AMLGA: A Genetic Algorithm with Adaptive and Memory-Assisted Local Operators

Sumaiya Iqbal and Md Tamjidul Hoque, Computer Science, University of New Orleans, LA 70148.

***Abstract*— Genetic algorithm (GA) is a popular population-based stochastic search technique that mimics the natural process of evolution. The GA with chromosomes encoded using binary alphabet operates by discovering, prioritizing and recombining good building blocks or schema (a binary string) that can confer higher fitness to a chromosome. Crossover is considered as the heart of GA to perform recombination of good parent chromosomes to produce better offsprings. With the aim of improving the performance of crossover, we propose a novel mechanism to perform crossover operation locally in gene level. We store good local gene schema using a memory and emphasize on it at the time of crossover. We perform this memory-assisted local crossover in an adaptive manner that takes feedback from the search process on the memory as well the current population while producing improved off-springs. We further include a recently proposed adaptive gene replacement operator that boosts up the elites globally with their homologues good gene schema in the proposed algorithm. We call the new Genetic Algorithm with Adaptive and Memory-assisted Local operators as AMLGA. We further improved the basic AMLGA by strategically reducing the number of computations of the objective function while not affecting the performance. AMLGA is rigorously evaluated on a set of 40 bound-constrained benchmark numerical optimization problems. On a reduced set of 27 test functions, AMLGA outperformed 7 GA variants. It further showed competitive performance with two other state-of-the-art evolutionary algorithms on the full set. With the fast-growing size and complexity of modern optimization problems, we believe our evolutionary algorithm will continue to be attractive as an efficient tool for optimization.**

***Index Terms*—Evolutionary computing, Genetic Algorithm, Local operators, Adaptive operator, Memory-assisted operator, Benchmark test function.**

This paragraph of the first footnote will contain the date on which you submitted your paper for review. It will also contain support information, including sponsor and financial support acknowledgment. For example, "This work was supported in part by the U.S. Department of Commerce under Grant BS123456".

The next few paragraphs should contain the authors' current affiliations, including current address and e-mail. For example, F. A. Author is with the National Institute of Standards and Technology, Boulder, CO 80305 USA (e-mail: author@ boulder.nist.gov).

S. B. Author, Jr., was with Rice University, Houston, TX 77005 USA. He is now with the Department of Physics, Colorado State University, Fort Collins, CO 80523 USA (e-mail: author@lamar.colostate.edu).

T. C. Author is with the Electrical Engineering Department, University of Colorado, Boulder, CO 80309 USA, on leave from the National Research Institute for Metals, Tsukuba, Japan (e-mail: author@nrim.go.jp).

I. Introduction

GENETIC algorithm (GA) is a simple yet effective evolutionary algorithm that is inspired by the Darwinian principles of biological evolution and adaptation in nature [1]. The natural evolution is a combination of *inheritance* and *variation*. While inheritance is the major source of the living organisms, variations of the genetic materials inherited from parents through genetic rearrangements and/or random changes (usually in small amount) are the determinant of creating viable as well as ameliorated organisms. GA starts with a population of chromosomes (a finite linear string of symbols) that represents the solutions and a set of genetic operators (GOs) are applied on these chromosomes in an iterative manner. GA uses crossover and mutation operators as the major source of variations throughout the evolution to produce well-adapted individuals (or chromosomes). Crossover and mutation have been extensively studied in various flavors [2-4] on numerous applications. They are the basic handles of GA to intensify the existing knowledge (*exploitation*) about the search space and to discover new knowledge by diversification (*exploration*) across that space. In this article, we formulate a new crossover operator for optimization in continuous domain that focuses on local substrings of a full chromosome and is applied adaptively by taking feedback from the search space with the help of memory, called LAmC (local, adaptive and memory-assisted crossover).

A stochastic search based evolutionary algorithm (EA) is applicable to wide variety of scientific research and engineering applications that are non-convex and multimodal, primarily due to their invariance to the existence of initial guess, differentiability and continuity of the objective functions. A rich literature is available on various evolutionary computing and swarm intelligence based search algorithms with numerous applications [5-24]. To design effective nature-inspired EAs for solving combinatorial as well as continuous optimization problems remains to be a demanding research topic due to their wide applicability, especially with increasing size and complexity.

After John Holland (1975) described the idea of GA search for the first time, researchers have extensively investigated its theory, constituents and performance [25-31]. The *Simple GA* (SGA) was proposed to include *crossover* and *mutation* only to exploit and explore respectively [26, 32, 33]. Later, the *Breeder GA* (BGA) introduced the idea of *elitism* and applied it along with uniform crossover and mutation [34, 35]. The tuning of population size and the operator rates resulted

several variants of GA [36, 37]. Some formulation of GA includes (λ+μ)-*Evolutionary Strategy* (ES) [38] such as *Genitor* [39-41], CHC (*Cross generational elitist selection*, *Heterogeneous selection* and *Cataclysmic mutation*) algorithm [42] and executable GAs [43-45]. On the other hand, some modeling exploits parallelism by *Tournament Selection* [46, 47], *Island Models* [39, 48-51] and cellular automata [45, 52]. Further, *twin removal* (TR) operator is introduced in [53] and integrated in a new GA variant named TRGA to ensure improved diversification in several applications [53-55]. Moreover, a memory-based crossover is investigated in two different GAs, namely FNGA and KGA [56]. Recently, an improved elitism with *homologous gene replacement* (hGR) [57] is used to develop a new GA, hGRGA [58] to further boost up the functional capacity of the elite chromosomes.

According to the principle of GA [59] there exists good *schemata* or *templates* in chromosomes that contribute to that chromosome's higher fitness. In nature, each organism has specific functions that are determined by the instruction set encoded within its chromosomes. Chromosome, which is equivalent to a DNA sequence, consists of genes that are connected together in the chromosome. Each of these genes is responsible for a function of that organism. Therefore, it can be hypothesized that the fully functional genes can result in a fitter chromosome so as a fitter organism. Inspired by this, we promote the idea of local genetic operators in this study that are applied on fixed-length substrings (referred as *genes*) of the full binary string corresponding to a chromosome. Additionally, we call the classical operators that are applied on the full chromosomes as global operators. With the local crossover and mutation, we try to implement the traditional theory of GA [1, 32, 33], popularly known as Schemata Theorem that states – "GA works by discovering, emphasizing, and recombining good *building blocks* or *schemas* (a bit string), that are specifically *short*". These short schemas are estimated to have above-average-fitness and thus confer higher fitness to the chromosomes or individuals they belong.

However, the truism behind any genetic variance is either beneficial or deleterious [60]. Even though crossover is regarded as the heart of the constructive power of GA, it has its detrimental effects while incorporating variance in the evolution [1, 33, 45, 53, 59]. Furthermore, the local gene-based crossover and mutation are more disruptive than their global counterparts. The reason seems straightforward: local application of crossover and mutation increases the total amount of variations. We empirically justify this argument in this study. However, we note that the local operators can outperform the global operators, especially in solving modern-world high dimensional problems if applied adaptively and guided appropriately. In view of the above considerations, we introduce a new *Adaptive and Memory-assisted Local operator based GA* (AMLGA) that uses an improved hGR based elitism [57] to ensure non-decreasing performance of GA, a new *memory-assisted local crossover* (LAmC) to intensify the search within the currently available solutions (proposed in this article). AMLGA further includes local mutation and TR [53] to explore the full domain space and inhibit premature convergence into local optimal solution.

The classical crossover operator relies on the selection method to choose relatively fitter chromosomes as parents so that the genetic materials of those parents can be recombined to produce even fitter offsprings. Thus, the classical crossover can use the genetic materials of the mating parents only. In the LAmC, we keep an alternative source of genetic materials in associative memory (AM) that can be used at the time of crossover. The memory stores the best chromosome (or gene) since the evolution begin and updated throughout the evolution. Thus, this new crossover can back-up the disruptions of the schemas to some extent that may result either from destroying a fit individual through mutation or not getting that individual selected by the selection method. In this framework, the crossover does not blindly accept the genetic material from the other parents, however adaptively evaluates the available one in the memory. Such adaptive application by taking feedback from the search space helps us to avoid the tedious optimization trials required for tuning of the rates. Further, we strategically reduce the number of computations of objective function required in the basic AMLGA (*bAMLGA*) to develop a cost-improved version, named *iAMLGA*.

We tested AMLGA on a comprehensive test suite of 40 benchmark functions for numerical optimization. Under this study, we implemented 7 other GA variants with different combination of genetic operators in both global and local scopes and compared their performances with the basic AMLGA to understand the relative advantages and disadvantages of global and local operators. The promising simulation results of AMLGA support the hypothesis that adaptive and memory-guided local operators are effective in constructing short and local schema within genes. These schemas with above-average fitness can further boost up the fitness of the full chromosome. Moreover, we compared the reliability and success performance of bAMLGA and iAMLGA with two other state-of-the-art EAs, Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) [8] and Social Spider Algorithm (SSA) [18].

The rest of the paper is organized as follows. **Section II** reviews the basics of genetic algorithm and the problem definition in terms of GA elements. **Section II** introduces the genetic operators that are studied in this study. **Section IV** briefly describes seven GA variants with different combination of global and local operators. The idea and development of basic AMLGA and iAMLGA is elaborately presented in **Section V**. **Section VI** contains the test function definitions, simulation results and performance comparison. At last, we briefly conclude in **Section VII**.

II. Genetic Algorithm (GA) Basics

Genetic algorithm imitates the rules of Mother Nature, "*survival of the fittest*", among a set of individuals for selection and evolution over consecutive generations (epochs) to solve an optimization problem. GA starts with a randomly generated initial population of chromosomes, also referred as individuals or candidate solutions. The values of the solution variables are encoded in the chromosomes. Then, the fitness

values of the chromosomes are quantified by evaluating the objective function under consideration. After that, a selection procedure is employed for the selection of individuals that are given higher preferences to pass on their genes to the next generation. We utilize roulette wheel selection algorithm that performs proportionate selection of the chromosomes based on their fitness values. These selected individuals are then tweaked using genetic operators according to the predefined rate of each operator to generate new individuals for the next generation. Thus, GA evolves through the natural adaptation process in which the fittest chromosomes or candidate solutions tend to survive, breed and propagate their genetic information to future generations. It is heuristically assumed that each successive generation will contain fitter individuals than the previous generation as the new individuals are produced using better individuals with good genes of the current population. GA is an iterative algorithm that runs for a predetermined number of generations (epochs) and terminates if the desirable solution is found. The basic steps of GA are shown in **Table I**.

TABLE I
BASIC STEPS OF GA

Step 1:	Initialize the population of individuals randomly
Step 2:	Compute the fitness scores of the individuals
Step 3:	Record the best result. If the optimal solution is found, go to step (8)
Step 4:	Perform selection of fitter individuals using roulette wheel selection procedure
Step 5:	Apply genetic operators to tweak the selected individuals
Step 6:	Generate new population of individuals for next generation
Step 7:	Repeat from step (2) to (6) until the predefined number of generations are evaluated
Step 8:	Report the best solution as output

A. Representation of Solution

GA uses a population of chromosomes, $pop = [C_1, \ \dots, \ C_N]$ where N is the number of chromosomes or individuals within the population or the size of population (pop_{size}). A solution vector, $\boldsymbol{X}_i = [x_{i,1}, \ \dots, \ x_{i,d}]^T$ is mapped to a chromosome, $C_i = [g_1, \ \dots, \ g_d]^T$ where $i \in \{1, 2, \dots, N\}$. Therefore, a chromosome is referred as a candidate solution. We represent each decision variable of the solution vector, $x_{i,j}$ as a gene $C_i(g_j)$ in the i^{th} chromosome, where $j \in \{1, 2, \dots, d\}$. Here, d is the dimension of the solution or search space, thus the problem size. We apply binary encoding scheme to store the values of the variables within genes. A gene is presented as $g = [b_1, \ \dots, \ b_m]^T$ where, m is the number of bits used to encode one variable. Therefore, the bit length of each chromosome is $L_C = d \times m$. Each bit position in a chromosome is called a locus, l where $l \in \{1, 2, \dots, L_c\}$. Each locus can have two possible alleles, 0 or 1. Further, the sequence of bits within each gene has three parts: (i) sign bit, m_s; (ii) decimal part, m_d and (iii) fractional part, m_f. Therefore, $m_s + m_d + m_f = m$. We used one bit to present the sign, specifically '0' for positive and '1' for negative real number. The number of bits used for decimal (m_d) and fractional part (m_f) may vary for different objective functions to ensure required precision to present numbers within the lower and upper bound of that function's search space, $[lb, ub]$. Therefore, the value of each variable is bounded so that $lb \leq x_{i,j} \leq ub$. The representation of a solution in terms of GA elements is shown in **Fig. 1**.

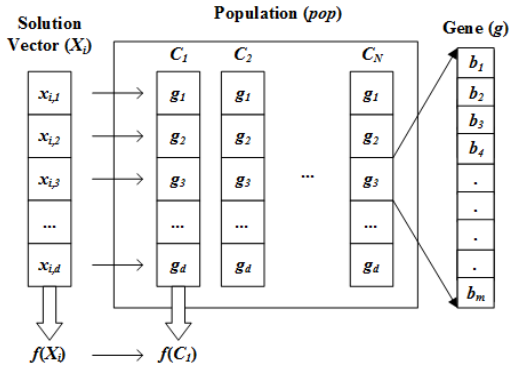

Fig. 1. Problem definition in terms of GA elements. Here, N is the population size or the number of chromosomes (C), d is the number of genes (g) in each chromosome or the number of dimension of search space or the number of variables (x) in each solution ($\boldsymbol{X}$), and m is the number of bits (b) to represent each gene. The variable i is used to index solution vectors and their elements.

A bound constrained optimization problem can be defined as follows:

$$\underset{\boldsymbol{X}}{\text{minimize}}\, f(\boldsymbol{X}), f(\boldsymbol{X}): S \subset \mathbb{R}^d \to \mathbb{R}$$

Here, $f(\boldsymbol{X})$ is the mathematical objective function to be optimized (minimized) over $\boldsymbol{X}$. S is the set of feasible solutions. A solution vector $\boldsymbol{X}^*$ is called optimal if $f(\boldsymbol{X}^*) \in S$ is less than or equal to any other element of S. The above form of optimization problem describes the minimization problem, and the counterpart (maximization problem) can be treated by negating the objective function. Fitness of each chromosome ($C_i, i \in \{1, \dots, N\}$) is denoted by $f(C_i)$. In this paper, a higher fitness value implies lower numerical value as we are after minimization of objective function value.

III. GENETIC OPERATORS

In this section, we describe the genetic operators that are utilized in different GA variants studied in this article. We further introduce necessary notations and terminologies here that facilitate the understanding of different GAs discussed in the following section. The operators are discussed in both global (when applied to a full chromosome) and local (when applied to each gene of a chromosome) scopes.

GA starts with a random initial population of individuals. Finally, the one that adapts with the best (minimum) fitness value through consecutive generations approaches to the optima. Each real-valued random individual, $\boldsymbol{X}_i = [x_{i,j}]^T$ is generated by **(1)** and mapped to binary chromosome C_i using the function $B(.)$ by **(2)**.

$$\forall x_{i,j} \in \boldsymbol{X}, x_{i,j} = (ub - lb).rand(0,1) + lb \quad (1)$$

$$B(\boldsymbol{X}_i) \to C_i \text{ such that } \forall x_{i,j} \in \boldsymbol{X}_i, B(x_{i,j}) \to C_i(g_j) \quad (2)$$

Here, $i = 1, 2, \dots, N$ and $j = 1, 2, \dots, d$. The function $rand(0,1)$ generates uniformly distributed random number within [0, 1].

A. Elitism with Homologous Gene Replacement (hGR)

Elitism is the genetic operator that segregates a subset of individuals from current population at the beginning of a generation. These predefined proportion of chromosomes are relatively fitter than the other chromosomes in the population, thus called *elites*. Elitism aims at the survival of these highly fitted individuals to guarantee non-decreasing performance from GA over time. The well-known foundation of GA, the Schemata Theorem states that GA tends to increase good schema (a set of bit strings with above-average fitness values) in the population through evolution and searches by prioritizing and recombining those schemata that confer higher fitness to the chromosomes. However, both crossover and mutation can create or destroy schema. These disruption effects of crossover and mutation under different scenarios have been formulated in the literature [45, 53, 61] in the form of quantifying the existence probability of a schema after crossover and mutation. As elites are passed to the next generation without any modification, elitism is useful in keeping those schemata intact through evolution. The elitism is controlled by a predefined rate denoted by r^E. In *classical elitism*, $n_E = (N \times r^E)$ number of highly fitted elites are directly passed from the current population to the next generation population. We consider this elitism as a global operator, presented in **Fig. 2(a)**.

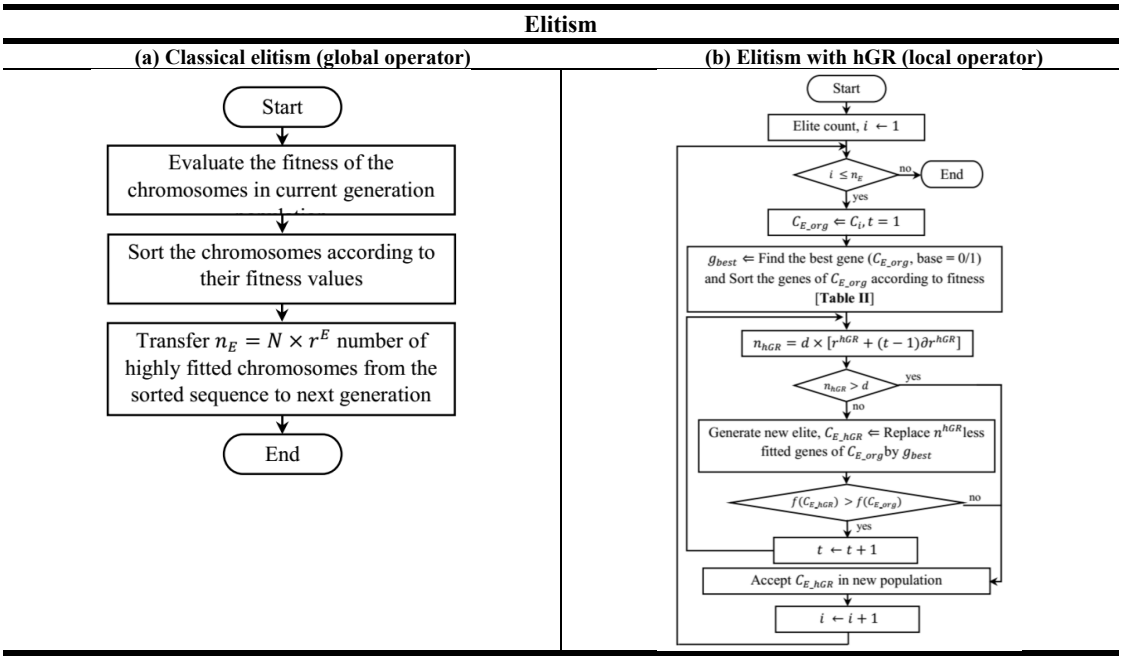

Fig. 2. Flowcharts of **(a)** classical elitism (global operator) and **(b)** elitism with homologous gene replacement (local operator).

Very recently an improved elitism is proposed in [58, 62] that enhances the benefits provided by elitism one step ahead by distributing the healthiest gene of each elite chromosome to replace relatively less fitted genes of that corresponding elite. This operator is called *homologous gene replacement*, in short hGR. The *elitism with hGR* is considered as a local operator as it works on the genes to improve the elite chromosomes further. This operator is motivated to mimic the natural phenomena that the combination of good genes can form a fitter chromosome [60]. The working principle of hGR operator is to identify the best gene (g_{best}) with good schema of each elite chromosome and replace the relatively weaker genes of the corresponding elite if these replacement improves that elite's fitness.

Therefore, hGR involves the evaluation of relative fitness of the local genes to determine the best gene (g_{best}) of an elite. While computing the relative fitness of a gene in a chromosome (one variable in a solution), we assign a common *base* value to other variables to generalize the effect of other variables. The pseudo code of the function that determines the best gene of a chromosome is given in **Table II**. It takes the chromosome (C) of which the best gene is to be determined and the *base* value as input and, outputs the best gene schema (g_{best}) and a vector containing relative fitness of all the genes of that chromosome ($\boldsymbol{\partial fv}$). For each elite chromosome, the hGR operation in **Fig. 2(b)**, is applied twice with base value '0' and '1' to generate improved elites. Finally, the better elite between the two that are generated using two different base values is accepted in the next generation population. The reasoning behind using two different base values is discussed in [58]. In short, the relative fitness of a variable can be effectively determined by deactivating the effect of other variables using '0' only if the variables are separable. When the effect of a variable is dependent on the values of other variables in case of inseparable functions, it is useful to generalize the other variables using a common non-zero value.

TABLE II
PSEUDO CODE OF FIND THE BEST GENE OF A CHROMOSOME

Function [$g_{best}, \partial \boldsymbol{fv}$] = *Find the Best Gene of a Chromosome* (*C, base*)
Begin
  Find $\boldsymbol{X}$ such that $B(\boldsymbol{X}) \rightarrow C$ and $B(x_j) \rightarrow C(g_j)$
  Let $\partial \boldsymbol{fv}$ stores the fitness values of individual variables or genes
  For j = 1 to d
    $\boldsymbol{X}' = [base]^d$;
    $x'_j = x_j$;
    $\partial \boldsymbol{fv}(j) = f(\boldsymbol{X}')$; // f(.) is the objective function
  End For
  $g_{best} \leftarrow g_j \; with \; minimum \; \nabla f$
End

The hGR operator is proposed to be applied only on the elites to prevent the saturation of population with similar bit pattern of the best gene [58]. Moreover, this operator aims to spread the schemas of elites only that have high fitness values and to bias the selection towards those schemas in the following crossover operator. We denote an original elite before applying hGR as C_{E_org} and the one after hGR by C_{E_hGR}. Further, the hGR operator is an adaptive operator that treats each elite differently as the bit pattern of different elites can accept different number of gene replacement. Initially hGR is controlled by a low rate r^{hGR} that determines the number of less fitted genes to be replaced, $n_{hGR} = r^{hGR} \times d$ in each elite. After each application of hGR on an elite, a feedback from the search space is taken on the current status of the elite in the form of fitness. If the fitness is improved, the rate is increased by ∂r^{hGR}. Such adaptive application of hGR ensures optimal number of gene replacement without time consuming tuning of the control parameters of hGR. Thus, hGR is applied on an elite for t number of times until the number of genes to be replaced (n_{hGR}) exceeds the total number of genes (d) in a chromosome or the performance of the elite is decreased. The number of genes to be replacement at each application or trial, t is formulated in [58] by **(3)**. With predefined r^{hGR} and ∂r^{hGR}, the maximum possible number of trial (t_{max}) of hGR on an elite can be quantified by (4), as derived in [58].

$$n_{hGR} = \lceil d \times [r^{hGR} + (t-1)\partial r^{hGR}] \rceil \quad (3)$$

$$t_{max} = 0\left(\left(\frac{1 - r^{hGR}}{\partial r^{hGR}}\right) + 1\right) \quad (4)$$

Equation **(3)** shows that the number of weaker genes to be replaced at each application of hGR is dynamically adjusted by the dimension of solution or scale of the problem d. Therefore, the maximum number of application of hGR given by **(4)** is independent of the scale of a problem or the dimension of the problem.

B. Selection

In GA, selection operator is used to select potentially good parents for crossover to produce children. The idea behind such selection is that better parents can produce better off-springs after mating. We use fitness proportionate selection, also known as *roulette wheel selection* procedure that assigns a selection probability to each individual or chromosome depending on its fitness. Thus the chromosomes with higher fitness have higher chance to be selected as parents. The pseudo code of roulette wheel selection is outlined in **Table III**. The function takes a vector containing unsorted fitness values ($\boldsymbol{FV}$) as input and returns the *index* of the candidate selected as parent for crossover.

TABLE III
PSEUDO CODE OF ROULETTE WHEEL SELECTION

Function *index = Roulette Wheel Selection* ($\boldsymbol{FV}$)
Begin
  $\boldsymbol{FV} = [fv_1, fv_2, \ldots, fv_N]^T$ is the unsorted fitness vector of current population; // bold font indicates vector form
  [$\boldsymbol{sFV}$, $\boldsymbol{id}$] = sort($\boldsymbol{FV}$, '*ascend*'); // $\boldsymbol{sFV}$ is the sorted fitness vector // $\boldsymbol{id}$ is the index of sorted individuals
  Let $\boldsymbol{p} = [p_i]^T$, i = 1, 2, …, N // probability vector
  $\forall i, p_i = fv_i / \sum_{i=1}^{N} fv_i$;
  $\boldsymbol{p} = 1 - \boldsymbol{p}$; /* to assign high probability to individuals with low numerical value as fitness for minimization problem */
  $r = rand(0,1)$; // random number between 0 to 1
  i = 1;
  While $r > 0$
    $r = r - p_i$;
    $i = i + 1$;
  End While
  $index \leftarrow id(i-1)$;
End

C. Crossover

Crossover is the process of combining two selected individuals, called *parents* to produce two new individuals for the next generation, called *offsprings*. The motivation behind this process is to pass the recombined genetic material, possibly improved, of parents from one generation to next generation. The parents are selected based on their fitness using roulette wheel selection procedure (**Table III**). We call this operator as *global crossover* when it is applied on full chromosomes. On the other hand, the operator can be applied locally on each of the gene pair of the parent chromosomes. Then we refer it as *local crossover*. We utilized single point crossover that recombines two parent chromosomes (global) or genes (local) of parent chromosomes around one crossover point to generate new offspring chromosomes (global) or genes (local).

In single point global crossover operation, we select one locus as crossover point, p^C where $1 \leq p^C \leq (L_C - 1)$ using **(5)**. Here, L_C is the bit length of a chromosome. We exchange the parts of bit string beyond p^C between the two parent chromosomes (C_{P1} and C_{P2}) to create two offspring chromosomes (C_{O1} and C_{O2}). We denote the rate of crossover by r^C. Then, the number of crossover is $n_C = {(N \times r^C)}/{2}$ as each crossover operation results two new individuals. In case of local crossover, a similar operation is performed on all the local genes of n_C pairs of parent chromosomes simultaneously. As there are d number of genes in each chromosome, the local crossover can be considered as a d-point crossover in d-dimensional space. For different genes, the crossover point p_g^C, where $1 \leq p_g^C \leq (m-1)$ and m is

the bit length of a gene, can be different loci and determined using **(6)**.

$$p^C = \lceil (L_C - 1).rand(0,1) \rceil \quad (5)$$

$$p_g^C = \lceil (m - 1).rand(0,1) \rceil \quad (6)$$

The function $rand(0,1)$ generates uniformly distributed random number within [0, 1]. The flow of operations along with the sample instances of global and local crossover are demonstrated in **Fig. 3** and **4**.

