Strong Controllability of Disjunctive Temporal Problems with Uncertainty

Bart Peintner[1], Kristen Brent Venable[2], and Neil Yorke-Smith[1]

[1] Artificial Intelligence Center, SRI International. {peintner,nysmith}@ai.sri.com
[2] University of Padova, Italy. kvenable@math.unipd.it

Abstract. The Disjunctive Temporal Problem with Uncertainty (DTPU) is an extension of the Disjunctive Temporal Problem (DTP) that accounts for events not under the control of the executing agent. We investigate the semantics of DTPU constraints, refining the existing notion that they are simply disjunctions of STPU constraints. We then develop the first sound and complete algorithm to determine whether Strong Controllability holds for a DTPU. We analyze the complexity of our algorithm with respect to the number of constraints in different classes, showing that, for several common subclasses of DTPUs, determining Strong Controllability has the same complexity as solving DTPs.

1 Introduction

The Simple Temporal Problem (STP) [1] is a temporal constraint formalism widely used for modeling and solving real-world planning and scheduling problems. Several extensions of this framework have been proposed in the literature. The Disjunctive Temporal Problem (DTP) [2] allows for non-convex and non-binary constraints by introducing disjunctions of STP constraints. The *Simple Temporal Problem with Uncertainty* (STPU) [3] extends the STP by allowing two classes of events, *controllable* and *uncontrollable*. Uncontrollable events are controlled by exogenous factors, often referred to as 'Nature'. The concept of *consistency* of an STP is replaced by varying notions of *controllability* of an STPU. The level of controllability for a problem describes the conditions under which an executor can guarantee all constraints will be satisfied, w.r.t. Nature's behavior. In problems that exhibit Strong Controllability (SC), there exists a time assignment to all events that ensures all constraints will be satisfied whatever Nature's realisation of the uncontrollable events.

The recently introduced *Disjunctive Temporal Problem with Uncertainty* (DTPU) [4] allows for both disjunctive constraints and contingent events. Such a coexistence is intrinsic to many real-world planning and scheduling problems (e.g., [5]). In this paper we focus on Strong Controllability of DTPUs, which provides an appropriate notion of solution for application domains such as production planning, and in situations where the entire schedule of actions must be known in advance. We present a sound and complete algorithm to determine whether Strong Controllability of a DTPU holds. We then analyze the complexity of the algorithm with respect to the quantity of different constraint types and we show that for several common subclasses of DTPUs, determining SC has the same complexity as solving a classical DTP without uncertainty.

2 Background

Temporal Problems A *Simple Temporal Problem* [1] is defined by a set of time-point variables X, which represent instantaneous events, and a set of quantitative constraints $\mathcal{C}$, which restrict the temporal distance between events. STP constraints are binary and convex, taking the form $X_j - X_i \in [a_{ij}, b_{ij}]$. A distinguished event, denoted *TR*, marks the start of time. Unary domain constraints thus can be modeled as binary relations to *TR*. Solving an STP equates to deciding consistency and deriving its minimal network. An STP is consistent iff it has a *solution*: an assignment to all variables such that all constraints are satisfied. Consistency can be tested with an All-Pairs Shortest Path algorithm, requiring $\mathcal{O}(n^3)$ time for n variables [1]. The *minimal network* is the tightest representation of the constraints that includes all solutions to the problem.

The *Disjunctive Temporal Problem* [2] generalizes the STP by admitting non-binary temporal relations. DTP constraints consist of disjunctions of STP constraints: $X_{j1} - X_{i1} \in [a_{i1j1}, b_{i1j1}] \vee X_{j2} - X_{i2} \in [a_{i2j2}, b_{i2j2}] \vee \cdots \vee X_{j\ell} - X_{i\ell} \in [a_{i\ell j\ell}, b_{i\ell j\ell}]$. Note that a variable may appear in more than one disjunct. While the worst-case complexity of solving a DTP is **NP**-hard [2], in practice efficient solving techniques have been developed (e.g., [6]), and tractability results are known for some classes of DTPs (e.g., [7]). A simple way to solve a DTP is to consider the *component STPs* obtained by selecting one disjunct from each constraint. A DTP is consistent iff it contains a consistent component STP. A search through the meta space of component STPs is the heart of most constraint-based DTP solvers (e.g., [6]); these solvers have been shown to be very efficient. The time complexity of the search depends on the number of component STPs. In the worst case, for m constraints with a maximum of k disjuncts each, there are $\mathcal{O}(k^m)$ component STPs, and the complexity is $\mathcal{O}(n^2 m k^m)$ [6].

