Proceedings of the Twenty-Ninth International Conference on Automated Planning and Scheduling (ICAPS 2019)

Mixed Discrete Continuous Non-Linear Planning through Piecewise Linear Approximation

Elad Denenberg, Amanda Coles
Department of Informatics, King's College London
Bush House, 30 Aldwych,
London, United Kingdom WC2B 4BG
{elad.denenberg,amanda.coles}@kcl.ac.uk

Abstract

Reasoning with continuously changing numeric quantities is vital to applying planners in many real-world scenarios. Several planners capable of doing this have been developed recently. Scalability remains a challenge for such planners, especially those that reason with non-linear continuous change. In this paper, we present a novel approach to reasoning with non-linear domains. Bounding the problem using linear over and under-estimators, allows us to use scalable planners that handle linear change to find plans for non-linear domains. We compare the performance of our approach to existing planners on several domains and demonstrate that our planner can achieve state-of-the-art performance in non-linear planning.

Introduction

Planning is fundamental to intelligent autonomous systems: deciding what to do, and when to do it, in order to achieve desired goals. One of the key challenges to deploying automated planners in real-world applications is reasoning with complex and expressive models that capture the temporal and numeric constraints, both discrete and continuous, that arise in these scenarios. The last twenty years have seen great focus from the community in developing increasingly expressive planners to address these problems: beginning with those capable of reasoning about time, e.g. (Do and Kambhampati 2001; Gerevini, Saetti, and Serina 2006), discrete numeric change e.g. (Hoffmann 2003) and most recently the most expressive *hybrid* planners that combine this with reasoning about continuous numeric change over time.

Broadly speaking there are two approaches to planning with continuous effects. The first is to discretize time according to a user-selected quantum ϵ. Planners such as UPMurphi (Della Penna et al. 2009), DiNo (Piotrowski et al. 2016) and ENHSP (Scala et al. 2016) take this approach. They are flexible in terms of the types of non-linear numeric functions supported, but depend heavily on an ϵ value selection that permits both scalability and solving of problems. Since the planners consider applying actions every ϵ, small values lead to poor scalability; whereas large values exclude solutions leading to incompleteness.

The second approach reasons about continuous time: using propositional planning techniques to generate action sequences that satisfy propositional preconditions; and then using a sub-solver to schedule this action sequence to satisfy the temporal and numeric constraints (or to determine that this is impossible). These planners typically scale better, as they are not affected by the ϵ issue; however, most of them are restricted to reasoning only with constraints and continuous changes that are linear because they use linear programming sub-solvers. Such planners include the forward-search planners COLIN (Coles et al. 2012), POPF (Coles and Coles 2014) and OPTIC (Benton, Coles, and Coles 2012). Of the few planners that reason with continuous time and non-linear effects SMTPlan (Cashmore et al. 2016) and PluReal (Bryce 2016) can handle polynomial effects, their scalability is limited by the use of a SAT-compilation based approach which must add a happening at every turning point of each non-linear effect.

In this paper, we present a novel, scalable approach to planning with discrete and continuous non-linear numeric effects. Our approach builds on the planner OPTIC, which uses forward search combined with a Mixed Integer Programming (MIP) solver ensuring the temporal and numeric constraints of the problem are met. In its original form OPTIC is restricted to continuous linear change because MIPs can only represent linear functions. In our work we generalize the approach to non-linear change by generating piecewise linear upper and lower bound approximations of non-linear functions and use the MIP to find solutions that are correct according to the most pessimistic bound (e.g. a greater than constraint must be satisfied by the lower bound), iteratively refining our approximations if necessary.

Linearization facilitates non-linear planning without the need to discretize time, while still handling a large class of non-linear functions. Our results show that the resulting planner OPTIC++, outperforms state-of-the-art hybrid planners on a number of domains; and further, is able to handle a wider class of functions (e.g. logarithmic functions) than other planners that remain indifferent to a user-selected ϵ.

Background

Here we define the problem of planning with continuous numeric change, and discuss existing approaches.

Problem Definition

A temporal planning problem with continuous and discrete numeric effects is a tuple $\langle I, G, \mathbf{A}, \mathbf{P}, \mathbf{V} \rangle$, where $\mathbf{P}$ is a proposition set and $\mathbf{V}$ is a numeric variable set. I and G are respectively complete/partial value assignments to these propositions and numeric variables, representing the initial and goal state of the problem. $\mathbf{A}$ is a set of actions.

To define actions, we require the following definitions:

Definition 1 (Linear Numeric Condition). *A condition of the form* $v^i \{\geq, \leq, =\} w^1 v^1 + w^2 v^2 + \ldots c,\ v^i \in \mathbf{V}$, $w^i, c \in \mathbb{R}$

Definition 2 (Instantaneous Linear Change). *An equation of the form* $v^i \{\text{+=}, =, \text{-=}\} w^1 v^1 + w^2 v^2 + \ldots c,\ v^i \in \mathbf{V}$, $w^i, c \in \mathbb{R}$

Note this instantaneous change does not depend on time.

Definition 3 (PDDL Expressible Continuous Change). *Any continuous change that can be expressed in PDDL: any rational function of the form:* $\frac{\mathrm{d}v^i}{\mathrm{dt}} \{\text{+=}, =, \text{-=}\} \frac{P(t)}{Q(t)}$ *where* $P(t)$ *and* $Q(t)$ *are polynomials:* $P(t) = \sum_i w^i \prod_j \left(v^j\right)^{m^j}$ *,and* $v^j \in \mathbf{V}$,$w^i \in \mathbb{R}$, $m^j \in \mathbb{N}$

Note that any function that can be expressed as an ODE can be defined as a continuous effect in PDDL. This includes natural exponents and logarithms, and any rational function.

Definition 4 (Continuous Linear Change). *Change of the form:* $\frac{\mathrm{d}v^i}{\mathrm{dt}} \{\text{+=}, =, \text{-=}\} c, v^i \in \mathbf{V}, c \in \mathbb{R}$*, i.e. linear in time.*

Definition 5 (Initial Time Indifferent Continuous Change). *Let* $v^i \in \mathbf{V}$ *be a variable, and* $f\left(v^i, t\right)$ *a PDDL expressible change acting on* v^i*. f will be initial time indifferent if for any start time* t_0^i *and* t_0^j*:*

$$\frac{\mathrm{d}^n v^i\left(t_0^i + \tau\right)}{\mathrm{dt}^n} = \frac{\mathrm{d}^n v^i\left(t_0^j + \tau\right)}{\mathrm{dt}^n} \ \forall n \in \mathbb{N},\ \tau \in \mathbb{R}$$

This defines any change with a gradient that does not depend on the time at which the change has started, and is not altered by actions that started before this change.

