2013

An Improved Hybrid Genetic Algorithm with a New Local Search Procedure

Wen Wan
Virginia Commonwealth University, wwan@vcu.edu

Jeffrey B. Birch
Virginia Polytechnic Institute and State University

Recommended Citation

Wen Wan and Jeffrey B. Birch, "An Improved Hybrid Genetic Algorithm with a New Local Search Procedure," Journal of Applied Mathematics, vol. 2013, Article ID 103591, 10 pages, 2013. doi:10.1155/2013/103591

Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2013, Article ID 103591, 10 pages
http://dx.doi.org/10.1155/2013/103591

Research Article

An Improved Hybrid Genetic Algorithm with a New Local Search Procedure

Wen Wan[1] and Jeffrey B. Birch[2]

[1] *Department of Biostatistics, Virginia Commonwealth University, Richmond, VA 23298-0032, USA*
[2] *Department of Statistics, Virginia Polytechnic Institute and State University, Blacksburg, VA 24061-0439, USA*

Correspondence should be addressed to Wen Wan; wwan@vcu.edu

Received 9 January 2013; Accepted 26 August 2013

Academic Editor: Bin Wang

One important challenge of a hybrid genetic algorithm (HGA) (also called memetic algorithm) is the tradeoff between global and local searching (LS) as it is the case that the cost of an LS can be rather high. This paper proposes a novel, simplified, and efficient HGA with a new individual learning procedure that performs a LS only when the best offspring (solution) in the offspring population is also the best in the current parent population. Additionally, a new LS method is developed based on a three-directional search (TD), which is derivative-free and self-adaptive. The new HGA with two different LS methods (the TD and Neld-Mead simplex) is compared with a traditional HGA. Four benchmark functions are employed to illustrate the improvement of the proposed method with the new learning procedure. The results show that the new HGA greatly reduces the number of function evaluations and converges much faster to the global optimum than a traditional HGA. The TD local search method is a good choice in helping to locate a global "mountain" (or "valley") but may not perform the Nelder-Mead method in the final fine tuning toward the optimal solution.

1. Introduction

Genetic algorithms (GAs) perform well as a global search technique, but they may often take a relatively long time to converge to a global optimum [1–4]. Local search (LS) techniques have been incorporated into GAs to improve their performance through what could be termed as learning. Such HGAs, often known as memetic algorithms (MAs), were first introduced by Moscato [5, 6] and are viewed as a form of population-based genetic algorithms hybridized with an individual learning procedure capable of fine tuning the global search.

MAs represent one of the recent growing areas of research in evolutionary computation [7, 8]. Any population-based metaheuristic search method (inspired by Darwinian principles of natural selection) hybridized with any individual learning (inspired by Dawkins' notation "meme" [9]) procedure that belongs to the class of MAs [7]. In diverse contexts, MAs have also been referred to as hybrid evolutionary algorithms, Baldwinian evolutionary algorithms, Lamarkian evolutionary algorithms, cultural algorithms, or a genetic local search.

MAs have been successfully applied to hundreds of real-world problems in a wide range of domains [3, 7, 8, 10]. An important challenge of MAs is the tradeoff between global searching and local searching in terms of the time and computational effort [3, 10–14]; that is, the yet unanswered questions are when to apply a LS technique; to which individuals in the GA (or any other evolutionary algorithms) population should the LS technique be applied; and how much computational effort should be devoted to the LS technique. Recent literature presented several nonclassical MA methods that have been successful in reducing the total computational costs associated with an LS technique and that produce a profitable synergy from the hybridization of the GA (or any other evolutionary algorithms) and LS methods [7, 14–19]. But none of the nonclassical MAs are commonly accepted [10, 14]. Additionally, some of these methods, such as Seront and Bersini [15], Tang et al. [17], and Molina et al. [18, 19], may require the need for extra parameters.

Another challenge of MAs is the choice of successful LS techniques. Ning et al. [20] investigated the choice of LS techniques in HGAs and concluded that the choice affects

the search performance significantly and no single HGA always performs best on a diverse set of benchmark test functions.

In this study, to reduce the computational effort of an LS method without any extra parameters, a new HGA, called "a best-offspring HGA," denoted by BOHGA, is developed with a new individual learning procedure; that is, BOHGA performs an LS only when the best offspring (solution) in the offspring population is also the best in the current parent population. Additionally, a new LS method, a three-directional local search (TD), is introduced which is derivative-free and self-adaptive. The main idea of TD is that when the offspring performs better than both of its parents, three potential directions are constructed from parents to one of their offspring with a certain step length. We compare the new individual-learning HGA, BOHGA, with a traditional HGA, each using two memes: our TD method and the Neld-Mead simplex method. Both of these memes are derivative-free and suitable for real applications.

The remainder of this paper is organized as follows. We first briefly review the traditional GA and HGA. Our new HGA is introduced with its new individual learning procedure on when to perform the LS and on which offspring. We then present the two memes, respectively: one is the three-directional search (TD) and the other is the Nelder-Mead simplex meme. Through two benchmark functions, we present results for comparing the four HGAs for eight different settings of the GA operators and two different stopping rules. Finally, we present conclusions, discussions, and suggestions for future work.

2. The Genetic Algorithm and Hybrid Genetic Algorithm

Genetic algorithms (GAs) are iterative optimization procedures that repeatedly apply GA operators (such as selection, crossover, and mutation) to a group of solutions until some criterion of convergence has been satisfied. In a GA, a search point (solution), a setting in the search space with k dimensions (k variables), is coded into a string, $\mathbf{x} = [x_1, \ldots, x_k]'$, which is analogous to a *chromosome* in biological systems. The string/chromosome is composed of k characters, $x_1, \ldots, x_k$, which are analogous to the k *genes*. A set of multiple concurrent search points or a set of chromosomes (or individuals) is called a *population*. Each iterative step where a new population is obtained is called a *generation*. A GA hybridized with a local search procedure is called a hybrid genetic algorithm (HGA).

A basic HGA procedure has the following steps.

(1) Define an objective/fitness function, and set the GA operators (such as population size, parent/offspring ratio, selection method, number of crossovers, and mutation rate).

(2) Randomly generate the initial population as the current parent population.

(3) Evaluate the objective function for each individual (chromosome or solution) in the initial population.

(4) Generate an offspring population by using GA operators (such as selection/mating, crossover, and mutation).

(5) Evaluate the objective function of each individual in the offspring population.

(6) Perform a local search on each offspring, evaluating fitness of each new location, and replace the offspring if there exists a locally improved solution.

(7) Decide which individuals to include in the next population. This step is referred to as "replacement" in that individuals from the current parent population are "replaced" by a new population consisting of those individuals from the offspring and/or the parent populations.

(8) If a stopping criterion is satisfied, then the procedure is halted. Otherwise, go to Step 4.

