Send Orders for Reprints to reprints@benthamscience.ae

The Open Cybernetics & Systemics Journal, 2017, *11*, 1-12

BENTHAM OPEN

CrossMark

The Open Cybernetics & Systemics Journal

Content list available at: www.benthamopen.com/TOCSJ/

DOI: 10.2174/1874110X01711010001

RESEARCH ARTICLE

Brief Review of Techniques Used to Develop Adaptive Evolutionary Algorithms

José Alberto Bonilla-Vera, Jaime Mora-Vargas*, Miguel González-Mendoza, Iván Adrian López-Sánchez and César Jaime Montiel-Moctezuma

Escuela de Ingeniería y Ciencias, Tecnológico de Monterrey, Monterrey Area, Mexico

Received: October 07, 2015 Revised: August 24, 2016 Accepted: October 04, 2016

Abstract: This paper presents a brief review of techniques used to allow evolutionary algorithms to adapt to optimization problems in dynamic environments, through exploration of the control parameters of genetic algorithms as well as genotypic interpreters. A description of some of the most used evolutionary techniques is included, with major emphasis on genetic algorithms and their relationship with the problem of adaptation to the environment. The article also discusses state used models to tackle these kinds of problems, including self-adaptation and genotype- phenotype mapping.

Keywords: Adaptation, Evolutionary algorithms, Genetic algorithms, Genotype-phenotype mapping.

1. INTRODUCTION

NP-complete problems are hard enough to solve. However, in reality many problems not only pose a structural complexity in their underpinnings, but also present dynamism over time. The objective function is bound to change - stochastically or otherwise - and this situation lays additional difficulties to algorithms, because the search space variations further hinder their exploration and exploitation abilities. In this context, this paper presents some alternatives to cope with dynamic environ- ments in the setting of evolutionary computation, in particular, genetic algorithms. The alternatives considered in this paper due to their relevance are adaptation, self-adaptation and the mapping between individuals and their genetic information also known as genotype-phenotype mapping.

This paper presents a review of some of the most important works regarding adaptation, self-adaptation and the concept of representation. A brief overview of evolutionary computing is presented first, then in section 3 the details of origin, of adaptation, self-adaptation and genotype-phenotype mapping are presented, in section 4 it is presented reflections and con- siderations regarding performance of each of previos concepts discussed. In section 5 final remarks are presented.

2. EVOLUTIONARY ALGORITHMS

Research works from Darwin, Mendel, and Lamarck about genetics and biology are the foundation over which evolutionary computing has stemmed. Evolutionary computing is a group of techniques for problem solving that mimic the mechanisms found in nature. Evolutionary algorithms are some of the newest research fields in this area. A quick glance evolutionary algorithms (evolutionary strategies, evolutionary programming and genetic algorithms) is included below.

Evolutionary algorithms are a group of stochastic and iterative techniques for problem resolution, its main characteristic being that it is based on a population of feasible solutions instead of a single point. An evolutionary

* Address correspondence to this author at the Carretera Lago de Guadapule km 3.5. Colonia Margarita Maza de Juarez, CP 52926, Atizapan de Zaragoza, EdoMex, Mexico; Tel: +525558645957; Fax: +525558645969; E-mails: jmora@itesm.mx, j_mora_vargas@hotmail.com

algorithm uses iterative evolutionary mechanisms inspired in biology such as natural selection, reproduction, and mutation to generate new sets of individuals (*generations*), in hopes of producing candidates that converge to the problem solution.

2.1. Evolutionary Strategies

These strategies were invented by the pioneer of evolutionary computation, Ingo Rechenberg. Its main distinction is the use of real numbers for encoding individuals and the use of mutation as its main evolutionary operator. These techniques were mainly applied by Rechenberg himself to design aerodynamic surfaces.

2.2. Evolutionary Programming

They were invented by Lawrence J. Fogel and are based on finite state machines [1] and use the mutation operator to evolve their structure and behavior.

2.3. Genetic Algorithms

They are the most used and studied evolutionary meta-heuristics. The idea is to represent candidate solutions as a chromosome in which genes may have a series of expression values (alleles). The evolution is done by means of basic operators such as selection, in which the environment selects the fittest representatives (with respect to the objective function), crossover, in which genetic material is exchanged to form a new individual, and mutation, which changes the allele due to external factors.

These algorithms were originally proposed by John Henry Holland in 1975 and refined by several theorists, including David Goldberg.

2.4. Genetic Programming

It is an adaptation of genetic algorithms running over a predefined solution space consisting of computer source code. Programs themselves are represented using trees that can contain functions, operators and values. This technique was proposed by John Koza in 1990.

2.5. Taxonomies of Adaptation in Evolutionary Algorithms

Adaptation within evolutionary algorithms means, in a wide sense, incorporating feedback from the performance of the algorithm in a particular environment in order to modify parameter values or evolutionary mechanisms as a function of time. Hence, adaptation methods produce time-varying parameters that can operate on different levels. In this area, one of the the most known works is Hinterding *et al*. [1]. This paper classifies many adaptation methods using a 4×4 matrix, where X-Y represents the kind of adaptative method, X represents structure level and Y represents time level, using * when represent all levels, for example *-S (static level) represents E-S, P-S, I-S and C-S.

Table 1. Adaptative methods classification.

