Optimal Stopping Time of Compact Genetic Algorithm on Deceptive Problem Using Real Options Analysis

Sunisa Rimcharoen, Daricha Sutivong and Prabhas Chongstitvatana

***Abstract*— This paper proposes using a decision contour derived from real options analysis, which is an evaluation tool for investment under uncertainty, to suggest an optimal stopping time of the compact genetic algorithm on the trap problem. The proposed criterion provides a stopping boundary, where termination is optimal on one side and continuation is on the other. A generic stopping function is formulated with an exercise region that scales well. The new stopping policy helps save on computational effort, and the evolutionary process reaches a higher solution quality when the reset method is incorporated. The proposed technique can be applied to analyze other problems.**

I. Introduction

STOPPING criterion is a common setting condition in many learning algorithms. It plays an important role in deciding when to obtain an appropriate solution. However, identifying a suitable criterion is critical because it affects solution quality. Since evolutionary algorithms provide no reliable stopping criterion [1], practitioners use some heuristics to set bounds of running the algorithms. No one knows whether the genetic algorithms will finish soon, later or never. The problem is whether we should terminate the algorithm or try to wait for solution. Therefore, this paper suggests a new approach to optimally terminate the algorithm using a contour derived from real options analysis. This contour provides a stopping boundary, where termination is optimal on one side and continuation is on the other. If the current fitness value falls into the stopping region, the algorithm should decide to stop because, with the current population, it is unlikely to achieve a better result.

The proposed criterion is different from few theoretical stopping criteria of the genetic algorithms in the literature. We will classify them in three main domains: the traditional stopping criteria, the theoretical upper bound, and the cost-benefit stopping criteria.

1) The Traditional Stopping Criteria

Michalewicz [2] and Zielinski et al. [3] identified the following kinds of termination conditions employed for genetic algorithms.

- An upper limit on the number of generations is reached.
- An upper limit on the number of fitness function evaluations is reached.
- The optimum is reached.
- The chance of achieving significant changes in the next generation is relatively slim.
- The algorithm terminates when the movement of individuals is below a threshold.
- The algorithm terminates when the individuals are close to each other.
- The algorithm uses several criteria in combination.

The current trends of these traditional stopping criteria move towards the employment of adaptive termination conditions instead of using a fixed number of iterations [4] because the search space in a real world problem is large and complex. There are some research works that modify these basic criteria to a specific problem. For example, Hernandez et al. [5] proposed a stopping criterion that is applied to a generic evolutionary algorithm (GEA) using coupling from the past generation. It checks whether the equilibrium distribution has been reached and concludes that the GEA should stop once it reaches stationary state where the underlying populations are the same for all the realizations. Meyer and Feng [6] proposed a fuzzy stopping criterion for a genetic algorithm which provides an evaluation of the genetic algorithm's real-time performance. The performance level is estimated using past population fitness and the acceptance parameters can be modified during the genetic algorithm process. The iteration is stopped when the estimated value is above a user-defined acceptance level.

2) The Theoretical Upper Bound Stopping Criteria

Several researchers propose the stopping criteria of the genetic algorithms based on a level of confidence that an optimal solution is found. The number of iterations required in a genetic algorithm can be obtained by a convergence analysis. Aytug and Koehler [7-8] estimated an upper bound of the number of iterations required to achieve a level of confidence to guarantee that a simple genetic algorithm converges. This model is based on Markov chain given by Nix and Vose [9]. Pendharkar and Koehler [10] extended this work to provide the bound based on the average performance that differs from the worst case performance bound presented in [7-8].

S. Rimcharoen, D. Sutivong and P. Chongstitvatana are with the Department of Computer Engineering, Faculty of Engineering, Chulalongkorn University, Thailand (e-mail: suni16@hotmail.com, daricha.s@chula.ac.th and prabhas@chula.ac.th).

3) The Cost-Benefit Stopping Criteria

Hulin [11] proposed the loss minimization criterion that considers the effect of two losses: the cost of computational time and the cost of obtaining a suboptimal solution. The algorithm will terminate if the total loss is expected to increase. The cost distribution model is estimated using Bayes' formula and is assumed that the prior distribution follows a Dirichlet distribution. The essential assumption of this work is that the cost distribution of two successive generations is similar. Therefore, if this assumption is false, the loss minimization should not be used.

