

Conditional and Composite Temporal CSPs

Malek Mouhoub and Amrudee Sukpan

University of Regina, Canada

Abstract

In order to deal with dynamic CSPs where the information regarding any possible change is known *a priori* and can thus be enumerated beforehand, *conditional constraints* and *composite variables* have been studied in the past decade. Indeed, these two concepts allow the addition of variables and their related constraints in a dynamic manner during the resolution process. More precisely, a *conditional constraint* restricts the participation of a variable in a feasible scenario while a composite variable allows us to express a disjunction of variables where only one will be added to the problem to solve. In order to deal with a wide variety of real life applications under temporal constraints, we present in this paper a unique temporal CSP framework including numeric and symbolic temporal information, conditional constraints and composite variables. We call this model, a Conditional and Composite Temporal CSP (or CCTCSP). To solve the CCTCSP we propose several search techniques based on constraint propagation. In order to assess the efficiency in time of these search techniques, experimental tests have been conducted on randomly generated CCTCSPs. The results demonstrate the superiority of a variant of the Maintaining Arc Consistency (MAC) (that we call MAX+) over the other constraint propagation strategies (FC and its variant) for both consistent and inconsistent problems.

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 (Baptiste and Pape 1995; Ghallab and Laruelle 1994; Laborie and Ghallab 1995; Laborie 2003), natural language processing (Song and Cohen 1991; Hwang and Shubert 1994), molecular biology (Golombic and Shamir 1993) and temporal database (Dean 1989; Theodoulidis, Loucopoulos, and Wangler 1991). 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 (Allen 1983; Vilain and Kautz 1986; Dechter, Meiri, and Pearl 1991; van Beek 1992). 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 (Dechter 2003; Montanari 1974; Mackworth 1977; Haralick and Elliott 1980). 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 express the relative position between the temporal objects these variables represent. The relative position between temporal objects 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 concerned with reasoning on the metric or the symbolic aspects of time (respectively through metric or qualitative networks), little work such as (Kautz and Ladkin 1991; Meiri 1996; Ghallab and Laruelle 1994; Thornton et al. 2004) has been developed to manage both types of information. In (Mouhoub 2004), we have developed a temporal model, TemPro, based on Allen's interval algebra (Allen 1983) 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 CSP¹ called Temporal CSP (or TCSP²) where variables are temporal events defined on domains of numeric intervals and binary constraints between variables correspond to disjunctions of Allen primitives (Allen 1983). 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

¹In a binary CSP, constraints can only be unary or binary relations.

²Note that the acronym TCSP was used in (Dechter, Meiri, and Pearl 1991). 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.

variable domains and the path consistency on symbolic relations. A backtrack search algorithm using constraint propagation is then performed in the second stage to check the consistency of the TCSP by looking for a feasible scenario. A feasible scenario (or solution to the TCSP) is a complete assignment of intervals to all the events such that all the constraints are satisfied. In order to deal with a large variety of dynamic real world temporal applications and where the information regarding any possible change can be enumerated beforehand, we present in this paper an extension of the modeling framework TemPro including the following.

(1) Variable status: each variable has either active or inactive status. Only active variables require an assignment from their domain of values. Inactive 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), through an activity constraint or a composite variable.

(2) Composite temporal variables: composite temporal variables are variables whose values are temporal events. During the resolution process, an active composite variable will be assigned (replaced with) one event from its domain. This latter event will then be activated. For instance, if we have a list of events Evt_1, \dots, Evt_n where only one Evt_i should be present in the problem to solve, then we can create a composite variable X defined on the domain $D_X = \{Evt_1, \dots, Evt_n\}$.

