TREE STRUCTURED RULES IN GENETIC ALGORITHMS

Arthur S. Bickel
Riva Wenig Bickel

Department of Computer and Information Systems
Florida Atlantic University
Boca Raton FL 33432

ABSTRACT

We apply genetic algorithm techniques to the creation and use of lists of tree-structured production rules varying in length and complexity. Actions, conditions and operators are randomly chosen from tables of possibilities. Use of the techniques is facilitated by the notions of time delay and dependency in performing tests or observations and by accounting for indeterminate test results.

GENES, a general program, can adapt to a variety of systems by changing the contents of the various tables to accomodate the domain of a particular problem and by substituting a library of appropriate cases.

INTRODUCTION

The genetic algorithm is an iterative adaptive search technique which has been applied successfully to learning systems having large search spaces and to problems which cannot easily be reduced to closed form. Holland, who originated the notion of computer constructs which mimic natural adaptive mechanisms, noted that in order to apply this technique, the subject to be treated must be capable of being represented by some data structure and the solutions must be capable of being evaluated and ranked (1,2).

APPLICATION TO EXPERT SYSTEMS

Conventionally in genetic algorithm implementations, the rules containing the expertise have been expressed in the form of bit strings of fixed length. This approach works quite well in terms of the ease of applying the genetic operators - mutations may turn a 0 to a 1 or to some other symbol taken from a similar simple alphabet; crossovers are implemented by randomly choosing two crossover points along the string and exchanging the information between those two points. However there are many problems which cannot be easily expressed in terms of a simple alphabet, and still be amenable to the genetic operators (see, e.g. work on the traveling salesman problem (3, 4)). The commonly used dual valued alphabet is especially troublesome for two reasons. First, since the number of possible actual interpretations is unlikely to be a power of two, some interpretations will be redundant, weighting the possible choices unevenly. Secondly, some mutations would be extremely unlikely, e.g. 000 would probably not mutate to 111. These problems are both eliminated by having each decisional element be an index into an appropriate table. Choices for each position and changes due to point mutations are then made merely by randomly chosing one of the possible indices into that table.

Schaffer and Grefenstette (5), building upon the LS-1 learning system originated by Smith (6), used a data structure where each expert was not an individual rule but, rather, an entire set of rules, in order to deal with multi-objective learning. These rule sets were of fixed length, although all the rules might not necessarily be used. But there may be insufficient flexibility in solving difficult classes of problems where the expert is composed of linear data structures of fixed length, since practicable computer expert systems are often presented as a system of non-linear rules (7). These rules tend to be of the form:

IF condition [{and|or condition}] THEN action

where the action may be either an attempt to obtain more information on the system (an action rule), or a terminal decision in the nature of deciding the answer to the presented problem (a decision rule).

CREATING AN EXPERT GENERATOR

We created GENES, a program to develop expert systems. Each expert in our system consisted of a linked list of rules, each rule parsed syntactically as a tree structure. The actual number of rules in each linked list was randomly determined via an average length parameter passed to the program upon its initiation. The number of rules allowed was optionally bounded.

A different program parameter determined the maximum number of nodes for each rule. The minimal possible value was one, this single node being an action node - which would mean that some prescribed action (randomly chosen from a table of possible actions) would always be carried out. One type of action was the undertaking of a specific additional test or observation; the other possible action was a final determination as to the state of the system - e.g. if the expert was a prospector, then this might be that a given mineral exists at a certain location in recoverable concentrations.

There was only one action node per rule. Additional nodes each represented a possible condition precedent to carrying out the action, linked by boolean operators (again, randomly chosen). Conditions precedent nodes are of the form:

IF {test-result-yields RELATIONAL OPERATOR scalar quantity}.

As an example, if the expert created were a chemist, a possible node might be "if melting point > 180.51 degrees". In creating (but not in evaluating) the expert, booleans, relational operators, possible tests, and possible range of results were all randomly chosen from a permissible domain of actions, conditions and values appropriate to the area of expertise desired for the system. Figure 1 shows a possible rule set for an unspecified expert.

