INT J COMPUT COMMUN, ISSN 1841-9836
9(3):325-339, June, 2014.

An Adaptive GA in Partitioned Search Space

F. Nadi, A.T. Khader

Farhad Nadi*
School of computer schiences
Universiti Sains Malaysia, 11800 Penang, Malaysia
*Corresponding author: nadi@cs.usm.my

Ahamad Tajudin Khader
School of computer schiences
Universiti Sains Malaysia, 11800 Penang, Malaysia
E-mail: tajudin@cs.usm.my

Abstract: Evolutionary algorithms are population based meta-heuristics inspired from natural survival of fittest phenomena. Despite their reasonable performance, these algorithms suffer from some weaknesses including the need for finding the values of their parameters that affect their performance. A new algorithm is proposed that divide the search space into equal sized partitions. Each partition is assigned with two parameters that determine the intensification and diversification rates. The partitions will be intensified or diversified adaptively with regards to the corresponding parameters. Traditional crossover and mutation operators are replaced with two new parameter-free operators. The experiments conducted on a wide range of multi-modal and epistatic problems showed the superiority of the proposed method in comparison to other algorithms in literature.
Keywords: Genetic Algorithms,Adaptive Parameter Control, Crossover Rate, Mutation Rate

1 Introduction

Evolutionary algorithm (EA) is a search meta-heuristics that improvise the quality of a set of candidate solutions iteratively using variation and selection operators [8, 13, 27].

Two major issues in the design of global search methods are intensification and diversification [3]. The ability to visit different regions of the search space is generally referred to as diversification. The ability to obtain high quality solutions within those locations is generally referred to as intensification [3]. The balance between these two and the way it is conducted is the main factor that differentiate these algorithms from each other [19]. EAs consist of different components (operators), each with different role for intensification or diversification or both. Usually, each operator in EAs come with some parameters that could be used to change the role of that operator, and consequently the so called balance in the search. This way, the parameters are found to have important effects on performance of the algorithms [9, 24, 27].

The literature on parameter adjusting could be divided into two main approaches [23] as follows. The first approach is known as *parameter assignment*, where values for the parameters will be provided using different methodologies. Whereas, in the second approach, which will be referred to as *parameter-reduced*, removal of the parameters is of interest.

Adaptive GA in a Partitioned Search Space (AGAPSS) is a new EA that assign the parameters to portions of the search space. It divides the search space into a predefined number of regions and tries to search the regions with carefully designed intensification and diversification operators. Two parameters are assigned to each partition of the search space holding the intensification and diversification rates. The probabilities for intensification or diversification of each

region will be determined from the behaviour of the regions. The region's behaviour is a function of overall fitness of the region and the number of visits from that region. The population will be distributed among the regions depending on their behaviour. The designed operators for intensification and diversification are working based on the probability vectors that holds the probability of the alleles on each loci for each individual. The probability vectors will be updated based on the changes in fitness of an individual. The AGAPSS iteratively evolves until it reaches a stopping criteria. Algorithm 1 shows a pseudo code of the proposed method.

Algorithm 1 Pseudo code for the main loop of the proposed method.

```
1: while stopping criteria is not met do
2:   for i = 0 to m   //m is the number of regions in the search space See § 3.2 do
3:     if U(0,1) < G_i.μ_D then
4:       //U(0,1) returns a random generated number between [0,1] based on the uniform distribution
5:       Diversify(I_1, I_2)   I_1, I_2 ∈ G_i
6:     end if
7:     if U(0,1) < G_i.μ_I then
8:       Intensify(I_1)   I_1 ∈ G_i
9:     end if
10:    Update related regions informations   //See sub-section 3.2
11:    Update: G_i.F, G_i.V, G_i.I_B, G_i.I_W   See § 3.2
12:    Update: G_i.μ_D.   See § 3.2
13:    Update: G_i.μ_I.   See § 3.2
14:  end for
15:  Adjust region population sizes. See § 3.2
16: end while
```

The remainder of this paper is organised as follows. Section 2 briefly looks into the background and related works. Explanation of the proposed method will be introduced in section 3. How the experiment has been conducted is mentioned in section 4. Experimental results will be reported in section 5. Finally, the last section, that is section 6, will be the conclusion and future works.

2 Background

This section will briefly review the literature on the research focusing on parameters.

2.1 Parameter Calibration Approaches

Categorizing approaches that are dealing with parameters could be done in different ways [8]. To this end, the literature could be divided into two main branches [23], which are parameter calibration methods and parameter-reduced methods.

In parameter calibration approaches, the parameters are actually within the algorithm and different methodologies are used for determining their values. Whereas in parameter-reduced approaches, the effort is towards removal of the parameters of the algorithms. The ideal form would have no parameters within the algorithm.

Parameter Control

This category covers all of the approaches on which the optimal values for the parameters will be provided by different methods. This covers those that try to find the optimal parameter

values prior to the run, or the methods that are trying to find the values of the parameters during the run.

REVAC [6], tries to find the parameters of a given EA by refining the possible parameter vectors iteratively. Here, an Estimation of Distribution Algorithm (EDA) [25], is used for finding the best parameter set for a given EA.

