Solving Conditional and Composite Temporal Constraints

Malek Mouhoub and Amrudee Sukpan
Department of Computer Science, University of Regina
3737 Wascana Parkway, Regina SK, Canada, S4S 0A2
{mouhoubm,supkan1a}@cs.uregina.ca

Abstract

One of the main challenges when designing constraint based systems in general and those involving temporal constraints in particular, is the ability to deal with conditional constraints and composite variables. Indeed, in this particular case the set of variables involved by the constraint problem to be solved is not known in advance. More precisely, while some variables (called initial variables) 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 variables whose values are the possible variables each composite variable can take. We propose in this paper a method based on constraint propagation for solving efficiently constraint problems involving numeric and symbolic temporal constraints, composite variables and activity constraints. We call these latter problems Conditional and Composite Temporal Constraint Satisfaction Problems (CCTCSPs). Experimental evaluation conducted on randomly generated CCTCSPs demonstrates the efficiency of our method to solve these problems especially when using the forward check strategy during the search.

1. Introduction

Representing and reasoning about numeric and/or symbolic aspects of time is crucial in many real world applications such as planning, scheduling, natural language processing, molecular biology and temporal database. 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 [1, 19, 4, 17]. 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 [12, 9, 7].

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 recently proposed to reasoning on the metric or the symbolic aspects of time (respectively through metric and qualitative networks), little work has been developed to manage both types of information. Meiri [10] 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[8] have proposed a model allowing the representation and processing of metric temporal information in the form of a system of simple linear inequalities to encode metric relations between time points, and systems of binary constraints in Allen's qualitative temporal calculus to encode qualitative relations between time points. In the Kautz and Ladkin approach, both kinds of constraints are independently processed in separate networks.

In a previous work [14, 13], we have developed a temporal model, TemPro, based on Allen's interval algebra [1] 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 CSP (CSP constraints can be unary or binary) where variables are temporal events defined on domains of numeric intervals and binary constraints between variables correspond to disjunctions of Allen primitives. We call it Temporal Constraint Satisfaction Problem (TCSP). Note that this name and the corresponding acronym was used in [4]. The 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 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 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 [10]. 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 :

- Handling conditional temporal constraints. This is the case where temporal variables 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 temporal variable can be activated by default (in the initial problem) or by an activity constraint. Here an activity constraint, having the following form : $Ev_i, \cdots, Ev_j \overset{condition}{\rightarrow} Ev_q, \cdots, Ev_r$ is fired if *condition* holds and the events $Ev_i, \cdots, Ev_j$ are active. This activity constraint will then activate the events $Ev_q, \cdots, Ev_r$.

- 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.

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 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 $O(D^N \ x \ d^M)$. To overcome this difficulty in practice, we propose a solving method based on constraint propagation techniques and backtrack search. The backtrack search is used here for both the generation of the possible CTCSPs the CCTCSP involves, and the resolution of each CTCSP. The algorithm will stop if a consistent CTCSP has been found. The constraint propagation through local consistency is used here before and during the backtrack search to detect earlier any inconsistent CTCSP.

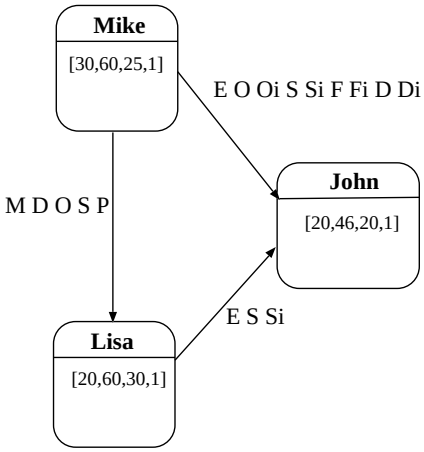


Figure 1. A Temporal Constraint Satisfaction Problem (TCSP).

This will save a lot of efforts needed to check the consistency of the CCTCSP. In order to evaluate the performance in time of the method we propose, we have conducted experimental tests on randomly generated CTCSPs and CCTCSPs. The results demonstrate the efficiency of our method to solve these problems especially when using the forward check strategy during the search.

