Tractable Classes of Metric Temporal Problems with Domain Rules

T. K. Satish Kumar
Computer Science Division
University of California, Berkeley
tksk@eecs.berkeley.edu

Abstract

In this paper, we will deal with some important kinds of metric temporal reasoning problems that arise in many real-life situations. In particular, events $X_0, X_1 \ldots X_N$ are modeled as time points, and a constraint between the execution times of two events X_i and X_j is either *simple temporal* (of the form $X_i - X_j \in [a, b]$), or has a *connected* feasible region that can be expressed using a finite set of *domain rules* each in turn of the form $X_i \in [a, b] \rightarrow X_j \in [c, d]$ (and conversely $X_j \in [e, f] \rightarrow X_i \in [g, h]$). We argue that such rules are useful in capturing important kinds of non-monotonic relationships between the execution times of events when they are governed by potentially complex (external) factors. Our polynomial-time (deterministic and randomized) algorithms for solving such problems therefore enable us to efficiently deal with very expressive representations of time.

Introduction

Efficient algorithms for solving problems involving rich representations of time are crucial to many applications in AI. Several tasks in planning and scheduling, for example, involve reasoning about temporal constraints between actions and propositions in partial plans (Smith *et al* 2000). These tasks may include threat resolution in partial order planning, analyzing resource consumption envelopes to guide the search for a good plan (see (Muscettola 2002) and (Kumar 2003)), etc. Among the important formalisms used for reasoning with metric time are *simple temporal problems* (STPs) and *disjunctive temporal problems* (DTPs).

Unlike DTPs, STPs can be solved in polynomial time, but are not as expressive as DTPs. An STP is characterized by a graph $G = \langle \mathcal{X}, \mathcal{E} \rangle$, where $\mathcal{X} = \{X_0, X_1 \ldots X_N\}$ is a set of events (X_0 is the "beginning of the world" node and is set to 0 by convention), and $e = \langle X_i, X_j \rangle \in \mathcal{E}$, annotated with the bounds $[LB(e), UB(e)]$, is a *simple temporal constraint* between X_i and X_j indicating that X_j must be scheduled between $LB(e)$ and $UB(e)$ seconds after X_i is scheduled ($LB(e) \leq UB(e)$).

DTPs are significantly more expressive than STPs, and allow for disjunctive constraints. The general form of a DTP is as follows. We are given a set of events $\mathcal{X} = \{X_0, X_1 \ldots X_N\}$ (X_0 is the "beginning of the world" node and is set to 0 by convention), and a set of constraints $\mathcal{C}$. A constraint $c_i \in \mathcal{C}$ is a disjunction of the form $s_{(i,1)} \vee s_{(i,2)} \ldots s_{(i,T_i)}$. Here, $s_{(i,j)}$ $(1 \leq j \leq T_i)$ is a simple temporal constraint of the form $L_{(i,j)} \leq X_{b_{(i,j)}} - X_{a_{(i,j)}} \leq U_{(i,j)}$ for $0 \leq a_{(i,j)}, b_{(i,j)} \leq N$.

Although DTPs are expressive enough to capture many tasks in planning and scheduling (like threat resolution and plan merging), they require an exponential search space. The principal approach taken to solve DTPs has been to convert the original problem to one of selecting a disjunct from each constraint, and then checking that the set of selected disjuncts forms a consistent STP. Checking the consistency of, and finding a solution to an STP can be performed in polynomial time using shortest path computations. The computational complexity of solving a DTP comes from the

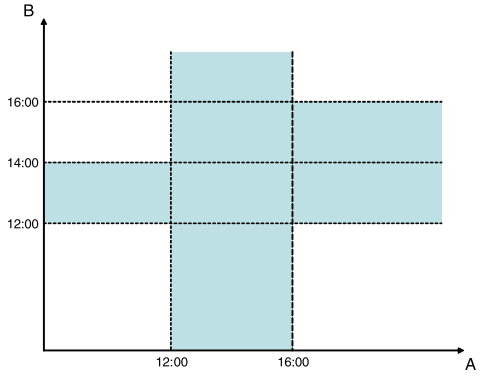

Figure 1: Illustrates a rover-scenario in which the execution times of two events (beginning times of tasks A and B) are constrained by non-monotonic relationships. If we run A before noon (when solar energy is not in abundance, and battery power has to be used), then we can run B only during a specific time—between noon and 2:00pm—when solar energy is abundant. However, if we run A between noon and 4:00pm (when solar energy is readily available), then B can be run at pretty much any time (either on solar energy or battery power). Similarly, if we run A after 4:00pm (possibly using the battery power built up to a certain maximum during the day and perhaps also drained for other purposes), then B can be executed only during a specific window of time—namely between noon and 4:00pm (so that at least a fair amount of battery power and/or solar energy would still be available). The figure illustrates the feasible region of such a constraint.

