13

Performance of Varying Genetic Algorithm Techniques in Online Auction

Kim Soon Gan, Patricia Anthony, Jason Teo and Kim On Chin
Universiti Malaysia Sabah, School of Engineering and Information Technology, Sabah Malaysia

1. Introduction

Genetic algorithm is one of the successful optimization algorithm used in computing to find exact or approximate solutions for certain complex problems. This novel algorithm was first introduced by John Holland in 1975 (Holland, 1975). Besides Holland, many other researchers have also contributed to genetic algorithm (Davis, 1987; Davis, 1991; Grefenstte, 1986; Goldberg, 1989; Michalewicz, 1992). This is an algorithm that imitates the evolutionary process concept based on the Darwinian Theory which emphasizes on the law of "the survival of the fittest". This algorithm used techniques which are inspired from evolution biology such as inheritance, selection, crossover and mutation (Engelbrecht, 2002).

There are several important components in genetic algorithm which includes representation, fitness function, and selection operators (parent selection and survivor selection, crossover operator and mutation operator). Genetic algorithm starts by generating an initial population of individuals randomly. The individuals are represented as a set of parameter which is the solution to the problem domain. Normally, individuals are fixed length binary string. The individuals are then evaluated using fitness functions. The evaluation will give a fitness score to individuals indicating how well the solutions perform in the problem domain. The individuals that have been evaluated using the fitness function will be selected to be parents to produce offspring through the crossover and mutation operators. The genetic algorithms will repeat the above process except for the population initialization until the termination criteria is met. Fig. 1 shows the structure of a genetic algorithm.

GAs have been applied successfully in many applications including job shop scheduling (Uckun *et al.* 1993), the automated design of fuzzy logic controllers and systems (Karr 1991; Lee & Takagi, 1993), hardware-software co-design and VLSI design (Catania *et al.* 1997; Chandrasekharam *et al.* 1993). In this chapter, variations of genetic algorithms are applied in optimizing the bidding strategies for a dynamic online auctions environment.

Auction is defined as a bidding mechanism and is expressed by a set of auction rules that specify how the winner is determined and how much he or she has to pay (Wolfstetter, 2002). Jansen defines an online auction as an Internet-based version of a traditional auction (Jansen, 2003). In today's e-commerce market, online auction has acted as an important tool

in the services for procuring goods and items either for commercialize purposed or for personal used. Online auctions have been reported as one of the most popular and effective ways of trading goods over the Internet (Bapna *et al*. 2001). Electronic devices, books, computer software, and hardware are among the thousands items sold in the online auctions every day. To date, there are 2557 auction houses that conduct online auctions as listed on the Internet (Internet Auction List, 2011). These auction houses conduct different types of auctions according to a variety of rules and protocols. eBay, as one of the largest auction house alone has more than 94 million registered users and had transacted more than USD 92 billion worth of goods during 2010 (eBay, 2010). These figures clearly show the importance of online auctions as an essential method for procuring goods in today's e-commerce market.

```
Begin
Generation = 0
Randomly Initialize Population
While termination criteria are not met
      Evaluate Population Fitness
      Crossover Process
      Mutation Process
      Select new population
      Generation = Generation + 1
End
```

Fig. 1. The structure of a Genetic Algorithm

The auction environment is highly dynamic in nature. Since there are a large number of online auction sites that can be readily accessed, bidders are not constrained to participate in only one auction; they can bid across several alternative auctions for the same good simultaneously. As the number of auction increases, difficulties such as monitoring the process of auction, tracking bid and bidding in multiple auctions arise when the number of auctions increases. The user needs to monitor many auctions sites, pick the right auction to participate, and make the right bid in order to have the desired item. All of these tasks are somewhat complex and time consuming. The task gets even more complicated when there are different start and end times and when the auctions employ different protocols. For this reasons, a variety of software support tools are provided either by the online auction hosts or by third parties that can be used to assist consumers when bidding in online auctions.

The software tools include automated bidding software, bid sniping software, and auction search engines. Automated bidding software or proxy bidders act on the bidder's behalf and place bids according to a strict set of rules and predefined parameters. Bid sniping software, on the other hand, is a practice of placing of bid a few minutes or seconds before an auction closes. These kinds of software, however, have some shortcomings. Firstly, they are only available for an auction with a particular protocol. Secondly, they can only remain in the same auction site and will not move to other auction sites. Lastly, they still need the intervention of the user, that is, the user still needs to make decision on the starting bid (initially) and the bid increments.

To address the shortcomings mentioned above, an autonomous agent was developed that can participate in multiple heterogeneous auctions. It is empowered with trading capabilities and it is able to make purchases autonomously (Anthony, 2003; Anthony & Jennings, 2003b). Two primary values that heavily influenced the bidding strategies of this agent are the k and β. These two values correspond to the polynomial function of the four bidding constraints, namely the remaining time left, the remaining auction left, the user's desire for bargain and the user's level of desperateness. Further details on the strategies will be discussed in Section 3. The k value ranges from 0 to 1 while the β value is from 0.005 to 1000. The possible combinations between these two values are endless and thus, the search space for the solution strategies is very large. Hence, genetic algorithms were used to find the nearly optimal bidding strategy for a given auction environment.

This work is an extension of the solution above, which has been successfully employed to evolve effective bidding strategies for particular classes of environment. This work is to improve the existing bidding strategy through the optimization techniques. Three different variations of genetic algorithm techniques are used to evolve the bidding strategies in order to search for the nearly optimal bidding solution. The three techniques are parameter tuning, deterministic dynamic adaptation, and self-adaptation. Each of this method will be detailed in Section 4, 5 and 6. The remainder of the chapter is organized as follow. Section 2 discusses related work. The bidding strategy framework is discussed in Section 3. The parameter tuning experiment is discussed in Section 4. Section 5 and 6 discussed the deterministic adaptive experiment and self-adaptive experiment. A comparison between all the schemes is discussed in Section 7. Finally, the conclusion is discussed in Section 8.

2. Related work

Genetic algorithm has shown to perform well in the complex system by which the old search algorithm has been solved. This is due to the nature of the algorithms that is able to discover optimal areas in a large search space with little priori information. Many researches in auctions have used genetic algorithm to design or enhance the auction's bidding strategies. The following section discusses works related to evolving bidding strategies.

An evolutionary approach was proposed by Babanov (2003) to study the interaction of strategic agents with the electronic marketplace. This work describes the agents' strategies based on different methodologies that employ incompatible rules in collecting information and reproduction. This work used the information collected from the evolutionary framework for economic studies as many researches have attempted to use evolutionary frameworks for economics studies (Nelson, 1995; Epstein & Axtell, 1996; Roth, 2002; Tesfatsion, 2002). This evolutionary approach allows the strategies to be heterogeneous rather than homogenous since only a particular evolutionary approach is applied. This work has shown that the heterogeneous strategies evolved from this framework can be used as a useful research data.

ZIP, introduced by Cliff, is an artificial trading agent that uses simple machine learning to adapt and operate as buyers or sellers in online open-outcry auction market environments (Cliff, 1997). The market environments are similar to those used in Smith's (Smith, 1962)

experimental economics studies of the CDA and other auction mechanisms. The aim of each zip agent is to maximize the profit generated by trading in the market. A standard genetic algorithm is then applied to optimize the values of the eight parameters governing the behavior of the ZIP traders which previously must be set manually. The result showed that GA-optimized traders performed better than those populated by ZIP traders with manually set parameter values (Cliff, 1998a; Cliff, 1998b). This work is then extended to 60 parameters to be set correctly. The experiment showed promising result when compared to the ZIP traders with eight parameters (Cliff, 2006). Genetic algorithm is also used to optimize the auction market parameters setting. Many tests have been conducted on ZIP to improve the agent traders and the auction market mechanism using genetic algorithm (Cliff, 2002a; Cliff, 2002b). Thus, ZIP was able to demonstrate that genetic algorithm can perform well in evolving the parameters of bidding agents and the strategies.

In another investigation, a UDA (utility-based double auction) mechanism is presented (Choi et, al. 2008). In UDA, a flexible synchronous double auction is implemented where the auctioneer maximizes all traders' diverse and complex utility functions through optimization modeling based on genetic algorithm. It is a double auction mechanism based on dynamic utility function integrating the notion of utility function and genetic algorithm. The GA-optimizer is used to maximize total utility function, composed of all participants' dynamic utility functions, and matches the buyers and sellers. Based on the experimental result, it performance is better than a conventional double auction.

3. The bidding strategy framework

As mentioned, this work is an extension of Anthony's work (Anthony, 2003) to tackle the problem of bidding in multiple auctions that employ varying auctions protocols. This section details the electronic marketplace simulation, the bidding strategies and the genetic algorithm implemented in the previous work.

3.1 The electronic market place simulation

The market simulation employed three different auction protocols, English, Vickrey and Dutch that run simultaneously in order to simulate the real auction environment. The market simulation is used in this work to evaluate the performance of the evolved bidding strategies. The following section explains how the market simulation works.