	Static	Deterministic	Adaptive	Self-Adaptive
Environment	S	E-D	E-A	E-SA
Population	S	P-D	P-A	P-SA
Individual	S	I-D	I-A	I-SA
Component	S	C-D	C-A	C-SA

In terms of adaptation mechanics, if the parameters are *static* (*-S) then its value is adjusted "manually" by the programmer. This is a complex task as it can affect the performance in a negative way, and is time consuming and does not guarantee good performance throughout the simulation. In *deterministic adaptation* (*-D), a heuristic rule is used to produce variation. This kind of method does not incorporate algorithm feedback, but instead relies on preset deterministic functions. In this case, the parameter change is generally a function of the number of generations executed. If the scheme is *adaptive* (*-A), the parameters are adjusted by incorporating feedback from the algorithm in the rule. Finally, *self-adaptive* mechanics (*-SA) imply that parameters themselves are encoded as a part of the genotype of the individuals, allowing evolutionary procedures to operate on parameter values. The hypothesis is that better parameter values will produce fitter individuals, which will in turn help propagate these parameter values among

[1] One can think of a finite state machine as a read-only Turing machine in which the head can traverse the tape from left to right.

future generations.

In terms of at what level adaptation occurs, Hinterding identifies four: *environment-level adaptation* (E-*), in which the fitness function changes as a result of niching or other considerations; population level (P-*), where parameter variations affect all individuals equally, or when several populations are used in parallel as a means of adapting parameters (such as mutation rates); at the individual level (I-*), when adjustments are performed at the individual level and resulting parameters apply only to that individual (this is the most prominent usage in self-adaptation); and finally at the component level (C-*), where the modified parameters are specific to a given gene of a more complex individual.

3. ADAPTATION, SELF-ADAPTATION AND GENOTYPE-PHENOTYPE MAPPING

This section summarizes a sample of the papers that have been produced regarding adaptation, self-adaptation and genotype-phenotype in the past twenty years.

3.1. Adaptation and Self-adaptation

While adaptation and self-adaptation have been explored over most of the genetic algorithm components, by far the most studied parameter has been the mutation rate. In fact, Ingo Rechenberg proposed the first deterministic adaptation rule for the mutation parameter, known as the $\frac{1}{5}$ *rule*. In essence, in each iteration of the algorithm, both the number of successful mutations (those which improve fitness) and the number of unsuccessful mutations (those which do not improve fitness) are counted. Rechenberg states that the optimal ratio of successful mutations must be approximately $\rho = \frac{1}{5}$, in such a way that, if the ratio of a particular generation is greater than ρ, then it is necessary to decrease the mutation rate μin order to allow exploitation; if, however, the ratio is below the mentioned threshold, mutation rate should be increased to incentive exploration of the solution space. Additionally, the same work proposes the idea of coupling the evolution of an individual with its corresponding mutation rate. This is the basic idea of self-adaptation as currently understood and as defined in this document.

In 1992, Thomas Back questioned whether previous works, which were focused in finding an "appropriate" (fixed) value for μ, were appropriate (such as Schaffer's proposal of $\mu = \frac{1.75}{N_0\sqrt{L}}$ where N is the number of individuals and L the length of the chromosome, and other authors' proposal of $\mu = \frac{1}{L}$).

Back states that a different mutation ratio per individual, encoded as part of the genotype, can yield better results, by exposing the parameter to evolutionary changes as a result of selection and genetic operators. Back compared a traditional, extinctive genetic algorithm with a self-adaptive mutation rate algorithm over different test functions and showed substantial improvements in algorithm performance.

Potential limitations of self-adaptation are also shown in works such as Rudolph [2], which shows that an elitist algorithm with Rechenberg's rule has premature convergence to a local optimum with positive probability, even using a infinite horizon of time, and proposes a modification that allows convergence to global optima.

A "canonical" version of a self-adaptive genetic algorithm is modeled and theoretically shown to converge using an extension of the Markov chain model in the work of Agapie [3].

Different features of self-adaptation in evolutionary strategies are explored in Boumaza [4]. The author also studies the correlation between movements of optima and adaptation of the mutation step. Also, this study reinforces how having a higher mutation parameter value in the exploration phase of the algorithm is more beneficial (to incentivize search of a wider portion of the space). In contrast, in the exploitation phase a lower mutation parameter value is desired to allow exploitation of better search directions.

Examples regarding self-adaptive genetic algorithms, where the performance is not improved (with respect to canonical genetic algorithms) on dynamic environments are presented in Rand and Riolo [5]. The changing environment is provided by a test suite called *Shaky Ladder Hyperplane-Defined Functions*. Several alternatives of the same model are explored; the first one, a lower bound for the mutation parameter is not established, allowing self-adaptation to gradually reduce the parameter value to zero leading to diversity loss. Even after imposing a lower bound and tweaking the algorithm, the self-adaptive model is consistently outperformed by the traditional fixed mutation genetic algorithm. The reasons cited by the authors revolve around the fact that a genetic algorithm in a dynamic environment should be able to select among high and low μ values along the execution; in average if μ is large, it is

possible that the algorithm finds an individual with a low mutation rate, and if that individual has good fitness, then it will probably generate more individuals with low μ, decreasing the overall mutation rate average; however, it is improbable that an algorithm with low μ could generate *many* individuals with high probability of mutation and hence transition from exploitation to exploration easily (such as in the case when the algorithm is atop a local optimum). A suggestion provided by the authors is to tweak self-adaptation in order to be able to produce high values of μ at different stages of the execution process (for example, having a Boolean allele representing one of two possible mutation rates).

Similar results that those shown by Rand and Riolo [5] in terms of the hurdles of self adaptation to improve the performance of the algorithms on certain environments are observed in Breukelaar [6]. Breukelaar reached the same conclusion and indicated that mutation rate should be allowed to be increased or decreased with equal probability.

