Active and Interactive Discovery of Goal Selection Knowledge

Jay Powell[1], Matthew Molineaux[2], and David W. Aha[3]

[1]Computer Science Department; Indiana University; Bloomington, IN 47405
[2]Knexus Research Corporation; Springfield, VA 22153
[3]Navy Center for Applied Research in Artificial Intelligence; Naval Research Laboratory (Code 5514); Washington, DC 20375
jhpowell@cs.indiana.edu | matthew.molineaux@knexusresearch.com | david.aha@nrl.navy.mil

Abstract

If given manually-crafted goal selection knowledge, goal reasoning agents can dynamically determine which goals they should achieve in complex environments. These agents should instead *learn* goal selection knowledge through expert interaction. We describe T-ARTUE, a goal reasoning agent that performs case-based active and interactive learning to discover goal selection knowledge. We also report tests of its performance in a complex environment. We found that, under some conditions, T-ARTUE can quickly learn goal selection knowledge.

1. Introduction

Modern autonomous agents can plan, learn, reason, and solve problems in the context of many diverse tasks set by a human. However, they require a human to specify all their goals. An important aspect of autonomy is the ability to self-select goals. We are studying methods for a new generation of *goal reasoning* agents that can select their own goals without human guidance (Ram and Leake 1995; Cox 2007).

Some agent architectures (e.g., Soar (Laird and Rosenbloom, 1990)) achieve many goals by recursively decomposing top-level goals into all other goals the agent might ever pursue. While this can define many interesting agents, it is restrictive. Agents that represent the relative importance of each goal and explicitly manage their *pending* goals (i.e., those they are currently pursuing) should perform more robustly in dynamic environments.

In this paper, we extend the Autonomous Response to Unexpected Events (ARTUE) agent, which performs goal formulation and goal management in the context of a Goal-Driven Autonomy (GDA) model for continuous planning (Molineaux et al. 2010). ARTUE formulates its goals using rule-based *principles*, which describe situations where specific goals should be formulated and their relative importance. Although ARTUE can formulate and manage new goals to properly respond to developing situations, it cannot respond to novel situations (i.e., those for which it lacks manually-encoded knowledge for goal selection and prioritization). However, if ARTUE could learn this knowledge, it would exhibit even greater autonomy.

We extend ARTUE with the ability to learn goal selection knowledge through interaction with an expert. We frame this as a case-based supervised learning task that employs *active learning* (AL) (Settles 2009): the agent can query a human expert for knowledge rather than be limited to consulting its own knowledge sources. We call this extension the *Trainable Autonomous Response to Unexpected Events* (T-ARTUE) agent.

We next describe the GDA model, followed by its implementation in ARTUE and related work. We describe how goal selection knowledge can be actively and interactively learned in §4, T-ARTUE's learning algorithms in §5, and our empirical study in §6. Our results indicate that, under some assumptions, T-ARTUE's performance quickly attains the level of ARTUE.

2. Goal-Driven Autonomy

GDA is a conceptual model for online planning in autonomous agents (Molineaux et al. 2010). It separates the planning process from procedures for goal formulation and goal management. Special formalisms (e.g., Dal Lago et al. 2002) exist for managing goals during planning. However, these require a specific planner, whereas the GDA model can be paired with an arbitrary planner. In Figure 1, we illustrate how GDA extends Nau's (2007) model of online planning: it expands and details the scope of the *Controller*, which interacts with a *Planner* Π and a *State Transition System* Σ (an execution environment).

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

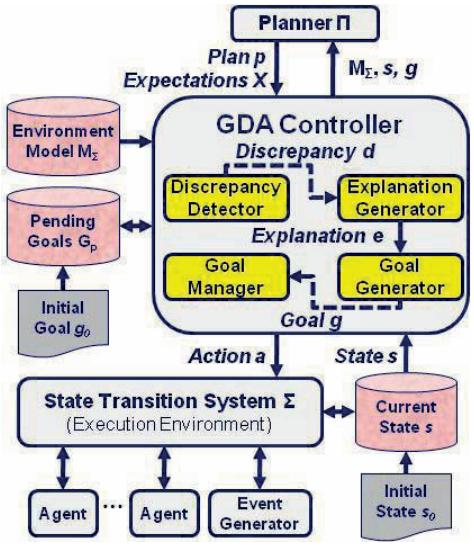

Figure 1: Goal-Driven Autonomy Conceptual Architecture

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

The Controller takes as input initial state s_0, initial goal g_0, and M_Σ, and gives them to the Planner to generate plan p_0 and expectations X_0. The Controller then forwards p_0's actions to Σ for execution and processes the resulting observations, which may also reflect the processing of other agents' actions or events from an Event Generator.