The marketplace simulator shown in Fig. 2 consists of concurrent running auctions that employ different protocols. These protocols are English, Dutch and Vickrey. All of these auctions have a known starting time and only English and Vickrey auctions have a known ending time. The bidding agent is given a deadline (t_{max}) by when it must obtain the desired item and it is told about the consumer's private valuation (p_r) for this item. The agent must only buy an instance of the desired item.

The marketplace announces the current bid values and the current highest bids for English auctions and the current offers for Dutch auctions at each time step. At the end of a given auction, it determines the winning bid and announces the winner. This set of information is used by the agent when deciding in which auction to participate, at what value to bid and in which time to bid.

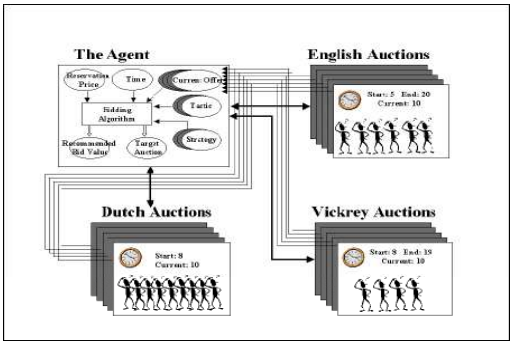


Fig. 2. The Marketplace Simulator

3.2 Bidding strategy

The bidding algorithm for this framework is shown in Fig. 3. Let *Item_NA* be a boolean flag to indicate whether the target item has already been purchased by the agent. Assume that the value of p_r is based on the current reliable market prices observed from past auctions and that the marketplace is offering the item which the agent is interested in. While the bidder agent has not obtained the desired item, the bidder agent needs to build an active auctions list in order to keep track of the current active auction. Active auction is defined as auction that is ongoing or just started but has not reach the ending time yet.

```
while (t ≤ tmax ) and (Item_NA = true)
    Build active auction list;
    Calculate current maximum bid using the agent's strategy;
    Select potential auctions to bid in, from active auction list;
    Select target auction as one that maximizes agent's expected utility;
    Bid in the target auction using current maximum bid as reservation price at this time;
Endwhile
```

Fig. 3. The bidding agent's algorithm

```
for all i ϵ A
    if ((t ≥ σi ) and (t ≤ ηi ) or (Si (t) = ongoing)
        add i to L(t)
    endif
endfor
```

Fig. 4. Building active auction list algorithms

In order to build the active auction list, the bidder agent follows the algorithm as shown in Fig. 4. $S_i(t)$ is a boolean flag representing the status of auction i at time t, such that $i \in A$ and $S_i(t) \in$ *(ongoing; completed)*. Each auction $i \in A$, has a starting time σ_i, and its own ending time η_i. The active auction list is built by taking all the auctions that are currently running at time t. In English and Vickrey auctions, any auction that has started but has not reached its ending time is considered as active. $S_i(t)$ is used in Dutch auctions since the ending time of this type of auction is not fixed.

After the bidder agent builds the active auctions list, the bidder agent will start calculating the current maximum bid based on the agent strategy. The current maximum bid is defined as the amount of the agent willing to bid at the current time that is lesser than or equal to the agent's private valuation. Four bidding constraints are used to determine the current maximum bid namely the remaining time left, the remaining auction left, the desire for bargain and the level of desperateness.

The remaining time tactic considers the amount of bidding time the bidder agent has to obtain the desire item. This tactic determines the bid value based on the bidding time left. Assuming that the bidding time t is between 0 and t_{max} ($0 \le t \le t_{max}$), the current bid value is calculated based on the following expression:

$$f_{rt} = \alpha_{rt}(t) P_r \tag{1}$$

where $\alpha_{rt}(t)$ is a polynomial function of the form:

$$\alpha_{rt}(t) = k_{rt} + (1 - k_{rt}) \left(\frac{t}{t_{\max}} \right)^{\frac{1}{\beta}} \tag{2}$$

This function is a time dependent polynomial function where the main consideration is the time left from the maximum time allocated. k_{rt} is a constant that determines the value of the starting bid of the agent in any auction multiplied by the size of the interval. This time dependent functions can be defined as those that start bidding near p_r rapidly to those only bid near p_r right at the end along with all the possibilities in between with variation of the value $\alpha_{rt}(t)$. Different shapes of curve can be obtained by varying the values of β by using the equation defined above. There are unlimited numbers of possible tactics for each value of β. In this tactic, β value is defined between $0.005 \le \beta \le 1000$. It is possible to have two different behaviors for β. When $\beta < 1$, the tactic will bid with a low value until the deadline is almost reached, whereby this tactic concedes by suggesting the private valuation as the recommended bid value. When $\beta > 1$, the tactic starts with a bid value close to the private valuation and quickly reaches the private valuation long before the deadline is reached. Fig. 5 shows the different shape of the curves with varying β values.

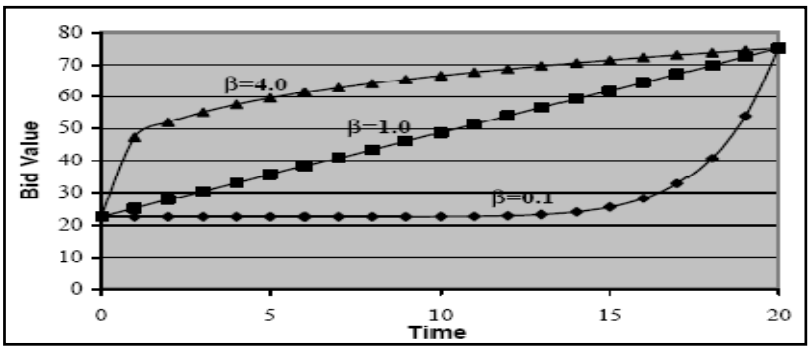


Fig. 5. The curve with varying β value. (Anthony, 2003)

The remaining auction left tactic, on the other hand, considers the number of remaining auctions that the bidder agent is able to participate in order to obtain the item. This tactic bids closer to p_r as the number of the remaining auctions decreases when the bidder agent is running out of opportunities to obtain the desired item. The current bid value is calculated based on the following expression:

$$f_{ra} = \alpha_{ra}(t) p_r \tag{3}$$

where $\alpha_{ra}(t)$ is a polynomial function of form:

$$\alpha_{ra} = k_{ra} + (1 - k_{ra}) \left(\frac{c(t)}{|A|} \right)^{\frac{1}{\beta}} \tag{4}$$

The polynomial function α_{ra} is quite similar to the terms use in α_{rt}, whereby the only difference between the two function is the *c(t)*. *c(t)* is a list of auctions that have been completed between time 0 and t. The β value for this tactic is identical to the remaining time tactic between $0.005 \leq \beta \leq 1000$.

The desire for a bargain tactic is the bidder agent that is interested in getting a bargain for obtaining the desired item. In this scenario, the bidder agent needs to take into account all the ongoing auctions and the time left to obtain the item. The current bid value is calculated based on the following expression:

$$f_{ba} = \omega(t) + \alpha_{ba}(t)\left(p_r - \omega(t)\right) \tag{5}$$

In the expression above, the variable $\omega(t)$ takes into account all the ongoing auctions along with the current bid value. The Dutch and English are considered solely in this expression as only these two auctions have current bid value. As a consequence, the minimum bid value is calculated based on the current bid value and also the proportion of the time left in the auction. These values are summed and averaged with respect to the number of active auctions at that particular time. The expression for $\omega(t)$ is calculated based on the formula as below:

$$\omega(t) = \frac{1}{|L(t)|} \left(\sum_{1 \leq i \leq L(t)} \frac{t - \sigma_i}{\eta_i - \sigma_i} v_i(t) \right) \tag{6}$$

where v_i is the current highest bid value in an auction *I* at time *t,* and $I \in L(t)$;

σ_i, and η_i is the start and end time of auction *i*

The expression for $\alpha_{ba}(t)$ is defined as:

$$\alpha_{ba}(t) = k_{ba} + (1 - k_{ba}) \left(\frac{t}{t_{\max}} \right)^{\frac{1}{\beta}} \tag{7}$$

The valid range for the constant k_{ba} is $0.1 \leq k_{ba} \leq 0.3$ and the β value is $0.005 \leq \beta \leq 0.5$. The β value is lower than 1 as bidder agent that is looking for bargain will never bid with the behavior of $\beta > 1$. The β value is, therefore, constantly lower than 1 in order to maintain a low bid until the closes to the end time. Hence, the value of $\beta < 0.5$ is used.

The level of desperateness tactic is the bidder agent's desperateness to obtain the target item within a given period and thus, the bidder agent who possesses this behavior tend to bid aggressively. This tactic utilizes the same minimum bid value and the polynomial function as the desire for bargain tactic but with a minor variation to the β and k_{de} value. The valid range for the constant k_{de} for this tactic is $0.7 \leq k_{de} \leq 0.9$ while the β value is $1.67 \leq \beta \leq 1000$. The β value is higher than 1 in this case as the bidder agent that is looking for bargain will never bid with the behavior of $\beta < 1$. As a result, the β value is always higher than 0.7 since the bidder agent will bid close to the private valuation.

There is a weight associated to each of this tactic and this weight is to emphasize which combination of tactics that will be used to bid in the online auction. The final current maximum bid is based on the combination of the four tactics by making use of the weight. Fig. 6 shows various combinations of the bidding constraints based on the different weight associated to the bidding tactics. It can also be seen that different bidding patterns are generated by varying the value of weights of the bidding constraints.