This paper presents a different approach to analyze an optimal stopping time by using the real options approach. The optimal stopping problem is an important class of a stochastic control problem that arises in economics and finance, such as finding optimal exercise rules for financial options. Fortunately, there are similarities in the problem of finding an optimal stopping time in genetic algorithms and finding optimal exercise rules for financial options. The concept behind this technique is that finding an optimal stopping time of the algorithm can be viewed as deciding when to exercise a call option. To explore this approach, Rimcharoen et al. [12] proposed finding optimal stopping time in the compact genetic algorithm. Using this special class of genetic algorithms, the compact genetic algorithm [13], the underlying uncertainty can be viewed as a probability distribution. This distribution automatically captures the underlying uncertainty of the problem, which can be simulated to obtain an evolutionary process of the algorithm. This forms a basis in using the real options valuation in order to determine when it is worth stopping the algorithm. In this paper, we present some contours, or so called exercise region, that suggest boundaries of preference fitness values in each generation. If the solution's models do not satisfy these bounds, the algorithm should decide to stop. Only appropriate distribution model should continue. We also substantiate the proposed criterion with an analysis on scalability of a trap problem. We formulate a generic stopping function with the exercise regions that scale well, and show that the new stopping policy can help save on computational effort when stopping early and the evolutionary process reaches a higher solution quality when the reset method is incorporated.

The paper is organized as follows. We introduce the concept of real options and the compact genetic algorithm in section II and III. Section IV describes detailed techniques of using option-based methodology to find optimal stopping time of the algorithm. Section V shows how to formulate the stopping policy of the trap problem as an example. Section VI presents the results and analysis of using the proposed criterion on various problem sizes. Finally, the concluding remarks of this study are in section VII.

II. Real Options

Real options approach is a financial concept that applies financial option theory to investments in real assets (as opposed to financial assets that are traded in the market). A financial option is the right, but not an obligation, to buy or sell an asset. An option that gives the holder the right to purchase an asset at a specified price is a call option, while an option that gives the holder the right to sell an asset at a specified price is a put option. The financial options are useful for managing risks in the financial world. For example, a call option can limit possible loss by paying an upfront premium to have this right, or it can open the possibility of unlimited gains. Black and Scholes [14], and Merton [15] have inspired the rapid development in financial option pricing. For example, the two basic methods for pricing financial options are the binomial lattice [16] and the Black-Scholes formula [14].

The financial option concept was extended to real assets when Myers [17] identified the fact that many corporate real assets can be viewed as call options. The real options approach addresses an investment decision problem by analyzing not only the expected net present value (NPV), but also considering the value of an option to wait, expand, abandon, etc.

One of the techniques to find an option value is a dynamic programming method. The idea of dynamic programming is to split a whole sequence of decisions into two parts: the immediate choice and the remaining decisions. The detailed technique is described in Dixit and Pindyck [18].

The value $F_t(x_t)$ is the expected net present value (NPV) when the firm makes all the decisions optimally from this point onwards. The value function called Bellman equation or the fundamental of optimality is shown in (1).

$$F_t(x_t) = \max_{u_t} \left\{ \pi_t(x_t, u_t) + \frac{1}{1+\rho} \varepsilon_t[F_{t+1}(x_{t+1})] \right\} \qquad (1)$$

At each period t, choices available to the firm are represented by the control variable(s) u_t. The value u_t must be chosen using only the information available at the time t, namely x_t. When the firm chooses the control variables u_t, it gets an immediate profit flow $\pi_t(x_t, u_t)$. The discount factor between any two periods is $1/(1+\rho)$, where ρ is the discount rate. The term $\varepsilon_t[F_{t+1}(x_{t+1})]$ is the expected value from time $t+1$ on called a continuation value.

An optimal stopping time is found by selecting the maximum value between the termination payoff $\Omega(x)$ and the continuation value. The Bellman equation becomes

$$F(x) = \max \left\{ \Omega(x), \pi(x) + \frac{1}{1+\rho} \varepsilon[F(x') \mid x] \right\}. \qquad (2)$$

From (2), there is a payoff value as a function of x achieved by termination and a payoff value as a function of x achieved through continuation. The x values that produce the boundary payoff values, where termination is optimal on one side and continuation is on the other, form an exercise region.

III. The Compact Genetic Algorithm

This section gives an overview of the compact genetic algorithm [13]. It reduces the size and power requirements of the system by representing the population as a probability vector rather than a collection of bitstrings. At each generation, the compact genetic algorithm 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 pseudocode of this algorithm is shown below.

1) initialize probability vector (p)
   for i := 1 to l do $p[i]$:= 0.5;
2) generate two individuals from the vector
   a := generate(p);
   b := generate(p);
3) let them compete
   winner, *loser* := compete(a, b);
4) update the probability vector towards the better one
   for i := 1 to l do
     if *winner*[i] ≠ *loser*[i] then
       if *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;

The parameters are the updating step size(n) and chromosome length(l). Notice that the parameter n is related to the population size in the simple genetic algorithm. The detail is provided in the original paper [13].

