Real Options Approach to Finding Optimal Stopping Time in Compact Genetic Algorithm

Sunisa Rimcharoen, Daricha Sutivong and Prabhas Chongstitvatana
Department of Computer Engineering,
Faculty of Engineering, Chulalongkorn University
Pathumwan Road, Bangkok 10330 Thailand
suni16@hotmail.com, daricha.s@chula.ac.th, prabhas@chula.ac.th

Abstract: Real options technique has emerged as an important tool for investment evaluation under uncertainty. It explicitly recognizes the future decisions, and the exercise decision is based on the future value of an underlying real asset. Real options approach has been applied to many economic and finance problems but few are in computer science and engineering. The novelty of this work is to apply real options to a computational problem. This paper proposes using real options technique to find an optimal stopping decision for the compact genetic algorithm. The compact genetic algorithm, a kind of genetic algorithms, represents the population as a probability distribution over the set of solutions. Using this distribution, underlying uncertainty of the problem is automatically captured. The compact genetic algorithm allows us to model the underlying uncertainty by simulating an evolutionary process of the algorithm many times. In the experiment, we show preliminary results of employing the real options approach to determine the optimal stopping time for the compact genetic algorithm. The proposed technique can be applied to analyze the other learning algorithms, such as neural networks or other variations of genetic algorithms.

1. Introduction

Genetic algorithms are becoming a common technique to solve difficult real world problems, and there are many research works that apply genetic algorithms to financial and real options. In the real options, mostly, these works are based on option pricing. Chen and Lee [1] studied the application of genetic algorithms to option pricing. Chidambaran et. al. [5] proposed a new methodology that used genetic programming to approximate the relationship between the price of a stock option and properties of the underlying stock price. In the same time, and the same conference, Chen et. al [3] also provide some initial evidence of the empirical relevance of genetic programming to option pricing. The pricing formulas is derived from the genetic programming and then compared with the Black-Scholes model. In the next year, Chen et. al [2] proposed an extended version of hedging derivative securities with genetic programming. Lastly, Chidambaran [4] used Monte Carlo simulations to generate stock and option price data to develop a genetic option pricing program.

Other research work related to genetic algorithms and real options is finding an optimal decision rule for oil field development. In this research, Lazo et. al. [8], the Monte Carlo simulation is employed within the genetic algorithm for simulating the possible paths of oil prices.

All of them used genetic algorithm or genetic programming to improve an option models. Thus, on the other hand, we propose the idea using real options analysis to deal with the uncertainty in the genetic algorithm. Section 2 describes the detail technique of the compact genetic algorithm. Section 3 defines the problem used in the experiment. Section 4 shows how to model underlying uncertainty of the problem. Section 5 formulates the value function of the real options. Section 6 presents the result and analysis. Finally, the concluding remarks of this study is contained in section 7.
2. The compact genetic algorithm

The genetic algorithms, the branches of evolutionary computation, are based upon the principle of natural evolution and the principle of the survival of the fittest. Evolutionary computation techniques abstract these evolution principles into algorithms. In an evolutionary algorithm, a representation scheme is chosen by a researcher to define a set of solutions that form the search space for the algorithm. The representation of genetic algorithm is a fixed-length bit string and that of the compact genetic algorithm is a probability vector. In general genetic algorithm, a number of candidate solutions are created and evaluated using a fitness function that is specific to the problem being solved. A number of solutions are chosen to be parents for the creation of new individuals or offspring. The survivors are selected from the original population and the offspring to form a new population of the next generation using their fitness values.

The compact genetic algorithm (cGA), proposed by Harik et al. [7], is a special class of genetic algorithms. It represents the population as a probability distribution over the set of solution; thus, the whole population needs not to be stored. At each generation, cGA samples individuals according to the probabilities specified in the probability vector. The individuals are evaluated and the probability vector is updated towards the better individual. The cGA has an advantage in using a small amount of memory and achieves comparable quality with approximately the same number of fitness evaluations as simple genetic algorithm. The pseudocode of cGA is shown in fig. 1. The parameters are population size($n$) and chromosome length($l$).

