Hybrid Planning with Temporally Extended Goals for Sustainable Ocean Observing

Hui Li
Boeing Research & Technology, The Boeing Company
hui.li3@boeing.com

Brian Williams
CSAIL, Massachusetts Institute of Technology
williams@csail.mit.edu

Abstract

A challenge to modeling and monitoring the health of the ocean environment is that it is largely under sensed and difficult to sense remotely. Autonomous underwater vehicles (AUVs) can improve observability, for example of algal bloom regions, ocean acidification, and ocean circulation. This AUV paradigm, however, requires robust operation that is cost effective and responsive to the environment. To achieve low cost we generate operational sequences automatically from science goals, and achieve robustness by reasoning about the discrete and continuous effects of actions.

We introduce Kongming2, a generative planner for hybrid systems with temporally extended goals (TEGs) and temporally flexible actions. It takes as input high-level goals and outputs trajectories and actions of the hybrid system, for example an AUV. Kongming2 makes two major extensions to Kongming1: planning for TEGs, and planning with temporally flexible actions. We demonstrated a proof of concept of the planner in the Atlantic ocean on Odyssey IV, an AUV designed and built by the MIT AUV Lab at Sea Grant.

Introduction

The control of real-world autonomous systems generally involves a mix of discrete and continuous actions. For example, a typical AUV mission involves discrete actions like `get GPS` and `set sonar`, and continuous actions that involve continuous dynamics of the vehicle, such as `descend` and `ascend`. Currently most real-world unmanned missions are controlled on the low level by engaging a set of behaviors, and on the high level by a mission script consisting of pre-defined action sequences that are specified by the human operator. Having to design the action sequence for every new mission puts a significant cognitive burden on the human operator. Some robotic systems (Thompson et al 2010) address the problem by using a path planner loosely coupled with a timeline planner. While this two-stage approach usually requires less computational power, a unified hybrid planning representation is more optimal and is the focus of this paper.

Significant progress has been made in planning with actions that not only have discrete but also continuous effects. PDDL2.1 (Fox and Long 2003) offers a standard language for encoding planning domains with temporal and metric dimensions. Zeno (Penberthy and Weld 1994) is among the early work that explores planning with continuous processes. LPSAT (Wolfman and Weld 1999) solves resource planning problems where real-valued quantities are involved. TM-LPSAT (Shin and Davis 2005) extends LPSAT to handle durative actions and linear continuous change. Sapa (Do and Kambhampati 2001) plans with durative actions and metric resource constraints, using a heuristic forward chaining approach. COLIN (Coles et al. 2009) is a forward-chaining temporal planner that handles actions with linear continuous effects using linear programming.

However, none of the planners above can plan for temporally extended goals (TEGs). TEGs refer to properties that must hold over intermediate and/or final states of a plan. It is of great interest to be able to plan for not only the final goal state but a series of goal states along various stages of the execution of a plan. Often times we would like to specify goals to be achieved in a sequential order, for example, sea gliders to identify algal bloom regions first, and then propeller-driven AUVs to take samples in such regions. Every so often we also want to specify goals to be achieved in parallel, for example, multiple sea gliders to monitor different parts of a region at the same time for correlation studies.

There has been work on TEG planning. PDDL3.0 (Gerevini and Long 2005) includes state trajectory constraints in the PDDL language to express TEGs. TLPlan (Bacchus and Kabanza 1996) and TALPlan (Kvarnström and Doherty 2000) treat TEGs as temporal domain control knowledge. (Biere et al. 1999; Latvala et al. 2004; Mattmuller and Rintanen 2007) extend the planning as satisfiability approach (Kautz and Selman 1992). However, none of them can plan with both discrete and continuous actions.

We introduce Kongming2, a planner that plans for TEGs with both discrete and continuous actions. It makes two extensions to Kongming1 (Li and Williams 2008). First, goals can be temporally extended instead of a single final goal. Second, actions are durative and can have arbitrary length. In August 2009, we demonstrated a proof of concept of Kongming2 in the Atlantic ocean on Odyssey IV, an AUV designed and built by the MIT AUV Lab at Sea Grant.

