Bounded Expectations for Discrepancy Detection in Goal-Driven Autonomy

Mark A. Wilson[1], James McMahon[2], and David W. Aha[1]

[1]Navy Center for Applied Research in Artificial Intelligence; Naval Research Laboratory, Code 5514; Washington, DC

[2]Physical Acoustics Branch; Naval Research Laboratory, Code 7130; Washington, DC

Abstract

Goal-Driven Autonomy (GDA) is a model for online planning extended with dynamic goal selection. GDA has been investigated in the context of numerous abstract planning domains, and there has been recent interest in applying GDA to control unmanned vehicles. In robotic domains, certain continuous state features from sensor data must be modeled for reasoning. However, modeling these features precisely during planning and execution monitoring may be problematic, due to the inefficiency of computing exact values or sensitivity to noise. We present PHOBOS, a Hierarchical Task Network planner with *bounded expectations*, which we apply with a GDA agent in an underwater vehicle domain. Bounded expectations allow an agent to plan and detect discrepancies more efficiently and with fewer false discrepancies (i.e., detected but semantically meaningless differences from expectations during execution). We describe an initial simulation study that supports this claim.

1. Introduction

While there is a large body of work on motion and task planning for autonomous underwater vehicles (AUVs), current approaches are not designed to reason about self-selected goals, which may hinder the vehicle's ability to act without human supervision. To address this shortcoming, we are augmenting the planning processes of an AUV with *goal reasoning*: the ability to dynamically formulate, prioritize, and assign goals.. This is valuable in long duration missions in complex environments, such as the AUV domain, where the agent is likely to encounter unpredictable hazards and opportunities too complex to enumerate *a priori*. The ability to choose an appropriate goal to pursue enables an agent to select useful actions in a broader range of situations without supervision.

We will use Goal-Driven Autonomy (GDA), a model that responds to unexpected situations by formulating and reprioritizing goals (Molineaux, Klenk, and Aha 2010a), to control an AUV with goal reasoning. We posit that GDA will enable an AUV to conduct long duration, independent missions with varying objectives. GDA has previously been applied in simulated domains inspired by real-world scenarios (Molineaux et al. 2010a) and game environments (Weber, Mateas, and Jhala 2012; Jaidee, Muñoz-Avila, and Aha 2013), where it has performed well in comparison to alternative approaches such as dynamic replanning. Our AUV control task is one of the first applications of GDA in support of a hardware platform. However, in this paper we focus on challenges related to task planning and discrepancy detection in the AUV domain and leave evaluation of GDA as an AUV control technology for future work.

During plan execution, a GDA agent detects *discrepancies* (i.e., unexpected situations) by monitoring the world state and comparing it to *expectations*. Our agent is inspired by ARTUE (Molineaux et al. 2010a), which uses predicted states from its planner as expectations. We describe other methods for creating expectations in §3.

To apply GDA on an AUV for long-term missions, we must model continuous state features that cannot be adequately discretized or represented symbolically (e.g., the AUV's location). Existing task planning techniques can model these values' continuous change in a plan, enabling discrepancy detection throughout execution. However, these approaches typically use precise descriptions of continuous changes, which can require extensive domain engineering and lead to *false discrepancies* (i.e., discrepancies that do not affect task execution or goal suitability) and other undesirable consequences in uncertain domains.

ARTUE uses the SHOP2$_{\text{PDDL+}}$ planner (Molineaux, Klenk, and Aha 2010b) to model nonlinear continuous effects while employing Hierarchical Task Network (HTN) planning techniques (Erol, Hendler, and Nau 1994). HTN planners can quickly generate plans using detailed representations of complex domains. However, the PDDL+ model (Fox and Long 2006) employed in SHOP2$_{\text{PDDL+}}$ uses precise descriptions of continuous effects, which can lead to the issues described above. We instead present PHOBOS (Planner for HTN Objectives with Bounding OperatorS), a variant of SHOP (Nau et al. 1998) that abstracts state features during planning by allowing operator effects to set bounds (i.e., constraints) on continuous values. We demonstrate its utility as a planner for our GDA agent in AUV simulation studies.

