A Parameter-less Genetic Algorithm with Customized Crossover and Mutation Operators

Farhad Nadi and Ahamad Tajudin Khader
School of Computer Sciences, Universiti Sains Malaysia, 11800 USM
Penang, Malaysia
fn.com07@student.usm.my, tajudin@cs.usm.my

ABSTRACT

Genetic algorithm is one of the well-known population based meta-heuristics. The reasonable performance of the algorithm on a wide variety of problems as well as its simplicity made this algorithm a first choice in lots of cases. However, the algorithm has some weaknesses such as the existence of some parameters that need to be carefully set before the run. The capability of the parameters to change the balance between exploration and exploitation make them crucial. Exploration and exploitation are the bases of every evolutionary algorithm. Conducting a balance between these elements is crucial for the success of any evolutionary algorithm. In this research a GA is proposed on which the crossover and mutation rates are removed. A probability vector holds the probability of the alleles for every locus within the individual. The probability is with regards to the contribution of the allele on either increasing or decreasing the fitness of the chromosome. The probability of an allele will increase if the fitness of the chromosome increases by a change or vise versa. The experiments conducted on a wide range of multi-modal and epistatic problems show good performance of the proposed method in comparison to other algorithms in literature.

Categories and Subject Descriptors

G.2.1 [**DISCRETE MATHEMATICS**]: Combinatorics— *Combinatorial algorithms*; G.1.6 [**NUMERICAL ANALYSIS**]: Optimization— *Global optimization*

General Terms

Algorithms, Design, Experimentation, Performance, Theory

Keywords

Genetic Algorithms, Parameter Control, Exploration, Exploitation, Crossover Rate, Mutation Rate

GECCO'11, July 12–16, 2011, Dublin, Ireland.

1. INTRODUCTION

Evolutionary algorithm (EA) is a sub-group of search meta-heuristics that generally improves the quality of a set of candidate solutions iteratively using variation and selection operators[9, 20, 32]. Parameters play an important role in EA. In particular, variation operators, parent selection, and survivor selection are reported as the more crucial components of EA [28]. Michalewicz [27] refers to population diversity and selective pressure as two important factors in the evolutionary process of a genetic search. He asserts that it is important to maintain a balance between these two issues. Values of parameters could be used to control this balance as they have effect on both diversity and selection pressure of the population. It could be concluded that it is the right balance between the exploration and exploitation that really matter for a genetic algorithm and not the parameters per se. However, in traditional EAs this balance could be conducted using the existing parameters.

This paper focuses on Genetic Algorithm (GA). John Holland introduced GA in early seventies [19]. In GA, generally a set of parents will be selected from a population of solutions. Through variation operators, normally crossover and mutation, a set of offspring will be created. The set of offspring will replace a set of old individuals in the population.

The motivation for the research in parameter control stems from the fact that it has been shown that there is no universal optimal parameter set for GAs [17], in other words, GAs need to be tuned for each problem independently. Even if a set of parameter values are optimal in the beginning of the run, it does not necessarily means that those values remain optimal during the whole run. In other words, it is essential for the algorithm to be instantiated with different parameter values during different stages of the run. Another motivation is based on the goal of designing black-box optimization algorithms where the reduction of the number of parameters is favored [16].

The literature on parameter control could be divided into two different approaches as follows. The first approach is known as *parameter assignment*, on which values for the parameters will be provided using different methodologies. Whereas, in the second approach, which will be referred to as *parameter-less*, removal of the parameters is of interest.

From a black box point of view, it does not matter if an algorithm has the automated parameters or does not have any parameters. What matters is that there should be no need for tuning any parameter.

A parameter-less GA is proposed where crossover and mutation rates are removed. However, customized versions of

crossover and mutation operators still exist. An improvement in fitness of a given chromosome would increase the probability of occurrence of the alleles on the loci that has been changed and vice versa.

The rest of this paper is organized as follows. First we briefly look into the background and related works in section 2. Then we introduce the proposed method in section 3. How the experiment has been conducted is mentioned in section 4. Experimental results will be discussed in section 5. Last section, i.e. section 6, provides the conclusion.

2. BACKGROUND

In this section parameter control methods shall be introduced. These could be classified as parameter assignment approaches and parameter-less approaches. In parameter assignment approaches the parameters are actually within the algorithm and different methodologies are used for determining their values. Whereas in parameter-less approaches the parameters of the algorithms will be removed and thus the algorithm has no more parameters to be assigned.

2.1 Parameter assignment approaches

Although this approach could be categorized into more detailed sub-approaches (for example following the categorization introduced in [9]), here we reviewed a few of them regardless of their sub-approach.