Hybrid Planning with TEGs

We first define the input and output of the planner.

Problem Statement

Kongming2 takes as input a set of *state variables*, *control variables*, *initial conditions*, *temporally extended goals (TEGs)*, *hybrid durative action types*, *external constraints*, and an *objective function*. It outputs an optimal sequence of actions and state trajectory from the given action types that achieve the TEGs without violating any of the constraints. We next define each item.

The state variables, $\mathbf{s}$, represent the continuous and discrete state of the system. $\mathbf{s} = \langle \mathbf{x}, \mathbf{p} \rangle$, where $\mathbf{x} \in \Re^n$ is a vector of real-valued variables, and $\mathbf{p}$ is a vector of propositional variables.

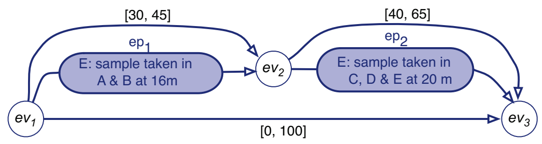
The control variables, $\mathbf{u}$, represent the control input to the system. $\mathbf{u} \in \Re^m$ is a vector of real-valued control variables, $\mathbf{u} = \langle u_1, u_2, \ldots, u_m \rangle$.

The initial conditions, $\mathcal{I}$, specify constraints on the initial value of the state variables $\mathbf{s}$. $\mathcal{I} = (\mathbf{Ax} \leq b) \wedge_i l_i$, where $\mathbf{Ax} \leq b$ is a system of linear inequalities over $\mathbf{x}$, and each l_i is a positive or negative literal for $p_i \in \mathbf{p}$.

We define the set of temporally extended goals (TEGs) as a qualitative state plan (QSP), which specifies the desired state trajectory of the system under control. It consists of the following.

- A finite set of *events*, each representing a point in time.
- A finite set of *episodes* that specify the desired state evolution over time. Each episode has a start event, an end event, and a state constraint over the state variables.
- A finite set of *temporal constraints* that specify the lower and upper temporal bounds between two events.

Fig. 1 shows a QSP example of an ocean observing mission. It consists of three events, shown as nodes, two episodes, specifying the goal states, and temporal bounds between events. Episode ep_1 requires samples taken in region A and B at depth 16m at the end of the episode. It should take between 30 and 45 time units to achieve this episode.



[30, 45]
ep1
E: sample taken in A & B at 16m
ev2
[40, 65]
ep2
E: sample taken in C, D & E at 20 m
ev1
[0, 100]
ev3

Figure 1: QSP example. Events are ev_1, ev_2 and ev_3. Episodes are ep_1 and ep_2. Temporal bounds are specified between events.

Hybrid durative action types specify conditions, discrete effects, dynamics and duration of the action types that a system can perform. Conditions are requirements that must be satisfied in order for an action to take place. Discrete effects capture the changed propositional truth values as a result of the action. Both conditions and discrete effects are specified in different stages of the action: *start*, *overall*, and *end*. Dynamics describe the state transition and the actuation limits of a system when performing an action. Actions have flexible durations. Each duration is a real-valued variable bounded by a lower bound and an upper bound. We show two action examples in Fig. 2 for the underwater domain.

```
(:durative-action descend
 :condition ((start (rudder))
             (overall (rudder)))
 :discrete-effect ()
 :dynamics (u ∈ [u_lb , u_ub]
          x(t_i+1) = Ax(t_i) + Bu(t_i))
 :duration (d ∈ [0, +∞)))
```

```
(:durative-action take-sample
 :parameter (?r - region)
 :condition ((start((x,y,z)∈?r)(gulper)(¬sample))
             (overall((x,y,z)∈?r)(gulper))
             (end((x,y,z)∈?r)))
 :discrete-effect (end(sample))
 :dynamics ()
 :duration (d ∈ [2, 5]))
```

Figure 2: Action `descend` requires rudder to be on at the start and during the action. It has no discrete effect. Its dynamics are governed by the bounds and equation. Its duration is unlimited. Similar with action `take-sample`. The main differences are its dynamics are unspecified and its discrete effect is there is sample at the end.

