Goal-Driven Autonomy in a Navy Strategy Simulation

Matt Molineaux[1], Matthew Klenk[2], and David W. Aha[2]

[1]Knexus Research Corporation; Springfield, VA 22153

[2]Navy Center for Applied Research in Artificial Intelligence;

Naval Research Laboratory (Code 5514); Washington, DC 20375

matthew.molineaux@knexusresearch.com | {matthew.klenk.ctr,david.aha}@nrl.navy.mil

Abstract

Modern complex games and simulations pose many challenges for an intelligent agent, including partial observability, continuous time and effects, hostile opponents, and exogenous events. We present ARTUE (Autonomous Response to Unexpected Events), a domain-independent autonomous agent that dynamically reasons about what goals to pursue in response to unexpected circumstances in these types of environments. ARTUE integrates AI research in planning, environment monitoring, explanation, goal generation, and goal management. To explain our conceptualization of the problem ARTUE addresses, we present a new conceptual framework, *goal-driven autonomy*, for agents that reason about their goals. We evaluate ARTUE on scenarios in the TAO Sandbox, a Navy training simulation, and demonstrate its novel architecture, which includes components for Hierarchical Task Network planning, explanation, and goal management. Our evaluation shows that ARTUE can perform well in a complex environment and that each component is necessary and contributes to the performance of the integrated system.

1. Introduction

Many modern video games and training simulations are complex environments that are continuous in time and space, partially observable, open with respect to the introduction of new objects, and unpredictable due to hostile opponents and exogenous events. These complications make the environment difficult to predict, and plans quickly become obsolete; mechanisms for handling surprises and other prediction failures are of high importance. To operate autonomously in these environments, intelligent agents must perform situation assessment, select appropriate goals, create plans to satisfy these goals, and execute them. During execution, opportunities and obstacles may occur outside the scope of the agent's current goals, but which are important to its central mission. Our focus is on a new generation of agents that generate and reason about their goals as a primary focus of their reasoning process.

This differs from approaches such as *online planning*, in which an agent generates new plans for user-provided goals during a plan's execution. We extend online planning with a conceptual model of g*oal-driven autonomy* (GDA), in which an agent reasons about its goals, identifies when they need to be updated, and changes or adds to them as needed for subsequent planning and execution. We present a conceptual model for GDA that integrates four reasoning tasks: environment monitoring, discrepancy explanation, goal generation, and goal management. Our hypothesis is that GDA enables an agent to outperform planning alone in complex environments.

We instantiate the GDA model in the ARTUE system, which integrates: (1) a novel Hierarchical Task Network (HTN) planner that reasons about exogenous events by projecting future states in dynamic continuous environments, (2) an explanation component that reasons about hidden information in the environment, (3) a component that uses domain knowledge in the form of *principles* to reason about and generate new goals, and (4) a goal management component responsible for prioritizing and issuing goals to the planner. ARTUE is novel in its approach to handling unexpected changes in the world by first explaining those changes, then generating new goals which incorporate the explained knowledge about hidden aspects of the environment. This approach allows ARTUE to handle challenges from new and unobservable objects within the framework of planning. Unlike most modern agents, ARTUE explicitly reasons about hidden state, the passage of time, continuous and discrete state, *and* exogenous events. To demonstrate its utility, we describe an evaluation of ARTUE on three scenarios from a Navy training simulation, the Tactical Action Officer (TAO) Sandbox (Auslander *et al.* 2009). Our ablation study illustrates the importance of the four GDA subtasks, showing that each contributes significantly to performance.

2. Related Work

Classical planning makes assumptions about how an agent finds a sequence of actions that transform an initial state into some goal state (Ghallab *et al*. 2004).

GDA relaxes several of these assumptions simultaneously, in contrast to many efforts that focus on relaxing only some subset of these assumptions.

