Constraint-Based Strategies for the Disjunctive Temporal Problem: Some New Results

Angelo Oddi*
IP-CNR, Italian National Research Council
Viale Marx 15, I-00137 Rome, Italy
odd@ip.rm.cnr.it

Abstract

The Disjunctive Temporal Problem (DTP) involves the satisfaction of a set of constraints represented by disjunctive formulas of the form $x_1 - y_1 \leq r_1 \lor x_2 - y_2 \leq r_2 \lor \ldots \lor x_k - y_k \leq r_k$. DTP is a general temporal reasoning problem which includes the well-known Temporal Constraint Satisfaction Problem (TCSP) introduced by Dechter, Meiri and Pearl. This paper describes a basic constraint satisfaction algorithm where several aspects of the current literature are integrated, in particular the so-called forward checking. Hence, two new extended solving strategies are proposed and experimentally evaluated. The new proposed strategies are very competitive with the best results available in the current literature. In addition, the analysis of the empirical results suggests future research directions concerning in particular the use of arc-consistency filtering strategies.

Introduction

In many recent Artificial Intelligence applications the need of a more expressive temporal reasoning framework has clearly emerged. For example, in continual planning applications (DesJardin et al. 1999) a relevant capability is the continuous management of temporal plans (Pollack and Horty 1999; Tsamardinos, Pollack, and Horty 2000) such that the representation of temporal disjunctions leverages the system’s capabilities, by avoiding early commitments on action orderings. The Temporal Constraint Satisfaction Problem (TCSP) (Dechter, Meiri, and Pearl 1991) is a well-known paradigm to express temporal disjunctions, as it allows to represent constraints of the form $x - y \leq r_1 \lor x - y \leq r_2 \lor \ldots \lor x - y \leq r_k$. A further generalization of the TCSP was recently proposed in (Stergiou and Koubarakis 1998; 2000) where a problem has constraints of the form $x_1 - y_1 \leq r_1 \lor x_2 - y_2 \leq r_2 \lor \ldots \lor x_k - y_k \leq r_k$. In (Armando, Castellini, and Giunchiglia 1999) this last problem is referred to as Disjunctive Temporal Problem (DTP) and we will use this name in the paper.

DTPs have been studied in several previous works. In (Stergiou and Koubarakis 1998; 2000) several constraint-based (CSP) algorithms (in the line of (Prosser 1993)) are defined and experimentally compared. One of them, based on forward checking (Haralick and Elliott 1980), is shown to be the best. In addition, (Stergiou and Koubarakis 1998; 2000) describe interesting applications of the DTP model ranging from scheduling and planning to temporal database with indefinite information. In (Armando, Castellini, and Giunchiglia 1999) DTP is modeled as a propositional satisfiability (SAT (Cook and Mitchell 1998)) problem and solved with a state-of-the-art SAT-solver plus some additional processing. Experiments show an improvement of up to two orders of magnitude with respect to the results obtained in (Stergiou and Koubarakis 1998). Finally, in (Oddi and Cesta 2000) the constraint-based algorithm proposed in (Stergiou and Koubarakis 1998; 2000) is improved exploiting the quantitative temporal information in the solution “distance graph”. In addition to the studies performed in previous works, the paper (Oddi and Cesta 2000) basically investigates how the effects of quantitative temporal information can be exploited to improve global performance in solving DTPs. Using such knowledge the authors introduce an incremental forward checking algorithm which has comparable performance (assessed in terms of number of forward checks) with the best SAT-based version proposed in (Armando, Castellini, and Giunchiglia 1999).

This paper follows the same CSP-based approach and again focuses on quantitative temporal information for reasoning on DTPs. In particular, we propose a new heuristic strategy for variable ordering used in our CSP framework and an arc-consistency filtering algorithm. The rationale behind the new results is quite general and can be exploited in other solvers that rely on “quantitative temporal reasoning”. The paper is structured as follows. The following section introduces the basic concepts used in the paper. The subsequent two sections introduce: (i) a basic CSP algorithm which integrates results from previous works, and (ii) two new additional algorithms for solving DTP instances. A following experimental section describes the empirical evaluation of the proposed approaches. Finally, some conclusions end the paper.

