Solving Hard Qualitative Temporal Reasoning Problems: Evaluating the Efficiency of Using the ORD-Horn Class

Bernhard Nebel
Universität Ulm
Fakultät für Informatik
D-89069 Ulm, Germany
e-mail: nebel@ki.informatik.uni-ulm.de

February 22, 1996

Abstract

While the worst-case computational properties of Allen's calculus for qualitative temporal reasoning have been analyzed quite extensively, the determination of the empirical efficiency of algorithms for solving the consistency problem in this calculus has received only little research attention. In this paper, we will demonstrate that using the ORD-Horn class in Ladkin and Reinefeld's backtracking algorithm leads to performance improvements when deciding consistency of hard instances in Allen's calculus. For this purpose, we prove that Ladkin and Reinefeld's algorithm is complete when using the ORD-Horn class, we identify phase transition regions of the reasoning problem, and compare the improvements of ORD-Horn with other heuristic methods when applied to instances in the phase transition region. Finally, we give evidence that combining search methods orthogonally can dramatically improve the performance of the backtracking algorithm.

Contents

1 Introduction

Representation of qualitative temporal information and reasoning with it is an integral part of many artificial intelligence tasks, such as general planning [Allen, 1991], presentation planning in a multi-media context [Feiner *et al.*, 1993], natural language understanding [Song and Cohen, 1988], and diagnosis of technical systems [Nökel, 1991]. *Allen's* [1983] *interval calculus* is well suited for representing qualitative temporal relationships and reasoning with it. In fact, it is used in all the applications mentioned above.

While the worst-case computational properties of Allen's calculus and fragments of it have been quite extensively analyzed [Golumbic and Shamir, 1993; Ladkin and Maddux, 1994; Nebel and Bürckert, 1995; van Beek and Cohen, 1990; Vilain and Kautz, 1986], design and empirical evaluation of reasoning algorithms for Allen's calculus has received much less research attention. In this paper, we address the latter problem and analyze in how far using the *ORD-Horn subclass* [Nebel and Bürckert, 1995] of Allen's relations can improve the efficiency of existing reasoning algorithms. As it turns out, the ORD-Horn class can significantly enhance the performance in search-intensive cases.[1]

Since reasoning in the full calculus is NP-hard [Vilain and Kautz, 1986], it is necessary to employ some sort of *exhaustive search method* if one wants complete reasoning in the full calculus. Allen [1983] proposed in his original paper to search through all possible "atomic" temporal constraint networks that result from instantiating disjunctive relations to one disjunct and to test for consistency using the *path-consistency algorithm* [Montanari, 1974] that is incomplete for the full calculus, but complete for atomic relations.

A more efficient algorithm has been proposed by Ladkin and Reinefeld [1992]. This algorithm uses path-consistency as a *forward checking* technique [Haralick and Elliot, 1980] during the backtrack search, which results in pruning the search tree significantly. As pointed out by Ladkin and Reinefeld [1992], this algorithm allows to instantiate disjunctive relations not only by atomic relations but by any set of relations the path-consistency method is complete for, which can considerably reduce the branching factor in the backtrack search. However, if non-atomic relations are used, it is not any longer obvious that the backtracking algorithm is a complete reasoning method. As we show in Section 3, however, Ladkin and Reinefeld's suggestion is indeed correct.

Since the ORD-Horn subclass of the qualitative relations in Allen's calculus is the unique maximal set containing all atomic relations such that path-consistency is sufficient for consistency [Nebel and Bürckert, 1995], it would seem that employing this set in the backtracking algorithm is clearly advantageous over using other subclasses. However, the experiments that have been performed so far do not seem to justify this conjecture. Ladkin and Reinefeld

[1] The C-programs that were used for the evaluation are available from the author.

[1992; 1993] concluded from the experiments they performed that "in practice one can expect the number of path-consistency computation almost constant," i.e., in practice there won't be much search. Van Beek and Manchak [1996], who further developed Ladkin and Reinefeld's backtracking algorithm, were able to generate problem instances that led to significant search. However, they did not observe that using the ORD-Horn subclass led to an performance improvement over using the smaller *pointizable subclass* [Ladkin and Maddux, 1994; van Beek and Cohen, 1990].

It may be the case, however, that Ladkin and Reinefeld [1992; 1993] missed generating hard instances and that van Beek and Manchak [1996] did not look for the right performance indicators. In Section 5, we identify the *phase transition region* [Cheeseman *et al.*, 1991] for reasoning in Allen's calculus, which contains arbitrarily hard instances. We use these problems to evaluate the usage of the ORD-Horn class in Section 6 and demonstrate its advantage. Further, we demonstrate in Section 7 that combining the ORD-Horn subclass with other search strategies in an orthogonal way can dramatically improve the performance on van Beek and Manchak's [1996] hard problem instances.

2 Allen's Calculus

Allen's [1983] approach to reasoning about time is based on the notion of *time intervals* and *binary relations* on them. A *time interval* X is an ordered pair (X^-, X^+) such that $X^- < X^+$, where X^- and X^+ are interpreted as points on the real line. Given two concrete time intervals, their relative positions can be described by *exactly one* of the elements of the set $\mathbf{A}$ of thirteen *atomic interval relations*. Atomic relations are, for example, $\equiv$, $\prec$, $\succ$, and d, meaning that the first interval equals, is before, is after, or is strictly inside the second interval, respectively. These interval relations can be defined in terms of their *interval endpoint relations*, e.g., $X \mathsf{d} Y$ can be defined by $X^- > Y^- \wedge X^+ < Y^+$ (see Table 1).