The rest of the paper is structured as follows. In the next section we present our modeling framework TemPro and the corresponding solving methods. Section 3 introduces the CCTCSP through different examples. In Section 4 we describe a resolution method for solving CCTCSPs. Section 5 describes the experimental tests we have conducted on random CTCSPs and CCTCSPs. Finally, concluding remarks are covered in Section 6.

2. Managing Numeric and Symbolic Temporal Constraints

Example 1

Let us consider the following temporal reasoning problem :

> *John, Mike and Lisa work for a company. It takes John **20 minutes**, Mike **25 minutes** and Lisa **30 minutes** to get to work. Every day, John left home **between 7:20 and 7:26**. Mike arrives at work **between 7:55 and 8** and Lisa arrives at work **between 7:50 and 8**. We also know that John and Mike meet on their way to work. Mike arrives to work before Lisa and that Lisa and John goes to work at the same time.*

Using our modeling framework TemPro [14, 13], the problem above is translated to the TCSP represented by the graph in figure 1. The nodes of the graph correspond to the three

events of our story, namely: John, Mike and Lisa are going to work. Events have here a uniform reified representation made up of a proposition and its temporal qualification : $Evt = OCCUR(p,I)$ defined by Allen [1] and denoting the fact that the proposition p occurred over the interval I. For the sake of notation simplicity, an event is used in this paper to denote its temporal qualification. The domains of the three events are the possible time intervals each event can take. Each event domain is expressed by the fourfold [*begintime*, *endtime*, *duration*, *step*] where *begintime* and *endtime* are respectively the earliest start 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 temporal window. Arcs are labeled with the disjunctive Allen primitives [1] between events. Table 1 illustrates the 13 Allen primitives. For example, the relation S $\vee$ Si $\vee$ E (denoted E S Si in the graph) between John's and Lisa's events indicates the fact that the start times of John and Lisa are equal. After we translate the above story into the TCSP shown in figure 1, we may want to ask queries such as : is the story consistent? what are the possible arrival times of Lisa? what is the earliest start time of Mike? check if a given scenario is consistent. To do that, we have defined a solving method structured in the following steps :

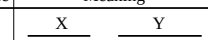
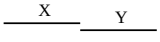
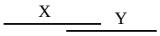
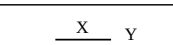
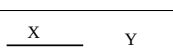
Numeric $\rightarrow$ Symbolic Conversion. Perform the numeric $\rightarrow$ symbolic conversion on all the constraints. If one symbolic relation becomes empty then the constraint network is not consistent. The numeric $\rightarrow$ symbolic conversion works as follows : from the numeric information, 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. We use the following rules to extract the symbolic relations from the numeric ones. We assume here that e_i and e_j are two events, r_{ij} is the symbolic relation between them (initially set to the disjunction of the 13 Allen primitives), and inf_i, inf_j, sup_i, sup_j, d_i and d_j are respectively the earliest start time of e_i, earliest start time of e_j, latest end time of e_i, latest end time of e_j, duration of e_i and duration of e_j.

1. if $inf_i > sup_j$ then $r_{ij} \leftarrow$ Bi,
2. if $sup_i < inf_j$ then $r_{ij} \leftarrow$ B,
3. if $d_i < d_j$ then remove { E, Si, Fi, Di } from r_{ij},
4. if $d_i > d_j$ then remove { E, S, F, D } from r_{ij},
5. if $d_i = d_j$ then remove { D, Di, S, Si, F, Fi } from r_{ij},
6. if $inf_i + d_i > sup_j - d_j$ then remove { M, B } from r_{ij},
7. if $sup_i - d_i < inf_j + d_j$ then remove { Mi, Bi } from r_{ij},
8. if $inf_i > sup_j - d_j$ then remove { E, B, M, S, Si, O, Di } from r_{ij},
9. if $inf_i + d_i > sup_j$ then remove { E, B, M, F, Fi, D} from r_{ij},
10. if $sup_i < inf_j + d_j$ then remove { F, Fi } from r_{ij},
11. if $sup_i - d_i < inf_j$ then remove { S, Si, E } from r_{ij}.

Local Consistency. Perform the path consistency algorithm PC-2 [18] on the symbolic relations and the arc consistency algorithm AC-3 [9, 20, 2] on the temporal windows. If the resulting graph is not path or arc consistent then it is not consistent.

Backtrack Search. Perform a backtrack search algorithm in order to look for a possible solution to the problem. Forward check through local consistency is used here during the backtrack search in order to prevent earlier later failure.