Examples of self-adaptation for crossover can be found in Kramer and Koch [7] where the authors develop versions of commonly used crossover operators with self-adaptive control parameters. A propose of the use of a crossover operator named *simulated binary reproduction* (SBX) on real genetic algorithms is presented by Deb and Beyer [8]. The purpose is to obtain similar results as evolutionary strategies, but using genetic algorithms. In this case, if parents are away from each other (with respect to the Euclidean distance) then there is a high probability that offspring will also be distant from each other (and *vice versa*). The operator is in function of a parameter $\eta > 0$. If η is big, the probability of generating similar children is very high. Using this self-adaptive operator, diversity of offspring solutions is controlled by diversity of parents.

The distinction between adaptation and self-adaptation explained in the context of the crossover operator is studied in Esquivel *et al.* [9]. The basic hypothesis of self-adaptation is that counting with fitter solutions also produces good values in the parameters. That work uses a genetic operator known as *multiple crosses per couple* (MCPC) in which the number of crossovers allowed per individual is encoded in the chromosome. As conclusion, this paper shows benefits over several test functions in terms of performance when self-adaptive number of crossovers is used compared to a fixed number of crossovers.

Similar results are obtained *via* the design of self-adaptive crossover operator, specifying rules such as preservation of the statistical moments of the population distribution and the degree of diversity in future offspring in Kita *et al*. [10]. As result of this work, *Uniformly unimodal distribution crossover*(UNDX) is defined, exhibiting features comparable with those of evolutionary strategies over a range of test functions.

The size of the population has been also subjected to self- adaptation. The role of population in the performance of the algorithms: a small size does not allow enough diversity, producing premature convergence is shown in Lobo and Lima [11]; also this study shows that a large populations hinder the computational efficiency of the algorithm without significant gain in performance; furthermore, if the algorithm does not consider mutation, then the genetic diversity is produced entirely by the population. The authors explore different schemes and produce a series of recommendations regarding adaptive population adjustments.

Adaptation schemes have also been explored in the context of distributed genetic algorithms. In particular in Bemtsson and Tang [12] a model based on three genetic algorithms in parallel, one denoted *DGA* with n islands of size d, another called DGA_1 with $2n$ islands of size d and another called DGA_2 with n islands of size $2d$. If the algorithm *DGA* converges, execution is terminated; otherwise, the parameters of *DGA* are tweaked based on the parameters of the winning algorithm, and the fittest individuals are seeded into the remaining algorithms for reinitialization.

A treatise on how selection, mutation, and crossover benefit the population diversity when they become adaptive is developed in McGinley *et al*. [13]. The *population diversity* measure is formally defined to be equal to the empiric coefficient of determination of the population fitness, and *weighted population diversity*, which takes into account the relative aptitude of each phenotype. The results indicate that the adaptive concept is useful mainly to improve the exploration phase of the search.

The issue of *-A adaptation can be reviewed in full detail in several other articles, such as Uyar and Eryigit [14], where an adaptive evolutionary strategy (over the mutation parameter) is used in a context of constrained optimization. The algorithm contains two parameters of mutation, one for 0-valued *loci* and the other for 1-valued *loci*, and they adapt with respect to the number of successful individuals containing those particular features in each *loci*. The contribution of Whitacre *et al*. [15, 16] lies in the exploration of different crossover operators that are selected by an adaptive mechanism.

Other works that can be reviewed related to these topics are Pytel *et al.* [17] and Tang *et al.* [18], Tang focus on population diversity on Island Model Parallel Memetic Algorithm, consider the use of online entropy measure to adapting the local search frequency of parallel memetic algorithm, the dynamic local search frequency can be defined in the online entropy ratio given by the population entropy measure by generation. Hansen [19] formulate a real code evolutionary algorithm with truncation selection and with self-adaptive using a mutative strategy parameter control of one global step-size to lead a point symmetrical distribution of the complete population before the selection operator.

Wickramasinghe *et al.* [20] worked in an autonomous selection for individuals to lead the survival and reproduction independently, without any central control, adding adding an adaptation mechanism that allow individuals regulate their own selection pressure. Curran [21] focuses on genomes for evolutionary algorithms, he defined three genomes: one containing the solution to the NK landscape, one encoding imitation probability and, finally, one encoding teaching rounds; all three genomes are allowed to undergo the processes of crossover and mutation. Rossi *et al.* [22] worked in two mutation operators, the first one, Kmut-N mutation operator, sets the values of the gens ignoring their previous values and the second one, Kmut-P mutation, is through a gaussian perturbation, this generate values close to an estimation, consider two terms, the fist one is the perturbation direction and the second one is the perturbation range [23].

Wang *et al.* [44] used an improved immune genetic algorithm to optimize the parameters in a Support Vector Machine to solve landscape design that refers to an independent profession and a design and art tradition combining nature and culture. Qingyang *et al.* [24] worked on an adaptive learning rate elitism Estimation of Distribution Algorithm (EDA), a kind of Evolutionary Algorithm, combining chaos perturbation (ALREEDA) to improve the performance of traditional EDA to solve high dimensional optimization problems. Chen-Yang *et al.* [25] focused on a hybrid algorithm called Hybrid Algorithm-Ant Colony Algorithm Genetic Algorithm (HA- ACAGA) to find optimal solutions for users on dynamic web service composition where user's personal preference is different and web services are massive and dynamic; they used a function to control individuals and a function to update pheromones.