External constraints, impose external requirements on the state of the system under control. They are in conjunctive normal form (CNF), and are specified by a conjunction of clauses, each of which is a disjunction of linear inequalities over the state variables $\mathbf{x}$.

The objective function is a linear or quadratic function over real-valued state variables $\mathbf{x}$, control variables $\mathbf{u}$, and time. Typical objective functions are to minimize battery use, distance traveled, or mission completion time.

The output is an optimal hybrid plan, consisting of a sequence of durative actions, control inputs, and the resulting states. The sequence of control inputs $\{\mathbf{u}^*(t_1), \mathbf{u}^*(t_2), \ldots, \mathbf{u}^*(t_{N-1})\}$ is an optimal assignment to the control variables $\mathbf{u}$, and $\mathbf{u}^*(t_i)$ is the assignment to $\mathbf{u}$ at time t_i. The sequence of states $\{\mathbf{s}^*(t_1), \mathbf{s}^*(t_2), \ldots, \mathbf{s}^*(t_N)\}$ is an optimal assignment to the continuous and discrete state variables $\mathbf{s} = \langle \mathbf{x}, \mathbf{p} \rangle$, and $\mathbf{s}^*(t_i)$ is the assignment to $\mathbf{s}$ at time t_i. Part of the output is shown in the example in Fig. 3, namely, the sequence of continuous states and actions.

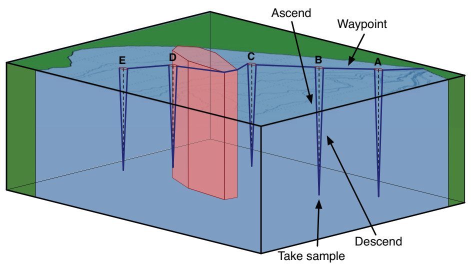


Figure 3: In a segment of the ocean. We show the state trajectory and action sequence of the planner output. The solid blue line represents the trajectory. Continuous and discrete actions take place along the trajectory.

Next we introduce the approach, starting with a review of Kongming1 (Li and Williams 2008). We then introduce the two extensions in Kongming2 for TEGs and for temporally flexible actions.

Review of Kongming1

Kongming1 employs a compact representation of all hybrid plans, called a *hybrid flow graph*, which combines the strengths of a Planning Graph for discrete actions and Flow Tubes for continuous actions. It then encodes the hybrid flow graph as a mixed logic linear program (MLLP), which it solves using CPLEX, an off-the-shelf solver.

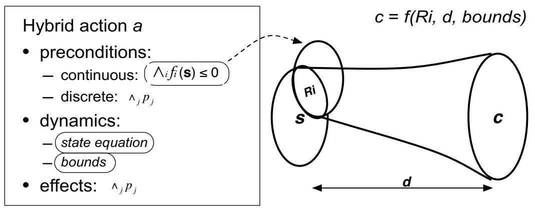


Figure 4: Mapping from a continuous action to a flow tube. The intersection of the current state with the continuous precondition of the action forms the initial region of the flow tube R_i. c is the end region of the flow tube, which is a function of R_i, duration d and dynamic bounds.

Flow tubes are used to as an abstraction of an infinite number of trajectories of continuous actions. A flow tube is a collection of all trajectories going from an initial region to the end region, which is a function of the initial region, duration and dynamic bounds. Each continuous action can be modeled as a flow tube, as shown in Fig. 4. Two flow tubes are connected when the continuous precondition of the second flow tube intersects with the end region of the first flow tube, as shown in Fig. 5.

