An Explorative and Exploitative Mutation Scheme

Fatemeh Vafaee and Peter C. Nelson

***Abstract*— Exploration and exploitation are the two cornerstones which characterize Evolutionary Algorithms (EAs) capabilities. Maintaining the reciprocal balance of the explorative and exploitative power is the key to the success of EA applications. Accordingly, in this work the canonical Genetic Algorithm is augmented by a new mutation scheme that is capable of exploring the unseen regions of the search space, and simultaneously exploiting the already-found promising elements. The proposed mutation operator specifies different mutation rates for different sites (loci) of the individuals. These site-specific rates are wisely derived based on the fitness and structure of the population individuals. In order to retain the balance of the required exploration and exploitation, the mutation rates are adapted during the evolution. To demonstrate the efficacy of the proposed algorithm, the method is evaluated using a set of benchmark problems and the outcome is compared with a series of well-known relevant algorithms. The results demonstrate that the newly suggested method significantly outperforms its rivals.**

I. Introduction

Evolutionary Algorithms belongs to the family of stochastic generate-and-test search algorithms [1] which achieves their power from the two sources: *exploration* of the search space and *exploitation* of the already found promising information [2]. Exploration is the ability of a search algorithm to discover unseen regions of the search space, and to avoid convergence to local optima. On the other hand, exploitation is defined as "*the ability of an algorithm to step into the direction of the desired improvement*" [3] or the capability of the "*good use of good information*" [2]. A search algorithm with well-adjusted exploitative power can improve the possibility of generating "better" solutions from already found "good" ones. In Genetic Algorithms (GAs), genetic operators—mutation and crossover—are known to be major sources which are supposed to introduce the required explorative and exploitative power into GAs and retain their reciprocal balances.

Traditionally, there has been an assumption that the exploitative power of GAs results from the crossover operator (beside the selection pressure) [4, 5]. The intuition behind such a supposition rests on the building-block hypothesis [4]. According to this hypothesis, highly fit individuals are formed from basic building blocks (schemata), and by way of crossover, fitter individuals may be constructed through mixing and matching building blocks. The following sentences from [5] summarizes the former common assumption in GA community on the role of the crossover operator:

"*the population contains not just a sample of n ideas, rather it contains a multitude of notions and rankings of those notions for task performance. Genetic Algorithms ruthlessly exploit this wealth of information by 1) reproducing high quality notions according to their performance and 2) crossing these notions with many other high-performance notions from other strings*."

However, new evidence has cast doubt on the validity of the schemata theorem [6, 7, 8], and the crossover viability in exploiting information rests on available schemas [9, 10, 11]. For instance, in [9] the results of investigating the usefulness of crossover on NK-landscapes is expressed as: "*recombination is useless on uncorrelated landscapes, but useful when high peaks are near one another and hence carry mutual information about their joint locations in the fitness landscape*." As an example of a more strong critique on crossover efficacy, it is argued by [11] that crossover does not have any competitive advantage over mutation.

Nevertheless, exploiting the available pieces of information (schemata) is necessary to GA performance, regardless of whether crossover can tackle it or not. Accordingly, in this work we put aside the crossover operator, and propose a new mutation scheme with both explorative and exploitative capabilities. The proposed mutation operator assigns specific rates to different sites (loci) of individuals in the population. The rates are derived in such a way that good individuals are exploited (by wisely varying a few number of alleles appearing at some particular sites), while bad individuals are freely disrupted to allow a vast exploration of the search space.

Another widespread opinion is that exploration and exploitation are opposite forces. Therefore, the success of a search algorithm resides in a well-found balance of the two [12, 13, 14]. It is well understood that excessive exploratory power usually leads to the discovery of the global optimum, but critically slows down the convergence rate. Conversely, the excessiveness of the exploitation rapidly results in the convergence to a local optima. In order to retain the reciprocal balance of exploration and exploitation we *adaptively* control the mutation rates throughout the evolution. In this regard, our proposed method can be suited within the parameter control Evolutionary Algorithms in which the parameters are subject to change *during* the algorithm run (as opposed to parameter tuning methods wherein the parameter values are set *before* a run).

Parameter control techniques can be divided into two major categories (referring to the taxonomy introduced by [15]): *self-adaptive* and *adaptive* approaches. In the self-adaptive paradigm, the parameters are encoded into the representation of the individual solutions and undergo mutation and

Fatemeh Vafaee and Peter C. Nelson are with the Artificial Intelligence Laboratory, the Department of Computer Sciences, University of Illinois at Chicago, Chicago, Illinois 60607, USA (phone: +1 (312) 413-4075; emails: {fvafae2, nelson}@uic.edu).

recombination. The intuition is that the parameter values (e.g., the genetic operator rates) have a chance to co-evolve with problem solutions to hopefully (near)-optimal settings (see [16], [17], [18], and [19] as a few examples).

On the other hand, in an adaptive scheme, the direction and/or magnitude of the parameters are determined based on some form of feedback (e.g., the quality of solutions) from the search. Among different parameters, adapting operator probabilities, specifically the mutation rate, has long been recognized as a commendable path toward improving the performance of Evolutionary Algorithms (see for example [20], [21], [22], and [23]).

The mutation operator proposed in this paper fits into the category of adaptive parameter control Evolutionary Algorithms. Accordingly, we selected two mature and frequently-cited operator rate control methods, one adaptive and the other self-adaptive, as benchmark algorithms to evaluate the performance of our suggested method. As the experimental results demonstrate, our proposed algorithm shows a significant improvement in the quality of the best found solutions compared to the benchmark algorithms.

