2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

Balancing the Exploration and Exploitation in an Adaptive Diversity Guided Genetic Algorithm

Fatemeh Vafaee, György Turán, Peter C. Nelson, and Tanya Y. Berger-Wolf

***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, this work is concerned with proposing a diversity-guided genetic algorithm with a new mutation scheme that is capable of exploring the unseen regions of the search space, as well as exploiting the already-found promising elements. The proposed mutation operator specifies different mutation rates for different sites of an encoded solution. These site-specific rates are carefully derived based on the underlying pattern of highly-fit solutions, adjusted to every single individual, and adapted throughout the evolution to retain a good ratio between exploration and exploitation. Furthermore, in order to more directly monitor the exploration vs. exploitation balance, the proposed method is augmented with a diversity control process assuring that the search process does not lose the required balance between the two forces.**

I. Introduction and Related Work

EVOLUTIONARY Algorithms are nature-inspired search algorithms whose efficacy stems from *exploring* unseen regions of the search space and *exploiting* already encountered promising solutions. Exploration is the ability of a search algorithm to discover entirely new regions of the search space in order to avoid convergence to local optima, while exploitation is the capability of the "*good use of good information*" [1] by visiting regions of search space within the neighboring of previously found promising points. Accordingly, a search algorithm with well-adjusted exploitative power can improve the possibility of generating "better" solutions from already found "good" ones.

It is widely understood that exploration and exploitation are opposite forces, and thus, the success of a search algorithm resides in achieving equilibrium between the two [2]. 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 local optima.

There has been a considerable amount of discussion and sometimes contradictory studies on how exploration and exploitation can be achieved in evolutionary algorithms. Wong *et al.* [3], for instance, argued that variation operators—mutation and crossover—are supposed to introduce the required explorative and exploitative power into genetic algorithms (GAs). On the other hand, a consensus view is that variation operators explore the search space, while exploitation is attained through the selection process. Eiben and Schippers [1], however, questioned this common opinion, and concluded that there is no generally accepted perception about exploration and exploitation in EAs, and more intensive research is required for deeper understanding of the fundamentals of evolutionary search processes.

Črepinšek *et al*. [4] recently presented a more comprehensive and systematic treatment on evolutionary exploration and exploitation. They have argued that both variation and selection operators as well as other factors (e.g., population size and representation) can individually or collectively contribute to exploration and exploitation. For instance, while selection is mainly seen as an exploitation operator by moving the search toward the regions of the best individuals, it can control the level of both exploration and exploitation by varying the selection pressure—i.e., higher selection pressure moves the search towards more exploitation, while lower selection pressure pushes the search towards more exploration [5]. Furthermore, mutation and crossover operators are also involved in exploration as well as exploitation. Variation operators modify or exchange genetic information, and thus increase the population diversity. From this point of view, they are counted as exploration operators. However, they are also aimed to generate better neighbors from the existing good individuals, and thus have an exploitation role; the exploitatory power of variation operators enhances if the locality property[1] holds [6].

While evolutionary exploration and exploitation can be achieved by variation and/or selection operators, the reciprocal balance between these two complementary forces has been usually managed by proper *parameter-control* settings in which the parameter values (e.g., variation operator rates) are subject to change during an EA run. Note that EAs are intrinsically dynamic processes requiring variable amounts of exploration vs. exploitation at different times or states of the evolution. Parameter control techniques can be divided into two major categories, referring to the taxonomy introduced by [7]: *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 recombination. The intuition is that the parameter values (e.g., the genetic operator rates) have a chance to coevolve with problem solutions to hopefully (near)-optimal settings; see [8], [9], and [10] 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

Fatemeh Vafaee is with the Charles Perkins Centre, University of Sydney, Sydney, NSW, Australia (email: fatemeh.vafaee@sydney.edu.au). György Turán is with the Department of Mathematics, Statistics and Computer Science, University of Illinois at Chicago, Chicago, IL, USA, and MTA-SZTE Research Group on Artificial Intelligence (email: gyt@uic.edu). Peter C. Nelson and Tanya Y. Berger-Wolf are with the Department of Computer Science, University of Illinois at Chicago, Chicago, IL, USA (email: {nelson, tanyabw}@uic.edu).

[1]The concept of locality describes how well neighboring genotypes correspond to neighboring phenotypes.

978-1-4799-1488-3/14/$31.00 ©2014 IEEE

the search. Among different parameters, adapting operator probabilities has long been recognized as a commendable path toward improving the performance of Evolutionary Algorithms; see for example [11],[12], [13], and [14].

In this work, we propose a 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, while bad individuals are freely disrupted to allow a vast exploration of the search space. The mutation rates are also *adaptively* adjusted throughout the evolution. Furthermore, in order to more directly monitor the exploration vs. exploitation balance, we augmented the proposed EA with a *diversity* control process assuring that the search process does not lose the required balance between the two forces.

In brief, diversity refers to differences among individuals, which can be at the genotype or phenotype levels. In the genotype level, differences among genomes/structures within the population is of interest, while in the phenotype level, differences among fitness values are sought. Loss of diversity corresponds to the lack of exploration, resulting in premature convergence and trapping into local optima. Excessive diversity, on the other hand, is counterproductive when high exploitation is required. It is widely accepted within the EA community that well-adjusted diversity of a population greatly contributes to EA performance [15]. Both genotype and phenotype diversity measures are classified to *difference-based*, *distance-based* and *entropy-based* measures [4]. The latter is a very succinct measure representing the amount of population disorder as a measure of the diversity. It also incorporates the distribution of values within the population. Interested readers may wish to consult [4] for a comprehensive review of different types of diversity measures. In this work we proposed a new genotypic entropy-based diversity measure which is used to dynamically monitor the exploration and exploitation balance.

The method proposed in this paper fits into the category of adaptive parameter control Evolutionary Algorithms. Accordingly, we selected several adaptive and self-adaptive operator rate control methods as benchmark evolutionary algorithms to evaluate the performance of our suggested method.