Without Step 6, an HGA is just a GA. Therefore, HGAs have all the properties possessed by GAs. Like GAs, HGAs are a large family of algorithms that have the same basic structure but differ from one another with respect to several strategies such as stopping rules, operators which control the search process, and the local search meme.

Based on previous experiences, in this study, we use a continuous HGA where chromosomes are coded as continuous measurement variables. Suppose there are k variables; that is, there are k genes in each chromosome. We also make the following assumptions. The (parent) population size is $2k$ and the offspring population size is also $2k$. The type of selection we utilize is random pairing. The blending crossover is utilized and the number of crossover points depends on the number of dimensions of a specific objective function. Random uniform mutation is utilized and the mutation rate is set around or equal to $1/k$. The type of replacement over both parent and offspring populations is either ranking or tournament. For details on the setting of the GA operators; see, for example, [21–25].

There are many choices of local search memes [20], two of which are used in this study. One meme is our newly developed "three-directional LS (TD)," introduced in Section 4. A second meme is a popular LS meme, the Nelder-Mead Simplex method, introduced in Section 5.

3. The Best-Offspring Hybrid Genetic Algorithm

As mentioned, our goal is to reduce the total costs associated with the LS. It has been noticed that the LS may be repeatedly performed on the same "mountain" (for finding a maximum) or "valley" (for finding a minimum) [15]. Therefore, it is possible that, after local searching, several chromosomes in a generation are very close to each other, standing on the same top of a mountain or at the same bottom of a valley. This may make it harder for the GA to maintain diversity in its population, an important consideration in avoiding converging to a local optimum [25]. Therefore, we propose the best-offspring HGA (BOHGA) where the LS is only performed on the best offspring in the offspring

population when it is also the best overall chromosomes in the current parent population. When such a best offspring appears, it is very likely that the best offspring is located on a new, higher mountain or on a new lower valley. As will be soon demonstrated, this action tends to make BOHGA more computationally efficient and helps to prevent converging to a local optimum.

The general procedure for BOHGA is the same as that of HGA, except that in the ith generation we change Step 6 from the original HGA procedure into Steps 6.1–6.3 as follows.

(6.1) Is the best offspring in the offspring population also the best over the current parent population?

(6.2) If no, directly go to Step 7; that is, there is no LS in this generation.

(6.3) If yes, then perform an LS on the best offspring considered as a starting point. Find the best locally improved solution and replace the best offspring by it. Then go to Step 7.

Actually, the BOHGA process is a special HGA process where an LS is not performed on every new offspring but only on the offspring which are best in both the offspring and the current parent populations. It is possible that not every generation of BOHGA requires an LS. The BOHGA procedure, therefore, strongly agrees with the original idea of MA, first introduced by Mascato in 1989 [5]; that is, initially let the GA explore a wide search space. Once a potential search solution is found by a GA, a fine tuning search will be conducted by an LS. Similar to both the GA and the HGA, the whole process is iterated until some appropriate stopping rule is satisfied.

4. A Three-Directional (TD) Meme

The idea of the TD meme is to construct three potential directions for an offspring whose performance is better than both of its parents in a generation. Thus, three paths are declared without requiring the gradient. When an offspring shows improvement from its parents in terms of the objective function, it may be possible to make continuous improvements by moving along the directions/paths from its parents to the offspring; that is, some search points are "collected" along the paths until no further improvement can be found. These parents can be considered as two different starting points. Both of their first steps from the two starting points go to the same point: the offspring. So two directions are established: one direction is from one of the parents to the offspring; the other is from the second of the parents to the offspring. Both directions have obtained improvement, since the best offspring of interest is an improvement over both its parents in terms of values of an objective function.

For example, consider a 2-dimensional ($k = 2$) problem along with the contours of a response (or values of an objective function) as illustrated in Figure 1. The offspring is denoted by O (expressed as $\mathbf{x}_O = [x_{O1}, \ldots, x_{Ok}]'$) and its parents are denoted by $P1$ ($\mathbf{x}_{P1} = [x_{P11}, \ldots, x_{P1k}]'$) and $P2$ ($\mathbf{x}_{P2} = [x_{P21}, \ldots, x_{P2k}]'$). Obviously, there are two directions: one is from $P1$ to O, expressed as $\delta_{P1O} = \mathbf{x}_O - \mathbf{x}_{P1} = [\delta_{11}, \delta_{12}, \ldots, \delta_{1k}]'$, and the other is from $P2$ to O, expressed as $\delta_{P2O} = \mathbf{x}_O - \mathbf{x}_{P2} = [\delta_{21}, \delta_{22}, \ldots, \delta_{2k}]'$. We refer to these two directions as Parent 1 and Parent 2 directions.

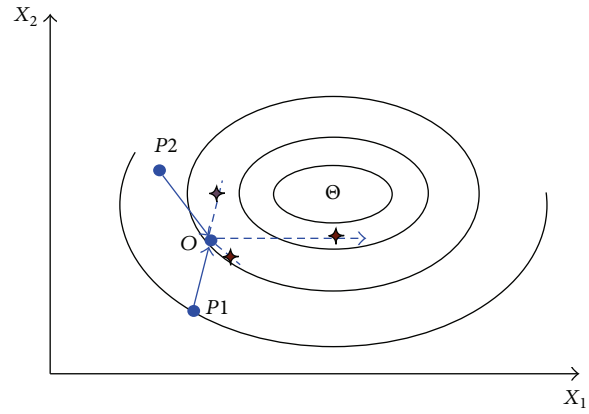


Figure 1: A contour plot of a 2-dimensional problem with the three directions indicated: Parent 1 direction is from $P1$ to O; Parent 2 direction is from $P2$ to O; the common direction is a horizontal dotted line, starting at O towards the positive values on the X_1 axis. The three "stars" represent the three points stopped on the three paths with no further improvement.

The third direction is the "common" direction, expressed as $\delta = [\delta_{31}, \delta_{32}, \ldots, \delta_{3k}]'$, and based on the two parent directions. If δ_{1i} and δ_{2i}, for $i = 1, \ldots, k$, are both positive (negative), then δ_{3i} is positive (negative); that is, if both the parent directions are in common, say, both positive (negative) along the X_i axis, then the third direction is positive (negative) along the X_i axis. If δ_{1i} and δ_{2i}, for $i = 1, \ldots, k$, are opposite in direction, then δ_{3i} is set to 0; that is, if the parent directions are not in common on the X_i axis, then the third direction has no movement along the X_i axis. For more details on the three directions and determining their moving distances for each moving step, see the Appendix.

