Augmenting Disjunctive Temporal Problems with Finite-Domain Constraints

Michael D. Moffitt and Bart Peintner and Martha E. Pollack
Department of Electrical Engineering and Computer Science
University of Michigan
Ann Arbor, MI 48109
{mmoffitt, bpeintne, pollackm}@eecs.umich.edu

Abstract

We present a general framework for augmenting instances of the Disjunctive Temporal Problem (DTP) with finite-domain constraints. In this new formalism, the bounds of the temporal constraints become conditional on the finite-domain assignment. This hybridization makes it possible to reason simultaneously about temporal relationships between events as well as their nontemporal properties. We provide a special case of this hybridization that allows reasoning about a limited form of *spatial* constraints; namely, the travel time induced by the locations of a set of activities. We develop a least-commitment algorithm for efficiently finding solutions to this combined constraint system and provide empirical results demonstrating the effectiveness of our approach.

Introduction

Temporal reasoning is an important tool in many areas of artificial intelligence, including both planning and scheduling. Often times, it may be the case that temporal constraints alone are not sufficient to represent a practical planning or scheduling problem. For instance, if a person is to attend two meetings that each could occur in one of several locations, it must be ensured that adequate travel time between these locations is reserved. Traditional temporal representations are typically unable to express such finite-domain considerations, and likewise, algorithms for temporal reasoning are unable to ensure their consistency.

In this paper, we present a method for augmenting instances of the Disjunctive Temporal Problem (DTP) (Stergiou & Koubarakis 1998) with finite-domain constraints. DTPs are a particularly expressive form of temporal constraint satisfaction problem that subsumes Simple Temporal Problems and Temporal Constraint Satisfaction Problems (Dechter, Meiri, & Pearl 1991). In our hybrid formalism, the bounds of the temporal constraints become conditional on the finite-domain assignment. This hybridization allows simultaneous reasoning about temporal relationships between events as well as their nontemporal properties.

After providing the details about this combined constraint system, we develop a least-commitment algorithm for efficiently generating solutions. While least-commitment strategies are prevalent in areas such as partial-order planning (Weld 1994), our approach is unique in that it is able to take advantage of the linear inequalities that temporal representations employ to express differences in time between pairs of temporal events. We also provide a special tractable case of this hybridization that allows reasoning about the travel time induced by the locations of a set of activities.

It should be noted that there have been several previous projects extending temporal formalisms to account for nontemporal considerations. Recently, the notion of a Resource Temporal Network (Laborie 2003) has been introduced, which augments a temporal network with a number of fluent resources, each represented by a numerical level that changes over time. The approach in (Bockmayr & Kasper 1998) is somewhat more closely related to our technique, as it combines integer programming and finite domain constraints into a single framework. However, they take the approach of constructing an expressive mixed constraint language grounded in first-order-logic, whereas our formulation instead resembles ideas found in conditional CSP literature (Mittal & Falkenhainer 1990).

A Motivating Example

Consider a faculty advisor at a major university. This advisor has three students: Pete, Mark, and Julie. She would like to set up meetings with these students to discuss upcoming paper deadlines. The meetings with Pete and Mark have already been predetermined, and now a meeting with Julie must be scheduled.

Encoding this problem with temporal constraints (such as those of a DTP) is fairly straightforward. In addition to the duration of the meeting, it must be expressed that the meeting with Julie cannot overlap with the meetings with Pete and Mark. Suppose the intervals $[P_S, P_E]$, $[M_S, M_E]$, and $[J_S, J_E]$ represent these meetings. The non-overlap constraints can be written as follows:

$$P_E - J_S \leq 0 \vee J_E - P_S \leq 0$$
$$M_E - J_S \leq 0 \vee J_E - M_S \leq 0$$