In a series of parameter control methods referred to as meta-algorithm, two algorithms will be utilized. The first algorithm (meta-EA) will be used to tune the parameters of the second algorithm (EA) that solves the problem [9]. This way, the meta-algorithms tune the parameters of a given problem. The meta-EA evaluate different combinations of the parameters based on their suitability for the given problem [7]. In a method proposed in [12], a two-level GA is utilized. In the meta-level, a set of parameters will evolve. In the other level, referred to as basic-level, the real problem will operate on the best set of parameter values found by the meta-level. REVAC is another work proposed in [7] where the parameters of a given EA will be refined iteratively over the possible parameter vectors. This method utilizes *Estimation of Distribution Algorithm* (EDA) [29] for finding the best parameter set for a given EA. EDA is type of algorithm that is mainly designed for maximizing the entropy in continuous domains [29].

Deterministic parameter control methods refer to all of the methods that change the parameters values based on some deterministic rules [10]. As an example, in Deterministic GA [30], the mutation rate (p_m) will change deterministically during the course of run.

Lobo [23] in his PhD thesis investigates a parameter assignment method referred to as parameter-less GA. The parameters were adaptively set either automatically (population size) or rationally (selection rate and crossover rate) by the algorithm itself based on theoretical foundations. An extension of Lobo's work [22] integrates local search with the previous work. They have shown that using local search for exploitation of the search space is beneficial.

An adaptive GA based on fuzzy logic controller was introduced in [25]. Static rule, inference engine and feedback are used in the proposed methodology. Another work based on fuzzy logic was proposed in [5]. An intelligent fuzzy controller changes the GA's parameters through monitoring the GA's status. Frequency of best individuals within the population, number of duplicate individuals, and number of expected optimal values show the status of the GA during the run. Subsequently, fuzzy rules will be fired according to the status of the GA, which would in turn change the values of the parameters.

Adaptive genetic algorithm (AGA) [36] adaptively controls both mutation and crossover rates. The adapted rates get adjusted for every individual based on its fitness and current state of the population convergence. The crossover and mutation rates are defined as,

$$p_c = \begin{cases} k_1(f_{max} - f')/(f_{max} - \bar{f}) & \text{if } f' \geq \bar{f} \\ k_3 & \text{otherwise} \end{cases}$$

and

$$p_m = \begin{cases} k_2(f_{max} - f)/(f_{max} - \bar{f}) & \text{if } f \geq \bar{f} \\ k_4 & \text{otherwise} \end{cases}$$

where $k_1, k_2, k_3, k_4 \leq 1.0$ are constants, f is fitness of the individual, f_{max} is best existing fitness, f' is the largest fitness of the parents that are selected for crossover, and $\bar{f}$ is the average fitness of the population.

A self-adaptive parameter control method was proposed in [34]. Mutation and crossover rates in this method have been controlled using a self-adaptive method. Each individual has a mutation rate in itself and the global mutation rate will be calculated based on the individual mutation rate and the global value. In another work based on a probabilistic rule-based adaptive methodology [18], the mutation and crossover rates get adapted during the run. Different subpopulation use different parameter rates in this method. The rates of the parameters will be based on the rates of the parameters on the best performing sub-population.

Eiben [8] introduced Hybrid Self-adaptive GA (HSGA). In the proposed method an extra gene is added to the end of normal chromosome, representing the size of tournament. The crossover operator applied on the whole chromosome, including the added gene. However, the mutation operator has been restricted to the normal chromosome and a special mutation has been designed for the extended part. The author tried to bias the adaptation towards a predefined heuristic, where the less fit individuals get less selection pressure while fitter individuals get higher selection pressure.

In a self-adaptive method (SAGA) [3], the mutation rate is adapted. Every individual is extended with an extra bit (μ) holding the mutation rate. Mutating this extended bit will be done using,

$$\acute{\mu} = \left(1 + \frac{1-\mu}{\mu} exp(-\gamma \cdot N(0,1))\right)^{-1}.$$

Where γ is a constant value, and $N(0,1)$ is a normal distributed number with mean 0.0 and standard deviation 1.

2.2 Parameter-less approaches

This approach, covers all of those methods where even one parameter is reduced from the algorithm. Ideally, all of the parameters should be removed in order to be able to refer to an algorithm as parameter-less. It is worthy noting that there are approaches that remove some of the parameters from the algorithms while introduce some other new ones.

