Proceedings of the 2002 IEEE/RSJ
Intl. Conference on Intelligent Robots and Systems
EPFL, Lausanne, Switzerland • October 2002

A Hybrid Cognitive-Reactive Multi-Agent Controller

Magdalena D. Bugajska[1], Alan C. Schultz[1], J. Gregory Trafton[1], Matthew Taylor[1], Farilee E. Mintz[2]

[1]*Naval Research Laboratory (NRL), Washington, DC, USA, {magda, schultz, trafton, mtaylor}@aic.nrl.navy.mil*
[2]*ITT Industries, AES Division, Alexandria, VA, USA, mintz@aic.nrl.navy.mil*

Abstract

The purpose of this paper is to introduce a hybrid cognitive-reactive system, which integrates a machine-learning algorithm (SAMUEL, an evolutionary algorithm-based rule-learning system) with a computational cognitive model (written in ACT-R). In this system, the learning algorithm handles reactive aspects of the task and provides an adaptation mechanism, while the cognitive model handles cognitive aspects of the task and ensures the realism of the behavior. In this study, the controller architecture is used to implement a controller for a team of micro-air vehicles performing reconnaissance and surveillance.

1. Introduction

This research focuses on the design and implementation of intelligent autonomous agents, or teams of agents, which can be easily integrated into mixed-initiative human-agent teams. Successful integration is supported by use of common representation between human and artificial team members, and cognitively plausible agent behaviors. The adaptation of behaviors to changes in the environment and the capabilities of the team has to be permitted as well. This study addresses some of these issues by creating an architecture that supports cognitively plausible behaviors of a team of intelligent unmanned vehicles.

The purpose of this research is to merge a cognitive model and a reactive system into a hybrid control architecture for autonomous agents. In this study, SAMUEL, an evolutionary algorithm-based reactive rule-learning system [13], is integrated with a cognitive model written in ACT-R, a computational cognitive architecture [5]. In this hybrid system, the learning algorithm handles reactive aspects of the task such as local navigation and obstacle avoidance, and provides an adaptation mechanism. The cognitive model handles higher-level aspects of the task such as planning and cognition, and ensures the cognitive realism of the behavior.

For this study, the hybrid architecture was adapted to provide a controller for a group of micro air vehicles whose task was reconnaissance and surveillance. The following sections of the paper discuss related work in cognitive science and robotics, and describe the hybrid architecture. The description of the task domain and the experimental approach along with current results are provided as well.

2. Related Work

In early robotics work, the architectures tended to be hierarchical, stressing the classical sense-think-act cycle. One example of this approach is the earlier work of Albus on reference model architecture for real-time intelligent control systems (ARTICS) [1]. Later, behavior-based approaches stressed rapid reaction to the environment and tried to model the sense-react cycle of insects and animals. This work is best exemplified in the subsumption architecture [8] where all behaviors have direct access to sensors, and each can produce direct actions in the world.

Many researchers saw the need to incorporate both approaches into their architectures. The current popular approach adapts something similar to a multi-level architecture with reactive behaviors or skills at the lower levels, and more traditional planning systems at the upper layer. Between these are additional mechanisms, which attempt to bridge the two approaches. This middle layer typically involves sequencing of skills based on goals produced by the planner. The 3T architecture [7], for example, uses reactive action plans (RAPs) [12] to instantiate the preconditions, skills, and stopping criteria to achieve the planner's goals. Albus's more recent work has extended the real-time control system (RCS) to incorporate different levels of reactivity into his architecture, moving it from a hierarchical to a heterarchical architecture [2], [3].

Our low-level controller, SAMUEL, implements the behaviors or skills, which it learns as stimulus-response rules [13]. This representation is derived from behaviorist tradition as mentioned above. SAMUEL's strength lies in its ability to learn relatively simple condition-action rules to solve complex tasks. SAMUEL and other evolutionary algorithm-based reinforcement learning systems [15] are good at learning reactive strategies for sequential decision problems, but cannot take advantage of the higher-level information that facilitates cognition.

