Incremental Tractable Reasoning about Qualitative Temporal Constraints

Alfonso Gerevini
DEA - Universita degli studi di Brescia, via Branze 38,1-25123 Brescia, Italy
gerevini@ing.unibs.it

Abstract

In many applications of temporal reasoning we are interested in reasoning incrementally In particular, given a CSP of temporal constrains and a new constraint, we want to *maintain* certain properties in the extended CSP (e.g., a solution), rather than recomputing them from scratch. The Point Algebra (PA) and the Interval Algebra (IA) are two well-known frameworks for qualitative temporal reasoning. Most of the existing algorithms for PA and the known tractable fragments of IA, such as ORD-Horn, has been designed for "static" reasoning. In this paper we study the incremental version of some fundamental problems of temporal reasoning, proposing new algorithms that amortize their complexity when processing a sequence of input constraints. After analyzing the role of path-consistency for incremental satisfiability, we propose algorithms for maintaining a solution of a CSP over either PA or ORD-Horn, and the minimal labels of a CSP over PA. Our algorithms improve the complexity of using existing techniques by a factor of $O(n)$ or $O(n^2)$, where n is the number of variables involved in the CSP.

1 Introduction

Constraint-based qualitative temporal reasoning is a widely studied area of AI in which the most prominent approaches are Allen's Interval Algebra (IA) [1983] and Vilain and Kautz's Point Algebra (PA) [1986]. Given a CSP of temporal constraints, fundamental reasoning problems are deciding satisfiability of the CSP, finding a solution, and computing its "minimal labels". These problems arc NP-hard for the full IA, while they are polynomial for PA and for several fragments of IA, such as Nebel & Burckert [19951 ORD-Horn subalgebra, which is the unique maximal tractable subclass of IA containing all the basic relations.

In many applications of temporal reasoning we are interested in "dynamic" or *on-line* reasoning. For instance, given an initial (possibly empty) CSP of temporal constraints, whenever a new constraint is asserted, we want to in *maintain* a solution of the CSP, rather than recomputing it from scratch using a "static" algorithm.

The design of algorithms for dynamic polynomial problems is an important research field in operation research and theoretical computer science (e.g., [Ausiello *et ai,* 1991]). A dynamic problem can be either *semi-dynamic* or *fully-dynamic.* In the semi-dynamic version we deal with information that is incrementally given (or decrementally retracted). In the fully-dynamic version, we deal with both assertions and retractions. Typically, the worst-case performance of an algorithm for a dynamic problem is specified in terms of its *amortized complexity* for a single operation (assertion or retraction), that can be defined as the average runtime per operation over a worst-case *sequence* of operations [Tarjan, 1985].

The large majority of the algorithms that have been developed for reasoning within tractable classes of qualitative temporal constraints are static. In this paper we investigate the incremental version of some reasoning problems for PA and ORD-Horn, discussing the behavior of existing techniques and proposing new ones. Given a sequence s of $O(n^2)$ input constraints involving n variables, our incremental algorithms improve the total runtime required for processing s using existing techniques by a factor of $O(n)$ or $O(n^2)$.

In Section 2 we give the necessary background on PA and IA. In Section 3, after analyzing the use of a path-consistency algorithm for incremental satisfiability checking, we propose new incremental algorithms for finding a solution of a CSP over either PA or ORD-Horn, and for computing the minimal labels of a CSP over PA. Finally, in Section 4 we briefly discuss the fully-dynamic version of the problems considered, and we mention further results and future work.

2 Background on PA and IA

Vilain and Kautz's Point Algebra [1986] consists of three basic relations between time points $(<, >, =)$, all possible unions of them $(\leq, \geq, \neq,$ and $?$, where ? is the universal relation), and of the empty relations $(\emptyset)$. Allen's Interval Algebra [1983] consists of thirteen basic relations between temporal intervals, all possible unions of these relations, and the empty relation. PA and IA are closed under the operations union $(\cup)$, intersection $(\cap)$, difference $(\setminus)$, converse $(^{\smile})$, and composition (o). The first four operations are defined in the standard way. The composition for two relations can be derived from the *transitivity tables* of the basic relations in PA and IA.

A *qualitative temporal constraint satisfaction problem* (or briefly *temporal CSP)* in our context is a set of constraints of the kind *xRu,* where *x* and *y* are *cither point variables* or *interval variables,* and *R* is either a PA-relation (if *x* and *y* are point variables) or an IA-relation (otherwise).

