and randomly assign to it a value from its domain as shown in the previous step.

3. If a complete solution is not found and the maximum number of iterations is not reached, randomly select an active variable v and proceed with one of the following cases:

   - If v belongs to the domain of a given composite variable X then select the pair $< v_i, int_{v_i} >$ that increases the quality of the current solution (number of solved constraints). v_i belongs here to the domain of X and int_{v_i} is a value of v_i's domain,

- otherwise, assign to v a value that increases the quality of the solution.

4. Deactivate any variable activated by the old assignment of v and goto 2.

5 Experimentation

In order to evaluate the methods we propose, we have performed experimental tests on randomly generated consistent CCTCSPs. The experiments are performed on a PC Pentium 4 computer under Linux system. All the procedures are coded in C/C++. Consistent CCTCSPs are generated from consistent TCSPs. A consistent TCSP of size N (N is the number of variables) has at least one complete numeric solution (set of N numeric intervals satisfying all the constraints of the problem). Thus, to generate a consistent TCSP we first randomly generate a numeric solution (set of N numeric intervals), extract the symbolic Allen primitives that are consistent with the numeric solution and then randomly add other numeric and symbolic constraints to it. After generating a consistent TCSP, some of the temporal events are randomly picked and grouped in subsets to form composite variables. Each activity constraint $V_i \stackrel{V_i=a}{\rightarrow} V_j$ is generated by randomly choosing a pair of variables (V_i , V_j) and a value a from the domain of V_i. This activity constraint activates the variable V_j if V_i is activated and is assigned the value a. The generated TCSPs are characterized by their tightness, which can be measured, as shown in [31], as the fraction of all possible pairs of values from the domain of two variables that are not allowed by the constraint. The tests we have performed compare the following four propagation strategies.

Forward Check (FC). This is the strategy we have described in Section 3 which consists basically of maintaining the arc consistency, during the search, between the current variable (the variable that we are assigning a value) and the future active variables (variables not yet assigned).

Full Look Ahead (FLA). This strategy maintains a full arc consistency on the current and future active variables.

FC+. Same as FC except that the applicability of the arc consistency is extended to non active variables as well.

FLA+. Same as FLA except that the applicability of the arc consistency is extended to non active variables as well.

Figure 3 and table 2 present the results of comparative tests performed on random CCTCSPs where the total number of variables is 150, the number of composite variables is 10 and their domain size is 5, the domain size of the events is 30 and the number of activity constraints is 500. In each test, the methods are executed on 100 instances and the average running

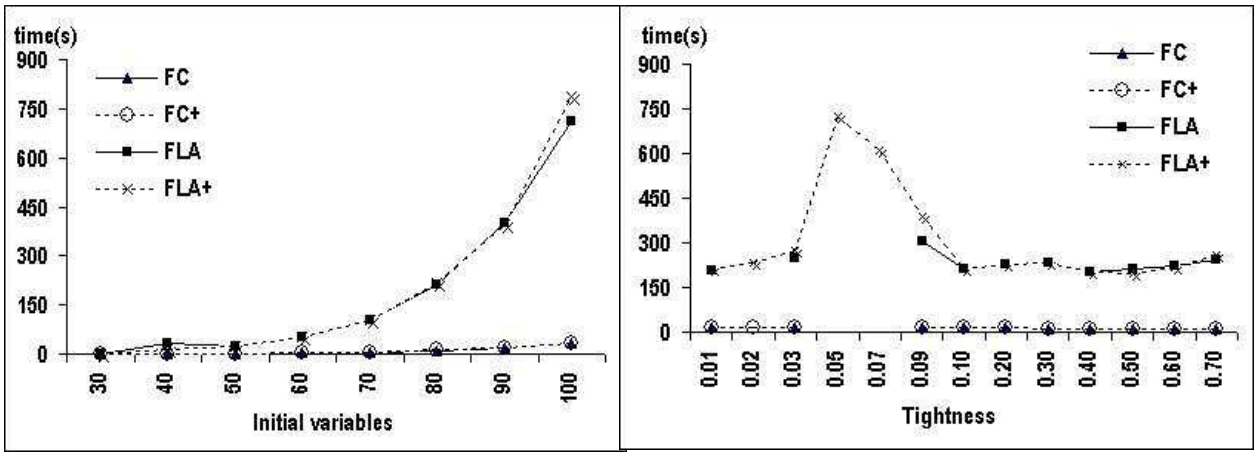


Figure 3. Comparative tests on random CCTCSPs.