Computational cognitive modeling is primarily concerned with building running models that not only imitate what a person does but also how they do it. There are several modeling languages prevalent in the literature today: ACT-R [5], Soar [16] and EPIC [14]. Each of these modeling languages has a different emphasis and, thus, different strengths and weaknesses [11]. Our work will focus on ACT-R. ACT-R is a production system language that allows creation of models based on human cognition. ACT-R has an excellent track record of providing fits to human data in a wide variety of domains including memory for goals [4], human computer interaction [6], and scientific discovery [20]. Most current work in ACT-R has focused on high-level cognition and much less work has been done on reactive skill modeling and learning [17].

Our hypothesis is that an integration of SAMUEL with ACT-R will create a robust control architecture that combines the best of both reactivity and high-level cognition (e.g. planning), with learning at both the reactive level and at the cognitive level.

3. Hybrid Architecture

To facilitate the implementation of the cognitive layer of the controller, ACT-R, a computational cognitive modeling system, was used. SAMUEL, an evolutionary algorithm-based rule learning system, was used for implementation of the reactive layer of the controller. The two layers of the controller communicated with each other using Unix Inter-Process Communication protocols (shared memory and message queues). This design allowed for independent decision cycles for low- and high-level behaviors as well as distributed implementation.

3.1 Cognitive Layer of the Controller

A cognitive model implemented using ACT-R augmented the cognitive layer of the controller. ACT-R is a production-system architecture based on condition-action rules, which execute the specified actions when the specified conditions are met. ACT-R was chosen for this research because it provides a rigorous framework for cognitive modeling as well as a set of built-in parameters and constraints on cognition to facilitate a priori predictions about behaviors and more psychologically plausible models [5].

Building a cognitive model, which is to serve as an agent's controller, consists of implementing the declarative and procedural knowledge of the domain and the task. The declarative knowledge is implemented using structures called chunks, which contain information about the current perceived state of the world as well as facts related to the domain. Examples of declarative knowledge include *"There is an object in front of agent number 5"*, *"The object in front of agent number 5 is green"*, and *"Tanks are green."* The procedural knowledge is implemented as sets of productions where each production is a condition-action pair. The condition specifies what must be known for the production to apply and the action specifies the things to do if the production applies. The conditions test the knowledge contained in declarative memory, while the actions modify the declarative memory or perform physical actions. Examples of procedural knowledge (production rules) include *"IF the goal is to find objects THEN send an agent to search for objects"*, *"IF the goal is to search for objects and object has been found THEN determine the type of an object"*, and *"IF the goal is to determine the type of an object and the color of the object is green THEN the object is a tank."* Given this set of productions and chunks from the previous example, the model would be able to deduce that a tank has been found.

ACT-R implements a fixed-attention architecture, which means that at any point in time, it is focused on a single goal and only one production can fire during each cycle. The actions of productions can create new goals as well as change the focus, which allows the system to support multiple tasks in a single model. ACT-R models much of the qualitative structure of human cognition at a subsymbolic level (also referred to as "rational" level) rather than using the symbolic structures described above. The subsymbolic level contains quantities that participate in neural-like processes, which determine the chunk and production access in memory. It also has a set of learning processes that can modify those subsymbolic quantities.

3.2 Reactive Layer of the Controller

SAMUEL was used to implement the reactive layer of the controller. SAMUEL is a machine learning system that uses genetic algorithms (GAs), reinforcement learning, and Lamarckian learning to solve sequential decision problems. SAMUEL is designed for handling problems in which feedback is delayed (payoff occurs only at the end of an episode that spans many decision steps). This learning system has been previously used to learn behaviors such as local navigation and collision avoidance for an autonomous underwater vehicle [18], shepherding [19], and tracking and herding for mobile robots. The original system implementation is described in detail in [13].

SAMUEL implements behaviors as a collection of stimulus-response rules. Each stimulus-response rule consists of conditions that match against the current sensors of the autonomous vehicle, and an action that defines action to be performed by it. An example of a rule might be:

```
RULE 4
 IF   range2 > 25
     AND  range5 > 0
     AND  bearing = 0
      THEN SET turn = 0
```

This rule should be interpreted as follows: if the MAV's range sensor 2 is returning a value greater than 25 units, the range sensor 5 is sensing anything, and the MAV is going towards the goal, then the MAV should go straight. Each rule has an associated strength with it as well as a number of other statistics. During each decision cycle, all rules that match the current state are identified. Conflicts are resolved in favor of the rules with higher strength. Rule strength is updated based on the reward received after each training episode.

