Metric Hybrid Factored Planning in Nonlinear Domains with Constraint Generation

Buser Say and Scott Sanner

Department of Mechanical & Industrial Engineering, University of Toronto, Canada
{bsay,ssanner}@mie.utoronto.ca

Abstract. We introduce a novel planner SCIPPlan for metric hybrid factored planning in nonlinear domains with general metric objectives, transcendental functions such as exponentials, and instantaneous continuous actions. Our key contribution is to leverage the spatial branch-and-bound solver of SCIP inside a nonlinear constraint generation framework where we iteratively check relaxed plans for temporal feasibility using a domain simulator, and repair the source of the infeasibility through a novel nonlinear constraint generation methodology. We experimentally evaluate SCIPPlan on a variety of domains, showing it is competitive with, or outperforms, ENHSP in terms of run time and makespan and handles general metric objectives. SCIPPlan is also competitive with a general metric-optimizing unconstrained Tensorflow-based planner (TF-Plan) in nonlinear domains with exponential transition functions and metric objectives. Overall, this work demonstrates the potential of combining nonlinear optimizers with constraint generation for planning in expressive metric nonlinear hybrid domains.

Keywords: Metric hybrid planning · Nonlinear optimization · Constraint generation.

1 Introduction

Metric optimization is at the core of many real-world nonlinear hybrid [6] planning domains where the *quality* of the plan matters. Most nonlinear hybrid planners in the literature either ignore metric specifications [12, 3, 4], or leverage heuristics to guide their search for finding a plan quickly [9, 13] with the notable exceptions COLIN [5] and ENHSP [17], which can handle metric optimization for a subset of PDDL+ [6] domains.

In this paper, we leverage the nonlinear constrained optimization solver SCIP [10] to present SCIPPlan for solving metric hybrid factored [2] nonlinear planning problems by decomposing the original problem into a master problem and a subproblem. In the master problem, we relax the original problem to a system of sequential function updates [1], which allows us to handle arbitrary nonlinear functions (such as polynomial, exponential, logarithmic etc.) in

[1] Relaxation refers to the omission of temporal constraints from the master problem.

the transition and metric objectives as well as instantaneous continuous action inputs that are beyond the expressivity of existing hybrid planners.

In the nonlinear hybrid planning literature, the time at which a conditional expression (e.g., a mode switch condition) is satisfied is known as a *zero-crossing* and when the dynamics of the planning problem are piecewise linear, one can use the TM-LPSAT compilation to find valid plans that avoid zero-crossings [18]. When the continuous change can be described more generally as polynomials, one can use the SMTPlan [4] compilation of the hybrid planning problem to avoid zero-crossings between two consecutive decision points (i.e. happenings). However, in general, problem dynamics can include arbitrary nonlinear change and only ENHSP [17] approaches the expressive dynamics of SCIPPlan.

In SCIPPlan, the candidate solution found by solving the master problem can include zero-crossings of general transcendental nonlinear conditions between two consecutive decisions which can either (i) violate the global constraints of the original problem, or (ii) contain mode switches that are not accounted for by the master problem. To identify and repair the source of zero-crossings, we use the simulate-and-validate approach [7] in the subproblem where domain simulators are used to simulate the candidate plan, and if the candidate plan is found to be infeasible, temporal constraints associated with zero-crossings are generated and added back to the master problem. SCIPPlan iteratively solves the master problem and the subproblem until a valid plan is found.

Experimentally, we show that SCIPPlan outperforms the state-of-the-art metric nonlinear hybrid planner ENHSP in almost all problem instances with respect to makespan and run time performance. We further experiment with the capabilities of SCIPPlan beyond the expressiveness limitations of ENHSP in the optimization of general metrics on a subset of modified domains, and verify its competitiveness versus an unconstrained Tensorflow-based planner (TF-Plan) [19] on a nonlinear metric domain with exponential transitions.

2 Preliminaries

In this section, we present the preliminary definitions, notation and solution methodologies required to define and solve the metric hybrid factored planning problem.

3 Metric Discrete Time Factored Planning: $\boldsymbol{\Pi}$

Before we dive into the notationally heavy details of general nonlinear hybrid factored planning, we begin with a straightforward mixed-integer nonlinear program (MINLP) compilation of a *discrete time* factored nonlinear planning domain. A discrete time metric factored planning problem is a tuple $\Pi = \langle S, A, C, T, I, G, Q \rangle$ where

- $S = \{S^d, S^c\}$ is a set of discrete S^d and continuous S^c domains with state variables/assignments denoted $\boldsymbol{s} \in S$,

- $A = \{A^d, A^c\}$ is a set of discrete A^d and continuous A^c domains with action variables/assignments denoted $\boldsymbol{a} \in A$,
- $C : S \times A \rightarrow \{true, false\}$ is a function that returns true if action $\boldsymbol{a} \in A$ and state $\boldsymbol{s} \in S$ pair satisfies constraints that represent global constraints on state and action variables,
- $T : S \times A \rightarrow S$ denotes the state transition function between discrete time steps t and $t+1$, $T(\boldsymbol{s}^t, \boldsymbol{a}^t) = \boldsymbol{s}^{t+1}$ if $C(\boldsymbol{s}^t, \boldsymbol{a}^t) = true$, and is undefined otherwise, and
- $Q : S \times A \rightarrow \mathbb{R}$ is the metric reward function to optimize.

In addition, I represents the initial state constraint $\boldsymbol{s}^1 = \bar{\boldsymbol{s}}^1$ and $G : S \rightarrow \{true, false\}$ represents goal state constraints. Given a finite planning horizon of H decision stages, a solution $\pi = \langle \bar{\boldsymbol{a}}^1, \ldots, \bar{\boldsymbol{a}}^H \rangle$ (i.e. plan) to Π is a fixed value assignment to actions $\boldsymbol{a}^t = \bar{\boldsymbol{a}}^t$ that induces an assignment to state variables $\boldsymbol{s}^t$ satisfying the initial state I, transition T, goal G, and global C constraints for all $t \in \{1, \ldots, H\}$. Our objective in solving metric planning problem Π is to find the action sequence π that maximizes the sum of rewards over the time horizon by optimizing the following model:

$$\begin{aligned}
&\max_{\pi = \langle \boldsymbol{a}^1, \ldots, \boldsymbol{a}^H \rangle} \sum_{t=1}^{H} Q(\boldsymbol{s}^{t+1}, \boldsymbol{a}^t) && (1)\\
&\text{subject to } I : \boldsymbol{s}^1 = \bar{\boldsymbol{s}}^1 \\
&\qquad G(\boldsymbol{s}^{H+1}) \\
&\qquad T(\boldsymbol{s}^t, \boldsymbol{a}^t) = \boldsymbol{s}^{t+1} \quad \forall t \in \{1, \ldots, H\} \\
&\qquad C(\boldsymbol{s}^t, \boldsymbol{a}^t) \quad \forall t \in \{1, \ldots, H\}
\end{aligned}$$

Note that Π is a standard discrete-time model that does not consider the (potentially) changing values of states between pairs of consecutive time steps $t, t+1 \in \{1, \ldots, H\}$. Under this simplifying assumption, there is no need to consider zero-crossing constraints that will become critical for relaxing time to be continuous in our subsequent hybrid generalization of the above framework. Before we present the hybrid generalization, however, we discuss the compilation and solution of the above problem followed by an example.

