Topology-based Variable Ordering Strategy for Solving Disjunctive Temporal Problems

Yuechang Liu, Yunfei Jiang, Hong Qian
Software Research Institute, Sun Yat-Sen University
135 West Xingang Rd., 510275 Guangzhou, P.R.China
ychangliu@gmail.com,issjyf@mail.sysu.edu.cn,honggzb@yahoo.com.cn

Abstract

Many temporal problems arising in automated planning and scheduling can be expressed as Disjunctive Temporal Problems (DTPs). Most of DTP solvers in the literature treat DTPs as Constraint Satisfaction Problems (CSPs) or Satisfiability Problems (SATs), and solve them using standard CSP (SAT) techniques. Basically DTPs are represented through logically related topological relations between temporal variables, however, unfortunately little work has been done on exploiting the topological information to direct the search for DTP resolving. According to the "fail-first"(FF) principle for dynamic variable ordering (DVO) heuristics in CSP literature, this paper proposes a DVO which is based on the topological structure of DTP (which is defined to be Disjunctive Temporal Network). Experimental results reveal that the proposed DVO outperforms Minimal Remaining Values heuristics–a DVO that is widely used in existing DTP solvers, especially for the hard and large-scale problems. And, a CSP based procedure with the best of the heuristics wins TSAT++ on most of the test problems.

1 Introduction

In this paper we study variable ordering strategy in solving Disjunctive Temporal Problems (DTPs), which are an essential part for building systems that reason about time and actions [13, 10]. DTP models events and their relationships (as distances between events), and provide the means to specify the temporal elements of an episode with a temporal extent. Examples of such an episode are as diverse as a story, a discourse, a manufacturing process, the measurements executed by the Hubble space telescope, the activities of a robot, or the scheduling for a production plan. The ability to efficiently process DTPs is a prerequisite for automate the planning and execution of complex tasks [13, 10, 11]. This paper describes an efficient dynamic variable ordering heuristics for the meta Constraint Satisfaction Problem (meta-CSP) solution of DTP [12, 2, 9, 13, 1].

A major research effort in the Constraint Processing community is finding effective variable/value ordering heuristics, especially dynamic variable ordering (which is generally superior to static one). According to "fail-first" (FF) principle [8], Minimum Remaining Values heuristics (MRV) is proved to be a very effective DVO [3]. And, as only variable ordering heuristics it is used in almost all existing DTP solvers that treat DTPs as meta-CSPs [12, 9, 13]. Those solvers every time select the variable with minimum available values to instantiate. Though particularly simple at the conceptual level, it performs well in DTP solving. However, considering the fact that many other heuristics proposed in DTP literature, such as removal of subsumed variables (RSV), semantic branching (SB), pursuit the same effect as DVO–focusing the search on the variables which are more likely to conflict with each other (similar to the "back-door" variables in CSP literature), development of new DVO heuristics is still essential in DTP as well as CSP solution.

To the best of our knowledge, there is still little work reported in literature on exploiting the internal topological structure to guide variable selection (we called it TVO) in DTP solving. In contrast to MRV heuristics (which only concerns the number of candidate values of variables), our approach evaluates the potentiality of individual values involving in conflicts with instantiated variables for pending variables.

The proposed approach works upon a graphical model of DTP which we defined as Disjunctive Temporal Network (DTN), in which each simple temporal constraint (an inequation) corresponds to a labelled edge. For each pending variable TVO evaluates the potentiality of its edges involving in negative cycles (the cycles with negative total weight) with instantiated variable, and select the variable with the most potentiality to instantiate. By conducting experiments on random DTPs, it is revealed that TVO is really helpful

1530-1311/08 $25.00 © 2008 IEEE
DOI 10.1109/TIME.2008.23

to reduce both the consistency checks and spent CPU time, and it outperforms MRV on most of, especially hard and large-scale, test problems.

This paper is structured as follows: first of all, some necessary definitions, notations and the task we address will be introduced in next section. Then, a subsequent section will be dedicated to a detailed description of TVO, followed by another section which displays the experimental results. The last section concludes this paper.