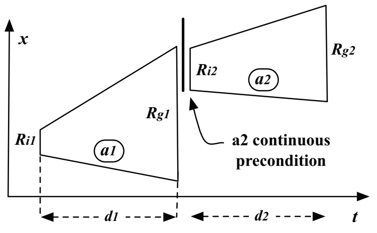


Figure 5: Connecting one flow tube to another.

Hybrid Flow Graphs build upon Planning Graphs introduced by Graphplan (Blum and Furst 1997), augmented with flow tubes, to represent all valid plans and collections of feasible state trajectories. A *hybrid flow graph* is a directed, leveled graph that alternates between fact levels and action levels. A fact level contains two types of fact nodes: continuous fact nodes and discrete fact nodes. An action level contains hybrid action nodes, and each node can be either continuous or discrete. When continuous dynamics are not involved, a hybrid action is discrete. Otherwise it is continuous and is denoted by a flow tube, connecting an initial region and an end region over a duration.

Similar to a Planning Graph, in a Hybrid Flow Graph, two action nodes, either continuous or discrete, at a given action level are marked as mutually exclusive (mutex) if no valid plan could contain both. Likewise, two facts at a given fact level are mutex if no valid plan could satisfy both. Different from Graphplan are the exclusion rules that involve continuous facts or continuous actions.

Kongming1 then encodes the Hybrid Flow Graph as an MLLP, based on a series of encoding rules. The rules are analogous to those for Blackbox (Kautz and Selman 1999), which encodes a Planning Graph as a SAT problem. The main difference from Blackbox is the introduction of continuous variables and constraints.

Extension 1: Temporally Extended Goals

We extend Kongming1 to plan for TEGs. This enables us to specify requirements along various stages of the execution of a plan, rather than only in the final stage. As defined in the problem statement, the TEGs are in the form of a QSP. Our approach is based on the novel idea that each episode in the QSP can be achieved through enforcing the conditions of a "pseudo" action. We create a hybrid durative action for each episode in a QSP. The conditions of the action include all the state constraints of the episode. In order to make the effects of the action true, the conditions have to be satisfied, which enforces the state constraints to be achieved.

An example of a QSP is shown on the left of Fig. 6. We observe that episodes in a QSP are similar to hybrid durative actions in that they are durative, and have constraints specified at the start, over the duration, and at the end. An encoding that represents episodes as durative actions may be used. The main difference is that episodes establish conditions to be satisfied, while actions establish effects to be produced. In addition, a QSP specifies which episodes precede and follow other episodes, while there is no specific description on how durative action types are ordered.

Fig. 6 shows the reformulation of one episode in a QSP into an action. Let us focus on episode ep_2 in the QSP on the left of the figure. The right of the figure shows the action created for episode ep_2. The duration of the action is bounded by the simple temporal constraint on the start and end events of the episode, if there is any. The conditions of the action include the state constraints of the episode, and the fact that all predecessors of the episode have been achieved. The discrete effect of the action at the end is the fact that this episode has been achieved. Dynamics are unspecified.

We explain the reformulation procedure as follows.

- A hybrid durative action type a is created for every episode ep in the QSP.
- If there exists a simple temporal constraint in the QSP on the time between the start event and the end event of the episode, then the duration of a is bounded by the lower and upper bounds in the temporal constraint. Otherwise, the duration of a is unbounded, specified as $[0, +\infty)$.
- The state constraints of ep are encoded as the conditions of a. More specifically, the *start* constraints of ep are encoded as the *start* conditions of a; the *overall* constraints

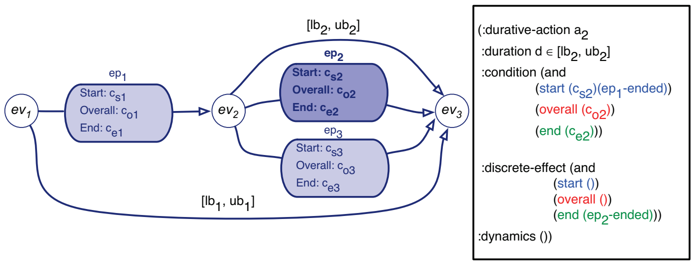


Figure 6: Hybrid durative action type a_2 is created for episode ep_2. ep_1-ended and ep_2-ended are literals, representing respectively the fact that ep_1 and ep_2 are achieved. c_{s2}, c_{o2} and c_{e2} are the constraints at different times of ep_2.

of ep are encoded as the *overall* conditions of a; and the *end* constraints of ep are encoded as the *end* conditions.