In summary, our proposed mutation scheme has the following distinct properties:
1) It specifies different mutation rates for different sites of the encoded solutions.
2) These site-specific rates are adapted generation-by-generation to adjust to the current state of the evolution.
3) The mutation operator explores unseen regions of the search space, and at the same time, exploits the promising elements discovered so far. In other words, it adjusts the tradeoff between the required exploration and exploitation power.
4) Instead of *randomly* assigning alleles to loci which are subject to mutation, our proposed mutation operator decides on a particular type of gene for each specific site. Thus, in binary-codded GAs, rather than flipping alleles which undergo mutation, the proposed scheme decides whether allele '1' or '0' should be chosen for a particular site, regardless of the allele which already exists in that site.
5) The method suggested in the current work is particularly designed for multimodal search spaces. However, it has the potential to get even extended to more challenging search environments (Section IV discusses the direction of possible improvements).

It is worth mentioning that the idea of attributing specific mutation rates to different sites has a root in biological models of evolution. The concept of "among-site rate variation" [24] which indicates that the rate of nucleotide substitution varies among different sites of a DNA strand has long been introduced in phylogenetic and molecular evolutionary studies. Several among-site rate variation models (e.g., gamma-distributed [25], and site-specific rates (SSR) models [26]) have emerged to properly describe the evolutionary behavior of nucleotide substitutions. Although the method proposed in this paper has not been directly inspired by such evolutionary models, we refer to GA augmented by our proposed mutation scheme as Site-specific Rate GA (SSRGA) since the idea of deriving mutation rates at single gene level came to us once we encountered the SSR model of evolution.

The remainder of this paper is organized as follows: Section II provides a detailed description of the SSRGA algorithms and the underlying ideas.The set of benchmark problems, the compared algorithms, and the experimental results are presented in Section III. Finally, Section IV gives a summary and a brief overview of the direction of future work.

II. Site-Specific Rate Genetic Algorithm

Exploitation in GAs can be achieved by instantiating "good" schemas. Intuitively, "good" schemas are those with sufficiently high fitness values (the fitness of a schema is usually defined to be the mean fitness of individuals matching the schema [5]). However, how can such schemata be found based on the available population of the individuals?

In order to derive some (seemingly) good schemas from the existing population, we introduce the process of "*schema-matching*." We informally explain this procedure by providing a simple example. Suppose that two highly-fit individuals $x = 001101$, and $y = 011111$ (of length $l = 6$) are (in some way!) drawn out of the population. A matched schema is then made based upon the commonness of alleles per site. For instance, since both individuals possess allele '0' at the first locus, the first position of the matched schema holds character '0'; but, as the second loci of the two individuals contains opposing allele types, the wild card (or unknown) character '*' would be chosen for this position of the schema. We then hypothesize that the schema $0 * 11 * 1$ can be considered as a good schema based on the information on hand.

Two major problems, however, arise here: 1) as the number of individuals in the population increases, the structure of the identified schema tends towards an *all-*-string*, which conveys no useful information, 2) all chosen individuals have a similar effect on defining the corresponding schema. Intuitively, better individuals should play a more important role in the procedure[1].

To resolve these obstacles, we introduce the refined version of the procedure referred to as "*probabilistic*-schema-matching." Instead of deterministically attributing a particular schema-character (i.e., '1', '0', and '*') to each schema site, probabilistic-schema-matching derives a site-specific probability $p_j(a)$ representing the possibility of possessing allele $a \in \{\text{'0'}, \text{'1'}\}$ at site j. In brief, the procedure accepts a set of selected individuals, and returns their corresponding probability vector of length l, called Site-Specific Rate (SSR) vector in the text. We discuss later how SSR vector is estimated and whether it can resolve the above-mentioned problems.

It is worth mentioning that the probabilistic-schema-matching should intuitively equate the schema-matching in some particular cases: assuming that all chosen individuals have an identical fitness value, if the estimated probability

[1]The intuition holds in *non-deceptive* landscapes which are the types of environments under study in this work.

Algorithm: *GA with Site-Specific Rates* (*SSRGA*)
Given: String length l, population size $n \geq 1$, and fitness function $f()$

1. $t \leftarrow 0$
2. $s_0 \leftarrow$INITIAL-POPULATION(n)
3. EVALUATE(s_0)
4. **repeat**
5. $\quad \mathcal{P}_t \leftarrow$PARTITIONING$(s_t)$
6. $\quad$ **for** $i \leftarrow 1$ **to** $|\mathcal{P}_t|$ **do**
7. $\quad\quad \mathbf{P}_t^i \leftarrow$COMPUTE-SSR$(s_t^i)$
8. $\quad\quad$ MUTATION$(s_t^i, \mathbf{P}_t^i)$
9. $\quad$ SELECT-EVALUATE(s_t)
10. $\quad t \leftarrow t+1$
11. **until** stop condition not met

Fig. 1. SSRGA algorithm

$p_j(a = \text{`1'})$, in probabilistic-schema-matching, would be either 1.0, 0.0, or 0.5, that should be equivalent with assigning characters '1', '0', or '*', respectively, to site j using the schema-matching approach.

Now, a valid question is which individuals are selected for the procedure and how? At the first look, highly fit individuals may seem to be proper candidates. However, by a deeper consideration, we realize that not every good individual can be picked for deriving a particular SSR vector. More precisely, in multimodal environments equally-fit individuals may be under different optima. If they all selected for a single SSR vector, the resulting probabilities may carry no useful information or can be even misleading. For instance, assume that two individuals are picked; one with allele '1' and the other with allele '0' at all loci. Furthermore, suppose that they both have an identical fitness value which can imply that they are under disparate peaks.[2] Then, all elements of the estimated SSR vector would be 0.5, which degenerates our method to the pure random search.