Figure 1 illustrates the three defined directions. The optimal point is denoted by "Θ." It is easy to see the two parents directions, expressed as $\delta_{P1O} = [\delta_{11}, \delta_{12}]'$ and $\delta_{P2O} = [\delta_{21}, \delta_{22}]'$, respectively. The third direction $\delta = [\delta_{31}, \delta_{32}]'$. Obviously, $\delta_{31} > 0$ since both $\delta_{11} > 0$ and $\delta_{21} > 0$; that is, the common direction in this case is positive along the X_1 axis. And $\delta_{32} = 0$ since $\delta_{12} > 0$ and $\delta_{22} < 0$; that is, the common direction has no relative movement along the X_2 axis.

Once the three directions are defined, starting at O, the TD method moves along the three directions/paths, with some appropriate step length for each moving step until no improvement is found in terms of an objective function. In Figure 1, the three "stars" on the paths denote that the three best points found on each path and the processes of moving along the paths will be stopped at their next points due to no further improvement.

The choice of the size of step length d depends on the degree of bumpiness of the surface of an objective function. We recommend that d should be in the physical range of 0.01 to 1.0. If the surface is very bumpy relative to the region of the domain, then the appropriate d should be relatively small. Otherwise, the appropriate d should be relatively large to make HGA more efficient.

In our BOHGA procedure, the TD meme will only be performed for the best offspring in the offspring population that is also the best in the current parent population. In our HGA procedure, the TD meme will be performed for those offspring whose performances are better than both of their parents. Since not every offspring performs better than either one of its parents, the TD meme will not be performed on every offspring, which is the major difference from a traditional HGA.

5. Nelder-Mead Simplex Meme

The Nelder-Mead simplex method [26] is a very popular derivative-free method for finding a local minimum of a function [8]. For a two-dimensional problem, a simplex is a triangle, and the method is a pattern search that compares function values at the three vertices of a triangle. The worst vertex, where $f(x, y)$ is largest, is rejected and replaced with a new vertex. A new triangle is formed and the search is continued. The process generates a sequence of triangles (which might have different shapes), for which the function values at the vertices get smaller and smaller. The size of the triangles is iteratively reduced and the coordinates of the minimum point are found. The simplex algorithm can easily be extended to higher dimensions [26]. In many numerical tests, the simplex method succeeds in obtaining a good reduction in the function value using a relatively small number of function evaluations but it is easy to converge to a local optimum and is generally not suitable for a highly nonlinear objective function [26].

Like the TD meme, the simplex meme requires a prespecified step length parameter, representing a guess of the problem's characteristic length scale. In this study, the step length parameter is set to the same size as d for the fair comparison between the simplex and TD memes. The C code for the simplex method is obtained from Numerical Recipes in C [27].

6. Examples: Benchmark Functions

Using benchmark functions, our goal is to compare our BOHGA with a traditional HGA, with each procedure using one of the two LS techniques: our new TD method or the simplex method; that is, we compare the computational efficiency of four MAs: BOHGA with simplex (denoted as "BOHGA_S"), BOHGA with TD ("BOHGA_TD"), HGA with simplex ("HGA_S"), and HGA with TD ("HGA_TD") in computational efficiency for the four objective benchmark functions. As mentioned, HGA_TD is different from the traditional HGA in that the TD local search will be performed only for those offspring whose performances are better than both their parents.

To make the comparisons comparable, the settings of the GA operators and the starting random numbers that are used to generate the initial populations are the same for each of the four MAs. In addition, since different starting random seeds may result in a different number of function evaluations to find an optimum, a Monte Carlo experiment is performed 100 times; that is, these four algorithms are run 100 times with 100 different starting random seeds. The four methods will be compared by averaging the results over the 100 replications of the experiment.

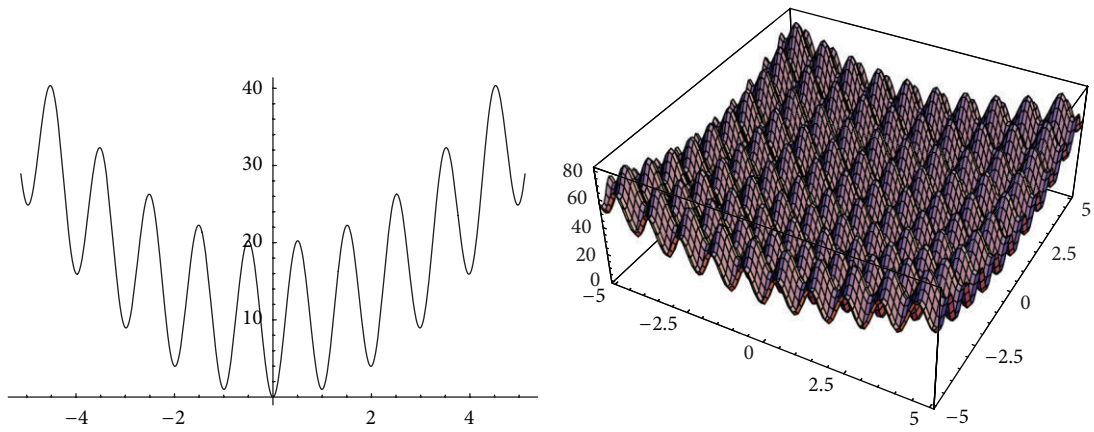
A different setting of GA operators may result in a different number of function evaluations. We choose 20 ($k = 20$) as a number of dimensions for the four benchmark functions. Therefore, as indicated in Section 2, both parent and offspring population sizes are 40. The number of crossover points is 4 or 8. The mutation rate is 0.05 ($= 1/k$) or 0.06, a slightly larger value than $1/k$. The type of replacement over both parent and offspring populations is ranking or tournament. Therefore, there are a total of eight combinations of crossover, mutation, and replacement type; that is, there are eight GA settings used for comparisons.

Also two stopping rules are utilized for the experiment. The first stopping rule (rule 1) is that a method will be halted when a preset cut-off value (considered as a near-global optimum) is achieved. The cut-off value represents the user's best guess of the optimal value of the objective function. Rule 1 can be used to compare the computational efficiencies of the four methods in finding a near-global optimum of an objective function. The mean of a total number of function evaluations over 100 replications of each MA will be used for comparisons. Since sometimes the global and near-global optimal values are unknown, a second stopping rule (rule 2) is also considered. The second stopping rule is that a method will be halted at a preselected number of generations. Under rule 2, the number of function evaluations it takes for the four methods to converge to a global "mountain" or "valley" or even to a global optimum is compared; that is, the rate of convergence to a near-global or global optimum is compared across the four methods. Obviously, it is not relevant to compare the total number of function evaluations required given a fixed total number of generations. Graphs will be used to illustrate the comparisons of the four methods, by plotting mean best values of the objective function over 100 replications at each generation found by each method versus mean cumulative number of function evaluations at each generation by each algorithm. Four benchmark functions (the Rastrigin's, Schwefel's, Rosenbrock's, and Griewank's) are used for the comparisons, but, due to similar results and limited space, only the first two functions are presented as follows.