- Another *start* condition of a requires that all the predecessors of ep have been achieved. Naturally, an episode cannot be started if any of its predecessors have not completed.
- The only discrete effect of a is that ep has completed, specified by a literal ep-$ended$. This effect is the *end* effect, because naturally, before the end of an episode, it cannot be considered completed. The *start* and *overall* effects of a are empty.
- The dynamics of a are unspecified, because the actions are "pseudo" actions that do not produce any actual effects. They are created to enforce the state constraints of episodes in the QSP.
- The final goal state requires that all episodes in the QSP are achieved, specified by $\wedge_i\ ep_i$-$ended, \forall ep_i \in qsp$.

This reformulation procedure can be adopted by PDDL planners. If one chooses to represent the TEGs as a QSP, then the procedure remains unchanged, except that the step about dynamics is removed. If an MITL-type (Alur et al 1996) representation is used, (Li 2010) shows that any non-disjunctive MITL can be expressed in QSP.

Extension 2: Flexible Action Durations

The second extension to Kongming1 is allowing actions to be durative and have flexible durations. Our approach builds upon LPGP (Long and Fox 2002), which encodes each discrete durative action as a *start*, an *end* and one or more intermediate actions. Kongming2 combines the LPGP encoding with flow tubes to reformulate hybrid durative actions as hybrid atomic actions. Because the LPGP encoding only applies to actions with discrete conditions and effects, the main challenge lies in incorporating the flow tubes of the hybrid actions in the encoding.

Fig. 7 shows a simple flow tube representation of a hybrid durative action with flexible duration. The specification of a hybrid durative action is listed on the right. Its flow tube representation is on the left. The *start*, *overall* and *end* conditions are checked at the start, in the middle and at the end of the flow tube. The *start*, *overall* and *end* discrete effects are added at the start, in the middle and at the end of the flow tube.

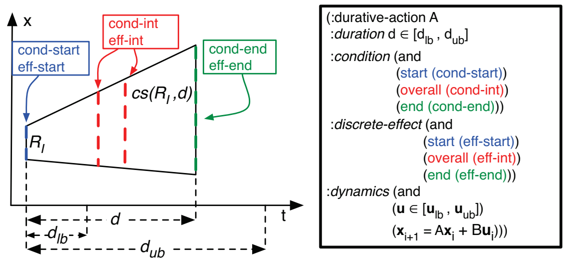


Figure 7: The specification of a hybrid durative action is listed on the right. Its flow tube representation is on the left. *cond-start* represents the *start* conditions; *cond-int* represents the *overall* conditions; *cond-end* represents the *end* conditions. *eff-start* represents the *start* discrete effects; likewise for *eff-int* and *eff-end*. The duration of the action is flexible. The *start*, *overall* and *end* conditions are checked at the start, in the middle and at the end of the flow tube. The *start*, *overall* and *end* discrete effects are added at the start, in the middle and at the end of the flow tube.

We encode each hybrid durative action as a set of atomic actions, by "slicing" the flow tube into multiple pieces. All flow tube slices have the same duration, Δt. We do not adopt the continuous timeline in LPGP because we need to keep the constraints in the mixed logic quadratic program encoding linear. When both the control input and duration are variables, non-linear constraints are introduced. This is because the state equations need to be satisfied in each action level. For example, a simple first-order state equation: $x(t_i) = x(t_{i-1}) + \dot{x}(t_{i-1}) \times (t_i - t_{i-1})$. The two terms in the product, $\dot{x}(t_{i-1})$ and $(t_i - t_{i-1})$, can't both be variable. We discuss the alternative at the end of the paper.

