Controllability of Soft Temporal Constraint Problems

Francesca Rossi[1], Kristen Brent Venable[1], and Neil Yorke-Smith[2]

[1] University of Padova, Italy. {frossi,kvenable}@math.unipd.it
[2] IC–Parc, Imperial College, London, UK. nys@icparc.ic.ac.uk

Abstract. In real-life temporal scenarios, uncertainty and preferences are often essential, coexisting aspects. We present a formalism where temporal constraints with both preferences and uncertainty can be defined. We show how three classical notions of controllability (strong, weak and dynamic), which have been developed for uncertain temporal problems, can be generalised to handle also preferences. We then propose algorithms that check the presence of these properties and we prove that, in general, dealing simultaneously with preferences and uncertainty does not increase the complexity beyond that of the separate cases. In particular, we develop a dynamic execution algorithm, of polynomial complexity, that produces plans under uncertainty that are optimal w.r.t. preference.

1 Motivation

Research on temporal reasoning, once exposed to the difficulties of real-life problems, can be found lacking both expressiveness and flexibility. To address the lack of expressiveness, preferences can be added to the temporal framework; to address the lack of flexibility to contingency, reasoning about uncertainty can be added. In this paper we introduce a framework to handle both preferences and uncertainty in temporal problems. This is done by merging the two pre-existing models of *Simple Temporal Problems with Preferences* (STPPs) [5] and *Simple Temporal Problems under Uncertainty* (STPUs). [12]. We adopt the notion of controllability of STPUs, to be used instead of consistency because of the presence of uncertainty, and we adapt it to handle preferences.

The proposed framework, *Simple Temporal Problems with Preferences and Uncertainty* (STPPUs), represents temporal problems with preferences and uncertainty via a set of variables, which represent the starting or ending times of events (which may be controllable or not), and a set of soft temporal constraints over the variables. Each constraint includes an interval containing the allowed durations of the event or the allowed interleaving times between events, and a preference function associating each element of the interval with a value corresponding to how much its preferred. Such soft constraints can be defined on both controllable and uncontrollable events.

Examples of real-life problems with temporal constraints, preferences, and uncertainty can easily be found in several application domains (e.g. [7]). Here we describe in detail one such problem, arising in an aerospace application domain. The problem refers to planning for fleets of *Earth Observing Satellites* (EOS) [4]. This planning problem involves multiple satellites, hundreds of requests, constraints on when and how to service each request, and multiple resources. Scientists place requests to receive earth images from space. After the image data is acquired by an EOS, it can either be downlinked in real time or recorded on board for playback at a later time. Ground stations or other satellites are available to receive downlinked images. Each satellite can communicate only with a subset of other satellites and/or ground stations, and transmission rates differ. Further, there may be different costs associated with using different communication resources. In [4], the EOS scheduling problem is dealt with through a constraint-based interval representation. Candidate plans are represented by variables and constraints which reflect the temporal relationships and ordering decisions between actions.

This problem contains all the aspects we address in this paper. It has temporal constraints which include duration and ordering constraints associated with the data collecting, recording, and downlinking tasks. Moreover, solutions are preferred based on objectives such maximising the number of high priority requests serviced, maximising the expected quality of the observations, and minimising the cost of downlink operations (notice that most of these preferences can be directly translated into preferences on durations of tasks). Finally, there is uncertainty due to weather: specifically to the duration and persistence of cloud cover, since image quality is reduced by the amount of cloud cover over the target. In the rest of the paper, we will use EOS as a running example to illustrate our STPPU framework.

The specific contributions of this paper are: a way to model simple temporal problems with both uncertainty and preferences; definitions of strong, weak, and dynamic controllability for such problems; an overall view of the logical relationship among these notions; algorithms to check controllability, complexity results, and a general scheme which guides in the use of the algorithms. In particular, we show that checking dynamic controllability of a Simple Temporal Problem with Preferences and Uncertainty can be done in polynomial time. Thus we prove that adding preferences does not make the problem more difficult, given that uncontrollable events may occur.

Most proofs have been omitted for lack of space. This paper is a revised and updated version of [15], a poster paper focused mainly on strong controllability.

2 Background

In a *Temporal Constraint Problem* [2], variables denote timepoints and constraints represent the possible temporal relations between them. The constraints are quantitative, describing restrictions either on durations of events or on distances (interleaving times) between events, in terms of intervals over the timeline. In general such problems are **NP**-complete. However, if each temporal constraint has just one interval — hence the constraints have form $l_{ij} \leq x_j - x_i \leq u_{ij}$, where the x denote timepoints — then we have a *Simple Temporal Problem* (STP) that can be solved in polynomial time by enforcing path consistency. An STP is said to be *path consistent* iff any consistent assignment to two variables can be extended to a consistent assignment to three variables. Path consistency in the context of STPs is enforced by performing two operation on temporal intervals: *intersection* $(\oplus)$ and *composition* $(\otimes)$. Intersection is defined as the usual interval intersection. Composition is defined as: $I_1 \otimes I_2 = \{t = a + b \,|\, a \in I_1, b \in I_2\}$. Given an STP and a constraint c_{ij} on variables x_i and x_j with interval I_{ij}, it is possible to show that the STP is path consistent iff $\forall i, j$, $I_{ij} \subseteq I_{ij} \oplus (I_{ik} \otimes I_{kj})$ $\forall k$ [10].

To address the lack of flexibility in execution of standard STPs, the *Simple Temporal Problem under Uncertainty* (STPU) framework [12] divides the timepoints into two classes: *executable* (or requirement) and *contingent* (or uncontrollable). The former, as in an STP, are decided by the agent, but the latter are decided by ‘Nature’: the agent has no control over when the activity will end; it observes rather than executes. The only information known prior to observation is that Nature will respect the interval on the duration. Durations of contingent links are assumed independent.

Controllability of an STPU is the analogue of consistency of an STP. Controllable implies the agent has a means to execute the timepoints under its control, subject to all constraints. Three notions have been proposed in [12]. Firstly, an STPU is *Strongly Controllable* (SC) if there is a fixed execution strategy that works in all *situations* (an observation of all contingent timepoints). Checking strong controllability is in **P** [12]. Secondly, an STPU is *Dynamically Controllable* (DC) if there is an online execution strategy that depends only on observed timepoints in the past and that can always be extended to a complete schedule whatever may happen in the future. Checking dynamic controllability is also in **P**; and we will call the algorithm proposed in [8] to test this property CHECK-DC. Thirdly, an STPU is *Weakly Controllable* (WC) if there exists at least one execution strategy for every situation. Checking weak controllability is co-**NP** [12]. The three notions are ordered by their strength: Strong $\Rightarrow$ Dynamic $\Rightarrow$ Weak.