In order to express indefinite information, unions of the atomic interval relations are used, which are written as sets of atomic relations. The formula $X\{\equiv, \mathsf{d}\}Y$ means, e.g., that X equals Y or is inside Y. Since there are 13 atomic relations, there are 2^{13} possible unions of atomic relations, which form the set of binary *interval relations* (denoted by r)—including the *empty relation* $\emptyset$ and the *universal relation* $\mathbf{A}$. The set of all binary interval relations $2^{\mathbf{A}}$ is denoted by $\mathcal{A}$. On this set, we can define the operations *intersection* $(r \cap r')$, relational *converse* $(r^{\smile})$, and relational *composition* $(r \circ r')$:

$$\begin{array}{lrcl}
\forall X, Y\colon & X r^{\smile} Y & \leftrightarrow & Y r X \\
\forall X, Y\colon & X\,(r \cap r')\,Y & \leftrightarrow & X r Y \wedge X r' Y \\
\forall X, Y\colon & X\,(r \circ r')\,Y & \leftrightarrow & \exists Z\colon (X r Z \wedge Z r' Y).
\end{array}$$

Basic Interval Relation	Symbol	Pictorial Example	Endpoint Relations
X before Y	$\prec$	xxx	$X^- < Y^-$, $X^- < Y^+$,
Y after X	$\succ$	yyy	$X^+ < Y^-$, $X^+ < Y^+$
X meets Y	m	xxxx	$X^- < Y^-$, $X^- < Y^+$,
Y met-by X	$\mathsf{m}^\smile$	yyyy	$X^+ = Y^-$, $X^+ < Y^+$
X overlaps Y	o	xxxx	$X^- < Y^-$, $X^- < Y^+$,
Y overlapped-by X	$\mathsf{o}^\smile$	yyyy	$X^+ > Y^-$, $X^+ < Y^+$
X during Y	d	xxx	$X^- > Y^-$, $X^- < Y^+$,
Y includes X	$\mathsf{d}^\smile$	yyyyyyy	$X^+ > Y^-$, $X^+ < Y^+$
X starts Y	s	xxx	$X^- = Y^-$, $X^- < Y^+$,
Y started-by X	$\mathsf{s}^\smile$	yyyyyyy	$X^+ > Y^-$, $X^+ < Y^+$
X finishes Y	f	xxx	$X^- > Y^-$, $X^- < Y^+$,
Y finished-by X	$\mathsf{f}^\smile$	yyyyyyy	$X^+ > Y^-$, $X^+ = Y^+$
X equals Y	$\equiv$	xxxx	$X^- = Y^-$, $X^- < Y^+$,
		yyyy	$X^+ > Y^-$, $X^+ = Y^+$

Table 1: The set **A** of the thirteen atomic relations. The endpoint relations $X^- < X^+$ and $Y^- < Y^+$ that are valid for all relations have been omitted.

Together with these operations, $\mathcal{A}$ forms an algebra,[2] which is called *Allen's interval algebra*.

A *qualitative description of an interval configuration* is usually given as a set of formulae of the above form, or, equivalently, as a *temporal constraint graph* with nodes as intervals and arcs labeled with interval relations—the *constraints*. These graphs are often represented as matrices of size $n \times n$ for n intervals, where $M_{ij} \in \mathcal{A}$ is the constraint between the ith and jth interval. Usually it is assumed (without loss of generality) that $M_{ii} = \{\equiv\}$ and $M_{ji} = M_{ij}{}^\smile$.

The *fundamental reasoning problem* in this framework is to decide whether a given qualitative description of an interval configuration is satisfiable, i.e., whether there exists an *assignment* of real numbers to all interval endpoints, such that all constraints in the corresponding constraint graph are satisfied. This problem, called ISAT, is fundamental because all other interesting reasoning problems polynomially reduce to it [Golumbic and Shamir, 1993] and because it is one of the most important tasks in practical applications [van Beek and Manchak, 1996].

The most often used method to determine satisfiability of a temporal constraint graph is the *path-consistency method*,[3] which was already proposed by

[2]Note that we obtain a relation algebra if we add *complement* and *union* as operations [Ladkin and Maddux, 1994].

[3]An alternative method for a subset of Allen's interval algebra has been developed by Gerevini and Schubert [1993].

Allen [1983]. Essentially, it consists of computing repeatedly

$$M_{ij} \leftarrow M_{ij} \cap (M_{ik} \circ M_{kj}) \tag{1}$$

for all i, j, k until no more changes occur. Obviously, the restriction on M_{ij} does not remove any *possible assignment*, but only deletes atomic relations that are not satisfiable in any way. This method—if implemented in a sophisticated way—runs in $O(n^3)$ time, where n is the number of intervals. In the following, a matrix that has been "reduced" in this way is called *path-consistent* and is denoted by $\widehat{M}$.

If $\widehat{M}_{ij} = \emptyset$ for some i, j, then it follows obviously that M is not satisfiable. The converse implication is not valid, however, as Allen [1983] already demonstrated using an example attributed to H. Kautz. Since ISAT is NP-complete [Vilain and Kautz, 1986], it is very unlikely that any polynomial algorithm can solve ISAT. However, there exist subsets of $\mathcal{A}$ such that ISAT is a polynomial problem if only relations from these subsets are used. These subsets are the *continuous endpoint class* $\mathcal{C}$ [Nökel, 1991; van Beek and Cohen, 1990], the *pointizable class* $\mathcal{P}$ [Ladkin and Maddux, 1994; van Beek and Cohen, 1990], and the *ORD-Horn class* $\mathcal{H}$ [Nebel and Bürckert, 1995], which form a strict hierarchy. Interestingly, these classes lead also to completeness of the path-consistency method.

