Mixed Discrete-Continuous Planning with Complex Behaviors

Enrique Fernandez-Gonzalez
Massachusetts Institute of Technology
Computer Science and Artificial Intelligence Laboratory
32 Vassar Street, Building 32-224, Cambridge, MA 02139
efernan@mit.edu

Abstract

Over the last few years, we have witnessed a tremendous increase in the capabilities of robots. However, robots are still largely teleoperated by humans or controlled by scripts written by experts in advance, in a time-consuming and costly manner. Many practical applications require more autonomous robots, but the current state of the art in planning is not well suited for this task.

This thesis aims to fulfill this need by developing a mixed discrete-continuous temporal planner, *Scotty*, capable of reasoning with complex robot behaviors and accepting high level temporally extended mission goals. Scotty leverages the proven performance of heuristic forward search for symbolic planning with the versatility of trajectory optimization for resolving complex non-linear robot behaviors.

Introduction

Our generation is witnessing a revolution in robotics. Over the last decade we have seen tremendous improvements in robot hardware, perception and control. Robots have transitioned from being, in general, expensive repetitive machines in automotive factories or relatively simple experiments with limited capabilities in research labs, to sophisticated machines able to walk, run, jump, fly or even drive autonomously on highways.

All these recent achievements are due to advances in hardware, perception, control, and the availability of small factor high performance computing power. However, as impressive as these robots are, most of them are either completely scripted in advance, or teleoperated by humans. This is works well for some applications. However, these approaches for operating robots will not be sufficient for many important robotic applications, such as robotic exploration of the Solar System. We need robots able to operate in remote hostile environments in which the availability of direct fast communication with human operators cannot be assumed. Such missions will require a more advanced level of autonomy than what we currently have.

To this day, we are largely not doing any automated planning with robots, as the current state of the art does not fulfill this need. The activity planning community has made impressive advances in symbolic planning, especially since the introduction of heuristic forward search. However, most activity planners, only reason with discrete conditions and effects. Some have been extended to consider limited forms of continuous linear time-evolving effects (Coles et al. 2012; 2010), but still focus mainly on logistic or planning competition problems and are unable to handle the more complicated non-linear effects required to model robot dynamics.

On the other hand, the AI community has expressed an increasing interest in the joint problem of activity and motion planning. Many interesting approaches have emerged over the last few years (Srivastava et al. 2014; Cambon, Alami, and Gravot 2009; Garrett, Lozano-Pérez, and Kaelbling 2014). However, most of these approaches focus on the manipulation problem, which is hard and interesting but requires specific assumptions, and often neglect dynamics, as they are not needed to model common manipulation scenarios.

Finally, we have lately seen great advances in the control of complex underactuated robots, in part thanks to the success of trajectory optimization. There are many examples of successful robot behaviors being implemented using this approach that were impractical to solve in a reasonable amount of time just a few years ago (Fallon et al. 2015; Dai, Valenzuela, and Tedrake 2014). To our knowledge there are no planners able to reason with these complex behaviors.

This thesis aims to address this need by developing a mixed discrete-continuous temporal planner, **Scotty**, able to plan with complex robot behaviors. The Scotty planner leverages the proven performance of heuristic forward search for symbolic planning and the recent advances in trajectory optimization to reason with complex robot behaviors requiring non-linear dynamics.

Problem Description

Scotty is a hybrid temporal planner and, as such, its problem description inherits the main elements from the activity planning literature. Scotty's problem statement is similar to PDDL2.1(Fox and Long 2011) in that the state of the system is given by true/false propositions and values to numeric state variables, and that activities have preconditions (*at start*, *over all*, *at end*) and effects (*discrete at start* and *at end* and also *continuous*). Scotty's problem statement differs from PDDL2.1 in that it allows more expressive goal specifications and more complex activities with non-linear dynamics and conditions that we call **behaviors**.

Scotty takes as inputs an *initial state*, a *planning domain* and *time-evolved goal representation* and produces a *plan*. The input to Scotty is a tuple $\langle I, A, G \rangle$ where

- I is the initial state of the system at $t = 0$. The state of the system at time t, $X(t)$ is given by:
  - The set of propositions that hold at time t
  - An assignment to all the continuous state variables of the system, $x_i(t)$

  the initial state of the system is then $X(0)$
- D is the planning domain that defines the allowed *activities* and *behaviors*
- G is the temporally-extended goal specification

The output of Scotty is a temporal plan satisfying the goal specification.

