Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence (AAAI-17)

# Mixed Discrete-Continuous Planning with Convex Optimization

**Enrique Fernández-González**
MIT CSAIL
Cambridge, MA
efernan@mit.edu

**Erez Karpas**
Technion
Haifa, Israel
karpase@technion.ac.il

**Brian Williams**
MIT CSAIL
Cambridge, MA
williams@mit.edu

## Abstract

Robots operating in the real world must be able to handle both discrete and continuous change. Many robot behaviors can be controlled through numeric parameters (called *control variables*), which affect the rate of the continuous change. Previous approaches capable of reasoning efficiently with control variables impose severe restrictions that limit the expressivity of the problems that can be solved. A broad class of robotic applications require, for example, convex quadratic constraints on state variables and control variables that are jointly constrained and that affect multiple state variables simultaneously. However, extensions to prior approaches are not straightforward, since these characteristics are non-linear and hard to scale. We introduce cqScotty, a heuristic forward search planner that solves these problems efficiently. While naive formulations of consistency checks are not convex and do not scale, cqScotty uses an efficient convex formulation, in the form of a Second Order Cone Program (SOCP), that is very fast to solve. We demonstrate the scalability of our approach on three new realistic domains.

## Introduction

In order to achieve their missions, autonomous robots need to reason with both discrete and continuous change (e.g. dynamics and temporal constraints). Over the last few years, planners like Kongming (Li and Williams 2008), COLIN (Coles et al. 2012) or Scotty (Fernandez, Karpas, and Williams 2015) have emerged to address this problem. Of these, heuristic forward search (HFS) approaches, such as COLIN's and Scotty's, have shown to perform best. An important characteristic of these two is that they do not require time discretization. This is essential for efficiently planning for typical scenarios with long horizons and activities with multiple time scales.

Scotty significantly increased the expressivity of the problems that could be modeled by introducing continuous control variables in this continuous time heuristic forward search setting. However, there is still a large class of problems that cannot be modeled with this approach. A broad range of robotic applications need to be able to specify constraints between the control variables (that often represent velocities) in order to limit, for example, the magnitude of the velocity of the robot, instead of its principal components independently $(v_x, v_y)$. Moreover, these velocities, apart from affecting the position of the robot, need to be able to also affect other magnitudes simultaneously (e.g. to model velocity dependent battery drain). Some applications also require robots to be subject to complex state constraints, such as staying within a maximum distance of another robot or object, or being inside ellipsoidal regions. Finally, preferred plans often minimize magnitudes other than the typical plan length, such as distances traveled or fuel consumption. None of these requirements can be modeled with Scotty or other HFS planners. Satisfying these requirements while maintaining the same performance of Scotty is challenging due to the non-linearity of these characteristics.

In this work we achieve this goal through three insights. First, due to recent advances in optimization, a restricted form of quadratically constrained programs, called Second Order Cone Programs (SOCPs), can be solved efficiently for real world problems. Second, nearly all of the requirements outlined above can be encoded with cone constraints, with the exception of a non-convex term resulting from the product of control variables and time. Third, an encoding trick allows us to eliminate this non-convex term, resulting in a SOCP encoding that is very fast to solve and that our planner repeatedly uses to test the consistency of partial plans. Our new SOCP encoding allows us to impose upper bound constraints on the norm of vectors of control variables (e.g. $v_x^2 + v_y^2 \leq v_{max}^2$), enforce convex quadratic state constraints (such as being inside ellipsoidal regions or ensuring a maximum distance between objects) and use the same control variables in as many simultaneous effects as needed. We do this without resorting to time, state or control variable discretization, which allows us to maintain the high performance of continuous time HFS planners. Since SOCPs are convex programs and can be solved very efficiently, we keep the same properties that Scotty had (optimality and completeness of consistency checks) at the cost of a slight increase in computation.

### Motivating example

Our example scenario (Figure 1) consists of an underwater ocean science mission carried out by a Remotely Operated Vehicle (ROV), an Autonomous Underwater Vehicle (AUV) and a ship. Initially the ship is transporting both the AUV and the ROV to the science site. The AUV needs to

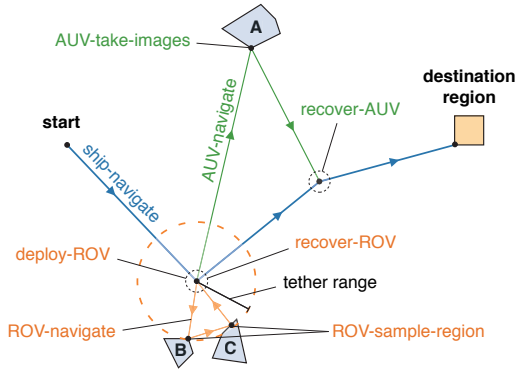


Figure 1: Ocean Science Mission Scenario

take images at region A, while the ROV needs to take samples in regions B and C. All three vehicles need to reach the destination region at the end, can navigate on their own and have their own $v_x, v_y$ velocities. The velocities can be freely chosen, but their norms are upper-bound constrained ($v_{x_i}^2 + v_{y_i}^2 \leq v_{max_i}^2 \quad \forall i \in \{\text{ship}, \text{ROV}, \text{AUV}\}$). Whenever the ROV is deployed, the ship needs to remain still at the deployment location until the ROV is recovered again. Moreover, the ROV is tethered to the ship, and therefore it can only move within a circle centered at the ship with radius the tether length ($(x_R - x_S)^2 + (y_R - y_S)^2 \leq R_{tether}^2$). Both the AUV and the ROV can be picked up when at most 2 meters away from the ship. The AUV can navigate on its own once deployed, but it has a finite battery that limits how long it can travel on its own ($\dot{b}_{\text{AUV}} = -k \cdot \|\mathbf{v}_{\text{AUV}}\|$).

