Incremental Forward Checking for the Disjunctive Temporal Problem

Angelo Oddi[1] and **Amedeo Cesta**[2]

Abstract. This paper studies algorithms for the Disjunctive Temporal Problem (DTP) a quite general temporal reasoning problem introduced in [12]. This problem involves the satisfaction of a set of constraints represented by disjunctive formulas of the form $x_1 - y_1 \leq r_1 \vee x_2 - y_2 \leq r_2 \vee \ldots \vee x_k - y_k \leq r_k$. The paper starts sketching why DTPs are potentially very useful in plan management applications, then analyzes the current solutions to DTP, and introduces a constraint-satisfaction problem solving algorithm where different aspects of current DTP's literature are integrated. This basic algorithm is then improved exploiting the quantitative temporal information in the "distance graph". Using this knowledge an incremental version of the forward checking is obtained and shown to be competitive with current best results. The whole approach allows to understand pros and cons of the current algorithms for the DTP and suggests further future developments as discussed in the final part of the paper.

1 INTRODUCTION

Reasoning about temporal constraints has always deserved major attention in AI (e.g., [1]) due to its relevance for applications (e.g., planning, scheduling, diagnosis). Quantitative temporal constraints have been studied in [5] where a basic problem (called STP) and a disjunctive problem (called TCSP) have been defined.

The STP is a constraint satisfaction problem where the generic temporal variables x and y should satisfy bounds on their distance of the kind $x - y \leq r$ with r being a real value. No disjunction is allowed among constraints and, as a consequence, checking consistency of an STP constraint network is polynomial. For such efficiency reasons the STP is commonly used in several planning and scheduling algorithms (e.g., [7] and many others) where consistency checking of the temporal constraints is continuously performed during problem solving.

In several recent applications the importance of more expressive quantitative temporal reasoning is underscored. For example in continual planning applications [6] a relevant capability is the continuous management of "rich temporal plans" [9, 13]. To this purpose, the representation of temporal disjunction allows a leverage of systems' capability. For example, it avoids a too early commitment on action orderings. In fact, when storing information of temporal plans that should not immediately be executed, powerful features are: (a) storing plans with potentially multiple executions; (b) knowing that at least one temporally consistent scenario exists.

One way to represent disjunction in quantitative temporal networks is the TCSP mentioned above. It allows constraints of the form $x - y \leq r_1 \vee x - y \leq r_2 \vee \ldots \vee x - y \leq r_k$. A further generalization of the TCSP has been proposed in [12] where a problem is defined whose constraints have the form $x_1 - y_1 \leq r_1 \vee x_2 - y_2 \leq r_2 \vee \ldots \vee x_k - y_k \leq r_k$. In [2] this last problem is referred to as Disjunctive Temporal Problem (DTP) and we use this name in the paper.

DTPs have been studied in two previous works: (a) in [12] several constraint-based (CSP) algorithms (in the line of [10]) are defined and experimentally compared. One of them, based on *forward checking* [8], is shown to be the best; (b) in [2] DTP is modeled as a propositional satisfiability (SAT [4]) 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 in [12].

Our starting point has been the observation of the sharp difference between these two results. Some possible questions arise: does this mean that the SAT-based approach is the best? Is it possible to improve a CSP-based approach? Is it of any use the knowledge about temporal distances that is usually represented in a specialized temporal constraint network?

This paper follows a CSP-based approach and focus on using the "quantitative reasoning" that can come out from a temporal constraint network representation. We basically observe that the effects of quantitative temporal information to improve global performance of DTPs have not been explored enough in previous works. Using such knowledge we are able to produce an *incremental forward checking* algorithm which has comparable performance (measured as number of forward checks) with the best SAT-based version proposed in [2]. Furthermore, the rationale behind incremental forward checking is quite general and can be exploited in other solvers that rely on "quantitative temporal reasoning".

The paper is structured as follows. Section 2 introduces the basic concepts used in the paper. Section 3 gives a basic CSP algorithm which integrates results from both previous works. Incremental forward checking is introduced in Section 4 and an experimental evaluation of the different approaches is given in Section 5. Section 6 ends the paper discussing some remarks from our experiences and drawing some directions for future research.