1) initialize probability vector
   for $i := 1$ to $l$ do $p[i] := 0.5$;

2) generate two individuals from the vector
   $a := \text{generate}(p)$;
   $b := \text{generate}(p)$;

3) let them compete
   $\text{winner, loser} := \text{compete}(a, b)$;

4) update the probability vector towards the better one
   for $i := 1$ to $l$ do
     if $\text{winner}[i] \neq \text{loser}[i]$ then
       if $\text{winner}[i] = 1$ then $p[i] := p[i] + 1/n$
       else $p[i] := p[i] - 1/n$;

5) check if the vector has converged
   for $i := 1$ to $l$ do
     if $p[i] > 0$ and $p[i] < 1$ then
       return to step 2;

Fig. 1. Pseudocode of the cGA

For example, the updating method of compact genetic algorithm has shown in figure 2, assuming step size 0.25.
3. Description of the problem

In this experiment, we choose 10 bit one-max problem as a test problem. It is a simple test problem or toy problem for genetic algorithm. This problem finds a maximum value in which all bits are one. The fitness value is assigned according to the number of bits that are one in the chromosome. Thus, the maximum value is equal to chromosome length. An example is shown in table 1.

Table 1. Example of one-max problem

<table>
<thead>
<tr>
<th>Chromosome String</th>
<th>Fitness</th>
</tr>
</thead>
<tbody>
<tr>
<td>101100010</td>
<td>5</td>
</tr>
<tr>
<td>1110010101</td>
<td>6</td>
</tr>
<tr>
<td>0010001011</td>
<td>4</td>
</tr>
<tr>
<td>1111100000</td>
<td>5</td>
</tr>
<tr>
<td>1111111111</td>
<td>10</td>
</tr>
</tbody>
</table>

4. Modeling Underlying Uncertainty

The underlying uncertainty of the compact genetic algorithm is a fitness value. According to the algorithm, when a candidate solution is sampling from the probability distribution, it is evaluated and the fitness value is assigned. This value depends on the distribution. In order to address about the fitness movement in the compact genetic algorithm, we capture the characteristic of all possible values in each time step. Generally, the fitness value will increase over time because the probability distribution is evolved.

To model the uncertainty in the real options application, principally, most people identify the key uncertainty and model it using stochastic process that fit to the problem. On the contrary, in the compact genetic algorithm the uncertainty can view as the changes of the fitness values in each state. In the beginning, the average fitness value is 5.0 because we initial the probability vector with the uniform distribution. In the next step, this fitness maybe rises from 5.0 to 6.0, 7.0, …,
10.0 or fall to 4.0, 3.0, …, 0.0. We can find the occurrence probability of these values and use it as the underlying uncertainty of the compact genetic algorithm.

In this work, we model the uncertainty of the compact genetic algorithm by observed the fitness values in many runs and keep track over time. Because the underlying uncertainty of this problem can obtain from the compact genetic algorithm, we do not need any model such as geometric brownian motion or mean-reverting process. We can construct a tree with a probability obtained from the compact genetic algorithm. By this method, it opens to use real options in wide varieties of applications that use the learning method as the genetic algorithm.

Fig. 3. lattice of all possible value

When running the compact genetic algorithm, we have a fitness value in each generation (time step). We accumulate the possible changes of fitness in each generation many runs and then calculate the probability of all possible values in each state. In this experiment, 10 bit one-max problem, the possible average values are 0.0, 0.5, 1.0, .., 9.0, 9.5, 10.0. Fig. 3 shows the lattice tree of all possible value associated with their probability.