3 The Backtracking Algorithm

If an application needs more expressiveness than is granted by the above mentioned subclasses and if complete reasoning is required, then some sort of backtracking search is necessary. The backtracking algorithm shown in Figure 1, which has been proposed by Ladkin and Reinefeld [1992], appears to be the most efficient version of such an algorithm.

The procedure "path-consistency" transforms a matrix C to $\widehat{C}$. The set *Split* is a subset of $\mathcal{A}$ such that path-consistency is complete for ISAT. The algorithm deviates slightly from the one published in [Ladkin and Reinefeld, 1992] in that it makes the choice of the constraint to be processed next nondeterministic, but is otherwise identical.

When the algorithm is implemented, a number of design choices are necessary that can influence the practical efficiency considerably [van Beek and Manchak, 1996]. Some of these choices will be discussed in Section 6 below. The choice of what subset of $\mathcal{A}$ to use for the set *Split* seems obvious, however, namely, the largest such set, which is the ORD-Horn class [Nebel and Bürckert, 1995]. This subclass covers 10% of Allen's interval algebra (compared with 1% for $\mathcal{C}$ and 2% for $\mathcal{P}$), and for this reason the ORD-Horn class should reduce the branching factor in the backtrack search much more than any other class. Unfortunately, the reduction is less dramatic than the previous figures suggest. Based on the assumption that the interval relations are uniformly distributed, a straightforward

Input: Matrix C representing a temporal constraint graph
Result: true iff C is satisfiable

function consistent(C)
  path-consistency(C)
  if C *contains empty relation*
    then return false
    else
      choose an unprocessed label C_{ij} *and split* C_{ij}
        into $r_1, \ldots, r_k$ *s.t. all* $r_l \in \mathrm{Split}$
      if *no label can be split* **then return** true
      endif
      for all labels r_l $(1 \leq l \leq k)$ **do**
        $C_{ij} \leftarrow r_l$
        if consistent(C) **then return** true
        endif
      endfor
      return false
  endif
endfunction

Figure 1: Backtracking algorithm

computer-based analysis gives the following average branching factors:[4] **A** 6.5, $\mathcal{C}$ 3.551, $\mathcal{P}$ 2.955,[5] $\mathcal{H}$ 2.533.

The main problem with the algorithm is, however, that it is not obvious that it is complete if *Split* differs from the set of atomic relations. In this case, it is possible that during the backtrack search a constraint M_{ij} that has been restricted to a relation from the set *Split* is further constrained by the path-consistency procedure to a relation that is not in *Split*. Hence, it is not obvious that all constraints belong to the class *Split* for which path-consistency is complete when the recursive function terminates, which may lead to incompleteness.

In order to show that the above backtracking algorithm is nevertheless complete, we need first some definitions. We write $M \leq N$ iff $M_{ij} \subseteq N_{ij}$ for all i, j. Further we denote by $M[i, j/r]$ the matrix that is identical to M except that $M[i, j/r]_{ij} = r$. The following lemma is straightforward [Montanari, 1974].

[4] As noted by Ladkin and Reinefeld [1993], this is a worst-case measure, because the interleaved path-consistency computations reduce the branching factor considerably.

[5] This number deviates from [Ladkin and Reinefeld, 1993] but has been confirmed by Peter van Beek in personal communication.

Lemma 1 *$\widehat{M} \leq M$, $\widehat{\widehat{M}} = \widehat{M}$, and if $M \leq N$ then $\widehat{M} \leq \widehat{N}$.*

Now let σ_k denote the k-th choice of the backtracking algorithm, i.e. the choice of the pair (i,j) and the selected relation r_l. Then $M[\sigma_k]$ denotes the replacement of the constraint M_{ij} by r_l. Assuming that C denotes the original temporal constraint graph, we define the following sequences of matrices:

$$
\begin{aligned}
C^0 &= C && (2)\\
C^k &= \widehat{C^{k-1}}[\sigma_k] && (3)\\
S^0 &= C && (4)\\
S^k &= S^{k-1}[\sigma_k] && (5)
\end{aligned}
$$

In other words, C^k corresponds to the matrix C after the kth choice in the backtracking algorithm and S^k reflects the first k choices without having applied path-consistency.

Lemma 2 *$\widehat{C^k} = \widehat{S^k}$, for all k.*

Proof. $\leq$: We prove $C^k \leq S^k$ by induction, from which $\widehat{C^k} \leq \widehat{S^k}$ follows by Lemma 1. The hypothesis holds for $k = 0$ by definition. Assume that it holds for k. From that it follows by Lemma 1 that $\widehat{C^k} \leq S^k$ and $\widehat{C^k}[\sigma_{k+1}] \leq S^k[\sigma_{k+1}]$, since the $k+1$th choice is always a subset of the corresponding relation in $\widehat{C^k}$. By applying the definition of C and S, we get $C^{k+1} \leq S^{k+1}$, as desired.

$\geq$: We prove $\widehat{C^k} \geq \widehat{S^k}$ by induction. The hypothesis holds for $k = 0$ by definition and Lemma 1. Assuming that it holds for k, it follows that $\widehat{C^k}[\sigma_{k+1}] \geq \widehat{S^k}[\sigma_{k+1}]$ (*). Since $S^k \geq S^k[\sigma_{k+1}]$, we have $\widehat{S^k} \geq S^k\widehat{[\sigma_{k+1}]}$. Let σ_{k+1} be r_l at (i,j). Clearly, $S^k\widehat{[\sigma_{k+1}]}_{ij} \subseteq r_l$. Hence, also $\widehat{S^k}[\sigma_{k+1}] \geq S^k\widehat{[\sigma_{k+1}]}$. From that and (*) it follows that $C^{k+1} \geq \widehat{S^{k+1}}$, from which the the claim follows by applying Lemma 1 twice. ■

In other words, if the recursive function terminates, the temporal constraint graph is equivalent to one which results from applying all choices (which select constraints from *Split*) and using path-consistency in the end. Since soundness is obvious and completeness follows from Lemma 2, the backtracking algorithm described above is indeed sound and complete.