We combine the flow tube slices with the LPGP encoding, to ensure that first, the conditions at various stages are enforced at the beginning of each flow tube slice; second, the effects at various stages are added at the end of each flow tube slice; and finally, the *start* flow tube slice is performed first, followed by one or more *intermediate* flow tube slices, and ending with the *end* flow tube slice.

As shown on the left of Fig. 8, the flow tube is divided into multiple flow tube slices, each having duration Δt. As shown on the right of Fig. 8, the flow tube slices are combined with the LPGP encoding. More specifically, the first flow tube slice corresponds to atomic action A-$start$. The last flow tube slice corresponds to atomic action A-end. The flow tube slices in the middle correspond to atomic action A-int. The duration for all the atomic actions is Δt. Moreover, the atomic actions all keep the same dynamics as in the durative action.

This encoding of a hybrid durative action not only discretizes the flow tube into slices with length of Δt each, but also uses the conditions of the action type at different stages to prune the invalid regions in state space and hence invalid trajectories in the flow tube.

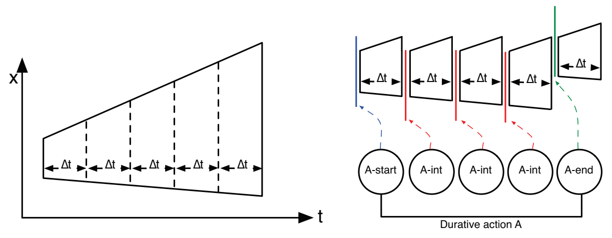


Figure 8: Kongming2 combines the flow tube slices with the LPGP encoding. On the left, the flow tube of a hybrid durative action is divided into slices, each with length Δt. On the right, the flow tube slices are combined with the LPGP encoding. The blue line represents the continuous condition of *A-start*. The red lines represent the continuous condition of *A-int*. The green line represents the continuous condition of *A-end*. The initial region of a flow tube slice is the intersection of the continuous condition of its corresponding atomic action and the end region of its previous flow tube slice.

Proof of Concept on An AUV

We ran Kongming2 on Odyssey IV (Fig. 9) in the Atlantic ocean in Cape Cod. One of the main goals of the demonstration was to show Kongming2's capability of producing optimal hybrid plans to achieve temporally extended goals. Another important goal of the demonstration was to show Kongming2's capability of generating mission scripts that run seamlessly on an AUV in a natural environment. Prior to this field demonstration, Odyssey IV and other AUVs at the MIT AUV Lab had only been controlled by mission scripts written by the engineers. Kongming2 automatically generates optimal mission scripts, and enables cost effective and robust operations.

Figure 9: Odyssey IV, an AUV from the MIT AUV Lab at Sea Grant, is $0.7 \times 1.4 \times 2.2$m, weighs 500kg dry and about 1000kg wet, and has rated depth 6000m.

As this was our first field demonstration, although Kongming2 is fully capable of planning in 3D, for safety reasons, the vehicle was kept on the surface, in order to maintain constant communication. As the vehicle was limited to operate on the surface, this reduced planning to one continuous action type, namely `GoToWaypoint`. Moreover, at the time of the demonstration, the vehicle could not perform any discrete action, so `GoToWaypoint` was the only hybrid action type available for the field demonstration.

We conducted two major tests. Due to limited space in the paper, we describe one of them. The scenario consists of reaching three waypoints before going to the pickup point. The graphical user interface used by Kongming2 for the scenario is shown in Fig. 10. As we can see, an obstacle prevents Odyssey IV from following a straight line from waypoint 2 to 3. The obstacle is imaginary in the ocean.