Separately, to address the lack of expressiveness in standard STPs, the *Simple Temporal Problem with Preferences* (STPP) framework [5] merges STPs with semiring-based soft constraints. Soft temporal constraints are specified by means of a *preference function* on the constraint interval, $f : [l, u] \rightarrow A$, where A is a set of preference values. The set A is part of a semiring. Recall that a *semiring* is a tuple $\langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$ such that: A is a set and $\mathbf{0}, \mathbf{1} \in A$; $+$ is commutative, associative and $\mathbf{0}$ is its unit element; $\times$ is associative, distributes over $+$, $\mathbf{1}$ is its unit element and $\mathbf{0}$ is its absorbing element. Further, in a *c-semiring* $+$ is idempotent, $\mathbf{1}$ is its absorbing element and $\times$ is commutative. In [1] semiring-based soft constraints are presented as a unifying, expressive framework since they can model various types of preferences and different ways of aggregating them. In this paper we will use, as underlying structure for handling preference, the *fuzzy* semiring $S_{\text{FCSP}} = \langle [0, 1], \max, \min, 0, 1 \rangle$.

In general, STPPs are **NP**-complete. However, under certain restrictions — if preference functions are semi-convex (i.e. having at most one peak), constraints are combined via an idempotent operation (like $\min$), and preference values are totally ordered (like $[0, 1]$) — then finding an optimal solution of an STPP is a polynomial problem [5].

Two solvers for STPPs with these restrictions are presented by [9]. We now outline CHOP-SOLVER, since some of the algorithms we propose here employ a similar *chopping procedure*. This procedure takes an STPP satisfying the above conditions, and a preference level β, and returns an STP (i.e. a problem without preferences). For each soft constraint of the STPP, $c = \langle [l, u], f \rangle$, there is a hard constraint on the same variables in the STP, $c' = [l', u'] = \{t : t \in [l, u], f(t) \geq \beta\}$. The consistency of the STP obtained can be tested, e.g. by enforcing path consistency. We recall that path consistency applied to an STP leaves the problem (if consistent) in its *minimal* form: its intervals contain only elements that belong to at least one solution. CHOP-SOLVER takes as input an STPP and searches for the highest preference level, *opt*, at which chopping the STPP gives a consistent STP; *opt* can be found performing a binary search in the preference interval $[0, 1]$. It can be easily proven that all and only the solutions of the STP obtained chopping at *opt* are optimal solutions of the input STPP (i.e. it has no solution with preference $>$ *opt*). Thus the minimal problem obtained by enforcing path consistency at level *opt* is minimal also with respect to the optimal solutions of the STPP. Summarising, CHOP-SOLVER takes an STPP and returns the optimal level *opt* and an STP in minimal form containing all and only the optimal solutions of the given STPP.

3 Simple Temporal Problems with Preferences and Uncertainty

An intuitive definition of an STPPU is an STPP for which the timepoints are partitioned into two classes, executable and contingent, just as in an STPU. Symmetrically an STPPU can be viewed as an STPU to which preference functions are added. Contingent constraints become *soft contingent constraints* and requirement constraints become *soft requirement constraints*.

Both soft contingent and requirement constraints are defined as a pair $\langle I, f \rangle$, where (as before) the interval $I = [l, u]$ contains the possible durations or distances between the two constrained timepoints, and $f_{ij} : I \rightarrow A$ is a *preference function* mapping each element of the interval into an element of the preference set of the semiring $S = \langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$. Since we are assuming the fuzzy semiring as a underlying structure, the optimisation criterion in this paper is to maximise the minimum preference obtained by an assignment on any constraint. Although it may appear conservative, this 'weakest link' criterion applies in many cases and it is attractive computationally because of the idempotency of the multiplicative operator $\min$. Further, without loss of generality, and following the assumptions made for STPUs [8], we may assume no two contingent constraints end at the same timepoint.

In both types of constraints, the preference function represents the preference of the agent on the duration of an event or on the distance between two events. The difference is that, while for soft requirement constraints the agent has control and can be guided by the preferences in choosing values for the timepoints, for soft contingent constraints the preference represents merely a desire of the agent on the possible outcomes of Nature; there is no control on the outcomes. We will illustrate shortly.

We can now state formally the definition of an STPPU:

Definition 1 (STPPU). *A* Simple Temporal Problem with Preferences and Uncertainty (STPPU) *P is a tuple $P = (N_e, N_c, L_r, L_c, S)$ where: N_e is the set of executable points; N_c is the set of contingent points; L_r is the set of soft requirement constraints over semiring S; L_c is the set of soft contingent constraints over semiring S; and $S = \langle A, +, \times, \mathbf{0}, \mathbf{1} \rangle$ is a c-semiring.*

Once we have a complete assignment to all timepoints we can compute its global preference. This is done according to the semiring-based soft constraint schema: first we project the assignment on all soft constraints, obtaining an element of the interval and the preference associated to that element; then we combine all the preferences obtained with the multiplicative operator of the semiring ($\min$ in this paper). Given two assignments with their preference, the best is chosen using the additive operator ($\max$ in this paper). An assignment is *optimal* if there is no other with a higher preference.