4. Experimental Details

In this study, the hybrid architecture (Section 3) was used to develop a system to control a team of simulated micro air vehicles whose task was to perform reconnaissance and surveillance. Each vehicle was able to detect obstacles, including other MAVs, and certain ground features below the vehicle within a defined range. As a group, the MAVs needed to maximize the information gain, concentrating on areas of more importance, and minimizing duplication of effort. In previous work, we successfully used GAs to evolve MAV control behaviors that could accomplish this task [22], [9].

4.1 Simulation

The Micro Air Vehicle Simulator (MAVSIM) includes a simple 2D model of the MAV's motion, sensors, and the environment. The sensors modeled for this study include eight range sensors (one for each compass direction), each of which outputs a floating-point values representing distance to the nearest obstacle or fellow MAV in that direction, and a "vision" sensor, which provides the information (interest level, size, mobility, etc.) about the ground features beneath the vehicle within its sensing range. The MAV's environment (Figure 1) consists of static as well as dynamic regions of varied military interest which model real world features such as roads, buildings, ground vehicles, etc., as well as static obstacles. The interest regions are only visible to the controller if they have been sensed by a MAV's vision sensor (i.e. if a MAV has flown over it). MAVs are destroyed either when they leave the flight zone (i.e., fly off the screen) or when they collide with fellow MAVs or obstacles.

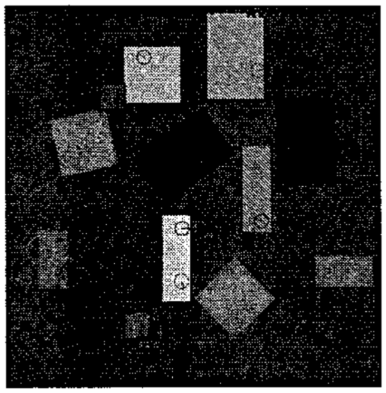

Figure 1: Screenshot of MAVSIM interface showing the plan view of the environment. Regions of interest (rectangles of various shades of gray or red), obstacles (black obstacles) and MAVs (circles) are shown.

The goal of the task was to maximize the score, which was determined as follows. An individual MAV's instantaneous value was equal to the sensed area weighted by the interest of the visible regions within the sensor. If the sensor only partially covered an area on interest, it would obtain a lower value than if it sat completely over the area of interest. The average score, the one to be maximized, was the total value of all MAVs averaged over time. Note that an area of interest could not be accumulated by more than one MAV in the same instant of time, i.e. only one MAV would receive credit if two or more sensors overlapped on some portion of an area of interest.

4.2 Controller

In this system, SAMUEL controlled the behavior of individual MAVs, while the cognitive model was responsible for the team behavior.

The cognitive model implemented in ACT-R was based on the data collected during human-subject experiments performed at NRL and described in greater detail in [21]. In those experiments, the human operators would control the MAVs by directing them to goal locations using a point-and-click interface to the simulator. In this study, ACT-R, just like a human operator, was responsible for providing 2D goals to individual MAVs based on the current perception of the world. ACT-R's perception of the environment was closely matched to the perception of the human operator. ACT-R could "see" the position and state of all MAVs, and the position and value of discovered regions of interest. This perceptual information was represented as ACT-R chunks, just as ACT-R/PM (perceptual-motor) does [10]. The top-level goal of the model was to perform the task, which required both exploration of the area as well as surveillance of the discovered regions.

The model begins by sending the MAVs to explore the unexplored regions of the environment. When a region of interest is discovered, the model delegates MAVs to determine the boundaries of the region. If the discovered regions are of high enough interest (based on the amount of explored territory and time left), the model positions MAVs on them. Once the MAV is positioned on the region of interest, it will be moved only if there exists an unoccupied region with higher interest value.