However, there's a problem: the advisor has two offices – one in an administrative building (L_1), and the other in a research lab (L_2). Furthermore, the locations of the meetings with Pete and Mark have been restricted:

$$loc(\{P_S, P_E\}) = L_1$$

$$loc(\{M_S, M_E\}) = L_2$$

It takes 10 minutes to get from L_2 to L_1, and 15 minutes to get from L_1 to L_2 (since the university bus takes a slight detour in this reverse direction). As a result, it is no longer sufficient to maintain the same non-overlap constraints as before – depending on *where* the meeting with Julie is scheduled, one of these two constraints must be tightened to allow for the travel time. For instance, if the meeting with Julie is to be at L_1, the constraints would be tightened as follows:

$$P_E - J_S \leq 0 \vee J_E - P_S \leq 0$$
$$M_E - J_S \leq -10 \vee J_E - M_S \leq -15$$

Alternatively, if the meeting with Julie is to be in L_2, a different tightening would be required:

$$P_E - J_S \leq -15 \vee J_E - P_S \leq -10$$
$$M_E - J_S \leq 0 \vee J_E - M_S \leq 0$$

To model this problem with a DTP, the location for the meeting with Julie would have to be specified in advance; otherwise, all possible instantiations of the meeting location would need to be enumerated and solved as separate DTPs until one proved to be consistent. The reason is that in a DTP, the bounds of the linear inequalities are always held constant. One can imagine more complicated extensions to this example – for instance, when the location and time of several meetings have been left only partially specified. Or, suppose the advisor must wait for a faculty candidate to arrive at L_1. This waiting activity can overlap with Julie's meeting if the meeting is in L_1, but not if the meeting is in L_2. Problems such as these can easily arise in domains other than meeting scheduling, and demand more expressive power than existing temporal representations allow.

Background

Finite-Domain Constraint Networks

A *constraint network* (Dechter 2003) is a constraint satisfaction problem defined by a triple $\langle X, D, C\rangle$, where $X = \{x_1, ..., x_n\}$ is a set of variables, $D = \{D_1, ..., D_n\}$ contains a domain $D_i = \{v_1, ..., v_k\}$ for each variable that lists the possible values it may take, and $C = \{C_1, ..., C_t\}$ is a set of constraints, where each constraint C_i is a relation R_i defined on a subset of variables $S_i \subseteq X$. A solution to a constraint network is an assignment $\bar{a} = (a_1, ..., a_n)$ such that each $a_i \in D_i$, and for each constraint C_i, the projected assignment $\bar{a}[S_i] \in R_i$.

Finite-domain constraint networks provide a powerful means for representing many constraint satisfaction problems, and as a result, several inference and search techniques have been developed for improving the efficiency of constraint solving algorithms. However, these networks are typically inadequate for representing and reasoning about temporal relationships. The domains of the temporal variables often draw from the set of real numbers, which makes search of the traditional assignment space impractical. In response, several alternative representations (both qualitative and quantitative) have been developed especially for the purpose of temporal reasoning. In this paper we concentrate on the quantitative variety.

Disjunctive Temporal Problems

A *Disjunctive Temporal Problem* (DTP) (Stergiou & Koubarakis 1998) is a constraint satisfaction problem defined by a pair $\langle X, C\rangle$, where each element in X designates a time point whose domain is $\Re$, and C is a set of constraints of the form $(c_{i1} \vee c_{i2} \vee ... \vee c_{in})$ where each c_{ij} is a linear inequality $x_{ij} - y_{ij} \leq b_{ij}$; $x_{ij}, y_{ij} \in X$ and $b_{ij} \in \Re$. DTPs are thus a generalization of Simple Temporal Problems (STPs) in which each constraint is limited to a single inequality. A solution to a DTP is an assignment of values to time points such that all constraints are satisfied.