Temporally-extended goals

The goal specification in the classic temporal planning problem simply consists of the set of proposition that need to hold at the end of the plan. This definition is static: it does not matter what happens between the start and the end of the plan as long as the objectives are satisfied at the end. However, in most real world problems objectives evolve over time and there may be temporal restrictions between them.

We will use Qualitative State Plans (QSPs)(Li and Williams 2011; Léauté and Williams 2005; Hofmann and Williams 2006) to represent these temporally-extended goals. QSPs use episodes and temporal constraints to specify the users' objectives and requirements on states, resources and time. Formally, a QSP is represented as a 3-tuple $< E, EP, TC >$, where

- E is a set of events. Each event $e \in E$ can be assigned a non-negative real value which denotes a point in time.
- EP is a set of episodes. Each episode specifies an allowed state trajectory between a starting and an ending event. They are used to represent the state constraints. An episode, ep, is a tuple $< e_S, e_E, l, u, sc >$ where:
  - e_S and e_E in ep are the start and end events of the state trajectory.
  - l and u are the lower and upper bounds on the temporal duration of the episode.
  - sc is a state constraint that must hold true over the duration of the episode. The state constraint sc can be either a discrete predicate, or a condition over the state variables.
- TC is set of simple temporal constraints. It represents the temporal requirement between events in E, and can be viewed as a special type of episode without state constraint. A simple temporal constraint(Dechter, Meiri, and Pearl 1991) is a tuple $< e_S, e_E, lb, ub >$ where:
  - e_S and e_E in E are the start and end events of the temporal constraint.
  - lb and ub represent the lower and upper bounds of the duration between events e_S and e_E , where $lb \leq Time(e_E) - Time(e_S) \leq ub$, ($lb \in \mathbb{R} \cup -\infty, ub \in \mathbb{R} \cup +\infty$).

Activities and composable behaviors

The *domain* of the planning problem is given by the *durative activities* and the *behaviors*.

Durative activities are similar to the ones defined in the temporal planning literature and have a bounded controllable duration, and a set of discrete effects and conditions defined *at start*, *over all* and *at end*. The conditions can also be continuous.

In Scotty we define a new type of activities that we call **behaviors**. These behaviors preserve the main elements of the durative activities but differ from these in that they also encode continuous non-linear effects that can represent, for example, the dynamics of a robot moving in an environment. We will consider that *durative activities* are a special case of *behaviors* with no continuous effects.

Consider for example a behavior *collect-water-column-data* of an autonomous underwater vehicle. This behavior represents the AUV taking measurements in a water column, and has the discrete effect *at end* of having collected the data (*data-collected*). This behavior also has the discrete *over all* conditions that the engine needs to be on and the sensors activated while the behavior is being executed. It also has continuous *over all* conditions that specify that the AUV needs to stay within some coordinates while taking the measurements ($x, y \in Region$). Finally, the depth before collecting the data needs to be within 100 and 120 meters ($100 \leq z \leq 120$), and the depth at the end has to be smaller than 40 meters ($z \leq 40$). These represent the continuous conditions *at start* and *at end*.

Moreover, the AUV needs to move in an ascending spiral of fixed radius while collecting the data. This is represented by the non-linear dynamics of the behavior:

$$\dot{x}(t) = v \cdot cos(\theta) \tag{1}$$
$$\dot{y}(t) = v \cdot sin(\theta) \tag{2}$$
$$\dot{z}(t) = v_z(t) \tag{3}$$
$$\dot{\theta}(t) = \omega \tag{4}$$
$$\frac{v}{\omega} = R \tag{5}$$

The dynamics of this behavior affect state variables x, y, z and θ and their velocities. These dynamics are driven by control variables v_x, v and ω which are assumed to be controllable within some bounds (actuation limits) during execution. Additionally, there may be other state constraints affecting the execution of this behavior. These constraints can be user-specified as part of the input QSP ('ensure the battery level is always greater than 25%' or 'stay within the high reception region' for example), or imposed by other activities or behaviors being executed at the same time.