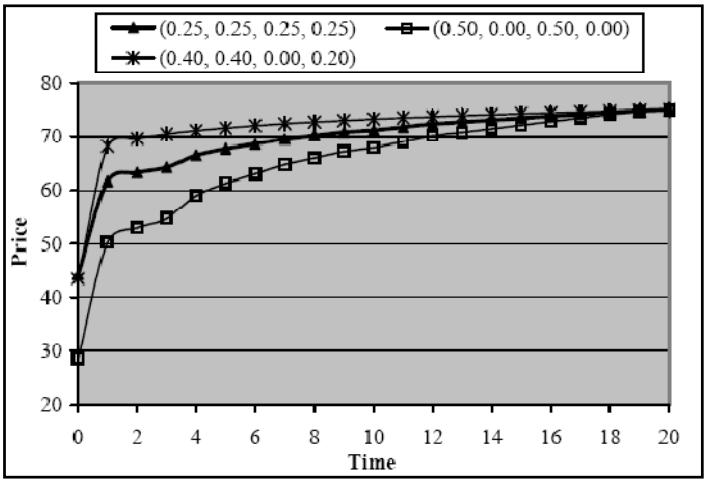


Fig. 6. Various combinations of the bidding constraints

3.3 Genetic algorithm

3.3.1 Representation

Floating point encoding is applied in this particular work as floating point encoding has shown to produce faster, more consistent and more accurate results (Janikow & Michalewicz, 1991). The floating encoding is, therefore, represented using an array of structure. The individuals that are represented in a floating point array structure are shown in Table 1.

p_r	Agent's private valuation
t_{max}	Deadline given to the agent to obtain the desired item
k_{rt}	k for the remaining time tactic
β_{rt}	β for the remaining time tactic
k_{ra}	k for the remaining auction tactic
β_{ra}	β for the remaining auction tactic
k_{ba}	k for the desire for a bargain tactic
β_{ba}	β for the desire for a bargain tactic
k_{de}	k for the desperateness tactic
β_{de}	β for the desperateness tactic
w_{rt}	Relative weight for the remaining time tactic
w_{ra}	Relative weight for the remaining auction tactic
w_{ba}	Relative weight for the desire for a bargain tactic
w_{de}	Relative weight for the desperateness tactic
fitness	Fitness score for the individual

Table 1. Bidding strategies representation

3.3.2 Representation

Fitness function is an objective function that quantifies the optimality of a solution in a genetic algorithm so that the particular chromosome may be ranked against all the other chromosomes. The main focus of the strategies evaluation in this work is the success rate and average utility of the strategies. Three fitness equations are used to evaluate the performance of the strategies namely the success rate, the agent's utility function and agent's utility with penalty. The success rate is the rate in obtaining the desired item and the second fitness function is the agent's utility

$$U_i(v) = \left(\frac{p_r - v}{p_r} \right) + c \tag{8}$$

where v represents the winning bid and c is an arbitrary constant 0.001 to ensure that the agent receives some value when the winning bid is equivalent to its private valuation. The third fitness equation involves a variation of the agent utility. If the agent fails to get the item, a penalty that ranges from 0.01 to 0.05 is incurred. Basically, Fitness Equation 1 is used if the delivery of the item is of utmost importance to the user. Fitness Equation 2 is used when the agent is looking for a bargain. Fitness Equation 3 is used when both the delivery of the item and looking for a bargain are equally important. The fitness score is then computed by taking the average utility from a total of 2000 runs.

3.3.3 Selection operators

Elitism is an operator used to retain some number of the best individuals in each generation to the next generation in order to ensure that the fittest individual is not lost during the evolution process (Obitko, 1998). Elitism is applied in this work to retain ten percent of the

best individuals to the new population and to ensure that a significant number of the fitter individuals will make it to the next generation. Tournament selection is applied in the genetic algorithm for selecting the individuals to the mating pools for the remaining ninety percent of the population (Blickle & Thiele, 2001). Tournament selection technique was chosen because it is known to perform well in allowing a diverse range of fitter individuals to populate the mating pool (Blickle & Thiele, 1995). By implementing the tournament selection, fitter individuals can contribute to the next generation genetic construction and the best individual will not dominate in the reproduction process compared to the proportional selection.

3.3.4 Crossover process

The extension operator floating point crossover operator is used this work (Beasley *et al.* 1993b). This operator works by taking the differences between the two values, adding it to the higher value (giving the maximum range), and subtracting it from the lower value (giving the minimum range). The new values for the genes are then generated between the minimum and the maximum range that were derived using this operator (Anthony & Jennings, 2002).

3.3.5 Mutation process

Since the encoding is a floating point, the mutation operator used in this work must be a non-binary mutation operator. Beasley has suggested a few non-binary mutation operators such as random replacement, creep operator and geometric creep (Beasley *et al.* 1993b) that can be used. The *creep* operator which adds or subtracts a small randomly generated amount from selected gene is used to allow a small constant of 0.05 to be added or subtracted from the selected gene depending on the range limitation of the parameter (Anthony & Jennings, 2002).

3.3.6 Stopping criteria

The genetic algorithm will repeat the process until the termination criteria are met. In this work, the evolution stops after 50 iterations. An extensive experiment was conducted to determine the point at which the population converges. It was decided to choose 50 as the stopping criterion since it is was observed that the population will always converge before or at the end of the 50 iterations.

Anthony's work has some shortcoming where the crossover and mutation rate used in the work is based on literature review recommended values. However, researches have shown that the crossover rate and mutation rate applied in the application are application dependent, thus, simulation need to be conducted in order to find the suitable crossover and mutation rate. Besides that, other variations of genetic algorithm have proven to perform better that traditional genetic algorithm which is worthwhile to be investigated.

4. Parameter tuning

Many researchers such De Jong, Grefenstte, Schaffer and others have contributed considerable efforts into finding the parameters values which are good for a number of

numerical test problems. The evolution of the bidding strategies by Anthony and Jennings (Anthony & Jennings, 2002) employed a fixed crossover and mutation probability based on the literatures. However, these recommended values may not perform at its best in the genetic algorithm as it has been proven that the parameter values are dependent on the nature of problems to be solved (Engelbrecht, 2002). In this experiment, the crossover and mutation rates are fine tuned with different combination of probabilities in order to discover the best combination of genetic operators' probabilities. Thus, the main objective of this experiment is to improve the effectiveness of the bidding strategies by "hand tuning" the values of the crossover rate and mutation rate to allow a new combination of static crossover and mutation rates to be discovered. By improving the algorithm, more effective bidding strategies can be found during the exploration of the solution.

The experiment is subdivided to two parts. The first one varies the crossover rate and the second one varies the mutation rate. At the end of this experiment, the combination rate discovered is compared and empirically evaluated with the bidding strategies evolved in Anthony's work (Anthony, 2003).

4.1 Experimental setup

Table 2 and 3 show the evolutionary and parameter setting for the genetic algorithm. The parameters setting in the simulated environment for the empirical evaluations are shown in Table 4. These parameters include the agent's reservation price; the agent's bidding time and the number of active auctions. The agent's reservation price is the maximum amount that the agent is willing to pay for the item while the bidding time is the time allocated for the agent to obtain the user's required item. The active auctions are the list of auctions that is ongoing before time $t_{max.}$ Fig. 7 shows the pseudocode of the genetic algorithm.

Representation	Real Values Number
Crossover	Extension Combination Operator
Mutation	Creep Operator
Selection	Tournament Selection

Table 2. Genetic algorithm evolutionary setting

Number of Generations	50
Number of Individuals	50
Elitism	10%
Crossover Probability	0.2, 0.4, 0.6, 0.8
Mutation Probability	0.2, 0.02, 0.002
Termination Criteria	After 50 Generation
Number of Run	10

Table 3. Genetic algorithm parameter setting

Agent reservation price	$73 \leq p_r \leq 79$
Bidding time for each auction	$21 \leq t_{max} \leq 50$
Number of active auction	$20 \leq L(t) \leq 45$

Table 4. Configurable parameters for the simulated marketplace

```
Begin
   Randomly create initial bidder populations;
   While not (Stopping Criterion) do
     Calculate fitness of each individual by running the
     marketplace 2000 times;
     Create new population
       Select the fittest individuals (HP);
       Create mating pool for the remaining population;
       Perform crossover and mutation in the mating
       pool to create new generation(SF);
       New generation is HP + SF;
  Gen = Gen + 1
  End while
End
```

Fig. 7. Genetic algorithm

4.2 Experimental evaluation

The performance of the evolved strategies is evaluated based on three measurements. Firstly, the average fitness is the fitness of the population at each generation over 50 generations. The average fitness shows how well the strategy converges over time to find the best solution.