In an adaptive GA [20], a combination of static rules, inference engine of fuzzy logic controller and feedback from the algorithm are used for determining the values of the parameters.

Frequency of the best individuals within the population, number of duplicate individuals, and number of expected optimal values are used [4] for determining the parameter's values.

In a self-adaptive method [28], each individual is extended with a part that holds the mutation rate in itself. The global mutation rate will be calculated based on the individual mutation rate and a global value. Similarly, in self-Adaptive GA (SAGA) [1], every individual is extended with an extra bit (μ) holding the mutation rate. On each update, the new mutation rate (μ') will be derived based on the previous rate (μ).

Hybrid Self-adaptive GA (HSGA) [7], add an extra gene to the end of the chromosomes, representing the size of tournament. In this method, the less fit individuals get less selection pressure while fitter individuals get higher selection pressure.

Lobo in his Ph.D. thesis [18] addressed all of the parameters of the GA, either automatically (population size) or rationally (selection rate and crossover rate) based on theoretical foundations. An extension of this work [17] integrates local search with the previous work. It has been shown that the use of local search for exploitation of the search space is beneficial.

Parameter-reduced Approaches

This approach covers all of those methods where even one parameter is removed from the algorithm. Generally, in parameter-reduced GAs, the population and variation operators will be replaced with probabilistic model representation and generation models [2, 21, 26].

Compact GA (cGA) was proposed as a variation of GA where population is represented as a probability distribution over the set of solutions [12]. It processes each gene independently while it tries to create the same distribution like the previous population.

In a canonical GA referred to as SSRGA [32] crossover operator is removed from the algorithm, and mutation is replaced with another customised operator. In this method, firstly, the population will be divided into a set of sub-populations, secondly, a site-specific rate vector will be calculated from each sub-populations, and finally each sub-population will be mutated using its corresponding site-specific vector.

The extended version of the SSRGA, referred to as SSRGA-II [31], explores and exploits the search space simultaneously using a different strategy used in SSRGA. The algorithms will select top m individuals from the population and a position frequency matrix will be constructed based on them. Each component of this matrix refers to the probability of occurrence of allele i in site j. The probability for each site will be calculated proportionate to the fitness of the individuals.

3 Proposed Method

In the proposed method, the search space will be divided into regions. Each region is given two probability rates, one for diversification and the other one for intensification. In the beginning of the search, these probabilities will be initialised randomly. In other words, some of the regions will have more chance for either diversification or intensification. However, these probabilities will be updated as the search progresses. The probabilities of the regions change over time depending on the behaviour of the regions, which in turn will be determined by a rank given

to each region. The rank of each region is with regards to its overall fitness and the number of visits that have been done from that region. Population of each region will also change over time, though in the beginning of the search, the same number of individuals will be assigned to each region. The rank of the region will also be used to determine the eligibility of a given region for attracting more individuals. Regions will be prioritised based on their eligibility and individuals will be assigned to regions proportional to their eligibility. Each individual is assigned with a counterpart probability vector, which holds the probability of having allele 1 for each loci of a given individual. The occurrence probabilities of the alleles will get updated depending on the changes in the fitness and the number of changes in each individual.

The AGAPSS is mainly inspired from PGA [23] however, these two methods differ in some aspects including the variation operators. Besides that, the AGAPSS is an adaptive method due to utilization of parameter for intensification and diversification.

The remaining of this section is organized as follows. First, a formal definition of the concepts that have been used in this research will be introduced in section 3.1. Details on dividing the search space and management of the divisions will be explained in section 3.2.

3.1 Terminology

The formal definitions of the concepts used in the algorithm will be introduce in this section. For an individual (I),

$$I = i_1, ..., i_l \quad | \quad i_k \in A, 1 \leq k \leq l,$$

where l is the length of individual, and $A = \{0, 1\}$ is the alphabet of each locus.

There would be a probability vector (Z) for every individual,

$$\forall \ I_i \quad \exists \quad Z_i \ , \quad 1 \leq i \leq N,$$

where N is the population size, and

$$Z = z_1, ..., z_l \quad | \quad z_j \in [0, 1], 1 \leq j \leq l,$$

where z_j holds the probability of 1 in j^{th} locus of individual (I).

Depending on the number of changes (τ) that have occurred on a given individual, its respective probability vector (Z) will get updated [23] using,

$$\acute{Z} = Z + R.$$

Where $R \subseteq Z$ is defined as in Eq. 1 and $\acute{Z}$ is the updated probability vector. Initially all of the Z vectors will be initialized, in all of their loci, with 0.5. Each locus in Z vector will be bounded between 0 and 1 This way, equal chance will be given to each loci for being either 0 or 1. As the search progresses, the Z vectors will get updated based on Eq. 1, which is with regards to Table 1. According to Eq. 1 only those loci in Z vector will be updated that their corresponding loci in individuals have been changed. The locations of the changed loci will be determined using another vector, U, (see eq. 2) which will be constructed by applying logical *exclusive OR* operator on a given individual before and after changes.