(3) Activity (or conditional) constraints: an activity constraint has the following form: $X_1 \wedge \dots \wedge X_p \xrightarrow{\text{condition}} Y$ where X_1, \dots, X_p and Y are temporal variables (composite or events). This activity constraint will activate Y if X_1, \dots, X_p are active and *condition* holds on these variables. *condition* corresponds here to the assignment of particular values to the variables X_1, \dots, X_p . We call a Composite TCSP (CTCSP) a TCSP including composite temporal variables. A CTCSP represents a finite set of possible TCSPs where each TCSP corresponds to a complete assignment of values (temporal events) to composite variables. Solving a CTCSP consists of finding a feasible scenario for one of its possible TCSPs. Solving a TCSP 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 (number of values in the domain). The possible number of TCSPs the CTCSP involves is d^M where M is the number of composite variables and d their domain size. Thus, solving a CTCSP requires a backtrack search algorithm of complexity $O(d^M \times D^{N+M})$ in the worst case. We call Conditional CTCSP (CCTCSP) a CTCSP augmented by activity constraints. Solving a CCTCSP can be seen like solving a CTCSP dynamically i.e when some of the variables (events or composites) and their corresponding constraints are added or removed dynamically during the resolution of the CSP. To overcome this difficulty in practice, we propose in this paper a search method based on constraint propagation for solving efficiently CCTCSPs. Constraint propagation includes arc consistency (Mackworth 1977) as well as forward check and MAC strategies (Haralick and Elliott 1980). In order to assess the efficiency in time of the solving method we propose, experimental comparative tests have been conducted on ran-

dom CCTCSPs generated using the model RB (Xu and Li 2000) as this latter has the ability to generate asymptotically hard instances. The test results demonstrate the superiority of a variant of the MAC technique (that we call MAX+) over the other constraint propagation strategies (FC and its variant FC+) for both consistent and inconsistent problems. The rest of the paper is structured as follows. In the next section we introduce the CCTCSP framework through an example. Section 3 is dedicated to the constraint propagation techniques for solving CCTCSPs. Section 4 describes the experimental comparative tests we have conducted on random CCTCSPs. Finally, concluding remarks are covered in Section 5.

2. Conditional and Composite Temporal Constraint Satisfaction Problems (CCTCSPs)

Managing conditional, composite and dynamic CSPs has already been reported in the literature (Mittal and Falkenhainer 1990; Sabin and Freuder 1996; Sabin, Freuder, and Wallace 2003; Gelle and Faltings 2003; Dechter and Dechter 1988; Jónsson and Frank 2000; J. Frank 2003; Tsamardinou, Vidal, and Pollack 2003). (Mittal and Falkenhainer 1990) 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 (Sabin and Freuder 1996), 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 (Jónsson and Frank 2000) 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 (J. Frank 2003). 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 (Jónsson and Frank 2000) 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 (Jónsson and Frank 2000) is also more general than the one in (Dechter and Dechter 1988) since

in this latter work variable domains are predetermined. Finally, in (Tsamardinos, Vidal, and Pollack 2003), Tsamardinos et al propose the Conditional Temporal Problem (CTP) formalism (and its particular case CSTP involving STP-like constraints) for Conditional Planning under temporal constraints. This model extends the well known qualitative temporal networks, STP (Dechter, Meiri, and Pearl 1991) and DTP (Tsamardinos and Pollack 2002) by adding instantaneous events (called observation nodes) representing conditional constraints and attaching labels to all nodes to indicate the different situations. We adopt both the CCSP (Mittal and Falkenhainer 1990) and the composite CSP (Sabin and Freuder 1996) paradigms and extend the modeling framework TemPro (Mouhoub 2004) 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 and those we mentioned in introduction on hybrid temporal constraints (Kautz and Ladkin 1991; Meiri 1996; Ghallab and Laruelle 1994; Thornton et al. 2004), ours has the following specifics.

1. Our work focuses on temporal constraints while the previous literature is on general constraints, if we exclude the work in (Tsamardinos, Vidal, and Pollack 2003) and (J. Frank 2003). Both these two latter formalisms, however, handle only quantitative time information while ours combines both quantitative and qualitative temporal constraints.
2. Our model is application independent and is not restricted to a particular area such as planning or scheduling. It can thus 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 (Allen 1983) which offers more expressiveness. Although this will lead to NP-hard problems, the solving techniques that we will present in the next two Sections overcome this difficulty, in practice, as demonstrated by the test results 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 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 and not, for example, like in (Kautz and Ladkin 1991) where numeric and symbolic temporal constraints are managed in two separate systems.
5. Our model is more expressive than IxTeT (Ghallab and Laruelle 1994) since this latter is restricted to the Point Algebra (PA) while ours is based on the full Allen Interval Algebra (IA). Also, when using a discrete representa-

tion of time, temporal problems represented by our model include those represented by the well known formalisms proposed in (Meiri 1996) and (Thornton et al. 2004) as we will show later in example 2.