During plan execution, a GDA Controller performs the following four knowledge-intensive tasks:

Discrepancy detection: GDA detects unexpected events by comparing s_{c+1} with $x_c \in X$ (i.e., it tests for constraint violations corresponding to unexpected observations). If a discrepancy d of one or more differences exists, then explanation generation is performed to explain it.

Explanation generation: The cause for a detected discrepancy d must be revealed so that it can be resolved. Given a state s_c and d, this task hypothesizes an explanation e of its cause.

Goal generation: Resolving a discrepancy may warrant a change in the current goal(s). This task generates a goal g in response to d, given e and s_c.

Goal management: Given a set of pending goals G_P and new goal g, this task will update G_P (e.g., by adding g or deleting/modifying other pending goals) and select the next goal $g' \in G_P$ to be given to the Planner.

GDA does not prescribe specific types of algorithms for these tasks, and treats the Planner as a black box. In contrast to reactive planners (e.g., Firby 1987)), GDA agents use expectation failures for goal formulation.

3. ARTUE and Related Work

ARTUE is a GDA agent. It uses set-difference to detect discrepancies and an assumption-based truth maintenance system (ATMS) (de Kleer 1986) to generate explanations. ARTUE calls SHOP2 (Nau et al. 2003) to generate plans, where we assume a mapping exists from any goal to be achieved to a SHOP2 task that achieves it. Molineaux et al. (2010) reported that ARTUE performs well on scenarios defined using the TAO Sandbox (see §4).

To perform goal generation and management, ARTUE uses expert-provided *principles*, which encode goal selection knowledge. T-ARTUE instead learns it via a case-based active and interactive learning algorithm. Research on AL, both case-based and otherwise, tends to focus on strategies for determining which labels to request in the context of classification tasks (e.g., Hu et al. 2010; Sculley 2007). We also focus on a classification task, but in a GDA online planning context.

Two groups have investigated case-based GDA for controlling agents in complex video games. First, Weber *et al.* (2010) introduced a case-based approach for goal formulation. They define it as the tasks of (1) locating a case c whose goal state $c.s$ is most similar to the current goal g, (2) computing difference $d=c'.s-c.s$, where $c'.s$ is a future goal state of c after executing n actions, and (3) adding d to g. Our work differs in that goal selection is triggered by a state discrepancy and cases are learned using AL and interactive feedback processes. Second, Muñoz-Avila *et al.* (2010) used two manually-engineered case bases for their GDA algorithm, which (1) fetches the next expected state $c_i.x$ from a case c_i whose state and goal states are maximally similar to the current versions, (2) detects whether $c_i.x$ differs from x_{next}, and if so (3) retrieves a goal state corresponding to this discrepancy. In contrast, T-ARTUE learns to acquire cases for goal selection.

Finally, while goal reasoning research has focused on a variety of interesting tasks, no prior work exists on AL for GDA (Aha et al. 2010).

4. Discovery of Goal Selection Knowledge

We describe three types of structured interactions between an agent and an expert through which the agent can learn goal selection knowledge: (1) *goal selection queries*, (2) *generalization confirmation queries*, and (3) *goal selection criticism*. The first two are system-initiated AL techniques (occurring when the system is unable to formulate a goal), whereas the third is user-initiated. These interactions facilitate online learning, allowing an agent to derive a procedure for goal selection that meets the expert's needs.