Several algorithms have been developed for solving DTPs (Stergiou & Koubarakis 1998; Oddi & Cesta 2000; Tsamardinos & Pollack 2003). Typically, these algorithms view the DTP as a collection of alternative STPs. Using this approach, the algorithm selects a single disjunct from each constraint of a given DTP D. The resulting set forms an STP, called a *component STP* of D, which can then be checked for consistency in polynomial-time using a shortest-path algorithm (Dechter, Meiri, & Pearl 1991). Specifically, an STP is consistent if and only if it contains no negative cycles, which can be determined by computing the all-pairs shortest path matrix and checking that the values along the main diagonal are non-negative. Clearly, a DTP D is consistent if and only if it contains at least one consistent component STP. Furthermore, any solution to a consistent component STP of D is also a solution to D itself. Consequently, it is standard in the DTP literature to consider any consistent component STP to be a solution of the DTP to which it belongs.

A number of pruning techniques can be used to focus the search for a consistent component STP of a given DTP. These include conflict-directed backjumping, removal of subsumed variables, and semantic branching. The DTP solver Epilitis (Tsamardinos & Pollack 2003) integrated all of these techniques as well as no-good recording. At the time it was developed, Epilitis was the fastest existing DTP solver, although it was recently surpassed by TSAT++ (Armando *et al.* 2004).

While DTPs offer a flexible language for expressing temporal relationships, they do not provide the ability to capture finite-domain constraints, unlike traditional constraint networks. To obtain the best of both worlds, we need some way of exploiting the advantages of both representations.

Disjunctive Temporal Problems with Finite-Domain Constraints

We define a *Disjunctive Temporal Problem with Finite-Domain Constraints* (DTP$_{FD}$) as a tuple $\langle X, D, C_T, C_F\rangle$. Each element X_i in X designates a pair $\langle x_i^t, x_i^f\rangle$, where x_i^t is the temporal component of X_i, and x_i^f is the finite-domain component of X_i. The set $D = \{D_1, ..., D_n\}$ contains a domain $D_i = \{v_1, ..., v_k\}$ for each variable that lists the possible values for that variable's finite-domain component x_i^f;the domain of each x_i^t is $\Re$. C_T is a set of temporal constraints of the form $(c_{i1} \vee c_{i2} \vee ... \vee c_{in})$ where each *conditional disjunct* c_{ij} is a linear inequality

$$x_{ij}^t - y_{ij}^t \leq B_{ij}(x_{ij}^f, y_{ij}^f)$$

x^f \ y^f	L_1	L_2	L_3
L_1	0	–15	–5
L_2	–10	0	–5
L_3	–5	–5	0

Figure 1: The bounds matrix for the difference $x^t - y^t$.

where $x_{ij}, y_{ij} \in X$ and the bounds function B_{ij} is a mapping $D(x_{ij}^f) \times D(y_{ij}^f) \rightarrow \Re$. When the context is clear, we will drop the subscripts and abbreviate the bound as B. $C_F = \{C_{F1}, ..., C_{Ft}\}$ is a set of finite-domain constraints. A solution to a DTP$_{FD}$ consists of two parallel assignments – an assignment to the finite-domain component of each variable that satisfies the set of constraints C_F, and an assignment to the temporal component of each variable that satisfies the set of temporal constraints C_T, whose bounds are grounded by the finite-domain assignment.

Returning to our initial example, the set of variables is $X = \{P_S, P_E, M_S, M_E, J_S, J_E\}$. Presumably, each meeting should start and end at the same place, and so the constraint network reflects the partition $P = \{\{P_S, P_E\}, \{M_S, M_E\}, \{J_S, J_E\}\}$. The finite-domain constraints (C_F) in this network would thus be $\{P_S^f = P_E^f, M_S^f = M_E^f, J_S^f = J_E^f\}$. The domains of the elements in each equivalence class draw from different subsets of locations:

Equiv. Class	Locations
$P_1 = \{P_S, P_E\}$	$D(P_1) = \{L_1\}$
$P_2 = \{M_S, M_E\}$	$D(P_2) = \{L_2\}$
$P_3 = \{J_S, J_E\}$	$D(P_3) = \{L_1, L_2\}$