6. As we will see in Section 3, unlike the work in (Sabin, Freuder, and Wallace 2003) where constraint propagation is performed during search (using forward checking and MAC strategies) through activity constraints, in our model the propagation is performed through compatibility constraints which will reduce the size of the search space before and during the resolution process.
7. Unlike the model in (Sabin and Freuder 1996) where a variable can unfold into sub CSPs in a recursive manner, in our framework a composite variable is defined on a set of events and can thus be replaced by only one event from this set.

Definition

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

$E = \{e_1, \dots, e_n\}$ is a finite set of temporal variables that we call events. In the same way as in (Allen 1983), we define an event e as the proposition that occurs over the smallest possible time interval I for it to occur:

$$OCCURS(e, I) \wedge I' \subset I \Rightarrow \neg OCCURS(e, I')$$

where I' is a subinterval of I . For the sake of notation simplicity, an event is used in this paper to denote its temporal qualification (time interval during which the event occurs).

$D_E = \{D_{e_1}, \dots, D_{e_n}\}$ is the set of domains of the events. Each domain D_{e_i} is the finite and discrete set of numeric and continuous time intervals the event e_i can take. D_{e_i} is expressed by the four-fold $[EarliestStart_{e_i}, LatestEnd_{e_i}, Duration_{e_i}, Step_{e_i}]$ where $EarliestStart_{e_i}$ and $LatestEnd_{e_i}$ are respectively the earliest start time and the latest end time of e_i , $Duration_{e_i}$ is its duration and $Step_{e_i}$ defines the distance (number of time units) between the starting time of two adjacent intervals within D_{e_i} . The discretization step $Step_{e_i}$ allows us to handle temporal information with different granularities as in (Bettini, Wang, and Jajodia 2002). $X = \{x_1, \dots, x_m\}$ is the finite set of composite variables.

$D_X = \{D_{x_1}, \dots, D_{x_m}\}$ 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. $IV \subseteq E \cup X$.

$C = \{C_1, \dots, C_p\}$ 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 (Allen 1983).

Act is the set of *activity constraints*. Each activity constraint has the following form: $X_1 \wedge \dots \wedge X_p \xrightarrow{condition} Y$

where X_1, \dots, X_p and Y are temporal variables (composite or events). This activity constraint will activate Y if X_1, \dots, X_p are active and *condition* holds on these variables. *condition* corresponds here to the assignment of particular values to the variables X_1, \dots, X_p .

Let us illustrate the CCTCSP through the following example. *John, Mike and Lisa are going to see a movie on Friday. John will pick Lisa up and Mike will meet them at the theatre. If John arrives at Lisa's before 7:30, then they will stop at a convenient store to get some snacks and pops. It will take them 35 minutes to reach the theater if they stop at the store and 15 minutes otherwise. There are three different shows playing: movie₁, movie₂ and movie₃. 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; movie₁, movie₂ and movie₃ start at 7:30, 7:45 and 7:35 and finish at 9:00, 9:10 and 9:20 respectively.*

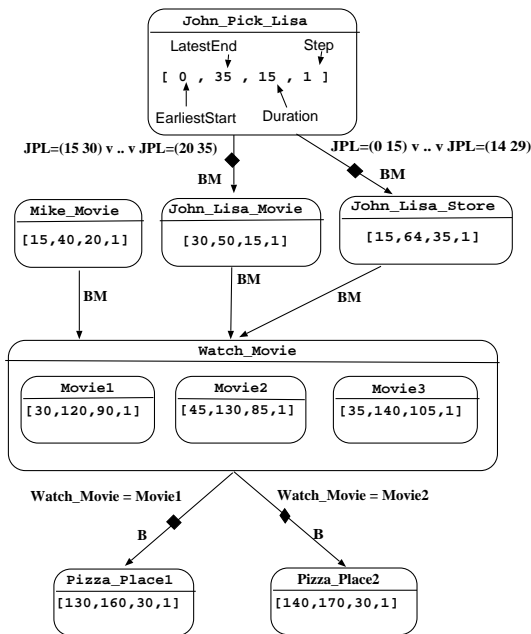


Figure 1: CCTCSP of example 1.

