Adapting Crossover and Mutation Rates in Genetic Algorithms.

Article *in* Journal of Information Science and Engineering · September 2003

Source: DBLP

CITATIONS
92

READS
745

3 authors, including:

Wen-Yang Lin
National University of Kaohsiung
115 PUBLICATIONS **877** CITATIONS
SEE PROFILE

Tzung-Pei Hong
National University of Kaohsiung
679 PUBLICATIONS **7,437** CITATIONS
SEE PROFILE

Adapting Crossover and Mutation Rates* in Genetic Algorithms

WEN-YANG LIN, WEN-YUNG LEE AND TZUNG-PEI HONG**
Department of Information Management
I-Shou University
Kaohsiung, 840 Taiwan
E-mail: wylin@isu.edu.tw
+*R&D Department*
TransAsia Telecommunications Inc.
Kaohsiung, 806 Taiwan
E-mail: anttyl@tat.com.tw
**Department of Electrical Engineering*
National University of Kaohsiung
Kaohsiung, 811 Taiwan
E-mail: tphong@nuk.edu.tw

It is well known that a judicious choice of crossover and/or mutation rates is critical to the success of genetic algorithms. Most earlier researches focused on finding optimal crossover or mutation rates, which vary for different problems, and even for different stages of the genetic process in a problem. In this paper, a generic scheme for adapting the crossover and mutation probabilities is proposed. The crossover and mutation rates are adapted in response to the evaluation results of the respective offspring in the next generation. Experimental results show that the proposed scheme significantly improves the performance of genetic algorithms and outperforms previous work.

Keywords: genetic algorithms, self-adaptation, progressive value, crossover rate, mutation rate

1. INTRODUCTION

Genetic Algorithms (GAs) are robust search and optimization techniques that were developed based on ideas and techniques from genetic and evolutionary theory [9, 19]. Beginning with a random population of chromosomes, a genetic algorithm chooses parents from which to generate offspring using operations analogous to biological processes, usually *crossover* and *mutation*. All chromosomes are evaluated using a fitness function to determine their "fitness." These fitness values are then used to decide whether the chromosomes are eliminated or retained. According to the *principle of survival of the*

Received January 11, 2002; revised August 26, 2002; accepted January 24, 2003.
Communicated by Hsu-Chun Yen.

*A preliminary version of this paper was presented at the *Sixth Conference on Artificial Intelligence and Applications*, Kaohsiung, Taiwan, in November 9, 2001, sponsored by the Taiwanese Association for Artificial Intelligence.

fittest, the more adaptive chromosomes are kept, and the less adaptive ones are discarded in the course of generating a new population. The new population replaces the old one, and the whole process is repeated until specific termination conditions are satisfied. Fig. 1 depicts a typical genetic process.

There is evidence showing that the probabilities of crossover and mutation are critical to the success of genetic algorithms [3-6, 12, 20]. Traditionally, determining what probabilities of crossover and mutation should be used is usually done by means of trial-and-error. The optimal crossover or mutation rates, however, vary along with the problem of concern, even within different stages of the genetic process in a problem. In the past few years, though some researchers have investigated schemes for automating the parameter settings for GAs, the problem of making GAs self-adaptive in choosing the optimal parameters remains an unexplored area.

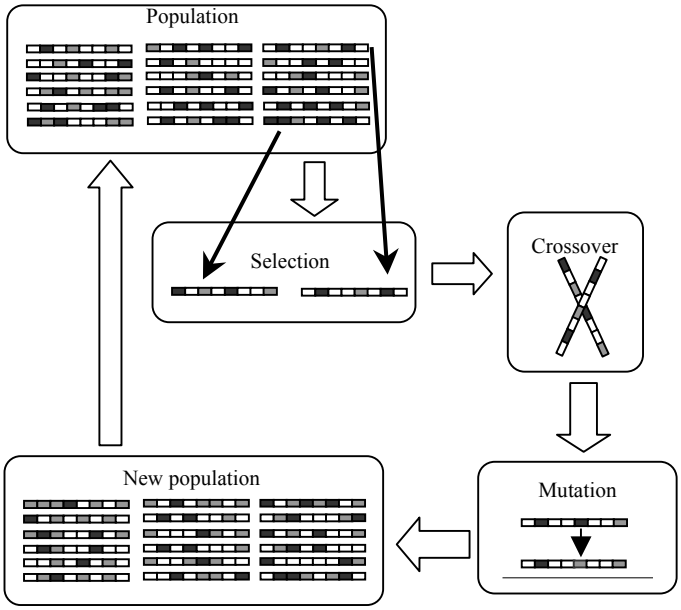

Fig. 1. A typical genetic process.

In this paper, a generic scheme for adapting the crossover and mutation probabilities is proposed. The crossover and mutation rates are adapted in response to the evaluation results of the respective offspring in the next generation. Experimental results show that the proposed scheme significantly improves the performance of genetic algorithms and outperforms previous proposed methods.