Each action in $\mathbf{A}$ can now be defined by a tuple:

$$\langle d, pre_{\vdash}, \mathit{eff}_{\vdash}, pre_{\leftrightarrow}, \mathit{eff}_{\leftrightarrow}, pre_{\dashv}, \mathit{eff}_{\dashv} \rangle \tag{1}$$

where d is the action's duration defined by conjunction of linear numeric conditions constraining d. $pre_{\vdash}$ and $pre_{\dashv}$ are conjunctions of preconditions: facts and linear numeric conditions that must be true at the action's start and end. $pre_{\leftrightarrow}$ is the invariant condition: a conjunction of preconditions that must be true throughout the action's duration.

$\mathit{eff}_{\vdash}$ and $\mathit{eff}_{\dashv}$ are effects that occur at the action's start and end, such effects may add or delete a proposition $p \in \mathbf{P}$ (eff_{+}, eff_{-}) or update a numeric variable $v^i \in V$ according to a linear instantaneous change (eff_{num}). $\mathit{eff}_{\leftrightarrow}$ is a conjunction of continuous effects operating throughout the action's duration. In general, this effect can be any continuous PDDL expressible change; here we focus on planning with monotonic non-linear initial time indifferent change.

The solution is a *plan*: a timestamped action sequence, with all preconditions satisfied on execution, transforming I into to G.

Related Work

Here we explore more existing planners capable of reasoning with continuous change. dReal (Gao, Kong, and Clarke 2013) uses the discretizing method. It uses an ICP solver which discretizes intervals to find a solution; dReach (Bryce et al. 2015) uses SMT encoding prior to utilizing dReal.

Planners reasoning about time include the work of (Piacentini et al. 2018) which compiles the entire problem to a MIP, and Graphplan/SAT/SCOP compilation such as TMLPSAT (Wolfman and Weld 1999) and Kongming (Li and Williams 2011). All can reason with linear change only.

A few existing planners can handle non-linearity: SCIPPlan (Say and Sanner 2018) uses Spatial Branch-and-Bound to solve Mixed Integer Non-Linear Programming problems. The planner cqScotty (Fernández-González, Karpas, and Williams 2017) can handle non-linear constraints, but is restricted to linear effects. uNICOrn (Bajada, Fox, and Long 2015), as well as (Alaboud and Coles 2019) use a linearize-validate cycle to compile the non-linear problem, allowing a linear planner to be used. (Denenberg and Coles 2018) considered using OPTIC with a linearized model but that approach proved inefficient. uNICOrn relies on the accuracy level of the linearization, with a low accuracy level it may produce plans that are not valid, and therefore, does not carry an advantage over discretizing planners in this regard.

Forward Search Planners with MIP Schedulers

This work builds on OPTIC (an extension of COLIN). The relevant mechanism is described in (Coles et al. 2012). OPTIC performs forward search using WA* with COLIN's relaxed plan heuristic and no helpful actions. The search branches from the initial state in the space of instantaneous snap-actions (instantaneous actions representing starting and ending of durative actions).

Snap-action $\mathrm{A}_{\vdash}$ represents the durative action A's start and has precondition $pre_{\vdash}\mathrm{A} \bigcup pre_{\leftrightarrow}\mathrm{A}$ and effects $\mathit{eff}_{+\vdash}\mathrm{A}$, $\mathit{eff}_{-\vdash}\mathrm{A}$; $\mathit{eff}_{num\vdash}\mathrm{A}$; $\mathrm{A}_{\dashv}$ is the analogous action for A's end. Search proceeds from the initial state, at each state branching over applicable actions, exploring partially-ordered but un-time-stamped snap-action sequences.

A snap-action is applicable if its propositional preconditions are satisfied, and its effects do not delete the propositional invariants of any previous actions that have started but have not yet finished. Each action yields a new propositionally consistent *partial plan* and a new resulting state updated according to its effects. We refer to each snap-action in the partial plan as a *step*.

OPTIC adds *ordering constraints* as necessary to enforce the plan's propositional consistency. If $step_i$ deletes proposition p or requires p as a precondition, it is ordered after the last step that added p. If step$_i$ adds p, it is ordered after the last step that deleted p. For numeric conditions, any step that conditions on, or refers to, variable v^i in an effect is ordered after the most recent step that modified/referred to v^i, giving a total ordering on all steps modifying each variable.

While this forward search enforces propositional consistency of the plan, it is possible that the partial plan cannot be scheduled to satisfy the temporal and numeric constraints. To determine temporal and numerical consistency

step	variables	constraints
Generate$_{\vdash}$	t_0	≥ 0
	Gen_FL_0	$= 80$
	$Gen\_FL'_0$	≥ 0
Refuel$_{\vdash}$	t_1	$-t_0 \geq \epsilon$
	Gen_FL_1	$= Gen\_FL'_0 - 1(t_1 - t_0)$
		≥ 0
	$Gen\_FL'_1$	$= Gen\_FL_1$
		≤ 90
		≥ 0
Refuel$_{\dashv}$	t_2	$-t_1 \geq \epsilon$
		$-t_1 \leq 15$
	Gen_FL_2	$= Gen\_FL'_1 + 1(t_2 - t_1)$
		≤ 90
		≥ 0
	$Gen\_FL'_2$	$= Gen\_FL_2$
		≥ 0
now	t_{now}	$-t_2 \geq \epsilon$
	Gen_FL_{now}	$= Gen\_FL'_2 - 1(t_{now} - t_2)$
		≥ 0

Table 1: MIP Equations of a Partial Plan

OPTIC represents the partial plan's constraints as a set of MIP equations to which a feasible solution represents valid time-stamps for plan steps. If no solution exists, the partial plan cannot be scheduled and is pruned from the search space (no further attempt to extend it).

Running Example - The Linear Generator

We demonstrate the MIP building process with the generator domain. A generator is required to supply energy for 100 minutes. The planner can take two actions: first, generate energy. Generating consumes fuel with a change rate of $1^{\text{litre}/\text{min}}$. While generating, the fuel level must not drop below 0. The second action is to refuel from an auxiliary tank. The refuel rate is $2^{\text{litres}/\text{min}}$. While refuelling the auxiliary tank fuel level must not drop below 0, and the generator fuel level must not exceed the main tank capacity of 90 litres. The maximum duration of the refuel action is 15min.