Generally, in parameter-less GAs, the population and crossover operators are replaced with probabilistic model representation and generation models [4, 14, 24, 26, 31]. *GA with variable population size* (GAVaPS) [1] could be categorized

as a parameter-less method. In this method, the population size is eliminated from the algorithm with the introduction of age and maximum life time to each individual. Age of the individuals will be increased by one in each generation and the individuals that reached the maximum life time will be removed from the population. The *Adaptive Population Size GA* (APGA) is another variation of GAVaPS [2]. Here, a steady-state GA is used but the age of the best individuals will remain unchanged.

Compact GA (cGA) was proposed as a variation of GA where population is represented as a probability distribution over the set of solutions [15]. It processes each gene independently while it tries to create the same distribution like the previous population. However, cGA considers population size as a parameter.

In another work presented in [38] two mutation probabilities, p_m^0, p_m^1, are defined for each locus. The appropriate mutation rate will be applied for each locus based on the allele . In each generation, the mutation probability for each locus will be updated based on correlated statistic of the population with respect to the relative success of individual with specific allele and allele distribution of the locus. A convergence mechanism is also utilized in this methodology by which a set of randomly selected individuals get complemented when the population converged to a certain degree. The results have shown that the methodology has improved GA's performance.

SSRGA [37] is a canonical GA on which crossover operator is removed from the algorithm. In this method both exploration and exploitation is carried out through a customized mutation operator. The main steps of this method is defined as follows:

- partitioning the population into a set of suitable subpopulations.
- Computing site-specific rate vector for each of the partitions.
- Mutate each subpopulation using the corresponding site-specific vector.

The partitioning supposed to divide the population based on *basins of attraction*. The authors asserts that this site specific rates will be more accurate especially in the cases where the problem is multi-modal. This algorithm continues by computing a probability for all the loci of an individual. For a given subpopulation the probability of value $a, a \in [0,1]$ for each loci will be calculated proportional to a function of the individual fitness,

$$p_j^i(a) = p_{min} + (P_{max} - p_{min}) \frac{\sum_{x \epsilon s^i} F(x)\delta(x_j = a)}{\sum_{x \epsilon s^i} F(x)},$$

where x is individual, x_j is the allele at j^{th} locus, $\delta(c)$ returns 1 if condition c is satisfied or otherwise 0 will be returned, and $F(x) = e^{5f(x)}$. In the last step, the respective mutation operator for each partition will be applied to each subpopulation.

3. PROPOSED METHOD

As mentioned earlier, the main idea is to credit the alleles which caused fitness improvement. In other words, for every loci, the allele which played more role in improving the fitness will be more probable to occur. For a given individual,

$$I = \{i_1, i_2, \ldots, i_{l-1}, i_l\}, i \in \{0,1\}$$

with length l , there will be a probability vector (Z),

$$Z = \{z_1, z_2, \ldots, z_{l-1}, z_l\}, z \in [0,1]$$

where z_i is the occurrence probability value in loci i_i. Unlike the conventional GA, here there will be no rates for recombination and mutation. Mutation operator simply create an individual based on the probability vector. Mutating the individuals based on the probability vector would result in exploitation of the search space. This is because the probability vector is biased towards creation of the solutions met so far. Whereas, crossover operator works in two-fold by exploring and exploiting the the search space.

As mentioned earlier, Z vector contains the occurrence probability of alleles on each locus of the chromosome based on the previous solutions. Creation of individuals according to Z would result in chromosomes in the adjacency of each other. In other words, reproduction of chromosomes in this manner would exploit the search space.

The more the recreation of new solutions be with regards to Z vector, the more the search space will get exploited. Therefore, exploration of the search space could be achieved, by a lesser regards to Z vector. The crossover operator (see Algorithm 1) explore the search space in two different ways. Firstly, in reproduction of off-springs exploration occurs implicitly (lines 12-16 in Algorithm 1). The genes will be exchanged according to Z vector. This will results in an offspring highly according to Z vector while the other offspring is not in accordance with the Z vector. The second way of exploration, lines 4-10 in Algorithm 1, happens explicitly as part of crossover. In the designed crossover operator, the values for a random sequence of the locus, $I_i ... I_j, j > i, i > 0, j < l$, will be chosen randomly with a random distance with Z_i for each locus.