In this study, SAMUEL was used to evolve stimulus-response rules to perform the collision-free navigation behavior for the simulated MAVs. Each MAV used the same behavior evolved by SAMUEL in conjunction with the goals provided by ACT-R to safely navigate to a specified location. The current MAV sensor information was mapped to the conditions of the stimulus-response rules. The activated rule specified the action of the vehicle. SAMUEL's conditions included *range0* - *range7* representing the eight MAV's range sensor readings (values between 0 and 50 units in 5-unit increments), and *range* (values between 0 to 1200 in 10-unit increments) and *bearing* (values between 0 and 345 degrees in 45-degree increments) to the goal. The *turn_rate* action, which specified the MAV's turning angle per decision cycle (values between –180 to 180 degrees in 45-degree increments), was the only allowed action.

In summary, ACT-R performed the high-level thinking and reasoning while SAMUEL controlled the low-level mobility. ACT-R controlled exploration of the environment (reconnaissance) and the allocation of MAVs over the regions of interest (surveillance). In contrast, SAMUEL controlled the local navigation and collision avoidance of individual MAVs.

5. Results

In this section, the performance of the hybrid architecture is compared to the performance of human operator and SAMUEL controller performing the same task are discussed. The SAMUEL learning results are presented first for completeness.

5.1 Reactive Behavior Learning Results

In order to allow us to focus on the important aspects of the collision-free navigation behavior, the environment in which the behavior was evolved differed from the environment used for the task of surveillance and reconnaissance. The learning environment contained no interest regions, but it contained significantly more obstacles. Also, to speed up and simplify the implementation, there was only one MAV present in the environment during the episode. Others were simulated as small static obstacles.

During each simulation run, the MAV was given a single goal location, which was at least half of the width of the environment away. The positions of the obstacles and the goal as well as the initial position of the MAV were generated randomly for each episode. Each learning evaluation consisted of a maximum of 250 decision cycles at the end of which the behavior was evaluated. The fitness function used in this study was defined as in Figure 2.

$$f(x) = \begin{cases} 0.0\text{-}0.3, & \text{if MAV crashed, based on the distance away from the goal} \\ 0.5\text{-}1.0, & \begin{cases} 0.5\text{-}0.8, & \text{if MAV survived, based on the distance away from the goal} \\ +0.2, & \text{if MAV made it to the goal} \end{cases} \end{cases}$$

Figure 2: Task fitness function.

The learning experiment was allowed to run for 100 generations with a population of 100 rulebases. For each individual evaluation, 40 runs of the simulator were performed in order to provide the learning system with statistics about rulebase's performance for Lamarckian learning, rule strength updates, as well as the genetic algorithm. The system was initialized with three heterogeneous sets of rules, which implemented behaviors including random walk, simple obstacle avoidance, and local navigation.

Every generation, the best rule set from the current population (based on the fitness function) was evaluated 100 times in randomly generated environments. The values of these evaluations are plotted in Figure 3. As seen in this figure, the fitness of the best behavior was at approximately 0.61, which shows a considerable improvement over the initial behavior (0.229).

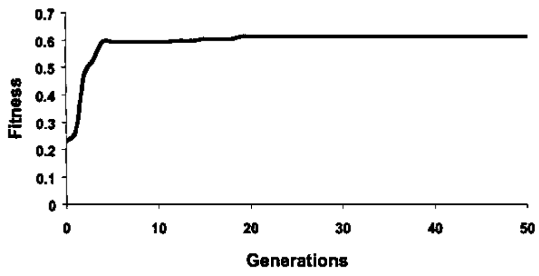

Figure 3: Average payoff (over 100 trials) of the best individuals throughout first 50 generations tested in learning environment.

5.2 Controller Performance Results

The task performance metrics used to evaluate the controller include the MAV survival rate defined as the number of MAVs controllable at the end of a trial and the total score calculated as an average MAV group score over the length of the trial (Section 4.1).

For this study, the performance of the hybrid controller was compared against the human controller [21] and SAMUEL controller performance [9] of the task across different MAV group sizes. There were two group sizes considered. The smaller MAV group consisted of six agents while the larger MAV group consisted of sixteen agents.