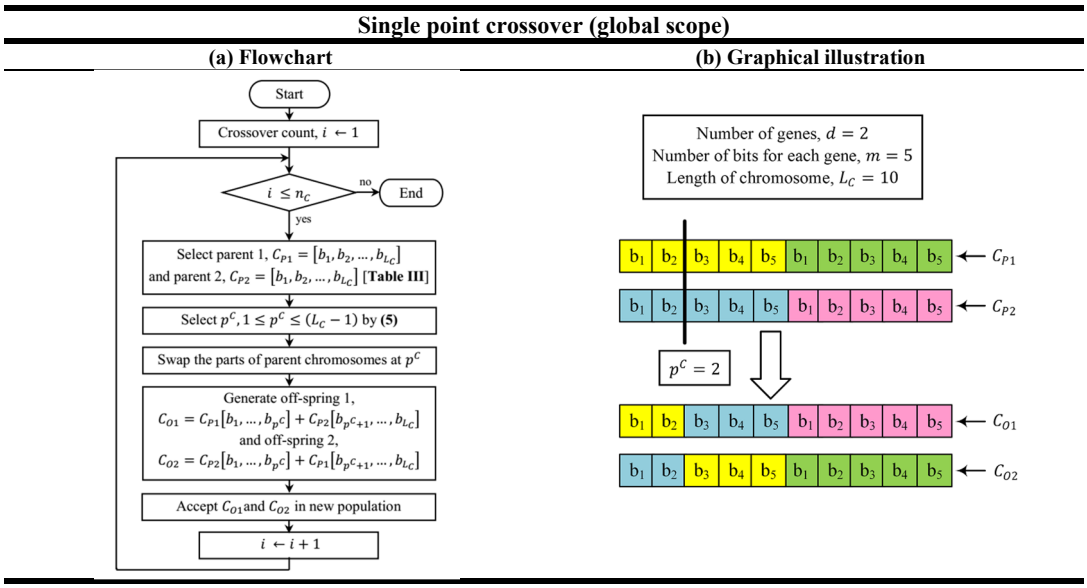

Fig. 3. **(a)** Flowchart of global crossover operation and **(b)** graphical illustration of a sample global crossover application.

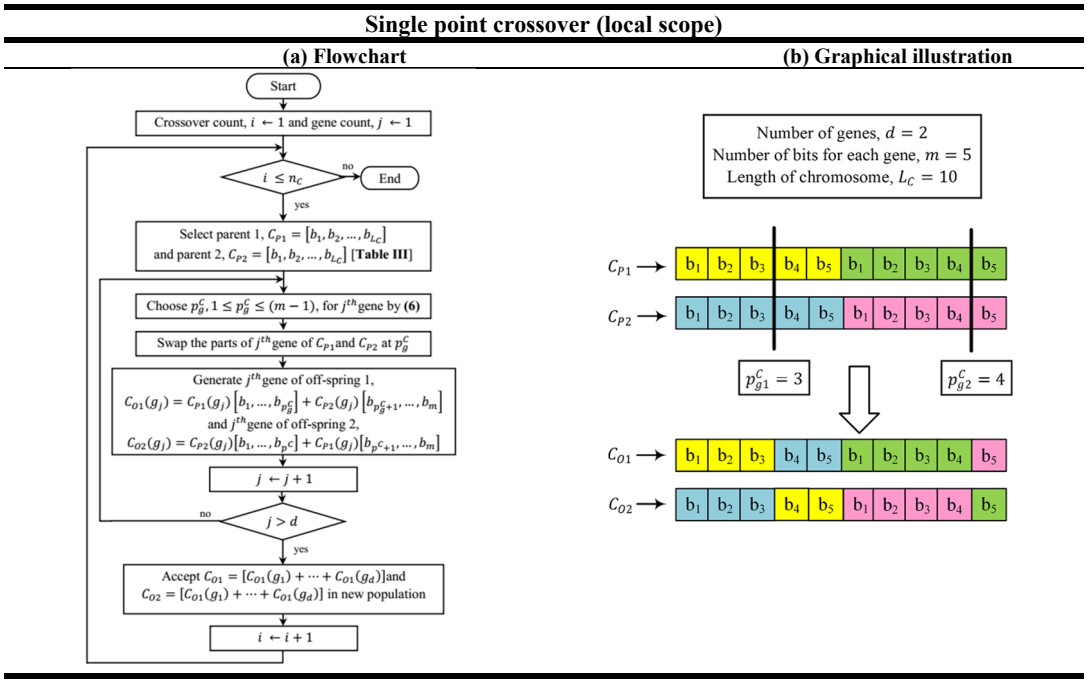

Fig. 4. **(a)** Flowchart of local crossover operation and **(b)** graphical illustration of a sample local crossover application.

D.Memory-Assisted Crossover

GA's primary productive power comes from crossover operation that selects highly fitted chromosomes with above-average-fitness schemata by the fitness proportionate selection and rearranges them to produce even better individuals. This process gradually bias the selection procedure toward the instances of schemas whose fitness values are estimated to be higher. To make it happen, the classical crossover operation depends on the fitness proportionate selection. In the novel memory-assisted crossover operation, the process of discovering schemata is further guided using two associated-memory (AM) which is first proposed in [56] and named *AM-based Crossover* (AmC). AM consists of two triangular memories that stores the parts (bit strings) of the current best individual at all crossover point. Unlike the classical crossover in **Fig. 3** that blindly swaps the subparts of parents, AmC considers two different subparts while performing the

crossover to produce two off-spring candidates: one is from the other participating parent (basic crossover) and other is from the AM. AmC takes feedback from the search space by evaluating the two potential offsprings and keeps the better one. Thus, the search of schemata in AmC operation is not limited to the other parent individual, rather assisted by memory and applied adaptively if useful. Moreover, the AM is updated if a better schema is found from the mating partner. Thus AM can essentially contain the best-performing subparts of a chromosome at different crossover point through consecutive generations. **Fig. 5** shows the flowchart and graphical illustration of global AmC operation.

In this article, we extended this idea of memory-assisted crossover operation from full chromosome level to local gene level, called *Local AM-based Crossover* (LAmC) operator. The LAmC operator stores the schema of current best gene at every crossover point and assists the basic local crossover operation (**Fig. 4**) with the memory adaptively. Therefore, we perform crossover in each gene, however we check two potential subparts: one from the gene of parent and other from AM. We consider the better subpart to produce the genes of the offsprings. The motivation of this operator is to direct the GA search towards the local good schemata within the genes and recombine them to produce better genes. From evolution point of view, the LAmC operator aims to emulate the evolution of each individual trait of an organism (encoded in genes) unlike the global one that focuses on the evolution of the full organism (chromosome). The results justify that LAmC enhances the exploitation capacity of classical crossover (both local and global scopes) and global AmC to a higher extent. The flow of operations of LAmC along with a sample application are demonstrated in **Fig. 6**.

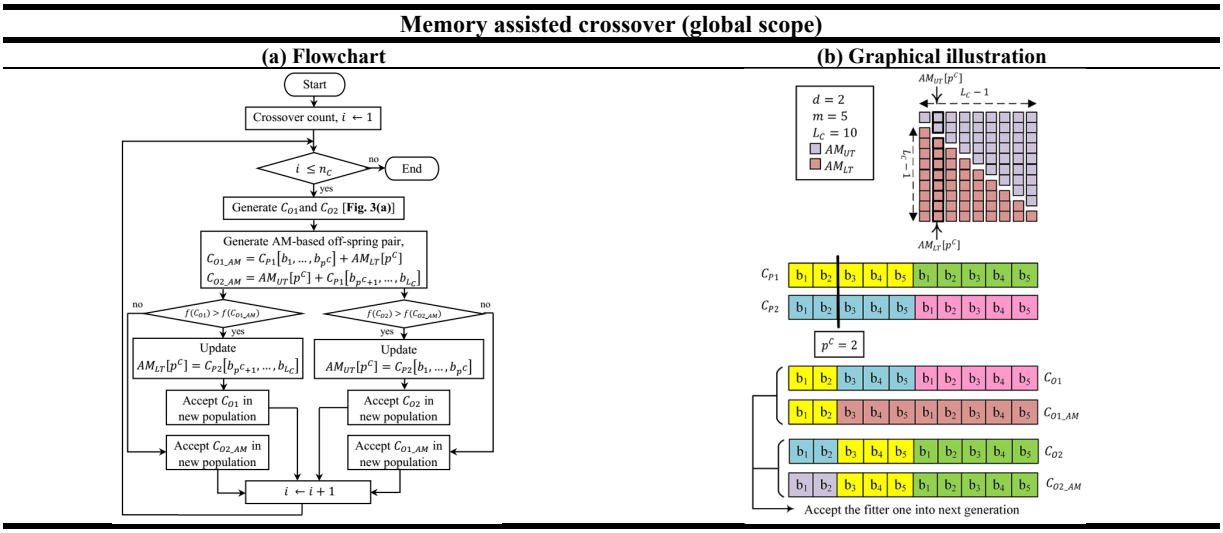

Fig. 5. **(a)** Flowchart of global AM-based crossover (AmC) and **(b)** graphical illustration of a sample application of AmC in global scope.

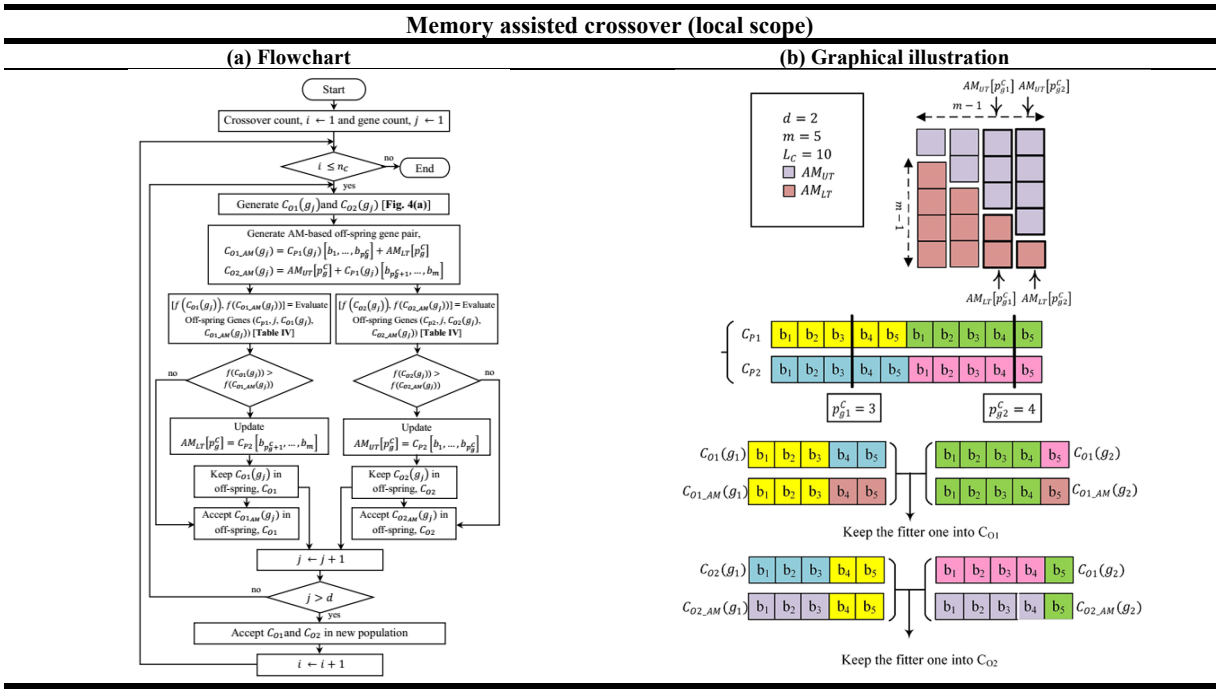

Fig. 6. **(a)** Flowchart of local AM-based crossover (LAmC) and **(b)** graphical illustration of a sample application of LAmC.

In case of global scope, AM is of dimension L_C by $(L_C - 1)$ (**Fig. 5(b)**). It consists of one upper triangular (AM_{UT}) and one lower triangular (AM_{LT}) matrix that respectively store the upper parts and lower parts of the currently available best chromosome at all possible crossover points ($p^C \in [1, \dots, L_C - 1]$). We generate one pair of offsprings by classical crossover (C_{O1}, C_{O2}) and another pair of offsprings by taking the sub part from AMs at p^C (C_{O1_AM}, C_{O2_AM}). We accept the fitter ones into the next generation population. However, if the sub part of the parent is better, the AM is updated with that part at respective p^C. In case of LAmC, the dimension of AM is m by $(m - 1)$, shown in **Fig. 6(b)**. Here, AM_{UT} and AM_{LT} store the upper parts and lower parts, respectively of the currently available best gene at all possible crossover points ($p_g^C \in [1, \dots, m - 1]$). We search for the best gene in all the chromosomes of initial population using the function in **Table II** and initialize the AM with best available gene. While doing crossover at each pair of genes of the parent chromosomes, we keep either the part of the parent chromosome's gene or that of AM, whichever gives higher additive fitness to the parent chromosome. This involves the evaluation and comparison of two offspring genes, one generated using the subpart from parent and other using that from AM. The evaluation of offspring genes includes a parent chromosome in which the newly generated genes are plugged in to evaluate the additive fitness provided by the new genes. The pseudo-code of this evaluation process is given in **Table IV**. The inputs of this function are one parent chromosome (C_p), index of gene under crossover operation (j) and the two offspring genes generated from parent (g_{parent}) and AM (g_{AM}). The function returns the fitness values of g_{parent} and g_{AM}. Finally, AMs are always upgraded by the best gene's schema, thus the next crossover is guided by the best schema through consecutive generations.

TABLE IV
PSEUDO CODE OF EVALUATING TWO OFF-SPRING GENES IN LAMC

Function $[f_{T1}, f_{T2}]$ = *Evaluate Off-spring Genes* (C_p, j, g_{parent}, g_{AM})

```
Begin
    C_p is a parent chromosome;
    j is the index of the gene under current local AmC;
    C_T1 ← C_p and C_T2 ← C_p;    // create two temporary chromosomes
    C_T1(g_j) ← g_parent;    /* new chromosome with off-spring gene
                                having subpart of other parent's gene */
    C_T2(g_j) ← g_AM;        /* new chromosome with off-spring gene
                                having subpart of AM */
    Generate X_T1 so that B(X_T1) → C_T1;
    Generate X_T2 so that B(X_T2) → C_T2;
    f_T1 = f(X_T1) ;
    f_T2 = f(X_T2) ;
End
```

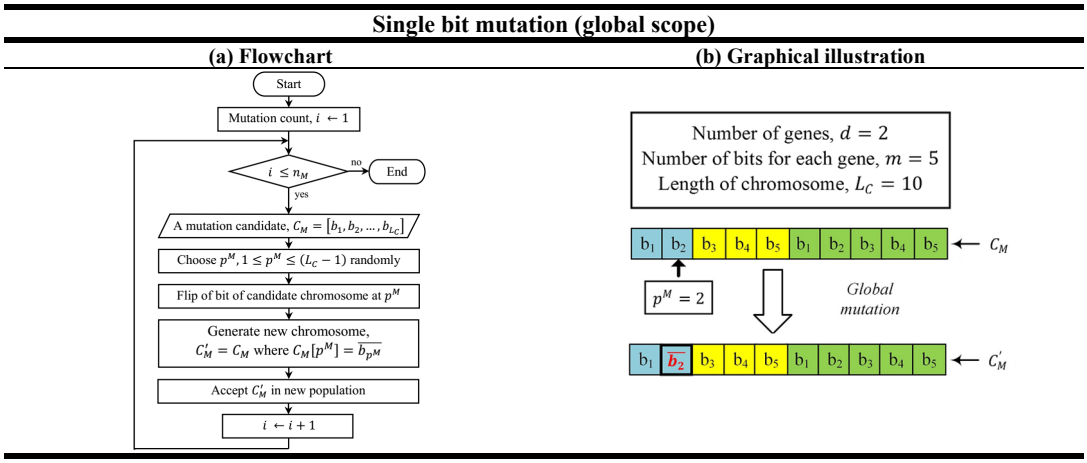

Fig. 7. **(a)** Flowchart of global mutation operation and **(b)** graphical illustration of a sample global mutation application.

E. Mutation

Mutation is the process of changing some individuals of current population randomly to produce new individuals for the next generation population. Mutation interprets the process of having random genetic diversity in nature. Mutation is usually performed at a relatively low probability [4]. Mutation can be applied once on the full chromosome, called *global mutation* or it can be exercised on all genes of a chromosome simultaneously, called *local mutation*. The flowcharts and sample instances of global and local mutation operations are presented in **Fig. 6** and **7**.

In mutation process, we randomly select a candidate chromosome (C_M). For single bit global mutation, we then choose one random locus as mutation point p^M using **(7)**, where $1 \leq p^M \leq L_C$ and L_C is the bit length of a chromosome. After that, we flip the allele value of the bit at p^M. Specifically, if $C_M[p^M] = 0$ (or 1) then for the mutated new chromosome, $C'_M[p^M] = 1$ (or 0). We denote the rate (probability) of mutation by r^M. Thus, the number of mutation $n_M = N \times r^M$.

$$p^M = \lceil L_C . (rand(0,1)) \rceil \quad (7)$$

$$p_g^M = \lceil m . (rand(0,1)) \rceil \quad (8)$$

For local mutation, a similar operation is performed on all the local genes ($C_M(g)$) of a mutation candidate. The random mutation points (p_g^M), determined by (8), for different genes can be different loci, where $1 \leq p_g^M \leq (m - 1)$ and m is the bit length of a gene.

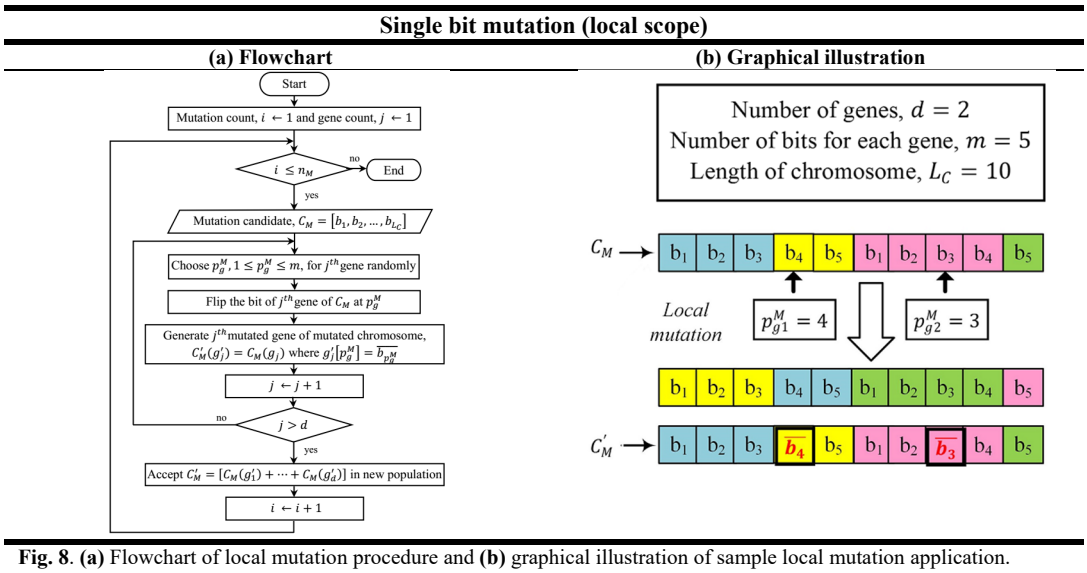

Fig. 8. **(a)** Flowchart of local mutation procedure and **(b)** graphical illustration of sample local mutation application.

F. Twin Removal (TR)

The Schemata Theorem states that the schemata with above-average fitness values are likely to be sustained in the population as the GA proceeds. Thus the similar chromosomes called *twins* tend to increase in the population [63]. The growth of such correlated twins inevitably debilitates the impact of mutation in producing new random solution when the underlying landscape is very complex with higher number of minima [64-66]. As a result of such effective diversification, GA can miss prominent solutions that are not yet investigated and can keep on searching in a local peak that ends in a premature convergence. The TR operation includes the detection and removal of the twins along with introduction of new random chromosomes within the population which is proven to be able to ensure optimal diversity and effective to avoid trap of local minima [56, 67].

The TR operation is controlled by *Chromosome Correlation Factor* (CCF) which defines the allowable similarity between chromosomes. In this process, we count the number of loci ($nSim$) of the chromosomes under comparison with identical allele values ('0' or '1'). If $nSim$ is higher or equal than the allowable similarity ($CCF \times L_C$), we replace the chromosome of relatively lower fitness with a new randomly generated chromosome. TR is performed when the new population for next generation has been generated after applying all other genetic operators. At the beginning of the evolution, CCF value is set to 1.0 (100% similarity). However, the population becomes more correlated as the generations proceed. Thus we allow lesser degree of correlation by decreasing the CCF at a low rate (∂CCF) in consecutive generations until the value is 0.8 (80% similarity). It has been empirically justified that CCF equal to 0.8 is effective to avoid the stall condition due to reduced diversity [53, 56] in GA population. We mark a chromosome as '*new*' if it has been detected as a twin and replaced by a random new chromosome to avoid redundant computation involved in repetitive twin evaluation. Note that, the scope of applying TR is global. However, it inherently checks the similarity of bits of the constituent genes of the chromosomes. Thus, we did not explore TR operator in local scope. The flow of operations in TR is displayed in **Fig. 9**.

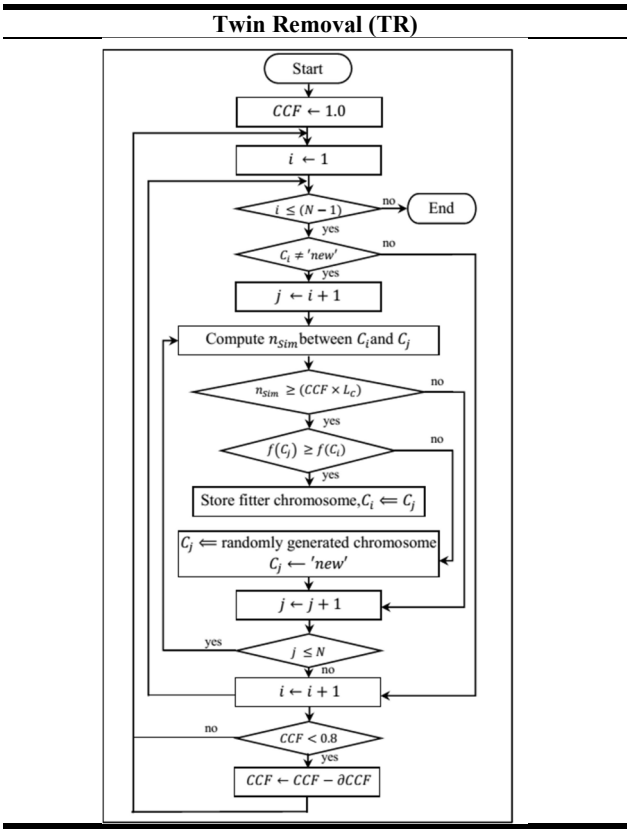

Fig. 9. Flowchart of twin removal (TR) operation.

IV. GA VARIANTS

In this section, we outline seven different GA variants that are studied and implemented under this work and compare with the proposed variant, AMLGA. The most crucial design parameter involved in the development of an evolutionary algorithm is to ensure an appropriate balance between

exploitation and exploration. When the search space is high-dimensional, multimodal and complex, it is compulsory to diversify the search to explore all possible locations as well as to intensely exploit a potential location near global optima. The seven algorithms considered in this section include different combinations of the genetic operators discussed in the previous section to effectively sample the search space. Four variants apply the operators in global scope only whereas there variants explore the operators in local scope. This section gives a convenient way to understand the reasoning of using different operators as well as their respective advantages and disadvantages in different GA variants.

A. Simple Genetic Algorithm (SGA)

SGA is the canonical version of GA that employs crossover and mutation on the chromosomes (global scope) for exploitation and diversification respectively. In our implementation of SGA (**Table V**), we include the elitism in addition to single-point crossover and single-bit mutation. Elitism ensures the survival of the fitter individuals (elites) from the disruption effects [1, 33, 45, 59] of crossover and mutation so that the elites can pass their genetic materials to next generation through evolution.

TABLE V
PSEUDO CODE OF SGA

```
     Procedure of SGA
01   Begin
02       Initialize population of individuals randomly by (1) – (2);
03       While (allowable number of generations does not exceed)
04           Evaluate fitness of the chromosomes;
05           Perform global elitism [Fig. 2(a)];
06           Perform selection of parents for crossover [Table III];
07           Perform global single-point crossover [Fig. 3];
08           Perform global single-bit mutation [Fig. 7];
09           Create new population of individuals for next generation;
10       End While
11   End
```

B. Fast Navigating Genetic Algorithm (FNGA)

FNGA [56] first used AM-based crossover (AmC) in global scope in place of the classical crossover to enhance the constructive exploitation power of crossover. As discussed earlier, the AmC operator uses the associated-memory (AM) to mitigate the disturbance effect of crossover. AmC does not generate off-springs deterministically by taking subpart from the other parent rather adaptively checks the potential of the available subpart from AM. Therefore, AmC has higher ability to construct a better schema and can offer improved exploitation. This argument has been empirically justified in [56] and in this work by comparing the performance of FNGA and SGA. In addition to global AmC, FNGA (**Table VI**) applies elitism and mutation on full chromosomes (global scope) to produce new individuals (or chromosomes) for the next generation.

TABLE VI
PSEUDO CODE OF FNGA

```
     Procedure of FNGA
01   Begin
02       Initialize population of individuals randomly by (1) – (2);
03       Evaluate fitness pf initial population;
04       Initialize AM with the best available chromosome;
05       While (allowable number of generations does not exceed)
06           Evaluate fitness of the chromosomes;
07           Perform global elitism [Fig. 2(a)];
08           Perform selection of parents for crossover [Table III];
09           Perform global AM-based crossover (AmC) [Fig. 5];
10           Perform global single-bit mutation [Fig. 7];
11           Create new population of individuals for next generation;
12       End While
13   End
```

C. Twin Removal Genetic Algorithm (TRGA)

TRGA [53, 67] is first proposed by Hoque *et al*. to include an improved diversification operator called twin removal (TR) in GA. TR operator can back up the reduced exploration power of mutation due to the similar chromosomes (twins) by introducing a new random chromosomes in place of similar chromosomes. Diversification is one of the key requisites to search a multimodal space effectively by avoiding the premature convergence in local optima. Thus TRGA can achieve better search results than the classical SGA (justified in the result section). TRGA (**Table VII**) further uses elitism to keep the schemas of highly fit individuals intact, single-point global crossover for exploitation and single-bit global mutation for exploration.

TABLE VII
PSEUDO CODE OF TRGA

```
     Procedure of TRGA
01   Begin
02       Initialize population of individuals randomly by (1) – (2);
03       While (allowable number of generations does not exceed)
04           Evaluate fitness of the chromosomes;
05           Perform global elitism [Fig. 2(a)];
06           Perform selection of parents for crossover [Table III];
07           Perform global single-point crossover [Fig. 3];
08           Perform global single-bit mutation [Fig. 7];
09           Perform twin removal (TR) [Fig. 9]
10           Create new population of individuals for next generation;
11       End While
12   End
```

D. Kite Genetic Algorithm (KGA)

KGA [56] combines the enhanced exploitation capacity of global AmC and improved diversification power of TR in one generation. Thus, KGA has advantages over both FNGA and TRGA in having balanced exploitation and exploration in every generation. It has been laid down by the Schemata Theorem that GA works by prioritizing and sustaining instances of schemas with above-average-fitness. The AM-based crossover (AmC) further ensures this by guiding the crossover towards better schemas stored in the AM. Thus after, it is more likely to have similarity between chromosomes that can make the GA search static. TR plays a complimentary role here by reducing similarity and introducing new random solutions. Therefore, KGA (**Table VIII**) combines classical elitism, global AmC, single-bit global mutation and TR in one generation.