In §2, we describe the GDA model and our architecture for applying it on an AUV. Related work is described in §3. We explain our novel method for planning and discrepancy detection in §4, describe our empirical study in §5, and discuss future research tasks in §6 before concluding.

2. GDA and AUV Control

2.1 The GDA Conceptual Model

GDA is a goal reasoning model for online planning in autonomous agents (Klenk, Molineaux, and Aha 2013). Figure 1 illustrates GDA as an extension of Nau's (2007) model of online planning. The GDA Controller interacts with a Planner and a State Transition System Σ, which is a tuple $\langle S, A, F, \gamma \rangle$ with states S, actions A, exogenous events F, and state transition function $\gamma: S \times (A \cup F) \rightarrow S$ that describes how an action or event transforms the environment. In stochastic or partially observable environments, the agent has only partial models of S, F, and γ. During execution, the Controller receives observations O, which are representations of S.

The Planner takes as input a planning problem $\langle M_\Sigma, o_c, g_c \rangle$, where M_Σ is a model of Σ, o_c is the current observation representing the current state s_c, and g_c is a goal from the set of all possible goals G. The Planner outputs (1) a plan p_c, which is a sequence of plan operators (i.e., actions and events) $R_c = [r_{c+1}, \ldots, r_{c+n}]$, and (2) a corresponding sequence of expectations $X_c = [x_{c+1}, \ldots x_{c+n}]$, where each $x_i \in X_c$ is the expectation that should follow when the corresponding action or event r_i in the sequence R_c takes place.

The Controller takes as input initial observation o_0, initial goal g_0, and M_Σ, and sends them to the Planner to generate plan p_0 and expectations X_0. The Controller forwards p_0's actions to Σ for execution and processes the resulting observations.

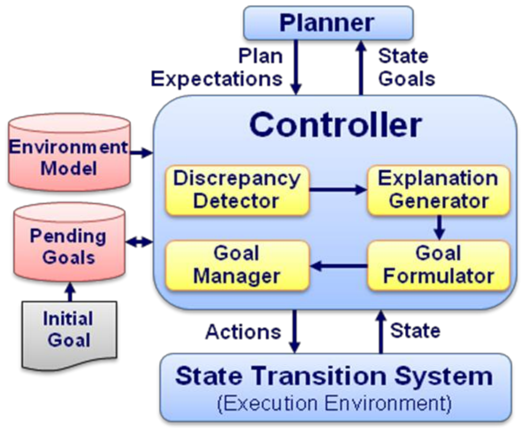

Figure 1: The GDA Conceptual Model

During plan execution, the Controller performs the following operations:

• *Discrepancy detection*: GDA detects a discrepancy d by comparing o_c with $x_t \in X_q$, which corresponds to the most recent r_t in current plan p_q.

• *Explanation generation*: Given o_c, x_t, and d, this operation hypothesizes one or more explanations of the discrepancy's cause e.

• *Goal formulation*: Resolving a discrepancy may warrant a change in the current goal(s). This operation may formulate a goal $g \in G$ in response to d, given e and s_c.

• *Goal management*: Formulating a goal may warrant its immediate focus or removal of some existing goals. Given a set of pending goals $G_P \subset G$ and new goal g, this operation may update G_P (e.g., by adding g or deleting other pending goals) and then select the next goal $g' \in G_P$ to be given to the Planner.

2.2 GDA Architecture for AUV Control

In our hybrid control architecture, the GDA Controller monitors the AUV's state and directs the AUV to perform sensing and navigation tasks, delegating them to lower-level control components. To address the challenges of motion control in dynamic environments that may be only partially known *a priori*, we employ the reactive MOOS-IvP autonomy architecture (Benjamin et al. 2010), a widely used, open source robotic control framework. MOOS is a message-passing suite with a centralized publish-subscribe model. The MOOS application IvP Helm is a behavior-based controller that sets navigation parameters to generate collision-free trajectories, using an interval programming technique that optimizes over behaviors' objective functions.