In Eq. 1, $\epsilon = 1$ is the learning factor, $f : I \rightarrow \mathbb{R}$ is assumed to be the objective function of a maximization problem,

$$\tau = \sum_{i=1}^{l} u_i, \quad U = u_1, ..., u_l | u_j = i_j \oplus \acute{i_j}, 1 \leq j \leq l \tag{2}$$

where $\oplus$ is bitwise *exclusive OR* operator.

$$R = r_1, ..., r_l | 1 \leq j \leq l, \quad r_j = \begin{cases} \frac{\epsilon}{\tau} \cdot u_j & (f(I) > f(\acute{I}) \wedge i_j = 0) \vee (f(I) < f(\acute{I}) \wedge i_j = 1) \\ -\frac{\epsilon}{\tau} \cdot u_j & (f(I) < f(\acute{I}) \wedge i_j = 0) \vee (f(I) > f(\acute{I}) \wedge i_j = 1) \\ 0 & otherwise \end{cases} \tag{1}$$

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.

Change in locus		Fitness	Credit
From	To		
0	1	Improved	+
0	1	Worsen	-
1	0	Improved	-
1	0	Worsen	+

As an example, assuming the OneMax function as optimization problem on which the number of ones within the individual will be the fitness. For an individual $I = \{1, 1, 0, 0, 1, 0, 0, 1\}$, the fitness will be 4. Let assume this individual is changed to $I' = \{1, 1, \textit{1}, \textit{1}, \textit{0}, 0, 0, 1\}$, the new fitness will be 5. Here the fitness improved after the changes. Accordingly, $U = \{0, 0, 1, 1, 1, 0, 0, 0\}$ and $\tau = 3$. Consequently, $R = \{0, 0, 1/3, 1/3, -1/3, 0, 0, 0\}$. Therefore, if we assume before the update $Z = \{0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5, 0.5\}$, after the changes it will be updated to $Z' = \{0.5, 0.5, \textit{0.83}, \textit{0.83}, \textit{0.17}, 0.5, 0.5, 0.5\}$.

3.2 Dividing the Search Space

Initially, the search space will be divided into a predefined number (m) of regions (G) with the same size [16, 18, 22].

Representation

A set of loci in each individual will be reserved as indicators of regions [22]. It has to be noted that no extra genes will be added to the individuals. Each individual in the population will be member of a region, formally,

$$I_i \in G_j | 1 \leq i \leq N, 0 \leq j \leq m - 1. \tag{3}$$

where N is population size, and m is the number of regions in the population,

$$m = |A|^x, x \geq 1 \tag{4}$$

where x is the number of loci in the individual that is needed to be reserved for the region indicator, and $|A|$ is the cardinality of loci.

Binary representation of the region index (i), in the first x loci of an individual, will be used for indication of the members of the i^{th} region (G_i). This way the search space could be divided into equal sized partitions.

As an example, assume the case of dividing the search space into $m = 4$ regions. Referring to equation 4, the number of loci needed to be reserved is $\log_{|A|} m$ which in this case will be $x = 2$. Consequently, values of the first two loci of each individual, will be set to either of $00, 01, 10, or 11$,

that refers to the regions 0, 1, 2, and 3 respectively. A sample population of this example is shown in Figure 1. Here, three individuals will be assigned to each region assuming $N = 12$.

Region	I					f	F	V	δ	γ	P_I	P_D	r_I	r_D	χ	N
G0	**0**	**0**	1	1	0	2	1.67	3	$\overline{H}$	$\overline{H}$	3	1	4	3	1	2
	0	**0**	0	1	1	2										
	0	**0**	0	0	1	1										
G1	**0**	**1**	0	0	1	2	3	3	$\overline{H}$	$\overline{H}$	3	1	2	1	3	3
	0	**1**	1	0	1	3										
	0	**1**	1	1	1	4										
G2	**1**	**0**	0	1	1	3	2.33	3	$\overline{H}$	$\overline{H}$	3	1	3	2	2	3
	1	**0**	0	1	0	2										
	1	**0**	1	0	0	2										
G3	**1**	**1**	0	0	0	2	2.67	3	$\overline{H}$	H	1	2	1	4	4	4
	1	**1**	1	1	0	4										
	1	**1**	0	0	0	2										

Figure 1: The very first locus on each individual (highlighted in bold) will be used to indicate the regions. The onemax function is assumed as optimization problem in this example.

Region Parameters

As mentioned earlier, each region is assigned with two rates, one for intensification (μ_D) and the other for diversification (μ_I). In the beginning of the search, these probabilities will be set randomly for all of the regions. However, these rates change adaptively over time (iteration), depending on the ranks of the regions. The diversification rate will be adjusted using,

$$\mu'_D(i) = \mu_D(i) + 0.1(r_D(i) - r'_D(i)), \tag{5}$$

where, $r'_D(i)$ and $r_D(i)$ are diversification ranks of region G_i at previous and current iterations respectively. On the other side, changes in intensification rates will be based on,

$$\mu'_I(i) = \mu_I(i) + 0.1(r_I(i) - r'_I(i)), \tag{6}$$

where $r'_I(i)$ and $r_I(i)$ are intensification ranks of region G_i at previous and current iterations respectively. In both of the cases, the rates are bounded between 0.1 and 0.9 to prevent either neglect or over looking into a region.