Table 1. Allen primitives.

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	

When applying the above method to the problem of example 1, we obtain the following solution : { Mike : (30 55), John : (26 46), Lisa : (26 56) }.

3. Managing Composite Variables and Conditional Constraints

Managing conditional and composite CSPs has already been reported in the literature [11, 16, 3, 6]. [11] 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 [16], 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.

We adopt the above two paradigms (CCSPs and composite CSPs) and extend our modeling framework TemPro 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 defined in introduction. Let us illustrate this transformation using the following examples.

Example 2

Consider the temporal reasoning problem we have seen in example 1. Let us assume now that we add the following information :

> *John may stop at a gas station. It takes him **10 minutes** to drive to the gas station and fill up the gas. If John stops at the gas station, it takes him **15 minutes** to arrive at work **between 7:50 and 7:55**. However John will not stop at the gas station if he is late. John will not meet Mike if he stops at the gas station.*

John's event will be represented here as a composite temporal event whose domain contains the following values : *JWork* and *JGas* which represent the events "*John goes directly to work*" and "*John goes to the gas station*" respectively. In addition, we define the following activity constraint : *JGas*$\longrightarrow$*GWork*. *GWork* represents the event : *John goes from the gas station to work*. Figure 2 describes the CCTCSP corresponding to the examples 1 and 2, including composite events and activity constraints. In the figure, temporal events are represented by rectangle solid line nodes while composite events are represented by rectangle dash line nodes. The activity constraint is denoted by a solid arrow with small diamond in the middle and is labeled with the condition and the temporal constraint that will link the two events if the activity constraint is fired. A temporal constraint between two temporal events is represented by a solid arrow while a temporal constraint involving at least one composite event, has one of the following forms :

Case 1 : The temporal constraint involves an event Evt **and a composite variable** X**.**

- A solid arrow labeled with a temporal relation R, in the case where all the temporal events in the domain of X have the same temporal relation R with Evt. This is, for example, the case of the relation "E S Si" between the event Lisa and the composite variable John.
- A dash arrow labeled with a list of temporal constraints, $R_1, \ldots, R_n$, where each of the R_is denotes respectively a temporal constraint between Evt and each temporal event in the domain of X. This is the case of the relations "E O Oi S Si F Fi D Di" and "I" between the event Mike and the composite variable John. "I" denotes here the disjunction of the 13 Allen primitives.

Case 2 : The temporal constraint involves 2 composite variables X **and** Y**.**

- A solid arrow labeled with a temporal relation R, in the case where all the temporal events in the domain of X have the same temporal relation with all the temporal events in the domain of Y.
- A dash line labeled with the list : $Evt_{X_1}\ R_{11}\ Evt_{Y_1}, \ldots, Evt_{X_n}\ R_{nm}\ Evt_{Y_m}$ where $\{Evt_{X_1}, \ldots, Evt_{X_n}\}$ and $\{Evt_{Y_1}, \ldots, Evt_{Y_m}\}$ are respectively the domains of X and Y and each of the R_{ij}s is a temporal constraint between Evt_{X_i} and Evt_{Y_j}.

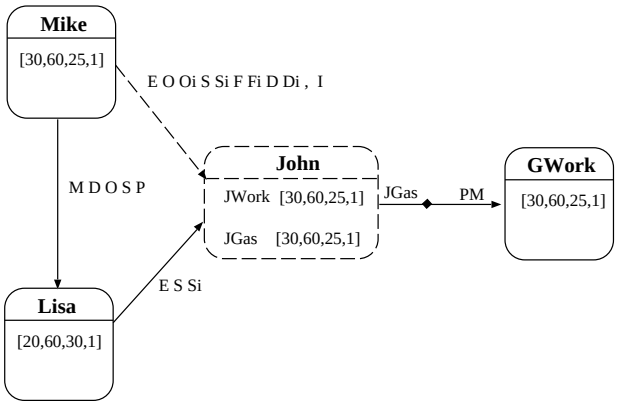


Figure 2. CCTCSP representing the examples 1 and 2.

Let us consider a more complex example that can be represented by a CCTCSP.

