EuroGP2002
5th European Conference on Genetic Programming
3-5 April, Kinsale, Ireland

Genetic programming applied to strategies learning.

James Cunha Werner

Terence C. Fogarty

SCISM
South Bank University
103 Borough Road
London SE1 0AA
{wernerjc,fogarttc}@sbu.ac.uk

Abstract. This study addresses the problem of knowledge acquisition to decision taken, applied to Tic-tac-toe game: a fix structure and rules, with a reasonable number of solutions, each one carrying to a different result (win, lost or tie). The comparative study focus the case where all possible states are modelled by genetic programming, and a set of rules are applied against the opponent. The other approach is the acquisition of knowledge on fly, applying each solution for a number of trials, and using genetic programming to obtain the better solution.

Introduction.

Genetic programming (GP) is applied in mathematical problems solution, with a previous defined set of operations and terminals, to obtains the best solution according to a fitness function. The problem when working with strategies is that it would not be represented by a mathematical modeling, but through a sequence of rules with express the knowledge. A problem would have more than one possible action, or more than one rule attended.

The decision take problem consists of two steps: define what are the possible actions available and select between then the best one, modeled by a different three by GP.

Tic-tac-toe is a good decision taking problem, where the goal consist in obtain two different strategies: to win the game or to impede the opponent win (tie). The choice is due to the fix structure and size of the problem, consisting of 9 cells, with a total of 19683 states combinations with 2781 possible moves. There is a program available in Internet [1]- termed guru, whose solutions would be analyzed comparatively with genetic programming results. To avoid repetitive results, the first move is a random cell, the second is due by the guru, and then GP plays, then the Guru, … until obtain a final result.

A previous approach available in literature is Peter Angeline [2] solution of Tic-tac-toe problem obtaining with GP the programming with the complete move sequence implemented into the solution.

The modeling algorithm.

The genetic programming (GP) algorithm ([3], [4] and [5]) mimics the evolution and improvement of life through reproduction, when each individual contributes with its own genetic information to building a new individuals with greater fitness to the environment and higher chances of survival. Each 'individual' in a generation represents, with its chromosome, a feasible solution to the problem; in our case, a discriminate function to be evaluated by a fitness function.

The best individuals are continuously being selected, and crossover and mutation take place. Following a number of generations, the population converges to the solution that best represents the discrimination function (Fig. 1).

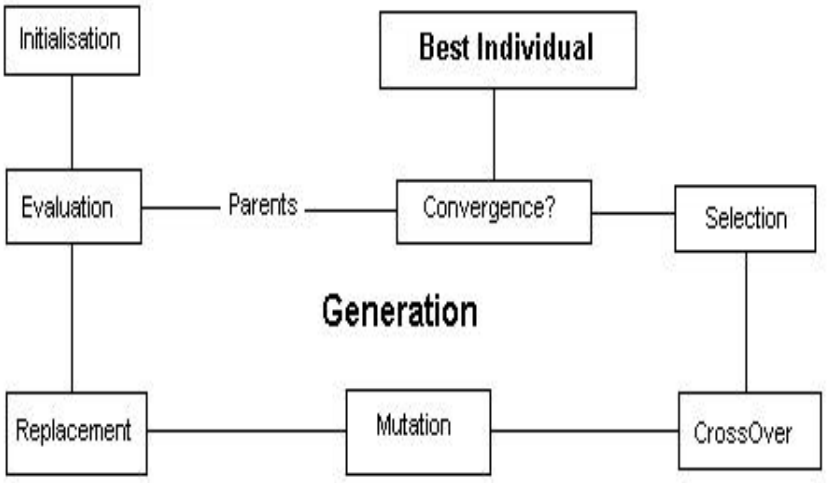

Fig. 1. Block diagram of Genetic programming.

There are two kinds of information defined for the algorithm: terminals (variable values and random numbers) and functions (mathematical functions used in the generated model).

The functions defined for genetic programming are: AND, OR, NOT and XOR, and the terminals are: random number [0..1] and each possible cell with guru or GP move.

The software application architecture.

Two different approaches are compared in this paper. The first try to model all possible moves from a static dataset with all possibilities. Genetic programming works in three different phases:

1. Training: where a set of complete information feeds the software to extract the rules (or models).
2. Result application (or test): a data set with was not used into the training is applied to the results, and the accuracy is measured, to check how complete the rules are.
3. Inference of new rules: usually, the training data set is not enough to represent all different states of the system, and the rules have to be updated.

To study the solution effectiveness, an integer random generator (1 to 9) plays against the guru to measure the performance of a unlearn algorithms. Then all possible states are applied to guru (fig 2), to obtain all possible best plays with are submitted to GP to extract the rules and then 100 games are played with these rules against the Guru. The fitness evaluate hit the mark number.

The errors total would be against all possibilities (with means that the rule would result 0 when the best move isn't the one), only considering the specific move or only the cases where the sequence wins.