Finally, we consider the temporal constraints of the problem (C_T). Some of them will look like conventional DTP constraints; for instance, if Julie's meeting is to last 30 minutes, these constraints are added:

$$J_E^t - J_S^t \leq +30$$
$$J_S^t - J_E^t \leq -30$$

Here, each function B_{ij} maps to a constant value, independent of the location of Julie's meeting. However, other constraints are now conditional on the finite-domain assignments. These new constraints are written as follows:

$$P_E^t - J_S^t \leq B(P_E^f, J_S^f) \vee J_E^t - P_S^t \leq B(J_E^f, P_S^f)$$
$$M_E^t - J_S^t \leq B(M_E^f, J_S^f) \vee J_E^t - M_S^t \leq B(J_E^f, M_S^f)$$

We represent the values of the bounds in these constraints with the bounds matrix $B(x^f, y^f)$ in Figure 1; the $(r, c)^{th}$ value represents the bound B (for $x^t - y^t \leq B$) when x^f is assigned the r^{th} finite-domain value and y^f is assigned c^{th} finite-domain value. For example, given the partial assignment $(x^f, y^f) = (L_1, L_2)$, we would have $x^t - y^t \leq -15$. Note that we have added a location L_3 that is 5 minutes from both L_1 and L_2, although it so happens that this location never occurs as a legal value for any variable's finite-domain component.

Solving DTP$_{FD}$'s

In this section, we present three methods for solving a DTP$_{FD}$. The first is a brute-force approach that fully instantiates the finite-domain components before addressing the temporal constraints. The second takes the reverse approach, and starts by solving the temporal constraints using a least-commitment strategy. The third technique builds on the second, but uses additional information obtained during the temporal search to infer possible assignments for the finite-domain components, and is thus able to prune large regions of the search space. Such an approach resembles, at least conceptually, many of the strategies used in constraint logic programming (CLP) (Jaffar & Maher 1994).

A Brute-Force Approach

Since the bound on each conditional disjunct of a temporal constraint depends on the finite-domain assignments to the variables in its scope, the most obvious method for solving a DTP$_{FD}$ is to first instantiate all finite-domain components, and then solve the *induced DTP* (that is, the DTP obtained by fixing all bounds relative to the finite-domain assignment). If a solution is found, the process may stop; otherwise, alternative finite-domain assignments will be attempted until satisfiability is achieved or search is exhausted.

There are two fundamental problems with this approach. First, it fails to take advantage of any shared structure that may exist between two induced DTP's generated from similar (or even dissimilar) finite-domain assignments. Indeed, some constraints (such as an activity's duration) may not depend on the finite-domain constraints at all. Thus, to solve each induced DTP separately in sequence could potentially repeat many of the same steps in search. Second, for some types of problems, it may be the case that the number of possible finite-domain assignments is disproportionately larger than the number of consistent component STPs, especially if the finite-domain constraints are fairly loose (e.g, if a meeting can take place in one of a hundred conference rooms). As a result, enumeration of the finite-domain assignments may prove to be prohibitively expensive, particularly if the density of the temporal constraints admits no solution.

A Least-Commitment Approach

A potentially more effective method for solving a DTP$_{FD}$ is to perform a *least-commitment* search through the temporal constraints, and consider the finite-domain assignments only when a leaf-node is reached. By least-commitment, we mean that for each disjunct c_{ij} in the DTP$_{FD}$, the conditional bound will temporarily be replaced with the constant

$$\max_{v_x \in D(x_{ij}^f), v_y \in D(y_{ij}^f)} B_{ij}(v_x, v_y).$$

In words, we compute the largest possible bound available for any feasible pair of assignments, and use that to create the loosest possible constraint, which we call a *weakened disjunct*. If there are $|C|$ temporal constraints and $|X|$ variables, the top $|C|$ levels of this search will assign these weakened disjuncts to the temporal constraints of the DTP, and the bottom $|X|$ levels will assign values to the finite-domain