Formulation of MIP

For every partial plan we formulate the temporal constraints then the continuous numeric constraints. Table 1 shows the MIP equations for the variable representing the main tank fuel level and the temporal constraints for the partial plan Generate$\vdash$,Refuel$\vdash$,Refuel$\dashv$ in the linear generator domain.

Temporal Constraints: We create a MIP variable t_i for each plan step i representing the application time of the action that step i represents. Two types of temporal constraints are possible: duration and ordering.

Duration constraints are conjunctions of linear constraints as per Definition 1. For any durative action A in the partial plan, with time-stamp variables t_i and t_j for its start $A_{\vdash}$ and end $A_{\dashv}$, we write each constraint in the conjunction:

$$t_j - t_i \{\geq, \leq, =\} w^1 v^1 + w^2 v^2 + \ldots c \tag{2}$$

To enforce the *ordering constraints* generated during search, stating $step_j$ comes after some other step $step_i$ we write:

$$t_j - t_i \geq \epsilon \tag{3}$$

where ϵ is a small constant.

Table 1 shows the ordering constraints for each step in our example (e.g. $t1 - t0 \geq \epsilon$). In addition, the duration constraint of refuel is transformed to $t_2 - t_1 \leq 15$.

Numeric Constraints: For each plan step i and for each variable $v \in \mathbf{V}$ we define three MIP variables: v_i and v'_i representing the variable's value before and after the step; and δv_i representing the coefficient of all the continuous linear change active on v (used to calculate the value of v_{i+1}).

Let A be a durative action with a continuous linear effect as per Definition 4, and let c_A be the constant defining said linear change. Then δv_i is calculated thus:

$$\delta v_i = \begin{cases} \delta v_{i-1} + c_A & \text{if} A_i = A_{\vdash} \\ \delta v_{i-1} - c_A & \text{if} A_i = A_{\dashv} \end{cases} \tag{4}$$

i.e. at step i which starts A, c_A will be added to δv_i and at step j which ends A, c_A will be removed from δv_j. Thus δv_i is the sum of all effects currently active on $v \in \mathbf{V}$ (if we begin indexing steps from i=0, $\delta_{v0} = c_A$ for $step_0$). All steps affecting or referring to v are totally ordered, so δv_i is always correctly known when required.

v'_i's value immediately after the application of an instantaneous snap action is calculated by summing up all instantaneous numerical effects eff^i_{num} acting at that step. Numeric effects are linear equations (Definition 2), so:

$$v'_i = v_i + w^1 v^1_i + w^2 v^2_i + \ldots c \tag{5}$$

The value of v prior the action's application is calculated according to the active continuous linear change:

$$v_i = v'_{i-1} + \delta v_{i-1} \left(t_i - t_{i-1}\right) \tag{6}$$

where t_{i-1} is the time-stamp of the previous step, and δv_{i-1} is the sum of all rates of the continuous linear effects that operated on v between the two time-stamps.

We formulate the numeric preconditions $pre_{\vdash}$, $pre_{\dashv}$ (Definition 1) of the action at $step_i$ over the respective variables v_i. We formulate invariant conditions $pre_{\leftrightarrow}$ after the action's start step, i (over v'_i), before the action's end step and at every step between them. This is sound because all effects are linear: turning points are only possible at steps that alter δv, so a condition satisfied at all steps is satisfied throughout.

Table 1 shows the the numeric constraints formulation for the generator fuel level in our example. The value of δGen_FL_i is calculated for each step: when the generator is working it consumes fuel at a rate of 1 ($\delta Gen\_FL_0 = -1$), refuelling increases fuel at a rate of 2 ($\delta Gen\_FL_1 = -1 + 2 = 1$). Equations calculate the value of Gen_FL and Gen_FL' before and after each step. Finally, constraints at appropriate steps ensure that the fuel level does not go below 0 or rise above 90.

So far all constraints have been Linear Programming (LP) constraints; however, this approach has been extended by making use of MIP to reason with preferences that impose soft constraints on the plan trajectory by adding big M constraints (Benton, Coles, and Coles 2012).

A feasible solution to this MIP is a valid time-stamp for each of the actions, that respects all the temporal and numeric constraints. A valid schedule for a propositionally sound plan that satisfies the goal is a solution to the planning problem. The search and MIP work together to achieve this: search proposing propositionally sound plans and the MIP solver determining whether there is a valid schedule.

Partial Plan Optimization

We can optimize a partial plan according to a given metric F, or if none is specified minimize plan makespan. To calculate the makespan or the variable values for the objective OPTIC defines an additional temporal variable t_{now} which is ordered after all steps and calculates the value of the variables at that time point. The last three rows in Table 1 demonstrate this. The MIP objective is defined over the values of v_{now}; or set to minimize t_{now} for makespan.

Planning with Non-Linear Continuous Effects

Now we extend OPTIC to reason with non-linear continuous change; using piecewise linear estimators defined thus:

Definition 6 (Piecewise Linear Function). *A piecewise linear function is a continuous function of the form:*

$$f(t) = \begin{cases} a_0 t + b_0 & 0 < t \leq \tau_1 \\ a_1 t + b_1 & \tau_1 < t \leq \tau_2 \\ \vdots & \\ a_{n-1} t + b_{n-1} & \tau_{n-1} < t < \tau_n \end{cases} \quad (7)$$

where a_i, b_i and $\tau_i \in \mathbb{R}$ are constants defining the linear pieces, n is the number of pieces and $\forall i \quad a_i \tau_i + b_i = a_{i+1} \tau_i + b_{i+1}$.

Definition 7 (Over (Under) estimators). *A function $f(t)$ is said to be an over (under) estimator of function $g(t)$, if $\forall t \in \mathbb{R} \quad f(t) \geq (\leq)\, g(t)$*

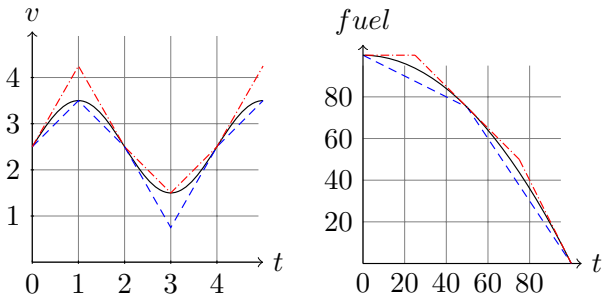
Using over and under-estimators we can say two things about a solution to the partial plan MIP equations:

Case 1: If a feasible MIP solution is found, with the over estimator adhering to all less-than constraints and the under estimator adhering to all greater-than constraints, that plan is valid in the non-linear case.

