Improving Tactical Plans with Genetic Algorithms

Alan C. Schultz John J. Grefenstette

Navy Center for Applied Research in Artificial Intelligence (Code 5514),
Naval Research Laboratory, Washington, DC 20375-5000, U.S.A.
EMAIL: schultz@aic.nrl.navy.mil
(202) 767-2685

Abstract

The problem of learning decision rules for sequential tasks is addressed, focusing on the problem of learning tactical plans from a simple flight simulator where a plane must avoid a missile. The learning method relies on the notion of competition and employs genetic algorithms to search the space of decision policies. In the research presented here, the use of available heuristic domain knowledge to initialize the population to produce better plans is investigated.

Introduction

This paper addresses problems of learning rules for sequential decision tasks. Sequential decision tasks may be characterized by the following general scenario: A decision making agent interacts with a discrete-time dynamical system in an iterative fashion. At the beginning of each time step, the agent observes a representation of the current state and selects one of a finite set of actions, based on the agent's decision rules. As a result, the dynamical system enters a new state and returns a (perhaps null) payoff. This cycle repeats indefinitely. The objective is to find a set of decision rules that maximizes the expected total payoff.[1] Several sequential decision tasks have been investigated in the machine learning literature, including pole balancing (Selfridge, Sutton & Barto, 1985), gas pipeline control (Goldberg, 1983), and the animat problem (Wilson, 1985; Wilson, 1987). For many problems, including the one considered here, payoff is delayed in the sense that non-null payoff occurs only at the end of an episode that may span several decision steps. In fact, the paradigm is quite broad since it includes any problem solving task by defining the payoff to be positive for any goal state and null for non-goal states (Barto, Sutton & Watkins, 1989).

For many interesting sequential decision tasks, there exists neither a database of examples nor a complete and tractable domain theory. In these cases, one method for manually developing a set of decision rules is to test a hypothetical set of rules against a simulation model of the task environment, and to incrementally modify the decision rules on the basis of the simulated experience. This paper presents an approach toward using machine learning to automate the process of learning sequential tasks with a simulation model. In this approach, each decision policy, or *tactical plan*, is represented as a set of condition-action rules. A simulation of the sequential decision task provides the basis for measuring the performance for any proposed plan, and a genetic algorithm (Holland, 1975) is used to evolve high-performance plans. The approach has been implemented in a system called SAMUEL[2] (Grefenstette, 1989b; Grefenstette, Ramsey & Schultz, 1990).

One of the interesting features of SAMUEL is that, unlike many previous genetic learning systems (Smith, 1980; Goldberg, 1983; Holland, 1986), the knowledge representation consists of symbolic condition-action rules, rather than low-level binary pattern matching primitives. The use of a high level language for rules offers several advantages. First, it is easier to transfer the knowledge learned to human operators. Second, it makes it possible to combine empirical methods such as genetic algorithms with analytic learning methods that explain the success of the empirically derived rules (Gordon & Grefenstette, 1990). Finally, it makes it easier to incorporate existing knowledge, whether acquired from experts or by symbolic learning programs. This paper addresses this final point by comparing two mechanisms for initializing the knowledge structures in SAMUEL. These results should provide an interesting contrast with most published work on genetic algorithms, which usually assume *tabula rasa* initial conditions, although some studies have investigated seeding the initial population with available knowledge (Grefenstette, 1987). The results presented here show that genetic algorithms can be used to improve partially correct decision rules, as well as to

[1] If payoff is accumulated over an infinite period, the total payoff is usually defined to be a (finite) time-weighted sum (Barto et. al, 1989).

[2] SAMUEL stands for Strategy Acquisition Method Using Empirical Learning. The name also honors Art Samuel, one of the pioneers in machine learning.

learn rules from scratch.

The paper is organized as follows: First, section 2 presents the target problem and illustrates a partially successfully set of rules that was derived manually. Section 3 briefly describes the operation of SAMUEL, focusing on how its knowledge structures are initialized in the absence of prior knowledge. Section 4 present two mechanisms for initializing the knowledge structures with prior knowledge. Section 5 presents an empirical study comparing these two mechanisms against a mechanism that uses no prior knowledge. The final section summarizes our findings and discusses topics for further study.