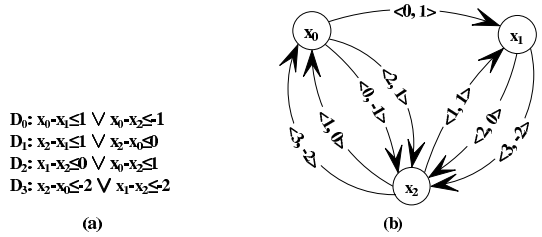


Figure 1. A DTP and its DTN

2 Preliminaries

We first introduce some necessary terminology and theoretic background.

2.1 Definitions

Definition 1. ***Disjunctive Temporal Problem (DTP)*** *A disjunctive temporal problem is a tuple $< X, D >$, where $X = \{x_1, x_2, ..., x_n\}$ is a set of* ***temporal variables***[1]*, and D is a set $\{D_1, D_2, \ldots, D_m\}$ of* ***clauses****, each of which is a disjunction of the form $c_{i1} \vee c_{i2} \vee \ldots \vee c_{ik_i}$ (each c_{ij} is a* ***simple temporal constraint*** *of the form $x_i - x_j \leq c$, where x_i and x_j are temporal variables and c a constant.*

Specially, a DTP with $|D_i| = 1, i = 1, 2, ..., m$ is called a **Simple Temporal Problem (STP)**. An STP can be represented by a weighted digraph named **Simple Temporal Network (STN)** [5], in which each vertex corresponds to a temporal variable, and each directed edge represents a simple temporal constraint, and the weight attached to an edge is the constant part of the inequation.

Like STN for STP, we can also define a graphical model for DTP, which is defined to be Disjunctive Temporal Network (DTN).

Definition 2. ***Disjunctive Temporal Network (DTN)*** *Given a DTP $P =< X, D >(X = \{x_1, x_2, ..., x_n\}$, $D = \{D_1, D_2, \ldots, D_m\})$, its* ***Disjunctive Temporal Network*** *(notated as N) is a tuple $< V, E >$, where $V = \{v_1, v_2, ..., v_n\}$ is the set of vertices with each v_i corresponds to x_i in P, $E = \{e_r =< v_i, v_j, l, c > |D_l \in D, x_i - x_j \leq w \in D_l, r = 1, 2, ..., \sum_{i=1}^{m} |D_i|\}$ is the set of directed edges leading from v_i to v_j. For an edge $e =< v_i, v_j, l, c > \in E$, l and c are usually called* ***label*** *and* ***weight*** *of e respectively.*

Example 1. *As running example, Figure 1 gives a DTP and its DTN.*

[1] As is customary in DTP literature temporal variables are assumed to be integer-valued (which can be easily extended to rational case) in this paper.

Compared to STN, an edge of DTN still corresponds to a simple temporal constraint; but in contrast, there is extra information attached to each edge (in addition to "weight") that labels the conjunction which the simple temporal constraint belongs to. For example, the edge $x_0 \xrightarrow{<0,1>} x_1$ corresponds to the constraint $x_0 - x_1 \leq 1$ in D_0. Mapped in this way, it can be asserted that there is a unique DTN **corresponding to** a given DTP, and vice versa. For convenience's sake, we take $label(e)$ and $weight(e)$ for short to indicate the conjunction which the edge "e" belongs to and its weight, and $source(e)$ and $dest(e)$ for e's starting and ending vertex, respectively.