Solve-Temporal-Layer(A_T, U_T, U_F)
1. If ($U_T = \oslash$) return **Solve-Finite-Domain-Layer**($\oslash$, U_F, A_T)
2. $C_i \leftarrow$ *select-variable*(U_T), $U'_T \leftarrow U_T - \{C_i\}$
3. For each weakened disjunct c_{ij} of *d*(C_i)
4. $\quad A'_T \leftarrow A_T \cup \{C_i \leftarrow c_{ij}\}$
5. $\quad$ If **consistent**(A'_T)
6. $\quad\quad$ If **Solve-Temporal-Layer**(A'_T, U'_T, U_F) = *success*
7. $\quad\quad\quad$ return *success*
8. return *failure*

Solve-Finite-Domain-Layer(A_F, U_F, A_T)
1. If ($U_F = \oslash$) return **Tighten-Component-STP**(A_F, A_T)
2. $X_i^f \leftarrow$ *select-variable*(U_F), $U'_F \leftarrow U_F - \{X_i^f\}$
3. For each value v of *D*(X_i^f)
4. $\quad A'_F \leftarrow A_F \cup \{X_i^f \leftarrow v\}$
5. $\quad$ If **Solve-Finite-Domain-Layer**(A'_F, U'_F, A_T) = *success*
6. $\quad\quad$ return *success*
7. return *failure*

Tighten-Component-STP(A_F, A_T)
1. $A'_T \leftarrow A_T$
2. For each ($C_i \leftarrow c_{ij}$) in A_T
3. $\quad A'_T \leftarrow A'_T \cup \{x_{ij} - y_{ij} \leq B_{ij}(A_F)\}$
4. $\quad$ Unless **consistent**(A'_T) return *failure*
5. return *success*

Figure 2: A least-commitment approach for solving a DTP_{FD}

components of the variables.[1] The effect of these bottom $|X|$ levels will be the tightening of disjuncts that compose the component STP of the weakened DTP. The pseudocode for this general procedure is given in Figure 2. The function **Solve-Temporal-Layer** is called with A_T (the set of assignments from disjuncts to constraints) initialized to $\oslash$, U_T (the set of uninstantiated temporal constraints) initialized to C_T, and U_F (the set of unassigned finite-domain variable components) initialized to contain all X_i^f. It recursively attempts to assign weakened disjuncts to constraints, backtracking whenever failure is encountered. At leaf nodes of this weakened DTP search, the function **Solve-Finite-Domain-Layer** is called to perform instantiations to the finite-domain components. For each such instantiation, the function **Tighten-Component-STP** is called in an attempt to appropriately tighten the bounds of the currently weakened disjuncts.[2]

The least-commitment strategy addresses both of the concerns presented in the previous section. Since a single DTP is examined, any common structure can easily be exploited by the search procedure. Furthermore, if the temporal constraints of the problem are sufficiently strong, consideration of any finite-domain assignments will be postponed until a promising component STP has been constructed.

[1]Recall that DTP solving involves selecting a disjunct for each constraint to construct a component STP.

[2]This pseudocode significantly simplifies a practical implementation, since the DTP pruning techniques (such as semantic branching and removal of subsumed variables) as well as traditional CSP techniques are not illustrated, but these are still very powerful and should be used.

This algorithm remains slightly unappealing in one aspect. Despite the fact that there exists a strong connection between the temporal and finite-domain constraints, not much has yet been done to exploit this hybridization. In the top $|C|$ levels of search, the various finite-domain constraints are, for the most part, ignored entirely, and only at the bottom $|X|$ levels are they considered at all. In contrast, the most successful types of constraint hybridizations make some attempt to share information between both sets of constraints while search progresses.

Forward Checking and Propagation of Bounds

Forward checking is one of the most basic mechanisms for dead-end detection and pruning in CSPs. It works by examining the domains of all unassigned variables against the current assignment, removing any values discovered to be inconsistent. As is the case with most CSPs, forward checking is an essential component in any DTP solving algorithm such as Epilitis. If any disjunct of a currently uninstantiated constraint is inconsistent with the underlying temporal network, it can be removed from that constraint's domain, and backtracking will occur if all the disjuncts of a constraint become obliterated. Forward checking can be performed in $O(v + |X|^2)$ time, where X is the set of all of time points, and v is the number of remaining legal values, if the all-pairs shortest paths matrix is maintained and updated using incremental full path consistency (Mohr & Henderson 1986).