Preliminaries

The Disjunctive Temporal Problem (DTP) involves a finite set of temporal variables $x_1, y_1, x_2, y_2, \ldots, x_n, y_n$ ranging over the reals and a finite set of constraints $C^*$ =
c$ \in C$ at least one disjunct $x_i - y_i \leq r_i$ is satisfied. One way to check for consistency of a DTP consists of choosing one disjunct for each constraint $c_i$ and see if the conjunction of the chosen disjuncts is consistent. It is worth observing that this is equivalent to extracting a “particular” STP (the Simple Temporal Problem defined in (Dechter, Meiri, and Pearl 1991)) from the DTP and checking consistency of such a STP. If the STP is not consistent another one is selected, and so on. Both previous approaches to DTP (Stergiou and Koubarakis 1998; 2000; Armando, Castellini, and Giunchiglia 1999; Oddi and Cesta 2000) do this basic search step.

**Previous Work.** All (Stergiou and Koubarakis 1998; 2000; Armando, Castellini, and Giunchiglia 1999; Oddi and Cesta 2000) share a “two layered” algorithmic structure. An upper layer of reasoning is responsible for guiding the search that extracts the set of disjuncts, a lower layer represents the quantitative information of the temporal reasoning problem. In (Stergiou and Koubarakis 1998) a general CSP formulation is used at the upper level while the quantitative information is managed by using the incremental directional path consistency (IDPC) algorithm of (Chleq 1995). In (Armando, Castellini, and Giunchiglia 1999) at the upper level the DTP is encoded as a SAT problem, a SAT-solver extracts an STP to be checked, a simplified version of the Simplex algorithm is used at the lower level to check for its consistency. Stergiou and Kubarakis define different backtracking algorithms for managing the upper-level and experimentally verify that the version using forward checking is the best. Forward checking is used after each choice to test which of the possible next choices are compatible with current partial STP. In the rest of the paper their best algorithm is called SK. Armando, Castellini and Giunchiglia focus their attention on the SAT encoding, each disjunct is a propositional formula, and they use a state of the art SAT-solver enriched with a form of forward checking biased by the temporal information. Their basic version is called TSAT and is shown to improve up to one order of magnitude with regard to SK. Then they add a further preprocessing step called IS that basically produces a more accurate SAT encoding because it codifies mutual exclusion conditions among propositions that exist in the temporal information, but were lost by the first standard encoding.

**DTP Consistency Checking as a Meta-CSP.** Before introducing our algorithm we underscore the possibility of representing the consistency checking problem as a meta-CSP problem, where each DTP constraint $c \in C$ represents a (meta) variable and the set of disjuncts represents variable’s domain values $D_c = \{\delta_1, \delta_2, \ldots, \delta_k\}$. A meta-CSP problem is consistent if exists at least an element $S$ (solution) of the set $D_1 \times D_2 \times \ldots \times D_m$ such that the corresponding set of disjuncts $S = \{\delta_1, \delta_2, \ldots, \delta_m\}$ $\delta_i \in D_i$ is temporally consistent.