Deterministic environments: Classical planning assumes that each future state is determined by the action executed in the current state. *Contingency planning* relaxes this by generating conditional plans that are executed only when an action does not achieve its intended effects (Dearden *et al.* 2003). Uncertainty in future state prediction is often captured as *partial observability*, which can be modeled using Markov decision processes (Puterman 1994). Likhachev and Stentz (2009) observe that these approaches scale poorly, and cannot incorporate domain-specific heuristic knowledge about the environment. To address this, their PCPP planner instead reasons about preferences between unknown values of the state when generating the plan.

Static environments: Another classical assumption is that the environment does not change other than through the execution of agent actions. Plan monitoring can be used to detect changes in the environment that can cause plan failure. For example, *incremental planners* plan for a fixed time horizon, execute the plan, and then generate a new plan from the current state. This process continues until a goal state is reached. For example, CPEF (Myers 1999) generates plans to achieve air superiority in military combat and replans when unexpected events occur during execution (e.g., a plane is shot down). Some recent approaches instead focus solely on dynamic replanning (e.g., HoTRiDE regenerates only part of its plan when an action fails (Ayan *et al.* 2007)).

Discrete effects: Complex environments are subject to continuous change. For example, a unit's location, health, and fuel all can continuously change over time. However, few systems can process continuous effects (e.g., COLIN can plan using durative actions with linear continuous effects (Coles *et al.*, 2009)).

Static goals: Classical planning assumes that the goals are all-or-nothing and static. If no plan can achieve all of the goals, then classical planners will fail. *Partial satisfaction planning* (PSP) relaxes this all-or-nothing constraint, and instead focuses on generating plans that achieve some "best" subset of goals (i.e., the plan that gives the maximum trade-off between total achieved goal utilities and total incurred action cost) (van den Briel *et al.* 2004). Other researchers have addressed the limitations of static goals. For example, Coddington and Luck (2004) bestowed agents with *motivations*, which generate goals in response to specific states. For example, if a rover's battery charge falls below 50%, then a goal to attain a full battery charge will be generated (Meneguzzi and Luck 2007). Another approach is to allow for goals to reference objects that are unknown at planning time. *Open world*
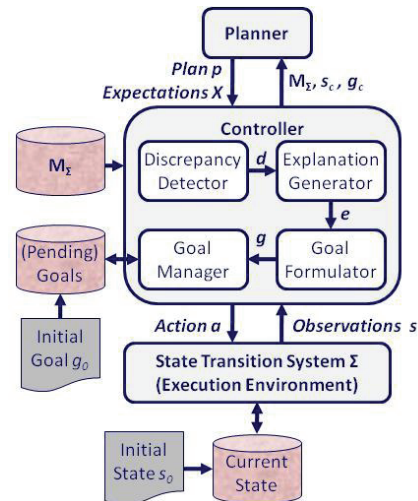

Figure 1: A Conceptual Model for Goal-Driven Autonomy

quantified goals combine information about sensing objects and generating goals into an existing PSP system (Talamadupula *et al.* 2009). Similarly, Goldman (2009) describes a system with universally quantified goals that allows planning for sets of entities whose cardinality is unknown at planning time. Several systems (e.g., PECAS (Hawes et al, 2009)) generate goals at execution time based on a human's requests or commands.

Although these assumptions characterize complex environments, none of these previous efforts relax all four simultaneously, which is the focus of GDA.

There is a rich history of developing agent architectures for increasingly sophisticated environments, e.g., TACAIR-SOAR (Jones *et al.* 1999). Unlike reactive architectures, such as ICARUS (Langley and Choi 2006), GDA separates environmental and goal reasoning from action selection, which permits additional reflection as required. Recently, Choi (2010) has been working on extensions to the Icarus architecture which create goals using constraint-like goal descriptions, Furthermore, the goals considered here may differ substantially from the current goals and consequently should not be considered subgoals. They may be autonomously generated, and involve objects that are not known or available until execution time. We detail the GDA framework in Section 3.

