A New Approach to Temporal Planning with Rich Metric Temporal Properties

Son Thanh To,[1] Benjamin Johnson,[2] Mark Roberts,[2] David W. Aha[3]
[1]Knexus Research Corporation, Springfield, VA 22153
[2]NRC Postdoctoral Fellow; Naval Research Laboratory, Code 5514; Washington, DC 20375
[3]Navy Center for Applied Research in AI; Naval Research Laboratory, Code 5514; Washington, DC 20375
son.to@knexusresearch.com | blj39@cornell.edu | {mark.roberts.ctr; david.aha}@nrl.navy.mil

Abstract

Temporal logics have been used in autonomous planning to represent and reason about temporal planning problems. However, such techniques have typically been restricted to either (1) representing actions, events, and goals with temporal properties or (2) planning for temporally-extended goals under restrictive assumptions. We introduce Mixed Propositional Metric Temporal Logic (MPMTL) where formulae are built over mixed binary and continuous real variables. We introduce a planner, MTP, that solves MPMTL problems and includes a SAT-solver, model checker for a polynomial fragment of MPMTL, and a forward search algorithm. We extend PDDL 2.1 with MPMTL syntax to create MPDDL and an associated parser. The empirical study shows that MTP outperforms the state-of-the-art PDDL+ planner SMTPlan+ on several domains it performed best on and MTP performs and scales on problem size well for challenging domains with rich temporal properties we create.

1 Introduction

Temporal planning has attracted the attention of researchers in the robotics and AI communities. Numerous systems have advanced temporal planning (Benton, Coles, and Coles 2012; Coles *et al.* 2009; 2010; 2008; Della Penna *et al.* 2009; Do and Kambhampati 2003; Dvorak et al. 2014; **?**; Penberthy and Weld 1994). However, these systems typically consider limited temporal properties. For example, they may assume that action conditions and effects exist at only the beginning, the end, or over the entire duration of an action, but not some other point or interval. Furthermore, the temporal goals they might support are limited to goals with a deadline.

Other approaches (Bacchus and Kabanza 2000; 1998; Bauer and Haslum 2010; Kabanza and Thiebaux 2005; Patrizi, Lipoveztky, and Geffner 2013; Patrizi *et al.* 2011) extend classical planning with temporally extended goals, typically expressed in some temporal logic formalism (e.g., Linear Temporal Logic), while assuming other properties of the problem are restricted to conditions of classical planning (e.g., no durative actions, no temporal conditions or effects).

In this paper, we develop *Mixed Propositional Metric Temporal Logic (MPMTL)*, a hybrid of a (partial) satisfiability modulo theory (SMT) (Barrett *et al.* 2009) and a metric temporal logic (Alur and Henzinger 1989; Koymans 1990). Formulae in MPMTL are built over *mixed* binary and continuous real variables with *metric* modalities and can solve three challenges by representing and reasoning about: (1) conditions and effects for discrete and continuous variables and changes over time points and intervals other than the start and end points and durations of the actions; (2) conditions and goals over boolean and numeric variables in *second-order* temporal relations (equivalent to double time quantifiers); and (3) durative discrete and linear continuous changes in the *initial world*. To our knowledge, such a hybrid formalism has not been previously proposed, despite being essential for temporal planning with continuous change.

To demonstrate the utility of this approach, we extend PDDL 2.1 with MPMTL into a language we call MPDDL. We then present the MPMTL Temporal Planner (MTP), which includes a SAT-solver and model-checker for a polynomial fragment of MPMTL, a parser for MPDDL, and a forward search algorithm. We introduce four domains and problem instances that embody the aforementioned temporal features, and use them to demonstrate and test MTP. We compare it with the state-of-the-art temporal planner SMTPlan+ using well-known PDDL+ benchmarks on which it was reported to perform the best, as well as on one of our newly created benchmarks that can be translated into PDDL+. We found that MTP outperforms SMTPlan+, can find a correct plan of hundreds complex actions within a second for the new challenging problems, and that MTP's pruning and heuristic techniques substantially improve its performance.

2 Motivation: Domains and Scenarios

In this section we present two problem domains and use them to illustrate relevant challenges that cannot be adequately addressed by other techniques.

2.1 Domain 1: Maintaining a Relay

The first problem domain extends the one we presented in (To *et al.* 2016); it concerns the maintenance of a continuous communications relay for a specified person $p0$ (at a known disaster location), given a set of UAVs that share a given base station $b0$. We will use this domain to illustrate some of the limitations of current temporal planning approaches.

Challenge 1: Durative actions with intermediate effects. To continuously maintain a communications relay, multiple UAVs must coordinate so that at least one provides a relay at all times. To provide the relay, a UAV must fly from $b0$ to $p0$'s location and hover; it can remain at $p0$ for only a finite amount of time before returning to $b0$ to refuel. When one UAV leaves $p0$ another must arrive to maintain the continuous relay. We model each UAV v with a known fuel capacity $(fuel\_cap\ v)$ and specified fuel consumption rates when transiting $(fly\_rate\ v)$ and hovering $(hover\_rate\ v)$. The takeoff, landing, and refueling of a UAV at $b0$ each takes a non-zero amount of time, and can be performed by only one UAV at any given time.

The $relay$ action is a complex, durative action; it is challenging to model in current temporal planners. It can be modeled in PDDL 2.1 with a set of sub-actions (e.g., takeoff, fly to $p0$, hover, fly to $b0$, land) using *clips* to tie each subaction together (Fox and Long 2004). However, this can be difficult to use, create an unnatural model, massively increase the search space and computational complexity due to the number of subactions and their permutations during plan construction, and give rise to subtle, semantic issues (Smith 2003; Fox and Long 2006; Coles and Coles 2014). Using PDDL+ to decompose a complex action into instantaneous actions, processes, and events could be a better solution, yet it still encounters similar problems (as will show). Another possible method is to express it as an HTN action (e.g., in ANML (Smith, Frank, and Cushing 2008)). However, this requires synchronizing HTN action instances because they affect the same variables. This synchronization problem is challenging due to temporal constraints and has not been thoroughly studied.

Challenge 2a: Second-order metric temporal goal. Because of the initial world state, and the UAVs' durative actions, the relay cannot be established at the initial time, but must instead be defined such that it starts at or before a specified goal time t_g. The length of time that the UAVs must maintain the relay is substantially longer than the length of time that any one UAV can maintain it, due to their physical constraints (e.g., fuel capacity). This requires multiple UAVs to alternate providing the relay for different sub-intervals.

If the relay need only be established, and not maintained, then the goal can be expressed with a single temporal quantifier: $\exists t \in [0, t_g] relayed$. Many languages can adequately describe this goal. The additional duration d_g required to describe the problem of *maintaining* the relay instead requires that expressing the goal using two nested, temporal quantifiers: $\exists t \in [0, t_g] \forall t' \in [0, \in [0, d_g](att + t' relayed)$. Expressing such a goal directly in existing temporal planners is not, to our knowledge, possible. In contrast, MPMTL represents this goal: $\Diamond_{[0,t_g]} \langle relayed, [0, d_g] \rangle$.