We also evaluated our proposed method against a set of Estimation of Distribution Algorithms (EDAs), a class of stochastic optimization techniques closely related to evolutionary algorithms. EDAs search the space of potential solutions by building and sampling explicit probabilistic models of promising candidate solutions [16]. The main difference between EDAs and most conventional evolutionary algorithms is that EAs generate new candidate solutions using a distribution model *implicitly* defined by the current state of the search and the set of variation and selection operators, whereas EDAs use an *explicit* probability distribution encoded by probability vectors, Bayesian networks, or another model class [17].

The method proposed in this paper is a modified version of our recently developed site-specific rate algorithm [18]; it is henceforth referred to as *DiG-SiRG* which stands for *Diversity-Guided Site-specific Rate Genetic-algorithm*.

The remainder of this paper is organized as follows: Section II provides a detailed description of the DiG-SiRG algorithm and the underlying ideas. Sections IV, V, and VI respectively describe the benchmark problems, the compared algorithms, and the experimental results for the MAX-SAT problem domain. Finally, Section VII briefly summarizes the proposed algorithm.

II. A Diversity Guided GA with an Explorative and Exploitative Mutation Scheme

Exploration in GAs can be simply achieved by a random perturbation of individuals. However, achieving an exploitative power, and then, retaining its balance with the required exploration are properties which are not readily attainable. As it is formerly mentioned, exploitation is *good use of good information* to move the search towards the desirable improvements. However, what is good information, and how can it be extracted and used for further improvements?

In non-deceptive environments, the underlying pattern of already found highly-fit individuals seems to be a reliable source of the currently available good information. To learn such patterns, we made use of a revised version of *motif representation* in biology (roughly speaking, motifs are biologically important patterns frequently recurring in different DNA sequences). Once the pattern of good individuals has been derived, it is used to compute mutation rates specified to each site of every individual.

Note that prior to pattern derivation, highly-fit individuals should be extracted out of the population. Such individuals should be carefully selected to preserve the genetic diversity in the resultant pattern. We accordingly, developed a genotypic entropy-based diversity measure and incorporated it into the "elite set" selection phase.

To better understand the proposed motif-based approach it is worth to make a very brief detour into DNA motif representation: in genetics, a DNA[2] *motif* is defined as a nucleotide acid sequence pattern that is widespread and has some biological significance [19]. Given a set of m DNA sequences, the problem of *motif discovery* can be roughly defined as finding a set of m subsequences $\{x_1, x_2, \ldots, x_m\}$, one from each input sequence, that maximizes a predefined scoring criterion. The set of discovered patterns is usually represented in a matrix form referred to as *position frequency matrix* (PFM). Considering n subsequences of length l and $b = 4$ nucleotide bases, a PFM is a matrix $F = (f_{i,j})$ of size $b \times l$, where $f_{i,j}$ is the frequency of the occurrence of nucleotide basis b_i at site j of all n sequences.

A. DiG-SiRG Algorithm (General Ideas)

In our framework, highly-fit individuals can be thought of as motifs since they are "important" sequences in a sense that they possibly carry the underlying patterns of promising solutions. Assume that out of a population of n binary solutions, $m(\leq n)$ individuals have been drawn based on the superiority of their fitness values. Intuitively, we interpret the underlying pattern of (available) promising solutions as the

[2]On its most elementary level, the structure of DNA sequence can be thought as a long succession of four letters—A, C, G, and T—representing the four nucleotide bases of a DNA strand.

possibility of the occurrence of each allele (i.e., '0' and '1') appearing at each site of high-quality individuals selected out of the current population.

To derive such a pattern, we can construct a revised version of position frequency matrix, denoted as $PFM' = (f'_{i,j})$ in which each component $f'_{i,j}$ corresponds to the *probability* of the occurrence of allele $i \in \{0, 1\}$ at site j (i.e., simply divide the allele frequency by the size of the selected population).

However, the PFM' representation suffers from an important drawback: all chosen individuals have an identical effect on determining the pattern of promising solutions. However, intuitively, better individuals should play a more important role in the procedure.To resolve this problem, we further refine the representation by constructing a *position weight matrix*, $PWM = (w_{i,j})$, wherein the individuals are weighted by their fitness values. Accordingly, $w_{i,j}$ is derived by Equation 1 where $f(x_k)$ is the fitness of individual x_k.

$$w_{i,j} = \frac{\sum_{k=1}^{m} f(x_k).\delta(x_{k,j} = i)}{\sum_{k=1}^{m} f(x_k)}, \tag{1}$$

PWM is then used to derive site-specific mutation rates. Nonetheless, the key to PWM construction is the selection of highly-fit individuals such that the genetic diversity is preserved among the selected individuals. Let $s_t = \{x_1^t, \ldots, x_n^t\}$ be the population of n individuals at time t, and $\mathcal{E}_t \subseteq s_t = \{x_1, \ldots, x_m\}$ is the elite set containing individuals with comparatively higher fitness values. If $\mathcal{E}_t$ happens to contain genotypically identical individuals—i.e., no diversity—all the allele probabilities ($w_{i,j}$'s) of the corresponding PWM matrix drop off to 0 or raise to 1. Such PWM will possibly lead the evolution towards trapping into local optima and premature convergence resulted from the loss of enough exploration. Conversely, by an excessively diverse $\mathcal{E}_t$, PWM loses the information content required for an effective exploitative search. Hence, the number of individuals in the elite set should be carefully adjusted at each generation in order to preserve the required diversity throughout the evolution.

As previously mentioned, diversity refers to the differences among the individuals, either in genotypic (structure) or phenotypic (fitness) levels. In EAs, identical genotypes produce the same fitness, and thus, a decrease in genotype diversity necessarily decreases the phenotype diversity. Therefore, to measure the diversity, one may favour to define some structural measures. Entropy is a succinct measure of diversity, and it is shown to be a useful measure for genotypic diversity (e.g., [20]).