3. Goal-Driven Autonomy

Cox's (2007) INTRO system provides the inspiration for several concepts in goal-driven autonomy with its focus on integrated planning, execution, and goal reasoning. We extend these ideas and consider them as a general agent framework.

GDA is a conceptual model of online planning in autonomous agents. Figure 1 illustrates how GDA

extends Nau's (2007) model of online planning. The GDA model primarily expands and details the scope of the *Controller*, which interacts with a *Planner* and a *State Transition System* Σ (an execution environment). We present only a simplified version of this model, and the ARTUE system is only one possible implementation.

System Σ is a tuple (S,A,E,γ) with states S, actions A, exogenous events E, and state transition function γ: $S\times(A\cup E)\rightarrow 2^S$, which describes how an action's execution or an event's occurrence transforms the environment from one state to another. In complex environments, the agent has only partial access to the state, set of events, and state transition function.

The Planner receives as input a planning problem (M_Σ,s_c,g_c), where M_Σ is a model of Σ, s_c is the current state, and $g_c\in G$ is a goal that can be satisfied by some set of states $S_g\subseteq S$. The Planner outputs a plan p_c, which is a sequence of actions $A_c=[a_c,...,a_{c+n}]$. In the GDA model, the Planner generates a corresponding sequence of expectations $X_c=[x_c,...x_{c+n}]$, where each $x_i\in X_c$ is a set of state constraints corresponding to the sequence of states $[s_{c+1},...,s_{c+n+1}]$ expected to occur when executing A_c in s_c using M_Σ.

The Controller sends the actions in the plan to Σ and processes the resulting observations. The GDA model takes as input initial state s_0, initial goal g_0, and M_Σ, which are sent to the Planner to generate plan p_0 and expectations X_0. When executing p_0, the Controller performs the following four knowledge-intensive tasks, which uniquely distinguish the GDA model:

1. *Discrepancy detection*: GDA must first detect unexpected events before deciding how to respond to them. This task compares the observations s_{c+1} obtained from executing action a_c in state s_c with the expectation $x_c\in X$ (i.e., it tests for constraint violations corresponding to unexpected observations). If one or more discrepancies $d\in D$ are found, then explanation generation is performed to explain them.
2. *Explanation generation*: The cause for a detected discrepancy must be revealed so that it can be addressed. Given a state s_c and discrepancy(ies) $d\in D$, this task hypothesizes one or more explanations $e\in E$ of their cause.
3. *Goal generation*: Resolving the discrepancies may warrant a change in the current goal(s). This task generates goal(s) $g\in G$ in response to D, given explanation(s) $e\in E$ and the current state $s_c\in S$.
4. *Goal management*: The generation of a new goal may warrant its immediate focus and/or removal of some existing goals. Given a set of pending goals $G_P\in G$ (one or more of which may be the focus of the current plan execution) and new goal(s) $g\in G$, this task may update G_P (e.g., by adding g and/or deleting/modifying other pending
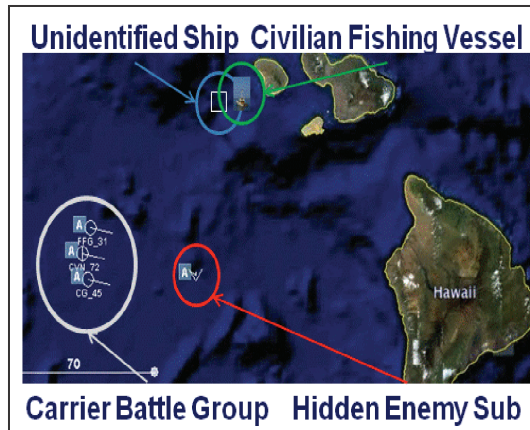

Figure 2: A screen shot from the TAO Sandbox with callouts highlighting friendly and enemy units.

goals) and will select the next goal(s) $g'\in G_P$ to be given to the Planner. (It is possible that $g=g'$.)