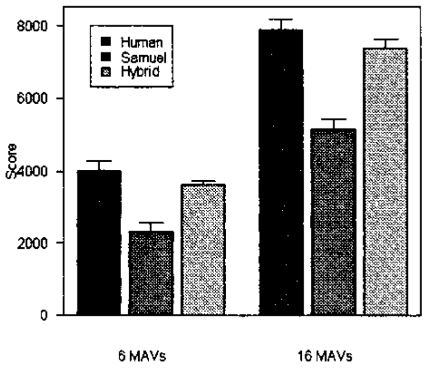

Figure 4: The total score. The graph shows the performance as measured by the total score across different controllers (from the left: Human, SAMUEL, and Hybrid) and different MAV group sizes (from the left: 6 and 16 MAVs). The error bars show standard error of means.

As the results show (Figures 4 and 5), the hybrid controller obtained an average score of 7367.67 with the larger MAV team and 3626.9 with the smaller team. The average MAV survival rates for this controller were 14.8 MAVs for the larger team and 5.36 for the smaller team. It can be seen that the hybrid controller obtained a better score than the SAMUEL controller independent of the group size, even though the SAMUEL controller maintained slightly higher survival rates for smaller MAV group than the hybrid. The hybrid controller's score as well as the survival rates were comparable to human controller's performance independent of the group size. Once the quality of the evolved collision-free navigation behavior is improved, the hybrid controller should outperform the human controller.

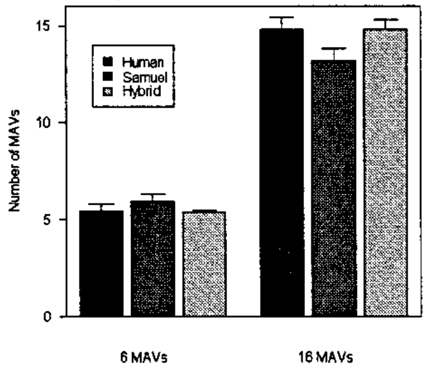

Figure 5: The MAV survival rate. The graph shows the performance as measured by the number of MAVs surviving at the end of the trial across different controllers (from the left: Human, SAMUEL, and Hybrid Architecture) and different MAV group sizes (from the left: 6 and 16 MAVs). The error bars show standard error of means.

6. Conclusions and Future Work

This paper presented hybrid cognitive-reactive control architecture for autonomous vehicles. The deliberative module of traditional robotics architecture was augmented by a cognitive model of the task. We believe our architecture will allow for more intelligent, more capable, and robust autonomous agents. Our architecture should also allow for easier integration of the agents into mixed-initiative human-agent teams.

A detailed description of the study was given in which the hybrid architecture was used to implement a controller for a distributed team of micro air vehicles. Even though the performance of the hybrid controller was shown to be only comparable with the performance of the human controller, it does seem to capture some of the human's behavior and performance. This comparability suggests that our hybrid model is adequately modeling the humans' high-level cognitive functions, as well as the low level reactive aspects.

The immediate research will focus on increasing the quality of the evolved behavior for collision-free navigation for the MAVs in order to improve the performance of the hybrid controller as compared to a human operator. An attempt will also be made to examine how much autonomy is appropriate for a given agent within a team. We will explore autonomy by examining collision avoidance, local navigation, full route planning, situation awareness, and complete mission planning. Our hybrid architecture will allow us to explore issues of reactivity and cognitive complexities in a straightforward manner.

Acknowledgments

This work was supported by the DARPA CoABS program and the Office of Naval Research under work order number N0001402WX20374.

References

[1] Albus, J. S., R. Quintero, R. Lumia., M. Herman, R. Kilmer., K. Goodwin (1991). "It is Time to Adopt a Reference Model Architecture for Real-Time Intelligent Control Systems (ARTICS)," ASME Manufacturing Review.

[2] Albus, J. A. (1997). "The NIST Real-time Control System (RCS) An Approach to Intelligent Systems Research," Special Issue of the Journal of Experimental and Theoretical Artificial Intelligence 9, pgs 157-174.

[3] Albus, J.S. (1999). "4-D/RCS Reference Model Architecture for Unmanned Ground Vehicles," Proceedings of the SPIE AeroSense Technical Conference 3693, Orlando, FL.

