Processing Disjunctions of Temporal Constraints

E. Schwalb R.Dechter

Information and Computer Science Dept.
University of California, Irvine

Abstract

This paper describes new algorithms for processing quantitative Temporal Constraint Satisfaction Problems (TCSP). In contrast to discrete CSPs, enforcing path-consistency on TCSPs is exponential due to the fragmentaiton problem. We present an efficient polynomial constraint propagation algorithms, called Loose Path Consistency, which is shown to improve the performance backtrack search algorithms by orders of magnitude. The tradeoffs between the effectiveness and efficiency of LPC are analyzed. We report the presence of a phase transition on this domain and perform the empirical evaluation on problems which lie in the transition region.

1 Introduction

Problems involving temporal constraints arise in various areas such as scheduling [12] and planning with temporal databases [3, 14]. Several formalisms for expressing and reasoning about temporal constraints have been proposed, most notably, Allen's interval algebra [1], Vilain, Kautz and van Beek's point algebra [15], Dean's Time Map Management [3], Dechter, Meiri and Pearl's Temporal Constraint Satisfaction Problems (TCSP) [4] and Meiri's combined model of quantitative and qualitative networks [10]. Improved algorithms for processing Allen's interval algebra were presented in [9]. Here we extend the work on TCSPs by providing a new algorithm which improves on the algorithms presented in [4, 13].

In this paper, we present a new polynomial approximation algorithm for processing *quantitative* constraint networks, which is extended to process combined qualitative and quantitative constraints. In the reminder of this paper we use TCSP to refer to the combined qualitative and quantitative model.

The TCSP model facilitates the following tasks:

1. Finding one or more consistent scenarios.
2. Finding all feasible times at which a given event can occur.
3. Finding all relationships between two given events.

All these tasks are known to be NP-complete [4, 10]. The source of complexity stems from allowing disjunctive relationships between pairs of variables. Disjunctive constraints often arise in scheduling and planning applications. As an example of a disjunctive relation, consider the constraint specifying that the time it takes to ship cargo depends on whether it is shipped by air or ground transports.

Example 1 : A large cargo needs to be delivered from New York to Los Angeles within 8-10 days. From New York to Chicago the delivery requires 1-2 days by *air or* 10-11 days on the *ground*. From Chicago to L.A. the delivery requires 3-4 days by air *or* 13-15 days on the ground.

Given the above constraints, we are interested in answering questions such as: "are these constraints satisfiable ?" or "when should the cargo be in Dallas ?" or "can the cargo arrive in L.A. on Jan 8-9 ?". The model of Temporal Constraint Satisfaction Problems (TCSP) provides a representation with which such questions can be answered.

The central task in constraint processing is to decide consistency of a set of constraints. Since deciding consistency is intractable, it is common to use approximation algorithms such as path-consistency. In contrast to discrete CSPs, enforcing path-consistency on TCSPs may be exponential, as noted in [4] and observed empirically in [13]. This motivated a polynomial algorithm for approximating path-consistency, called Upper-Lower-Tightening (ULT). Here we identify the cause of the exponential behavior and present an improved algorithm called Loose Path-Consistency (LPC).

For randomly generated problems, a phase transition from easy to hard is reported on 3-CNF formulas in [2, 11]. It is observed that the most difficult problems lie in the transition region. Here we show that randomly generated TCSPs also exhibit a phase transition. The empirical evaluation is performed on relatively hard problems which lie in the phase transition. We compare our new algorithm, LPC, and the previously proposed algorithm ULT, with respect to efficiency and effectiveness. By effectiveness we refer to the ability to detect inconsistencies and by efficiency we refer to the execution time. We show that LPC is significantly more effective than ULT in detecting in-

0-8186-7528/96 $5.00

consistencies and improving performance of backtrack search by orders of magnitude.