Case 2: If no feasible solution can be found with the over estimator satisfying the greater-than constraints and under estimator satisfying less-than constraints, the partial plan is not valid in the non-linear case and can be pruned.

When neither of the statements is true, the estimators need to be refined, until we determine solution existence or inexistence. We explain the refinement process in a later section.

Figure 1a demonstrates this logic. Assume that the black curve is a description of the sum of all effects on v. The red (dashed-dotted) lines are the piecewise linear over-approximations, the blue (dashed) under-estimators. If the constraints are $v > 0$ and $v < 5$ between time 0 and 5 then clearly the plan which affected the variable v in this manner is valid, since the over estimator is always less than 5, and the function is always less than the over estimator. Similarly since the under estimator is always greater than 0 so is v. If



(a) The Linearization logic (b) Fuel Loss During Generation

Figure 1: Linearization

the constraint was $v > 5$, by examining the over estimator one can conclude that the plan is not valid. However, if the constraints were $v > 1$ and $v < 3.6$ then the $>$ constraint fails on the lower estimator, and the $<$ constraint on the upper estimator; they are however satisfied by the upper and lower estimator respectively, so we know we need to refine the approximation in order to find out whether this might be a valid plan.

Linearization Process

The use of piecewise linear functions has a long history in the field of control. A classic reference is (Bemporad, Ferrari-trecate, and Morari 2000), which uses piecewise affine linear functions to prove the observability and controllability of a hybrid problem. Our approach differs as we implement the above logic to solve two instances of the linearized problem while not relying on a constant discretization of the function, rather, we define in this section an automatic linearization process with a reducing segment size.

The planner described in this work is capable of reasoning with PDDL expressible, initial time indifferent, monotonic continuous change. Since it inherits the PDDL definition of change, it requires the effects to be differentiable. One can calculate the value of an effect at any time t by integrating the given derivative (assuming that the effect in question alone is acting on the variable).

We generate piecewise linear upper and lower bounding functions (Definitions 6 and 7) for all continuous non-linear effects, these are of the form :

$$v_k = a_k t + b_k \; \forall k \quad (8)$$

where $k = 0 \ldots N-1$ represents the kth linear segment v_k.

The framework of OPTIC++ is depicted roughly in Fig 2. The initial linearization is done in the action grounding phase. We assume input domains in which all continuous effects are defined by monotonic, initial time indifferent change only. We exploit the monotonicity in the initial linearization by not searching for turning points in the duration of the action. Determining whether extrema exist, and finding them, is undecidable; numerical methods for root finding exist, however they add exponential complexity. We plan in future to extend our work to some classes of non-monotonic functions, where root finding might be feasible, but for now

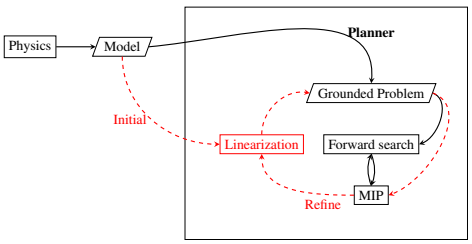


Figure 2: OPTIC++ Framework

we focus on the monotonic case. Initial time indifference allows us to linearize and refine effects before the formulation of the MIP, if the linearization depended on the relative timestamps assigned to actions we would have to present the MIP with several possible linerizations (determined at each state) for each effect. Again we leave this to future work.

Linearization Process - Concave: We define a linear segment on the concave side of the effect by sampling the values of at two different time-points τ_{k+1} and τ_k (τ_k being the kth sampled time-point) thus:

$$a_k = \frac{v(\tau_{k+1}) - v(\tau_k)}{\tau_{k+1} - \tau_k} \tag{9}$$

$$b_k = v(\tau_k) - \frac{v(\tau_{k+1}) - v(\tau_k)}{\tau_{k+1} - \tau_k}\tau_k \tag{10}$$

Each segment is defined on the interval $[\tau_k, \tau_{k+1}]$

Initially, we sample two time-points: time 0, and at the maximal duration of the action initiating the effect. If a maximum duration is not defined for the action, then we use a large time-horizon. The first approximation on the concave side is therefore a straight line between the effect's start and end. To refine the estimator the value at an additional point is sampled (we explain how this is selected later). The segment in which the sampled time-point falls is replaced by two new segments, calculated as above.

Linearization Process - Convex: We define a convex linear segment by the function's value and the derivative's value at a selected time-point τ_k. The piece is a tangent at the sampled point and so a_k is equal to the derivative and b_k is:

$$b_k = v_k - a_k\tau_k \tag{11}$$

The interval defining each segment is found by its intersection with the estimator's previous and next segments:

$$\tau_k = \frac{b_{k-1} - b_k}{a_k - a_{k-1}} \tag{12}$$

The initial convex approximation has two segments calculated from the start and end points. A segment is added to the estimator by sampling a point, the new segment constants are calculated as described, the time on which the segment is defined is found by the intersection with existing segment.

Fig 1b demonstrates the linearization: the effect shown is hyperbolic and taken from the generate action in the non-linear generator domain. The linear under-approximation is calculated using the values at the effect start, end and an additional sampled point that was added during the search.

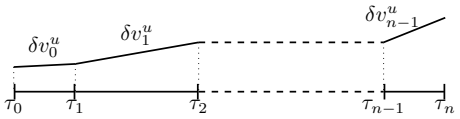


Figure 3: An Approximation of Non-linear Effect

The linear over-approximation is calculated using these values and the derivatives at these three time-points.

Selecting a Point for Refinement: If Case 1 of the over (under) estimator adhering to the less(greater)-than constraints **does not** have a solution; but Case 2 of requiring the over (under) estimator to adhere to the greater(less)-than constraints **does** find a solution then we cannot prove or disprove whether a solution exists so a refinement takes place. Since we cannot know (without expensive computations) which condition/effect caused constraint violation we refine both estimators for the effects of all actions that appeard in the plan for which the refinement was required.

The solution to the Case 2 problem is in the form of a time-stamped partial plan, i.e., a sequence of start and end snap-actions, and the times at which they occur. The first candidate point considered for refinement is the time value assigned to the end of the action that started the effect.

If the candidate point is close to any of the points already sampled (this can happen, when the duration constraint is equality) or no end action has yet been applied in the partial plan, we select the point in the middle of the largest estimator section. This prevents the approximations converging in one area and impeding finding a feasible solution.