The content of this paper is organized as follows. A brief review of operator rate adaptation in genetic algorithms and related works is given in section 2. Our scheme for adapting the crossover and mutation rates is proposed in section 3. There we also discuss its convergence property. Section 4 describes our experiments. Conclusion and future work are given in section 5.

2. OPERATOR RATE CONTROL IN GENETIC ALGORITHMS AND RELATED WORK

When genetic algorithms are applied to solve a problem, the first step is to define a representation that describes the problem states. The most commonly used representation is the bit string. An initial population is then defined, and three genetic operations (crossover, mutation, and selection) are performed to generate the next generation. This procedure is repeated until the termination criterion is satisfied. This so-called Simple Genetic Algorithm (SGA) [9] is described in Algorithm 1.

Algorithm 1. A simple Genetic Algorithm.

```
Initialize the parameters;
Generate a population P randomly;
generation ← 1;
while generation ≤ max_gen do
    Clear the new population P';
    Use a fitness function f(·) to evaluate each individual in P;
    while |P'| ≤ N do
        Select two parents from P;
        Perform crossover with rate p_c;
        Perform mutation with rate p_m;
        Insert the offspring to P';
    endwhile
    P ← P';
    generation ← generation + 1;
endwhile
```

The power of genetic algorithms arises primarily from crossover and mutation. The crossover operation is used to generate offspring by exchanging bits in a pair of individuals (parents) chosen from the population, with the possibility that good solutions can generate better ones. Crossover occurs only with a probability p_c (the *crossover rate* or *crossover probability*). When individuals are not subjected to crossover, they remain unmodified. The mutation operator is used to change some elements in selected individuals with a probability p_m (the *mutation rate* or *mutation probability*), leading to additional genetic diversity to help the search process escape from local optimal traps.

The choice of p_c and p_m is known to critically affect the behavior and performance of GAs. The crossover rate controls the capability of GAs in exploiting a located hill to reach the local optima. The higher the crossover rate, the quicker exploitation proceeds. A p_c that is too large would disrupt individuals faster than they could be exploited. The mutation rate controls the speed of GAs in exploring a new area. Small p_m values are commonly adopted in GAs. A number of guidelines exist in the literature for setting the values for p_c and p_m [3, 6, 9, 23]. Typical values of p_c are in the range 0.5~1.0, while typical values of p_m are in the range 0.001~0.05. These general guidelines were drawn from empirical studies on a fixed set of test problems, and were inadequate because the optimal use of p_c and p_m is specific to the problem under consideration. Some studies focused particularly on finding optimal crossover or mutation rates [3, 11, 12, 22]. These

heralded the need for self-adaption in the crossover or mutation rates [7].

Fogarty [8], to our knowledge, was the first to use a varying mutation rate. He concluded that a mutation rate that decreased exponentially over generations had superior performance. His idea is quite similar to simulated annealing, where the mutation rate takes on a role analogous to that of *temperature*. Indeed, a mutation-only GA that evolves at a constant mutation rate is analygous to simulated annealing at a constant temperature [22] (which corresponds to the Metropolis process). Bäck [2] followed Fogarty's investigation and devised a deterministic formula for adjusting the mutation rate to dynamically approach the optimal value. An analogous way of cyclically varying the mutation rate reported in [13] exhibited a similar effect.

In [24], Srinivas and Patnaik proposed a mechanism for adapting operator probabilities in a generational GA. Each chromosome has its own crossover probability p_c and mutation probability p_m necessary for it to undergo crossover and mutation, respectively. During the execution of GAs, both p_c and p_m are adapted in proportion to the population maximum and mean fitness, with probabilities being larger when the mean fitness is near the maximum, and smaller for particular chromosomes with larger fitnesses. Similar concepts were adopted and further explored in [16, 25].

3. SCHEME FOR ADAPTING CROSSOVER AND MUTATION RATES

In the classic genetic algorithm, the involved genetic operators, such as crossover and mutation, work at an a priori, constant probability. Different crossover and mutation rates can, however, traverse different search directions in the state space, thus affecting the performance of the applied genetic algorithm. In fact, the overall performance of a genetic algorithm depends on it maintaining an acceptable level of productivity throughout the process of evolution. It is, thus, essential to design a genetic algorithm that adapts itself to the appropriate crossover and mutation rates. Our intuition is to dynamically adjust the operator's applied probability according to its "contribution" (or productivity); thus, there is the potential to produce children of improved fitness. In this section, we propose a rate-adapting scheme to achieve this goal.

The rationale behind this approach is as follows: It may be advantageous to employ a method that dynamically adjusts the genetic algorithm's settings according to a measure of the performance of each operator. To measure the performance of an operator, we develop a scheme involving the ability of an operator to produce new, preferably fitter, children. The purpose of the dynamic operator adapting methods is to exploit information gained regarding the current ability of each operator to produce children of improved fitness.

3. 1 Description

The first step in our approach is determining an initial p_c and p_m pair. For classic genetic algorithms, the general rule is to use a high crossover rate and a low mutation probability. However, inspired by previous works [3, 8], a large initial mutation rate is employed to help explore more local hills and locate a prospective area quickly. During this time, crossover should occur with a small probability to retain diversity. For these

reasons, we set the initial crossover rate and mutation rate at 0.5 and 0.5, respectively, and adjust them using the *progress-value* concept [15] described in the following.