Figure 1 shows a valid plan for this mission that presents interesting characteristics. First, by stationing the ship and deploying the ROV at the appropriate location, both sampling regions can be visited without violating the tether range constraint, which saves time and fuel. Second, the AUV battery is not large enough to reach the destination region on its own. Therefore, the ship meets the AUV at a non fixed nor discretized intermediate location, picks it up and transports it to the destination region. Finally, the battery decrease function is also interesting, as it depends on the norm of the velocity of the AUV.

## Related Work

Other than Scotty (Fernandez, Karpas, and Williams 2015), Kongming is perhaps the planner that is closest to ours in the features it supports. Kongming (Li and Williams 2008) was the first planner to support control variables affecting continuous effects. Although powerful and expressive, Kongming does not scale well in practice due to its fixed time discretization and a solving method based on a continuous analog of the Planning Graph and a MILP encoding.

COLIN (Coles et al. 2012) was the first planner capable of reasoning with continuous effects in a heuristic forward search setting. In order to do that COLIN uses linear programs to check the consistency of partial plans. However, COLIN's rates of change are fixed and cannot be chosen as continuous controllable parameters and, therefore, it cannot model the velocities in the example scenario. Scotty combines the strengths of Kongming that allows more expressive continuous actions through flow tubes, and an approach based on COLIN's method of heuristic forward search and linear programming for consistency checking. In doing so, Scotty supports continuous control variables without resorting to discretization. However, Scotty cannot model the example scenario since its control variables cannot affect multiple effects simultaneously (e.g. velocity affecting both position and battery) and can only be limited by their independent fixed lower and upper bounds (i.e. the principal axes of the velocity can be constrained, but not its norm). Recently, POCPORN (Savas et al. 2016) formalized the notion of continuous control parameters as an additional element of actions. Contrary to Scotty and our planner, POPCORN's control parameters can only be used in discrete numeric effects and not as rates of change in continuous effects, which we need for problems like the motivating scenario. Other recent approaches have also considered continuous control parameters, but are limited to discrete time and change (Pantke, Edelkamp, and Herzog 2016).

Recently, the planning community has made advances in supporting more expressive planning problems as those modeled in PDDL+ (Fox and Long 2006). Some of these planners use a discretize and validate approach (Della Penna et al. 2009; Piotrowski et al. 2016) while others have used SMT based techniques with good results (Bogomolov et al. 2015; Cashmore et al. 2016). Most PDDL+ planners are, in general, more expressive than our planner in several ways, like their support of processes, events, must-happen semantics and continuous change that is non-linear in time. However, their semantics do not represent robot dynamics very well, since they do not support control variables and are unable to model the motivating example shown earlier.

The robotics community, on the other hand, has recently shown interest in combined Task and Motion Planning (TAMP) (Srivastava et al. 2014; Lozano-Pérez and Kaelbling 2014). Like our planner, some interesting approaches combine optimization with discrete search (Toussaint 2015; Hadfield-Menell et al. 2016). These planners excel at highly constrained manipulation problems but do not scale to long planning horizons due to time discretization.

## Problem Statement

Our mixed discrete-continuous planning problem is given by the tuple $\langle P, V, I, G, CV, CC, A, O \rangle$, where:

- $P$ is a set of propositions, which can be true or false.
- $V$ is a set of real valued state variables.
- $I$ is the initial state, which is a complete assignment to $P$ and $V$.
- $G$ is the goal, which consists of a partial assignment to $P$ as well as conditions on the real state variables, and both of which need to hold at the end of the plan.
- $CV$ is a set of *control variables*. Each $c \in CV$ is a real valued parameter with a lower bound $c_l$ and an upper bound $c_u$.
- $CC$ is a set of global constraints on the control variables.

- $A$ is the set of durative activities as described in PDDL2.1 (Fox and Long 2003).
- $O$ is the plan optimization objective.

Our activities support two types of continuous effects that depend on control variables: *controllable linear time-varying effects* and *resource-constrained norm effects*.

**Definition** (*Controllable linear time-varying continuous effect, CLTE*)**.** A *CLTE*, is given by a set of pairs $\langle k_i, c_i \rangle$, where each $c_i$ is a control variable and $k_i \in \mathbb{R}$ is its associated constant. A *CLTE* changes a state variable linearly in time with a rate of change that is a linear combination of its control variables. The change on a continuous state variable $x \in V$ up to time $t$ due to an ongoing *CLTE* that started at time 0 is given by

$$\Delta x_{CLTE}(t) = \int_0^t \sum_{\langle k_i, c_i \rangle \in CLTE} k_i \cdot c_i(\tau) d\tau \qquad (1)$$

**Definition** (*Resource-constrained norm effects*)**.** Resource-constrained norm effects are continuous linear time effects that decrease the value of a constrained resource with a rate of change proportional to the $l_2$-norm or square of the $l_2$-norm of a set of control variables.