2.2 Meta-CSP encoding of DTP

Naturally, a DTP ($< X, C = \{D_1, D_2, \ldots, D_n\} >$) is a CSP (called original-CSP): $< Vars, Doms, Cons >$, where $Vars(= X)$ is the set of temporal variables, $Doms$ is a set of domains of the variables in $Vars$, and $Cons$ a set of constraint (which is exactly the disjunctive constraints $\{D_1, D_2, \ldots, D_m\}$). A solution (called **exact solution** [13]) to original-CSP is assignments to the temporal variables that satisfy all of the disjunctive constraints. A DTP is said to be **consistent** if and only if there exists an exact solution for it. However, a DTP can also be another CSP (called meta-CSP): $< V, D, C >$, where V is a set of variables with each standing for a disjunction D_i, D is the set of domains for the variables in V with each the set of disjuncts in the disjunction ($D(v_i) = \{c_{i1}, c_{i2}, \ldots, c_{ik_i}\}$), and C is a singleton set of constraint that constrains the assignments of all variables in V form a consistent STP (such an STP is called **solution STP** [13]). Then the consistency decision of given DTP is cast to the problem of searching for any solution STPs for it.

As is customary, this paper aims at computing solution STPs, instead of exact solutions, for DTPs. Actually, given a solution STP, an exact solution can be achieved in polynomial time [9, 13].

Treating DTPs as meta-CSPs, Stergiou and Koubarakis designed a backtracking procedure that solves DTPs with the techniques of forward checking, backjumping, Minimal Remaining Values ordering heuristics [12]. Based on the

procedure BASIC-DTP
input: A–partial assignment, U–un-assigned variables
output: complete assignment, or failure

1. if $U = \oslash$ return(A);
2. $C = SelectVariable(U), U' = U - C$;
3. for each value c of domain(C)
4. $A' = A \cup \{C = c\}$;
5. if forward-check(A', U')
6. Basic-DTP(A', U');
7. un-forward(U');
8. return(failure);

forward-check(A, U)
9. for each variable C in U
10. for each value c in domain(C)
11. if not STP-consistency-check($A \cup \{C = c\}$)
12. remove c from domain(C);
13. if $domain(C) = \oslash$ return(failure);
14. return(true);

Figure 2. A basic CSP procedure for solving DTPs

similar CSP algorithm, Oddi and Cesta developed CSPi, a DTP solver that uses an incremental version of forward checking technique [9]. Based on the core concept of "no-good", Tsamardino integrated many CSP technique, such as forward checking, conflict-based backjumping, no-good recording, semantic branching , RSV and MRV [13], and developed another DTP solver–Epilitis. A basic CSP procedure with forward checking is illustrated in Figure 2 ([13]). In the Basic-DTP procedure, the function STP-consistency-check() is a function that checks the consistency of the current instantiation. Our implementation is maintaining a distance array (or called "distance graph") for current partial solution STP and testing whether the selected edge involves in any negative cycles with other edges within $O(1)$ time cost. For Example 1, Figure 3 illustrates a search tree of the examplified DTP under the Basic-DTP procedure (with each node denoting an assignment of a variable, solid lines showing the search progress and dashed lines witnessing backtracks to previous nodes when dead-ends are met).

3 Topology-based Variable Ordering Heuristics (TVO)

Problem topology usually helps to solve constraint problems. For general constraint network, Freuder proposed a sufficient condition identifying CSP subset that can be solved without backtracking [7]. Epstein and Wallace proposed an approach to identify "crucial subproblems" in CSPs by finding the structure of "near cliques" and "near clusters" in constraint network [6]. Xu and Choueiry designed a variable ordering heuristics for temporal constraint satisfaction problem (TCSP) which is based on problem topology [14, 4]. Although also studying constraint based temporal problem, Xu's approach can not be straightforward immigrated to the solution of DTPs because of the particularity of its topology, i.e. the disjuncts of a disjunction are scattered in the DTN instead of being attached to one common edge.