This work is part of an on-going study of genetic algorithms for learning tactical plans. The current system is detailed in (Grefenstette, Ramsey & Schultz, 1990). The design of SAMUEL owes much to Smith's LS-1 system (Smith, 1980), and draws on some ideas from classifier systems (Holland, 1986). An analysis of the credit assignment methods in SAMUEL appears in (Grefenstette, 1988). A study of the effects of sensor noise on SAMUEL appears in (Schultz, Ramsey & Grefenstette, 1990), and techniques for applying explanation-based learning methods to the empirically derived rules learned by SAMUEL appears in (Gordon & Grefenstette, 1990).

The Evasive Maneuvers Problem

We will focus our discussion on a particular sequential decision task called the *Evasive Maneuvers (EM)* problem, inspired in part by Erickson and Zytkow (1988). There are two object of interest in this problem, a plane and a missile. The tactical objective is to maneuver the plane to avoid being hit by the approaching missile. The missile tracks the motion of the plane and steers toward the plane's anticipated position. The initial speed of the missile is greater than that of the plane, but the missile loses speed as it maneuvers. If the missile speed drops below some threshold, it loses maneuverability and drops out of the sky. There are six sensors that provide information about the current tactical state:

- *last-turn*, the current turning rate of the plane;
- *time*, a clock that indicates time since detection of the missile;
- *range*, the missile's current distance from the plane;
- *bearing*, the direction from the plane to the missile;
- *heading*, the missile's direction relative to the plane; and
- *speed*, the missile's current speed measured relative to the ground.

The sensors *time*, *last-turn*, *range*, and *speed* are linearly ordered numeric sensors, and the sensors *bearing* and *heading* are cyclic numeric sensors, taking values from the cyclic ranges 1 to 12 o'clock and 0 to 360 degrees, respectively. Finally, there is a discrete set of actions available to control the plane. In this study, we consider only actions that specify discrete turning rates for the plane.[3] The learning objective is to develop a tactical plan, i.e., a set of decision rules that map current sensor readings into actions, that successfully evade the missile whenever possible.

The EM problem is divided into *episodes* that begin when the threatening missile is detected and that end when either the plane is hit or the missile is exhausted. For the experiments described here, the missile begins each episode at a fixed distance from the plane, traveling toward the plane at a fixed initial speed. The direction from which the missile approaches is selected at random. It is assumed that the only feedback provided is a numeric payoff, supplied at the end of each episode, that reflects the quality of the episode with respect to the goal of evading the missile. Maximum payoff is given for successfully evading the missile, and a smaller payoff, based on how long the plane survived, is given for unsuccessful episodes.

The EM problem is clearly a laboratory-scale model of realistic tactical problems. Nevertheless, it includes several features that make it a challenging machine learning problem, e.g. a weak domain knowledge (e.g., no predictive model of missile), incomplete state information provided by discrete (possibly, noisy) sensors, a large state space, and, of course, delayed payoff.

We have developed a simulator for EM that can be played in interactive mode. It presents a challenging task for human players. It does not seem to be solvable by any simple, fixed tactical plan that ignores the position of the missile, such as making tight loops or flying straight ahead at full speed. Making random turns with the plane evades the missile about 35% of the time. However, after a moderate exposure to the problem, users can often suggest partially correct tactics. The

[3] The current statement of the problem assumes a two-dimensional world. Future experiments will adopt a three-dimensional model and will address problems with multiple control variables, such as controlling both the direction and the speed of the plane.

following tactical plan for EM was manually developed by one of the authors:

> If the missile is far enough away, turn so that it is behind the plane. When the missile is closing in, make hard turns such that the missile losses velocity. If the missile is heading away from the plane and going slow, ignore it and continue in the current direction.