The goal here is to check if this story is consistent (has a feasible scenario). The story can be represented by the CCTCSP in figure 1. Each event domain is represented by the fourfold $[EarliestStart, LatestEnd, Duration, Step]$. In the case of *John_Pick_Lisa*, 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. This domain corresponds to the following set of intervals: $\{[0, 15], [1, 16], \dots, [20, 35]\}$. 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 (Allen 1983). For example, the relation *BM* between *John_Pick_Lisa* and *John_Lisa_Store* denotes the disjunction *Before* \vee *Meets*. The *JPL* (*John_Pick_Lisa*) related activity constraint shown in Figure 1 expresses the fact that if *JPL* ending time is before or equal 30 (7:30) then the event *John_Lisa_Store* is activated, otherwise (if *JPL* starting time is after 20) *John_Lisa_Movie* is activated. This activity constraint is a representation of the sentence “If John arrives at Lisa’s before 7:30, then they will stop at a convenient store to get some snacks and pops”.

3. Constraint Propagation for Solving CCTCSPs

Different methods for solving conditional CSPs have been reported in the literature (Gelle and Faltings 2003; Mittal and Falkenhainer 1990; Sabin, Freuder, and Wallace 2003; Gelle 1998). In (Gelle and Faltings 2003), 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 acyclic 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 (Mittal and Falkenhainer 1990; Sabin, Freuder, and Wallace 2003) 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 (Sabin, Freuder, and Wallace 2003). 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 (Gelle 1998; Sabin, Freuder, and Wallace 2003), 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 meth-

ods, 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 perform the propagation through activity constraints (while it is done through compatibility constraints in our case). The other problem of the above methods (since the goal is to look for all solutions) is the redundant work done when checking at each time the consistency of the same set of variables (subset of a given generated CSP). Our aim in this paper is to check for the consistency of the CCTCSP (finding a possible solution to one of its possible TCSPs) and to return one solution in case this latter is consistent. 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 (Mittal and Falkenhainer 1990; Sabin, Freuder, and Wallace 2003), 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 (FC) principle (Haralick and Elliott 1980), pick an active variable v , assign a value to it and perform arc consistency between this variable and each of the unassigned 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 shown in the four cases below. We will assume in the following that evt_1 and evt_2 are two events while x_1 and x_2 are two composite variables. **(1) The constraint is (evt_1, evt_2) .** Arc consistency (Mackworth 1977) is applied here i.e. each interval a of evt_1 should have a support in the domain of evt_2 .

(2) The constraint is (evt_1, x_1) . Each interval a , from the domain of evt_1 , should have a support in at least one domain of the variables within x_1 . **(3) The constraint is (x_1, evt_1) .** Each interval a , from the domain of every event evt within x_1 , should have a support in the domain of evt_1 . **(4) The constraint is (x_1, x_2) .** Apply case (b) between each interval evt within x_1 , and x_2 . Note that, in addition to the Forward Check (FC) principle we described earlier in phase 2 of our constraint propagation method, we have explored other propagation strategies. More precisely we are considering the following four techniques. **FC.** As we mentioned earlier, this strategy consists of checking the arc consistency, during the search, between the current variable (the variable that we are assigning a value) and each of the future active variables (active variables that are not assigned yet) sharing a constraint with the current variable. **Maintaining Arc Consistency (MAC).** This strategy maintains a full arc consistency on the current and future active variables. **FC+.** Same as FC except that the applicability of arc consistency is extended to inactive variables as well. This means that the arc consistency is performed between the current variable and each of the future active and non active variables sharing a constraint with the current one. **MAC+.** Same as MAC except that the applicability of the arc consistency is extended to inactive variables as well. This means that the full arc consistency is enforced on the current and the future variables (active and inactive).