Example 3

> *Tom is going to ski. He can ski either at Snowbird or Park City. The top picture of figure 3 shows the path from his house to the two skiing places and the time distance between each pair of points. If he wants to ski at Snowbird, he should arrive after 1 pm, because they offer great discounts in the afternoons. If he skis at Park City, he should arrive at point B no later than 11 am, since the traffic is very heavy afterward. Today Tom leaves home between 10:30 am and 11:00 am.*

We have three major composite variables in this story :

Rest_AB : Tom is going from *Rest area* to point A or from *Rest area* to point B (denoted respectively by the temporal events *Rest_A* and *Rest_B*).

A_SnowB : Tom is going from point A to *SnowBird* or from point A to point B (denoted respectively by the temporal events *A_SnowBird* and *A_B*).

B_ParkA : Tom is going from point *B* to *ParkCity* or from point *B* to point *A* (denoted respectively by the temporal events *B_ParkCity* and *B_A*).

A_SnowB will be activated if and only if *Rest_A* is selected whereas *B_ParkA* will be activated if and only if *Rest_B* is selected. The CCTCSP of this problem is shown in the temporal graph of figure 3.

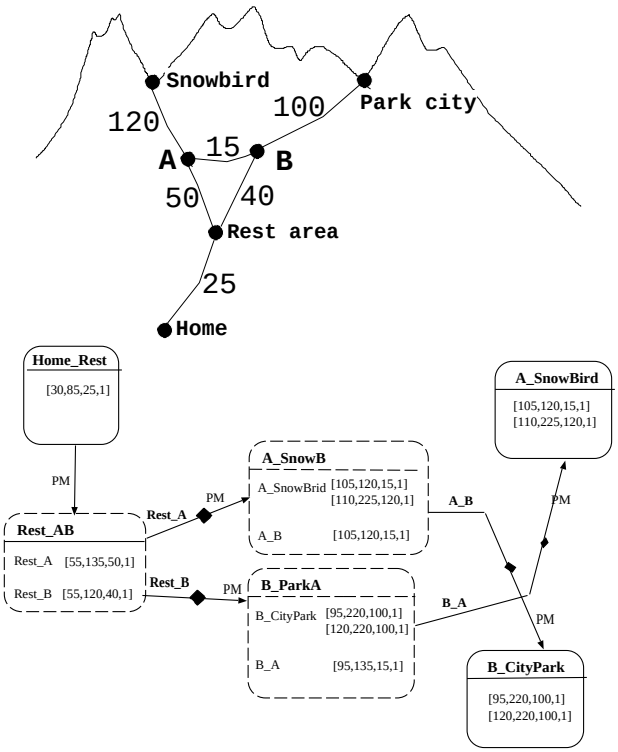


Figure 3. Skiing example.

4. Solving CCTCSPs

Different methods for solving conditional CSPs have been reported in the literature [6, 11, 3, 5]. In [6], 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 [11, 3] 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 [3]. 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 [5, 3], 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 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. Thus, in addition to the fact that our method handles composite variables, conditional constraints, and numeric and symbolic temporal constraints, the advantages of our method comparing to the above methods are as follows :

- The consistency check performed on a given subset of variables is performed once. The result is reused anytime the same subset is considered.
- In the same way as reported in [11, 3], we use constraint propagation with the forward check strategy 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. Numeric $\rightarrow$ symbolic conversion and arc

consistency are applied on the initial temporal events and composite variables in order to reduce respectively some inconsistent Allen primitives and temporal intervals which will reduce the size of the search space. If the temporal events are not consistent (in the case of an empty symbolic relation or an empty domain) then the method will stop. The CCTCSP is inconsistent in this case. When applied to temporal events, the numeric $\rightarrow$ symbolic conversion is used in the same way as explained in Section 2. When numeric $\rightarrow$ symbolic conversion is applied to pairs of variables involving at least one composite variable then this conversion method is processed between each temporal event within the first and the second composite variables (in the case where both variables are composite variables) or between each temporal event within the composite variable; and the temporal variable (in the case where only the first variable is composite). We use a modified version of AC-3 algorithm [9, 20, 2] to check the arc consistency when some of the variables are composite. The algorithm we propose is illustrated in figure 4. Here the arc consistency between a temporal event and a composite variable is handled as follows. If the first variable is composite, such as a constraint (X Evt) where X is composite and Evt is a temporal event, then each value a, from the domain of a given event Evt_{X_k} within X, should have a support in the domain of Evt (value in Evt domain which is consistent with a) otherwise a will be removed. In the case where the second variable is composite, such as a constraint (Evt X) where X is composite and Evt is a temporal event then each value a, from the domain of Evt, should have a support in at least one domain of the events within X otherwise a will be removed. The arc consistency involving a pair of composite variables consists of applying the above method to each event from the first composite variable, and the second composite variable.