2 Processing Constraints

A Temporal Constraint Satisfaction Problem (TCSP) consists of a set of variables $X_1, \ldots, X_n$, having *rational* domains, each representing a time point. Each constraint C is a set of intervals

$$C \stackrel{\text{def}}{=} \{I_1, \ldots, I_n\} = \{[a_1, b_1], \ldots, [a_n, b_n]\}.$$

A unary constraint C_i restricts the domain of the variable X_i to the given set of intervals

$$C_i \stackrel{\text{def}}{=} (a_1 \leq X_i \leq b_1) \cup \ldots \cup (a_n \leq X_i \leq b_n).$$

A binary constraint C_{ij} over X_i, X_j restricts the permissible values for the distance $X_j - X_i$; it represents the disjunction

$$C_{ij} \stackrel{\text{def}}{=} (a_1 \leq X_j - X_i \leq b_l) \cup \ldots \cup (a_n \leq X_j - X_i \leq b_n).$$

All intervals are assumed to be open and pairwise disjoint.

A tuple $X = (x_1, \ldots, x_n)$ is called a *solution* if the assignment $X_1 = x_1, \ldots, X_n = x_n$ satisfies all the constraints. The network is *consistent* iff at least one solution exists.

Definition 1 : [composition]
Let $T = \{I_1, I_2, \ldots, I_r\}$ and $S = \{J_1, J_2, \ldots, J_s\}$ be two constraints. The *composition* of T and S, denoted by $T \otimes S$, admits only values r for which there exists $t \in T$ and $s \in S$ such that $r = t + s$.

2.1 Path-Consistency and Fragmentation

A commonly used constraint propagation algorithm enforces path-consistency. A constraint T_{ij} is *path-consistent* iff $T_{ij} \subseteq \cap_{\forall k}(T_{ik} \otimes T_{kj})$ and a network is *path-consistent* iff all its constraints are *path-consistent.*

Path-consistency can be enforced by applying the operator $T_{ij} \leftarrow T_{ij} \cap (T_{ik} \otimes T_{kj})$ until a fixed point is reached.

Lemma 1 : *[4] Algorithm PC computes a path-consistent network and terminates in $O(n^3R^3)$, where R is the range of the quantitative constraints. Algorithm DPC terminates in $O(n^3R^2)$ steps.*

As we enforce path-consistency on quantitative TCSPs, some intervals may be broken into several smaller subintervals and the number of intervals per constraint (i.e. disjunction size) may increase. This may result in exponential blowup (bound by $O(n^3R^3)$). This is called a *fragmentation problem.*

Consider the 3 variable network in Figure 3. Each node represents a *point variable*, and the intervals on the edges represent a disjunctive constraint, namely the interval label $[a_1, b_1]$ $[a_2, b_2]$ $[a_3, b_3]$ representing the constraint $X_j - X_i \in [a_1, b_1] \cup [a_2, b_2] \cup [a_3, b_3]$. Enforcing path consistency on the network in Figure 3a results in increasing the number of intervals of T_{13} from 3 to 10 as shown in Figure 3b. Clearly, if this pattern is repeated throughout the network, the computation may become exponential in the number of triangles processed. This was observed empirically in [13].

Algorithm PC

1. $Q \leftarrow \{(i,k,j) \mid (i < j)$ *and* $(k \neq i,j)\}$
2. **while** $Q \neq \{\}$ **do**
3. select and delete a path (i,k,j) from Q
4. **if** $T_{ij} \neq T_{ik} \otimes T_{kj}$ **then**
5. $T_{ij} \leftarrow T_{ij} \cap (T_{ik} \otimes T_{kj})$
6. **if** $T_{ij} = \{\}$ **then** exit (inconsistency)
7. $Q \leftarrow Q \cup \{(i,k,j) \mid \forall k \neq i,j\}$
8. **end-if**
9. **end-while**

Figure 1: Algorithm PC.