Algorithm 1 Pseudo code for crossover operator

```
1: {inputs to the function are two parent chromosomes
   namely Chrom_1, Chrom_2.}
2: TmpZ ← Z
3: if Rnd() < 0.5 then
4:    { RndInt(x) returns a random integer value between
      0 and x}
5:    start ← RndInt(l/2 − 2)
6:    end ← start + RndInt(l − start)
7:    for i = start to end do
8:       { Rnd() returns a random number between [0,1)}
9:       TmpZ[i] = |Rnd() − Z[i]|
10:   end for
11: end if
12: for i = 0 to l do
13:   if Rnd() < TmpZ[i] .AND. Chrom_1[i]! = Chrom_2[i]
      then
14:      Chrom_1[i] = 1
15:      Chrom_2[i] = 0
16:      τ++
17:   end if
18: end for
```

Recombination and mutation operators changes the alleles on different loci of chromosomes. Any change in a given

chromosome will have an effect on the fitness of the chromosome. Depending on the number of changes (τ) that have been done on a given chromosome, the Z vector will get updated. A credit of $\frac{\epsilon}{\tau}$ will be given to each corresponding loci on Z. The credit could either reduce or increase z_i with regards to the type of change in allele and also the change in fitness as shown in Table 1.

Table 1: Probability vector updates is based on the change in fitness prior and after the changes on individual. Depending on the fitness the probability vector will get positive or negative credit.

from	to	Fitness	credit
Gene A	Gene B		
0	1	improved	+
0	1	worsen	-
1	0	Improved	-
1	0	worsen	+

As an example, assume individual $I = \{1, 1, 0, 0, 1, 0, 0, 1\}$ with fitness value 4. Let assume this individual changed to $I = \{1, 1, \mathit{1}, \mathit{1}, \mathit{0}, 0, 0, 1\}$ with the new fitness 5. Here the fitness improved after the changes. As three loci are changed then three loci in the probability vector get updated, each with $\frac{1}{3}$, assuming $\epsilon = 1$. Therefore, if we assume before the update $Z = \{0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5\}$. Then after the changes $Z = \{0.5, 0.5, \mathit{0.83}, \mathit{0.83}, \mathit{0.83}, 0.5, 0.5, 0.5\}$.

4. EXPERIMENT SETUP

The performance of the proposed method assessed over multimodal and epistatic problems. However, all of the benchmarks are under the domain of maximum satisfiability (MAX-SAT) problems.

MAX-SAT is a generalized version of satisfiability (SAT) decision problem that belongs to the family of NP-hard optimization problems.

A SAT problem is consisted of,

- A set of m variables.
- A set of literals, where a literal is a variable or its negation.
- A set of n distinct clauses, each of them combined just using logical $OR(\vee)$.

The problem in SAT is to find if there exists any assignment for the variables that satisfies the following formula which is in Conjunctive Normal From (CNF),

$$C_1 \wedge C_2 \wedge \ldots \wedge C_n.$$

However, in MAX-SAT the problem is to find an assignment which maximizes the number of satisfied clauses [6, 11, 13].

The binary nature of MAX-SAT problem makes it very compatible with GA. However, unlike the nature of the problem, selection of an efficient fitness function for the problem is crucial. A simple fitness function which translate 0 and 1 values to $True$ and $False$ respectively would be inefficient. Such a function is not capable of evaluating of the intermediate solutions as it would assign 0 to almost all of the points in the search space.

A better fitness function is suggested in [35] on which a fitness could be assigned to individual subexpressions in the original expression. The combination of fitness values then would result in a final fitness for the expression.

Another similar fitness function is proposed by Smith [33] where 0 represents a *false* and 1 represents *true* fitness of every subexpression e_i of an expression e will be calculated as follows:

$$f(\bar{e}) = 1 - f(e),$$

$$f(e_1 \vee e_2 \vee \ldots \vee e_n) = Max(f(e_1), f(e_2), \ldots, f(e_n)),$$

$$f(e_1 \wedge e_2 \wedge \ldots \wedge e_n) = Ave(f(e_1), f(e_2), \ldots, f(e_n)),$$

where Ave returns the average value of its parameters, and Max returns the parameter with maximum value.

Two classes of benchmarks are used in the comparisons, including multi-modal boolean satisfiability and epistatic problems. These problems will be introduced in the following subsections. Degree of multi-modality in a problems could be defined as a measure of difficulty of the problem. In these problems, the number of false peaks grows with the number of modality. In the other side, the degree of epistasis of a problem expresses the relationship between the genes in a chromosome. Dependency of a large number of alleles at other loci is a sign of high epistasis in a system [35]. The degree of epistasis has direct relation with difficulty level of the problem.

The compared algorithms are canonical GA with a randomly chosen constant mutation rate, SSRGA [37], a *self-adaptive (SAGA)* [3] and an *adaptive (AGA)* [36] parameter control methods.

4.1 Multi-modal problems

The multi-modal problems used for the course of experiment are created using a mechanism proposed by Spears [35].