Each value $\delta_i \in D_i$ represents an inequality of the form $x_i - y_i \leq r_i$ and a solution $S$ can be represented as a labeled graph $G_d(V_S, E_S)$ called “distance graph” (Dechter, Meiri, and Pearl 1991). The set of nodes $V_S$ coincides with the set of DTP variables $x_1, y_1, x_2, y_2, \ldots, x_n, y_n$ and each disjunct $x_i - y_i \leq r_i$ is represented by a direct edge $(y_i, x_i)$ from $y_i$ to $x_i$ labeled with $r_i$. A path from a node $x_i$ to $y_j$ on the graph is a set of contiguous edges $(x_i, y_i), (y_i, y_1), (y_1, y_2), \ldots, (x_i, y_j)$ and the length of the path is the sum of the edges’ labels. The set of disjuncts $S$ corresponds to an STP. $S$ is a solution to the meta-CSP problem if $G_d$ does not contain closed path with negative length (negative cycles) (Dechter, Meiri, and Pearl 1991). From the graph $G_d$ a numerical solution of the problem can be extracted as follows. Let $d_{x_i,y_i}$ be the shortest path distance on $G_d$ from the node $x_i$ to $y_i$ without loss of generality we can assume a variable $x_i$ as reference point, for example $x_1$, in this way the tuple $(d_{x_1,x_1}, d_{x_1,x_2}, \ldots, d_{x_1,x_n})$ is a solution of the original DTP problem. In fact, the previous values represent the shortest distance from the reference node $x_1$ to all the other ones (in particular $d_{x_1,x_1} = 0$). For each edge $x_i - y_i \leq r_i$ in $G_d$ as it is well known values $(d_{x_1,x_1}, d_{x_1,x_2}, \ldots, d_{x_1,x_n})$ must hold the Bellman’s inequalities: $d_{x_1,x_1} \leq d_{x_1,y_1} + r_1$, that is $d_{x_1,x_1} - d_{x_1,y_1} \leq r_1$. Hence $(d_{x_1,x_1}, d_{x_1,x_2}, \ldots, d_{x_1,x_n})$ is a solution for the DTP.

This view of the consistency checking problem is used to define our CSP approach and in particular is useful to understand our incremental forward checking method.

**A CSP Algorithm for DTP**

In this work we mainly follow the constraint-based approach of Stergiou and Kubarakis (Stergiou and Koubarakis 1998; 2000) for solving DTP instances. Figure 1 shows a CSP procedure which starts from an empty solution $S$ and basically executes three steps: (a) the current partial solution is checked for consistency (Step 1) by the function CheckConsistency. This function filters also the search space from inconsistent states. If the partial solution is a complete solution (Step 2) the algorithm exits. If the solution is still incomplete the following two steps are executed: (b) a (meta) variable (a constraint $c_i$) is selected at Step 5 by a variable ordering heuristic; (c) a disjunct $x_i - y_i \leq r_i$ is chosen (Step 6) from the domain variable $D_i$ and added to $S$ (represented at the lower level as a $G_d$ graph). Hence the solver is recursively called on the partial updated solution $S \cup \{\delta\}$.

The CheckConsistency function is the core of the CSP al-
algorithm, it both updates the set of distances $d_{x_i,y_i}$ and the domain variables $D_i$ by forward checking. In particular it executes two main steps:

**Temporal propagation.** every time a new inequality $x_i - y_i \leq r_i$ is added to the $G_d$ graph, the set of distances $d_{x_i,x_j}$ is updated by a simple $O(n^2)$ algorithm.

**Forward checking.** After the previous step, for each not assigned meta-variable the domain $D_i$ is checked for consistency (forward checking). Given the current solution represented by $G_d$, each value $x_i - y_i \leq r_i$ belonging to a not assigned variable and which induces a negative cycles on $G_d$ is removed. In other words, each time a value $\delta_i \equiv x_i - y_i \leq r_i$ satisfies the test $r_i + d_{x_i,y_i} < 0$, then $\delta_i$ is removed from the corresponding domain $D_i$. In the case that one domain $D_i$ becomes empty, the function CheckConsistency returns false.

The CheckConsistency step contributes to avoid investigation of search states proved inconsistent and the other two steps (Steps 5 and 6 of Figure 1) are used to guide the search according to heuristic estimators.