Given a temporal CSP Θ over either PA or IA, a fundamental reasoning problem is deciding the *satisfiability* of Θ. Θ is satisfiable if and only if there exists a *solution* for Θ, i.e., an assignment of temporal values to the variables of Θ (rational numbers for point variables, pairs of rational numbers for interval variables) such that all the constraints in 0 are satis-

fied. The problem of deciding the satisfiability of Θ will be called PSAT, if Θ is over PA, and ISAT if Θ is over IA. The problem of finding a solution for Θ will be called PSOL, if Θ is over PA, and ISOL otherwise.

A related reasoning problem is finding a *scenario* that refines a given temporal CSP A scenario of a CSP Θ is a satisfiable *refinement* of Θ, where the constraints between all pairs of variables are basic relations. A CSP Θ' is a refinement of Θ if and only if Θ' and Θ involve the same variables, and for every pair of variables *(x,y)* such that $xR'y \in \Theta'$ and $xRy \in \Theta$, $R' \subseteq R$.[1] Any scenario of a satisfiable temporal CSP over PA (IA) identifies a total order σ of the point (interval endpoint) variables of the CSP that is consistent with the constraints of the scenario, and in which point (interval endpoint) variables that must have the same interpretation are mapped to the same position. Clearly, from *a* we can derive a solution for the CSP by just assigning to the points (endpoints) of *a* numbers consistent with their relative order.[2]

Any temporal CSP Θ involving *n* variables can be processed using a cubic time algorithm that refines Θ to an equivalent *path consistent* CSP [Montanari, 1974]. A CSP is path consistent if for every subset of constraints involving three variables *i, j,* and *k,* the relation R_{ik} between *i* and A; is stronger or equal than (i.e., is a subset of) the composition of R_{ij} and R_{jk}.

A temporal CSP is *minimal* if, for every pair of variables i, j, the relation R_{ik} between i and *k* is the strongest relations between *i* and *k* that is entailed by the CSP, i.e., for each basic r_{ij} relation in R_{ij} there exists a scenario for the CSP in which R_{ij} is refined to f We call the problem of computing the minimal CSP PMIN for PA, and IMIN for IA.

From a computational point of view, PA and IA have different properties. The reasoning problems introduced above are polynomial for PA, while they are NP-hard for IA [Vilain and Kautz, 1986]. In particular, PSAT and PSOL can be solved in $O(n^2)$ time using van Beek's method [1992], while PMIN requires $O(n^4)$ time Ivan Beek, 1992; Gerevini and Schubert, 1995]. These techniques use a graph-based representation of the CSP that in [Gerevini and Schubert, 1995] is called *temporally labeled graph (TL-graph).* A TL-graph is a graph with a set of labeled edges, where each edge connects a pair of distinct vertices v, *w* representing the point variables of the temporal CSP. The edges are either directed and labeled $\leq$ or $<$, or undirected and labeled $\neq$ (=-constraints are represented by a pair of $\leq$-constraints), van Beek's method for finding a scenario for a satisfiable CSP of PA-constrains is based on first identifying the strongly connected components (SCC) of the TL-graph representing the CSP [Cormen *et* a/., 1990] using only its $\leq$-edges. Each SCC is then collapsed into a single vertex representing an equivalent class of variables that must be interpreted with the same temporal value. If we omit the $\neq$-edges from the resultant graph, we obtain a directed acyclic graph that we call the induced *precedence graph* of the CSP. From a topological sort of the vertices of this graph we can easily derive a scenario for the CSP.[3]

Enforcing path-consistency to a temporal CSP requires cubic time and is sufficient to decide PSAT fLadkin and Maddux, 1988], but it does not solve PMIN. In order to solve PMIN for a CSP over PA, in addition to enforce path-consistency to it, we need to identify particular 4-variable constraint subsets of the path consistent CSP containing a $\leq$-constraint that should be refined to "$<$". Such subsets are *called forbidden subgraphs* in Ivan Beek, 1992] and $\neq$-*diamonds* in [Gerevini and Schubert, 1995].

Regarding IA, several tractable fragments have been identified. The Simple Interval Algebra (SIA) [van Beek, 1992; Ladkin and Maddux, 1988] is formed by the IA-relations that can be translated into a conjunction of PA-constraints between interval endpoints. All reasoning problems for a CSP Ω over SIA can be solved by applying the corresponding algorithms to the PA-translation of Ω. The most interesting tractable fragment of IA is Nebel and Biirckert's ORD-Horn class [1995]. ORD-Horn subsumes SIA and is the unique maximal tractable sub-algebra of IA containing all the basic relations. Each constraint *C* over ORD-Horn can be translated into a set of disjunctions of PA-constraints of the form $P = q$. $P \leq q$ *or* $p \neq q$, where *p* and *q* are endpoints of intervals in C, and at most one literal is of type " = " or "$\leq$". Given a temporal CSP Ω over ORD-Horn, $\pi_1(\Omega)$ will denote the CSP of PA-constraints formed by the *unary* disjunctions (PA-constraints) in the point translation of all interval constraints in Ω.