2 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 $C = \{c_1, c_2, \ldots c_m\}$ of the form $x_1 - y_1 \leq r_1 \vee x_2 - y_2 \leq r_2 \vee \ldots \vee x_k - y_k \leq r_k$, where r_i are real numbers. A DTP is consistent if an assignment to the variables exists such that in each constraints in the set $C = \{c_1, c_2, \ldots c_m\}$ at

[1] IP-CNR, Italian National Research Council, Viale Marx 15,I-00137 Rome, Italy, email: oddi@ip.rm.cnr.it
[2] IP-CNR, Italian National Research Council, Viale Marx 15,I-00137 Rome, Italy, email: cesta@ip.rm.cnr.it

least one disjunct 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 [5]) 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 [12, 2] do this basic search step.

Previous Work. Both [12, 2] 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 [12] 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 [3]. In [2] 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 use a state of the art SAT-solver that they enrich 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 w.r.t. SK. Then they add a further preprocessing step called IS that basically produces a more accurate SAT encoding because it codifies mutual exclusion conditions between propositions that exists in the temporal information, but were lost by the first standard encoding. Their second algorithm called $TSAT_{IS}$ furtherly improves up to one order of magnitude w.r.t. $TSAT$.

DTP Consistency Checking as a Meta-CSP. Before introducing our algorithm we underscore the possibility of seeing 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" [5]. 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 $(x_i, y_i), (y_i, y_{i1})(y_{i1}, y_{i2}) \ldots (x_{il}, y_j)$ 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 iff G_d does not contain closed path with negative length (*negative cycles*) [5].

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: $d_{x_1 x_i} \leq d_{x_1 y_i} + r_i$, that is $d_{x_1 x_i} - d_{x_1 y_i} \leq r_i$. 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.

```
CSP-DTP-SOLVER(dtp,S)
1.  if CheckConsistency(dtp)
2.     then if IsaSolution(dtp)
3.             then return(S)
4.             else begin
5.                c ← SelectVariable(dtp)
6.                δ ← Choose Value(dtp,c)
7.                CSP-DTP-SOLVER(dtp,S ∪ {δ})
8.             end
9.     else return(Fail)
10. end
```

Figure 1. A CSP solver for the DTP.

3 A CSP ALGORITHM FOR DTP

In this work we mainly follow the CSP approach of Stergiu and Kubarakis looking for improvements of a constraint satisfaction procedure for DTP. A basic CSP algorithm for solving DTP instances is shown in Figure 1. The procedure 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 algorithm, it both updates the set of distances $d_{x_i y_j}$ 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 unassigned 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 an unassigned 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 δ_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 in one of the two internal analyses. The other two steps (Steps 5 and 6 of Figure 1) are used to guide the search according to heuristic estimators.

SelectVariable. It applies the simple and effective minimum remaining values heuristic. Hence, 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.

Analyzing the SAT approach. In [12] the exact implementation of SK is not given but it is likely to be close to the algorithm described till here [3]. The current version of our CSP solver integrates also the so-called *semantic branching* [2]. 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 an unassigned 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 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.

A further observation concerns [2]. Their best performance is obtained when at propositional level the IS preprocessing is added. This function adds to the original disjunction —which represents the problem constraints— a set of binary propositional relations (up to a quadratic number of propositional relations in the number of disjuncts) which represents mutual exclusions between pairs of disjuncts belonging to different constraints. It is possible to verify that in our CSP approach it is not useful to add the latter set of new constraints because we work directly on a quantitative representation of the problem where this mutual exclusive information is implicitly encoded.

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.

[3] Stergiou and Koubarakis use also backjumping together with standard forward checking but the results did not show major differences [11]

4 INCREMENTAL FORWARD CHECKING

The algorithms for solving DTP introduced in the previous section is based on the *meta*-CSP schema with some additional features. In particular, it uses the enriched backtracking schema called *semantic branching*. In the experimental section we show that such algorithm improves the performance of the original KS but is not comparable to both $TSAT$ and $TSAT_{IS}$.

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 [12, 2]. 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 δ is added to a partial solution, the algorithm *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 distance 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 δ_i in all cases in which 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 $\langle x_i, y_j \rangle$ the set of *affected meta values* $AMV(x_i, y_j)$.

Affected meta-values w.r.t. a pair $\langle x_i, y_j \rangle$. 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 $\langle x_i, y_j \rangle$) 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 ($AMV(x_i, y_j) \doteq \{x - y \leq r : x = y_j, y = x_i\}$).