3.1 High-level Syntax and SCIP MINLP Compilation

In order to compile the optimization formulation of (1) into a Mixed-Integer Nonlinear Programming (MINLP) formulation that can be solved via an off-the-shelf MINLP solver (e.g., SCIP [10]), we need (i) a high-level syntax such as the RDDL language [15] for specifying all constraints and functions and (ii) a compilation that can translate any formula in this syntax into the MINLP language. For example, piecewise functions induced by if-then-else constructs require use of the big-M trick to encode conditional constraints, while boolean

Table 1. (left column) Grammar to recursively generate expression syntax of the RDDL language [15] extending [14] to nonlinear expressions in the last four rows. E_1 and E_2 belong to the same language as E and are acyclic. (middle column) Conditions on grammar rule application. (right column) MINLP compilation of the grammar rule: every RDDL expression E is represented by an MINLP variable v_E that evaluates to the value of that expression ($\{0,1\}$ if boolean). M is a large constant.

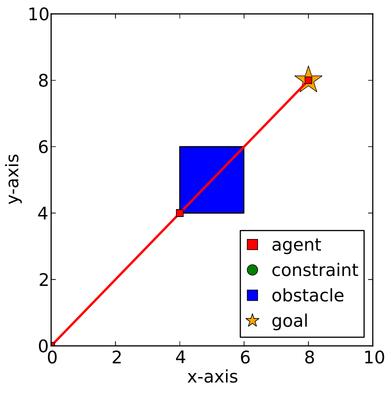
Expression	**Condition**	**Constraints**
$E \rightarrow k$	k is a constant	$v_E = k$
$E_b \rightarrow \top$ (or $\bot$)		$v_{E^b} = 1$ (or 0)
$E_b \rightarrow p$	p is state or action variable	$v_E = v_p$
$E \rightarrow \wedge_{i=1}^{n} E_b^i \equiv \forall_i E_b^i$	E_b^i is a boolean expression	$nv_E \leq \sum_{i=1}^{n} v_{E_b^i} \leq n - 1 + v_E$
$E \rightarrow \vee_{i=1}^{n} E_b^i \equiv \exists_i E_b^i$	E_b^i is a boolean expression	$v_E \leq \sum_{i=1}^{n} v_{E_b^i} \leq nv_E$
$E \rightarrow \neg E_b$	E_b is a boolean expression	$v_E = 1 - E_b$, $v_E, v_{E_b^i} \in \{0,1\}$
$E \rightarrow kE_1$	k is a constant	$v_E = kv_{E_1}$
$E \rightarrow E_1 \; op \; E_2$	$op \in \{+,-\}$	$v_E = v_{E_1} \; op \; v_{E_2}$
$E_b \rightarrow E_1 \geq E_2$	E_b is a boolean expression	$Mv_{E_b} - M \leq v_{E_1} - v_{E_2}$ $\leq Mv_{E_b}, v_{E_b} \in \{0,1\}$
$E \rightarrow$ if E_b then E_1 else E_2	E_b is a boolean expression	$v_{E_1} + Mv_{E_b} - M \leq v_E$ $\leq M + v_{E_1} - Mv_{E_b}$, $v_{E_2} - Mv_{E_b} \leq v_E$ $\leq v_{E_2} + Mv_{E_b}, v_{E_b} \in \{0,1\}$
$E \rightarrow E_1 \; op \; E_2$	$op \in \{\times, \div\}$	$v_E = v_{E_1} \; op \; v_{E_2}$
$E \rightarrow \exp(E_1)$		$v_E = e^{v_{E_1}}$
$E \rightarrow \log(E_1)$	E_1 is a positive expression	$v_E = log(v_{E_1})$
$E \rightarrow \text{abs}(E_1)$		$v_E = \lvert v_{E_1} \rvert$

expressions in constraints and if-then-else conditions require special encodings as arithmetic expressions over integers.

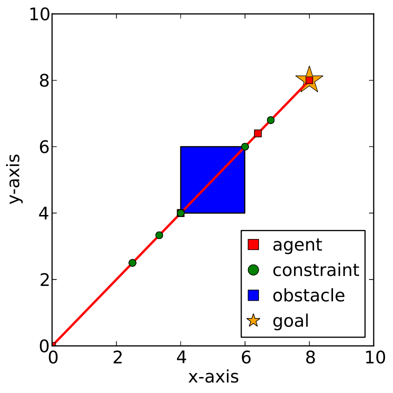
In Table 1, we provide a grammar for the (nonlinear) expression syntax of the ground RDDL language and a compilation of each grammar rule to the SCIP MINLP format assuming each sub-expression has been recursively compiled.

3.2 Spatial Branch-and-Bound

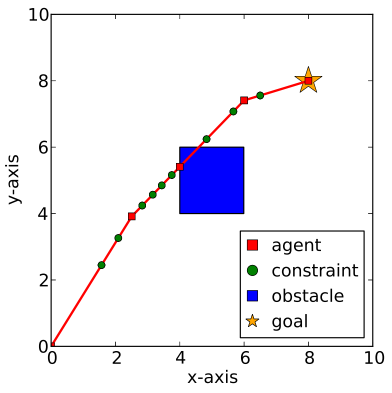
To solve the compiled MINLP, SCIP uses Spatial Branch-and-Bound (SBB) [11] – an algorithm based on the *divide-and-conquer* strategy for solving MINLPs in the form of min $f(\mathbf{x})$ subject to $\mathbf{g}(\mathbf{x}) \leq 0$ where function $f(x)$ and function vector $\mathbf{g}(\mathbf{x})$ contain nonlinear expressions, and the decision variable vector $\mathbf{x}$ can have continuous and/or discrete domains. The SBB algorithm uses tree search where branching decisions are made on candidate solutions $\bar{\mathbf{x}}$, and the optimal value of the objective function $f(\bar{x})$ is bounded at each search node until a preset optimality gap is reached.



