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

Bernhard Nebel[1]

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.

1 INTRODUCTION

Representation of qualitative temporal information and reasoning with it is an integral part of many artificial intelligence tasks, such as presentation planning [3], natural language understanding [16], and diagnosis of technical systems [14]. *Allen's* [1] *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 [6, 8, 13, 17, 19], 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* [13] 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.[2]

Since reasoning in the full calculus is NP-hard [19], it is necessary to employ some sort of *exhaustive search method* if one wants complete reasoning in the full calculus. Ladkin and Reinefeld [9] have proposed a backtracking algorithm that uses path-consistency as a *forward checking* technique [7] during the backtrack search, which results in pruning the search tree significantly compared with the algorithm proposed by Allen [1]. As pointed out by Ladkin and Reinefeld [9], 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 backtrack 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 [13], 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 [10, 18] do not seem to justify this conjecture.

It may be the case, however, that in previous experiments the authors missed generating hard instances or did not look for the right performance indicators. In Section 5, we identify the *phase transition region* [2] 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 heuristics in an orthogonal way can dramatically improve the performance on van Beek and Manchak's [18] hard problem instances.

2 ALLEN'S CALCULUS

Allen's [1] approach to reasoning about time[3] 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.

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 $\mathcal{A}$. This set forms together with the operations *intersection* ($r \cap r'$), relational *converse* ($r^\smile$), and relational *composition* ($r \circ r'$) an algebra, which is called *Allen's interval algebra*.

[1] Institut für Informatik, Albert-Ludwigs-Universität Freiburg, Am Flughafen 17, D-79110 Freiburg, Germany; e-mail: nebel@informatik.uni-freiburg.de

[2] The C-programs that were used for the evaluation are available by anonymous ftp from ftp.informatik.uni-freiburg.de as /pub/documents/papers/ki/allen-csp-solving.programs.tar.gz.

[3] Because of lack of space, we only sketch the basic notions of Allen's calculus. For a complete description one should consult, e.g., [1, 8, 13, 17].

ECAI 96. *12th European Conference on Artificial Intelligence*
Edited by W. Wahlster
Published in 1996 by John Wiley & Sons, Ltd.

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 [6] and because it is one of the most important tasks in practical applications [18].

The most often used method to determine satisfiability of a temporal constraint graph is the *path-consistency method*,[4] which was already proposed by Allen [1]. 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 [1]. Since ISAT is NP-complete [19], 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* $\mathcal{C}$ [14, 17], the *pointizable class* $\mathcal{P}$ [8, 17], and the *ORD-Horn class* $\mathcal{H}$ [13], 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 following backtracking algorithm, which has been proposed by Ladkin and Reinefeld [9], appears to be the most efficient version of such an algorithm:

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, ..., r_k s.t. all r_l ∈ Split
      if no label can be split then return true
      endif
      for all labels r_l (1 ≤ l ≤ k) do
        C_ij ← r_l
        if consistent(C) then return true
        endif
      endfor
      return false
  endif
endfunction
```

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 [9] 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 [18]. 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 [13]. 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 computer-based analysis gives the following average branching factors: $\mathbf{A}$ 6.5, $\mathcal{C}$ 3.551, $\mathcal{P}$ 2.955, $\mathcal{H}$ 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 [11].

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:

$$C^0 = C \tag{2}$$
$$C^k = \widehat{C^{k-1}}[\sigma_k] \tag{3}$$
$$S^0 = C \tag{4}$$
$$S^k = S^{k-1}[\sigma_k] \tag{5}$$

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.

[4] An alternative method for a subset of Allen's interval algebra has been developed by Gerevini and Schubert [5].

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 [18]. The DNA sequencing instance from molecular biology that has been suggested by van Beek and Manchak [18] is unfortunately not adequate for our purposes because the structure of constraints leads to identical results for $\mathcal{P}$ and $\mathcal{H}$ [18].

For these reasons, the only possibility to evaluate the usefulness of the ORD-Horn class is to randomly generate temporal constraint networks as in [9, 10, 18]. We use two models to generate constraint networks, denoted by $A(n, d, s)$ 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 $S(n, d, s)$, 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 [18], and they reported that a large fraction of instances generated by $S(100, 25, 6.5)$ 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* [2] conjectured that "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 [10] 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 1 demonstrates,[5] this is indeed the case for $A(n, d, 6.5)$.

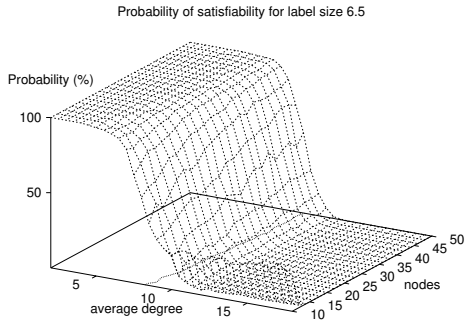

Figure 1. Probability of satisfiability for $A(n, d, 6.5)$

The probability that the instance is satisfiable drops from 1 to 0 around $d = 9.5$. For other values of s and other distributions of constraints, we observed a similar behavior (see full paper [12]). The general picture was that with higher values of s the phase transition moves to higher values of d.

As expected, hard instances appear around the phase transition, meaning that the median value of CPU time peaks around the phase transition. Similarly, the mean value has a peak there, as shown in Figure 2 (the solid line marks the phase transition).

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

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

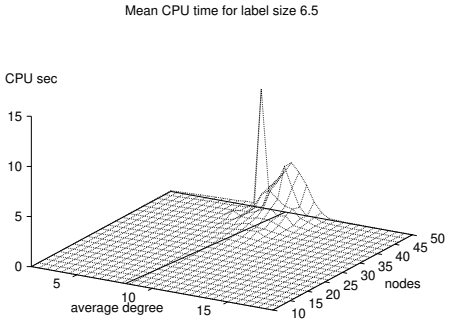

Figure 2. CPU time for $A(n, d, 6.5)$

a significant performance enhancement in some cases (Figure 3).

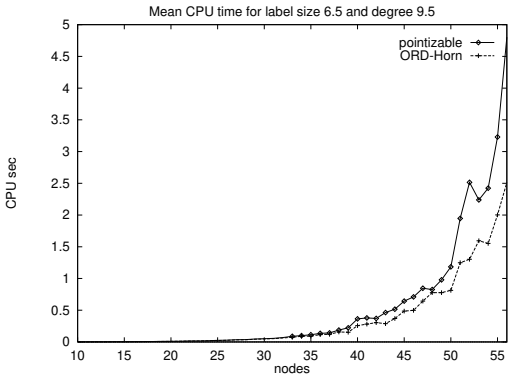

Figure 3. Comparison between $\mathcal{P}$ and $\mathcal{H}$

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. One question might be, however, where the performance enhancements came from. As Figure 4 shows, the median CPU time value is almost identical for using $\mathcal{H}$ and $\mathcal{P}$ and the main differences appear in 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.

The results described above were achieved by using all techniques described in [18] 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 [18].

queue/no queue: The path-consistency procedure uses a weighted *queue* scheme for the constraints to be processed next [18] or the scheme described in [9], which uses *no queue*.

As it turns out, the improvement of using $\mathcal{H}$ instead of $\mathcal{P}$ is small

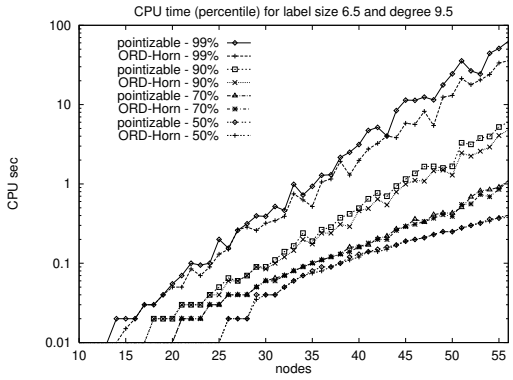

Figure 4. Comparison by percentiles

compared with the improvements achievable by other means (Figure 5).

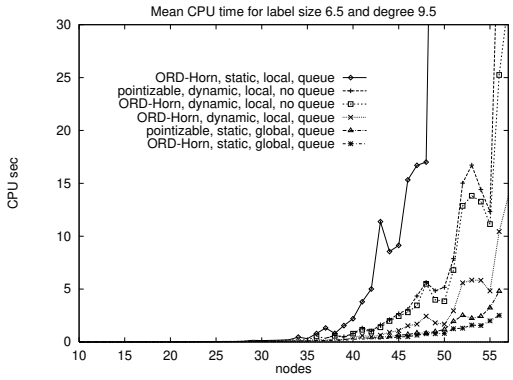

Figure 5. Using ORD-Horn and other design choices

The two lower curves in Figure 5 correspond to the curves in Figures 3. 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 [18] used $S(n, d, 6.5)$-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 are several orders of magnitude harder to solve than the typical instances, a phenomenon similar to what Gent and Walsh have also observed for kSAT in the satisfiable region [4].