The GDA Controller executes plans by activating, deactivating, and changing parameters of IvP Helm behaviors. (While IvP Helm can alter behaviors reactively, it cannot deliberate about what goal the vehicle should pursue, which is the focus of GDA.) Figure 2 depicts our

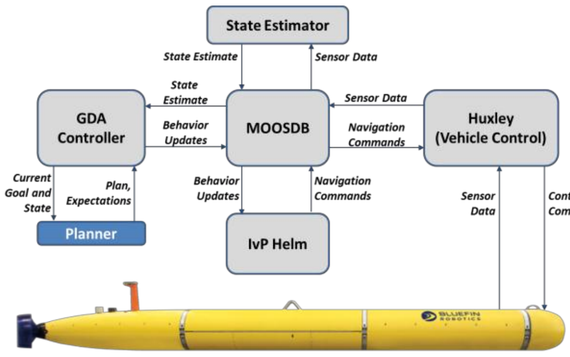

Figure 2: Our GDA agent architecture for controlling an AUV with MOOS-IvP

agent architecture, which includes Huxley, the low-level control software provided by the AUV's manufacturer, Bluefin Robotics.

3. Related Work

Several architectures have been developed to control AUVs using AI techniques. Orca (Turner 1995) is a context sensitive architecture that applies schemas to create sequences of actions for the AUV to execute. It uses an event handler to react to unexpected states, and an *agenda manager* to change the agent's active goal in response (similarly to GDA's Goal Manager). However, unlike our GDA agent, Orca assumes externally-specified goals; it does not formulate new goals.

COLA2 (Palomeras et al. 2012) is an AUV control architecture that uses reinforcement learning techniques to learn and execute motion primitives, Petri nets to model behavior-like structures for plan execution, and a STRIPS-like planner for mission planning. It replans when expectations are violated, but does not perform goal reasoning.

T-REX (Rajan, Py, and Barreiro 2012) uses constraint-based planning to guide robots, primarily AUVs, using multiple reactors that collaborate to produce a plan. Although T-REX recognizes plan expectation violations and can apply response strategies at any affected reactor, it does not address goal formulation or management, which is the focus of GDA. Although we present planning extensions employing constraints, we do so in the context of HTN planning, not constraint-based planning. Finally, unlike our GDA agent, T-REX uses multiple reactors that operate on the same plan timeline, which requires synchronization and reactor dependency graphs. Py, Rajan, and McGann (2010) argue that monolithic planning such as we employ may reduce responsiveness, which we leave as a future research topic to investigate.

Automated planning models have been developed for uncertain domains. The Probabilistic PDDL (PPDDL) model (Younes and Littman 2004) provides probability distributions over action effects, allowing an agent to use Markov Decision Processes in planning. However, it assumes that actions are instantaneous and does not model continuous change (such as AUV motion) probabilistically. Concurrent Probabilistic Temporal Planning (Mausam and Weld 2008) can represent probabilistic durations for actions, but, like PPDDL, does not model probability distributions over continuous change.

The challenge of modeling uncertain motion during task planning for AUVs has been the focus of some research. The PANDORA project employs PDDL planning over Probabilistic Roadmap (PRM) models of the environment (Cashmore et al. 2013), which provide an abstraction for use in planning. However, due to the abstract nature of PRMs, the agent performs discrepancy detection only during sensing actions; unlike our agent, it does not detect discrepancies during motion actions. Plaku and McMahon (2013) also model the AUV environment using PRMs, but employ LTL to express task-level requirements. Because their framework is not yet adapted to dynamic replanning, they do not address the problem of discrepancy detection.

Many goal reasoning agents employ expectations. LGDA (Jaidee, Muñoz-Avila, and Aha 2011) learns probabilistic state expectations that are stored in a case base; during execution, the most likely expected state is used for discrepancy detection. Jaidee et al. (2013) use an expectation that relevant state features will monotonically increase. Although their domain model specifies relevant state features, the constraint itself is fixed. Cox et al. (2012) use A-distance on symbolic state representations to detect discrepancies in MIDCA. This technique requires training on normal sequences of states. Unlike these approaches, our agent uses deterministic expectation models and does not have fixed constraints or require training.