Therefore, to correctly estimate an SSR vector, we need to select individuals which are the *basins of attraction*[3] of a single optimum. To this end, we propose a partitioning method which divides the population into several subpopulations. Each part should ideally contain only the individuals under a particular peak. Clearly, the number of subpopulations should be determined on-the-fly based on the fitness and the structure of the individuals contained in the population.

By keeping these introductory ideas in mind, in the remainder of this section we first thoroughly explain the proposed SSRGA algorithm, and then specifically focus on the employed partitioning method.

A. SSRGA Algorithm

SSRGA is the canonical (binary-coded) GA augmented by a new mutation scheme which is designed to cope with both explanatory and exploitatory responsibilities of genetic operators. Figure 1 shows the proposed SSRGA algorithm. The overall theme of the algorithm is to perform the following tasks at the onset of each generation:

- Partition the current population into a set of "suitable" subpopulations: PARTITIONING function (line 5)
- For each subpopulation, compute the Site-Specific Rate vector: COMPUTE-SSR function (line 7)
- Mutate each subpopulation using the corresponding SSR vector: MUTATION function (line 8)

These three functions will be precisely described in here. The rest of the lines of the SSRGA algorithm depicted in Figure 1, are all identical to the canonical genetic algorithm, and thus will not be reconsidered anymore.

1) PARTITIONING(s_t)*:* This function splits the current population s_t into a set of distinct (possibly one) part(s). The number of parts/subpopulations is determined based on the fitness and the structure of individuals contained in the population. Let us denote each subpopulation as s_t^i. Then, assuming that m is the total number of recognized parts, the returned set of subpopulations is formally defined as $\mathcal{P} = \{s_t^1, \cdots, s_t^m\}$ such that:

$$\forall s_t^i, s_t^j \wedge (i \neq j) \Rightarrow s_t^i \cap s_t^j = \emptyset, \text{ and } \bigcup_{i=1}^{m} s_t^i = s_t.$$

As mentioned earlier, each subpopulation should (ideally) comprise the individuals which are the basins of attraction of a single optimum. The partitioning method is discussed in detail below in subsection B.

2) COMPUTE-SSR(s_t^i)*:* This function gets a subpopulation s_t^i as input parameter and returns the corresponding Site-Specific Rate vector denoted as $\mathbf{P}_t^i(a) = \langle p_0^i(a), \cdots p_l^i(a) \rangle$ (subscript t is eliminated from the vector elements to simplify the notation) where l is the string length and $p_j^i(a)$ is the probability of acquiring allele $a \in \{\text{`0'}, \text{`1'}\}$ at site j of all individuals contained in subpopulation s_t^i. To avoid zero probabilities, $p_j^i(a)$ is bounded to $[p_{min}, p_{max}]$ where p_{min} is a small positive number (0.05 in our settings) and $p_{max} = 1.0 - p_{min}$. The calculation of $p_j^i(a)$ is based on the formulation given in Equation 1. Using this Equation, one can easily show that $p_j^i(a = \text{`0'}) + p_j^i(a = \text{`1'}) = 1.0$.

$$p_j^i(a) = p_{min} + (p_{max} - p_{min}) \frac{\sum\limits_{x \in s^i} \mathcal{F}(x)\delta(x_j = a)}{\sum\limits_{x \in s^i} \mathcal{F}(x)} \quad (1)$$

Here, x refers to an individual, x_j corresponds to the allele appeared at the individual's j^{th} locus, and $\delta(c)$ returns 1 if condition c (i.e., $x_j = a$) holds and 0 otherwise. $F(x) = e^{5 * f(x)}$ is an exponential function of the individual fitness value $f(x)$. Thus, as f decreases (increases), $\mathcal{F}$ falls (rises) exponentially. Such exponential relation to f mitigates the negative effect of outliers on estimated probabilities.

Note that the probabilistic-schema-matching introduced earlier is implemented by COMPUTE-SSR function. We have already brought up schema-matching two basic shortages which directed us towards proposing the probabilistic version:

[2]The notion of peak is usually used in place of optimum in function maximization problems

[3]Informally, the basins of attraction are the set of initial points from which the search leads to a specified optimum.

1) The matched schema tends to an *all-*-string* as the number of selected individuals increases. In other words, we are either *certain* on choosing a particular allele ('0' or '1') for a schema position, or we pick up an unknown '*' character. However, increasing the number of individuals, inevitably intensifies the uncertainty, and thus leads us an towards informationless schema. Instead of making discrete decisions, probabilistic-schema-matching assigns a continuous value to the schema positions indicating the degree of *belief* on the dominant allele per site. Capturing the uncertainty by introducing $p_j^i(a)$ probabilities automatically resolves the first problem.

2) In Schema-matching, individuals with different fitness values have an identical effect on determining the matched schema. Assuming that all selected individuals are basins of attraction of a single optimum, better individuals are "closer" to the optima, and thus should intuitively more affect the derived schema (assuming a non-deceptive search space). Equation 1 illustrates that the effect of having allele a at site j of the individuals contained in the same subpopulation is exponentially scaled by the individuals' fitness values. Thus, in contrast to schema-matching, better individuals play a more significant role on controlling the probabilities. Furthermore, by scaling individuals with an exponential factor, the effect of outliers (i.e., very bad individuals) would be negligible on the estimated SSR probabilities. In other words, the effect of good individuals are diminished only in cases when the subpopulation contains a few good and many bad individuals.