4. Experimentation

The goal of the tests reported in this section is to evaluate and compare the performance for solving randomly generated CCTCSPs using the methods we presented in the previous two sections. More precisely we will compare the following four constraint propagation strategies we described in Section 3 (FC, FC+, MAC and MAC+). All the experiments are performed on a PC Pentium 4 computer running Linux. All the procedures are coded in C/C++. CCTCSPs are build from TCSPs randomly generated using the model RB proposed in (Xu and Li 2000). This model is a revision to the standard Model B (Gent et al. 1998; Smith and Dyer 1996), has exact phase transition and the ability to generate asymptotically hard instances. As demonstrated in (Xu and Li 2000), when the number of variables approaches infinity the phase transition occurs when the constraint tightness $p = 1 - e^{-\frac{\alpha}{r}}$. Thus the phase transition is an asymptotic phenomenon since, only for infinite number of variables, we can have sharp phase transitions. In addition, the number of variables and constraints of the possible CSPs, each CCTCSP contains, is slightly different from the one of the CCTCSP they are generated from. In the following we will present the results of tests performed by each of the four strategies described in Section 3 on several CCTCSP sets. Each set is generated by varying one of the following parameters: p, N, D, I, a, α and r where N is the number of composite variables, D their domain size (number of events within each composite variable), I

the percentage of variables that are initially active and a the density of activity constraints. For each CSP within the CCTCSP, the number of constraints is $rn \ln n$ and the domain size of the events is n^α (where n is the number of variables)³. For each test, each of the four methods is executed on 100 instances and the average running time in seconds is taken.

4.1 Easy versus Hard Problems: Varying p .

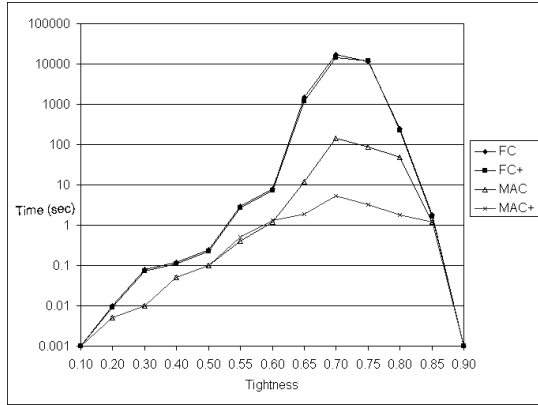


Figure 2: Comparative tests when varying the tightness p .

Here the parameters are set as follows. $n = 140$, $N = 10$, $D = 5$, $\alpha = 0.8$, $I = 0.8$, $a = 0.2$ and $r = 0.6$. p varies from 0.2 to 0.9 (in order to consider under and over constrained problems as well as those near the phase transition). As mentioned before, according to the RB model, the phase transition is computed as follows: $p = 1 - e^{-\frac{\alpha}{r}} = 1 - e^{-\frac{0.8}{0.6}} = 0.73$. In practice it is around 0.7 as we can see from the test results shown in figure 2. In the case of over constrained problems (tightness greater than 0.8), all the four methods have comparable running time. This is because the inconsistency is detected in the first stage of the resolution method we provided in Section 3 to the initial problem). Indeed, the first stage is the same for all the four methods. All the methods have also similar running times in the case of under constrained problems. Indeed, in this particular case the extra effort done by MAC and MAC+ does not remove much of the inconsistent values and thus does not improve the overall running time to find a solution. However, when we move toward the phase transition the extra work performed by MAC and especially MAC+ starts to pay off. At the phase transition MAC+ is almost 10,000 times faster than FC and FC+; and 100 times faster than MAC. Note that the superiority of MAC over FC for hard CSPs has already been noticed and reported in (Sabin and Freuder 1994). Also, considering inactive variables during the propagation through MAC+ does help a lot on the complexity peaks. Indeed, at the phase transition more search

³According to the RB model, in order to be able to compute accurately the phase transition, the number of constraints should be equal to $rn \ln n$ (Xu and Li 2000).

space is explored through the backtrack and many of the inactive variables are activated. Reducing the domains of these latter is thus very relevant in this particular situation.

