Controlling Genetic Algorithms with Reinforcement Learning

James E. Pettinger
Dept. of Computer Science,
University of Exeter,
Exeter, EX4 4QT, UK.
J.E.Pettinger@exeter.ac.uk

Richard M. Everson
Dept. of Computer Science,
University of Exeter,
Exeter, EX4 4QT, UK.
R.M.Everson@exeter.ac.uk

Abstract

Here we present a hybrid system that uses a reinforcement learning agent to improve the performance of a genetic algorithm on the travelling salesman problem (TSP). The agent uses Q(λ) learning to estimate state-action utility values, which it uses to implement high-level adaptive control over the genetic algorithm. In this way the agent influences selection of both crossover and mutation operators as well as the selection of individuals for breeding, at every generation.

1 RL-GA MODEL

One of the weaknesses of genetic algorithms (GAs) is that there is a myriad of choices to be made in their implementation, such how frequently to crossover and mutate, how to determine which individuals will be selected for breeding, what operators to use for crossover and mutation, and so on. In study this we attempt to enhance the performance of a GA by using a reinforcement learning agent to learn to how adaptively control some of these aspects.

Our model itself contains two parts: a genetic algorithm and a reinforcement learning (RL) agent. The GA is capable of functioning independently but it is non-adaptive. The GA population itself forms the current state of the RL agent, which is represented using a set of state features. At each timestep the agent has certain actions available to it. Each action represents two things, the particular crossover or mutation operator the GA will use at the current timestep and the fitness class of the breeding pair or individual used with the selected operator. Each potential parent was classed as Fit (F) if it lay in the top 10% of the population and Unfit (U) if it did not. Thus, a crossover operator can be used with one of 4 types of parent pair, {FF, FU, UF, UU} and a mutation operator can be used with 2 types of individual, {F, U}. We use 3 crossover operators and 4 mutation operators – giving 20 composite actions in total.

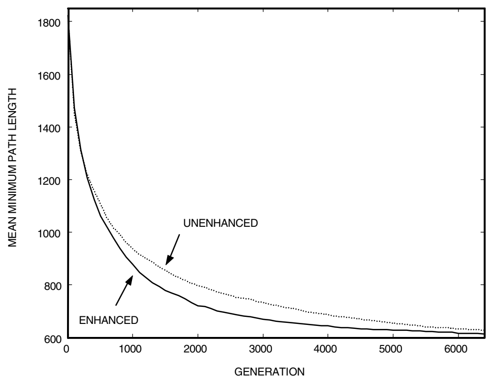

Figure 1 : Enhanced and Unenhanced GA Performance

Each action has a set of state-action utilites associated with it where each utility is an estimate of the reward that will be obtained if that action is used in a given GA state. An agent selects actions with a probability proportional to each state-action utility estimate. This scheme permits the agent to control both the genetic operator used and the particular individuals acted upon, dependant on the current population characteristics.

The system is trained by rewarding the agent after each action is taken. Reward is calculated by comparing the fitness of the parents and offspring. The better the offspring are, in comparison to the parents, the larger the reward and *visa versa*. This reward is used to update the utility of the action taken using Q(λ) learning. In this way actions that give larger rewards in a certain GA state are selected from that state with a higher probability. We then froze the learned utility estimates and applied the trained system to a test problem. Figure 1 shows the mean best solution across 50 test runs of a 40-city TSP for an RL enhanced GA and an unenhanced control GA.

2 CONCLUSIONS

Initial results on our test problem are encouraging with an RL-GA hybrid system outperforming an analogous non-adaptive GA system, on a 40-city TSP test problem. The hybrid system produces a better solution overall and learns faster. Results on a larger 100-city problem also show improvement.