Challenge 2b: Second-order metric temporal constraint for quantity. To maintain a continuous relay during the switch between UAVs, more than one UAV must hover over $p0$ for some non-zero period of time. However, longer durations will result in a higher probability of UAV collisions. Thus, we define a maximum period of time d_{tr} during which more than one vehicle may hover over $p0$. If d_{tr} is set to zero, this constraint can be easily expressed in PDDL 2.1 and PDDL+ as *(over all (< $hover\_num$ 2))*. But when $d_{tr} > 0$, this constraint cannot be easily expressed in current approaches. MPMTL represents this constraint: $\neg \Diamond_{[start,end]} \langle hover\_num \geq 2, [0, d_{tr}] \rangle$.

Challenge 3: Durative changes in the initial world. Finally, consider a scenario in which one or more UAVs is initially en route to $b0$. The temporal changes (e.g., the intervals when the base is clear and each UAV is available, and how their fuel levels vary) depend on the initial world model. As such, they must be represented and accounted for to find a correct plan. Later we will show how such initial changes can be directly, succinctly represented in the initial model by MPDDL without using additional dummy actions, processes, events, Timed Initial Literals (TILs) (Hoffmann and Edelkamp 2005), or Numeric Timed Initial Fluents (TIFs) (Piacentini, Fox,& Long 2015), as used in other work (e.g., Tierney et al. 2012 used TILs to model ships moving in the initial state).

2.2 Domain 2: Evacuation

Our second domain concerns the evacuation of people and cargo from several cities c_i in advance of a severe storm. The storm is modeled as having different arrival times ts_i per city, where it will last for a duration of te_i. This problem requires that the people be evacuated from the dangerous cities (prior to the arrival of the storm at their city) and moved to safe cities using a set of buses b_i, each with a defined maximum capacity cap_i, where the buses may need to ferry between the cities to complete the evacuation. Problems in this domain require representating several impacting factors, such as the number of people per city, the initial locations of the buses, their capacity and speed, the distances between cities, and the (un)loading durations. The goal of safely evacuating all people from the dangerous cities can be expressed in MPMTL as $\langle (persons\_nearby\ c_i) = 0, [ts_i, te_i] \rangle$. Such a temporal goal for a numeric value cannot be represented using other temporal planners.

3 Related Work

Fox and Long (2003) proposed PDDL 2.1, which extends PDDL (McDermott 1998) to represent numeric expressions and durative actions with duration-dependent effects. In 2006, they proposed another extension of PDDL, PDDL+, for modeling mixed discrete and continuous changes through the use of processes and events. Smith et al. (2008) developed the *Action Notation Modeling Language (ANML)*, which succinctly represents finite-duration actions with numeric changes and supports a limited form of HTN actions. ANML then was first used in the development of the temporal planner FAPE (Dvorak et al. 2014). A few efforts (Do and Kambhampati 2003; Rintanen 2015) extend PDDL2.1 to allow (first order temporal) conditions and effects (for binary variables) over time points and intervals other than the start and end points and durations of the actions. We propose MPDDL, an extension of PDDL based on MPMTL (To *et al.* 2016), for representing rich temporal features in planning such as high order temporal constraints and durative complex actions for discrete and continuous variables (and changes). These features cannot be directly expressed in the other languages.

Many temporal planners use PDDL2.1 and employ a forward search with a relaxed graph plan heuristic in the state space for planning (e.g., SAPA (Do and Kambhampati 2003), CRIKEY3 (Coles *et al.* 2008) and its successors COLIN (Coles *et al.* 2009), POPF (Coles *et al.* 2010), and OPTIC (Benton, Coles, and Coles 2012)). Cushing et al. 2007 proposed to reduce temporal planning into non-temporal planning by compiling temporal actions into instantaneous actions with the same net effects for problems whose plans must contain concurrent actions. By using STN to check the consistency of temporal constraints, CRIKEY3 can solve problems that require concurrent actions of discrete changes. COLIN extended CRIKEY3 by using LP to handle concurrent actions with linear continuous changes to the same variables. MTP instead employs a forward search in the space of models with duration and quantity based heuristics for solutions.

Many temporal planners adopt PDDL+, including TM-LPSAT (Shin and Davis 2005), UPMurphi (Della Penna *et al.* 2009) and its successor DiNo (Piotrowski *et al.* 2016), SHOP2$_{PDDL+}$, dReal (Bryce *et al.* 2015), and SMTPlan+ (Cashmore *et al.* 2016). Approaches in TM-LPSAT, dReal, and SMTPlan+ and those in (Rintanen 2015; 2015b) translate PDDL+ problems into a SAT problem. SHOP2$_{PDDL+}$ extended SHOP2 (Nau *et al.* 2003) as the first PDDL+ HTN planner. UPMurphi and DiNo rely on uniform time discretization and plan validation. All these planners support non-linear continuous changes, unlike MTP. However, MTP supports high order temporal properties and durative complex actions for discrete and linear continuous changes, unlike the others.

4 Mixed Proposition Metric Temporal Logic

In this section we briefly review MPMTL (To *et al.* 2016), focusing on its use in temporal planning.

4.1 Literals and Timeline

A *literal* ℓ is a proposition p or its negation $\neg p$. A *numeric literal* is an (in)equality of the form xra, where x is a numeric (discrete or linear continuous) variable (i.e., a fluent), $r \in \{=, \neq, <, >, \leq, \geq\}$ is a relational operator, and $a = (b, c)$ is a pair of real values. For example, $(fuel\ \ v) \geq (0, 5.5)$ and $(fly\_ratev) = (3, 0)$ are numeric literals. An *interval literal* (or *i-literal*) is a pair $\langle \ell, I \rangle$, where ℓ is a literal, and I is a (time) interval. The i-literal $\langle x = (b, c), [t, t'] \rangle$ denotes that at time t, $x = c$, and then x linearly changes with the rate b until time t' (i.e., at any time $t" \in [t, t']$, $x = c + b(t"-t)$ holds). For example, the i-literal $\langle relayed, [5, 20) \rangle$ denotes that $relayed$ holds true at any time point in the interval $[5, 20)$, while $\langle (fuel\ \ v) = (-3, 80), [5, 9] \rangle$ means that at time 5 $(fuel\ \ v) = 80$ and at any time $t \in [5, 9]$ $(fuel\ \ v) = 80 - 3(t - 5)$.

An *instant literal* of an i-literal $\langle \ell, I \rangle$ at time $t \in I$, denoted by $\langle \ell, I \rangle(t)$, is the (in)equality $x\ r\ c{+}b(t{-}I^-)$, where I^- denotes the left endpoint of I, if ℓ is the pair (b, c), or the literal ℓ otherwise. An i-literal $\langle \ell, I \rangle$ implies $\langle \ell', I' \rangle$ of the same variable if $I' \subseteq I$ and $\forall t \in I'\ \langle \ell, I \rangle(t)$ implies $\langle \ell', I' \rangle(t)$. $\langle \ell, I \rangle$ contradicts $\langle \ell', I' \rangle$ if $I \cap I' \neq \emptyset$ and $\exists t \in I \cap I'$ such that $\langle \ell, I \rangle(t)$ contradicts $\langle \ell', I' \rangle(t)$.