Completeness: Branches are pruned from the search only if they are proved inconsistent. For a continuous effect, by using Case2. This is a worst-case scenario: The value of the variable will always be less than an overestimator; therefore, if an overestimator violates a greater-than constraint the effect is bound to violate the same constraint.

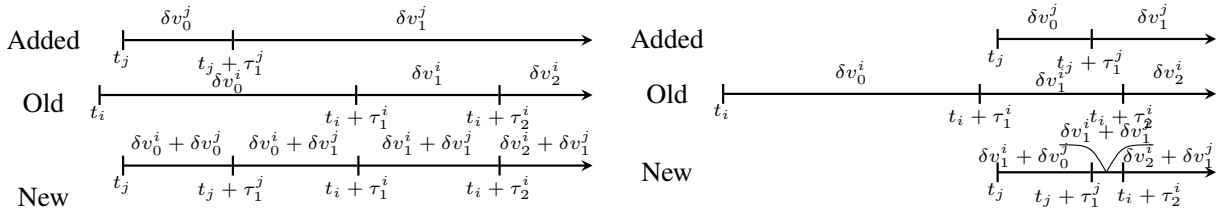
When refining during the search, in order not to linger too long at a given state, a limit of L refinements is defined. If more refinements are required, then the state is put on a second open list, to be used when all other branches have been explored. When the state is revisited, it can be refined L more times before being put in the second list again. Therefore refining does not affect completeness.

Encoding Piecewise Linear Effects in the MIP

We now detail new MIP variables describing the estimator, and their computation in MIP construction at each state.

MIP Representation of a Single Non-linear Effect

To accommodate a non-linear effect starting at step i, the first modification we make is to replace v_i at each step with upper and lower estimations of v: v_i^u, v_i^l. The same is done for v_i' and δv_i. Instantaneous effects operate on both estimators: if a constant value is assigned, added or subtracted, then the effect is applied to both estimators. If the value of a different variable is used, we use the upper or lower bound on that variable as appropriate in order to maximize (minimize)



(a) A Possible Option for Combining Two effects

(b) Another Possible Option for Combining Two effects

Figure 4: Combining an Additional Effect to an Existing Single Effect

the upper (lower) bound on v. For instance when a variable u is subtracted then v_i^u would be:

$$v_i'^u = v_i^u - u_i^l \tag{13}$$

The next modification ensures the correct value of v_i^u (v_i^l) is computed, by applying the appropriate continuous change to $v_{i-1}'^u$ ($v_{i-1}'^l$). In the linear case we defined δv_i as the sum of all derivatives acting at step i and then wrote $v_i = v'_{i-1} + \delta v_{i-1}(t_i - t_{i-1})$. However, we now have a piecewise linear representation with a sequence of different derivatives, acting on the variable at different time-points. To demonstrate why a single value δv_i no longer suffices, consider a single linearized effect, illustrated in Fig 3. The value of v at t_j depends on which segment of the piecewise linear function t_j occurs in.

The linearization can be thought of as a set of pairs $\boldsymbol{\delta v}^u_{\mathbf{eff}} = ((\delta v_0^u, \tau_0) \ldots, (\delta v_n^u, \tau_n))$ each stating that at a constant time τ_k after the effect begins the gradient is approximated by δv_k^u. If a future $step_j$ falls in segment m ($\tau_m \leq t_j - t_i \leq \tau_{m+1}$) v_j can be computed by adding the cumulative effects of the previous segments $prev(m)$ (a known constant for a given segment) to the effect according to the distance between τ_m and t_j, $since(m)$:

$$prev(m) = \sum_{k=1..m} \delta v_{k-1}^u (\tau_k - \tau_{k-1}) \tag{14a}$$

$$since(m) = \delta v_m^u (t_j - (t_i + \tau_m)) \tag{14b}$$

Now we have:

$$v_j = v'_i + prev(m) + since(m) iff \tau_m \leq t_j - t_i \leq \tau_{m+1} \tag{15}$$

In a single effect, the change rates δv_j^u are the estimator's slopes (a_ks) of Eq. (8), and the time points correspond to the time points defining the estimator τ_k.

To enable the MIP to choose which effect segment a future step (t_j) will occur in an effect that started at step$_i$, we use Big-M constraints. For each segment $m \in [0, n)$ in which t_j could occur we write the following constraints:

$$(t_j - t_i) + MB_{j,m} \geq \tau_m \tag{16a}$$

$$(t_j - t_i) - MB_{j,m} \leq \tau_{m+1} \tag{16b}$$

$$v_j^u \geq v_i'^u + prev(m) + since(m) - MB_{j,m} \tag{16c}$$

$$v_j^u \leq v_i'^u + prev(m) + since(m) + MB_{j,m} \tag{16d}$$

where M is a large constant, and $B_{j,m}$ is a binary variable for step j and segment m. We additionally write:

$$\sum_{k \in [0,n)} B_{j,k} = n - 1 \tag{17}$$

These constraints force one variable $B_{j,m}$ to be set to zero; hence, the underlying temporal constraints in equations 16a and 16b must be satisfied for that m (i.e. t_j occurs in segment m). Equations 16c and 16d ensure that v_j^u is calculated according to the equation for segment m.

If step j starts an action A with a linear effect at t_j, while a non-linear effect with n sections is executing, the change rate δv_A is added to all sections of $\boldsymbol{\delta \mathbf{v}_{\mathbf{eff}}}$ that appear after t_j. However, since we do not know in which section t_j is, we generate n vectors $\boldsymbol{\delta \mathbf{v}_{\mathbf{eff}}}$, selecting from them using Eq.(16)

Combining Several Non-Linear Effects

Assume the first non-linear effect acting on v^u, having a set of change rates $\boldsymbol{\delta \mathbf{v}_i}^u = \boldsymbol{\delta \mathbf{v}_{\mathbf{eff}}}$, has started at step i. In addition, assume that at step $j > i$ an additional non-linear effect has started. For the calculation of v_{j+1} at $step_{j+1}$, after j, we need to sum all active effects. Therefore we need to combine both of the linearized effects to properly represent the way in which they are acting on v^u. Combining those effects will generate several vectors each representing the total change in way a similar to that for a single effect.

Fig 4a presents merging two effects. The added effect (top) is weaved with the existing effect (middle) creating a new effect (bottom). If the relative time at which j started after step i is known, then we can describe a new vector $\boldsymbol{\delta \mathbf{v}_\mathbf{j}}^u$.