A unimodal problem of length 30 could be created as follows,

$$1Peak \equiv (x_1 \wedge x_2 \wedge \ldots \wedge x_{30}).$$

Using the above unimodal problem, a bimodal problem will be,

$$2Peak \equiv 1Peak \vee (x_1 \wedge \bar{x}_1 \wedge \bar{x}_2 \wedge \ldots \wedge \bar{x}_{30}).$$

As it could be inferred from the definition of bimodal problem, the extended sub-expression could not get fully satisfied because of the existence of $x_1 \wedge \bar{x_1}$. However, here the second expression could get a value very near to the value of global optima, which is 1. This second peak is usually referred to as "false peak". Obviously the probability of finding the global optima will decrease with the growth of the number of false peaks. For the course of this experiments, multi-modal problems up to 5 peaks have been used. The higher degrees of multi modality used in the experiments are defined as,

$$3Peak \equiv 2Peak \vee (x_1 \wedge \bar{x}_1 \wedge \bar{x}_2 \wedge \ldots \wedge \bar{x}_{15} \wedge x_{16} \wedge \ldots \wedge x_{30})$$

$$4Peak \equiv 3Peak \vee (x_1 \wedge \bar{x}_1 \wedge x_2 \wedge \ldots \wedge x_{15} \wedge \bar{x}_{16} \wedge \ldots \wedge \bar{x}_{30})$$

$$5Peak \equiv 3Peak \vee (x_1 \wedge \bar{x}_1 \wedge x_2 \wedge \bar{x}_3 \wedge x_4 \wedge \bar{x}_5 \ldots \wedge \bar{x}_{29} \wedge x_{30})$$

4.2 Epistatic problems

Spears [35] has also introduced a method for the creation of epistatic problems using boolean expressions. His proposed method has the capability for increasing the level of

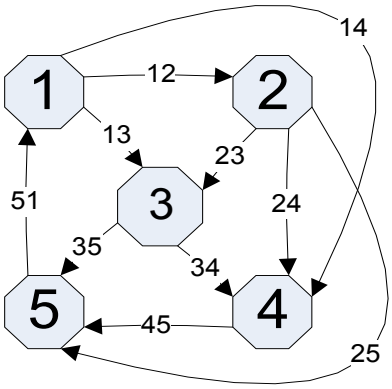

Figure 1: An example of a Hamiltonian circuit with five nodes. The edges are connected following the proposed method by Spears for creating epistasis problems.

epistasis, thus making different epistasis problems with different levels of difficulty. The Spears method is based on conversion of the Hamiltonian Circuit (HC) problems into the SAT expressions. In a directed HC problem, the aim is to find if a given graph has a hamiltonian cycle. By definitions, a hamiltonian cycle is a cycle in a graph that meet all of the vertex exactly once. The definition of HC constraint the nodes of feasible solutions to have only one input edge and one output edge. Having this constraint, any tour that does not satisfies this constraint cannot be a solution [21]. Using conjunction of terms that indicate the valid edge combinations for each node, the equivalent satisfiability boolean expression for this constraint could be constructed.

Figure 1 depicts a hamiltonian circuit that uses Spears method for creation of epistatic problems. Spears defined a graph $G = (V, E)$ with N nodes where nodes are labeled using sequential integers from 1 to N. The connection of the nodes by the edges will be as follows.

- The first node has output edges to all other nodes except the last one.
- All of the next $N - 2$ nodes have edges to all other nodes with the higher label.
- The last node has a direct edge back to the first node.

As such, the only valid Hamiltonian tour is the tour that the nodes are labeled in the increasing order, and the last node finishes the cycle. By increasing N, the level of epistasis and thus the difficulty of the problem increases [21].

As for an example, the boolean expression for Figure 1 with regards to the Spears method will be as follow:

$$
\begin{gathered}
12 \wedge 23 \wedge 34 \wedge 45 \wedge 51 \wedge \\
(12 \wedge \bar{13} \wedge \bar{14}) \vee (\bar{12} \wedge 13 \wedge \bar{14}) \vee (\bar{12} \wedge \bar{13} \wedge 14) \wedge \\
(23 \wedge \bar{24} \wedge \bar{25}) \vee (\bar{23} \wedge 24 \wedge \bar{25}) \vee (\bar{23} \wedge \bar{24} \wedge 25) \wedge \\
(34 \wedge \bar{35}) \vee (\bar{34} \wedge 35) \wedge \\
(23 \wedge \bar{13}) \vee (\bar{23} \wedge 13) \wedge \\
(14 \wedge \bar{24} \wedge \bar{34}) \vee (\bar{14} \wedge 24 \wedge \bar{34}) \vee (\bar{14} \wedge \bar{24} \wedge 34) \wedge \\
(25 \wedge \bar{35} \wedge \bar{45}) \vee (\bar{25} \wedge 35 \wedge \bar{45}) \vee (\bar{25} \wedge \bar{35} \wedge 45)
\end{gathered}
$$

The hamiltonian circuit in Figure 1 has 5 nodes, with 15 , i.e. $\frac{N(N-1)}{2}$, edges and 21 clauses. Following this, the resultant boolean expression should be converted into CNF format and thus be used as MAX-SAT problem.