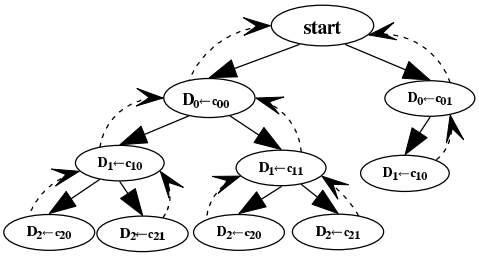


Figure 3. Search tree of Example 1 under Basic-DTP procedure

The idea behind TVO is relatively simple. Intuitively, the edge that most shortens the distance between its end points should be the one that most constrains the instantiations of pending variables. In other words, such an edge would be the one that will most probably involve in any negative cycles with values of pending variables. As a result, each pending variable can be estimated (notated as $est()$ function) by evaluating the extent (notated as $EXT()$ function) to which each of its value can reduce the distances on current distance graph. Therefore each time the variable with the maximal $est(x)$ value will be selected as the first to instantiate next. Formally, $EXT()$ can be defined to be:

$$EXT(e) = Dist(source(e), dest(e)) - weight(e) \quad (1)$$

where $Dist(v_i, v_j)$ represents the distance from vertex v_i to v_j on the distance graph. Figure 4 roughly depicts how TVO works. In Figure 4, the solid lines form current partial solution STP , the dashed lines are the edges of pending variables among which the edges with same styles belongs to the same variables (disjunctions), i.e. $label(e_1) = label(e_2)$ and $label(e_3) = label(e_4)$. The figure illustrates the fact that the variable D_i is selected to instantiate (e.g. with e_1 in the figure) before D_j because $est(D_i) > est(D_j)$.

Then, several definitions of est function can be considered:

H 1. *MAX estimation*

$$est(D_i) = max\{EXT(e)|e \in D_i\} \quad (2)$$

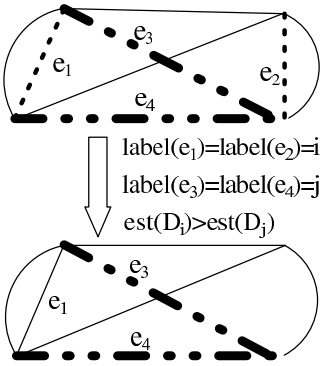


Figure 4. TVO heuristics

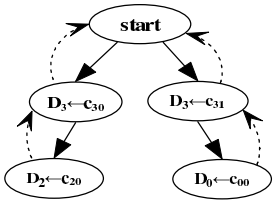


Figure 5. The search tree under H 2 and TVO_{INF} heuristics

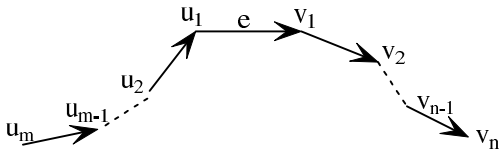


Figure 6. Factorization

H 1 represents the maximal extent to which the variable can reduce the distance graph. However, it neglects the contribution of all other values. In contrast, H 2 solves this problem:

H 2. *SUM estimation*

$$est(D_i) = \sum_{j=1}^{k} (EXT(e_j)|e_j \in D_i) \tag{3}$$

By its definition H 2 emphasizes the big-cardinality variables, which is opposite to usual knowledge (e.g. MRV heuristics prefers the variable with less values). As a compromise, H 3 improves it by averaging the SUM estimation:

H 3. *AVG estimation*

$$est(D_i) = \frac{\sum_{j=1}^{k} (EXT(e_j)|e_j \in D_i)}{k} \tag{4}$$

Generally speaking, H 3 would more accurately characterize the distance shortening ability of the variables. However, considering again that cardinality of variable is useful information for effective CSP/DTP solution, we have another heuristics:

H 4. *AVG* estimation*

$$est(D_i) = \frac{\sum_{j=1}^{k} (EXT(e_j)|e_j \in D_i)}{k^2} \tag{5}$$

3.1 Implementations of ∞

For any edge e with $Dist(source(e), dest(e)) = \infty$ (which means that there is no path between $source(e)$ and $dest(e)$ on current partial solution STP) yet, $EXT(e)$ can be evaluated as $-\infty$ (with the effect of $-\infty \pm x = -\infty$, annotated as TVO_{∞}) or $INF - weight(e)$ (annotated as TVO_{INF}) where INF is a pre-determined number (big enough to satisfy $n * INF > m * INF \pm x$ for $n > m$ and x is an integer that occurs in the calculation). The two alternatives will result in different instantiating behaviors. In contrast to TVO_{∞}, TVO_{INF} admits the potential difference among edges even when there are no path between the end points of them in current distance graph.