This plan can be expressed in the rule representation language of SAMUEL, as shown in Figure 1.

```
Rule 1 ; In front of and slightly to right,
       ; so make hard 180 to left.
if (and (range 500 1500) (bearing 12 1))
then (and (turn 180 180))

Rule 2 ; In front of and slightly to left,
       ; make hard 180 to right.
if (and (range 500 1500) (bearing 11 12))
then (and (turn -180 -180))

Rule 3 ; If missile is to right,
       ; make 90 degree turn to left
if (and (range 500 1500) (bearing 2 4))
then (and (turn 90 90))

Rule 4 ; If missile is  to left,
       ; make 90 degree turn to right
if (and (range 500 1500) (bearing 8 10))
then (and (turn -90 -90))

Rule 5 ; If missile is behind the plane,
       ; let it get closer.
if (and (range 500 1500) (bearing 5 7))
then (and (turn 0 0))

Rule 6 ; Wait for it to get close,
       ; and then make a sharp turn.
if (and (range 0 400))
then (and (turn -180 -180))

Rule 7 ; If missile is heading away and
       ; going slow, just go straight.
if (and (heading 160 200) (speed 0 300))
then (and (turn 0 0))
```

Fig. 1. Manually Derived Tactical Plan for EM

In this language, each condition expresses a (possibly cyclic) numerical range for each of the named sensors. Although the plan in Figure 1 was produced with minimal knowledge engineering effort, it produces a 77% success rate when tested in the EM simulation environment. Further manual improvement is possible, but becomes increasingly more difficult. Previous studies have shown that SAMUEL could learns rules at the 95% performance level, starting from *tabula rasa*. The question under consideration here is whether the learning system could more efficiently learn high performance rules if given a reasonable starting level, represented by these manually derived rules. First, it is necessary to take a brief look at the operation of SAMUEL.

SAMUEL

SAMUEL was designed to learn rules for sequential decision problems that meet the follows requirements:

- There is a fixed set of sensors that provide a feature vector representation of the current state. Features may be numeric or tree-structured symbolic values.
- There is a fixed set of control variables available to be set by the decision making agent. These may also take numeric or symbolic values.
- There exists a critic module that provides numeric payoff to indicate the quality of the system's behavior at the end of each problem-solving episode.

SAMUEL consists of three major components: a problem specific module, a performance module, and a learning module. The problem specific module consists of the task environment simulation, or world model (in this case, the EM model), and its interfaces. The performance module is called CPS (Competitive Production System), a production system that interacts with the world model by reading sensors, setting control variables, and obtaining payoff from a critic. In addition to matching, CPS implements conflict resolution as a competition among rules based on rule strength and performs credit assignment based on payoff (Grefenstette, 1988). A *tactical plan* in SAMUEL consists of a set of condition-action rules of the form:

```
if   (and c1  . . .  cn)
then (and a1  . . .  am)
```

where each c_i is a condition on one of the sensors and each action a_j specifies a setting for one of the control variables.

The learning module uses a genetic algorithm to develop tactical plans. Each plan is evaluated on a number of tasks in the world model. As a result of this evaluation, plans are selected for replication using a procedure that is biased in favor of high performance

plans. Replicated plans are then recombined to form plausible new plans. Recombination of two selected plan is controlled by a CROSSOVER operator that exchanges a randomly chosen number of rules. The rules to be exchanged are selected so that rules that fire in sequence are more likely to be inherited as a group (Grefenstette, 1988). A background MUTATION operator makes small changes to plans by randomly altering individual conditions or actions. For details about the genetic learning operators, see (Grefenstette, 1989a). The remainder of this section focuses on how to form the knowledge structures in the initial population, if no prior knowledge about the task is available.

In the absence of available knowledge, the usual way to initializes the population of knowledge structures in a genetic algorithm is to create random plans, consisting of randomly generated rules. This has the advantage of giving the genetic algorithm highly diverse material to work with. Unfortunately, this approach is also likely to result in a lengthy search before plausible rule sets are developed through the action of the genetic operators. As an alternative, we have developed an approach called *adaptive initialization*. Each plan starts out as a set of completely general rules, but its rules are specialized according to its early experiences. That is, each rule in the initial population says:

for any sensor readings, take action X

where *X* is one of the possible actions. A tactical plan consisting of only such rules executes essentially a random walk, since every rule matches on every cycle, and all rules start with equal strength.

In order to introduce plausible new rules, an intelligent mutation operator called SPECIALIZE is applied after each evaluation of a plan. SPECIALIZE is similar in spirit to Holland's *triggered operators* (Holland, 1986). The trigger in this case is the conjunction of the following conditions:

(1) There is room in the plan for at least one more rule.