Like PSAT, ISAT for ORD-Horn can be decided in cubic time by using a path-consistency algorithm. ISOL for ORD-Horn can be solved in square time, if the input CSP is known to be path consistent[Gerevini and Cristani, 1997], in cubic time otherwise [Ligozat, 1996].

In the rest of the paper, the incremental version of the reasoning problems for a temporal CSP is indicated by adding the prefix " I " to the name of the corresponding static problem, e.g., the incremental version of PSOL is I-PSOL. Without loss of generality, we assume that the initial temporal CSP already involves all variables, possibly constrained only by the universal relation. Moreover, our amortized complexity analysis is based on input sequences of length quadratic in the number of the variables. These assumptions are discussed in an extended version of the paper [Gerevini, 2003].

3 Incremental Tractable Reasoning

3.1 Path-consistency for I-PSAT and I-ISAT

Although van Beek's method for solving PSAT requires only quadratic time, when we consider the incremental version of this problem (I-PSAT), enforcing path-consistency turns out to be more efficient than a simple iterative application of van Beek's (static) algorithm. In fact, we can use a path-consistency algorithm like the one given in Figure 1 (PC), which was proposed in [Allen, 1983] and slightly reformulated and improved in [Vilain and Kautz, 1986; Bessiere, 1996], to incrementally process a sequence of $O(n^2)$ PA-

Without loss of generality we assume that, if no information between *x* and *y* is provided, *R* is the *universal relation,* and that for every pair of variables (x, *y)* such that $xRy \in \Theta, yR^{\smile}x \in \Theta$.

[2]E.g., we assign an integer *i* to the points in the first position of o, *i* + 1 to the points in the second position, etc. If we have a solution, we can derive a total order (and hence a scenario) in a trivial way.

[3]A topological sort for a DAG is a linear order *a* of its vertices such that if *v* is a successor of *w* in the graph, then *v* precedes *win a.*

Algorithm: PC
Input: the matrix representation M of a temporal CSP over either PA or IA.
Output: `fail`, if enforcing path-consistency to the input CSP generates the empty relation; the matrix representation of a path consistent CSP equivalent to the input CSP, otherwise.

1. $Q := \{(i,j) \mid i < j\}$
2. **while** $Q \neq \emptyset$ **do**
3. select and delete an element from Q;
4. **for** $k \neq i, k \neq j$ **do**
5. **if** REVISE(i,j,k) **then**
6. **if** $M[i,k] = \emptyset$ **then return** `fail`
7. **else** add (i,k) to the end of Q;
8. **if** REVISE(k,i,j) **then**
9. **if** $M[k,j] = \emptyset$ **then return** `fail`
10. **else** add (k,j) to the end of Q;
11. **return** M.

Subroutine: REVISE(i,k,j)
1. **if** $M[i,k]$ or $M[k,j]$ is the universal relation **then return** `false`;
2. $S := M[i,k] \circ M[k,j]$;
3. **if** $M[i,j] \subseteq S$ **then return** `false`
4. $M[i,j] := M[i,j] \cap S$; $M[j,i] := M[i,j]^{\smile}$;
5. **return** `true`;

Figure 1: A path-consistency algorithm for qualitative temporal constraints as formulated in [Bessière, 1996].

constraints in $O(n^3)$ total time (i.e., the amortized time complexity per input constraint is linear).This is better than reapplying van Beek's algorithm after each constraint assertion, which requires $O(n^4)$ total time. Similarly, for any fragment of IA for which enforcing path-consistency is sufficient to decide satisfiability, we can use PC to solve I-ISAT in $O(n^3)$ total time, instead of $O(n^5)$. More precisely, let $\langle C_1, C_2, ..., C_k \rangle$ be the sequence of the input constraints, Θ_0 an initial (possibly empty) satisfiable temporal CSP, and $\Theta_i = \Theta_0 \cup \{C_j \mid j \leq i\}$ $(i > 0)$.[4] To decide the satisfiability of each Θ_i, we consider three cases:

(1) if the relation R of $C_i = xRy$ is weaker than the relation R' of the corresponding constraint in Θ_{i-1}, then clearly Θ_i is satisfiable, and no further processing is needed;
(2) if $R \cap R'$ is the empty relation, then Θ_i is unsatisfiable and we return `fail`;
(3) if none of the previous cases holds, then, first we revise the relation between x and y to $R' \cap R$ (and between y and x to $R'^{\smile} \cap R^{\smile}$), and then we run PC on the resultant revised CSP with step 1 modified so that Q initially contains only the item (x,y).