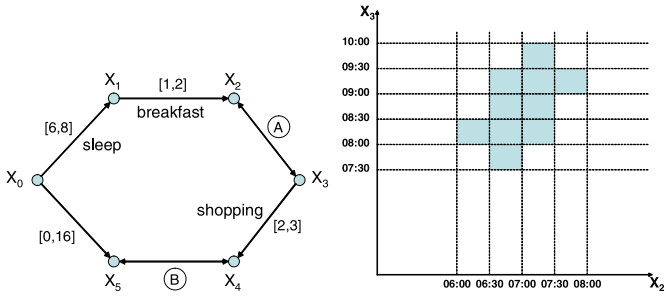

Figure 2: Presents an elaborate real-life example about an agent having to plan her day's schedules. Suppose X_0 corresponds to 12:00am. The agent chooses to sleep between 6 and 8 hours, and have breakfast for anytime between 1 and 2 hours. After this, she has to go to the shopping store; shop there for anytime between 2 and 3 hours; and come back home before 4:00pm. The constraint (between X_2 and X_3) describing her travel options between home and the shopping store is best described by the set of domain rules as in the accompanying diagram. If the agent starts out between 6:00am and 6:30am, then she can accompany a friend going to the same destination in his/her car and reach her destination anytime between 8:00am and 8:30am. However, between 6:30am and 7:30am, the only option she has is to take the city bus. If she takes the bus between 6:30am and 7:00am, she can reach her destination anytime between 7:30am and 9:30am—with the large uncertainty coming from the (un)availability of an express bus and/or external traffic-related factors. Similarly, if she takes the bus between 7:00am and 7:30am, she can get to her destination anytime between 8:00am and 10:00am. Finally, if she is willing to wait until 7:30am and possibly spend more money, she can take an express train to reach her destination anytime between 9:00am and 9:30am. Her travel options between the shopping store and back home (constraint between X_4 and X_5) can be similarly argued.

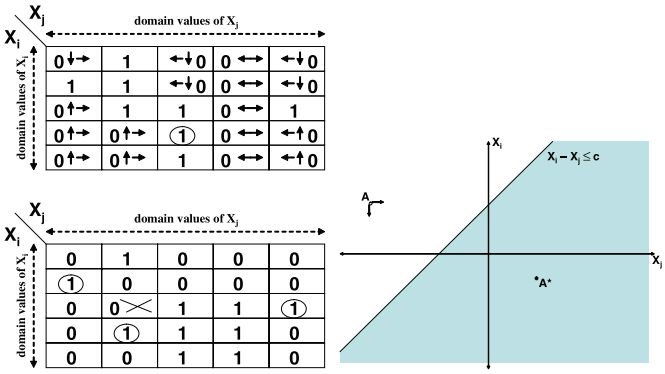

Figure 3: Illustrates the notion of a smooth constraint (left side). The right side illustrates the fact that a simple temporal constraint is smooth. At any infeasible point (A) there exist two directions such that moving along at least one of them (by a tiny amount) decreases the L_1-distance to the solution (A^*), no matter where it is placed in the feasible region of the constraint.

fact that there are an exponentially large number of disjunct combinations possible. This "disjunct selection problem" can also be cast as a constraint satisfaction problem (CSP), or a satisfiability problem (SAT) and solved using standard search techniques applicable for them (see (Tsamardinos and Pollack 2003) and (Armando *et al* 2004)).

In (Kumar 2005a), a tractable class of DTPs called *restricted DTPs* (RDTPs) is identified. In RDTPs, any $c_i \in \mathcal{C}$ is restricted to be of one of the following types: (Type 1) $(L \leq X_b - X_a \leq U)$, (Type 2) $(L_1 \leq X_a \leq U_1) \vee (L_2 \leq X_a \leq U_2) \ldots (L_{T_i} \leq X_a \leq U_{T_i})$, or (Type 3) $(L_1 \leq X_a \leq U_1) \vee (L_2 \leq X_b \leq U_2)$. RDTPs are amenable to very simple polynomial-time algorithms (Kumar 2005a).

In this paper, however, instead of dealing with disjunctions explicitly, we will deal with important kinds of metric temporal reasoning problems where the direct constraint between the execution times of two events X_i and X_j is itself more naturally expressive. In particular, the events $X_0, X_1 \ldots X_N$ are modeled as time points, and a constraint between the execution times of two events X_i and X_j is either simple temporal or has a connected feasible region that can be expressed using a finite set of domain rules each in turn of the form $X_i \in [a, b] \rightarrow X_j \in [c, d]$ (and conversely $X_j \in [e, f] \rightarrow X_i \in [g, h]$). We argue that such rules are useful in capturing important kinds of non-monotonic relationships between the execution times of events when they are governed by potentially complex (external) factors. Our polynomial-time (deterministic and randomized) algorithms for solving such problems therefore enable us to efficiently deal with very expressive representations of time.

We note that although a rule of the form $X_i \in [a, b] \rightarrow X_j \in [c, d]$ is equivalent to the disjunction $(X_i < a) \vee (X_i > b) \vee (X_j \in [c, d])$, it is not of a type allowed by RDTPs. Further, treating the disjunctions explicitly will result in a DTP which requires an exponential search space. The key idea in this paper is to show that although each rule considered *individually* is equivalent to a disjunction of the above mentioned kind, when all the rules are treated *together*, they can be made amenable to very simple polynomial-time (deterministic and randomized) algorithms.