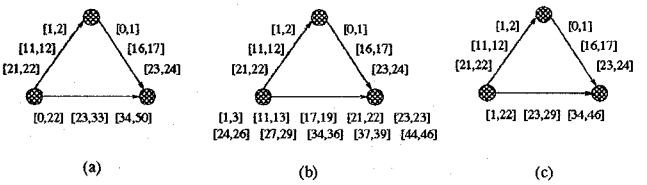


Figure 2: The fragmentation problem.

3 Algorithm LPC

In algorithm *Loose Path-Consistency (LPC)* we control the fragmentation by replacing the intersection operator $\cap$ with the loose intersection operator $\lhd$.

Definition 2 : The *loose intersection*, $T \lhd S$ consists of the intervals $\{I'_1, \ldots, I'_r\}$ such that $\forall i \;\; I'_i = [L_i, U_i]$ where $[L_i, U_i]$ are the lower and upper bounds of the intersection $I_i \cap S$.

The number of intervals in the constraint between X_i and X_j, denoted T_{ij}, is not increased by the operation $T_{ij} \leftarrow T_{ij} \lhd (T_{ik} \otimes T_{kj})$. Note that the $\lhd$ operator is asymmetric, namely $T \lhd S \neq S \lhd T$.

Example 2 : Let $T = \{[1,4],[10,15]\}$ and $S = \{[3,11],[14,19]\}$. Then $T \lhd S = \{[3,4],[10,15]\}$, $S \lhd T = \{[3,11],[14,15]\}$ while $S \cap T = \{[3,4],[10,11],[14,15]\}$.

Algorithm LPC is presented in Figure 3. The network N' is a relaxation of N and therefore loosely intersecting it with N results in an equivalent network. At every iteration if LPC (except the first and the last) at least one interval is removed. This allows us to conclude that:

Algorithm Loose Path-Consistency (LPC)

1. **input:** N
2. $N'' \leftarrow N$
3. **repeat**
4. $\quad N \leftarrow N''$
5. $\quad$ Compute N' by assigning $T'_{ij} = \cap_{\forall k}(C_{ik} \otimes C_{kj})$, for all i, j.
6. $\quad$ Compute N'' by loosely intersecting $T''_{ij} = C_{ij} \lhd T'_{ij}$, for all i, j.
7. **until** $\exists i, j \ \ (T''_{ij} = \phi)$; inconsistency, or
   or $\forall i, j \ \ |T''_{ij}| = |C_{ij}|$; no interval removed.
8. **if** $\exists i, j \ \ (T''_{ij} = \phi)$ **then output** "inconsistent."
   else output: N''.

Figure 3: The Loose Path-Consistency (LPC) algorithm.

Lemma 2 : *Algorithm LPC (see Figure 2) computes a network which is equivalent to the input network and terminates in $O(n^3k^3e)$, where n is the number of variables, k is the maximum number of disjuncts per constraint and e is the total number of constraints.*

Example 3 : Consider the constraints:

$$\begin{array}{lll} X_1 - X_0 & \in & [10,20] \cup [100,110] \\ X_2 - X_1 & \in & [20,40] \cup [100,130] \\ X_3 - X_0 & \in & [80,100] \cup [150,160] \cup [180,190] \\ X_3 - X_1 & \in & [30,40] \cup [130,150] \\ X_3 - X_2 & \in & [50,70] \cup [110,120] \cup [130,140] \cup [160,190] \end{array}$$

After 3 iterations, algorithm LPC terminates with the network:

$$\begin{array}{lll} X_1 - X_0 & \in & [10,20] \\ X_2 - X_0 & \in & [30,50] \\ X_2 - X_1 & \in & [20,30] \\ X_3 - X_0 & \in & [150,160] \\ X_3 - X_1 & \in & [130,140] \\ X_3 - X_2 & \in & [110,120] \end{array}$$

3.1 Variants of LPC

A variant of algorithm LPC can be obtained by replacing, in algorithm PC (Figure 1), the intersection $\cap$ operator with loose-intersection $\lhd$. Another variant not presented here is called Directional LPC (DLPC).