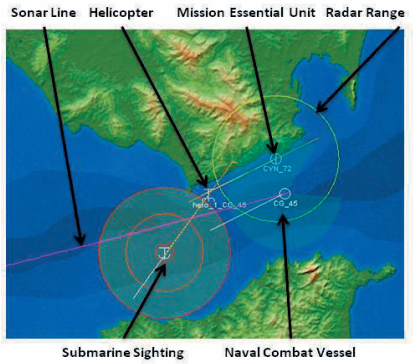

Figure 2: An Annotated Screenshot from the TAO Sandbox

We define scenarios from the TAO Sandbox (Auslander et al. 2009) to exemplify these interaction types. It is a US Navy simulator for training Tactical Action Officers in anti-submarine warfare (Figure 2). Trainees control assets (ships, planes, helicopters) by giving instantly-executed orders. Autonomous mission planning in the TAO Sandbox is a continuous planning problem (desJardins et al. 1999). This environment is partially observable, dynamic, and open with respect to the introduction of new objects. Thus, opportunities and failures can arise that benefit from a goal reasoning process. However, this requires substantial engineering effort, which motivates the development of agents that can learn new goal selection knowledge.

4.1 Goal Selection Queries

In this interaction type, the agent queries an expert for the goal to select in the current state. This resembles other AL processes, but is more constrained due to the nature of the online task, where the agent must experience and respond to states in an order determined by its environment, and has no knowledge of future states. Therefore, it only makes sense for the agent to query an expert regarding the current state, rather than an arbitrarily chosen example, as is common in AL research (Settles 2009).

The agent's query includes (1) a comprehensive state description (that the expert can use to make a decision) and (2) a set of goals (that the Planner can accept). The expert must respond with a single selected goal. For example, a TAO Sandbox agent may request a goal when observing the first signs of an approaching storm. To do this, it forms a state description (e.g., locations and velocities of all known vessels and possible destinations), and enumerates a list of the goal types it understands. It communicates these to the expert, who responds by selecting any of the infinite set of possible goal for this state, such as (*sheltered ship1*).

4.2 Generalization Confirmation Queries

Here the GDA agent requests confirmation of its learned goal selection knowledge. This enables it to get feedback on its hypothesized generalizations of user-provided goal selection knowledge with respect to the current state. These queries are less constrained than goal selection queries, as the agent can hypothesize arbitrary connections based on prior learning, and are thus more similar to traditional AL techniques.

These queries include (1) a description of the current state and (2) a set of hypothesized generalizations, where each describes the process the agent used for goal selection and the selected goal. The user can reply by confirming zero or more of the hypothesized explanations. For example, suppose a TAO Sandbox agent sees signs of another storm. It recalls prior similar occasions where the expert advised it to select the goal (*sheltered ship1*). Thus, it forms a query by (1) creating a state description and (2) a list of generalizations (e.g., the deductive rule "(*storm-signs*) → (*goal-select* (*sheltered ship1*))" and the selected goal (*sheltered ship1*)). These are displayed, and the expert responds by indicating which (if any) are confirmed.

4.3 Goal Selection Criticism

This interaction is *expert*-initiated. After an expert observes an agent operating in an environment, they may critique it (e.g., indicate that a goal selected in a given state was *not* correct). This may help the agent to recover from learned over-generalizations.

In this interaction: (1) the user requests a justification for goal selections (e.g., within a given time interval); (2) the agent presents its decisions (i.e., a list of states and goals selected); and (3) the user provides a critique (i.e., selected states and goal recommendations). For example, suppose the agent learned to return to its original goal (*transport cargo destination*) after a storm passes. However, in state *s* a nearby *ship2* requests assistance. Not recognizing this as a significant difference, the agent selects the (*transport cargo destination*) goal. Seeing this incorrect behavior, the expert later requests a justification. The agent responds with its relevant decisions and the user highlights the agent's mistake, explaining that in state *s*, the goal (*render-aid ship2*) should be selected.

5. Learning in T-ARTUE

During training, we run a simulator that gives T-ARTUE a stream of TAO Sandbox states, from which it learns its Goal Generator knowledge. When its prediction confidence is low, T-ARTUE uses the AL techniques described in §4.1 and §4.2, and the expert can also provide feedback after a trial ends (§4.3).