A *timeline* μ of a variable x is a set of equality i-literals φ of the same variable such that their intervals cover $[0, \infty)$ and there does not exist a pair of those intervals that overlap. Thus, a timeline of a variable specifies the value of the variable over all the interval $[0, \infty)$.

The instant literal of a timeline μ at time t, denoted by $\mu(t)$, is the instant literal $\langle \ell, I \rangle(t)$ such that $\langle \ell, I \rangle \in \mu$ and $t \in I$. The timeline μ implies $\langle \ell, I \rangle$ if $\forall t \in I\ \mu(t)$ implies $\langle \ell, I \rangle(t)$. For example, timeline $\{\langle x{=}(1, 0), [0, 5) \rangle, \langle x{=}(0, 5), [5, \infty) \rangle\}$ implies $\langle x{=}(1, 0), [0, 4) \rangle$ and $\langle x{\leq}(1, 4), [4, \infty) \rangle$.

4.2 MPMTL Formula

Let ℓ be a literal and I be a time interval. An *MPMTL-formula* φ is defined by the following grammar:

$$\varphi ::= \top \mid \bot \mid \langle \ell, I \rangle \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \Box_I \varphi \mid \Diamond_I \varphi$$

where $\top$ and $\bot$ denote $true$ and $false$, and $\Box_I$ and $\Diamond_I$ are metric *always* and *eventually* modalities. The formulas $\Box_I \varphi$ and $\Diamond_I \varphi$ mean φ will be true after t time units for *all* $t \in I$ (i.e., universal quantifier) and *some* $t \in I$ (i.e., existential quantifier), respectively. The *left-shift* of a timeline μ by t time units, denoted by $\mu - t$, is a timeline defined as: $\mu - t = \{\langle \ell, I - t \rangle \mid \langle \ell, I \rangle \in \mu \wedge I - t \neq \emptyset\}$. Let t and t' be time points such that $t' {\geq} t$. Then we have $(\mu - t)(t' - t) = \mu(t')$.

Let $\vec{x} = \langle x_1, \ldots, x_n \rangle$ be a vector of variables. Let $M = \langle \mu_1, \ldots, \mu_n \rangle$ be a vector of *timelines* such that each μ_i is a timeline of x_i, for $i = 1, \ldots, n$. Let $M - t = \langle \mu_1 - t, \ldots, \mu_n - t \rangle$ and φ be an MPMTL-formula built over literals of variables in $\vec{x}$. Then φ is *true* in M, or M is a *model* of φ (denoted $M \models \varphi$) if: $\varphi = \Box_I \psi$ and $\forall t \in I \mid M - t \models \psi$ and $\varphi = \Diamond_I \psi$ and $\exists t \in I \mid M - t \models \psi$

4.3 Polynomial Fragment of MPMTL

An *n-order* literal is a formula of the form $\bigcirc_1 \ldots \bigcirc_{n-1} \varphi$, where φ is an i-literal and $\bigcirc_i$ denotes $\Box_I$, $\neg\Box_I$, $\Diamond_I$, or $\neg\Diamond_I$. This implies that there are n nested modalities in the n-order literal, as an i-literal $\langle \ell, I \rangle$ is a 1^{st}-order literal and it can be written as $\Box_I \ell$. Let D_k be the set of DNF formulae of n-order literals, where $n \leq k$. Then checking whether a formula φ in D_2 has a model (satisfiability) requires polynomial time, and checking whether a vector of *timelines* M of variables in φ is a model of φ also requires polynomial time. Two formulae are equivalent iff they have the same set of models. Let Γ_2 be the set of MPMTL formulae that can be converted to an equivalent formula in D. Conditions, constraints, and goals are expressed as formulas in Γ_2, and effects are represented by a set of i-literals.

5 MPMTL Representation for Planning

This section describes how MPMTL can be used to model the world, actions, problem constraints, and goals with rich temporal features as presented earlier in the scenarios. In our approach, the world is represented by a model of the variables in the problem, and temporal constraints are expressed by a MPMTL formula in Γ_2.

5.1 Representing the World Model (Timelines)

Unlike most approaches that adopt state models of the world, ours represents the world by a (1) timeline vector (*model*) and

(2) a set of constraints on the variables defined in the problem. We call such a pair a *world model*. The execution of an action will change the world model instead of a state. This allows the world to be represented as a continuous process, relevant to temporal planning with durative changes, rather than as a (sequence of) discrete *snapshot(s)* with some auxiliaries for outstanding durative changes. Also, the initial durative discrete and continuous changes can be precisely captured in the initial world model. For example, suppose UAV v_1 is en route to base b_0 in the initial world as mentioned in the relay scenario. Then the initial changes can be captured in the timelines of the corresponding variables in the initial model. The following is an example of such timelines:

$\langle \neg at(v_1, b_0), [0, 7) \rangle, \langle at(v_1, b_0), [7, \infty) \rangle$
$\langle clear(b_0), [0, 4) \rangle, \langle \neg clear(b_0), [4, 7] \rangle, \langle clear(b_0), (7, \infty) \rangle$
$\langle fuel(v_1) = (-2, 20), [0, 7] \rangle, \langle fuel(v_1) = (0, 6), (7, \infty) \rangle$

In this example, we assume v_1 will arrive at time 7, its flying consumption rate is 2, its fuel level at time 0 is 20, and the landing duration is 3 time units.

The constraint about concurrent hovering duration can be represented as $\neg\Diamond_{[0,\infty)}\langle hover\_num \geq 2, [0, d_{tr}]\rangle$ in the initial constraint set. Alternatively, this constraint can also be expressed in the relay action as presented next.

5.2 Actions

An action (schema) a is composed of a duration constraint $dur(a)$, a condition $cond(a)$, a constraint $C(a)$, and an effect eff(a). $dur(a)$ is a set of literals of the form $d\ r\ e$, where $r \in \{=, <, >, \leq, \geq\}$ and e is a numeric value or an expression of numeric variables, specifying the (range of) value of the duration d. There is no explicit distinction between instantaneous and durative actions, as the former can be defined as an action with $d = 0$ and instantaneous effect.