A solution of an STPPU, a *schedule*, is a complete assignment of values (times) to all the timepoints, executable and contingent, that satisfies the constraints with preference ≥ 0. We can distinguish two parts of a schedule: a *situation* ω, the duration of all contingent events, and a *control sequence* δ, the set of assignments to all executable timepoints. One the one hand, ω represents the assignments made by Nature; on the other hand, δ represents the decisions made by the agent. We define $Sol(P)$ to be the set of all schedules. Given a situation ω, the *projection* P_ω of an STPPU P is the STPP obtained by replacing in P the contingent events with their values in ω and the associated preferences. We indicate with $opt(P_\omega)$ the preference value of an optimal solution

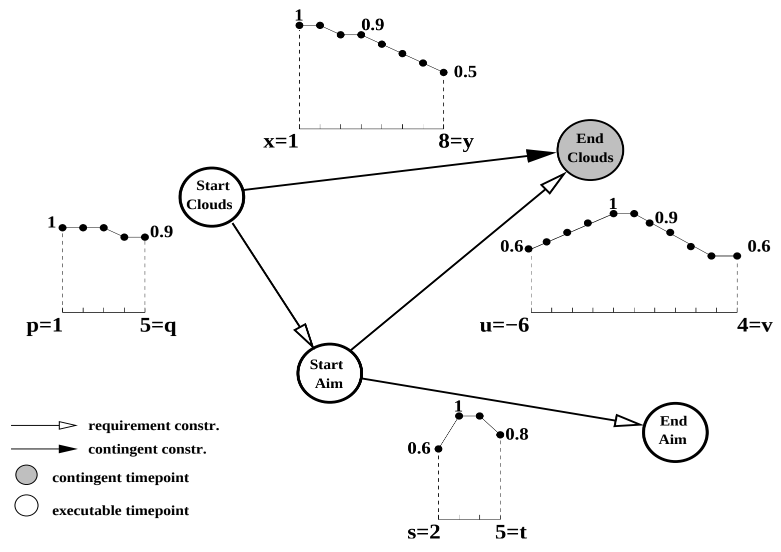

Fig. 1. Example STPPU from the Earth Observing Satellites domain

of P_ω. We define $Proj(P)$ to be the set of all projections. Then a *strategy* is a map S: $Proj(P) \rightarrow Sol(P)$ such that for every projection P_ω, $S(P_\omega)$ is a schedule which includes ω. Regarding notation, given an executable timepoint x, we will write $[S(P_\omega)]_x$ to indicate the value assigned to x in $S(P_\omega)$, and $[S(P_\omega)]_{<x}$ to indicate the durations of the contingent events that finish prior to x in $S(P_\omega)$.

Example 1. Consider as an example the following scenario from the Earth Observing Satellites domain [4] described in Sect. 1. Suppose a request for observing a region of interest has been received and accepted. To collect the data, the instrument must be aimed at the target before images can be taken. It might be, however, that for a certain period during the time window allocated for this observation, the region of interest is covered by clouds. The earlier the cloud coverage ends the better, since it will maximise both the quality and the quantity of retrieved data; but coverage is not controllable.

Suppose the time window reserved for an observation is from 1 to 8 units of time and that we start counting time when the cloud occlusion on the region of interest is observable. Suppose, in order for the observation to succeed, the aiming procedure must start before 5 units after the starting time, ideally before 3 units, and it actually can only begin after at least 1 time unit after the weather becomes observable. Ideally the aiming procedure should start slightly before the cloud coverage will end. If it starts too early then, since the instrument is activated immediately after it is aimed, clouds might still occlude the region and the image quality will be poor. On the other hand, if it waits until the clouds have disappeared then precious time during which there is no occlusion will be wasted aiming the instrument instead of taking images. The aiming procedure can be controlled by the mission manager and it can take anywhere between 2 and 5 units of time. An ideal duration is 3 or 4 units, since a short time of 2 units would put the instrument under pressure, while a long duration, like 5 units, would waste energy.

This scenario, rather tedious to describe in words, can be compactly represented by the STPPU shown in Fig. 1 with the following features: (1) a set of executable timepoints SC, SA, EA; (2) a contingent timepoint EC; (3) a set of soft requirement constraints on $\{\mathsf{SC} \to \mathsf{SA}, \mathsf{SA} \to \mathsf{EC}, \mathsf{SA} \to \mathsf{EA}\}$; (4) a soft contingent constraint $\{\mathsf{SC} \to \mathsf{EC}\}$; and (5) the fuzzy semiring $S_{\text{FCSP}} = \langle [0,1], \max, \min, 0, 1\rangle$.

A solution of the STPPU in Fig. 1 is the schedule $s = \{\mathsf{SC} = 0, \mathsf{SA} = 2, \mathsf{EC} = 5, \mathsf{EA} = 7\}$. The situation associated with s is the projection on the only contingent constraint, $\mathsf{SC} \to \mathsf{EC}$, i.e. $\omega_s = 5$, while the control sequence is the assignment to the executable timepoints, i.e. $\delta_s = \{\mathsf{SC} = 0, \mathsf{SA} = 2, \mathsf{EA} = 7\}$. The global preference is obtained considering the preferences associated with the projections on all constraints, that is $\mathrm{pref}(2) = 1$ on $\mathsf{SC} \to \mathsf{SA}$, $\mathrm{pref}(3) = 0.6$ on $\mathsf{SA} \to \mathsf{EC}$, $\mathrm{pref}(5) = 0.9$ on $\mathsf{SA} \to \mathsf{EA}$ and $\mathrm{pref}(5) = 0.8$ on $\mathsf{SC} \to \mathsf{EC}$. The preferences must then be combined using the multiplicative operator of the semiring, which is $\min$, so the global preference of s is 0.6. Another solution $s' = \{\mathsf{SC} = 0, \mathsf{SA} = 4, \mathsf{EC} = 5, \mathsf{EA} = 9\}$ has global preference 0.8. Thus s' is a better solution than s according to the semiring ordering since $\max(0.6, 0.8) = 0.8$. $\square$

4 Strong and Weak Controllability with Preferences

We now consider how it is possible to extend the notion of controllability to accommodate preferences. In general we are interested in the ability of the agent to execute the timepoints under its control not only subject to all constraints but also in the best possible way w.r.t. preferences. It transpires the meaning of 'best possible way' depends on the types of controllability we introduced earlier.