Desirable properties for selecting different genetic operators are expained by Thierens [26]. The conditions presented are:

1. If several operators are available, they should all have a positive probability of being selected at any stage of the process.
2. To maximize the performance of the search, one must apply the best operator possible in a given step, such that the previous property still holds.
3. When a new operator becomes the best available one, operator selection probabilities must be updated as soon as possible to reflect this.
4. When all operators are performing similarly, their selection probabilities should be equal.

Finally, two comprehensive reviews in the field of evolutionary adaptation can be found in Thierens [27] and in Meyer-Nieberg and Beye [28], these works are compared in the Table **2** in the Appendix.

3.2. Representation and Genotype-Phenotype Mapping

The main idea around genotype-phenotype mapping revolves around the fact that the vast majority of changes at the genotype level are phenotypically *neutral*, *i.e.*, several genotypes encode the same phenotype. This observation is based on the biomolecular research of Mooto Kimura in 1968 and later works by Banzhaf in 1994. This result is very important in the field of the genetics and evolutionary theory and it is known as the *neutral theory of molecular evolution*.

The applications of non-trivial genotype-phenotype mappings in evolutionary computing were first used in genetic programming. In Korejo and Yang [29], where, in the context of creating valid programs as individuals, instead of forcing closure at the time of individual generation and evolution, the use of a genotype- phenotype mapping which repairs genotypes containing invalid syntax with operators or functions that are nearby to that individual in order to conform a valid program (with respect to the Hamming distance, for example), is explored with favorable results.

A discussion about the fact that evolutionary search is influenced by the environment is discussed in Keller and Banzhaf [30]. The authors explain the existence in nature of synonym codons that produce the same phenotype, but vary greatly in their mutability. Thus, having synonyms in the genotype population (in order to be able to perform genetic operators over them) allows for better adaptation to the environment.

The role of the genotype-phenotype mapping in genetic programming is explored in the context of information recovery in Vargas *et al*. [31]. The performance of a genetic program with a binary tree and a genetic algorithm with a non trivial genotype-phenotype mapping is compared. In simple problems, both algorithms are comparable, but in more complex instances the genetic algorithm outperforms the canonical genetic program; this is attributed to a better ability to escape from local optima.

The role of the synonyms is analyzed in more detail by Martn and Shackleto [32]. This bibliographic review argues that not all types of redundant representations are useful in the context of the performance of evolutionary algorithms. A representation is defined like *synonymously redundant* if the genotypes associated to same phenotype are similar. This concept is formally defined in terms of the sum of all distances (relative to a metric d (x_1, x_2)) between pairs of genotypes obtained from the set of genotypes that codifies a particular phenotype. Also, if every phenotype is represented by the same number of genotypes, it is said that the representation is *uniformly redundant*. Lastly, two genotypes are *neighbors* if their distance is the minimum possible (a small fixed number). If neighboring genotypes encode neighboring phenotypes, it is said that the representation has *high locality* (a kind of "continuity" in the representation function). It is shown that simple problems solved with a genetic algorithm with low locality yield bad performance; also, genetic operators do not seem to work appropriately if the representation is not synonymously redundant. The paper comes to the conclusion that non injective genotype-phenotype mappings are useful when the optima are over-represented in the genotype space corresponding to the initial population (and *vice versa*). Finally, if no prior information about the problem is available, a uniform, synonymously redundant representation is recommended.

Another example of redundant mappings that improve performance of genetic algorithms is shown in Rothlauf and Goldberg [33]. This paper explores three different representations for the same scheduling problem in real-time parallel applications, which allow failing to meet certain deadlines or deliverables. A particular non injective representation presents better performance than others. An adaptation of the genotype-phenotype mapping under the context of the project scheduling problem with resource constrains is presented in Dandass and Bugde [34]. By using a *locus* that indicates one of two possible specific individual mapping functions within the individuals genome. As a result, the paper shows performance improvement on several other heuristics mentioned in the literature, such as simulated annealing, traditional genetic algorithms, and tabu search.

A solution to the problem of optimizing investment portfolios is exhibited using a novel representation based on threes (inspired by concepts of genetic programming representations) instead of a classical array-based representation which models the weights of assets in the portfolio is proposed in Hartmann *et al*. [35]. This work shows that using tree representation improves the solution quality.

An adaptive mechanism of selecting a representation is explored in Aranha and Iba [36]. The concept developed is denoted *states- based evolutionary algorithm* (SEA). The algorithm chooses variants of a particular representation over time. Execution of a two-state SEA shows improvement in algorithm results for some test functions and the authors assert that changing the representation during the search will yield better performance than fixed representations.

People who work with hybrid algorithms using phenotype mapping, Ning *et al.* [37] and Ning *et al*. [38] worked with a hybrid algorithm called improved double chains Quantum Genetic Algorithm which realized the diversity and parallelism of the population through replacing the chromosome coding with quantum bits probability amplitude and searching with the quantum gates to solve problems such as Flexible Job-Shop scheduling problem and Vehicle Routing Problem.

Finally, some other research papers of interest related to the problem of phenotype representations can be found, for example, in Ning *et al.* [39] where they used three different neural network representations and are compared using a coevolutionary setup, these representations are: a direct weighted mapping of input features to a heuristic value, a complexifying neural network using NeuroEvolution of Augmenting Topologies (NEAT), and an implicit representation based on properties of genetic regulatory networks. Other work is Reisinger and Miikkulainen [40] where they used a new class of representations for real valued parameters called Center of Mass Encoding (CoME). CoME is based on variable length strings and it allows the choice of redundancy degree of the genotype-phenotype map and the choice of redundancy distribution for the space of phenotypes. In the Appendix, Table **3** shows a compilation of research about genotype-phenotype mapping and a comparison between them.