In the example scenario, there are six control variables corresponding to the $v_x, v_y$ velocities of each vehicle. These are constrained so that the norm of each vehicle velocity is limited. The vehicles change their positions thanks to the navigate activities of each vehicle having two *CLTE* effects, each operating on the $x$ and $y$ axes. Finally, the AUV navigate activity also has a *resource-constrained norm effect* that makes its battery drain at a rate proportional to the norm of its velocity ($\dot{b}_{AUV}(t) = -k \cdot \|\mathbf{v_{AUV}}\|$ with $k > 0$). The optimization objective is given as a linear combination of the plan makespan, the values of the state variables at the end and the sumproducts of the norms (or squared norms) of the control variable vectors and the times they were active for. In the example, one of the terms is $\int \|\mathbf{v_{SHIP}}\| \, dt$, which minimizes the distance traveled by the ship. A similar term involving the square of the norm is also possible.

## Approach

Our planner consists of two components that are tightly integrated: a discrete search and a convex optimization model. We use a heuristic forward search algorithm, Enforced Hill-Climbing (Hoffmann and Nebel 2001), to find a sequence of starts and ends of activities that are analogous to the start and end snap actions used by many temporal planners (Long and Fox 2003; Coles et al. 2008). The convex optimization serves two purposes. First, it is used to check the consistency of the partial plans that the search visits. This entails checking the feasibility of the convex program, that tries to find an assignment of execution times and trajectories of the state and control variables that is consistent with the constraints imposed by the activities in the partial plan. Second, upon finding a sequence of activities that satisfies the goals, this convex program finds the optimal trajectories of state and control variables for that sequence in order to return the plan.

**Heuristic**

The heuristic is based on the Temporal Relaxed Planning Graph (Do and Kambhampati 2003; Coles et al. 2008) and is very similar to that of COLIN and Scotty except for some minor differences. As in the case of many other planners, the heuristic value is the number of start or end events to reach the goal in the relaxed graph. In the spirit of MetricFF (Hoffmann 2003), each state variable has its associated minimum and maximum bounds in each fact layer of the planning graph. The state variable bounds for the first layer are computed by solving the convex optimization program twice per state variable (to minimize and maximize each variable). Like Scotty, we use the minimum and maximum bounds of the control variables to expand the bounds of the state variables in the next layer due to active continuous effects. New in our heuristic is that we need to take into account the convex quadratic conditions that neither COLIN nor Scotty support. In order to skip uneventful layers, COLIN's heuristic computes the future time when a currently unmet state condition will be satisfied as a result of an active continuous effect modifying a state variable. This is straightforward in the case of linear inequality constraints, but hard to compute for general convex quadratic constraints. Our solution is to use linear over-approximations to the quadratic constraints in the heuristic, which are then handled efficiently as in COLIN's heuristic. If the user specifies the convex quadratic conditions as primitives, such as ellipsoidal regions, our planner computes the linear over-approximations automatically (e.g. axis aligned bounding boxes). Otherwise, the user can specify the approximations directly by providing lists of linear inequalities. These linear over-approximations constitute valid relaxations since they ensure that actions can always be executed earlier than they would be if the actual quadratic constraints were used.

The other difference in our heuristic is that *resource-constrained norm effects* also need to be considered. These effects only reduce the availability of constrained resources and their application can only make activities infeasible (and never new activities possible). Therefore, in the spirit of delete relaxations, these effects are ignored in the heuristic. The optimization-based consistency check, that accurately computes these effects, rejects states that become infeasible due to this and backtracking takes the search through a different route. However, there is room to improve the current heuristic and make it aware of these interactions.

**Consistency checking with convex optimization**

To check the consistency of partial plans, we use Second Order Cone Programs, which allow us to impose convex quadratic constraints. While SOCPs are harder to solve than LPs, there are efficient, polynomial-time algorithms that can easily solve problems with thousands of variables. Since SOCPs are convex problems, our consistency check is complete and only infeasible partial states are pruned from the search. While our planner is not optimal due to the greedy search, this also guarantees that the returned solution is optimal with respect to the chosen sequence of activities.

Search nodes define partial plans that are given by an ordered sequence of start and end events (start and end of activities). The execution times of these events, $t_j$, and the val-

ues of the state variables at these times, $x_k(t_j)$, are decision variables. We call the period of time between consecutive events a *stage* and denote by $\Delta t_j = t_{j+1} - t_j$ the duration of the j-th stage.

The constraints in the model describe: a) the temporal constraints, b) the at start, at end and over all numeric constraints on the state variables imposed by the activities, and c) the continuous change in the state variables due to continuous effects. As in other temporal planners (Coles et al. 2012), the temporal constraints describe that consecutive events need to be at least $\varepsilon$-separated ($\Delta t_j \geq \varepsilon$) and that the minimum and maximum activity durations need to be respected.

In order to simplify the model, we restrict the control variables to only change values at start or end events and therefore, stay constant throughout each stage. This results in state variables having piecewise linear trajectories, with the switch points being the events. This does not affect completeness, since our linear time dynamics and the absence of obstacles or curvature constraints ensure that any problem solvable with an arbitrarily changing state trajectory is also solvable with a piecewise linear one. Additionally, because the numeric conditions are convex and the trajectories are linear, we only need to enforce the over all conditions at the events (i.e., switch points) to ensure that these are satisfied at all times, without requiring time discretization.