We call this simple method *incremental path-consistency* (IPC). This technique has the following important property, that will be exploited by the algorithms in the next sections.[5]

Theorem 1 *IPC solves* I-PSAT *and* I-ISAT *for ORD-Horn in* $O(n^3)$ *total time, where* n *is the number of temporal variables.*

Proof. Since a pair (x,y) enters into Q only when the relations between x and y is revised, and each relation R can be revised at most twice, if $R \in$ PA, or at most 12 times, if $R \in$ IA, it follows that the total time complexity is $O(n^3)$. □

It is worth noting that, as pointed out by Bessière [Personal Communication, 1997], the $O(n^3)$ time complexity of Ligozat's algorithm [1996] for finding a scenario of a path consistent CSP over ORD-Horn follows from Theorem 1.

3.2 An Algorithm for I-PSOL

For the incremental version of PSOL (I-PSOL) the simple application from scratch of van Beek's static algorithm [1992] after each constraint assertion requires $O(n^4)$ total time.[6]

In this section we propose an algorithm for solving I-PSOL in $O(n)$ amortized time, i.e., the total time complexity for processing a sequence on $O(n^2)$ constraint assertions is $O(n^3)$. The algorithm is based on maintaining a topological sort S for the precedence graph of the input CSP in which the constraints are incrementally given. Let S_{i-1} be a topological sort for Σ_{i-1} (Σ_{i-1} is satisfiable). When we process a new constraint C_i we perform two main operations:

(i) we check whether $\Sigma_i = \Sigma_{i-i} \cup \{C_i\}$ is satisfiable;
(ii) if Σ_i is satisfiable, we update S_{i-1} to derive S_i.

As we have seen, (i) can be solved in $O(n^3)$ total time by using IPC. Regarding (ii), there are two related difficulties to address: C_i induces an edge (or precedence constraint) in the precedence graph that invalidates S_{i-1}; if the relation of C_i is "=" or "$\leq$", new equalities may be generated (i.e., in the TL-graph representation a new SCC may be generated, and the precedence graph should be revised accordingly). The algorithm INCREMENTAL-PA-SOL given in Figure 2 accomplishes (i) and (ii) taking these cases into account. The rest of this section is devoted to the description of the algorithm.

The current path consistent CSP is represented by a matrix M, while the current topological sort is represented by an array T such that $T[j] = v$ if and only if v is the j-th element of the sort. $Ord(v)$ denotes the position of v in the topological sort (i.e., the index of v in T). INCREMENTAL-PA-SOL uses a modification of IPC, called IPC-1, in which each variable has a flag (EQ) which is set to `false` before running INCREMENTAL-PA-SOL. The EQ-flag of a variable v is revised to `true` if the relation of the new input constraint is "=" and v is one of its variables, or if during the propagation of the new input constraint, the relation between v and *any* other variable is revised to "=". The second case is handled in IPC-1 by using an extension of the subroutine REVISE in which, if $M[i,j]$ is revised to "=", then $EQ[i]$ and $EQ[j]$ are set to `true`. Clearly, this modification does not increase the complexity of the incremental path-consistency method.

If the relation of the new PA-constraint xRy is in $\{<, \leq\}$, then xRy may induce a precedence constraint $x \prec y$ for S_i that is not satisfied in S_{i-1}. (Similarly for $R \in \{>, \geq\}$ inducing $y \prec x$.) This is the case when Σ_i is satisfiable, $Ord(x) > Ord(y)$ (or $Ord(y) > Ord(x)$), and xRy induces no new equality in Σ_i (i.e., the set E of step 3 is empty).

[4] Θ_0 may already contain up to $O(n^2)$ constraints different from the universal relation. For CSPs over PA $k \leq 6 \cdot (n^2 - n)/2$, because we assume that there is no duplication in the input sequence, and there are at most 6 different input PA-constraints between any pair of variables (the empty and the universal relations are not valid input relations). Similarly, for CSPs over IA $k \leq (2^{13} - 2) \cdot (n^2 - n)/2$.

[5] For lack of space we omit proof details, which are available in [Gerevini, 2003].

[6] A temporal CSP over PA may involve $O(n^2)$ constraints with relation different from "?", and each of these can be revised at most twice by additional input constraints.

Algorithm: INCREMENTAL-PA-SOL
Input: the matrix representation M of the path consistent CSP of Σ_{i-1}; a topological order T for the precedence graph G of Σ_{i-1}; a new PA-constraint xRy.
Output: A solution for $\Sigma_i = \Sigma_{i-1} \cup \{xRy\}$, if Σ_i is satisfiable, `fail` otherwise.