**SelectVariable.** This applies the simple and effective Minimum Remaining Values (MRV) heuristic: variables with the minimum number of values are selected first. It is worth noting that the heuristic just ranks the possible choices deciding which one to do first but all the choices should be done (it is not a non deterministic search step).

**ChooseValue.** This represents a non deterministic operator, which starts a different computation for each domain values. Obviously in our implementation we use a depth-first search strategy, where there is no particular values ordering heuristic. However, in the case a constraint (variable) is always satisfied by the current partial solution $S_p$, that is, a constraint disjunct $x_i - y_i \leq r_i$ exists such that holds the condition $d_{y_i,x_i} < r_i$, no branching is created. In fact, the current constraint is implicitly “contained” in the partial solution and it will be satisfied in all the solution created from $S_p$.

**Integrating SAT Features**

The current version of our CSP solver integrates also the so-called semantic branching (Armando, Castellini, and Giunchiglia 1999). This is a feature that in the SAT approach comes for free and that in the CSP temporal representation is to be explicitly inserted. It avoids to test again certain conditions previously proved inconsistent. The idea behind semantic branching is the following, let us suppose that the algorithm builds a partial solution $S_p = \{\delta_1, \delta_2, \ldots, \delta_p\}$ and a not assigned meta-variable is selected which has a disjunct set of two elements $\{\delta', \delta''\}$. Let us suppose that the disjunct $\delta'$ is selected first and no feasible solution exists from the partial solution $S_p \cup \{\delta'\}$. In other words, each search path from the node $S_p \cup \{\delta'\}$ arrives to an infeasible state. In this case the depth-first search process removes the decision $\delta'$ from the current solution and tries the other one ($\delta''$). However, even if the previous computation is not able to find a solution, it demonstrates that with regard to the partial solution $S_p$ no solution can contain the disjunct $\delta'$. If we simply try $\delta''$, we lose the previous information, hence, before trying $\delta''$, we add the condition $\neg \delta'$ (that is $x' - y' > r_i$) to the partial solution. It is worth nothing that in this case it is important to make explicit the semantic branching by adding the negation of the disjunct, because the values in the domains $D_i$ are not self-exclusive. In other cases, for example a scheduling problem, where branching is done with regard to the temporal ordering of pairs of activities $A$ and $B$, semantic branching is not useful. In fact when $A$ before $B$ is chosen the case $B$ before $A$ is implicitly excluded.

In this section we have described our basic algorithm that integrates some of the previous analysis in a meta-CSP search framework. From now on we call this algorithm CSP and it is the base for the description of the incremental forward checking of the next section.

**Incremental Forward Checking**

The algorithms for solving DTP introduced at the beginning of this section is based on the meta-CSP schema with some additional features. In particular, it uses the enriched backtracking schema called semantic branching. To further improve the performance of the CSP approach we have investigated aspects connected to the quantitative temporal information. This aspect has received less attention in (Stergiou and Koubarakis 1998; Armando, Castellini, and Giunchiglia 1999; Stergiou and Koubarakis 2000). In particular, in this section we introduce a method to significantly decrease the number of forward checks by using the temporal information. Its general idea is relatively simple.

**Rationale.** When a new disjunct $\delta$ is added to a partial solution, the temporal propagation algorithm inside CheckConsistency updates only a subset of the distances $d_{x_i,x_j}$(usually a “small” subset). The forward checking test on disjuncts is performed w.r.t. the distances in the graph $G_d$. It is of no use to perform a forward checking test of the form $d_{x_i,y_i} + r_i < 0$ on a disjunct $\delta_i$ when the distance $d_{x_i,y_i}$ has not been changed w.r.t. the previous state.

This basic observation can be nicely integrated in CSP with the additional cost of a static preprocessing needed to create for each pair of nodes $(x_i, y_j)$ the set of affected meta values $AMV(x_i, y_j)$.