Take the DTP of Example 1 for instance, the search tree when using H 2 and TVO_{∞} is just the same as the one of Figure 3. But the search tree under H 2 and TVO_{INF} estimation would be the one in Figure 5. Just like the demonstration of Figure 5, the algorithm using TVO_{INF} usually achieve less node visits in search tree than that with TVO_{∞}.

3.2 Factorization

When evaluating a single edge of a variable, H 1–H 4 defined above only concern the distance decreasing between the end vertices of the edge. However, once an edge is inserted into current STP, more distances can be decreased. Figure 6 illustrates this observation. When the edge e (that will decrease the distance of u_1 and v_1) is selected to insert into the current STP, the distances between u_i and v_j will also be decreased.

To take account of the above observation, two arrays, Pre and $Succ$, are maintained along with the edge insertion into and deletion from partial STP. For any vertex v, $Pre[v]$ and $Succ[v]$ respectively record (non-duplicately) the number of predecessors and successors of v in current partial STP. Take the one in Figure 6 for instance, we have

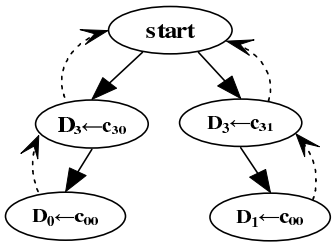


Figure 7. The search tree under H 2, TVO_{INF} and EXT_{fac}

$Pre[u_1] = m$ and $Succ[v_1] = n$. To evaluate the edge e, a factorized EXT function would be:

$$EXT_{fac}(e) = (Succ[souce(e)] + Pre[dest(e)]) * (Dist(source(e), dest(e)) - weight(e)) \quad (6)$$

Intuitively, EXT_{fac} highlights the variable whose edges (values) have more vertices that can reach (or be reached from) them. Such preference is reasonable, considering that more vertices that an edge can reach or be reached from implies its stronger power of constraining the future instantiation. For Example 1, the search tree under EXT_{fac} and TVO_{INF} is the one illustrated in Figure 7.

4 Experimental Results

In this section, some experimental results will be displayed to show the effectiveness of TVO heuristics.

4.1 Experiment Design

As is customary, the empirical evaluations are conducted on random DTPs which are generated by the random DTP generator implemented by Stergiou and Koubarakis [12]. The benchmark DTPs are instantiated according to the parameters $< k, n, m, L >$ where k is the number of disjuncts per constraint, n the number of temporal variables, m the number of disjunctive constraints, and L is a positive integer used to limit the constants in the DTP to be within $[-L, L]$. In the version of DTP generator provided by Claudio Castellini, k is limited to be 2, and, we examine the data sets with $L = 100$, which is also conventional. For each data set, we examine the number of consistency checks and CPU time cost (in seconds) of our algorithm on different data sets with the ratio $r(= \frac{m}{n})$ ranging from 1 to 14. For each r/n combination the generator produced 50 sample DTPs. And, we compute the median value of the interesting output values on the 50 sample DTPs as the statistical result. When the consistency checking exceeds 10000000, the algorithms stop running and exit. The algorithms are implemented in C, and the experiments are running on Red Hat Linux 9.0, which is installed on a VMware 4.5 virtual machine with allocated ram of 200Mb, and the master platform is Windows XP sp2 running on the machine with 512Mb ram and a clock speed of 1.8G Hz.