TABLE VIII
PSEUDO CODE OF KGA

	Procedure of KGA
01	**Begin**
02	Initialize population of individuals randomly by **(1) – (2)**;
03	Evaluate fitness pf initial population;
04	Initialize AM with the best available chromosome;
05	**While** (allowable number of generations does not exceed)
06	Evaluate fitness of the chromosomes;
07	Perform **global elitism** [**Fig. 2(a)**];
08	Perform **selection** of parents for crossover [**Table III**];
09	Perform **global AM-based crossover** (AmC) [**Fig. 5**];
10	Perform **global single-bit mutation** [**Fig. 7**];
11	Perform **twin removal** (TR) [**Fig. 9**]
12	Create new population of individuals for next generation;
13	**End While**
14	**End**

E. Local Operator based Simple Genetic Algorithm (LSGA)

LSGA employs the local crossover and local mutation in a similar framework of SGA, thus named LSGA. Therefore, LSGA applies single-point crossover and single-bit mutation in all the genes (substring of the full binary string of a chromosome) separately which can considered as *d*-point crossover and *d*-bit mutation when we have *d* number of variables in the solution or the search space is *d*-dimensional. However, tweaking all the variables (or genes) simultaneously can prohibit LSGA to have small changes in the solution, especially needed when a solution is close to global optima. Therefore, the algorithm may roam around the full search space, nevertheless cannot converge into a specific optima. We experienced such effects of local operators, given in the results section, where the performance of LSGA is found inferior than the other four GAs with global operators. LSGA (**Table IX**) combines classical elitism, local single-point crossover, and local single-bit mutation in one generation. We intentionally introduced LSGA in this study to rationalize that local operators can be way ineffective if not appropriately controlled as carried out in the proposed AMLGA, specially for high dimensional problems.

TABLE IX
PSEUDO CODE OF LSGA

	Procedure of LSGA
01	**Begin**
02	Initialize population of individuals randomly by **(1) – (2)**;
03	**While** (allowable number of generations does not exceed)
04	Evaluate fitness of the chromosomes;
05	Perform **global elitism** [**Fig. 2(a)**];
06	Perform **selection** of parents for crossover [**Table III**];
07	Perform **local single-point crossover** [**Fig. 4**];
08	Perform **local single-bit mutation** [**Fig. 8**];
09	Create new population of individuals for next generation;
10	**End While**
11	**End**

F. Local Operator based Twin Removal Genetic Algorithm (LTRGA)

LTRGA has a similar workflow of LSGA (**Table IX**) except it uses TR operator in addition to the operators of LSGA. This variant is implemented as an effort to minimize the disruption effect of local crossover and mutation at all the genes in LSGA using TR. Therefore, LTRGA (**Table X**) uses elitism, local single-point crossover for exploitation and local single-bit mutation along with TR for exploration. Our simulation results suggest that using TR can effectively reduce such disruption effects by introducing new random solutions only. Thus, performance of LTRGA is found better than that of LSGA.

TABLE X
PSEUDO CODE OF LTRGA

	Procedure of LTRGA
01	**Begin**
02	Initialize population of individuals randomly by **(1) – (2)**;
03	**While** (allowable number of generations does not exceed)
04	Evaluate fitness of the chromosomes;
05	Perform **global elitism** [**Fig. 2(a)**];
06	Perform **selection** of parents for crossover [**Table III**];
07	Perform **local single-point crossover** [**Fig. 4**];
08	Perform **local single-bit mutation** [**Fig. 8**];
09	Perform **twin removal** (TR) [**Fig. 9**]
10	Create new population of individuals for next generation;
11	**End While**
12	**End**

G. Homologous Gene Replacement based Genetic Algorithm (hGRGA)

The hGRGA [57, 58] is the most recent variant of GA studied in the literature that uses an improved elitism with homologous gene replacement (hGR). This operator investigates the genes (local scope) of elites only. It locates the best gene in each elite and distributes that best gene to replace the relatively weaker genes of that elite. The application of hGR in hGRGA is controlled adaptively to ensure non-decreasing performance of GA. Moreover, the distribution of gene schemas have immediate influence on the selection so that these schemas are prioritized in the following crossover operation for offspring generation. However, hGRGA (**Table XI**) includes classical crossover (in global scope) after hGR based elitism for exploitation. Besides, global single-bit mutation and TR are used for exploration. The superior simulation results of hGRGA in comparison to LSGA and LTRGA supports that the adaptive hGR operator is way more effective than classical crossover and mutation in local scope for constructing short schemas in chromosomes.

TABLE XI
PSEUDO CODE OF HGRGA

	Procedure of hGRGA
01	**Begin**
02	Initialize population of individuals randomly by **(1) – (2)**;
03	**While** (allowable number of generations does not exceed)
04	Evaluate fitness of the chromosomes;
05	Perform **elitism with hGR** [**Fig. 2(b)**];
06	Perform **selection** of parents for crossover [**Table III**];
07	Perform **global single-point crossover** [**Fig. 3**];
08	Perform **global single-bit mutation** [**Fig. 7**];
09	Perform **twin removal** (TR) [**Fig. 9**]
10	Create new population of individuals for next generation;
11	**End While**
12	**End**

V. Adaptive and Memory-assisted Local Operator based GA (AMLGA)

In this part of the article, we discuss the theory, rationale and components of the proposed algorithm, AMLGA. It combines four operators that are discussed in **Section III**. Altogether these operators allow both *inheritance* and *variations* in the cumulative process of evolution to ensure balanced exploitation and exploration of the search space to produce well-adapted solutions. The operators of AMLGA are: (*i*) elitism with homologous gene replacement (hGR, shown in **Fig. 2(a)**), (*ii*) memory-assisted local crossover (LAmC, shown in **Fig. 6**), (*iii*) local mutation (**Fig. 8**), and (*iv*) twin removal (TR, shown in **Fig. 9**). In the following, we explain the three basic properties of these operators and their benefits:

1) Local*:* To investigate the concept and effectiveness of local genetic operators, we use a similar set of biological terminologies used as GA vocabulary [60] with a different meaning. In the classical encoding of a candidate solution into a chromosome with a bit string, a single gene is a single bit having two possible allele values ('0' or '1') [61]. In our encoding scheme (**Fig. 1**), a gene is a short block of adjacent bits, thus a shorter substring having fixed start and end loci within the full long bit string of the chromosome. A chromosome is the representative of a solution vector whereas the genes represent the variables of that vector. When a genetic operator like recombination or mutation works on the substrings (or genes) rather than the full binary string (or chromosomes), it is considered as a local operator.

From biological point of view, a gene can be defined as the functional and physical unit of the heredity passed from parent to offspring whereas the chromosomes contain the complete set of functional instructions of an individual [60]. Therefore, the combination of fully functional genes, arranged linearly in a chromosome, can produce a highly fit chromosome or individual. The layout of this evolution process in nature has similarity with the underlying landscape of a high-dimensional optimization problem where each dimension can be mapped to a decision variable and a combination of the optimized variables can generate an optimum solution. This motivated us to hypothesize that the gene level (variable or dimension wise) evolution can result a well-adapted individual (full solution vector). Thus schemas can be located and recombined locally within genes of a chromosome. A separate study conducted by us [57, 58] justifies the effectiveness of a local operator, hGR in continuous optimization. However, operators like local crossover that modifies a chromosome in each gene can increase the extent of genetic variance in a full chromosome. As a genetic variance can either build or break a schema, local operations need to be controlled carefully. We observe the deleterious effect of local operators in the performance of LSGA and LTRGA, whereas the performance is unalike for AMLGA in which the local operators are applied adaptively and guided by memory.

2) Memory-assisted*:* The local AM-based crossover (LAmC) used in AMLGA has this feature that can facilitates in recovering the disruption effects of classical crossover. The classical crossover swaps a subpart of one parent with a subpart of other parent deterministically to produce an offspring. Therefore, it relies only on the selection method to choose fitter chromosomes as parents so that the fitter schemas of these parents can be rearranged to generate even better children. However, the consequence can be otherwise as well. The rearrangement can break an existing schema of the current best individual and there is no mechanism in classical crossover to prevent this.

In LAmC (**Fig. 6**), two associated-memories (AMs) stores the best gene string at every crossover points from the beginning of evolution. Therefore, we have an option to compare the potential of the sub-part available in the memory with that of other parent while producing offsprings. AMs are continuously updated if a better sub-part is available from the mating partner. Thus, it can further help in rebuilding schema that was lost earlier. In a separate study [56], we found that the global AmC (**Fig. 5**) operator were more successful than classical global crossover (**Fig. 3**). We investigate the effectiveness of memory-assisted crossover in local scope (LAmC) under this study, and found it to outperform classical crossover in global (SGA and TRGA) and local scopes (LSGA and LTRGA) by a higher extent.

3) Adaptive*:* Adaptive EAs becoming increasingly popular as it does not require the computationally costly trial-and-error parameter tuning step. To overcome the inconveniences involved in such time-consuming parameter tuning, researchers have investigated the adaptation and self-adaptation of parameters in EAs [68-73]. However, even with a pre-specified setting of parameters involved in an operator, the execution of operators can be either *deterministic* or *adaptive*. A deterministic operator tweaks a candidate solution according to a predetermined rationale, whereas an adaptive operator takes feedback from the search space and finally execute it only if useful.

At the beginning of our experiments with local operators in LSGA and LTRGA, we observe a higher amount of disturbance effect of crossover and mutation when applied in each gene of the chromosomes, especially for high-dimensional scalable problems. However, we notice a significant improvement in the performance when the recombination in crossover is guided using memory and applied adaptively by LAmC (**Fig. 5**) after taking feedback from the search space. Moreover, we consider elitism with hGR (**Fig. 2(b)**) in AMLGA that adaptively attempts to boost up the fitness of the elites further before accepting them in the next generation directly. The performance gap between all other GA variants and those of hGRGA and AMLGA clearly highlights the usefulness of this characteristics of operators.

A. Components of AMLGA

Before describing the components of AMLGA, we briefly review the notations in here to facilitate the understanding of the readers.

- N is the number of chromosomes or individuals or candidate solutions
- d is the number of genes in a chromosome or decision variables in a solution or dimensions of search space or problem size

- m is the bit-length of a gene
- L_C is the bit-length of a chromosome
- $C = [g_1, \dots, g_d]^T$ is a chromosome where $C_i(g_j)$ is the j^{th} gene of i^{th} chromosome, $i \in (1, \dots, N)$ and $j \in (1, \dots, d)$
- $g = [b_1, \dots, b_m]^T$ is a binary gene where b indicates a bit
- $\boldsymbol{X}_i = [x_{i,1}, \dots, x_{i,d}]^T$ is a real-valued solution where $x_{i,j}$ is the j^{th} variable of i^{th} solution, $i \in (1, \dots, N)$ and $j \in (1, \dots, d)$

The AMLGA algorithm consists of two main steps: *1*) Initialization and *2*) Evolution. In the following we describe the steps which are outlined in **Table XII**. In this description, we did not employ any strategy to reduce the cost (in terms of number of function evaluation) of AMLGA. Thus we refer it as *basic AMLGA*, in short *bAMLGA*.

1) Initialization:

We randomly generate the initial population first and then initialize the associative-memory (AM) needed to perform memory-assisted crossover.

a) ***Initialize population*** *(line 09–13 of Table XII)*: We maintain two populations in our implementation of AMLGA; one real-valued, $pop_{real} = [\boldsymbol{X_1}, \dots, \boldsymbol{X}_N]^T$ and another binary, $pop_{binary} = [C_1, \dots, C_N]^T$. We uniformly randomize the N individuals of pop_{real} using **(1)** within the allowable bounds ($[lb, ub]$) to cover the entire search space as much as possible. The real-valued candidate solutions are then converted into binary to create pop_{binary} using **(2)**.

b) ***Initialize AM*** *(line 14–25 of Table XII)*: In the AM initialization phase, we intend to find the best available gene in the population as we adopt local AM-based crossover strategy (LAmC in **Fig. 6**). For this purpose, we call the function *Find the Best Gene of a Chromosome (C, base)* in **Table II** for all the chromosomes as C with two base values ('0' and '1') due to the aforementioned reasons discussed in Section III. Essentially we use the gene with highest relative fitness contribution to initialize the AM. The AM_{UT} and AM_{LT} (**Fig. 6(b)**) store the best gene schema at every crossover point. Therefore, we can consider the best gene's sub-parts as potential candidate at the time of crossover in addition to the mating parents.

2) Evolution

In this phase, the population of individuals are evolved to produce an optimum individuals. The steps under evolution are described below.

a) ***Fitness Evaluation*** *(line 29–32 of Table XII)*: This is the first step of a generation where we evaluate the individuals (or chromosomes) of the current generation population by passing them to the objective function to be minimized. After that, we sort the individuals according to their fitness values.

b) ***Elitism with hGR*** *(line 33–58 of Table XII)*: This step mimics the concept of '*Survival of the fittest*' in biological evolution where $n_E = N \times r^E$ number of fitter individuals are separated as elites from others. Here, n_E is the number of elites and r^E is the rate of elites.

In AMLGA, we applied hGR (**Fig. 2(b)**) on each of the elite to improve them further. The hGR is an adaptive and local operator that does not allow the decrement of the fitness of original elite (C_{E_org}), yet attempt to execute hGR only if a better elite (C_{E_hGR}) can be produced. To do this, we find the best gene (g_{best}) of the elite chromosomes using *Find the Best Gene of a Chromosome (C, base)* in **Table II**. Then, we try to replace the less healthy genes by the best gene in multiple trials (t) where we slowly increase the number of genes (n_{hGR}) to be replaced in hGR adaptively with d if the performance is improved (line 44 – 52) and n_{hGR} does not exceed the maximum number of genes (d). We can perform maximum of t_{max} trials, defined by **(4)** in which n_{hGR} is quantified using **(3)** according to the rate of gene replacement. This process is repeated with two different base values in line 53. Finally, we accept the resulting elite after hGR (C_{E_hGR}) in the next generation population (line 54 – 57). In a separate study [58], we have empirically shown that hGR is scalable with the problem size (d) or dimension of the search space as the n_{hGR} is adapted according to it.

The improvement of the elites with hGR has significant impact on the subsequent crossover operation of AMLGA. These improvement of elites creates a reasonable gap between the fitness of elite set with the rest of the individuals in the population. Thus, the selection becomes strongly biased towards those improved elites to select these elites as parents and the highly fitted schemas of those elites are therefore prioritized to be recombined through LAmC to produce better offsprings. This increases the exploitation power of crossover.

c) ***Fitness proportionate selection*** *(line 63–68 of Table XII)*: In this phase, we perform the selection of the parent chromosomes using the function *Roulette Wheel Selection* ***(FV)*** outlined in **Table III** according to their fitness. This process resembles the concept of biological fitness which is defined by the success of reproduction and relies on the idea that better parents can produce better off-spring. This is consistent with the working foundation of GA [1] as well that states that GA works by prioritizing and recombining the good schemas present in highly fit individuals.

TABLE XII
PSEUDO CODE OF BAMLGA

```
    Procedure of bAMLGA
01  Begin
02    I ← generation index;                    // I = current and I+1 = next generation
03    I_max ← maximum number of generations;
04    optima ← Inf;
03    pop_real = [X_1^I, X_2^I, ..., X_N^I];   // real-valued population of solutions
04    pop_binary = [C_1^I, C_2^I, ..., C_N^I]; // binary population of chromosomes
05    FV = [fv_1, fv_2, ..., fv_N]^T;          // fitness values
06    f(.) ← objective function;
07
08    Step 1: Initialization
09      Step 1(a): Initialize population
10        For i = 1 to N
11          ∀x_{i,j} ∈ X_i, x_{i,j} = (ub − lb).rand(0,1) + lb;
12          B(X_i) → C_i where ∀x_{i,j} ∈ X_i, B(x_{i,j}) → C_i(g_j);
13        End for
14      Step 1(b): Initialize AM
15        AM[] = m × (m-1);
16        /* Find the best gene of the population */
17        For i = 1 to N
18          [g_{best_0}, ~] = Find the Best Gene of a Chromosome(C_i^I, 0);
19          [g_{best_1}, ~] = Find the Best Gene of a Chromosome(C_i^I, 1);
20          g_best ← g_{best_0} or g_{best_1}, whichever has better fitness;
21        End For
```

/* Assign the best gene into AM ***/**
For i = 1 to *N*
AM[:, k] = g_{best};
End For

Step 2: *Evolution*
While ($I \leq I_{max}$)
Step 2(*a*): *Fitness Evaluation*
For i = 1 to *N*
$\boldsymbol{FV}[i] = fv_i = f(\boldsymbol{X}_i^I)$;
End For

Step 2(*b*): *Elitism with hGR*
/* segregate the elites */
[***sFV***, ***id***] = sort(***FV***, '*ascend*'); /* ***sFV*** is the sorted fitness vector and ***id*** is the index of sorted individuals */
$n_E = N \times r^E$; // number of elite

/* Apply hGR on each elite */
For i = 1 to n_E
$\boldsymbol{X}_{E\_org} = \boldsymbol{X}_{\boldsymbol{id}(i)}^I$, $C_{E\_org} = C_{\boldsymbol{id}(i)}^I$; // original real-valued and binary elites
$[g_{best}, \boldsymbol{\partial fv}] = Find\ the\ Best\ Gene\ of\ a\ Chromosome(C_i^I, 0)$;
[$\boldsymbol{\partial sfv}, \boldsymbol{iid}$] = sort($\boldsymbol{\partial fv}$, '*descend*'); /* $\boldsymbol{\partial sfv}$ is the sorted fitness of the genes and ***iid*** is the index of sorted genes in descending order (worst gene first) */
$n_{hGR} = 1$, t = 1;$B(x_{best}) \rightarrow g_{best}$;
$\boldsymbol{X}_{E\_hGR\_0} = \boldsymbol{X}_{E\_org}$; // New elite by hGR using base value 0
While ($n_{hGR} \leq d$) // Stop if all genes are replaced
/* Number of genes to be replaced */
$n_{hGR} = \lceil d \times [r^{hGR} + (t-1)\partial r^{hGR}] \rceil$;
$\boldsymbol{X}_{E\_hGR\_0}[iid(1:n_{hGR})] = x_{best}$;
IF $f(\boldsymbol{X}_{E_{hGR_0}}) < f(\boldsymbol{X}_{E_{org}})$
Break; // stop if fitness decreases
End If
$\boldsymbol{X}_{E\_org} = \boldsymbol{X}_{E\_hGR\_0}$; t = t + 1;
End While
Repeat line 39-52 to produce elite ($\boldsymbol{X}_{E\_hGR\_1}$) by hGR using base value 1 from original elite ($\boldsymbol{X}_{E\_org}$)
$\boldsymbol{X}_{E\_hGR} \leftarrow \boldsymbol{X}_{E\_hGR\_0}$ or $\boldsymbol{X}_{E\_hGR\_1}$, whichever has better fitness;
$\boldsymbol{FV}[\boldsymbol{id}(i)] = fv_{\boldsymbol{id}(i)} = f(\boldsymbol{X}_{E\_hGR})$; // update fitness vector
/* Accept new elites as next generation individuals */
$\boldsymbol{X}_i^{I+1} = \boldsymbol{X}_{E\_hGR}$, $\boldsymbol{C}_i^{I+1} = \boldsymbol{C}_{E\_hGR} = B(\boldsymbol{X}_{E\_hGR})$;
End For

/* Start of crossover loop */
$n_C = (N \times r^C)/2$; // number of crossover
For i = 1 to n_C
Step 2(*c*): *Fitness Proportionate Selection*
/* Select parent - 1 */
$[index\_p1] = Roulette\ Wheel\ Selection(\boldsymbol{FV})$;
$C_{P1} = C_{index\_p1}^I$;
/* Select parent - 2 */
$[index\_p2] = Roulette\ Wheel\ Selection(\boldsymbol{FV})$;
$C_{P2} = C_{index\_p2}^I$;
Step 2(*d*): *Local AM-based crossover (LAmC)*
For j = 1 to d
$p_g^C = \lceil (m-1).rand(0,1) \rceil$; // point of crossover for this gene
/* Generate offspring gene – 1 */
$\boldsymbol{C}_{O1}(g_j) = C_{P1}(g_j)\left[b_1, \dots, b_{p_g^C}\right] + C_{P2}(g_j)\left[b_{p_g^C+1}, \dots, m\right]$;
$\boldsymbol{C}_{O1\_AM}(g_j) = C_{P1}(g_j)\left[b_1, \dots, b_{p_g^C}\right] + AM_{LT}[p_g^C]$;
$\left[f\left(\boldsymbol{C}_{O1}(g_j)\right), f\left(\boldsymbol{C}_{O1_{AM}}(g_j)\right)\right] =$
$Evaluate\ Offspring\ Genes\left(C_{P1}, j, \boldsymbol{C}_{O1}(g_j), \boldsymbol{C}_{O1\_AM}(g_j)\right)$;
If $(f(\boldsymbol{C}_{O1}(g_j)) > f(\boldsymbol{C}_{O1_{AM}}(g_j)))$
$AM_{LT}[p_g^C] = C_{P2}(g_j)\left[b_{p_g^C+1}, \dots, m\right]$; // Update AM
Accept $\boldsymbol{C}_{O1}(g_j)$ in nest generation population;
Else If
Accept $\boldsymbol{C}_{O1\_AM}(g_j)$ in nest generation population;
End If
/* Generate offspring gene – 2 */
$\boldsymbol{C}_{O2}(g_j) = C_{P2}(g_j)\left[b_1, \dots, b_{p_g^C}\right] + C_{P1}(g_j)\left[b_{p_g^C+1}, \dots, m\right]$;
$\boldsymbol{C}_{O2\_AM}(g_j) = AM_{UT}[p_g^C] + C_{P1}(g_j)\left[b_{p_g^C+1}, \dots, m\right]$;
$\left[f\left(\boldsymbol{C}_{O2}(g_j)\right), f\left(\boldsymbol{C}_{O2\_AM}(g_j)\right)\right] =$
$Evaluate\ Offspring\ Genes\left(C_{P2}, j, \boldsymbol{C}_{O2}(g_j), \boldsymbol{C}_{O2\_AM}(g_j)\right)$;
If $(f(\boldsymbol{C}_{O2}(g_j)) > f(\boldsymbol{C}_{O2\_AM}(g_j)))$
$AM_{UT}[p_g^C] = C_{P2}(g_j)\left[b_1, \dots, b_{p_g^C}\right]$; // Update AM
Accept $\boldsymbol{C}_{O2}(g_j)$ in nest generation population;
Else If
Accept $\boldsymbol{C}_{O2\_AM}(g_j)$ in nest generation population;
End If
End For
End For

Step 2(*e*): *Local Mutation*
$n_M = (N \times r^M)$; // number of mutation
For i = 1 to n_M
$M = round\left((N - n_E).rand(0,1)\right) + n_E$;
$C_M = C_M^I$; //Candidate for mutation
For j = 1 to d
$p_g^M = \lceil m.(rand(0,1)) \rceil$; // point of mutation in this gene
$C_M'(g_j)[b_{p_g^M}] = C_M'(g_j)[\overline{b_{p_g^M}}]$;
End For
Accept the mutated chromosome C_M' in the next generation population;
End For

Step 2(*f*): *Twin Removal (TR)*
For i = 1 to (N-1)
If $C_i^{I+1} \neq new$
For j = i+1 to N
n_{Sim} = Number of identical bit of C_i^{I+1} and C_j^{I+1};
If $n_{Sim} \geq (CCF \times L_C)$
If $f(C_j^{I+1}) > f(C_i^{I+1})$, $C_i^{I+1} \leftarrow C_j^{I+1}$; **End If**;
End If
Replace C_j^{I+1} with a new random chromosome;
End For
End If
End For
End While
End

d) ***LAmC*** *(line 69–93 of Table XII):* In LAmC (**Fig. 6**), we rearrange the genetic materials, encoded in the parent chromosome's (C_{P1} and C_{P2}) genes to produce genes of the off-spring chromosomes (C_{O1} and C_{O2}). For each of the jth gene (g_j), we randomly select a locus as crossover point, p_g^C within $[1, m)$ using **(6)** in line71. Then, we produce two different off-spring gene candidates, $C_{O1}(g_j)$ and $C_{O1\_AM}(g_j)$ using (9) as the jth gene of the first off-spring (C_{O1}). In both of the candidate off-spring genes, we keep the upper part of jth gene of C_{P1} fixed, and attach the lower part of jth gene of C_{P2} and the corresponding part from AM_{LT} at p_g^C to produce $C_{O1}(g_j)$ and $C_{O1\_AM}(g_j)$ respectively. After that, we evaluate the two off-spring genes using function *Evaluate Off-spring Genes* ($C_P, j, g_{parent}, g_{AM}$) outlined in **Table IV**. Here, C_{P1} is passed as the parent chromosome (C_P) in which the new genes created from parent's sub-part ($g_{parent} = C_{O1}(g_j)$) and that from AM ($g_{AM} = C_{O1\_AM}(g_j)$) is integrated at jth gene position to compute the additive fitness values of the off-spring gene candidates with respect to the parent and finally we keep the better one. The AM_{LT} at p_g^C is updated by that corresponding lower part of $C_{P2}(g_j)$ if the parent's sub-part is found fitter. Thus AM always contains the best available sub-part at that crossover point.

$$\begin{matrix} C_{O1}(g_j) \\ C_{O1\_AM}(g_j) \end{matrix} = upper\ part\ of\ C_{P1}(g_j) + \begin{matrix} lower\ part\ of\ C_{P2}(g_j) \\ part\ of\ AM_{LT}\ at\ p_g^C \end{matrix} \quad (9)$$

$$\begin{matrix} C_{O2}(g_j) \\ C_{O2\_AM}(g_j) \end{matrix} = \begin{matrix} upper\ part\ of\ C_{P2}(g_j) \\ part\ of\ AM_{UT}\ at\ p_g^C \end{matrix} + lower\ part\ of\ C_{P1}(g_j) \quad (10)$$

Similarly, we produce two different off-spring gene candidates, $C_{O2}(g_j)$ and $C_{O2\_AM}(g_j)$ using **(10)** as the j[th] gene of the second off-spring (C_{O2}). Here, the upper parts of the offspring gene candidates are different; one is the upper part of j[th] gene of C_{P2} and other part from the upper triangle AM_{UT} at p_g^C. We attach the lower part of j[th] gene of C_{P1} with both of these sub-parts to produce $C_{O2}(g_j)$ and $C_{O2\_AM}(g_j)$. After that, we evaluate the two off-spring genes using the same function *Evaluate Off-spring Genes* ($C_P, j, g_{parent}, g_{AM}$) outlined in **Table IV**. Here, C_{P2} is passed as the parent (C_P) in which the new genes created from parent's sub-part ($g_{parent} = C_{O2}(g_j)$) and that from AM ($g_{AM} = C_{O2\_AM}(g_j)$) is plugged in the position of j[th] gene to compute the additive fitness values of the off-spring gene candidates and finally we keep the better one. If $C_{O2\_AM}(g_j)$ provides higher benefit, we consider it instead of using parent's sub-part. However, the parent's sub-part is used if $C_{O2}(g_j)$ is found better and the AM_{UT} at p_g^C is updated by that upper-part of $C_{P2}(g_j)$.

This local crossover operation creates a difference in the performance between hGRGA [57, 58] that introduced hGR operator for the first time and AMLGA, proposed in the study. After propagating the best gene schema of an elite chromosome in the other locations of the genes, hGRGA relies on classical global crossover applied on the full chromosome for recombining those schemas. Elitism with hGR operator can optimize all the genes (or variables) simultaneously using the best gene and can lead directly to the global optima if that best gene or, variable is the optimum value necessary at all the variable positions to achieve the global optima. However, it is possible the best gene cannot be identified properly due to the complex property of the objective function, such as *inseparability* of the variables. Then it is useful to recombine the sub-parts of the bit strings of each of those genes to ensure appropriate exploitation of those good schemas distributed by hGR, rather than applying single-point global crossover on the chromosome. Thereafter, the other properties of this crossover such as memory-assisted and adaptability strengthen the power of AMLGA.

e) ***Local Mutation*** *(line 94–102 of Table XII)*: After finishing the recombination of parents and producing off-springs, we carry out single bit local mutation (**Fig. 8**) at each gene of the candidate chromosome selected randomly for mutation. However, we do not consider elites for mutation so that the good schemas of elites or improved elites after hGR operation survive through consecutive generations of evolution. The index M for the candidate chromosome for mutation, C_M is determined by **(11)** in line 97.

$$C_M, M = round\big((N - n_E).rand(0,1)\big) + n_E \quad (11)$$

Here, N is the total number of chromosomes, n_E is the number of elites that are indexed from location 1 to n_E within a population, and *rand*(0, 1) generates uniformly distributed random number within [0, 1]. To perform the local mutation, we select a locus randomly in the bit string of each gene as mutation point, p_g^M within [1, m] by **(8)** where, m is the bit length of a gene. Then, we flip the allele value of that locus (**Fig. 8**).