In the information theory, entropy is a measure of uncertainty or disorder in a signal or a random event. Shannon [21] defines the entropy of a random event X, with possible states $\{X_1, \ldots, X_n\}$ as:

$$H(X) = -\sum_{i}^{n} p_i \log_2(p_i) \tag{2}$$

function: DiG-SiRG-MUTATION(s_t)
inputs: population $s_t = \{x_1^t, \ldots, x_n^t\}$
returns: mutated population

1. $\mathcal{E}_t \leftarrow$FORM-ELITE-GROUP(s_t)
2. $PWM_t \leftarrow$CONSTRUCT-PWM($\mathcal{E}_t$)
3. **for** *each individual* $\boldsymbol{x}_k^t = \langle x_{k,0}^t, \ldots, x_{k,l-1}^t \rangle \in s_t$ **do**
4. **for** *each site* $j = 0, \ldots, l-1$ **do**
5. $\mu_{k,j}^t \leftarrow$COMPUTE-MU($\boldsymbol{x}_k^t, j, PWM_t$)
6. **if** $rand(0,1) \leq \mu_{k,j}^t$ **do**
7. $x_{k,j}^t \leftarrow x_{k,j}^t \oplus 1$
8. **return** s_t

Fig. 1. Site-specific Rate Mutation scheme

where p_i is the probability of observing the i^{th} outcome, X_i. When more states are available to a system, the systems' unpredictability and disorder/diversity increase, and thus, entropy rises. When only one outcome is observed, there is no uncertainty, and therefore, the entropy of the system is zero. Conversely, entropy becomes maximal if all the outcomes are equally likely—i.e., if the system has n states which are equiprobable ($p_i = 1/n$), the entropy is maximum: $H_{max} = -\sum_{i}^{n} \frac{1}{n} \log_2(\frac{1}{n}) = \log_2 n$

Here, we define the entropy for a set of selected population $\mathcal{E}_t = \{x_1, \ldots, x_m\}$ by first placing each individual x_i into a genotype class C_i. $C_0, C_1, \ldots, C_l$ are possible genotype classes where l is the individual length and C_i comprises individuals whose *hamming distance*[3] with the best individual in the current population $x^*(s_t)$ is i. p_i in Equation 2 is therefore the proportion of the population occupied by the population partition C_i. More formally, p_i is derived by Equation 3 where $h^*(x_k^t) =$ hamming-distance$(x^*(s_t), x_k^t)$, and $\delta(c)$ returns 1 if condition c (i.e., $h^*(x_k^t) = i$) holds and 0 otherwise.

$$p_i^t = \frac{1}{m} \sum_{k=1}^{m} \delta(h^*(x_k^t) = i) \tag{3}$$

In the next section, we describe how this entropy measure is used to preserve the elite set diversity.

B. DiG-SiRG Algorithm (Pseudo-code)

DiG-SiRG starts by randomly initializing a population of n individuals of length l. Then, following the inherent mechanism of evolutionary algorithms, it incorporates two primary operations of *selection* and *reproduction* during the main evolutionary cycle. As for the selection mechanism, DiG-SiRG employs a modified version of the canonical elitist *fitness-proportional* selection [22] in which the best individual survives with the probability of one and replaces the *worst* individual in the reproduced population. The reproduction operation is then accomplished by means of the newly proposed mutation operator. Fig. 1 gives the detailed pseudo-code of the proposed mutation scheme. As the Figure shows, the DiG-SiRG-MUTATION function begins by forming

[3]Hamming distance between two strings of equal length is the number of positions at which the corresponding symbols are different.

an elite set containing highly-fit individuals selected out of the population s_t. An individual $x_k^t \in s_t$ is eligible for being selected if

$$f(x_k^t) \geq \alpha_t f_{max},$$

where f_{max} is the best individual in the current population, and α_t is the reduction factor at time t which should be adjusted dynamically to preserve $\mathcal{E}_t$ diversity. Let $H(\mathcal{E}_t)$ be the entropy diversity measure of elite set $\mathcal{E}_t$ (based on Equations 2 and 3). We define $[d_{min}, d_{max}] \subset [0, H_{max}]$ to be a *healthy diversity* interval such that if $H(\mathcal{E}_t)$ falls within this interval, the exploration and exploitation are assumed to be in a good ratio, and thus the process is proceeded without concern. Otherwise, $H(\mathcal{E}_t) < d_{min}$ implies that evolution is starved of exploration, while $H(\mathcal{E}_t) > d_{max}$ means that the process falls into excessive exploration. We empirically assigned $\frac{1}{5}H_{max}$ to d_{min} and $\frac{3}{5}H_{max}$ to d_{max}. A better practice, however, is to make the healthy diversity interval time-sensitive based on the intuition that at the earlier stages of evolution more exploration is required while towards the end of the process exploitation is more demanding.

$$\alpha_t = \begin{cases} \alpha_{t-1} - 0.01 & \text{if } H(\mathcal{E}_t) < d_{min} \\ \alpha_{t-1} + 0.01 & \text{if } H(\mathcal{E}_t) > d_{max} \\ \alpha_{t-1} & \text{otherwise} \end{cases} \quad (4)$$

According to Equation 4, when the elite set diversity falls below the given d_{min} threshold, reduction factor decreases by the step-size of 0.01 to increase the genotypic diversity by including individuals with slightly lower fitness values. On the contrary, if the diversity exceeds the d_{max} threshold, α_t increases; this will restrict the number of individuals in the elite set to reduce the diversity, and consequently the exploration rate.

Once a controlled elite set is formed, the selected individuals are considered as the currently available promising solutions to construct the position weight matrix (PWM) according to Equation 1. The resultant PWM gives the *weighted* probability of occurrence of each allele at every site of high-quality individuals. This has been considered as the underlying pattern of the promising solutions discovered so far.