Secondly, success rate is the percentage of time that an agent succeeds in acquiring the item by the given time at any price less than or equal to its private valuation. This measure will determine the efficiency of the agent in terms of guaranteeing the delivery of the requested item. Individual will be selected from each of the data set to compete in the simulated marketplace for 200 times. The success is calculated based on the number of time the agent is able to win the item over 200 runs. The formula below is used to calculate the success rate.

$$\text{Success Rate} = \frac{(\text{Number of winning}) \times 200}{100} \tag{9}$$

Finally, the third measurement is the average payoff which is defined as

$$\frac{\sum_{1 \leq x \leq 100} \left(\frac{p_r - v_i}{p_r} \right)}{n} \tag{10}$$

where p_r is the agent's private valuation, n is the number of runs, v_i is the winning bid value for auction i. This value is then divided by the agent's private valuation, summed and average over the number of runs. The agent's payoff is 0 if it is not successful in obtaining the item.

A series of experiments was conducted using the set of crossover and mutation rate described in Table 2. It was found that 0.4 crossover rate and 0.02 mutation rate performed better than the other combinations (Gan *et al*, 2008a, Gan *et al*, 2008b). An experiment was conducted with the newly discovered crossover rate p_c = 0.4 and mutation rate p_m = 0.02. The result was then compared with the original combination of the genetic operators' (p_c = 0.6 and p_m = 0.02). Figures 8, 9 and 10 shows the comparison between the strategies evolved using a combination of crossover rate 0.4 and a mutation rate of 0.02 and the combination of crossover rate 0.6 with a mutation rate of 0.02. The new strategies evolved from the combination of the crossover rate of 0.4 and mutation rate of 0.02 produced better result in terms of the average fitness, the success rate and the average payoff. It can be observed that the mutation rate of 0.02 evolved better strategies when compared to other mutation rates as well (0.2 and 0.002). This rate is similar to the research outcome by Cervantes (Cervantes & Stephen, 2006) in which a mutation rate below the 1/N and error threshold is recommended. Besides, the results of the comparison showed that the combination of 0.4 crossover rate and 0.02 mutation rate can achieve better balance in the exploration and exploitation in evolving the bidding strategies as well. T-test is performed to show the significant improvement of this newly discovered combination of genetic operator probabilities. The symbol of ⊕ in Table 5 indicates that the P-value is less than 0.05 and has significant improvement.

	P Value
Average Fitness	⊕
Success Rate	⊕
Average Payoff	⊕

Table 5. P value of the t-test statistical analysis for comparison between newly discovered genetic operator probabilities with the old set of genetic operator probabilities

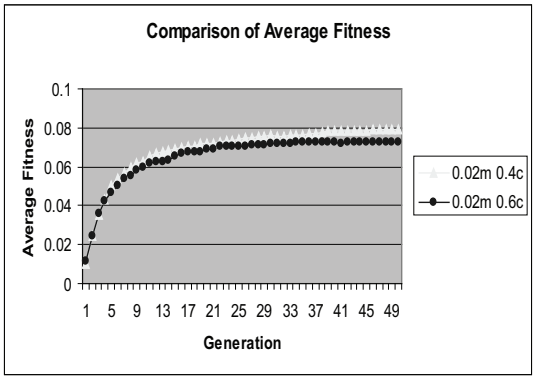


Fig. 8. Comparison of Average Fitness between the benchmark and the newly discovered rate.

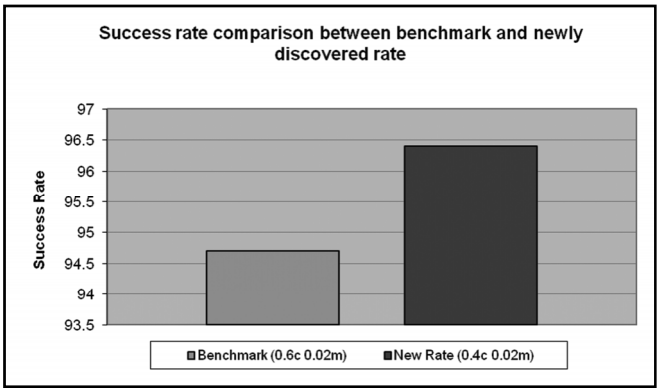


Fig. 9. Success rate for strategies evolved with the benchmark and the newly discovered rate

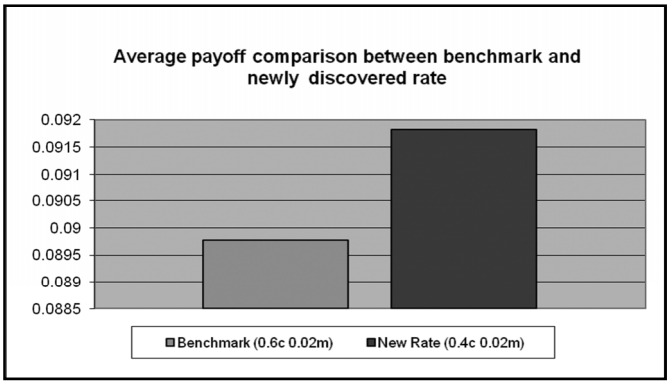


Fig. 10. Average payoff for strategies evolved with the benchmark and the newly discovered rate

This section investigated the performance of various combinations of predetermined sets of genetic operators' rates in genetic algorithm on a flexible and configurable heuristic decision making framework that is capable to tackle the problem of bidding across multiple auctions that applied different protocols (English, Vickrey and Dutch). As mentioned earlier, the optimal combinations of operators' probabilities of applying these operators are problem dependent. Thus, experiments have to be conducted in order to discover a new operator of combinations genetic operator probability which can improve the effectiveness of the bidding strategy. This experiment has proven that the crossover rate and mutation rate which were applied in the previous work are not the best value to be used in this framework. With this new combination of genetic operators, the experimental evaluation has also shown that the strategies evolved performed better than the other strategies evolved from the other combinations in terms of success rate and average payoff when bidding in the online auction marketplace. By discovering a better combination of genetic operator's probabilities, the improved performance of the bidding strategies as shown in Fig. 8, 9, and 10 are achieved. From this parameter tuning experiment, it can be confirmed that the parameters are problem dependent. However, trying out all of the different

combinations systematically is practically impossible as hand tuning the parameter is very time consuming. Therefore, in the second stage of the experiment, deterministic dynamic adaptation is applied to genetic algorithm to evolve the bidding strategies in order to overcome the manual tuning problem.

5. Deterministic dynamic adaptation

Many researchers have applied deterministic dynamic adaptation in evolutionary algorithms as a method to improve the limitation in the performance of evolutionary algorithms. This type of adaptation alters the value of strategy parameter by using some deterministic rule (Fogarty, 1989; Hinterding *et al.* 1997). The value of the strategy parameter is modified by the deterministic rule which is normally a time-varying schedule. It is different from the standard genetic algorithm since GA applies a fixed mutation rate over the evolutionary process. Most of the practical applications often favor larger or non-constant settings of the genetic operators' probabilities. (Back & Schutz, 1996). Some of the studies have proved the usefulness and effectiveness of larger, varying mutation rates (Back, 1992; Muhlenbein, 1992).

In this work, a time-variant dependent control rule is applied to change the control parameters over time without taking into account any present information by the evolutionary process itself (Eiben *et al.* 1999; Hinterding *et al.* 1997). Several studies have shown that a time dependent schedule is able to perform better than a fixed constant control parameter (Fogarty, 1989; Hesser & Manner, 1990; Hesser & Manner, 1992; Back & Schutz, 1996). The control rule is used to change the control parameter over the generation of the evolutionary process. The newly discovered crossover and mutation rates from the first experiment will be used in this particular schedule to serve as the midpoint in the time schedule. The parameter step size will change equally over the generation of the evolutionary process as well. This experiment is intended to discover the best deterministic dynamic adaptation by varying the genetic operators' probability scheme in exploring the bidding strategies.

The deterministic increasing and decreasing schemes for the crossover and mutation are different due to the changing scale of the values. The newly discovered crossover rates obtained from Section 3 is used as the midpoint for the time variant schedule because the convergence period of the evolution occur around the 25th generation. Consequently, the deterministic increasing scheme for the crossover rate will change progressively from p_c = 0.2 to p_c = 0.6 over the generation whereas the decrease scheme for the crossover rate is vice versa. The mutation rate obtained from the previous experiment is used as the midpoint of the time variant schedule for the increasing and decreasing schemes. The deterministic increasing scheme for the mutation rate, in contrast, will change progressively from p_m = 0.002 to p_m = 0.2 over the generation and vice versa for the deterministic decreasing schemes. The changing scale during each generation is decided by taking the difference between ranges of the rate divided by the total number of generation.

5.1 Experimental setup

Table 6 shows the parameter setting for the deterministic dynamic adaptation genetic algorithm. The evolutionary setting and parameter setting in the simulated environment is

the same as Tables 2 and 4. Fig. 11 shows the pseudocode of the deterministic dynamic adaptive genetic algorithm.