Theorem 3 *The backtracking algorithm is sound and complete if the set* Split *is a subclass of Allen's interval algebra such that the path-consistency algorithm is complete.*

4 Test Instances and Measurement Methods

In order to test empirically the usefulness of employing the ORD-Horn class in the backtracking algorithm, some set of test instances is necessary. Ideally, a set of "benchmark" instances that are representative of problem instances that appear in practice should be used. However, such a collection of large benchmark problems does not exist for qualitative temporal reasoning problems [van Beek and Manchak, 1996]. The DNA sequencing instance from molecular biology [Benzer, 1959]that has been suggested by van Beek and Manchak [1996] is unfortunately not adequate for our purposes because the structure of constraints leads to identical results for $\mathcal{P}$ and $\mathcal{H}$ [van Beek and Manchak, 1996].

For these reasons, the only possibility to evaluate the usefulness of the ORD-Horn class is to randomly generate temporal constraint networks as in [Ladkin and Reinefeld, 1992; Ladkin and Reinefeld, 1993; van Beek and Manchak, 1996]. We use three models to generate constraint networks, denoted by $A(n,d,s)$, $H(n,d)$, and $S(n,d,s)$.

For $A(n,d,s)$, random instances are generated as follows:

1. A graph with n nodes and an average degree of d for each node is generated. This is accomplished by selecting $nd/2$ out of the $n(n-1)/2$ possible edges using a uniform distribution.

2. If there is no edge between the ith and jth node, we set $M_{ij} = M_{ji} = \mathbf{A}$.

3. Otherwise a non-null constraint is selected according to the parameter s, such that the average size of all non-universal constraints is s. This is accomplished by selecting one of the atomic relations with uniform distribution and out of the remaining 12 relations each one with probability $(s-1)/13$.

For $H(n,d)$, the random instances are generated as in steps 1–2 above, but in step 3, we select a constraint from a particular set of 3006 probably very hard constraints with a uniform distribution. The conjecture that these constraints are hard is based on the fact that their translation to a *logical form* requires clauses with at least three literals and the observation that the path-consistency algorithm is similar to positive unit-resolution on the *logical form*.[6] As our experiments demonstrate, these constraints lead indeed to hard reasoning problems.

Finally, for $S(n,d,s)$, the random instances are generated as in $A(n,d,s)$, but in a post-processing step the instances are made satisfiable by adding atomic relations that result from the description of a randomly generated scenario, i.e., these instances are always satisfiable. This model was proposed by van Beek and Manchak [1996], and they reported that a large fraction of instances generated by

[6]See [Nebel and Bürckert, 1995] for a precise definition of logical form of a temporal constraint and for the similarity between path-consistency and positive unit resolution.

$S(100, 25, 6.5)$ are very hard, sometimes requiring more than half a day of CPU time on a Sun 4/20.

Using these random models, we analyze the effect of varying the parameters and evaluate the runtime efficiency of different implementations of the backtracking algorithm. As the performance indicator we use CPU time on a SparcStation 20. Although this indicator is more dependent on the particular implementation and platform than indicators such as the number of compositions performed or the number of search nodes explored, it gives a more realistic picture of the effect of applying different search techniques.

5 Phase Transitions for Reasoning in Allen's Calculus

Cheeseman *et al* [1991] conjectured:

> All NP-complete problems have at least one order parameter and the hard to solve problems are around a critical value of this order parameter. This critical value (a *phase transition*) separates one region from another, such as overconstrained and underconstrained regions of the problem space.

Instances in the phase transition are obviously particularly well suited for testing algorithms on search intensive instances.

Ladkin and Reinefeld [1993] observed that reasoning in Allen's calculus has a phase transition in the range $6 \leq c \times n \leq 15$ for $c \geq 0.5$, where c is the ratio of non-universal constraints to all possible constraints and n is the number of intervals. This phase transition is, however, not independent of the instance size, and for this reason does not allow to generate arbitrarily hard instances.

Our conjecture was that the *average degree* of the constraint graph is a critical order parameter that can lead to a size-independent phase-transition. As Figure 2 demonstrates,[7] this is indeed the case for $A(n, d, 6.5)$.

The probability that the instance is satisfiable drops from 1 to 0 around $d = 9.5$. As expected, the typical instances around the phase transition are hard, meaning that the median value of CPU time has a peak in the phase transition region, as shown in Figure 3 (the solid line marks the phase transition). Further, the mean value has a peak there as well, as also shown in Figure 3.

For other values of the average label size s, we get qualitatively similar results, as Figure 4 shows for $s = 7.0$. The general picture that emerged from varying s from 5.0 to 8.0 was that with larger values of s the phase transition region moves to higher values of d and the runtime requirements grow.

[7] Each data point in this and the following graphs is based on 500 randomly generated instances.