1. M := IPC-1(M, xRy);
2. **if** M = `fail` **then return** `fail`;
3. E := set of the point variables x such that $EQ[x]$ = `true` (E is a global variable used also in UPDATE-TSORT);
4. **if** E is empty **then**
5. **if** $R \in \{<, \leq\}$ and $Ord(x) > Ord(y)$ **then**
6. UPDATE-TSORT(T,x,y);
7. **if** $R \in \{>, \geq\}$ and $Ord(y) > Ord(x)$ **then**
8. UPDATE-TSORT(T,y,x)
9. **else** collapse the vertices of G in E to x;
10. $S := \{v \mid v \notin E - \{x\} \wedge Ord(y) \leq Ord(v) \leq Ord(x)\}$;
11. L := list of the variables in S topologically sorted using G;
12. **for** $j := Ord(y)$ **to** $Ord(x) - |S| + 1$ **do**
13. $T[j] := Pop(L)$; $Ord(T[j]) := j$;
14. **for** $j := Ord(x) - |E| + 2$ **to** $V(G)$ **do**
15. $T[j] := T[j + |E| - 1]$; $Ord[T[j]] := j$;
16. **return** an assignment to the variables in Σ_i consistent with T.

Subroutine: UPDATE-TSORT(T, x, y)
(The topological sort T for Σ_{i-1} is updated using the new precedence constraint $x \prec y$, and it becomes a topological sort for Σ_i.)

1. *shift* := 1;
2. L := a list containing only y;
3. **for** $i := Ord(y) + 1$ **to** $Ord(x)$ **do**
4. **if** $M[y, T[i]] \in \{<, \leq\}$ **then**
5. add $T[i]$ to the end of L;
6. *shift* := *shift* + 1;
7. **else** $Ord(T[i]) := i - shift$;
8. $T[i - shift] := T[i]$;
9. **for** $i := Ord(x) - shift + 1$ **to** $Ord(x)$ **do**
10. $z := Pop(L)$; $T[i] := z$; $Ord(z) := i$;

Figure 2: Algorithm for I-PSOL. $V(G)$ denotes the current number of vertices in the precedence graph G. $Pop(L)$ returns the first element of L and removes it from L.

Under these conditions, the new constraint requires to update S_{i-1}, which is accomplished by steps 4–8. Specifically, if the induced precedence constraint is $x \prec y$, we run the subroutine UPDATE-TSORT(T, x, y), which revises the current topological sort (T) in the following way.[7]

UPDATE-TSORT considers each temporal variable v such that $Ord(y) \leq Ord(v) \leq Ord(x)$. (Notice that the only part of S_{i-1} than can be invalidated by $x \prec y$ regards the variables from $T[Ord(y)]$ to $T[Ord(x)]$.) If Σ_i entails $y \leq v$, then in S_i y (and hence also x) must precede v, because in the precedence graph there is a path from y (x) to v. Since by step 1 of the main algorithm M is the path consistent CSP of Σ_i, these entailment can be checked in constant time by simply looking up at the appropriate entries of M (step 4 of the subroutine). The v-variables that must follow x are added to the end of a list L that initially contains only y. Once all variables in the relevant part of S_{i-1} have been considered, those that are in L should be postponed, while the others can

[7]UPDATE-TSORT is similar to a technique used by Marchetti *el ai* [1993] in the context of an algorithm for detecting cycles in a DAG.

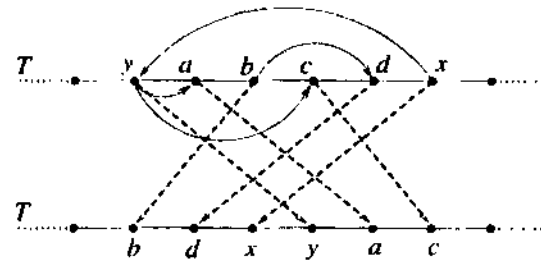

Figure 3: Example of an update of the topological order T with $x \prec y$ using UPDATE-TSORT(x, y, T). Directed solid arcs represent edges in the precedence graph of Σ_i. The variables preceding x in the current topological sort, and that in the revised sort must follow it, are y, a and c. When step 10 is executed the values of *shift* and L are 3 and (y a c), respectively.