Figure 8 and Figure 9 illustrate all of the results. Figure 8(f) displays the ratio of consistent DTPs to total samples against the ratio r. Figure 8(a)-(e) confirm the existence of "critical range" of r ($r = 5, 6, 7, 8$) (where the problems are hard to solve) that is first pointed out in Stergiou and Koubarakis's paper [12]. In the illustrations, "meta-CSP" represents a plain meta-CSP DTP solver which exploits the techniques of incremental forward checking, semantic branching and removal of subsumed variable. "hx" stands for the algorithm integrating meta-CSP procedure with Hx heuristics. Moreover, "inf" and "fac" respectively correspond to the implementation of TVO_{INF} and EXT_{fac}. The other curves that are not notated as "inf" are implemented as TVO_{∞} by default. From Figure 8 and Figure 9 several conclusions can be drawn:

1. Different heuristics of TVO can improve meta-CSP procedure by one to four order-of-magnitude. Moreover, as expected, the number of consistency checks decreases when using the heuristics from H1 to H4. However, the curves are interleaved when factorization and different implementation of ∞ are considered. For instance, H2_inf_fac outperforms all implementations of H3 on $n \in \{15, 20, 25, 30\}$.

2. As the case shown in Figure 5 TVO_{INF} usually performs better than TVO_{∞}, though H1_inf is outperformed by H1 on some values of r. On the other hand, EXT_{fac} always improves EXT on the experimented data set.

3. The best of the examined 12 heuristics, H4_inf_fac, wins MRV on the hardest region (r=5, 6, 7, 8) almost on all examined values of n (except for $n = 10$), and, the superiority appears more obvious when n gets bigger. For the case n=30, H4_inf_fac always outperforms MRV for $r > 4$, and, especially, it wins MRV by one order of magnitude on the hardest problems (for instance, H4_inf_fac does 55148 consistency checks, which is much lower than MRV's 659943). Figure 9(a) and (b) show that the superiority also lies in the CPU time cost, **which suggests that the complexity of our ordering strategy is minor to overall solution complexity**. The figures reveal the fact that H4_inf_fac is superior to MRV, particularly on hard and large-scale problems. In addition, compared to MRV, H4_inf_fac demonstrates better scalability.

4. Compared to the fastest DTP solver reported in the DTP literature, TSAT++ v0.5[2], the algorithm with

[2]It can be found via: http://www.star.dist.unige.it/ marco/Tsat/default.htm

H4_inf_fac also demonstrates competitive (faster for most of problems) solution speed.

5 Conclusions and future work

In this paper, we have examined the problem of designing new dynamic variable ordering (DVO) heuristics for solving Disjunctive Temporal Problems (DTPs). The heuristics (TVO for short) is based on the evaluation of the potentiality of a variable's involvement in some conflicts with other variables, which is again based on the estimation of each value's (or edge's) possibility of participating in any negative cycles in the problem topology that is defined to be Disjunctive Temporal Network (DTN). The experimental results show that the proposed heuristics can effectively reduce the consistency checks in solving DTPs, which results in higher performance in deciding the consistency of given DTPs. Meanwhile, the experiment figures reveal that our approach has a good trade-off between ordering complexity and overall performance. Moreover, compared with MRV, TVO achieves less consistency checking, as well as time cost, on most of, especially for the hard and large-scale, problems (where TVO can achieve one order-of-magnitude saving of consistency checking over MRV). Compared to TSAT++, the CSP based algorithm with H4_inf_fac heuristics also demonstrates competitive solution speed.

As for the future work, more evaluation of TVO on other hand-made or practical problems would be made. And, comparison between TVO with other DVOs is helpful to further evaluate the particular advantage of TVO. Moreover, how to extend the approach to other constraint based temporal formulation or arithmetic logic is another valuable topic.

References

[1] A. Alessandro, C. Castellini, E. Giunchiglia, M. Idini, and M. Maratea. Tsat++: an open platform for satisfiability modulo theories. In *Proceedings of PDPAR 2004*, 2004.

[2] A. Armando, C. Castellini, and E. Giunchiglia. Sat-based procedures for temporal reasoning. In *Proceedings of the 5th European Conference on Planning (Durham, UK)*, 1999.

[3] F. Bacchus and P. van Run. Dynamic variable ordering in csps. In *Proceedings First International Conference on Constraint Programming*, pages 258–275. Springer-Verlag, 1995.