Consider the offspring generated from two parents after a crossover operation is performed. Let f_sum_S be the fitness sum of the two offspring, and let f_sum_P denote the fitness sum of the parent individuals. We define the progress value of crossover CP as the gain obtained by

$$CP = f\_sum_S - f\_sum_P. \tag{1}$$

For a generation that undergoes n_c crossover operations, the average crossover progress value $\tilde{CP}$ is

$$\tilde{CP} = \frac{1}{n_c}\sum CP \tag{2}$$

Thus, $\tilde{CP}$ measures the overall performance of the crossover operator within a generation run.

Similarly, consider the resulting offspring after a mutation operation is performed; the progress value of the mutation MP is

$$MP = f_{new} - f_{old}, \tag{3}$$

where f_{new} is the fitness of the new offspring and f_{old} the fitness of the original individual. For a generation that undergoes n_m mutation operations, the average mutation progress value $\tilde{MP}$ is

$$\tilde{MP} = \frac{1}{n_m}\sum MP \tag{4}$$

Before the end of each generation, the crossover and mutation rates are adjusted using these average progress values. The operator that performs better in the previous run (with a larger average progress value) should participate more frequently (increase its probability) in the next generation run, and vice versa. The adjustment is executed as shown below:

$$p_c = p_c + \theta_1 \quad \text{if } \tilde{CP} > \tilde{MP},$$
$$p_c = p_c - \theta_1 \quad \text{if } \tilde{CP} < \tilde{MP},$$

and

$$p_m = p_m + \theta_2 \quad \text{if } \tilde{CP} < \tilde{MP},$$
$$p_m = p_m - \theta_2 \quad \text{if } \tilde{CP} > \tilde{MP},$$

where θ_1 and θ_2 represent the amount of adjustment of p_c and p_m, respectively. Now we

are confronted with another problem: What values of θ_1 and θ_2 should be chosen? Should θ_1 and θ_2 be identical or different? And, should they be constant values or adapting functions? We will examine these aspects later through experiments.

Notice that after each adjustment, we should make sure that the crossover and mutation operations have the chance to work continuously. For this reason, we set the minimum crossover and mutation rates to 0.001. If the crossover or mutation rate is less than or equal to 0.001, the adjustment operation stops decreasing the probability.

The refined genetic algorithm incorporating this adaptation approach (called the *progress rate genetic algorithm*, or PRGA) is described in Algorithm 2.

Algorithm 2. Progress Rate Genetic Algorithm.

Initialize the parameters;
Generate a population P randomly;
$generation \leftarrow 1$;
while $generation \leq max\_gen$ **do**
  Clear the new population P';
  Use a fitness function $f(\cdot)$ to evaluate each individual in P;
  while $|P'| \leq N$ **do**
    Select two parents from P;
    Perform *crossover* and accumulate the crossover progress value CP;
    Perform *mutation* and accumulate the mutation progress value MP;
    Insert the offspring to P';
  endwhile
  Compute $\tilde{CP}$ and $\tilde{MP}$, and adjust the crossover rate p_c and mutation rate p_m;
  $P \leftarrow P'$;
  $generation \leftarrow generation + 1$;
endwhile

Let us illustrate this approach using a single generation run with the following function:

$f(t) = t^A |\sin(5\pi t)|$, $t \in [0.000, 1.024]$, find the max.

Let the value of t be represented by a bit string with length = 10. For example, the genetic representation for $t = 0.738$ is 1011100010. Let $N = 8$. An example of the initial population is shown in Table 1.

Assume that the crossover and mutation rates are initially set to $p_c = 0.5$ and $p_m = 0.5$. The chosen parents for crossover and the resulting offspring are shown in Table 2. Here, we assume that one-point crossover is employed, and that for $p_c = 0.5$, only the pairs (4, 8) and (1, 8) are subjected to crossover. The crossover point is underlined.

The average crossover progress value is calculated as follows. Say parents 4 and 8 are chosen to produce two offspring, N3 and N4. According to Equation 1, the progress value of the crossover operation is $(0.0004 + 0.0021) - (0.0149 + 0.0004) = -0.0128$, and the progress value for offspring N7 and N8 is $(0.0010 + 0.1562) - (0.2453 + 0.0004) = -0.0885$. The average crossover progress value is -0.0506.

Table 1. An example initial population.

No	Old string	t	$f(t)$
1	1011100010	0.738	0.2453
2	1100001100	0.78	0.1144
3	1100101010	0.81	0.0673
4	0110101101	0.429	0.0149
5	0110100111	0.423	0.0113
6	0100100111	0.295	0.0076
7	0011111001	0.249	0.0027
8	0010010101	0.149	0.0004

Table 2. Eight offspring generated by the crossover operator.