6.1. Comparisons for the Rastrigin's Function in 20 Dimensions. A generalized Rastrigin's function is given by

$$f(\mathbf{x}) = \sum_{i=1}^{k} \left(x_i^2 - 10\cos\left(2\pi x_i\right) + 10\right), \quad \text{where } -5.12 \le x_i \le 5.12, \tag{1}$$

where k is the number of dimensions of the function ($k = 20$ in the study). Figure 2 shows its 1- and 2-dimensional surfaces. The surfaces are very bumpy in a narrow range $(-5.12, 5.12)$. The goal is to find a minimal value and its corresponding location. The minimum of this function is known as $\min(f(\mathbf{x})) = f(0, \ldots, 0) = 0.0$. From the left plot of



(a) (b)

Figure 2: Surface of Rastrigin's function: (a) 1-dimension; (b) 2-dimension.

Table 1: Comparisons of $BOHGA_S$, HGA_S, $BOHGA_{TD}$, and HGA_{TD} in terms of mean number of evaluations under the eight settings of GA operators for the Rastrigin's function in 20 dimensions by stopping rule 1.

8 settings of GA operators			Mean (evaluation)			
Replacement	Crossover	Mutation	$BOHGA_S$	HGA_S	$BOHGA_{TD}$	HGA_{TD}
Ranking	4	0.05	**29174**	510436	41352	39420
		0.06	**29688**	549620	46608	40174
	8	0.05	33951	463661	**30980**	34720
		0.06	**30052**	510260	35135	33627
Tournament	4	0.05	**62758**	988761	132271	112930
		0.06	**113071**	1424709	258692	265373
	8	0.05	**91747**	1054834	214672	208569
		0.06	**212320**	1538475	765658	880024
Overall average			**75345**	880095	190671	201855

Figure 2, a solution must be located on the global valley where the value of the objective function is less than about 1.0.

The step length for the TD meme is set to 0.05, the same value as for the simplex meme. The cut-off value used by rule 1, which is a near-global optimum, is set to 0.05. The preselected number of generations used by stopping rule 2 is 5,000.

Under stopping rule 1, Table 1 presents the mean number of function evaluations as a summary of the 100 repetitions for the Rastrigin's function in 20 dimensions for comparisons of the four algorithms. Table 1 shows that the number of evaluations required to obtain a value of the objective function is within 0.05 of the true minimum. $BOHGA_S$ consistently performs the best with much smaller mean numbers of function evaluations than the $BOHGA_{TD}$ and HGA_{TD}, which are quite competitive to each other. In most of all the GA settings, $BOHGA_S$ has the smallest mean number of function evaluations, followed by $BOHGA_{TD}$, HGA_{TD}, and HGA_S. In addition, the mean number of function evaluations greatly depends on the GA settings. The GA using ranking replacement obviously performs much better than the GA with tournament replacement in all of the four methods indicating that tournament replacement in MAs is not as efficient as ranking replacement. The mutation rate of 0.05 performs better than the rate of 0.06 in most cases.

Under stopping rule 2 with 5,000 generations, Figure 3 shows the mean best minimums of Rastrigin's function versus mean cumulative number of function evaluations at each generation over 100 replications by $BOHGA_S$, HGA_S, $BOHGA_{TD}$, and HGA_{TD}, respectively. The GA parameters were set at the ranking replacement, four crossover points, and 0.05 mutation rate. This figure illustrates that HGA_S did not converge in the 50,000 mean function evaluations but the other three methods did converge. In the left plot of Figure 3, the $BOHGA_S$ procedure converged the fastest to the global "valley," followed closely by the $BOHGA_S$ and the HGA_{TD} methods. The right plot in Figure 3, in an expanded scale, reveals that the $BOHGA_S$ procedure is actually the first to converge to the cutoff of 0.05 at about 38,000 mean cumulative function evaluations, followed by the HGA_{TD} and the $BOHGA_{TD}$ methods. It is clear that these three methods have very similar behavior for this function.

6.2. Comparisons for the Schwefel's Function in 20 Dimensions. A generalized Schwefel function from Schwefel [28] is given by

$$\sum_{i=1}^{k} -x_i \sin\left(\sqrt{|x_i|}\right), \quad \text{where } -500 \le x_i \le 500, \tag{2}$$

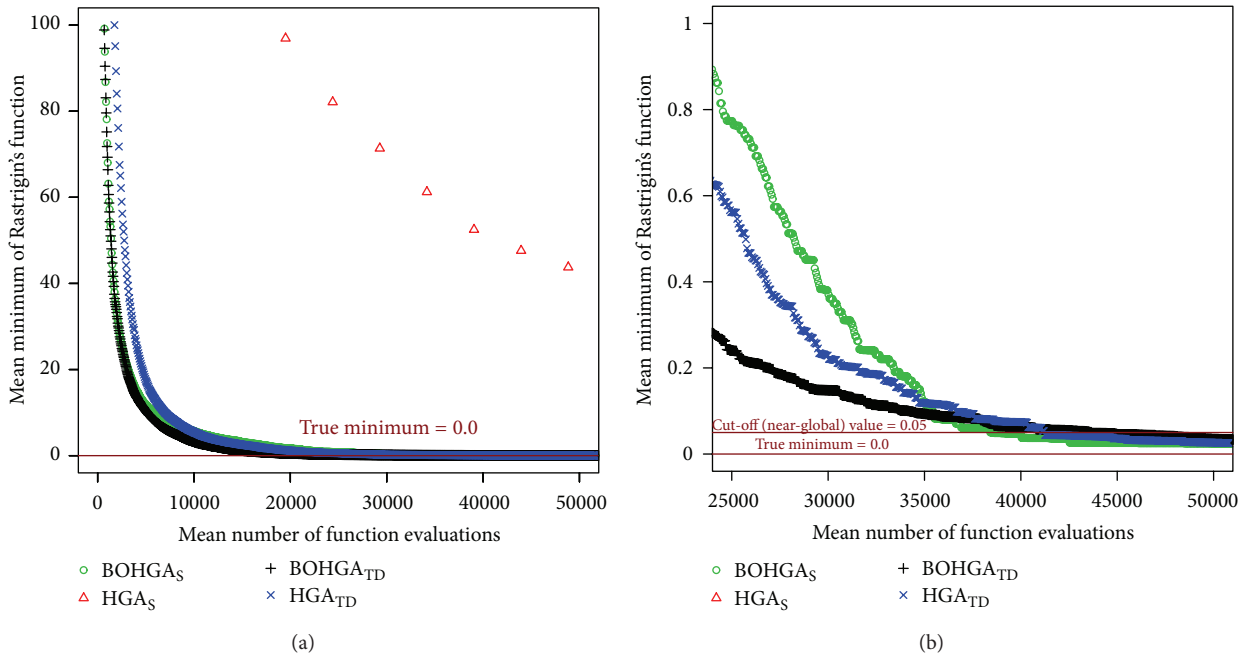


Figure 3: Best minimums of Rastrigin's function in 20 dimensions versus number of function evaluations at each generation averaged over 100 replications of the four MA methods under the GA setting (ranking replacement, 4 crossover points, and 0.05 mutation rate). (a) Overall view in a full scale; (b) highlighted view in an expanded scale.