A simple GA is used for the course of the experiments where its attributes are shown in Table 2.

Table 2: Simple GA's attributes

Population model	steady state
Parent selection	Tournament selection
Survival selection	delete oldest
Selection pressure	8
Population size	50
Max. no. of generations	500 (epistatic), 100(multimodal)

The algorithms performance is measured over 50 independent runs for each of the problems. The average (*avg.*) and standard deviation (*stdev.*) of the best results are derived for all of the experiments. Highest average is highlighted with **bold** font while the lowest standard deviation is underlined for each case.

5. RESULTS AND DISCUSSION

The results of the conducted experiments on multi-modal problems are listed in Table 3. Table 4 shows the results of epistatic problem.

In case of the multi-modal problems, the performance of the proposed method in all of the cases shown to be better with the compared methods. However, in terms of the robustness, the proposed method is comparable to the other methods.Only in one case, $p = 1$, the proposed method has a better standard deviation in comparison to the other compared methods.

As for the epistasis problems, the performance is better than all the compared algorithms, with exception when $N = 6$. The standard deviations have been better in three cases and comparable in the other cases.

In order to visualize the performance behavior of the proposed method, we have derived the differences between the proposed method's results and results of all other methods. Figure 2 and Figure 3 visualizes the differences for multi-modal and epistatic problems respectively.

According to Figure 3, the performance of the proposed method is better in the mid-range valued epistasis degrees. The differences narrows as the level of epistasis increases. However, in all of the cases the performance of the proposed method is shown to be better.

In the multi-modal case (see Figure 2), the performance of the algorithm is better in all of the cases. The gap between the proposed method and the compared methods seems to be constant in all of the cases. $SSGA$ [37] has the smallest gap with the proposed method.

We have done some experiments by running mutations and crossover alone. The results have shown to be not as good as those that are reported when both of them are in use.

Considering different probability vectors for different parts of the search space could be studied in future works. The idea seems to be very prominent specially in the case of multi-modal problems where there are several peaks within the search space.

Table 3: Comparison of the proposed method with benchmark methods on literature on multi-modal MAX-SAT problem with different levels of multi-modality.

Modality		$p=1$	$p=2$	$p=3$	$p=4$	$p=5$
Proposed Method	avg.	**0.999**	**0.976**	**0.970**	**0.964**	**0.961**
	stdev.	0.007	0.023	0.025	0.029	0.026
SSRGA	avg.	0.952	0.945	0.931	0.940	0.923
	stdev.	0.020	0.022	0.025	0.025	0.028
AGA	avg.	0.874	0.870	0.866	0.868	0.876
	stdev.	0.022	0.029	0.026	0.022	0.025
SAGA	avg.	0.827	0.842	0.840	0.853	0.846
	stdev.	0.029	0.029	0.022	0.027	0.024
CGA	avg.	0.834	0.843	0.848	0.850	0.842
	stdev.	0.029	0.029	0.022	0.027	0.024

Table 4: Comparison of the proposed method with benchmark methods of literature on epistatic problem with different levels of epistasis.

Degree of epistasis		N=6	N=11	N=16	N=21	N=26	N=31	N=36	N=41
Proposed Method	avg.	0.991	**0.990**	**0.994**	**0.967**	**0.909**	**0.869**	**0.842**	**0.827**
	stdev.	0.019	0.004	0.001	0.005	0.007	0.007	0.006	0.006
SSRGA	avg.	**1**	0.968	0.931	0.893	0.866	0.848	0.833	0.823
	stdev.	0	0.004	0.006	0.006	0.005	0.007	0.004	0.005
AGA	avg.	**1**	0.96	0.922	0.888	0.865	0.847	0.836	0.826
	stdev.	0	0.007	0.008	0.007	0.006	0.005	0.004	0.004
SAGA	avg.	0.980	0.943	0.904	0.873	0.853	0.837	0.827	0.817
	stdev.	0.019	0.007	0.01	0.006	0.007	0.005	0.004	0.004
CGA	avg.	0.989	0.948	0.906	0.876	0.856	0.840	0.827	0.819
	stdev.	0.017	0.011	0.009	0.007	0.008	0.005	0.005	0.004