No	Parents	Offspring	t	$f(t)$
N1	(7, 8)	0011111001	0.249	0.0027
N2	(7, 8)	0010010101	0.149	0.0004
N3	(4, 8)	0110010101	0.405	0.0021
N4	(4, 8)	0010101101	0.173	0.0004
N5	(4, 6)	0110101101	0.429	0.0149
N6	(4, 6)	0100100111	0.295	0.0076
N7	(1, 8)	1010010101	0.661	0.1562
N8	(1, 8)	0011100010	0.226	0.0010

Next, the offspring are mutated with probability p_m. Here, we assume that bit-flip mutation is employed, and that individuals N2, N5, N6 and N8 in Table 2 are chosen to undergo mutation. The results are shown in Table 3, where the values in bold denote the positions at which mutation occurs. According to Equation 2, the mutation progress value for N2 is 0.1562 ─ 0.0004 = 0.1558, and the mutation progress values are –0.0137, 0.0024, 0.0005 for N5, N6 and N8, respectively. The average mutation progress value is 0.0362.

Table 3. Offspring that undergo mutation.

No	Offspring	t	$f(t)$
N2’	**1**010010101	0.661	0.1562
N5’	0110**0**01101	0.397	0.0012
N6’	010**1**100111	0.359	0.0100
N8’	0011101**1**010	0.234	0.0015

Since $\tilde{MP} = 0.0362 > \tilde{CP} = -0.0506$, the mutation rate obtains an increment, while the crossover rate decreases.

3.2 Convergence Property of PRGA

Since genetic algorithms are stochastic search processes, there is no convincing theory that can completely deduce the convergence rate for GAs to reach the optimal solution. Nevertheless, we can examine a sort of convergence in probability.

Let N be the population size, and let n be the length of each individual. Given a probability α, we may ask what the smallest number of generations $g(\alpha)$ required to obtain an optimal solution is. Aytug and Koehler [1] modeled genetic algorithms with the Markov chain and derived the following bound:

$$\begin{aligned} \mathrm{INT}\left[\max_j\left\{\frac{\ln(1-\alpha)}{\ln(\delta(Q-Qu_j u_J^T))}\right\}\right] &\le g(\alpha) \\ &\le \mathrm{INT}\left[\frac{\ln(1-\alpha)}{\ln(1-\min\{(1-P_m)^{nN}, {p_m}^{nN}\})}\right] \end{aligned}, \tag{5}$$

where INT[x] denotes the smallest integer greater than or equal to x, for $x \ge 0$; Q is the Markov chain transition matrix; u_j is the jth unit vector; $\delta(A)$ is the spectral radius of a matrix A; and p_m is the mutation probability. The lower bound is difficult to evaluate because the maximum is taken over all states of the Markov chain. Greenhalgh and Marshall [10] showed that the upper bound can be improved further as

$$g(\alpha) \le \mathrm{INT}\left[\frac{\ln(1-\alpha)}{N\ \ln(1-\min\{P_m(1-P_m)^{n-1}, {P_m}^n\})}\right], \tag{6}$$

and that the bound holds independent of the crossover and selection schemes.

The above result can be further refined in the case where the mutation rate is adapted. Note that the mutation rate p_m under the proposed adaptation scheme is always bounded between 0 and 1. Without loss of generality, let $p_1 \le p_m \le p_2$ for $p_1 < 1$ and $p_2 > 0$. Then, it is easy to show that, following the derivation in [10],

$$g(\alpha) \le \mathrm{INT}\left[\frac{\ln(1-\alpha)}{N\ \ln(1-\min\{P_1(1-P_1)^{n-1}, {P_2}^n\})}\right]. \tag{7}$$

Note that the above derivation is based on the simple GAs, without taking into account the effect of repairing chromosomes. Because the degree of population diversity is reduced after repairing is performed, we can expect that the population will converge more quickly. The question of what the exact bound on the number of generations will be when the repairing effect is considered deserves further investigation.

4. EXPERIMENTS

To examine the performance of the progress rate genetic algorithm (PRGA), we compared it with four variants of GAs. They included the simple genetic algorithm (SGA) with fixed crossover and mutation rates; the decreasing mutation rate genetic algorithm (DMRGA) proposed by Bäck [2]; the cyclic-parental, low-offspring mutation (CPLO) in [13]; and the adaptive genetic algorithm (AGA) proposed by Srinivas and Patnaik [24]. The test problem was the 0/1 knapsack problem, which belongs to the class of knapsack-type problems and is well known to be NP-hard [17].

The 0/1 knapsack problem is as follows: given a set of objects, a_i, for $1 \le i \le n$, together with their profits P_i, weights W_i, and a capacity C, find a binary vector $\boldsymbol{x} = \langle x_1, x_2, \ldots, x_n \rangle$, such that

$$\sum_{i=1}^{n} x_i \cdot W_i \le C \quad \text{and} \quad \sum_{i=1}^{n} x_i \cdot P_i \quad \text{is maximal.}$$

Because the difficulty of knapsack problems is greatly affected by the correlation between profits and weights [18], we employed the three randomly generated sets of data used in [18]:

(1) uncorrelated
W_i and P_i: random(1..v);

(2) weakly correlated
W_i: random(1..v);
P_i: W_i + random(–r..r);

(3) strongly correlated
W_i: random(1..v);
P_i: $W_i + r$.