3) MUTATION$(s_t^i, \mathbf{P}_t^i(a))$*:* The mutation function is separately applied on each subpopulation using the corresponding SSR vector. The mutation of each individual $x = \langle x_1, \cdots, x_l \rangle \in s_t^i$ according to the SSR vector $\mathbf{P}_t^i(a)$ yields a new individual $\acute{x} = \langle \acute{x}_1, \cdots, \acute{x}_l \rangle$ where $\acute{x}_j = 1$ if $U[0,1] \leq p_j^i(a=1)$ (otherwise, $\acute{x}_j = 0$), and $U[0,1]$ denotes a uniform random number sampled from the interval $[0,1]$.

We have already claimed that our proposed mutation scheme retains the exploitation and exploration reciprocal balance. How can this claim be fulfilled?

As we mentioned earlier, exploitation is defined to be the *good use of good information*. The exploitatory power in this framework is achieved by mutating highly-fit individuals. These individuals are not radically disrupted by mutation since the corresponding SSR vectors are highly affected by the good-quality individuals. In other words, only the small proportion of loci for which most of the good individuals do not agree on their corresponding alleles are subject to change.[4] That is how good information concealed in good-quality individuals is wisely used to instantiate good schemata and consequently exploit the good regions of the search space.

On the other hand, the exploratory power of the proposed mutation scheme is achieved by mutating low-fit individuals. These individuals in contrast are subject to high disruption which results in the discovery of possibly new regions of the search space.

[4]Not many of such loci (which good individuals disagree on them) should exist based on the employed partitioning algorithm.

function: PARTITIONING(s_t) **returns** a set of partitions
inputs: Current population s_t
local variables: $\alpha \in (0,1)$, reduction factor
ε, distance threshold

1. $\mathcal{U} \leftarrow$GET-POTENTIAL-ATTRACTORS(s_t, α)
2. $\mathcal{F} \leftarrow$GET-FINAL-ATTRACTORS$(\mathcal{U}, \varepsilon)$
3. $\mathcal{P}_t \leftarrow$CONSTRUCT-PARTITIONS$(s_t, \mathcal{F})$
4. **return** $\mathcal{P}_t$

Fig. 2. Partitioning function

B. Partitioning Method

We have already mentioned that the proposed partitioning method splits the population into a number of disjoint subpopulations. Each subpopulation should ideally comprise the basins of attraction of an identical optima. The proposed partitioning method works based upon a primary assumption: highly-fit individuals which are far-apart (based on an appropriate distance measure), are considered to be under different peaks in the search space. If the distance measure is properly defined for the problem under study, and the chosen individuals are sufficiently high, the assumption is intuitively valid (at least for non-deceptive landscapes).

Figure 2 demonstrates that the partitioning procedure is broken up into three distinct sub-procedures. First, the population is scanned to detect the best available individuals referred to as "attractors" (the GET-POTENTIAL-ATTRACTORS function). These temporarily selected individuals are then filtered to take out proximate individuals and return a final set of attractors which are mutually faraway in the search space (the GET-FINAL-ATTRACTORS function). Once the final set of individuals is determined, each selected individual is considered as the representative of a distinct partition. Then, every other individual in the population would be placed in a partition with the closest representative. (the CONSTRUCT-PARTITIONS function)

1) GET-POTENTIAL-ATTRACTORS(s_t, α)*:* By this function the population s_t is searched to find "good" available individuals. Note that the individuals' goodness should be inevitably defined based on the information on hand. The selection criteria is the relative individual fitness values. Accordingly, the selected individuals are supposed to have fitness values quite close (or identical) to the fitness of the best individual(s) in the population. More formally, we state that the function returns a set of individuals denoted as $\mathcal{U}$ such that:

$$\forall x \in \mathcal{U} \;\; f(x) \geq \alpha f_{max}(s_t),$$

where $f_{max}(s_t) = \max_{x \in s_t} f(x)$. α is set to 0.9 in our framework.

2) GET-FINAL-ATTRACTORS$(\mathcal{U}, \varepsilon)$*:* The potential set of attractors $\mathcal{U}$ is refined by this function to result in the final set $\mathcal{F}$ containing only the individuals with mutual distances more than a threshold value ε. The appropriate choice of a distance measure is problem-dependant. As we briefly discuss

later, Hamming distance is a proper distance metric for our framework.

The GET-FINAL-ATTRACTORS function keeps on seeking pairs of individuals x and y in $\mathcal{U}$ which are closer than a predefined distance threshold ε. It then retains the individual with a higher fitness value, say x, and discards the other one. The resultant set $\mathcal{F}$ contains only individuals which are sufficiently far from each other. More formally, $\forall x, y \in \mathcal{F}\ x, y \in \mathcal{U}$ and $d(x, y) > \varepsilon$, where d denotes a distance measure.

3) CONSTRUCT-PARTITIONS$(s_t, \mathcal{F})$*:* To partition population s_t, this function first constructs $m = |\mathcal{F}|$ sets each containing one of the individuals in $\mathcal{F}$ as the set representative. Let us denote each set as s_t^i and its representative as r_t^i. Then, individual $x \in s_t$ is placed in set s_t^i if and only if

$$d(x, r_t^i) = \min_{\substack{j \in \{1, \cdots, m\} \\ j \neq i}} \{d(x, r_t^j)\}.$$

In summary, the proposed partitioning procedure extracts the *distinct* attractors (i.e., local optima) into separate classes and assigns to each class all individuals which are the *basins of attractions* of the corresponding attractor. Now, we would like to briefly justify why the individuals chosen as attractors are distinct, and why the individuals close to an attractor are supposed to be within its basins of attraction.