Representation	Floating Points Number
Number of Generations	50
Number of Individuals	50
Elitism	10%
Selection Operator	Tournament Selection
Crossover Operator	Extension Combination Operator
Crossover Probability	Change(Range from 0.4 to 0.6) / Fixed (0.4)
Mutation Operator	Creep Operator
Mutation Probability	Change (Range from 0.2 to 0.002) / Fixed (0.02)
Termination Criteria	After 50 Generation
Numbers of Repeat Run	30

Table 6. Deterministic dynamic adaptation parameter setting

```
Begin
    Randomly create initial bidder populations;
    While not (Stopping Criterion) do
      Calculate fitness of each individual by running the
      marketplace 2000 times;
      Create new population
        Select the fittest individuals (HP);
        Create mating pool for the remaining population;
        Perform crossover and mutation in the mating
        pool to create new generation(SF);
        New generation is HP + SF;
   Change the control parameter value (Crossover / Mutation)
   Gen = Gen + 1
   End while
End
```

Fig. 11. The Deterministic Dynamic Adaptation Genetic Algorithm

Crossover Rate	Mutation Rate	Abbreviation
Fixed	Increase	CFMI
Fixed	Decrease	CFMD
Increase	Fixed	CIMF
Decrease	Fixed	CDMF
Increase	Increase	CIMI
Decrease	Decrease	CDMD
Increase	Decrease	CIMD
Decrease	Increase	CDMI

Table 7. The Deterministic Dynamic Adaptation testing sets

5.2 Experimental evaluation

The performance of the evolved bidding strategies is evaluated based on three measurements discussed in Section 4.2. As before, the average fitness of the each population is calculated over 50 generations. The success rate of the agent's strategy and the average payoff is observed over 200 runs in the market simulation.

A series of experiments were conducted with the deterministic dynamic adaptation using the testing sets in Table 7. From the experiments, CFMD and CDMI performed better than the other combinations (Gan *et al*, 2008a, Gan *et al*, 2008b). Fig. 12 shows that the population evolved with deterministic dynamic adaptation is able to perform a lot better than the fixed constant crossover and mutation rates. This result is similar to the ones observed by other researches where non-constant control parameter performed better than fixed constant control parameter (Back 1992; Back 1993; Back & Schutz 1996; Fogarty 1989; Hesser & Manner, 1991; Hesser & Manner, 1992). Even though, the point of convergence for the different dynamic deterministic scheme is similar, the population with CDMI achieved a higher average fitness when compared to the populations with CFMD. The CDMI scheme with the increase mutation rate is able to maintain exploration velocity in the search space till the end of the run with the decreasing crossover rate achieving a balance between exploitation with the exploration in the search space and also to achieve a balance between exploration and exploitation in this setting.

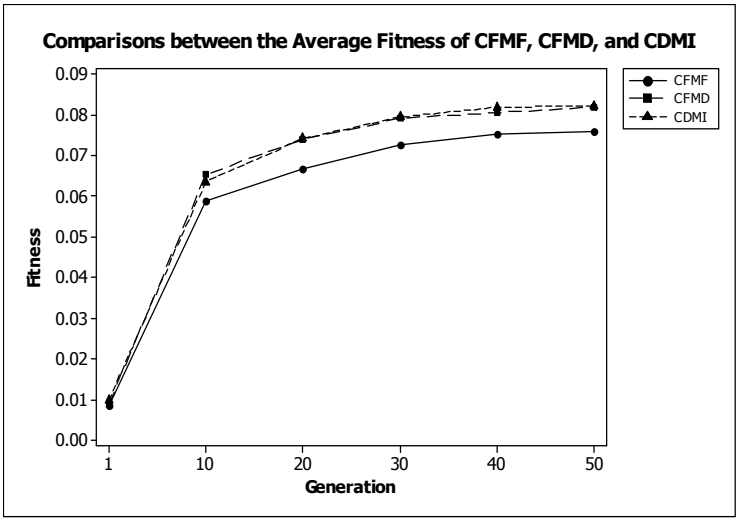


Fig. 12. Comparisons between the average fitness of CFMF, CFMD, and CDMI

Based on Fig. 13 and Fig. 14 CDMI outperformed CFMF and CFMD in both the success rate and the average payoff. This shows that the strategy evolved by using the CDMI does not only generate a better average fitness but also evolves better effective strategies compared to the strategy evolved for the other deterministic schemes and they are able to gain a higher profit when procuring the item at the end of the auction. It achieved a higher average fitness function during the evolution process as well.

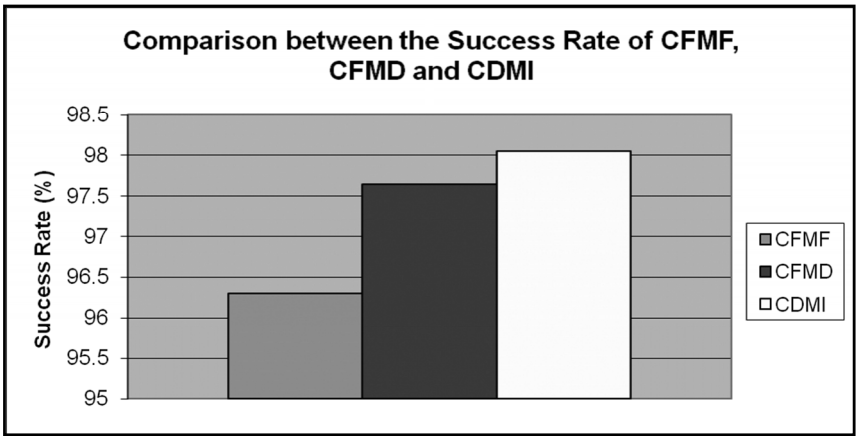


Fig. 13. Success rate comparison between CFMF, CFMD and CDMI

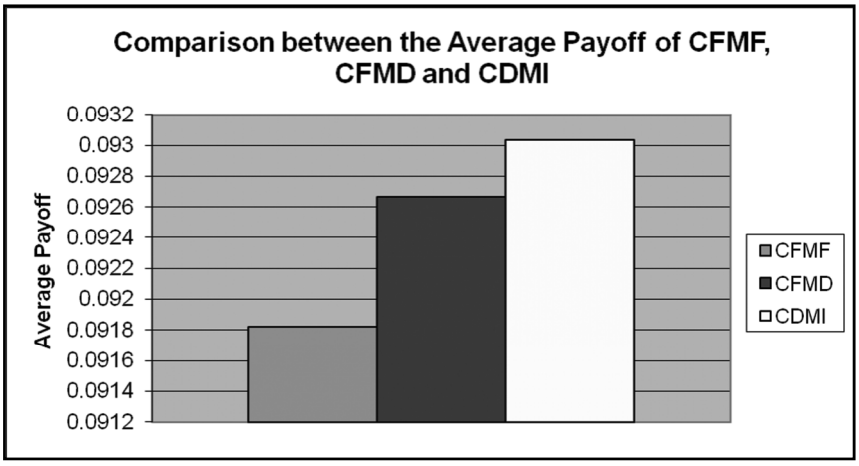


Fig. 14. Average payoff comparison between CFMF, CFMD and CDMI

This experiment has proven that non-constant genetic probabilities are more favorable than constant genetic probabilities. However, the deterministic dynamic adaptation may change the control parameter without taking into account the current evolutionary process as it does not take feedback from the current state evolutionary process whether the genetic operators' probabilities performed best at that current state of evolutionary process. The third stage of the experiment applies another adaptation method known as self-adaptation. The self-adaptation method is different from the deterministic dynamic adaptation where the self-adaptation evolves the parameter based on the current status of the evolutionary process. The self-adaptation method incorporates the control parameters into the chromosomes, thereby, subjecting them to evolution. In the last stage of the experiment, the self-adaptation is applied to genetic algorithm in order to evolve the bidding strategies.

6. Self-adaptation

The idea of self-adaptation is based upon the evolving of evolution. Self-adaptation has been used as one of the method to regulate the control parameter. As the name implies, the algorithm controls the adjustment of the parameters itself. This is done by encoding the parameter into the individual genomes by undergoing mutation and recombination. The control parameters can be any of the strategy parameters in evolutionary algorithm such as mutation rate, crossover rate, population size, selection operators and others (Back et al. 1997). However, the encoded parameters do not affect the fitness of the individuals directly, but rather, "better" values will lead to "better" individuals and these individuals will be more likely to survive and produce offspring and hence, proliferating these "better" parameter values. The goal of the self-adaptation is not only to find the suitable adjustment but also to execute it efficiently. The task is further complicated when the optimizer faced by a dynamic problem is taken into account since a parameter setting that was optimal at the beginning of an EA-run might become unsuitable during the evolution process. This scenario has been shown in some of the researches that different values of parameters might be optimal at different stages of the evolutionary process (Back, 1992a; Back, 1992b; Back, 1993; Davis, 1987; Hesser & Manner, 1991). Self-adaptation aims at biasing the distribution towards appropriate regions of search space and maintains sufficient diversity among individuals in order to enable further evolvability (Angeline, 1995; Meyer-Nieberg & Beyer, 2006).

The self-adaptation method has been commonly used in evolutionary programming (Fogel, 1962; Fogel, 1966) and evolutionary strategies (Rechenberg, 1973; Schwefel, 1977) but it is rarely used in genetic algorithms (Holland, 1975). This work applies self-adaptation in genetic algorithm which aims to adjust the crossover rate and mutation rate. The optimal rate for different phases of the evolution is obtained when different self-adaptation is capable in improving the algorithm by adjusting the crossover rate and mutation rate based on the current phase of the algorithm. Researchers have shown that the self-adaptation is able to improve the crossover in genetic algorithm (Schaffer & Morishima, 1987; Spears, 1995). In addition, studies also showed that the self-adaptive mutation rate does perform better than fixed constant mutation rate by incorporating the mutation rate into the individual genomes (Back, 1992a; Back, 1992b). In this section, three different self-adaptation schemes will be tested to discover the best self-adaptation scheme from this testing set. The self-adaptation requires the crossover and mutation rates to be encoded into the individual's genomes. Thus, some modification the encoding representation needs to be performed. The crossover and mutation rate become part of the genomes which will go through the crossover and mutation processes similar to the other alleles.