Temporal Problem with Uncertainty The STP and DTP formalisms assume that all events are under the complete control of the execution agent. Recognizing that this assumption is often not valid, the *Simple Temporal Problem with Uncertainty* (STPU) [3] distinguishes two classes of variables, *controllable* V_c and *uncontrollable* V_u. The values of controllable variables are chosen by the execution agent and correspond to events in standard STPs. The values of uncontrollable variables, by contrast, are determined by exogenous factors ('Nature'); such a *realisation* is observed but cannot be controlled by the execution agent. The only information known prior to observation of an uncontrollable variable λ is that Nature will ensure that its value respects a single *contingent constraint* $\lambda - \mathbf{X} \in [\mathbf{a}, \mathbf{b}]$, with $a \geq 0$. Contingent constraints are assumed independent, and Nature is assumed to be consistent. Besides contingent constraints, which we distinguish by using a bold typeface, all other constraints in an STPU are *executable*.

The semantics of STPU constraints can be summarized as follows:

1. ***Contingent STPU constraints*** (S) model a priori information the agent is given about when an event controlled by Nature can occur (e.g., "An experiment will end (uncontrollable) between 5 and 10 minutes after it begins (controllable)").
2. ***Executable STPU constraints*** (S_e) model requirements the agent has between variables it controls and those controlled by Nature (e.g., "Data cannot be sent (controllable) until 3 minutes after the experiment ends (uncontrollable)").

3. **(Executable) STP constraints** (S_c) model temporal constraints between events under the control of the agent (e.g., "The experiment cannot start until after 2 p.m.").

Controllability of an STPU is the analogue of consistency of an STP. Three forms have been defined [3]: strong, weak and dynamic. In this paper we extend the notion of *Strong Controllability* (SC), a guarantee that a single assignment to all controllable variables in the STPU will satisfy all constraints, regardless of the realisation. The time complexity of determining strong controllability of an STPU is in class **P** [3].

3   Extending the DTP with Contingent Events

In the sequel, we will indicate with *constraint* a disjunctive temporal constraint and with *disjunct* a single simple temporal constraint. Representationally, the extension of an STPU to a DTPU is straightforward: we relax the restriction that each constraint be binary and convex, and allow disjunctive constraints [4]. Like the STPU, we divide the variables into two classes, controllable and uncontrollable, and retain the restriction that each process of uncertain duration is represented by a single contingent constraint between the process's start and end time-points.

Definition 1 (DTPU) *A* Disjunctive Temporal Problem with Uncertainty *is a tuple* $\langle V_c, V_u, \mathcal{C}, \mathcal{C}_u \rangle$, *where* V_c, V_u *are the sets of executable and uncontrollable variables, respectively,* $\mathcal{C}$ *is a finite set of disjunctive temporal constraints over* $V_c \cup V_u$, *and* $\mathcal{C}_u \subseteq \mathcal{C}$ *is the set of binary contingent constraints, one for each element of* V_u.

Notice that w.r.t. the original definition in [4], the set of contingent constraints now appears explicitly. A solution to a DTPU, $s = s_c \cup s_u$ is a complete assignment to all variables $V = V_c \cup V_u$ that satisfies all constraints in $\mathcal{C}$. The controllable part of the assignment, s_c, is the *decision*, and the uncontrollable part, s_u, is the *realisation*.