2. Activate any variable if the activating condition is true. Use numeric $\rightarrow$ symbolic conversion and arc consistency as shown above to maintain the local consistency of the old and the new activated events. The method will stop if a local inconsistency is detected since the CCTCSP is inconsistent in this case.

3. Pick a variable from the list of composite variables.

4. Assign a value (temporal event) to the chosen composite variable. In other words, we replace the composite variable with a chosen event from its domain. Run numeric $\rightarrow$ symbolic conversion and arc consistency on the old events and the new one. If an inconsistency is detected backtrack and choose another temporal event for the composite variable. This process will continue until we find an event from the domain of the composite variable which is consistent with the old events. If such event is not found, backtrack to the previously assigned composite variable and goto step 4.

5. Activate any variable generated by the added event of the last step and run numeric $\rightarrow$ symbolic conversion and arc consistency. If a local inconsistency is detected, backtrack to step 4 and choose another value for the current composite variable.

6. Repeat steps 3, 4 and 5 until all the composite variables are processed. As a result, an arc consistent TCSP will be generated.

7. Perform path consistency on the generated TCSP. If the resulting TCSP is not path consistent, backtrack to step 4 and choose another event for the last assigned composite variable.

8. Perform a backtrack search algorithm on the generated TCSP. Return a solution if it is found otherwise backtrack to step 4 and choose another event for the last assigned composite variable. If all events of a given composite variable have been tried without success then the CCTCSP is inconsistent.

5. Experimentation

In order to evaluate the method we propose, we have performed experimental tests on randomly generated consistent CTCSPs and CCTCSPs. The experiments are performed on a SUN SPARC Ultra 5 station. All the procedures are coded in C/C++. Consistent CTCSPS and 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 [15], as the fraction of all possible pairs of values from the domain of two variables that are not allowed by the constraint.

The top chart of figure 5 presents the results on random CTCSPs. The number of variables and domain size are respectively 70 and 35. The percentage of initial variables is 70% of the total number of variables (namely 50 variables). The number of activity constraints is 0.1% of the total number of possible activity constraints. The total number of possible

$REVISE(D_i, D_j)$
    $REVISE \leftarrow false$
    For each value $a \in D_i$ do
        if not compatible(a,b) for any value $b \in D_i$ 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
        $D_{i_{tmp}} = 0$ (temporal domain of i)
        For each value $D_{jk} \in D_j$ do
            $D_{ik} \leftarrow D_i\ XOR\ D_{i_{tmp}}$
            $REVISE\_COMP \leftarrow REVISE\_COMP\ ||\ REVISE(D_{ik}, D_{jk})$
            $D_{i_{tmp}} = D_{i_{tmp}} \cup D_{ik}$
        end for
        $i = D_{i_{tmp}}$
    end if
    if i is a composite variable and j is a single variable
        For each value $D_{ik} \in D_i$ do
            $REVISE\_COMP \leftarrow REVISE\_COMP\ ||\ REVISE(D_{ik}, D_j)$
        end for
    end if
    if i and j are composite variables
        For each value $D_{ik} \in D_i$ do
            $REVISE\_COMP \leftarrow REVISE\_COMP\ ||\ REVISE\_COMP(D_{ik}, D_j)$
        end for
    end if

AC_COMP
    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
                $Q \leftarrow \cup\{(k,i)|(k,i) \in U \cap k \neq j\}$
            end if
        else if $REVISE(D_i, D_j)$ then
                $Q \leftarrow \cup\{(k,i)|(k,i) \in U \cap k \neq j\}$
            end if
        end if
    end while

where :
D_i : domain of i
D_{jk} : domain of variable k of composite variable j
XOR : exclusive or operation

Figure 4. AC-3 for CCTCSPs.

activity constraints is $N(N-1)D$ where D is the domain size of the temporal events. Each curve of the chart corresponds to a given search strategy [7] used within the method we propose. The forward check strategy (FC) consists of maintaining the arc consistency, during the search, between the current variable (the variable that we are assigning a value) and the future variables (variables not yet assigned). The full look-ahead strategy (FLA) maintains a full arc consistency on the current and future variables. As we can easily see on the figure, the results of FC outperforms those of FLA for all kinds of problems (from under-constrained to over-constrained problems). Indeed, although the FLA strategy removes more inconsistent values than the FC one during the backtrack search, FLA requires more time to do the arc consistency check which affects the total running time of the method. The total time needed by the forward check technique in the case of the hardest CTCSPs (corresponding to the phase transition) is less than 125 seconds.