(2) A maximally general rule fired during an episode that ended with a successful evasion.

If these conditions hold, SPECIALIZE creates a new rule with the same right hand side as the maximally general rule, but with a more specialized left-hand side. Specifically, for each sensor, the condition for the sensor in the new rule covers approximately half the legal range for that sensor, splitting the difference between the extreme legal values and the sensor reading obtained in (2) above. For example, suppose the initial plan contains the maximally general rule:

```
Rule 6:  if (  ) then (and (turn 90))
```

Suppose further that the following step is recorded in the evaluation trace during the evaluation of this plan:

```
Trace:

 :
sensors: ... (time 4) (range 500)
             (bearing 6) ...
action: (turn 90)
rule fired: Rule 6
 :
```

Then SPECIALIZE would create the following new rule:

```
Rule 10:
if   (and ... (time 2 11)
              (range 300 1000)
              (bearing 3 9) ...)
then (and (turn 90))
```

The resulting rule is given a high initial strength, and added to the tactical plan. The new rule is plausible, since its action is known to be successful in at least one situation that matches its left hand side. Of course, the new rule is likely to need further modification, and is subject to further competition with the other rules.

Two Methods for Using Prior Knowledge

For simple laboratory problems such as EM, the adaptive initialization methods outlined above gives the genetic algorithm an adequate supply of plausible rules. These rules serve as building blocks for high performance plans, generated by the genetic algorithm. For more realistic problems, it is likely that some heuristic domain knowledge is available that can serve as a reasonable starting point for the learning system. This section presents two methods for initializing the population of knowledge structures in SAMUEL with existing domain knowledge, expressed in the appropriate rule language.

We call the first method the *homogeneous population* approach, since each member of the population, i.e. each rule set, contains identical rules. Recall that in the adaptive initialization method, each plan in the initial population consists of a set of maximally general rules, which are then specialized by the SPECIALIZE operator. One approach to using existing knowledge is to augment the maximally general rules in the initial plans with the previously developed heuristic rules.

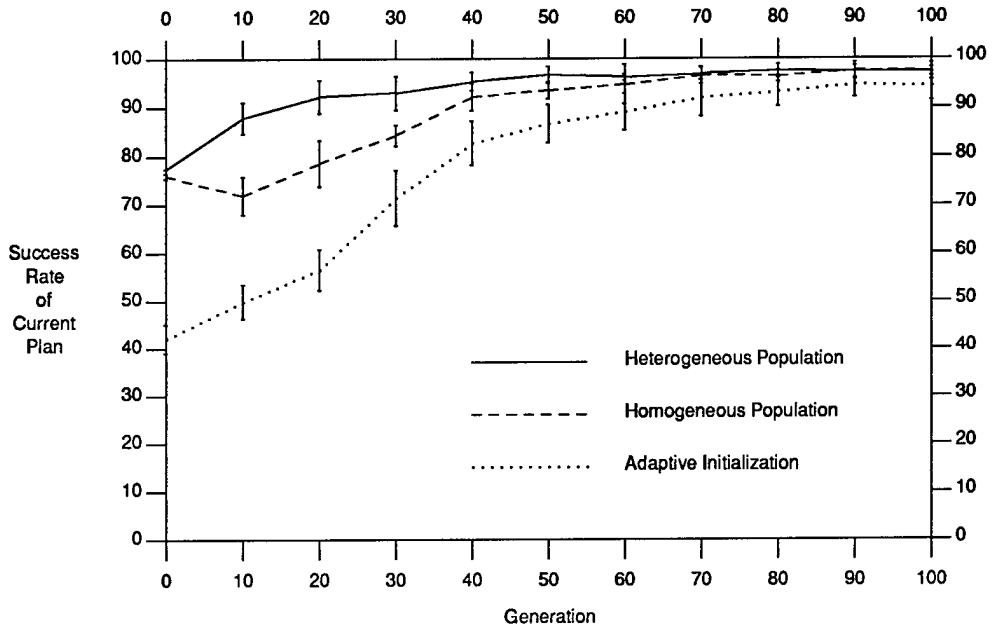

Fig. 2. Comparison of Methods for Using Domain Knowledge

Therefore, each rule set, or plan, consists of the maximally general rules plus the heuristic rules. The heuristic rules are given high strength, and the maximally general rules are given low strength. This implies that, when the heuristic rules match the sensors, they tend to take precedence over the maximally general rules.[4] On the other hand, when no heuristic rule matches, the plan falls back to the maximally general rules. If the heuristic rules match most situations that arise (as do the rules in Figure 1, for example), the net effect is that SPECIALIZE operates at a highly reduced rate, since SPECIALIZE only applies when maximally general rules fire.