Definition 2 (Optimal Strong Controllability). *An STPPU P is* Optimally Strongly Controllable *(OSC) iff there is an execution strategy S s.t. $\forall P_\omega \in Proj(P)$, $S(P_\omega)$ is an optimal solution of P_ω, and $[S(P_1)]_x = [S(P_2)]_x$, $\forall P_1, P_2$ projections and for every executable timepoint x.*

In other words, an STPPU is OSC if there is a fixed control sequence that works in all possible situations and is optimal in each of them. In the definition, 'optimal' means that there is no other assignment the agent can choose for the executable timepoints that could yield a higher preference in any situation. Since this is a powerful restriction, we can instead look at just reaching a certain quality threshold:

Definition 3 (α-Strong Controllability). *An STPPU P is α-*Strongly Controllable *(α-SC), with $\alpha \in A$ a preference, iff there is a strategy S s.t.:*

1. *$\forall P_\omega \in Proj(P)$, $S(P_\omega)$ is a solution of P_ω such that $[S(P_1)]_x = [S(P_2)]_x$, $\forall P_1, P_2$ projections and for every executable timepoint x; and*
2. *the global preference of $S(P_\omega)$ is at least preference $opt(P_\omega)$ if $opt(P_\omega) \leq \alpha$.*

In other words, an STPPU is α-SC if there is a fixed control sequence that works in all situations and results in optimal schedules for those situation where the optimal preference level of the projection is $\leq \alpha$. Clearly, OSC implies α-SC $\forall\alpha$.

It is possible to check whether an STPPU is Optimally Strongly Controllable or α-Strongly Controllable in polynomial time [15]. To check if an STPPU is OSC we chop it at every preference level, starting from the minimum preference. At each level we obtain a different STPU; the SC of each STPU is checked using the algorithm proposed in [12], which reduces the problem to checking the consistency of an STP only on the executable timepoints. The original STPPU is OSC iff at each preference level the STP obtained is consistent and, considering each constraint of the STPs, the intersection of its intervals across all preference values is non-empty.

Notice, however, that it is very unlikely for an STPPU to be OSC, since OSC is opt-SC where opt is the optimal preference value of the STPPU considered as an STPP. The algorithm proposed for checking OSC can be easily modified to check, given a certain preference α, if the STPPU is α-SC; or even to find the highest α such that this property is satisfied. For example, the STPPU in Fig. 1 is not OSC since there is no choice for SA that will be optimal whatever happens on the contingent constraint SC $\rightarrow$ EC. However the strategy that assigns a time to SA that is 4 units after that assigned to SC works in all possible situations and is optimal for all those situations that have an optimal value of their projection ≤ 0.9. Hence the STPPU in Fig. 1 is 0.9-SC.

Secondly, we extend similarly Weak Controllability:

Definition 4 (Optimal Weak Controllability). *An STPPU is* Optimally Weakly Controllable *(OWC) iff $\forall P_\omega \in Proj(P)$ there is a strategy S_ω s.t. $S_\omega(P_\omega)$ is an optimal solution of P_ω.*

In other words, an STPPU is OWC if, for every situation, there is a fixed control sequence that results in an optimal schedule for that situation.

Optimal Weak Controllability of an STPPU is equivalent to Weak Controllability of the corresponding STPU obtained by ignoring preferences. The reason is that if a projection P_ω has at least one solution then it must have an optimal solution (i.e. one with the highest preference $opt(P_\omega)$). This also implies that an STPPU is such that its underlying STPU is either WC or not. Hence it does not make sense to define a notion of α-Weak Controllability. To check OWC, it is enough to apply the algorithm proposed in [13] to the underlying STPU.

It is easy to see that α-SC for any α implies OWC. In Sect. 8 we discuss the relations between the properties defined in this and the next section. For instance, in Example 1, from the fact that the STPPU is 0.9-SC we can derive that it is also OWC.

5 Dynamic Controllability with Preferences

Dynamic Controllability is seen as the more useful notion of controllability in practice. It addresses the ability of the agent to execute a schedule by choosing incrementally the values to be assigned to executable timepoints, looking only at the past. When preferences are available, it is desirable that the agent acts not only in a way that is guaranteed to be consistent with any possible future outcome but also in a way that ensures the absence of regrets w.r.t preferences.

Definition 5 (Optimal Dynamic Controllability). *An STPPU P is* Optimal Dynamic Controllable *(ODC) iff there is a strategy S such that $\forall P_1, P_2$ in $Proj(P)$ and for any executable timepoint x:*

1. if $[S(P_1)]_{<x} = [S(P_2)]_{<x}$ then $[S(P_1)]_x = [S(P_2)]_x$;

2. *$S(P_1)$ is a consistent complete assignment for P_1 and $S(P_2)$ is a consistent complete assignment for P_2;*
3. *$\mathrm{pref}(S(P_1))$ is optimal in P_1 and $\mathrm{pref}(S(P_2))$ is optimal in P_2.*

In other words, an STPPU is ODC is there exists a means of extending any current partial control sequence to a complete control sequence in the future in such a way that the resulting schedule will be optimal. As before, we also soften the optimality requirement to having a preference reaching a certain threshold.

Definition 6 (α-Dynamic Controllability). *An STPPU P is α-*Dynamic Controllable *(α-DC) iff there is a strategy S such that $\forall P_1, P_2$ in $Proj(P)$ such that $opt(P_1) \leq \alpha$ and $opt(P_2) \leq \alpha$, and for any executable x, the three conditions of Definition 5 hold.*

In other words, an STPPU is α-DC if there is a means of extending any current partial control sequence to a complete sequence; but optimality is guaranteed only for situations with preference less or equal to α.

6 Checking Optimal Dynamic Controllability

In this section we describe an algorithm that tests whether an STPPU is ODC, and we prove that the test is performed in polynomial time. The idea is to chop the STPPU at different preference levels, in the same way as CHOP-SOLVER does (recall Sect. 2), except now the output of this chopping procedure is an STPU. Starting from the lowest preference and moving up in the preference ordering, at each step the STPPU is chopped and the Dynamic Controllability of the STPU obtained is checked.

We next give more details on how Dynamic Controllability of an STPU can be checked in polynomial time using the algorithm proposed in [8]. We call the algorithm CHECK-DC. It works locally on triangles of simple temporal constraints (with no preferences), on two executable timepoints A and B and a contingent timepoint C. Without loss of generality, let the intervals be $[x, y]$ on the contingent constraint AC, $[p, q]$ on the requirement constraint AB, and $[u, v]$ on the other requirement constraint BC. For example, in Fig. 1, $A = \mathsf{SC}$, $B = \mathsf{SA}$, and $C = \mathsf{EC}$. The idea is to consider how executable C can be ordered w.r.t. the ending time of the contingent event represented by AC. This is determined by looking at interval $[u, v]$ of constraint BC, which contains the allowed interleaving times that can occur between B and C. Based on the signs of u and v, three different cases arise.