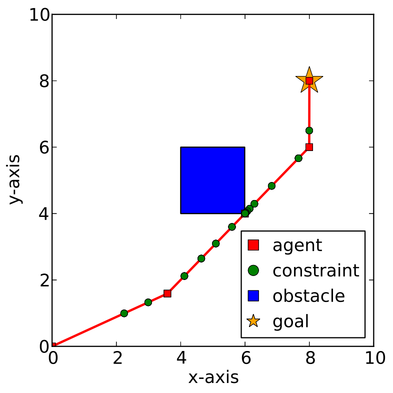
(a) First Iteration



(b) Iteration 6



(c) Iteration 9



(d) Final Iteration 16

Fig. 1. Visualization of iterative plan generation of SCIPPlan for the example *hybrid* navigation domain. In the first six iterations, the plan steps π (in red) generated to reach the goal (in orange) pass through the obstacle (in blue), violating zero-crossing constraint c_3 that is detected during plan simulation. At each iteration, additional zero-crossing constraints are generated *symbolically* at the midpoints of each violation interval (in green) to eliminate these zero-crossings from the solution space of the master problem. By iteration 9, SCIPPlan starts to converge to a valid plan and by iteration 16, SCIPPlan returns a valid plan. Note that sometimes there are overlaps of the position of the agent between time steps (i.e., the agent does not move).

3.3 Illustrative Example

To illustrate how the MINLP compilation and solution works for the metric factored planning problem, we consider the following simple navigation domain with (i) three continuous action variables $(a_x, a_y, \Delta) \in A^c$ that move the agent a_x and a_y in respective x and y directions for duration Δ, (ii) two continuous state variables $(s_x, s_y) \in S^c$ representing agent location, (iii) and three constraints in

C:

$$\begin{aligned}
&c_1: 0 \leq s_x^t + a_x^t \Delta^t, s_y^t + a_y^t \Delta^t \leq 10 \quad \forall t \in \{1, \dots, H\},\\
&c_2: -1 \leq a_x^t, a_y^t \leq 1 \quad \forall t \in \{1, \dots, H\}, \text{and}\\
&c_3: 4 \geq s_x^t + a_x^t \Delta^t \vee 6 \leq s_x^t + a_x^t \Delta^t\\
&\quad \vee 4 \geq s_y^t + a_y^t \Delta^t \vee 6 \leq s_y^t + a_y^t \Delta^t \quad \forall t \in \{1, \dots, H\}.
\end{aligned}$$

Here, constraints c_1, c_2 denote bounds on the domains of state variables s_x and s_y, and constraint c_3 represents an obstacle located in the middle of the maze. Initial and goal constraints are compiled as follows for $H = 4$:

$$I: s_x^1, s_y^1 = 0, \quad G: s_x^{H+1}, s_y^{H+1} = 8\ ,$$

Given the transition function

$$T: s_x^{t+1} = s_x^t + a_x^t \Delta^t, \quad s_y^{t+1} = s_y^t + a_y^t \Delta^t \quad \forall t \in \{1, \dots, H\}$$

and reward metric $Q(\boldsymbol{s}^{t+1}, \boldsymbol{a}^t) = -\Delta^t$ (minimize total time, a.k.a. makespan), the SSB solver can return a plan π as visualized by Figure 1 (a). The plan π passes through the obstacle since the discrete time formalization only checks constraints at the start and end points of each decision stage; we remedy this with a hybrid extension in the next section.

4 Metric Hybrid Factored Planning: $\boldsymbol{\Pi^\delta}$

In this section, we define the metric hybrid factored planning problem Π^δ by building on the notation, definitions and the solution methodology presented for the metric factored planning problem Π. But before we define Π^δ, first we need to distinguish one continuous action variable as the control duration $\Delta \in A^c$ to specify the duration of time step $t \in \{1, \dots, H\}$ such that $0 \leq \Delta$ [2]. Similarly, we update the notation we use for the global constraint function $C(\boldsymbol{s}^t, \boldsymbol{a}^t, \Delta^t)$ and the state transition function $T(\boldsymbol{s}^t, \boldsymbol{a}^t, \Delta^t)$ to explicitly specify the duration Δ^t of time step $t \in \{1, \dots, H\}$. Finally, we need to distinguish the set of boolean expressions that appear in if-else conditions of the state transition function T as transition modes $\mathtt{M}$, that is,

$$\begin{aligned}
T(\boldsymbol{s}^t, \boldsymbol{a}^t, \Delta^t) = &\text{ if } E_b^1(\boldsymbol{s}^t, \boldsymbol{a}^t, \Delta^t) \text{ then } E_1(\boldsymbol{s}^t, \boldsymbol{a}^t, \Delta^t)\\
&\dots\\
&\text{elif } E_b^n(\boldsymbol{s}^t, \boldsymbol{a}^t, \Delta^t) \text{ then } E_n(\boldsymbol{s}^t, \boldsymbol{a}^t, \Delta^t)\\
&\text{else } E_{n+1}(\boldsymbol{s}^t, \boldsymbol{a}^t, \Delta^t)
\end{aligned}$$

[2] In this work, we focus on hybrid planning problems where duration Δ is completely controlled by the planner. When there are exogenous events or processes that can change the total duration of a time step, we need to define a continuous state variable $\Delta' \in S^c$ as a function of $\boldsymbol{s}, \boldsymbol{a}, \Delta$ such that $f(\boldsymbol{s}, \boldsymbol{a}, \Delta) = \Delta'$ and transfer zero-crossing definitions onto Δ'. In this work, we assume $\Delta = \Delta'$ and omit Δ' for notational simplicity.

where $E_b^1(\boldsymbol{s}^t, \boldsymbol{a}^t, \Delta^t), \dots, E_b^n(\boldsymbol{s}^t, \boldsymbol{a}^t, \Delta^t) \in \mathtt{M}$. We denote $\mathtt{M}^\delta : S \times A \times \Delta \rightarrow P(\mathtt{M})$ as a function that returns the set of transition modes evaluating to true for given values of state $\bar{\boldsymbol{s}}^t$ and action $\bar{\boldsymbol{a}}^t$ variables and duration $\bar{\Delta}^t$ for $t \in \{1,\dots,H\}$ where notation $P(S)$ denotes the power set of S.