For example, assuming that in the beginning of the search, the diversification rank ($r_D(1)$) for the first region is 3, and diversification rate ($\mu'_D(1)$) is 0.10. Also assume that in the subsequent generation, the diversification rank ($r_D(1)$) will be changed to 2. Based on Eq. (5) the diversification rate for the first region will change to 0.20.

Rank of the Regions

As mentioned, the regions' parameters will be updated based on the ranks of the regions. The rank of the regions will be determined with regards to the priority and the fitness of the regions.

The priorities of the regions will be determined based on the fitness of the regions and the number of visits that have been done from them. To this end, two behavioural statuses will be given to any given region, one for fitness and the other for the number of visits from a region.

If the fitness value of a given region (See Eq. (7)) was larger than mid-range value on that region, its status will be considered as high (H), otherwise it will be determined as not high ($\overline{H}$).

$$\gamma(i) = \begin{cases} H & G_i.F > \frac{f(G_i.I_B)+f(G_i.I_W)}{2} \\ \overline{H} & \text{otherwise.} \end{cases} \tag{7}$$

The number of visits from a given region (see Eq. 8), will be determined as high (H) if the mid-range value of the number of visits among all of the regions is lower than the number of the visit from the given region.

$$\delta(i) = \begin{cases} H & G_i.V > \frac{\underset{0\leq i\leq m-1}{\text{argmax}}\ \text{G}_\text{i}.\text{V} + \underset{0\leq i\leq m-1}{\text{argmin}}\ \text{G}_\text{i}.\text{V}}{2} \\ \overline{H} & \text{otherwise} \end{cases} . \tag{8}$$

As the first step, priorities of the regions will be retrieved using Eq. (11) or Eq. (12) for diversification or intensification respectively. A region with a low fitness and a relatively high number of visits would get the lowest priority (lowest value in Eq. (11)) for either diversification or intensification. It is because that the expectation of finding a good solution within that region is low. This way, lower attention will be on this kind of regions.

In the next step, ranks of the regions ($r_D(i)$ or $r_I(i)$) will be determined. A list will be constructed where the regions are sorted first based on their priorities and then their fitnesses. The row index of each region in the sorted list will be returned as the ranks for the regions. In other words,

$$\begin{aligned} r_D(i) > r_D(j) \Leftrightarrow (p_D(i) > p_D(j)) \\ \vee (p_D(i) = p_D(j) \wedge\ G_i.F > G_j.F) \\ \mid 0 \leq i, j \leq m-1. \end{aligned} \tag{9}$$

$$\begin{aligned} r_I(i) > r_I(j) \Leftrightarrow (p_I(i) > p_I(j)) \\ \vee (p_I(i) = p_I(j) \wedge\ G_i.F > G_j.F) \\ \mid 0 \leq i, j \leq m-1. \end{aligned} \tag{10}$$

where in both Eq. 9 and Eq. 10, $\vee$ is logical OR operator, and $\wedge$ is logical AND operator.

$$p_D(i) = \begin{cases} 2 & (\gamma(i) = H) \wedge (\delta(i) = H) \\ 3 & (\gamma(i) = H) \wedge (\delta(i) = \overline{H}) \\ 1 & (\gamma(i) = \overline{H}) \wedge (\delta(i) = H) \\ 4 & (\gamma(i) = \overline{H}) \wedge (\delta(i) = \overline{H}) \end{cases} \tag{11}$$

$$p_I(i) = \begin{cases} 3 & (\gamma(i) = H) \wedge (\delta(i) = H) \\ 4 & (\gamma(i) = H) \wedge (\delta(i) = \overline{H}) \\ 1 & (\gamma(i) = \overline{H}) \wedge (\delta(i) = H) \\ 2 & (\gamma(i) = \overline{H}) \wedge (\delta(i) = \overline{H}) \end{cases} \tag{12}$$

As another example, the overall fitness (F) of the regions of the example population, and the number of visits from each population are also reported in Figure 1 on page 330. Consequently, δ, and γ are calculated for all of the regions. Using the required information, the diversification and intensification priorities and also the ranks for the regions (according to Eq. (8) and Eq. (7)) are also calculated and shown in Figure 1.

Region Population

As mentioned earlier, each region will be instantiated with the same number of individuals. However, during the progress of the search, the number of individuals within each region varies. It has to be noted that the population size is fixed and it is the membership of individuals that changes.

The population of regions changes with proportion to the diversification rank of the regions. The lower is the diversification rank of a given region, the higher is the eligibility (χ) of that region for attracting new individuals. Formally,

$$\chi_i = m - r_D(i) + 1. \tag{13}$$

In each iteration of the run, the new population size of each region will be calculated using,

$$G_i.N = G_i.N_{min} + \left\lfloor \frac{\chi_i}{\sum_{1 \le i \le m} \chi_i} G_i.N_{max} \right\rfloor,$$

where $N_{max} = N - N_{min}(m-1)$.

The extra individuals of each region will be distributed among the other regions with proportionate to their eligibility.

The eligibility rank and population size of the regions for the sample population in the previous example are shown in Figure 1.

4 Experimental setup

Three different test functions are chosen from two different family of problems. Selection of the compared algorithms was based on their relevance to the proposed method. The test functions are the same as those that have been used by the chosen algorithms for comparison [1,23,30–32]. This facilitate, the comparison of the results with other methods in literature.