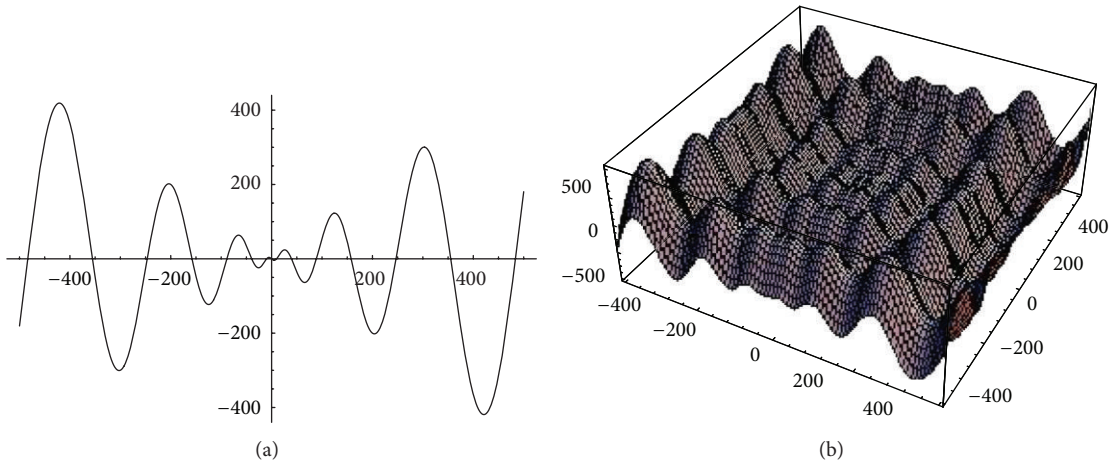


Figure 4: Surface of Schwefel's function: (a) 1-dimension; (b) 2-dimension.

where k is the number of dimensions of the function. The minimum of the objective function is given by $\min(f(\mathbf{x})) = f(420.9687, \ldots, 420.9687)$. The minimum is dependent on k, the number of dimensions. When $k = 20$, the minimum value is −8,379.66. Figure 4 shows the 1- and 2-dimensional surfaces for the Schwefel function. In the left plot of the figure, a solution must be located in the deepest valley, when value of the objective function is less than about −300.0 in the 1-dimensional case.

Although the Schwefel function has a nonlinear bumpy surface, its surface is relatively smooth in a range (−500, 500) when compared to the surface of the Rastrigin's function. The step length for the TD meme is set to 0.5, the same as for the simplex meme. The preselected number of generations used by stopping rule 1 is 1,000. The cut-off near-global value is set to −8,379.0.

Similar to Table 1, under stopping rule 1, Table 2 presents the mean total number of function evaluations as a summary of the 100 repetitions for the Schwefel's function for comparison of the four algorithms. Table 2 shows that the numbers of evaluations required to obtain a value of the objective function smaller than −8,379.0 by BOHGA$_S$, BOHGA$_{TD}$, and HGA$_{TD}$ are all consistently much less than required by HGA$_S$ overall settings. BOHGA$_S$ consistently performs the best with much smaller mean numbers of function evaluations than the HGA$_{TD}$ and BOHGA$_{TD}$, which

Table 2: Comparisons of $BOHGA_S$, HGA_S, $BOHGA_{TD}$, and HGA_{TD} in terms of mean of the number of evaluations under the eight settings of GA operators for the Schwefel's function in 20 dimensions by stopping rule 1.

8 settings of GA operators			Mean (evaluation)			
Replacement	Crossover	Mutation	$BOHGA_S$	HGA_S	$BOHGA_{TD}$	HGA_{TD}
Ranking	4	0.05	**13595**	471668	26792	31243
		0.06	**15049**	471251	26972	31590
	8	0.05	**13230**	518101	20588	28070
		0.06	**13792**	546366	20207	29972
Tournament	4	0.05	**28631**	1059412	47893	132294
		0.06	**37991**	1404750	94763	281186
	8	0.05	**37792**	1408634	74805	221730
		0.06	**57270**	1824465	214563	815384
Overall average			**27169**	963081	65823	196434

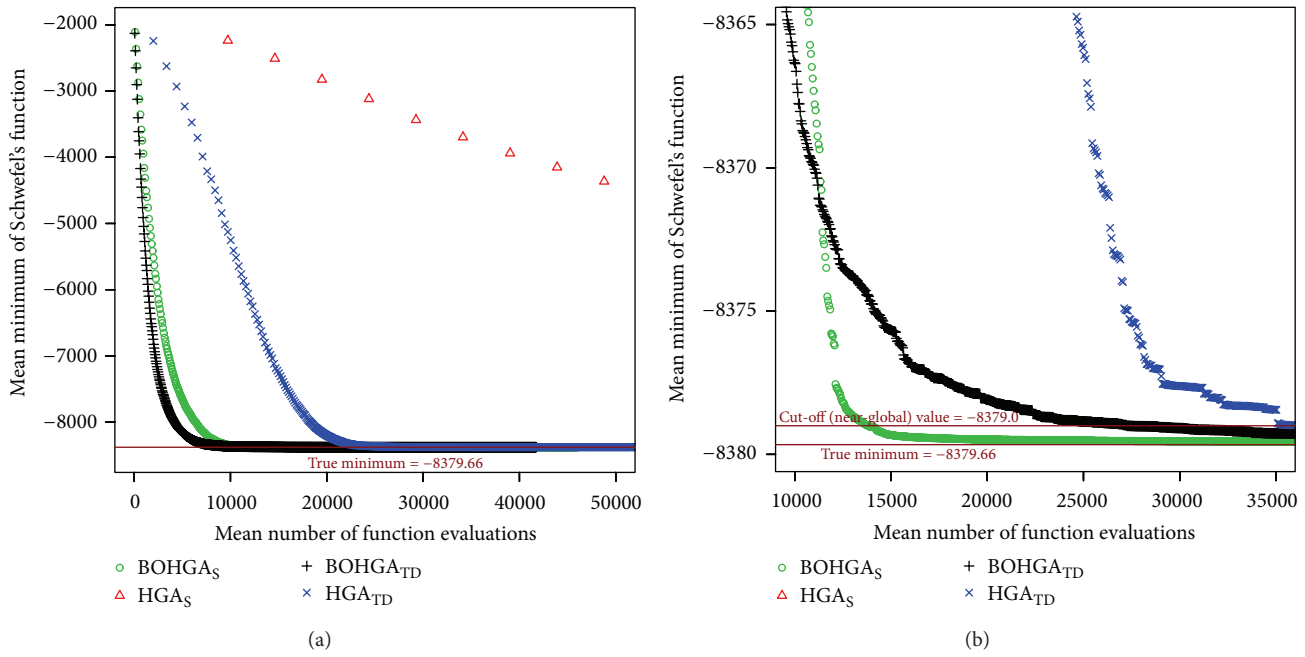


Figure 5: Best minimums of Schwefel's function in 20 dimensions versus number of function evaluations at each generation averaged over 100 replications of the four MA methods under the GA setting (ranking replacement, four crossover points, and 0.05 mutation rate). (a) Overall view in a full scale; (b) highlighted view in an expanded scale.