Definition 1. *(Zero-Crossing Certificate): Given the values of state $\bar{\boldsymbol{s}}^t$ and action $\bar{\boldsymbol{a}}^t$ variables and duration $0 < \bar{\Delta}^t$ for $t \in \{1, \dots, H\}$, we say $x^t \in (0, \bar{\Delta}^t)$ is a zero-crossing certificate for time step t if at least one of the following holds:*

1. *Global Constraint Violation: $C(\bar{\boldsymbol{s}}^t, \bar{\boldsymbol{a}}^t, x^t) = false$,*
2. *Mode [3] Switch: $\mathtt{M}^\delta(\bar{\boldsymbol{s}}^t, \bar{\boldsymbol{a}}^t, x^t) \neq \mathtt{M}^\delta(\bar{\boldsymbol{s}}^t, \bar{\boldsymbol{a}}^t, \bar{\Delta}^t)$.*

Given the definition of the zero-crossing certificate, the metric hybrid factored planning problem is a tuple $\Pi^\delta = \langle S, A, C, C^\delta, T, I, G, Q \rangle$ where $C^\delta : S \times A \times \Delta \rightarrow \{true, false\}$ is a function defined as $C^\delta(\boldsymbol{s}^t, \boldsymbol{a}^t, \Delta^t) = true$ if and only if there does not exist $x^t \in (0, \Delta^t)$ that is a zero-crossing certificate. Given a planning horizon H, a plan $\pi^\delta = \langle \bar{\boldsymbol{a}}^1, \bar{\Delta}^1 \dots, \bar{\boldsymbol{a}}^H, \bar{\Delta}^H \rangle$ to Π^δ is a plan π to Π where $C^\delta(\bar{\boldsymbol{s}}^t, \bar{\boldsymbol{a}}^t, x^t) = true$ for all $x^t \in (0, \bar{\Delta}^t)$ and $t \in \{1,\dots,H\}$. Note that the definition Π^δ extends deterministic RDDL [15] to continuous time and allows instantaneous continuous actions $A^c \subseteq A$ that are not functions of time. Unlike the PDDL+ [6] formalism, we do not assume that the effects of instantaneous actions are realized ϵ time after their execution.

5 Solving Π^δ with Constraint Generation

In this section, we introduce our novel SCIP-based planner (SCIPPlan) to plan in metric hybrid planning problems with nonlinear dynamics. But before we outline SCIPPlan, we first need to define the zero-crossing interval.

Definition 2. *(Zero-Crossing Interval): Given the values of state $\bar{\boldsymbol{s}}^t$ and action $\bar{\boldsymbol{a}}^t$ variables, and the duration $0 < \bar{\Delta}^t$ of time step $t \in \{1, \dots, H\}$, a zero-crossing is an interval $|_L x_1^t, x_2^t|_R$ where $|_L \in \{[, (\}$ and $|_R \in \{],)\}$ if and only if:*

1. *Non-empty: $0 < x_1^t \leq x_2^t < \bar{\Delta}^t$ such that if $x_1^t = x_2^t$ then $|_L x_1^t, x_2^t|_R$ is not an open interval (x_1^t, x_2^t), and*
2. *Uninterrupted and Contiguous: $\forall x \in |_L x_1^t, x_2^t|_R$ where x is a zero-crossing certificate.*

The novelty of SCIPPlan is that it decomposes Π^δ into a master problem $\mathcal{M}(\Pi, H)$ and a subproblem $\mathcal{S}(\pi, \epsilon)$, where $\mathcal{M}(\Pi, H)$ solves the metric factored planning problem Π for a given horizon H using a SBB solver, and $\mathcal{S}(\pi, \epsilon)$ checks whether π is also a plan for Π^δ using a domain simulator with respect to a time discretization parameter ϵ. If π is not a plan for Π^δ, $\mathcal{S}(\pi, \epsilon)$ returns the first zero-crossing interval $|_L x_1^t, x_2^t|_R$ with minimum time step $t \in \{1, \dots, H\}$, and a temporal constraint is added back to the master problem $\mathcal{M}(\Pi, H)$ to update either function C or T, depending on whether the zero-crossing is due to a global constraint violation or a mode switch, respectively.

[3] The concept of a mode is analogous to its counterpart in the field of Hybrid Automata [8].

5.1 Master Problem

The master problem $\mathcal{M}(\mathit{\Pi}, H)$ solves the metric factored planning problem $\mathit{\Pi}$ for a given horizon H, using the compilation presented for $\mathit{\Pi}$ in the MINLP formulation of (1) assisted by complex expression compilation of $\mathit{\Pi}$ to MINLP form provided in Table 1. We optimize the MINLP in (1) using the SCIP SBB solver [10].

5.2 Subproblem

Given a plan π for $\mathit{\Pi}$ and a discretization parameter ϵ, the subproblem $\mathcal{S}(\pi, \epsilon)$ uses a domain simulator to check for a zero-crossing certificate by simulating the state transition T sequentially $\lfloor \frac{\bar{\mathit{\Delta}}^t}{\epsilon} \rfloor$ times for all time steps $t \in \{1, \dots, H\}$ such that $T(\bar{\boldsymbol{s}}^t, \bar{\boldsymbol{a}}^t, \epsilon) \dots T(\bar{\boldsymbol{s}}^t, \bar{\boldsymbol{a}}^t, \epsilon \lfloor \frac{\bar{\mathit{\Delta}}^t}{\epsilon} \rfloor)$. If a zero-crossing certificate is found, $\mathcal{S}(\pi, \epsilon)$ returns (i) the first zero-crossing interval $|_L x_1^t, x_2^t|_R$ with minimum time step $t \in \{1, \dots, H\}$ such that there does not exist another zero-crossing certificate $x^t \ ? \ x_1^t$ found by $\mathcal{S}(\pi, \epsilon)$ where the relational operator ? is $<$ (i.e., greater) if $|_L = [$ (i.e., minimum bound of the interval is closed) and $\leq$ (i.e., greater or equal to) otherwise, and (ii) the set of compilation constraints $\boldsymbol{g}^t$ that cause the zero-crossing interval $|_L x_1^t, x_2^t|_R$.

Precisely, the zero-crossings due to (i) global constraint violation can be mapped to a set of compilation constraints representing the global constraint function C (as presented in the Illustrative Example Revisited section). Zero-crossings due to (ii) mode switch can be mapped to a set of boolean expressions E_b^t and to their respective compilation boolean decision variables $v_{E_b}^t$ — these evaluate to different values within zero-crossing interval $|_L x_1^t, x_2^t|_R$ compared to the end of control duration $\bar{\mathit{\Delta}}^t$ at time step $t \in \{1, \dots, H\}$.