GDA makes no commitments to specific types of algorithms for the highlighted tasks (e.g., goal management may involve comprehensive goal transformations (Cox and Veloso 1998)), and treats the Planner as a black box. In Section 5, we describe one instantiation of this model.

4. The TAO Sandbox Environment

The TAO Sandbox is a strategy simulator used by the US Navy to train Tactical Action Officers in anti-submarine warfare (Auslander *et al.* 2009). Figure 2 shows a screenshot from the TAO Sandbox. In this simulation, trainees accomplish their objectives by giving orders to naval ships, planes, and helicopters. Vessel positions, fuel levels, heading, and speed are important *fluents*, continuously varying numeric quantities, in this domain. The trainee's actions are orders, which occur instantaneously. The effects of these orders may be instantaneous (e.g., launch a helicopter), of fixed duration (e.g., move to a specific location), or of indefinite duration (e.g., follow another vessel). Therefore, agents interacting with the simulation must reason about instantaneous occurrences and continuous effects.

Attempting missions in the TAO Sandbox autonomously is a continuous planning problem (desJardins *et al.* 1999). Opportunities and failures arise that require an effective response. In addition to being continuous, we define *complex* environments to be partially observable, and open with respect to new objects and unpredictable events. Therefore, our GDA agent monitors both the continuous and discrete state of the environment during plan execution.

In order to describe and reason about the TAO Sandbox environment, we modeled it using the domain language PDDL+. PDDL+ (Fox and Long 2006) was designed to support reasoning about mixed

discrete-continuous domains, such as the TAO Sandbox. In addition to the *actions* of traditional planning, changes to the discrete state occur as the result of instantaneous *events* in the environment, some of which can be predicted, and others of which can only be recognized afterward. To capture changes in the continuous state, PDDL+ introduces ongoing *processes*, which are defined by their *participant types, conditions*, and *effects*.

5. The ARTUE Prototype

To explain ARTUE's novel features and the integration of components resulting in a GDA system, we describe a cycle of ARTUE's execution in a simplified version of a Norwegian transport scenario from the TAO Sandbox domain. This scenario involves four objects: a transport ship, two ports, and a destination. ARTUE's user-supplied goal is to move the transport ship to the destination. This is at first a simple goal to achieve, but as the scenario progresses, a severe storm will rise quickly that can sink the ship; this storm cannot be (directly) observed until a time too late to save the ship. To protect itself, the ship may seek shelter in one of two ports. Further complicating the situation, unseen icebergs are in the water that can stop the ship. To safely guide the ship to its destination, ARTUE will need to execute plans, monitor them for failures, determine why those failures occurred, and generate new goals.

5.1. HTN Planning

In GDA, the Planner generates (1) plans to satisfy the goals selected by the system and (2) expectations about how the environment will change during plan execution. For ARTUE, we extended the SHOP2 HTN planner (Nau *et al.* 2003) to reason about PDDL+ domains. Our extensions include the addition of a *wait* action, which allows SHOP2 to incorporate the passage of time in its plans, and a state projection algorithm, which projects the continuous effects of active processes and the timing of exogenous events. For further details, see (Molineaux *et al.* 2010).

ARTUE begins with the task: `(MoveShip Ship1 Destination1)`. Our HTN methods decompose this task into the following actions: `(navigate Ship1 Destination1)`, `(wait 10)`. Using the state projection algorithm, the Planner predicts changes to the continuous and discrete state throughout the plan's execution (e.g., the expected changes in the ship's location throughout its movement).

5.2. Discrepancy Detection

In the GDA framework, an agent must monitor the state for unexpected events. ARTUE does this by examining the state whenever a wait action completes and at fixed intervals during longer waits.

Discrepancies between the observed state and the expected state projected by the Planner trigger the explanation generation process.