In our combined constraint system, forward checking can be applied to more than just the (weakened) disjuncts. Suppose we have just induced the constraint $P_S - J_E \leq 15$. There may be entries in the bounds matrix for another disjunct, say $J_E - P_S \leq B$, that are not consistent with this new constraint. For instance, the bounds of -20 and -35 on this difference are not feasible, and such entries can be removed from the bounds matrix.

Removing these entries alone actually has no effect on the algorithm, since the least-commitment approach only operates on the maximum bound of any disjunct. However, consider the situation that ensues when all entries in a single row or column of the bounds matrix are deleted. In this case, the finite-domain value associated with that row or column is no longer available. As an illustration, consider the two bounds matrices in Figure 3, in which we have extended our example to include more locations. Again, suppose that the temporal network induces the constraint $P_S - J_E \leq 15$. As a result, any bound for $J_E - P_S$ must be at least -15. For the left matrix, we have crossed out all inconsistent values. If the disjunct $J_E - P_S \leq B$ is part of a current meta-CSP assignment, then the location L_6 cannot be assigned to J_E^f, nor can it be assigned to J_S^f since these are elements of the same equivalence class. Now consider the right matrix. The current maximum bound for this disjunct is 0, but this requires J_S^f to be L_6. If we propagate the constraint $L(\{J_S^f, J_E^f\}) \neq L_6$, the new maximum value becomes -5.

There are several ways to make the forward checking and bounds propagation procedures extremely efficient, such as performing a preprocessing step to create a sorted list of the possible bounds for each disjunct, and including a pointer to

J_E^t \ P_S^t	L_2	L_3	L_5
L_1	–15	–5	~~–20~~
L_3	–5	0	–10
L_6	~~–35~~	~~–35~~	~~–20~~

M_E^t \ J_S^t	L_1	L_3	L_6
L_2	–10	–5	–35
L_4	–20	–25	–15
L_6	–40	–35	0

Figure 3: Forward checking of the bounds matrix can eliminate location assignments

the current minimum-allowable value. We omit the details of how to maintain these structures due to space limitations. It should be noted that forward checking and bounds propagation process must be implemented as a cyclical procedure, since finite-domain removals and bound tightenings can directly induce additional finite-domain removals.

Partitions as a Special Case

One can expect DTP$_{FD}$'s in general to be quite difficult to solve, as they are comprised of two networks each of which being NP-hard. When forward checking and propagation of bounds are employed, ensuring consistency of the finite-domain network could require exponential time in the worst case. However, certain special cases can reduce this complexity. One such case, reflected in our running example, occurs when the finite-domain constraints over the variables form a partition – that is, where the variables are divided into a collection of $|P|$ equivalence classes. Each class is constrained so that its members must all be given the same assignment from a common set of possible values, with these being the only finite-domain constraints. Such a restriction is reasonable for many planning and scheduling scenarios, where the start and end of an activity must occur in the same location. Travel time between locations, reflecting a limited variety of spatial constraints, is still easily expressed with this additional restriction. Thus, it is useful to define a special case of a DTP$_{FD}$, called a *Partitioned DTP$_{FD}$*, where a partition replaces the arbitrary constraint network. The experimental results described in the following section are performed on instances of this special case.

Experimental Results

In this section, we describe the results of a set of experiments that were performed on three variations of Hybrilitis, an implementation of our algorithm for solving DTP$_{FD}$'s. The three variations include: 1) the brute force algorithm that enumerates finite-domain assignments, and checks each projected DTP separately, 2) the least-commitment algorithm without the additional forward checking and bounds propagation, and 3) the least-commitment algorithm with these advanced mechanisms.