5. Value Function of option

This paper proposes the real options analysis that applies to the compact genetic algorithm. We select the compact genetic algorithm because the uncertainty is obvious to see. The underlying uncertainty depends on a probability vector. In each time step, two individuals are sampling from the distribution and values of these candidates are assigned by the evaluation routine. The probability vector drives these values and then using these values to update the probability vector according to the best candidate. It has a certain cost per one sampling. The average fitness of these candidates is the outcome. Although the maximum value has been expected, it is an optimistic view of events. As the candidate solutions are sampling from the probability vector, they have a chance that one sampling is good and another is bad. Therefore, we use the average value to be a representative of information in order to neutralize the event.
Let \( \pi(x) \) denote the profit, and \( \Omega(x) \) the termination payoff. We apply Bellman equation corresponds to Dixit and Pindyck [6]. Then the value function becomes

\[
F(x) = \max \left\{ \Omega(x), \frac{\pi(x)}{1 + \rho} \varepsilon[F(x') | x] \right\}.
\] (1)

The termination payoff is shown in eq. 2.

\[
\Omega(x) = \text{fitness\_value}(x) \times \text{fitness\_price} \tag{2}
\]

We illustrate the method with a simple example. In this case, there is no profit and no discounting. Equation 1 is becomes

\[
F(x) = \max \left\{ \Omega(x), \varepsilon[F(x') | x] \right\}. \tag{3}
\]

Noted that in this work we do not use the discount factor because in each state the compact genetic algorithm takes a few milliseconds to run and it is no effect to the future value, in other words, the future value does not distinguish from the present value. We also ignore the profit term \( \pi(x) \) because the compact genetic algorithm does not produce the profit flow in the future. The solution value is obtained by the time the algorithm terminated.

To implement this idea, we assume that one sampling must pay one dollar and one fitness has value of 100 dollar. The compact genetic algorithm sampling two individual so it must pay two dollar in each generation. Here, the termination payoff is a fitness value multiply by 100 dollar and the continuation must pay two dollar for a new sampling because the compact genetic algorithm requires two evaluation cost per time step. We formulate the option value of this case below:

\[
F(x) = \max \left\{ \text{fitness\_value}(x) \times 100, \varepsilon[F(x') | x] - 2 \right\}. \tag{4}
\]

However, in the real world problem the fitness price can define as a real value. For example, in a bin packing problem we know the cost of each piece and if the pack is success we will gain a certain profit. Thus, equation (4) can adapt to a real pricing.

6. Result and Analysis

For this preliminary study, we use 10 bit one-max problem as a test example. At first, we run this problem with the compact genetic algorithm and keep the probability distribution of each value over time. The probability is an average of 1,000,000 runs. We use these data to construct a lattice tree of a possible value with a probability of them. Second, we calculate an option value according to eq. 2 using dynamic programming approach. The option value is an average of 100 runs. Finally, we get an option value and exercise policy. Fig. 4 illustrates the exercise region of this problem. Note that the reported exercise thresholds are biased high.

stop
As shown in Fig. 4, the algorithm should decide to stop the search when the fitness value rises above the upper threshold because the value is worth. On the other hand, if the value is lower than the lower threshold, the algorithm is also decided to stop because it is not worth to invest. We observe that the exercise threshold graph has a step that jump to another value. This is because it may a characteristic of the discreteness of the fitness value. In this problem, the option values play an importance role in the deciding to stop or continuation. If an option has a value the algorithm will decide to put an effort to search more sampling. On the other hand, if an option has no value, the algorithm will terminate or reset. This decision helps the algorithm to save the waste effort and time.

In terms of genetic algorithm, the optimal fitness value is expected. Most of people who use the genetic algorithm to find the optimal solution spend more time and effort to achieve it. But, in an investor’s point of view, the algorithm should pay the adequate effort for the problem because the small profits are not worth the while. The real options analysis can show that we can analyze the worth optimal stopping time for running the algorithm. In this experiment, we show a preliminary result that employs the real options approach to determine the compact genetic algorithm whether to stop.

7. Concluding Remarks

This paper proposes a model based on real options and compact genetic algorithm which designed to find an optimal stopping decision for compact genetic algorithm. The novelty of this work is to apply real options in a computational problem. In the experiment, we show a preliminary result that employs the real options approach to analyze the 10 bit one-max problem. The result shows that the proposed technique can provide a stopping decision to the algorithm. The algorithm should decide to stop the search when the fitness value rises above the upper threshold or should
decides to stop when the fitness value below the lower threshold. This study also leads to a more application domains of using real options to analyze the other learning algorithm such as neural network or more complicate problems in genetic algorithms.

8. References