**Affected meta-values w.r.t. a pair $(x_i, y_j)$**. Given a distance $d_{x_i,y_j}$ on $G_d$ the set of affected meta values discriminates which subset of disjuncts are affected by an update of $d_{x_i,y_j}$. The set $AMV(x_i, y_j)$ associated to the distance $d_{x_i,y_j}$ (or the pair $(x_i, y_j)$) is defined as the set of disjuncts $x - y \leq r$ whose temporal variables $x$ and $y$ respectively coincide with the variables $y_j$ and $x_i$ ($AMV(x_i, y_j) = \{(x - y \leq r : x = y_j, y = x_i)\}$).

Given a DTP, the set of its $AMV$s is computed once for all with a preprocessing step with a space complexity $O(m + n^2)$ and a time complexity $O(n^3 \ln n)$ (as explained below each set $AMV$ is represented as a sorted list according to
the values $r$). The information stored in the AMVs can be used in a new version of CSP (CSPi) we call “incremental forward checking” (CSPi). It requires a modification of the CheckConsistency function. The new incremental version of the CheckConsistency works in two main steps:

1. The distances $d_{x_i, y_j}$ are updated and the set of distances that have been changed is collected.

2. Given such set, for each $d_{x_i, y_j}$ the corresponding AMV($x_i, y_j$) is taken, and its values are forward checked. In particular, all the set AMV($x_i, y_j$) are represented as a list of disjuncts sorted according to the value of $x$ and the forward checking test $d_{x_i, y_j} + r < 0$ is performed from the disjunct with the smallest value of $r$. In this way, when a test fails on the list element $\delta$, it will fail also on the rest of the list and the forward checking procedure can stop on AMV($x_i, y_j$).

In the experimental section we show that the algorithm CSPi (constraint-based solver with incremental forward checking) strongly improves with respect to the basic CSP and becomes competitive with the best results available in the literature.

**New Solving Strategies for the DTP**

In this section we propose two new solving strategies for the DTP based on the work (Oddi and Cesta 2000). In particular we propose: (1) a new variable ordering heuristic; (2) an arc-consistency (Mackworth 1977) filtering strategy.

The rational behind the first method is based on the observation that given a DTP problem, and considered a value $\delta_i = x_i - y_i \leq r_i$, during the solving process $\delta_i$ is removed by forward checking from its domain $D_i$ when induces negative cycles in the current solution represented by the $G_d$ graph. On the basis of the previous observation we propose the following variable ordering strategy: **select the subset of variables with minimum number of remaining values $x_i - y_i \leq r_i$, and within this subset, the variable with maximal number of negative coefficients $r_i$.** The values $\delta_i$ with negative coefficients $r_i$ are crucial to the existence of a solution to a DTP. In fact, it is simple to see that a DTP instance without negative $r_i$ values has always a solution. On the other hand, the presence of negative $r_i$ values generate negative cycles on the graph $G_d$ and induces inconsistent partial solutions. This strategy has the main purpose of pruning the search tree in its early stages, trying to create as many as possible negative cycles, in this way the strategy maximizes the probability of finding negative cycles at the early steps of the search tree. As we will see in the experimental section this strategy is effective in the transition phase (Liu 1996) of a DTP problem, where the probability of find a solution is very low.

The second solving method can be explained by giving a new version of the CheckConsistency algorithm used in the general algorithmic template described in Figure 1. The aim of this solving method is reducing the dimension of the search tree by the application of a more effective filtering strategy and to explore the possibility of finding tradeoffs among number of consistency checks, number of visited search nodes, and CPU time. In particular, we propose an arc-consistency filtering algorithm such that, among the set of filtering methods analyzed during our preliminary experimentation, is the one which gave the better performance both in CPU time and number of consistency checks. The proposed filtering algorithm works in two main steps:

1. It applies the incremental forward checking method described in the previous section, when at least one variable domain becomes empty then CheckConsistency returns false, otherwise the following second step is executed.