Accordingly, for each site $j = 0, \ldots, l-1$ of every individual $x_k^t \in s_t$, a distinct mutation rate $\mu_{k,j}^t$ is computed (COMPUTE-MU function in line 5 of Fig. 1), and used to mutate site j of individual x_k^t. $\mu_{k,j}^t$ is computed according to Equation 5, where ε is a small positive number to avoid zero probabilities. Furthermore, $w_{i,j}$ is the (i,j)th element of matrix PWM where $i = x_{k,j}^t$ is the allele number (i.e., either 0 or 1 corresponding to alleles '0' and '1') appearing in site j of individual x_k^t.

$$\mu_{k,j}^t = \begin{array}{l} [\varepsilon + (1-\varepsilon)(1 - f(x_k^t)/f_{max})] * \\ \left[\varepsilon + (1-\varepsilon)(1 - w_{i,j}^t)\right]. \end{array} \quad (5)$$

By using the above equation, the mutation rate at each site is gauged based on 1) the probability of having the corresponding allele at that site based on the PWM constructed out of high-quality individuals, and 2) the fitness of the individual which is about to get mutated. In general, better individuals are more probable to be close to the promising regions of the search space, and they usually carry more valuable information. Accordingly, they need to be modified more carefully. In other words, assigning them high mutation rates is rather risky and may cause destruction of the useful information.

III. EXPERIMENTS

We evaluated the performance of our proposed method on the maximum satisfiability (MAX-SAT) problem, a standard problem in the theoretical computer science. Note that DiG-SiRG is a general-purpose (domain-independent) search algorithm and it can be applied to any GA-compatible optimization problem. As it is thoroughly discussed in the next Sections, we were interested in MAX-SAT problem mainly due to the fact that it can be simply used to generate problems with controllable degree of "difficulty".

IV. PROBLEM FORMULATION

Given a Boolean formula in conjunctive normal form (CNF), MAX-SAT is the problem of finding an assignment of variables that maximizes the number of satisfied clauses of the given formula. MAX-SAT generalizes the problem of Boolean satisfiability (SAT) which is a decision problem aimed to determine whether or not variables of the given formula can be assigned in such a way that makes the given formula evaluate to true [23].

We have chosen the MAX-SAT domain since because of the natural binary string representation of Boolean expressions [24], GAs can be simply applied and evaluated over the MAX-SAT problem domain.

More importantly, MAX-SAT problems can be realized as "GA-hard" problems as they are shown to be particularly difficult for standard genetic algorithms to solve them [25]. A problem would be characterized as GA-hard if it is identified to be hard from a *GA point-of-view*. Although the terminology recalls that of computational complexity, fundamentally different criteria (e.g., epistasis, deception, massive multi-modality, and long paths to a single global optimum) were used in this context to define the term "GA-hard".[4] Hence, once a genetic algorithm performs effectively over a GA-hard search environment, its efficacy can be extrapolated to a wide range of problem domains.

Furthermore, as shown by [26], MAX-SAT problems share similar structures with NK-landscapes [27], and both belong to the encompassing class of embedded landscapes. NK-landscapes are often used as benchmark problems of scalable difficulty (epistasis) for genetic algorithms. However, from a practical perspective, there is a problem in using a generator of NK-landscapes, that is, the space required to store the tables used to compute fitness is exponential in K. This restricts us to small models in which the performance difference among the competing methods may not be decisive enough. Furthermore, all genes in NK-landscape models have an identical degree of dependency (K), whereas

[4]Note that, unlike the notion of NP-hardness, GA-hardness is an informal notion with no known formal counterpart.

in many real-world problems, the dependencies among the problem variables vary considerably.

We instead used the Boolean satisfiability problem generator developed by [28], which helps us to avoid concerns associated with the NK model, and at the same time, allows us to systematically generate MAX-SAT problems with controllable degree of difficulty in terms of the problem's scale and "epistasis". Epistasis expresses the relationship between separate genes in a chromosome. More precisely, a system has low (high) epistasis if the optimal allele for any locus depends on small (large) number of alleles at other loci [28]. Epistasis can be interpreted as expressing non-linearity in the fitness function, thus, the more epistatic the problem is, the harder it is to find an optimum [29]. We note that there is no general agreement on a formal definition of epistasis. In what follows, we use a formal measure (the degree of the Walsh representation of a Boolean function) which can be considered as one specific approach to define this notion.

A. Boolean Satisfiability and Genetic Algorithms

Prior to applying GAs to any particular problem two critical decisions should be made: 1) selecting an appropriate representation for the solution space, and 2) defining an external evaluation function to determine the utility of the candidate solutions. (MAX-)SAT appears to have highly GA-compatible string representation, namely, binary strings of length l in which the i^{th} bit represents the truth value of the i^{th} Boolean variable (assuming that l Boolean variables appear in the given Boolean expression).

Regarding the evaluation function, a common practice is to assign fitness to individual subexpressions in the original expression and combine them in some way to generate a total fitness value [28]. We accordingly follow Smith's suggestion [24] which defines the value of *true* to be 1 and the value of *false* to be 0. He then determines the fitness f of a simple subexpressions e_i according to the following formulae.

$$f(\bar{e}) = 1 - f(e),$$
$$f(e_1 \vee e_2 \vee \ldots \vee e_n) = Max(f(e_1), f(e_2), \ldots, f(e_n)),$$
$$f(e_1 \wedge e_2 \wedge \ldots \wedge e_n) = Ave(f(e_1), f(e_2), \ldots, f(e_n)),$$

B. Creating epistatic problems

As proposed by [28], Boolean satisfiability expressions can also be easily used to gauge the difficulty of the problem under study by changing the degree of the epistasis of the interactions. The proposed technique is mainly centered on creating specific Hamiltonian circuit problems which are then converted to SAT expressions. The generated Boolean expressions have always only one solution which makes the problem harder to solve. The (directed) Hamiltonian circuit (HC) problem consists of finding a tour in a directed graph that touches every node exactly once. The definition of the HC problem implies that, for any solution, each node must have exactly one incoming edge and one outgoing edge. Thus any subgraph which violates this constraint cannot be a solution [30]. Thus, a satisfiability instance expressing this necessary condition for the existence of a Hamiltonian cycle can be formulated as the conjunction of terms indicating valid edge combinations for each node. Assigning value 1 to an edge variable indicates that the edge *is* in the tour. Conversely, an edge variable with the value of 0 would not be considered to be in the tour. Please refer to [28] for complementary examples and more detailed explanations.