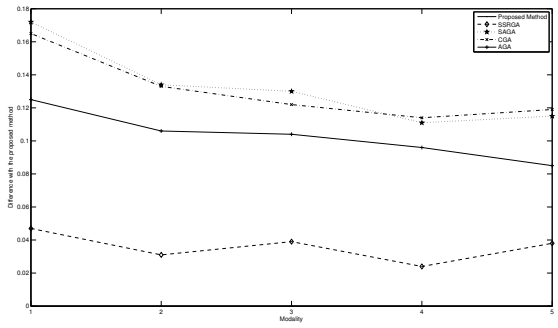

Figure 2: Difference of the results of the proposed method with compared methods over multi-modal problems.

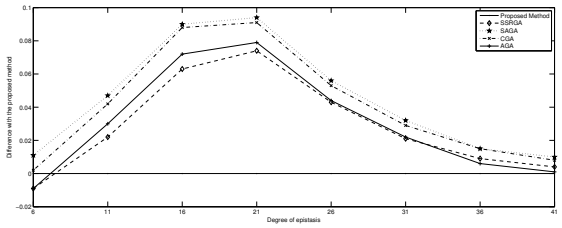

Figure 3: Difference of the results of the proposed method with compared methods over epistasis problems.

6. CONCLUSIONS

A new GA proposed on which the mixing and mutating of the individuals was done with regard to a probability for each loci. A probability vector holds the probability of occurrence for each allele on each loci. The probability on each loci changes with regards to the effect of that loci on the performance of a given individual. The probability of an allele is related to the changes in fitness.Experiments conducted on a wide range of MAX-SAT problems with different levels of multi-modality and epistatsis have shown that the performance of the proposed algorithm is comparable to other adaptive and self-adaptive methods. The proposed method is found to be superior over the compared algorithms.

7. ACKNOWLEDGMENTS

The first author thanks IPS, Universiti Sains Malaysia for supporting this research under the graduate fellowship programme. The first author would also like to thanks Dr. Fatemeh Vafaee and Prof. György Turán for their valuable supports.

References

[1] J. Arabas, Z. Michalewicz, and J. Mulawka. Gavaps-a genetic algorithm with varying population size. In *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*, volume 1, pages 73–78, USA, 1994.

[2] T. Bäck, A. E. Eiben, and N. A. L. van der Vaart. An empirical study on gas "without parameters". In *PPSN*, pages 315–324, 2000.

[3] T. Bäck and M. Schutz. Intelligent mutation rate control in canonical genetic algorithms. In *Foundations of Intelligent Systems*, pages 158–167. 1996.

[4] S. Baluja. Population based incremental learning: A method for integrating genetic search based function optimization and competitve learning, 1994.

[5] F. H. d. Brito, A. N. Teixeira, O. N. Teixeira, and R. C. L. Oliveira. A fuzzy approach to control genetic algorithm parameters. *SADIO Electronic Journal of Informatics and Operations Research*, 1(1):12–23, 2007.

[6] S. A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the third annual ACM symposium on Theory of computing*, STOC '71, pages 151–158, New York, NY, USA, 1971. ACM.

[7] W. de Landgraaf, A. Eiben, and V. Nannen. Parameter Calibration using Meta-Algorithms. In *IEEE Congress on Evolutionary Computation*, pages 71–78. IEEE, 2007.

[8] A. Eiben, M. Schut, and A. de Wilde. Boosting genetic algorithms with (self-) adaptive selection. In *Proceedings of the IEEE Conference on Evolutionary Computation (CEC 2006)*, pages 1584–1589. IEEE, 2006.

[9] A. Eiben and J. Smith. *Introduction to Evolutionary Computing*. Natural Computing Series. Springer, 2 edition, 2007.

[10] G. Eiben and M. C. Schut. New ways to calibrate evolutionary algorithms. In *Advances in Metaheuristics for Hard Optimization*, pages 153–177. 2008.

[11] M. R. Garey and D. S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[12] J. Grefenstette. Optimization of control parameters for genetic algorithms. *Systems, Man and Cybernetics, IEEE Transactions on*, 16(1):7, 1986.

[13] J. Gu. Local search for satisfiability (sat) problem. *Systems, Man and Cybernetics, IEEE Transactions on*, 23(4):1108 –1129, 1993.

[14] G. Harik. Linkage learning via probabilistic modeling in the ecga. Technical report, Illinois Genetic Algorithms Laboratory, 1999.