[4] Altmann, E. M. And J. G. Trafton (2002). "An activation-based model of memory for goals." Cognitive Science, 39-83.

[5] Anderson, J. R. and C. Lebiere (1998). The Atomic Components of Thought. Mahwah, NJ: Lawrence Erlbaum.

[6] Anderson, J. R., M. Matessa, and C. Lebiere (1997). "ACT-R: A theory of higher level cognition and its relation to visual attention." Human-Computer Interaction, 12 (4), 439-462.

[7] Bonasso, R. P., R. J. Firby, E. Gat, D. Kortenkamp, D. P. Miller, and M. Slack (1997). "Experiences with an Architecture for Intelligent, Reactive Agents," Journal of Experimental and Theoretical Artificial Intelligence, 9(1).

[8] Brooks, R.A (1991). "Intelligence without representation." Artificial Intelligence, 47, pp. 139-159.

[9] Bugajska, M. D, A. C. Schultz, J. G. Gregory, S. Gittens, and F. Mintz (2001). "Building Adaptive Computer Generated Forces: The Effect of Increasing Task Reactivity on Human and Machine Control Abilities." In Late-Breaking Papers Proceedings of GECCO-2001, San Francisco, CA.

[10] Byrne, M. D. and J. R. Anderson (1998). "Perception and action." In J. R. Anderson and C. Lebiere (Eds.) The Atomic Components of Thought, (pp. 167-200). Mahwah, NJ: Lawrence Erlbaum.

[11] Chong, R. S. (in press). Unified theories of cognition. In Encyclopedia of Cognitive Science.

[12] Firby, R. J. (1989). "Adaptive Execution in Complex Dynamic Domains," Ph.D. Thesis, Yale University Technical Report YALEU/CSD/RR #672.

[13] Grefenstette, J., C. L. Ramsey and A. C. Schultz (1990). "Learning sequential decision rules using simulation models and competition." Machine Learning, 5(4), 355-381, October 1990, Kluwer.

[14] Kieras, D. and D. E. Meyer (1997). "An overview of the epic architecture for cognition and performance with application to human-computer interaction." Human Computer Interaction, 12, 391-438.

[15] Moriarty, D. E., A. C. Schultz, and J. J. Grefenstette (1999). "Evolutionary Algorithms for Reinforcement Learning." Journal of Artificial Intelligence Research, 11: 199 -229.

[16] Newell, A. (1990). Unified theories of cognition. Cambridge, MA: Harvard University Press.

[17] Salvucci, D. D., E. R. Boer, and A. Liu (2001). "Toward an Integrated Model of Driver Behavior in a Cognitive Architecture." In *Transportation Research Record*, No. 1779.

[18] Schultz, A. C. (1991). "Using a Genetic Algorithm to Learn Strategies for Collision Avoidance and Local Navigation," Proceedings of the Seventh International Symposium on Unmanned Untethered Submersible Technology, 213-225, University of New Hampshire, September 23-25.

[19] Schultz, A.C., J. J. Grefenstette, and W. Adams (1996). "Robo-Shepherd: Learning Complex Robotic Behaviors," Proc. Of the International Symposium on Robotics and Automation, Robotics and Manufacturing: Recent Trends in Research and Applications, Volume 6, (Eds. Mohammad Jamshidi, Francois Pin, and Pierre Dauchez), ASME Press: New York, 763-768.

[20] Schunn, C. D. and J. R. Anderson (1998). "Scientific discovery." In J. R. Anderson, & C. Lebiere (Eds.), Atomic Components of Thought. Mahwah, NJ: Erlbaum.

[21] Trafton, J. G, A. C. Schultz, M. D. Bugajska, S. Gittens, and F. Mintz (2001). "An Investigation of How Humans and Machines Deal with Increases in Reactivity." In Proceedings of 10th Conference on Computer Generated Forces and Behavioral Represenation (CGF&BR), Norfolk, VA.

[22] Wu, A. Schultz and A. Agah (1999). "Evolving Control for Distributed Micro Air Vehicles," proc. Of the IEEE Conference on Computational Intelligence in Robotics and Automation, IEEE, 174-179.