Condition $cond(a)$ and constraint $C(a)$ of action a is an *extension* of a MPMTL formula in Γ_2, as follows: (1) the endpoints of intervals in $cond(a)$ and $C(a)$ can be an expression of duration d and (2) the endpoints of the leftmost intervals in $cond(a)$ and $C(a)$ can be an expression of the start time t_s. The effect eff(a) is a set of extended i-literals of the form $\langle \ell, I \rangle$, where the endpoints of I can be an expression of the action start time ts and duration d, ℓ can be $x = b$ or $x = \neg x$ (only if x is a proposition), or $x+ = b$ or $x- = b$ (only if x is a numeric variable). In such an expression, b is either a constant in $\{0, 1\}$ (if x is binary) or in $\mathbb{R}$, or an expression of other numeric variables, or a pair (c, d), where c and d can be real valued or an expression of other variables. The complex $relay$ action with a UAV v, a base b, and a person p in the example scenario can then be represented as

Action: $relay(v, b, p)$
Duration: $d \geq 3 \times d_{fly}(b, p) \wedge$
$d = 2 \times d_{fly}(b, p) + (fuel(v) - 2 \times d_{fly}(b, p)$
$\times rate_{fly}(v) - fuel_{min}(v))/rate_{hv}(v)$
Condition: $\langle at(v, b), [t_s, t_s + d]\rangle\ \wedge$
$\langle avail(v), [t_s, t_s{+}d{+}d_{cool}]\rangle\wedge$
$\langle clear(b), [t_s, t_s + d_{off}]\rangle\ \wedge$
$\langle clear(b), [t_s + d{-}d_{land}, t_s + d]\rangle$
Constraint: $\langle fuel(v) \geq fuel_{min}(v), [t_s, t_s{+}d]\rangle\wedge$
$\neg\Diamond_{[t_s, t_s+d]}\langle hover\_num \geq 2, [0, d_{tr}]\rangle$
Effect: $\{\langle \neg at(v, b), (t_s, t_s{+}d)\rangle,$
$\langle \neg avail(v), [t_s, t_s{+}d{+}d_{cool}]\rangle,$
$\langle fuel(v){-}{=}(rate_{fly}(v), 0), [t_s, t_s{+}d)\rangle,$
$\langle fuel(v){+}{=}(rate_{dif}(v), 0), [t_s{+}d_{fly}(b, p), [t_s{+}d -$
$d_{fly}(b, p))\rangle,$
$\langle \neg clear(b), [t_s, t_s + d_{off}]\rangle,$
$\langle \neg clear(b), [t_s + d{-}d_{land}, t_s + d]\rangle,$
$\langle relayed, [t_s + d_{fly}(b, p), t_s + d - d_{fly}(b, p)]\rangle,$
$\langle hover\_num{+}{=}1, [t_s{+}d_{fly}(b, p), t_s{+}d{-}d_{fly}(b, p)]\rangle\}$

In this example, $fuel_{min}(v)$ is the minimum fuel level allowed for v during the scenario. The fuel consumption rates when flying and hovering are given by $rate_{fly}$ and $rate_{hv}$, and $rate_{dif}$ is their difference. The variable $d_{fly}(b, p)$ denotes the flying time of v from b to p, and d_{off}, d_{land}, and d_{cool} are the durations for v to take off, land, and cool down after a flight (before it is available for refueling or flying). The second duration constraint, above, specifies the maximum duration for the action before the fuel level is lower than the minimum value allowed, while the first duration constraint ensures that the fuel level of the UAV is sufficiently large, such that the hovering time is at least as long as half of the total flying time.

This action modeling allows for expressing all temporal properties in the example scenarios. This includes temporal constraints and effects at different times and for different durations other than the start and end points and the duration of action, as well as interval effects for discrete and linear continuous changes, and second-order temporal constraints.

Conditions: The condition of the action is similar to the precondition of a durative action in PDDL, with some semantic differences. Although the condition in our model is temporal (i.e., it must be true at the action start time and for a duration or at some time in the future), it is subject to only the current world model (which has been updated with the effects of all previously applied actions), and it is not required to hold in the new world model updated by the action. For example, the condition $\langle clear(b), [t_s + d{-}d_{land}, t_s + d]\rangle$ denotes that the base b must be clear over the entire future interval when the UAV will be landing, with respect to the current model. However, in the updated model (once the UAV has landed), this condition is no longer true due to the effect of the landing action: $\langle \neg clear(b), [t_s + d{-}d_{land}, t_s + d]\rangle$. This prevents other UAVs from taking off or landing during that time.

A similar precondition (e.g., *over all (clear b)* in PDDL 2.1), may be used to represent a property that must be true for the entire duration of the action's execution. The action constraint in this model may be used to define such a condition. For example, the constraint $\langle fuel(v) \geq fuel_{min}(v), [t_s, t_s{+}d]\rangle$ ensures that the vehicle's fuel level stays above the defined minimum, for the current model *and* all successor models. For example, during the relay action, another action that changes the UAV's fuel level (e.g., it provides fuel to another UAV) would be allowed only if the fuel constraint is not violated.

Effects: An effect of the form $\langle x{+}{=}(b, c), I\rangle$ specifies that x is increased by c at the left point I^- of the interval I, and its linear change rate is increased by b over the interval I. If $b = 0$ then the pair (b, c) can be replaced with only c. Usually

$c = 0$ if $b \neq 0$, which allows for the expression of both discrete and continuous changes, as well as their combination (i.e., both $b \neq 0$ and $c \neq 0$). If b or c is an expression of numeric variables, then the expression is evaluated with the instant value of the variables at the action start time t_s, with respect to the variable's timelines in the model. Effects with the assignment sign $=$ are called *absolute* effects while those with the sign $+=$ or $-=$ are *relative*. A variable is called *inertial* if its value remains unchanged after an action effect, until another effect changes it. In the $relay$ example, $fuel(v)$ is inertial but $relayed$ and $clear$ are not.

We adopt the following assumptions and rules:

1. Variables can only change due to the effects of actions.
2. An action is executable in a world model only if (1) its condition and duration constraint are satisfied in the model, (2) all constraints, including that of the action, hold in the resulting model after applying the action, and (3) there do not exist two absolute effects to the same variable x that simultaneously assign different values to x.
3. An absolute effect to x overrides any other effect to x (i.e., the value of x set by the absolute effect in its interval are unchanged by other effects). A time point in this interval is called a fixed point of x.
4. An effect e to an inertial proposition x will negate the outcome of all relative effects that start after e and before any absolute effect after t. The value of x will extend until it is set by another effect thereafter, and it is limited to the effect interval, otherwise. We denote by the superscript '+' an *extendable* effect to an inertial variable.
5. The linear continuous changes of relative effects are combined in their intersection(s). For example, during the time a UAV v is flying, another UAV relays fuel to v for the interval I with the rate $rfuel$. The fuel level of v will change with the combined rate $rfuel+\delta fuel(v)$ over I.
6. All numeric variables are inertial. Effects to a numeric variable x are limited to the following forms: (1) assign a constant or a line segment to x for a time interval (e.g., $\langle x = 5, [0, d] \rangle$, $\langle x = (2, 5), [0, d] \rangle$), (2) add (or subtract) a constant to x at a time (e.g., $\langle x \mathrel{+}= 2.3, [t, \infty) \rangle$), (3) add a *linear* continuous change for an interval (e.g., $\langle x \mathrel{+}= (b, 0), [0, d] \rangle$), and (4) a combination of the last two forms (e.g., $\langle x \mathrel{+}= (b, \mathbf{c}), [0, d] \rangle$). Note that the first form is absolute and the rest are relative.
7. If an effect on a numeric variable x over an interval I increases (or decreases) the value of x by δ at the right end point I^+, then its value will be increased (or decreased) by δ over the interval (I^+, ∞), if there is no absolute effect to x after I^+. Alternately, if t_f is the smallest fixed point of x after I^+, the value of x will increase (or decrease) over the interval (I^+, t_f).