Figure 1 shows an example of the kinds of constraints expressible by domain rules (that are going to be dealt with in this paper). In particular, it illustrates a rover scenario in which the execution times of two events (beginning times of two tasks A and B respectively) are constrained by non-monotonic relationships. (Note that the problem is allowed to contain other such constraints or even other simple temporal constraints.) Non-monotonic relationships between the execution times of events occur fairly naturally in many problem domains, and are mostly a result of the effects of complex (external) factors like the presence of reservoirs of resources (e.g. battery power and sunlight), presence of multiple options (see next example), etc. Such non-monotonic relationships can very often be appropriately expressed using a finite set of domain rules, but cannot be captured by simple temporal constraints alone. Figure 2 presents a more elaborate real-life example.

Smoothness and Temporal Problems

In (Kumar 2005b), a theory of random walks is shown to be useful in identifying tractable classes of constraints referred to as *smooth constraints*. A binary constraint between X_i and X_j (under an ordering of their domain values) is said to

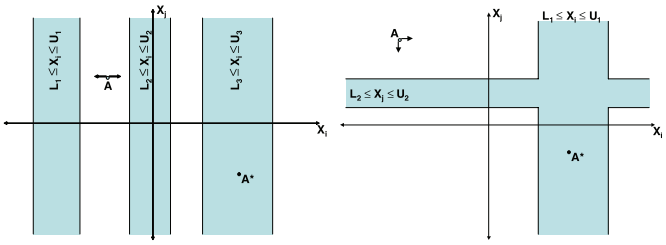

Figure 4: Two other kinds of constraints that are smooth. The left and right sides of the figure respectively correspond to Type 2 and Type 3 disjunctions in RDTPs. In both cases, the property of smoothness is satisfied: namely, at any infeasible point (A) there exist two directions such that moving along at least one of them (by a tiny amount) decreases the L_1-distance to the solution (A^*), no matter where it is placed in the feasible region of the constraint.

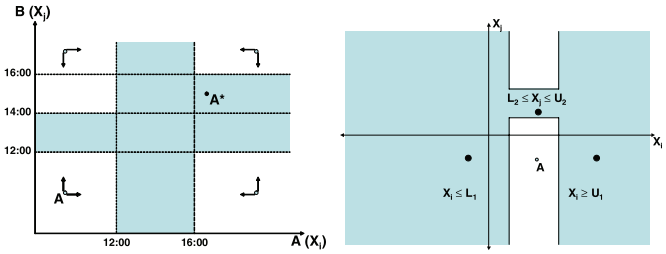

Figure 5: Shows that although the geometry of a constraint between X_i and X_j is smooth when defined by domain rules (left side), each rule by itself is not necessarily smooth (right side).

be *smooth* if the following property is true of its matrix representation:[1] "At every '0', there exist two directions such that with respect to every other '1' in the matrix, moving along at least one of these two directions (by 1 unit) decreases the Manhattan distance to it".

Figure 3 (left side) illustrates the notion of smoothness. One of the constraints shown is smooth, and the other is not. In the first case (top), every '0' is marked with two directions—indicating that no matter which '1' we have in mind (encircled in figure), moving along at least one of these two directions decreases the Manhattan distance between the '0' and the '1'. In other words, if we randomly choose to move in one of these two directions, we decrease the Manhattan distance between the '1' and the '0' by at least 1 with a probability at least 0.5, and increase it by at most 1 with a probability at most 0.5. In the second case (bottom), no such pair of directions exists for one of the '0's (marked with a cross), and it is therefore not smooth.

Simple randomized algorithms for solving smooth constraints are presented in (Kumar 2005b). Roughly, the idea is to start out with an initial random assignment to all the variables from their respective domains, and use the violated constraints in every iteration to guide the search for the satisfying assignment A^* (if it exists). In particular, in every iteration, a violated constraint is chosen, and the rank of the assignment of one of the participating variables (according to the domain orderings) is either increased or decreased. Since we know that the true assignment A^* satisfies all constraints, and therefore the chosen one too, randomly moving along one of the two directions associated with the '0' corresponding to the current assignment A will reduce the L_1-distance to A^* with a probability ≥ 0.5. Much like in a random walk on a linear graph, therefore, we can bound the convergence time to A^* by a quantity that is only quadratic in the maximum L_1-distance between any two complete assignments (which in turn is ND where N is the number of variables, and D is the size of the largest domain); see (Kumar 2005c) for more details about this algorithm.