An important property of behaviors is that they are **composable**. That is, behaviors can be pieced together sequentially one after another. Behaviors have **inlet** and **outlet con-**

nectors. In this example both the inlet and outlet of the behavior are of type *pose*. That is, behavior *collect-water-column-data* takes a starting pose and *transforms* it into a different pose at the end of its execution. Behaviors with compatible connectors can be connected together. For example, in order to start the *collect-water-column-data* behavior, the AUV first needs to reach the water column region at a certain depth. A prior behavior *navigate*, also with *pose* connectors, could be the one taking the AUV from it's starting pose to a valid starting pose for *collect-water-column-data*. In the same way, *collect-water-column-data* could be followed by an additional *navigate* behavior that could take the AUV to its final rendezvous region. A third behavior *surface* could have a *position* inlet requiring the AUV to be in the rendezvous region before ascending to the surface. Because the *position* connector (x, y, z) is less specific than the *pose* connector (*position* and *orientation*), the *navigate* behavior can connect to the *surface* behavior. Scotty ensures that the connector constraints are satisfied in the solution plan. While multiple durative activities can be executed simultaneously, we only allow simultaneous execution of behaviors that do not affect the same state variables.

Solution

Scotty returns a plan satisfying the temporally-extended goals and conditions described in the input QSP. The solution plan consists of a list of scheduled behaviors where each behavior b has an execution start time t_b, duration d_b, and a trajectory of the value of each of its control variables from t_b to $t_b + d_b$. The returned plan also needs to satisfy all the behavior discrete and continuous conditions as well as the connection constraints between behaviors.

Work so far

In the initial stage of this thesis we first approached the problem of combining heuristic forward search symbolic planning with trajectory optimization by solving a simplified problem. Instead of considering arbitrary dynamics, in this stage continuous effects are assumed to be linear time varying. While we allow full discrete activity planning, we restrict the continuous effects to the ones described before, and we assume the environment is obstacle free. Therefore we do not use the robot behaviors defined earlier yet, but durative activities with some modifications.

We solve this problem by combining heuristic forward search and trajectory optimization through solving linear programs that are used to resolve the continuous effects and to test the consistency of partial plans.

This stage was completed in Spring 2015 and resulted in an IJCAI-15 publication (Fernandez, Karpas, and Williams 2015).

Simplified Problem Statement

The simplified problem statement is framed as a standard PDDL 2.1(Fox and Long 2011) planning problem with some modifications that allow us to define activities with continuous effects that depend on *bounded control variables*. These bounded control variables represent, in general, velocities

```
(:durative-action navigate
:control-variables ((velX) (velY))
:duration (and (<= ?duration 5000))
:condition (and
                (over all (>= (velX) -4)) (over all (<= (velX) 4)))
                (over all (>= (velY) -4)) (over all (<= (velY) 4)))
                (over all (<= (x) 700)) (over all (>= (x) 0))
                (over all (<= (y) 700)) (over all (>= (y) 0))
                (at start (AUV-ready)))
:effect (and
            (at start (not (AUV-ready)))
            (at end (AUV-ready))
            (increase (x) (* 1.0 (velX) #t))
            (increase (y) (* 1.0 (velY) #t))))
```

Figure 1: Navigate activity modified by continuous control variables *velX* and *velY*.

that are bounded (the actuation limits of the system). Similarly to PDDL 2.1, durative activities have a bounded controllable duration, discrete effects and continuous and discrete conditions defined *at start*, *over all* and *at end*. Continuous conditions are limited to linear inequalities over the state variables according to:

$$\mathbf{c}^{\mathbf{T}}\mathbf{x}' \leq 0 \tag{6}$$

, where $\mathbf{x}' = (x_1, \ldots, x_{n_x}, 1)^T$ and $\mathbf{c} \in \mathbb{R}^{n_x+1}$ is a vector of coefficients, with n_x being the number of state variables of the system.

The simplified problem statement differs from PDDL 2.1 in the effects on continuous variables. Each activity has a set of control variables, which can be seen as continuous parameters — each of those constrained by lower and upper bounds. The continuous effects of the activity are similar to those of PDDL 2.1, except they are affected by the value chosen for the control variables. Here we restrict each continuous effect to involve only a *single* control variable, c_{var}, and thus each continuous effect can be defined by $\langle x, c_{var}, k\rangle$, where x is a state variable, c_{var} is a control variable, and k is a constant.