In the *Follow case* ($v < 0$), executable B will always follow C. If the STPU is path consistent then it is also DC since, given the time at which C occurs after A, it is always possible to find a consistent value for B. In the *Precede case* ($u \geq 0$), B will always precede or happen simultaneously with C. Then the STPU is DC if $y - v \leq x - u$ since, if so, any assignment to B, v_B, such that $v_B - v_A$, where v_A is the assignment to A, is in $[y - v, x - u] \subseteq [p, q]$, and so is consistent with any assignment to C, v_C (i.e. $v_C - v_B \in [u, v]$). In this case, interval $[p, q]$ on AB should be replaced by $[y - v, x - u]$. Lastly, in the *Unordered case* ($u < 0$ and $v \geq 0$), B can either follow or precede C. To ensure DC, B must wait either for C to occur first, or for $t = y - v$ units of time to pass after A. In other words, either C occurs and B can be activated at the first value consistent with C's time, or B can safely be executed t units of time after A's execution.

This can be described by an additional constraint which is expressed as a *wait* on AB and is written $\langle C, t\rangle$. Of course if $x \geq y - v$ then we can raise the lower bound of AB to $y - v$ (*unconditional Unordered reduction*), and in any case we can raise it to x if $x > p$ (*general Unordered reduction*). Waits can be propagated, or *regressed*, from one constraint to another. For example, a wait on AB may induce a wait on other constraints involving A, e.g. AD, depending on the type of constraint DB.

Algorithm CHECK-DC [8] applies these rules to all triangles in the STPU and propagates all possible waits. This propagation is based on the intuition that a wait that must be respected by some executable, B, can (1) affect the allowed execution times of another related (via some constraint path) executable, say B'; or (2) can impose a new, possibly, longer wait on B'. The actual rules that guide such propagation are rather complex and cannot be thoroughly described here due lack of space; we refer to [8].

If no inconsistency is found — no requirement interval becomes empty and no contingent interval is squeezed (i.e. removing possible uncontrollable outcomes) — then the STPU is DC. In that case, the algorithm returns an STPU where some constraints may have waits to satisfy, and the intervals contain only the elements that appear in at least one possible dynamic strategy, i.e. the minimal form of the STPU.

Returning now to the algorithm we propose for testing ODC, the basic idea is to move, bottom up, in the preference set chopping the STPPU and testing the DC of each STPU obtained. If at a given level the STPU is DC then its minimal form is saved. Once a preference level is found, say γ, such that the corresponding STPU is not DC, then the STPPU will not be β-DC for all $\beta \geq \gamma$; thus we can stop the chopping procedure. At this point, the intuitive idea is too keep only elements that guarantee DC at all preference levels lower than γ, performing an 'intersection' of the minimal STPUs obtained up to $\gamma - 1$. If such intersection is non-empty then the STPPU is $(\gamma - 1)$-DC. Applying CHOP-SOLVER to the STPPU will give the highest preference, opt, of any complete assignment. If $\gamma - 1 = opt$ then the STPPU is also ODC.

We now define formally what we have sketched above.

Theorem 1. *Consider an STPPU P and the STPU Q^α obtained chopping P at level α and applying path consistency. Let $[p^\alpha_{AB}, q^\alpha_{AB}]$ be the interval obtained on any requirement constraint AB, by applying* CHECK-DC *to Q^α, and let $\langle C, t^\alpha\rangle$ be its wait, if any. Then P is ODC iff the following conditions hold: (1) for every α, Q^α is DC; (2) for each requirement constraint AB, the intersection $I = \bigcap_\alpha [t^\alpha_{AB}, q^\alpha_{AB}]$ is not empty.* □

We refer to the two conditions in the theorem as property **M**. It is not hard to see why **M** is necessary for ODC. First, assume the first condition **M**1 does not hold. Since $Proj(Q^\alpha) \subseteq Proj(P)$, and since $Proj(Q^\alpha)$ contains all the projections that have optimum preference at least α, then, since Q^α is not DC, there is no global strategy S for P such that $S(P_1)$ has optimal preference in P_1 and $S(P_2)$ in P_2, $\forall P_1, P_2$ with preference at least α. This allows us to conclude that P is not ODC. Secondly, assume instead that **M**1 holds but **M**2 does not. Each requirement constraint AB might, at different preference levels α, be squeezed in different ways and have different waits t^α_{AB}. At each level α the sub interval $[t^\alpha_{AB}, q^\alpha_{AB}]$ contains the assignments for B that are consistent with all future values of any contingent timepoint and that guarantee a preference $\geq \alpha$. If intersecting these sub-intervals across all preference levels gives an empty interval, then there is no unique value for B that satisfies the properties mentioned above (consistency and optimality) for every preference level. Thus, P is not ODC.

Table 1. First column: preference level α. Next three columns: intervals obtained chopping at all preference levels the constraint triangle on variables SC, SA, and EC shown in Fig. 1. Last column: subintervals obtained on constraint $\mathsf{SC} \rightarrow \mathsf{SA}$ by applying CHECK-DC and considering only elements following the wait

preference	$(\mathsf{SC} \rightarrow \mathsf{EC})^{\alpha}$	$(\mathsf{SC} \rightarrow \mathsf{SA})^{\alpha}$	$(\mathsf{SA} \rightarrow \mathsf{EC})^{\alpha}$	$[t^{\alpha}_{\mathsf{SC,SA}}, q^{\alpha}_{\mathsf{SC,SA}}]$
0.5	$[1, 8]$	$[1, 5]$	$[-6, 4]$	$[4, 5]$
0.6	$[1, 7]$	$[1, 5]$	$[-6, 4]$	$[3, 5]$
0.7	$[1, 6]$	$[1, 5]$	$[-5, 2]$	$[4, 5]$
0.8	$[1, 5]$	$[1, 5]$	$[-4, 1]$	$[4, 5]$
0.9	$[1, 4]$	$[1, 5]$	$[-3, 0]$	$[4, 5]$
1	$[1, 2]$	$[1, 3]$	$[-2, -1]$	$[3, 3]$

Example 1 is not ODC. Consider the triangle of constraints on variables SC, SA, EC where EC is the only contingent timepoint. Table 1 shows the intervals obtained on each of the constraints chopped at all preference levels. In the last column we show intervals $[t^{\alpha}_{AB}, q^{\alpha}_{AB}]$ where constraint AB is $\mathsf{SC} \rightarrow \mathsf{SA}$.