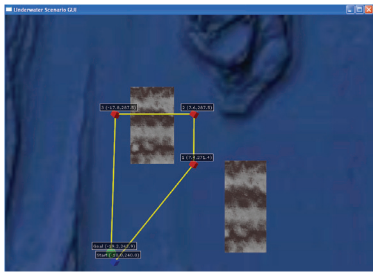


Figure 10: Display of the graphical user interface for the scenario. The mission requires Odyssey IV to reach three waypoints before going to the pickup point. There is an obstacle between Waypoint 2 and 3.

The QSP input to Kongming2 is shown in Fig. 11. Two episodes, ep_3 and ep_4, to reach the obstacle corners are added. The initial condition for Kongming2 is to be at the start point, as seen in Fig. 10. The objective function is a quadratic function to minimize the total distance traveled.

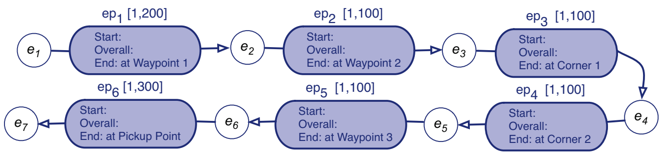


Figure 11: The QSP for the test. Each episode in the QSP specifies the state of reaching a waypoint. The temporal constraint for each episode is specified in square brackets.

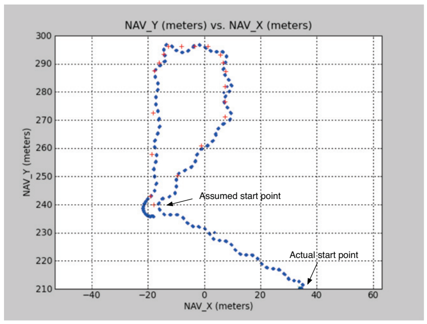


Figure 12: The plot of the actual trajectory of Odyssey IV executing the mission script generated by Kongming2. AUV's trajectory is in blue dots, and the waypoints planned by Kongming2 are in red crosses.

The mission script generated by Kongming2 consists of 22 action instances, and can be found in (Li 2010). The mission script commands Odyssey IV to go to the waypoints specified in the QSP as well as all the intermediate waypoints generated by Kongming2. The behavior-based controller of the vehicle then maneuvers it to those waypoints within specified time bounds. The trajectory of the vehicle is shown in Fig. 12, where the wavy parts were due to currents in the ocean and the fact that the controller was under-tuned. The extra leg close to the bottom of the plot occurs because at the start of the mission, the vehicle was not at the assumed start location. Hence the controller had to maneuver the vehicle to the assumed start location first, which was not part of the plan Kongming2 output.

Evaluation and Discussion

We empirically evaluated Kongming2 in both the underwater domain and the air vehicle domain. We describe two main results here. Please refer to (Li 2010) for details. First, Kongming2 scales better in terms of the number of discrete action types than in terms of the number of continuous action types. The reason is that in each action level, for one continuous action type, there can be a large number of flow tubes based on different initial regions, which then propagate to a large number of continuous facts in the next fact level, and so on. Second, the time discretization factor Δt significantly affects the computation time of Kongming2, as it decides the number of levels in the Hybrid Flow Graph.

Because in Kongming2 the control input of each action is a variable, we have to discretize time in order to keep the dynamical equations linear. This discretization of time makes the approach less scalable. An alternative is to keep time continuous and discretize control input. The Maneuver Automaton (Frazzoli 2001; Schouwenaars et al 2003; Greytak 2009) is a representation of the discretized dynamics of a vehicle, consisting of a finite collection of two types of motion primitives: trim conditions and maneuvers. Trim conditions are similar to the hybrid action types in Kongming2, except that each trim condition has a fixed control value. For vehicles whose dynamics can be approximated with a reasonable number [1] of trim conditions, Maneuver Automata may provide a more computationally efficient solution to continuous-time planning.

Conclusion

In this paper, we introduce Kongming2, a planner that plans for TEGs with both discrete and continuous actions that are temporally flexible. We focus on the two extensions to Kongming1. First, goals can be temporally extended instead of a single final goal. Second, actions are durative and can have arbitrary length. We demonstrated a proof of concept of Kongming2 on an AUV in the Atlantic ocean. It opens up possibilities of robust and cost effective AUV paradigm for ocean observing.