Example 1. Consider the example of a Mars rover (Figure 1) tasked with drilling into a rock (D denotes the start of the drilling action, D' its end), taking an image (I, I'), and sending data collected during each action back to Earth (S, S'). The drilling task consists of a preparation phase followed by two minutes of drilling. The preparation phase has variable duration, depending in part on whether the type of rock requires a different drill bit than is currently installed. The image task must occur at time 15, which is when the object to photograph will be visible. Drilling causes significant vibration, so the image cannot be taken during the last two minutes or during the minute after drilling ends. Only in the small window $[25, 30]$ can the rover begin data transmission.

The problem can be formalized as a DTPU as follows. The constraints describe the durations and temporal ordering between the activities. In this example, $V_c = \{TR, D, S, I, I'\}$, $V_u = \{D', S'\}$, and $\mathcal{C}$ contains nine constraints, two of which are disjunctive. The filled arrows represent contingent constraints: $\mathcal{C}_u$ contains two constraints, $\mathbf{S'} - \mathbf{S} \in [\mathbf{4}, \mathbf{5}]$ and $\mathbf{D'} - \mathbf{D} \in [\mathbf{5}, \mathbf{10}] \vee [\mathbf{15}, \mathbf{20}]$.

We define a *component STPU* P' of a DTPU P to be an STPU obtained by selecting one disjunct from each constraint. Note that P' may include only a subset of the
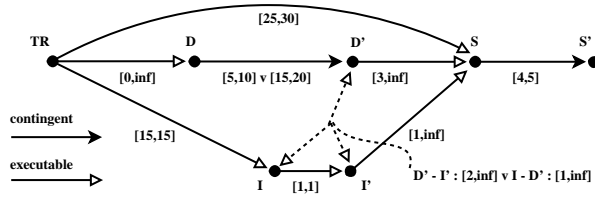
Fig. 1. Dashed arrows depict disjunctive constraints that involve more than two variables.

uncontrollable variables of P. The DTPU in Figure 1 has four component STPUs, one for each combination of disjuncts in the two disjunctive constraints.

In addition to constraints of type S, S_c and S_e, a DTPU features the types:

1. **DTP** (D): A disjunction of two or more STP constraints (e.g., "The image action must be ended 2 minutes before drilling begins (controllable) or the image action must be started after drilling begins (controllable)").
2. **Executable DTPU** (D_e): A disjunction of two or more executable STPU disjuncts (e.g., "The image action must end 2 minutes before drilling ends (uncontrollable) or the image action must start at least 1 minute after drilling ends (uncontrollable)").
3. **Mixed executable DTPU** (D_{me}): A disjunction of STP and executable STPU constraints (e.g., "The image action must end before drilling starts (controllable) or the image action must start at least 1 minute after drilling ends (uncontrollable)").
4. **Contingent DTPU** (D_c): A disjunction of two or more contingent STPU constraints. *Nature chooses which disjunct will be satisfied.* (e.g., "Drilling can take 5–10 minutes or 15–20, depending on whether the correct bit is installed").

It is important to recognize that by choosing a duration for uncertain processes (i.e., choosing a time for an uncontrollable variable), Nature is in effect 'choosing' a disjunct from the contingent constraint modeling that process.

An eighth constraint type would be *mixed DTPU constraints* that contain both contingent and executable disjuncts. Such constraints are outside our definition of a DTPU, which specifies exactly one contingent constraint to model each uncontrollable variable. Contingent constraints model the behavior of Nature: what the modeler declares *will* happen, not what should happen. Thus, by definition, one of contingent disjuncts of a mixed DTPU constraint will always be satisfied, making the executable disjuncts superfluous. If there were a realisation not satisfying one of the contingent disjuncts, then the model of Nature would be incomplete.

4 Strong Controllability of a DTPU

We now address the problem of testing the Strong Controllability of a DTPU. Motivating examples for SC include production planning, where schedules must be known in advance because of possible dependencies among activities performed by other agents, and safety-critical domains where advanced approval is mandated of the schedule. Additionally, when the executing agent lacks resources to reason about dynamic controllability, as may be the case for a Mars rover, it can operate with SC.