It is easy to see that chopping the problem at any preference level gives a STPU that is Dynamically Controllable. At preference level 1 we have an instance of the Follow case, so consistency is equivalent to controllability. At levels 0.5 and 0.7–0.9, SA will either have to wait for the cloud coverage to end or wait for at least 4 units of time after the clouds have been observed at first (i.e. after SC). At preference level 0.6, SA must wait only for 3 time units. However the STPPU is not ODC since the intersection of the intervals in the last column is empty. Consider the scenario in which the clouds coverage lasts for 2 units of time. Since this is consistent with SA occurring 3 unit after SC and this gives a solution with preference value 1, the optimal preference of the projection of situation $\omega = 2$ on contingent constraint $\mathsf{SC} \rightarrow \mathsf{EC}$ is 1. However if we execute SA at 3 time units after SC and EC happens, say, at 4 units after SC, then the solution obtained has preference 0.8, which is not optimal for the STPP corresponding to situation $\omega = 4$ on contingent constraint $\mathsf{SC} \rightarrow \mathsf{EC}$ (which has 0.9 as optimal preference value, the preference of the solution obtained in which SA is executed 5 units of time after SC). This shows that there is no way of dynamically assigning values to the executables to guarantee optimality in every possible situation.

6.1 Determining ODC by checking property M

We have just informally shown that property **M** is necessary for ODC. To see that it is also sufficient for ODC, we present the pseudocode of an algorithm that, given as input an STPPU P, checks if **M** is satisfied. We then show that if the algorithm reports success then P is ODC. Fig. 6.1 shows the pseudocode of the algorithm that checks **M**.

Algorithm CHECK-ODC takes as input an STPPU P. It checks whether the STPU Q obtained ignoring the preference functions, *IP*(P) (line 1), is DC, by applying CHECK-DC (line 2). If it is not the case, there is no chance for the original STPPU to be ODC. Otherwise, CHOP-SOLVER is applied to *IU*(P), i.e. to P considered as an STPP (line 4). This allows us to find the global optimal preference opt. At this point the algorithm checks if the STPUs obtained by chopping P and applying path consistency with PC-2 at different preference levels are DC (lines 5–9). It starts bottom-up from preference α_{min} up to

```
CHECK-ODC(triangular STPPU $P$)
 1  STPU $Q \leftarrow IP(P)$
 2  if CHECK-DC($Q$) returns 'not DC'
 3    then return 'not ODC'
 4  CHOP-SOLVER($IU(P)$)
 5  for $\alpha \leftarrow \alpha_{min}$ to $opt$
 6  do STPP $Q^\alpha \leftarrow$ PC-2(CHOP($P, \alpha$))
 7      if CHECK-DC($Q^\alpha$) returns 'not DC'
 8        then return 'not ODC'
 9      save $DC(Q^\alpha)$
10  for requirement links $AB$ $s.t.$ $\exists\alpha : t^\alpha_{AB} < q^\alpha_{AB}$
11  do $I \leftarrow \bigcap_\alpha [t^\alpha_{AB}, q^\alpha_{AB}]$
12      if $I = \emptyset$
13        then return 'not ODC'
14      $I_{AB} \leftarrow [\min_\alpha\{p^\alpha_{AB}\}, \min_\alpha\{q^\alpha_{AB}\}]$
15      wait $t_{AB} \leftarrow \max_\alpha\{t^\alpha_{AB}\}$
16  PC-2($IP(IU(P))$)
```

Fig. 2. Algorithm for checking ODC of an STPPU by determining whether **M** holds

opt, where α_{min} is the minimum preference on any constraint.[3] If at any level the DC test fails, the algorithm stops and returns failure (line 8). The dynamically controllable versions of the STPUs at each the preference level are saved (line 9).

Now, for every preference level α, each requirement constraint AB will have interval $I^\alpha_{AB} = [p^\alpha_{AB}, q^\alpha_{AB}]$ and a wait t^α_{AB} which is the maximum wait that B must satisfy for any related contingent timepoint C, for the subproblem at preference level α. CHECK-ODC thus considers each requirement constraint and checks if it is in an Unordered or Precede case with at least one contingent timepoint at some preference level (line 10). If so, the algorithm computes the intersection of the sub-intervals $[t^\alpha_{AB}, q^\alpha_{AB}]$ (line 11). If this intersection is empty then the algorithm again stops and returns failure (line 13). While checking this property, the algorithm updates P, replacing the interval on AB with $I_{AB} = [\min_\alpha\{p^\alpha_{AB}\}, \min_\alpha\{q^\alpha_{AB}\}]$, the preference function with its restriction to this interval, and imposing wait $t_{AB} = \max_\alpha\{t^\alpha_{AB}\}$ (lines 14 and 15).

The last step is to apply path consistency to the STP obtained ignoring preferences and uncertainty (line 16). This reduces the intervals by leaving out the elements that are not in any solution of the STP. Running path consistency in line 16 does not squeeze any contingent interval since it is possible to show that no element of a contingent interval loses its support, because of the reductions in line 14.

6.2 Executing an ODC STPPU

We now reconsider the execution algorithm, EXECUTE, presented in [8], in light of preferences. Adapting it, we show that if given as input an STPPU which has passed CHECK-ODC, it produces dynamically an optimal execution strategy. After initial propagation from the start timepoint, EXECUTE performs iteratively the following steps until

[3] In general the minimum preference, α_{min}, will be assigned to an element of a contingent constraint, since it is reasonable to leave in the requirement constraints only elements with a preference higher than that of the worst possible uncontrollable outcome.

the execution is completed. First, it immediately executes any executable timepoints that have reached their upper bounds. Then it arbitrarily picks and executes any executable timepoint x such that (1) the current time is within its bounds; (2) all the timepoints, y, that must be executed before x have been executed; and (3) all waits on x have been satisfied. The effect of the execution is propagated. The current time is advanced, propagating the effect of any contingent timepoints that occur. For an STPPU, the only difference is that the propagation now involves also preferences. Propagating the effect of an execution implies chopping the problem at the minimum preference level obtained by that execution on any constraint up to the current time. We denote EXECUTE equipped with this new type of propagation as EXECUTE-ODC.