To come up with a fair experiment, the algorithms are tuned as it was reported in literature, which is reported in Table 2.

Table 2: Attributes of the comparing algorithms

Feature	epistatic	multimodal	MPG
Population model	steady state		
Parent selection	Tournament selection		
Survival selection	delete oldest		
Selection pressure	8	8	100
Population size	50	50	100
Max. no. of generations	500	100	10000
Chromosome length	based on the epistasis level	30	100
Number of regions (m)	8		
Region population size	8		

The performance of each algorithm is measured over 50 independent runs for each of the problems. The average (*avg.*) of the best results are derived for all of the experiments. However, standard deviation (*stdev.*) of the results are reported when the compared methods have reported them. Best average is highlighted in **bold**, while the lowest standard deviation is underlined in each case.

The following will be the explanation on the utilized test problems for the course of experiments.

4.1 MAX-SAT problems

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

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 maximises the number of satisfied clauses [5, 10, 11].

Multi-modal boolean satisfiability and epistatic problems are the two classes of benchmarks used in the comparisons. These problems will be constructed based on the MAX-SAT problems. 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 grow with the number of modality. While, 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 [29]. The degree of epistasis has direct relation with difficulty level of the problem.

The compared algorithms are canonical GA (CGA) [32] with a randomly chosen constant mutation rate, SSRGA [32], *self-adaptive (SAGA)* [1], *adaptive (AGA)* [30] parameter control methods, Compact GA (cGA) [12] and a proposed method, [23] which will be referred to as *PGACMCO*.

Epistatic problems

Spears [29] has introduced a method for the creation of epistatic problems using the boolean expressions. His proposed method could be tuned for different levels of epistasis. The method is based on conversion of the Hamiltonian Circuit (HC) problems into equivalent 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 constrains the nodes of feasible solutions to have only one input edge and one output edge. As such, any tour that does not satisfies this constraint cannot be a solution [14].

MAX-SAT Multi-modal problems

The multi-modal problems used for the course of experiment are created using a mechanism proposed by Spears [29].

A uni-modal problem of length 30 could be created as follows,

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

Using the above uni-modal problem, a bimodal problem will be,

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

For the course of this experiments, multi-modal problems up to 5 peaks have been used.

4.2 Multi-modal Problem Generator

Multi-modal Problem Generator (MPG) is another benchmark, proposed by Spears [15, 29]. The benchmark has the ability to create problems with multiple peaks where the number of peaks defines level of problem difficulty. For all of the peaks we have used, the heights are linearly distributed and lowest height is 0.5. Fitness value of a given individual will be calculated according to the Hamming Distance (HD) between the individual and the nearest peak [33].

The compared algorithms are Compact GA (cGA), [12] the PGA , [23] GASAT [9], GAHSAT [9], hand-tuned [9], meta-GA [6] and REVAC [6]. It has to be noted that the hand-tuned algorithm here is a canonical GA on which the values of the parameters are carefully tuned.

5 Results and discussion

The Mean of Best Fitness (MBF) [8] for all of the algorithms are reported. The MBFs of the MPG function reported in Table 3 show that the AGAPSS has obtained better performance in all of the instances of this function in comparison to the other compared methods.

Table 3: Comparison of the AGAPSS with compared methods over MPG benchmark in [6, 7]

Peaks	AGAPSS	GASAT	GAHSAT	GA	meta-GA	REVAC	cGA	PGA	SSRGA-II
1	1	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.959
2	1	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.958
5	1	1.0	1.0	1.0	0.988	1.0	1.0	1.0	0.962
10	**1**	0.9956	0.9939	0.9961	0.993	0.996	0.9989	0.9994	0.960
25	**0.9992**	0.9893	0.9879	0.9885	0.994	0.991	0.9863	0.9933	0.963
50	**0.9988**	0.9897	0.9891	0.9876	0.994	0.995	0.9794	0.9896	0.962
100	**0.9976**	0.9853	0.9847	0.9853	0.983	0.989	0.9847	0.9909	0.965
250	**0.9969**	0.9867	0.9850	0.9847	0.992	0.966	0.9739	0.9879	0.969
500	**0.9949**	0.9834	0.9876	0.9865	0.989	0.970	0.9673	0.9885	0.967
1000	**0.9935**	0.9838	0.9862	0.9891	0.987	0.985	0.9649	0.9884	0.959

The results of the MAX-SAT epistatic problem are reported in Table 4. The rustles are very comparable with the results of the cGA even though the AGAPSS has obtained higher MBFs in five instances of the epistasis problems.

Table 4: Comparison of the AGAPSS with benchmark methods [23, 32] over epistatic problem with different levels of epistasis

Deg. of epistasis		N=6	N=11	N=16	N=21	N=26	N=31	N=36	N=41
AGAPSS	avg.	0.986	0.986	0.982	0.965	**0.946**	**0.926**	**0.91**	**0.896**
	stdev.	0.014	0.003	0.005	0.008	0.011	0.0081	0.011	0.012
PGA	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-II	avg.	0.953	0.926	0.885	0.858	0.839	0.823	0.811	0.805
	stdev.	0.013	0.007	0.013	0.014	0.01	0.007	0.007	0.007
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
Compact GA	avg.	0.985	0.983	0.983	**0.979**	0.945	0.926	0.909	0.894
	stdev.	0.014	0.002	0.002	0.003	0.005	0.006	0.006	0.005

The results of the MAX-SAT multi-modal problem are reported in Table 5. In this benchmark the AGAPSS has performed better in four out of the six algorithms. However, the results of the AGAPSS and the PGA are comparable for all the instances. It has to be noted that this test function as like the MPG function is a multi-modal problem. However, the obtained results of the AGAPSS was far different from what that have been obtained over MPG function.