Given a DTP, the set of its AMVs is computed once for all with a preprocessing step with a space complexity $O(m + n^2)$ and a time complexity $O(n^3 \ln n)$. The information stored in the AMVs can be used in a new version of CSP 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 $d_{x_i y_j}$ in it 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 r 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 δ, 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 next section we experimentally show that the new algorithm $CSPi$ strongly improves with respect to the basic CSP and becomes competitive with the best results available in the literature.

5 EXPERIMENTAL EVALUATION

We adopt the same evaluation procedure used in [12, 2] and use the random DTP generator defined by Stergiou. 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 disjunction (temporal constraints); L: a positive integer such that all the constants r_i are sampled in the interval $[-L, L]$).

As in the works [12, 2] experimental sets are generated with $k = 2$ and $L = 100$. As done in these works, the domain of r_i is on integers not on reals as in the general definition of DTP. This helps us to implement the negation used in *semantic branching*. Experimental results are plotted for $n \in \{10, 12, 15, 20, 25\}$, where each curve represents the number of consistency checks versus the ratio $\rho = m/n$. The median number of checks over 100 random samples for different values of ρ is plotted in Figures 2(a)-2(e). Figure 2(f) plots the percentage of problems solvable by $CSPi$ on different n. The algorithm is implemented in Allegro Common Lisp and the reported results are obtained on a SUN UltraSparc 30 (266MHz). All the results are obtained setting a timeout of 1000 seconds of CPU time.

Several comments are worth doing: (a) first of all, all the curves of the CSP and $CSPi$ algorithms 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 $< 10\%$. When the number of variables increases the hardest region narrows; (b) the median number of forward checks show that $CSPi$ significantly improves over CSP. This fact: (1) shows that the incremental forward checking implemented using the AMVs is very effective. This is confirmed by the fact that the CPU time spent by $CSPi$ is half the time spent by CSP; (2) indirectly confirms that there could be further space for investigating improvements of the CSP approach; (c) the $CSPi$ compares very well with the preexisting approaches; (d) indirectly our results confirm the goodness of the results obtained in [2]. Further work will be needed to clearly outperform $TSAT_{IS}$ on all n. In particular $TSAT_{IS}$ turns out to be very good for the values at the transition phase of the problem. Seeing the other part of the coin, this confirm that $CSPi$ contains good ideas because $TSAT_{IS}$ uses one of the best SAT-solvers available; (e) finally, very important is the fact, confirmed in all the figures, that for lower values of the ratio $\rho = m/n$ (≤ 5) the $CSPi$ is significantly better with respect to all the others. It is worth reminding that the ratio ρ represents the number of disjunctive constraints w.r.t. the number of temporal events. This means that when the disjunctive constraints are "not so many" the principle of "locality of computation" implemented by the AMVs is rather useful. It is to be noted also that in many practical applications the condition $\rho \leq 5$ is likely to be verified.

6 CONCLUSIONS

This paper has extended the CSP approach, initially introduced in [12], to solving the DTP temporal problem. The DTP is going to become very relevant in many planning application (see the short discussion in the introduction). Our algorithm integrates a new method to perform incremental forward checking based on the use of quantitative temporal information with additional features based on [2]. In particular, the use of the semantic branching schema allows our CSP to constantly outperform SK but is not enough for competing with the SAT-based approaches. The incremental forward checking schema improves performance up to a further order of magnitude allowing competition with the best SAT approach. An interesting area where $CSPi$ constantly outperforms all other approaches (when $\rho \leq 5$) emerges from an experimental evaluation.

Going back to our original motivating questions we say that the CSP approach may compete with the SAT-based one provided that the implicit information encoded in the temporal constraints is actually exploited. Ours is a first result in this direction but more issues still deserve investigation.

We are currently following two different directions: (a) we are exploring more sophisticated filtering algorithms (generally applied in solving discrete CSP problems). In particular we are working at an incremental version of 2-consistency (arc consistency) and 3-consistency filtering algorithms. Our goal is to analyze the tradeoffs among number of forward checks, number of search nodes explored, and CPU time; (b) we are also developing algorithms to compute upper bounds of the shortest distances $d_{y_i x_i}$ useful to improve performance by the execution in advance of a forward checking test of the form $d_{y_i x_i} + r_i < 0$.