Since the selected individuals are *close in fitness value* and *faraway in the search environment*, we can conclude that they are under distinct peaks. The distance between an individual and an attractor (a class representative) in this framework can be considered to be proportional to the number of steps an operator requires to alter the individual to an attractor (or vice versa). Here we need to have a short detour to the concept of "operator landscape" first introduced by Jones [27]. Jones treated the fitness landscape from an operator point of view. Describing landscapes in Hamming space is an ideal view for mutation, since the mutation of a parent yields a child that is nearby in Hamming space. The Hamming distance however is not necessarily useful when considering crossover. SSRGA has no crossover operator. Therefore, adopting a Hamming-proportional distance measure can be relative to the number of steps required to be traversed by the mutation operator. Hence, any individual x which is placed in subpopulation s_t^i is more probable to be within the basins of attraction of r_t^i than r_t^j $(\forall j \in \{1, \cdots, m\}$ and $j \neq i)$ since in operator landscape fewer steps are needed to move from x to r_t^i than to any other representative r_t^j.

Nevertheless, we believe that the proposed partitioning scheme still has room for improvement. Some ideas for improvement are discussed as future work in Section IV.

III. EXPERIMENTS AND RESULTS

A. Problem Formulation

In this work we used the multimodality test-problem generator proposed by [28]. In general, test-problem generators can create random problems from certain classes of problems with user-controllable characteristics. We opted for a controllable multimodal framework primarily to investigate how well SSRGA can explore the landscape and exploit true optima as the number of peaks in the search space varies. Furthermore, it has been shown that the multimodality of a search space is an important characteristic for determining the utility of crossover in an evolutionary algorithm [28]. Therefore, the chosen test-problem generator can provide a good framework to compare the performance of our method equipped with an exploitive mutation with GAs using the traditional crossover operator.

In the multimodality problem generator employed in this paper, the number of peaks $\mathcal{P}$ (the degree of multimodality) can be controlled easily and methodically by the experimenter. The idea is to randomly generate $\mathcal{P}$ peaks each represented by an l-bit string in the space. To evaluate an arbitrary binary string, the nearest peak (in Hamming space) is first located. Its fitness is then the number of bits the string has in common with that nearest peak, divided by l. Formally, the fitness of an arbitrary string j denoted by f_j is derived from the following formula:

$$f_j = \frac{1}{l} \max_{i=1}^{\mathcal{P}} \{l - \text{HAMMDIST}(j, peak_i)\} \tag{2}$$

B. Compared Algorithms

The performance of the proposed algorithm, SSRGA, has been first compared to the canonical GA with a randomly chosen constant mutation and crossover rates. Furthermore, we were also interested in comparing our method to some former works concerned with controlling the GA operator rates. To this end, we have selected a *self-adaptive* [16] and an *adaptive* [22] operator rate control scheme. We have selected these works because they are frequently cited in other related literature, and thus, they seem to be well-matured benchmark methods.

The self-adaptive compared algorithm uses a mechanism for controlling individual mutation rates following the principle of self-adaptation as used in evolutionary algorithms (the method does not possess any crossover operator). The adaptive compared algorithm (referred to as AGA in this paper) controls both mutation and crossover rates for each individual based on the population diversity and the individuals' fitness values. This method is particularly designed for function optimization in multimodal landscapes. We refer readers to the original papers for the detailed description of these methods.

C. Simulation Settings

We assessed different algorithms using problems with different levels of "difficulty."[5] To gauge the difficulty of the problem, we changed the length of the individuals l and the population size n. Clearly, as l increases the problem becomes more challenging since the state space is exponentially extended as the length of individuals grows. Increasing the population size, however, reduces the chance of premature

[5]Here the problem difficulty is defined in terms of the size of the search space and the availability of the resources (i.e., the population size), rather than the deceptiveness of the fitness landscape.

TABLE I

END-OF-RUN RESULTS ACHIEVED BY DIFFERENT ALGORITHMS FOR DIFFERENT PROBLEM SETTINGS

Problem settings		SSRGA		Canonical GA		Adaptive GA		Self-adaptive GA	
Peak No.	(l, n)	average	std.dev.	average	std.dev.	average	std.dev.	average	std.dev.
$\mathcal{P} = 1$	(10,10)	**1.000**	0.000	0.944	0.055	0.986	0.035	0.910	0.054
	(20,30)	**0.999**	0.005	0.883	0.042	0.907	0.040	0.834	0.034
	(30,50)	**0.961**	0.022	0.846	0.39	0.845	0.033	0.787	0.031
$\mathcal{P} = 5$	(10,10)	**1.000**	0.000	0.966	0.048	0.993	0.026	0.972	0.045
	(20,30)	**0.976**	0.028	0.885	0.038	0.912	0.028	0.879	0.031
	(30,50)	**0.914**	0.035	0.833	0.31	0.853	0.023	0.827	0.033
$\mathcal{P} = 10$	(10,10)	**1.000**	0.000	0.987	0.034	0.999	0.010	0.994	0.024
	(20,30)	**0.960**	0.038	0.897	0.032	0.918	0.031	0.900	0.032
	(30,50)	**0.892**	0.034	0.845	0.30	0.856	0.022	0.843	0.025

convergence to non-optimal values, and increases the possibility of finding better solutions at the end of the optimization process (i.e., increases the optimization reliability) [29].

We set up different levels of problem difficulty by setting the parameter pair (l, n) to (10,10), (20,30), and (30,50). Additionally, to investigate the optimization performance in multimodal landscapes, we separately applied each of the three problem settings on search spaces with different degrees of multimodality by setting the total number of peaks $\mathcal{P}$ to 1, 5, and 10. Each peak is represented by a randomly generated l-bit binary string which determines the location of the peak in the search environment. For SSRGA, the minimum distance threshold ε is set to 4 for problems with $l = 3$, and to 2 for the rest of problem settings. The parameters (e.g., crossover and mutation initial rates) of the compared algorithms are set up as recommended by the original papers [16, 22].