IV. Finding Optimal Stopping Time Using Option-Based Methodology

This section provides an overview of the proposed methodology. We employ the real options approach to determine when to stop running a genetic algorithm, which is analogous to deciding when to exercise a call option. In each generation, the algorithm can stop or continue running. If the algorithm decides to stop, the payoff from stopping is obtained. If the algorithm decides to continue, further computation may add value, but it also incurs a computational cost. To determine when to terminate, the algorithm needs to know the probability distribution of the fitness value (underlying uncertainty) and the payoff model (value function of option). At every generation, we compute the expected payoff from stopping and continuing using the underlying uncertainty and the value function of option. The algorithm should continue if the expected payoff from continuing is higher than that of stopping. The stop or continue decision is solved starting from the last time step and working backward to the first generation, as in dynamic programming.

The methodology of finding an optimal stopping time in genetic algorithm described above can be summarized in the following process.

1) Modeling Underlying Uncertainty

In this step, we need to know the movement of fitness values in each generation. We can obtain this distribution by running the genetic algorithms many times. For example, suppose the average fitness value in the first generation is 5.0. Assume that in the first run the fitness value increases to 6.0 and the second run the fitness value falls to 4.0. The fitness movement of these two runs can be shown in Fig. 1.

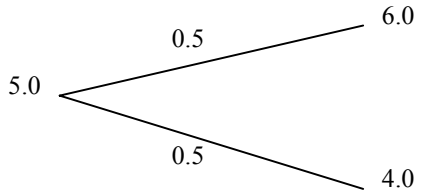


Fig. 1. The fitness movement

From this example, it means that the average fitness value in the second generation is 6.0 with probability 0.5 and 4.0 with probability 0.5.

2) Defining the Value Function of Option

In this step, we formulate the function that indicates the value of a solution in each generation. The termination payoff and the computational cost is defined specific to the problem.

For example, assume that one fitness value is worth 100 dollars and computing one generation costs 50 dollars. The value function of option is

$$F(x) = \max\{\text{ fitness\_value} * 100,\ \mathcal{E}[F(x') \mid x] - 50\ \}.$$

The first term of the maximization is the value if the algorithm stops now; thus, we receive the outcome that is the value of the current fitness value. The second term is the value if the algorithm continues. We choose the maximum of these terms as a policy to stop or continue the algorithm, when we reach x.

3) Calculating the Option Value According to the Value Function of Option

Using the probability distribution of the fitness value in step 1) and the value function of option in step 2), we can

calculate the option value in each generation by working backward from the last time step. The option value of the above example is shown in Fig. 2.

Given that the termination payoffs in the last time step are 600 and 400 dollars when the fitness values are 6.0 and 4.0 respectively, we work backward one time step. In this generation, the termination payoff is 500 dollars for the fitness value of 5.0 whereas the continuation value is 450 dollars. Therefore, the algorithm should stop and gets 500 dollars.

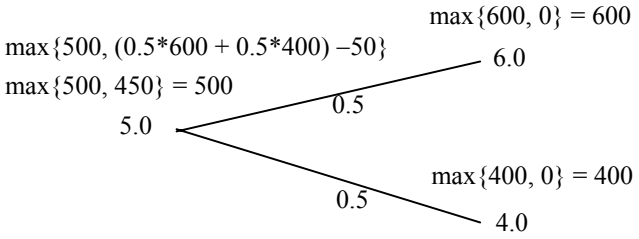


Fig. 2. Calculating option values

4) Summarizing an Option Value and an Exercise Policy

From step 3, we obtain the maximum values that may arise from stopping or continuing the algorithm. The underlying values, where the termination is optimal on one side and continuation is on the other, produce the boundary which forms the exercise region

V. The Optimal Stopping Criterion of Deceptive Problem

Deceptive problem [19] is a difficult test problem for a genetic algorithm. The general *k*-bit trap function is defined as:

$$F_k(b_0 \ldots b_{k-1}) = \begin{cases} f_{\text{high}} & ; \quad \text{if } u = k \\ f_{\text{low}} - u\dfrac{f_{\text{low}}}{k-1} & ; \quad \text{otherwise} \end{cases} \qquad (3)$$

where $b_i \in \{0, 1\}$, $u = \sum_{i=0}^{k-1} b_i$, and $f_{\text{high}} > f_{\text{low}}$. Usually, f_{high} is set at k and f_{low} is set at k-1. The test function $m \times F_k$ is defined as:

$$m \times F_k(B_0 \ldots B_{m-1}) = \sum_{i=0}^{m-1} F_k(B_i),\ B_i \in \{0,1\}^k \qquad (4)$$

This function fools gradient-based optimizers to favor zeroes, but the optimal solution is composed of all ones. The k and m may vary to produce a number of test functions.