4.2 More versus less Dynamic Problems: Varying a , I , N and D

The top left chart of figure 3 reports the results of comparative tests conducted when the parameter a varies from 0.02 to 0.3. The other parameters are fixed as follows: $n = 50$, $N = 10$, $D = 5$, $p = 0.5$, $\alpha = 0.8$, $I = 0.8$, $r = 0.6$. The top right chart of figure 3 reports the results of comparative tests conducted when the parameter I varies from 0.2 to 0.9. The other parameters are fixed as follows: $n = 50$, $N = 10$, $D = 5$, $p = 0.5$, $\alpha = 0.8$, $a = 0.2$, $r = 0.6$. The lower the value of I is, the more “dynamic” (and difficult to solve) is the problem. In both figures 11 and 12 the winners are again MAC and MAC+. While MAC+ does more efforts than MAC when I decreases (since the difference between the two strategies is that MAC+ does the propagation to non active variables as well as active variables) it is the only method solving all the problem instances when the tightness is equal to 0.6. Indeed we have conducted other tests (that we did not report here) where $p = 0.6$. The results indicate that FC and FC+ fail to solve all the problems when $I < 0.9$. The bottom left chart of figure 3 reports the results of the tests when varying N . Here the number of composite variables N varies from 5 to 20. The other parameters are fixed as follows: $n = 50$, $D = 5$, $p = 0.5$, $\alpha = 0.8$, $a = 0.2$, $r = 0.6$ and $I = 0.8$. The bottom right chart of figure 3 reports the results of the tests when varying D from 2 to 10. The other parameters are fixed as follows: $n = 50$, $N = 10$, $p = 0.5$, $\alpha = 0.8$, $a = 0.2$, $r = 0.6$ and $I = 0.8$. When N or D is increasing, the generated problems become more dynamic. Here again, MAC and MAC+ outperform the other two methods.

4.3 Small versus Large Size Problems: Varying α and r

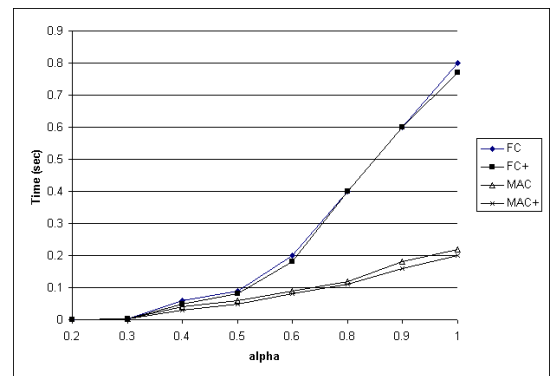


Figure 4: Comparative tests when varying α .

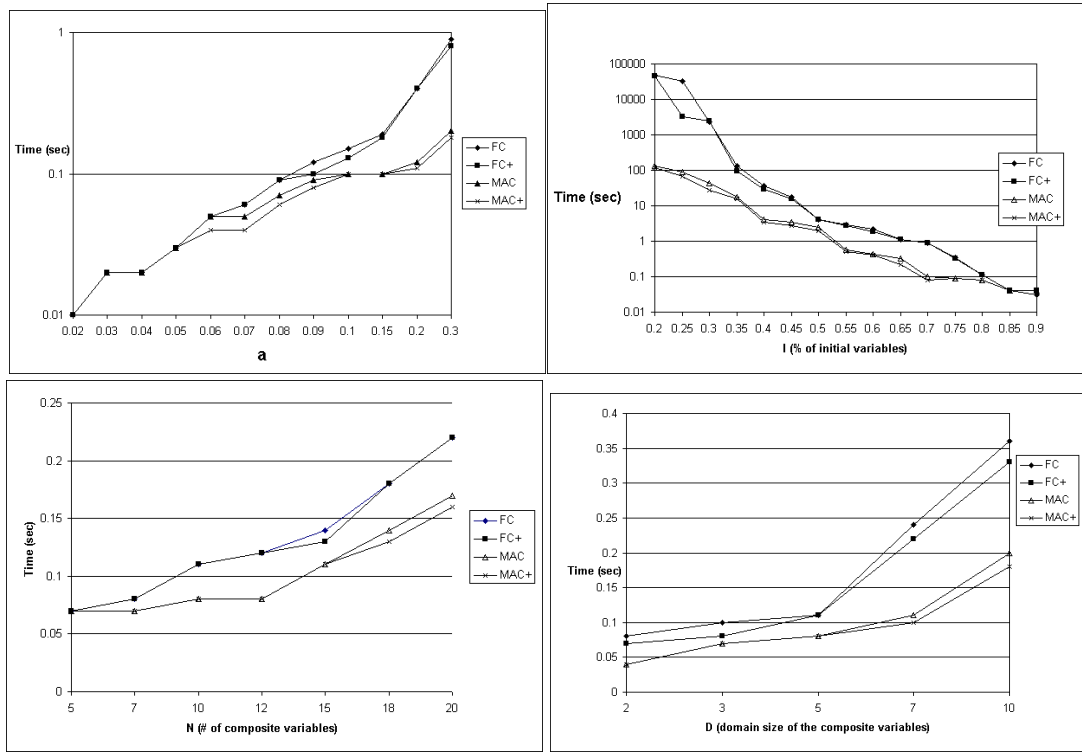


Figure 3: Comparative tests when varying a (% of possible activity constraints), I , the number of composites or their domain size.