The statistical test reported in table 6 allow systematic comparison of the AGAPSS with the other compared methods. The proposed method is compared pairwise with other compared methods and the results of the statistical are reported for each pair.

The statistical tests confirms that the results of the AGAPSS have been significantly different from all of the other compared methods over MPG test function. According to the obtained ranks of the AGAPSS it could be inferred that the proposed methods has obtained significantly better results in comparison to the other compared methods over this test function.

Considering the obtained rankings and the p-values, the statistical tests confirms the superiority of the proposed method over epistasis test function in four out of the six compared

algorithms. Although the AGAPSS has performed better than the PGA and cGA, regarding to the obtained ranks (R^-), however the statistical difference between them was not significant.

The statistical results have confirmed significant difference in favour of the proposed method for five out of six compared algorithm over MAX-SAT multi modal problem. However, the proposed algorithm has performed significantly worse than the PGA method. It would be more interesting to note that two different results have been obtained for the same family of problems. Both of the MAX-SAT multi-modal and MPG functions are multi-modal functions. While the proposed method has performed significantly better than the PGA over MPG method, it has performed worst over MAT-SAT multi-modal problem.

The main difference between the MPG and MAX-SAT multi-modal function is that the MPG function is ran for 10000 fitness calls, while the MAX-SAT problem stopped after 100 fitness calls. It suggest that the AGAPSS might be able to outperform the PGA over higher number of iterations. In order to check the validity of the aforementioned hypothesis, a new experiment has been conducted. Both of the PGA and AGAPSS methods have been ran over MAX-SAT multi-modal problem while the maximum number of fitness calls was set to 10000. Figure 2 depicts the progress of improvement in fitness as the search progress over different instances of MAX-SAT multi-modal function.