Let us partition the constraints $\mathcal{C}$ into three groups: multi-disjunct contingent constraints, $\mathcal{C}_C = \{D_c\}$; multi-disjunct executable constraints with at least one executable STPU disjunct, $\mathcal{C}_E = \{D_e, D_{me}\}$; and all other constraints, $\mathcal{C}_S = \{S, S_e, S_c, D\}$. Note that $\mathcal{C}_S$ includes all single disjunct constraints and all multi-disjunct constraints over controllable variables. In Example 1, $\mathcal{C}_C = \{\mathbf{D'} - \mathbf{D} \in [\mathbf{5}, \mathbf{10}] \vee [\mathbf{15}, \mathbf{20}]\}$, $\mathcal{C}_E = \{D' - I' \in [2, \infty) \vee I - D' \in [1, \infty)\}$, and the remaining seven constraints in $\mathcal{C}_S$.

In general, it is not true that a DTPU is SC if it contains an SC component STPU, since Nature can in effect choose a disjunct, and thus a decision strongly controlling only one disjunct of a contingent DTPU constraint is not sufficient. However, this relationship does hold if $\mathcal{C}_C = \oslash$, which we say is a *Simple-Natured* DTPU. The converse is false unless $\mathcal{C}_E = \oslash$, since an SC solution may satisfy different disjuncts in different realisations. When $\mathcal{C}_E = \oslash$, we say that the DTPU is *Simple-Nature-Dependent*, since all executable constraints that depend on uncontrollables are simple (single-disjunct).

We combine the idea of using the SC test for STPUs [8] with the now-standard meta-CSP search of a DTP solver [6]. In our algorithm DTPU-SC, the DTPU constraints are treated as CSP variables whose values correspond to the disjuncts of the DTPU constraint. Hence, for each DTPU constraint $C_i \in \mathcal{C}$ with disjuncts $D(C_i)$, a meta-CSP variable C_i is created with a domain consisting of the disjuncts $c_{ij} \in D(C_i)$.

Pseudocode of DTPU-SC is given in Algorithm 1. The three methods correspond to the three classes of constraints, $\mathcal{C}_S$, $\mathcal{C}_C$, and $\mathcal{C}_E$, defined above. Initially, DTPU-SC is called with partial component STPs A and A_C empty, i.e., DTPU-SC $(\oslash, \oslash, \mathcal{C}_S, \mathcal{C}_C, \mathcal{C}_E)$. DTPU-SC either returns the empty set, indicating that DTPU P is not SC, or it returns a set of decisions S such that any $s_c \in S$ strongly controls P.

Since finding a single SC component STPU is not sufficient for determining SC of DTPUs, DTPU-SC must efficiently keep track of which component STPUs each assignment controls during search. To help explain the operation of the algorithm, we will first look at three special cases. Then we will address the general case.

Case 1: STPU First, consider the trivial case in which all constraints are single-disjunct. Since only $\mathcal{C}_S$ contains constraints, and all these constraints are simple, the DTPU P is an STPU; thus SC can be determined with the existing STPU algorithm [8].

Case 2: Totally Simple Second, consider the case in which multi-disjunct constraints are present, but the sets $\mathcal{C}_C$ and $\mathcal{C}_E$ are empty. In this common case, all constraints involving uncontrollable variables have only a single disjunct. The DTPU is Simple-Natured, and moreover is also Simple-Nature-Dependent; altogether, we describe the DTPU as *Totally Simple*. An example of a Totally Simple DTPU is $V_d = \{TR, X\}, V_u = \{Z\}, \mathcal{C} = \{X - TR \in [1, 2] \vee [5, 8], \mathbf{Z} - \mathbf{TR} \in [\mathbf{8}, \mathbf{10}], Z - X \in [1, 5]\}$.