In (Kumar 2005a), smoothness was observed to occur naturally in some temporal reasoning problems. In particular, as Figure 3 shows, simple temporal constraints are smooth, although they are not the only kinds of smooth constraints. Figure 4 shows that Type 2 and Type 3 disjunctions as allowed in RDTPs are also smooth. Therefore, it is conceivable that one simple way of solving RDTPs is to start out with an initial random assignment to all the variables, identify a violated constraint in the current iteration, and move in one of the two associated directions by an amount δ. However, the problem with this algorithm is in large because we have to deal with continuous domains. In particular, the projections of the current assignment A and the satisfying assignment A^* onto a violated constraint could be arbitrarily close to each other. In such a case, moving by an amount δ may overshoot A^* for any fixed δ. To address this problem and regain strongly polynomial running times, the random walk is not conducted over assignments to the variables, but on the disjuncts of an RDTP—which works directly on the associated *distance graph* (Kumar 2005a).

In the context of dealing with domain rules, we first note that when each domain rule is treated *independently*, it does not yield a smooth constraint. Figure 5 illustrates this; the rule $X_i \in [L_1, U_1] \rightarrow X_j \in [L_2, U_2]$ is the same as $(X_i < L_1) \vee (X_i > U_1) \vee (X_j \in [L_2, U_2])$, the feasible region for which is shown on the right side of the figure. In particular, there are infeasible points (such as A in the figure) for which no pair of directions satisfying the conditions for smoothness exists. (A^* can be in any of the three directions indicated by black dots with respect to A.) Interestingly, however, the set of domain rules put *together* exhibits smoothness. In particular, as the left side of Figure 5 shows, the required pair of directions exists at every infeasible point for the example constraint from Figure 1. We note, however, that this observation by itself does not lead to a strongly polynomial-time algorithm (because of the same problem that arises in RDTPs of having to deal with continuous domains). The rest of this paper therefore concentrates on designing strongly polynomial-time algorithms for handling domain rules and simple temporal constraints. Essentially, we will achieve strongly polynomial running times by combining the intuition in the above observation with the power of the *distance graph* representation of STPs.

Strongly Polynomial-time Algorithms

In this section, we will present strongly polynomial-time (deterministic and randomized) algorithms for solving tem-

[1] '1's and '0's respectively represent allowed and disallowed combinations of values to X_i and X_j.

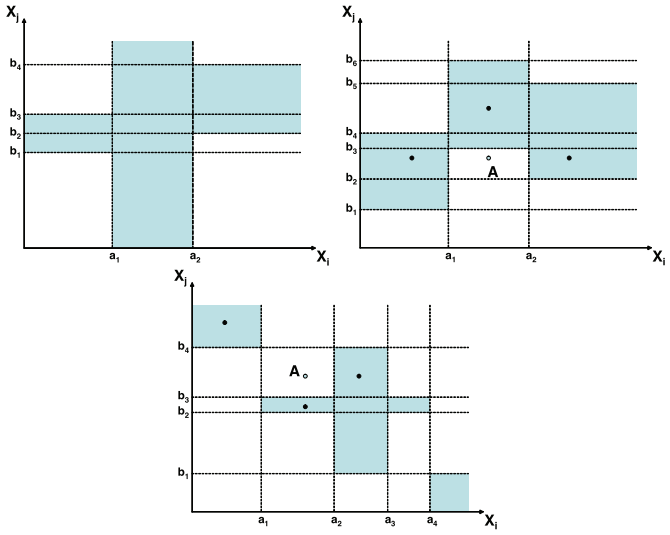

Figure 6: Illustrates constraints that can be expressed using domain rules. The (connected) feasible region on the top left side of the figure can be expressed using domain rules with either X_i or X_j as the tail variable. The (connected) feasible region on the top right side of the figure, however, can be expressed using domain rules only with X_i as the tail variable (and not with X_j). In the last case (bottom), the feasible region can be expressed using domain rules with either X_i or X_j as the tail variable, but is not connected. Only the first case qualifies as a smooth constraint.

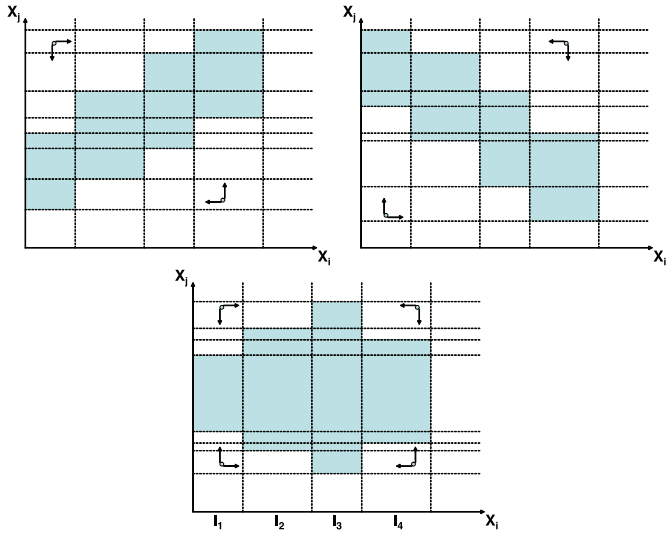

Figure 7: Three basic cases in which a constraint between X_i and X_j (with a connected feasible region) can be expressed using domain rules with either X_i or X_j as the tail variable. Cases (1) and (2) (top two cases) have a monotonic behavior in terms of the lower and upper end-points of the intervals $I_{j1}, I_{j2} \ldots I_{jk}$ (using X_i as the tail). In case (3) (bottom case), this is not true. However, the subsumption relationships between the intervals $I_{j1}, I_{j2} \ldots I_{jk}$ follow a *bitonic* sequence—i.e. $I_{j1} \subseteq I_{j2} \subseteq I_{j3} \supseteq I_{j4}$.