ACKNOWLEDGEMENTS

Cesta and Oddi's work is supported by ASI (Italian Space Agency) under ASI-ARS-99-96 contract and by the Italian National Research Council. We would like to thank Alessandro Armando, Claudio Castellini and Enrico Giunchiglia for useful discussions on this subject.

REFERENCES

[1] J. Allen, 'Maintaining Knowledge About Temporal Intervals', *Communications of ACM*, **26**, 832–843, (1983).

[2] A. Armando, C. Castellini, and E. Giunchiglia, 'SAT-based Procedures for Temporal Reasoning', in *Proceedings 5th European Conference on Planning (ECP-99)*, (1999). (available at `http://www.mrg.dist.unige.it/~drwho/Tsat`).

[3] N. Chleq, 'Efficient Algorithms for Networks of Quantitative Temporal Constraints', in *Proceedings of the Workshop CONSTRAINTS'95 (held in conjunction with FLAIRS-95)*, pp. 40–45, (1995).

[4] S. Cook and D. Mitchell, 'Finding Hard Instances of the Satisfiability Problem: a Survey', in *Satisfiability Problems: Theory and Applications*, DIMACS Series in Discrete Mathematics and Computer Science N.35, (1998).

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

[6] M.E. desJardin, E.H. Durfee, C.L. Ortiz, and M.J. Wolverton, 'A Survey of Research in Distributed, Continual Planning', *AI Magazine*, **20**(4), 13–22, (1999).

[7] M. Ghallab and H. Laruelle, 'Representation and Control in IxTeT, a Temporal Planner', in *Artificial Intelligence Planning Systems: Proceedings of the Second International Conference (AIPS-94)*, (1994).

[8] R.M. Haralick and G.L. Elliott, 'Increasing Tree Search Efficiency for Constraint Satisfaction Problems', *Artificial Intelligence*, **14**, 263–313, (1980).

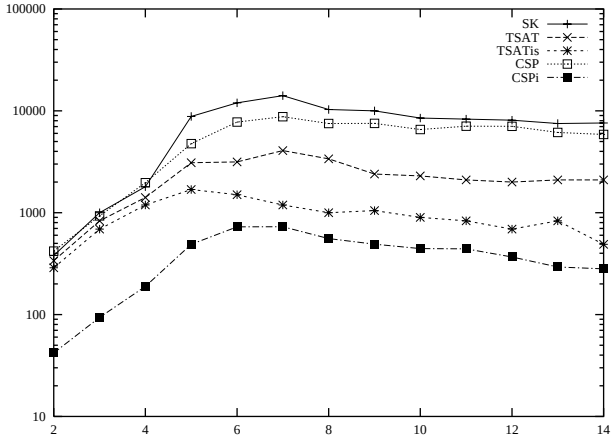
[9] M.E. Pollack and J.F. Horty, 'There's More to Life Than Making Plans: Plan Management in Dynamic Multiagent Environment', *AI Magazine*, **20**(4), 71–83, (1999).

[10] P. Prosser, 'Hybrid Algorithms for the Constraint Satisfaction Problem', *Computational Intelligence*, **9**(3), 268–299, (1993).

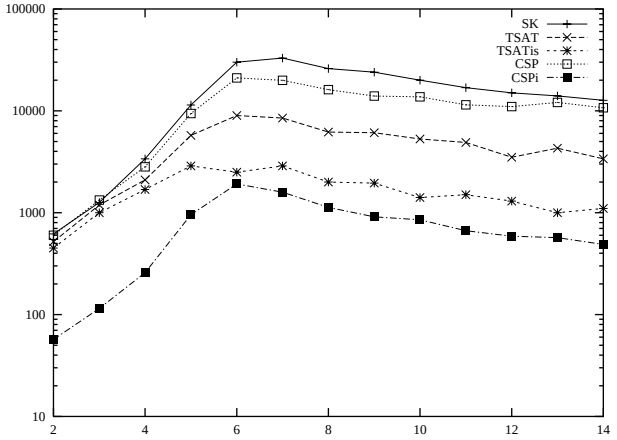
[11] C. Stergiou, 'Personal communication, may 2000'.

[12] K. Stergiou and M. Koubarakis, 'Backtracking Algorithms for Disjunctions of Temporal Constraints', in *Proceedings 15th National Conference on AI (AAAI-98)*, (1998).