We proceed to describe now how we encode the constraints related to the continuous effects. Continuous effects are cumulative. The continuous change in a state variable between consecutive events ($j$ and $j+1$) is the sum of the changes due to each effect:

$$x_k(t_{j+1}) = x_k(t_j) + \sum_{eff_i \in E_{x_k}(j)} \Delta x_{k eff_i}(j) \qquad (2)$$

where $E_{x_k}(j)$ is the set of all active continuous effects operating on $x_k$ at stage $j$. Each $\Delta x_{k eff_i}(j)$ is a decision variable describing the change due to a continuous effect during stage $j$.

The change due to a *CLTE* during stage $j$ is given by

$$\Delta x_{kCLTE}(j) = \left( \sum_i k_i \cdot c_{var_i}(j) \right) \cdot (t_{j+1} - t_j) \qquad (3)$$

where $c_{var_i}(j)$ is the constant value that the i-th control variable takes during stage $j$. Given that the event times are decision variables, the previous equation is non-linear and non-convex if the control variables are also decision variables. To overcome this problem, Scotty used alternative interval equations that introduced severe limitations to the problems that could be solved and that we address here. To do that, we define the decision variable $c_{i\Delta t_j} = c_i \cdot \Delta t_j$ for every control variable being used in each stage $j$. $c_{i\Delta t_j}$ is subject to the linear inequality constraints

$$c_{il} \cdot \Delta t_j \leq c_{i\Delta t_j} \leq c_{iu} \cdot \Delta t_j \qquad (4)$$

where $c_{il}$ and $c_{iu}$ are the lower and upper bounds of $c_i$. The non-linear CLTE equation (3) can then be rewritten as the linear equation

$$\Delta x_{kCLTE}(j) = \sum_i k_i \cdot c_{i\Delta t_j} \qquad (5)$$

In effect, instead of asking the solver to pick a value for each control variable, we ask the solver to pick the times of the events and the product of the values of the control variables and the elapsed time between consecutive events. The key idea that makes this encoding work is that for most purposes we only need the values of the products of control variables and time intervals instead of the actual values of the control variables. A key advantage of this formulation is that, contrary to Scotty, control variables can now be used in as many continuous effects as needed, since the $c_{i\Delta t}$ decision variables can be reused in other equations, and the values will be consistent with each other. Moreover, we can now impose constraints on the control variables. For example, we can impose that two control variables, $c_1, c_2$ always satisfy $c_1 + c_2 \leq 5$. While the constraint cannot be encoded directly since $c_1$ and $c_2$ are not explicit decision variables, we can multiply the equation by $\Delta t_j$ and impose the condition $c_{1\Delta t_j} + c_{2\Delta t_j} \leq 5 \cdot \Delta t_j$ at every stage. This can be generalized to any linear constraint, since $\Delta t_j$ is always positive by construction.

Similarly, we can also impose maximum $l_2$-norm constraints on sets of control variables. In many robotic applications it is useful to represent the velocity of a robot with its principal components $(v_x, v_y)$ but still limit the total velocity by some fixed amount. This can be expressed with the constraint $||\mathbf{c}|| \leq v_{max}$, where $\mathbf{c} = [v_x, v_y]^T$ is a vector of control variables. To encode this constraint, we again multiply the whole equation by $\Delta t_j$ to obtain the second order cone constraint

$$\|\mathbf{c}\| \cdot \Delta t_j = \left\|\mathbf{c_{\Delta t_j}}\right\| \leq v_{max} \cdot \Delta t_j \qquad (6)$$

We can also represent *resource constrained norm effects* using cone constraints. In this work we focus on *linear norm effects* (LNE) and *linear squared norm effects* (LSNE). The change due to the first effect on a constrained resource (state variable $r_k$) is given by

$$\Delta r_{kLNE}(j) = -k_{LNE} \cdot \|\mathbf{c}\| \cdot \Delta t_j, \quad k_{LNE} \geq 0 \qquad (7)$$

where $k_{LNE}$ is a constant and $\mathbf{c}$ is a vector of control variables. Equation (7) is not convex and cannot be encoded directly. However, we can transform (7) into a cone constraint by defining the bound variable $b$ and rewriting the equation as

$$||\mathbf{c_{\Delta t_j}}|| \leq b, \quad b \geq 0 \qquad (8)$$

$$\Delta r_{kLNE}(j) = -k_{LNE} \cdot b \qquad (9)$$

Equations (7) and (9) do not represent the same, since $b$ is simply an upper-bound on $||\mathbf{c_{\Delta t_j}}||$ . This is the reason why we restrict these effects to constrained resources. In general, the bound $b$ will not be tight and the computed value for resource $r_k$ may not be accurate. However, these equations can perfectly model that a constrained resource decreases with the norm of a control variable vector and the bound will become tight to ensure that a resource never dips below some threshold. This is useful to model, for example, how battery decreases in a vehicle as a function of the speed it is traveling at, regardless of the $x, y$ direction. Unfortunately, we cannot use this resource constrained norm effects to impose

that a certain resource is below some level, since the artificial bound $b$ could take any arbitrary large value to satisfy the constraint trivially without changing the actual value of the norm of the control variable vector. The second resource constrained norm effect that we support is the *linear squared norm effect* (LSNE), which is subject to the same limitations as the LNE effect, but in which monotonic decrease of the resource variable is proportional to the squared norm.