Execution monitoring has received substantial attention, particularly in robotics research. We list a small sample of such work here; see (Petersson 2005) for one survey. At the sensor level, Gat et al. (1990) define envelopes of expected sensor values, including mathematical functions on physical sensors, based on motion plans for a Mars rover. At the task level, Fichtner, Großmann, and Thielscher (2003) use extensions to the fluent calculus to represent temporal and uncertain information for a robot in a dynamic environment, but their representation is focused on symbolic knowledge, not continuous-valued domains. The SKEMon process (Bouguerra, Karlsson, & Saffiotti 2008) uses semantic knowledge about a robot's domain to infer implicit expectations as results of actions, which can be tested probabilistically or otherwise. Although semantically rich, this technique requires semantic knowledge and additional processing that may be more suitable for terrestrial robots than the limited perception and processing power of AUVs.

4. Planning and Discrepancy Detection

In many domains, a GDA Controller will need to reason about continuous values. For instance, deciding whether to pursue a goal of mapping a region of the ocean floor may depend on predicting the vehicle's position and battery health in the future. Concrete values may also be necessary because *a priori* discretization is not feasible (e.g., segmentation of an ocean into polygonal regions suitable to a variety of tasks). Therefore, these values must be modeled in M_Σ.

Existing task-planning techniques, such as PDDL2.1's continuous durative actions (Fox and Long 2003) and PDDL+'s process-event model, describe continuous change to state features. This knowledge allows the agent to detect discrepancies at any point during execution. However, these techniques often require exact models of continuous change, which can create several difficulties for planning and discrepancy detection in the AUV domain.

First, in stochastic or partially observable domains, precise modeling can lead to false discrepancies, which require extra computation and may be detrimental to agent performance. For instance, while executing a motion action, ocean currents may cause an AUV to move off its projected course. Reactive motion controllers can adjust the vehicle's controls and correct this deviation without intervention from the GDA Controller. However, the Discrepancy Detector may incorrectly treat this deviation as a discrepancy requiring reevaluation of the agent's goals. Employing a threshold during discrepancy detection is a common approach to such challenges, but a threshold value may not generalize to all continuous features in a domain.

Second, precise domain modeling can be burdensome. Automated task-planning techniques (e.g., durative actions or processes) may require extensive knowledge engineering to describe complex actions or processes.

Third, precise modeling causes redundant computation. The planner may need to solve complex process constraints to determine how long an action will take and to compute intermediate states for discrepancy detection. During plan execution on an AUV, the navigation and motion controllers will perform lower-level computations to guide the vehicle on the same path. These computations are redundant and need not both be executed.

PHOBOS and our agent's Discrepancy Detector will address these challenges by accepting bounds on state values as part of M_Σ and applying these bounds during planning and discrepancy detection.

4.1 Planning with PHOBOS

ARTUE, the inspiration for our agent, uses SHOP2$_{\text{PDDL+}}$ to plan with nonlinear models of continuous processes. In the PHOBOS planner, we revise this model to exclude processes but include *bounding effects* in plan operators. Bounding effects specify constant bounds on state values or *derived values*, i.e., mathematical functions on state values.

Syntactically, a bounding effect is an expression of the form *(set v (range lb ub))*, where v is a continuous fluent or derived value, and lb and ub are numeric values or variables which were unified in the operator's precondition, representing the value's lower and upper bounds, respectively. For example, *(set (speed) (range ?l ?u))* creates a bound on the fluent *speed*, where *?l* and *?u* were unified in the precondition.