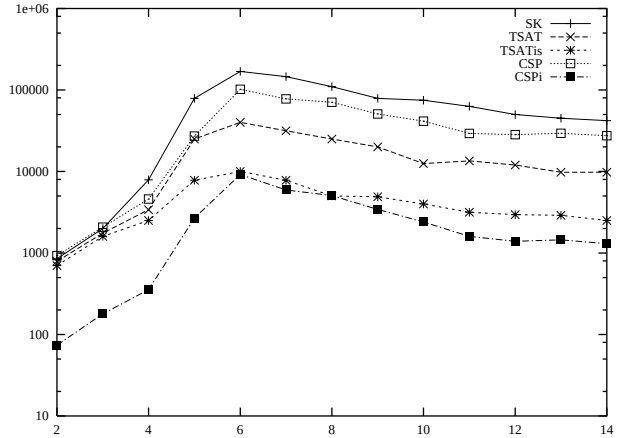
[13] I. Tsamardinos, M. E. Pollack, and J. F. Horty, 'Merging Plans with Quantitative Temporal Constraints, Temporally Extended Actions, and Conditional Branches', Technical report, Intelligent Systems Program, University of Pittsburgh, (1999).



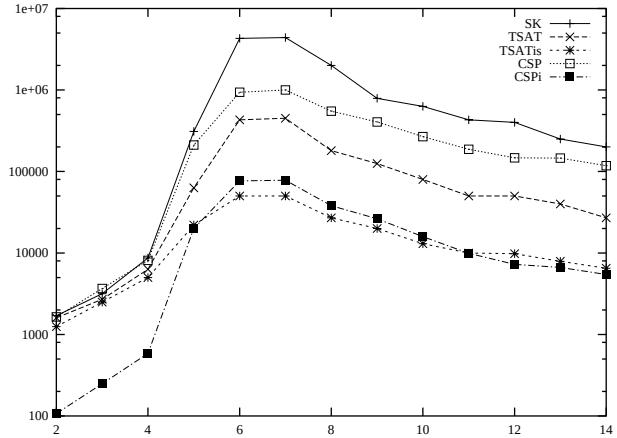
(a) Median number of forward checks for $n = 10$



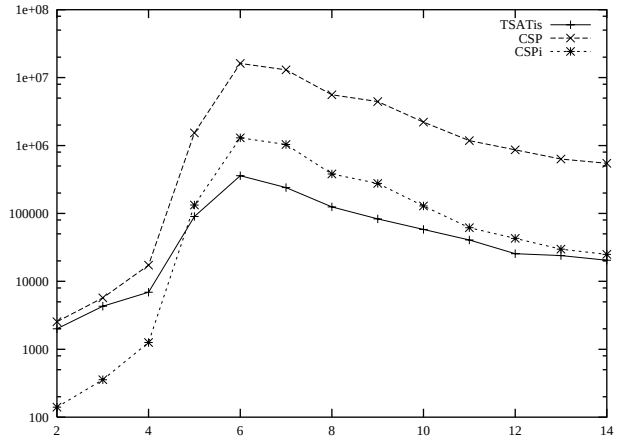
(b) Median number of forward checks for $n = 12$



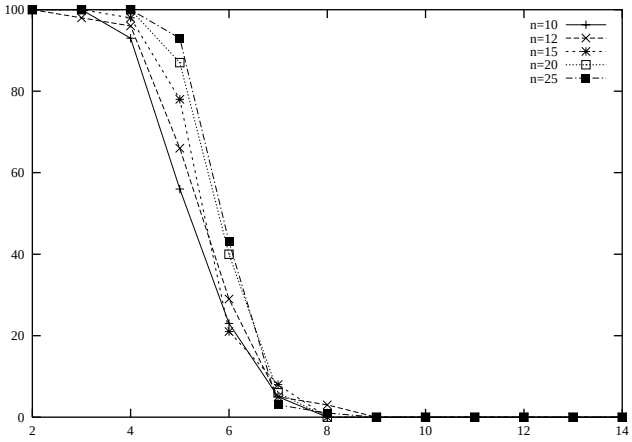
(c) Median number of forward checks for $n = 15$



(d) Median number of forward checks for $n = 20$



(e) Median number of forward checks for $n = 25$



(f) Percentage of solvable problems for different values of n

Figure 2. Experimental results.