The data were generated with the following parameter settings: $v = 10$, $r = 5$, and $n = 250$. Following a suggestion made in [17], we adopted two different types of capacity C: 1) $C = 2v$, for which the optimal solution contained very few items; and 2) $C = 0.5\sum W_i$, in which about half of the items were in the optimal solution.

The test suit represented two different types of problem instances. Those data sets in Group 1 under the constraint $C = 2v$ were more difficult than those in Group 2 under the constraint $C = 0.5\sum W$, because the fitness landscape in Group 1 abounded with local hills. This increased the difficulty for GAs to locate the global optimum. The instance difficulty in each group was further classified by means of the correlation between profits and weights.

To be consistent with the crossover and mutation operators considered, we used the binary encoding scheme: each bit represented the inclusion or exclusion of an object. It was, however, possible to generate infeasible solutions with this representation. That is, the total weights of the selected objects would exceed the knapsack capacity. In the literature, two different ways of handling this constraint violation [18] have been proposed. One way is to use a penalty function to penalize the fitness of the infeasible candidate to

diminish its chance of survival. Another approach is to use a repair mechanism to correct the representation of the infeasible candidate. In [18], the repair method was more effective than the penalty approach. Hence, we adopted the repair approach in our implementation.

The repair scheme that we used was a greedy approach. All the objects in a knapsack represented by an overfilled bit string were sorted in decreasing order of their profit weight ratios. The last object was then selected for elimination (the corresponding bit of "1" was changed to "0"). This procedure was executed until the total weight of the remaining objects was less than the total capacity.

The common parameters set in this experiment were those listed below:

crossover: one-point crossover;
mutation: bit-flip mutation;
selection: random selection;
replacement: (μ+λ) replacement;
population size: 100;
generations: 500;
experimental runs: 10.

In a preliminary experiment, we noticed that the original AGA algorithm [24] that adopted proportional selection with generational replacement was inferior to all of the other methods that used random selection and (μ+λ) replacement. For this reason, the AGA algorithm was changed to use random selection with (μ+λ) replacement as well.

The crossover and mutation rates employed for each variant are shown below:

SGA: $p_c = 0.65$, $p_m = 0.01$;
DMRGA: $p_c = 0.65$,

$$p_m(t) = \left(2 + \frac{L-2}{T} \cdot t\right)^{-1},$$

where t is the generation count, L the length of chromosomes, and T the maximum generation;

CPLO: $p_c = 0.65$, $p_m = 0.1$ for parent mutation, and 0.001 for offspring mutation;

AGA: the expressions for p_c and p_m were

$$p_c = (f_{\max} - f') / (f_{\max} - f_{\text{avg}}), \qquad f' \geq f_{\text{avg}},$$
$$p_c = 1.0, \qquad f' < f_{\text{avg}},$$

and

$$p_m = 0.5\ (f_{\max} - f') / (f_{\max} - f_{\text{avg}}), \qquad f \geq f_{\text{avg}},$$

$$p_m = 0.5, \qquad f < f_{\text{avg}},$$

where $f_{\max}$ represents the maximum fitness value of the population, f_{avg} the average fitness value of the population, f the fitness value of the solution undergone mutation, and f' the larger of the fitness values of the solutions to be crossed.

We first examined the effects of different step sizes for adaptation of p_c and p_m. To understand the effect of varying θ_1 and θ_2 under constant step sizes, we first measured the PRGA performance using four different settings: 1) $\theta_1 = \theta_2 = 0.01$; 2) $\theta_1 = 0.01$ and $\theta_2 = 0.001$; 3) $\theta_1 = 0.001$ and $\theta_2 = 0.01$; 4) $\theta_1 = 0.001$ and $\theta_2 = 0.001$. The results are presented in Table 4.

Table 4. Best fitness obtained by PRGA using different constant step sizes of θ_1 and θ_2.

	$\theta_1 = 0.01$ $\theta_2 = 0.01$	$\theta_1 = 0.01$, $\theta_2 = 0.001$	$\theta_1 = 0.001$, $\theta_2 = 0.01$	$\theta_1 = 0.001$ $\theta_2 = 0.001$
Uncorrelated, $C = 2v$	114	89	87	78
Uncorrelated, $C = 0.5\Sigma W_i$	1078	983	1089	1069
Weakly, $C = 2v$	54	54	58	54
Weakly, $C = 0.5\Sigma W_i$	1063	962	1079	947
Strongly, $C = 2v$	60	65	65	65
Strongly, $C = 0.5\Sigma W_i$	1470	1435	1500	1460

As Table 4 shows, no combination was superior to the others in all test suits. This observation led us to seek a self-adaptive control of the step sizes. Inspired by [24], we tuned the values of θ_1 and θ_2 according to the population convergence. When the population converged, we increased the step sizes to reduce the probability of the GA getting stuck in a local optimum. The resulting self-adapting function is expressed as follows:

$$\theta_1 = \theta_2 = 0.01 \frac{f_{\max} - f_{avg}}{f_{\max} - f_{\min}}, \text{ if } f_{\max} > f_{\min},$$

$$\theta_1 = \theta_2 = 0.01, \text{ if } f_{\max} = f_{\text{avg.}},$$

where the convergence is measured as the difference between the maximum and average population fitness, $f_{\max} - f_{\text{avg}}$, and normalized by $f_{\max} - f_{\min}$.