{IF (NOT ((C1 = 2) AND (C4 > 3)) OR (C2 >= 12)) THEN A1} ->

{A5} ->

{IF ((C5 = TRUE) OR NOT (C1 <> 17)) THEN A1}

Figure 1

Statements within {} indicate a single set of rules Statements beginning with C are conditions
Statements beginning with A are actions
-> indicates the linking of rules which are evaluated in order

APPLYING THE GENETIC OPERATORS TO THE EXPERT POPULATION

Three types of genetic operators were applied to the population of experts - mutation, crossover and inversion. The tree structured nature of each rule called for some modifications to the way these operators normally deal with fixed length strings of bits.

Mutation was performed as an operation upon a single rule. A number of varieties of mutation were possible and the type of mutation occurring in each given case was chosen randomly from the list of possibilities, as was the rule to be mutated and the node of the rule at which the mutation would occur. The simplest type of mutation was a point mutation. This involved the change of one operator to another (e.g., > becoming >=). It was also possible for an operand of either a condition or an action type to be exchanged with another (e.g., in a chemical setting boiling point becoming melting point or 450 degrees might become 200). A third possibility is exemplified by the exchange of an OR for an AND or the addition or deletion of a NOT in the boolean expression. The last mutation possible was the addition or deletion of a rule or a portion of a rule - the latter being equivalent to a randomly selected branch of the parse tree.

The next simplest genetic operation was an order inversion. Each inversion was carried out on a single set of rules. This involved randomly choosing, via the MOD operator, two points along a rule set of given length, and then exchanging the order of rules that existed between the two points. This operation did not affect the rules themselves but, rather, the order in which they were evaluated. The possible effects of inversion were twofold. First, the firing of an earlier rule (that is conditions evaluating positively so that an action is taken) could cause later rules to evaluate differently. With a new ordering, given preconditions now may or may not exist. Second, due to rule evaluation halting once a decision rule fired, new rules might come under consideration or old ones might not now be reached.

The most complex genetic operator used involved crossover. Crossover, in this system, occurred at points between rules, and involved exchanges of identical amounts of genetic material (in terms of number of rules) between experts. Because the rule sets of each of the pair of experts chosen to undergo crossover were unequal in length, one point for crossover was chosen to be MOD the length of the shorter expert and one MOD the longer expert. If both values turned out to be less than the length of the shorter rule set then a double crossover occurred: a central list of rules was exchanged between the two and both experts retained their original size. If only one point was less than the shorter rule set, then only a single crossover occurred, with the tail end of the shorter expert now grafted on to what was once the longer one, and vice versa. Both offspring of each inversion survived. Figures 2a and 2b demonstrate the two types of crossovers.

R1A--R1B--R1C-|-R1D--R1E--R1F-|-R1G--R1H--R1I--R1J--R1K

R2A--R2B--R2C-|-R2D--R2E--R2F-|-R2G

crossover of rule sets 1 & 2 yield

R1A--R1B--R1C-|-R2D--R2E--R2F-|-R1G--R1H--R1I--R1J--R1K

R2A--R2B--R2C-|-R1D--R1E--R1F-|-R2G

DOUBLE CROSSOVER - CROSSOVER POINTS INDICATED BY |

Figure 2a

R1A--R1B--R1C--R1D-|-R1E--R1F--R1G--R1H--R1I--R1J--R1K

R2A--R2B--R2C--R2D-|-R2E--R2F--R2G

crossover of rule sets 1 & 2 yield

R1A--R1B--R1C--R1D-|-R2E--R2F--R2G

R2A--R2B--R2C--R2D-|-R1E--R1F--R1G--R1H--R1I--R1J--R1K

SINGLE CROSSOVER - CROSSOVER POINT INDICATED BY |

Figure 2b

THE ROLE OF THE CRITIC

Experts, naturally, relate to some field of expertise. The general set of problems which these experts were to solve involved a set of related problems, where a number of different tests could be undertaken in order to determine some information about the system. The goal of each expert was to generate the proper set of tests, in proper order, so that the correct solution for each presented problem would be obtained at a minimal cost in terms of testing expenses.

The role of the critic was to apply each expert's rule set, in order, (and repeatedly if necessary), to each member of a library of actual cases, and to obtain thereby some figure of merit for each expert. If the conditions precedent in the rule were met, then the rule fired and the action called for was taken. If the rule action called for an observation about the case, information might be unmasked, and the expert's state of knowledge increased (but only if the information was available in the actual library case). Of course costs may have been incurred in making such observations. These costs could involve time, money, or injury to the subject being observed. Results might be inconclusive or even unavailable.