In this section, we apply the methodology described in section IV to model the optimal stopping policy of trap problem. We study the stopping regions on problems of 3 and 4-trap and vary size with 5, 10 and 15 copies. A generic function that represents the behavior of this problem is then formulated. The experiments of using the proposed criterion on various problem sizes, such as 5-trap, are also provided in the next section.

As mentioned earlier, to find the optimal stopping time using the real options approach, the underlying uncertainty of the problem and the value function of option must be defined. In section *A*, we describe how to model underlying uncertainty of the compact genetic algorithm. Section *B* provides the value function of option. Exercise regions and the stopping function of trap problem are presented in section *C*.

A. Modeling Underlying Uncertainty

The underlying uncertainty of the compact genetic algorithm is naturally its fitness value. According to the algorithm, when a candidate solution is sampled, it is evaluated and the fitness value is assigned. In order to characterize change in the fitness value in the compact genetic algorithm, the algorithm is simulated many times, and statistics of the fitness movement are collected. Generally, the fitness value will increase over time, as the solution is evolved.

To model uncertainty in the real options application, the general process is to identify the key uncertainties and to model them using a stochastic process that fits the problem, such as a geometric Brownian motion or a mean-reverting process. In the compact genetic algorithm, however, the uncertainty can be viewed as the change of the fitness value in each step. We can find the probability of occurrence of these values and use it to characterize the underlying uncertainty of the genetic algorithm.

In the preliminary experiment, we model the uncertainty of the compact genetic algorithm by observing the fitness values from many runs and keeping track of them over time. Because the underlying uncertainty of this problem can be automatically obtained by simulating the compact genetic algorithm, we do not need to employ any particular stochastic process, such as a geometric Brownian motion or a mean-reverting process. We can construct a probability tree of the compact genetic algorithm straightforwardly. Using this method, real options can be applied to a wide variety of applications that use the learning method including the genetic algorithms.

By running the compact genetic algorithm, we have fitness values in each generation (time step). We accumulate the possible changes of fitness values in each generation over many runs and then calculate the probability of all possible values in each state. For example, in a 5 x 3-trap problem, the possible average values are 0.0, 0.5, 1.0, .., 14.0, 14.5, and 15.0. The step size of these values is 0.5 because the compact genetic algorithm has population of two, so the average fitness of two individuals ends with .0 or .5.

Therefore, this problem has 31 possible values. Fig. 3 shows the lattice of all possible values along with their associated transitional probability.

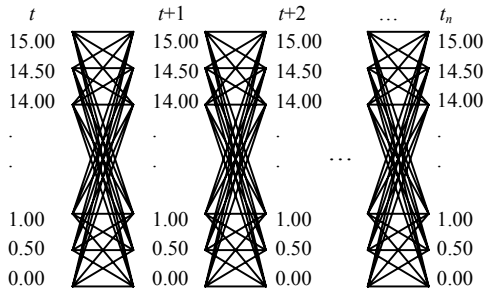


Fig. 3. Lattice of a 5 x 3-trap problem

B. Value Function of Option

The underlying uncertainty of the compact genetic algorithm depends on a probability vector. In each time step, two individuals are sampled from the distribution and fitness values of these candidates are assigned by the evaluation routine. The probability vector drives these values, and the algorithm uses these values to update the probability vector according to the best candidate. A certain cost per one sampling is assigned in order to account for an effort spent in running the algorithm. The average fitness of these candidates is used as a representative fitness value. As the candidate solutions are sampled from the probability vector, there is a chance that one sampling is good and the other is bad. Therefore, we use the average value to be a representative of the information in order to neutralize the event.

Let $\pi(x)$ denote the profit, and $\Omega(x)$ is the termination payoff. We apply the Bellman equation (2), where the value function is

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

The termination payoff is shown in (5)

$$\Omega(x) = g(x) * v \tag{5}$$

where $g(x)$ is the fitness value of x, and v is the price. We illustrate the method with a simple example. In this case, there is no profit and discounting. Equation (2) becomes

$$F(x) = \max \left\{ \Omega(x),\ \varepsilon[F(x') \mid x] \right\}. \tag{6}$$

Note that in this experiment we do not use the discount factor because in each state the compact genetic algorithm takes a few milliseconds to run; thus, the future value is not distinguishable from the present value. We also ignore the profit term $\pi(x)$ because the compact genetic algorithm does not produce any immediate profit flow. The solution value is obtained from the fitness value at the time the algorithm terminates.

To implement this idea, we assume that one fitness value is worth one unit price and no computation cost. We formulate the option value of this case as below:

$$F(x) = \max \{ g(x),\ \varepsilon[F(x') \mid x] \}. \tag{7}$$

In this trap problem, we assume artificial values in order to test the model. However, in the real-world problem, the fitness value's worth and the algorithm cost can be determined according to the application. For example, in a bin packing problem, we know how a profit depends on the number of pieces packed into the bin. Thus, equation (7) can be adapted to real-world parameter values.