Another method to using heuristic domain knowledge is called the *heterogeneous population* approach, since all members of the population are not identical. In this method, we seed part of the population with the heuristic knowledge, and let the remainder of the population consist of the maximally general rules, as in the adaptive initialization method. This approach seems to be more in line with the overall philosophy of genetic algorithms, since it puts the heuristic plans in direct competition with the plans generated through the SPECIALIZE operator. In the experiments below, we elected to seed half of the initial population with the heuristic plans, with the remaining half being maximally general.

[4] However, conflict resolution is probabilistic, so it is possible that a low strength rule might be selected over a conflicting high strength rule (Grefenstette, 1988).

An Experimental Comparison

This section presents an empirical study of the use of existing domain knowledge to initialize the knowledge structures that comprise the initial population. The experiments described here reflect two important methodological assumptions:

> (1) Since learning may require experimenting with decision rules that might occasionally produce unacceptable results if applied to the real world, we assume that hypothetical plans will be evaluated in a simulation model.
>
> (2) SAMUEL is designed to continue learning indefinitely, rather than to run for a fixed number of cycles.

In this methodology, a set of rules is periodically extracted from the learning system to represent the learning system's current plan. This plan is tested in the target environment, and the resulting performance is plotted in a learning curve. Specifically, the genetic algorithm in SAMUEL evaluates the fitness of each plan in its population by measuring its performance on a number of episodes (currently, 10 episodes per plan) in the simulation model of EM. A plan's fitness determines its reproductive probability for the next generation. At periodic intervals (currently, 10 generations), a single plan is extracted from the current population to represent the learning system's current hypothetical plan, and the plots show the performance of the

extracted plan.

Because SAMUEL employs probabilistic learning methods, all graphs represent the mean performance over at least five independent runs of the system, each run using a different seed for the random number generator. Error bars indicate one standard deviation from the mean. This device allows the reader to see significant differences between two approaches at various points during the learning process.

Three experiments were performed and are plotted in Fig. 2. The dotted line shows the learning curve when the no heuristic knowledge is available, i.e., the adaptive initialization method. The dashed line shows the learning curve when heuristic knowledge is incorporated in the initial population of knowledge structures using the homogeneous population method. The solid line is the learning curve for the heterogeneous population method.

As can be seen from the graph, the use of existing domain knowledge gives SAMUEL initially better behavior than without using that knowledge. With the homogeneous population method, we see an initial decrease in performance before the performance improves, whereas in the heterogeneous population method, no initial drop in performance occurs. This can be explained by considering the effects of the SPECIALIZE operator in the first few generations. In the homogeneous population, when SPECIALIZE fires, it introduces new high-strength rules into a rule set that also contains heuristic rules. Since the rules created by SPECIALIZE are also fairly general, they tend to compete immediately with the heuristic rules, and quickly lead to worse behavior than the original rules did. In the heterogeneous population, SPECIALIZE only applies to the maximally general rule sets, since the heuristically seeded rule sets do not contain maximally general rules. As a result, the heuristic plans maintain their original behavior much longer, and only gradually recombine with rule sets influenced by SPECIALIZE.

It is encouraging that in both methods that incorporate heuristic knowledge, the learning curves rise at a significantly faster rate than with the adaptive initialization method. The differences become insignificant after 50 generations, by which time even the *tabula rasa* system finds high performance plans. However, even after 100 generations, the experiments using the heuristically initialized populations exhibited a slightly better mean and a smaller variance in the performance of the learned plans, compared to the experiments without heuristic knowledge. This probably means that the presense of the heuristic rules causes the genetic algorithm to focus its search more consistently within the space of all plans. This effect will be analyzed in more detail in a future paper.