5.3 Temporal Constraint Generation

Given the interval $|_L x_1^t, x_2^t|_R$ identified by the domain simulator for a time step $t \in \{1, \dots, H\}$ and the respective set of constraints $\boldsymbol{g}^t$, SCIPPlan generates a nonlinear constraint

$$\boldsymbol{g}^t(k\mathit{\Delta}^t) \leq 0, \quad k = \frac{x_2^t + x_1^t}{2\bar{\mathit{\Delta}}^t}, \tag{2}$$

where Constraint 2 symbolically substitutes all $\mathit{\Delta}^t$ with $k\mathit{\Delta}^t$. Note that $k \in [0, 1]$ is a constant coefficient representing the ratio of the mid-point of the zero-crossing interval $|_L x_1^t, x_2^t|_R$ to the complete duration at time step t. There are four benefits of our constraint generation methodology: (i) We generate a symbolic [4] constraint ensuring the zero-crossing violation of the current plan is enforced, while generalizing as a valid constraint for all other plans. (ii) Instantiation of zero-crossing constraints at the violation midpoint is intended to induce a

[4] Symbolic refers to the fact that Constraint (2) is a function of decision variables (i.e., $\boldsymbol{s}^t, \boldsymbol{a}^t, \mathit{\Delta}^t$) whose values are decided at optimization time.

binary search refinement in the constraint generation process. (iii) SCIPPlan only generates temporal constraints as needed, thus substantially reducing MINLP size. (iv) Constraint (2) only perturbs $\boldsymbol{g}^t$ by changing its coefficients and not adding any additional decision variables, thus allowing SBB solvers to reuse information between iterations (e.g., warm start features). Given the descriptions of $\mathcal{M}(\Pi, H)$, $\mathcal{S}(\pi, \epsilon)$ and Constraint (2), SCIPPlan is outlined by Algorithm 1.

Algorithm 1 SCIPPlan

```
1: $H \leftarrow 1$, $\pi \leftarrow \emptyset$, $x_1^t, x_2^t, \boldsymbol{g}^t \leftarrow \emptyset$, $\epsilon \leftarrow$ small numerical constant
2: while $\pi$ is $\emptyset$ do
3:     $\pi \leftarrow \mathcal{M}(\Pi, H)$
4:     if $\pi$ is $\emptyset$ then
5:         $H \leftarrow H + 1$.
6:     else $|_L x_1^t, x_2^t|_R, \boldsymbol{g}^t \leftarrow \mathcal{S}(\pi, \epsilon)$
7:         if $|_L x_1^t, x_2^t|_R, \boldsymbol{g}^t$ are $\emptyset$ then
8:             return $\pi$
9:         else $\mathcal{M}(\Pi, H) \leftarrow \boldsymbol{g}^t(k\Delta^t) \leq 0$ where $k = \frac{x_2^t + x_1^t}{2\Delta^t}$
```

5.4 Illustrative Example Revisited

We have previously ended the illustrative example where the master problem $\mathcal{M}(\Pi, H)$ (i.e., the SSB solver) returned the plan $\pi = \langle \bar{a}_x^1 = 0, \bar{a}_y^1 = 0, \bar{\Delta}^1 = 0, \bar{a}_x^2 = 1, \bar{a}_y^2 = 1, \bar{\Delta}^2 = 4, \bar{a}_x^3 = 0, \bar{a}_y^3 = 0, \bar{\Delta}^3 = 0, \bar{a}_x^4 = 1, \bar{a}_y^4 = 1, \bar{\Delta}^4 = 4 \rangle$ as visualized by Figure 1 (a). The subproblem $\mathcal{S}(\pi, \epsilon)$ will detect the zero-crossing interval by simulating the transition function T for all time steps $t \in \{1, \dots, H\}$ and detect the first violation of constraint c_3 which occurs within the interval $[0, 2]$ over duration $\bar{\Delta}^4 = 4$. Given the identified zero-crossing interval $[0, 2]$ for time step $t = 4$ and the violated constraint c_3, the following constraint (i.e., checking for the obstacle at the midpoint of the zero-crossing)

$$\begin{aligned} g_1^4 : 4 \geq s_x^4 + (0.25)a_x^4\Delta^4 \vee 6 \leq s_x^4 + (0.25)a_x^4\Delta^4 \\ \vee\, 4 \geq s_y^4 + (0.25)a_y^4\Delta^4 \vee 6 \leq s_y^4 + (0.25)a_y^4\Delta^4 \end{aligned}$$

will be added to the master problem. As visualized by Figure 1 (b-d), the master problem would then be re-solved and further constraints will be generated if needed. Once no zero-crossings are detected in a solution, that plan would be returned as the final plan π^δ in Figure 1 (d).

6 Experimental Results

In this section, we test the computational efficacy of SCIPPlan on three metric hybrid factored planning problems Π^δ, namely `HVAC` [1], `ComplexPouring` [17], [3], `NavigationJail`, against ENHSP [17], and on one metric factored planning problem Π, namely `NavigationMud` [16], against TF-Plan [19] [5] with respect to run time and solution quality. Unless otherwise stated, all domains minimize total time (i.e., makespan) $Q(\boldsymbol{s}^{t+1}, \boldsymbol{a}^t) = -\Delta^t$.

6.1 Domain Descriptions

In this section, we describe the benchmark domains in detail. The domains were chosen to test the capabilities of SCIPPlan on metric optimization, handling nonlinear transitions and concurrency.

Heating, Ventilation and Air Conditioning is the problem of heating different rooms $r \in R$ of a building upto a desired temperature by sending heated air b_r. The temperature of a room h_r^{t+1} is bilinear function of its current temperature h_r^t, the volume of heated air sent to the room b_r, the temperature of the adjacent rooms $h_{r'}^t$ and the duration Δ^t of the control input at time step t where $r' \in Adj(r) \subset R$ denotes the set of adjacent rooms to room r. The dynamics of the domain are described as follows:

$$h_r^{t+1} = h_r^t + \frac{\Delta^t}{C_r}(b_r + \sum_{r' \in Adj(r)} \frac{h_{r'}^t - h_r^t}{W_{r,r'}}) \tag{3}$$

for all $r \in R, t \in \{1, \dots, H\}$ where C_r and $W_{r,r'}$ are parameters denoting the heat capacity of room r and the heat resistance of the wall between r and r', respectively. Moreover, the initial and the goal constraints are described as $h_r^1 = H_r^{init}$ and $h_r^{H+1} = H_r^{goal}$ for all rooms $r \in R$ where the parameters H_r^{init} and H_r^{goal} denote the initial and goal temperatures of the rooms, respectively.