4. REFLECTIONS ABOUT PERFORMANCE IN TIME-VARYING ENVIRONMENTS

Bibliographic review of adaptation and self-adaptation seems to indicate mixed results regarding performance of

modified evolutionary algorithms in dynamic environments. There is still doubt regarding the particular aspects and mechanics of a self-adaptive algorithm that yield better performance than a traditional genetic algorithm, and whether the proposed mechanism is robust enough to detect changes in the solution space.

Most likely, the prime feature to be analyzed in order to be successful using a self-adaptive algorithm is appropriate balance between exploration and exploitation phases during the search. During exploration, it is desirable that the algorithm generates sufficient diversity in order to fully explore search directions; the initial population plays an important role as well as the mutation parameter in this stage. In general, it can be observed that a larger initial μ is preferred. During exploitation phases, the algorithm has already found a sequence of solution candidates, and it seeks to refine those candidates to find the (quasi) optimal solution. In this phase, the crossover operator seems to obtain relevance, and decreasing (but not eliminating) variability of the search by means of some mechanism is needed such as decreasing the value of μ. As it was pointed out in Rand and Riolo [5] and Breukelaar and Back [6], it is important that the algorithm can still generate certain number of candidates away from the currently explored peak. This reduces the probability of being stuck in local optima.

In the context of time dynamic problems, the role of the mutation seems to take special importance. During the search, the algorithm needs to switch between exploration and exploitation multiple times, and it must react fast enough to avoid being stuck. The parameters that control genetic diversity (such as μ) need a lower bound that can prevent decrease of genetic diversity that can lead to premature convergence.

The selection mechanism can affect the algorithm performance as well. Selective pressure can lead to premature convergence. This is especially true for extinctive selection mechanism, since ergodicity of the process can be lost.

The role of the representation and genotype-phenotype mapping seems to be determinant for a good performance. It should favor high locality and synonymic redundancy. Performance of synonymously redundant representations might be enhanced by seeding the initial population with genotypes that represent "best estimates" or *a prior* guesses of the global optimum.

Some articles seem to lead to performance improvements when several variants of a particular genetic operator (such as crossover), or a particular representation, exist, and selection of which variant to use on each generation and/or individual is made *via* adaptive or self-adaptive mechanisms. For this purpose, it seems important to ensure "diversity" of these operators or representations through positive probabilities of selection at any stage of the process, as well as probabilities that reflect corresponding fitness gains or losses by using each variant.

FINAL REMARKS

After reviewing and analyzing the state of the art, there is still wide room for research both theoretical and experimental regarding self-adaptive mechanisms and representations for genetic algorithms in order to consistently solve hard problems, and particularly problems with time-varying fitness landscapes. Many research lines can originate from the review of the bibliography.

The classification developed in mentioned works is too complete because it considers the characteristics of the parameters, and with the behavior of the values, and it classifies an algorithm according to adaptation type (deterministic, static, adaptive, self-adaptive) and adaptation level(environment, population, individually or component).

Adaptation and self-adaptation have been explored over the majority of genetic algorithm components, but these mechanisms do not have relevant improvements on all of them. The most considerable improvement is in mutation and crossover operations, where rules or new genetic operators are specified to preserve the population distribution and a degree of diversity in future offspring.

Adaptive and self-adaptive mechanisms have an advantage in the parameters that control diversity in individuals in different process stages, to assure an exhaustive explorative search; and to adapt the results generated in time-dynamic environments. Some disadvantages involve problems such as operator selection and the accurate adjustment in settings for each operator, because if the value found by the mechanism is too low, such as mutation operator, it could not ensure the diversity in whole process.

Genotype-phenotype mapping in simple problems is comparable with other algorithms, but in more complex instances is more efficient; regarding synonymous concept, not all types of redundant representations are useful. It is necessary to have high locality to obtain good results, because in simple problems using genetic algorithm with low locality always yield bad performance. If no prior information about the problem is available, it is recommendable to

define synonymously redundant representations; these representations might be enhanced by seeding the initial population with genotypes that represent prior guesses of the global optimum.

There are some hints to implement these mechanisms in a proper manner such as clearly defining the balance between explorative and exploitative searches in all stages of the algorithm, to bring off enough diversity to fully explore the solution space, and to decrease the probabilities of being stuck in local optima increasing the number of individuals far away from the neighborhood where the exploitative search is taking place. In the case of several operators being available, it is necessary consider that all should have a positive probability of being selected at any stage. One of them must apply the best operator possible in a given step to maximize performance. When an operator becomes the best available, operator selection probabilities must be updated, and when all operators are performing similarly, their selection probabilities should be equal.

Some research lines can be the combination of two or more mechanisms to modify several operators in the algorithm; to work in mechanisms to adapt others operators instead of mutations and crossover, combining mechanisms working together in different adaptation levels at the same time. The genotype phenotype mapping has a favorable trend with improvements in settings to obtain high locality in representations of genotypes and phenotypes neighbors, and to define a synonymously redundant representation.

CONFLICT OF INTEREST

The authors confirm that this article content has no conflict of interest.

ACKNOWLEDGEMENTS

Declared none.

APPENDIX

Table 2. Comparative table of adaptive self-adaptive strategies.