Rather than projecting new states with exact continuous values, PHOBOS uses bounding effects to project an expectation that includes the bounds specified by M_Σ. A ***bounded expectation*** is a tuple $X = \langle C, V, B, K \rangle$, where C is the set of true facts, V is the set of exact fluent values, B is the set of bounded continuous fluent values, and K is the set of bounded values derived from fluent values. Each value $b \in B$ is a tuple $\langle b_{id}, b_l, b_u \rangle$, where b_{id} is an identifier for the value, which is constrained to lie within the given bounds b_l, b_u. Each $k \in K$ is similarly constrained but contains a mathematical function in place of an identifier.

When projecting a new expectation X_n using a plan operator, PHOBOS applies each bounding effect by placing the bounded value, with the values of its bounds as computed in the precondition, in the appropriate element of X_n. If the effect constrains a state value, the value and its bounds are placed in B. If it constrains a derived value, the function for computing it from observed states and its bounds are placed in K. C and V are projected in the usual manner, using positive and negative fact effects and instantaneous fluent change effects.

When testing preconditions of operators, PHOBOS considers a value v_1 to be less than a bounded value v_2 iff the upper bound or precise value of v_1 is less than the lower bound of v_2 (similarly for greater-than). Arithmetic expressions are not applicable on a bounded value, but are applicable on the upper and lower bounds of a bounded value *?v*, which are expressed *(upper-bound ?v)* and *(lower-bound ?v)*.

Table 1 shows the action definition that represents the start of an AUV survey maneuver, simplified for presentation. The action preconditions define the boundaries of the region in which the vehicle will operate, and the effects constrain the vehicle's motion, creating an expectation defining a bounding box around the survey area and the vehicle's current position. This expectation will not be violated as long as the vehicle remains en route to or within the survey area. Figure 3 depicts an example survey maneuver in MOOS-IvP's simulation viewer with possible bounds on the x and y coordinates.

Table 1: Action specification for surveying an area, demonstrating bounding effects to be used during state

Action Name	survey-area
Parameters	?area-lower-x (type real) ?area-upper-x (type real) ?area-lower-y (type real) ?area-upper-y (type real)
Conditions	(assign ?lower-x (min (lower-bound (x-pos)) ?area-lower-x)) (assign ?upper-x (max (upper-bound (x-pos)) ?area-upper-x)) (assign ?lower-y (min (lower-bound (y-pos)) ?area-lower-y)) (assign ?upper-y (max (upper-bound (y-pos)) ?area-upper-y)) (assign ?s (lower-bound (speed)))
Effects	(vehicle-maneuvering) (set (x-pos) (range ?lower-x ?upper-x)) (set (y-pos) (range ?lower-y ?upper-y)) (set (depth) (range 0 5)) (set (heading) (range 0 360)) (set (speed) (range ?s 2.1))

To permit the vehicle time to execute a maneuver, the HTN designer can use a built-in action, *wait-for*, that causes the GDA Controller to suspend plan execution. The *wait-for* action takes as parameters an event (such as *maneuver-finished*, shown in Table 2) and the event's arguments to indicate the point when plan execution should resume. It also optionally takes a timeout for the event's occurrence. If *wait-for* is operator r_i in the plan, PHOBOS copies x_{i-1} to create x_i (*wait-for* does not affect the world). PHOBOS uses the event specified in arguments to *wait-for* (e.g., *maneuver-finished*) as r_{i+1} and projects the effects of that event to create x_{i+1}. The use of these expectations is described in §4.2.

In planning, placing bounds on continuous values is equivalent to replacing each continuous value v with two continuous values v_u and v_l (i.e., v's upper and lower bounds). During discrepancy detection, it is necessary to understand the semantics of v_u and v_l as bounds on v. Our representation provides this knowledge by associating v with its bounds in the tuple X.

4.2 Discrepancy Detection

During plan execution, the Discrepancy Detector must monitor the resulting states to ensure that they meet the requirements set forth in the expectations. In addition to the set comparisons used in ARTUE to check facts and fluents, our Detector must compute derived values in K from the current observation o_c, and compare them and bounded state values in B with the most recent expectation x_t to verify that they lie within the expected ranges.