Typically, DTP solvers in the past have been benchmarked by using a random generator described in (Stergiou & Koubarakis 1998). This generator is, unfortunately, insufficient for our purposes, as it does not generate the finite-domain constraints or conditional bounds that we wish to represent. In response, we created a generator that takes as

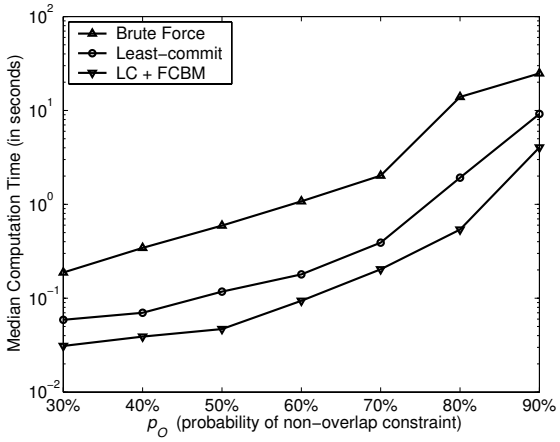

Figure 4: Median computation time required by the three algorithms

arguments the parameters $\langle A, L, B, p_L, p_O, C_{\max}\rangle$, where A is the number of activities, L is the number of locations, B is a bounds matrix shared by all conditional disjuncts, p_L is the probability that a location will appear in the set of allowable locations for an activity, p_O is the probability that a non-overlap constraint will be enforced between any pair of activities, and $C_{\max}$ is the maximum allowable makespan of the schedule (where in any solution, all activities must occur between times 0 and $C_{\max}$). With the addition of the temporal reference point, the number of time points in these test cases is $2 * A + 1$. For our experiments, we used the parameters $A \in \{8, 9, ..., 13\}$, $L = 6$, $p_L = 33\%$, and $p_O \in \{30\%, 40\%, ..., 90\%\}$, and created 50 test cases for each combination of parameter settings. The activities are all 30 minutes in length, and entries in the bounds matrix B range from 5 to 40 minutes of travel time (except for the zeros along the diagonal). To ensure that all problems were feasible yet nontrivial, we calculated the minimum possible $C_{\max}$ for each test case (by running Hybrilitis multiple times with smaller horizons until the problem became inconsistent), and subsequently enforced this makespan for all tests. Our implementations of Hybrilitis were developed in Java, and our experiments were conducted on a Windows XP machine with a 3 GHz processor and 1 GB of memory.

In Figure 4, we plot the median computation time of the three algorithms as a function of the probability of non-overlap p_O, since this parameter influences how constrained – and thus, how hard – the problem is. For these experiments, the number of activities A was fixed at 10. Note that the y-axis is shown on a logarithmic scale. Not surprisingly, the worst of these three algorithms is the brute force approach, which on the most constrained problems, requires a median time of roughly 24.8 seconds. The least-commitment algorithm does significantly better (9.17 seconds), and after adding forward checking of the bounds matrix, the time drops to 4.04 seconds. One can clearly see that this difference in performance is relatively consistent for other values of p_O.

		Number of Activities					
	%ile	8	9	10	11	12	13
BF	25^{th}	.031	.031	.031	.063	.219	2.30
	50^{th}	.093	.125	.453	1.01	3.81	15.7
	75^{th}	.219	.640	3.84	3.86	26.2	152
LC	25^{th}	.031	.016	.031	.125	.157	.797
	50^{th}	.031	.047	.125	.531	1.21	5.48
	75^{th}	.063	.141	.422	3.33	10.8	69.7
LC+FC	25^{th}	.031	.031	.047	.063	.125	.250
	50^{th}	.031	.046	.062	.086	.250	.476
	75^{th}	.032	.062	.063	.172	.797	2.19

Figure 5: Computation time (in seconds) at quartile boundaries for the three algorithms

In Figure 5, we fix the probability of non-overlap constraint p_O at 50%, and vary the number of activities to evaluate how well these algorithms scale with problem size. We report the computation time at the boundaries of the quartiles for the 50 test cases. As an example, we found that 75% of all problems with $A = 13$ can be solved in under 152 seconds with the brute force algorithm. While the least-commitment algorithm reduces this to 69.7 seconds, a more substantial drop to 2.19 seconds is achieved when the forward checking and bounds propagation mechanisms are enabled (almost two orders of magnitude faster than the brute-force technique). Thus, this more advanced algorithm is particularly good at coping with problems falling at the more difficult end of the spectrum.