Let δv_i^k be a change rate of section k that is acting on the variable v by an effect that started at step i. Also, let δv_j^m be a change rate of section m of the effect that started at step j. If the sections overlap in time, the result is a section (numbered n) in the new vector $\boldsymbol{\delta \mathbf{v}_\mathbf{j}}$, that has the rate:

$$\delta v_j^n = \delta v_i^k + \delta v_j^m \tag{18}$$

In the single effect vector, each section's start and end points were defined as constants offsets from step i that started the effect. When effects of multiple steps are defined, we need to define the active gradient with respect to the appropriate time step. For example, in Figure 4a, the first section's end in the new vector is defined with respect to t_j as its time-point $t_j + \tau_1^j$ depends on the time at which j started; whereas the second segment $t_i + \tau_1^i$ occurs a constant time after step i started. Hence, the combined effect vector records not just τ_m but also t_n the step index that the effect happens τ_m after. Thus the new effect $\boldsymbol{\delta \mathbf{v}_\mathbf{j}}^u$ is $\boldsymbol{\delta \mathbf{v}_\mathbf{j}}^u = ((\delta v_0^u, (t_0, \tau_0)) \ldots, (\delta v_n^u, (t_n, \tau_m)))$ where t_n is the nth step in the vector, which may relate to any previous step in the partial plan, and τ_m the corresponding constant offset.

There are many ways to create $\boldsymbol{\delta \mathbf{v}_j}$, depending on the scheduled time of t_j. The difference between Fig 4a and 4b shows this. The number of possible $\boldsymbol{\delta \mathbf{v}_j}$s created depends on the number and size of segments in the current and added effects. In the worst case if $\boldsymbol{\delta \mathbf{v}_i}$ has n segments and the added effect $\boldsymbol{\delta \mathbf{v}_{eff}}$ m, then there are $(m+n)!$ possible $\boldsymbol{\delta \mathbf{v}_j}$ vectors. We reduce that: first, thanks to total ordering we consider only interleavings that have j starting after i. Second, we have information about the section sizes, and check and exclude invalid permutations. Third, as in the linear case, here too ending actions removes their effect, and thus reduces the number of possible vectors. While this may still leave us with many vectors, we inevitably have to solve this combinatorial part of the problem, and we show in our evaluation that the planner's scalability is reasonable.

In the general case, step i might already have multiple effects operating on variable v and therefore multiple $\boldsymbol{\delta \mathbf{v}_i}^m$s. If step $j > i$ adds another non-linear effect on v the process above is used to create a new $\boldsymbol{\delta \mathbf{v}_j}$ for each existing $\boldsymbol{\delta \mathbf{v}_i}$.

To describe the problem in the MIP, all feasible $\boldsymbol{\delta \mathbf{v}}_j^m$s are created, where m is the mth possible vector. In addition, Big-M constraints are written corresponding to each vector:

$$(t_j - t_i) + MP_{j,m} \geq \pi_1 \tag{19a}$$

$$(t_j - t_i) - MP_{j,m} \leq \pi_2 \tag{19b}$$

where π_i defines the interval in which the t_j needs to occur for the corresponding $\boldsymbol{\delta \mathbf{v}}_j^m$ to be active, and $P_{j,m}$ are binary variables, as the $B_{j,m}$s of Eq. (16).

At step$_{j+1}$, we write Eq. (15) according to each $\boldsymbol{\delta \mathbf{v}}_j^m$. t_i in Eq. (16a)-(16b) will be replaced with the MIP variables representing the correct section start. To ensure the we calculate the value of v_{j+1} according to the right vector, $P_{j+1,m}$ will be added to Eq. (17) as follows:

$$-P_{j+1,m} + \sum_{i=0..n} B_i = n - 1 \tag{20}$$

Though the input effects are monotonic, a function composed of two or more monotonic effects is not guaranteed to remain monotonic. Therefore, $\boldsymbol{\delta \mathbf{v}_i}$ vector sections are checked for sign changes, indicating a possible minimum or maximum. The values of v^u and v^l are calculated at such points, and the invariant constraints are checked at these points in the MIP.

Evaluation

We implemented our approach in OPTIC++ and tested in four domains[1]. Our evaluation uses the following constants: time horizon (10^6), refinement limit (L=20) and big M (10^5). We have found these constants sufficient in all the domains we have explored. With a time horizon lower than the duration of the longest action OPTIC++ could not find a solution. Too small an L would degrade performance, we seldom required more than 5. Too small an M would not force the equations, and too large an M would cause floating point errors. For OPTIC++ and SMTPlan we set ϵ = 0.001, changing this value does not affect their performance. All tests were performed on an Intel i7-6700T CPU@2.80GHz×8 with 15GB RAM. The results are presented in Table 2.

We compare to three state-of-the-art hybrid planners SMTPlan, DiNo and ENHSP. We chose these planners because they take input in standard PDDL, and have been shown to outperform other planners, e.g. dReach (Piotrowski et al. 2016); which are more difficult perform a fair comparison with due to the need to translate PDDL to drm.

The comparison with ENHSP must be interpreted cautiously: ENHSP does not support durative actions, whereas the other planners do not support global constraints, due the the different parsers they use. This difference requires that the domains be rewritten for ENHSP. Therefore, though the physics was the same, the models on which the planners were compared are not exactly equivalent.

Linear Generator: This is the running example. Its aim here is to demonstrate the overhead of using upper and lower estimators on a linear domain (in practice, OPTIC++ could trivially detect the case where all effects are linear and revert to OPTIC). The main tank capacity is 1000 litres, all auxiliary tank capacities are 10 litres.

As expected, OPTIC++ was slower. However, the difference becomes less significant as the problem becomes bigger, this implies that the overhead is in the linearization phase, rather than search itself being slower in OPTIC++.

Non-linear Generator: Similar to the linear version but for the refuelling effect. The fuel level added to the main generator's main tank (and reduced from an auxiliary) is:

$$\frac{\mathrm{d}F_{gen}\,(t)}{\mathrm{dt}} = 0.4t \tag{21}$$

The generation fuel consumption function remains linear. In this domain, there is just one non-linear effect, which interacts with a linear effect. Each tank capacity here is 100 litres. This domain differs from the non-linear variant from (Cashmore et al. 2016) in two ways: first, the duration of the refuel action is not governed by a duration constraint, rather, by the requirement for the tank's fuel level to not fall below zero. Second, only one tank at a time can refuel the generator. This forces a different happening for each refuel action.