Last but not least, due to the stochastic nature of genetic algorithms, the performance of all algorithms over each problem setting is evaluated based on statistics obtained from 100 independent runs.

D. Simulation Results and Analysis

1) Comparison with Benchmark Algorithms: The detailed simulation results including the average and the standard deviation of the best-found fitness values over 100 trials are listed in Table I. The average best fitness (abbreviated as *average* in the Table) is calculated from all of the end-of-run best fitness values which are either obtained when the maximum number of generations is encountered, or if a perfect solution is found. Any bold average value indicates that the corresponding algorithm outperforms the rest for the given parameter setting. Additionally, the underlined average values are those with the smaller corresponding standard deviations. Smaller standard deviations represent steadier and consequently more reliable ultimate solutions. Thus, a bold and underlined average value indicates that the corresponding algorithm provides both more accurate and more consistent final results.

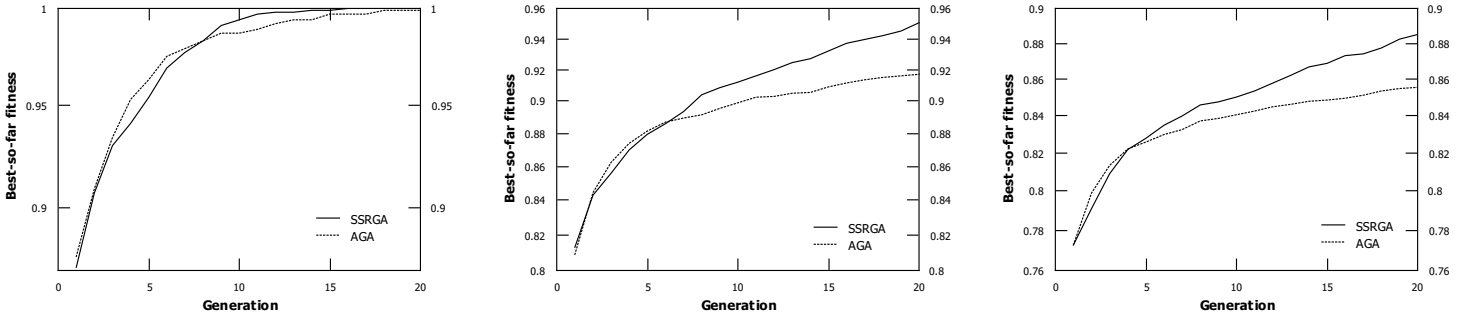
As the statistics displayed in Table I demonstrate, in all experiments, SSRGA clearly outperforms all the compared algorithms. To illustrate the significance of our proposed method, we performed t-tests comparing the averaged optimal fitness values achieved by SSRGA to those obtained from other algorithms. Considering a 95% confidence level, we observed that the comparison of SSRGA with other benchmark algorithms is statistically significant. The only exception was the comparison with AGA on the problem setting of $\mathcal{P} = 10$ and $(l, n) = (10, 10)$ in which we achieved the confidence interval of 68.2%. In this case, both SSRGA and AGA perform (almost) perfectly. Thus, clearly the difference is not significant.

Upon inspection of the end-of-run results displayed in Table I, it can be seen that the performance of SSRGA on identical (l, n)-settings slightly declines as the number of peaks increases (with the exception of $(10, 10)$-setting). However, the trend is reversed in other compared algorithms. In general, as the number of desirable peaks[6](i.e., true optima) increases the possibility of randomly finding one of them grows. Hence, the performance of stochastic search methods (e.g., GAs) which are meant for finding *an* optimum (not *all* optima) should be naturally improved by the growth of the number of the true optima. This would justify the behavior of the compared methods as $\mathcal{P}$ increases. However, why is the trend reversed in the case of SSRGA?

It might be possible to find a clue to this question if we divert our attention from the end-of-run results, and instead focus on the behavior of each process as a whole by examining the convergence curves of different algorithms. The convergence or "best-so-far" curves plot the fitness of the best individual that has been achieved by the current generation. A fitness convergence curve with a rapid ascension to the maximum fitness (1.0) shows a more desirable evolutionary process on average. Where as a convergence curve with a shallower incline may be accordingly viewed as depicting a slow pursuit of improved solutions and possibly premature convergence to non-optimal values. Figure 3 depicts the convergence curves of SSRGA and AGA—the second best algorithm—on the last three problem-settings in which the difference between the performance of the two methods is smaller.

It can be seen from Figure 3 convergence curves that AGA

[6]Some environments possess undesirable or *extraneous* peaks. Such environments will be studied in the full version paper.



(a) $l = 10$, $n = 10$ (b) $l = 20$, $n = 30$ (c) $l = 30$, $n = 50$

Fig. 3. Best-so-far convergence curves of SSRGA (solid lines) and AGA (dashed lines). $\mathcal{P} = 10$ in all (l, n)-settings.

demonstrates a faster rate of convergence on the first 5 to 10 generations. However, its convergence rate rather quickly decreases as generations pass. On the contrary, SSRGA reveals a steady and at the same time rapid rate of convergence. Hence, we expect that as the maximum number of generation increases, AGA traps in some local optima and experiences the premature convergence, while SSRGA converges to a (quasi)optimal solution. To examine our expectation, we increased the maximum number of generations to 100 and reran the algorithms. Figure 4 depicts the achieved convergence curves for a sample problem setting. Similar results have been observed in other test-cases. Figure 4 shows that the behaviors of the two algorithms perfectly meet our expectation.