To refine the tradeoff between effectiveness and efficiency, we restrict the algorithm to induce constraints from only a subset of the triangles, where at least one of the constraints was non-universal in the input network. The intuition being that second level constraint recording is relatively week. Algorithm Partial LPC (PLPC) approximates LPC by applying the relaxation operation $T_{ij} \leftarrow T_{ij} \lhd (T_{ik} \otimes T_{kj})$ only in case T_{ij} and at least one of T_{ik}, T_{kj} is non-universal in the input network.

The partial order between the different constraint propagation algorithms we experimented with is presented in Figure 7. A directed edge from algorithm $\mathcal{A}_1$ to $\mathcal{A}_2$ indicates that $\mathcal{A}_2$ computes a tighter network than $\mathcal{A}_1$ on an *instance by instance* basis, and that its complexity is higher accordingly. Together these

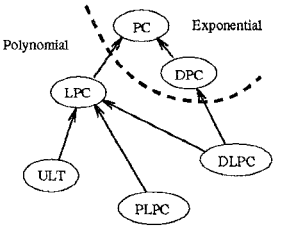


Figure 4: The partial order on the effectiveness of the variants of LPC.

algorithms form a spectrum of sound but incomplete propagation algorithms.

4 Empirical Evaluation

Algorithm ULT was the first algorithm psoposed for processing disjunctions in TCSPs [13]. This section is organized as follows: In section 4.1, we determine relative efficiency and effectiveness of algorithm ULT, algorithm LPC and its variants DLPC and PLPC. By effectiveness we refer to the ability to detect inconsistencies and by efficiency we refer to the execution time. In section 4.2 we examine the effectiveness of LPC and ULT as a forward checking procedure within backtracking search. The empirical evaluation of these search methods is performed on relatively hard problems which lie in the phase transition.

Problems were generated with the following parameters: n and e are the number of variables and constraints, and k is the number of intervals per quantitative point-point constraint. These quantitative constraints specify integers in $[-R, R]$, and the tightness α of a constraint $T = \{I_1, \ldots, I_k\}$ is $(|I_1| + \cdots + |I_k|)/2R$ where $|I_i|$ is the size of I_i; we used uniform tightness for all constraints.

4.1 Comparing LPC and ULT

We compare the effectiveness of incomplete constraint propagation algorithms by counting the fraction of cases in which the weaker algorithm detected inconsistency provided that the stronger algorithm also detected inconsistency (recall Figure 4). PC may

32 *vars and* 200 *reps.*									
# of Consts	*Acc of PLPC*	*Acc of ULT*	*# Op. LPC*	*# Op. PLPC*	*# Op. DLPC*	*Time LPC*	*Time PLPC*	*Time DLPC*	*Time ULT*
150	98%	90%	$25K$	$12K$	$5K$	0.546	0.400	0.165	0.132
200	99%	15%	$27K$	$17K$	$8K$	0.623	0.533	0.259	0.162
250	100%	45%	$14K$	$11K$	$10K$	0.380	0.350	0.315	0.181
300	100%	77%	$9K$	$8K$	$8K$	0.287	0.275	0.270	0.164
350	100%	94%	$7K$	$7K$	$7K$	0.244	0.241	0.235	0.126
400	100%	100%	$6K$	$6K$	$6K$	0.211	0.212	0.204	0.105

Table 1: Effectiveness and efficiency of LPC, DLPC, Partial LPC and ULT.

be exponential in the number of intervals per constraint in the input network while ULT's execution time is almost constant in the number of intervals. Nevertheless, ULT is able to detect inconsistency in about 70% of the cases in which PC does [13]. Here we demonstrate that algorithm LPC is capable of detecting even more inconsistencies than ULT. Therefore, we show LPC computes a better approximation to PC than ULT.