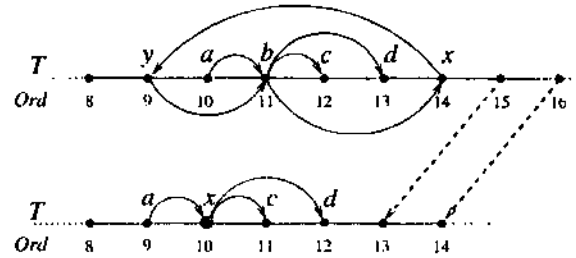

Figure 4: Example of revision of the topological sort T when a set E of variables become equal by adding $x \leq y$ to Σ_{i-1} ($E = \{x, y, b\}$). Directed solid arcs represent edges in the precedence graph of Σ_i before and after collapsing the variables in E to x. Dashed arrows indicate the shift performed by steps 14-15 of INCREMENTAL-PA-SOL.

be anticipated. In particular, if a variable is not added to L, we can shift it backward a number of positions equal to the number of items currently in L (which is given by the value of *shift).* This is done by steps 3-8 of the subroutine, while steps 9-10 move forward the variables in L to the positions of T that were occupied by the variables shifted backward. Notice that when UPDATE-TSORT(T,X,;U) terminates, the relative order of the variables that entered into L, as well as of those that were shifted backward, are not changed in the revised topological sort. Figure 3 shows a simple example.

If xRy induces new equalities *(E* is not empty), then the variables in *E,* together with those possibly represented by them, form an equivalence class. Step 9 collapses the vertices in the precedence graph G of Σ_{i-1} corresponding to the variables in E into a single representative vertex ;r, transferring to x all incoming and outgoing edges of the collapsed vertices. In order to update T, we need now to reorder the variables from $T[Ord(y)]$ to $T[Ord(x)]$ that have not been collapsed. These variables are topologically ordered using G by step 11, and assigned to T in positions from $Ord(y)$ to $Ord(x) - |E| + 1$. Then, steps 13 and 14 *shrinks T* by shifting backward the variables appearing after x in T, starting from position $Ord(x) - |E| + 2$ and with a shift equal to the number of variables (vertices) eliminated from G (i.e., $|E| - 1$). Figure 4 gives an example illustrating steps 9-15.

Theorem 2 INCREMENTAL-PA-SOL *solves* I-PSOL *in* $O(n^3)$ *total time, where n is the number of temporal variables.*

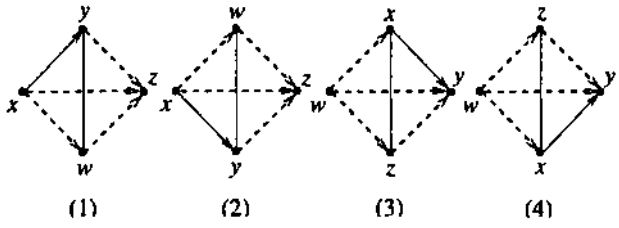

Figure 5: Types $\neq$-diamonds that can be generated in $\hat{\Sigma}_i$ by the new constraint $x < y$. Directed (undirected) edges represent $\leq$-constraints ($\neq$-constraints). In the minimal CSP of Σ_i, for cases (1)-(2) R_{xz} should be $<$, for cases (3)-(4) R_{wy} should be "$<$".

Proof (sketch). Correctness of the algorithm follows from Theorem 1 and the discussion in the description of the algorithm. By Theorem 1 step 1 requires $O(n^3)$ total time. The other most costly steps are 9, 11 requiring $O(n^2)$ time. They are performed at most $O(n)$ times for any input sequence. □

3.3 An Algorithm for I-PMIN

The simple use of van Beek's (static) algorithm for PMIN after each constraint assertion to solve I-PMIN requires $O(n^6)$ total time. In this section we propose an algorithm for solving I-PMIN, INCREMENTAL-PA-MIN, requir$O(n^4)$ total time. This is the best possible worst-case complexity bound.

In order to maintain a CSP over PA minimal when a new PA-constraint xRy is given, in addition to enforce path-consistency, we need to identify and remove all possibly $\neq$-diamonds that are generated (see Figure 5). Since $\neq$-diamonds consist only of constraints of type $\leq$ and $\neq$, clearly a new constraint can generate additional $\neq$-diamonds only if $R \cap R' \in \{=, \leq, \geq, \neq\}$, where R' is the relation between x and y in the minimal CSP of Σ_{i-1}.

Let $\overline{\Sigma}_{i-1}$ be minimal CSP of $\sum_{i}$_i, and $\sum_i$ of the path consistent CSP of $\overline{\Sigma}_{i-1} \cup \{xRy\}$. If $R \cap R' \in \{=, \leq, \geq\}$ (and $R \subset R'$), then the new constraint and its propagation can generate new $\leq$-constraints in $\hat{\Sigma}_i$. It is easy to see that any *new* possible $\neq$-diamonds in $\hat{\Sigma}_i$ must involve one of these new $\leq$-constraints. Hence, in order to make $\hat{\Sigma}_i$ minimal, it suffices to identify and eliminate all diamonds involving one of these constraints. Figure 5 shows the only four possible types of $\neq$-diamonds that can be generated in $\hat{\Sigma}_i$ by the new constraint $x \leq y$. Note that cases (1) and (2) are symmetric and can be reduced to the same case (analogously for (3) and (4)).