ComplexPouring is the problem of filling buckets $b \in B$ upto a desired volume with the water that is initially stored in the tanks $u \in U$. The volume of a bucket v_b^{t+1} (or a tank) is a nonlinear function of its current volume v_b^t (or v_u^t), volume of water poured in (and out) from (and to) other tanks and Δ^t at time step t. The dynamics of the domain are described as follows:

$$v_b^{t+1} = v_b^t + \overrightarrow{v}_b^t - \overleftarrow{v}_b^t \qquad \forall b \in B \cup U \tag{4}$$

$$\overrightarrow{v}_b^t = \sum_{u \in U} \Delta^t p_{u,b}^t (2R_u \sqrt{v_u^t} - R_u^2) \qquad \forall b \in B \cup U \tag{5}$$

$$\overleftarrow{v}_b^t = \sum_{u \in B \cup U} \Delta^t p_{b,u}^t (2R_b \sqrt{v_b^t} - R_b^2) \qquad \forall b \in U \tag{6}$$

$$0 \le v_b^t + \overrightarrow{v}_b^t - \overleftarrow{v}_b^t \le {V^{max}}_b^t \qquad \forall b \in B \cup U \tag{7}$$

[5] We note that TF-Plan does not handle i) discrete variables, ii) global or goal constraints, or iii) support dynamic time discretization, but can handle exponential transitions and complex metric objectives (e.g., `NavigationMud`).

for all $t \in \{1, \dots, H\}$ where $p^t_{b,u} \in \{0, 1\}$ is a binary decision variable denoting whether tank b pours into bucket (or tank) u at time step t, and R_b and $V^{max}{}^t_b$ are parameters denoting the flow rate and capacity of bucket (or tank) b, respectively. Further, the initial and the goal constraints are described as $v^1_b = V^{init}_b$ for all buckets and tanks $b \in B \cup U$ and $v^{H+1}_b \geq V^{goal}_b$ for all buckets $b \in B$ where the parameters V^{init}_b and V^{goal}_b denote the initial and goal volumes of tanks and buckets, respectively.

NavigationJail is a two-dimensional $d \in \{x, y\} = D$ path- finding domain that is designed to test the ability of planners to handle instantaneous events. The location of the agent l^{t+1}_d is a nonlinear function (i.e., cubic polynomial) of its current location l^t_d, speed v^t_d, acceleration a^t_d and Δ^t at time step t. Moreover, the agent can be instantaneously relocated to its initial position L^{init}_d for all dimensions $d \in D$ and set its speed to 0 if it travels through a two-dimensional jail area that is located in the middle of the maze with the corner points J^{min}_d, J^{max}_d for all $d \in D$. The system dynamics of the domain is described as follows:

$$l'^t_d = l^t_d + v^t_d \Delta^t + 0.5 a^t_d (\Delta^t)^2 \qquad \forall d \in D \qquad (8)$$

$$v'^t_d = v^t_d + a^t_d \Delta^t \qquad \forall d \in D \qquad (9)$$

$$\textbf{if} \quad \forall d \in D \quad J^{min}_d \leq l'^t_d \leq J^{max}_d \qquad (10)$$

$$\textbf{then} \quad l^{t+1}_d = L^{init}_d,\ v^{t+1}_d = 0 \qquad \forall d \in D \qquad (11)$$

$$\textbf{else} \quad l^{t+1}_d = l'^t_d,\ v^{t+1}_d = v'^t_d \qquad \forall d \in D \qquad (12)$$

$$L^{min}_d \leq l'^t_d \leq L^{max}_d, \quad A^{min}_d \leq a^t_d \leq A^{max}_d \qquad \forall d \in D \qquad (13)$$

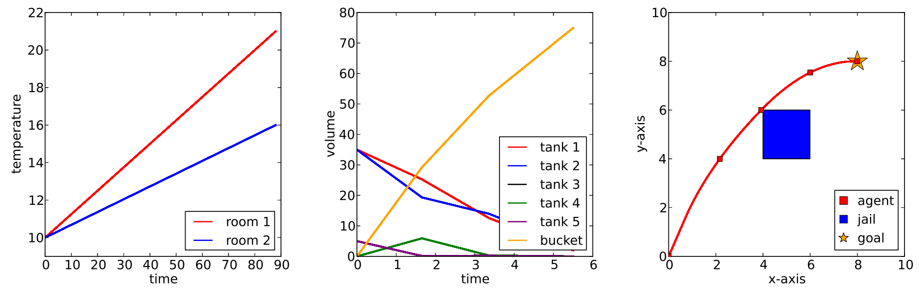
for all $t \in \{1, \dots, H\}$ where (L^{min}_d,L^{max}_d) and (A^{min}_d,A^{max}_d) are the minimum and the maximum boundaries of the maze and the control input for dimension $d \in D$, respectively. The goal of the domain is to find a path from the initial location L^{init}_d to the goal location L^{goal}_d for all dimensions $d \in D$. The initial and the goal constraints are described as $l^1_d = L^{init}_d$, $v^1_d = 0$, and $l^{H+1}_d = L^{goal}_d$ for all dimensions $d \in D$, respectively.

NavigationMud is a two-dimensional $d \in \{x, y\} = D$ domain that is designed to test the ability of planners to handle transcendental functions with general optimization metrics. The location of the agent l^{t+1}_d is a nonlinear function (i.e., exponential) of its current location l^t_d and positional displacement action p^t_d at time step t due to higher slippage in the center of the maze. The system dynamics of the domain is described as follows:

$$l^{t+1}_d = l^t_d - 0.99 + p^t_d \frac{2.0}{1.0 + e^{-2y^t}} \qquad \forall d \in D \qquad (14)$$

$$y^t = \sqrt{\sum_{d \in D} (l^t_d - \frac{L^{max}_d - L^{min}_d}{2.0})^2} \qquad (15)$$

$$L^{min}_d \leq l^t_d \leq L^{max}_d, \quad P^{min}_d \leq p^t_d \leq P^{max}_d \qquad \forall d \in D \qquad (16)$$



(a) HVAC (b) ComplexPouring (c) NavigationJail

Fig. 2. Visualization of example plans generated by SCIPPlan. The inspection of plan traces show from left to right: linear, piecewise linear, and nonlinear state transitions as a function of time. As observed in Table 2, we remark that the nonlinear domain (right) requires significantly more compute time than the linear (left) and piecewise linear (middle) domains.

for all $t \in \{1, \dots, H\}$ where (P_d^{min},P_d^{max}) are the minimum and the maximum boundaries of the positional displacement for dimension $d \in D$, respectively.

The objective of the domain is to find a path from the initial location L_d^{init} that is described by the constraint $l_d^1 = L_d^{init}$ for all dimensions $d \in D$ with the minimum total Manhattan distance $\sum_{t \in \{1,\dots,H\}} \sum_{d \in D} |l_d^{t+1} - L_d^{goal}|$ from the goal location L_d^{goal} over all time steps t.