For discrete states, discrepancies are found using a set difference operation between the set of expected literals and the set of observed literals. For continuous states, the observed and expected value of each fluent is compared; a discrepancy is considered to occur whenever these values differ by less than 0.1% of the (absolute) observed value.

In our example, after 5 minutes, a lightning strike is observed in the environment. As this was not predicted, the literal `(see lightning)` is present in the observed discrete state but not the expected discrete state, which triggers explanation generation.

5.3. Explanation Generation

Discrepancies between an expected state and an actual state arise as a result of one of three circumstances:

1. A hidden factor is influencing the state.
2. The dynamics according to which the state was projected were incorrect, meaning that the agent's domain knowledge is flawed.
3. The perception of the state is incorrect.

ARTUE generates explanations of discrepancies in search of the hidden factors that affect the state in the first case; the other two cases are primarily distractions for our purposes (see below). Explanations are produced by abduction over the conditions and effects found in the planning domain using an Assumption-based Truth Maintenance System (ATMS; de Kleer 1986).

Before execution time, ARTUE transforms the PDDL+ domain into a set of ATMS rules that infers the effects of all processes and events in the domain over some known period of time during which no non-wait actions occur. For reasoning about hidden state, the planning domain includes a set of hidden predicates, which refer to information that cannot be observed directly, but can be abduced during explanation, as well as processes and events that are defined using these hidden predicates.

During explanation, a list of possible hidden facts is generated using a list of hidden discrete predicates belonging to the domain, a list of all known objects from the observed state, and skolem objects which stand for possible unobserved objects. All of these possible literals become *assumptions*, facts that may be assumed to be part of the state if no contradiction occurs.

Using this information, the ATMS searches for possible worlds containing only observed facts and (assumed) hidden facts that are consistent with the generated rules.

One complication to this process comes from inaccurate perceptions. In continuous environments,

measuring and rounding errors often make exact perceptions impossible. Therefore, ARTUE considers possible worlds to be consistent when fluent values are "close enough" to the observed values. Just as in discrepancy detection, measurements within 0.1% of the observed value are considered equivalent. This is, admittedly, a poor model of perception error, and can be improved in future work.

Another complication occurs when domain knowledge is incomplete. ARTUE allows for the possibility that not all processes and events in the environment are specified completely and accurately. If no consistent possible worlds are found, a partial explanation can still be constructed. Therefore, ARTUE searches for possible worlds in which each successor fact, by itself, is consistent with all the prior facts. When no such world can be found, the fact that cannot be explained is discarded. Explanation then takes place over all remaining facts that are true in some known possible world. In future work, explanation failures due to incomplete domain knowledge could lead to the construction of learning goals (Ram and Leake 1995) to refine ARTUE's knowledge.

Ideally, after explanation generation ARTUE will have found one or more possible worlds in which all of the observed facts are true. If so, then the assumptions shared by all possible explanations will be added to the agent's beliefs about the current state and can be used in goal generation and future planning steps.

In the running example, explanation generation finds a single possible world, containing one assumption, `(stormApproaching Ship1)`, which explains the lightning strike. This fact is adopted as a belief, and goal generation is triggered.

5.4. Goal Generation

In GDA, the agent considers the explained discrepancy(ies) and the current state to determine what goals, if any, should be generated. ARTUE uses background knowledge for this task in the form of *principles,* which are schemas whose components are a set of *participants*, a *condition*, an *intensity level*, and a *goal form*. Each participant is assigned a type (e.g., FriendlyShip). Conditions are statements concerning the participants that must hold in the agent's beliefs to generate the goal specified by the goal form. The intensity level is a fixed value proportional to the importance or urgency of satisfying the generated goal. Once a goal is generated, the intensity of the principle used to generate it is passed with the goal to the goal manager.

In our example, given the approaching storm explanation and the current state, ARTUE attempts to generate goals for each of its principles. When some set of objects from the agent's beliefs correspond to

the principle's participant types and satisfy its condition, the corresponding goal is generated.