Now, we are ready to justify the behavior of SRRGA as the degree of the landscape multimodality increases. As the number of peaks grows, SSRGA needs more time to distinguish between different peaks. Ideally each recognized partition should comprise the basins of attraction of a particular peak. However, clear discrimination of the individuals into correct partitions in multimodal environments is a goal which can be gradually achieved during the evolution, (possibly) via an adaptive procedure. Nevertheless, wiser methods clearly result in the faster achievement of the goal. In summary, it is quite possible that in multimodal landscapes the SSRGA partitioning algorithm forces some individuals into incorrect partitions because, for instance, proper representatives of the corresponding correct peaks have not yet been observed. The possibility, however, decreases by observing more individuals along the evolution. Improving the partitioning algorithm will surely enhance SSRGA performance in more challenging problem domains.

2) A Closer Look Over the Trends of the Estimated SSRs: Once observing the SRRGA performance superiority, we were naturally encouraged to take a closer look at the *trend* of the estimated site-specific rates (for the recognized partitions) and compare them with the corresponding ideal probabilities. To concisely visualize and simply analyze the trend of the rate adaptation, here, we focus on studying the SSRGA behavior in a *unimodal* search space in which usually a single

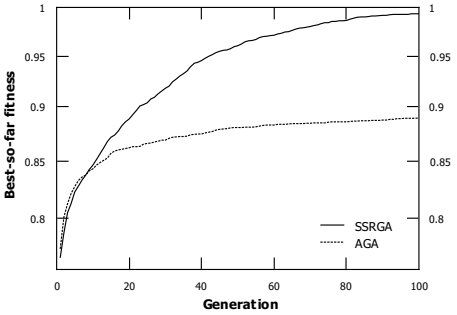


Fig. 4. Best-so-far convergence curves of SSRGA (solid lines) and AGA (dashed lines) for problem settings of $l = 30$, $n = 50$, and $\mathcal{P} = 10$. Maximum nember of generations is set to 100.

partition would be identified. Analyzing the progression of the estimated site-specific rates in multimodal environments is rather complex since it requires tracing the emergence and disappearance of different partitions over generations. We leave such analysis for the full paper version.

SSRGA behaves rather similarly in unimodal environments for all problem-settings. Thus, to succinctly visualize the trend, we pick the $(l, n) = (10, 10)$ setting in which the optimum is usually found within the first 10 generations. Figure 5 depicts the trend of the estimated rates of each site for a sample run. The unique peak is manually triggered to be '1111100000'. Thus, the ideal rates are $p_{max} = 0.95$ for the first five loci, and $p_{min} = 0.05$ for the remaining sites. Figure 5 shows that the estimated site-specific rates are getting closer to the ideal probabilities as the generations pass. In this sample run the optimum has been found in generation 5 wherein the derived rates are sufficiently accurate estimations of the corresponding ideal values.

IV. Summary and Future Work

In this paper we proposed a new mutation scheme which is aimed to tackle both explorative and exploitative responsibilities of genetic operators. the suggested scheme first partitions the population into a number of subpopulations each supposed

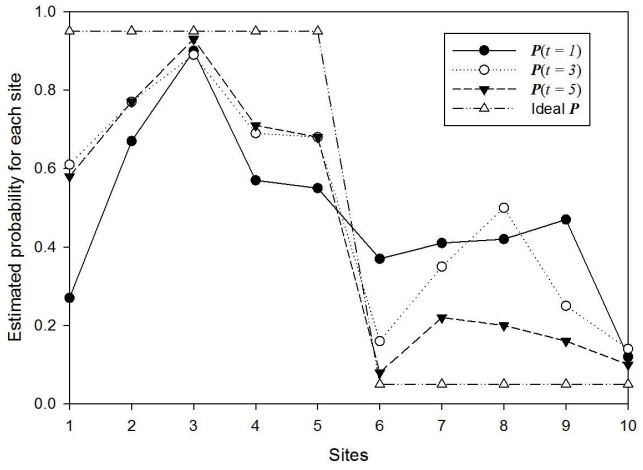


Fig. 5. Trends of probabilities derived for each site for problem settings of $l = 10$, $n = 10$, and $\mathcal{P} = 10$. The probability vectors are shown component-wise at times (generations) $t = 1$, $t = 3$, and $t = 5$. The ideal probabilities for each site are also depicted in the Figure.

to contain the basins of attraction of an identical optima. Then, it tries to identify the pattern of the optimum individual corresponding to each subpopulation via a procedure called probabilistic-schema-matching. The outcome of this procedure is a probability vector (SSR vector) with elements corresponding to the probability of possessing a particular allele at each site. SSR vectors are adapted throughout the evolution to retain the mutual balance of the exploitation and exploration required for a successful search.

To evaluate the performance of our proposed method, we applied it to a range of test problems and compared the results to the canonical genetic algorithm, and genetic algorithms augmented by adaptive and self-adaptive operator rate control schemes. The results demonstrated the superiority of the proposed method over the compared rivals.

In the future, we plan to extend the proposed partitioning algorithm to solve *multiobjective* function optimization problems in which *all* global optima are required to be found. Additionally, we intend to assess SSRGA functionality in more challenging environments which possess possibly many more undesirable peaks than the desirable ones (refer to [30] for definition of desirable vs. undesirable peaks). Moreover, we would like to enhance the proposed method to handle deceptive problems [5] in which undesirable peaks have larger basins of attraction than the desirable peaks. We believe that by choosing and amending some proper GA *niching methods* (see [30] for a thorough review), SSRGA can be extended to multiobjective optimization problems, and also enhanced to perform properly in much more challenging landscapes.