time (in seconds) is taken. The left chart of figure 3 presents comparative results of the four constraint propagation strategies when the number of initial variables varies from 30 until 100. The tightness is equal here to 0.06 which corresponds to the hardest problems. The right chart of figure 3 presents the comparison of the four strategies when the tightness of the TCSPs, from which the CCTCSPs are generated, varies from 0.1 to 0.7. The number of initial variables is equal to 80. As we can easily see FC and FC+ outperform FLA and FLA+ in all cases in the left chart and in most of the cases in the right chart. However, in the right chart, the strategy of choice in the phase transition is FLA+ since it is the only strategy which returns a complete solution in these situations. In both charts FC and FC+ have similar running times. Table 2 compares the four constraint propagation strategies and the MCRW method we described in Section 4, when the percentage of constraints varies from 0.1 (10%) until 1 (100% which corresponds to a complete constraint graph). Since MCRW is an approximation method, we report in each case the number of times (in percent) this method succeded to provide a complete solution. As we can easily see, MCRW is the fastest method for under constrained problems (where the percentage of constraints is between 0.1 and 0.3) while some propagation strategies (FC and FLA) fail sometimes to to find a solution. For middle and over constrained problems, MCRW does not always guarantee a complete solution when the percentage of constraints is between 0.4 and 0.5 and fails in all the cases when the percentage is between 0.6 and 0.9. FC, FC+ and FLA also have difficulty to find a solution in the case of middle constrained problems and the method of choice in this case is FLA+. In the case of over constrained problems (0.8 to 1) FC+ is the fastest complete methods while MCRW has better running time but succeds only in 30% of the cases for complete graphs.

% of Cons	MCRW		FC	FC+	FLA	FLA+
	Time	success(%)				
0.1	0.1	100	10	11	173	173
0.2	0.1	100	10	10	174	173
0.3	0.1	100	-	11	-	193
0.4	8.7	70	-	-	-	209
0.5	14.8	70	-	19	-	987
0.6	18.3	0	12	12	200	209
0.7	18.5	0	-	-	201	187
0.8	16.9	0	-	16	-	220
0.9	17.9	0	16	16	198	194
1	14.9	30	22	22	217	204

Table 2. Comparative tests on random CCTCSPs.

6 Conclusion

We have presented in this paper a CSP based framework for representing and managing numeric and symbolic temporal constraints, activity constraints and composite variables with a unique constraint network that we call Conditional Composite Temporal Constraint Satisfaction Problem (CCTCSP). Solving a CCTCSP consists of finding a solution for one of its possible TCSPS. This requires an algorithm with $O(D^N d^M)$ time cost where N, D, M and d are respectively the number of events and their domain size, the number of composite variables and their domain size. In order to overcome this difficulty in practice, we have proposed 2 methods respectively based on constraint propagation and stochastic local search. Constraint propagation prevents earlier later failure which improves, in practice, the performance in time of the backtrack search. On the other hand, because of its polynomial time cost, the stochastic local search method has better time performance than constraint propagation but does not always guarantee a complete solution. Experimental tests we have performed on randomly generated CCTCSPs demonstrates the efficiency of MCRW method for under constrained problems while variants of the full look ahead and the forward check strategies are the methods of choice respectively for middle constrained and over constrained problems. For these kinds of problems MCRW can be used in case we want to trade search time for the quality of the solution returned (number of solved constraints).