By the above, we can check SC by checking each component STPU for SC. If a decision controls any component STPU, it controls the DTPU as well. It is straightforward to adapt existing meta-CSP DTP solvers to implement this algorithm. The first method in Algorithm 1 does exactly this; the call to the second method, ALL-PATHS-SC, in line 4 simply returns the minimal network of A in the case of a Totally Simple DTPU. Before the incremental consistency check for a disjunct over an uncontrollable variable (line 10), we convert the disjunct into a set of STP constraints using a call to sc-transform, which is a transformation adapted from STPUs [8]. sc-transform converts

---
Algorithm 1 Determine SC, return controlling minimal network

DTPU-SC(A, A_C, $\mathcal{C}_S$, $\mathcal{C}_C$, $\mathcal{C}_E$)

1: $S \leftarrow \varnothing$
2: **if** $\mathcal{C}_S = \varnothing$ **then**   *{ A is a disjunct combination of C_S }*
3:    $G \leftarrow$ minimal-network(A)
4:    $S \leftarrow$ ALL-PATHS-SC(A, A_C, $\mathcal{C}_C$, $\mathcal{C}_E$, G)
5: **else**
6:    $C_i \leftarrow$ select-variable($\mathcal{C}_S$), $\mathcal{C}'_S \leftarrow \mathcal{C}_S - \{C_i\}$
7:    **for** each disjunct c_{ij} of $D(C_i)$ **do**
8:       $A'_C \leftarrow A_C \cup c_{ij}$ *{ A'_C is an STP that ignores uncontrollable variables }*
9:       $A' \leftarrow A' \cup$ sc-transform(A'_C, c_{ij})   *{Transform c_{ij} into disjuncts over controllable variables}*
10:      **if** consistent(A') **then**
11:        $S \leftarrow$ DTPU-SC(A', A'_C, $\mathcal{C}'_S$, $\mathcal{C}_C$, $\mathcal{C}_E$)
12:        **if** $S \neq \varnothing$ **then return** S
13: **return** S *{Set of SC decisions or nil}*

ALL-PATHS-SC(A, A_C, $\mathcal{C}_C$, $\mathcal{C}_E$, G)

1: **if** $\mathcal{C}_C = \varnothing$ **then** $G \leftarrow G \cap$ SATISFY-$\mathcal{C}_E$(A, A_C, $\mathcal{C}_E$)
2: **else**
3:    $C_i \leftarrow$ select-variable($\mathcal{C}_C$), $\mathcal{C}'_C \leftarrow \mathcal{C}_C - \{C_i\}$
4:    **for** each disjunct c_{ij} of $D(C_i)$ **do**
5:       $A'_C \leftarrow A_C \cup c_{ij}$
6:       $A' \leftarrow A \cup$ sc-transform(A'_C, c_{ij})
7:      **if** consistent(A') **then**
8:        $G \leftarrow$ ALL-PATHS-SC(A', A'_C, $\mathcal{C}'_C$, $\mathcal{C}_E$, G)
9:        **if** $G = \varnothing$ **then return** $\varnothing$ *{Fail if ANY disjunct fails}*
10:     **else return** $\varnothing$
11: **return** G

SATISFY-$\mathcal{C}_E$ (A, A_C, $\mathcal{C}_E$)

1: $H \leftarrow \varnothing$ *{ H will hold all assignments that satisfy any combination of C_E }*
2: **if** $\mathcal{C}_E = \varnothing$ **then** $H \leftarrow$ minimal-network(A)   *{ A represents a complete component STPU }*
3: **else**
4:    $C_i \leftarrow$ select-variable($\mathcal{C}_E$), $\mathcal{C}'_E \leftarrow \mathcal{C}_E - \{C_i\}$
5:    **for** each disjunct c_{ij} of $D(C_i)$ **do**
6:       $A' \leftarrow A \cup$ sc-transform(A_C, c_{ij})   *{Returns at most 1 constr.}*
7:      **if** consistent(A') **then** $H \leftarrow H \cup$ SATISFY-$\mathcal{C}_E$(A',A_C, $\mathcal{C}'_E$)
8: **return** H
---

a single disjunct (i.e., STPU constraint) over uncontrollable variables into an equivalent set of disjuncts over controllable variables. If the input disjunct is contingent, this set can be as large as the number of constraints over uncontrollable variables; if it is executable, only a single disjunct will be produced.