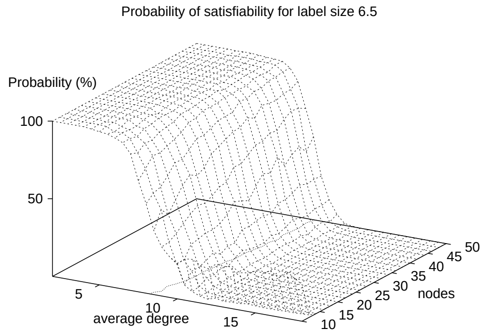

Figure 2: Probability of satisfiability for $A(n, d, 6.5)$

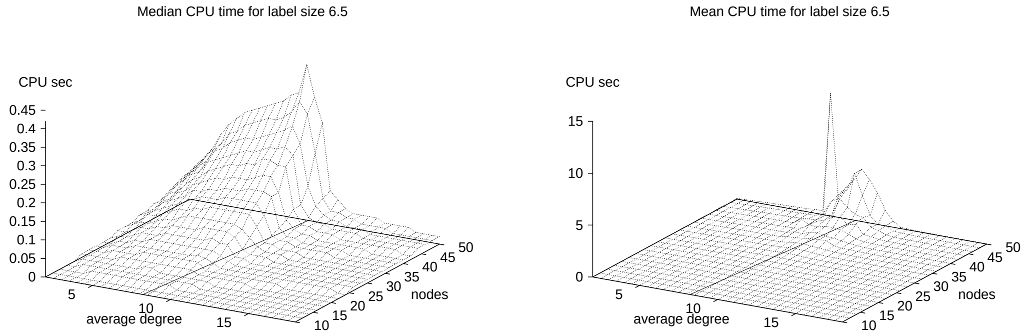

Figure 3: CPU time for $A(n, d, 6.5)$

These results on phase transitions provide us with test cases on which we can evaluate different backtracking methods. However, the predictive value of the results is, of course, limited. The average label size together with the average degree gives indications of whether instances may be hard or easy to solve. However, the particular distribution of constraints is much more important than

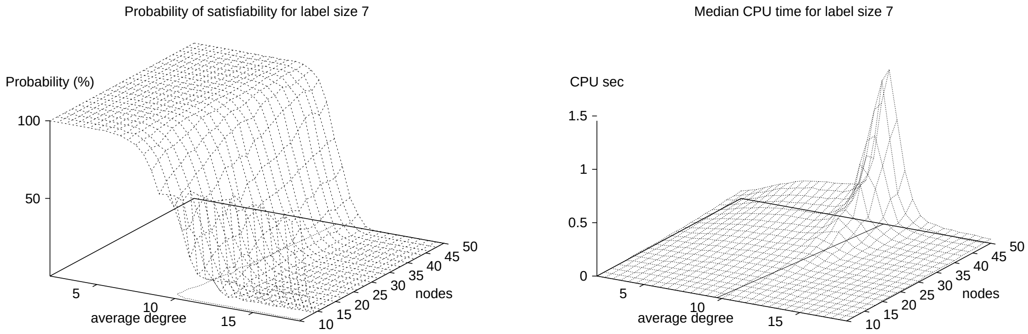

Figure 4: Probability of Satisfiability and Median CPU time for $A(n, d, 7.0)$

the average label size. If we use, for instance, a uniform distribution over the ORD-Horn relations—which results in an average label size of 6.83—no runtime peak is observable in the phase transition region. Using the "hard relations" from $H(n, d)$—resulting in an average label size of 6.97—one would expect significant more search and perhaps a move of the phase transition compared with $A(n, d, 7.0)$. This expectation is confirmed by our experiments, as shown in Figure 5.

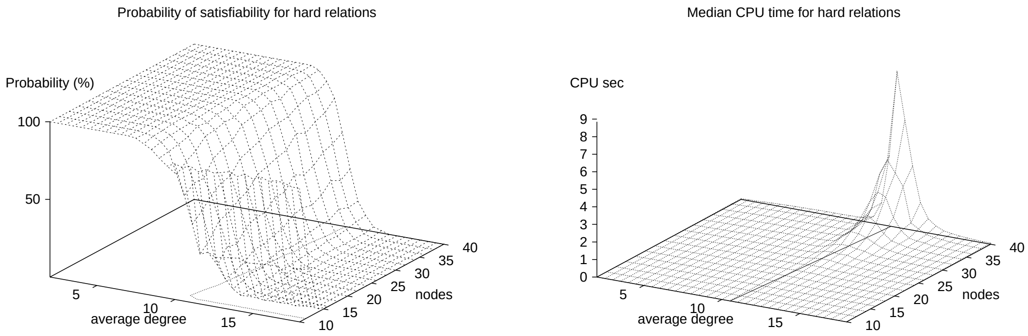

Figure 5: Probability of Satisfiability and Median CPU time for $H(n, d)$

6 Using the ORD-Horn Class

Comparing the backtracking algorithm for $Split = \mathcal{H}$ and $Split = \mathcal{P}$ in the phase transition region shows that the ORD-Horn class provides a significant performance enhancement in some cases (Figure 6).