are quite competitive to each other. Overall the GA settings, $BOHGA_S$ has the smallest mean numbers of function evaluations, followed by $BOHGA_{TD}$, then HGA_{TD}, and finally HGA_S. In addition, the GA setting with ranking replacement performs much better than with tournament replacement in all of the four methods. This again indicates that tournament replacement in MAs is not as efficient as ranking replacement. The mutation rate of 0.05 performs better than the rate of 0.06 in most cases.

Similar to Figure 3, under stopping rule 2 with 1,000 generations, Figure 5 shows the mean best minimum of the Schwefel's function versus mean cumulative number of function evaluations at each generation over 100 replications obtained by each MA in the GA setting with the ranking replacement, four crossover points, and 0.05 mutation rate. The left plot of Figure 5 shows that HGA_S is the slowest to converge while $BOHGA_{TD}$ and $BOHGA_S$ have converged to a global "valley" at a similar yet faster rate than HGA_{TD}. The right plot of Figure 5, in an expanded scale, shows in detail that $BOHGA_S$ is the fastest to converge to the cutoff of −8379.0 at about 1,400 mean cumulative function evaluations, followed by the $BOHGA_{TD}$, followed by the HGA_{TD}.

7. Conclusion and Discussion

The importance of memetic algorithms in both real-world applications and academic research has lead to the establishment of the series of international Workshops On Memetic Algorithms (WOMA) and a dedicated book [13]. From these

workshops, the following important questions are raised: (1) when to apply local improvement heuristics, (2) to which individuals in the evolutionary algorithms population should local searches be applied, and (3) how much computational efforts to devote to local search algorithms. These questions remain unanswered, and more research effort is required to gain the understanding and insights that may lead to guidelines for the design of efficient and effective algorithms [13].

This paper presents an improved and simplified MA, BOHGA, with a novel individual learning procedure on when to perform a local search (or individual learning). Unlike a classical MA/HGA procedure, where a local search is performed on each offspring (solution), our new MA performs a local search when the best offspring is also found to be the best among the current parent population. This new learning procedure does not require any extra parameters.

We also develop a new meme, a three-directional local search, TD, which is derivative-free and self-adaptive. The main idea of TD is that three potential directions are constructed from parents to their offspring with a certain step length, when the offspring performs better than both of its parents.

The four well-known benchmark functions with very different experimental ranges are used to compare BOHGA_{S}, HGA_{S}, BOHGA_{TD}, and HGA_{TD}. The results under stopping rule 1 (that an algorithm is halted when a near-global-optimum cutoff is achieved) match the results under stopping rule 2 (that an algorithm is halted at a preselected number of generations) for both functions. These results indicate that BOHGA with the new individual learning procedure works much more efficiently than the traditional HGA, whichever meme is chosen. HGA_{TD}, where the LS is performed only on those offspring that have an objective function values superior to both of their parents, is quite competitive to BOHGA. These results also indicate that the TD meme is likely to help algorithms converge faster to a global "valley" but does not appear to converge as quickly during the final fine tuning stage as the simplex meme. The results from the Rosenbrock's and the Griewank's functions (which are not presented here) are similar to those from the Rastrigin's and the Schwefel's functions.

During the comparisons of the four MAs, we used eight different settings of GA operators and found that ranking replacement performs uniformly better than tournament replacement for both functions. The mutation rate of 0.05 (which is $1/k$, $k = 20$ in both of the benchmark functions) performed better than the rate of 0.06 in most cases. The different number of crossover points had no obvious effect on the number of function evaluations.

In summary, our new HGA with an individual learning procedure performs a LS only when the best offspring is also the best within the parent population. Our new HGA not only reduces the number of function evaluations required by the LS, but also improves accuracy and efficiency in finding an optimal solution. The TD meme is a good choice in helping finding a global "valley" or "peak" but may not perform as well as the Nelder-Mead method at the final fine tuning. It is noted that our HGA has combined our new meme with a GA. We speculate that our new procedure would also be effective when combined with other evolutionary algorithms.

Several issues remain for further study. For example, the three derivative-free directions defined in the TD meme may not be optimal. Another issue concerns the appropriate step length, once the directions are chosen. The size of a step length, arbitrarily chosen by us, may affect the efficiency of the MAs. We found that the TD may converge faster to a global "valley" or "peak" than the simplex meme but may be not as fast at finding an optimum at the fine tuning stage. In a future study, we may combine the TD and simplex memes together, using TD first to reach the global "valley" or "peak," followed by the simplex meme to fine tune the solution. A further issue involves the optimal settings of the GA operators. In this study, the three main GA operators: the type of replacement, the number of crossover points, and the mutation rate, have been studied. However, there may be some other operators affecting the GA performance, such as the population size and the parent/offspring ratio. We plan to study these issues in future work.

C++ code is available upon request from the authors.

Appendix

Mathematical Representation of the Three-Direction: A Local Search

We first introduce our notation. Parent 1 ($P1$) is given by $\mathbf{x}_{P1} = [x_{P11}, \ldots, x_{P1k}]'$, where $\mathbf{x}$ is a vector of size $k \times 1$ where k is the number of factors or the number of dimensions. Similarly, Parent 2 ($P2$) is given by $\mathbf{x}_{P2} = [x_{P21}, \ldots, x_{P2k}]'$, and their offspring ($O$) is expressed as $\mathbf{x}_{\text{O}} = [x_{O1}, \ldots, x_{Ok}]'$. Parent 1 direction (from $P1$ to O) is expressed as δ_{P1O} and Parent 2 direction (from $P2$ to O) is as δ_{P2O}. And the common direction is simply denoted as δ. The new points after the first step along the three directions are expressed as $\mathbf{x}_{\text{New1}} = [x_{\text{New11}}, \ldots, x_{\text{New1}k}]'$, $\mathbf{x}_{\text{New2}} = [x_{\text{New21}}, \ldots, x_{\text{New2}k}]'$, and $\mathbf{x}_{\text{New}} = [x_{\text{New1}}, \ldots, x_{\text{New}k}]'$, corresponding to Parent 1, Parent 2, and their common direction, respectively. The appropriate moving distance on each axis in each moving step is expressed as d.