If at time t T-ARTUE observes a discrepancy d_t (between x_t and s_t), it calls the Explanation Generator to generate explanation e_t. Then, the Goal Generator predicts a goal g_t to resolve d_t (given e_t and s_t) using case base C. We next describe T-ARTUE's case representation, retrieval and reuse algorithms, and its learning techniques.

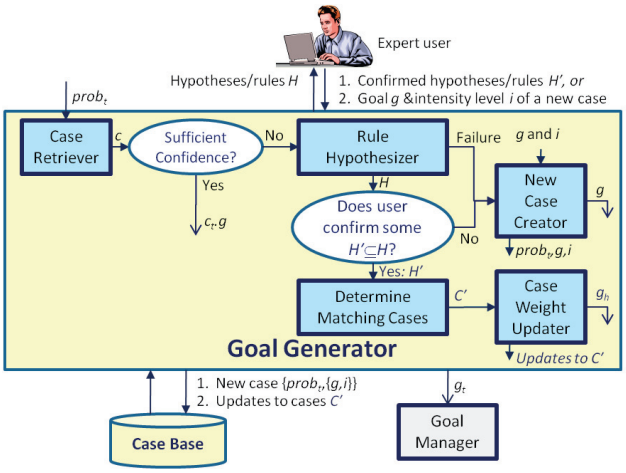

Figure 3: T-ARTUE's Active and Interactive Learning Processes

5.1 Case Representation

Each case is a pair $c = \{prob,sol\}$, where $prob=\{s,d,e\}$, s is the recorded state, d is the discrepancy at that time, and e is the explanation (i.e., a set of *beliefs*) T-ARTUE generated in response to d in s. A case's solution is a tuple $sol=\{g,i\}$, where g is the goal that T-ARTUE selected to resolve d, and i is its discrete *goal intensity level* (i.e., a fixed value proportional to the importance of satisfying g).

Each problem component is represented by a set of *literals*, which are logical expressions that ascribe a predicate to zero or more constants. There are 40-70 literals per problem. T-ARTUE maintains a weight with each literal in a problem, and these are all initially set to 1.

5.2 Retrieval and Reuse

T-ARTUE uses a weighted nearest neighbor rule to compute the feature similarity of new problem $prob_t$ at time t with the problem $c_i.prob$ of each case $c_i \in C$ as follows:

$$fsim(prob_t, c_i.prob) = \sum_{j \in prob_t} \sum_{k \in c_i.prob} \frac{w_{i,k}}{W} 1(j,k) \quad (1)$$

where j and k are literals, $w_{i,k}$ is k's weight, W is the sum of $c_i.prob$'s literal's weights, and $1(j,k)$ tests whether $j=k$. Equation 1 yields a value in [0,1]. Equation 2 transforms this value to account for case intensity:

$$sim(prob_t, c_i.prob, c_i.int) = \frac{c_i.int}{1 + e^{\alpha - \beta * fsim(prob_t, c_i.prob)}} \quad (2)$$

where $c_i.int$ is case c_i's intensity, and we set α=6 and β=10. This prevents cases with high intensity and low similarity from overriding cases with the inverse.

At time t, the Goal Generator retrieves the goal $c_t.g$ of the most similar case c_t and outputs it to the Goal Manager.

5.3 Retention and Maintenance: Active Learning

Figure 3 displays T-ARTUE's learning processes. The probability that it will request a label (i.e., a goal to formulate and its intensity) is $p(1\text{-}sim(prob_t,c_t.prob,c_t.int))$, where c_t is the retrieved case. (Initially, T-ARTUE will always request a label to seed the case base.) If it requests a label, it will (1) request it from the user, or (2) present a set of hypotheses describing the goals it believes apply, from which the user can confirm or reject some subset.

Goal Selection Queries

These queries yield a new case. Many literals in a case c's problem (e.g., positions and velocities of vehicles) are contextually irrelevant to c's goal and will vary greatly during trial runs. A few literals of a case problem (typically 1-5) will suffice to identify the goal's applicable context. Relying on initially equal weights for case retrieval can yield poor performance. Thus, T-ARTUE adapts weights to increase goal selection accuracy, as described below.