Technique	Algorithm	Operator	Features
1/5 Rule	GA	Mutation	-Successful mutations must be p=1/5 in a particular generation
Fixed M [41]	GA	Mutation	-Based on 1/5 Rule
Modification Fixed M [2]	GA	Mutation	-Based on Fixed M
Canonical version [3]	GA	Mutation	-Using markov chain model to converge
Self-Adaptation features [4]	GA	Mutation	Exploration phase is more beneficial high mutation Exploitation phase is more beneficial low mutation
Shaky ladder Hyperplane [5, 42, 43] Defined Functions	GA	Mutation	Reduce value to zero of mutation parameter Bolean allele represent one or two possible mutation rates Produce high values of mutation at different generations
Based on Shaky ladder [6]	GA	Mutation	Equal probability to allow mutation on different generations
Evolutionary mutation control parameter [19]	EA	Mutation	Real code evolutionary algorithm with truncation selection and with self-adaptive
Kmut-N and Kmut-P [22]	EA	Mutation	-Kmut-N mutation operator sets the values of the gens ignoring their previous values -Kmut-p mutation is a gaussian perturbation
Simulated Binary Reproduction (SBX) [7]	GA	Crossover	Crossover operator with self adaptive control parameters
SBX [8]	GA	Crossover	Produce offspring between parents using euclidian distance
Multiple Crosses per Couple [9]	GA	Crossover	Number of crossovers allowed per individual is encoded in the chromosome
Uniformly unimodal distribution crossover [10]	GA	Crossover	Rules such as preservation of statistical moments of the population distribution and degree of diversity in future offsprings
HA-ACAGA [25]	GA	Population	-Use a function to control individuals -Extra amount of pheromones are deposited on path found
Memetic Algorithm parameter control [18]	MA	Population	Define a parameter to control diversity in the population for Parallel memetic algorithm

(Table 2) contd.....

Technique	Algorithm	Operator	Features
Adaptation in population [20]	**EA**	**Selection**	**-Regarding survival and reproduction are taken by the individuals themselves independently, without any central control. -Adding an adaptation mechanism allowing individuals to regulate their own selection pressure -Algorithm enables individuals to maintain estimates on the size and the fitness of the population**
Immune genetic algorithm [44]	**GA**	**Algorithm**	**-Describe solutions using symbolic coding and full binary tree in the chromosomes -Cross point can be selected from the intermediate nodes and the root nodes -Mutation operator is modified**
Adaptive paramters for EAs [23]	**EA**	**Algorithm**	**-is based on micropopulation evolutionary algorithm -main mechanims: elitism and adaptive behaviour -mechanism mixed on mutation, crossover and replacement operators -3 adaptive paramters: ambient pressure (related with population), step size por mutation operator (related with the number of variables) and crossover balance**
Estimation of Distribution Algorithm [24]	**EA**	**Learning Rate**	**-Adaptive learning rate with different learning rule -Chaos perturbation and elitistm strategy**
Different genomes encoding by individuals [21]	**EA**	**Imitation probability**	**-Each individual carries three genomes: one containing the solution to the NK landscape, one encoding imitation probability and, finally, one encoding teaching rounds -All three genomes are allowed to undergo the processes of crossover and mutation.**

Table 3. Genotype-phenotype mapping research table.

Technique	Algorithm	Features
Keller [29]	Genetic Programming	Use of a genotype-phenotype mapping which repairs genotypes containing invalid syntax
Fernandez-VillicanasMartin [31]	Genetic Programming	Fernandez-VillicanasMartin [31] Genetic Programming Genotype-phenotype mapping in genetic programming is explored in the context of information recovery
CastroAranha [35]	Genetic Programming	Using tree representation improves the solution quality for optimizing investment portfolios problem
Mora [30]	Genetic Algorithm	Evolutionary search is influenced by the environment Existence in nature of synonym codons that produce the same phenotype, but vary greatly in their mutability Having synonyms in the genotype population allows better adaptation to the environment
Rothlauf [32]	Genetic Algorithm	Not all types of redundant representations of synonyms are useful in the context of the performance of evolutionary algorithms Non injective genotype-phenotype mappings are useful when the optima are over-represented in the genotype space corresponding to the initial population
Dandass [33]	Genetic Algorithm	Explores three different representations for the same scheduling problem in real-time parallel applications Non injective representation presents better performance than others realized
TaoNing [37]	Genetic Algorithm	The diversity and parallelism of the population through replacing the chromosome coding with quantum bits probability amplitude and searching with the quantum gates
Hartman [34]	Genetic Algorithm,Simulated Annealing and Tabu search	Indicates one of two possible specific individual mapping functions within the the individuals genome
Bercachi [36]	States-Based Evolutionary Algorithm	Chooses variants of a particular representation over time, changing the representation during the search will yield better performance than fixed representations
Reisinger [39]	Evolutionary Algorithm	Use of three different neural network representations
Mattiussi [40]	Evolutionary Algorithm	A new class of representations for real valued parameters called Center of Mass Encoding

REFERENCES

[1] R. Hinterding, Z. Michalewicz, and A. Eiben, "Adaptation in evolutionary computation: A survey", In: *Proceedings of the IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence, IEEECEP,* 1997, pp. 65-69. [http://dx.doi.org/10.1109/ICEC.1997.592270]

[2] G. Rudolph, "Self-adaptive mutations may lead to premature conver- gence", *IEEE Trans. Evol. Comput.,* vol. 5, no. 4, pp. 410-414, 2001. [http://dx.doi.org/10.1109/4235.942534]

[3] A. Agapie, "Theoretical analysis of mutation-adaptive evolutionary algorithms", *Evol. Comput.,* vol. 9, no. 2, pp. 127-146, 2001. [http://dx.doi.org/10.1162/106365601750190370] [PMID: 11382353]