Table 1: A principle directing the agent to spawn a *find shelter* goal to avoid an incoming storm.

Name	FindShelter
Participants	`?vehicle type = FriendlyShip`
Condition	`(stormApproaching ?vehicle)`
Intensity Level	HighIntensity
Goal form	`(DirectShipToShelter ?vehicle)`

Using the principle in table 1, ARTUE finds one set of objects that satisfies the conditions according to its current beliefs (i.e., `?vehicle = Ship1`). For this set of entities, the agent creates an instantiation of the principle's goal form and asserts it as a goal (i.e., `(DirectShipToShelter Ship1)`). This goal and its intensity are added to the list of pending goals, which becomes:

- `(MoveShip Ship1 Destination1)`
  - LowIntensity
- `(DirectShipToShelter Ship1)`
  - HighIntensity

5.5. Goal Management

Given multiple goals, the agent must decide which one to pursue next, and how to act in the presence of competing goals. We call this task *goal management*. Currently, ARTUE can only execute one plan at a time. When multiple goals exist, the goal with the highest intensity is selected. If the Planner cannot generate a plan to achieve that goal, the goal with the next highest intensity is selected, until an achievable goal is found.

In our example, two goals are active: `(DirectShipToShelter Ship1)` and `(MoveShip Ship1 Destination1)`. The first goal has a higher intensity, and given that the Planner can generate a plan to satisfy it, it becomes the selected goal.

6. Evaluation

We present results for three scenarios defined in the TAO Sandbox environment. In each of these scenarios, a situation arises outside the scope of the agent's goals that is nonetheless highly important to address. Performance is scored using a scenario-specific score metric based on satisfaction of a user-specified goal as well as the response to an unexpected situation. To demonstrate the contribution of each component, we constrast the ARTUE agent with three ablated versions: PLAN1, REPLAN, and EXPLAIN. PLAN1 performs planning once and never changes its plan. REPLAN monitors the environment for discrepancies and changes its

plan when any discrepancies are detected, but does not generate explanations or change its goals. EXPLAIN explains discrepancies it detects in the environment, adding to its beliefs about the current state, but forms no new goals. Finally, the complete GDA agent, ARTUE, monitors, explains, and generates and manages its own goals.

The three scenarios that we use to test ARTUE differ widely in their tasks, so as to illustrate its generality. For ARTUE, parameterization is minimal and knowledge creation costs are high. Domain knowledge for ARTUE consists of the PDDL+ domain, HTN methods and tasks, a set of principles, and rules used in goal generation. The amounts of knowledge created for the TAO Sandbox domain, covering all 3 scenarios, is as follows: 22 defined types, 115 predicate forms (17 hidden), 8 fluent forms, 33 action models, 37 event models, 7 process models, 42 tasks, 148 methods, 9 principles, and 28 goal generation rules.

In the first scenario, *Scouting*, ARTUE is given a goal to identify nearby ships. To do this, a task group vessel must be sent close enough to visually identify each unknown ship. During this exercise, a hidden submarine torpedoes a nearby ship, causing it to sink. The score is based on identifying each of the nearby ships as well as the sub (which requires special sensors), and also destroying the sub, which is outside the scope of the user-supplied goal.

In the second scenario, *Iceberg*, a ship is transporting cargo to a destination in Norway. During its transport, a storm arises, which is presaged by a lightning strike. This strike causes a large iceberg to calve, blocking the entrance to a nearby port. Due to the storm's severity, all ships must seek shelter, and a nearby ship, which does not detect the iceberg, founders on it. The score for this scenario is based on how close the ship comes to its destination, how long it survives, and how early it's able to arrive to rescue the passengers aboard the foundering ship, which is outside the scope of its supplied goals.

The third scenario, *SubHunt*, involves a search for an enemy sub that has been spotted nearby. A ship is sent out to find and engage it. However, this sub has been laying mines which can incapacitate the searching ship. Points are awarded for finding and destroying the sub, as well as sweeping the mines.