References

[1] Baptiste, P., Pape, C.L.: Disjunctive constraints for manufacturing scheduling : Principles and extensions. In: Third International Conference on Computer Integrated Manufacturing, Singapore (1995)

[2] Ghallab, M., Laruelle, H.: Representation and Control in IxTeT, a Temporal Planner. In: AIPS 1994. (1994) 61–67

[3] Laborie, P., Ghallab, M.: Planning with Sharable Resource Constraints. In: IJCAI-95. (1995) 1643–1649

[4] Laborie, P.: Resource Temporal Networks: Definition and Complexity. In: Eighteenth International Joint Conference on Artificial Intelligence (IJCAI'03). (2003) 948–953

[5] Song, F., Cohen, R.: Tense interpretation in the context of narrative. In: AAAI'91. (1991) 131–136

[6] Hwang, C., Shubert, L.: Interpreting tense, aspect, and time adverbials: a compositional, unified approach. In: Proceedings of the first International Conference on Temporal Logic, LNAI, vol 827, Berlin (1994) 237–264

[7] Golumbic, C., Shamir, R.: Complexity and algorithms for reasoning about time: a graphic-theoretic approach. Journal of the Association for Computing Machinery **40(5)** (1993) 1108–1133

[8] Dean, T.: Using Temporal Hierarchies to Efficiently Maintain Large Temporal Databases. JACM (1989) 686–709

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

[10] Vilain, M., Kautz, H.: Constraint propagation algorithms for temporal reasoning. In: AAAI'86, Philadelphia, PA (1986) 377–382

[11] Dechter, R., Meiri, I., Pearl, J.: Temporal Constraint Networks. Artificial Intelligence **49** (1991) 61–95

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

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

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

[15] Haralick, R., Elliott, G.: Increasing tree search efficiency for Constraint Satisfaction Problems. Artificial Intelligence **14** (1980) 263–313

[16] Kautz, H., Ladkin, P.: Integrating metric and qualitative temporal reasoning. In: AAAI'91, Anaheim, CA (1991) 241–246

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

[18] Thornton, J., Beaumont, M., Sattar, A., Maher, M.: A Local Search Approach to Modelling and Solving Interval Algebra Problems. Journal of Logic and Computation **14** (2004) 93–112

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

[20] Mouhoub, M.: Reasoning with numeric and symbolic time information. Artificial Intelligence Review **21** (2004) 25–56

[21] Selman, B., Kautz, H.: Domain-independent extensions to gsat: Solving large structured satisfiability problems. In: IJCAI-93. (1993) 290–295

[22] Mittal, S., Falkenhainer, B.: Dynamic constraint satisfaction problems. In: Proceedings of the 8th National Conference on Artificial Intelligence, Boston, MA, AAAI Press (1990) 25–32

[23] Sabin, D., Freuder, E.C.: Configuration as composite constraint satisfaction. In Luger, G.F., ed.: Proceedings of the (1st) Artificial Intelligence and Manufacturing Research Planning Workshop, AAAI Press (1996) 153–161

[24] D. Sabin, E.C.F., Wallace, R.J.: Greater efficiency for conditional constraint satisfaction. Proc., Ninth International Conference on Principles and Practice of, Constraint Programming - CP 2003 **2833** (2003) 649–663

[25] Gelle, E., Faltings, B.: Solving mixed and conditional constraint satisfaction problems. Constraints **8** (2003) 107–141

[26] Dechter, R., Dechter, A.: Belief maintenance in dynamic constraint networks. In: 7th National Conference on Artificial Intelligence, St Paul (1988) 37–42

[27] Jónsson, A.K., Frank, J.: A framework for dynamic constraint reasoning using procedural constraints. In: ECAI 2000. (2000) 93–97

[28] J. Frank, A.K.J.: Constraint-based attribute and interval planning. Constraints **8** (2003) 339–364

[29] Tsamardinos, I., Vidal, T., Pollack, M.E.: CTP: A New Constraint-Based Formalism for Conditional Temporal Planning. Constraints **8** (2003) 365–388

[30] Gelle, E.: On the generation of locally consistent solution spaces in mixed dynamic constraint problems. Ph.D.thesis **1826** (1998) 101–140

[31] Sabin, D., Freuder, E.C.: Contradicting conventional wisdom in constraint satisfaction. In: Proc. 11th ECAI, Amsterdam, Holland (1994) 125–129