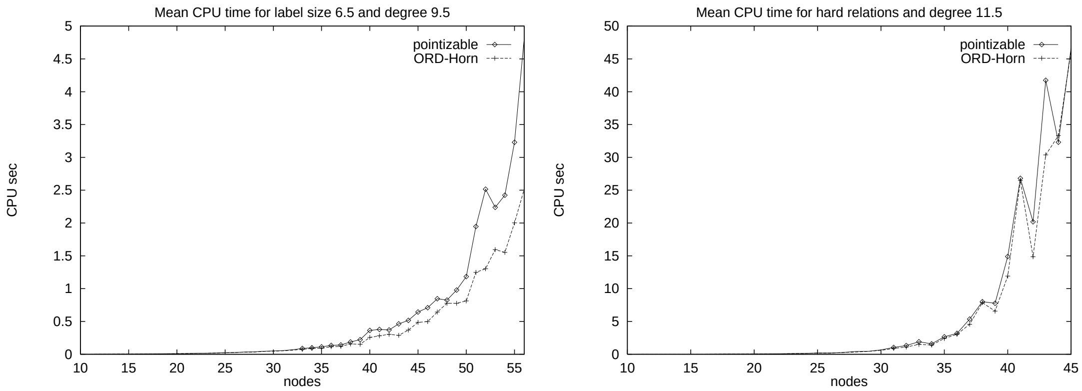

Figure 6: Comparison between $\mathcal{P}$ and $\mathcal{H}$ for $A(n, 9.5, 6.5)$ and $H(n, 11.5)$

This means that contrary to previous observations, it can pay off for search intensive cases to use the ORD-Horn subclass instead of the pointizable subclass.

The difference between $A(n, 9.5, 6.5)$ and $H(n, 11.5)$ is probably explainable by the fact that the distribution of labels in the two different random models lead to a reduction of the branching factor of 15.3% in the former case and 9.3% in the latter case when going from the pointizable to the ORD-Horn class.

One question might be, however, where the performance enhancements came from. As Figure 7 shows, the median CPU time value is almost identical for using $\mathcal{H}$ and $\mathcal{P}$ and the main differences appear for the very hard instances. For this reason, the main value of using the ORD-Horn subclass seems to be that it reduces the runtime of extreme cases.

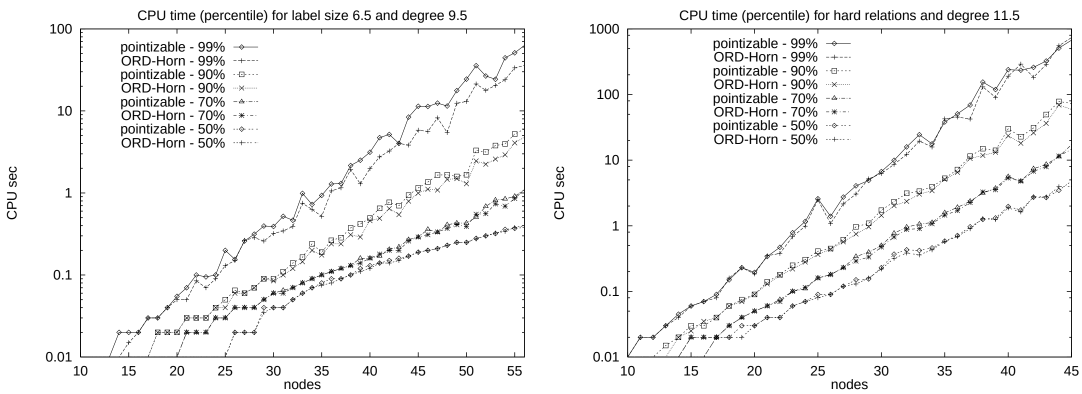

Figure 7: Comparison by percentiles

The results described above were achieved by using all techniques described in [van Beek and Manchak, 1996] and varying only the set *Split*. So the question arises how changing the set *Split* in our backtracking algorithm compares to other design decisions. We varied the following design decisions in order to answer this question:

ORD-Horn/pointizable: The subclass used for the set *Split*.

static/dynamic: Constraints are processed according to a heuristic evaluation of their constrainedness which is determined *statically* before the backtracking starts or *dynamically* during the search.

local/global: The evaluation of the constrainedness is based on a *local* heuristic weight criterion or on a *global* heuristic criterion [van Beek and Manchak, 1996].

queue/no queue: The path-consistency procedure uses a weighted *queue* scheme for the constraints to be processed next [van Beek and Manchak, 1996] or the scheme described in [Ladkin and Reinefeld, 1992], which uses *no queue*.

As it turns out, the improvement of using $\mathcal{H}$ instead of $\mathcal{P}$ is small compared with the improvements achievable by other means (Figure 8).

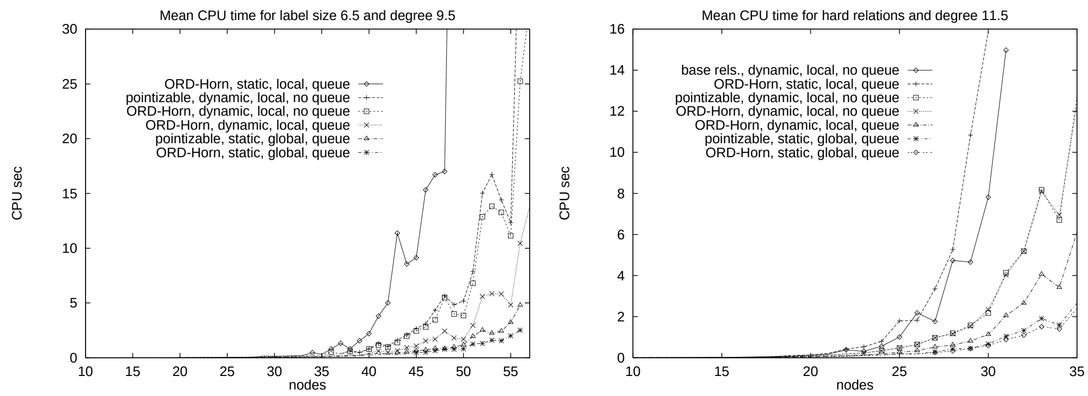

Figure 8: Using ORD-Horn and other design choices

The two lower curves in the graphs shown in Figure 8 correspond to the curves in Figures 6. The results show that node ordering and the heuristic evaluation of constrainedness can have a much more significant effect on the efficiency than the choice of the tractable subset used for the set *Split* in the algorithm.