TABLE I. THE NO. OF VARIABLES AND CLAUSES OF THE CORRESPONDING CNF BOOLEAN FORMULA, AS THE DEGREE OF EPISTASIS CHANGES.

No. of Nodes	No. of variables	No. of clauses
$N = 5$	15	43
$N = 10$	55	333
$N = 15$	120	1123
$N = 20$	210	2663
$N = 25$	325	5203
$N = 30$	465	8993
$N = 35$	630	14283
$N = 40$	820	21323

Using the above idea, Spears uses the following family of HC instances to create MAX-SAT instances with different degrees of epistasis. Consider a graph of $N + 1$ nodes $G = (V, E)$ where nodes are labeled using consecutive integers from 0 to N (i.e., $V = \{0, 1, \ldots, N\}$). Suppose the first node has directed edges to all other nodes except the last one. The next $N - 1$ nodes have edges to all other nodes with the higher label, and the last node has a direct edge back to the first node. Having such a graph, the only valid Hamiltonian tour is to follow the nodes labeled in the increasing order, and then from the last node travel back to the first one. Increasing N increases the epistasis (see [30] for more details).

Note that the resultant Boolean formula should be first converted to CNF format to be used as input to the MAX-SAT problem. Due to the specific form of Boolean expressions employed in this work, the equivalent CNF expression can be achieved quite efficiently, i.e., the resulting CNF expression has size $O(N^2)$.

It is also worth mentioning that Rana *et al.* [25] considered the epistasis of MAX-SAT problems defined as the degree (the maximal number of variables in a nonzero term) is equal to the size of the largest clause. In our examples the largest clause size is $N - 2$, where N is the number of nodes in the corresponding graph.

Table I shows the number of variables (i.e., the number of edges in Hamiltonian circuits) and the number of clauses in the equivalent CNF expressions as the number of nodes (i.e., the degree of epistasis) varies from 5 to 40 with the step size of 5.

V. COMPARED ALGORITHMS

The performance of the proposed method on a wide range of Boolean satisfiability problems has been compared to the canonical GA as well as some evolutionary algorithms formerly designed to control the rates of variation operators. We have selected the following adaptive/self-adaptive evolutionary algorithms which support the binary representation, and thus applicable to the MAX-SAT problem:

1) auto-tuning based genetic algorithm (atGA) [31],
2) canonical genetic algorithm (CGA),
3) adaptive genetic algorithm based on fuzzy mechanism (FuzzyAGA) [32],

4) adaptive genetic algorithm with mutation and crossover matrices (Matrix-GA) [33],
5) ranked based genetic algorithm (Rank-GA) [13],
6) self-adaptive genetic algorithm (SAGA) [34], and
7) site specific rate genetic algorithm (SSRGA) [11].

The compared algorithms are either recently devised and shown to outperform some of the former mutation rate heuristics, or they are pioneer works in this context, frequently cited, and seem to be well-matured benchmark algorithms. Due to the space limitation, we refer readers to the corresponding references for the description of the selected EAs.

We also chose a set of estimation of distribution algorithms (EDAs) to compare our method against a benchmark suite of stochastic optimization techniques, which are closely related to evolutionary algorithms. EDAs explore the space of possible solutions by building and sampling explicit probabilistic models of high-quality candidate solutions [17]. The main difference between EDAs and EAs including DiG-SiRG, is that EDAs estimate an *explicit* distribution to sample new solutions, while EAs use selection and variation operators together with the current population to define an *implicit* probability distribution over the populations of solutions; the new population can be then viewed as a sample of such distribution [17].

EDAs are usually categorized according to the complexity of distribution models they use. Univariate EDAs assume that the variables in a problem are independent, while multivariate EDAs are designed to capture multivariate interactions among problem variables [16]. We have selected the following well-known examples of univariate and multivariate EDAs for comparison:

1) Univariate Marginal Distribution Algorithm (UMDA) [35]
2) Extended Compact Genetic Algorithm (ECGA) [36]; we used ECGA's C++ package developed by [37], and
3) Bayesian Optimization Algorithm (BOA) [38]; we used BOA's C++ package implemented by [39].

We have selected these EDAs as they have been applied to different problems, and have shown successful records of performance (e.g., [40] and [41]). Furthermore, as the implementations of multivariate EDAs are generally complex, to avoid any replication bias, we sought for EDAs whose source code is publicly available.

Note that besides the above-mentioned general difference between EDAs and GAs, DiG-SiRG individualizes the mutation vector μ_j to each candidate solution based on the individual's fitness value, while EDAs typically sample an entire population using an identical distribution probability vector. There are, of course, some other differences between DiG-SiRG's mutation scheme and each particular EDA. For instance, both DiG-SiRG and UMDA work on a set of high-quality solutions to derive position-dependent sampling probabilities or mutation rates. However, at every site of high-quality individuals, DiG-SiRG uses the weighted probability of occurrence of each allele, while UMDA uses the relative frequency of '1'. Please refer to the corresponding references for the description of the selected EDAs.