DiNo and SMTPlan employ symmetry breaking methods: DiNo breaks symmetry in the model checker, and SMTPlan in the SMT solver. Since OPTIC++ does not employ symmetry breaking it was implemented in the model. We made two versions of this domain: one with symmetry breaking (SB) (requiring the tanks to be used in order - one before two and so on), and one without. The headline comparison between the planners should be OPTIC++ on the domain which required defined order (termed SB) compared with DiNo and SMTPlan on the domains which do not. The results are shown in Table 2. The results in the table refer to DiNo running with a time step $\epsilon = 0.1$. Using $\epsilon = 1$ resulted in DiNo failing to find a solution because the discretization isn't fine enough to admit a solution, $\epsilon = 0.01$ timed out on all instances. ENHSP timed out when using $\epsilon = 0.1$ and smaller. The results shown in the table are for $\epsilon = 1$. The plans ENHSP found with this larger ϵ are valid within said epsilon, i.e. the generator could be ran for 1001 seconds -

[1]The domains can be found in https://github.com/eladden/ICAPS2019Benchmark

Linear Generator									
tanks	10	20	30	40	50	60	70	80	90
OPTIC	0.67	3.70	12.04	99.97	390.65	63.65	117.61	190.52	224.67
OPTIC++	1.87	10.19	30.80	152.00	420.27	63.77	119.11	192.85	235.50

Non Linear Generator										
tanks	1	2	3	4	5	6	7	8	9	10
OPTIC++ SB	0.12	0.80	2.38	10.19	8.27	32.57	51.10	143.55	492.65	827.92
OPTIC++	0.13	2.52	48.91	-	-	-	-	-	-	-
SMTPlan SB	0.11	-	-	-	-	-	-	-	-	-
SMTPlan	0.11	0.12	0.14	0.20	0.31	-	-	-	-	-
DiNo SB 0.1	-	-	-	-	-	-	-	-	-	-
DiNo 0.1	6.78	741.95	-	-	-	-	-	-	-	-
ENHSP SB 1	5.27	7.93	9.15	11.33	-	-	-	-	-	-
ENSHP 1	4.56	7.64	9.09	11.17	-	-	-	-	-	-

3D Printer					
secondary cart.	1	2	3	4	5
OPTIC++ SB	0.31	239.64	-	-	-
OPTIC++	0.25	-	-	-	-
SMTPlan SB	0.01	-	-	-	-
SMTPlan	0.02	0.08	0.48	1.84	23.54
DiNo SB 1	-	-	-	-	-
DiNo 1	-	-	-	-	-
ENHSP SB 1	-	-	-	-	-
ENHSP 1	-	-	-	-	-

Powered Landing																
height	100	200	300	400	500	600	700	800	900	1000	1100	1200	1300	1400	1500	1600
OPTIC++	4.54	2.59	3.30	1.37	1.49	1.51	1.53	1.37	1.71	1.33	1.63	1.50	1.59	1.34	1.77	1.05
DiNo 0.1	8.51	22.97	41.78	61.72	86.92	113.62	141.65	170.47	199.69	230.84	263.62	299.05	340.17	380.33	382.16	-
ENHSP 0.1	0.47	1.16	4.88	-	-	-	-	-	-	-	-	-	-	-	-	

Table 2: Runtime (s): "-" indicates planner failure to report a solution in (1000s); Numbers after planner names indicate the ϵ value used: we use (in their favor) the highest value that permits a valid solution.

but this would not be deemed valid by the other planners. All planners produced the same solution plans (modulo ϵ).

When SMTPlan found a solution it did so faster than other planners. Because it represents effects as polynomials and solves an SMT problem, polynomial change is defined directly in the equations, thus it fits this problem best. SMTPlan's disadvantage is with many happenings: as the number of actions grow, SMTPlan scales badly. OPTIC++ scales better to plans that require larger number of happenings. Indeed in our evaluation SMTPlan was unable to solve any problem (in any domain) requiring more than 6 happenings.

3D Printer: This domain describes the working of a 3D printer. The "ink" of the 3D printer is PLA plastic. The printer draws PLA from a main cartridge of radius $R_m = 0.1\text{m}$, it has additional secondary cartridges from which PLA can be drawn, each with its own radius $Rc = 0.04\text{m}$. The cartridge is built with an internal spring assuring that the radius remains constant. The engines of this printer supply all axes with a constant torque (T_m and T_c). Remember that $T = I\alpha$, I is the moment of inertia and α is the angular acceleration. We assume here that the moment of inertia remains constant through the duration of the printing.

The length of the PLA string that is used by either the print or the feed action is therefore:

$$L = L_0 + v_0 t + \frac{1}{2} R\alpha t^2 \tag{22}$$

The printer can print, and draw PLA from the main cartridge, and it can, simultaneously feed additional PLA to the main cartridge from a secondary. In our domain the engines begin from rest, therefore all $\omega_0 = 0$. The torques and moments of inertia were such that $\alpha_m = -0.1^{\text{rad}}/\text{s}^2$ in printing, and each secondary cartridge acceleration was $\alpha_c = 10^{\text{rad}}/\text{s}^2$. In this domain the planner is required to work with two interacting non-linear (polynomial) effects.

DiNo timed out on all instances of the problem with $\epsilon = 1$, OPTIC++ was able to plan with 2 cartridges before timing out, SMTPlan was able to plan with 5, ENHSP was unable to find any solution, moreover, it incorrectly deemed the problem unsolvable. Here SMTPlan outperforms OPTIC++ as it is avoiding the combinatorial blow-up OPTIC++ encounters in the overlapping non-linear effects by exploiting the fact that the effects are polynomial (as it only supports polynomials it can rely on additional assumptions); as OPTIC++ is more general, and doesn't assume effects are polynomial it has to deal with the combinatorial overlapping of effects. Since solutions to the larger problems here have small numbers of happenings SMTPlan is not affected by the issues we saw in Non-Linear Generator. This is the one case in our evaluation where we found another planner to be superior to OPTIC++: specifically if all effects are known to be polynomial, these effects overlap, and there are a small number of happenings in the solution plan, SMTPlan is the better option. Again, all planners generated identical plans.

Powered Landing: This domain was used in (Piotrowski et al. 2016) to demonstrate DiNo's ability to solve plans with different required makespans. It serves here to show the great advantage OPTIC++ gains by reasoning with continuous time rather than discretizing, as well as to demonstrate OPTIC++ solving a non-polynomial domain: something SMTPlan cannot do. The domain models a 1D rocket powered landing. A rocket is free falling to the surface of a planet from an initial height d_0 and a given velocity v_0. The goal is to reach a marginal area of no more than d_f above the ground with a velocity no larger than v_f.