If $R \cap R' =$ "$\neq$" (and $R \subset R'$), then the propagation of xRy cannot give rise to new $\neq$-constraints, and hence in order to make $\hat{\Sigma}_i$ minimal, it suffices to identify and eliminate all new diamonds involving $x \neq y$.

Our algorithm for solving I-PMIN is given in Figure 6. It uses a modification of IPC, called 1PC-2, which like IPC-I is a revision of IPC to keep track of certain constraints generated by the propagation of *xRy*. IPC-2 adds all pairs of variables whose relation has been refined to "$\leq$" to a list *L*. *L* is a global variable that is initialized by steps 1-2 of INCREMENTAL-PA-MIN to the list containing either *(x,y), (y, x)* or no pair. A pair of variables is added to *L* by the subroutine REVISE-2 of IPC-2, a modification of REVISE in which after step 4 we perform the following additional steps to update *L:*

Algorithm: INCREMENTAL-PA-MIN
Input: the matrix representation M of the minimal CSP of Σ_{i-1}; a new PA-constraint xRy.
Output: The matrix representation of the minimal CSP for $\Sigma_i = \Sigma_{i-1} \cup \{xRy\}$, if Σ_i is satisfiable, `fail` otherwise.

```
                = "≤" ⊂ M[x, y] then L := ((x, y)) else L := ();
2.                M x, y] then L := ((y, x)) else L := ();
3. if R ∩ M[      = "≠" ⊂ M x, y] then N:=true else N:=false;
4. M := IPC-2(M, xRy);
5. if M = fail then return fail;
6. for each (u, v) ∈ L do
7.    for each w, z such that w ≠ z and w, z ≠ u, v do
8.      if M[u, w] = M[v, z] = M[w, z] = "≤" ∧ M[v, w] = "≠"
9.          then M[u, z] := "<"; M[z, u] := ">";
10.     if M[w, u] = M[w, z] = M[z, v] = "≤" ∧ M[u, z] = "≠"
11.         then M[w, v] := "<"; M[v, w] := ">";
12.if N then
13    for each w, z such that w ≠ z and w, z ≠ x, y do
14.     if M[w, x] = M[w, y] = M[x, z] = M[y, z] = "≤"
15.         then M[w, z] := "<"; M[z, w] := ">";
16.return M.
```

Figure 6: Algorithm for solving I-PMIN. L is a global variable that is modified by IPC-2 as described in the text.

5. **if** $M[i, j] =$ "$\leq$" **then** add (i, j) to L
6. **else if** $M[j, i] =$ "$\leq$" **then** add (j, i) to L

Step 4 of INCREMENTAL-PA-MIN runs IPC-2 to (a) decide PSAT for Σ_i, (b) enforce path-consistency to M, and (c) update L with the new implied $\leq$-constraints. If Σ_i is satisfiable, then steps 6–11 identify all new $\neq$-diamonds that are generated in M by the new constraint $x \leq y$. These diamonds are eliminated by revising to "$<$" or "$>$" the relation of the "non-minimal" constraints (steps 9 & 11). Finally, steps 13–15 eliminate all new $\neq$-diamonds involving $x \neq y$.

Theorem 3 INCREMENTAL-PA-MIN *solves* I-PMIN *in* $O(n^4)$ *total time, where n is the number of temporal variables.*

Proof (sketch). Since the algorithm eliminates all new $\neq$-diamonds, correctness follows from Theorem 1 and fact that any path consistent CSP over PA with no $\neq$-diamond is minimal [Gerevini and Schubert, 1995] Complexity follows from Theorem 1 and the fact that the total number of pairs that can enter into *L* is $O(n^2)$. □

3.4 An Algorithm for I-ISOL over ORD-Horn

Gerevini and Cristani [1997] showed that ISOL for a path consistent CSP Ω over ORD-Horn can be solved in $O(n^2)$ time by using a solution for the π_1-translation of Ω. Figure 7 gives an algorithm solving I-ISOL for Ω which exploits this property and uses INCREMENTAL-PA-SOL. Our algorithm requires $O(n^3)$ total time, while the application from sketch of a known static algorithm each time a new constraint is asserted requires $O(n^5)$ total time. This cost can be reduced to $O(n^4)$ total time by using IPC combined with Gerevini & Cristani's algorithm, which is still one order worse than the complexity of the algorithm presented here.