[4] B. Y. Choueiry and L. Xu. An efficient consistency algorithm for the temporal constraint satisfaction problem. *AI Commun.*, 17(4):213–221, 2004.

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

[6] S. L. Epstein and R. J. Wallace. Finding crucial subproblems to focus global search. In *ICTAI '06: Proceedings of the 18th IEEE International Conference on Tools with Artificial Intelligence*, pages 151–162, 2006.

[7] E. C. Freuder. A sufficient condition for backtrack-free search. *J. ACM*, 29(1):24–32, 1982.

[8] V. Kumar. Algorithms for constraint-satisfaction problems: A survey. *AI Magazine*, 13(1):32–44, 1992.

[9] A. Oddi and A. Cesta. Incremental forward checking for the disjunctive temporal problem. In *Proceedings of ECAI 2000*, pages 108–112, 2000.

[10] P. J. Schwartz and M. E. Pollack. Planning with disjunctive temporal constraints. In *Proceedings of ICAPS Worskhop on Integrating Planning into Scheduling*, 2004.

[11] P. J. Schwartz and M. E. Pollack. Two approaches to semi-dynamic disjunctive temporal problems. In *Proceedings of CAPS Workshop on Constraint Programming for Planning and Scheduling*, 2005.

[12] K. Stergiou and M. Koubarakis. Backtracking algorithms for disjunctions of temporal constraints. In *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98)*, pages 248–253, 1998.

[13] I. Tsamardinos and M. E. Pollack. Efficient solution techniques for disjunctive temporal reasoning problems. *Artificial Intelligence*, 151(1-2):43–89, 2003.

[14] L. Xu and B. Y. Choueiry. A new efficient algorithm for solving the simple temporal problem. In *Proceedings of TIME2003*, pages 212–222, 2003.