VI. Results

A. Comparison with benchmark EAs

Using the Spear's mechanism [28] described in section IV-B, we have constructed several graphs where N was set to 5, 10, 15, 20, 25. 30, and 40, derived the Boolean expressions of the corresponding Hamiltonian circuits,[5] and converted the expressions to conjunctive normal forms.

The remaining shared free parameters include the population size which is set to 50 in all experiments, and the maximum number of generations which is set to 500. The initial value of the reduction factor, α_0 in DiG-SiRG is chosen to be 0.99. Due to the stochastic nature of evolutionary algorithms, the performance of all algorithms over each problem is evaluated based on statistics obtained from 50 independent runs.

Table II lists the simulation results of the epistatic problems when N is set to $5, 10, \ldots, 40$. The results include the average, the standard deviation of the best-found fitness values, and the average CPU time (in seconds) of a single run. The average best fitness (abbreviated as *ave* 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 in the Table indicates that the corresponding algorithm outperforms the rest for the given problem.

As the statistics displayed in Tables II demonstrate, in all experiments, except for the epistatic problem with $N = 5$, DiG-SiRG outperforms the compared algorithms regarding both the efficiency (running time) and the precision (average fitness) of the best found solutions. Rank-GA and Matrix-GA are the second and the third winners, respectively. Even though Rank-GA and then Matrix-GA perform relatively well as compared to other benchmark algorithms, they both suffer from excessive fitness evaluations and run time complexity. Comparing to DiG-SiRG and other benchmark algorithms, number of fitness evaluations are doubled per generation. In addition, sorting the population two times at every iteration further raises the algorithmic time complexity as the results in Table II demonstrate.

Last but not least, in order to illustrate the significant statistical differences between DiG-SiRG and the benchmark algorithms, we decided to perform a statistical hypothesis test among the compared methods. Prior to opt for an appropriate parametric test, we employed Shapiro-Wilk test to ensure the data being used follows a Normal distribution. As the data were normally distributed according to Shapiro-Wilk test, we were open to choose *t*-test to detect significant differences between our proposed method and the compared algorithms. According to the obtained results, except for problems with $N = 5$ in which all algorithms perform almost perfectly, p-value of every comparison is almost always quite close to 0 ($p \approx 0$) indicating that the superiority of DiG-SiRG over all benchmark evolutionary algorithms is significant at any typical significance level.

[5]Note that the number of Boolean variables would be identical to the number of the edges of the constructed graphs that is equal to $N(N+1)/2$ where the number of nodes is $N + 1$.

TABLE II. BEST FOUND FITNESS OF COMPARED EAS AS THE DEGREE OF EPISTASIS CHANGES.

Epistasis level		$N = 5$	$N = 10$	$N = 15$	$N = 20$	$N = 25$	$N = 30$	$N = 35$	$N = 40$
DiG-SiRG	ave (std.)	0.989 (0.020)	**0.992** (0.003)	**0.993** (0.001)	**0.994** (0.001)	**0.993** (0.001)	**0.986** (0.002)	**0.975** (0.003)	**0.963** (0.002)
	time(sec.)	0.013	0.202	0.602	1.388	2.588	4.373	6.953	10.752
SSRGA	ave (std.)	**1.000** (0.000)	0.974 (0.003)	0.944 (0.006)	0.907 (0.005)	0.879 (0.007)	0.859 (0.006)	0.842 (0.004)	0.830 (0.004)
	time(sec.)	0.026	0.297	0.808	1.806	3.283	5.378	8.347	12.413
SAGA	ave (std.)	0.985 (0.019)	0.944 (0.006)	0.904 (0.008)	0.874 (0.007)	0.852 (0.005)	0.838 (0.005)	0.826 (0.004)	0.817 (0.004)
	time(sec.)	0.036	0.257	0.826	1.593	2.875	4.840	7.465	11.713
Rank-GA	ave (std.)	**1.000** (0.000)	0.989 (0.004)	0.991 (0.002)	0.984 (0.003)	0.969 (0.005)	0.950 (0.005)	0.933 (0.006)	0.915 (0.005)
	time(sec.)	0.017	0.579	1.820	4.225	7.924	13.435	21.077	32.028
Matrix-GA	ave (std.)	0.998 (0.009)	0.985 (0.005)	0.981 (0.005)	0.965 (0.007)	0.943 (0.007)	0.920 (0.006)	0.904 (0.006)	0.888 (0.007)
	time(sec.)	0.166	1.847	4.025	8.029	13.514	20.727	30.113	41.935
FuzzyAGA	ave (std.)	**1.000** (0.000)	0.932 (0.017)	0.891 (0.016)	0.864 (0.018)	0.845 (0.011)	0.832 (0.012)	0.821 (0.008)	0.817 (0.011)
	time(sec.)	0.109	0.328	1.026	2.380	4.439	7.553	11.852	18.521
CGA	ave (std.)	0.977 (0.037)	0.915 (0.017)	0.886 (0.018)	0.861 (0.017)	0.843 (0.019)	0.820 (0.020)	0.810 (0.014)	0.800 (0.014)
	time(sec.)	0.047	0.589	1.796	4.122	7.908	13.760	21.956	33.661
atGA	ave (std.)	**1.000** (0.000)	0.950 (0.016)	0.917 (0.025)	0.880 (0.020)	0.857 (0.022)	0.834 (0.016)	0.821 (0.015)	0.813 (0.012)
	time(sec.)	0.031	0.622	2.086	5.081	9.570	16.352	27.017	42.194

TABLE III. END OF RUN STATISTICS OF THE PROPOSED METHOD AND THE COMPARING EDAS.