In the simple case, where a single continuous effect $\langle x, c_{var}, k\rangle$ is active from time t_{start} to time t_{end} with c_{var} fixed to a constant value of c throughout the duration, then $x(t)$, the value of state variable x at time t is defined by $x(t) = x(t_{start}) + k \cdot c \cdot (t - t_{start})$ with $t_{start} \leq t \leq t_{end}$.

Multiple continuous effects on the same state variable are additive, and thus $x(t)$ is defined by:

$$x(t) = x(0) + \int_0^t C_x(\tau)d\tau \tag{7}$$

where $C_x(t)$ is the sum of the values of the control variables in active continuous effects modifying x at time t (represented by the set E).

$$C_x(t) = \sum_{\langle x, c_{var}, k\rangle \in E} k \cdot c_{var}(t) \tag{8}$$

where $c_{var}(t)$ denotes the value chosen for the control variable c_{var} at time t. An example *navigate* activity for a robot is shown in Figure 1. Note the bounded *control variables* velX and velY.

In this stage the *goal* simply consists of the discrete and continuous conditions that need to hold at the end. Temporally-extended goals are not supported yet.

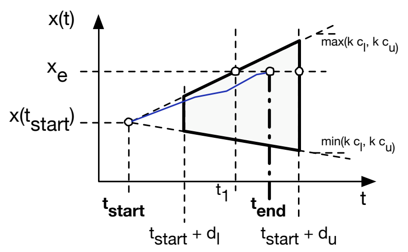

Figure 2: Flow tube with its reachable region (shaded area). The solid blue line represents an example valid state trajectory. The flow tube contains all valid state trajectories.

Approach

In order to solve this planning problem, we merge the powerful representation of continuous effects based on *flow tubes* from Kongming (Li and Williams 2008; Li 2010) with the efficient solving method based on heuristic forward search and linear programs for consistency checking that COLIN (Coles et al. 2009) uses.

Each continuous effect of an activity is represented by a flow tube. Flow tubes represent the reachable state space region, that is, the values that the state variable can take after the activity is started. Remember that here we restrict continuous effects to linear time varying effects.

If no other flow tubes affect state variable x between t_{start} and t_{end}, then the reachable region of x represented by the flow tube $f(d_l, d_u, c_{var}, x, k)$ of an activity executed between t_{start} and t_{end} is defined by the following equations:

$$x(t_{end}) = x(t_{start}) + k \cdot \int_{t_{start}}^{t_{end}} c_{var}(t) \cdot dt \quad (9)$$

$$\text{with} \qquad c_l \leq c_{var}(t) \leq c_u \quad (10)$$

$$d_l \leq t_{end} - t_{start} \leq d_u \quad (11)$$

where $x(t_{start})$ is the value of the state variable before the activity is executed. Note that, if the value of the control variable c_{var} is constant during the execution of the activity, equation 9 reduces to $x(t_{end}) = x(t_{start}) + k \cdot c_{var} \cdot (t_{end} - t_{start})$.

Figure 2 shows a flow tube. Note that any point in the shaded region (reachable region) can be reached at the end of the activity by carefully choosing the appropriate activity duration and control variable value. In the figure, we can see how the state value x_e can be reached as fast as in $t_{end} = t_1$ if the control variable c_{var} is constant and takes its maximum possible value (c_u), or as late as $t_{end} = t_{start} + d_u$ if $c_{var}(t)$ takes smaller values.

An important characteristic of flow tubes, is that they provide a compact encoding of all feasible trajectories. In order to find a feasible plan, we connect flow tubes forward using heuristic forward search, as explained next.

We use a method based on heuristic forward search and linear programs for consistency checking that is borrowed from COLIN. The main difference is that in our formulation, continuous effects support control variables and are represented by flow tubes and, therefore, the state evolution constraints are different from COLIN's.