During a *wait-for* action, each observation represents a situation after the event at which execution should resume, a normal situation during the wait period, or an unexpected situation requiring goal reasoning. The Controller tests

Table 2: Event specification for finishing a motion, demonstrating effects used to detect the event during execution

Event Name	maneuver-finished
Conditions	(assign ?s (upper-bound (speed)))
Effects	(not (vehicle-maneuvering)) (set (speed) (range 0 ?s))

observations against the expectation corresponding to the specified event (x_{i+1}, where *wait-for* is r_i). If no discrepancy is detected, the Controller assumes the event has occurred and resumes plan execution. If a discrepancy is detected, the Controller next tests the observation against the expectation corresponding to the *wait-for* action (x_i). If no discrepancy is detected, the Controller assumes the event has not yet occurred and waits for the next observation. If a discrepancy is detected, the Controller begins the GDA cycle to respond to it. In our example, when the motion controller completes the survey operation, it removes the *vehicle-maneuvering* fact from the state, causing the GDA Controller to detect the event and resume plan execution.

5. Empirical Study

We claim that using bounded expectations in GDA can reduce the number of false discrepancies and planning time for tasks in an AUV environment. To evaluate this hypothesis, we conducted tests on a simulated AUV using the uSimMarine utility included with MOOS-IvP. We applied our GDA agent to direct a simulated AUV in three simple missions. In each mission, we used thirty randomized scenarios.

We compared a configuration of the agent using PHOBOS to a configuration using V-PHOBOS, a planner based on the same code as PHOBOS. V-PHOBOS does not provide bounding effects, but integrates a vehicle dynamics model from uSimMarine. It uses this model to project expected states at intervals during motion actions, similar

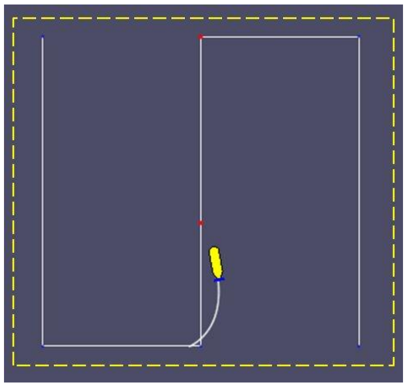

Figure 3: An AUV survey maneuver (solid line) with bounded expectation (dashed line) and AUV trajectory (curved line)

to projections that might be produced by a continuous-time planning model, but requiring less domain engineering. In other respects the agent configurations were identical. For the experiments, we used rule-based goal formulation and priority values for goal management. More advanced techniques for goal selection have been investigated (e.g., see Powell, Molineaux, and Aha 2011), but were not used in this study. Discrepancy detection was performed with a fixed threshold for continuous values.

Mission 1 - Waypoint Following: In this mission, the vehicle visits a sequence of five waypoints and returns to its starting point. There are no other vessels in the area. The waypoints and starting point were drawn randomly from a uniform distribution over a rectangular region.

Mission 2 - Survey with Surface Vessel: In this mission, the AUV surveys an area using a lawnmower pattern. Meanwhile, a surface vessel traverses the area on a fixed route. The AUV can detect the vessel, but does not assess it as threatening since it is not actively searching for the AUV. The AUV starting point was chosen randomly as in Mission 1, the survey region's center was similarly selected from a smaller subregion, the region's extent in x and y was randomly chosen from a set of predefined values, and the approaching vessel's start and end points were randomly selected from areas to each side of the possible survey region, in the y direction.

Mission 3 - **Survey with Hostile Surface Vessel**: In this mission, the AUV surveys an area with a traversing vessel as in Mission 2. However, the surface vessel uses active sensors to search for the AUV, which causes the GDA agent to detect discrepancies resulting from the pings. The agent uses goal formulation rules to respond to the pinging by avoiding the vessel. For simplicity of modeling, the goal directs the AUV to move toward a given "safe" point. Parameter values were randomly selected as in Mission 2, and the safe point was determined in advance from the aggressor's start and end points.