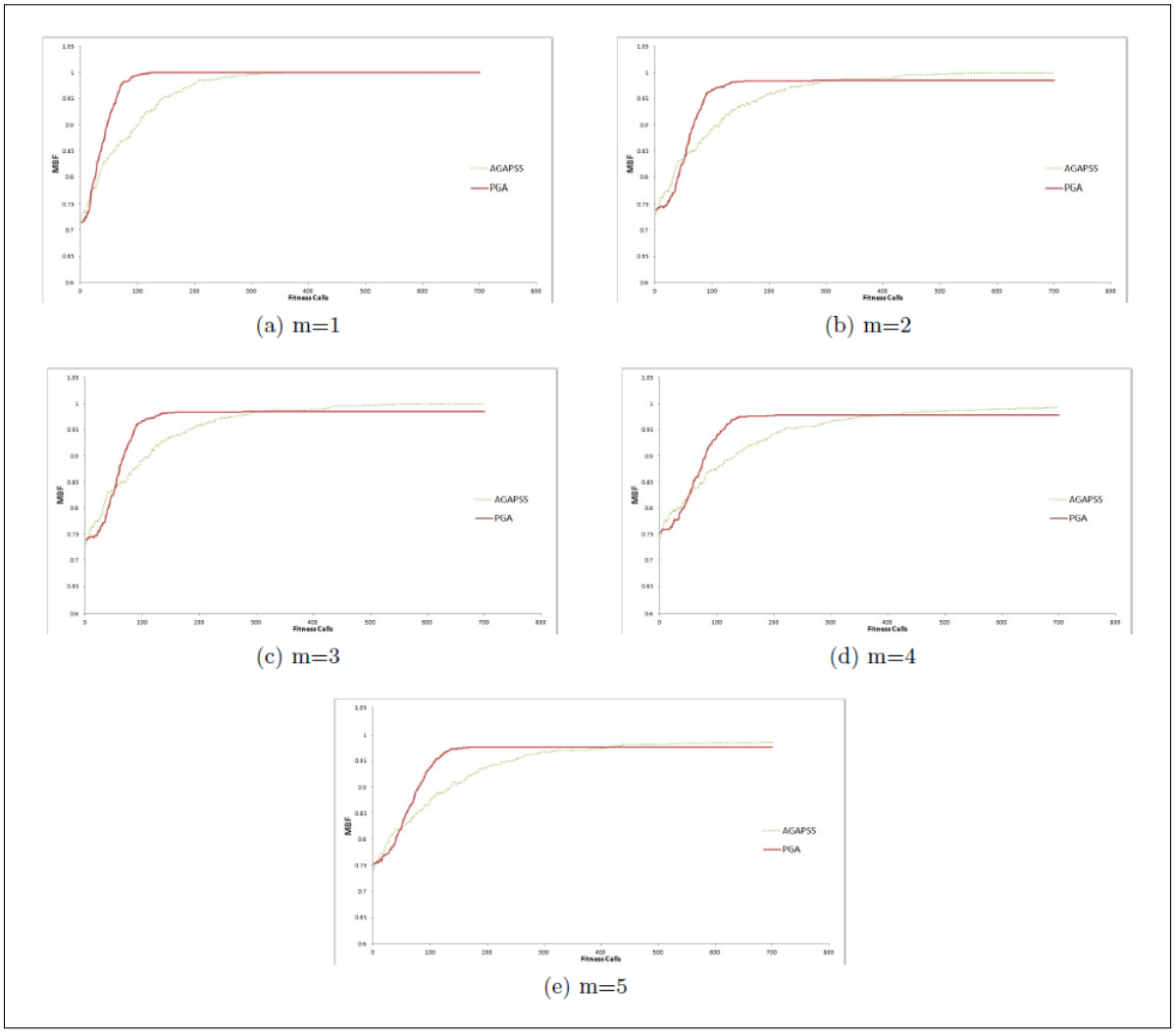

Figure 2: Comparison of the progress in the performance of the AGAPSS and PGA over the MAX-SAT multi-modal function in.

As it could be seen in all of the instances of the compared algorithms, the AGAPSS have been able to obtain better results in comparison with the PGA method. However, the AGAPSS has obtained the better performance after some iterations. In other words, the performance of the PGA method was better in the early stages of the run. The statistical analyses, on the results of the last four instances of the MAX-SAT multi-modal function, confirmed that the AGAPSS has performed significantly better than the PGA . The performance of the AGAPSS has also been significantly better than PGA in longer iterations over the epistasis problems.

As a result it has been shown that the assignment of the parameters to the parts of the search space could be beneficial. However, it take some generations, in the early stages of the run, for the AGAPSS to conduct the proper balance between the partitions of the search space.

In other words, it takes time for the proposed method to find the promising partition and focus on it. This time would slow down the algorithm from finding the promising area in the search space, however, the accuracy of the proposed method in comparison to the other methods has shown that the proposed method is able in finding significantly better solutions.

Table 5: The AGAPSS in comparison to some of the relevent methods from literature over multi-modal MAX-SAT problem with different levels of multi-modality.

Modality		$p = 1$	$p = 2$	$p = 3$	$p = 4$	$p = 5$
AGAPSS	avg.	0.936	0.9155	0.9015	0.8996	0.9117
	stdev.	0.023	0.033	0.031	0.031	0.037
PGA	avg.	**0.999**	**0.976**	**0.970**	**0.964**	**0.961**
	stdev.	0.007	0.023	0.025	0.029	0.026
SSRGA-II	avg.	0.871	0.835	0.842	0.835	0.846
	stdev.	0.030	0.040	0.034	0.034	0.029
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
cGA	avg.	0.8360	0.7642	0.7909	0.7729	0.7794
	stdev.	0.0386	0.0274	0.0347	0.0292	0.0280

6 Conclusions and future works

A new adaptive algorithm is proposed that divides the search space into predefined number of regions. Different search strategies will be applied on different regions based on ranks that are given to each region based on their behaviour. The behaviour of a regions will be determined in relation to the overall fitness of the regions and the number of visits that have been done from them.

The proposed method has been examined using different instances of three benchmark problems: MAX-SAT multi-modal problems, MAX-SAT epistatic problems, and MPG problems.

The performance of the proposed method shown to be superior over the MPG in comparison to all of the compared methods. However, the results are negotiable with the MAX-SAT multi-modal test function. The proposed method shown to be able to perform better than the PGA when the maximum number of fitness calls increased. Ignoring the first instance of this benchmark, the statistical analysis shown significant difference The results over the epistasis problem has also shown that the AGAPSS was significantly better in compare to the most of the compared methods. Overall, based on the results of the benchmarks, the performance of the proposed method is superior compared to the compared algorithms.

Table 6: The Wilcoxon signed-rank test between the AGAPSS and other algorithms. p-values below 0.05 are shown in bold.

AGAPSS vs.		multi-modal	epistatic	MPG
PGA	R^-	0	26	53.5
	R^+	15	10	1.5
	p-value	0.043*	0.263**	**0.012**
SSRGA-II	R^-	15	36	55
	R^+	0	0	0
	p-value	**0.043**	**0.012**	**0.005**
AGA	R^-	15	35	
	R^+	0	1	
	p-value	**0.043**	**0.017**	
SAGA	R^-	15	36	
	R^+	0	0	
	p-value	**0.043**	**0.012**	
Hand-tuned	R^-	15	35	52
	R^+	0	1	3
	p-value	**0.043**	**0.017**	**0.018**
Compact GA (cGA)	R^-	15	25	52
	R^+	0	11	3
	p-value	**0.043**	0.327**	**0.018**
GASAT	R^-			52
	R^+			3
	p-value			**0.018**
GAHSAT	R^-			52
	R^+			3
	p-value			**0.018**
meta-GA	R^-			53.5
	R^+			1.5
	p-value			**0.012**
REVAC	R^-			52
	R^+			3
	p-value			**0.018**

** : No significant difference between the AGAPSS and the compared algorithm.

* : Significant difference between the AGAPSS and the compared algorithm in favour of the compared algorithm.