Explanation Confirmation Queries

T-ARTUE generalizes cases to increase predictive accuracy. For example, the first time signs of a storm are observed, it creates a case describing the current state (e.g., positions and velocities of all vehicles, and the literal (*storm-signs*)) and the new goal (*sheltered ship1*). The next time a storm is observed, it will not retrieve a case that recommends (*sheltered ship1*) because the new state will differ, causing it to retain another case. However, after it acquires several cases with the same goal, it can generate hypotheses that generalize its case knowledge.

T-ARTUE's rule-based hypotheses describe problem similarities among cases with the same goal, and define when a goal should be formulated. Rules are generated using Apriori (Agrawal and Srikant 1994), which takes a set of cases as input and searches for frequently occurring *itemsets* (conjuncts of literals) within their problems. A *frequent* itemset is one that appears in at least τ cases (we set τ=3) and all of its subsets are frequent. Frequent itemsets of length k are generated from frequent itemsets of length *k-1*. This process recurses, with increasing values of k, until none are found. Rules are then generated whose antecedent is a maximal-length frequent itemset and whose consequent is a corresponding goal.

This set of hypotheses H is presented to the user, who can confirm a subset $H' \subseteq H$ or reject them all. If the former, then T-ARTUE locates all cases $C' \subseteq C$ whose problems each match at least one hypothesis in H'. For each case $c_i \in C'$, the weight $w_{i,j}$ for each of its literals $l_{i,j}$ is updated. The squared error of the existing weight is defined relative to the probability that $l_{i,j}$ appears in H':

$$E(l_{i,j} \epsilon c_i) = \left(w_{i,j} - P(l_{i,j} \mid H' \subseteq H)\right)^2 \quad (3)$$

The space of possible weights per case is searched using stochastic gradient descent, where weight updates correspond to the derivative of the squared error (Equation 3) multiplied by a learning rate η (we set η=0.6).

If the user rejects all the hypotheses, then the user must provide a goal and intensity, which triggers case retention.

5.4 Goal Selection Criticism

After a trial a user can critique T-ARTUE's sequence of goal selection decisions (i.e., the goal chosen and the

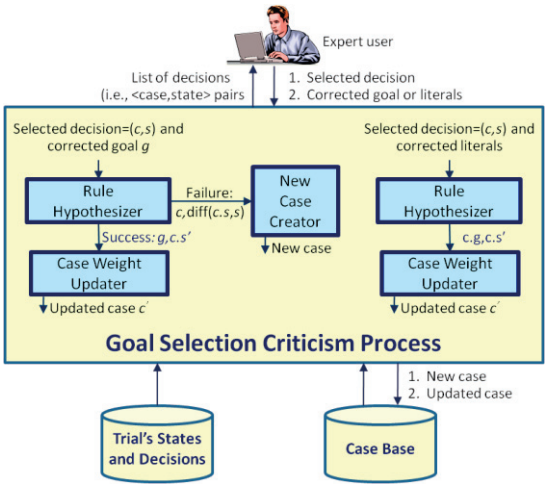

Figure 4: Process for Goal Selection Criticism

problem literals influencing a case's selection). For each decision, the user may (1) recommend a different goal, or (2) criticize the weights T-ARTUE used. (See Figure 4.)

T-ARTUE may fail to transition between goals. For example, in one scenario it must rescue a foundered vessel before heading to port. Initially, it will not recognize when the goal (*rescue ship2*) has been satisfied, and (*at ship1 destination*) should instead be pursued. After a ship is rescued, the current state includes (*rescued ship2*). This may not appear in cases with the goal (*at ship1 destination*), or its weight may be too low to correctly classify the appropriate goal. However, after the user provides a correct goal, T-ARTUE creates rules describing the differences between the case's and current problem, and automatically adapts the literals' weights (see §5.3). These adaptations are not immediately confirmed by a user, but may be corrected after subsequent trials. T-ARTUE makes many mistakes during its initial learning stages, which could cause it to create many similar cases. Thus, it attempts to hypothesize generalizations for goals provided by the user during criticism (see §5.3).