poral problems with domain rules. The correctness of the algorithms is proved in the Lemmas that follow. For notational convenience, we will refer to the constraint $(L \leq X_a \leq U)$ as $X_a \in [L, U]$. For any interval $I = [L, U]$, we will denote its left end-point (viz. L) by $\mathcal{L}(I)$, and its right end-point (viz. U) by $\mathcal{R}(I)$. We will also say that an interval $I_1 = [L_1, U_1]$ *subsumes* another interval $I_2 = [L_2, U_2]$ (denoted $I_2 \subseteq I_1$) if and only if $L_1 \leq L_2$ and $U_2 \leq U_1$.

We note once again that we are dealing with temporal constraints that are either simple temporal (between X_i and X_j), or have a connected feasible region that can be expressed by a finite set of domain rules, each in turn of the form: $X_i \in I_{i1} \rightarrow X_j \in I_{j1}$ (and conversely $X_j \in I_{j2} \rightarrow X_i \in I_{i2}$). It is important that the constraint be expressible in both ways and have a connected feasible region in order for it to be smooth (as illustrated in Figure 6). This is because *binary* smooth constraints have been shown to be equivalent to *connected row-convex* (CRC) constraints (see (Deville *et al* 1999) and (Kumar 2005c)). The top left side of the figure has a connected feasible region which can be expressed as three rules of the form $X_i \in I_p \rightarrow X_j \in I_q$ (namely $X_i \in [0, a_1] \rightarrow X_j \in [b_1, b_3], X_i \in [a_1, a_2] \rightarrow X_j \in [0, \infty]$ and $X_i \in [a_2, \infty] \rightarrow X_j \in [b_2, b_4]$). Here, X_i is referred to as the *tail* variable, and X_j as the *head* variable. Moreover, the constraint can also be expressed using five rules of the form $X_j \in I_q \rightarrow X_i \in I_p$ i.e. with X_j as the tail variable (namely $X_j \in [0, b_1] \rightarrow X_i \in [a_1, a_2]$, $X_j \in [b_1, b_2] \rightarrow X_i \in [0, a_2]$, $X_j \in [b_2, b_3] \rightarrow X_i \in [0, \infty]$, $X_j \in [b_3, b_4] \rightarrow X_i \in [a_1, \infty]$ and $X_j \in [b_4, \infty] \rightarrow X_i \in [a_1, a_2]$). Thus, it satisfies the properties required for smoothness. The top right side of the figure, however, cannot be expressed in this fashion, and is therefore not smooth. Although it can be expressed as three rules of the form $X_i \in I_p \rightarrow X_j \in I_q$ (namely $X_i \in [0, a_1] \rightarrow X_j \in [b_1, b_4]$, $X_i \in [a_1, a_2] \rightarrow X_j \in [b_3, b_6]$ and $X_i \in [a_2, \infty] \rightarrow X_j \in [b_2, b_5]$), it cannot be expressed using rules of the form $X_j \in I_q \rightarrow X_i \in I_p$. In particular, this is because when $X_j \in [b_2, b_3]$, X_i is in $[0, a_1] \cup [a_2, \infty]$ and not within a continuous interval. The bottom case in the figure shows that although the constraint can be expressed as a finite set of domain rules using either X_i or X_j as the tail variable, it is still not a smooth constraint because the feasible region is not connected.[2] Put together, not all non-monotonic relationships are smooth; but those constraints that have a connected feasible region and that can be expressed as domain rules using any of the two variables as the tail variable are certainly smooth. As we will see later in this section, these cases can be made amenable to simple polynomial-time (deterministic and randomized) algorithms. We also note that the examples presented in the Introduction satisfy our requirements for smoothness.

Consider a constraint between X_i and X_j (with a connected feasible region) that can be expressed as a finite set of domain rules using either of the variables as the tail variable. Without loss of generality, let us assume that we use

[2]Strictly speaking, smoothness does not a priori prohibit disconnected feasible regions (e.g. Type 2 disjunctions of RDTPs) provided they become connected after removing entire horizontal and vertical strips of infeasible regions (analogous to rows and columns with only '0's in the discrete case). Although these cases can also be taken care of in the various Lemmas, we will not deal with them explicitly in the rest of this paper (to retain simplicity).