REFERENCES

[1] J. Heitkotter and D. Beasley, "The hitchhiker's guide to evolutionary computation," *FAQ for comp.ai.genetic*, vol. 3, no. 5, 1997.

[2] A. E. Eiben and C. A. Schipper, "On evolutionary exploration and exploitation," *Fundamenta Informaticae*, vol. 35, pp. 35–50, 1998.

[3] H. G. Beyer, "On the "explorative power" of es/eplike algorithms," *Proceedings of the 7th Annual Conference on Evolutionary Programming, , Lecture Notes in Computer Science. Springer, Berlin*, 1998.

[4] J. H. Holland, *Adaptation in natural and artificial systems*. University of Michigan Press, 1975.

[5] D. E. Goldberg, *Genetic algorithms in search, optimization and machine learning*. Reading, MA: Addison Wesley, 1989.

[6] L. Altenberg, "The schema theorem and price's theorem," *Found. of Genetic Algorithms*, vol. 2, pp. 23–49, 1995.

[7] J. J. Grefenstette and J. E. Baker, "How genetic algorithms work: a critical look at implicit parallelism," *In Proceedings of the 3rd Int. Conf. on Genetic Algorithms, Morgan Kaufmann*, pp. 20–27, 1989.

[8] M. D. Vose, "A critical examination of the schema theorem," *Technical Report CS-93-212, University of Tennessee*, 1993.

[9] W. Hordijk and B. Manderick, "The usefulness of recombination," *Advances in Artificial Life, 3rd Int. Conf. on Artif. Life*, vol. 929, pp. 908–919, 1995.

[10] J. Schaffer and L. Eshelman, "On crossover as an evolutionary viable strategy," *In Proc. of the 4th Int. Con. on GAs*, pp. 61–68, 1991.

[11] D. Fogel and J. Atmar, "Comparing genetic operators with gaussian mutations in simulated evolutionary processes using linear systems," *Biological Cybernetics*, vol. 63, pp. 111–114, 1990.

[12] K. A. DeJong and W. M. Spears, "A formal analysis of the role of multipoint crossover in genetic algorithms," *Ann.of Math. and Art. Int.*, vol. 5, pp. 1–26, 1992.

[13] S. Tsutsui *et al.*, "A real coded genetic algorithm with an explorer and an exploiter populations," *In Proc. of the 73th Int. Con. on GAs*, pp. 238–245, 1997.

[14] L. Eshelman, R. Caruana, and J. Schaffer, "Biases in the crossover landscape," *In Proc. of the 3th Int. Con. on GAs*, pp. 10–19, 1989.

[15] A. Eiben, R. Hinterding, and Z. Michalewicz, "Parameter control in evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 2, pp. 124–141, July 1999.

[16] T. Back and M. Schutz, "Intelligent mutation rate control in canonical genetic algorithms," *In Foundations of Intelligent Systems, Springer*, pp. 158–167, 1996.

[17] T. Back, "Self-adaptation in genetic algorithms," *In Proceeding of The 1st European Conference on Artificial Life*, pp. 263–271, 1991.

[18] J. Smith and T. Fogarty, "Self adaptation of mutation rates in a steady state genetic algorithm," *In Proceeding of congress on evolutionary computation*, pp. 318–323, 1996.

[19] F. Vafaee *et al.*, "Adaptively evolving probabilities of genetic operators," *In IEEE Proceedings of the 7th Int. Conf. on Machine Learning and Applications*, vol. 3, pp. 292–299, 2008.

[20] L. Davis, "Adapting operator probabilities in genetic algorithms," *In Proceedings of the 3rd Int. Conf. on Genetic Algorithms*, pp. 61–69, 2002.

[21] T. C. Fogarty, "Varying the probability of mutation in the genetic algorithms," *In Proceedings of the the 3rd Int. Conf. on Genetic Algorithms, Morgan Kaufmann*, pp. 104–109, 1989.

[22] M. Srinivas and L. M. Patnaik, "Adaptive probabilities of crossover and mutation in geneticalgorithms," *IEEE Transactions on Systems, Man and Cybernetics*, vol. 24, no. 4, pp. 656–667, 1994.

[23] F. Vafaee and P. C. Nelson, "A genetic algorithm that incorporates an adaptive mutation based on an evolutionary model," *In IEEE Proceedings of the 8th Int. Conf. on Machine Learning and Applications*, pp. 101–107, 2009.

[24] Z. Yang, "Maximum likelihood estimation of phylogeny from dna sequences when substitution rates differ over sites," *J. Mol. Biol. Evol.*, vol. 10, pp. 1396–1401, 1993.

[25] ——, "Maximum likelihood phylogenetic estimation from dna sequences with variable rates over sites: Approximatemethods," *J. Mol. Biol. Evol.*, vol. 39, pp. 306–314, 1994.

[26] D. L. Swofford *et al.*, *Phylogenetic inference*. Massachusetts: Sinauer Associates.

[27] T. Jones, "Evolutionary algorithms, fitness landscapes, and search," *Ph.D. thesis, University of New Mexico, Albuquerque, NM*, 1995.

[28] W. M. Spears, "Evolutionary algorithms, the role of mutation and recombination," *Natural Computing, Springer-Verlag, Berlin*, 2000.

[29] J. Koljonen and J. T. Alander, "Effects of population size and relative elitism on optimization speed and reliability of genetic algorithms," *In Proc. of the 9th Scand. Conf. on Artif. Intel.*, pp. 25–27, 2006.

[30] M. Srinivas and L. M. Patnaik, "Niching methods for genetic algorithms," *PH.D. Thesis, Univ. Illinois at Urbana-Champaign*, 1995.