T-ARTUE may choose the correct case but using the wrong literal weights. For example, it may choose the goal (*sheltered ship1*) due to the literals (*storm-signs*), (*at-x helicopter1 35*), and (*at-y helicopter1 49*). The helicopter's position should not influence the decision to seek shelter. Thus, the user can tell T-ARTUE that (*storm-signs*) is the only relevant literal. Given this, it would create a rule associating the literals selected by the user with the case's goal, and use the algorithm in §5.3 for weight learning.

6. Empirical Study

We claim that T-ARTUE can quickly and accurately learn to respond effectively in TAO Sandbox scenarios, given access to an expert. To test this, we trained it with 30 trials for each of two TAO Sandbox scenarios, where trials differ in their randomly-generated state conditions. As a baseline, we used ARTUE (given its expert-designed goal selection knowledge – *principles*) using these conditions.

Scenario 1 (*Sub Hunt*) requires T-ARTUE to respond to a nearby ship in distress, submarines, and underwater mines. Scenario 2 (*Iceberg*) requires it to learn to respond to a fast-approaching storm, the formation of an iceberg, and the foundering of a nearby vessel. Each scenario requires T-ARTUE to learn goals to respond effectively. We created oracles that automatically respond to T-ARTUE's queries, and used them in a set of online learning tests. We ran 10 repetitions per scenario, each with a different random seed.

Figure 5 (top) shows the average percentage of this "optimum" performance attained by T-ARTUE per scenario, using the same random seeds, throughout training. For each trial, the oracle provided criticism each time T-ARTUE erred. As shown, T-ARTUE maintains 90% of optimal performance after only 5 trials in Sub Hunt and after 15 trials in Iceberg.

A mixed-initiative system is often more useful if it requires less attention from the human collaborator. Therefore, we investigated the effect of lowering the probability p that the expert would provide criticism by repeating the experiments with p={0, .2, .6, 1.0}. For example, with a setting of .6, an expert "notices" and criticizes T-ARTUE when it chooses a non-optimal goal with probability p=.6. Otherwise, the error was ignored.

The lower two graphs in Figure 5 display the results. Lower amounts of criticism increase the time required to attain the same level of performance, although long term performance is not severely affected. However, the lack of any criticism can prevent T-ARTUE from attaining a high performance level, and can even cause catastrophic failure. This is due to an incorrect assumption made early on that cannot be corrected through the query processes.

To determine how much expert interaction was required, we examined the number of queries T-ARTUE per training repetition. For Iceberg, it averaged 11.2 queries. This decreased quickly as time progressed: 90% of these occurred during the first ten learning trials. This indicates that T-ARTUE can survive on its own fairly quickly. However, Sub Hunt required 26.9 queries on average, and they were distributed more evenly throughout the trial. Examining the causes of this is a topic of future research.

7. Conclusions

Our study demonstrates that an agent can learn goal selection knowledge for immediate use in an online setting. This is useful for goal reasoning agents that must respond

effectively to unexpected states in dynamic environments. Our results show that active and interactive learning techniques allow our agent to perform comparably to when its knowledge is manually crafted, given access to a sufficiently attentive expert. Our tests also show that a high level of expert attentiveness is needed to guarantee good performance. Our future work will include improving the generalization algorithms so that a lower level of attentiveness suffices to constrain the concepts learned by T-ARTUE. Finally, we will also examine other means to learn goal selection knowledge in goal reasoning agents.