Parent 1 direction, which essentially is the different distances on each dimension between points $P1$ and O, is expressed as

$$\delta_{P1O} = \mathbf{x}_{\text{O}} - \mathbf{x}_{P1} = [\delta_{11}, \delta_{12}, \ldots, \delta_{1k}]'. \tag{A.1}$$

Similarly, the Parent 2 direction is expressed as

$$\delta_{P2O} = \mathbf{x}_{\text{O}} - \mathbf{x}_{P2} = [\delta_{21}, \delta_{22}, \ldots, \delta_{2k}]'. \tag{A.2}$$

To keep the same directions and move along the three paths, the moving distance on each axis should be in constant proportion to each other, as the method of steepest ascent/descent in response surface methodology (RSM). (In RSM, the constant proportion on the ith dimension is defined as $\widehat{\beta}_i/\widehat{\beta}^{\times}$, where the $\widehat{\beta}_i$ is the ith estimated coefficient in the estimated first-order model and the $\widehat{\beta}^{*}$ is the largest coefficient in magnitude among the k estimated coefficients,

that is, $\widehat{\beta}^* = \max_{i=1,\ldots,k}|\widehat{\beta}_i|$.) From this ratio, we can see that the proportion only depends on the β_i, the ith coefficient. The moving distance on the ith dimension is defined as $(\widehat{\beta}_i/\widehat{\beta}^*) * \rho$, where the ρ is an appropriate fixed distance. (For more details, please see Myers and Montgomery [29, Pages 205–207]).

In our GA application, the main idea in moving along the Parent 1 path is the same as that in the method of steepest ascent/descent; that is, to keep the constant proportion in each dimension and move some appropriate fixed distance (which is d in our case) along Parent 1 path. But the difference between our GA case and RSM is the starting point. In the GA case, the starting points are $P1$ and $P2$, not O; that is, the first step has already been completed. So the next moving step starts at O. The largest moving distance in the first step is also not d, but $\max_{i=1,\ldots,k}|\delta_{1i}|$, where the δ_{1i} is the moving distance on ith axis in (A.1). Let δ_1^* denote $\max_{i=1,\ldots,k}|\delta_{1i}|$. In our study, if $\delta_1^* < d$, then the moving distant in the next step will be δ_1^*. Otherwise, the distance in the next step will be d. The distance d is obviously utilized to control the next moving distance.

The procedure of moving along the Parent 1 direction is as follows.

(1) Calculate δ_{P1O} and then find $\delta_1^* = \max_{i=1,\ldots,k}|\delta_{1i}|$, the largest distance in the first moving step.

(2) If $\delta_1^* < d$, then the next new position on the ith axis, $i = 1, \ldots, k$, is defined as $x_{\text{New1}i} = x_{Oi} + (\delta_{1i}/\delta_1^*) \times d$. Otherwise, the new position is $x_{\text{New1}i} = x_{Oi} + \delta_{1i}$.

(3) Check the region of the new point $\mathbf{x}_{\text{New1}} = [x_{\text{New11}}, \ldots, x_{\text{New1}k}]'$. If $x_{\text{New1}i}$ is greater than its upper bound (which is the largest value in the ith domain), then let it be the upper bound. Similarly, if it is less than its lower bound (which is the lowest value in the ith domain), then let it be the lower bound. (Usually, the upper bounds and lower bounds have been given through defining the objective function.)

(4) Evaluate the new point $\mathbf{x}_{\text{New1}}$ by the objective function. If the new point performs worse than the point $\mathbf{x}_O$, then the process of moving along the Parent 1 direction is halted. If the new point performs better than the $\mathbf{x}_O$, then replace the point $\mathbf{x}_{\text{New1}}$ by the next new point $\mathbf{x}_{\text{New1}} + \mathbf{\Delta}_{N1O}$, where $\mathbf{\Delta}_{N1O} = \mathbf{x}_{\text{New1}} - \mathbf{x}_O$. (The "N1O" means "New point from Parent 1" to "Offspring.") Then return to Step 3.

The procedure for moving along the Parent 2 direction is the same as that for the Parent 1 direction. However, the procedure for the common direction is slightly different from them, due to the different starting points. The starting points from the parents directions are $P1$ or $P2$, while the starting point in the common direction is O.

As mentioned earlier, building the common direction depends on whether both parent directions are consistent or not. If they are consistent on ith axis (either both positive or both negative), then move the same direction on the ith axis as the parent directions. Otherwise, stay on that axis without any movement, due to inconsistent directions. There is a special case: one of the moving distances on an axis in the parent directions is zero and the other is nonzero. In this case, we recommend movement in the same direction with the parent direction with nonzero moving distance on the axis.

The procedure for movement along the common direction is as follows.

(1) Calculate δ_{P1O} and δ_{P2O} as (A.1) and (A.2).

(2) The next new point is defined as $\mathbf{x}_{\text{New}} = [x_{\text{New1}}, \ldots, x_{\text{New}k}]'$ along the path from the common direction. To establish the common direction, three situations on each axis/dimension are possible: (a) the $\delta_{1i} \times \delta_{2i} > 0$ which means that there is a common direction on the ith axis; (b) The $\delta_{1i} \times \delta_{2i} < 0$ which means that there is not a common direction on the ith axis; and (c) the $\delta_{1i} \times \delta_{2i} = 0$ which means that at least one of δ_{1i} and δ_{2i} equals zero.

(2.1) If the situation is (a), then the new point position on the ith axis is given by $x_{\text{New}i} = x_{Oi} + \min(|\delta_{1i}|, |\delta_{2i}|, d)$ if both δ_{1i} and δ_{2i} are positive, or $x_{\text{New}i} = x_{Oi} - \min(|\delta_{1i}|, |\delta_{2i}|, d)$ if both δ_{1i} and δ_{2i} are negative.

(2.2) If the situation is (b), the new point position on the ith axis is given by $x_{\text{New}i} = x_{Oi}$ (no movement on the ith axis in this situation).

(2.3) If the situation is (c), there are three subcases: (1) $\delta_{1i} = 0$ and $\delta_{2i} \neq 0$; (2) $\delta_{1i} \neq 0$ and $\delta_{2i} = 0$; and (3) $\delta_{1i} = 0$ and $\delta_{2i} = 0$.

(2.3.1) For case (1), if $|\delta_{2i}| \geq d$, then $x_{\text{New}i} = x_{Oi} + d$ (when $\delta_{2i} > 0$) or $x_{\text{New}i} = x_{Oi} - d$ (when $\delta_{2i} < 0$). Otherwise, $x_{\text{New}i} = x_{Oi} + \delta_{2i}$.

(2.3.2) For case (2), similar to case (1), if $|\delta_{1i}| \geq d$, then $x_{\text{New}i} = x_{Oi} \pm d$. Otherwise $x_{\text{New}i} = x_{Oi} + \delta_{1i}$.