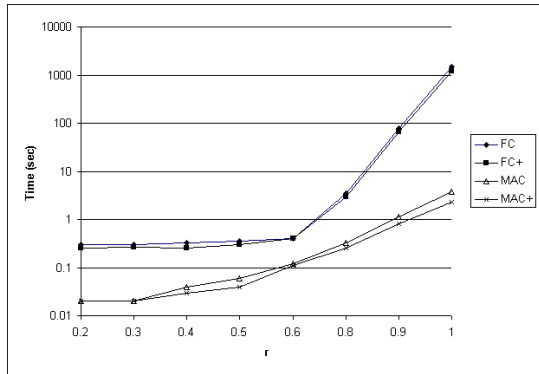


Figure 5: Comparative tests on random CCTCSPs when varying r .

Figure 4 reports the results of the tests when varying α from 0.2 to 1. The other parameters are fixed as follows: $n = 50$, $N = 10$, $D = 5$, $p = 0.5$, $a = 0.2$, $r = 0.6$ and $I = 0.8$. Figure 5 reports the results of the tests when varying r from 0.2 to 1. The other parameters are fixed as follows: $n = 50$, $N = 10$, $\alpha = 0.8$, $D = 5$, $p = 0.5$, $a = 0.2$ and $I = 0.8$. According to the RB model, the domain size d of the events is equal to n^α , and the number of generated constraints is equal to $rn \ln n$. Thus, when α increases the domain size of the events becomes very large, and when r increases, the

number of constraints increases and the related generated problem becomes more constrained. In both figures 4 and 5 MAC and MAC+ outperform FC and FC+.

5. 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. When considering only composite TCSPs (CTCSPs) the solving algorithm requires $O(D^{N+M}d^M)$ time cost where N , D , M and d are respectively the number of events and their domain size, and the number of composite variables and their domain size. Adding activity constraints will make the problem even harder. In order to overcome this difficulty in practice, we have proposed a solving method based on constraint propagation. Constraint propagation prevents later failure earlier which improves, in practice, the performance in time of the backtrack search especially when the propagation is performed through the MAC principle.