5.3 MPMTL Temporal Planning Problem

An MPMTL temporal planning problem P is defined as a tuple of the form $\langle \vec{x}, A, M_I, C_I, G \rangle$, where $\vec{x}$ is a vector of variables, A is a set of actions, M_I is a timeline vector of $\vec{x}$, C_I is a (possibly empty) set of formulae in D_2 over $\vec{x}$ denoting the initial constraint, and G is a formula in D_2 over $\vec{x}$ denoting the goal.

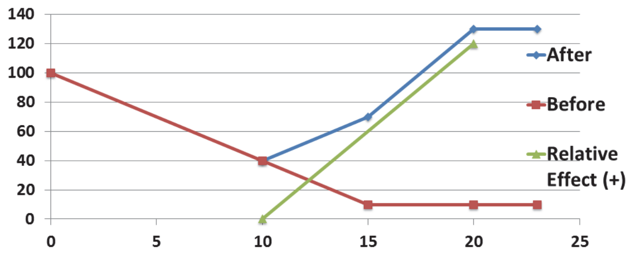

Figure 1: Update timeline of the fuel level of UAV v_2 with the relative effect $\langle (fuel\ v2) \mathrel{+}= (12, 0), [10, 20) \rangle$, suppose during v_2's flight another UAV starts to relay fuel to v_2 with the rate 12 for the interval [10,20). Thus, $(fuel\ v2)$ continuously increases with the combined rate $-6 + 12 = 6$ over interval [10,15]. The green, red, and blue lines denote the effect, the timeline before and after (partial) the update.

Instant Action Let a be an action. If the start time and the duration of a are assigned with constants t_s and d (by a planner) then the endpoints of intervals in the condition $cond(a)$, constraint $C(a)$, and effect eff(a) become constant. Furthermore, any variables in the right side of literals in these formulae are replaced by their instant value of their timelines in the current model at t_s. Hence, $cond(a)$ and $C(a)$ become formulae in D_2. We call such an action an *instant action* (or *action*, if there is no need to make such a distinction). Let M be a model and C be a constraint set. An (instant) action a is *executable* in the world model (M, C) iff a satisfies rule 2, as given above.

Update Timelines and Constraints The result of executing a in (M, C) is, intuitively, a new world model (M', C'), where C' is the new constraint set $C' = C \cup C(a)$, if a is executable in (M, C), or $null$ otherwise (rule 2). We need to define an *update* function that, given an action a and a model M, returns a new model $M' = update(a, M)$. This is defined through another function that returns a new timeline $update(e, \mu)$ given an effect e (in eff(a)) and a timeline μ of the same variable, as shown in Algorithm 1. This function can be defined based on rules 3-7. We omit a formal definition of this function, since it is straightforward, yet tedious and lengthy. Instead, one can see how this update function works in Figure 1, which shows how an effect to a real variable changes its timeline.

Due to the above assumptions/rules, one can prove that $update(a, M)$ does not depend on the order in which the effects in eff(a) are applied.

Solution A solution to a problem P is a sequence of instant actions that are executable in the initial world model (M_I, C_I) and results in a world model (M_g, C_g) such that M_g satisfies goal G.

6 MPDDL: a PDDL Extension for MPMTL

MPDDL is an extension of PDDL2.1 (Fox and Long 2003) that has been used in many state-of-the-art temporal plan-

Algorithm 1 $update(a, M)$

```
Input: Instant action $a$, timeline vector $M$
Output: Timeline vector $update(a, M)$
1: Let $X = M$
2: for each effect $e$ in eff($a$) do
3:     Let $\mu$ be the timeline in $X$ of the same variable in $e$
4:     Set $\mu = update(e, \mu)$ {replace $\mu$ with $update(e, \mu)$ in $X$}
5: end for
6: return $X$
```

ners (e.g., CRIKEY3, COLIN, POPF). We focus on the new features of MPDDL; other features and syntax of PDDL 2.1 (e.g., cost metric and preferences) remain the same. The only feature of PDDL 2.1 that this new language does not support is the non-linear continuous change, as MPMTL is subject to discrete and linear continuous changes.

New Syntax and MPMTL Formulae Expression

In MPDDL, temporal constraints in (pre)conditions, effects, constraints, and goals are all expressed via MPMTL formulae. For example, the $relay$ action is expressed in MPDDL as shown in Figure 2. The MPDDL parser supports infix syntax for expressions, (in)equalities, and formulae.

Closed and opened left endpoints (resp. right endpoints) of an interval are expressed by the characters '[' and '(' (resp. ']' and ')'). Likewise, 'and' and 'or' denote conjunction and disjunction in the prefix expression, while '&' and '|' stand for $\wedge$ and $\vee$ in the infix expression. The modalities $\Diamond$ and $\Box$ are expressed by the pairs '<>' and '[]'. A negation can be expressed by '-' instead of 'not', as shown in the second constraint of the action. The second i-literal in the condition is expressed using the always modality (i.e., '[]') instead surrounded by angle brackets. MPDDL uses several special keywords: '?ts' and '?duration' for start time and duration of an action, and 'inf' to denote infinity.

MPDDL Domain and Problem Descriptions In general, the structure of a domain remains the same as for PDDL 2.1. However, MPDDL allows some new syntax and action description as previously mentioned. Also, MPDDL supports *indirect functions* that are defined as an expression of other functions (i.e., numeric variables or values given in problem instances) (e.g., the flying time *(fly_dur ?v ?b ?p)* and consumption rate difference *(rate_dif ?v)* in Figure 2). This allows for compact and readable description of actions and problem instances.

MPDDL specifies timelines for initial changes and initial constraints expressed in MPMTL formulae. Figure 3 shows a partial problem instance for the second relay domain used in our experiments. The constraint in the problem description limits the number of UAVs that can hover concurrently and the amount of fuel used for the relay task.

7 MPDDL Temporal Planner (MTP)

We developed the MPDDL Temporal Planner (MTP), which includes a parser for MPDDL, a SAT-solver and model checker for MPMTL, and a forward search algorithm.

7.1 A Parser for MPDDL

Currently, the parser supports only a portion of MPDDL (i.e., the features presented in Section 6, including math expressions with the operators '+', '-', '*', '\', functions 'min', 'max', 'pow', and MPMTL formulae defined in the domain). Not yet supported are universal and existential quantifiers, preferences, cost metrics, and indirect functions.

Preprocessing and Grounding The parser can read any MPMTL formula (e.g., in conditions) and reduce it to a DNF of ordered literals (To *et al.* 2016). If the resulting formula is not in D_2, then the parser will notify the user. After parsing all the domain and problem instance descriptions, it performs grounding and preprocessing to eliminate redundant variables and grounded actions. The parser uses the MPMTL SAT-solver to check the satisfiability of conditions and constraints in actions, discarding those whose condition or constraints are not satisfiable.