(2.3.3) For case (3), $x_{\text{New}i} = x_{Oi}$.

(3) Check the range of the new point $\mathbf{x}_{\text{New}}$.

(4) Evaluate the point $\mathbf{x}_{\text{New}}$. If the new point performs worse than the point $\mathbf{x}_O$, then the process for moving along the common direction is stopped. If the new point is better than $\mathbf{x}_O$, then replace the point $\mathbf{x}_{\text{New}}$ by the next new point $\mathbf{x}_{\text{New}} + \mathbf{\Delta}_{\text{NCO}}$, where $\mathbf{\Delta}_{\text{NCO}} = \mathbf{x}_{\text{New}} - \mathbf{x}_O$. (The "NCO" means "New from Common directions" and "Offspring"). Return to Step 3.

Acknowledgments

The authors wish to thank the anonymous referees and the editors for their constructive comments on an earlier draft of this paper.

References

[1] L. Davis, *Handbook of Genetic Algorithms*, Van Nostrand Reinhold, New York, NY, USA, 1991.

[2] D. E. Goldberg and S. Voessner, "Optimizing global-local search hybrids," in *Proceedings of the 1st International Conference of*

Genetic and Evolutionary Computation (GECCO '99), pp. 220–228, 1999.

[3] M. Lozano, F. Herrera, N. Krasnogor, and D. Molina, "Real-coded memetic algorithms with crossover hill-climbing," *Evolutionary Computation*, vol. 12, no. 3, pp. 273–302, 2004.

[4] Z. Michalwicz, *Genetic Algorithms + Data Structure = Evolution Programs*, AI, Springer, New York, NY, USA, 3rd edition, 1996.

[5] P. A. Moscato, "On evolution, search, optimization, genetic algorithms and martial arts: towards memetic algorithms," Tech. Rep. 826, Caltech Concurrent Computation Program, 1989.

[6] P. A. Moscato, "Memetic algorithms: a short introduction," in *New Ideas in Optimization*, D. Corne, M. Dorigo, and F. Glower, Eds., p. 219234, McGraw-Hill, London, UK, 1999.

[7] Y.-S. Ong, N. Krasnogor, and H. Ishibuchi, "Special issue on memetic algorithms," *IEEE Transactions on Systems, Man, and Cybernetics B*, vol. 37, no. 1, pp. 2–5, 2007.

[8] X. Chen, Y.-S. Ong, M.-H. Lim, and K. C. Tan, "A multi-facet survey on memetic computation," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 5, pp. 591–607, 2011.

[9] R. Dawkins, *The Selfish Gene*, Oxford University Press, New York, NY, USA, 1990.

[10] D. Sudholt, "Local search in evolutionary algorithms: the impact of the local search frequency," in *Algorithms and computation*, vol. 4288 of *Lecture Notes in Computer Science*, pp. 359–368, Springer, Berlin, Germany, 2006.

[11] N. Krasnogor and J. E. Smith, "Emergence of profitable search strategies based on a simple inheritance mechanism," in *Proceedings of the International Conference on Genetic and Evolutionary Computation*, pp. 432–439, Morgan Kaufmann, San Mateo, Calif, USA, 2001.

[12] M. Lozano, F. Herrera, and J. R. Cano, "Replacement strategies to maintain useful diversity insteady-state genetic algorithms," *Information Sciences*, vol. 178, pp. 4421–4433, 2008.

[13] W. E. Hart, N. Krasnogor, and J. E. Smith, "Editorial introduction special issue on memetic algorithms," *Evolutionary Computation*, vol. 12, no. 3, 2004.

[14] T. A. El-Mihoub, A. A. Hopgood, L. Nolle, and A. Battersby, "Hybrid genetic algorithms: a review," *Engineering Letters*, vol. 13, pp. 2–11, 2006.

[15] G. Seront and H. Bersini, "A new GA local search hybrid for continuous optimization based on multi-level single linkage clustering," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '00)*, pp. 90–95, Morgan Kaufmann, Las Vegas, Nev, USA, 2000.

[16] M. Lozano and C. García-Martínez, "Hybrid metaheuristics with evolutionary algorithms specializing in intensification and diversification: overview and progress report," *Computers & Operations Research*, vol. 37, no. 3, pp. 481–497, 2010.

[17] K. Tang, Y. Mei, and X. Yao, "Memetic algorithm with extended neighborhood search for capacitated arc routing problems," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 1151–1166, 2009.

[18] D. Molina, M. Lozano, and F. Herrera, "Memetic algorithm with local search chaining for continuous optimization problems: a scalability test," in *Proceedings of the 9th International Conference on Intelligent Systems Design and Applications (ISDA '09)*, pp. 1068–1073, IEEE Computer Society, December 2009.

[19] D. Molina, M. Lozano, and F. Herrera, "MA-SW-Chains: memetic algorithm based on local search chains for large scale continuous global optimization," in *Proceedings of the 6th IEEE World Congress on Computational Intelligence (WCCI '10)*, July 2010.

[20] Z. Ning, Y. S. Ong, K. W. Wong, and M. H. Lim, "Choice of memes in memetic algorithm," in *Proceedings of the 2nd International Conference on Computational Intelligence, Robotics and Autonomous Systems (CIRAS '03)*, 2003.

[21] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, Addison-Wesley, Reading, Mass, USA, 1989.

[22] M. Hamada, H. F. Martz, C. S. Reese, and A. G. Wilson, "Finding near-optimal Bayesian experimental designs via genetic algorithms," *The American Statistician*, vol. 55, no. 3, pp. 175–181, 2001.

[23] D. G. Mayer, J. A. Belward, and K. Burrage, "Robust parameter settings of evolutionary algorithms for the optimisation of agricultural systems models," *Agricultural Systems*, vol. 69, no. 3, pp. 199–213, 2001.

[24] F. Ortiz Jr., J. R. Simpson, J. J. Pignatiello Jr., and A. Heredia-Langner, "A genetic algorithm approach to multiple-response optimization," *Journal of Quality Technology*, vol. 36, no. 4, pp. 432–450, 2004.

[25] R. L. Haupt and S. E. Haupt, *Practical Genetic Algorithms*, Wiley-Interscience [John Wiley & Sons], Hoboken, NJ, Second edition, 2004, With 1 CD-ROM (Windows).

[26] J. A. Nelder and R. Mead, "A simplex method for function minimization," *Computer Journal*, vol. 7, pp. 308–313, 1965.

[27] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in C*, Cambridge University Press, Cambridge, UK, 2nd edition, 1992.

[28] H. P. Schwefel, *Evolution and Optimum Seeking*, John Wiley & Sons , New York, NY, USA, 1995.

[29] R. H. Myers and D. C. Montgomery, *Response Surface Methodology: Process and Product Optimization Using Designed Experiments*, John Wiley & Sons, 2002.