Assuming that an STPPU P has passed CHECK-ODC, then the corresponding STPU obtained ignoring preferences, $Q = \mathit{IP}(P)$, is DC. Let P' be the STPPU returned by running CHECK-ODC on P, and Q' be the STPU returned by procedure CHECK-DC on Q. Observe that the contingent intervals in $\mathit{IP}(P')$ and Q' are exactly the same (in fact they are those of P and Q respectively); both algorithms CHECK-ODC and CHECK-DC leave contingent intervals unchanged, as required by the definitions of ODC and DC. Further, given a requirement constraint AB and and a contingent timepoint C, if $\langle C, t\rangle$ is the wait in P' and $\langle C, t'\rangle$ is the wait in Q', then $t \geq t'$ (intuitively: the wait necessary to be optimal may be longer than the one necessary to be only consistent). These relationships allow us to inherit directly from [8] that running EXECUTE-ODC on STPPU P' cannot fail due to any of the following events: a deadlock caused by a wait that never expires, an un-respected wait, or a squeezed contingent interval.

At this point we know that a dynamic schedule is completed by the execution algorithm, so it only remains to prove that it is optimal.

Theorem 2. *If STPPU P has successfully passed* CHECK-ODC *then the dynamic schedule produced by* EXECUTE-ODC *is optimal.* □

Since all waits have been respected, it must be $([S(P)]_B - [S(P)]_A) \geq t_{AB}$, for any requirement constraint AB. This means that $([S(P)]_B - [S(P)]_A) \in \bigcap_\alpha [t^\alpha_{AB}, q^\alpha_{AB}]$. But hence $f_{AB}([S(P)]_B - [S(P)]_A) \geq \alpha = \mathit{opt}_P$ since P has passed CHECK-ODC.

We now consider the complexity of CHECK-ODC. Assuming there is a maximum number of points R in an interval, in [8] it is shown that checking DC of an STPU can be done in time $O(n^2R)$. If there are a finite number ℓ of different preference levels, we can state the following result for CHECK-ODC, since the complexity of applying CHOP-SOLVER in line 5 of algorithm, $O(n^3\ell R)$, dominates that of all other steps.

Theorem 3. *The complexity of determining ODC of an STPPU with n variables, ℓ preference levels, and intervals of maximum size R is $O(n^3\ell R)$.* □

7 Checking α-Dynamic Controllability

Optimal Dynamic Controllability is a strong property. Instead, one may be satisfied by knowing, for a given preference value α, whether the STPPU is α-Dynamically Controllable. As stated earlier, this implies that a control sequence can be incrementally created only on the basis of past assignments. It is guaranteed to be optimal if the 'partial' preference obtained by combining the preferences on contingent constraints is smaller or equal to α. In other words, what is guaranteed is that, up to preference

Table 2. Solutions of the STPPU in Fig. 1. First four rows: assignments to the variables. Fifth row: global preference of the solution. Last row: optimal preference level of the STPP that is the projection of the corresponding situation

SC	0	0	0	0	0	0	0	0
EC	1	2	3	4	5	6	7	8
SA	4	4	4	4	4	4	4	4
EA	7	7	7	7	7	7	7	7
pref	0.9	0.9	0.9	0.9	0.8	0.7	0.6	0.5
opt	1	1	0.9	0.9	0.8	0.7	0.6	0.5

$$\begin{tikzcd}
\text{OSC} \arrow[r, leftrightarrow] \arrow[d] & \textit{opt}\text{-SC} \arrow[r] \arrow[d] & \alpha\text{-SC} \arrow[r] \arrow[d] & \alpha_{min}\text{-SC} \arrow[r] \arrow[d] & 0\text{-SC} \arrow[r] \arrow[d] & \text{SC} \arrow[d] & & \\
\text{ODC} \arrow[r, leftrightarrow] & \textit{opt}\text{-DC} \arrow[r] & \alpha\text{-DC} \arrow[r] & \alpha_{min}\text{-DC} \arrow[r] & 0\text{-DC} \arrow[r] & \text{DC} \arrow[r] & \text{WC} \arrow[r, leftrightarrow] & \text{OWC}
\end{tikzcd}$$

Fig. 3. Comparison of controllability notions. α_{min} is the smallest preference over any constraint; $opt \geq \alpha \geq \alpha_{min}$

level α, there is always a choice for the executables that will not worsen the overall complexity; for preferences above α this might not be the case.

We can state a theorem analogous to that presented for ODC. The two conditions in Theorem 1 now must hold only for all $\beta \leq \alpha$, rather than for all α. In fact, an STPPU is ODC iff it is opt-DC, where opt is the optimal preference value of STPPU P considered as an STPP. Consequently, the algorithm used to test α-DC is exactly the same as CHECK-ODC, except for line 4, which is no longer necessary (since we do not need to compute value opt), and for line 5, where the **for** loop need only go up to preference level α and not opt. Thus the worst case complexity of checking α-DC is the same as that of checking ODC.

A further query we might ask is: what is the highest level α at which P is α-DC? This query can be answered by modifying CHECK-ODC in order to find the highest preference level at which a dynamic schedule exists. We will call this algorithm MAX-α-DC. The only change needed is in line 12. Assuming the intersection proceeds bottom-up from the lowest preference α_{min}, if a preference level β is found such that the intersection becomes empty, the algorithm does not stop; instead it saves β and continues until it has considered all the requirement constraints. It then returns the minimum of all such β preferences found, β_{min}. It is easy to see that β_{min} is the highest preference α_{max} level for which there exists a dynamic schedule. Again the complexity of this algorithm is the same as that of checking ODC and hence is polynomial.

Consider now our Example 1 and the results in Table 1. It is easy to see that the STPPU shown is, for example 0.7-DC and 0.8-DC. The highest preference α such that it is α-DC is 0.9. In fact, if we choose to assign to SA either 4 or 5 units of time (i.e. any element in the intersection of intervals $[t^{\alpha}_{\mathsf{SC,SA}}, q^{\alpha}_{\mathsf{SC,SA}}]$ for α from 0.5 to 0.9), the preference of the complete solution is at least greater or equal to that of the corresponding projection, for those projections that have optimal preference ≤ 0.9. We obtain the set of solutions represented in Table 2, according to the value assigned by Nature to EC.