Here, we perform gene level mutation to have random changes in the bit strings of each of the genes that is another difference between hGRGA [58] and AMLGA. The hGRGA [58] uses global mutation in one bit of the full chromosome, whereas the local mutation of a bit in each gene string is more complementary with the other operations of AMLGA. Distribution of the best schema by hGR [57] can create saturation of identical gene string in the population. Subsequently, the crossover of those identical bit strings of the gene pair of the parents have neither constructive nor destructive impact on the schemas [9, 32, 53, 61]. Thus we randomly mutate each of genes of the chromosomes to have higher random changes.

f) ***Twin Removal*** *(line 104–115 of Table XII)*: TR (Fig. 9) is the last operation that is executed at the end of every generation. This step filters out the identical chromosomes (twins) from the population to assure appropriate amount of diversified individuals in the population. Here, we define an upper bound on the allowable similarity between two chromosomes by Chromosome Correlation Coefficient (CCF). We compare each pair of chromosomes and if a pair is found having higher number of identical bits than that of determined by CCF, the pair is marked as twins. The one with the relatively lower fitness value is then replaced by a new random solution generated by **(1)** and **(2)**. The new chromosome is then marked as 'new' and not considered further in the twin evaluation to avoid redundant comparison.

B. Complexity of bAMLGA

Adaptive application of genetic operator is costly in terms of Number of Function Evaluation (NFE) as it involves taking feedback about the effectiveness of the operation before execution by evaluating the objective function. Here, we analyze the cost of basic idea of AMLGA, outlined in **Table XII** as bAMLGA that do not employ any cost optimization. We define a cost function $\mathcal{C}(.)$ that measures the complexity of an operation or procedure in terms of absolute value of NFE required to reach the target. Such absolute value of NFE is useful to compare the speed of the algorithms as it is not dependent on machines or coding languages. There are five potential steps or sub-steps of AMLGA that involves function evaluation.

1) Step 1(b): Initialize AM

AM initialization needs function evaluations (FEs) to determine the best gene (line 17 – 21 of **Table XII**) of the initial population. It calls *Find the best gene of a chromosome* (*C, base*) function twice for N times and this function (shown in **Table II**) d number of Fes. Therefore,

$$\mathcal{C}(Initiliza\ AM) = N \times (d + d) = 2Nd \quad (12)$$

2) Step 2(a): Fitness Evaluation

This step evaluated the objective function for N times to determine the fitness values of N individuals (line 30 – 32 of **Table XII**). Thus,

$$\mathcal{C}(Fitness\ Evaluation) = N \quad (13)$$

3) Step 2(a): Elitism with hGR

In this phase, hGR operator is applied on each of the n_E elites in line 30 – 32 of **Table XII**. It calls *Find the best gene of a chromosome* (*C, base*) function that requires d number of function calls. In addition, the while loop from line 44 to 52 is executed maximum of t_{max} number of times, formulated by **(4)** that involves one FE. However, the hGR operator is applied twice using two different base values (line 53). Therefore,

$$\mathcal{C}(Elitism\ with\ hGR) = n_E \times \left(d + 0\left(\frac{1-r^{hGR}}{\partial r^{hGR}} \times 1\right)\right) = 0\left(n_E\left[d + \left(\frac{1-r^{hGR}}{\partial r^{hGR}}\right)\right]\right) \quad (14)$$

4) Step 2(d): Local AM-based Crossover (LAmC)

The memory-assisted crossover is costlier than classical crossover as it checks two potential source of genetic materials while producing children, one from the other parent and another from the memory. The LAmC is applied for n_C times (line 61) on d number of genes (line 70). Each application requires two calls (line 75 and 85) of *Evaluate Offspring Genes* function (**Table IV**) and each of this function call requires two, in total 4 FEs per gene.

$$\mathcal{C}(LAmC) = n_c \times d \times 4 = 4n_c d \quad (15)$$

5) Step 2(f): Twin Removal (TR)

TR (line 105 – 115 of **Table XIII**) involves a search of twins by comparing all the $(n-1) \times (n-1)$ number of pairs of individuals and evaluates the objective function at line 110 only if a twin is found. Therefore we can only determine an upper bond on cost given below.

$$\mathcal{C}(TR) = 0(N-1) \times 0(N-1) = 0(N^2) \quad (16)$$

Step 1(b) and 2(f) that initializes population and performs local mutation respectively do not need FEs. Therefore, the cost of bAMLGA in one generation or epoch,

$$\mathcal{C}(bAMLGA) = \mathcal{C}(Initialize\ AM) + \mathcal{C}(Fitness\ Evaluation) + \mathcal{C}(Elitism\ with\ hGR) + \mathcal{C}(LAmC) + \mathcal{C}(TR)$$
$$\mathcal{C}(bAMLGA) = Nd + N + 0\left(n_E\left[d + \left(\frac{1-r^{hGR}}{\partial r^{hGR}}\right)\right]\right) + 4n_c d + 0(N^2) \quad (17)$$

C. Reducing Cost of bAMLGA to develop improved AMLGA: iAMLGA

Here, we discuss three strategies that we adopt to reduce the NFE (or cost) required in the bAMLGA, quantified in previous section, to make our algorithm robust and useful to deal with expensive optimization problems. We name the cost-reduced AMLGA as *improved AMLGA*, in short *iAMLGA*. The iAMLGA is briefly outlined in **Table XIII**. We reduce the cost of three steps (marked with *) among the five that involves function evaluations, discussed above.

*1) Step 1(b): Initialize AM**

To reduce the cost of this step (line 07 – 17 of **Table XIII**), we hypothesize that it is more likely that the best gene will be a part of the best chromosome. Therefore, we look for the best gene in the current best chromosome, rather than in all the chromosomes. We determine the best chromosome at the cost of N NFEs (line 10) and then we call *Find the best gene of a chromosome* (*C, base*) function twice that requires d NFEs. Thus, the cost of Initialize AM* linear which is less than that of bAMLGA which is quadratic (Equation **(12)**).

$$\mathcal{C}(Initialize\ AM^*) = N \times (d+d) = 2Nd \quad (18)$$

TABLE XIII
PSEUDO CODE OF IAMLGA

	Procedure of iAMLGA
01	**Begin**
02	line 01 – 06 of bAMLGA (Table XII)
03	NFE_{max} ← maximum number of function evaluation;
04	FE ← Function evaluation counter
05	**Step 1: *Initialization***
06	**Step 1(*a*): *Initialize population***: line 10 – 13 of bAMLGA (Table XII)
07	**Step 1(*b*): *Initialize AM****
08	AM[] = m × (m-1);
09	**/*** Find the best gene of the best individual ***/**
10	**For** i = 1 to *N*, $\boldsymbol{FV}[i] = fv_i = f(X_i^l)$; End For;
11	[***sFV, id***] = sort(***FV***, '*ascend*'); // sort according to fitness
12	$\boldsymbol{C}_{best}^l = \boldsymbol{C}_{id(1)}^l$; // pick the best individual
13	$[g_{best\_0}, \sim] = Find\ the\ Best\ Gene\ of\ a\ Chromosome(C_{best}^l, 0)$;
14	$[g_{best\_1}, \sim] = Find\ the\ Best\ Gene\ of\ a\ Chromosome(C_{best}^l, 1)$;
15	$g_{best} \leftarrow g_{best\_0}$ or g_{best_1}, whichever has better fitness;
16	**/*** Assign the best gene into AM ***/**
17	line 23 – 25 of bAMLGA (Table XII)
18	**Step 2: *Evolution***
19	**While** ($FE \leq NFE_{max}$)
20	**Step 2(*a*): *Fitness Evaluation***: line 30 – 32 of bAMLGA (Table XII)
21	**Step 2(*a*): *Elitism with hGR***: line 34 – 58 of bAMLGA (Table XII)
22	Line 59 – 60 of bAMLGA (Table XII)
23	**For** i = 1 to n_C
24	**Step 2(*c*): *Selection***: line 63 – 68 of bAMLGA (Table XII)
25	**Step 2(*d*): *Local AM-based crossover (LAmC*)***
26	**For** j = 1 to d
27	/* Generate candidates for offspring gene – 1 */
28	Line 73 – 74 of bAMLGA (Table XII)
29	**If** $((i\%2 = 1)$ && $(j\%2 = 1))$, Line 75 – 81 of bAMLGA (Table XII)
30	**Else** Accept $\boldsymbol{C}_{O1}(g_j)$ in nest generation population;
31	/* Generate candidates for offspring gene – 2 */
32	Line 83 – 84 of bAMLGA (Table XII)
33	**If** $((i\%2 = 0)$ && $(j\%2 = 0))$, Line 85 – 91 of bAMLGA (Table XII)
34	**Else** Accept $\boldsymbol{C}_{O2}(g_j)$ in nest generation population;
35	**End For**
36	**End For**
37	**Step 2(*e*): *Local Mutation***: line 95 – 103 of bAMLGA (Table XII)
38	**Step 2(*f*): *Twin Removal (TR*)***
39	**For** i = 1 to (N-1)
40	**For** j = i+1 to N
41	n_{Sim} = Number of identical bit of C_i^{l+1} and C_j^{l+1};
42	**If** $n_{Sim} \geq (CCF \times L_C)$, Replace C_j^{l+1} with a new chromosome;
43	**End If**
44	**End For**
45	**End For**
46	**End While**
47	**End**

*2) Step 2(d): LAmC**

In the LAmC of bAMLGA, we apply the memory-assisted crossover for both of the upper and lower parts of all the d genes using **(9)** and **(10)**. It can be considered as d-point LAmC. In iAMLGA, we perform $\frac{d}{2}$-point LAmC (line 23 – 36 of **Table XIII**). While implementing iAMLGA, we executed LAmC only on the odd-indexed genes of the odd-indexed crossover operation (line 29) and, respectively on the even-indexed genes for the even-indexed crossover (line 33). For the even-indexed genes of the odd-indexed crossover and odd-indexed genes of even-indexed crossover, we performed the classical crossover with the parents sub-part only. Moreover,

we considered either lower or upper part of the genes for each application of memory-assisted crossover. Thus, each application requires 2 FEs whereas the previous one requires 4 evaluations. While generating 1st offspring gene (line 27), we applied memory-assisted crossover on the upper part using **(9)** and classical crossover on the lower part. On the other, we applied memory-assisted crossover on the lower part of the gene using **(10)** and classical crossover on the upper part to produce the 2nd offspring gene (line 31). These two strategies reduced the cost of crossover to one-fourth than that of bAMLGA, defined by (15).

$$\mathcal{C}(LAmC) = n_C \times \frac{d}{2} \times 2 = n_C d \tag{19}$$

*3) Step 2(f): TR**

The initial implementation of TR requires FEs as it checks the fitness of the twins to replace the less fit one. In iAMLGA, we perform a greedy selection of the twin to be replaced. TR* always inserts a new random chromosome in place of the one having higher index (or stays in the later part of the population). Due to this greedy strategy, the TR* operation of iAMLGA (line 39 – 45 of **Table XIV**) involves no function evaluation.

Therefore, the upper bound of the NFEs needed in one generation or epoch of iAMLGA is,

$$\begin{aligned}\mathcal{C}(iAMLGA) &= \mathcal{C}(Initialize\ AM^*) + \mathcal{C}(Fitness\ Evaluation) \\ &\quad + \mathcal{C}(Elitism\ with\ hGR) + \mathcal{C}(LAmC^*) \\ \mathcal{C}(iAMLGA) &= N + d + N + \mathrm{O}\left(n_E\left[d + \left(\frac{1-r^{hGR}}{\partial r^{hGR}}\right)\right]\right) + n_C d \end{aligned} \tag{20}$$

We comparatively analyzed the performance of bAMLGA and iAMLGA in the following section which demonstrates that the cost reduction strategies not only speed up the evolution process but also result better solutions for most of the test functions. However, iAMLGA mostly compromises the amount of exploitation (**(15)** versus **(19)**) to reduce the cost. Therefore, iAMLGA can be less effective than bAMLGA for the problems where greater exploitation is required. This can be justified under the No-Free-Lunch (NFL) theorem that states that it is impossible to develop a universal and general-purpose optimization algorithm that can solve all classes of problems [74, 75].

VI. Numerical Experiments and Results

In this section, we elaborate the results of the simulations conducted to evaluate and compare the performance of the proposed AMLGA.

A. Test Functions

In order to benchmark the performance of the proposed algorithms, we carried out simulations on 40 test functions that are established in the literature as challengeable enough for performance evaluation [76-78]. There are several characteristics of the test function that features the complexity of the function's landscape like modality, separability, scalability [79]. Modality is defined by the number of ambiguous peaks in the function landscape. A function is called *separable* if the variables of the solution are independent and can be optimized separately. On the other hand, the *inseparable* class of functions are highly difficult as the variables are interrelated and affect each other. The functions that are extendable to arbitrary dimensionality are called *scalable* and otherwise are call *non-scalable.* The scalability often introduces high extent of complexity to the search space such as some test function landscapes become multimodal from unimodal when the dimension are scaled up.

There are some additional features that can make a function landscape even more challenging while searching for minima such as flat surface (less informative area), basins and narrow curved valley. However, some algorithms are found to exploit several shortcomings [80] of popularly used test functions such as the global optimum with same values for different variables [81], global optimum lying in the center of the landscape, at the origin [81], along the axes [82, 83], on the bounds [84] etc. Thereafter, the conventional test functions are rotated and/or shifted before using them to test optimization algorithms [18, 76, 77, 80, 85]. In addition to using rotated and shifted functions, we hybridized functions according to [77] to generate highly complex real-world optimization problems to test the capacity of the algorithms to a limit. We collected the rotation matrix, shifting vector and shuffled-indices vector for rotation, shifting and hybridization, respectively from a shared link of data used in CEC 2014[1].

Among 40, 27 functions are classical functions whereas 13 functions are rotated and/or shifted, and hybridized to construct modified functions. Depending on the three basic properties (modality, separability and scalability) mentioned above, we classify the 27 functions under consideration into six groups (Group I through VI). The rest of the 13 modified functions are categorized into three groups according their complexity. In the following, we list up the definition of the functions under different groups with their corresponding search ranges ($[lb, ub]$) of solution variable ($\boldsymbol{X}$), global optimum ($\boldsymbol{X}^*$) and the function value at global optimum, $f(\boldsymbol{X}^*)$.

1) Group I: Unimodal, Separable and Scalable functions

Group I is comprised of 4 unimodal functions with only one minima in the landscape. The number of variables (d) is extensible to a higher value, however the variables are separable. Therefore, it is relatively easier to minimize these functions and are mainly utilized to test the convergence speed of the algorithms.

1. Sphere function

$$f_{Sphere}(\boldsymbol{X}) = \sum_{i=1}^{d} x_i^2$$
$$[lb, ub] = [-100, 100]^d, \boldsymbol{X}^* = [0, \dots, 0]^T, f_{Sphere}(\boldsymbol{X}^*) = 0$$

2. Cigar or Bent Cigar function

$$f_{Cigar}(\boldsymbol{X}) = x_1^2 + 10^6 \sum_{i=2}^{d} x_i^2$$
$$[lb, ub] = [-100, 100]^d, \boldsymbol{X}^* = [0, \dots, 0]^T, f_{Cigar}(\boldsymbol{X}^*) = 0$$

3. Discus function

$$f_{Discus}(\boldsymbol{X}) = 10^6 x_1^2 + \sum_{i=2}^{d} x_i^2$$
$$[lb, ub] = [-100, 100]^d, \boldsymbol{X}^* = [0, \dots, 0]^T, f_{Discus}(\boldsymbol{X}^*) = 0$$

[1] http://web.mysites.ntu.edu.sg/epnsugan/PublicSite/Shared Documents/Forms/AllItems.aspx

4. Rotated Hyper-ellipsoid (RHE) function

$$f_{RHE}(\boldsymbol{X}) = \sum_{i=1}^{d}\sum_{j=1}^{i} x_j^2$$
$$[lb, ub] = [-100, 100]^d, \boldsymbol{X}^* = [0, \dots, 0]^T, f_{RHE}(\boldsymbol{X}^*) = 0$$

2) Group II: Unimodal, Inseparable and Non-scalable

Group II contains 3 inseparable challenging functions in spite of being unimodal and 2-dimensional ($d = 2$). The *Leon* function has its minima in a narrow valley of the search space. The landscape of *Easom* function has large flat area around a very small area containing the global minima.

5. Zettl function

$$f_{Zettl}(\boldsymbol{X}) = \frac{1}{4}x_1 + (x_1^2 - 2x_1 + x_2^2)^2$$
$$[lb, ub] = [-5, 5]^d, \boldsymbol{X}^* = [-0.0299, 0]^T, f_{Zettl}(\boldsymbol{X}^*) = -0.003791237$$

6. Leon function

$$f_{Leon}(\boldsymbol{X}) = 100(x_2 - x_1^2)^2 + (1 - x_1)^2$$
$$[lb, ub] = [-1.2, 1.2]^d, \boldsymbol{X}^* = [1, 1]^T, f_{Leon}(\boldsymbol{X}^*) = 0$$

7. Easom function

$$f_{Easom}(\boldsymbol{X}) = -cos(x_1)cos(x_2)e^{[-(x_1-\pi)^2-(x_2-\pi)^2]}$$
$$[lb, ub] = [-100, 100]^d, \boldsymbol{X}^* = [\pi, \pi]^T, f_{Easom}(\boldsymbol{X}^*) = -1$$

3) Group III: Unimodal, Inseparable and Scalable

Functions of Group III simultaneously test the algorithm's convergence speed and capacity to handle inseparable variables with higher dimensions.

8. Zakharov function

$$f_{Zakharov}(\boldsymbol{X}) = \sum_{i=1}^{d} x_i^2 + \left(\frac{1}{2}\sum_{i=1}^{d} ix_i\right)^2 + \left(\frac{1}{2}\sum_{i=1}^{d} ix_i\right)^4$$
$$[lb, ub] = [-5, 10]^d, \boldsymbol{X}^* = [0, \dots, 0]^T, f_{Zakharov}(\boldsymbol{X}^*) = 0$$

9. Schwefel 1.2 or Ridge function

$$f_{Schwefel1.2}(\boldsymbol{X}) = \sum_{i=1}^{d}\left(\sum_{j=1}^{i} x_j\right)^2$$
$$[lb, ub] = [-100, 100]^d, \boldsymbol{X}^* = [0, \dots, 0]^T, f_{Schwefel1.2}(\boldsymbol{X}^*) = 0$$

10. Schwefel 2.2 function

$$f_{Schwefel2.2}(\boldsymbol{X}) = \sum_{i=1}^{d}|x_i| + \prod_{i=1}^{d}|x_i|$$
$$[lb, ub] = [-100, 100]^d, \boldsymbol{X}^* = [0, \dots, 0]^T, f_{Schwefel2.2}(\boldsymbol{X}^*) = 0$$

4) Group IV: Multimodal, Separable and Scalable

Group IV encompasses 4 multimodal functions with many local minima which is considered as the most difficult property of test functions for many well-known global optimization algorithms [18]. The functions of Group IV are scalable and separable. *Rastrigin* function [86] has large number of regularly distributed local minima, wheras *Schwefel 2.26* [86] includes additional difficulty of being less symmetrical. *Michalewicz* function [86] has highly difficult landscape with steep valleys. The steepness (s) defines the level of complexity. Moreover, the global optimum function value changes with the number of variables.

11. Rastrigin function

$$f_{Rastrigin}(\boldsymbol{X}) = 10d + \sum_{i=1}^{d}[x_i^2 - 10cos(2\pi x_i)]$$
$$[lb, ub] = [-5.2, 5.2]^d, \boldsymbol{X}^* = [0, \dots, 0]^T, f_{Rastrigin}(\boldsymbol{X}^*) = 0$$

12. Schwefel 2.26 function

$$f_{Schwefel2.6}(\boldsymbol{X}) = 418.9829d - \sum_{i=1}^{d} x_i sin\left(\sqrt{|x_i|}\right)$$
$$[lb, ub] = [-500, 500]^d, \boldsymbol{X}^* = [420.9687, \dots, 420.9687]^T,$$
$$f_{Schwefel2.6}(\boldsymbol{X}^*) = 0$$

13. Michalewicz function

$$f_{Michalewicz}(\boldsymbol{X}) = -\sum_{i=1}^{d} sin(x_i)\, sin^{2s}\left(\frac{ix_i^2}{\pi}\right)$$
$$steepness, s = 10$$
$$[lb, ub] = [0, \pi]^d, \boldsymbol{X}^* = [2.20, 1.57]^T, d = 2$$
$$f_{Michalewicz}(\boldsymbol{X}^*) = -1.8013, d = 2$$
$$f_{Michalewicz}(\boldsymbol{X}^*) = -4.687658, d = 5$$
$$f_{Michalewicz}(\boldsymbol{X}^*) = -9.66015, d = 10$$

14. Styblinski-Tang (ST) function

$$f_{ST}(\boldsymbol{X}) = \frac{1}{2}\sum_{i=1}^{d}(x_i^4 - 16x_i^2 + 5x_i)$$
$$[lb, ub] = [-5, 5]^d, \boldsymbol{X}^* = [-2.903534, \dots, -2.903534]^T,$$
$$f_{ST}(\boldsymbol{X}^*) = -39.16599d$$

5) Group V: Multimodal, Inseparable and non-Scalable

Group V contains 5 multimodal functions with addition complexity of having inseparable variables. However, these function are nonscalable with two variables ($d = 2$). Out of these functions, Schaffer's F2 and F6 have their global minima surrounded by many maxima as well as local minima. Therefore, they are utilized to stress the algorithms ability to overcome these maxima in addition with the trap of local minima before getting into the global minima.

15. Schaffer F2 function

$$f_{ScafferF2}(\boldsymbol{X}) = 0.5 + \frac{sin^2(x_1^2 - x_2^2) - 0.5}{[1 + 0.001(x_1^2 + x_2^2)]^2}$$
$$[lb, ub] = [-100, 100]^d, \boldsymbol{X}^* = [0, 0]^T, f_{ScafferF2}(\boldsymbol{X}^*) = 0$$

16. Schaffer F6 function

$$f_{ScafferF6}(\boldsymbol{X}) = 0.5 + \frac{sin^2\left(\sqrt{x_1^2 + x_2^2}\right) - 0.5}{[1 + 0.001(x_1^2 + x_2^2)]^2}$$
$$[lb, ub] = [-100, 100]^d, \boldsymbol{X}^* = [0, 0]^T, f_{ScafferF6}(\boldsymbol{X}^*) = 0$$

17. Bird function

$$f_{Bird}(\boldsymbol{x}) = (x_1 - x_2)^2 + sin(x_1)e^{[1-cos(x_2)]^2} + cos(x_2)e^{[1-sin(x_1)]^2}$$
$$[lb, ub] = [-2\pi, 2\pi]^{\boldsymbol{d}}, \boldsymbol{X}^* = [4.701056, 3.152946]^T,$$
$$\boldsymbol{X}^* = [-1.582142, -3.130247]^T, f_{Bird}(\boldsymbol{x}^*) = -106.7645367198034$$

18. Levy 13 function

$$f_{Levy13}(\boldsymbol{X}) = sin^2(3\pi x_1) + (x_1 - 1)^2[1 + sin^2(3\pi x_2)] + (x_2 - 1)^2[1 + sin^2(2\pi x_2)]$$
$$[lb, ub] = [-10, 10]^d, \boldsymbol{X}^* = [1, 1]^T, f_{Levy13}(\boldsymbol{X}^*) = 0$$

19. Carrom Table (CT) function

$$f_{CarromTable}(\boldsymbol{X}) = -\frac{1}{30}e^{2\left|1-\frac{\sqrt{x_1^2+x_2^2}}{\pi}\right|}cos^2(x_1)cos^2(x_2)$$

$[lb, ub] = [-10, 10]^d, \boldsymbol{X}^* = [\pm 9.646157, \pm 9.646157\ ]^T$
$f_{CarromTable}(\boldsymbol{X}^*) = -24.1568155$

6) Group VI: Multimodal, Inseparable and Scalable

Group – VI is the one of the most challenging set that aggregates 8 multimodal functions which are scalable as well as inseparable. Therefore, they can be extended to arbitrary dimensionality (variables) allowing for scaling investigation and these variables are interrelated with each other. The *Ackley* function consists of a single funnel with many local minima around it that has potential relevance with real-world applications such as the free energy hypersurface of proteins [87]. *Rosenbrock* function is a difficult function to be optimized as the global optima resides in a narrow, long flat valley [88]. For *Griewank* function, the number of local minima increases exponentially with the dimension [89]. The *Sine Envelope Sine Wave* (SESW) function is the scalable form of Schaffer's function F6 which gives repeating couplets of optimal values and has a concentric barrier around the global minima [90]. The *Lunacek* function's, also known as bi- or, double Rastrigin function, landscape has two funnels to be deceptive for the search algorithms.