$$\Delta r_{kLNSE}(j) = -k_{LNSE} \cdot \|\mathbf{c}\|^2 \cdot \Delta t_j, \quad k_{LNSE} \geq 0 \quad (10)$$

Again, equation (10) is non-convex, but we can use the same principle as before to represent it as a SOCP constraint. Since $\mathbf{c_{\Delta t_j}} = \mathbf{c} \cdot \Delta t_j$, we can write

$$||\mathbf{c}||^2 \cdot \Delta t_j = \frac{\mathbf{c_{\Delta t_j}}^T \mathbf{c_{\Delta t_j}}}{\Delta t_j} \leq b, \quad b \geq 0 \qquad (11)$$

where $b$ is, again, an auxiliary positive bound variable. Finally, equation (11) can be rewritten as

$$\mathbf{c_{\Delta t_j}}^T \mathbf{c_{\Delta t_j}} \leq b \cdot \Delta t_j \qquad (12)$$

$$\Delta r_{kLNSE}(j) = -k_{LNSE} \cdot b \qquad (13)$$

which is a *rotated cone constraint*, a valid type of convex SOCP constraint, since $b$ and $\Delta t_j$ are positive.

The remaining constraints in our model correspond to the conditions imposed by the activities. The numeric state conditions can be linear inequalities or convex quadratic constraints. The *at start* and *at end* conditions of an activity are imposed at the start and end events of an activity. For *over all* conditions, we further require that all intermediate events between the start and end (the switch points) also satisfy the condition. As previously mentioned and due to the convexity properties, the full piecewise linear trajectory will be contained in the convex set.

While checking the consistency of partial plans, there may be activities that have started, but not ended yet. We use the same $t_{now}$ trick that COLIN uses to ensure that the partial plan is feasible. An event at $t_{now}$ is placed after all the other events in the partial plan and before all the future end events of activities that have started but not ended yet. This helps identify partial plans that are not feasible because the temporal deadlines have been violated.

Finally, once a plan has been found, our planner solves the optimization problem with an optimization objective. This objective is specified as part of the problem and can be a linear combination of the state variables at the end of the plan and the sumproducts of the norms (or square norms) of control variable vectors and stage durations. This can be useful for finding a plan that minimizes actuation control. An important advantage of this approach is not committing early to the times or values of the state variables at the switch points. Since we do not require discretization of state, control variables or time, we prevent early bad choices that could lead to infeasible plans later on and we leave as much flexibility as possible for the solver to make the best choice according to the optimization objective (Toussaint 2015).

## Experimental Results

To showcase the new capabilities of our planner and to show that our optimization framework is fast and scalable, we present three new expressive domains and benchmark our planner against them. Since no other planner can solve these domains, we also provide a simplified, linear version of some of these domains that we use to compare our planner to Scotty and POPCORN. We compare against these planners since they support control variables, which are essential for these domains. Although Kongming supports the same capabilities as Scotty, we do not compare against Kongming since it was shown to perform significantly worse than Scotty for the same features (Fernandez, Karpas, and Williams 2015).

**The AUV domain:** In this domain an AUV needs to visit and take samples at multiple regions. This domain is similar to POPCORN's 2D-AUV-Power domain (Savas et al. 2016), that is based on prior Kongming and Scotty domains. There are two main differences between POPCORN's domain and ours. First, since POPCORN does not support controllable rates of change, the effects of the *glide* action are modeled as discrete numeric displacements on the $x, y$ variables at the end of the action, whereas we model the motion as a continuous effect that takes place while the action is being executed. Moreover, since POPCORN only supports linear constraints, its authors model the maximum power of the vehicle as a simple linear constraint on the displacements at the end of the action ($3d_x + 4d_y \leq 60$), while we can use the new features of our SOCP model to limit the magnitude of the velocity ($v_x^2 + v_y^2 \leq v_{max}^2$). In the simplified linear version of this domain we place no constraints on the $v_x, v_y$ velocities other than their simple independent bounds.

**The ROV domain:** This domain is based on the motivating example presented at the beginning of this paper but without an AUV. As in the motivating example, the ROV needs to take samples in multiple regions and end, together with the ship, in the destination region. Note that our planner decides where to station the ship while the ROV is taking samples, and that good selections of that position may allow the ROV to visit several regions without having to be recovered by the ship first. The optimization objective for this domain minimizes a linear combination of the plan makespan and the distance traveled by the ship. In the simplified linear version of this domain we remove the velocity norm constraints. Furthermore, the maximum distance constraints, which are modeled with the convex quadratic constraints of being inside a circle, are replaced by a simpler linear polygonal over approximation of such a circle (an octagon in this case). Since we cannot model the distance traveled by the ship without quadratic constraints, the simplified version only optimizes the makespan of the plan.