We compare the relative effectiveness and efficiency of algorithms LPC, DLPC, PLPC and ULT. The results are presented in Table 1. The columns labeled "Acc $< alg >$" specify the accuracy of algorithm $< alg >$ relative to LPC, namely fraction of cases algorithm $< alg >$ detected inconsistency given that LPC did. The columns labeled "# Op $< alg >$" describe the number of revision operations made by algorithm $< alg >$. The basic revision operation of LPC is $T_{ij} \leftarrow T_{ij} \triangleleft (T_{ik} \otimes T_{kj})$, while for ULT we use the relaxation operation given in [13]. This measure is machine and implementation independent, unlike execution time. The problems generated have 32 variables and constraint tightness of 45%. Each entry represents the average of 200 instances.

From Table 1 we can conclude that (1) LPC is more effective yet less efficient than ULT, and (2) PLPC is almost as effective as LPC yet more efficient than LPC. We conclude that on our benchmarks, algorithm PLPC is the best. Therefore, in the rest of this paper, whenever we mention LPC we refer to PLPC.

4.2 Backtracking

To find a solution to a given TCSP, we use a backtracking search algorithm. A solution of a TCSP is a selection of a single interval or relation from each constraint, called *constraint labeling*. Consistency of such a constraint labeling is decided by enforcing path-consistency. In the first part of this section we reports results of experiments performed on quantitative TCSPs, while in the second part we focus on qualitative networks.

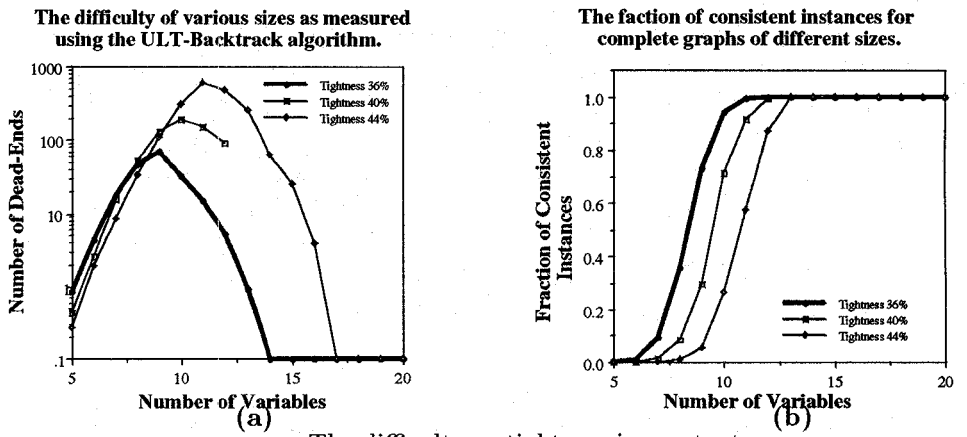
The experiments were performed with a backtrack search algorithm which uses the constraint propagation algorithms presented above (recall Figure 7) as a forward checking procedure. Given a TCSP, a single interval or relation is selected from a disjunctive constraint and consistency is tested using either PLPC or ULT. Thereafter, on the tightened network, we repeat selecting and testing consistency until either inconsistency is detected or a solution is found. When inconsistency is detected, a dead-end is encountered and the search algorithm backtracks undoing the last constraint labeling.

Constraint propagation algorithms can also be used as a preprocessing phase before backtracking, to reduce the number of dead-ends encountered during search. After preprocessing with algorithm PC, problems became even harder to solve, due to the increased fragmentation. In contrast, preprocessing with ULT results in problems on which naive backtrack search is manageable [13]. This algorithms is called "Old-Backtrack+ULT"; it was used as our reference point.