20. Ackley function

$$f_{Ackely}(\boldsymbol{X}) = -ae^{\left(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d} x_i^2}\right)} - e^{\left(\frac{1}{d}\sum_{i=1}^{d} cos(cx_i)\right)} + a + e^1$$
$a = 20, b = 0.2, c = 2\pi$
$[lb, ub] = [-32, 32]^d, \boldsymbol{X}^* = [0, \dots, 0]^T, f_{Ackley}(\boldsymbol{X}^*) = 0$

21. Rosenbrock function

$$f_{Rosenbrock}(\boldsymbol{X}) = \sum_{i=1}^{d-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$$
$[lb, ub] = [-30, 30]^d, \boldsymbol{X}^* = [1, \dots, 1]^T, f_{Rosenbrock}(\boldsymbol{X}^*) = 0$

22. Griewank function

$$f_{Griewank}(\boldsymbol{X}) = \sum_{i=1}^{d} \frac{x_i^2}{4000} - \prod_{i=1}^{d} cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$$
$[lb, ub] = [-600, 600]^d, \boldsymbol{X}^* = [0, \dots, 0]^T, f_{Griewank}(\boldsymbol{X}^*) = 0$

23. Sine Envelope Sine Wave (SESW) function

$$f_{SESW}(\boldsymbol{X}) = \sum_{i=1}^{d-1} \left(\frac{sin^2\left(\sqrt{x_1^2 + x_2^2}\right) - 0.5}{[1 + 0.001(x_1^2 + x_2^2)]^2} + 0.5 \right)$$
$[lb, ub] = [-100, 100]^d, \boldsymbol{X}^* = [0, \dots, 0]^T, f_{SESW}(\boldsymbol{X}^*) = 0$

24. Trigonometric function

$$f_{Trigonometric}(\boldsymbol{X}) = \sum_{i=1}^{d} \left[m + i(1 - cosx_i) - sinx_i - \sum_{j=1}^{d} cosx_j \right]^2$$
$[lb, ub] = [-1000, 1000]^d, \boldsymbol{X}^* = [0, \dots, 0\ ]^T, f_{Trigonometric}(\boldsymbol{X}^*) = 0$

25. Levy function

$$f_{Levy}(\boldsymbol{X}) = sin^2(\pi y_1) + \sum_{i=1}^{d-1} \left[(y_i - 1)^2 \left(1 + 10(sin^2 y_{i+1})\right) \right] + (y_n - 1)^2 \left(1 + sin^2(2\pi y_n)\right), y_i = 1 + \frac{1}{4}(x_i + 1)$$
$[lb, ub] = [-50, 50]^d, \boldsymbol{X}^* = [-1, \dots, -1\ ]^T, f_{Levy}(\boldsymbol{X}^*) = 0$

26. Schaffer F7 function

$$f_{ScafferF7}(\boldsymbol{X}) = \left[\frac{1}{(d-1)} \sum_{i=1}^{d-1} \left(\sqrt{y_i} + sin(50 {y_i}^{0.2}) \sqrt{y_i}\right) \right]^2$$
$y_i = \sqrt{x_i^2 + x_{i+1}^2}$
$[lb, ub] = [-100, 100]^d, \boldsymbol{X}^* = [0, \dots, 0\ ]^T, f_{ScafferF7}(\boldsymbol{X}^*) = 0$

27. Lunacek Function

$$f_{Lunacek}(\boldsymbol{x}) = \left[min\left(\left\{ \sum_{i=1}^{d} (x_i - \mu_1)^2 \right\}, \left\{ 1.d + s \sum_{i=1}^{d} (x_i - \mu_2)^2 \right\} \right) \right] + 10 \sum_{i=1}^{d} 1 - cos 2\pi (x_i - \mu_1)$$
$\mu_1 = 2.5, s = 1 - \frac{1}{2\sqrt{d + 20} - 8.2}, \mu_2 = -\sqrt{\frac{\mu_1^2 - 1}{s}}$
$[lb, ub] = [-10, 10]^d, \boldsymbol{X}^* = [\mu_1, \dots, \mu_1\ ]^T, f_{Lunacek}(\boldsymbol{X}^*) = 0$

7) Group XII: Rotated functions

Rotation of the landscape of the functions using an orthogonal rotation matrix ($\mathbb{R}$) [91] can prevent the optimum from lying along the coordinate axes. We rotate 7 scalable functions using **(21)** that are already listed above of which 2 are unimodal and the rests of the 5 are multimodal.

$$f(Z), \qquad \boldsymbol{Z} = \mathbb{R} \times \boldsymbol{X} \tag{21}$$

28. Rotated Cigar or Bent Cigar function

$f_{R-Cigar}(\boldsymbol{Z}) = f_{Cigar}(\boldsymbol{Z}), \boldsymbol{Z} = \mathbb{R} \times \boldsymbol{X}$
$[lb, ub] = [-100, 100]^d, \boldsymbol{X}^* = [0, \dots, 0]^T, f_{R-Cigar}(\boldsymbol{X}^*) = 0$

29. Rotated Discus function

$f_{R-Discus}(\boldsymbol{Z}) = f_{Discus}(\boldsymbol{Z}), \boldsymbol{Z} = \mathbb{R} \times \boldsymbol{X}$
$[lb, ub] = [-100, 100]^d, \boldsymbol{X}^* = [0, \dots, 0]^T, f_{R-Discus}(\boldsymbol{X}^*) = 0$

30. Rotated Rastrigin function

$f_{R-Rastrigin}(\boldsymbol{Z}) = f_{Rastrigin}(\boldsymbol{Z}), \boldsymbol{Z} = \mathbb{R} \times (\boldsymbol{X} \times (5.2/100))$
$[lb, ub] = [-5.2, 5.2]^d, \boldsymbol{X}^* = [0, \dots, 0]^T, f_{R-Rastrigin}(\boldsymbol{X}^*) = 0$

31. Rotated Ackley function

$f_{R-Ackley}(\boldsymbol{Z}) = f_{Ackley}(\boldsymbol{Z})\ , \boldsymbol{Z} = \mathbb{R} \times (\boldsymbol{X} \times (32/100))$
$[lb, ub] = [-32, 32]^d, \boldsymbol{X}^* = [0, \dots, 0]^T, f_{R-Ackley}(\boldsymbol{X}^*) = 0$

32. Rotated Griewank function

$f_{R-Griewank}(\boldsymbol{Z}) = f_{Griewank}(\boldsymbol{Z})\ , \boldsymbol{Z} = \mathbb{R} \times (\boldsymbol{X} \times (600/100))$
$[lb, ub] = [-600, 600]^d, \boldsymbol{X}^* = [0, \dots, 0]^T, f_{R-Griewank}(\boldsymbol{X}^*) = 0$

33. Rotated Schaffer F7 function

$f_{R-SchafferF7}(\boldsymbol{Z}) = f_{SchafferF7}(\boldsymbol{Z}), \boldsymbol{Z} = \mathbb{R} \times \boldsymbol{X}$
$[lb, ub] = [-100, 100]^d, \boldsymbol{X}^* = [0, \dots, 0]^T, f_{R-SchafferF7}(\boldsymbol{X}^*) = 0$

34. Rotated Lunacek function

$f_{R-Lunacek}(\boldsymbol{Z}) = f_{Lunacek}(\boldsymbol{Z}), \boldsymbol{Z} = \mathbb{R} \times (\boldsymbol{X} \times (10/100))$
$[lb, ub] = [-10, 10]^d, \boldsymbol{X}^* = [\mu_1, \dots, \mu_1\ ]^T, f_{R-Lunacek}(\boldsymbol{X}^*) = 0$

8) Group XIII: Shifted and Rotated functions

Shifting operation moves the global optimum to a new random position ($\boldsymbol{o}_{new}$) from the old optimum ($\boldsymbol{o}_{old}$) to avoid the limitation of having same values of the variables at optima or having optimum at the center. Under this group, we shift 3

multimodal functions, and further rotated to make them highly intractable using **(22)**.

$$f(Z), \qquad \boldsymbol{Z} = \mathbb{R} \times (\boldsymbol{X} - \boldsymbol{o}_{new} + \boldsymbol{o}_{old}) \tag{22}$$

35. Shifted and Rotated Rastrigin function

$f_{SR-Rastrigin}(\boldsymbol{Z}) = f_{Rastrigin}(\boldsymbol{Z})$,
$\boldsymbol{Z} = \mathbb{R} \times ((\boldsymbol{X} - \boldsymbol{o}_{new} + \boldsymbol{o}_{old}) \times (5.2/100))$
$[lb, ub] = [-5.2, 5.2]^d$, $\boldsymbol{X}^* = [0, \dots, 0]^T$, $f_{SR-Rastrigin}(\boldsymbol{X}^*) = 0$

36. Shifted and Rotated Ackley function

$f_{SR-Ackley}(\boldsymbol{Z}) = f_{Ackley}(\boldsymbol{Z})$,
$\boldsymbol{Z} = \mathbb{R} \times ((\boldsymbol{X} - \boldsymbol{o}_{new} + \boldsymbol{o}_{old}) \times (32/100))$
$[lb, ub] = [-32, 32]^d$, $\boldsymbol{X}^* = [0, \dots, 0]^T$, $f_{SR-Ackley}(\boldsymbol{X}^*) = 0$

37. Shifted and Rotated Lunacek function

$f_{SR-Lunacek}(\boldsymbol{Z}) = f_{Lunacek}(\boldsymbol{Z})$,
$\boldsymbol{Z} = \mathbb{R} \times ((\boldsymbol{X} - \boldsymbol{o}_{new} + \boldsymbol{o}_{old}) \times (10/100))$
$[lb, ub] = [-10, 10]^d$, $\boldsymbol{X}^* = [\mu_1, \dots, \mu_1]^T$, $f_{SR-Lunacek}(\boldsymbol{X}^*) = 0$

9) Group IX: Hybrid functions

Hybrid function defined by **(23)** represents real-world optimization problems where different subcomponents of the variables can have different characteristics. For these functions, the variables are randomly shuffled and divided into subcomponents and then different basic functions are used for different subcomponents. In our hybrid functions, we have further rotated the basic functions to be optimized.

$$f(\boldsymbol{Z}) = f_1(\mathbb{R}_1\boldsymbol{Z}_1) + f_2(\mathbb{R}_2\boldsymbol{Z}_2) + \cdots + f_n(\mathbb{R}_n\boldsymbol{Z}_n) \tag{23}$$

Here, $\boldsymbol{Z} = [Z_1, Z_2, \dots, Z_n]$
$Z_1 = \left[x_{S_1}, \dots, x_{S_{d_1}}\right]$, $Z_2 = \left[x_{S_{d_1+1}}, \dots, x_{S_{d_1+d_2}}\right]$,… $Z_n = \left[x_{S_{\sum_{i=1}^{n-1} d_i+1}}, \dots, x_{S_d}\right]$
n is number of basic functions
$S = random\_permutation(1{:}\,d)$
f_i is the i^{th} basic function
d_i is the dimension of the i^{th} basic function, $\sum_{i=1}^{n} d_i = d$
p_i is the percentage of the variables for i^{th} basic function
$d_1 = \lceil p_1 \times d \rceil$, $d_2 = \lceil p_2 \times d \rceil$, …, $d_n = d - \sum_{i=1}^{n-1} d_i$

38. Hybrid function – 1

$f_{Hybrid-1}(\boldsymbol{Z}) = f_{Rastrigin}(\mathbb{R}_1\boldsymbol{Z}_1) + f_{Cigar}(\mathbb{R}_2\boldsymbol{Z}_1) + f_{Griewank}(\mathbb{R}_n\boldsymbol{Z}_1)$
$[p_1, p_2, p_3] = [1/3, 1/3, 1/3]$ for d = 30
$[p_1, p_2, p_3] = [2/5, 1/5, 2/5]$ for d = 50
$[lb, ub] = [-100, 100]^d$, $f_{Hybrid-1}(\boldsymbol{X}^*) = 0$

39. Hybrid function – 2

$f_{Hybrid-2}(\boldsymbol{Z}) = f_{Sphere}(\mathbb{R}_1\boldsymbol{Z}_1) + f_{Ackley}(\mathbb{R}_2\boldsymbol{Z}_1) + f_{SchafferF7}(\mathbb{R}_n\boldsymbol{Z}_1)$
$[p_1, p_2, p_3] = [1/3, 1/3, 1/3]$ for d = 30
$[p_1, p_2, p_3] = [2/5, 1/5, 2/5]$ for d = 50
$[lb, ub] = [-100, 100]^d$, $f_{Hybrid-2}(\boldsymbol{X}^*) = 0$

40. Hybrid function – 3

$f_{Hybrid-3}(\boldsymbol{Z}) = f_{Discus}(\mathbb{R}_1\boldsymbol{Z}_1) + f_{Rosenbrock}(\mathbb{R}_2\boldsymbol{Z}_1) + f_{Griewank}(\mathbb{R}_n\boldsymbol{Z}_1)$
$[p_1, p_2, p_3] = [1/3, 1/3, 1/3]$ for d = 30
$[p_1, p_2, p_3] = [2/5, 1/5, 2/5]$ for d = 50
$[lb, ub] = [-100, 100]^d$, $f_{Hybrid-3}(\boldsymbol{X}^*) = 0$

B. Parameter Settings

Implementation of GA involves the setting of several control parameters. Following setup of the parameters has been used for the simulations conducted in this study.

- Number of iterations/epoch, I_{max} = 2000
- Number of chromosomes/Population size, N = 200
- Rate of elites, $r^E = 0.1$ (10%). Thus, the number of elites, $n_E = 200 \times 0.1 = 20$
- Initial rate of hGR, $r^{hGR} = 0.1$ (10%)
- Rate of increasing hGR, , $\partial r^{hGR} = 0.05$ (5%)
- Rate of crossover, $r^C = 0.8$ (80%). Thus, the number of crossover, $n_C = (200 \times 0.8)/2 = 160$
- Rate of mutation, $r^M = 0.05$ (5%). Thus, the number of mutation, $n_M = 200 \times 0.05 = 10$
- Range of Chromosome correlation factor (CCF) = [1.0, 0.8]
- Rate of decreasing CCF, $\partial CCF = 0.015\%$

C. Comparison among GA Variants

In this section, we compare the performances of 7 GA variants discussed in Section IV and the basic AMLGA (bAMLGA) introduced in the Section V of this article. Among these 8 variants, four algorithms use global operators: SGA (Table V), FNGA (Table VI), TRGA (Table VII), KGA (Table VIII) and four use local operators: LSGA (Table IX), LTRGA (Table X), hGRGA (Table XI) and bAMLGA (Table XII). This comparative analysis highlights the respective merits and detriments of global and local operators, and different combination of genetic operators in order to have appropriate exploitation and exploration. It further highlights the effectiveness of adaptive and memory-assisted application of the local operators. We ran the 8 algorithms for 30 cycles on each of the 27 basic test functions. A cycle was terminated if the known global optima was found or the maximum number of epochs has been executed. In **Table XIV**, we present the best (*), mean (μ), standard deviation (σ) and the average number of iterations/epoch (ε) required to achieve the best result out of 30 cycles. Here, the mean epoch has been reported as a measure of convergence speed. The table further shows the number of bits utilized (m) to encode each of the variables (gene) different functions which is set according to the domain range of the search space. For the scalable functions (Group I, III, IV, VI), we reported the results with dimension, $d = 30$. The only exception is Michalewicz function with $d = 10$. For non-scalable functions (Group II and V), the results correspond to $d = 2$.

Group I's separable and unimodal functions were used mainly to test the algorithms exploitation capacity and speed of convergence. GA variations with global single point crossover, mutation, AmC and TR, namely SGA, FNGA, TRGA and KGA's performances were approximately similar (**Table XIV(*a*)**). The disruption effect of d-point or, local crossover is explicitly visible here. LSGA and LTRGA could not reach the global minima ('0') for any of the functions and even to achieve their best result, the average number of epoch required was very high. The local hGR operator with single point crossover in hGRGA performed quite effectively. However, the combination of adaptive and memory-assisted local crossover and hGR in bAMLGA outperformed all the variants with the fastest convergence speed.

Three functions of Group II are only two dimensional, yet difficult to optimize due to inseparable variables and complex landscapes (**Table XIV(*b*)**). For Zettl function, all the algorithms could reach the single global minima. KGA with global AmC and TR operator converged very quickly. For Leon and Easom functions, LTRGA resulted best average minima. However, bAMLGA provided comparable results along with the fastest convergence. The overall scenario conveys that for these low-scale (d = 2) functions with relatively flat landscape, the global and local operators can give comparative performance.

Group III is the most difficult unimodal function set that are scalable and designed to test the intensification (exploitation) capacity of the algorithms defined by a large number of inseparable variables. Simulation results on these functions (**Table XIV(*c*)**) justify the enhanced ability of LAmC to exploit the search space than that of classical crossover (*Algorithms*: SGA, TRGA, LSGA, LTRGA) and global AmC (*Algorithm*s: FNGA, KGA). Moreover, hGRGA that includes a local operator, hGR failed to reach the minima for Zakharov and Schewfel 1.2 functions. Only bAMLGA could converge into the global minima for all three functions with the fastest convergence. These results show the benefit of LAmC operation in addition to the hGR operator.

TABLE XIV
EXPERIMENTAL RESULTS OF 8 GA VARIANTS ON 27 BASIC TEST FUNCTIONS

			(*a*) *Group I: Unimodal, Separable, Scalable*							
			GA variants with global operators				GA variants with local operators			
			SGA	**FNGA**	**TRGA**	**KGA**	**LSGA**	**LTRGA**	**hGRGA**	**bAMLGA**
f_{Sphere}	1+7+13=21	*	**0**	**0**	**0**	**0**	2.60e+02	2.99e+02	**0**	**0**
		μ	**0**	**0**	**0**	**0**	5.08e+02	4.93e+02	**0**	**0**
		σ	**0**	**0**	**0**	**0**	1.30e+02	9.63e+01	**0**	**0**
		ε	6.23e+02	6.21e+02	6.40e+02	6.12e+02	1.83e+03	1.80e+03	1.74e+01	**5.37e+00**
f_{Cigar}	1+7+12=20	*	**0**	**0**	**0**	**0**	3.05e+08	2.29e+08	**0**	**0**
		μ	**0**	**0**	**0**	**0**	5.23e+08	4.44e+08	**0**	**0**
		σ	**0**	**0**	**0**	**0**	1.27e+08	8.93e+07	**0**	**0**
		ε	5.86e+02	5.80e+02	5.80e+02	6.44e+02	1.84e+03	1.84e+03	1.53e+01	**5.17e+00**
f_{Discus}	1+7+12=20	*	**0**	**0**	**0**	**0**	7.23e+02	6.63e+02	**0**	**0**
		μ	**0**	**0**	**0**	**0**	1.21e+03	1.32e+03	**0**	**0**
		σ	**0**	**0**	**0**	**0**	2.82e+02	3.83e+02	**0**	**0**
		ε	5.65e+02	6.28e+02	5.98e+02	594.6000	1.87e+02	1.82e+03	1.44e+01	**4.00e+00**
f_{Rose}	1+7+12 =20	*	**0**	**0**	**0**	**0**	4.32e+03	3.04e+03	**0**	**0**
		μ	**0**	**0**	**0**	**0**	7.49e+03	1.58e+03	**0**	**0**
		σ	**0**	**0**	**0**	**0**	1.54e+03	8.94e+03	**0**	**0**
		ε	6.19e+02	5.91e+02	6.09e+02	5.67e+02	1.76e+03	1.83e+03	2.13e+02	**4.53e+00**

			(*b*) *Group II: Unimodal, Inseparable, Non-scalable*							
			GA variants with global operators				GA variants with local operators			
			SGA	**FNGA**	**TRGA**	**KGA**	**LSGA**	**LTRGA**	**hGRGA**	**bAMLGA**
f_{Zettl}	1+3+25=29	*	**-3.79e-03**	**-3.79e-03**	**-3.79e-03**	**-3.79e-03**	**-3.79e-03**	**-3.79e-03**	**-3.79e-03**	**-3.79e-03**
		μ	5.34e-03	3.03e-03	**-3.79e-03**	**-3.79e-03**	**-3.79e-03**	**-3.79e-03**	**-3.79e-03**	**-3.79e-03**
		σ	2.09e-02	1.86e-02	3.51e-06	3.91e-06	3.78e-06	3.70e-06	3.54e-06	3.89e-06
		ε	4.76e+02	4.25e+02	1.54e+02	**6.18e+01**	5.58e+02	5.35e+02	2.04e+02	2.84e+02
f_{Leon}	1+1+16=18	*	3.82e-08	8.31e-08	2.33e-10	1.89e-08	2.33e-10	**0**	**0**	**0**
		μ	1.88e-02	1.71e-02	1.35e-05	1.67e-05	2.29e-04	**3.20e-08**	3.55e-06	4.37e-08
		σ	4.02e-02	3.94e-02	1.99e-05	3.09e-05	7.39e-04	9.56e-08	1.53e-05	1.91e-07
		ε	9.67e+02	9.72e+02	9.02e+02	8.36e+02	4.45e+02	5.84e+02	5.84e+02	**2.79e+02**
f_{Easom}	1+7+19=27	*	**-1**	**-1**	**-1**	**-1**	**-1**	**-1**	**-1**	**-1**
		μ	-7.67e-01	-7.55e-01	-9.98e-01	-9.99e-01	-8.51e-01	**-1**	-9.97e-01	**-9.99e-01**
		σ	3.29e-01	3.63e-01	7.60e-03	3.20e-03	3.08e-01	**1.01e-06**	9.10e-03	3.36e-04
		ε	6.55e+02	7.51e+02	1.30e+03	1.17e+03	7.25e+02	9.55e+02	1.00e+03	**6.80e+02**

			(*c*) *Group III: Unimodal, Inseparable, Scalable*							
			GA variants with global operators				GA variants with local operators			
			SGA	**FNGA**	**TRGA**	**KGA**	**LSGA**	**LTRGA**	**hGRGA**	**bAMLGA**
$f_{Zakharov}$	1+4+12=17	*	2.07e+02	2.06e+02	1.52e+02	1.67e+02	5.27e+02	5.25e+02	2.40e-03	**0**
		μ	3.45e+02	3.33e+02	2.34e+02	2.33e+02	8.48e+01	8.58e+01	9.45e+01	**0**
		σ	7.81e+01	8.07e+01	3.99e+01	4.48e+01	1.77e+01	2.03e+01	6.50e+01	**0**
		ε	1.98e+03	1.96e+03	1.98e+03	1.93e+03	1.61e+03	1.60e+03	1.90e+03	**7.7**
$f_{Schewfel1}$	1+7+12=2	*	1.97e+04	2.05e+04	1.30e+04	1.01e+04	6.48e+04	5.90e+04	6.02e-02	**0**
		μ	3.20e+04	3.18e+04	2.21e+04	2.16e+04	1.25e+04	1.21e+04	7.25e+03	**0**
		σ	7.82e+03	7.29e+03	5.36e+03	4.67e+03	3.60e+03	3.72e+03	6.97e+03	**0**
		ε	1.88e+03	1.86e+03	1.95e+03	1.74e+03	1.73e+03	1.86e+03	1.64e+03	**6.5**
$f_{Schwefel22}$	1+7+12=20	*	**0**	**0**	**0**	**0**	2.12e+02	1.97e+03	**0**	**0**
		μ	**0**	**0**	**0**	**0**	2.74e+02	2.76e+02	**0**	**0**
		σ	**0**	**0**	**0**	**0**	4.07e+01	4.08e+01	**0**	**0**
		ε	6.12e+02	5.96e+02	5.96e+02	6.11e+02	1.77e+03	1.79e+03	1.11e+01	**4.63e+00**

			(*d*) *Group IV: Multimodal, Separable, Scalable*							
			GA variants with global operators				GA variants with local operators			
			SGA	**FNGA**	**TRGA**	**KGA**	**LSGA**	**LTRGA**	**hGRGA**	**bAMLGA**
$f_{Rastrain}$	1+3+17=21	*	8.95e+00	8.96e+00	7.96e+00	7.96e+00	5.01e+01	3.76e+01	**0**	**0**
		μ	1.30e+01	1.39e+01	1.23e+01	1.28e+01	7.13e+01	5.15e+01	**0**	**0**
		σ	2.29e+00	2.48e+00	2.58e+00	2.24e+00	9.10e+00	8.63e+00	**0**	**0**
		ε	6.67e+02	7.53e+02	6.67e+02	6.76e+02	1.68e+03	1.63e+03	1.56e+01	**8.2e+00**
$f_{Schwefel2.26}$	1+9+16=26	*	2.20e+03	1.62e+03	7.28e+02	1.35e+03	2.25e+03	2.17e+03	**3.82e-04**	**3.82e-04**
		μ	2.94e+03	2.86e+03	2.18e+03	2.31e+03	3.26e+03	3.22e+03	1.90e-03	**1.70e-03**
		σ	5.44e+02	5.91e+02	6.05e+02	5.30e+02	5.82e+02	5.46e+02	**1.80e-03**	**1.80e-03**
		ε	1.31e+03	1.22e+03	1.77e+03	1.48e+03	1.75e+03	1.86e+03	**2.02e+01**	2.08e+01
$f_{Michalewicz}$	1+2+19=22	*	-9.655	-9.630	-9.659	-9.659	-9.447	-9.654	-9.659	**-9.660**
		μ	-9.541	-9.438	-9.567	-9.584	-9.007	-9.323	-9.607	**-9.650**
		σ	1.28e-01	1.34e-01	8.22e-02	8.06e-02	2.20e-01	2.02e-01	5.03e-02	**2.96e-02**
		ε	1.64e+03	1.55e+03	1.60e+03	1.62e+03	1.79e+03	1.76e+03	1.54e+03	1.42e+03
f_{ST}	1+3+25=29	*	-1174.02	-1174.26	-1174.54	-1174.55	-1164.65	-1174.12	**-1174.99**	**-1174.99**
		μ	-1169.34	-1171.16	-1172.08	-1172.81	-1147.00	-1173.20	**-1174.99**	**-1174.99**
		σ	5.82e+00	4.26e+00	3.68e+00	3.16e+00	1.07e+01	4.32e-01	**3.69e-12**	3.39e-12
		ε	1.91e+03	1.83e+03	1.87e+03	1.84e+03	1.76e+03	1.80e+03	4.53e+02	3.38e+02

			(*e*) *Group V: Multimodal, Inseparable, Non-scalable*							
			GA variants with global operators				GA variants with local operators			
			SGA	**FNGA**	**TRGA**	**KGA**	**LSGA**	**LTRGA**	**hGRGA**	**bAMLGA**
$f_{SchafferF6}$	1+7+19=27	*	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
		μ	6.60e-03	5.00e-03	1.00e-04	5.09e-05	6.85e-04	**0**	**0**	**0**
		σ	1.52e-02	1.32e-02	1.53e-04	7.06e-05	2.20e-03	**0**	**0**	**0**
		ε	8.53e+02	9.38e+02	7.21e+02	8.64e+02	4.87e+02	1.41e+02	**3.99e+01**	4.53e+01
$f_{SchafferF6}$	1+7+15=23	*	9.70e-03	9.70e-03	**0**	**0**	**0**	**0**	**0**	**0**
		μ	2.44e-02	2.58e-02	7.40e-03	8.40e-03	1.09e-02	6.20e-03	3.23e-04	**0**
		σ	1.40e-02	1.69e-02	4.20e-03	3.40e-03	7.60e-03	4.80e-03	1.80e-03	**0**
		ε	5.48e+02	4.35e+02	3.95e+02	5.84e+02	6.46e+02	4.65e+02	**1.32e+02**	2.77e+02
f_{Rast}	1+3+25=29	*	**-106.764**	**-106.764**	**-106.764**	**-106.764**	**-106.764**	**-106.764**	**-106.764**	**-106.764**
		μ	-106.556	-106.584	-106.763	-106.762	-106.520	-106.763	**-106.764**	**-106.764**
		σ	3.35e-01	3.42e-01	1.50e-03	3.60e-03	3.88e-01	1.70e-03	1.10e-03	1.60e-03
		ε	3.46e+02	5.04e+02	5.92e+02	6.26e+02	9.29e+02	7.34e+02	7.34e+02	7.78e+02
$f^{\dagger}_{Levy13}$	1+4+25=30	*	**1.35e-31**	**1.35e-31**	**1.35e-31**	**1.35e-31**	8.88e-16	**1.35e-31**	**1.35e-31**	**1.35e-31**
		μ	8.07e-02	1.00e-01	4.56e-14	**2.71e-14**	4.12e-02	7.31e-14	5.38e-14	3.76e-14
		σ	1.23e-01	1.50e-01	4.02e-14	3.82e-14	5.52e-02	7.35e-14	3.87e-14	**4.09e-14**
		ε	2.83e+02	3.26e+02	8.84e+01	1.05e+02	5.57e+02	2.88e+02	2.87e+01	**1.47e+01**
$f_{CarromTable}$	1+4+35=40	*	**-24.1568**	**-24.1568**	**-24.1568**	**-24.1568**	**-24.1568**	**-24.1568**	**-24.1568**	**-24.1568**
		μ	-24.1544	-24.1541	-14.1567	**-24.1568**	**-24.1568**	**-24.1568**	**-24.1568**	**-24.1568**
		σ	4.10e-03	4.30e-03	2.31e-04	1.44e-04	2.80e-03	2.33e-04	**1.56e-06**	4.76e-05
		ε	1.03e+03	9.95e+02	1.41e+03	1.36e+03	1.22e+03	1.36e+03	1.15e+03	**8.04e+02**