The second approach (fig 3) consists evaluate each individual with 100 games played against the guru, and measure the fitness as:

$$Fitness = 5.0 * Num\_tie + 10.0 * Num\_win \quad (1)$$

Other point to consider when the goal is obtain the rules for each possible state is the choice sequence, because in one state there are more then one possible move. Two solutions are analyzed: by the distribution number of the state occurrence or using the victory information. Both are available after some training games, or using all possible configurations.

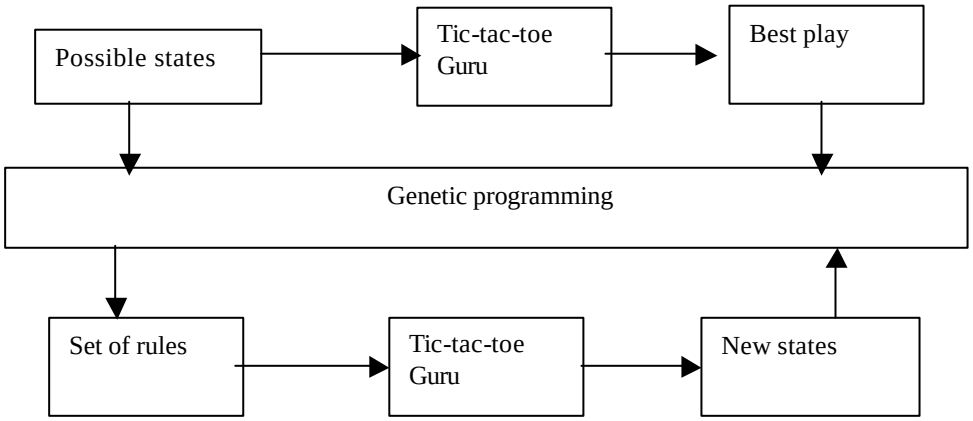

Fig. 2. Training and test architecture for complete state approach.

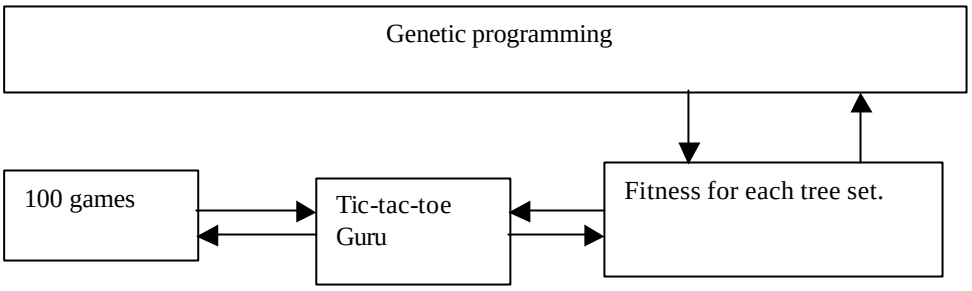

Fig. 3. Training and test architecture on fly.

Results.

Table 1 show the results for each different point of view, with a population of 30 individuals and 1000 generations. Other 100 games are played with the rules, with are not available during the training, and this results are evaluate to the total of Tie, wins and lost.

Table 1. Results of GP player – test condition.

Algorithm		Number of Tie	Number of lost	Number of wins
Random moves		6	94	0
Rule extraction	The fitness measures all possible move	12	88	0
	Fitness only for each position	33	67	0
	Fitness only where are possibility of win	25	75	0
On fly	Big frequency sequence	92	8	0
	Best opportunity and then big frequency sequence	92	8	0

Analysis and conclusions.

The extraction rule from the total possibilities space looked to be the better solution for the problem, because all states are mapped into the training set, and the on-fly training will not cover all possibilities, but only a few (100 games with 9 moves against 2781 possible moves).

However, genetic programming extracted the rules for each cell much better with the training on fly, and the strategy of move sequence with the big frequency or the best opportunity and then big sequence don't show any difference, probably because the big occurrence would be the best one.

References:

1. Chapel, S.; "Uses alpha-beta pruning minimax search to play a perfect game" www.programmersheaven.com/zone8/cat121/1020.html
2. Angeline, P. J. and Pollack, J. B. (1992) The Evolutionary Induction of Subroutines, In *The Proceedings of the 14th Annual Conference of the Cognitive Science Society*. Hillsdale, NJ: Lawrence Erlbaum, pp 236-241; Angeline, P. J. and Pollack, J. B. (1994) Coevolving High-Level Representations, In *Artificial Life III*, C. Langton (ed.), Addison-Wesley: Reading MA, pp. 55-71; see http://www.natural-selection.com/people/pja/
3. HOLLAND,J.H. "Adaptation in natural and artificial systems: na introductory analysis with applications to biology, control and artificial intelligence." Cambridge: Cambridge press 1992 reedição 1975.
4. KOZA,J.R. "Genetic programming: On the programming of computers by means of natural selection." Cambridge,Mass.: MIT Press, 1992.
5. LilGP "*Genetic Algorithms Research and Applications Group (GARAGe)*", Michigan State University; http://garage.cps.msu.edu/software/lil-gp/lilgp-index.html