Method	Epistatic level	Time to stop in sec.	Ave. fitness (std.)	Ave. gen. to conv.	Stop reason
DiG-SiRG	$N = 5$	0.046	1.000 (0.0)	—	optimum found
	$N = 10$	0.228	0.994 (1.0E-3)	—	
	$N = 15$	0.218	0.991 (4.4E-4)	—	
	$N = 20$	0.652	0.990 (3.2E-4)	—	
	$N = 25$	2.071	0.990 (2.2E-4)	—	
	$N = 30$	5.462	0.990 (1.6E-4)	—	
	$N = 35$	12.919	0.990 (9.8E-5)	—	
	$N = 40$	31.043	0.990 (1.6E-4)	—	
UMDA	$N = 5$	0.06	0.945 (0.027)	803.34	allele convergence
	$N = 10$	1.25	0.949 (0.015)	1,945.06	
	$N = 15$	4.98	0.937 (0.015)	2,292.6	
	$N = 20$	11.22	0.893 (0.017)	2,622.74	
	$N = 25$	20.07	0.857 (0.018)	2,820.94	
	$N = 30$	33.32	0.820 (0.019)	2,955.44	
	$N = 35$	53.93	0.793 (0.019)	3,106.98	
	$N = 40$	90.68	0.767 (0.014)	3,215.64	
ECGA	$N = 5$	0.24	0.947 (0.022)	69.92	allele convergence
	$N = 10$	2.96	0.980 (0.004)	82.14	
	$N = 15$	13.08	0.982 (0.006)	81.60	
	$N = 20$	45.78	0.979 (0.005)	74.40	
	$N = 25$	106.72	0.972 (0.005)	95.04	
	$N = 30$	148.75	0.965 (0.004)	71.65	
	$N = 35$	190.44	0.961 (0.005)	86.65	
	$N = 40$	373.84	0.955 (0.005)	97.08	
BOA	$N = 5$	0.04	0.933 (0.015)	68.82	allele convergence
	$N = 10$	0.08	0.939 (0.012)	20.08	
	$N = 15$	0.18	0.904 (0.014)	14.14	
	$N = 20$	0.50	0.883 (0.015)	16.12	
	$N = 25$	1.32	0.865 (0.012)	15.42	
	$N = 30$	2.68	0.851 (0.011)	12.48	
	$N = 35$	5.38	0.840 (0.009)	13.96	
	$N = 40$	8.08	0.832 (0.008)	14.60	

B. Comparison with benchmark EDAs

We set up similar experiments to evaluate DiG-SiRG's performance as compared to the selected EDAs. EDAs repeat the model building procedure in each generation, and the cost per generation is usually high (for multivariate EDA). Therefore, reaching the maximum number of generation is not a proper stop condition for runtime comparison. Instead, we set the algorithms to stop when the optimum is found (optimum is set to 0.99), or when the population has fully converged, i.e., when the population consists of n copies of the same individual, where n is the population size.

Table III shows the end-of-run statistics of each of the comparing methods as the degree of epistasis changes. The second column of Table III gives the average time to stop (in seconds) of a single run. The third column shows the average and the standard deviation of the best-found fitness values. Whenever the algorithm terminates due to the allele convergence, the forth column gives the average number of generations passed prior to the convergence.

As the results clearly show, while our proposed method can quickly find optimum in all the problem instances, EDAs suffer from premature convergence. UMDA has a very simple model building procedure, and thus, comparing to multivariate EDAs, the timing cost per generation is minimal. However, it performs poorly as the problem gets more difficult. ECGA is the best performing selected EDA in terms of the quality of the best found solutions prior to convergence. Its running time, however, is comparatively high, specifically for larger values of N. Compared to ECGA, BOA implements a faster model building procedure, but rapidly converges to a local optimum (in less than 21 generations, except for $N = 5$).

VII. SUMMARY

In this paper we proposed diversity-guided adaptive genetic algorithm which makes use of a site-specific mutation scheme enhanced by and entropy-based diversity control aimed to tackle both explorative and exploitative responsibilities of genetic operators. The proposed mutation scheme follows an approach similar to *motif representation* in biology, to derive the underlying pattern of highly-fit solutions discovered so far. This pattern is then used to derive mutation rates specified for every site along the encoded solutions. The site-specific rates are amended for every individual to balance the required explorative and exploitative power. Additionally, the mutation rates are reassigned at every generation to get adapted to the current state of the evolution.

Due to the space limitation, we only reported the performance of the proposed method over the MAX-SAT problem. DiG-SiRG, however, is a general-purpose (domain-independent) search algorithm which can be applied to various optimization problems. We, for instance, examined another problem from a totally different domain of genomic studies (visualization of functional relationships across disease loci), and similarly observed significantly higher performance when compared to the benchmark algorithms.

REFERENCES

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

[2] S. Tsutsui, A. Ghosh, D. Corne, and Y. Fujimoto, "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.

[3] Y. Y. Wong, K. H. Lee, K. S. Leung, and C. W. Ho, "A novel approach in parameter adaptation and diversity maintenance for genetic algorithms," *Soft Computing*, vol. 7, no. 8, pp. 506–515, 2003.

[4] M. Črepinšek, S. Liu, and M. Mernik, "Exploration and exploitation in evolutionary algorithms: a survey," *ACM Computing Surveys (CSUR)*, vol. 45, no. 3, p. 35, 2013.

[5] T. Back, "Selective pressure in evolutionary algorithms: A characterization of selection mechanisms," *In Proc. of the 1st Conf. on Evolutionary Computing*, 1994.

[6] E. Galván-López, J. McDermott, M. O'Neill, and A. Brabazon, "Towards an understanding of locality in genetic programming," *In Proc. of the 12th Annual Conf. on Genetic and Evolutionary Computation, ACM*, 2010.

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

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

[9] F. Vafaee, W. Xiao, P. C. Nelson, and C. Zhou, "Adaptively evolving probabilities of genetic operators," *In IEEE Proc. of Seventh Int. Conf. on Machine Learning and Applications*, vol. 3, pp. 292–299, 2008.

[10] W. J. Kruisselbrink, R. Li, E. Reehuis, J. Eggermont, and T. Back, "On the log-normal self-adaptation of the mutation rate in binary search spaces," *In Proc. of the 13th annual Conf. on Genetic and evolutionary computation (GECCO 11), ACM*, 2011.