We next compared the other four methods with our PRGA algorithm using the adaptive step size function. The results are shown in Figs. 2 to 7. Our PRGA outperformed the competitors in all test cases except for the weakly correlated case $C = 0.5\sum W_i$, where AGA led. For the most difficult problem, as shown in Fig. 2, SGA converged quickly to a local optimum and failed to perform further exploration. CPLO, with cyclic-rate mutation, exhibited the ability to escape from local traps, but the resulting improvement was not clear. DMRGA and AGA performed well initially but failed to perform further exploration when the population began to converge. Similar phenomena were observed for the other two instances in Group 1, as shown in Figs. 4 and 6.

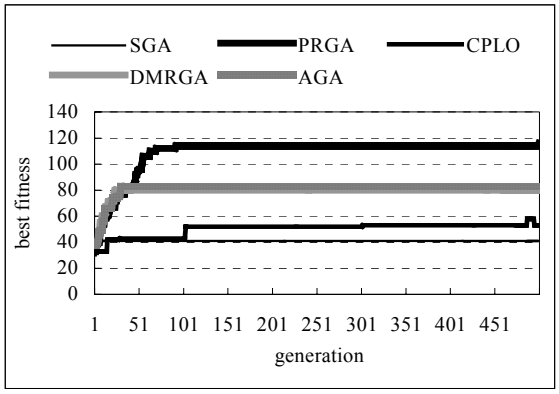

Fig. 2. Experimental results for the uncorrelated 0/1 knapsack problem with $C = 2v$.

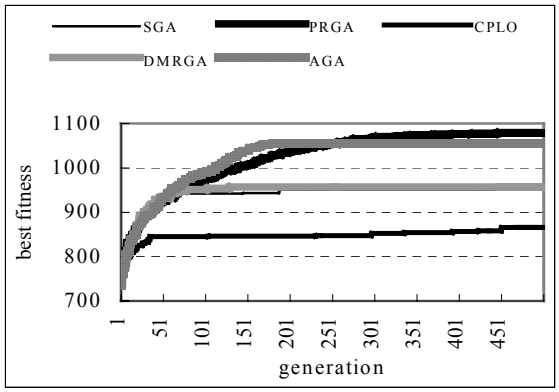

Fig. 3. Experimental results for the uncorrelated 0/1 knapsack problem with $C = 0.5\sum W_i$.

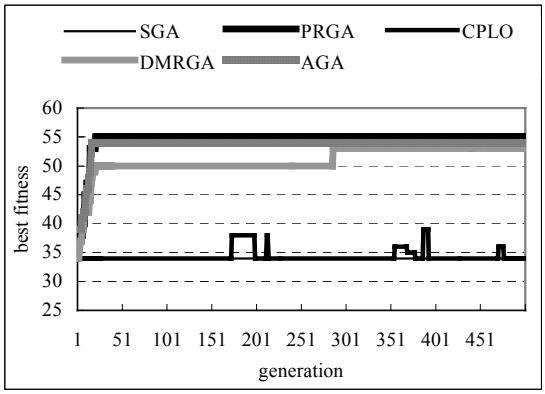

Fig. 4. Experimental results for the weakly correlated 0/1 knapsack problem with $C = 2v$.

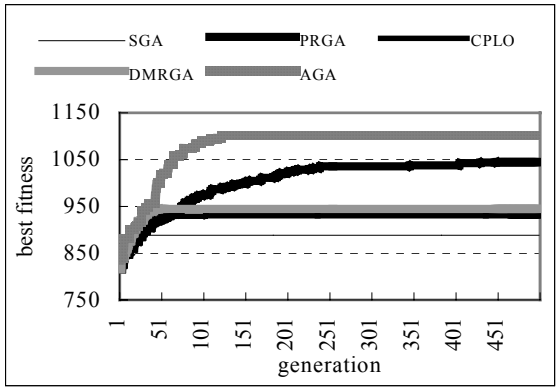

Fig. 5. Experimental results for the weakly correlated 0/1 knapsack problem with $C = 0.5\sum W_i$.

For the cases within Group 2, we found that the performance gap between PRGA and the competitors was smaller. Note that the ranking of SGA, CPLO, DMRGA, and AGA, however, was different. CPLO ranks lower in Fig. 3, while SGA ranks last in Fig. 5, and DMRGA ranks last in Fig. 7. This phenomenon reveals that when the interaction between crossover and mutation is not considered, solely adapting the mutation rate can guarantee no performance gain.

To understand how p_c and p_m evolved when our method was employed, we also recorded the rates at the end of each generation. We only show the results for the uncorrelated case with $C = 2v$; similar results were observed for the other cases.