Activities are represented by their start and end events, analogous to the start and end snap actions used by many temporal planners (Long and Fox 2003; Coles et al. 2008). The planner needs to find the ordered sequence of start and end events that takes the system from the initial conditions to the goal and the execution time of each event. We also need to find a trajectory for the values of each activity control variable between the start and end events of the activity ($c_{var}(t)$ with $t_{start} \leq t \leq t_{end}$).

As explained before, we use heuristic forward search to find the sequence of start and end events that form a fixed plan, and linear programs to check the consistency of partial plans. The search uses Enforced Hill Climbing, which has proven to be effective in this type of problems (Hoffmann and Nebel 2001).

Search states contain the set of propositions that are known to be true due to discrete effects, and are augmented with the ongoing activities list and the bounds for all state variables. The ongoing activities list keeps track of the activities that have started but not finished at that state and is needed to keep track of the active overall discrete and continuous constraints. The lower and upper bounds for the state variables are used to prune sections of the search tree that are necessarily not feasible.

Each search state defines a partial plan as the current sequence of start and end events, and is tested for consistency with a linear program. The partial plan is feasible if the linear program has a solution. In this linear program the decision variables are the event execution times and the values of the state variables at each event. The constraints include activity duration, start, end and overall conditions and state evolution constraints that are built from the current sequence of events. These constraints are the same ones that COLIN uses (Coles et al. 2012) except for the state evolution constraint, however, due to the presence of control variables. This constraint is given by the flow tube reachability equation 9. Because the values of the control variables can change during the activity execution, and the start and end times of the activity are variables of the linear program, this equation is not linear if control variables are decision variables of the linear program. However, we can redefine the reachability region of the flow tube with the following linear inequalities:

$$x_{end} \geq x_{start} + \min(k \cdot c_l, k \cdot c_u) \cdot (t_{end} - t_{start}) \quad (12)$$

$$x_{end} \leq x_{start} + \max(k \cdot c_l, k \cdot c_u) \cdot (t_{end} - t_{start}) \quad (13)$$

, where c_l and c_u are the bounds of the control variable. Note that $\min(k \cdot c_l, k \cdot c_u)$ represents the minimum rate of change of $k \cdot c_{var}$ and reduces to $k \cdot c_l$ when $k > 0$. The more complicated expression is needed to preserve generality when $k < 0$. The same applies to the maximum rate of change. These linear inequalities represent the same flow tube reachability region described by equation (9) if each of the activity's control variables appear in only one continuous effect.

The consistency program is solved for each state in the search tree to determine the feasibility of the partial plan, and to extract the event times ($t_1 \ldots t_6$), state variable values

$(x_1 \ldots x_6)$, and control variables. These values keep changing as more steps are added to the plan during search. In order to find the state variable lower and upper bounds, the LP is solved twice per state variable (to minimize and maximize its value).

If the current search state is determined to be consistent, its heuristic value is computed and the state is added to the queue. If the state satisfies the goal conditions, a valid fixed plan has been found and Scotty proceeds to extract the flexible plan next. The last linear program used to extract the fixed plan tries to minimize the makespan of the plan, although a different optimization objective could be chosen.

The **heuristic** function used by Scotty is essentially the same used by COLIN, with minor modifications due to the use of control variables. The heuristic value for a state is the number of start or end events to reach the goal in the relaxed plan. The planning graph that COLIN expands keeps track of the state variables lower and upper bounds for each fact layer, with the caveat that activities can only grow these bounds, in a similar fashion to Metric-FF (Hoffmann 2003). COLIN calculates the positive gradient affecting each state variable by adding the positive rates of change of each ongoing activity (similarly with the negative gradient). In Scotty's case, these positive and negative gradients are found by adding the maximum (and minimum) rates of change given by the bounds of the control variables affecting each activity.

Future research

Currently, I am working on moving from the limited linear action model to a more general non-linear model that supports more interesting robot behaviors.

In particular, I am working towards supporting the following:

1. Advanced robot behaviors with non-linear dynamics
2. Temporally-extended goals and user-specified global constraints
3. Search heuristics through behavior approximations

Now instead of planning with durative activities, we plan with **behaviors**. Behaviors still support the standard discrete conditions and effects, but are extended to allow non-linear dynamics and more expressive user-specified constraints. Also, as described in the problem statement, behaviors are *composable* elements that can be connected sequentially and transform **inlets** to **outlets**. Behaviors that produce an outlet of a certain kind can connect to a behavior accepting the same kind of inlet.