[4] A.M. Boumaza, "Learning environment dynamics from self-adaptation: a preliminary investigation", In: *Proceedings of the 7th Annual Workshop on Genetic and Evolutionary Computation,* Washington, D.C. USA, 2005, pp. 48-54. [http://dx.doi.org/10.1145/1102256.1102265]

[5] W. Rand, and R.L. Riolo, "The problem with a self-adaptative mutation rate in some environments: A Case Study Using The Shaky Ladder Hyperplane-Defined Functions", In: *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation,* Washington, D.C. USA, 2005, pp. 1493-1500. [http://dx.doi.org/10.1145/1068009.1068245]

[6] R. Breukelaar, and T. Bäck, "Self-adaptive mutation rates in genetic algorithm for inverse design of cellular automata", In: *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation,* Atlanta, GA, USA, 2008, pp. 1101-1102. [http://dx.doi.org/10.1145/1389095.1389298]

[7] O. Kramer, and P. Koch, "Self-adaptive partially mapped crossover", In: *Proceedings of the 9th Annual Conference on Genetic and Evolutionery, Computation,* ACM, New Yorks, 2007, p. 1523.

[8] K. Deb, and H.G. Beyer, "Self-adaptation in real-parameter genetic algorithms with simulated binary crossover", In: *Proceedings of the 1st Annual Conference on Genetic and Evolutionery, Computation,* ACM, Orlando, Florida, 1999, pp. 172-179.

[9] S.C. Esquivel, H.A. Leiva, and R.H. Gallard, "Self adaptation of parameters for mcpc in genetic algorithms", *J. Comput. Sci. Technol.,* vol. 2, pp. 1-8, 2000.

[10] H. Kita, "A comparison study of self-adaptation in evolution strategies and real-coded genetic algorithms", *Evol. Comput.,* vol. 9, no. 2, pp. 223-241, 2001. [http://dx.doi.org/10.1162/106365601750190415] [PMID: 11382357]

[11] F.G. Lobo, and C.F. Lima, "A review of adaptive population sizing schemes in genetic algorithms", In: *Proceedings of the 7th Annual Workshop on Genetic and Evolutionary Computation,* Washington, D.C. USA, 2005, pp. 228-234. [http://dx.doi.org/10.1145/1102256.1102310]

[12] J. Berntsson, and M. Tang, "Adaptive sizing of populations and number of islands in distributed genetic algorithms", In: *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation,* Washington, D.C. USA, 2005, pp. 1575-1576. [http://dx.doi.org/10.1145/1068009.1068266]

[13] B. McGinley, F. Morgan, and C. O'Riordan, "Maintaining diversity through adaptive selection, crossover and mutation", In: *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation,* Atlanta, GA, USA, 2008, pp. 1127-1128.

[14] S. Uyar, and G. Eryigit, "Improvements to penalty-based evolutionary algorithms for the multi-dimensional knapsack problem using a gene-based adaptive mutation approach", In: *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation,* Washington, D.C. USA, 2005, pp. 1257-1264. [http://dx.doi.org/10.1145/1068009.1068214]

[15] J.M. Whitacre, T.Q. Pham, and R.A. Sarker, "Credit assignment in adaptive evolutionary algorithms", In: *CoRR,* vol. abs/0907.0592. 2009, pp. 1-4.

[16] J.M. Whitacre, T.Q. Pham, and R.A. Sarker, "Use of statistical outlier detection method in adaptive evolutionary algorithms", In: *CoRR,* vol. abs/0907.0595. 2009, pp. 1-4.

[17] K. Pytel, and T. Nawarycz, *Genetic Algorithm with the Adaptation of the Probability of the Mutation.,* vol. 1. Academy of Human Ecology: Lodz, Poland, 2007, pp. 1-9.

[18] J. Tang, M-H. Lim, and Y-S. Ong, "Adaptation for parallel memetic algorithm based on population entropy", In: *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation,* Seattle, Washington, USA, 2006, pp. 575-582. [http://dx.doi.org/10.1145/1143997.1144100]

[19] N. Hansen, "An analysis of mutative sigma-self-adaptation on linear fitness functions", *Evol. Comput.,* vol. 14, no. 3, pp. 255-275, 2006. [http://dx.doi.org/10.1162/evco.2006.14.3.255] [PMID: 16903793]

[20] W.R. Wickramasinghe, M. van Steen, and A.E. Eiben, "Peer- to-peer evolutionary algorithms with adaptive autonomous selection", In: *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation,* London, England, 2007, pp. 1460-1467. [http://dx.doi.org/10.1145/1276958.1277225]

[21] D. Curran, C. O'Riordan, and H. Sorensen, "Self-adaptation of cultural learning parameters", In: *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation,* ACM, New York, 2007, p. 338.