Fig. 8 depicts the results, where p_m rises aggressively throughout evolution to reach the upper bound, while p_c behaves in the opposite manner. Note that the evolution map of p_m contradicts the conventional suggestion made by most researchers [2, 3, 8, 12] that p_m should behave as a descending curve. These results also confirm the necessity of using operator rate adaptation scheme to locate prospective solution.

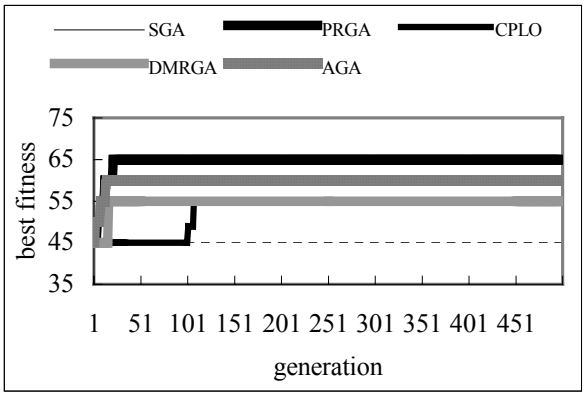

Fig. 6. Experimental results for the strongly correlated 0/1 knapsack problem with $C = 2v$.

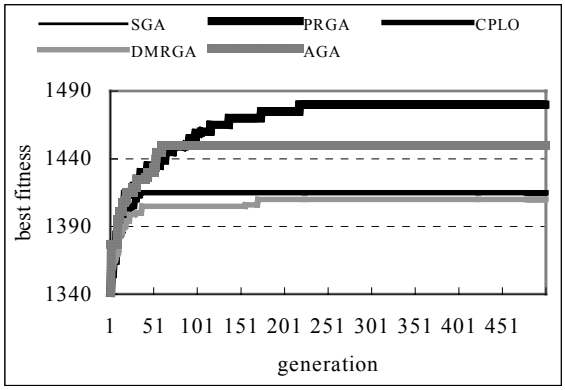

Fig. 7. Experimental results for the strongly correlated 0/1 knapsack problem with $C = 0.5\sum W_i$.

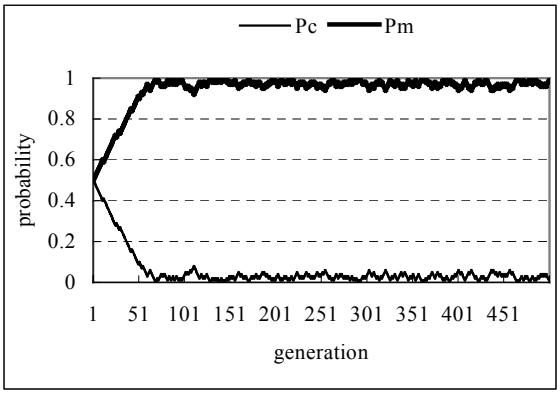

Fig. 8. Evolution of p_c and p_m for the uncorrelated 0/1 knapsack problem with $C = 2v$.

5. CONCLUSION AND FUTURE WORK

In this paper, we have presented an adaptive genetic algorithm for automatically adjusting suitable crossover and mutation rates to reduce the effort of searching for appropriate crossover and mutation rates. Our approach takes into account the interaction between crossover and mutation in adapting the operator rates. The performance of the proposed genetic algorithm has been empirically shown to be better than that of previous schemes. In the future, we will attempt to design other sophisticated adaptation schemes that include other parameters, such as the population size and the replacement rate.

REFERENCES

1. H. Aytug and G. J. Koehler, "Stopping criteria for finite length genetic algorithms," *INFORMS Journal on Computing*, Vol. 8, 1996, pp. 183-191.