**The air refueling domain:** In this final domain, an autonomous Unmanned Aerial Vehicle (UAV) needs to take pictures of several regions before landing at the destination location. Since the UAV has limited fuel, it needs to refuel in-air from a tanker plane. While refueling, both planes can keep moving but they need to stay within a maximum distance. The UAV fuel decreases as a function of the distance traveled and the square of the velocity ($\dot{f} = -k_1 v - k_2 v^2$). As in the ROV domain, the objective for this domain is to minimize a linear combination of the plan makespan and the distance traveled by the tanker plane. In instances 11-

| | AUV | | | | | ROV | | | | | Air Refueling | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | t | L | S | N | T | t | L | S | N | T | t | L | S | N | T |
| **01** | 0.53 | 4 | 4 | 17 | 2 | 1.16 | 16 | 19 | 153 | 3 | 0.97 | 8 | 11 | 111 | 3 |
| **02** | 0.55 | 8 | 10 | 41 | 2 | 1.74 | 20 | 35 | 281 | 4 | 1.68 | 12 | 19 | 191 | 5 |
| **03** | 0.66 | 12 | 18 | 73 | 2 | 2.79 | 24 | 57 | 457 | 4 | 2.44 | 16 | 30 | 301 | 6 |
| **04** | 0.84 | 16 | 28 | 113 | 2 | 4.34 | 36 | 79 | 633 | 5 | 5.78 | 18 | 74 | 669 | 7 |
| **05** | 0.94 | 20 | 40 | 161 | 2 | 7.26 | 40 | 119 | 953 | 6 | 3.82 | 20 | 45 | 433 | 7 |
| **06** | 0.89 | 20 | 40 | 161 | 2 | 10.71 | 52 | 157 | 1225 | 7 | 5.68 | 24 | 60 | 583 | 8 |
| **07** | 1.18 | 24 | 54 | 217 | 2 | 16.38 | 56 | 213 | 1665 | 9 | 11.18 | 28 | 98 | 927 | 11 |
| **08** | 1.24 | 24 | 54 | 217 | 2 | 19.75 | 68 | 236 | 1822 | 9 | 10.93 | 32 | 96 | 916 | 11 |
| **09** | 1.46 | 28 | 70 | 281 | 2 | 32.24 | 72 | 338 | 2607 | 11 | 12.03 | 32 | 105 | 997 | 11 |
| **10** | 1.49 | 28 | 70 | 281 | 2 | 35.09 | 84 | 350 | 2659 | 12 | 17.14 | 38 | 115 | 1124 | 14 |
| **11** | 1.87 | 32 | 88 | 353 | 3 | 40.10 | 88 | 392 | 2993 | 12 | 2.14 | 10 | 18 | 289 | 5 |
| **12** | 1.86 | 32 | 88 | 353 | 3 | 56.63 | 100 | 451 | 3410 | 15 | 10.77 | 14 | 64 | 1025 | 10 |
| **13** | 2.29 | 36 | 108 | 433 | 3 | 52.82 | 96 | 412 | 3094 | 15 | 15.43 | 16 | 80 | 1281 | 11 |
| **14** | 2.30 | 36 | 108 | 433 | 3 | 68.63 | 108 | 497 | 3683 | 17 | 21.33 | 18 | 98 | 1569 | 13 |
| **15** | 2.85 | 40 | 130 | 521 | 3 | 87.77 | 120 | 586 | 4315 | 18 | 49.61 | 22 | 165 | 2581 | 19 |
| **16** | 2.75 | 40 | 130 | 521 | 3 | 95.51 | 124 | 630 | 4659 | 18 | 60.67 | 24 | 191 | 2907 | 20 |
| **17** | 3.57 | 44 | 154 | 617 | 3 | 119.82 | 136 | 712 | 5301 | 20 | 78.57 | 26 | 222 | 3373 | 23 |
| **18** | 4.48 | 48 | 180 | 721 | 4 | 151.10 | 140 | 885 | 6531 | 21 | 658.30 | 32 | 1147 | 17003 | 38 |
| **19** | 5.04 | 52 | 208 | 833 | 4 | 161.05 | 144 | 923 | 6802 | 21 | 563.87 | 34 | 906 | 13972 | 39 |
| **20** | 6.15 | 56 | 238 | 953 | 4 | 218.39 | 156 | 1181 | 8658 | 22 | 249.25 | 36 | 419 | 6210 | 39 |