Our results are summarized in Figures 4-5. Because MOOS sensor readings may not be taken or delivered exactly when planned, V-PHOBOS caused discrepancies even in Mission 1, which has no true unexpected events. In Missions 2 and 3, V-PHOBOS's lengthy planning time often caused plan production to lag behind updates to the vehicle's position, causing discrepancies at the start of each plan. For our metrics, on average, the agent using PHOBOS performed better by at least an order of magnitude in all three missions. A t-test indicates that the GDA agent, when using PHOBOS, performed better than when using V-PHOBOS with $p < 10^{-14}$ for all metrics.

The results support our claim that planning and discrepancy detection with PHOBOS can reduce planning time and false discrepancies in a simulated AUV

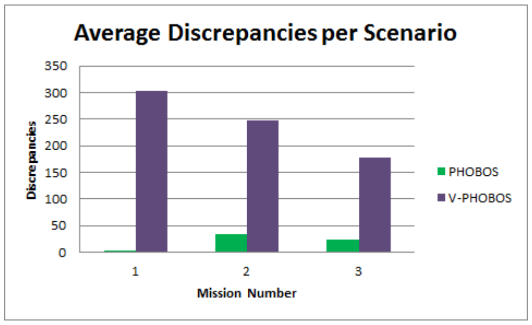

Figure 4: Discrepancies encountered per scenario, by mission

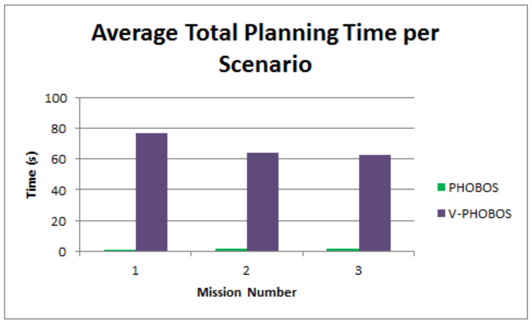

Figure 5: Average total planning time, by mission

environment. Also, PHOBOS permits goal reasoning on continuous state features such as position, speed, and battery health, allowing the agent to make more informed decisions than agents that use symbolic representations.

6. Conclusions

We introduced PHOBOS, an HTN planner with effects for creating bounded expectations. We claimed that this extension would reduce planning time and false discrepancies in a GDA agent. Our empirical study, in which we compared PHOBOS to a similar HTN planner that makes precise predictions over a complex motion, supports this claim.

Future research tasks include investigating PHOBOS's generality with respect to other domains that also include unpredictable continuous values. We will also test PHOBOS's utility in more challenging scenarios in our AUV domain, investigate more thoroughly the use of bounded derived values, and evaluate the utility of GDA as an AUV control technology in comparison to existing AUV frameworks.

Acknowledgements

The authors thank NRL for supporting this work. The views and opinions contained in this paper are those of the authors and should not be interpreted as representing the official views or policies, either expressed or implied, of NRL or the DoD.

References

Benjamin, M., Schmidt, H., Newman, P., & Leonard, J. 2010. Nested autonomy for unmanned marine vehicles with MOOS-IvP. *Journal of Field Robotics* 27:834-875.

Bouguerra, A., Karlsson, L., and Saffiotti, A. 2008. Monitoring the execution of robot plans using semantic knowledge. *Robootics and Autonomous Systems* 56:942-954.

Cashmore, M., Fox, M., Larkworthy, T., Long, D., and Magazzeni, D. (2013). Planning Inspection Tasks for AUVs. In *Proceedings of MTS/IEEE OCEANS 2013*. San Diego, CA.

Cox, M., Oates, T., Paisner, M., and Perlis, D. 2012. Noting anomalies in streams of symbolic predicates using A-distance. *Advances in Cognitive Systems* 2:167-184.

Erol, K., Hendler, J., & Nau, D.S. 1994. HTN planning: complexity and expressivity. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, 1123-1128. Atlanta, GA: AAAI Press.

Fichtner, M., Großmann, A., and Thielscher, M. 2003. Intelligent Execution Monitoring in Dynamic Environments. *Fundamenta Informaticae* 57:371-392.

Fox, M., & Long, D. 2003. PDDL2.1: An extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research* 20:61-124.