As an example, imagine that the dynamics of a *navigate* behavior are given by the following Dubin's equations:

$$\dot{x}(t) = v \cdot cos(\theta) \tag{14}$$
$$\dot{y}(t) = v \cdot sin(\theta) \tag{15}$$
$$\dot{\theta}(t) = u(t) \tag{16}$$
$$|u(t)| \leq \frac{1}{\rho} \tag{17}$$

, where v is the velocity, that can be fixed or not, and ρ is the minimum turning radius.

Note that now we not only deal with much more complex dynamics than before but we also need to handle **switch points**: the connection states between subsequent behaviors. These were trivial to handle in simplified case with the linear program formulation, as they corresponded directly with the decision variables of the LP (the state variables at the switch points). However, this is harder to handle in this case. The way we deal with this is by using the connector (**inlets** and **outlets**) specification of behaviors. In effect, the navigate activity has a *pose* ($\langle x, y, \theta \rangle$) inlet and a pose outlet, meaning that it takes the robot from a starting pose to an ending pose. When connecting the *navigate* behavior with a *collect-data* behavior with analogous specification, the ending pose after *navigate* becomes the start pose for *collect-data*. Another *deliver* behavior may have an inlet of type *position* ($\langle x, y \rangle$), a subset of the *pose* type. Because the outlet of behavior *navigate* is a *pose* which is more specific than a *position* inlet, both behaviors can connect (*navigate* finishes at some position and orientation, but *deliver* only enforces the position to be inside the delivery region). We handle these relations by framing a joint *trajectory optimization* program combining sequential behaviors in which the connection constraints between them are added explicitly. Although one may think that framing large optimization programs like this is not practical, some combined task and motion planners have followed this approach with good results (Toussaint 2015). As Toussaint notes in his paper, this approach has the great advantage that it does not require discretizing the continuous states in advance, and that it lets efficient solvers choose the best switch poses, which would otherwise require very fine discretization or many randomly sampled poses with many combinatorial possibilities if done outside the trajectory optimization paradigm.

We now also consider global temporally extended goals and user-specified constraints. This may be things such as completing one of the objectives of a mission within a given time or ensuring that the robot never leaves a certain safe region. This is represented by a *Qualitative State Plan* (QSP). The nodes of the graph represent *events*, and the black arcs in between them, *episodes*. *Episodes* denote conditions that need to be satisfied between events. The temporal relations between events are specified with simple temporal constraints. These specifications will be modeled as additional constraints in the global trajectory optimization program and will affect the selection of the switch points.

Finally, recall that the heuristic that we used for the simplified problem was based on the *relaxed planning graph*. Unfortunately, this approach cannot be easily adapted to complicated non-linear behaviors. In effect, computing the bounds of the state variables after executing some non-linear behavior is not only computationally expensive, but also, it cannot be determined in general if any arbitrary value inside those bounds can be reached (this was the case in the linear effects case).

In this thesis we will try to optimize the forward search by pruning options that cannot be feasible in order to avoid wasting time on them. We will do so by using approxima-

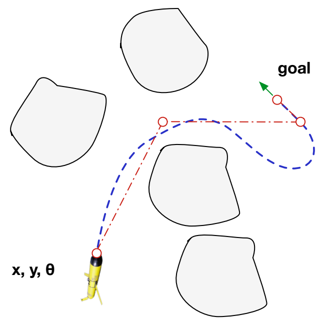

Figure 3: An outer approximation for the Dubin's smooth path given by an infinite curvature (straight-line) path

tions to the non-linear behaviors that are faster to compute. We want to use outer approximations that guarantee that if no solution can be found using the approximation, no solution exists for the more detailed model either. However, some returned solutions using the approximated model may not be feasible when checked with the more detailed model.

Figure 3 shows an example of an approximation for the Dubin's dynamics model that the *navigate* activity follows. The smooth path resulting from Dubin's equations (shown in blue) can be approximated by a piece-wise straight-line path (in red). The linear approximation is faster to compute and it can be shown that if no linear path exists from the starting conditions to the goal conditions, no Dubin's path exists either. On the other hand, there could exist an obstacle-free linear path between the start and the goal poses but no Dubin's equivalent path (curvature constraints may be violated, for example). We can use this approximation to calculate lower and upper bounds on duration (using minimum and maximum velocities) and to prune search states leading to infeasible outcomes.