Discussion and Future Work

In this paper, we have presented a method for augmenting instances of the Disjunctive Temporal Problem (DTP) with finite-domain constraints. In this hybridization, the bounds of the temporal constraints are made to be conditional on the finite-domain assignment. We have also described a special case of this formalism in which a partition on the finite-domain variables is established, and have shown how a limited form of spatial constraints can be represented with this approach. We have introduced a least-commitment approach for efficiently reasoning about this combined constraint system, and have improved upon this by developing a specialized forward checking mechanism to aid in search space pruning. Experimental results demonstrate the effectiveness of our hybrid algorithm.

As mentioned earlier, conditional bounds can express more than simple spatial relationships such as the travel time required between pairs of locations. A finite-domain assignment can potentially have an effect on a number of other temporal constraints. For instance, the duration of an activity such as *WatchTV* may range from thirty minutes to two hours, depending on what program is being watched. In addition, there exist several generalizations that could greatly enhance the power of the DTP_{FD}, such as decoupling the (currently fused) finite-domain and temporal variables, as well as allowing the bound of a conditional disjunct to depend on the entire finite-domain assignment as a whole (instead of just a single pair of finite-domain components). Further development of the expressivity allowed by this hybrid formalism is a topic worthy of continued research.

Acknowledgments

The authors thank Neil Yorke-Smith and Tomas Uribe for their invaluable input into preliminary versions of this work. This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. NBCHD030010 and the Air Force Office of Scientific Research under Contract No. FA9550-04-1-0043. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the view of DARPA, the Department of Interior-National Business Center, or the United States Air Force.

References

Armando, A.; Castellini, C.; Giunchiglia, E.; and Maratea, M. 2004. A SAT-based decision procedure for the boolean combination of difference constraints. In *Proceedings of the 7th International Conference on Theory and Applications of Satisfiability Testing (SAT-2004)*.

Bockmayr, A., and Kasper, T. 1998. Branch-and-infer: A unifying framework for integer and finite domain constraint programming. *INFORMS Journal on Computing* 10(3):287 – 300.

Dechter, R.; Meiri, I.; and Pearl, J. 1991. Temporal constraint networks. *Artificial Intelligence* 49(1-3):61–95.

Dechter, R. 2003. *Constraint Processing*. Morgan Kaufmann.

Jaffar, J., and Maher, M. J. 1994. Constraint logic programming: A survey. *Journal of Logic Programming* 19/20:503–581.

Laborie, P. 2003. Resource temporal networks: Definition and complexity. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, 948–953.

Mittal, S., and Falkenhainer, B. 1990. Dynamic constraint satisfaction problems. In *Proceedings of the 8th National Conference on Artificial Intelligence (AAAI-90)*, 25–32.

Mohr, R., and Henderson, T. C. 1986. Arc-consistency and path-consistency revisited. *Artificial Intelligence* 28:225–233.

Oddi, A., and Cesta, A. 2000. Incremental forward checking for the disjunctive temporal problem. In *Proceedings of the 14th European Conference on Artificial Intelligence (ECAI-2000)*, 108–112.

Stergiou, K., and Koubarakis, M. 1998. Backtracking algorithms for disjunctions of temporal constraints. In *Proceedings of the 15th National Conference on Artificial Intelligence (AAAI-98)*, 248–253.

Tsamardinos, I., and Pollack, M. E. 2003. Efficient solution techniques for disjunctive temporal reasoning problems. *Artificial Intelligence* 151(1-2):43–90.

Weld, D. S. 1994. An introduction to least commitment planning. *AI Magazine* 15(4):27–61.