(a) Benchmarking results

| | AUV-simplified | | | | | | | | | ROV-simplified | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | cqScotty | | | | | Scotty | | POPCORN | | cqScotty | | | | | Scotty | | POPCORN | |
| | t | L | S | N | T | t | L | t | L | t | L | S | N | T | t | L | t | L |
| **01** | 0.54 | 4 | 4 | 17 | 2 | 0.48 | 4 | 0.05 | 4 | 0.86 | 16 | 19 | 153 | 2 | 0.89 | 16 | 0.57 | 16 |
| **02** | 0.55 | 8 | 10 | 41 | 1 | 0.49 | 8 | 0.15 | 8 | 1.25 | 20 | 35 | 281 | 2 | 1.48 | 20 | 1.95 | 20 |
| **03** | 0.61 | 12 | 18 | 73 | 1 | 0.56 | 12 | 0.30 | 12 | 1.74 | 24 | 57 | 457 | 2 | 2.33 | 24 | 3.59 | 24 |
| **04** | 0.72 | 16 | 28 | 113 | 1 | 0.64 | 16 | 0.58 | 16 | 2.54 | 36 | 79 | 633 | 3 | 5.58 | 36 | 6.53 | 36 |
| **05** | 0.80 | 20 | 40 | 161 | 1 | 0.77 | 20 | 0.94 | 20 | 4.07 | 40 | 119 | 953 | 3 | 7.54 | 40 | 16.34 | 40 |
| **06** | 0.83 | 20 | 40 | 161 | 1 | 0.74 | 20 | 0.91 | 20 | 5.46 | 52 | 156 | 1214 | 3 | 12.79 | 52 | 24.77 | 52 |
| **07** | 0.92 | 24 | 54 | 217 | 1 | 0.92 | 24 | 1.50 | 24 | 7.85 | 56 | 213 | 1621 | 4 | 16.73 | 56 | 49.84 | 56 |
| **08** | 0.99 | 24 | 54 | 217 | 1 | 0.94 | 24 | 1.52 | 24 | 9.06 | 68 | 233 | 1753 | 4 | 55.40 | 84 | 77.21 | 68 |
| **09** | 1.15 | 28 | 70 | 281 | 1 | 1.24 | 28 | 2.22 | 28 | 14.36 | 72 | 328 | 2478 | 5 | 86.75 | 96 | 107.27 | 72 |
| **10** | 1.13 | 28 | 70 | 281 | 1 | 1.14 | 28 | 2.23 | 28 | 15.53 | 84 | 345 | 2565 | 5 | 119.23 | 100 | 150.69 | 84 |
| **11** | 1.43 | 32 | 88 | 353 | 1 | 1.44 | 32 | 3.29 | 32 | 18.34 | 88 | 396 | 2952 | 5 | 96.59 | 96 | 175.83 | 88 |
| **12** | 1.42 | 32 | 88 | 353 | 1 | 1.43 | 32 | 3.26 | 32 | 24.79 | 100 | 450 | 3335 | 6 | 142.33 | 108 | 242.64 | 92 |
| **13** | 1.49 | 34 | 92 | 369 | 2 | 1.53 | 34 | 3.89 | 34 | 23.36 | 96 | 411 | 3023 | 6 | 126.13 | 104 | 278.46 | 96 |
| **14** | 1.48 | 34 | 92 | 369 | 2 | 1.57 | 34 | 3.90 | 34 | 30.35 | 108 | 495 | 3639 | 7 | 180.93 | 116 | 343.66 | 108 |
| **15** | 1.80 | 38 | 114 | 457 | 2 | 2.12 | 38 | 4.88 | 38 | 38.62 | 120 | 578 | 4261 | 7 | 254.07 | 128 | 460.56 | 112 |
| **16** | 1.75 | 38 | 114 | 457 | 2 | 2.19 | 38 | 5.60 | 38 | 45.26 | 124 | 660 | 4826 | 8 | 158.47 | 108 | 525.74 | 116 |
| **17** | 2.22 | 42 | 138 | 553 | 2 | 2.50 | 42 | 6.58 | 42 | 56.49 | 136 | 743 | 5469 | 8 | 204.34 | 120 | 617.31 | 128 |
| **18** | 2.81 | 46 | 164 | 657 | 2 | 3.53 | 46 | 9.91 | 46 | 68.51 | 140 | 890 | 6484 | 8 | 271.64 | 132 | 783.90 | 140 |
| **19** | 3.31 | 50 | 192 | 769 | 2 | 4.12 | 50 | 13.01 | 50 | 73.40 | 144 | 926 | 6737 | 9 | 391.01 | 152 | 834.64 | 136 |
| **20** | 3.93 | 54 | 222 | 889 | 2 | 5.30 | 54 | 17.18 | 54 | 98.11 | 156 | 1177 | 8479 | 9 | 500.96 | 164 | 1028.76 | 148 |

(b) Benchmarking results for simplified domains

Table 1: Results of Empirical Evaluation. **t**: Planning time in seconds; **L**: Plan length; **S**: Number of nodes expanded; **N**: Number of optimization problems solved; **T**: Mean solving time for each optimization problem in milliseconds.

20 there is an additional UAV, and only one UAV can refuel at a time. This domain is challenging for several reasons. First, the planner needs to consider the simultaneous trajectories of multiple vehicles and also their fuel levels. Second, and more importantly, while our optimization model supports the *resource-constrained norm effects* (such as the fuel decrease depending on the norm or squared norm of the velocity), the heuristic does not consider these effects directly. Therefore, our planner only chooses the refuel activities by backtracking when reaching other regions becomes infeasible due to having insufficient fuel.

We do not present a simplified version of this domain because neither POPCORN nor Scotty would be able to solve a linear alternative. The reason is that the *refuel* activity requires continuous effects since both the tanker and the UAV have to be flying simultaneously while staying close to each other. POPCORN cannot model this since the numeric change can only be applied at the beginning or end of an activity and not continuously in time. This domain also requires that the fuel of each UAVs decreases as a function of the magnitude of their velocities, which neither POPCORN nor Scotty can model.

**Results:** We benchmarked our planner on an Intel Core i7-3770 3.40 GHz with the Gurobi 7.0.1 solver. As seen in column T of Table 1(a), our convex optimization model, a key contribution of our work, is solved very quickly. The mean optimization time per problem grows for more complicated instances since these have more state variables and require more activities, which results in far more decision variables and constraints at later stages of the search. However, most optimization problems are solved in less than 10 ms in average for small to medium domain instances and in less than 50 ms for larger ones. This is important since large domain instances require solving tens of thousands of optimization problems, as seen in the table. This kind of performance would not be possible if we used a straightforward non-convex non-linear optimization model with a general purpose non-linear optimizer.