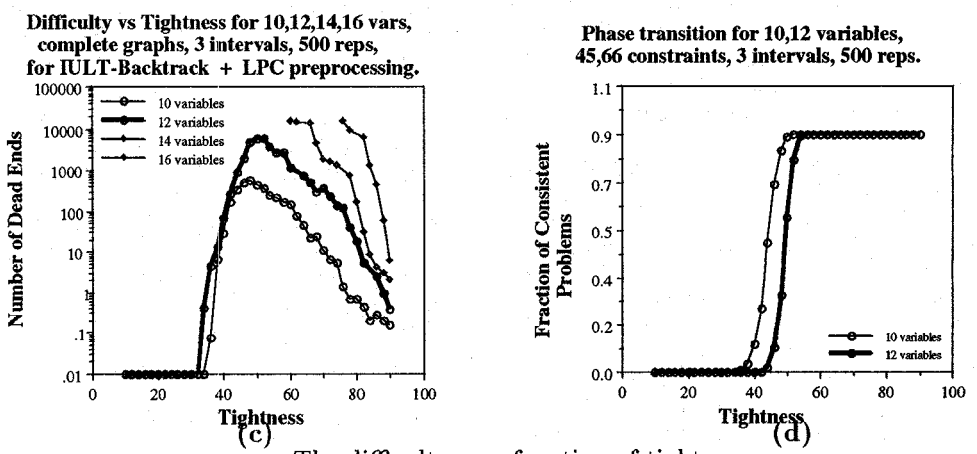
In this section, we compare three backtrack search algorithms: (1) "Old-Backtrack+ULT" which uses ULT as a preprocessing phase but no forward checking is used; (2) "ULT-Backtrack+ULT" which uses ULT both as a preprocessing phase before backtracking and as a forward checking procedure during the search; (3) "LPC-Backtrack+LPC" which uses LPC both as a preprocessing phase before backtracking and as a forward checking procedure during the search.

In Figure 5 we report an exponential increase in number of dead-ends and execution time at certain regions. This region, where about half of the problems are satisfiable, is called the *transition region* [2, 11]. In Figures 5a, 5b we observe a phase-transition when varying the size of the network while in Figures 5c, 5d we observe a similar phenomenon when varying the tightness of the constraints. Some theoretical insights into this phenomenon in discrete CSPs can be found in [16].

As observed in Figure 6, ULT and LPC are capable of pruning dead-ends and improving search efficiency on our benchmarks by orders of magnitude. In particular, Old-Backtrack+ULT is about 1000 times slower than ULT-Backtrack+ULT, which is about 1000 times slower than LPC-Backtrack+LPC. The latter encounters about 20 dead-ends on the peak (worst performance) on networks with 12 variables and 66 constraints with 3 intervals to instantiate each (see Figure 10).



The difficulty as tightness is constant.



The difficulty as a function of tightness.