The complexity of DTPU-SC for the case of a Totally Simple DTPU is the same as solving a DTP: $\mathcal{O}(n^2 S k^S)$, where $S = |\mathcal{C}_S|$. The only additions to the DTP algorithm are lines 8 and 9, both of which amortize to $\mathcal{O}(1)$ time.

Case 3: Simple-Nature-Dependent When $\mathcal{C}_C$ is non-empty, i.e., if the DTPU contains multi-disjunct contingent constraints, Nature chooses the disjunct that must be satis-

fied for each contingent constraint, and an SC decision must control *any* combination of disjuncts from $\mathcal{C}_C$. Contrary to intuition, we cannot simply break the disjunctive constraints apart. Consider the constraint $\mathbf{D'} - \mathbf{D} \in [\mathbf{5}, \mathbf{10}] \vee [\mathbf{15}, \mathbf{20}]$: if we break the constraint into two STPU constraints, they will simply be inconsistent with one another.

The second method, ALL-PATHS-SC, extends the partial component STP A (generated by the first method, i.e., by search through the $\mathcal{C}_S$ constraints) using every feasible disjunct combination of the constraints in $\mathcal{C}_C$. For each combination, it intersects the set of decisions that control it with the set of decisions that control the previously checked combinations. Thus, ALL-PATHS-SC returns the set G of all decisions that control each combination. The call to the third method, SATISFY-$\mathcal{C}_E$, in line 1 simply returns the minimal network of A if $\mathcal{C}_E$ is empty.

For the Simple-Nature-Dependent DTPU, where $\mathcal{C}_E$ is empty and $\mathcal{C}_C$ non-empty, the time complexity is $\mathcal{O}(n^2 S k^S + n^2(S) k^C k^S)$, where $C = |\mathcal{C}_C|$. The S element describes the maximum number of consistency checks in ALL-PATHS-SC in line 7.

Case 4: General DTPU In the general case, both $\mathcal{C}_C$ and $\mathcal{C}_E$ are non-empty. The constraints in $\mathcal{C}_E$ contain uncontrollable variables, but (since they are executable constraints) only one disjunct must be satisfied for any realisation, and the agent may choose which disjunct. With $\mathcal{C}_E$ constraints, however, it is possible that a single SC decision satisfies different disjuncts for different realisations. Hence, for each decision under consideration, we must determine for each feasible disjunct combination in $\mathcal{C}_C$ whether there is a feasible disjunct combination in $\mathcal{C}_E$ as well. While this is not difficult for a single decision, DTPU-SC is searching in the meta-CSP space, and therefore is effectively reasoning about sets of decisions at once (recall that a minimal network represents a set of decisions). Hence, method ALL-PATHS-SC must maintain a potentially *non-convex* list of decisions G: the union of decisions that satisfy at least one disjunct combination of $\mathcal{C}_E$ for each disjunct combination of $\mathcal{C}_C$ considered so far.

The third method in Algorithm 1, SATISFY-$\mathcal{C}_E$, searches through all disjunct combinations of $\mathcal{C}_E$ for those consistent with the SC decisions given as input (in the partial STP A). Upon finding consistent disjunct combinations, it adds the resulting minimal network to a union H. H therefore represents a non-convex set of decisions that control the assignment to $\mathcal{C}_C$. Upon return, in line 1 of ALL-PATHS-SC, H is intersected with G, assuring that G contains only assignments that control all combinations of $\mathcal{C}_C$ considered so far. We note that efficient data structures and simplification methods for G and H are key to reducing the complexity in practice of the repeated intersections.