Bibliography

[1] Bäck, T., Schutz, M. (1996). Intelligent mutation rate control in canonical genetic algorithms. *In Foundations of Intelligent Systems*, 158-167.

[2] Baluja, S. (1994). Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Science, *CMU-CS-94-(CMUCS- 94-163)*:1-41.

[3] Blum, C., Roli, A. (2003). Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Comput. Surv.*, 35(3):268-308.

[4] Houat de Brito,F., Noura Teixeira, A., Noura Teixeira, O., Oliveira R.C.L. (2007). A fuzzy approach to control genetic algorithm parameters. *SADIO Electronic Journal of Informatics and Operations Research*, 1(1):12-23.

[5] Cook, S.A. (1971). The complexity of theorem-proving procedures. *Proceedings of the third annual ACM symposium on Theory of computing, STOC 71*, 151-158.

[6] de Landgraaf, W.A., Eiben, A.E., Nannen, V. (2007). Parameter calibration using metaalgorithms. *In Evolutionary Computation*, 71-78.

[7] Eiben, A.E., Schut,M.C., de Wilde, A.R.(2006). Boosting genetic algorithms with (self-) adaptive selection. *Proceedings of the IEEE Conference on Evolutionary Computation (CEC 2006)*, 1584-1589.

[8] Eiben, A.E., Smith, J.E. (2007). *Introduction to Evolutionary Computing*. Natural Computing Series. Springer, 2 edition, 2007.

[9] Eiben, G., Martijn C. Schut, M.C. (2008). New ways to calibrate evolutionary algorithms. In *Patrick Siarry and Zbigniew Michalewicz, editors, Advances in Metaheuristics for Hard Optimization, Natural Computing Series*, 153-177.

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

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

[12] Harik, G.R., Lobo, F.G., Goldberg, D.E. (1999). The compact genetic algorithm. *Evolutionary Computation, IEEE Transactions on*, 3(4):287-297.

[13] De Jong, K.E. (2006). *Evolutionary Computation. A Unified Approach*. MIT Press, 2006.

[14] De Jong, K.E., Spears, W.M. (1989). *Using genetic algorithms to solve np-complete problems*. In J. David Schaffer, editor, ICGA, 124-132. Morgan Kaufmann, 1989.

[15] De Jong, K.E., Spears, W.M. (1992). *A formal analysis of the role of multi-point crossover in genetic algorithms*, 1992.

[16] Kuo, T. Hwang, S.-Y. (1996). Why DGAs work well on ga-hard functions? *New Gen. Comput.*, 14:459-479.

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

[18] Lobo F.G., (2000). *The Parameter-Less Genetic Algorithm: Rational and Automated Parameter Selection for Simplified Genetic Algorithm Operation*. PhD thesis, 2000.

[19] Lozano, M., Garcia-Martinez, C, (2010). Hybrid metaheuristics with evolutionary algorithms specializing in intensification and diversification: Overview and progress report. *Comput. Oper. Res.*, 37:481-497, March 2010.

[20] McClintock,S., Lunney, T., Hashim, A. (1997). A fuzzy logic controlled genetic algorithm environment. *Systems, Man, and Cybernetics*, 3:2181-2186.

[21] Muhlenbein, H., Mahnig, T., Ochoa Rodriguez, A. (1999). Schemata, distributions and graphical models in evolutionary optimization. *Journal of Heuristics*, 5:215-247.

[22] Nadi, F., Khader, A.T. (2010). Managing search in a partitioned search space in GA. *Cybernetics and Intelligent Systems (CIS), 2010 IEEE Conference on*, 114-119.

[23] Nadi, F., Tajudin, A., Khader, A.T. (2011). A parameter-less genetic algorithm with customized crossover and mutation operators. In *Natalio Krasnogor and Pier Luca Lanzi, editors, ACM, GECCO*, 901-908.

[24] Nannen, V., Smit,S.K., Eiben, A.E. (2008). Costs and benefits of tuning parameters of evolutionary algorithms. *Proceedings of the 10th international conference on Parallel Problem Solving from Nature*, 528-538.

[25] Pelikan, M., Goldberg,D.E., Lobo F. (1999). 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.

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

[27] Smit, S.K., Eiben, A.E. (2009). Comparing Parameter Tuning Methods for Evolutionary Algorithms. *IEEE Congress on Evolutionary Computation (CEC)*, 399-406.

[28] Smith, J., Fogarty, T.C. (1996). Adaptively parameterised evolutionary systems: Selfadaptive recombination and mutation in a genetic algorithm. *of Lecture Notes in Computer Science*, 1141:441-450.

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

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

[31] Vafaee, F. (2011). *Controlling Genetic Operator Rates in Evolutionary Algorithms*. PhD thesis, Dept. Computer Science, 2011.

[32] Vafaee, F., Nelson P.C. (2010). An explorative and exploitative mutation scheme. *IEEE Congress on Evolutionary Computation*, 1-8.

[33] Vajda, P., Eiben, A.E., Hordijk, W. (2008). Parameter control methods for selection operators in genetic algorithms. *Proceedings of the 10th international conference on Parallel Problem Solving from Nature*620-630.