ALGORITHM: SOLVE-STP-DOMAIN-RULES
INPUT: Events $\{X_0, X_1 \ldots X_N\}$, with a constraint between X_i and X_j being either simple temporal or having a connected feasible region expressible as a finite set of domain rules using either X_i or X_j as the tail variable.
OUTPUT: A solution s (if it exists).
(1) Cast the problem as a "disjunct selection problem" using meta-variables $\mathcal{W} = \{W_1, W_2 \ldots W_R\}$ such that:
(a) A variable $W \in \mathcal{W}$ corresponds to a constraint of the form $X_i \in I_{i1} \rightarrow X_j \in I_{j1}, X_i \in I_{i2} \rightarrow X_j \in I_{j2} \ldots X_i \in I_{ik} \rightarrow X_j \in I_{jk}$ (domain rules).
(b) The domain of W is $= \{(X_i \in I_{i1} \wedge X_j \in I_{j1}), (X_i \in I_{i2} \wedge X_j \in I_{j2}) \ldots (X_i \in I_{ik} \wedge X_j \in I_{jk})\}$
(c) The ordering on the domain values of W is the *nominal ordering*—i.e. ascending order of the intervals defined for the tail variable.
(2) For every W_i and W_j in $\mathcal{W}$, build a binary constraint as follows:
(a) An instantiation of disjuncts to W_i and W_j is disallowed, if and only if, together with the rest of the simple temporal constraints, they introduce a negative cycle in the underlying *distance graph*.
(3) Solve these binary constraints using the (deterministic or randomized) procedures for solving CRC constraints.
(4) RETURN: s = SOLVE-STP (induced STP).
END ALGORITHM

Figure 8: Strongly polynomial-time (deterministic and randomized) algorithms for solving simple temporal constraints with domain rules (of the special kinds identified in the paper). We note that a "disjunct" involves constraining intervals for two variables.

X_i as the tail variable. Then, a natural ordering exists on the rules $X_i \in I_{i1} \rightarrow X_j \in I_{j1}, X_i \in I_{i2} \rightarrow X_j \in I_{j2} \ldots X_i \in I_{ik} \rightarrow X_j \in I_{jk}$—namely, the increasing order of the intervals $I_{i1}, I_{i2} \ldots I_{ik}$.[3] We will refer to this ordering as the *nominal ordering* of the domain rules with respect to X_i. For example, the rules $X_i \in [0, 2] \rightarrow X_j \in [3, 5]$, $X_i \in [10, \infty] \rightarrow X_j \in [7, 14]$ and $X_i \in [2, 10] \rightarrow X_j \in [1, \infty]$ are arranged as $X_i \in [0, 2] \rightarrow X_j \in [3, 5]$, $X_i \in [2, 10] \rightarrow X_j \in [1, \infty]$ and $X_i \in [10, \infty] \rightarrow X_j \in [7, 14]$ in the nominal ordering with respect to the tail variable X_i.

Figure 7 shows the three basic cases in which a constraint between X_i and X_j (with a connected feasible region) can be expressed using domain rules with either X_i or X_j as the tail variable. (Other cases are possible by certain combinations of these basic cases, and their tractability can be established using arguments similar to the ones presented in this paper; but in the interest of simplicity, we will not deal with them explicitly.) Cases (1) and (2) have a monotonic behavior in terms of the lower and upper end-points of the intervals $I_{j1}, I_{j2} \ldots I_{jk}$ (using X_i as the tail variable). The more interesting case is (3) where the subsumption relationships between the intervals $I_{j1}, I_{j2} \ldots I_{jk}$ follow a *bitonic* sequence—i.e. $I_{j1} \subseteq I_{j2} \ldots I_{j(q^*-1)} \subseteq I_{jq^*} \supseteq I_{j(q^*+1)} \ldots I_{j(k-1)} \supseteq I_{jk}$ (for some $1 \leq q^* \leq k$). We will deal only with the more interesting case in (3). The cases in (1) and (2) are simple generalizations of the Lemmas presented in (Kumar 2005a). Case (3), however, requires different arguments (presented in the Lemmas below). We use the result that a consistent schedule exists for an STP if and only if its associated *distance graph* does not contain any negative cycles (Dechter *et al* 1991).

For any finite set of domain rules $X_i \in I_{i1} \rightarrow X_j \in I_{j1}$, $X_i \in I_{i2} \rightarrow X_j \in I_{j2} \ldots X_i \in I_{ik} \rightarrow X_j \in I_{jk}$, since the intervals $I_{i1}, I_{i2} \ldots I_{ik}$ are non-overlapping, all of the implications (rules) would be satisfied if we ensure that both $(X_i \in I_{ih})$ and $(X_j \in I_{jh})$ are true for some $1 \leq h \leq k$. Because of this, we can view our problem as a meta-level CSP where the meta-level variables are associated with every set of rules of the form $X_i \in I_{i1} \rightarrow X_j \in I_{j1}, X_i \in I_{i2} \rightarrow X_j \in I_{j2} \ldots X_i \in I_{ik} \rightarrow X_j \in I_{jk}$, and their domains are respectively $\{(X_i \in I_{i1} \wedge X_j \in I_{j1}), (X_i \in I_{i2} \wedge X_j \in I_{j2}) \ldots (X_i \in I_{ik} \wedge X_j \in I_{jk})\}$. The goal is then to instantiate these meta-level variables in such a way that the resulting simple temporal constraints (in addition to the other simple temporal constraints already specified in the problem) do not induce a negative cost cycle in the underlying *distance graph*.