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. 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. While the application of just one method using ORD-Horn, static ordering and global evaluation on 500 generated instances with a time limit of 1800 sec per instance solved only 85%, using all of the 16 methods with a time limit of 20 sec on each method resulted in 99% solved instances. Further, in 40% of all cases one of the $\mathcal{H}$-methods gave the fastest response, in 22% of all cases one of the $\mathcal{P}$-methods was the fastest, and the $\mathcal{C}$ and $\mathbf{A}$-methods contributed with 19%.

In Figure 6, 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.

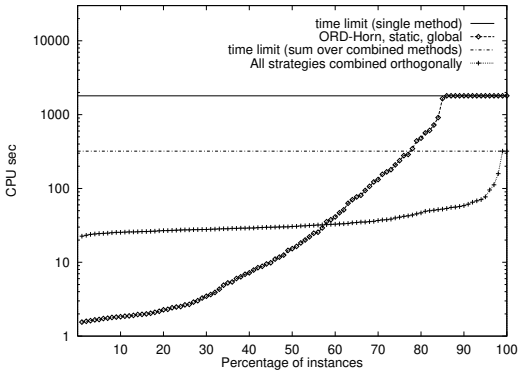

Figure 6. The effect of combining strategies orthogonally

One should note that the combination of different search strategies is completely 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 [9] leads to a complete reasoning algorithm and has—as conjectured in [13]—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 [18] of Ladkin and Reinefeld's [9] backtracking algorithm. For the hard satisfiable problems described in [18], 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 [15] 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 [13].

ACKNOWLEDGEMENTS

I would like to thank the two anonymous referees for helpful comments, 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

[1] J. F. Allen, 'Maintaining knowledge about temporal intervals', *Comm. ACM*, **26**, 832–843, (1983).

[2] P. Cheeseman, B. Kanefsky, and W. M. Taylor, 'Where the *really* hard problems are', in *Proc. 12th IJCAI*, pp. 331–337, Sydney, Australia, (1991). Morgan Kaufmann.

[3] S. K. Feiner, D. J. Litman, K. R. McKeown, and R. J. Passonneau, 'Towards coordinated temporal multimedia presentation', in *Intelligent Multi Media*, ed., M. Maybury, AAAI Press, Menlo Park, CA, (1993).

[4] I. P. Gent and T. Walsh, 'The hardest random SAT problems', in *KI-94: Advances in Artificial Intelligence*, eds., B. Nebel and L. Dreschler-Fischer, pp. 355–366, Saarbrücken, Germany, (1994). Springer-Verlag.

[5] A. Gerevini and L. Schubert, 'Efficient temporal reasoning through timegraphs', in *Proc. 13th IJCAI*, pp. 648–654, Chambery, France, (1993).

[6] M. C. Golumbic and R. Shamir, 'Complexity and algorithms for reasoning about time: A graph-theoretic approach', *Jour. ACM*, **40**, 1128–1133, (1993).

[7] R. M. Haralick and G. L. Elliot, 'Increasing tree search efficiency for constraint satisfaction problems', *Artif. Intell.*, **14**, 263–313, (1980).

[8] P. B. Ladkin and R. Maddux, 'On binary constraint problems', *Jour. ACM*, **41**, 435–469, (1994).

[9] P. B. Ladkin and A. Reinefeld, 'Effective solution of qualitative interval constraint problems', *Artif. Intell.*, **57**(1), 105–124, (1992).

[10] P. B. Ladkin and A. Reinefeld, 'A symbolic approach to interval constraint problems', in *Artificial Intelligence and Symbolic Mathematical Computing*, eds., J. Calmet and J. A. Campbell, 65–84, Springer-Verlag, Berlin, (1993).

[11] U. Montanari, 'Networks of constraints: fundamental properties and applications to picture processing', *Infor. Sci.*, **7**, 95–132, (1974).

[12] B. Nebel, 'Solving Hard Qualitative Temporal Reasoning Problems: Evaluating the Efficiency of Using the ORD-Horn Class', Technical Report UIB-96-02, Fakultät für Informatik, Univ. Ulm (1996).

[13] B. Nebel and H.-J. Bürckert, 'Reasoning about temporal relations: A maximal tractable subclass of Allen's interval algebra', *Jour. ACM*, **42**, 43–66, (1995).

[14] K. Nökel, *Temporally Distributed Symptoms in Technical Diagnosis*, Springer-Verlag, Berlin, 1991.

[15] B. Selman, H. J. Levesque, and D. Mitchell, 'A new method for solving hard satisfiability problems', in *Proc. 10th AAAI*, pp. 440–446, San Jose, CA, (1992). MIT Press.

[16] F. Song and R. Cohen, 'The interpretation of temporal relations in narrative', in *Proc. 7th AAAI*, pp. 745–750, Saint Paul, MI, (1988).

[17] P. van Beek and R. Cohen, 'Exact and approximate reasoning about temporal relations', *Comp. Intell.*, **6**, 132–144, (1990).

[18] P. van Beek and D. W. Manchak, 'The design and experimental analysis of algorithms for temporal reasoning', *Jour. of Artif. Intell. Res.*, **4**, 1–18, (1996).

[19] M. B. Vilain and H. A. Kautz, 'Constraint propagation algorithms for temporal reasoning', in *Proc. 5th AAAI*, pp. 377–382, Philadelphia, PA, (1986).