6.1 Experimental setup

Table 8 shows the parameter setting for the self-adaptive genetic algorithm. The evolutionary setting and parameter setting in the simulated environment is same as Table 2 and 4. Fig. 15 shows the pseudocode of the deterministic dynamic adaptive genetic algorithm. Fig. 16 shows the different encoding representation of the individual genome that will be used in the experiment. The crossover and mutation rate are encoded into the representation in order to go through the evolution process.

Representation	Floating Points Number
Number of Generations	50
Number of Individuals	50
Elitism	10%
Selection Operator	Tournament Selection
Crossover Operator	Extension Combination Operator
Crossover Probability	Self-Adapted / Fixed (0.4)
Mutation Operator	Creep Operator
Mutation Probability	Self-Adapted / Fixed (0.02)
Termination Criteria	After 50 Generation
Numbers of Repeat Run	30

Table 8. Self-adaptation genetic algorithm parameter setting

```
Generation = 0
Random initialize population
While generation not equal 50
    Evaluate population fitness
    Select the top 10% to next generation
    Tournament Selection Parents to Mating Pool
    Check Parents Crossover Rate
    Generating offspring through crossover process
    Check Individual Mutation Rate
    Mutate the offspring
    Select offspring to the next generation
    Generation = Generation + 1
```

Fig. 15. The self adaptation algorithm both genetic operators

k_{rt}	β_{rt}	k_{ra}	β_{ra}	k_{ba}	β_{ba}	k_{de}	β_{de}	w_{rt}	w_{ra}	w_{ba}	w_{de}	p_c	p_m

Fig. 16. Encoding of a bidding strategy for self-adaptation crossover and mutation rate

Crossover Rate	Mutation Rate	Abbreviation
Fixed	Self-Adapted	SAM
Self-Adapted	Fixed	SAF
Self-Adapted	Self-Adapted	SACM

Table 9. Self-adaptation testing sets

6.2 Experimental evaluation

The performance of the evolved bidding strategies is also evaluated based on the three measurements discussed in Section 4.2. As before, the average fitness of the each population is calculated over 50 generations. The success rate of the agent's strategy and the average payoff is observed over 200 runs in the market simulation.

A series of experiments were conducted with the self-adaptive testing sets described in Table 10. From the experiments, self-adapting both crossover and mutation rates performed better than the other combinations (Gan *et al*, 2009). The population with self adaptive crossover and mutation (SACM) achieved a higher average fitness compared to the population of self-adaptive crossover (SAC) and self –adaptive mutation schemes (SAM) as shown in Fig. 17. This scenario implies that the population with self adaptive crossover and mutation perform at its best among other populations and this is due to the self-adaptation crossover and mutation scheme which has combined the advantageous of the self-adaptive crossover and self-adaptive mutation scheme together. By having the two parameters to self-adapt, the control parameter can be adjusted to find the solution in different stages with the best control parameter which have been shown in the previous study indicating that different evolution stages will possess different optimal parameter values (Eiben *et al*. 1999).

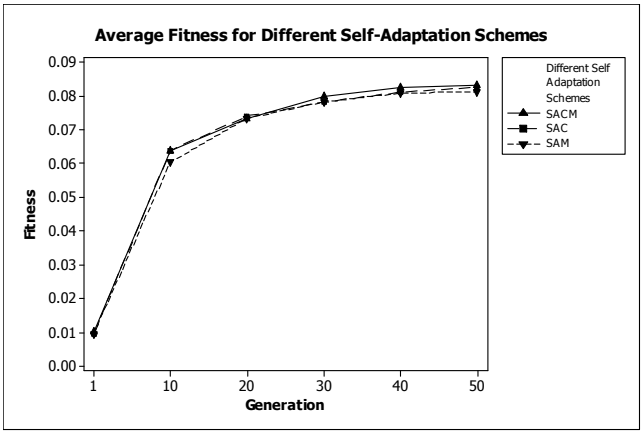


Fig. 17. Average fitness for different self-adaptation schemes

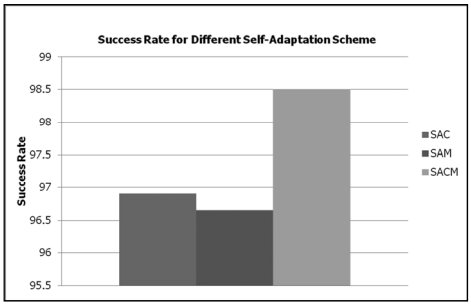


Fig. 18. Success rate for strategies evolved from different self-adaptation schemes

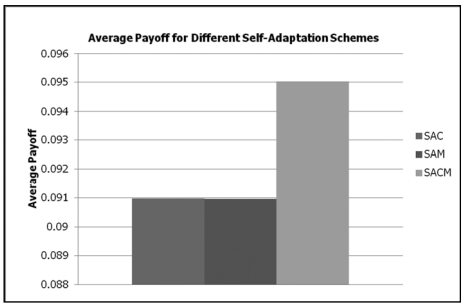


Fig. 19. Average payoff for strategies evolved from different self-adaptation schemes

All of the individuals generated a 4% increase in success rate and average payoff after employing the self adaptive crossover and mutation schemes as shown in Fig. 18 and Fig. 19 This has proven that the strategy evolved by using the self adaptive crossover and mutation does not only generate a better average fitness and success rate but also evolves better effective strategies compared to the strategy evolved for other self adaptive schemes.

7. Comparison between variations of genetic algorithm

In order to determine which of the three approaches perform the best in improving the effectiveness of the bidding strategies, the best result of each experiment is compared. The comparison is made by choosing the best performing schemes from the parameter tuning, deterministic dynamic adaptation and self-adaptation experiments. The main objective of this work is to improve the effectiveness of the existing bidding strategies by using different disciplines of the genetic algorithm.

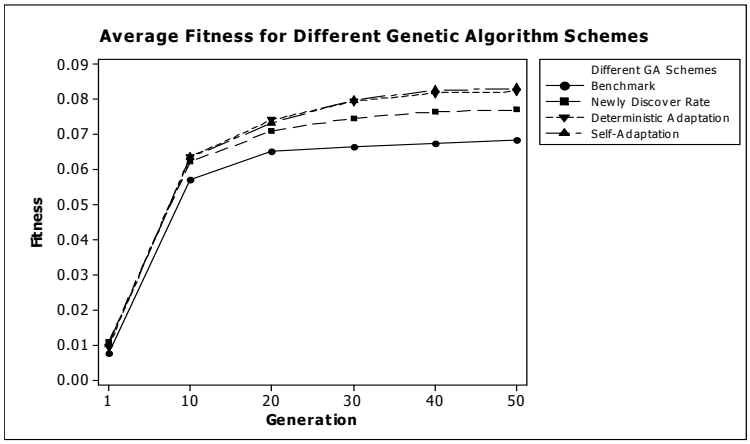


Fig. 20. Average fitness population with different genetic algorithm disciplines

Fig. 20 shows the average fitness for the evolving bidding strategy with different disciplines of the genetic algorithm. It can be seen clearly that there is an obvious differences between the convergence points in the different genetic algorithm disciplines. Self-adaptation

achieves a higher average fitness compared to benchmark, the newly discovered static rate and deterministic dynamic adaptation. Although average fitness of the self-adaptation and deterministic dynamic adaption is similar, self-adaptation achieves a higher average fitness when compared to deterministic dynamic adaptation.