2. T. Bäck, "Self-adaptation in genetic algorithms," in *Proceedings of the First European Conference on Artificial Life*, 1992, pp. 263-271.
3. T. Bäck, "Optimal mutation rates in genetic search," in *Proceedings of the Fifth International Conference on Genetic Algorithms*, 1993, pp. 2-8.
4. K. Deb and S. Argrawal, "Understanding interactions among genetic algorithm parameters," in *Foundations of Genetic Algorithms 5*, 1998, pp. 265-286.
5. K. A. De Jong, "An analysis of the behavior of a class of genetic adaptive systems," PhD thesis, University of Michigan, 1975.
6. K. A. De Jong, "Adaptive system design: A genetic approach," *IEEE Transactions on System, Man and Cybernetics*, Vol. 10, 1980, pp. 566-574.
7. A. E. Eiben, R. Hinterding, and Z. Michalewicz, "Parameter control in evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, Vol. 3, 1999, pp. 124-141.
8. T. C. Fogarty, "Varying the probability of mutation in genetic algorithms," in *Proceedings of the Third International Conference on Genetic Algorithms*, 1989, pp. 104-109.
9. D. E. Goldberg, *Genetic Algorithms in Search, Optimization & Machine Learning*, Addison Wesley, 1989.
10. D. Greenhalgh and S. Marshall, "Convergence criteria for genetic algorithms," *SIAM Journal on Computing*, Vol. 30, 2000, pp. 269-282.
11. J. J. Grefenstette, "Optimization of control parameters for genetic algorithms," *IEEE Transactions on System, Man and Cybernetics*, Vol. 16, 1986, pp. 122-128.
12. J. Hesser and R. Männer, "Towards on optimal mutation probability for genetic algorithms," in *Proceedings of Parallel Problem Solving from Nature Conference*, 1990, pp. 23-32.
13. T. P. Hoehn and C. C. Pettey, "Parental and cyclic-rate mutation in genetic algorithms: an initial investigation," in *Proceedings of Genetic and Evolutionary Computation Conference*, 1999, pp. 297-304.
14. J. H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, 1975.
15. T.-P. Hong, H.-S. Wang, W.-Y. Lin, and W.-Y. Lee, "Evolution of appropriate crossover and mutation operators in a genetic process," *Applied Intelligence*, Vol. 16, 2002, pp. 7-17.
16. B. A. Julstrom, "What have you done for me lately? Adapting operator probabilities in a steady-state genetic algorithm," in *Proceedings of the Sixth International Conference on Genetic Algorithms*, 1995, pp. 81-87.
17. S. Martello and P. Toth, *Knapsack Problems*, Jonh Wiley, UK, 1990.
18. Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, Springer-Verlag, 1994.
19. M. Mitchell, *An Introduction to Genetic Algorithms*, MIT press, 1996.
20. G. Ochoa, I. Harvey, and H. Buxton, "On recombination and optimal mutation rates," in *Proceedings of Genetic and Evolutionary Computation Conference*, 1999, pp. 488-495.
21. K. Park, "A comparative study of genetic search," in *Proceeding of the Sixth International Conference on Genetic Algorithms*, 1995, pp. 512-519.

22. J. D. Schaffer and A. Morishima, "An adaptive crossover distribution mechanism for genetic algorithms," in *Proceeding of the Second International Conference on Genetic Algorithms*, 1987, pp. 36-40.
23. J. D. Schaffer, et al, "A study of control parameters affecting online performance of genetic algorithms for function optimization," in *Proceeding of the Third International Conference on Genetic Algorithms*, 1989, pp. _____
24. M. Srinias and L. M. Patnaik, "Adaptive probabilities of crossover and mutation in genetic algorithms," *IEEE Transactions on System, Man and Cybernetics*, Vol. 24, 1994, pp. 656-667.
25. A. Tuson and P. Ross, "Cost based operator rate adaptation: an investigation," in *Proceedings of Parallel Problem Solving from Nature Conference*, 1996, pp. 461-469.

Wen-Yang Lin (林文揚) received his B.S. and M.S. both in Computer Science and Information Engineering from National Chiao-Tung University in 1988 and 1990, respectively. He then received his Ph.D. in Computer Science and Information Engineering from National Taiwan University in 1994. In 1996, he joined the faculty of the Department of Information Management at I-Shou University and now is an Associate Professor. He is primarily interested in the area of sparse matrix technology and large-scale supercomputing. Currently he is also interested in data warehousing, data mining and evolutionary computations. Dr. Lin is a member of SIAM, IEEE, the Taiwanese AI Association and the Institute of Information and Computing Machinery.

Wen-Yuan Lee (李文淵) was born in Kaohsiung, Taiwan in 1974. He earned his B.E. degree in Information Management in 1999, and M.S. degree in Information Engineering in 2001, both from the I-Shou University. In 2001, He joined the R&D Department of the TransAsia Telecommunications Inc. in Kaohsiung, Taiwan. In 2002, he designed new applications for VASMS (VAS monitoring system) to improve the quality of the short message service. In 2003, he developed WAP Push Engine for Java game and multi-media downloading. He is currently a VAS development engineer for the short message group and a technical project leader. His main research interest is Genetic Algorithms.

Tzung-Pei Hong (洪宗貝) received his B.S. degree in chemical engineering from National Taiwan University in 1985, and his Ph.D. in computer science and information engineering from National Chiao-Tung University in 1992.

From 1987 to 1994, he was with the Laboratory of Knowledge Engineering, National Chiao-Tung University, where he was involved in applying techniques of parallel processing to artificial intelligence. He was an associate professor at the Department of Computer Science in Chung-Hua Polytechnic Institute from 1992 to 1994, and at the Department of Information Management in I-Shou University (originally Kaohsiung Polytechnic Institute) from 1994 to 1999. He was a professor in I-Shou University from

1999 to 2001. He was in charge of the whole computerization and library planning for National University of Kaohsiung in Preparation from 1997 to 2000. He was also the first director of the library and computer center in National University of Kaohsiung from 2000 to 2001. He is currently a professor at the Department of Electrical Engineering in National University of Kaohsiung. He has published more than 250 research papers in international/national journals and conferences. He has also planned for more than fifty information systems. His current research interests include artificial intelligence, soft computing, data mining, parallel processing, management information systems and www applications.

Dr. Hong is a member of the Association for Computing Machinery, the IEEE, the Chinese Fuzzy Systems Association, the Taiwanese Association for Artificial Intelligence, and the Institute of Information and Computing Machinery.