7.2 MPMTL SAT-solver and Model Checker

We implemented a SAT-solver and model checker for the D_2 fragment of MPMTL. Checking the satisfiability and satisfaction of a D_2 formula in a model can be done in polynomial time (To *et al.* 2016). This follows from the property that to check whether a point or line segment is above (or on, below) another line segment, it is sufficient to check at their endpoints and intersection (if it exists). The algorithms for these modules are extensive, and not presented here for the sake of brevity. The SAT-solver is used to check whether the goal and the initial constraint of the problem are satisfiable, while the model checker is used to verify the satisfaction of action conditions, constraints, and the goal in a world model.

7.3 Heuristic Forward Search in the Model Space

We focus on problems with temporal features such as second-order temporal constraints, maintenance goals, and quantitative goals. These problems are not supported by other current planners, and the effective techniques they use (e.g., reduction to non-temporal planning, STN, and LP) do not work for these problems. We employ time discretization for the search, with a pruning technique. The discretization step is computed dynamically each time based on the average action duration and the number of actions (function $step(M, A, t)$ Alg.2). Action durations are computed based on their range constraints. If the action duration is assigned by a value or expression, then it must satisfy all other inequalities in the duration constraint, or the action is not executable. If no assignment exists then MTP computes the range $[a, b]$ based on the duration constraint, and chooses the midpoint $(a + b)/2$ if $b < \infty$ or a otherwise.

Search Algorithm Each search node represents a timestamped world model $\langle (M, C), t_s \rangle$ as shown in Algorithm 2. Using the set O of actions that have not been applied in the model at the timestamp ensures that each action is applied in the model at that time at most once (lines 7, 11). More importantly, the actions whose conditions will be satisfied by

```
(:functions (distance ?b - base ?p - vip) (speed ?v - uav) (fly_rate ?v  - uav)  [...]
            (fly_dur ?v  - uav ?b - base ?p - vip) = (distance ?b ?p)/(speed ?v)
            (rate_dif ?v - uav) = (fly_rate ?v) - (hover_rate ?v) )
(:action relay  :parameters (?v - uav  ?b - base ?p - vip)
   :duration ( ?duration = 2*(fly_dur ?v ?b ?p) + ((fuel ?v)-2*(fly_dur ?v ?b ?p)*
                                     (fly_rate ?v)-(min_fuel ?v))/(hover_rate ?v)  &
               ?duration >= 3*(fly_dur ?v ?b ?p) )
   :condition ( <(at ?v ?b),[?ts,?ts+?duration]> &
                ([][?ts,?ts+?duration](available ?v))  &
                <(cleared ?b), [?ts,?ts+takeoff_dur]> &
                <(cleared ?b), [?ts+?duration-landing_dur,?ts+?duration]> )
   :constraint( <(fuel ?v) >= (min_fuel ?v), [?ts,?ts+?duration]>   &
                -(<>[?ts,?ts+?duration] <(hover_num) >= 2, [0, transit_dur]>) )
   :effect (and <-(at ?v  ?b), (?ts,?ts+?duration)> <-(available ?v),[?ts,?ts+?duration]>
                <(fuel ?v) -= ((fly_rate ?v),0), [?ts, ?ts+?duration)>
                <(fuel ?v) += ((rate_dif ?v),0), [?ts+(fly_dur ?v ?b ?p),
                                                  ?ts+?duration-(fly_dur ?v ?b ?p))>
                <-(cleared ?b), [?ts, ?ts+takeoff_dur]>
                <-(cleared ?b), [?ts+?duration-landing_dur, ?ts +?duration]>
                <(relayed ?p), [?ts+(fly_dur ?v ?b ?p), ?ts+?duration-(fly_dur ?v ?b ?p)]>
                <hover_num += 1, [?ts+(fly_dur ?v ?b ?p),  ?ts+?duration-(fly_dur ?v ?b ?p)]>
```

Figure 2: Description of the functions $distance$, fly_dur and $rate\_dif$ and of the complex durative action $relay$ in MPDDL.

```
(define (problem 1)   (domain relay2)
(:objects v0 v1 v2 - uav)
(:init (at v0 b0) <-(at v2 b0), [0,15]> <(at v2 b0), (15,inf)>
   <(cleared b0), [0,11)> <-(cleared b0), [11, 15]>
   <(cleared b0), (15,inf)> <(fuel v2) = (-6,100),[0,15]>
   <(fuel v2) = 10,(15, inf)>((fuel_used) = 0) ...)
(:constraint (hover_num) < 3 & (fuel_used) < 10000 )
(:goal (<>[0,15] <(relayed p0), [0,100]>)))
```

Figure 3: A partial description of a relay problem instance with 3 UAVs. The initial durative changes as a result of UAV $v3$ flying to $b0$ and landing at time 15 are expressed in the timelines: *(at v2 b0)*, *(cleared b0)*, and *(fuel v2)*. The landing duration is 4 as expressed in the timeline of *(cleared b0)*.

the effects of other simultaneous actions will eventually be applied in the model at that time, before advancing the model with a new timestamp (lines 12-14).

Pruning for Continuous Space of Model The pruning technique is applied in lines 6, 11, and 14. In line 14, the $pruned(M, C, t)$ function checks whether the time t has already passed the earliest deadline of an unsatisfied goal. If so, it discards the search node. Also, this function computes the left shift model $M{-}t$ of M and compares it with that of other generated models. If there exists a search node M', C', t' such that $M{-}t = M'{-}t'$, then (M, C, t) is discarded if its heuristic $heu(M, t)$ is not better than $heu(M', t')$, or the search node M', C', t' is invalidated and will not be explored later (line 6). $M{-}t = M'{-}t'$ means that the rest of the model M from t is the same as the rest of M' from t'. However, it is not necessarily that $M = M'$ as their prefixes can differ.

Algorithm 2 Search($\vec{x}, A, M_I, C_I, G$)

```
   Input: A temporal planning problem $\langle \vec{x}, A, M_I, C_I, G \rangle$
   Output: A plan, if it is found, or $NULL$, otherwise
 1: Create Priority Queue $Q$ and Set $O = A$ {Actions not
    yet applied in the model at the timestamp}
 2: Initialize $Q$ with $(M_I, C_I, 0, O)$
 3: while $Q$ is not empty do
 4:    Pop the first element $N = (M, C, t, O)$ from $Q$
 5:    if $N$ is invalidated then continue
 6:    if $M$ satisfies $G$ then Construct and Output a plan
 7:    for each action $o$ in $O$ do
 8:       Compute the instant action $a$ of $o$ at time $t$
 9:       Compute $S = update(a, M, C)$
10:       if $S = null$ or $pruned(S, t)$ then continue
11:       Let $(M', C') = S$ and Set $O = O \setminus \{o\}$
12:       if $O \neq \emptyset$ then Insert $(M', C', t, O)$ to $Q$ {Heuristic
          of $(M', t)$ as Priority in $Q$}
13:       else if not $(pruned(M', C', t + step(M', A, t)))$
14:       then Insert $(M', C', t + step(M', A, t), A)$ to $Q$
15:    end for
16: end while
17: return $NULL$
```