Table 1(b) shows the results for the simplified domains. These results let us answer the question of what is the performance penalty of switching from a linear program formulation to a SOCP one. As seen in the table, the difference between the mean optimization time for the linear problems solved in the simplified domains and the SOCP ones from the full domains is very small for the simpler instances and significant for more complicated instances. However, this difference is always well within an order of magnitude. Moreover, we should highlight that the linearized version is significantly simpler, as it not only linearizes some constraints (such as the ROV tether range ones) but also drops many other constraints, like the norm ones or the ones required to minimize the traveled distances. We can conclude that using SOCPs for consistency checking is not only practical, but that the performance tradeoff is well worth it considering the added expressivity that they provide. Finally, we compare our planner in these simplified domains to Scotty and POPCORN. Since our optimization model is significantly more complex than theirs, even in these linear domains, given the extra variables and constraints that we require, we expected that our planner would be slower. However, Table 1 shows that this is not the case and our planner performs significantly better. We hypothesize that this is due to the superior performance of the Gurobi solver compared to the solvers used by Scotty (CPLEX 12.4) and POPCORN (lpsolve 5.5). Additionally, POPCORN's test were kindly run on a slower i5-M540 2.53GHz processor by its authors, since they could not share the planner with us.

## Conclusion

We have presented a new planner capable of solving more expressive hybrid problems than the current state of the art while maintaining the same level of performance. By using an efficient convex optimization model based on SOCPs, we can solve problems requiring convex quadratic constraints on control and state variables and control variables that affect multiple continuous effects. While this work ad-

vances the expressivity of the hybrid problems that can be solved with heuristic forward search techniques, our planner presents some limitations that we would like to address. First, our planner only supports linear dynamics and practical constraints such as maximum curvature cannot be handled in our model. Moreover, we currently do not support obstacles, since their associated non-convex constraints cannot be represented in our framework. Finally, while returned plans are optimal conditioned on the chosen sequence of activities, our search does not explicitly take the optimization objective into account, and we cannot make any guarantees with respect to the optimality of the chosen sequence. All of these are active areas of research that we are currently pursuing.

## Acknowledgments

We thank the SUTD-MIT Graduate Fellows Program for providing financial support to Enrique Fernández during this work. We would also like to thank Emre Savas for running the POPCORN benchmarks.

## References

Bogomolov, S.; Magazzeni, D.; Minopoli, S.; and Wehrle, M. 2015. PDDL+ Planning with Hybrid Automata: Foundations of Translating Must Behavior. In *Proceedings of the Twenty-Fifth International Conference on Automated Planning and Scheduling, ICAPS 2015*, 42–46.

Cashmore, M.; Fox, M.; Long, D.; and Magazzeni, D. 2016. A Compilation of the Full PDDL+ Language into SMT. In *Proceedings of the Twenty-Sixth International Conference on Automated Planning and Scheduling, ICAPS 2016*, 79–87.

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*, 892–897.

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.

Della Penna, G.; Magazzeni, D.; Mercorio, F.; and Intrigila, B. 2009. UPMurphi: A Tool for Universal Planning on PDDL+ Problems. In *Proceedings of the 19th International Conference on Automated Planning and Scheduling, ICAPS 2009*.

Do, M. B., and Kambhampati, S. 2003. Sapa: A Multi-objective Metric Temporal Planner. *J Artif Intell Res(JAIR)* 20:155–194.

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*, 1565–1572.

Fox, M., and Long, D. 2003. PDDL2. 1: An Extension to PDDL for Expressing Temporal Planning Domains. *J Artif Intell Res(JAIR)*.

Fox, M., and Long, D. 2006. Modelling Mixed Discrete-Continuous Domains for Planning. *J Artif Intell Res(JAIR)*.

Hadfield-Menell, D.; Lin, C.; Chitnis, R.; Russell, S.; and Abbeel, P. 2016. Sequential Quadratic Programming for Task Plan Optimization . In *In the proceedings of the 29th IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2016)*.

Hoffmann, J., and Nebel, B. 2001. The FF Planning System: Fast Plan Generation Through Heuristic Search. *J Artif Intell Res(JAIR)* 14:253–302.

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

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*, 206–213.

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*, 52–61.

Lozano-Pérez, T., and Kaelbling, L. P. 2014. A constraint-based method for solving sequential manipulation planning problems. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, 3684–3691. IEEE.

Pantke, F.; Edelkamp, S.; and Herzog, O. 2016. Symbolic discrete-time planning with continuous numeric action parameters for agent-controlled processes. *Mechatronics* 34:38–62.

Piotrowski, W. M.; Fox, M.; Long, D.; Magazzeni, D.; and Mercorio, F. 2016. Heuristic Planning for PDDL+ Domains. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016*, 3213–3219.

Savas, E.; Fox, M.; Long, D.; and Magazzeni, D. 2016. Planning Using Actions with Control Parameters. In *ECAI 2016 - 22nd European Conference on Artificial Intelligence*, 1185–1193.

Srivastava, S.; Fang, E.; Riano, L.; Chitnis, R.; Russell, S. J.; and Abbeel, P. 2014. Combined task and motion planning through an extensible planner-independent interface layer. In *2014 IEEE International Conference on Robotics and Automation, ICRA 2014*, 639–646. IEEE.

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*, 1930–1936.