Rule evaluation ceased when a decision rule fired. If the entire rule set was evaluated without a decision rule firing, then the set was reevaluated from its beginning (and at an additional cost), because the results of earlier fired rules were likely to have provided additional information that could result in different rules firing on subsequent passes. A maximum number of passes were allowed in order to ensure termination. A valuation of the expert's merit vis a vis that particular problem was made, and then the next problem in the set was presented to it, and so on. Obviously, the larger and more varied the set of problems, the more sophisticated and discriminating the expert.

Creation of a library of problems under direct program control can be a tedious and error-prone process. In order to facilitate both entry of and changes to the problem set, GENES allowed creation of the problem set via the normal Unix(TM) vi editor, and then fed the error-free result to the program.

In our trial system we were interested in evaluating our experts on a dual basis. One consideration was how often they reached the correct result. The other was how little cost was expended in reaching this result. An apportionment between these two factors is clearly a value judgment that must be made for each particular expert system. Thus each expert accrued a certain cost in its decision making, that being the sum of dollar costs incurred in running tests, degradation costs in the case of environmental misfortunes (e.g. a damaged drill in the instance of oil exploration) and penalties for bad evaluations. The merit of each expert was inversely related to the costs it incurred.

TIME DEPENDENT EVALUATIONS

Because the system we were studying was one where time was a factor, we introduced the notion of time dependency to our evaluations of the goodness of each expert. Many of the tests which could be performed by the experts would not yield immediate results. While such answers were pending other tests might or might not be undertaken, depending upon the rules (where actions might or might not be dependent upon previous test results).

Time dependency required that the entire rule set for each expert be reevaluated repeatedly, because new tests might be ordered once earlier test results came in. In many existing systems such delay involves additional daily overhead costs, so these costs were factored into the testing costs for each expert where appropriate.

A related notion was persistence, persistence being a quality associated with tests whose results do not change meaningfully over short intervals. In order to prevent the repeated ordering of such tests, they were each assigned a persistence value which was decremented with time. If a rule required that a test be done when that test showed a non-zero persistence value, the rule was ignored.

It should also be noted that there was a possibility that test results might be equivocal. To simulate this possibility, the individual problems in the sets presented were issued certain flags representing both the current and possible results of tests and observations that the expert might make. Thus, for example, in a medical situation, the patient's sex might generally be immediately knowable, the results of a blood culture would be available in 24 hours, but meaningful results of a CAT scan might actually turn out to be unobtainable. Because there was no prohibition on the repetition of rules within an expert's repertory, however, the CAT scan could be repeated and might show results the second time.

HANDLING CONVERGENCE

Researchers in the genetic algorithm field have repeatedly noted the difficulty of fine tuning the parameters in order to obtain results along the desired results. The algorithm shares this rather nasty fact with other types of computer programming: "what you ask for is what you get" (See, e.g., (8)). Due to the emphasis upon getting the correct result as compared to the cost of arriving at that result, the population of experts tended to converge rapidly to a rather homogeneous solution set, wherein a high percentage of the problem sets tested were solved properly, but at less than optimum cost. Rapid convergence is a common problem in these types of systems and has been solved in various ways (9). The solution used in GENES was to consider the system to have converged if the sum of all the expert scores remained within a narrowly bounded interval over several consecutive iterations.

If the system was judged to have converged, then a percentage of the population was replaced by newly generated experts, thereby providing a major influx of genetic material into the system.

PARAMETERS FOR GENETIC OPERATIONS

Rates of mutation, inversion and crossover, as compared with just plain duplication, were tunable parameters, as was the population size. Typically, rates in the range recommended by Grefenstette (10), namely a 30 percent crossover rate, 3 percent mutation rate, 5 percent inversion rate, and a 40 percent generation gap, were found to give good results and were used. Variations in these rates affected the time it took to reach a good solution, but did not effect the goodness of that solution. Other typical variable values were a population of experts set at 50 and 50 maximum rules; this was with 13 possible actions that could be taken and 12 possible tests-to-be-done where the possible variations for the bound of each test-to-be-done were extremely large since the constants chosen were from a continuous rather than a discrete population (e.g. temperature > 102.45). This, along with the variable number of branches in each rule and the variable number of rules in each set, makes the size of the search space difficult to estimate. It certainly is not small.

Using these parameters, the choice of which experts were to undergo each procedure was made via the weighted roulette wheel. In all cases at least one copy of the expert with the best value was retained.