[11] F. Vafaee and P. C. Nelson, "An explorative and exploitative mutation scheme," *IEEE Proc. of 2010 IEEE Congress on Evolutionary Computation (IEEE CEC 10)*, 2010.

[12] S. Y. Yuen and C. K. Chow, "A genetic algorithm that adaptively mutates and never revisits," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 454–472, April 2009.

[13] J. Cervantes and C. R. Stephens, "Limitations of existing mutation rate heuristics and how a rank GA overcomes them," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 369–397, 2009.

[14] S. Bottcher, B. Doerr, and F. Neumann, "Optimal fixed and adaptive mutation rates for the leadingones problem parallel problem solving from nature," *In Proc. of the 11th Int. Conf. on Parallel Problem Solving from Nature (PPSN XI), Springer*, 2010.

[15] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*. Berlin, Germany: Springer, 1996.

[16] M. Pelikan, D. E. Goldberg, and F. Lobo, "A survey of optimization by building and using probabilistic models," *Computational Optimization and Applications*, vol. 21, no. 1, pp. 5–20, 2002.

[17] M. Hauschild and M. Pelikan, "An introduction and survey of estimation of distribution algorithms," *Swarm and Evolutionary Computation*, vol. 1, no. 3, pp. 111–128, 2011.

[18] F. Vafaee, G. Turan, P. C. Nelson, and T. Y. Berger-Wolf, "Among-site rate variation: Adaptation of genetic algorithm mutation rates at each single site," *In ACM Proc. of 16th annual Conf. on Genetic and Evolutionary Computation (GECCO 14)*, 2014.

[19] M. K. Das and H.-K. Dai, "A survey of DNA motif finding algorithms," *BMC Bioinformatics*, p. 8(Suppl 7):S21, 2007.

[20] L.Masisi, V. Nelwamondo, and T. Marwala, "The use of entropy to measure structural diversity," *In Proc. of IEEE Int. Conf. on Computational Cybernetics*, 2008.

[21] C. E. Shannon, "A mathematical theory of communication," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 5, no. 1, pp. 3–55, 2001.

[22] K. A. DeJong, "Analysis of the behavior of a class of genetic adaptive systems," *Ph.D. thesis, University of Michigan, Ann Arbor, MI*, 1975.

[23] V. V. Vazirani, *Approximation Algorithms*. Springer, 2003.

[24] G. Smith, "Adaptive genetic algorithms and the boolean satisfiability problem," *Technical Report, University of Pittsburgh, Pittsburgh, PA*, 1979.

[25] S. Rana, R. B. Heckendorn, and D. Whitley, "A tractable walsh analysis of sat and its implications for genetic algorithms," *In Proc. of the 15th Conf. on Artificial intelligence/Innovative applications of artificial intelligence (AAAI 98), ACM*, 1998.

[26] R. B. Heckendorn, S. Rana, and D. Whitley, "Polynomial time summary statistics for a generalization of maxsat," *In Proc. of the annual Conf. on Genetic and evolutionary computation (GECCO 99), ACM*, 1999.

[27] S. Kauffman, *The Origins of Order: Self-Organisation and Selection in Evolution*. Oxford University Press, Oxford, UK, 1993.

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

[29] C. R. Reeves and J. E. Rowe, *Genetic Algorithms-Principles and Perspectives:A Guide to GA Theory*. Kluwer, 1995.

[30] K. A. DeJong and W. M. Spears, "Using genetic algorithms to solve NP-complete problems," *In Proc. of the third Int. Conf. on Genetic algorithms*, pp. 124–132, 1989.

[31] L. Lin and M. Gen, "Auto-tuning strategy for evolutionary algorithms: balancing between exploration and exploitation," *Soft Computing, Springer*, vol. 13, pp. 157–168, 2009.

[32] X. B. Hu and S. F. Wu, "A self-adaptive genetic algorithm based on fuzzy mechanism," *IEEE Congress on Evolutionary Computation*, pp. 4646–4652, 2007.

[33] N. Law and K. Y. Szeto, "Adaptive genetic algorithm with mutation and crossover matrices," *In Proc. of the 20th Int. joint Conf. on Artifical intelligence*, pp. 2330–2334, 2007.

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

[35] H. Mhlenbein and G. Paass, "From recombination of genes to the estimation of distributions i. binary parameters," *In Proceeding of the 4th Int. Conf. on Parallel Problem Solving from Nature (PPSN IV), Springer*, 1996.

[36] G. R. Harik, F. G. Lobo, and K. Sastry, "Linkage learning via probabilistic modeling in the ECGA," *IlliGAL Report 99010, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL*, 1999.

[37] ——, "Linkage learning via probabilistic modeling in the ECGA," *Extended compact genetic algorithm in C++ (version 1.1). IlliGAL Report No. 2006012, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL*, 2006.

[38] M. Pelikan, D. E. Goldberg, and E. Cantu-Paz, "Linkage problem, distribution estimation, and bayesian networks," *Evolutionary Computation 8*, 2000.

[39] M. Pelikan, "The bayesian optimization algorithm (boa) with decision graphs," *IlliGAL Report No. 2000025, University of Illinois at Urbana-Champaign, Illinois Genetic Algorithms Laboratory, Urbana, IL*, 2000.

[40] J. Bacardit, M. Stout, J. D. Hirst, K. Sastry, X. Llora, and N. Krasnogor, "Automated alphabet reduction method with evolutionary algorithms for protein structure prediction," *In Proc. of the 9th annual Conf. on Genetic and evolutionary computation, ACM*, 2007.

[41] P. Lipinski, "ECGA vs. BOA in discovering stock market trading experts," *In Proc. of the 9th annual Conf. on Genetic and evolutionary computation (GECCO 07), ACM*, 2007.