The bottom chart of figure 5 shows the results of tests performed on random CCTCSPs. The number of variables and domain size are 50 and 25 respectively. The number of composite variables is 5. The number of events within each composite variable is 2. We report here only the running times of the FC technique (FLA is even slower in the case of CCTCSPs). As we can see FC requires less than 20 seconds to solve even the hardest CCTCSPs.

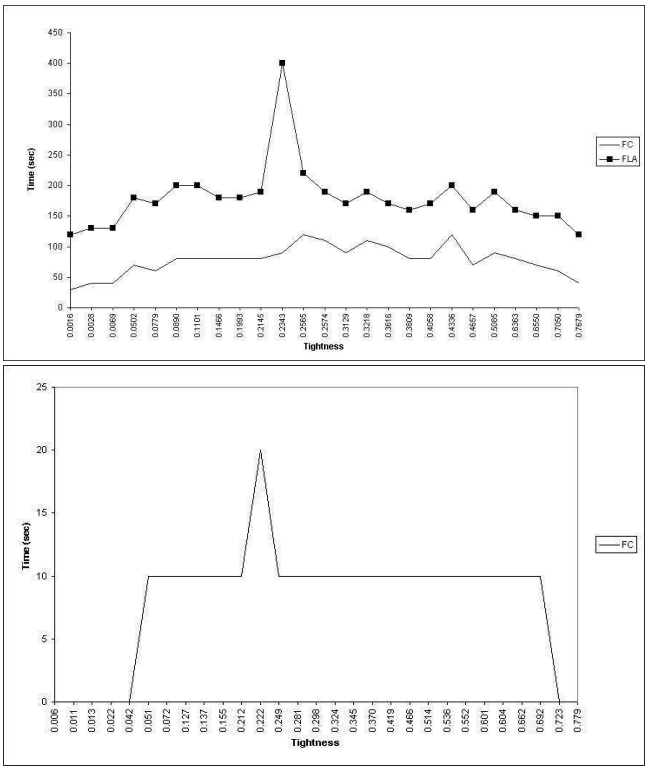


Figure 5. Experimental tests on random CTCSPs and CCTCSPs.

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. Solving a TCSP consists of finding a complete assignment that satisfies all the temporal constraints. This requires an exponential time algorithm, $O(D^N)$, where D is the domain size of the temporal events and N the number of temporal events involved by the TCSP. The number of possible TCSPs is exponential in number of composite variables, $O(d^M)$, where d is the domain size of the composite variables and M the number of composite variables. Solving a CCTCSP will then require an algorithm with $O(D^N d^M)$ time cost. In order to overcome this difficulty in practice, we have proposed a method based on constraint propagation and backtrack search. The goal of the method is to check the consistency of the CCTCSP and to return a feasible scenario (solution) if this latter is consistent. Constraint propagation is performed at both the symbolic and the numeric levels in order to prevent earlier later failure which will improve, in practice, the performance in time of the backtrack search algorithm.

In order to evaluate, in practice, the performance of the method we propose, we have conducted experimental tests on randomly generated CTCSPs and CCTCSPs. The results demonstrate the efficiency of our method especially when using the forward check strategy.

References

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

[2] 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.

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

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

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

[6] E. Gelle and B. Falting. Solving mixed and conditional constraint satisfaction problems. *Constraints*, 8:107–141, 2003.

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

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

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

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

[11] S. Mittal and B. Falkenhainer. Dynamic constraint satisfaction problems. In *Proceedings of the 8th National Conference on Artificial Intelligence*, pages 25–32, Boston, MA, Aug. 1990. AAAI Press.

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

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

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

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

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

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

[18] 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.

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

[20] Y. Zhang and R. 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.