7 The Power of Orthogonally Combined Strategies

Van Beek and Manchak [1996] used $S(n,d,s)$-instances for evaluating different strategies. They noted that in particular $S(100,25,6.5)$ leads to a large fraction of extraordinarily hard instances. Interestingly, the median value of the CPU time does not vary much when varying the average degree. However, around $d = 25$ very hard instances occur that for $n = 60$ are several orders of magnitude harder to solve than the typical instances (see Figure 9), a phenomenon similar to what Gent and Walsh have also observed for kSAT in the satisfiable region [Gent and Walsh, 1994].

When comparing ORD-Horn with the pointizable subclass on $S(100,25,6.5)$, van Beek and Manchak did not observe any significant performance difference, which our experiments confirmed. When running the backtracking algorithm on 500 instances with a time limit of 20 sec and varying the *Split* set and the strategy for selecting the constraints in the search, the number of solved instances as well as the runtime was almost the same for ORD-Horn and the pointizable set. The results of this experiment are displayed in Figure 10, in which the percentage of solved instances is plotted against the maximal CPU time necessary to solve one instance.

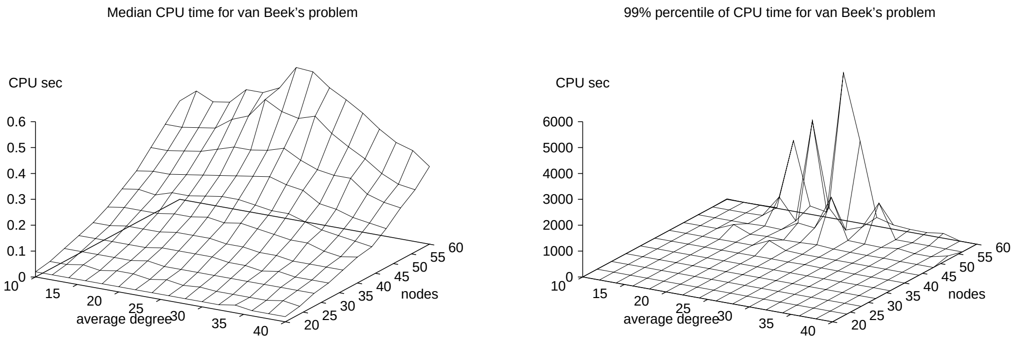

Figure 9: 50% and 99% percentile of CPU time on $S(n, d, 6.5)$

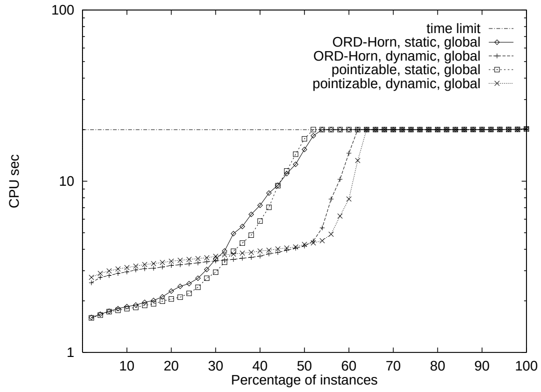

Figure 10: Runtime performance on $S(100, 25, 6.5)$

However, it is, of course, not evident that the same instances are solved by all methods. As a matter of fact, it turns out that by using different search methods, different instances get solved.

Based on this observation, we ran 16 different search strategies resulting from combining the four possible candidates $\mathbf{A}, \mathcal{C}, \mathcal{P}, \mathcal{H}$ for the split-set *Split*, with dynamic and static constraint ordering and local and global evaluation of the constrainedness. Using all of the 16 methods on 500 generated instances with a time limit of 20 sec on each method resulted in 99% solved instances, while the application of just one method using ORD-Horn, static ordering and global evaluation with a time limit of 1800 sec solved only 85%.

In Figure 11, the results of this experiment are displayed, plotting the percentage of solved instances against the maximal CPU time necessary to solve one instance. The line for the combined strategies results from multiplying the minimum CPU time to solve a particular instance by one method with 16, which would be the actual costs if all methods were applied in parallel.

One should note that the combination of different search strategies is com-

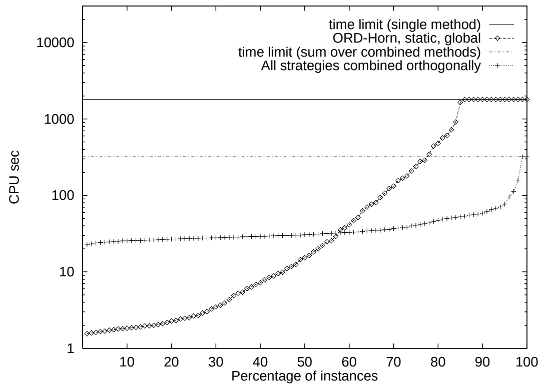

Figure 11: The effect of combining strategies orthogonally

pletely orthogonal and does not require any communication, which makes it very well suited for parallel implementations.

8 Conclusions and Outlook

We showed that using the ORD-Horn subclass in the backtracking algorithm proposed by Ladkin and Reinefeld [1992] leads to a complete reasoning algorithm and has—as conjectured in [Nebel and Bürckert, 1995]—the effect of enhancing search efficiency. On instances in the phase transition, which we have identified in this paper, the ORD-Horn subclass leads to an additional performance enhancement over the already highly optimized version [van Beek and Manchak, 1996] of Ladkin and Reinefeld's [1992] backtracking algorithm. For the hard satisfiable problems described in [van Beek and Manchak, 1996], the benefit of using the ORD-Horn class is not directly observable. However, when combining it orthogonally with other search strategies one notes that by using ORD-Horn some instances become solvable which are not solvable otherwise.