[22] C. Rossi, A. Barrientos, and J. del Cerro, "Two adaptive mutation operators for optima tracking in dynamic optimization problems with evolution strategies", In: *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation,* London, England, 2007, pp. 697-704.
[http://dx.doi.org/10.1145/1276958.1277102]

[23] F. Viveros Jimnez, E. Mezura-Montes, and A.F. Gelbukh, "Adaptive evolution: an efficient heuristic for global optimization", In: *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation,* Montreal, Québec, Canada, 2009, pp. 1827-1828.
[http://dx.doi.org/10.1145/1569901.1570184]

[24] W. Yao-Kuan, and L. Yu-Hong, "An improved immune genetic algorithm and its application in computer aided landscape design", *Open Cyber. Syst. J.,* vol. 8, pp. 1082-1090, 2014.
[http://dx.doi.org/10.2174/1874110X014080101082]

[25] J.S. Qingyang Xu, C. Zhang, and L. Zhang, "Adaptive learning rate elitism estimation of distribution algorithm combining chaos perturbation for large scale optimization", *Open Cyber. Syst. J.,* vol. 10, pp. 20-40, 2016.
[http://dx.doi.org/10.2174/1874110X01610010020]

[26] J.Q. Chen-Yang Zhao, J-L. Wang, and W-Q. Zhang, "A hybrid algorithm combining ant colony algorithm and genetic algorithm for dynamic web service composition", *Open Cyber. Syst. J.,* vol. 8, pp. 146-154, 2014.
[http://dx.doi.org/10.2174/1874110X01408010146]

[27] D. Thierens, "On benchmark properties for adaptive operator selection", In: *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers,* Montreal, Québec, Canada, 2009, pp. 2217-2218.
[http://dx.doi.org/10.1145/1570256.1570306]

[28] S. Meyer-Nieberg, and H-G. Beyer, "Self-adaptation in evolutionary algorithms", In: *Parameter Setting in Evolutionary Algorithms.,* Springer, 2007, pp. 47-75.
[http://dx.doi.org/10.1007/978-3-540-69432-8_3]

[29] I. Korejo, S. Yang, and L. Changhe, "A comparative study of adaptive mutation operators for genetic algorithms", In: *The VIII Metaheuristics International Conference,* Hamburg, Germany, MIC 2009.

[30] R.E. Keller, and W. Banzhaf, "Genetic programming using genotype- phenotype mapping from linear genomes into linear phenotypes", In: *Proceedings of the 1st Annual Conference on Genetic Programming,* MIT Press: Cambridge, MA, USA, 1996, pp. 116-122.

[31] J.M. Vargas, C.R. Stephens, H. Waelbroeck, and F. Zertuche, "Symmetry breaking and adaptation: evidence from a toy model of a virus", *Biosystems,* vol. 51, no. 1, pp. 1-14, 1999.
[http://dx.doi.org/10.1016/S0303-2647(98)00093-8] [PMID: 10426468]

[32] J.L. Fernndez Villicaas Martn, and M. Shackleton, "Investigation of the importance of the genotype-phenotype mapping in information retrieval", *Future Gener. Comput. Syst.,* vol. 19, no. 1, pp. 55-68, 2003.
[http://dx.doi.org/10.1016/S0167-739X(02)00108-5]

[33] F. Rothlauf, and D.E. Goldberg, "Redundant representations in evolutionary computation", *Evol. Comput.,* vol. 11, no. 4, pp. 381-415, 2003.
[http://dx.doi.org/10.1162/106365603322519288] [PMID: 14629864]

[34] Y.S. Dandass, and A.C. Bugde, "Comparison of genetic representation schemes for scheduling soft real-time parallel applications", In: *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation,* Seattle, Washington, USA, 2006, pp. 523-530.
[http://dx.doi.org/10.1145/1143997.1144093]

[35] S. Hartmann, "A self-adapting genetic algorithm for project scheduling under resource constraints", *Naval Res. Logis.,* vol. 49, pp. 433-448, 2002.
[http://dx.doi.org/10.1002/nav.10029]

[36] C. de Castro Aranha, and H. Iba, "A tree-based ga representation for the portfolio optimization problem", In: *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation,* Atlanta, GA, USA, 2008, pp. 873-880.

[37] M. Bercachi, P. Collard, M. Clergue, and S. Ve'rel, "Do not choose representation just change: an experimental study in states based ea", In: *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation,* Montreal, Québec, Canada, 2009, pp. 1799-1800.
[http://dx.doi.org/10.1145/1569901.1570168]

[38] R.C. Tao Ning, C. Guo, and H. Jin, "A novel hybrid method on vrp with pickup and delivery", *Open Cyber. Syst. J.,* vol. 10, pp. 56-60, 2016.
[http://dx.doi.org/10.2174/1874110X01610010056]

[39] R.C. Tao Ning, C. Guo, and H. Jin, "A novel hybrid method for solving flexible job-shop scheduling problem", *Open Cyber. Syst. J.,* vol. 10, pp. 13-19, 2016.
[http://dx.doi.org/10.2174/1874110X01610010013]

[40] J. Reisinger, and R. Miikkulainen, "Acquiring evolvability through adaptive representations", In: *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation,* London, England, 2007, pp. 1045-1052.

[41] C. Mattiussi, P. D¨urr, and D. Floreano, "Center of mass encoding: a self-adaptive representation with adjustable redundancy for real-valued parameters", In: *Proceedings of the Genetic and Evolutional Computation Conference, GECCO2007,* University College, London, New York: ACM Press, 2007, pp. 1304-1311.

[42] T. Bck, "Self-adaptation in genetic algorithms", In: *Proceedings of the 1st European Conference on Artificial Life,* MIT Press: Cambridge,

MA, USA, 1992, pp. 263-271.

[43] W. Rand, "*Controlled Observations of the Genetic Algorithm In a Changing Environment: Case Studies Using the Shaky Ladder Hyperplane-Defined Functions.* Dissertation, University of Michigan, Michigan", 2005.

[44] W. Rand, and R.L. Riolo, "Measurements for understanding the behavior of the genetic algorithm in dynamic environments", In: *Proceedings of the 7th Annual Workshop on Genetic and Evolutionary Computation,* ACM, New York, 2005.