Heuristic for Interval and Quantitative Goals Interval and quantitative goals specify a process rather than an instant fact as considered in most other approaches. For example, a partial plan that maintains a longer relay should be considered better than those with a shorter relay, even though none of them yet achieves the complete goal. Similarly, in the second example scenario, preference is given to plans that evacuate more people (or cargo). The (temporal) relaxed graph plan heuristic does not seem to work well for problems with these types of goals. Hence, we introduce a new heuristic based on progress towards achieving the goal. Specifically, the func-

Problem	SMTPlan+	MTP	Problem	SMTPlan+	MTP
gen-2	0.09/ 1000	0.04/ 1000	gen-6	0.22/ 1000	0.05/ 1000
gen-4	0.13/ 1000	0.04/ 1000	gen-8	0.42/ 1000	0.06/ 1000
car-1-1	0.11/ 32	0.04/ 10.95	evac-1-1	0.04/ 2.2	0.04/ 2.2
car-5-1	0.12/ 32	0.04/ 4.9	evac-1-2	TO	0.15/ 6.6
car-1-2	TO	0.05/ 4.85	evac-2-2	TO	0.2/ 4.22
evac-2-3	TO	38/ 7.48	evac2-2-3	N/A	0.54/ 7.04

Table 1: Performance of MTP and SMTPlan+ on multiple benchmark problems, given as the total time t (seconds) and makespan m for a generated plan. The performance is expressed as: t/m. 'TO' denotes a time out (>30 minutes), and N/A indicates that SMTPlan+ does not support problem $evac2$-2-3. For the benchmark problems used, gen-n denotes a linear generator problem with n tanks, car-m-n indicates a Car problem with maximum accelerator m that requires at least n runs in a plan, $evac$-m-n denotes an Evacuation problem with two cities and m buses that requires at least n loads of passengers.

Problem	MTP		MTP-h		MTP-p	
	time/pl	exp/gen	time/pl	exp/gen	time/pl	exp/gen
rel-50	0.07/7	28/41	1/5	1.4k/2.1k	same	
rel-100	0.08/11	52/75	150/9	232k/360k	same	
rel-150	0.11/17	83/120	TO		same	
rel-250	0.13/29	143/208	TO		same	
rel-1000	0.57/115	586/851	TO		same	
rel2-50	0.55/5	58/82	1/5	104/138	0.85/5	70/96
rel2-100	0.77/8	71/99	TO		0.84/8	83/113
rel2-150	0.79/12	99/136	TO		1/12	111/150
evac-2-3	38/10	30.6k/42k	TO		TO	
evac2-2-3	0.54/6	242/338	TO		1.5/6	825/861

Table 2: Comparison of MTP with MTP-h and MTP-p. For each planner, 'time', 'pl', 'exp', and 'gen' denote the running time, plan length, and number of nodes explored and generated by the search. rel-n (resp. $rel2$-n) indicates the instance of the $relay$ (resp. $relay2$) domain with the a goal requiring the maintenance of the communications relay for n time units. 'k' means about 1000, e.g., 2.1k is about 2100.

tion $heu(M, t)$ computes a value h as an increasing function of the duration, in which an interval goal has been partially achieved in the model, and of the value for a quantitative goal achieved so far in the model. If two search nodes have the same value h then the search will favor the node with the smaller timestamp.

8 Empirical Study

We compared the performance of MTP, on a shared problem space (http://www.knexusresearch.com/software/), with SMTPlan+ (Cashmore *et al.* 2016), a state-of-the-art PDDL+ planner. We also evaluated the effectiveness of MTP's search heuristics and pruning techniques by comparing its performance to that of its ablations MTP-h (MTP minus heuristics) and MTP-p (MTP minus pruning). In each case, we measured performance (on a set of benchmark problems) as the total time (seconds) for plan generation and the makespan for the computed plan.

Our benchmark problems include the relay and evacuation examples introduced earlier, as well as the Car and Generator examples, on which SMTPlan+ seems to perform the best among state-of-the-art temporal planners (Bryce *et al.* 2015; Cashmore *et al.* 2016). For a better evaluation, we modified the Car benchmark problem by limiting the duration for each run, such that the problem may require multiple runs in order to reach the goal. For MTP's executability, we reduced (no non-linear effect) and translated the problem to one with a complex durative $move$ action in MPDDL. This allowed us to use MPDDL to define the optimality of the $move$ action as the one with the highest average speed. For the first evacuation domain ($evac$), we defined three actions: $board$, $unboard$, and $drive$; the second domain ($evac2$) extends this with the complex action $transport$, which combines the other three actions. We simplified and translated the domain $evac$ into PDDL+ for SMTPlan+ to execute, but not $evac2$ because PDDL+ does not support complex actions.

Table 1 displays our results for MTP and SMTPlan+ plans for our benchmark problems. MTP outperformed SMTPlan+ on both measures for each domain. While SMTPlan+ cannot solve any of problems whose plan requires at least two actions at different times (e.g., car-1-2, $evac$-1-2) MTP can solve those problems within a second. We also observed that, although the second evacuation domain $evac2$ contains more actions, MTP found a solution in much less time, and with a smaller makespan, than in the first evacuation domain for the same problem instance. This is a result of MTP's heuristic, which is based on the progression of goal achievement, where the $transport$ action may help guide the search and illustrates the benefit of complex actions.

Table 2 displays our results for MTP and its ablations on the relay and evacuation domains. For the relay scenario, we created two domains $relay$ and $relay2$, each of which has two durative actions: $relay$ and $refuel$. The $relay2$ domain is more challenging than $relay$, as it contains the second-order constraint for concurrent hovering UAVs, and has more strict temporal constraints (e.g., narrower time windows) for the UAVs to operate. Due to the complex nature of these benchmarks as shown earlier, we find it impossible to translate them into PDDL+ or PDDL2.1 without much simplification. In the results, as shown, MTP performs and scales on problem size well on these challenging benchmarks, e.g., it can generate a plan of many complex actions at different times within a second (e.g., 115 actions for rel-1000). Furthermore, MTP outperforms and scales better than its ablations; this is reflected through the time to find a plan and the number of nodes generated and explored during search with only one exception: MTP-p performs virtually the same as MTP (denoted by "same") on the first relay domain.

9 Summary and Future Work

We showed how to use MPMTL to represent and solve planning problems with rich temporal properties. We presented MPDDL, extended PDDL2.1 to support MPMTL, and used it in MTP, the new planner we developed. We implemented in MTP a parser for MPDDL, a SAT-solver, and a model checker for a polynomial fragment of MPMTL. We created domains and problems for the example scenarios and tested

MTP with these problems. Our results showed that: (1) MTP outperforms the most competitive temporal planner SMTPlan+ on the domains it performed best on, (2) MTP scaled well on problem size for challenging domains with rich temporal properties, and (3) the heuristic and pruning technique we used in MTP improved its performance and scalability.

In future work, we will improve MTP by using a better discretization strategy or a more effective search technique.