C. Formulating the stopping function

We construct the stopping regions on problem of 3 and 4-trap using the methodology mentioned earlier. The exercise regions are shown in Fig. 4.

To formulate the stopping boundary of the trap problem, first, we solve this problem with the compact genetic algorithm and keep track of the probability distribution of each fitness value over time. The probabilities are averaged over 10,000 runs. We use these data to construct a lattice of the fitness distribution. Second, we calculate an option value according to (6) using a dynamic programming approach. The option values are averaged over 100 runs. Finally, we summarize an option value and an exercise policy.

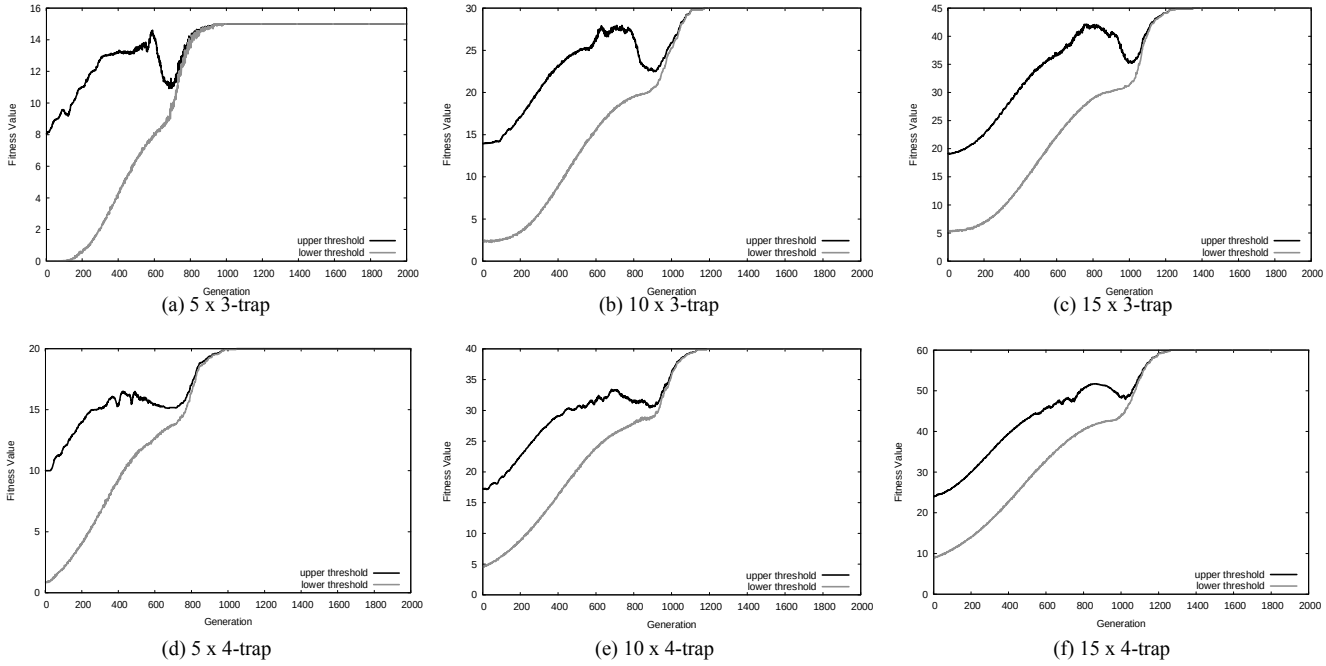
Using the behavior from 5, 10 and 15 copies of 3 and 4 trap, the generic function that represents the stopping criterion of trap problem is formulated. The boundary is approximated with a linear function by calculating slope from any two points. We denote some coordinates in order to represent intervals of the function in Fig. 5.

This stopping policy is formulated as a function of chromosome length (l) and generation (x) using the slope-intercept form of the line in (8).

$$y = mx + c \tag{8}$$

where m is slope
c is y-intercept

In case that the upper bound is approximated from coordinate $(0,\ \alpha)$ and $(\alpha_1,\ \beta_1)$, the slope is $(\beta_1 - \alpha)/\alpha_1$. Y-intercept of this function is α obviously.



(a) 5 x 3-trap (b) 10 x 3-trap (c) 15 x 3-trap

(d) 5 x 4-trap (e) 10 x 4-trap (f) 15 x 4-trap

Fig. 4. The plots illustrate the exercise regions of running trap problem on 5, 10 and 15 copies of 3 and 4 trap. On the top, (a) – (c) are 5 x 3, 10 x 3 and 15 x 3 trap problem. On the bottom, (d) – (f) are 5 x 4, 10 x 4 and 15 x 4 trap problrm. The stopping regions are the area where the fitness value rises above the upper threshold and where the fitness value falls below the lower threshold. If the fitness value falls into the stopping region, the algorithm should decide to stop because with the current population, it is unlikely to achieve a better result.