Summary

Our work to date with a simple tactical problem has shown that it is possible for learning systems based on genetic algorithms to effectively search a space of tactics and discover sets of rules that provide high performance. The current study has outlined two methods for utilizing existing heuristic rules in the SAMUEL learning system. The experimental results indicate that SAMUEL can learn more quickly when heuristic rules are available. More research is required to explore other methods of seeding the initial knowledge structures with heuristic knowledge. For example, it would be interesting to extend the heterogeneous population method by using variations of a single heuristic tactical plan, or by including a variety of quite different plans.

Current efforts are also aimed at augmenting the task environment to test SAMUEL's ability to learn tactical plans for more realistic scenarios. Multiple incoming threats will be considered, as well as multiple control variables (e.g., accelerations, directions, weapons, etc.). It is expected that, in more difficult domains, it will be much more important to utilize existing knowledge than in the current model of EM. Experiments in these domains should provide better guidance about the benefits of various initialization methods.

This line of research suggests that future system might benefit from the trade-offs between manual knowledge acquisition and machine learning. Ideally, future systems should use any easily available heuristic knowledge, and improve on that knowledge where possible. Further developments along these lines can be expected to reduce the manual knowledge acquisition effort required to build systems with expert performance on complex sequential decision tasks.

References

Barto, A. G., R. S. Sutton and C. J. C. H. Watkins (1989). Learning and sequential decision making. COINS Technical Report, University of Massachusetts, Amherst.

Erickson, M. D. & J. M. Zytkow (1988). Utilizing experience for improving the tactical manager. *Proceedings of the Fifth International Conference on Machine Learning*. Ann Arbor, MI. (pp. 444-450).

Goldberg, D. E. (1983). *Computer-aided gas pipeline operation using genetic algorithms and machine learning*, Doctoral dissertation, Department Civil Engineering, University of Michigan, Ann Arbor.

Gordon, D. G & J. J. Grefenstette (1990). Explanations of empirically derived reactive plans. *Proceedings of the Seventh International Conference on Machine Learning*. Austin, TX: Morgan Kaufmann.

Grefenstette, J. J. (1987). Incorporating problem specific knowledge into genetic algorithms. In *Genetic algorithms and simulated annealing*. L. Davis (ed.), London: Pitman Press.

Grefenstette, J. J. (1988). Credit assignment in rule discovery system based on genetic algorithms. *Machine Learning, 3(2/3)*, (pp. 225-245).

Grefenstette, J. J. (1989a). A system for learning control strategies with genetic algorithms. *Proceedings of the Third International Conference on Genetic Algorithms*. Fairfax, VA: Morgan Kaufmann. (pp. 183-190)

Grefenstette, J. J. (1989b). Incremental learning of control strategies with genetic algorithms *Proceedings of the Sixth International Workshop on Machine Learning*. Ithaca, NY: Morgan Kaufmann. (pp. 340-344).

Grefenstette, J. J., C. Loggia Ramsey, and A. C. Schultz (1990). Simulation-assisted learning by competition. To appear in *Machine Learning*.

Holland, J. H. (1975). *Adaptation in natural and artificial systems*. Ann Arbor: University Michigan Press.

Holland J. H. (1986). Escaping brittleness: The possibilities of general-purpose learning algorithms applied to parallel rule-based systems. In R.S. Michalski, J. G. Carbonell, & T. M. Mitchell (Eds.), *Machine learning: An artificial intelligence approach* (Vol. 2). Los Altos, CA: Morgan Kaufmann.

Selfridge, O., R. S. Sutton and A. G. Barto (1985). Training and tracking in robotics. *Proceedings of the Ninth International Conference on Artificial Intelligence*. Los Angeles, CA. August, 1985.

Schultz, A. C., C. L. Ramsey, & 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*. Austin, TX: Morgan Kaufmann.

Smith, S. F. (1980). *A learning system based on genetic adaptive algorithms*, Doctoral dissertation, Department of Computer Science, University of Pittsburgh.

Wilson, S. W. (1985). Knowledge growth in an artificial animal. *Proceedings of the International Conference Genetic Algorithms and Their Applications* (pp. 16-23). Pittsburgh, PA.

Wilson, S. W. (1987). Classifier systems and the animat problem. *Machine Learning, 2(3)*, (pp. 199-228).