8 Comparing and Using Controllability

In all, we have introduced five new notions of controllability. In Fig. 3 we show the relations between them. The proofs of such relations, while omitted for lack of space, are rather immediate.

As a general strategy given an STPPU, the first property to check is OSC. This can be done in polynomial time. If the problem is OSC, the solution obtained is valid and optimal in all possible situations. However, OSC is a strong property and holds infrequently. If the problem is not OSC, the highest preference level α for which α-SC holds can be found in polynomial time by using the algorithm described in Sect. 4. If such preference level is not satisfactory (for it can be very low), then we can turn to checking ODC or β-DC for $\beta > \alpha$. Both these things can be done in polynomial time.

If the problem is not even Dynamically Controllable, but the situation can be known (just) before execution, then the last possibility is to check WC, which is equivalent to OWC. This will at least allow the agent to know in advance if there is a way to cope with every situation. However, checking WC is not necessarily in **P** [12].

9 Related Work

Temporal reasoning is a diverse area, dividing roughly into qualitative and quantitative approaches. For the Temporal Constraint Problem (TCP), a survey is [10], which also discusses some of the hybrid qualitative/quantitative approaches. The Simple Temporal Problem under Uncertainty, which the STPPU builds on, introduces uncertainty due to contingent events. Besides STPUs, uncertainty has been introduced into the TCP with possibilistic reasoning; [14] is one framework for *Fuzzy Temporal Constraint Problems*. In principle, vagueness due to soft preferences can also be accommodated with fuzzy constraints. Closer to STPPUs, contingent events can be ascribed an explicit probability distribution (rather than the implicit uniform distribution of a STPU), to yield *Probabilistic STPs* [11]; preferences are not part of this framework. Note, in contrast to STPPUs, the complexity of solving both Fuzzy TCP and Probabilistic STPs is not polynomial in general. Probabilistic temporal reasoning is found outside the TCP, for instance in temporal synthesis [6], where the amount of memory required for a dynamic solving strategy is also considered. Separately, for the general CSP, [3] introduced the distinction between controllable and uncontrollable variables, to yield *mixed CSP*. Many authors have introduced preferences, to yield soft CSP frameworks: for instance the unifying semiring-based soft CSPs [1] which the STPPU builds on.

10 Summary

Temporal constraint problems in the real world feature both preferences and uncertainty. In this paper we have introduced the Simple Temporal Problem with Preferences and Uncertainty and defined five levels of controllability. We have provided algorithms to determine whether the different levels hold, and shown that the complexity of checking controllability in a STPPU is the same as that for the equivalent notion in a STPU. In particular, the key notion of Dynamic Controllability can be tractably extended to account for preferences.

We have implemented and tested the algorithms for OSC and ODC on randomly generated problems. The experimental results show that, as expected, the time needed to check OSC and ODC grows with the size of the problem. ODC is slower than OSC on Strongly Controllable problems. However, in both cases controllability checking can be performed in reasonable time (less than 3 minutes for 500 variables). Future work is to apply the algorithms to the Earth Observing Satellites and other real-world problem domains. We are also investigating the use of probabilities over contingent constraints and their combination with preferences and uncertainty.

Acknowledgements. We thank Robert Morris and Carmen Gervet for conversations on STPPUs, the participants of the Online'03 workshop for their discussions, and the reviewers for their constructive comments. This work is partially supported by ASI (Italian Space Agency) under project ARISCOM (Contract I/R/215/02), and by the EPSRC under grant GR/N64373/01.

References

1. S. Bistarelli, U. Montanari, and F. Rossi. Semiring-based constraint solving and optimization. *Journal of the ACM*, 44(2):201–236, 1997.
2. R. Dechter, I. Meiri, and J. Pearl. Temporal constraint networks. *Artificial Intelligence*, 49:61–95, 1991.
3. H. Fargier, J. Lang, and T. Schiex. Mixed constraint satisfaction: A framework for decision problems under incomplete knowledge. In *AAAI-96*, pages 175–180, 1996.
4. J. Frank, A. Jonsson, R. Morris, and D. Smith. Planning and scheduling for fleets of earth observing satelliets. In *Proc. of 6th i-SAIRAS*, 2001.
5. L. Khatib, P. Morris, R. A. Morris, and F. Rossi. Temporal constraint reasoning with preferences. In *IJCAI'01*, pages 322–327, 2001.
6. O. Kupferman, P. Madhusudan, P. S. Thiagarajan, and M. Y. Vardi. Open systems in reactive environments: Control and synthesis. *LNCS 1877*, 2000.
7. N. Layaida, L. Sabry-Ismail, and C. Roisin. Dealing with uncertain durations in synchronized multimedia presentations. *Multimedia Tools and Applications*, 18(3):213–231, 2002.
8. P. Morris, N. Muscettola, and T. Vidal. Dynamic control of plans with temporal uncertainty. In *IJCAI'01*, pages 494–502, 2001.
9. F. Rossi, A. Sperduti, K. Venable, L. Khatib, P. Morris, and R. Morris. Learning and solving soft temporal constraints: An experimental study. In *CP'02*, pages 249–263, 2002.
10. E. Schwalb and L. Vila. Temporal constraints: A survey. *Constraints*, 3(2/3):129–149, 1998.
11. I. Tsamardinos. A probabilistic approach to robust execution of temporal plans with uncertainty. In *Second Hellenic Conference on AI (SETN'02)*, pages 97–108, 2002.
12. T. Vidal and H. Fargier. Handling contingency in temporal constraint networks. *J. Experimental and Theoretical Artificial Intelligence*, 11(1):23–45, 1999.
13. T. Vidal and M. Ghallab. Dealing with uncertain durations in temporal constraint networks dedicated to planning. In *ECAI-96*, pages 48–52, 1996.
14. L. Vila and L. Godo. On fuzzy temporal constraint networks. *Mathware and Soft Computing*, 1(3):315–334, 1994.
15. N. Yorke-Smith, K. B. Venable, and F. Rossi. Temporal reasoning with preferences and uncertainty. In *Poster paper in IJCAI-03*, pages 1385–1386, 2003.