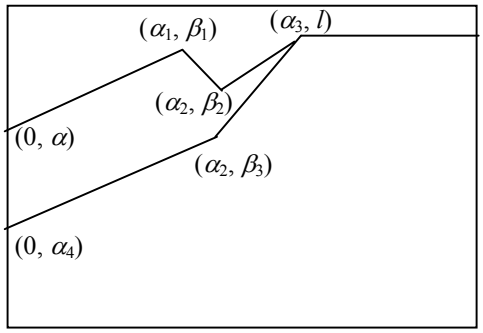


Fig. 5. Points' notation

Substitute these into equation (8), the upper bound where x less than α_1 is obtained. If $\alpha_1 \le x < \alpha_2$, the upper bound is approximated from (α_1, β_1) and (α_2, β_2), this bound is $\frac{\beta_2 - \beta_1}{\alpha_2 - \alpha_1}(x - \alpha_1) + \beta_1$. When $\alpha_2 \le x < \alpha_3$, the upper bound is approximated from (α_2, β_2) and (α_3, l), this bound is $\frac{l - \beta_2}{\alpha_3 - \alpha_2}(x - \alpha_2) + \beta_2$. The values of α and β are estimated from data in Fig. 4 when the sizes of problem change. It is interesting to note that for a trap problem, this trend is linear. We formulate the upper bound function in (9), where k is trap size, m is the number of trap copies and l is chromosome length.

$$upper\_bound = \begin{cases} \frac{(\beta_1 - \alpha)x + \alpha\alpha_1}{\alpha_1} & ; x < \alpha_1 \\ \frac{\beta_2 - \beta_1}{\alpha_2 - \alpha_1}(x - \alpha_1) + \beta_1 & ; \alpha_1 \le x < \alpha_2 \\ \frac{l - \beta_2}{\alpha_3 - \alpha_2}(x - \alpha_2) + \beta_2 & ; \alpha_2 \le x < \alpha_3 \end{cases} \quad (9)$$

Using the same method, the lower bound function is shown in (10).

$$lower\_bound = \begin{cases} \frac{(\beta_3 - \alpha_4)x + \alpha_4\alpha_2}{\alpha_2} & ; x < \alpha_2 \\ \frac{l - \beta_3}{\alpha_3 - \alpha_2}(x - \alpha_2) + \beta_3 & ; \alpha_2 \le x \le \alpha_3 \\ l & ; x \ge \alpha_3 \end{cases} \quad (10)$$

where $\alpha = (0.37m + 0.88)k$
$\alpha_1 = (68.70m + 1482.99) / k$
$\alpha_2 = 32.00m + 549.33$
$\alpha_3 = 33.50m + 796.33$
$\alpha_4 = (0.18m - 0.87)k$
$\beta_1 = (0.91m + 0.23)k$
$\beta_2 = (0.81m - 0.47)k$
$\beta_3 = (0.74m - 0.63)k$

TABLE I
COMPARISON OF THE PROPOSED CRITERION

m x *k*-trap	cGA with Traditional Stopping			cGA with Early Stopping			cGA with Reset Method		
	Eval.	Best fit.	Avg. fit.	Eval.	Best fit.	Avg. fit.	Eval.	Best fit.	Avg. fit.
5 x 3-trap	971	15*	13.15	108	14	10.84	986	15*	13.16
10 x 3-trap	1345	29	25.36	78	22	18.00	1350	28	25.54
20 x 3-trap	1809	55	49.98	66	38	30.73	1801	54	49.93
30 x 3-trap	2175	81	74.74	124	71	44.36	2194	81	75.19
5 x 4-trap	889	18	15.49	483	18	14.77	917	20*	16.47
10 x 4-trap	1252	34	30.62	521	31	25.74	1329	38	31.28
20 x 4-trap	1757	66	61.66	867	64	49.99	1831	74	62.55
30 x 4-trap	2165	99	92.59	912	97	70.54	2238	100	93.61
5 x 5-trap	874	20	20.00	601	20	19.20	905	25*	20.55
10 x 5-trap	1210	40	39.99	636	41	34.10	1286	50*	40.62
20 x 5-trap	1731	82	79.95	887	81	64.17	1803	99	80.51
30 x 5-trap	2147	122	119.58	1052	121	92.88	2231	125	119.67

* Optimal solution

VI. EXPERIMENTS WITH VARIOUS SIZES

To explore how to use the proposed criterion, we apply these functions to test on various sizes of trap problem. We test the proposed criterion compared with the compact genetic algorithm that stops when the probability fully converges. Using early stopping, the algorithm stops when the fitness value falls into the stopping region. With the reset method, the algorithm stops when the probability fully converges. The results in table I are averaged over 100 runs.