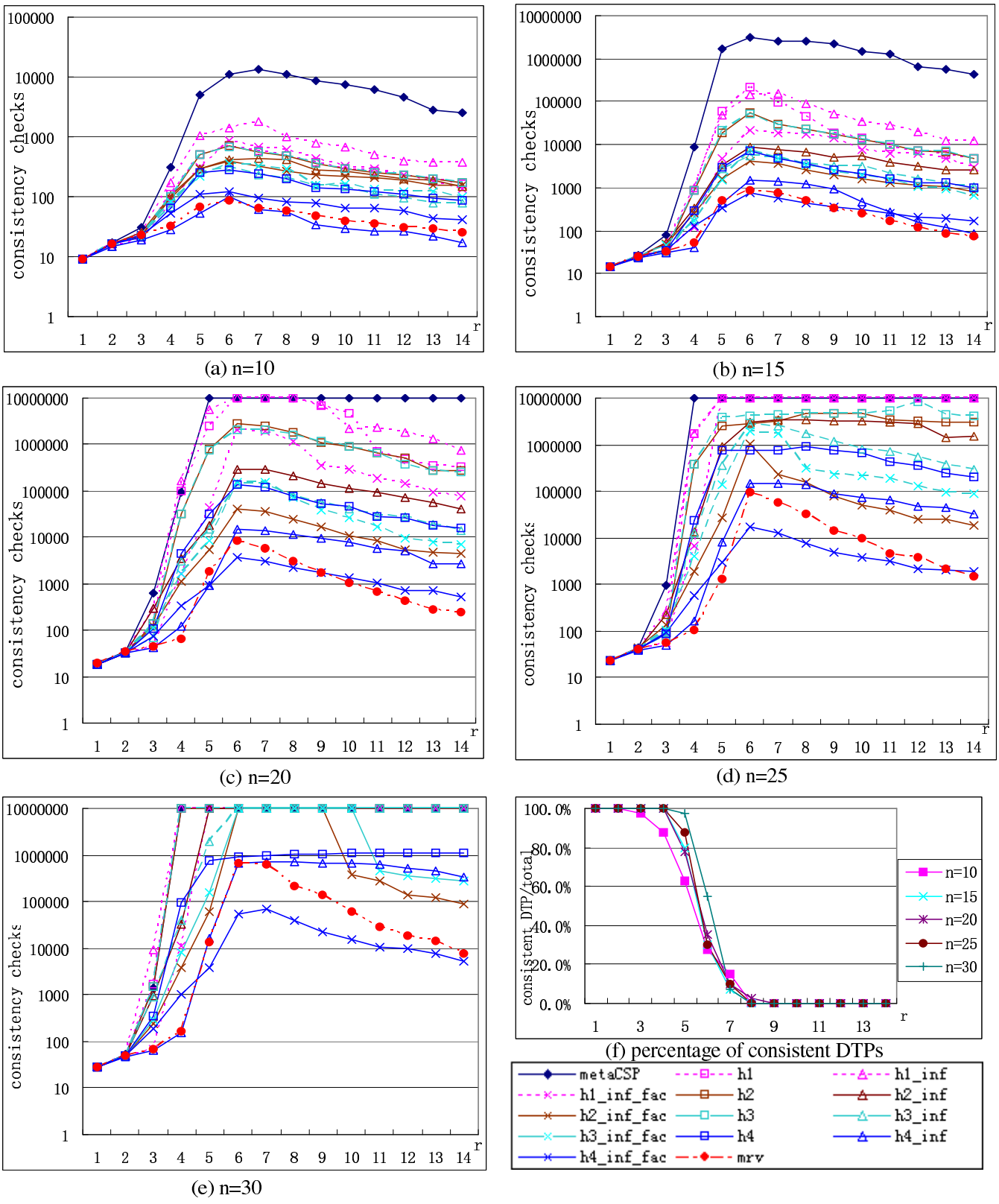
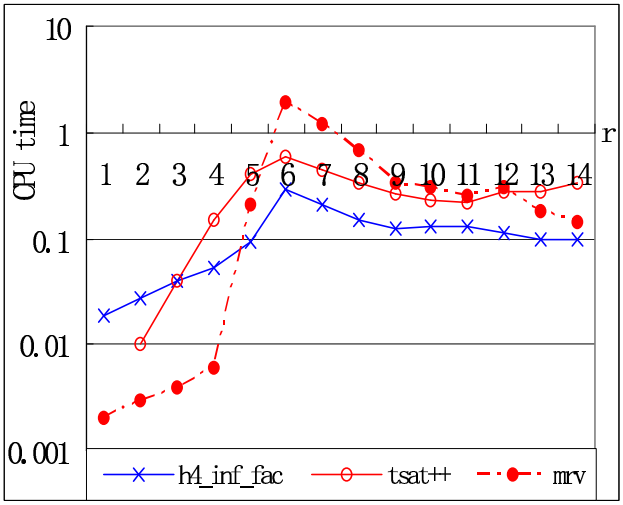
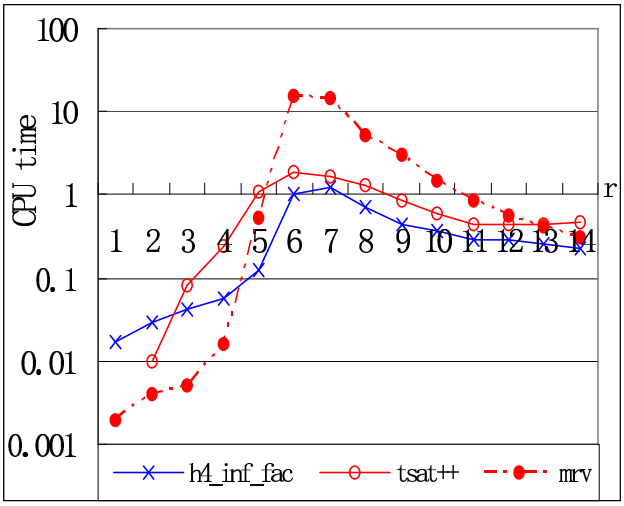


Figure 8. Comparison on consistency checks



(a) n=25



(b) n=30

Figure 9. Comparison on CPU time cost