References

- Allen, J. 1983. Maintaining knowledge about temporal intervals. *CACM* 26(11):832–843.
- Baptiste, P., and Pape, C. L. 1995. Disjunctive constraints for manufacturing scheduling : Principles and extensions. In *Third International Conference on Computer Integrated Manufacturing*.
- Bettini, C.; Wang, X.; and Jajodia, S. 2002. Solving multi-granularity temporal constraint networks. *Artificial Intelligence* 140(1-2):107–152.
- Dean, T. 1989. Using temporal hierarchies to efficiently maintain large temporal databases. *JACM* 686–709.
- Dechter, R., and Dechter, A. 1988. Belief maintenance in dynamic constraint networks. In *7th National Conference on Artificial Intelligence*, 37–42.
- Dechter, R.; Meiri, I.; and Pearl, J. 1991. Temporal Constraint Networks. *Artificial Intelligence* 49:61–95.
- Dechter, R. 2003. *Constraint Processing*. Morgan Kaufmann.
- Gelle, E., and Faltings, B. 2003. Solving mixed and conditional constraint satisfaction problems. *Constraints* 8:107–141.
- Gelle, E. 1998. On the generation of locally consistent solution spaces in mixed dynamic constraint problems. *Ph.D.thesis* 1826:101–140.
- Gent, I.; MacIntyre, E.; Prosser, P.; Smith, B.; and Walsh, T. 1998. Random constraint satisfaction: Flaws and structure.
- Ghallab, M., and Laruelle, H. 1994. Representation and Control in IxTeT, a Temporal Planner. In *AIPS 1994*, 61–67.
- Golumbic, C., and Shamir, R. 1993. Complexity and algorithms for reasoning about time: a graphic-theoretic approach. *Journal of the Association for Computing Machinery* 40(5):1108–1133.
- Haralick, R., and Elliott, G. 1980. Increasing tree search efficiency for Constraint Satisfaction Problems. *Artificial Intelligence* 14:263–313.
- Hwang, C., and Shubert, L. 1994. Interpreting tense, aspect, and time adverbials: a compositional, unified approach. In *Proceedings of the first International Conference on Temporal Logic, LNAI, vol 827*, 237–264.
- J. Frank, A. K. J. 2003. Constraint-based attribute and interval planning. *Constraints* 8(4):339–364.
- Jónsson, A. K., and Frank, J. 2000. A framework for dynamic constraint reasoning using procedural constraints. In *ECAI 2000*, 93–97.
- Kautz, H., and Ladkin, P. 1991. Integrating metric and qualitative temporal reasoning. In *AAAI'91*, 241–246.
- Laborie, P., and Ghallab, M. 1995. Planning with Sharable Resource Constraints. In *IJCAI-95*, 1643–1649.
- Laborie, P. 2003. Resource Temporal Networks: Definition and Complexity. In *Eighteenth International Joint Conference on Artificial Intelligence (IJCAI'03)*, 948–953.
- Mackworth, A. K. 1977. Consistency in networks of relations. *Artificial Intelligence* 8:99–118.
- Meiri, I. 1996. Combining qualitative and quantitative constraints in temporal reasoning. *Artificial Intelligence* 87:343–385.
- Mittal, S., and Falkenhainer, B. 1990. Dynamic constraint satisfaction problems. In *Proceedings of the 8th National Conference on Artificial Intelligence*, 25–32. Boston, MA: AAAI Press.
- Montanari, U. 1974. Fundamental properties and applications to picture processing. *Information Sciences* 7:95–132.
- Mouhoub, M. 2004. Reasoning with numeric and symbolic time information. *Artificial Intelligence Review* 21:25–56.
- Sabin, D., and Freuder, E. C. 1994. Contradicting conventional wisdom in constraint satisfaction. In *Proceedings of the Eleventh European Conference on Artificial Intelligence*, 125–129. Amsterdam, The Netherlands: John Wiley and Sons.
- Sabin, D., and Freuder, E. C. 1996. Configuration as composite constraint satisfaction. In Luger, G. F., ed., *Proceedings of the (1st) Artificial Intelligence and Manufacturing Research Planning Workshop*, 153–161. AAAI Press.
- Sabin, D.; Freuder, E. C.; and Wallace, R. J. 2003. Greater efficiency for conditional constraint satisfaction. *Proc., Ninth International Conference on Principles and Practice of, Constraint Programming - CP 2003* 2833:649–663.
- Smith, B., and Dyer, M. 1996. Locating the phase transition in binary constraint satisfaction problems. *Artificial Intelligence* 81:155–181.
- Song, F., and Cohen, R. 1991. Tense interpretation in the context of narrative. In *AAAI'91*, 131–136.
- Theodoulidis, C. I.; Loucopoulos, P.; and Wangler, B. 1991. A conceptual modelling formalism for temporal database applications. *Information Systems* 16(4):401–416.
- Thornton, J.; Beaumont, M.; Sattar, A.; and Maher, M. 2004. A Local Search Approach to Modelling and Solving Interval Algebra Problems. *Journal of Logic and Computation* 14(1):93–112.
- Tsamardinos, I., and Pollack, M. E. 2002. Efficient Solution Techniques for Disjunctive Temporal Problems. *URL* <http://citeseer.ist.psu.edu/tsamardinos02efficient.html>.
- Tsamardinos, I.; Vidal, T.; and Pollack, M. E. 2003. CTP: A New Constraint-Based Formalism for Conditional Temporal Planning. *Constraints* 8(4):365–388.
- van Beek, P. 1992. Reasoning about qualitative temporal information. *Artificial Intelligence* 58:297–326.
- Vilain, M., and Kautz, H. 1986. Constraint propagation algorithms for temporal reasoning. In *Proceedings of the Fifth National Conference on Artificial Intelligence (AAAI'86)*, 377–382. Philadelphia, PA: The MIT Press.
- Xu, K., and Li, W. 2000. Exact Phase Transitions in Random Constraint Satisfaction Problems. *Journal of Artificial Intelligence Research* 12:93–103.