[15] G. Harik, F. Lobo, and D. Goldberg. The compact genetic algorithm. In *Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence., The 1998 IEEE International Conference on*, pages 523 –528, 4-9 1998.

[16] G. R. Harik and F. G. Lobo. A parameter-less genetic algorithm. In I. W. B. e. al., editor, *Proceedings of the Genetic and Evolutionary Computation Conference GECCO-99*, pages 258–265, San Francisco, CA, Morgan Kaufmann, 1999.

[17] W. E. Hart and R. K. Belew. Optimising an arbitrary function is hard for the genetic algorithm. In R. K. Belew and L. B. Booker, editors, *ICGA*, pages 190–195. Morgan Kaufmann, 1991.

[18] C. Ho, K. Lee, and K. Leung. A genetic algorithm based on mutation and crossover with adaptive probabilities. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 1, pages –775 Vol. 1, 1999.

[19] J. Holland. *Adaptation in Natural and Artifical Systems*. University of Michigan Press, 1975.

[20] K. A. D. Jong. Evolutionary computation. a unified approach. 2006.

[21] K. A. D. Jong and W. M. Spears. Using genetic algorithms to solve np-complete problems. In J. D. Schaffer, editor, *ICGA*, pages 124–132. Morgan Kaufmann, 1989.

[22] F. G. Lobo and D. E. Goldberg. The parameter-less genetic algorithm in practice. *Information Sciences*, 167(1-4):217–232, 2004.

[23] F. M. P. d. G. Lobo. *The Parameter-Less Genetic Algorithm: Rational and Automated Parameter Selection for Simplified Genetic Algorithm Operation*. PhD thesis, 1999.

[24] E. C.-P. Martin Pelikan, David E. Goldberg. Boa: The bayesian optimization algorithm. In *in Proceedings of Genetic and Evolutionary Computation Conference*, pages 525–532. Morgan Kaufmann, 1999.

[25] S. McClintock, T. Lunney, and A. Hashim. A fuzzy logic controlled genetic algorithm environment. In *Systems, Man, and Cybernetics, 1997. 'Computational Cybernetics and Simulation'., 1997 IEEE International Conference on*, volume 3, pages 2181–2186 vol.3, 1997.

[26] H. Mühlenbein, T. Mahnig, and A. O. Rodriguez. Schemata, distributions and graphical models in evolutionary optimization. *Journal of Heuristics*, 5:215–247, 1999.

[27] Z. Michalewicz. Gas: Selected topics. In *Genetic Algorithms + Data Structures = Evolution Programs*, page 387. Springer, 1998.

[28] V. Nannen, S. K. Smit, and A. E. Eiben. Costs and benefits of tuning parameters of evolutionary algorithms. In *Proceedings of the 10th international conference on Parallel Problem Solving from Nature*, pages 528–538. Springer-Verlag, Berlin, Heidelberg, 2008.

[29] M. Pelikan, D. E. Goldberg, and F. Lobo. A survey of optimization by building and using probabilistic models. IlliGAL Report No. 99018, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL, 1999.

[30] R. Salomon. The deterministic genetic algorithm: implementation details and some results. In *Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on*, volume 1, pages –702 Vol. 1, 1999.

[31] K. Sastry and D. E. Goldberg. On extended compact genetic algorithm. Technical report, GECCO-2000, late breaking papers, Genetic And Evolutionary Computation Conference, 2000.

[32] S. K. Smit and A. E. Eiben. Comparing Parameter Tuning Methods for Evolutionary Algorithms. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 399–406, May 2009.

[33] G. Smith. Adaptive genetic algorithms and the boolean satisfiability problem. Technical report, University of Pittsburgh, Pittsburgh, PA, 1979.

[34] J. Smith and T. C. Fogarty. Adaptively parameterised evolutionary systems: Self-adaptive recombination and mutation in a genetic algorithm. In *PPSN*, pages 441–450, 1996.

[35] W. M. Spears. *Evolutionary Algorithms: The Role of Mutation and Recombination (Natural Computing Series)*. Springer, June 2000.

[36] M. Srinivas and L. M. Patnaik. Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(4):656–667, 1994.

[37] F. Vafaee and P. C. Nelson. An explorative and exploitative mutation scheme. In *IEEE Congress on Evolutionary Computation*, pages 1–8. IEEE, 2010.

[38] S. Yang and c. Uyar. Adaptive mutation with fitness and allele distribution correlation for genetic algorithms. In *Proceedings of the 2006 ACM symposium on Applied computing*, SAC '06, pages 940–944, New York, NY, USA, 2006. ACM.