References

Cambon, S.; Alami, R.; and Gravot, F. 2009. A Hybrid Approach to Intricate Motion, Manipulation and Task Planning. *The International Journal of Robotics Research* 28(1):104–126.

Coles, A.; Fox, M.; Long, D.; and Smith, A. 2008. Planning with Problems Requiring Temporal Coordination. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*, 892–897.

Coles, A. J.; Coles, A.; Fox, M.; and Long, D. 2009. Temporal Planning in Domains with Linear Processes. In *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*, 1671–1676.

Coles, A. J.; Coles, A.; Fox, M.; and Long, D. 2010. Forward-Chaining Partial-Order Planning. In *Proceedings of the 20th International Conference on Automated Planning and Scheduling, ICAPS 2010, Toronto, Ontario, Canada, May 12-16, 2010*, 42–49.

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

Dai, H.; Valenzuela, A.; and Tedrake, R. 2014. Whole-body motion planning with centroidal dynamics and full kinematics. In *Humanoid Robots (Humanoids), 2014 14th IEEE-RAS International Conference on*, 295–302. IEEE.

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

Fallon, M.; Kuindersma, S.; Karumanchi, S.; and Tedrake, R. 2015. An Architecture for Online Affordance-based Perception and Whole-body Planning. *Journal of Field* ....

Fernandez, E.; Karpas, E.; and Williams, B. C. 2015. Mixed Discrete-Continuous Heuristic Generative Planning Based on Flow Tubes. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, 1565–1572.

Fox, M., and Long, D. 2011. PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. *CoRR* abs/1106.4561.

Garrett, C. R.; Lozano-Pérez, T.; and Kaelbling, L. P. 2014. FFRob: An efficient heuristic for task and motion planning. *lis.csail.mit.edu*.

Hoffmann, J., and Nebel, B. 2001. The FF planning system: Fast plan generation through heuristic search. *Journal of Artificial Intelligence Research* 253–302.

Hoffmann, J. 2003. The Metric-FF Planning System: Translating "Ignoring Delete Lists" to Numeric State Variables. *J Artif Intell Res(JAIR)* 20:291–341.

Hofmann, A., and Williams, B. C. 2006. Robust Execution of Temporally Flexible Plans for Bipedal Walking Devices. In *Proceedings of the Sixteenth International Conference on Automated Planning and Scheduling, ICAPS 2006, Cumbria, UK, June 6-10, 2006*, 386–389.

Léauté, T., and Williams, B. C. 2005. Coordinating Agile Systems through the Model-based Execution of Temporal Plans. In *Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference, July 9-13, 2005, Pittsburgh, Pennsylvania, USA*, 114–120.

Li, H. X., and Williams, B. C. 2008. Generative Planning for Hybrid Systems Based on Flow Tubes. In *Proceedings of the Eighteenth International Conference on Automated Planning and Scheduling, ICAPS 2008, Sydney, Australia, September 14-18, 2008*, 206–213.

Li, H., and Williams, B. C. 2011. Hybrid Planning with Temporally Extended Goals for Sustainable Ocean Observing. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2011, San Francisco, California, USA, August 7-11, 2011*.

Li, H. X. 2010. *Kongming: a generative planner for hybrid systems with temporally extended goals*. Ph.D. Dissertation, Massachusetts Institute of Technology.

Long, D., and Fox, M. 2003. Exploiting a Graphplan Framework in Temporal Planning. In *Proceedings of the Thirteenth International Conference on Automated Planning and Scheduling (ICAPS 2003), June 9-13, 2003, Trento, Italy*, 52–61.

Srivastava, S.; Fang, E.; Riano, L.; and Chitnis, R. 2014. Combined task and motion planning through an extensible planner-independent interface layer. ... *(ICRA)*.

Toussaint, M. 2015. Logic-Geometric Programming: An Optimization-Based Approach to Combined Task and Motion Planning. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, 1930–1936.