6.2 Implementation Details

SCIPPlan is a compilation-based planner that consists of the constraints compiled from RDDL [15] using the syntax presented in Table 1, RDDLsim domain simulator [15] and the dynamically generated temporal constraints (2). At every iteration, SCIPPlan only adds the set of constraints that correspond to the first zero-crossing interval, or terminates if the plan is valid with respect to the discretization parameter ϵ. In SCIPPlan, we modeled the actions b_r^t and a_d^t from `HVAC` and `NavigationJail` domains as decision variables with continuous domains. In PDDL+, we incremented and decremented the actions as a function of time with some constant rate z. Further in the `NavigationJail` domain, we have modeled the if-else-then statements (10-12) using events in PDDL+ (as opposed to using global constraints) since going into the jail location can still lead to feasible plans. In the `HVAC` and `NavigationJail` domains, we tested ENHSP with relaxed goal settings where the respective equality goal constraints were relaxed to the following constraints:

$$H_r^{goal} - z \leq h_r^{H+1} \leq H_r^{goal} + z \qquad \forall r \in R \qquad (17)$$

$$L_d^{goal} - z \leq l_d^{H+1} \leq L_d^{goal} + z \qquad \forall d \in D \qquad (18)$$

Table 2. Comparison of plan quality produced and run times by SCIPPlan (SP), ENHSP (EP) and TF-Plan (TF) with respect to the given domain metrics. We optimize both Makespan metric objectives (middle four columns) and General metric objectives (last column). Lower values indicate better solution quality.

	Makespan				General
Domain	Quality		Run Time		Run Time
HVAC	SP^0	$EP^{0.1}$	SP^0	$EP^{0.1}$	SP^0
(2,R)	**88.00**	145.00	$\leq$ **0.01**	1.02	0.02
(2,D)	**88.00**	145.00	$\leq$ **0.01**	1.02	0.19
Pouring	SP^0	EP	SP^0	EP	SP^0
(3,1)	**4.30**	11.00	0.10	0.32	**0.01**
(5,1)	**5.51**	19.00	1.38	**0.41**	0.87
(4,2)	**7.67**	22.00	0.93	**0.37**	0.58
(9,2)	**1.69**	10.00	0.90	0.37	**0.08**
NJail	$SP^{0.05}$	$EP^{0.1}$	$SP^{0.05}$	$EP^{0.1}$	-
(-1.0,1.0)	**13.59**	-	**281.75**	$\geq$ 1800	-
(-0.5,0.5)	**13.63**	-	**60.94**	$\geq$ 1800	-
(-0.2,0.2)	**13.35**	-	**59.29**	$\geq$ 1800	-
NMud	$SP^{0.05}$	TF	$SP^{0.05}$	TF	-
(-1.0,1.0)	**64.25**	65.23	**15.46**	30.00	-
(-0.5,0.5)	140.35	**136.55**	232.84	240.00	-
(-0.2,0.2)	800.00	**360.38**	1800.00	**960.00**	-

due to the continuous domains of state $\boldsymbol{s} \in S^c$ and action $\boldsymbol{a} \in A^c$ variables, and the fact that ENHSP identified these domains to be infeasible with equality constraints. We tested SCIPPlan under different optimality gap parameters g for the underlying SBB solver. For both parameter settings z and g, we will use the notation $\mathbf{SP^x}$ to report results for SCIPPlan under the optimality gap setting $g = x$, and $\mathbf{EP^x}$ for ENHSP under the rate setting $z = x$. Finally, when the total makespan is not minimized, in SCIPPlan we constrained the total makespan by a large constant such that $\sum_{t\in\{1,\dots,H\}} \Delta^t \leq M$.

6.3 Comparison of the Solution Quality and Run Time Performance

In Table 2, we compare the quality of plans produced and the run times of SCIPPlan, ENHSP and TF-Plan with respect to the chosen optimization metric under the best performing parameter settings. From left to right, the first column of Table 2 specifies the domains and problem instances solved. The second and third columns present the optimal makespan found by the respective planners. The fourth and firth columns present the computational effort that is required to produce the metrics presented in the second and third columns. The sixth column presents the running time (seconds) that is required to optimize the general metric variants of the original domains.

6.4 Computational Performance

In this section, we investigate the efficiency of using SCIPPlan for solving metric hybrid factored planning problems in nonlinear domains. We ran the experiments on MacBookPro with 2.8 GHz Intel Core i7 16GB memory. We optimized the nonlinear encodings using SCIP 4.0.0 [10] with 1 thread, and 30 minutes total time limit per domain instance.

Comparison of solution qualities The detailed inspection of the columns associated with solution quality shows that SCIPPlan can successfully find high quality plans in almost all the instances with optimality gap parameter $g \leq 0.05$, except the largest domain `NavigationMud` (-0.2,0.2). In contrast, we observe that in `HVAC` and `ComplexPouring` domains, ENHSP can find plans with on average 60% lower quality compared to SCIPPlan. Moreover in `NavigationJail` domain, neither $EP^{0.1}$ nor $EP^{0.01}$ found feasible plans within time limit. In `NavigationMud` domain, we tested the scalability of SCIPPlan against TF-Plan. We found that SCIPPlan is competitive with TF-Plan with respect to the solution quality of the plans found in the small and medium size instances, whereas the large instance `NavigationMud(-0.2,0.2)` is hard to optimize (i.e., the plan does not contain actions other than no-ops) for SCIPPlan. We note that unlike TF-Plan, we currently do not leverage parallel computing, which is one of the main strengths of Tensorflow to handle large scale optimization problems. In Figure 2, we visualize the plan traces to get a better understanding of what makes a domain hard in terms of plan computability. The inspection of plan traces shows from left to right: linear, piecewise linear and nonlinear state transitions. Together with the computational results presented in Table 2, we confirm that domains with nonlinear state transitions (e.g., NavigationJail) are significantly computationally harder compared to linear (e.g., HVAC) and piecewise linear (e.g., ComplexPouring) domains.

Comparison of run time performances The inspection of the last three columns shows that SCIPPlan finds high quality plans with little computational effort in `HVAC` and `ComplexPouring` domains, whereas it takes on average 135 seconds and 125 seconds to find high quality plans for `NavigationJail` and `NavigationMud` domains, with the exception of the largest instance `NavigationMud` (-0.2,0.2) for horizon $H = 50$. The benefit of spending computational resources to provide stronger optimality guarantees is justified in Figure 3, where we plot the increase in plan quality as a function of optimality gap parameter g. Figure 3 shows that spending more computational resources can significantly improve the quality of the plan found as the instances get harder to solve.