2. The set of not-assigned variables which are modified by the application of the previous first step is considered and used to initialize the propagation queue $Q$ of the used arc-consistency filtering method. The filtering method is executed to remove further values, when at least one variable domain becomes empty then CheckConsistency returns false, otherwise returns true.

Figure 2 shows the arc-consistency filtering algorithm based on the well-known AC-3 schema (Mackworth 1977). It takes as an input the set $Q_{init}$ of modified variables and applies the 2-consistency filtering strategy by the Revise operator. In this case the operator has the following definition: Revise($c_i, c_j$) removes from the domains $D_i, D_j$ each value $x_i - y_i \leq r_i$ which does not have support. That is, a value $x_i - y_i \leq r_i$ is removed from the domain $D_i$ when there is no value $x_i - y_i \leq r_j$ in the set $D_j$ such that $r_i + d_{x_i, y_i} + r_j + d_{x_j, y_j} \geq 0$ holds. When the procedure stops, it returns the set of variables with reduced domain of values.

In the experimental section we compare this strategy with the other ones proposed in this work considering three different parameters: the number of consistency checks (we consider the test $r_i + d_{x_i, y_i} + r_j + d_{x_j, y_j} \geq 0$ performed inside the Revise operator as equivalent to a forward checking test), the total CPU time and the number of visited search nodes.

**Experimental Evaluation**

We adopt the same evaluation procedure used in (Stergiou and Koubarakis 1998; Armando, Castellini, and Giunchiglia 1999) and use the random DTP generator defined by Stergiou and Koubarakis. DTP instances are generated according to the parameters $\langle k, n, m, L \rangle$ ($k$: number of disjuncts per clause, $n$: number of variables, $m$: number of disjuncts (temporal constraints); $L$: a positive integer such that all the constants $r_i$ are sampled in the interval $[-L, L]$).
Figure 3: Median number of forward checks for $n \in \{10, 12, 15\}$.

In particular, according to (Stergiou and Koubarakis 1998; Armando, Castellini, and Giunchiglia 1999) experimental sets are generated with $k = 2, L = 100$ and the domain of $r_i$ is on integers not on reals, as in the general definition of DTP. Experimental results are plotted for $n \in \{10, 12, 15, 20, 25, 30\}$, where each curve represents the number of consistency checks versus the ratio $\rho = m/n$.

Figure 4: Median number of forward checks for $n \in \{20, 25, 30\}$.

(in all the results of Figures 3, Figures 4 and 5 $\rho = m/n$ is an integer value which ranges from 2 to 14). The median number of forward checks over 100 random problem instances for different values of $\rho$ is plotted in Figures 3 and 4 where three different type of results are compared: (1) the performance of the best algorithm proposed in (Stergiou and Koubarakis 1998) and labeled with $SK$; (2) the re-
sults of the SAT-based solving methods, there are two methods: the first one labeled with $TSAT_{I8(2)}$, which corresponds to the best results described in the work (Armando, Castellini, and Giunchiglia 1999), and a second one, labeled with $TSAT_{I8(3)}$, which represents some new results only published on the $TSAT$ web page (see the reference (Armando, Castellini, and Giunchiglia 1999) for the URL): (3) the performance of our constraint-based approach, in particular the curve labeled with $CSPi$ corresponds to the best results in the paper (Oddi and Cesta 2000), and the one labeled with $CSPineg$ represent the new results obtained with the heuristic strategy defined in the previous section. The algorithms are implemented in Common Lisp and the reported results are obtained on a SUN UltraSparc 10 (440MHz). All the results are obtained setting a timeout of 1000 seconds of CPU time.