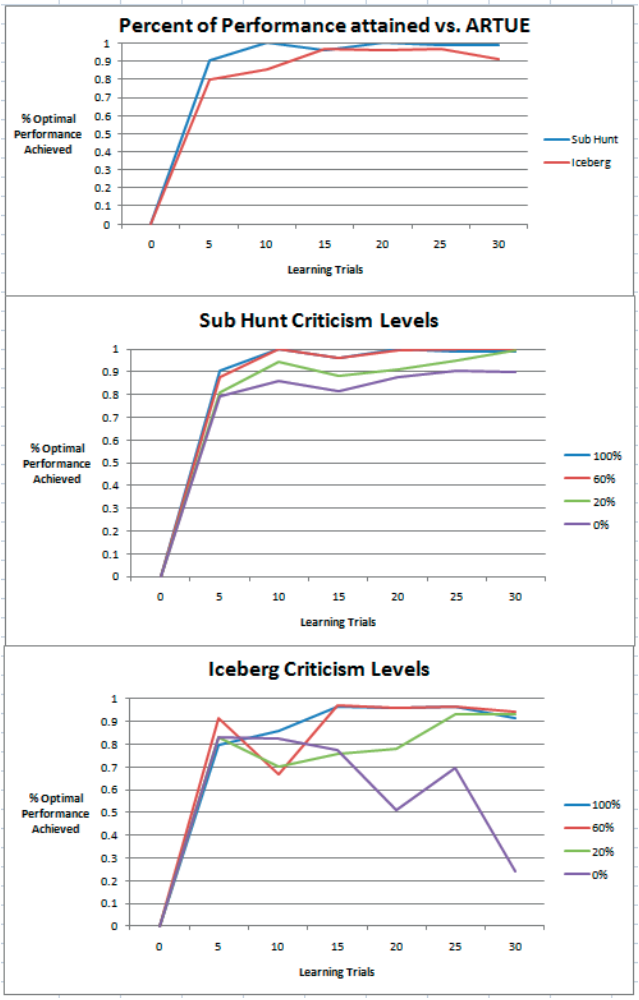

Figure 5: Empirical Results for T-ARTUE

References

Agrawal, R., & Srikant, R. (1994). Fast algorithms for mining association rules. *Proceedings of the 20th International Conf. on Very Large Data Bases* (pp. 487-499). Santiago de Chile, Chile: Morgan Kaufmann.

Aha, D.W., Klenk, M., Muñoz-Avila, H., Ram, A., & Shapiro, D. (Eds.) (2010). *Goal-Directed Autonomy: Notes from the AAAI Workshop (W4)*. Atlanta, GA: AAAI Press.

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

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

Dal Lago, U., Pistore, M., & Traverso, P. (2002). Planning with a language for extended goals. *Proceedings of the 18th National Conference on Artificial Intelligence* (pp. 447-454). Edmonton (Alberta), Canada: AAAI Press.

de Kleer, J. (1986). Problem solving with the ATMS. *Artificial Intelligence*, 28(2), 197-224.

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

Firby, R.J. (1987). An investigation into reactive planning in complex domains. *Proceedings of the Sixth National Conference on Artificial Intelligence* (pp. 202-206). Seattle, WA: AAAI Press.

Hu, R., Delaney, S.J., & Mac Namee, B. (2010). EGAL: Exploration guided active learning for TCBR. *Proceedings of the 18th International Conference on CBR* (pp. 156-170). Alessandria, Italy: Springer.

Laird, J.E., & Rosenbloom, P.S. (1990). Integrating execution, planning, and learning in Soar for external environments. *Proceedings of the Eighth National Conference on Artificial Intelligence* (pp. 1022-1029). Boston, MA: AAAI Press.

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

Muñoz-Avila, H., Jaidee, U., Aha, D.W., & Carter, E. (2010). Goal-driven autonomy with case-based reasoning. *Proceedings of the Eighteenth International Conference on Case-Based Reasoning* (pp. 228-241). Alessandria, Italy: Springer.

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.

Ram, A., & Leake, D. (1995). Learning, goals, and learning goals. In A. Ram & D. Leake (Eds.) *Learning, goals, and learning goals.* Cambridge, MA: MIT Press.

Sculley, D. (2007). Online active learning methods for fast label-efficient spam filtering. In *Proceedings of the Fourth Conference on Email and Anti-Spam*. Mountain View, CA: [http://www.ceas.cc/2007].

Settles, B. (2009). *Active learning literature survey* (Technical Report 1648). Madison, WI: University of Wisconsin, Department of Computer Science.

Weber, B.G., Mateas, M., & Jhala, A. (2010). Case-based goal formulation. In (Aha et al. 2010).