References

R. Alur and T. Feder and T. Henzinger 1996. The benefits of relaxing punctuality. *Journal of the ACM*.

F. Bacchus and F. Kabanza 1996. Planning for Temporally Extended Goals. In *Proceedings of AAAI*.

A. Biere and A. Cimatti and E. Clarke and Y. Zhu 1999. Symbolic Model Checking without BDDs. In *Proceedings of TACAS*.

A. Blum and M. Furst 1997. Fast planning through planning graph analysis. *Artificial Intelligence*.

A. J. Coles and A. I. Coles and M. Fox and D. Long 2009. Temporal Planning in Domains with Linear Processes. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*.

M. B. Do and S. Kambhampati 2001. Sapa: A Domain-Independent Heuristic Metric Temporal Planner. In *Proceedings of European Conference on Planning (ECP)*.

M. Fox and D. Long 2003. PDDL2.1: An Extension of PDDL for Expressing Temporal Planning Domains. *Journal of Artificial Intelligence Research*.

E. Frazzoli 2001. Robust Hybrid Control for Autonomous Vehicle Motion Planning. *PhD Thesis, MIT*.

A. Gerevini and D. Long 2005. Plan constraints and preferences in PDDL3 - the language of the fifth international planning competition. *Technical Report, Department of Electronics for Automation, University of Brescia, Italy*.

M. Greytak 2009. Integrated Motion Planning and Model Learning for Mobile Robots with Application to Marine Vehicles. *PhD Thesis, MIT*.

H. Kautz and B. Selman 1992. Planning as Satisfiability. In *Proceedings of European Conference on Artificial Intelligence (ECAI)*.

H. Kautz and B. Selman 1999. Unifying SAT-based and Graph-based Planning. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*.

J. Kvarnström and P. Doherty 2000. TALPlanner: A Temporal Logic based Forward Chaining Planner. *Annals of Mathematics Artificial Intelligence*.

T. Latvala and A. Biere and K. Heljanko and T. Junttila 2004. Simple Bounded LTL Model Checking. In *Proceedings of FMCAD*.

H. Li and B. Williams 2008. Generative Planning for Hybrid Systems based on Flow Tubes In *Proceedings of International Conference on Automated Planning and Scheduling (ICAPS)*.

H. Li 2010. Kongming: A Generative Planner for Hybrid Systems with Temporally Extended Goals. *PhD Thesis*. MIT.

D. Long and M. Fox 2002. Fast Temporal Planning in a Graphplan Framework. In *Proceedings of International Conference on Automated Planning and Scheduling (ICAPS)*.

R. Mattmuller and J. Rintanen 2007. Planning for Temporally Extended Goals as Propositional Satisfiability. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*.

J. Penberthy and D. Weld 1994. Temporal Planning with Continuous Change. In *Proceedings of AAAI*.

T. Schouwenaars and B. Mettler and E. Feron and J. How 2003. Robust Motion Planning Using a Maneuver Automaton with Built-in Uncertainties. In *Proceedings of ACC*.

J. Shin and E. Davis 2005. Processes and Continuous Change in a SAT-based Planner. *Artificial Intelligence*.

D. Thompson and S. Chien and Y. Chao and P. Li and B. Cahill and J. Levin and O. Schofield and A. Balasuriya and S. Petillo and M. Arrott and M. Meisinger 2010. Spatiotemporal path planning in strong, dynamic, uncertain currents. In *Proceedings of IEEE International Conference on Robotics and Automation (ICRA)*.

S. Wolfman and D. Weld 1999. The LPSAT Engine and Its Application to Resource Planning. In *Proceedings of International Joint Conference on Artificial Intelligence (IJCAI)*.

[1]The examples in (Frazzoli 2001; Schouwenaars et al 2003; Greytak 2009) are on the order of 10 trim conditions.