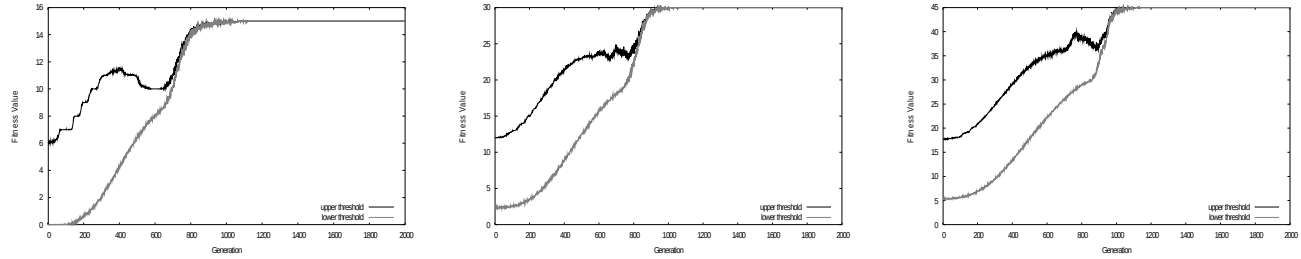
In the experiments, we employ the stopping region as a policy to stop running the algorithm. It suggests that it is unlikely to achieve a good solution when the fitness value falls into the stopping region. Thus, the algorithm will stop when an average fitness value falls into this region, and we call this an early stopping. Using early stopping can save computational effort when the fitness value is unlikely to lead to optimality. Table I shows that using the early stopping can save the number of function evaluations while the best solution is close to the compact genetic algorithm that uses traditional stopping criterion.

To improve the solution quality further, this paper also presents a technique called the reset method. This is another way of using the proposed criterion. Instead of using early stopping, the reset technique provides a chance of reaching a higher solution quality by reversing the probability vector. When the fitness values fall into the stopping region, it implies that the updating process may go wrong. Thus, we should try to search the other way. A simple approach to jump to other search points is adjusting the probability.

We suggest reversing the probability vector. Each dimension of the probability vector is reversed by 1 minus probability in that dimension. Using this approach, the algorithm can explore the solution in the counterpart and if the solution in that area is not good, the fitness value will fall into the stopping region and the probability vector is reversed again. This methodology provides an opportunity to search more candidate solutions when the model seems to go bad. From the experiment results in table 1, the reset method mostly provides a higher solution quality than the traditional compact genetic algorithm that uses traditional stopping criterion. Moreover, it tends to reach some optimal solutions when the trap size grows. This suggests that, for the trap problem, exploring different candidates when the fitness value is in the stopping region can be more effective than using the traditional or early stopping method, specifically when the problem is hard.

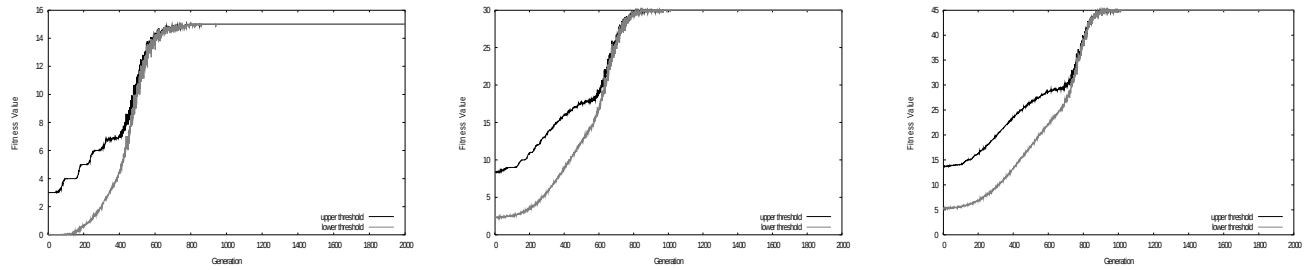
One of the insights of this study is to better understand the behavior of fitness movement in the compact genetic algorithm. The continuation region, the area between upper and lower bounds, indicates the values that have a chance to reach optimal. In the trap problem the last part of the continuation region that leads to the optimal narrows down. With fitness distribution over time, this opens up to an analysis of problem hardness. Furthermore, when the problem size grows, the exercise region tends to scale well according to the linear function in the previous section.