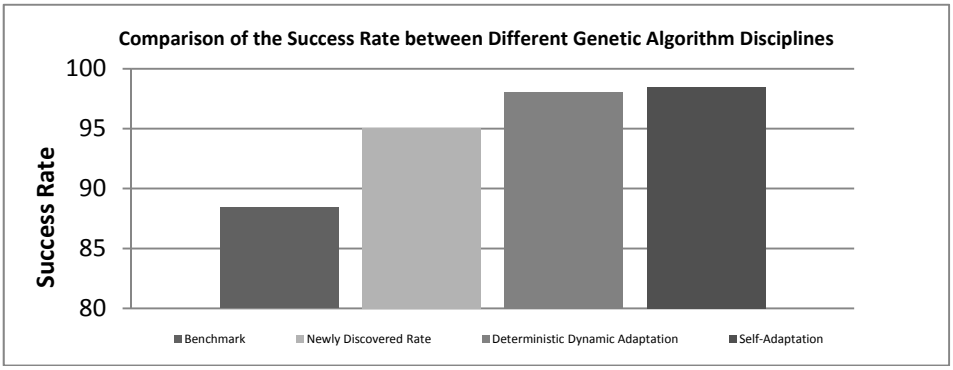


Fig. 21. Success rate for strategies evolved from different genetic algorithm disciplines

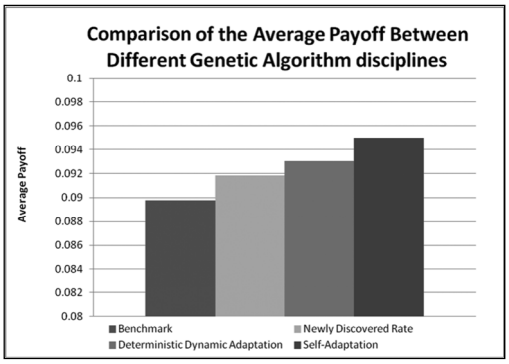


Fig. 22. Average payoff for strategies evolved from different genetic algorithm disciplines

The individuals evolved from the self adaptive genetic algorithm outperformed the other individuals from the other disciplines by delivering a more promising success rate. The strategy evolved is 1% higher than the strategies evolved from the deterministic dynamic adaptation. When compared to the benchmark value, an increase of 4% in the success rate is generated by the strategy which that employed the self-adaptation method. As a result, the strategy evolved from the self adaptive genetic algorithm can evolve better strategies and deliver higher success rate when bidding in online auctions which will eventually, improve the GA in searching for better bidding strategies.

All of the strategies evolved from the self adaptive genetic algorithm outperformed the rest with 2% higher average payoff when compared to the strategies which applied deterministic dynamic adaptation and 4% higher when compared to the strategies from the benchmark. This result obtained indicates that the strategy evolved by using the self adaptive genetic algorithm does not only produce a better average fitness and success rate but also evolves

better effective strategies compared to the other strategies evolved for other disciplines and they have gained higher profit when procuring the item.

SA	Benchmark	Newly Discovered Static Rate	DDA
Success Rate	⊕	⊕	⊕
Average Payoff	⊕	⊕	⊕

Table 10. P value for the comparison between different disciplines in term of success rate and average payoff

The symbol ⊕ in Table 10 indicates that the P-value is less than 0.05 and has significant improvement. The result of P value in the t-test in Table 10 shows the improvement generated by the self-adaptation is more significant compared to the other disciplines. Hence, it can be confirmed that self-adaptation is the best discipline in improving the effectiveness of the bidding strategies.

8. Conclusion

Based on the results of the experiments, the strategies evolved with self adaptive genetic algorithm achieved the most ideal result in terms of success rate and average payoff in an online auction environment setting. The strategies have also achieved a higher average fitness function during the evolution process.

The result in Figure 20, 21, 22 and Table 10 confirmed this conclusion by empirically proving that self adaptive genetic algorithm can evolve better bidding strategies compared to the other genetic algorithm disciplines. Among these different methods, the self-adaptation outperformed all of the other methods due to the nature of the method. In order to achieve better bidding strategies, the self-adaptation crossover and mutation scheme can be used to ensure better bidding strategies which in turn produces higher success rate, average fitness and average payoff.

Further investigation can be conducted by evolving the bidding strategies with two other evolution methods which are the evolution strategies and evolution programming. Evolving the bidding strategies with the evolution programming and evolution strategies may generate interesting result which different from genetic algorithm. A comparison between performances the evolutions strategies, evolution programming and genetic algorithm may produce interesting results.

9. References

Angeline, P. J. 1995. Adaptive and Self-Adaptive Evolutionary Computation. *In* Palaniswami, M., Attikiouzel, Y., Marks, R. J., Fogel, D., & Fukuda, T. (eds.). *A Dynamic System Perspective*, pp. 264-270. New York: IEEE Press.

Anthony, P. 2003. *Bidding Agents for Multiple Heterogeneous Online Auctions.* PhD's Thesis. University of Southampton.

Anthony, P. and Jennings, N. R. 2002. Evolving Bidding Strategies for Multiple Auctions. Amsterdam: *Proceedings of the 15th European Conference on Artificial Intelligence*, pp. 178-182. IOS Press.

Anthony, P. and N. R. Jennings. 2003a. Agents in Online Auctions. *In* Yaacob, S. Nagarajan, R., Chekima, A. and Sainarayanan, G. (Eds.). *Current Trends in Artificial Intelligence and Applications*, pp. 42-50. Kota Kinabalu: Universiti Malaysia Sabah.

Anthony, P. and N. R. Jennings. 2003b. Developing a Bidding Agent for Multiple Heterogeneous Auctions. *ACM Transactions on Internet Technology,* 3(3): 185-217.

Anthony, P. and N. R. Jennings. 2003c. A Heuristic Bidding Strategy for Multiple Heteregenous Auctions. *Proceedings of the Fifth International Conference on Electronic Commerce*, pp. 9-16. New York: ACM.

Babanov, A., Ketter, W. and Gini, M. L. 2003. An Evolutionary Approach for Studying Heterogeneous Strategies in Electronic Markets. *Engineering Self-Organising Systems 2003*. pp. 157-168.

Back T. and Schutz M. 1996. Intelligent Mutation Rate Control in Canonical Genetic Algorithms. *Proceedings of the International Symposium on Methodologies for Intelligent Systems*. *In* Ras, Z. W. and Michalewicz, Z. (Eds.) *Lecture Notes In Computer Science,* 1079: 158-167. London: Springer-Verlag.

Back, T. 1992a. The Interaction of Mutation Rate, Selection, and Self-Adaptation within a Genetic Algorithm. In Manner, R. and Manderick, B. (Eds). *Proceeding 2nd Conferences of Parallel Problem Solving from Nature,* pp. 85-94. Belgium: Elsevier.

Back, T. 1992b. Self-Adaptation in Genetic Algorithms. *In* Varela, F. J. and Bourgine, P. (Eds.) *Toward a Practice of Autonomous Systems: Proceeding 1st European Conference of Artificial Life,* pp. 263-271. Cambridge: MIT Press.

Back, T. 1993. Optimal Mutation Rates in Genetic Search. *Proceedings of the 5th International Conferences of Genetic Algorithms*, pp. 2-8. San Francisco: Morgan Kaufmann.

Back, T. Fogel, David. and Michalewicz, Z. Eds. 1997. *Handbook of Evolutionary Computation.* New York: Oxford University Press.

Bapna, R., P. Goes, and A. Gupta (2001). Insights and Analyses of Online Auctions. *Communications of the ACM,* 44 (11): 43-50.

Beasley, D., Bull, D. R. and Martin R. R. 1993. An Overview of Genetic Algorithms: Part 2, Research Topics. *University Computing* 15(4): 170 - 181.

Blickle, T. and Thiele, L. 1995. A Comparison of Selection Schemes Used in Genetic Algorithms. *Technical Report 11*. Zurich: Swiss Federal Institute of Technology.

Blickle, T. and Thiele, L. 2001. A Mathematical Analysis of Tournament Selection. *Proceedings of the Sixth International Conference on Genetic Algorithms*, pp. 9-16. San Francisco: Morgan Kaufmann.

Catania, V., Malgeri, M. and Russo, M. 1997. Applying Fuzzy Logic to Codesign Partitioning. *IEEE Micro* 17(3): 62-70.

Cervantes, J. and Stephens, C. R. 2006. "Optimal" mutation rates for genetic search. *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation Conference*, pp.1313 – 1320. New York: ACM Press.

Chandrasekharam, R., Subhramanian, S. and Chaudhury, S. 1993. Genetic Algorithm for Node Partitioning Problem and Application in VLSI Design. *IEE Proceedings Series E: Computers and Digital Techniques*, 140(5): 255–260.

Choi, J. H., Ahn, H., and Han, I. 2008. Utility-based double auction mechanism using genetic algorithms. Expert System Applicationl. 2008, pp. 150-158.

Cliff, D. 1997. Minimal Intelligence Agents for Bargaining Behaviours in Market Environment. *Technical Report HPL-97-91.* Hewlett Packard Laboratories.

Cliff, D. 1998a. Genetic optimization of adaptive trading agents for double-auction markets. *Proceedings Computing Intelligent Financial Engineering (CIFEr)*. pp. 252–258.

Cliff, D. 1998b. Evolutionary optimization of parameter sets for adaptive software-agent traders in continuous double-auction markets. *Artificial Society Computing Markets (ASCMA98) Workshop at the 2nd Int. Conference. Autonomous Agents*. (unpublished)

Cliff, D. 2002a. Evolution of market mechanism through a continuous space of auction types. *Proceeding Congress Evolutionary Computation*. pp. 2029–2034.

Cliff, D. 2002b. Visualizing search-spaces for evolved hybrid auction mechanisms. *Presented at the 8th Int. Conference. Simulation and Synthesis of Living Systems (ALifeVIII) Conference. Beyond Fitness: Visualizing Evolution Workshop, Sydney*.

Cliff, D. 2006. ZIP60: Further Explorations in the Evolutionary Design of Trader Agents and Online Auction-Market Mechanisms. *IEEE Transactions on Evolutionary Computation*.

Davis, L. 1987. *Genetic Algorithm and Simulated Annealling*. San Francisco: Morgan Kaufmann.

Davis, L. 1991a. *Handbook of Genetic Algorithms*. New York: Van Nostrand Reinhold.