We tested each agent 25 times per scenario with random variation in object starting locations and prearranged events, and held these constant across all versions. Table 2 shows the average scores for each agent on each scenario. All scores are scaled between 0 and 1, with 1 being the maximum performance. However, scores are not directly relatable between scenarios. The difference between GDA and each ablated agent is statistically significant ($p<.0001$) in each scenario. In addition, the differences between successive ablations are also significant ($p<.01$).

Table 2: Agent scores in the TAO Sandbox ablation study

Agent	Scouting	Iceberg	SubHunt
PLAN1	0.33	0.35	0.35
REPLAN	0.40	0.48	0.48
EXPLAIN	0.58	0.64	0.74
ARTUE	0.74	0.73	0.98

To illustrate the complexity of the reported scenarios, table 3 lists the following characteristics describing how much work ARTUE performed on each scenario averaged over 25 runs: number of discrepancy checks, number of replans, average (longest) plan length, average (longest) simulated time, and average (longest) clock time. Most of ARTUE's clock time is spent in explanation, which searches through a set of world states which is exponential in the number of discrepancies.

Table 3: Scenario Difficulty Characteristics

Characteristic	Scouting	Iceberg	SubHunt
# Discrepancy Checks	24.6	30.4	16.5
# Replans	10.4	5.5	11.5
Plan Length	50 (149)	36 (463)	10.1 (45)
Sim. Time (mins)	18 (49)	153 (240)	84 (126)
Clock Time (mins)	10 (22)	15 (28)	3.5 (6)

This study helps to identify the capabilities that each GDA task provides. PLAN1 cannot tolerate even minor changes expectation failures. REPLAN responds correctly to small unexpected events such as course changes that occur during the *Identify* scenario. This requires that the event and its repercussions be immediately apparent. EXPLAIN adapts well to both the unexpected storm and the enemy submarine because it deduces their presence without sighting them and plans around them. Finally, ARTUE's goal generation and management steps are useful when unexpected situations arise which suggest pursuing other goals that most agents would not consider, such as saving the sinking ship.

7. General Discussion

These results support our hypothesis that GDA agents like ARTUE can competently respond to unexpected events in complex environments. By conception, the GDA process integrates a diverse set of AI components. Our evaluation shows that each component produces a significant increase in performance, and each component makes the contribution of successor components possible.

In particular, ARTUE integrates HTN planning with continuous effects, discrepancy detection, explanation generation using an ATMS, and goal generation and management using principles. Each of these components is domain independent. To further evaluate its generality, we are currently applying ARTUE to other strategy simulations with complex environments (i.e., open, partially observable,

continuous environments with hostile opponents and exogenous events).

Further investigations are warranted for studying GDA and its ARTUE implementation in complex task environments. Additional research is required on individual components (e.g., the HTN planning extensions for continuous effects, categorization of discrepancies, and principles for goal generation) as well as their integration in GDA agents for complex environments. These investigations will extend ARTUE toward our vision of robust autonomy.

Acknowledgements

References

Auslander, B., Molineaux, M., Aha, D.W., Munro, A., & Pizzini, Q. (2009). *Towards research on goal reasoning with the TAO Sandbox* (Technical Report AIC-09-155). Washington, DC: Naval Research Laboratory, Navy Center for Applied Research on AI.

Ayan, N.F., Kuter, U., Yaman F., & Goldman R. (2007). Hotride: Hierarchical ordered task replanning in dynamic environments. In F. Ingrand, & K. Rajan (Eds.) *Planning and Plan Execution for Real-World Systems – Principles and Practices for Planning in Execution: Papers from the ICAPS Workshop.*Providence, RI: [http://www.mbari.org/autonomy/ICAPS07-workshop]

van den Briel, M., Sanchez, R., Do, M.B., & Kambhampati, S. (2004). Effective approaches for partial satisfaction (over-subscription) planning. *Proceedings of the Nineteenth National Conference on Artificial Intelligence* (pp. 562-569). San Jose, CA: AAAI Press.

