Combining Experience with Quantitative Models

John J. Grefenstette
Connie Loggia Ramsey

Navy Center for Applied Research in AI
Naval Research Laboratory
Code 5514
Washington, DC 20375-5337

Abstract

This is a progress report on our efforts to design intelligent robots for complex environments. The sort of applications we have in mind include sentry robots, autonomous delivery vehicles, undersea surveillance vehicles, and automated warehouse robots. We are investigating the issues relating to machine learning, using multiple mobile robots to perform tasks such as playing hide-and-seek, tag, or competing to find hidden objects. We propose that the knowledge acquisition task for autonomous robots be viewed as a cooperative effort between the robot designers and the robot itself. The robot should have access to the best model of its world that the designer can reasonably provide. On the other hand, some aspects of the environment will be unknown in advance. For such aspects, the robot itself is in the best position to acquire the knowledge of what to expect in its world. We have implemented these ideas in an arrangement we call *case-based anytime learning*. This system starts with a parameterized model of its world and then learns a set of specific models that correspond to the environmental cases it actually encounters. The system uses genetic algorithms to learn high-performance reactive strategies for each environmental model.

1 Introduction

This is a progress report on our efforts to design intelligent robots for complex environments. The sort of applications we have in mind include sentry robots, autonomous delivery vehicles, undersea surveillance vehicles, and automated warehouse robots. We are investigating the issues relating to machine learning, using multiple mobile robots to perform tasks such as playing hide-and-seek, tag, or competing to find hidden objects.

An intelligent robot will need extensive knowledge to interact effectively with its external environment. This challenge represents an important opportunity for machine learning. A key issue is not necessarily how complex the environment is, but how easy it is to provide the robot with the knowledge it needs, given the available knowledge we have. Several previous studies have explored different methods for automating knowledge acquisition for intelligent robots, each approach typically depending on different assumptions about what is already known about the task environment. If the robot's task environment is well understood, it may be most efficient to depend largely on a preprogrammed model of the environment, and to use learning to improve efficiency and reactivity (Laird et al., 1991). If the effects of the robot's actions can not be easily predicted but there are only a few important state variables that affect the robot's decisions, the robot might use an internal model to accelerate its learning of state-action mappings (Sutton, 1990). If the environment is assumed to exhibit perpetual novelty, it might be useful to enable the robot to learn a wide variety of cognitive structures based on low-level perceptual stimuli (Booker, 1988). We believe that in practical cases, there will be a mix of the above cases. Some aspects of the robot's world will be accurately known in advance, and some aspects can only be learned through the experience of the robot.

We propose that the knowledge acquisition task for autonomous robots be viewed as a cooperative effort between the robot designers and the robot itself. Some aspects of the environment will be known in great detail to the designer, for example, the size and weight of the robot, the characteristics of its sensors and effectors, and at least some of the physics of the task environment. Our first principle guiding the cooperative knowledge acquisition effort is:

1. The robot needs all the help it can get.

That is, the robot should have access to the best model of its world that the designer can reasonably provide. This is likely to include a quantitative simulation model of the robot and its environment. On the other hand, some aspects of the environment will be unknown in advance. These aspects could include such things as the frictional characteristics of the floor, the reflective attributes of the wall surfaces, the location of objects and obstacles, and the speed and maneuverability of the other agents in the environment. For such aspects, the robot itself is in the best position to

acquire the knowledge of what to expect in its world. This observation leads to our second principle:

> *2. The robot should use its experience to fill in the missing aspects of its world model.*

Again, this should be a cooperative effort. The designer's responsibility should include identifying the uncertain aspects of the environment, supplying plausible ranges of values, and making sure that the initial model provided to the robot treats these aspects as parameters that can be changed as necessary to reflect the actual environment. The robot can then modify its world model based on its observation of the actual environment. We have put these principles into practice in an arrangement we call *anytime learning* (Grefenstette and Ramsey, 1992; Ramsey and Grefenstette, 1993).

2 Anytime Learning

Anytime learning is a method for continuous learning in changing environments. We call the approach *anytime learning* to emphasize its relationship to recent work on anytime planning and scheduling (Dean and Boddy, 1988; Zweben, Deale and Gargan, 1990). The basic characteristics of anytime algorithms are: (1) the algorithm can be suspended and resumed with negligible overhead, (2) the algorithm can be terminated at any time and will return some answer, and (3) the answers returned improve over time. However, our use of the term *anytime learning* is meant to denote a particular way of integrating execution and learning. The basic idea is to integrate two continuously running modules: an execution module and a learning module (see Figure 1). The agent's learning module contains a simulation model with parameterized aspects of the domain. It continuously tests new strategies against the simulation model to develop improved strategies, and updates the knowledge base used by the agent (in the execution system) on the basis of the best available results. The execution module controls the agent's interaction with the environment, and includes a monitor that can detect changes in the environment, and dynamically modify the parameter ranges of the simulation model (used by the learning module) to reflect these changes. When the simulation model is modified, the learning process is restarted on the modified model. The learning system is assumed to operate indefinitely, and the execution system uses the results of learning as they become available. In short, this approach uses an existing simulation model as a base on which strategies are learned, and updates the parameters of the simulation model as new information is observed in the external environment in order to improve long-term performance.