			(*f*) *Group VI: Multimodal, Inseparable, Scalable*							
			GA variants with global operators				GA variants with local operators			
			SGA	**FNGA**	**TRGA**	**KGA**	**LSGA**	**LTRGA**	**hGRGA**	**bAMLGA**
$f^{\dagger}_{Ackley}$	1+6+16=23	*	1.65e+00	1.65e+00	7.44e-01	1.78e+00	5.51e+00	4.14e+00	**8.88e-16**	**8.88e-16**
		μ	2.28e+00	2.23e+00	2.18e+00	2.21e+00	6.53e+00	5.12e+00	**8.88e-16**	**8.88e-16**
		σ	3.06e-01	2.56e-01	3.88e-01	2.60e-01	4.79e-01	5.68e-01	**0**	**0**
		ε	1.59e+03	1.68e+03	1.64e+03	1.70e+03	1.75e+03	1.65e+03	1.73e+01	**7.33e+00**
$f_{Rosenbrock}$	1+5+16=22	*	6.80e+01	7.46e+01	5.74e+00	2.92e+01	4.15e+03	4.18e+03	**0**	**0**
		μ	1.04e+03	1.67e+03	1.38e+03	1.35e+03	1.38e+04	1.34e+04	1.70e-04	**9.84e-05**
		σ	1.10e+03	1.53e+03	1.87e+03	1.34e+03	5.20e+03	4.55e+03	6.95e-04	**5.25e-04**
		ε	1.59e+03	1.60e+03	1.61e+03	1.57e+03	1.85e+03	1.89e+03	3.72e+02	**6.04e+01**
$f_{Griewank}$	1+10+16=27	*	4.25e-02	0	2.04e-02	5.26e-02	2.24e+00	2.29e+00	**0**	**0**
		μ	3.58e-01	4.05e-01	4.15e-01	4.05e-01	3.36e+00	3.37e+00	**0**	**0**
		σ	2.91e-01	4.08e-01	2.40e-01	3.10e-01	7.71e-01	7.97e-01	**0**	**0**
		ε	1.03e+03	9.82e+02	1.05e+03	9.18e+02	1.85e+03	1.87e+03	2.96e+01	**1.30e+01**
f_{SSW}	1+7+15=23	*	3.73e-01	3.73e-01	4.30e-01	4.52e-01	3.47e-01	3.12e-01	**0**	**0**
		μ	4.45e-01	4.53e-01	4.60e-01	4.79e-01	4.19e-01	3.91e-01	3.17e-01	**1.43e-01**
		σ	3.15e-02	2.93e-02	1.83e-02	**1.40e-02**	2.39e-02	3.45e-02	1.65e-01	1.07e-01
		ε	1.76e+03	1.78e+03	1.38e+03	1.44e+03	1.75e+03	1.80e+03	8.66e+02	**5.01e+02**
$f_{Trigonometric}$	1+10+7=18	*	5.12e-02	1.85e-02	1.00e-02	8.77e-02	5.84e+03	5.12e-02	3.62e-09	**1.82e-10**
		μ	1.36e+01	1.77e+01	1.77e+01	1.75e+01	8.68e+03	1.36e+01	7.49e-05	**6.50e-05**
		σ	2.29e+01	2.87e+01	2.41e+01	2.15e+01	1.65e+03	2.29e+01	1.55e-04	1.17e-04
		ε	9.83e+02	8.21e+02	8.00e+02	8.71e+02	9.27e+02	9.83e+02	1.08e+02	1.15e+02
$f^{\ddagger}_{Levy}$	1+6+12=19	*	4.95e-01	4.95e-01	3.95e-07	9.58e-01	6.75e+01	8.02e+01	**1.50e-32**	**1.50e-32**
		μ	2.27e+00	1.81e+00	2.05e+00	2.56e+00	1.35e+02	1.25e+02	**1.34e-06**	6.51e-05
		σ	1.02e+00	9.20e-01	1.05e+00	1.07e+00	4.73e+01	3.75e+01	**4.75e-06**	2.44e-04
		ε	1.22e+03	1.15e+03	1.15e+03	1.14e+03	1.76e+03	1.70e+03	2.59e+02	**3.11e+01**
$f_{SatFerF7}$	1+2+19=22	*	2.20e-03	1.70e-03	1.60e-03	8.28e-04	2.30e-01	3.64e-01	**0**	**0**
		μ	4.65e-02	3.96e-02	4.87e-02	4.40e-02	1.04e+00	9.24e-01	**4.85e-05**	0.0095
		σ	5.13e-02	3.81e-02	6.29e-02	5.74e-02	5.21e-01	3.93e-01	**1.85e-04**	0.0253
		ε	1.60e+03	1.62e+03	1.75e+03	1.69e+03	1.53e+03	1.52e+03	6.06e+02	6.46e+02
f_{Ackley}	1+7+12=20	*	6.97e+00	1.39e+01	7.96e+00	9.96e+00	1.00e+02	1.21e+02	0	**0**
		μ	3.15e+01	3.43e+01	3.51e+01	3.59e+01	1.32e+02	1.61e+02	**1.06e-06**	**8.31e-07**
		σ	1.12e+01	8.54e+00	8.98e+00	8.31e+00	1.54e+01	1.90e+01	**5.96e-07**	6.90e-07
		ε	1.66e+03	1.50e+03	1.45e+03	1.58e+03	1.54e+03	1.67e+03	1.41e+01	8.33e+00

'*', 'μ', 'ε', 'σ' indicates best fitness, mean fitness, number of epoch required to achieve '*' and standard deviation.
The best results among the algorithms are marked by bold.
'†' The global minima of f_{Levy13} function changes to **1.35e-31** from **0** with precision of π fixed to 4 digits.
'†' The global minima of f_{Ackley} function changes to **8.88e-16** from **0** with precision of π fixed to 4 digits.
'‡' The global minima of f_{Levy} function changes to **1.50e-32** from **0** with precision of π fixed to 4 digits

The four functions of Group IV are highly multimodal and the main challenge for the algorithms is to skip the local minima and converge into the global minima. The simulation results in **Table XIV(*d*)** clearly portray the extended exploration capacity of TR operator than that of only mutation. Among GAs with global operators, TRGA and KGA with TR operator gave better performance than those of without TR, SGA and FNGA. Further, the performance of LTRGA was comparatively better than that of LSGA. However, the disruption effect of d-point local crossover and mutation was also explicit as LSGA and LTRGA gave poor performance than the global counterparts (SGA, FNGA, TRGA and KGA). The improved exploitation and exploration capacity by LAmC, hGR and TR operators are justified by the superior performances of hGRGA and bAMLGA for all the four functions. Therefore, it is empirically justified that the inclusion of LAmC, hGR and TR operators ensures the appropriate trade-off between intensification and diversification in bAMLGA.

Group V consists of five non-scalable functions with only two variables, yet have highly complex and multimodal landscape and inseparability property. The proposed algorithm bAMLGA could reach the global minima avoiding many local minima and maxima around it for all the functions in each cycle except Levy13 function. For $f_{ScafferF2}$, f_{Bird} and $f_{CarromTable}$, hGRGA provided a similar performance. Considering the best performance (*) only, we can again conclude that when the number of decision variables involved is low, the performances of global operator based GAs get relatively closer to newly introduced algorithms (**Table XIV(*e*)**). Here, we would like to highlight the possible numerical error due to the allowable precision of the floating point value of π for Levy13 function. We noticed that with the precision of π extended to 10, 15 and 25, the global minima changes to 4.49e-34, 4.02e-45 and 8.07e-65 respectively which are not exactly equal to 0. Here, we reported the result for precision of 4 digits.

Group VI contains the most challenging functions that are scalable, multimodal and have inseparable variables. For all of the eight functions, the bAMLGA with LAmC and hGR operators performed highly superior (**Table XIV(*f*)**) than that of SGA, FNGA, TRGA, KGA, LSGA and LTRGA. For 5 out of 8 functions, hGRGA performed comparably with bAMLGA. However, bAMLGA outperformed hGRGA for the rest of the functions. One more time, the high disruption effect of local d-point crossover and mutation (*Algorithm*: LSGA, LTRGA) was evident for all of these functions.

D. Comparison of AMLGA with Other Evolutionary Algorithms

Here we compare the performance of the cost-improved iAMLGA with its basic version, bAMLGA and, with two state-of-the-art evolutionary computing algorithms, Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) [8, 92, 93] and Social Spider Algorithm (SSA) [18]. The CMA-ES [8, 92, 93] is a population-based evolutionary algorithm (EA) for non-linear and non-convex problems in continuous domain that estimates a covariance matrix which is closely related to the inverse Hessian. It is a second order approach that approximates this positive definite matrix by an iterative manner. The source code of CMA-ES is collected from a publicly available link[2]. The default strategy parameters including the population size were kept as mentioned in (6)–(8) of [8] and the initial standard deviation or step size was set to half of the interval ($[lb, ub]$). As for many of the test functions used in this study CMA-ES failed for ill-conditioned covariance matrix, we adopted a restart strategy if the condition number of the covariance matrix exceeds 10^{14} with no additional changes in the initial setup. Unlike CMA-ES, the SSA is a swarm intelligence based EA that models the information sharing behavior of social spider into the formulation of a search algorithm [18]. It determines the potential direction towards food source locations (solutions) by analyzing the vibrations on the web, received due to the spider's cooperative movements. The control parameters for SSA such as population size, rate of vibration attenuation, parameters related to the dimension mask were fixed according to the recommendations made in [18]. The SSA is also obtained from online[3].

In our complete test suite with 40 functions, 32 are scalable to higher dimension and the rest of 8 functions are non-scalable with two variables. We tested SSA, CMA-ES, bAMLGA and iAMLGA on 31 scalable functions for two different dimension (d) values, 30 and 50. However, Michalewicz function was optimized with 10 variables only. Therefore, we conducted 71 simulations ($2 \times 31 + 1 + 8$) for each of the 4 algorithms, and each of these simulations were repeated for 30 runs to collect statistics like median, mean and standard deviations of the function values. To compare the cost of the algorithms consistently, we allowed a limited number of function evaluations, $NFE_{max} = d \times 1e^4$. Therefore, a run terminates if the solution is no worse than the global optima plus a tolerance ($f(\boldsymbol{X}^*) + 1e^{-10}$) or the number of function evaluations (NFE) exceed the NFE_{max}. A simulation is called as successful if the best found function values is less than or equal to the pre-specified optima ($f(\boldsymbol{X}^*) + 1e^{-10}$) using no greater than NFE_{max} function evaluations. In the following we compare the solution quality, convergence speed, reliability and overall success performance [94] of the four algorithms.

1) Assessment of Solution Quality

We assess the quality of the function values achieved using three different statistics, median (~), mean (μ) and standard deviation or stdev (σ) of the function values obtained in 30 independent runs of each simulation. It is useful to focus on the median in addition to the mean function value as median is invariant of outliers [95] whereas mean suffers from the limitations of emphasizing large values. The results on 30-dimensional and 50-dimensional scalable functions are reported in **Table XV** and **XVI** respectively. **Table XVII** separately shows the performance of the algorithms on non-scalable functions. The best values are bold faced and a value less than 1e-10 is reported as 0.

Table XV and **XVI** shows that the four unimodal and separable functions of Group I ($f_{Sphere}, f_{Cigar}, f_{Discus}, f_{RHE}$) were successfully optimized by all four algorithms for both 30

[2] https://www.lri.fr/~hansen/cmaes_inmatlab.html#matlab

[3] https://github.com/James-Yu/SocialSpiderAlgorithm

and 50 dimension values. The functions of Group III are relatively complex as their variables are interrelated and affect each other. For all three functions of this group, SSA failed to converge into the global optima for both dimension values. In case of $f_{Zakharov}$, the median solution quality of CMA-ES, bAMLGA and iAMLGA were similar for d = 30 and 50, however the mean function value achieved by iAMLGA were comparatively worse than CMA-ES and bAMLGA. Note that, we reduced the amount of exploitation in iAMLGA to make it cost effective than bAMLGA. Thus, $f_{Zakharov}$ is a benchmark for which higher exploitation is more rewarding. The CMA-ES, bAMLGA and iAMLGA consistently found the global minima of $f_{Schwefel1.2}$ in every run. For $f_{Schwefel2.2}$, both bAMLGA and iAMLGA, proposed in this work, outperformed SSA and CMAES for both dimension values.

Group IV consists of 4 multimodal functions with separable variables. These functions test how well an algorithm can forgo the local minima and continue looking for the global one. The SSA could optimize the 30-dimensional $f_{Rastrigin}$ (**Table XV**), however failed in case of 50 dimension (**Table XVI**). CMA-ES failed for both 30 and 50-dimensional $f_{Rastrigin}$, whereas iAMLGA successfully converged. Both of the SSA and CMA-ES were trapped into local minima for $f_{Schwefel2.26}$ which is a deceptive function as the global minimum lies at the edge of the search space and is geometrically distant from the next best local minima. Therefore, the search algorithms are potentially prone to convergence in the wrong direction. The bAMLGA and iAMLGA could reach very close to the global minima for both of the dimensions (**Table XV** and **XVI**), among which iAMLGA was better. For 10-dimensional $f_{Michelawicz}$ with highly complex search space containing $d!$ number of local optima, SSA outperformed CMA-ES. However, both of the bAMLGA and iAMLGA further outperformed SSA (**Table XV**). In case of $f_{Styblinski-Tang(ST)}$, CMA-ES performed better than SSA, and subsequently the both versions of AMLGA resulted better solutions than SSA and CMA-ES.

TABLE XV
SOLUTION QUALITY ASSESSMENT ON SCALABLE TEST FUNCTIONS WITH DIMENSION 30

Test functions		SSA [18]		CMA-ES [8]		bAMLGA [This work]		iAMLGA [This work]	
		Median (∼)	*Mean* (μ) ± *stdev*(σ)	*Median* (∼)	*Mean* (μ) ± *stdev*(σ)	*Median* (∼)	*Mean* (μ) ± *stdev*(σ)	*Median* (∼)	*Mean* (μ) ± *stdev*(σ)
Group I	f_{Sphere}	**0**	**0 ± 0**	**0**	**0 ± 0**	**0**	**0 ± 0**	**0**	**0 ± 0**
	f_{Cigar}	**0**	**0 ± 0**	**0**	**0 ± 0**	**0**	**0 ± 0**	**0**	**0 ± 0**
	f_{Discus}	**0**	**0 ± 0**	**0**	**0 ± 0**	**0**	**0 ± 0**	**0**	**0 ± 0**
	f_{RHE}	**0**	**0 ± 0**	**0**	**0 ± 0**	**0**	**0 ± 0**	**0**	**0 ± 0**
Group III	$f_{Zakharov}$	7.70E-03	8.47E-03 ± 4.46E-03	**0**	**0 ± 0**	**0**	**0 ± 0**	**0**	0.987 ± 5.405
	$f_{Schewfel1.2}$	94.762	97.961 ± 25.316	**0**	**0 ± 0**	**0**	**0 ± 0**	**0**	**0 ± 0**
	$f_{Schwefel2.2}$	5.78E+35	1.42E+36 ± 2.85E+36	**0**	20.973 ± 114.876	**0**	**0 ± 0**	**0**	**0 ± 0**
Group IV	$f_{Rastrigin}$	**0**	**0 ± 0**	48.753	49.449 ± 13.587	**0**	**0 ± 0**	**0**	**0 ± 0**
	$f_{Schwefel2.26}$	6.87E+03	6.75E+03 ± 8.29E+02	1.25E+04	1.25E+04 ± 0.00E+00	4.10E-03	2.48E-03 ± 1.86E-03	**3.82E-04**	**8.75E-04 ± 1.28E-03**
	$f_{Michalewicz}$⊙	-3.770	-3.949 ± 0.605	4.18E+03	4.18E+03 ± 1.618	**-11.021**	**-11.477 ± 1.284**	-10.941	-11.022 ± 0.938
	f_{ST}	-738.267	-738.490 ± 61.184	-977.071	-983.668 ± 4.85E+01	**-1174.98**	**-1174.983** ± 1.95E-03	**-1174.98**	**-1174.983 ± 1.84E-03**
Group VI	f_{Ackley}	9.82E-11	9.82E-11 ± 0	19.3787	12.969 ± 9.329	**8.88E-16**	**8.88E-16 ± 0**	**8.88E-16**	**8.88E-16 ± 0**
	$f_{Rosenbrock}$	0.478	0.807 ± 1.0304	**0**	**0.531 ± 1.378**	0.0562	3.625 ± 8.262	**0**	1.085 ± 5.265
	$f_{Griewank}$	**0**	**0 ± 0**	**0**	2.47E-04 ± 1.35E-03	**0**	4.21E-02 ± 1.81E-01	**0**	**0 ± 0**
	f_{SESW}	**0.0782**	**0.072727 ± 1.42E-02**	0.5	0.499946 ± 2.50E-05	0.2574	0.252398 ± 1.62E-01	0.2277	0.229392 ± 1.52E-01
	$f_{Trigonometric}$	5.19E+02	5.37E+02 ± 1.37E+02	1.10E-06	1.00E-05 ± 2.70E-05	8.56E-07	1.07E-04 ± 2.36E-04	**0**	**2.60E-05 ± 9.80E-05**
	f_{Levy}	**9.42E-11**	**9.56E-11 ± 0**	72.225	68.216 ± 47.375	5.28E-04	1.16E-02 ± 2.67E-02	1.00E-06	2.15E-03 ± 8.11E-03
	$f_{SchafferF7}$	0.0148	0.014632 ± 0.006392	0.1104	0.21457 ± 0.254529	0.0102	0.088966 ± 0.240811	**0**	**9.44E-04 ± 3.14E-03**
	$f_{Lunacek}$	41.601	41.956 ± 5.770	73.289	70.704 ± 19.125	**0**	**1.00E-06 ± 1.00E-06**	**0**	**1.00E-06 ± 1.00E-06**
Group VII	$f_{R-Cigar}$	40.839	44.202 ± 42.475	**0**	**0 ± 0**	**0**	**0 ± 0**	**0**	**0 ± 0**
	$f_{R-Discus}$	185.524	619.473 ± 813.876	**0**	**0 ± 0**	**0**	**0 ± 0**	**0**	**0 ± 0**
	$f_{R-Rastrigin}$	6.86E-08	0 ± 1.00E-06	0.4975	0.762802 ± 0.893109	**0**	**0 ± 0**	**0**	**0 ± 0**
	$f_{R-Ackley}$	3.50E-03	3.87E-03 ± 3.01E-03	15.414	15.381 ± 0.802	**0**	**0 ± 0**	**0**	**0 ± 0**
	$f_{R-Griewank}$	3.27E-05	3.60E-05 ± 3.00E-05	9.49E-11	1.40E-03 ± 3.22E-03	0	2.18E-02 ± 1.19E-01	**0**	**0 ± 0**
	$f_{R-SchafferF7}$	0.186	0.2134 ± **0.092**	**0.108**	**0.182** ± 0.248	0.169	0.496 ± 0.804	0.112	0.189 ± 0.239
	$f_{R-Lunacek}$	**128.046**	**127.735 ± 10.719**	189.845	187.934 ± 11.771	268.523	264.414 ± 17.651	178.613	180.295 ± 18.741
Group VIII	$f_{RS-Rastrigin}$	**175.182**	**174.194 ± 9.446**	205.458	205.458 ± 11.050	279.279	282.460 ± 16.372	247.0157	250.291 ± 15.042
	$f_{RS-Ackley}$	**1.7053**	**3.133 ± 4.359**	19.664	19.637 ± 0.101	18.714	18.805 ± 0.439	19.065	19.108 ± 0.371
	$f_{RS-Lunacek}$	**212.101**	**220.930 ± 35.227**	1172.5	1159.590 ± 220.689	1071.1	1072.507 ± 22.019	1022.509	1024.609 ± 34.624
Group IX	$f_{Hybrid-1}$	33.3546	42.273 ± 34.729	708.455	711.342 ± 312.738	**0**	**0 ± 0**	**0**	**0 ± 0**
	$f_{Hybrid-2}$	0.067	0.188 ± 0.366	20	19.887 ± 0.403	**8.88E-16**	0.111 ± 0.235	**8.88E-16**	**8.88E-16 ± 0**
	$f_{Hybrid-3}$	100.141	131.914 ± 124.918	**1.214**	**2.440 ± 2.345**	8.971	9.313 ± 1.695	8.994	16.409 ± 26.830

⊙ $f_{Michelawicz}$ is tested for 10 dimensions only.
Best values are bold faced.

TABLE XVI
SOLUTION QUALITY ASSESSMENT ON 32 SCALABLE TEST FUNCTIONS WITH DIMENSION 50

Test functions		SSA [18]		CMA-ES [8]		bAMLGA [This work]		iAMLGA [This work]	
		Median $(\sim)$	*Mean* $(\mu) \pm stdev(\sigma)$	*Median* $(\sim)$	*Mean* $(\mu) \pm stdev(\sigma)$	*Median* $(\sim)$	*Mean* $(\mu) \pm stdev(\sigma)$	*Median* $(\sim)$	*Mean* $(\mu) \pm stdev(\sigma)$
Group I	f_{Sphere}	**0**	**0 ± 0**	**0**	**0 ± 0**	**0**	**0 ± 0**	**0**	**0 ± 0**
	f_{Cigar}	**0**	**0 ± 0**	**0**	**0 ± 0**	**0**	**0 ± 0**	**0**	**0 ± 0**
	f_{Discus}	**0**	**0 ± 0**	**0**	**0 ± 0**	**0**	**0 ± 0**	**0**	**0 ± 0**
	f_{RHE}	**0**	**0 ± 0**	**0**	**0 ± 0**	**0**	**0 ± 0**	**0**	**0 ± 0**
Group III	$f_{Zakharov}$	9.925	9.904 ± 1.904	**0**	**0 ± 0**	**0**	**0 ± 0**	**0**	3.972 ± 20.548
	$f_{Schewfel1.2}$	5.92E+03	592E+03 ± 9.93E+02	**0**	**0 ± 0**	**0**	**0 ± 0**	**0**	**0 ± 0**
	$f_{Schwefel2.2}$	6.97E+62	8.12E+65 ± 4.29E+66	0	40.557 ± 188.115	**0**	**0 ± 0**	**0**	**0 ± 0**
Group IV	$f_{Rastrigin}$	16.602	16.079 ± 2.777	94.521	97.804 ± 19.932	**0**	4.40E-05 ± 2.43E-04	**0**	**0 ± 0**
	$f_{Schwefel2.26}$	1.35E+04	1.35E+04 ± 1.18E+03	2.08E+04	2.08E+04 ± 0	6.36E-04	2.90E-03 ± 3.02E-03	**6.36E-04**	**2.07E-03 ± 2.65E-03**
	f_{ST}	-1085.5	-1107.715 ± 92.863	-1604.9	-1607.717 ± 55.115	**-1958.3**	**-1958.3** ± 2.85E-03	**-1958.3**	**1958.3 ± 2.80E-03**
Group VI	f_{Ackley}	9.87E-11	9.87E-11 ± 0	19.384	12.956 ± 9.319	**8.88E-16**	**8.88E-16 ± 0**	**8.88E-16**	**8.88E-16 ± 0**
	$f_{Rosenbrock}$	11.290	29.132 ± 34.843	**9.40E-11**	0.398 ± 1.216	0.0405	1.149 ± 1.996	1.00E-06	**0.095 ± 0.423**
	$f_{Griewank}$	**0**	**0 ± 0**	**0**	6.57E-04 ± 2.50E-03	**0**	5.00E-06 ± 2.50E-05	**0**	**0 ± 0**
	f_{SESW}	**0.127**	**0.120484 ± 1.69E-02**	0.5	0.499981 ± 6.00E-06	0.353	0.326142 ± 1.58E-01	0.3288	0.287784 ± 1.91E-01
	$f_{Trigonometric}$	1.43E+04	1.40E+04 ± 1.90E+03	**5.57E-07**	**2.00E-06 ± 3.00E-06**	1.01E-06	7.40E-05 ± 2.13E-04	1.00E-06	4.00E-06 ± 7.00E-06
	f_{Levy}	**9.56E-11**	**9.56E-11 ± 0**	43.2246	58.584 ± 55.201	5.30E-03	3.02E-02 ± 5.22E-02	2.00E-06	1.78E-03 ± 6.65E-03
	$f_{SchafferF7}$	0.288	0.295 ± 0.073	0.118	0.156 ± 0.207	9.35E-04	0.0588 ± 0.128	**1.00E-06**	**2.11E-03 ± 1.14E-02**
	$f_{Lunacek}$	118.184	119.424 ± 15.752	141.4451	129.488 ± 36.498	**0**	**1.00E-06** ± **1.00E-06**	**0**	**1.00E-06 ± 1.00E-06**
Group VII	$f_{R-Cigar}$	13.776	26.542 ± 45.989	**0**	**0** ± **0**	**0**	**0** ± **0**	**0**	**0** ± **0**
	$f_{R-Discus}$	3.16E+03	3.29E+03 ± 1.27E+03	**0**	**0** ± **0**	**0**	**0** ± **0**	**0**	**0** ± **0**
	$f_{R-Rastrigin}$	3.04E-06	1.00E-05 ± 1.80E-05	0.995	0.895 ± 1.023	**0**	**0** ± **0**	**0**	**0** ± **0**
	$f_{R-Ackley}$	9.92E-11	2.00E-06 ± 5.00E-06	15.661	15.620 ± 0.635	**0**	**0** ± **0**	**0**	**0** ± **0**
	$f_{R-Griewank}$	5.35E-05	8.80E-05 ± 1.39E-04	**0**	7.39E-04 ± 2.97E-03	**0**	2.00E-06 ± 1.30E-05	**0**	**0** ± **0**
	$f_{R-SchafferF7}$	2.655	2.693 ± 0.647	0.168	0.439 ± 0.559	**0.071**	0.347 ± 0.681	0.158	**0.299** ± **0.406**
	$f_{R-Lunacek}$	**271.637**	**269.957 ± 14.514**	325.807	323.125 ± 14.610	518.077	523.519 ± 30.003	322.736	322.844 ± 27.761
Group VIII	$f_{RS-Rastrigin}$	**366.877**	**365.081 ± 24.838**	487.025	488.551 ± 17.840	622.462	619.645 ± 17.556	544.731	545.886 ± 16.183
	$f_{RS-Ackley}$	**19.785**	**18.561 ± 2.565**	19.7576	19.752 ± 0.065	20.1957	20.131 ± 0.272	19.670	19.677 ± 0.126
	$f_{RS-Lunacek}$	**366.028**	**365.735 ± 38.321**	1808.2	1809.710 ± 93.508	1708.9	1710.998 ± 34.582	1544.998	1543.264 ± 42.158
Group IX	$f_{Hybrid-1}$	51.572	52.774 ± 15.030	1534.949	1552.549 ± 423.393	**0**	4.167 ± 22.826	**0**	**0 ± 0**
	$f_{Hybrid-2}$	3.078	3.220 ± 1.419	20.259	20.405 ± 0.849	**8.88E-16**	0.183 ± 0.796	1.80E-03	**8.88E-16 ± 0.007154**
	$f_{Hybrid-3}$	556.828	703.164 ± 491.322	**0.115**	**1.343 ± 1.885**	8.959	29.105 ± 6.734	9.00	16.212 ± 23.827

Best values are bold faced.