Figure 5

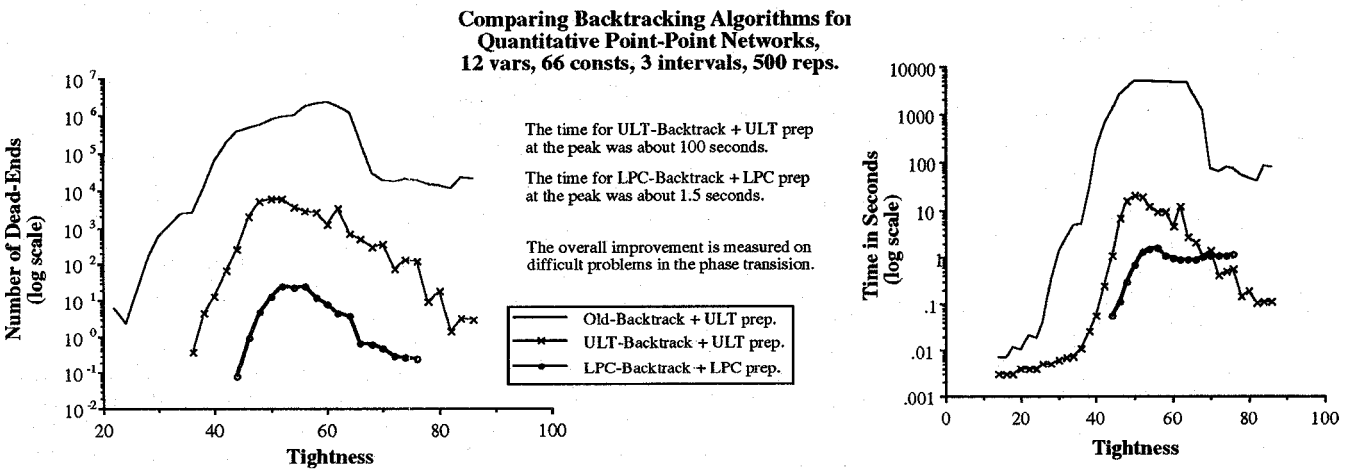


Figure 6: A comparison of various backtracking algorithms.

5 Conclusion

We discuss processing combined qualitative and quantitative Temporal Constraint Satisfaction Problems (TCSP). Using relatively hard problems which lie in the *transition region* we evaluate the effectiveness of algorithm LPC and its variants and show that they improve efficiency of backtrack search by orders of magnitude, even on small problems.

We identify a fragmentation problem which explains, in contrast to discrete Constraint Satisfaction Problems (CSP), why enforcing path-consistency on quantitative TCSPs is exponential. Identifying this problem allows us to design an efficient yet effective polynomial algorithm for processing TCSPs called Loose Path-Consistency (LPC), and its variants Directional LPC (DLPC) and Partial LPC (PLPC).

References

[1] Allen, J.F., 1983. Maintaining knowledge about temporal intervals, *CACM* 26 (11): 832-843.

[2] Cheesman, P, Kanefsky, B., Taylor, W., 1991. Where the Really Hard Problems Are. Proc. of IJCAI-91, 163-169.

[3] Dean, T.M., McDermott, D. V., 1987. Temporal data base management, *Artificial Intelligence* 32 (1987) 1-55.

[4] Dechter, R., Meiri, I., Pearl, J., 1991. Temporal Constraint Satisfaction Problems, *Artificial Intelligence* 49(1991) 61-95.

[5] Freuder, E.C. 1985. A sufficient condition of backtrack free search. *JACM* 32(4):510-521.

[6] Frost, D., Dechter, R., 1994. In Search of the Best Search: An empirical Evaluation, In *Proc. of AAAI-94*, pages 301-306.

[7] Frost, D., Dechter, R., 1994. Dead-End Driven Learning, In *Proc. of AAAI-94*, pages 294-300.

[8] Kautz, H., Ladkin, P., 1991. Integrating Metric and Qualitative Temporal Reasoning, In *Proc. of AAAI-91*, pages 241-246, 1991.

[9] Ladkin, P.B., Reinefeld, A., 1992. Effective solution of qualitative interval constraint problems, *Artificial Intelligence* **57** (1992) 105-124.

[10] Meiri, I., 1991. Combining Qualitative and Quantitative constraints in temporal reasoning In *Proc. AAAI-91*, pp. 260-268.

[11] Mitchell, D., Selman, B., Levesque, H., 1992. Hard and Easy Distributions of SAT Problems, Proc. of AAAI-92.

[12] Sadeh, N., 1991. Look-Ahead techniques for Micro-opportunistic Job Shop Scheduling, Ph.D. thessis, School of Computer Science, Carnegie Mellon University, March 1991.

[13] Schwalb, E., Dechter, R., 1993. Coping with Disjunctions in Temporal Constraint Satisfaction Problems, In Proc. AAAI-93, 127-132.

[14] Schwalb, E., Dechter, R., 1994. Temporal Reasoning with Constraints on Fluents and Events, In Proc. AAAI-94.

[15] Vilain, M., Kautz, H., Van Beek, P., 1989. Constraint Propagation Algorithms for Temporal Reasoning: A revised Report. In Readings in Qualitative Reasoning about Physical Systems, J. de Kleer and D. Weld (eds). 1989.

[16] Williams, C.P., Hogg, T., 1993. A typicality of phase transition search, *Computational Intelligence* 9(3):211-238.