Fox, M., & Long, D. 2006. Modelling mixed discrete-continuous domains for planning. *Journal of Artificial Intelligence Research* 27:235-297.

Gat, E., Slack, M.G., Miller, D.P., Firby, R.J. 1990. Path planning and execution monitoring for a planetary rover. In *Proceedings of the 1990 IEEE International Conference on Robotics and Automation*. Cincinnati, OH: IEEE Press.

Jaidee, U., Muñoz-Avila, H., & Aha, D.W. 2011. Case-based learning in goal-driven autonomy agents for real-time strategy combat tasks. In M.W. Floyd & A.A. Sánchez-Ruiz (Eds.) *Case-Based Reasoning in Computer Games: Papers from the ICCBR Workshop*. U. Greenwich: London, UK.

Jaidee, U., Muñoz-Avila, H., & Aha, D.W. 2013. Case-based goal-driven coordination of multiple learning agents. In *Proceedings of the Twenty-first International Conference on Case-Based Reasoning*, 164-178. Saratoga Springs, NY: Springer.

Klenk, M., Molineaux, M., & Aha, D.W. 2013. Goal-driven autonomy for responding to unexpected events in strategy simulations. *Computational Intelligence* 29:187-206.

Mausam and Weld, D.S. 2008. Planning with durative actions in stochastic domains. *Journal of Artificial Intelligence Research*, 31:33-82.

Molineaux, M., Klenk, M., & Aha, D.W. 2010a. Goal-driven autonomy in a Navy strategy simulation. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*. Atlanta, GA: AAAI Press.

Molineaux, M., Klenk, M., & Aha, D.W. 2010b. Planning in dynamic environments: Extending HTNs with nonlinear continuous effects. In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*. Atlanta, GA: AAAI Press.

Nau, D.S. 2007. Current trends in automated planning. *AI Magazine*, 28(4):43–58.

Nau, D.S., Cao, Y., Lotem, A., & Muñoz-Avila, H. 1998. SHOP: Simple hierarchical ordered planner. In *Proceedings of the Sixteenth International Joint Conferences on Artificial Intelligence*. Stockholm, Sweden.

Palomeras, N., El-Fakdi, A., Carreras, M., & Ridao, P. 2012. COLA2: A control architecture for AUVs. *IEEE Journal of Oceanic Engineering* 37:695-716.

Petersson, O. 2005. Execution monitoring in robots: a survey. In *Robotics and Autonomous Systems* 53:73-88.

Plaku, E., and McMahon, J. 2013. Combined mission and motion planning to enhance autonomy of underwater vehicles operating in the littoral zone. In *Workshop on Combining Task and Motion Planning at IEEE International Conference on Robotics and Automation*. Karlsruhe, Germany.

Powell, J., Molineaux, M., & Aha, D.W. 2011. Active and interactive learning of goal selection knowledge. In *Proceedings of the Twenty-Fourth Florida Artificial Intelligence Research Society Conference*. West Palm Beach, FL: AAAI Press.

Py, F., Rajan, K., & McGann, C. 2010. A systematic agent framework for situated autonomous systems. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, 583-590. International Foundation for Autonomous Agents and Multiagent Systems.

Rajan, K., Py, F., and Barreiro, J. 2012. Towards Deliberative Control in Marine Robotics. In *Marine Robot Autonomy*, ed. M. Seto. Springer Verlag.

Turner, R. 1995. Context-sensitive, adaptive reasoning for intelligent AUV control: Orca project update. In *Proceedings of the Ninth International Symposium on Unmanned, Untethered Submersible Technology*. Durham, NC.

Weber, B., Mateas, M., & Jhala, A. 2012. Learning from demonstration for goal-driven autonomy. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*. Toronto, Canada: AAAI Press.

Younes, H. L., & Littman, M. L. 2004. PPDDL1.0: An extension to PDDL for expressing planning domains with probabilistic effects. Technical Report, CMU-CS-04-162, Carnegie Mellon University, Pittsburgh, PA.