An interesting question is, whether the orthogonal combination of search strategies as described above can also lead to a better performance in the phase transition region. Another interesting question is, whether local search methods similar to GSAT [Selman *et al.*, 1992] can be applied to temporal reasoning. A direct application of GSAT, however, does not seem to be promising because translations from Allen's calculus to propositional logic lead to a cubic blowup [Nebel and Bürckert, 1995].

Acknowledgements

I would like to thank Peter Ladkin for discussions concerning the problems discussed in this paper and Peter van Beek for discussions and making available the

programs he used to evaluate different search strategies on temporal reasoning problems.

References

[Allen, 1983] James F. Allen. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843, November 1983.

[Allen, 1991] James F. Allen. Temporal reasoning and planning. In James F. Allen, Henry A. Kautz, Richard N. Pelavin, and Josh D. Tenenberg, editors, *Reasoning about Plans*, chapter 1, pages 1–67. Morgan Kaufmann, San Mateo, CA, 1991.

[Benzer, 1959] S. Benzer. On the topology of the genetic fine structure. *Proc. Nat. Acad. Sci. USA*, 45:1607–1620, 1959.

[Cheeseman *et al.*, 1991] Peter Cheeseman, Bob Kanefsky, and William M. Taylor. Where the *really* hard problems are. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, pages 331–337, Sydney, Australia, August 1991. Morgan Kaufmann.

[Feiner *et al.*, 1993] Steven K. Feiner, Diane J. Litman, Kathleen R. McKeown, and Rebecca J. Passonneau. Towards coordinated temporal multimedia presentation. In M. Maybury, editor, *Intelligent Multi Media*. AAAI Press, Menlo Park, CA, 1993.

[Gent and Walsh, 1994] Ian P. Gent and Toby Walsh. The hardest random SAT problems. In B. Nebel and L. Dreschler-Fischer, editors, *KI-94: Advances in Artificial Intelligence*, pages 355–366, Saarbrücken, Germany, 1994. Springer-Verlag.

[Gerevini and Schubert, 1993] Alfonso Gerevini and Lenhart Schubert. Efficient temporal reasoning through timegraphs. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 648–654, Chambery, France, August 1993.

[Golumbic and Shamir, 1993] Martin C. Golumbic and Ron Shamir. Complexity and algorithms for reasoning about time: A graph-theoretic approach. *Journal of the Association for Computing Machinery*, 40(5):1128–1133, November 1993.

[Haralick and Elliot, 1980] R. M. Haralick and G. L. Elliot. Increasing tree search efficiency for constraint satisfaction problems. *Artificial Intelligence*, 14:263–313, 1980.

[Ladkin and Maddux, 1994] Peter B. Ladkin and Roger Maddux. On binary constraint problems. *Journal of the Association for Computing Machinery*, 41(3):435–469, May 1994.

[Ladkin and Reinefeld, 1992] Peter B. Ladkin and Alexander Reinefeld. Effective solution of qualitative interval constraint problems. *Artificial Intelligence*, 57(1):105–124, September 1992.

[Ladkin and Reinefeld, 1993] Peter B. Ladkin and Alexander Reinefeld. A symbolic approach to interval constraint problems. In J. Calmet and J. A. Campbell, editors, *Artificial Intelligence and Symbolic Mathematical Computing*, volume 737 of *Lecture Notes in Computer Science*, pages 65–84. Springer-Verlag, Berlin, Heidelberg, New York, 1993.

[Montanari, 1974] Ugo Montanari. Networks of constraints: fundamental properties and applications to picture processing. *Information Science*, 7:95–132, 1974.

[Nebel and Bürckert, 1995] Bernhard Nebel and Hans-Jürgen Bürckert. Reasoning about temporal relations: A maximal tractable subclass of Allen's interval algebra. *Journal of the Association for Computing Machinery*, 42(1):43–66, January 1995.

[Nökel, 1991] Klaus Nökel. *Temporally Distributed Symptoms in Technical Diagnosis*, volume 517 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, Berlin, Heidelberg, New York, 1991.

[Selman *et al.*, 1992] Bart Selman, Hector J. Levesque, and David Mitchell. A new method for solving hard satisfiability problems. In *Proceedings of the 10th National Conference of the American Association for Artificial Intelligence*, pages 440–446, San Jose, CA, July 1992. MIT Press.

[Song and Cohen, 1988] Fei Song and Robin Cohen. The interpretation of temporal relations in narrative. In *Proceedings of the 7th National Conference of the American Association for Artificial Intelligence*, pages 745–750, Saint Paul, MI, August 1988.

[van Beek and Cohen, 1990] Peter van Beek and Robin Cohen. Exact and approximate reasoning about temporal relations. *Computational Intelligence*, 6:132–144, 1990.

[van Beek and Manchak, 1996] Peter van Beek and Dennis W. Manchak. The design and experimental analysis of algorithms for temporal reasoning. *Journal of Artificial Intelligence Research*, 4:1–18, 1996.

[Vilain and Kautz, 1986] Marc B. Vilain and Henry A. Kautz. Constraint propagation algorithms for temporal reasoning. In *Proceedings of the 5th National Conference of the American Association for Artificial Intelligence*, pages 377–382, Philadelphia, PA, August 1986.