The rocket may fire to slow its descent. The thrust velocity change is given by the Tsiolkovsky equation (Turner 2009).

$$\Delta v = I_{sp} g \ln\left(\frac{m_0}{m\,(t)}\right) \tag{23}$$

where I_{sp} is the specific impulse, g the gravitational constant, m_0 the fuel mass prior to the engine firing, and $m\,(t)$ the current mass, or the mass after t seconds of thrust. We assume that the mass flow is constant and equal to q, thus:

$$m\,(t) = m_0 - qt \tag{24}$$

The effect of falling is of course (Newton 1934):

$$d = d_0 + v_0 t - \frac{1}{2} g t^2 \tag{25}$$

The constants used were: $g = 9.8^{\text{m}}/\text{s}^2$, $I_{sp} = 311\text{s}$, $q = 50^{\text{kg}}/\text{s}$, $m_0 = 50,000\text{kg}$, $v_0 = 0$, $d_f = 10\text{m}$, and $v_f =$

$10^{\mathrm{m}}/_{\mathrm{s}}$. The initial distance d_0 varied from 100 to 20,000m in the problem instances. This domain differs from that in (Piotrowski et al. 2016): the duration of the fall action was originally defined by a non-linear condition on the mass. Here the maximal duration of the falling action is simply a large constant as this mass requirement was already subsumed by the preconditions and OPTIC++ doesn't support non-linear conditions. The crash event is redundant and was not used.

Note that the optimal plan requires the same number of actions as the fall duration increases: the rocket simply needs to thrust for a longer duration. ENHSP and DiNo do not scale well: degrading significantly as fall duration increases; due to time discretization, they have more time-points to reason about as the duration increases. Smaller epsilon values make this worse: neither planner solves any problem with $\epsilon = 0.01$ ($\epsilon = 1$ is not fine enough to admit valid solutions). As OPTIC++ reasons with continuous time, neither the duration of the plan, nor the value of ϵ affect its performance.

DiNo and ENHSP's found plans required the firing of the engine several times, leading to a longer makespan plan, whereas OPTIC++ found the point after which it fired the engine once until the required limit is reached. OPTIC++'s solution is the optimal one. This shows that OPTIC++'s inbuilt makespan optimisation, even in the absence of a metric function, allows production of better quality plans.

Conclusions

In this paper, we presented OPTIC++, a MIP planner capable of reasoning with monotonic, initial time indifferent, non-linear change. Our results show OPTIC++ outperforms state-of-the-art planners on a range of benchmark domains. In future work we plan to implement root-finding algorithms to allow reasoning with non-monotonic change, and to implement linearization in the MIP writing phase to allow reasoning with arbitrary non-linear effects, specifically those that depend on actions that happen before the effect starts. Additional research can be done by extending the linearization logic to non-linear preconditions and duration constraints.

Acknowledgements

This work was supported by the UK Engineering and Physical Sciences Research Council Grant EP/P008410/1 (AI Planning with Continuous Non-Linear Change).

References

Alaboud, F. K. M., and Coles, A. 2019. Personalized medication and activity planning in PDDL+. In *ICAPS*.

Bajada, J.; Fox, M.; and Long, D. 2015. Temporal planning with semantic attachment of non-linear monotonic continuous behaviours. In *IJCAI*.

Bemporad, A.; Ferrari-trecate, G.; and Morari, M. 2000. Observability and controllability of piecewise affine and hybrid systems. *IEEE Transactions on Automatic Control* 45:1864–1876.

Benton, J.; Coles, A. J.; and Coles, A. 2012. Temporal planning with preferences and time-dependent continuous costs. In *ICAPS*.

Bryce, D.; Gao, S.; Musliner, D. J.; and Goldman, R. P. 2015. Smt-Based nonlinear PDDL+ planning. In *AAAI*.

Bryce, D. 2016. A happening-based encoding for nonlinear PDDL+ planning. In *AAAI Workshop: Planning for Hybrid Systems*.

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

Coles, A. J., and Coles, A. I. 2014. PDDL+ planning with events and linear processes. In *ICAPS*.

Coles, A. J.; Coles, A. I.; Fox, M.; and Long, D. 2012. COLIN: Planning with continuous linear numeric change. *Journal of Artificial Intelligence Research* 44:1–96.

Della Penna, G.; Intrigila, B.; Magazzeni, D.; and Mercorio, F. 2009. UPMurphi: a tool for universal planning on PDDL+ problems. In *ICAPS*.

Denenberg, E., and Coles, A. 2018. Modelling sequences of processes in pddl+ for efficient problem solving. In *Workshop on Knowledge Engineering for Planning and Scheduling*. ICAPS.

Do, M. B., and Kambhampati, S. 2001. SAPA: a domain-independent heuristic metric temporal planner. In *European Conf. on Planning (ECP)*.

Fernández-González, E.; Karpas, E.; and Williams, B. C. 2017. Mixed discrete-continuous planning with convex optimization. In *AAAI*.

Gao, S.; Kong, S.; and Clarke, E. M. 2013. dReal: An SMT solver for nonlinear theories over the reals. In *International Conference on Automated Deduction*.

Gerevini, A.; Saetti, A.; and Serina, I. 2006. An approach to temporal planning and scheduling in domains with predictable exogenous events. *Journal of Artificial Intelligence Research* 25:187–231.

Hoffmann, J. 2003. The Metric-FF planning system: Translating "ignoring delete lists" to numeric state variables. *Journal of Artificial Intilligence Research* 20:291–341.

Li, H., and Williams, B. 2011. Hybrid planning with temporally extended goals for sustainable ocean observing. In *AAAI*.

Newton, I. 1934. Principia mathematica.

Piacentini, C.; Castro, M. P.; Ciré, A. A.; and Beck, J. C. 2018. Compiling optimal numeric planning to mixed integer linear programming. In *ICAPS*.

Piotrowski, W.; Fox, M.; Long, D.; Magazzeni, D.; and Mercorio, F. 2016. Heuristic planning for PDDL+ domains. In *IJCAI*.

Say, B., and Sanner, S. 2018. Metric nonlinear hybrid planning with constraint generation. In *ICAPS*.

Scala, E.; Haslum, P.; Thiébaux, S.; and Ramirez, M. 2016. Interval-based relaxation for general numeric planning. In *ECAI*.

Turner, J. L. M. 2009. *History and principles of rocket propulsion"*. Berlin, Heidelberg: Springer Berlin Heidelberg. 1–35.

Wolfman, S., and Weld, D. 1999. The LPSAT System and its Application to Resource Planning. In *IJCAI*.