We are still in the process of elaborating the anytime learning model. One aspect concerns the criteria for deciding when the external environment has changed. Currently, the monitor compares measurable aspects of the environment with the parameters provided by the simulation

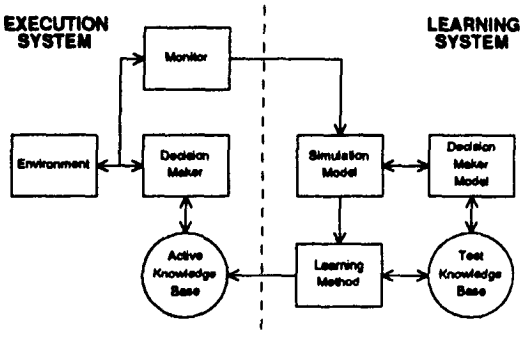

Figure 1. Anytime Learning System

design. In addition, the monitor might also detect differences between the expected and actual performance of the current strategy in the environment.[1] For example, if the performance level degrades in the environment, that is a sign that the current strategy is no longer applicable. If the performance of the current strategy improves unexpectedly, it may indicate that the environment has changed, and that another strategy may perform even better.

3 Case-Based Initialization of Genetic Algorithms

While this approach could be used merely to fill in particular values for parameters that are initially unknown, it is especially useful in changing environments. Each observed set of values for the parameters can be treated as an environmental *case*. The learning system can store strategies for different cases, indexed by the environmental parameters that characterize that case. When the environment changes, the set of previous cases can be searched for similar cases, and the corresponding best strategies can be used as a starting point for the new case.

Our particular instantiation of anytime learning uses a learning system called SAMUEL (Cobb and Grefenstette, 1991; Gordon, 1991; Grefenstette, 1991; Grefenstette, Ramsey and Schultz, 1990; Schultz, 1991). SAMUEL is a system for learning reactive strategies expressed as symbolic condition-action rules, given a simulation model of the environment. It uses a modified genetic algorithm and reinforcement learning to generate increasingly competent strategies. SAMUEL has successfully learned strategies for a number of multi-agent tasks in (simulated) static environments, including evading attackers, tracking other agents at a distance, and dogfighting. While the basic ideas of anytime learning could be applied using a number of other learning methods, especially other reinforcement learning methods, SAMUEL has some important advantages for this approach. In particular, SAMUEL can learn rapidly from

[1] Hart, Anderson and Cohen (1990) discuss related issues concerning the design of planners that monitor differences between expected and actual progress of a plan.

partially correct strategies and with limited fidelity simulation models (Schultz and Grefenstette, 1990; Ramsey, Schultz & Grefenstette, 1990). More importantly for this discussion, the genetic algorithms provide an effective mechanism for modifying previously learned cases. When an environmental change is detected, the genetic algorithm is restarted with a new initial population. This initial population can be "seeded" with the best strategies found for previous similar cases. We call this *case-based initialization* of the genetic algorithm. Our studies have shown that by using good strategies from several different past cases as well as exploratory strategies, default strategies and members of the previous population, the genetic algorithm can respond effectively to environmental changes, and can also recover gracefully from spurious false alarms (i.e., when the monitor mistakenly reports that the environment has changed).

A case-based anytime learning system can be viewed as one that starts with an underspecified model of its world (the original parameterized quantitative model), and then learns, on the basis of experience, a set of fully specified models that correspond to the environmental cases it actually encounters. The power of this approach derives from the cooperative effort between designer and robot. The designer provides a rich sets of models that can be used for learning, and the robot selects the appropriate models from that (possibly infinite) set of models. Each partner makes a significant contribution to the knowledge acquisition process.

4 Current Results

Preliminary experiments have shown the effectiveness of anytime learning in a changing environment (Grefenstette and Ramsey, 1992; Ramsey and Grefenstette, 1993). The task used in these experiments was a cat-and-mouse game in which one robot had to track another without being detected. The changing environmental parameters included the speed distribution of the target agent and the maneuverability of the target agent, as well as environmental variables that were in fact irrelevant to the performance of the task. The most promising aspect of these results is that, within each time period (epoch) after an environmental change, the case-based anytime learning system consistently showed a much more rapid improvement in the performance of the execution system than a baseline learning system (with its monitor disabled). Case-based initialization allows the system to automatically bias the search of the genetic algorithm toward relevant areas of the search space. More recent experiments show that the longer the epochs last and the longer the system runs and gathers a base of experiences of different environmental cases, the greater the expected benefit of case-based anytime learning is.