Lemma 1: The choice $(X_i \in I_p \wedge X_j \in I_q)$ is equivalent to adding the edges $\langle X_0, X_i \rangle$ annotated with $\mathcal{R}(I_p)$, $\langle X_i, X_0 \rangle$ annotated with $-\mathcal{L}(I_p)$, $\langle X_0, X_j \rangle$ annotated with $\mathcal{R}(I_q)$ and $\langle X_j, X_0 \rangle$ annotated with $-\mathcal{L}(I_q)$ to the *distance graph* without creating a negative cycle.
Proof: If we have to ensure that the variable X_i is in the interval I_p, we have to make sure that $X_i - X_0 \leq \mathcal{R}(I_p)$ and $X_i - X_0 \geq \mathcal{L}(I_p)$. Retaining the semantics of the *distance graph* where the constraint $X_b - X_a \leq w$ is specified by the edge $\langle X_a, X_b \rangle$ annotated with w, this corresponds to the addition of the edges $\langle X_0, X_i \rangle$ annotated with $\mathcal{R}(I_p)$, and $\langle X_i, X_0 \rangle$ annotated with $-\mathcal{L}(I_p)$, to the *distance graph* without creating an inconsistency (which is characterized by the presence of a negative cycle). Similarly, the other two edges are required for ensuring that $X_j \in I_q$.

We define a *conflict* to be an instantiation of a set of meta-level variables that results in an inconsistency with the rest of the simple temporal constraints. We define a *minimal conflict* to be a conflict no proper subset of which is also a conflict. It is easy to see that an instantiation of a set of meta-level variables is consistent (with the rest of the simple temporal constraints in the problem) if and only if there is no subset of this set that constitutes a minimal conflict.

Lemma 2: The size of a minimal conflict is ≤ 2.
Proof: Suppose we try to instantiate a set of meta-level variables $W_1, W_2 \ldots W_h$. Since instantiating any meta-level variable W requires committing to some variables X_{i_W} and X_{j_W} to respectively be within some intervals I_{p_W} and I_{q_W}, we would have to add the following edges to the *distance graph*: $\langle X_0, X_{i_W} \rangle$ annotated with $\mathcal{R}(I_{p_W})$, $\langle X_{i_W}, X_0 \rangle$ annotated with $-\mathcal{L}(I_{p_W})$, $\langle X_0, X_{j_W} \rangle$ annotated with $\mathcal{R}(I_{q_W})$, and $\langle X_{j_W}, X_0 \rangle$ annotated with $-\mathcal{L}(I_{q_W})$

[3] We will assume that $I_{i1}, I_{i2} \ldots I_{ik}$ are non-overlapping; otherwise, we can rewrite the rules in such a canonical form.

(for all $W \in \{W_1, W_2 \ldots W_h\}$). We will refer to these edges as "special" edges. Knowing that the *distance graph* initially does not contain any negative cycles (because any inconsistency in the given simple temporal constraints can be caught right away), if a negative cycle is newly created, it must involve one of the "special" edges. Since all "special" edges have X_0 as an end point, the negative cycle must contain X_0. Further, since a fundamental cycle can have any node repeated at most once, at most 2 "special" edges can be present in a newly created negative cycle. Finally, since "special" edges correspond to the instantiation of meta-level variables, the size of a minimal conflict is ≤ 2.

Lemma 3: Under the nominal domain orderings for all the meta-level variables, all the resulting (meta-level) binary constraints are CRC.