TABLE XVII
SOLUTION QUALITY ASSESSMENT ON 8 NON-SCALABLE TEST FUNCTIONS WITH DIMENSION 2

Test functions		SSA [18]		CMA-ES [8]		bAMLGA [This work]		iAMLGA [This work]	
		Median $(\sim)$	*Mean* $(\mu) \pm stdev(\sigma)$	*Median* $(\sim)$	*Mean* $(\mu) \pm stdev(\sigma)$	*Median* $(\sim)$	*Mean* $(\mu) \pm stdev(\sigma)$	*Median* $(\sim)$	*Mean* $(\mu) \pm stdev(\sigma)$
Group II	f_{Zettle}	0.0711	40.457 ± 204.976	**-0.0037**	**-0.0037 ± 0**	**-0.0037**	**-0.0037** ± 3.50E-05	**-0.0037**	**-0.0037 ± 0**
	f_{Leon}	0.499	1.671 ± 4.214	**0**	**0 ± 0**	5.69E-05	9.48E-04 ± 2.41E-03	3.66E-07	3.65E-04 ± 7.37E-04
	f_{Easom}	0	-0.0667 ± 0.208	0	3.33E+09 ± 4.79E+09	-0.162	-0.369 ± 0.394	**-0.969**	**-0.773 ± 0.325**
Group V	$f_{SchafferF2}$	0.4369	0.305 ± 0.209	0.403	0.273 ± 0.218	**0**	1.55E-04 ± 5.17E-04	**0**	**6.60E-05 ± 3.64E-04**
	$f_{SchafferF6}$	0.312	0.266 ± 0.168	0.178	0.232 ± 0.197	9.70E-03	9.09E-03 ± 1.24E-02	**9.70E-03**	**5.51E-03 ± 4.90E-03**
	f_{Bird}	-35.751	-47.972 ± 42.112	n/a	n/a	-106.576	-106.110 ± 0.997	**-106.760**	**-106.72 ± 0.08593**
	f_{Levy13}	3.6672	14.185 ± 19.562	**0**	1.69E-02 ± 5.84E-02	**0**	1.11E-02 ± 4.22E-02	**0**	**0 ± 0**
	$f_{CarromTable}$	-1.116	-6.622+02 ± 9.233	-5.74E+03	-4.12E+07 ± 1.85E+08	**-24.156**	-24.144 ± 5.08E-02	**-24.156**	**-24.152 ± 7.58E-03**

n/a CMA-ES failed for f_{Bird}, thus the results are not available.
Best values are bold faced

Group VI consists of complicated multimodal functions with inseparable variables. Here, we observed superior performance of iAMLGA than that of bAMLGA which validates that not exhaustive, yet a balanced amount of exploitation with proper exploration is effective in solving multimodal functions. The CMA-ES prematurely converged into one of the many local minima of f_{Ackley} (**Table XV** and **XVI**), whereas CMA-ES, bAMLGA and iAMLGA showed better and comparable performances. For $f_{Rosenbrock}$, CMA-ES and iAMLGA showed competitive median values for both dimensions. However, CMA-ES was better considering mean solution of 30-dimensional $f_{Rosenbrock}$ (**Table XV**) and iAMLGA was better for 50-dimensional $f_{Rosenbrock}$ (**Table XVI**). SSA and iAMLGA consistently converged into the global minima of $f_{Griewank}$ and provided better statistics than those of CMA-ES and bAMLGA for both of the dimension values. For f_{SESW}, none of the algorithms could achieve the global minima. In summary, SSA performed the best and CMA-ES gave the worst performance on this function, while iAMLGA was comparatively better than bAMLGA. For

$f_{Trigonometric}$, both versions of AMLGA and CMA-ES gave consistent outputs whereas the SSA was found stuck in the local minima. On the other hand, SSA provided the best result for f_{Levy}, whereas iAMLGA and bAMLGA were very close to the global minima. CMA-ES showed comparatively lower performance for this function. The iAMLGA provided superior quality solution for both 30 and 50-dimensional $f_{SchafferF7}$ than the other three algorithms under consideration (**Table XV** and **XVI**). For $f_{Lunacek}$ with 70% of the landscape covered by two funnels with local minima [96], both iAMLGA and bAMLGA provided far better quality solutions (**Table XV** and **XVI**) than others.

Group VII contains 7 rotated functions. For the two rotated unimodal functions ($f_{R-Cigar}$ and $f_{R-Discus}$), all the algorithms except SSA could achieve the global optima (**Table XV** and **XVI**). The iAMLGA and bAMLGA outperformed the other two algorithms in two complex rotated functions, $f_{R-Rastrigin}$ and $f_{R-Ackley}$ and consistently found the global optima for both dimension values 30 (**Table XV**) and 50 (**Table XVI**). For both of these functions, SSA provided the second best result and the performance of CMA-ES was comparatively lower than those of the others. For $f_{R-Griewank}$, iAMLGA found the global minima at every run which was the best performance. The CMA-ES resulted the best mean and median function values for 30-dimensional $f_{R-SchafferF7}$ (**Table XV**). However, bAMLGA and iAMLGA gave the best median and mean values respectively for 50-dimensional $f_{R-SchafferF7}$ (**Table XVI**). All the four algorithms prematurely converged into local minima in every run of $f_{R-Lunacek}$ (**Table XV** and **XVI**), however SSA provided a better solution quality.

Group VIII consists of 3 modified functions with landscape rotated as well as the global optima shifted to a random position. The SSA consistently outperformed the other algorithms in all three rotated and shifted functions ($f_{RS-Rastrigin}$, $f_{RS-Ackley}$ and $f_{RS-Lunacek}$) for both 30 and 50 variables (**Table XV** and **XVI**). However, iAMLGA provided better quality solutions for the hybrid functions for which random subsets of the variables are passed to different constituent rotated functions. For $f_{Hybrid-1}$ and $f_{Hybrid-2}$, the proposed iAMLGA resulted far better mean and median function values than those provided by the other algorithms. However, CMA-ES performed the best in optimizing $f_{Hybrid-3}$. The bAMLGA and iAMLGA were comparable and second best for this function, while SSA showed relatively poor performance.

Table XVII shows the simulation output on 8 non-scalable functions among which 3 are unimodal (Group II) and 5 are multimodal (Group V). Both CMA-ES and iAMLGA could successfully optimize f_{Zettl} ($f_{Zettl}(\boldsymbol{X}^*) = -0.0037$). The bAMLGA performed very closely with slightly higher dispersion, however SSA failed to converge. CMA-ES achieved the best performance for f_{Leon} and both of the AMLGAs were very close. Only iAMLGA could reach very close to the global optima of $f_{Easom}(\boldsymbol{X}^*) = -1$ that has a huge less informative flat area in the landscape. Both CAM-ES and SSA failed to converge for f_{Easom}. For all 5 multimodal non-scalable functions of Group V, iAMLGA outperformed SSA, CMA-ES and bAMLGA (**Table XVII**). The bAMLGA was close to iAMLGA for $f_{SchafferF2}$ and $f_{SchafferF6}$, whereas the SSA and CMA-ES prematurely converged into local minima. In case of $f_{Bird}(\boldsymbol{X}^*) = -106.76$, CMA-ES failed due to ill-conditioned covariance matrix. Note that, we adopted restart strategy to overcome this problem and the restart mechanism worked for several other functions including f_{Easom}, $f_{Schawefel2.2}$, $f_{SchafferF2}$, $f_{SchafferF6}$, f_{Levy13} and $f_{CarromTable}$. Notwithstanding, the mechanism did not work for f_{Bird}. The median solution quality given by CMA-ES and bAMLGA for f_{Levy13} were competitive to that of iAMLGA, however the quality of mean results was slightly lower. For $f_{CarromTable.2}$, AMLGA versions were comparable while SSA failed to reach the global minima. On the other hand, CMA-ES resulted highly fluctuating function values that are even lower than the known global minima with a very high standard deviation, thus we considered this result unreliable.

Overall, iAMLGA produced either better or similar median function values for 50 out of 71 simulations as reported in **Table XV**, **XVI** and **XVII**. To compare, a similar count for bAMLGA, CMA-ES and SSA are 43, 28 and 23, respectively.

2) Assessment of Reliability

Precise documentation of the performance of a stochastic process is critical [97] as the output varies in different independent run. Thus, the researchers run a stochastic search based EA on a test function for multiple times and analyze the performance statistically. We carried out 30 independent runs of each simulation and measured the reliability of the algorithms by computing the success rate or the percentage of successful runs (%SR) out of the total runs executed on a function. Thus, the success rate is calculated as the number of successful runs (S_{runs}) divided by the total number of runs (T_{runs}). The result is reported in **Table XVIII** (30-dimensional scalable functions), **XIX** (50-dimensional scalable functions) and **XX** (2-dimensional non-scalable functions).

All the four algorithms were found consistently reliable with 100% success rate for the 4 unimodal and separable functions of Group I with both dimension values 30 (**Table XVIII**) and 50 (**Table XIX**). SSA was stuck at local optima of the 3 functions of Group III (unimodal and inseparable), therefore resulted 0 success rate for all of them. For $f_{Zakharov}$, bAMLGA and CMA-ES were more reliable than iAMLGA for both dimension values. Note that, $f_{Zakharov}$ is an instance for which the solution quality of bAMLGA was also better than iAMLGA as bAMLGA allows higher exploitation. The reliability of CMA-ES and AMLGAs were similarly promising for 30 and 50-dimensional $f_{Schwefel1.2}$. On the contrary, AMLGA versions resulted higher success rate than CAM-ES for $f_{Schwefel2.2}$.

For Group IV functions, CMA-ES was not successful at any of the run (**Table XVIII** and **XIX**). On the other hand, iAMLGA was successful in optimizing $f_{Rastrigin}$, $f_{Michalewicz}$ and $f_{Styblinski-Tang(ST)}$ for all the runs regardless of problem size. SSA showed 100% reliability for $f_{Rastrigin}$ when the number of dimension was 30, whereas the performance degraded to 0% when the dimension value was increased to

50. Moreover, SSA was not reliable for any other functions of this group (**Table XVIII** and **XIX**). None of the algorithms were able to reach the pre-defined global minima of $f_{Schwefel2.26}$, however iAMLGA attained better quality solution (**Table XV** and **XVI**). Overall, AMLGA showed promising reliability compared to the other algorithms in avoiding the local minima for these highly multimodal functions.

For the functions of Group VI, SSA and CMA-ES showed comparable and overall less reliability than those of AMLGA versions. The f_{Ackley} was optimized by all the algorithms with 100% success except CMA-ES that obtained only 33.33% success (**Table XVIII** and **XIX**). On the contrary, CMA-ES achieved the best success rate of 86.67% and 90% for 30 and 50-dimansional $f_{Rosenbrock}$ respectively. Both of the bAMLGA and iAMLGA accomplished similar performances for f_{SESW} and outperformed the other competitors (**Table XVIII** and **XIX**). For $f_{Trigonometric}$, none of the algorithms could result reasonable reliability. SSA and iAMLGA were respectively the best and the second best reliable for f_{Levy}. The cost optimized iAMLGA resulted the highest reliability for $f_{Griewank}$, $f_{SchafferF7}$ and $f_{Lunacek}$ with 100%, 33.33% and 63.33% success rate respectively for d = 30 (**Table XVIII**), and 100%, 20% and 40% respectively for d = 50 (**Table XIX**). These promising performances of AMLGA on multimodal functions with inseparable variables underlines the algorithm's robust ability to avoid regions around local optima and reach the global optima.

TABLE XVIII
RELIABILITY, COST AND SUCCESS PERFORMANCE ASSESSMENT ON SCALABLE TEST FUNCTIONS WITH DIMENSION 30

		SSA [18]			CMA-ES [8]			bAMLGA [This work]			iAMLGA [This work]		
		%SR	$\overline{NFE}$	$\widehat{SP}$	%SR	$\overline{NFE}$	$\widehat{SP}$	%SR	$\overline{NFE}$	$\widehat{SP}$	%SR	$\overline{NFE}$	$\widehat{SP}$
Group I	f_{Sphere}	**100**	9.80E+04	15.489	**100**	**6.33E+03**	**1**	**100**	5.67E+04	5.375	**100**	2.31E+04	3.644
	f_{Cigar}	**100**	1.31E+05	8.987	**100**	**1.46E+04**	**1**	**100**	5.61E+04	3.438	**100**	2.21E+04	1.514
	f_{Discus}	**100**	9.96E+04	4.956	**100**	3.21E+04	1.597	**100**	5.46E+04	2.717	**100**	**2.01E+04**	**1**
	f_{RHE}	**100**	1.05E+05	12.375	**100**	**8.47E+03**	**1**	**100**	5.38E+04	5.639	**100**	2.28E+04	2.685
Group III	$f_{Zakharov}$	0	∅	∅	**100**	**1.67E+04**	**1**	**100**	7.56E+04	4.525	96.67	1.42E+05	8.795
	$f_{Schewfel1.2}$	0	∅	∅	**100**	**1.54E+04**	**1**	**100**	8.04E+04	5.232	**100**	4.22E+04	2.747
	$f_{Schwefel2.2}$	0	∅	∅	93.33	5.07E+04	2.616	**100**	8.04E+04	3.874	**100**	**2.07E+04**	**1**
Group IV	$f_{Rastrigin}$	**100**	2.25E+05	3.076	0	∅	∅	**100**	8.75E+04	1.195	**100**	**7.32E+04**	**1**
	$f_{Schwefel2.26}$	0	∅	∅	0	∅	∅	0	∅	∅	0	∅	∅
	$f_{Michalewicz}$⊙	0	∅	∅	0	∅	∅	**100**	1.62E+04	2.219	**100**	**7.31E+03**	**1**
	f_{ST}	0	∅	∅	0	∅	∅	**100**	6.07E+04	3.434	**100**	**1.77E+04**	**1**
Group VI	f_{Ackley}	**100**	1.43E+05	3.224	33.33	1.49E+04	1.009	**100**	7.60E+04	1.717	**100**	**4.43E+04**	**1**
	$f_{Rosenbrock}$	0	∅	∅	**86.67**	**4.91E+04**	**1**	43.33	1.44E+05	5.869	56.67	1.05E+05	3.264
	$f_{Griewank}$	90	2.27E+05	30.517	96.67	**7.99E+03**	**1**	80	1.07E+05	16.176	**100**	5.54E+04	6.702
	f_{SESW}	0	∅	∅	0	∅	∅	**6.67**	2.22E+05	1.586	3.33	**4.66E+04**	**1**
	$f_{Trigonometric}$	0	∅	∅	0	∅	∅	**3.33**	**5.16E+04**	**1**	0	∅	∅
	f_{Levy}	**100**	**1.18E+05**	**1**	0	∅	∅	3.33	2.95E+05	74.713	36.67	6.46E+04	1.489
	$f_{SchafferF7}$	0	∅	∅	0	∅	∅	0	∅	∅	**33.33**	**9.15E+04**	**1**
	$f_{Lunacek}$	0	∅	∅	0	∅	∅	56.67	9.50E+04	2.760	**63.33**	**3.85E+04**	**1**
Group VII	$f_{R-Cigar}$	0	∅	∅	**100**	**1.57E+04**	**1**	**100**	5.85E+04	3.721	**100**	2.28E+04	1.451
	$f_{R-Discus}$	0	∅	∅	**100**	**3.41E+04**	**1**	**100**	6.31E+04	1.849	**100**	5.10E+04	1.493
	$f_{R-Rastrigin}$	13.33	2.24E+05	152.134	50	**5.51E+03**	**1**	**100**	5.89E+04	5.342	**100**	2.28E+04	2.069
	$f_{R-Ackley}$	0	∅	∅	0	∅	∅	100	6.08E+04	1.804	**100**	**3.37E+04**	**1**
	$f_{R-Griewank}$	0	∅	∅	83.33	**9.12E+03**	**1**	80	1.01E+05	11.577	**96.67**	9.55E+04	9.029
	$f_{R-SchafferF7}$	0	∅	∅	**10**	**2.32E+04**	**1**	0	∅	∅	**10**	2.04E+05	8.810
	$f_{R-Lunacek}$	0	∅	∅	0	∅	∅	0	∅	∅	0	∅	∅
Group VIII	$f_{RS-Rastrigin}$	0	∅	∅	0	∅	∅	0	∅	∅	0	∅	∅
	$f_{RS-Ackley}$	0	∅	∅	0	∅	∅	0	∅	∅	0	∅	∅
	$f_{RS-Lunacek}$	0	∅	∅	0	∅	∅	0	∅	∅	0	∅	∅
Group IX	$f_{Hybrid-1}$	0	∅	∅	0	∅	∅	**100**	**4.39E+04**	**1**	**100**	5.75E+04	1.311
	$f_{Hybrid-2}$	0	∅	∅	0	∅	∅	30	1.61E+05	3.717	**100**	**1.44E+05**	**1**
	$f_{Hybrid-3}$	0	∅	∅	0	∅	∅	0	∅	∅	0	∅	∅

⊙ $f_{Michelawicz}$ is tested for 10 dimensions only.
∅ Mean NFE and normalized success performance cannot be computed due to 0 success rate.
Best values are bold faced.

TABLE XIX
RELIABILITY, COST AND SUCCESS PERFORMANCE ASSESSMENT ON SCALABLE TEST FUNCTIONS WITH DIMENSION 50

		SSA [18]			CMA-ES [8]			bAMLGA [This work]			iAMLGA [This work]		
		%SR	$\overline{NFE}$	$\widehat{SP}$	%SR	$\overline{NFE}$	$\widehat{SP}$	%SR	$\overline{NFE}$	$\widehat{SP}$	%SR	$\overline{NFE}$	$\widehat{SP}$
Group I	f_{Sphere}	**100**	1.76E+05	18.400	**100**	**9.59E+03**	**1**	**100**	9.09E+04	9.475	**100**	3.33E+04	3.469
	f_{Cigar}	**100**	2.49E+05	10.953	**100**	**2.28E+04**	**1**	**100**	8.30E+04	3.206	**100**	2.83E+04	1.241
	f_{Discus}	**100**	1.78E+05	6.499	**100**	6.99E+04	2.548	**100**	8.29E+04	3.022	**100**	**2.74E+04**	**1**
	f_{RHE}	**100**	1.93E+05	12.682	**100**	**1.52E+04**	**1**	**100**	9.10E+04	5.326	**100**	2.98E+04	1.961
Group II	$f_{Zakharov}$	0	∅	∅	**100**	**4.07E+04**	**1**	**100**	1.14E+05	2.795	93.33	6.26E+04	1.647

	$f_{Schewfel1.2}$	0	∅	∅	**100**	**3.57E+04**	**1**	**100**	1.17E+05	3.286	**100**	6.13E+04	1.718
	$f_{Schwefel2.2}$	0	∅	∅	86.67	2.69E+05	10.800	**100**	8.22E+04	2.864	**100**	**2.87E+04**	**1**
Group IV	$f_{Rastrigin}$	0	∅	∅	0	∅	∅	**93.33**	1.25E+05	1.402	**100**	**9.56E+04**	**1**
	$f_{Schwefel2.26}$	0	∅	∅	0	∅	∅	0	∅	∅	0	∅	∅
	f_{ST}	0	∅	∅	0	∅	∅	**100**	4.30E+04	1.724	**100**	**2.49E+04**	**1**
Group VI	f_{Ackley}	**100**	2.96E+05	7.876	33.33	**2.01E+04**	1.600	**100**	1.05E+05	2.799	**100**	3.76E+04	**1**
	$f_{Rosenbrock}$	0	∅	∅	**90**	1.29E+05	**1**	36.67	1.94E+05	3.694	50	**1.19E+05**	1.668
	$f_{Griewank}$	93.33	3.78E+05	28.194	93.33	**1.34E+04**	**1**	96.67	1.77E+05	12.719	**100**	7.06E+04	4.914
	f_{SESW}	0	∅	∅	0	∅	∅	**10**	2.31E+05	1.827	**10**	**1.27E+05**	**1**
	$f_{Trigonometric}$	0	∅	∅	**6.67**	**1.15E+04**	**1**	0	∅	∅	0	∅	∅
	f_{Levy}	**100**	**2.31E+05**	**1**	6.67	1.69E+05	10.966	3.33	4.49E+05	58.395	30	1.19E+05	1.716
	$f_{SchafferF7}$	0	∅	∅	0	∅	∅	0	∅	∅	**20**	**1.18E+05**	**1**
	$f_{Lunacek}$	0	∅	∅	0	∅	∅	36.67	1.61E+05	3.661	**40**	**4.78E+04**	**1**
Group VII	$f_{R-Cigar}$	0	∅	∅	**100**	**2.51E+04**	**1**	**100**	8.60E+04	3.43	**100**	2.76E+04	1.100
	$f_{R-Discus}$	0	∅	∅	**100**	**6.90E+04**	**1**	**100**	1.01E+05	1.458	**100**	9.67E+04	1.401
	$f_{R-Rastrigin}$	0	∅	∅	46.67	**8.60E+03**	**1**	**100**	9.28E+04	5.035	**100**	3.26E+04	1.767
	$f_{R-Ackley}$	63.33	4.43E+05	18.754	0	∅	∅	**100**	9.90E+04	2.654	**100**	**3.73E+04**	**1**
	$f_{R-Griewank}$	0	∅	∅	93.33	**1.63E+04**	**1**	96.67	1.50E+05	8.889	**100**	1.64E+05	9.388
	$f_{R-SchafferF7}$	0	∅	∅	0	∅	∅	3.33	2.66E+05	6.903	**13.33**	**1.54E+05**	**1**
	$f_{R-Lunacek}$	0	∅	∅	0	∅	∅	0	∅	∅	0	∅	∅
Group VIII	$f_{RS-Rastrigin}$	0	∅	∅	0	∅	∅	0	∅	∅	0	∅	∅
	$f_{RS-Ackley}$	0	∅	∅	0	∅	∅	0	∅	∅	0	∅	∅
	$f_{RS-Lunacek}$	0	∅	∅	0	∅	∅	0	∅	∅	0	∅	∅
Group IX	$f_{Hybrid-1}$	0	∅	∅	0	∅	∅	96.67	7.16E+04	1	**100**	**7.14E+04**	**1**
	$f_{Hybrid-2}$	0	∅	∅	0	∅	∅	56.67	**2.13E+05**	1.456	**93.33**	2.41E+05	**1**
	$f_{Hybrid-3}$	0	∅	∅	0	∅	∅	0	∅	∅	0	∅	∅

∅ Mean NFE and normalized success performance cannot be computed due to 0% success rate.
Best values are bold faced.

TABLE XX
RELIABILITY, COST AND SUCCESS PERFORMANCE ASSESSMENT ON NON-SCALABLE TEST FUNCTIONS WITH DIMENSION 2

		SSA [18]			**CMA-ES [8]**			**bAMLGA [This work]**			**iAMLGA [This work]**		
		%SR	$\overline{NFE}$	$\widehat{SP}$	%SR	$\overline{NFE}$	$\widehat{SP}$	%SR	$\overline{NFE}$	$\widehat{SP}$	%SR	$\overline{NFE}$	$\widehat{SP}$
Group II	f_{Zettl}	0	∅	∅	**100**	**3.93E+02**	**1**	96.67	4.36E+03	11.481	**100**	3.56E+03	9.046
	f_{Leon}	0	∅	∅	**100**	**6.54E+02**	**1**	**26.67**	9.04E+03	69.117	30	5.70E+03	29.058
	f_{Easom}	0	∅	∅	**6.67**	**579**	**1**	0	∅	∅	0	∅	∅
Group V	$f_{SchafferF2}$	6.67	6.92E+03	13.823	0	∅	∅	90	1.05E+04	1.547	**96.67**	**7.26E+03**	**1**
	$f_{SchafferF6}$	3.33	2.68E+03	3.907	0	∅	∅	43.33	1.20E+04	1.343	**43.33**	**8.90E+03**	**1**
	f_{Bird}	0	∅	∅	n/a	n/a	n/a	0	∅	∅	0	∅	∅
	f_{Levy13}	16.67	1.25E+03	14.448	90	**4.66E+02**	**1**	80	1.09E+04	26.216	**100**	7.52E+03	14.504
	$f_{CarromTable}$	**16.67**	**1.63E+03**	**1**	0	∅	∅	3.33	1.80E+04	55.183	13.33	7.78E+03	5.979

∅ Mean NFE and normalized success performance cannot be computed due to 0% success rate.
n/a CMA-ES failed for f_{Bird}, thus the results are not available.
Best values are bold faced.

The functions of Group VII test the search potency of the algorithms on the rotated landscapes. The CMA-ES, bAMLGA and iAMLGA performed competitively with 100% success rate for $f_{R-Cigar}$ and $f_{R-Discus}$, while SSA failed to converge at every run (**Table XVIII** and **XIX**). Among the five multimodal rotated functions, SSA was partially successful for 30-dimensional $f_{R-Rastrigin}$ (**Table XVIII**) and 50-dimensional $f_{R-Ackley}$ (**Table XIX**) only and totally failed for the others. The CMA-ES was partially reliable for $f_{R-Rastrigin}$, $f_{R-Griwank}$ and, almost failed for $f_{R-Ackley}$, $f_{R-SchafferF7}$ and $f_{R-Lunacek}$. To compare, iAMLGA showed the most robust search outcome with almost 100% success for $f_{R-Rastrigin}$, $f_{R-Ackley}$ and $f_{R-Griwank}$ irrespective of the dimension value. For $f_{R-SchafferF7}$, the success rate of iAMLGA was less, however still comparable with those of others. The rotated and further shifted functions of Group VIII were so difficult that none of the algorithms could find the global optima within the NFE_{max}. In case of the hybrid functions, AMLGA versions clearly outperformed the other approaches (**Table XVIII** and **XIX**). While SSA and CMA-ES totally failed for all three hybrid functions, iAMLGA achieved close to 100% success rate for two of them. The bAMLGA was less reliable than iAMLGA, however better than SSA and CMA-ES. For these functions, the performance of bAMLGA was either similar or lower than iAMLGA.

For unimodal non-scalable functions of Group II (**Table XX**), CMA-ES was found the most reliable algorithm. In contrary, iAMLGA was found better for the multimodal non-scalable functions of Group V (**Table XX**). The only exception was $f_{CarromTable}$ for which SSA was competitive.

In summary, iAMLGA outperformed the others in reliability test for 46 of the total simulations (71), whereas bAMLGA, CMA-ES and SSA was competitive for 33, 23 and 14 cases respectively. At this point, we further aim at visualizing the reliability of the algorithms on all test functions. Thus, we plot the empirical cumulative distribution of best objective values achieved in all runs for 30, 50 and 2-dimanional functions in **Fig. 10**. A larger area under curve is preferable in these plots as it resembles the proportion of functions for which an algorithm was successful to find a

minima close to global one. Note that, the global optima for several functions are negative. To plot the distribution plots in log scale, we translated the negative values as $f(\boldsymbol{X}) - min$(function values given by all the algorithms on that particular function) – 1. The constant 1 at the end was used to avoid 0. From **Fig. 10(a)-(b)**, we can notice that iAMLGA is more reliable in finding very low objective value (10^{-10}) for 60% of the scalable optimization problems. To compare, bAMLGA, CMA-ES and SSA obtained same objective value for approximately 50%, 40% and 25% of the problems. For the 2-dimanional problems, curves for iAMLGA, bAMLGA and CAM-ES were approximately similar. However, SSA fell far behind. Overall, we perceive the good reliability of the proposed AMLGAs by these distribution curves.

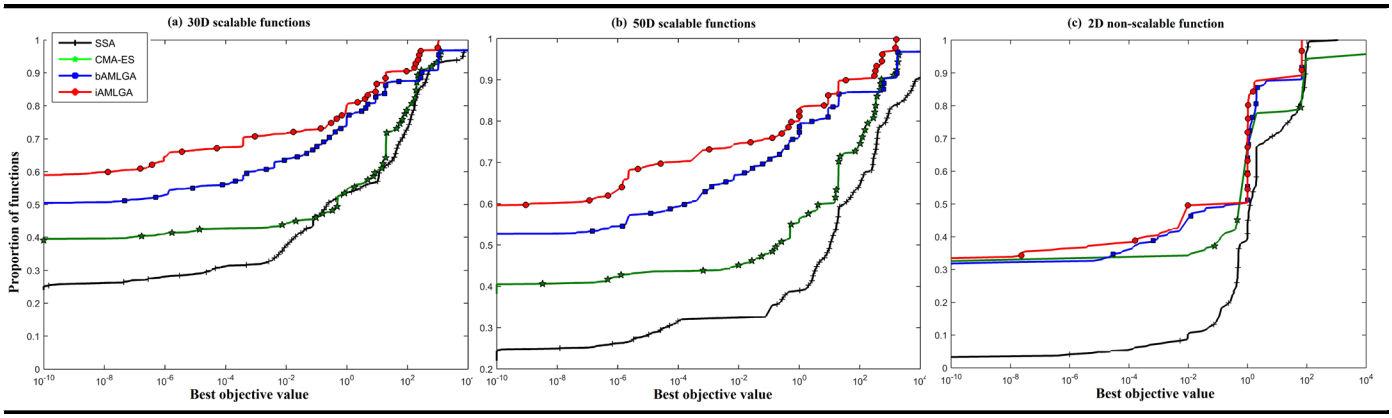

Fig. 10. Empirical cumulative distribution of best objective function values for (*a*) 30-dimanional, (*b*) 50-dimanional and (*c*) 2-dimensional test problems.