In the general case, DTPU-SC considers $k^S k^C k^E$ disjunct combinations, where $E = |\mathcal{C}_E|$. Therefore, SATISFY-$\mathcal{C}_E$ requires $\mathcal{O}(n^2 E k^E k^C k^S)$. Analyzing the intersection performed in line 1 of ALL-PATHS-SC yields $\mathcal{O}(n^2 k^S S k^{S^{k^C}})$. Since testing consistency of a DTP is **NP**-hard, it is no surprise that deciding SC with DTPU-SC is thus in the complexity class **EXPSPACE**.

Case 5: Simple-Nature Finally, we specialize the general case to the Simple-Natured DTPU, where $\mathcal{C}_E$ is non-empty but $\mathcal{C}_C$ is empty, i.e., all contingent constraints are simple. Using the above analysis, now $C = 0$, reducing the complexity to $\mathcal{O}(n^2 k^S (S + E k^S k^E))$. We suspect that Simple-Nature DTPUs occur frequently in practice because the need to model processes with uncertain, non-convex durations is relatively rare.

5 Related and Future Work

Uncertainty in planning and scheduling tasks has been addressed in the literature in different ways. For example in the *Conditional Temporal Problem* [9] it is the presence in the problem of the variables (all of which are controllable) that is unknown. A similar type of decision-point based uncertainty is captured in the *Temporal Plan Network* [10], where decision nodes are used to explicitly introduce choices in activity execution that the planner must make. Outside of the literature descended from the STP, there is a large body of work on fuzziness and uncertainty in temporal reasoning (e.g., [11]). Another interesting relation is that between DTPUs and Quantified CSPs [12]. Informally, Strong Controllability corresponds to a forall-then-exists quantification over uncontrollable and controllable variables, respectively.

For the future, we aim to analyze dynamic and weak controllability of DTPUs and derive algorithms for testing them. We would also like to explore the potential of mixed DTPU constraints with suitable semantics to capture decision-point based uncertainty.

References

1. Dechter, R., Meiri, I., Pearl, J.: Temporal constraint networks. Artificial Intelligence **49** (1991) 61–95
2. Stergiou, K., Koubarakis, M.: Backtracking algorithms for disjunctions of temporal constraints. Artificial Intelligence **120** (2000) 81–117
3. Vidal, T., Fargier, H.: Handling contingency in temporal constraint networks: From consistency to controllabilities. JETAI **11** (1999) 23–45
4. Venable, K.B., Yorke-Smith, N.: Disjunctive temporal planning with uncertainty. In: Proc. of IJCAI'05. (2005) 1721–1722
5. Muscettola, N., Nayak, P.P., Pell, B., Williams, B.C.: Remote Agent: To boldly go where no AI system has gone before. Artificial Intelligence **103** (1998) 5–47
6. Tsamardinos, I., Pollack, M.E.: Efficient solution techniques for disjunctive temporal reasoning problems. Artificial Intelligence **151** (2003) 43–89
7. Satish Kumar, T.K.: On the tractability of restricted disjunctive temporal problems. In: Proc. of ICAPS'05. (2005) 110–119
8. Vidal, T., Ghallab, M.: Dealing with uncertain durations in temporal constraint networks dedicated to planning. In: Proc. of ECAI-96. (1996) 48–52
9. Tsamardinos, I., Vidal, T., Pollack, M.E.: CTP: A new constraint-based formalism for conditional, temporal planning. Constraints **8** (2003) 365–388
10. Kim, P., Williams, B.C., Abramson, M.: Executing reactive, model-based programs through graph-based temporal planning. In: Proc. of IJCAI'01. (2001) 487–493
11. Dubois, D., Fargier, H., Prade, H.: Fuzzy scheduling. European J. of Operational Research **147** (2003) 231–252
12. Bordeaux, L., Monfroy, E.: Beyond NP: Arc-consistency for quantified constraints. In: Proc. of CP'02. (2002) 371–386