5 Future Plans

So far we have tested the anytime learning approach on simulated environments only. We have recently acquired two mobile robots from Nomadic Technologies, Inc. The robots are equipped with several sensor packages, including contact sensors, infrared, sonar, and structured light rangefinders. Our experiments will involve the same kind of cat-and-mouse tasks described above. A significant feature of the Nomadic robots is that they come equipped with a complete simulation model, allowing the designer to test control strategies in simulation before testing on the real robot. Our approach calls for putting these simulations to work as the learning test bed for the robots, exactly as described above. We will soon be able to report our initial findings on the physical robots.

The general approach described here is not limited to mobile robots, but could be used in a variety of other settings. There are many simulation models used for training human operators, for example, flight simulators, air traffic control simulators, and power plant simulators. Many aspects of such simulators, such as weather conditions, terrain characteristics, even the skill levels of the computer generated agents, are usually parameterized to allow a range of training experiences or to model probabilistic events. Besides training simulators, there are simulators that have been used to evaluate intelligent control systems prior to deployment in an operational setting (Fogarty, 1989). Both training simulators and testing simulators usually include an evaluation mechanism for the decisions made during the simulation run. These evaluation mechanisms provide feedback that can be used by autonomous learning agents. The approach described here could be used in many of these settings, allowing an automated power plant, for example, to learn from its own simulation model, and to adjust that model to reflect changing environmental conditions. We expect that the combination of experience and flexible quantitative models can reduce the overall knowledge acquisition effort required to produce effective autonomous systems.

References

Booker, L. B. (1988). Classifier systems that learn internal world models. *Machine Learning 3(3)*, 161-192.

Cobb, H. G. and J. J. Grefenstette (1991). Learning the persistence of actions in reactive control rules. *Proceedings of the Eighth International Machine Learning Workshop* (pp. 293-297). Evanston, IL: Morgan Kaufmann.

Dean, T. and M. Boddy (1988). An analysis of time-dependent planning. *Proceedings of the Seventh National Conference on AI (AAAI-88)* (pp. 49-54). St. Paul, MN: Morgan Kaufmann.

Fogarty, T. (1989). The machine learning of rules for combustion control in multiple burner installations.

Proceedings of the Fifth IEEE Conference on AI Applications (pp. 215-221).

Gordon, D. F. (1991). An enhancer for reactive plans. *Proceedings of the Eighth International Machine Learning Workshop* (pp 505-508). Evanston, IL: Morgan Kaufmann.

Grefenstette, J. J. (1991). Lamarckian learning in multi-agent environments. *Proceedings of the Fourth International Conference of Genetic Algorithms* (pp 303-310). San Diego, CA: Morgan Kaufmann.

Grefenstette, J. J. and C. L. Ramsey (1992). An approach to anytime learning. *Proceedings of the Ninth International Conference on Machine Learning* (pp 189-195), D. Sleeman and P. Edwards (eds.), San Mateo, CA: Morgan Kaufmann.

Grefenstette, J. J., C. L. Ramsey and A. C. Schultz (1990). Learning sequential decision rules using simulation models and competition. *Machine Learning 5(4)*, 355-381.

Hart, D. M., S. Anderson and P. R. Cohen (1990). Envelopes as a vehicle for improving the efficiency of plan execution. *Proceedings of a Workshop on Innovative Approaches to Planning, Scheduling and Control* (pp. 71-76). San Diego: Morgan Kaufmann.

Laird, J. E., E. S. Yager, M. Hucka and C. M. Tuck (1991). Robo-Soar: An integration of external interaction, planning, and learning using Soar. *Robotics and Autonomous Systems 8(1-2)*, 113-129.

Ramsey, C. L. and J. J. Grefenstette (1993). Case-based initialization of genetic algorithms. To appear in the *Proceedings of the Fifth International Conference on Genetic Algorithms.*

Ramsey, C. L., A. C. Schultz and J. J. Grefenstette (1990). Simulation-assisted learning by competition: Effects of noise differences between training model and target environment. *Proceedings of the Seventh International Conference on Machine Learning* (pp 211-215). Austin, TX: Morgan Kaufmann.

Schultz, A. C. (1991). Using a genetic algorithm to learn strategies for collision avoidance and local navigation. *Seventh International Symposium on Unmanned, Untethered, Submersible Technology* (pp 213-225). Durham, NH.

Schultz, A. C. and J. J. Grefenstette (1990). Improving tactical plans with genetic algorithms. *Proceedings of IEEE Conference on Tools for AI 90* (pp 328-334). Washington, DC: IEEE.

Sutton, R. S. (1990). Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. *Proceedings of the Seventh International Conference on Machine Learning (ML-90)*, Porter, B. W. and R. J. Mooney, eds., (pp 216-224). Austin, TX: Morgan Kaufmann.

Zweben, M., M. Deale and R. Gargan (1990). Anytime rescheduling. *Proceedings of a Workshop on Innovative Approaches to Planning, Scheduling and Control* (pp. 251-259). San Diego: Morgan Kaufmann.