3) Assessment of Convergence Quality and Cost

Here, we analyze the convergence speed or cost of the algorithms in terms of the mean NFE ($\overline{NFE}$) required to achieve the pre-defined global minima in the successful runs out of the 30 runs. Thus $\overline{NFE}$ can be meaningfully computed when an algorithm has non-zero success rate in optimizing a function. We use $\overline{NFE} = \varnothing$ (a dummy value) in case of zero success rate. Note that, we used the number of epochs or iterations required to evaluate convergence quality while comparing the GAs. However, we adopt a different measure $\overline{NFE}$ here as it can consistently compare the convergence cost of completely different EAs, ignoring their internal cost. The result discussed in this section empirically justifies that iAMLGA gives overall less costly and robust performance compared to bAMLGA.

For the unimodal functions of Group I and III, CMA-ES showed the fastest convergence for 5 out of 7 functions while iAMLGA was the best for the other 2 functions, f_{Discus} and $f_{Schwefel2.2}$. A similar scenario was observed for both of dimension values (**Table XVIII** and **XIX**). The SSA is found to be the most costly algorithm even when it had 100% success rate.

CMA-ES failed to converge for all of the 4 multimodal functions of Group IV (**Table XVIII** and **XIX**), thus no $\overline{NFE}$ was available to compare. Even though the performances of bAMLGA and iAMLGA seem likewise in terms of success rate, iAMLGA was found much less costly than bAMLGA. Therefore, iAMLGA can result faster convergence than bAMLGA and retain the solution quality at the same time. A similar scenario was observed for the multimodal functions of Group VI. The iAMLGA was found cost effective than bAMLGA with only one exception of $f_{Trigonometric}$.

In case of 7 rotated functions of Group VII, CMA-ES outperformed other algorithms with the lowest $\overline{NFE}$ given that it has successful runs for the function (5 and 4 out of 7 respectively for 30 and 50-dimesnional functions) (**Table XVIII** and **XIX**). In contrast, iAMLGA was successful as well as the least costly for $f_{R-Ackley}$ (both dimensions) and $f_{R-SchafferF7}$ (dimension value 50). The SSA had successful runs for one function only, however taking a very high NFE. None of the algorithms could obtain reportable success history so as $\overline{NFE}$ for very hard rotated and shifted functions of Group VIII. For hybrid functions, the comparison exists in between bAMLGA and iAMLGA only (**Table XVIII** and **XIX**) as they were the only successful algorithms. Exceptionally, bAMLGA took lower $\overline{NFE}$s than those of iAMLGA for 30-dimesnional $f_{Hybrid-1}$ (**Table XVIII**) and 50-dimensional $f_{Hybrid-2}$ (**Table XIX**). However, the success rate of bAMLGA was much lower. Thus, we implicate that it is worth to test both bAMLGA and iAMLGA to solve problems with real-life complexities like hybrid functions and use the best one.

Some other functions for which CMA-ES was found less costly were of Group II and V (**Table XX**). CMA-ES was better for 4 out of 7 non-scalable functions, whereas iAMLGA was better for 2 of them. However, CMA-ES failed to optimize f_{Bird} due to ill condition and no $\overline{NFE}$ was available.

4) Assessment of Overall Performance

In this part, we further assess the overall performance of the four algorithms by plotting the empirical cumulative distribution of normalized success performance ($\widehat{SP}$) [94]. The success performance (SP) is calculated using **(24)** that accounts both the reliability and cost of the algorithms. Therefore, it does not reward a high success rate at a cost of high NFE nor a low NFE with low and unreliable success rate. Rather, SP is a compound measure that indicates the overall significance of an algorithm.

$$SP = \overline{NFE} \times \frac{T_{runs}}{S_{runs}} \text{ and } \widehat{SP} = \frac{SP}{SP_{best}} \quad (24)$$

Here, T_{runs} is the total number of runs and set to 30, and S_{runs} is the number of successful runs. For each function, we first calculated the SP of each algorithm and then computed $\widehat{SP}$ by normalizing the SPs by the best algorithm's SP (SP_{best}) on that particular function. The $\widehat{SP}$s of the algorithms are reported in **Table XIX**, **XX** and **XXI**. Small values of SPs and so as $\widehat{SP}$s are preferable. The lowest and best value of $\widehat{SP}$ is 1 (bold faced in the tables). We plotted the distributions for different subsets of function groups (**Fig. 11**) to property-wise analyze the effectiveness of the algorithms. Note that, only the functions for which at least one algorithm had at least one successful run were considered in the distribution plot. The algorithm that quickly reaches the top of the curve with largest area underneath can be considered as the best.

Fig. 11(a) and **(b)** respectively show the distribution curves for 30D and 50D scalable and unimodal functions of both Group I (separable variables) and III (inseparable variables). All the 7 functions with these properties were considered in the plots. We observed that CMA-ES performed better than other algorithms for these unimodal functions.

Fig. 11(c) and **(d)** show respectively the distribution curves for 30D and 50D scalable and multimodal functions with both separable (Group IV) and inseparable variables (Group VI). There are 12 functions with these properties among which $f_{Schwefel2.26}$ was not successfully optimized to the global minima by any of the algorithm, therefore is not considered in the distribution plots. The performance of iAMLGA was clearly promising than other algorithms on these complicated multimodal functions. SSA and CMA-ES could optimize no greater than 40% and 50% of the functions respectively for both of the dimensions. To compare, iAMLGA and bAMLGA was successful for at least 90% and 80% of the functions respectively (**Fig. 11(c)-(d)**).

Plots for 13 modified functions of Group VII, VIII and IX are shown in **Fig. 11(e)** and **(f)**. We did not consider one rotated function ($f_{R-Lunacek}$), all three rotated and shifted functions and one hybrid function ($f_{Hybrid-3}$) for the distribution plots as none of the algorithms have success for these functions. Therefore, the distributions were generated for rest of 8 functions. The curves says that CMA-ES attained good performance, however on less number of functions. To compare, AMLGAs have higher coverage so as showed better overall performance.

Fig. 11(g) and **(h)** presents the distribution on all scalable functions of dimension 30 and 50 respectively. This two plots clearly demonstrates the robust as well as superior performance of the proposed algorithms than the others on a wide variety of problems. The last plot (**Fig. 11(i)**) shows the curves for 2D non-scalable functions with inseparable variables (both unimodal and multimodal). Only f_{Bird} was not included in the distributions as all algorithms failed for this function. This plot again focuses the better overall success performance of iAMLGA.

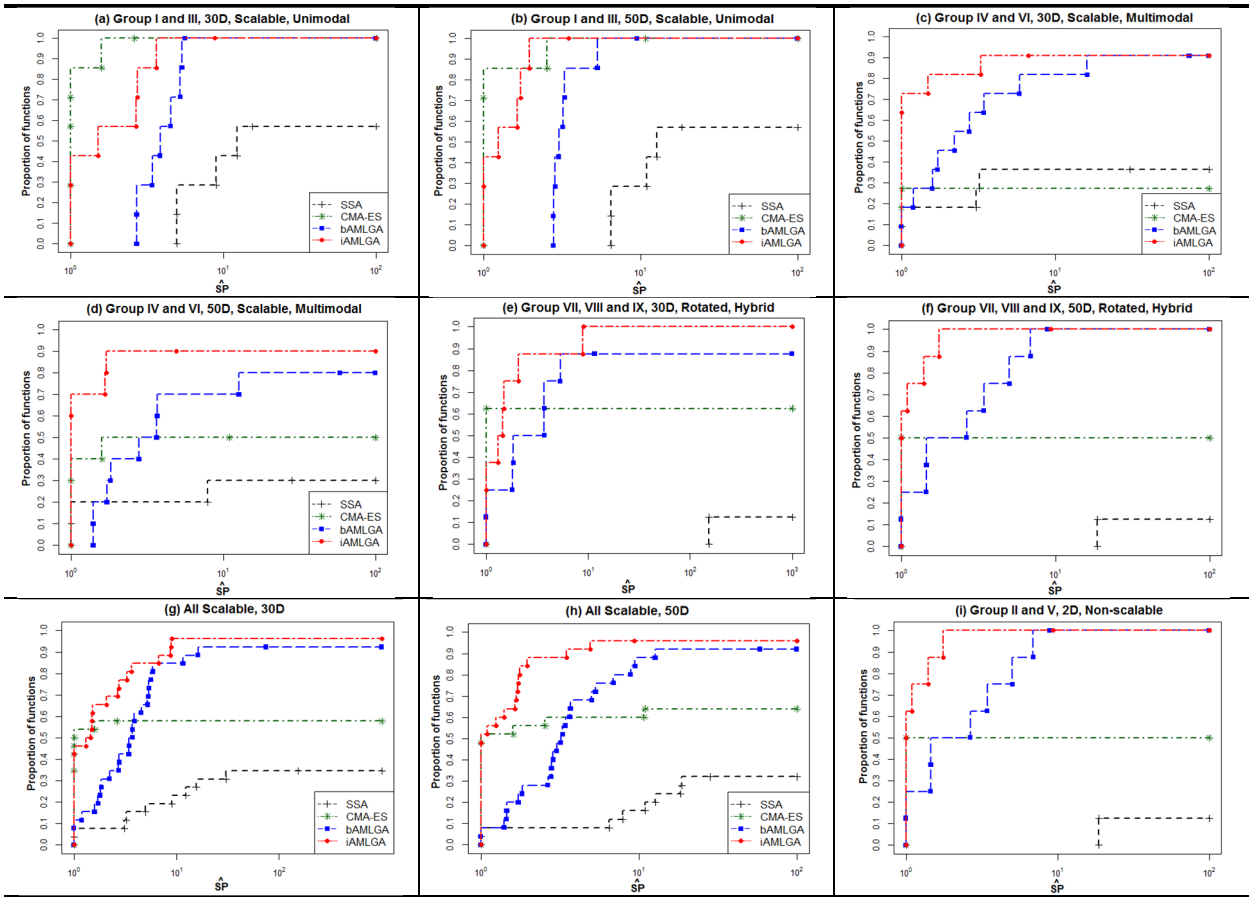

Fig. 11. Empirical cumulative distribution plots of normalized success performance.

VII. Conclusion

In this paper, we presented a new evolutionary algorithm called adaptive and memory-assisted local operator based genetic algorithm (AMLGA). The name of the algorithm features the three novel properties of the genetic operators included in this algorithm. First, it uses an improved elitism with hGR that not only ensures the survival of the fittest but also attempts to make the elites fitter. The hGR adaptively replaces the relatively weaker genes (substrings of a full binary chromosome) with the best gene schema if performance is improved. Subsequently, a novel crossover technique (LAmC) is employed locally on the genes to exploit the most information out of the recently distributed best gene schema. The LAmC is assisted by two piece of memories that stores the parts of currently available best gene at every crossover point. Thus after, these sub-parts is evaluated in addition to the sub-parts available from other partner and eventually the better one is used. These memories can essentially help in restoring the previously broken schema. After LAmC, a local single-point mutation is applied on the genes to complement the saturation impact of hGR. Finally, we employed a twin removal operator to ensure diversified individuals in the population.

In this study, we specially promoted the appropriate application of local (gene level) operators in GA. To justify our arguments, we investigated several combinations of both local and global operators in 8 different GA variants on 27 benchmark test functions. The simulation results empirically highlighted the effectiveness of adaptive and memory-assisted local operators especially in solving high-dimensional scalable optimization problems. We further quantified the cost of the basic AMLGA (bAMLGA) and proposed a cost-improved version of it named iAMLGA. The performances of iAMLGA and bAMLGA were evaluated with two state-of-the-art optimization algorithms, CMA-ES and SSA by attempting to minimize 40 test functions with various complexities. The promising results establish the proposed algorithm as a competitive optimization tool in terms of solution quality, reliability, and convergence speed and success performance. Specifically, we observed a superior performance from AMLGAs in case of multimodal problems. We further aim to apply the proposed algorithm on real-world engineering applications of optimization and are currently working on it.

References

1. Holland J. Adaptation in natural and artificial systems. Univ. of Michigan Press; 1975/1992.
2. Yoon H-S, Moon B-R. An Empirical Study on the Synergy of Multiple Crossover Operators. IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION. 2002;6(2):212 - 23.
3. Vasconcelos J, Ramirez J, Takahashi R, Saldanha R. Improvements in genetic algorithms. IEEE Transactions on Magnetics. 2001;37(5):3414-7.
4. Digalakis JG, Margaritis KG. An experimental study of benchmarking functions for genetic algorithms. International Journal of Computer Mathematics. 2002;79(4):403-16.
5. Kennedy J, Eberhart RC. Particle swarm optimization. Proceedings of IEEE International Conference on Neural Networks. 1995:1942 - 8.
6. Dorigo M, Maniezzo V, Colorni A. Ant system: optimization by a colony of cooperating agents. IEEE Transactions on Systems, Man, and Cybernetics Part B: Cybernetics. 1996;26(1):29-41.
7. Storn R, Price K. Differential Evolution – A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. Journal of Global Optimization. 1997;11(4):341 - 59.
8. Hansen N, Kern S, editors. Evaluating the CMA evolution strategy on multimodal test functions. International Conference on Parallel Problem Solving from Nature; 2004: Springer.
9. Holland J. Genetic Algorithms. Scientific American Journal. 1992:66 - 72.
10. Karaboga D. An idea based on honeybee swarm for numerica optimization. Technical Report TR06, Erciyes University. 2005.
11. Krishnanand K, Ghose D, editors. Detection of multiple source locations using a glowworm metaphor with applications to collective robotics. Swarm Intelligence Symposium, 2005 SIS 2005 Proceedings 2005 IEEE; 2005: IEEE.
12. Yang X-S, Deb S. Cuckoo Search via Lévy flights. World Congress on Nature & Biologically Inspired Computing (NaBIC). 2009:210 - 4
13. Fateen S-EK, Bonilla-Petriciolet A. Intelligent firefly algorithm for global optimization. Cuckoo Search and Firefly Algorithm: Springer; 2014. p. 315-30.
14. Yang X-S, Hossein Gandomi A. Bat algorithm: a novel approach for global engineering optimization. Engineering Computations. 2012;29(5):464-83.
15. Zhao R, Tang W. Monkey algorithm for global numerical optimization. J Uncertain Syst. 2008;2:165 - 76.
16. Gandomi AH, Alavi AH. Krill herd: A new bio-inspired optimization algorithm. Nonlinear Sci Numer Simul. 2012;17:4831 - 45.
17. Bayraktar Z, Komurcu M, Werner DH, editors. Wind Driven Optimization (WDO): A novel nature-inspired optimization algorithm and its application to electromagnetics. Antennas and Propagation Society International Symposium (APSURSI), 2010 IEEE; 2010: IEEE.
18. Yu JJQ, Li VOK. A Social Spider Algorithm for Global Optimization. Neural and Evolutionary Computing. 2015;30(C):614 - 27.
19. Gong M, Cai Q, Chen X, Ma L. Complex network clustering by multiobjective discrete particle swarm optimization based on decomposition. Evolutionary Computation, IEEE Transactions on. 2014;18(1):82-97.
20. Iqbal S, Kaykobad M, Rahman MS. Solving the multi-objective Vehicle Routing Problem with Soft Time Windows with the help of bees. Swarm and Evolutionary Computation. 2015;24:50-64.
21. Zhang Z, Zhang N, Feng Z. Multi-satellite control resource scheduling based on ant colony optimization. Expert Systems with Applications. 2014;41(6):2816-23.
22. Maher B, Albrecht AA, Loomes M, Yang X-S, Steinhöfel K. A firefly-inspired method for protein structure prediction in lattice models. Biomolecules. 2014;4(1):56-75.
23. Rashid MA, Khatib F, Hoque MT, Sattar A. An Enhanced Genetic Algorithm for Ab initio Protein Structure Prediction. IEEE Transactions on Evolutionary Computation. 2015;(99):1. doi: 10.1109/TEVC.2015.2505317.
24. Bao Z, Zhou Y, Li L, Ma M. A Hybrid Global Optimization Algorithm Based on Wind Driven Optimization and Differential Evolution. Mathematical Problems in Engineering. 2015;2015.
25. DeJong K. An analysis of the behavior of a class of genetic adaptive systems. Ph D Thesis, University of Michigan, Ann Arbor. 1975.
26. Goldberg DE. Genetic and evolutionary algorithms come of age. Communications of the ACM. 1994;37(3):113-20.
27. Muhlenbein H. How genetic algorithms really work: I. mutation and hillclimbing. Parallel Problem Solving from Nature 2. B. Manderick. Amsterdam, Elsevier; 1992.
28. Srinivas M, Patnaik LM. Genetic algorithms: A survey. Computer. 1994;27(6):17-26.
29. Vose MD. Modeling simple genetic algorithms. Foundations of genetic algorithms. 1993;2:63-73.
30. Davis L. Handbook of genetic algorithms. 1991.
31. Schmitt LM. Theory of genetic algorithms. Theoretical Computer Science. 2001;259(1):1-61.
32. Goldberg DE. Simple genetic algorithms and the minimal, deceptive problem. Genetic algorithms and simulated annealing. 1987;74:88.
33. Goldberg DE. Genetic Algorithms in Search. Optimization and Machine Learning, Reading, Massachusetts. 1989.
34. Mühlenbein H, Schlierkamp-Voosen D. Predictive models for the breeder genetic algorithm i. continuous parameter optimization. Evolutionary computation. 1993;1(1):25-49.
35. Mühlenbein H, Schlierkamp-Voosen D. The science of breeding and its application to the breeder genetic algorithm (BGA). Evolutionary Computation. 1993;1(4):335-60.
36. Koumousis VK, Katsaras CP. A saw-tooth genetic algorithm combining the effects of variable population size and reinitialization to enhance

performance. Evolutionary Computation, IEEE Transactions on. 2006;10(1):19-28.
37.Kühn M, Severin T, Salzwedel H. Variable Mutation Rate at Genetic Algorithms: Introduction of Chromosome Fitness in Connection with Multi-Chromosome Representation International Journal of Computer Applications 2013;72(17):31 - 8.
38.Back T, Hoffmeister F, Schwefel H, editors. A survey of evolutionary strategies. Proceedings of the 4th international conference on genetic algorithms Morgan Kaufmann publishers, San Mateo, CA; 1991.
39.Whitley D, Starkweather T. Genitor II: A distributed genetic algorithm. Journal of Experimental & Theoretical Artificial Intelligence. 1990;2(3):189-214.
40.Whitley LD, editor The GENITOR Algorithm and Selection Pressure: Why Rank-Based Allocation of Reproductive Trials is Best. ICGA; 1989.
41.Whitley D, Kauth J. GENITOR: A different genetic algorithm: Colorado State University, Department of Computer Science; 1988.
42.Eshelman LJ. The CHC adaptive search algorithm: How to have safe search when engaging. Foundations of Genetic Algorithms 1991 (FOGA 1). 2014;1:265.
43.Voset MD, Liepinsl GE. Punctuated equilibria in genetic search. Complex systems. 1991;5:31-44.
44.Whitley D. An executable model of a simple genetic algorithm. Foundations of genetic algorithms. 1993;2(1519):45-62.
45.Whitley D. A genetic algorithm tutorial. Statistics and computing. 1994;4(2):65-85.
46.Goldberg DE. A note on Boltzmann tournament selection for genetic algorithms and population-oriented simulated annealing. Complex Systems. 1990;4(4):445-60.
47.Goldberg DE, Deb K. A comparative analysis of selection schemes used in genetic algorithms. Foundations of genetic algorithms. 1991;1:69-93.
48.Gorges-Schleuter M, editor Explicit parallelism of genetic algorithms through population structures. International Conference on Parallel Problem Solving from Nature; 1990: Springer.
49.Belding TC. The distributed genetic algorithm revisited. arXiv preprint adap-org/9504007. 1995.
50.Starkweather T, Whitley D, Mathias K, editors. Optimization using distributed genetic algorithms. International Conference on Parallel Problem Solving from Nature; 1990: Springer.
51.Tanese R, editor Distributed genetic algorithms. Proceedings of the third international conference on Genetic algorithms; 1989: Morgan Kaufmann Publishers Inc.
52.Whitley D, editor Cellular genetic algorithms. Proceedings of the 5th International Conference on Genetic Algorithms; 1993: Morgan Kaufmann Publishers Inc.
53.Hoque MT, Chetty M, Dooley LS. Generalized schemata theorem incorporating twin removal for protein structure prediction. Pattern Recognition in Bioinformatics: Springer; 2007. p. 84-97.
54.Higgs T, Stantic B, Hoque MT, Sattar A, editors. Refining genetic algorithm twin removal for high-resolution protein structure prediction. IEEE Congress on Evolutionary Computation, ; 2012: IEEE.
55.Hoque MT, Chetty M, Lewis A, Sattar A. Twin removal in genetic algorithms for protein structure prediction using low-resolution model. IEEE/ACM Transactions on Computational Biology and Bioinformatics. 2011;8(1):234-45.
56.Hoque MT. Genetic Algorithms based Improved Sampling. Tech Report TR-2015/4. 2015.
57.Iqbal S, Hoque MT, editors. (Extended abstract) A Homologous Gene Replacement based Genetic Algorithm. GECCO; 2016; Denver, Colorado, USA.
58.Iqbal S, Hoque MT. A Scalable Gene Replacement Operator for Genetic Algorithm. 2016.
59.Mitchell M. Genetic Algorithm: An Overview. Complexity. 1995;1(1):31 - 9.
60.Townsend A. Genetic Algorithms–a Tutorial. Citeseer; 2003.
61.Mitchell M. An introduction to genetic algorithms: MIT press; 1998.
62.Iqbal S, Hoque MT, editors. (Abstract) A Homologous Gene Replacement based Genetic Algorithm. GECCO; 2016; Denver, Colorado, USA.
63.Ronald S, editor Duplicate genotypes in a genetic algorithm. Evolutionary Computation Proceedings, 1998 IEEE World Congress on Computational Intelligence, The 1998 IEEE International Conference on; 1998: IEEE.
64.Michalewicz Z. Genetic algorithms+ data structures= evolution programs: Springer Science & Business Media; 2013.
65.Whitley D. An overview of evolutionary algorithms: practical issues and common pitfalls. Information and software technology. 2001;43(14):817-31.
66.Haupt RL, Haupt SE. Practical genetic algorithms: John Wiley & Sons; 2004.
67.Hoque M, Chetty M, Dooley L. Critical Analysis of the Schemata Theorem: The Impact of Twins and the Effect in the Prediction of Protein Folding Using Lattice Model. GSIT, MONASH University. 2005.
68.Tuson A, Ross P. Adapting operator settings in genetic algorithms. Evolutionary computation. 1998;6(2):161-84.
69.Gómez J, Dasgupta D, González F, editors. Using adaptive operators in genetic search. Genetic and Evolutionary Computation Conference; 2003: Springer.
70.Julstrom BA. What have you done for me lately?{A} dapting operator probabilities in a steady-state genetic algorithm. Int Conf Genetic Algorithms; Jul. 15 - 19; Pittsburg, PA1995. p. 81 - 7.
71.Smith JE, Fogarty TC. Operator and parameter adaptation in genetic algorithms. Soft computing. 1997;1(2):81-7.
72.Angeline PJ, editor Adaptive and self-adaptive evolutionary computations. Computational intelligence: a dynamic systems perspective; 1995: New York: IEEE Press.
73.Eiben ÁE, Hinterding R, Michalewicz Z. Parameter control in evolutionary algorithms. IEEE Transactions on evolutionary computation. 1999;3(2):124-41.
74.Ho Y-C, Pepyne DL. Simple explanation of the no-free-lunch theorem and its implications. Journal of optimization theory and applications. 2002;115(3):549-70.
75.Wolpert DH, Macready WG. No free lunch theorems for optimization. Evolutionary Computation, IEEE Transactions on. 1997;1(1):67-82.
76.Liang J, Qu B, Suganthan P, Hernández-Díaz AG. Problem definitions and evaluation criteria for the CEC 2013 special session on real-parameter optimization. Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou, China and Nanyang Technological University, Singapore, Technical Report. 2013;201212.
77.Liang J, Qu B, Suganthan P. Problem definitions and evaluation criteria for the CEC 2014 special session and competition on single objective real-parameter numerical optimization. Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore. 2013.
78.Wahab MNA, Nefti-Meziani S, Atyabi A. A Comprehensive Review of Swarm Optimization Algorithms. PLOS One. 2015.
79.Dieterich JM, Hartke B. Empirical review of standard benchmark functions using evolutionary global optimization. arXiv preprint arXiv:12074318. 2012.
80.Liang J-J, Suganthan PN, Deb K, editors. Novel composition test functions for numerical global optimization. Proceedings 2005 IEEE Swarm Intelligence Symposium, 2005 SIS 2005; 2005: IEEE.
81.Zhong W, Liu J, Xue M, Jiao L. A multiagent genetic algorithm for global numerical optimization. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics). 2004;34(2):1128-41.
82.Van den Bergh F, Engelbrecht AP. A cooperative approach to particle swarm optimization. IEEE transactions on evolutionary computation. 2004;8(3):225-39.
83.Leung Y-W, Wang Y. An orthogonal genetic algorithm with quantization for global numerical optimization. IEEE Transactions on Evolutionary computation. 2001;5(1):41-53.
84.Coello CAC, Pulido GT, Lechuga MS. Handling multiple objectives with particle swarm optimization. IEEE Transactions on evolutionary computation. 2004;8(3):256-79.
85.Qin AK, Huang VL, Suganthan PN. Differential evolution algorithm with strategy adaptation for global numerical optimization. IEEE transactions on Evolutionary Computation. 2009;13(2):398-417.
86.Surjanovic S, Bingham D. Virtual Library of Simulation Experiments: Test Functions and Datasets 2013 [cited 2015 October 26]. Available from: http://www.sfu.ca/~ssurjano.
87.Dill KA, Chan HS. From Levinthal to pathways to funnels. Nature Structural Biology. 1997;4:10 - 9.
88.Tan Y, Shi Y, Coello CC. Advances in Swarm Intelligence: 5th International Conference, ICSI 2014, Hefei, China, October 17-20, 2014, Proceedings: Springer; 2014.
89.Locatelli M. A Note on the Griewank Test Function. Journal of Global Optimization. 2002-03;25(2):169 - 74.
90.Dieterich JM, Hartke B. Empirical review of standard benchmark functions using evolutionary global optimization. Applied Mathematics. 2012;3(10A):1552-64.
91.Salomon R. Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions. A survey of some theoretical and practical aspects of genetic algorithms. BioSystems. 1996;39(3):263-78.

92.Hansen N, Ostermeier A, editors. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. Proceedings of IEEE International Conference on Evolutionary Computation; 1996: IEEE.
93.Loshchilov I, Schoenauer M, Sebag M, editors. Self-adaptive surrogate-assisted covariance matrix adaptation evolution strategy. Proceedings of the 14th annual conference on Genetic and evolutionary computation; 2012: ACM.
94.Hansen N. Compilation of Results on the 2005 CEC Benchmark Function Set. Available: http://www.ntu.edu.sg/home/epnsugan/index_files/CEC-05/compareresults.pdf May 4, 2006.
95.Auger A. Performance Assessment in Optimization. Available: https://www.lri.fr/~auger/teaching-slides/05_PerformanceAssessment.pdf [online]. January 14, 2016 [03 August, 2016]. Available from: https://www.lri.fr/~auger/teaching.html.
96.Finck S, Hansen N, Ros R, Auger A. Real-parameter black-box optimization benchmarking 2009: Presentation of the noiseless functions. Citeseer, 2010.
97.Lawler GF. Introduction to stochastic processes: CRC Press; 2006.