Acknowledgements

Thanks to OSD ASD (R&E) for sponsoring this research. Thanks also to the reviewers and Frank Jeremy for their valuable comments, which led to a greatly improved paper.

References

Alur, R., and Henzinger, T. 1993. Real-Time Logics: Complexity and Expressiveness. *Infor. and Comput.*, 104:35-77.

Bacchus, F., and Kabanza, F. 2000. Using temporal logic to express search control knowledge for planning. *Artificial Intelligence*, 116(1-2), 123-191.

Bacchus, F., and Kabanza, F. 1998. Planning for Temporally Extended Goals. *Annals of Math & Artif. Intel.*, 22, 5-27.

Baier, J. A., and McIlraith, S. A. 2006. Planning with First-Order Temporally Extended Goals Using Heuristic Search. In *AAAI*.

Bauer, A., and Haslum, P. 2010. LTL Goal Specifications Revisited. In *Proc. ECAI*.

Benton, J.; Coles, Amanda; and Coles, Andrew. 2012. Temporal Planning with Preferences and Time-Dependent Continuous Costs. In *ICAPS*.

Cashmore, M.; Fox, M.; Long, D. ; and Magazzeni D. 2016. A Compilation of the Full PDDL+ Language into SMT. In *ICAPS*.

Piacentini, C.; Fox, M.; and Long, D. 2015. Planning with Numeric Timed Initial Fluents. In *AAAI*.

Coles, Amanda, and Coles, Andrew. 2014. PDDL+ Planning with Events and Linear Processes. In *ICAPS*.

Coles, Amanda; Coles, Andrew ; Fox, M.; and Long, D. 2010. Forward-Chaining Partial-Order Planning. In *ICAPS*.

Coles, A.; Coles, A.; Fox, M.; and Long, D. 2009. Temporal Planning in Domains with Linear Processes. In *IJCAI*.

Coles, A.; Fox, M.; Long, D.; and Smith, A. J. 2008. Planning with Problems Requiring Temporal Coordination. In *AAAI*.

Cushing, W.; Kambhampati, S.; Mausam; and Weld, D. 2007. When is temporal planning really temporal planning? In *Proc. Int. Joint Conf. on AI (IJCAI)*, pages 1852-1859.

Bryce, D.; Gao, S.; Musliner, D.; and Goldman, R. 2015. SMT-based Nonlinear PDDL+ Planning. In *AAAI*.

Della Penna, G.; Intrigila, B.; Magazzeni, D. ; and Mercorio, F. 2009. UPMurphi: a Tool for Universal Planning on PDDL+ Problems. In *ICAPS*.

Dechter, R.; Meiri, I.; and Pearl, J. 1991. Temporal constraint networks. *Artificial Intelligence*, 49.

Do, M., and Kambhampati, S. 2003. Sapa: A Multi-objective Metric Temporal Planner. *J. Art. Intel. Res*.

Dvorak, F.; Bit-Monnot, A. ; Ingrand, F. ; and Ghallab, M. 2014. A flexible ANML actor and planner in robotics. In *ICAPS Planning and Robotics Workshop*.

Fox, M., and Long, D. 2006. Modelling Mixed Discrete-Continuous Domains for Planning. *J. Art. Intel. Res.*, 27 235-297.

Fox, M., and Long, D. 2004. An Investigation into the Expressive Power of PDDL2.1. In *Proceedings of the Sixteenth European Conference on Artificial Intelligence (ECAI)*.

Fox, M., and Long, D. 2003. An Extension to PDDL for Expressing Temporal Planning Domains. *J. Art. Intel. Res.*, 61-124.

Hoffmann, J., and Edelkamp, S. 2005. The deterministic part of IPC-4: An overview. *J. Art. Intel. Res.*, 24, 519-579.

Kabanza, F., and Thiebaux, S. 2015. Search Control in Planning for Temporally Extended Goals. *ICAPS*.

McDermott, D., and the AIPS-98 Planning Competition Committee. 1998. PDDL-the planning domain definition language. Tech. rep. at: www.cs.yale.edu/homes/dvm. M. Molineaux, M. Klenk, and D. W. Aha. Planning in Dynamic Environments: Extending HTNs with Nonlinear Continuous Effects. In *Proceedings of 24th AAAI Conf.*, 2010.

D. Nau, T.-C. Au, O. Ilghami, U. Kuter, J. W. Murdock, D. Wu,& F. Yaman. SHOP2: An HTN planning system. In *J. Art. Intel. Res.*, 20:379-404, 2003.

F. Patrizi, N. Lipoveztky, G. Giacomo, and H. Geffner. Computing Infinite Plans for LTL Goals Using a Classical Planner. *IJCAI*, 2011.

F. Patrizi, N. Lipoveztky, and H. Geffner. Fair LTL Synthesis for Non-Deterministic Systems using Strong Cyclic Planners. *IJCAI*, 2013.

J. Penberthy and D. Weld. Temporal planning with continuous change. In *Proceedings of AAAI Conf.*, 1994.

W. Piotrowski, M. Fox, D. Long, D. Magazzeni, F. Mercorio. Heuristic Planning for PDDL+ Domains. In *Proceedings of 25th International Joint Conference on Artificial Intelligence (IJCAI)*, 2016.

J. Rintanen. Models of Action Concurrency in Temporal Planning. In Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI), 2015.

J. Rintanen. Discretization of Temporal Models with Application to Planning with SMT. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI)*, 2015.

J. A. Shin and E. Davis. Processes and continuous change in a SAT-based planner. In Art. Intel. (AIJ), 166(I):194-253, 2005.

D. E. Smith. Commentary: The Case for Durative Actions: A Commentary on PDDL2.1 In *J. Art. Intel. Res.*, pp. 149-154, 2003.

D. Smith, J. Frank, and W. Cushing. The ANML Language. The *ICAPS-08 Workshop on Knowledge Engineering for Planning and Scheduling (KEPS)*, 2008.

K. Tierney, Amanda Coles, Andrew Coles, C. Kroer, A. M. Britt, and R. M. Jensen. Automated Planning for Liner Shipping Fleet Repositioning. In *ICAPS*, 2012.

To, S.T., Roberts, M., Apker, T., Johnson, B., & Aha, D.W. Mixed propositional metric temporal logic: A new formalism for temporal planning. In *Planning for Hybrid Systems: Papers from the AAAI Workshop (Technical Report WS-16-13)*. Phoenix, AZ: AAAI Press, 2016.

C. W. Barrett, R. Sebastiani, S. A. Seshia, and C. Tinelli. Satisfiability modulo theories. IOS Press. 825-885, 2009.

A. Cimatti, A. Micheli, and M. Roveri. Solving strong controllability of temporal problems with uncertainty using SMT. In *AAAI*, 2014.

R. Alur and T. Henzinger. Real-Time Logics: Complexity and Expressiveness. *Information and Computation*, 104:35-77, 1993.

R. Koymans. Specifying real-time properties with metric temporal logic. *Real-Time Systems*, 2(4):255-299, 1990.