RESULTS

The GENES model, we tested at least on a small scale, converged in approximately 2000 iterations with a population size of 30, or in 8 iterations with a population size of 50, to results which appear optimal in that the correct decisions were made for all nine problems presented and at a reasonably low cost. This required no excessive or duplicative testing. Due to the very large possible variation of rules, there did, however, seem to be some strange rules that remained in the best expert's rule sets. These rules tended not to make a great deal of sense, but they did not actually affect the system as their very weirdness assured that they either always or never fired. Thus the rule interpreted as "if the patient's temperature is less than 120.3 degrees discover if abdominal pain is present", is practically equivalent to stating "always check for abdominal pain".

Additionally, it should be noted that all rules following one that resulted in a decision being made were never evaluated, so they did not actually contribute to the rule set, although they did provide genetic material to succeeding generations and might be activated during inversion.

FURTHER DIRECTIONS

This system is currently being examined with promising results in two areas of expertise.

The first area that we are investigating is the minimization of hospital costs during patient diagnostic evaluations in a prospective reimbursement environment. The simple environment we are studying is a rule set appropriate to a patient admitted with suspected gall bladder disease. The expert in this system should be able to arrive at a correct diagnosis using minimal hospital resources in terms of tests and time. It should be noted that this system does not involve interpretation of tests, but only the suggests the order of testing, as a function of previous results.

The second system under study involves server activation and deactivation in queueing problems. Costs here include both activation and use of the server, with the goal being minimization of costs, while providing adequate service to the queue under varying server loads.

A back end interpreter to translate the rule sets into terms understandable by semi-expert users is also under consideration.

ACKNOWLEDGMENT

The authors wish to acknowledge the assistance of Neal Coulter for his many helpful suggestions.

BIBLIOGRAPHY

1. John H. Holland. Outline for a Logical Theory of Adaptive Systems. In Essays on Cellular Automata, Edited by Arthur W. Burks. University of Illinois Press (1970).

2. John H. Holland, "Adaptation in Natural and Artificial Systems". Univ. Michigan Press, Ann Arbor (1975).

3. David E. Goldberg and Robert Lingle, Jr., Alleles, Loci, and the Traveling Salesman Problem, Proceedings of an International Conference on Genetic Algorithms and Their Applications, John J. Grefenstette, ed. (1985) pp 154-159.

4. John Grefenstette et al, Genetic Algorithms for the Traveling Salesman Problem, Proceedings of an International Conference on Genetic Algorithms and Their Applications, John J. Grefenstette, Ed. (1985) pp 160-165.

5. J. D. Schaffer and John Grefenstette, Multi-Objective Learning Via Genetic Algorithms, International Joint Conference on Artificial Intelligence (9th 1985) pp 593 - 595.

6. S. F. Smith, Flexible Learning of Problem Solving Heuristics Through Adaptive Search, International Joint Conference on Artificial Intelligence (8th 1983) 422 - 425.

7. A.J. Tomas and A. Th. Schreiber, HERMAN Computer Aided Medical Decision Making, Artificial Intelligence in Medicine, I. De Lotto, M. Stefanelli, Ed. North-Holland 1985, pp 1 - 9.

8. K. De Jong, Genetic Algorithms: A 10 Year Perspective, Proceedings of an International Conference on Genetic Algorithms and Their Applications, John J. Grefenstette, ed. (1985) pp 169 - 177.

9. J. E. Baker, Adaptive Selection Methods for Genetic Algorithms, Proceedings of an International Conference on Genetic Algorithms and Their Applications, John J. Grefenstette, ed. (1985) pp 101 - 111.

10. John Grefenstette, Optimization of Control Parameters for Genetic Algorithms, IEEE Transactions on Systems, Man, and Cybernetics, (Vol. SMC 16 No. 1 Jan/Feb 1986) pp 122 - 128.

GENETIC ALGORITHMS AND THEIR APPLICATIONS: Proceedings of the Second International Conference on Genetic Algorithms

July 28-31, 1987
at the
Massachusetts Institute of Technology
Cambridge, MA

Sponsored By

American Association for Artificial Intelligence

Naval Research Laboratory

Bolt Beranek and Newman, Inc.

John J. Grefenstette
Naval Research Laboratory
Editor

LEA LAWRENCE ERLBAUM ASSOCIATES, PUBLISHERS
1987 Hillsdale, New Jersey Hove and London