6.5 General Metric Specifications

Finally, we test SCIPPlan on general metrics of interest in `HVAC` and `ComplexPouring` domains and measure the effect on run time. In the `HVAC` domain, we modify

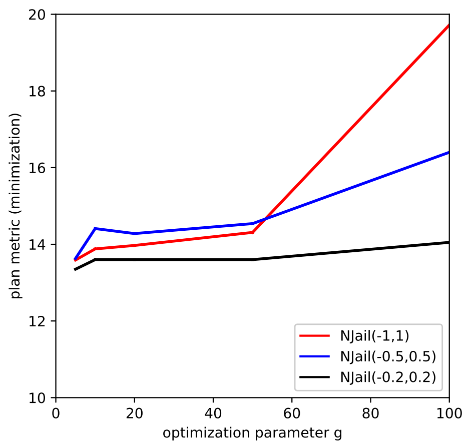


Fig. 3. The increase in plan quality (lower is better for minimization) as a function of optimality gap parameter g for SCIPPlan on NavigationJail domain.

the metric to minimize the total cost $\sum_{t\in\{1,\dots,H\}}\sum_{r\in R} c_r b_r^t$ of heating all rooms $r \in R$ of a building for all time steps where the parameter c_r denotes the unit cost of heating room $r \in R$. Similarly in `ComplexPouring` domain, we minimize the total number of times we pour from one tank to the bucket (or other tanks) across all time steps such that $\sum_{t\in\{1,\dots,H\}}\sum_{u\in U}\sum_{b\in B\cup U} p_{u,b}^t$. The results presented in the last column of Table 2 show that the performance of SCIPPlan is on average the same for general metric optimization and makespan optimization. As demonstrated in `NavigationMud` and modified `HVAC` and `ComplexPouring` domains, SCIPPlan finds high quality plans with respect to general metric specifications.

7 Conclusion

In this paper, we presented a novel SCIP-based planner (SCIPPlan) that can plan in metric hybrid factored planning domains with nonlinear transcendental functions such as exponentials and instantaneous continuous actions. In SCIPPlan, we leveraged the spatial branch-and-bound solver of SCIP inside a nonlinear constraint generation framework where candidate plans are iteratively checked for temporal infeasibility using a domain simulator, and the sources of infeasibilities are repaired through a novel nonlinear constraint generation algorithm. Experimentally, we have shown that SCIPPlan can plan effectively on a variety of domains and outperformed ENHSP in terms of plan quality and run time performance. We have further shown that SCIPPlan is competitive with the Tensorflow-based planner (TF-Plan) in highly nonlinear domains with exponential transitions and general metric specifications.

References

1. Agarwal, Y., Balaji, B., Gupta, R., Lyles, J., Wei, M., Weng, T.: Occupancy-driven energy management for smart building automation. In: ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Building. pp. 1–6 (2010)
2. Boutilier, C., Dean, T., Hanks, S.: Decision-theoretic planning: Structural assumptions and computational leverage. JAIR **11**(1), 1–94 (1999), http://dl.acm.org/citation.cfm?id=3013545.3013546
3. Bryce, D., Gao, S., Musliner, D., Goldman, R.: SMT-based nonlinear PDDL+ planning. In: 29th AAAI. pp. 3247–3253 (2015), http://dl.acm.org/citation.cfm?id=2888116.2888168
4. Cashmore, M., Fox, M., Long, D., Magazzeni, D.: A compilation of the full PDDL+ language into SMT. In: ICAPS. pp. 79–87 (2016), http://dl.acm.org/citation.cfm?id=3038594.3038605
5. Coles, A.J., Coles, A.I., Fox, M., Long, D.: COLIN: Planning with continuous linear numeric change. JAIR pp. 1–96 (2012)
6. Fox, M., Long, D.: Modelling mixed discrete-continuous domains for planning. JAIR **27**(1), 235–297 (2006), http://dl.acm.org/citation.cfm?id=1622572.1622580
7. Fox, M., Long, D., Magazzeni, D.: Plan-based policies for efficient multiple battery load management. CoRR **abs/1401.5859** (2014), http://arxiv.org/abs/1401.5859
8. Henzinger, T.A., Kopke, P.W., Puri, A., Varaiya, P.: What's decidable about hybrid automata? In: Proceedings of the Twenty-seventh Annual ACM Symposium on Theory of Computing. pp. 373–382. ACM, New York, NY, USA (1995). https://doi.org/10.1145/225058.225162, http://doi.acm.org/10.1145/225058.225162
9. Löhr, J., Eyerich, P., Keller, T., Nebel, B.: A planning based framework for controlling hybrid systems. In: ICAPS. pp. 164–171 (2012), http://www.aaai.org/ocs/index.php/ICAPS/ICAPS12/paper/view/4708
10. Maher, S.J., Fischer, T., Gally, T., Gamrath, G., Gleixner, A., Gottwald, R.L., Hendel, G., Koch, T., Lübbecke, M.E., Miltenberger, M., Müller, B., Pfetsch, M.E., Puchert, C., Rehfeldt, D., Schenker, S., Schwarz, R., Serrano, F., Shinano, Y., Weninger, D., Witt, J.T., Witzig, J.: The scip optimization suite 4.0. Tech. Rep. 17-12, ZIB, Takustr.7, 14195 Berlin (2017)
11. Mitten, L.G.M.: Branch-and-bound methods: General formulation and properties. Operations Research **18**(1), 24–34 (1970), http://www.jstor.org/stable/168660
12. Penna, G.D., Magazzeni, D., Mercorio, F., Intrigila, B.: UPMurphi: A tool for universal planning on PDDL+ problems. In: ICAPS. pp. 106–113 (2009), http://dl.acm.org/citation.cfm?id=3037223.3037238
13. Piotrowski, W.M., Fox, M., Long, D., Magazzeni, D., Mercorio, F.: Heuristic planning for hybrid systems. In: AAAI. pp. 4254–4255 (2016), http://www.aaai.org/ocs/index.php/AAAI/AAAI16/paper/view/12394
14. Raghavan, A., Sanner, S., Tadepalli, P., Fern, A., Khardon, R.: Hindsight optimization for hybrid state and action mdps. In: Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17). San Francisco, USA (2017)
15. Sanner, S.: Relational dynamic influence diagram language (rddl): Language description (2010)
16. Say, B., Wu, G., Zhou, Y.Q., Sanner, S.: Nonlinear hybrid planning with deep net learned transition models and mixed-integer linear programming. In: Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17. pp. 750–756 (2017). https://doi.org/10.24963/ijcai.2017/104, https://doi.org/10.24963/ijcai.2017/104

17. Scala, E., Haslum, P., Thiébaux, S., Ramírez, M.: Interval-based relaxation for general numeric planning. In: ECAI. pp. 655–663 (2016). https://doi.org/10.3233/978-1-61499-672-9-655, http://dx.doi.org/10.3233/978-1-61499-672-9-655
18. Shin, J.A., Davis, E.: Processes and continuous change in a sat-based planner. Artificial Intelligence **166**(1-2), 194–253 (Aug 2005). https://doi.org/10.1016/j.artint.2005.04.001, http://dx.doi.org/10.1016/j.artint.2005.04.001
19. Wu, G., Say, B., Sanner, S.: Scalable planning with tensorflow for hybrid nonlinear domains. In: Proceedings of the Thirty First Annual Conference on Advances in Neural Information Processing Systems (NIPS-17). Long Beach, CA (2017)