Our algorithm uses a modification of IPC, called IPC-3, which again is a revision of IPC to keep track of certain constraints generated by the propagation of the new input constraint xRy. Given the path consistent C S $\hat{\Omega}$ o f Ω_{i-1}, the

Algorithm: INCREMENTAL-ORDHORN-SOL
Input: the matrices M and M' representing the path consistent CSP $\hat{\Omega}$ of Ω_{i-1}, and the path consistent CSP of $\pi_1(\hat{\Omega})$, respectively; a topological order T for the precedence graph of $\pi_1(\hat{\Omega})$; a new interval constraint xRy.
Output: A solution for Ω_i, if Ω_i is satisfiable, `fail` otherwise.

1. **if** $(R \cap M[x,y]) \subset M[x,y]$ **then**
2. $P :=$ list formed by the constraints in $\pi_1(\{xRy\})$ **else** $P := ()$;
3. $M :=$ IPC-3(M, xRy);
4. **if** $M =$ `fail` **then return** `fail`;
5. **for each** $pSq \in P$ **do**
6. **if** $(S \cap M'[p,q]) \subset M'[p,q]$ **then** initialize the EQ-flags and
7. run steps 1–15 of INCREMENTAL-PA-SOL(M', T, pSq);
8. **return** an assignment to the interval endpoints in Ω_i consistent with T.

Figure 7: An algorithm solving I-ISOL for ORD-Horn.

complexity only for any *subsequence* of constraint assertions. E.g., consider an instance of the fully-dynamic version of PMIN in which we have k retractions during an input sequence of $O(n^2)$ assertions. When we process a retraction, we can recompute from scratch the minimal labels, while assertions are processed using our algorithm. The total time complexity is then $O(kn^4)$, which is still more efficient than applying van Beek's static $O(n^4)$ algorithm $O(kn^2)$ times.

While in this paper we have not addressed I-IMIN for ORD-Horn, very recently we have proved that this problem can be solved in $O(n^6)$ total time. Future work includes incremental algorithms for other tractable classes [Krokhin *et al,* to appear] and a deeper study of fully-dynamic problems.

References

[Allen, 1983] J.F.Allen. Maintaining knowledge about temporal intervals. *Comm. of the ACM,* 26(I):832-843,1983.

[Ausiello *et ai,* 1991] G. Ausiello, G. Italiano, A.Marchetti-Spaccamela, and U. Nanni. Incremental algorithms for minimal length paths. *J. of Algorithms,* 12:615-638,1991.

[Bessiere, 1996] C. Bessiere. A simple way to improve path-consistency in Interval Algebra networks. *Proc. of AAAI-*96,375-380,1996.

[Cormen etal., 1990] T. Cormen, C. Leiserson, and R. Rivest. *Introduction to Algorithms.* The MIT Press, 1990.

[Gerevini, 2003] A. Gerevini. Incremental Tractable Reasoning about Qualitative Temporal Constraints. *Technical Report,* DEA, University of Brescia, Italy, 2003.

[Gerevini and Cristani, 1997] A. Gerevini and M. Cristani. On finding solutions in temporal constraint networks. *Proc. of IJCAI-97,* 1460-1465, 1997.

[Gerevini and Schubert, 1995] A. Gerevini and L. Schubert. On computing the minimal labels in time point algebra networks. *Computational Intelligence,* 11(3):443-448, 1995.

[Krokhin et a/., to appear] A. Krokhin, P. Jeavons, and P. Jonsson. The Tractable Subalgebras of Allen's Interval Algebra. *J. of ACM,* to appear.

[Ladkin and Maddux, 1988] P. Ladkin, and R.Maddux. On binary constraint networks. TR, Kestrel Institute, 1988.

[Ligozat, 1996] G. Ligozat. A new proof of tractability for ORD-Horn relations. *Proc. of AAAI-96,* 715-720, 1996.

[Marchetti et a/., 1993] A. Marchetti-Spaccamela, U. Nanni, and H. Rohert. On-line graph algorithms for incremental compilation. LNCS 790, 113-151. Springer Verlag, 1993.

[Montanari, 1974] U. Montanari. Networks of constraints: Fundamental properties and applications to picture processing. *Information Science,* 7(3):95-132, 1974.

[Nebel and Burckert, 1995] B. Nebel and H. Burckert. Reasoning about temporal relations: A maximal tractable subclass of Allen's interval algebra. *J. of ACM,* 42(1), 1995.

[Tarjan, 1985] R.E. Tarjan. Amortized computational complexity. *SIAM J. of Alg. and Dis. Meth.,* 6:306-318, 1985.

[van Beek, 1992] P. van Beek. Reasoning about qualitative temporal information. *Artif Intell,* 58, 297-321, 1992.

[Vilain and Kautz, 1986] M. Vilain and H.A. Kautz. Constraint propagation algorithms for temporal reasoning. *Proc. ofAAAI-86,* 377-382,1986.