Choi, D. (2010). Coordinated Execution and Goal Management in a Reactive Cognitive Architecture. Ph.D. diss., Dept. of Aeronautics & Astronautics, Stanford University., Stanford, CA.

Coddington, A. & Luck, M. (2004). A motivation-based planning and execution framework. *International Journal on Artificial Intelligence Tools*. **13**(1). 5-25.

Coles, A.J., Coles, A., Fox, M., and Long, D. (2009). Temporal planning in domains with linear processes. *Proceedings of the International Joint Conference on Artificial Intelligence* (pp. 1671-1676). Pasadena, CA: AAAI Press.

Cox, M.T. (2007). Perpetual self-aware cognitive agents. *AI Magazine*, **28**(1), 32-45.

Dearden, R., Meuleau, N., Ramakrishnan, S., Smith, D., & Washington, R. (2003). Incremental contingency planning. In M. Pistore, H. Geffner, & D. Smith (Eds.)

Planning under Uncertainty and Incomplete Information: Papers from the ICAPS Workshop. Trento, Italy.

desJardins, M., Durfee, E., Ortiz, C., & Wolverton, M. (1999). A survey of research in distributed, continual planning. *AI Magazine*, **20**(4), 13–22.

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

Ghallab, M., Nau, D.S., & Traverso, P. (2004). *Automated planning: Theory and practice*. San Mateo, CA: Morgan Kaufmann.

Goldman, R.P. (2009). Partial observability, quantification, and iteration for planning: Work in progress. In *Generalized Planning*: *Macros, Loops, Domain Control: Papers from the ICAPS Workshop*. Thessaloniki, Greece: [http://www.cs.umass.edu/~siddhart/genplan09].

Hawes, N., Zender, H., Sjöö, K., Brenner, M., Kruijff, G.J.M., Jensfelt, P. (2009). Planning and Acting with an Integrated Sense of Space. In *Proceedings of the 1st International Workshop on Hybrid Control of Autonomous Systems -- Integrating Learning, Deliberation and Reactive Control (HYCAS)* (pp. 25-32).

Jones, R., Laird, J., Nielsen, P., Coulter, K., Kenny, P., and Koss, F. (1999). Automated Intelligent Pilots for Combat Flight Simulation. *AI Magazine*. **20**(1).

Langley, P. & Choi, D. (2006). A unified cognitive architecture for physical agents. In *Proceedings of the twenty-first AAAI conference on artificial intelligence*. Boston, MA: AAAI Press.

Likhachev, M. & Stentz, T. (2009). Probabilistic planning with clear preferences on missing information. *Artificial Intelligence*. **173**, 696-721.

Meneguzzi, F.R., & Luck, M. (2007). Motivations as an abstraction of meta-level reasoning. *Proceedings of the Fifth International Central and Eastern European Conference on Multi-Agent Systems* (pp. 204-214). Leipzig, Germany: Springer.

Myers, K.L. (1999). CPEF: A continuous planning and execution framework. *AI Magazine*, **20**(4), 63-69.

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

Nau, D., Au, T.-C., Ilghami, O., Kuter, U., Murdock, J. W., Wu, D., & Yaman, F. (2003). SHOP2: An HTN planning system. *Journal of Artificial Intelligence Research*. **20**, 379–404.

Puterman, M.L. (1994). *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. New York: John Wiley & Sons.

Ram, A. & Leake, D. (1995) *Goal-Driven Learning*. Cambridge, MA: MIT Press.

Talamadupula, K., Benton, J., Schermerhorn, P., Kambhampati, S., & Scheutz, M. (2009). Integrating a closed world planner with an open world robot: A case study. In *Bridging the Gap between Task and Motion Planning: Papers from the ICAPS Workshop*. Thessaloniki, Greece.