Proof: Consider the meta-level variables W_i and W_j with respective domains $\{(X_q \in I_{q1} \wedge X_r \in I_{r1}), (X_q \in I_{q2} \wedge X_r \in I_{r2}) \ldots (X_q \in I_{qK} \wedge X_r \in I_{rK})\}$ and $\{(X_s \in I_{s1} \wedge X_t \in I_{t1}), (X_s \in I_{s2} \wedge X_t \in I_{t2}) \ldots (X_s \in I_{sH} \wedge X_t \in I_{tH})\}$. We consider them in their nominal orders with respect to the tail variables (X_q and X_s respectively). Instantiating W_i implies that X_q and X_r are restricted to be within certain intervals I_{qk} and I_{rk} ($1 \leq k \leq K$) respectively; and similarly, instantiating W_j implies that X_s and X_t are restricted to be within certain other intervals I_{sh} and I_{th} ($1 \leq h \leq H$) respectively. Now, a conflict can arise in a number of ways. If $(X_q \in I_{qk})$ and $(X_r \in I_{rk})$ conflict, then the disjunct $(X_q \in I_{qk} \wedge X_r \in I_{rk})$ is simply removed from the domain of W_i; similarly, if $(X_s \in I_{sh})$ and $(X_t \in I_{th})$ conflict, then the disjunct $(X_s \in I_{sh} \wedge X_t \in I_{th})$ is simply removed from the domain of W_j. There are four other kinds of conflicts: (1) between X_q and X_s, (2) between X_r and X_t, (3) between X_s and X_r, and (4) between X_q and X_t. In each case, we will prove that the meta-level constraint resulting from such conflicts (under the nominal orderings of the disjuncts) is CRC. Since the intersection of CRC constraints is also CRC, the required meta-level constraint between W_i and W_j will also be CRC as required (Deville *et al* 1999). A detailed proof for case (1) appears in (Kumar 2005a). Consider case (2). Let $(X_r \in I_{r1}), (X_r \in I_{r2}) \ldots (X_r \in I_{rK})$ constitute the columns and $(X_t \in I_{t1}), (X_t \in I_{t2}) \ldots (X_t \in I_{tH})$ constitute the rows (in the matrix representation of the constraint). Now, since $I_{r1}, I_{r2} \ldots I_{rK}$ form a *bitonic* sequence (because X_r is a head variable), the '1's in any row appear continuously, and any non-empty row (row with at least one '1') will contain a '1' in the column corresponding to the largest interval of $I_{r1}, I_{r2} \ldots I_{rK}$ (say I_{ra^*} for $1 \leq a^* \leq K$). Therefore, any two non-empty rows will have a common '1' (namely at I_{ra^*}), and the bands of '1's in any two consecutive (non-empty) rows will touch each other. Similarly the '1's in any column appear continuously (because $I_{t1}, I_{t2} \ldots I_{tH}$ form a *bitonic* sequence). This proves that the constraint is CRC (Deville *et al* 1999). Now consider case (3) (case (4) is symmetric). Without loss of generality, let $(X_r \in I_{r1}), (X_r \in I_{t2}) \ldots (X_r \in I_{rK})$ constitute the columns, and let $(X_s \in I_{s1}), (X_s \in I_{s2}) \ldots (X_s \in I_{sH})$ constitute the rows. As before, since $I_{r1}, I_{r2} \ldots I_{rK}$ form a *bitonic* sequence, the '1's in any non-empty row appear continuously, and the bands of '1's in any two consecutive non-empty rows touch each other. Further, in any column the '1's appear continuously (see Lemmas in (Kumar 2005a)). This establishes that the constraint is CRC. Finally, since all the constraints are CRC, the meta-level constraint formed by taking the intersection of these constraints is also CRC (as required).

From the previous Lemma, we conclude that all the meta-level constraints are CRC. CRC constraints, in turn, are amenable to polynomial-time deterministic algorithms based on establishing path-consistency (Deville *et al* 1999). They are also amenable to extremely simple polynomial-time randomized algorithms (Kumar 2005b). Figure 8 presents the final algorithm for solving metric temporal problems involving simple temporal constraints and the special kinds of domain rules identified in this paper.

Conclusions and Future Work

We dealt with important kinds of metric temporal problems that arise in many real-life situations. In particular, we identified tractable classes of constraints that are useful in capturing many important kinds of non-monotonic relationships between the execution times of events. We provided simple polynomial-time (deterministic and randomized) algorithms for solving such problems—thereby enabling us to efficiently deal with very expressive representations of time. Future work includes possibly extending the theory developed in this paper to geometrical reasoning tasks and reasoning tasks that involve preferences in temporal problems.

References

Armando A., Castellini C., Giunchiglia E. and Maratea M. 2004. A SAT-based Decision Procedure for the Boolean Combination of Difference Constraints. *SAT'2004*.

Dechter R., Meiri I. and Pearl J. 1991. Temporal Constraint Networks. *Artificial Intelligence Journal 49. 1991*.

Deville Y., Barette O. and Van Hentenryck P. 1999. Constraint Satisfaction over Connected Row-Convex Constraints. *Artificial Intelligence, 109:243-271*.

Kumar T. K. S. 2003. Incremental Computation of Resource-Envelopes in Producer-Consumer Models. *9th Int. Conference on Principles and Practice of Constraint Programming (CP'2003)*.

Kumar T. K. S. 2005a. On the Tractability of Restricted Disjunctive Temporal Problems. *ICAPS'2005*.

Kumar T. K. S. 2005b. On the Tractability of Smooth Constraint Satisfaction Problems. *CP-AI-OR'2005*.

Kumar T. K. S. 2005c. Contributions to Algorithmic Techniques in Automated Reasoning about Physical Systems. *PhD Thesis, Computer Science Department, Stanford University, March 2005*.

Muscettola N. 2002. Computing the Envelope for Stepwise Constant Resource Allocations. *Proceedings of CP'2002*.

Smith D., Frank J. and Jonsson A. 2000. Bridging the Gap Between Planning and Scheduling. *Knowledge Engineering Review 15:1, 2000*.

Tsamardinos I. and Pollack M. E. 2003. Efficient Solution Techniques for Disjunctive Temporal Reasoning Problems. *Artificial Intelligence, 151(1-2):43-90, 2003*.