Obviously the exercise contours vary with the ratio of gain/cost (in eq. 6). To observe the sensitivity of these contours, we vary the computational cost where the ratio of gain/cost are 10:1 and 1:1. The results are shown in Fig. 6 and Fig. 7. They show that when the evaluating cost is expensive with respect to the gain, it is preferable to accept the solution even though they are not very good because the future computation is expensive (Fig. 7). When the sampled solution seems to be good, it should be accepted and stop running. In the case that the computation cost is low (Fig. 6) the decision is different. The continuation region is larger. There are higher chances to continue with the current solution to improve the solution because it is beneficial.



(a) 5 x 3-trap (b) 10 x 3-trap (c) 15 x 3-trap

Fig. 6. The plots illustrate the exercise regions of gain10cost1 on 5, 10 and 15 copies of 3 trap (a)-(c). The threshold is lower than running without cost.



(a) 5 x 3-trap (b) 10 x 3-trap (c) 15 x 3-trap

Fig. 7. The plots illustrate the exercise regions of gain1cost1 on 5, 10 and 15 copies of 3 trap (a)-(c). The threshold is shifted down when the cost is higher.

VII. Conclusion

The proposed stopping policy is obtained using real options analysis. It provides a boundary for deciding whether to continue or to stop the algorithm. The stopping function of the trap problem is illustrated as an example of applying this methodology. Using the proposed stopping criterion, the algorithm stops early and saves on the number of function evaluations. Furthermore, when utilizing the reset method, it can achieve a higher solution quality. The proposed methodology would be more beneficial to a harder problem. Experiments using the boundary function estimated from 3- and 4-trap shows that we can reach an optimal solution of some copies of 5-trap. The capability in solving other problems should be investigated in future works.

References

[1] P. Larranaga and J. Lozano, "An Estimation of Distribution Algorithms: A New Tool for Evolutionary Computation," Kluwer Academic Publishers, 2001.

[2] Z Michalewicz, "Genetics Algorithms + Data Structures = Evolution Programs," Springer-Verlag, NY, 1996.

[3] K. Zielinski, D. Peters and R. Laur, "Stopping criteria for single-objective optimization," in Proceeding of the Third International Conference on Computational Intelligence, Robotics and Autonomous Systems, 2005.

[4] M. Safe, J. Carballido, I. Ponzoni and N. Brignole, "On stopping criteria for genetic algorithms," SBIA, 2004.

[5] G. Hernandez, K. Wilder, F. Nino and J. Garcia, "Toward a self-stopping evolutionary algorithm using coupling from the past," in Proceeding of the 2005 Conference on Genetic and Evolutionary Computation, 2005.

[6] L. Meyer and X. Feng, "A fuzzy stop criterion for genetic algorithms using performance estimation," In proceedings of the Third IEEE Conference on Fuzzy Systems, 1994.

[7] H. Aytug and G. J. Koehler, "Stopping criterion for finite length genetic algorithms," INFORMS Journal on Computing, 1996.

[8] H. Aytug and G. J. Koehler, "New stopping criterion for genetic algorithm," European Journal of Operational Research, 2000.

[9] A. E. Nix and M. D. Vose, "Modeling genetic algorithms with Markov chains," Annals Mathematics and Artificial Intelligence, 1992.

[10] P. C. Pendharkar and G. J. Koehler, "A general steady state distribution based stopping criteria for finite length genetic algorithms," European Journal of Operational Research, 2006.

[11] M. Hulin, "An optimal stop criterion for genetic algorithms: A bayesian approach," in Proceeding of the International conference on Genetic Algorithms, 1997.

[12] S. Rimcharoen, D. Sutivong and P. Chongstitvatana, "Real option approach to finding optimal stopping time in compact genetic algorithm," In Proceeding of IEEE International Conference on Systems, Man, and Cybernetics, 2006.

[13] G. R. Harik, F. G. Lobo and D. E. Goldberg, "The compact genetic algorithm," IEEE Trans. on Evolutionary Computation, 1999.

[14] F. Black and M. Scholes, "The pricing of options and corporate liabilities," Journal of Political Economy, 1973, 81: 637-654.

[15] R. C. Merton, "Theory of rational option pricing," Bell Journal of Economics and Management Science, 1973, 4: 141-183.

[16] J. C. Cox, S. A. Ross and M. Rubinstein, "Option pricing: a simplified approach," Journal of Financial Economics, 1979.

[17] S. C. Myers, "Determinants of corporate borrowing," Journal of Financial Economics, 1977, 5(2): 147-175.

[18] A. K. Dixit and R. S. Pindyck, "Investment under uncertainty," Princeton University Press, Princeton, NJ, 1994.

[19] D. E. Goldberg, "Simple genetic algorithms and the minimal, deceptive problem," In L. Davis (ed.), Genetic Algorithms and Simulated Annealing, Morgan Kaufmann Publisher, 1987, 74-88.