There are several comments on the $CSPineg$ performance: (a) all the curves have the same behavior of the previous results. It is confirmed that the harder instances are obtained for $\rho \in \{6, 7\}$ and for such values the percentage of solvable problems becomes $< 10\%$. Figure 5(d) plots the percentage of problems solvable by $CSPineg$ on different $n$. When the number of variables $n$ increases the hardest region narrows; (b) the median number of forward checks show that $CSPineg$ significantly improves over $CSPi$. This fact shows that the new selection variable strategy is very effective, and indirectly confirms that there could be further space for investigating improvements of the CSP approach; (c) the $CSPineg$ compares very well with the pre-existing approaches, it outperforms the others for $n \in \{10, 12, 15, 20\}$ and it is competitive with $TSAT_{I8(3)}$ for $n \in \{25, 30\}$. However, further work will be needed to clearly outperform $TSAT_{I8(3)}$ on all $n$. One possible direction of research is the use of more effective filtering strategies to reduce the dimension of the search tree. However, the use of a more powerful filtering strategy has a price of an higher computational time. Hence, the challenge is to find a good tradeoff among number of consistency checks, number of search nodes and CPU time.

Figure 5(a) shows a comparison between the performance of our constraint-based algorithm $CSPineg$ and the other one which uses the arc-consistency filtering strategy (labeled with $CSPiac$) introduced in the previous section. With respect to number of forward checks $CSPiac$ performs about one order of magnitude worse than $CSPineg$, where in the case of the arc-consistency algorithm we consider the test $r_i + d_{x,y} + r_j + d_{x,y} \geq 0$ as equivalent to a forward check. On the other hand, if we consider the CPU time performance (Figure 5(c)), the ratio between the $CSPiac$ and $CSPineg$ CPU times is less than 3 in the transition phase. The analysis is completed by the results in Figure 5(b), which show that the $CSPiac$ strategy is able to reduce about 25% the number of explored search nodes respect to the $CSPineg$ performance.

About the results of Figure 5 we observe: (a) the arc-consistency strategy performs an higher number of consistency checks respect to the forward checking strategy and many of the performed checks are probably unnecessary, in fact, after each solution modifications, many distances on the $G_d$ graph remain unchanged, hence many tests of the form $r_i + d_{x,y} + r_j + d_{x,y} \geq 0$ are unnecessarily performed.

The experimental results confirm that the proposed CSP approach is competitive with the best results available in the current literature, in fact our results are comparable with the ones obtained by the $TSAT$ approach, in addition, for lower values of the ratio $\rho = m/n (\leq 5)$ the $CSPineg$ is significantly better with respect to all the others (it is to be noted also that in many practical applications the condition $\rho \leq 5$ is likely to be verified). On the other hand, further investigation is needed to realize a competitive and specialized arc-consistency solving algorithm (e.g., see the work (Liu 1996)). However, in this experimentation some useful observations about tradeoffs among number of forward checks, number of search nodes explored, and CPU time are pointed out and represent a good starting point for future research directions.

**Conclusion**

This paper has extended the constraint-based approach, initially introduced in (Stergiou and Koubarakis 1998) and later improved in (Oddi and Cesta 2000) for solving the DTP temporal problem. We propose two new solving methods for DTP. The first one is an heuristic strategy for variable ordering, which improves forward checking performance up to an order of magnitude respect to the results described in (Oddi and Cesta 2000) and allowing a real competition with the existing best SAT approach. An interesting area where $CSPineg$ constantly outperforms all other approaches (when $\rho \leq 5$) emerges from the proposed empirical evaluation. The second solving method uses a more sophisticated arc-consistency filtering algorithm. In this case the aim of the method is reducing the dimension of the search tree by the application of a more effective filtering strategy and to explore the possibility of finding better tradeoffs among the number of consistency checks, number of visited search nodes, and CPU time. The results proposed in the paper suggest that an useful research direction is the definition of an incremental and specialized version of the arc-consistency filtering algorithm.

**Acknowledgments**

This work is supported by ASI (Italian Space Agency) under ASI-ARS-99-96 contract and by the Italian National Research Council.

**References**


Figure 5: CSPineg and CSPiac performance.