Davis, L. 1991b. Hybridization and Numerical Representation, *In* Davis, L. (ed), *The handbook of Genetic Algorithm*, pp. 61-71. New York: Van Nostrand Reinhold.

eBay. 2008. "eBay Inc. Annual Report 2010," (19 October 2010). http://investor.ebayinc.com/annuals.cfm ..

Eiben, A. G., Hinterding, R., and Michalewicz, Z. 1999. Parameter Control in Evolutionary Algorithms. *IEEE Transactions on Evolutionary Computation, 3*(2). pp. 124 – 141.

Engelbrecht, A.P. 2002. *Computational Intelligence an Introduction*. New Jersey: John Wiley & Sons.

Epstein, J. M. and Axtell, R. 1996. *Growing Artificial Societies: Social Science from the Bottom Up*. Cambridge: MIT Press.

Fogarty, T. 1989. Varying the probability of mutation in genetic algorithm. *In* Schaffer, J. D. (Ed.) *Proceedings of the Third International Conference on Genetic Algorithms*, pp. 104-109. San Francisco: Morgan Kaufmann.

Fogel, D. B. 1992. *Evolving Artificial Intelligence*. PhD Thesis. Berkeley: University of California.

Fogel, L. J. 1962. Autonomous Automata. *Industrial Research,* 4: 14-19.

Gan K.S., Anthony P. and Teo J. 2008a. The Effect of Varying the Crossover Rate in the Evolution of Bidding Strategies. *4th International IASTED Conference on Advances in Computer Science and Technology (ACST-2008),* Langkawi, Malaysia, April 2008.

Gan K.S., Anthony P., Teo J. and Chin K.O. 2008b, Mutation Rate in The Evolution of Bidding Strategies, *The 3rd International Symposium on Information Technology 2008 (ITSim2008)*, Kuala Lumpur, Malaysia, August 2008

Gan K.S., Anthony P., Teo J. and Chin K.O. 2008c, Dynamic strategic parameter control in evolving bidding strategies. *Curtin University of Technology Science and Engineering (CUTSE) International Conference 2008*, Sarawak, Malaysia, November 2008.

Gan K.S., Anthony P., Teo J. and Chin K.O. 2008d, Evolving Bidding Strategies Using Deterministic dynamic adaptation. *The 4th International Conferences on Information Technology and Multimedia (ICIMU2008)*, Bangi, Malaysia, November 2008.

Gan K.S., Anthony P., Teo J. and Chin K.O. 2009, Evolving Bidding Strategies Using Self-Adaptation Genetic Algorithm, International Symposium on Intelligent Ubiquitous Computing and Education, Chengdu, China.

Goldberg, D. E. 1989. *Genetic Algorithms in Search, Optimization and Machine Learning*. New York: Addison-Wesley.

Hesser, J. and Manner, R. 1990. Towards an optimal mutation probability for genetic algorithms. *In* Schewefel, H. P. and Manner, R. (Eds.) *Proceedings for eh 1st Conferences on Parallel Problem Solving from Nature*, *Lecture Notes in Computer Science*, 496, pp 23-32. London: Springer-Verlag.

Hesser, J. and Manner, R. 1992. Investigation of the m-heuristic for optimal mutation probabilities, *Proceedinng of the 2nd Parallel Problem Solving from Nature*. pp. 115-124. Belgium: Elsevier.

Hinterding, R., Michalewicz, Z., and Eiben, A. E. 1997. Adaptation in Evolutionary Computation: A survey. *Proceeding 4th IEEE Conference of Evolutionary Computation*. pp. 65-69.

Hinterding, R., Michalewicz, Z., and Eiben, A. E. 1997. Adaptation in Evolutionary Computation: A survey. *Proceeding 4th IEEE Conference of Evolutionary Computation*. pp. 65-69.

Holland, J. H. 1975. *Adaption in Natural and Artificial System*. Michigan: MIT Press.

Internet Auction List. 2008. Listing Search in USAWeb.com, http://internetauctionlist.com/Search.asp. 19 October 2011.

Janikow, C. Z. and Michalewiz, Z. 1991. An experimental comparison of Binary and Floating Point Representations in Genetic Algorithms, *In* Belew, R. K. and Booker, L. B. (eds), *Proceedings of the 4th International Conferences in Genetic Algorithms*. pp 31-36. San Francisco: Morgan Kaufmann.

Jansen, E. 2003. Netlingo the Internet Dictionary. http://www.netlingo.com/. 10 November 2008.

Karr, C. 1991. Genetic Algorithms for Fuzzy Controllers. *AI Expert*. 6(2): 26-33.

Lee, M. A. and Takagi, H. 1993. Integrating Design Stages of Fuzzy Systems Using Genetic Algorithms. *Proceedings of the IEEE International Conference on Fuzzy Systems*. pp. 612–617.

Meyer-Nieberg, S. and Beyer, H-G. 2006. Self-Adaptation in Evolutionary Algorithms. *In* Lobo, F., Lima, C., and Michalewicz, Z. (Eds.) *Parameter Setting in Evolutionary Algorithm*. London: Springer-Verlag.

Michalewicz, Z. 1992. *Genetic Algorithms + Data Structure = Evolution Programs*. London: Springer-Verlag.

Muhlenbein, H. 1992. How Genetic Algorithm Really Work: I. Mutation and HillClimbing. *In* Manner, R. & Manderick, B. (Eds) *Parellel Problem Solving from Nature 2*. pp. 15-25. Belgium: Elsevier.

Nelson. R. R. 1995. Recent evolutionary theorizing about economic change. *Journal of Economic Literature*. 33(1): 48-90.

Obitko, M. 1998. Introduction to Genetic Algorithms. http://cs.felk.cvut.cz/ xobitko/ga/. 12 November 2008.

Rechenberg, I. 1973. *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution (Evolution Strategy: Optimization of Technical Systems by Means of Biological Evolution)*. Stuttgart: Fromman-Holzboog.

Roth, A. E. 2002. The economist as engineer: Game theory, experimentation, and computation as tools for design economics. *Econometrica*, 70(4): 1341-1378.

Schaffer, J. D. and Morishima, A. 1987. An Adaptive crossover distribution mechanism for Genetic Algorithms. *In* Grefensttete, J. J. (Ed) *Genetic Algorithms and their Applications: Proceedings of the Second International Conference on Genetic Algorithms.* pp. 36-40.

Schwefel, H. P. 1977. Numerishce Optimierung von Computer-Modellen mittels der Evolutionsstrategic. *Interdisciplinary System Research.* 26.

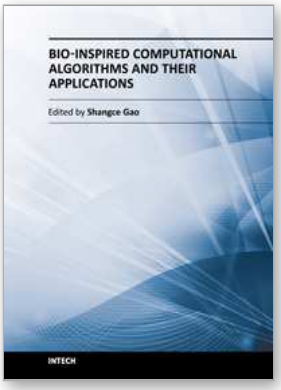
Smith, V. 1962. Experimental study of competitive market behavior. *Journal Political Economy*, 70: 111–137.

Spears, W. M. 1995. Adapting Crossover in Evolutionary Algorithm. *Proceedings of the Fourth Annual Conference on Evolutionary Programming.* pp. 367-384. Cambridge: MIT Press.

Tesfatsion, L. 2002. Agent-based computational economics: Growing economies from the bottom up. *Artificial Life*, 8(1): 55-82.

Uckun, S., Bagchi, S. and Kawamura, K. 1993. Managing Genetic Search in Job Shop Scheduling. *IEEE Expert: Intelligent Systems and Their Applications*, 8(5): 15-24.

Wolfstetter, E. 2002. Auctions: An Introduction. *Journal of Economic Surveys,* 10: 367-420.



Bio-Inspired Computational Algorithms and Their Applications

Edited by Dr. Shangce Gao

ISBN 978-953-51-0214-4
Hard cover, 420 pages
Publisher InTech
Published online 07, March, 2012
Published in print edition March, 2012

Bio-inspired computational algorithms are always hot research topics in artificial intelligence communities. Biology is a bewildering source of inspiration for the design of intelligent artifacts that are capable of efficient and autonomous operation in unknown and changing environments. It is difficult to resist the fascination of creating artifacts that display elements of lifelike intelligence, thus needing techniques for control, optimization, prediction, security, design, and so on. Bio-Inspired Computational Algorithms and Their Applications is a compendium that addresses this need. It integrates contrasting techniques of genetic algorithms, artificial immune systems, particle swarm optimization, and hybrid models to solve many real-world problems. The works presented in this book give insights into the creation of innovative improvements over algorithm performance, potential applications on various practical tasks, and combination of different techniques. The book provides a reference to researchers, practitioners, and students in both artificial intelligence and engineering communities, forming a foundation for the development of the field.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Kim Soon Gan, Patricia Anthony, Jason Teo and Kim On Chin (2012). Performance of Varying Genetic Algorithm Techniques in Online Auction, Bio-Inspired Computational Algorithms and Their Applications, Dr. Shangce Gao (Ed.), ISBN: 978-953-51-0214-4, InTech, Available from: http://www.intechopen.com/books/bio-inspired-computational-algorithms-and-their-applications/performance-of-varying-genetic-algorithm-techniques-in-online-auction

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821