Open Mathematics

Research Article

Shahram Golzari*, Mohammad Nourmohammadi Zardehsavar, Amin Mousavi, Mahmoud Reza Saybani, Abdullah Khalili, and Shahaboddin Shamshirband*

KGSA: A Gravitational Search Algorithm for Multimodal Optimization based on K-Means Niching Technique and a Novel Elitism Strategy

https://doi.org/10.1515/math-2018-0132
Received February 14, 2018; accepted August 2, 2018

Abstract: Gravitational Search Algorithm (GSA) is a metaheuristic for solving unimodal problems. In this paper, a K-means based GSA (KGSA) for multimodal optimization is proposed. This algorithm incorporates K-means and a new elitism strategy called "loop in loop" into the GSA. First in KGSA, the members of the initial population are clustered by K-means. Afterwards, new population is created and divided in different niches (or clusters) to expand the search space. The "loop in loop" technique guides the members of each niche to the optimum direction according to their clusters. This means that lighter members move faster towards the optimum direction of each cluster than the heavier members. For evaluations, KGSA is benchmarked on well-known functions and is compared with some of the state-of-the-art algorithms. Experiments show that KGSA provides better results than the other algorithms in finding local and global optima of constrained and unconstrained multimodal functions.

Keywords: Gravitational Search Algorithm (GSA), multimodal optimization, K-means, niching methods

1 Introduction

In addition to the need for finding several optima in many applications, solving multimodal problems can be useful at least for two reasons; first, it can increase the chance of finding the global optimum and second, it can help the researcher to become more familiar with the nature of the problem [1]. Population-based (or

***Corresponding Author: Shahram Golzari:** Department of Electrical and Computer Engineering, University of Hormozgan, Bandar Abbas, Iran, E-mail: golzari@hormozgan.ac.ir
Mohammad Nourmohammadi Zardehsavar: Department of Electrical and Computer Engineering, University of Hormozgan, Bandar Abbas, Iran, E-mail: nourmohammadi.z.mohammad@gmail.com
Amin Mousavi: Department of Electrical and Computer Engineering, University of Hormozgan, Bandar Abbas, Iran, E-mail: mousavi@hormozgan.ac.ir
Mahmoud Reza Saybani: Department of Computer Networks, Markaz-e Elmi Karbordi Bandar Abbas 1, University of Applied Science and Technology, 79199-33153 Bandar Abbas, Iran, E-mail: saybani@gmail.com
Abdullah Khalili: Department of Electrical and Computer Engineering, University of Hormozgan, Bandar Abbas, Iran, E-mail: khalili@hormozgan.ac.ir
***Corresponding Author: Shahaboddin Shamshirband:** Department for Management of Science and Technology Development, Ton Duc Thang University, Ho Chi Minh City, Vietnam
Faculty of Information Technology, Ton Duc Thang University, Ho Chi Minh City, Vietnam,
E-mail: shahaboddin.shamshirband@tdtu.edu.vn

meta-heuristic) algorithms have been used to solve optimization problems. Some of these algorithms are: Genetic Algorithm (GA) by [2, 3], Simulated Annealing (SA) by [4], Artificial Immune Systems (AIS) by [5], Ant Colony Optimization (ACO) by [6], Particle Swarm Optimization (PSO) by [7], and Gravitational Search Algorithm (GSA) by [8, 9]. Generally speaking, these algorithms are inspired by nature and are effective in solving unimodal optimization problems. However, they have not been very successful in solving multimodal problems. To resolve this, two solutions have been provided: 1- Niching techniques to converge to more than one solution by dividing the main population into non-overlapping areas and 2- Elitism strategy to accelerate the convergence by selecting the best individuals from the current population and its offspring.

In this study, the Gravitational Search Algorithm (GSA) is boosted with the K-means niching and a new elitism strategy called "loop in loop" to make an efficient algorithm (called KGSA) in solving multimodal problems. GSA was selected since it is less dependent on parameters and can find existing optima with less iterations without trapping in local minimum. In addition, K-means clustering technique was chosen for its simplicity, effectiveness and, low time complexity in dividing the main population into non-overlapping subpopulations. Results show that by incorporating K-means and "loop in loop" into the GSA, the proposed algorithm has increased both 'exploration' and 'exploitation'.

This paper is structured as follows: in section 2, some recent works regarding solving multimodal problems are studied. Afterwards, in section 3, GSA, niching concepts and K-means clustering techniques are described. Section 4, describes how the proposed algorithm is designed and how is "loop in loop" used. In section 5, after introducing constrained and unconstrained benchmark functions, evaluation criteria and parameters required for the suggestive algorithm are presented and results of implementation of the suggestive algorithm on the benchmark functions are analyzed, and then the sensitivity of parameter T_l relevant to the suggestive algorithm on some functions is measured. At the end, in section 6, strengths and weaknesses of the proposed algorithm are analyzed and some future works are suggested.

2 Literature review

Solving multimodal problems has always been one of the important and interesting issues for computer science researchers. Authors in [10] presented NichePSO algorithm to solve multimodal problems and showed its efficiency by solving some multimodal functions. In this algorithm, Guaranteed Convergence Particle Swarm Optimization (GCPSO) and Faure-sequences [11] were used to optimize the sub-swarms and the population initialization, respectively. In addition, two parameters δ and μ were defined in this algorithm. If an individual's variance was less than δ or, in other words, if an individual did not change over several generations, it may be an optimum member. Therefore, the individual and its closest member build a sub-swarm. On the other hand, parameter μ was used to merge the sub-swarms. Results indicated that NichePSO is highly dependent on parameter μ to explore the optimal solutions and this is a major weakness.

Authors in [12] presented Clustering-Based Niching (CBN) method to find global and local optima. The main idea of this method for exploring optima and keeping variety of population was to use sub-populations instead of one population. Species were formed using sub-populations and sub-populations were separated by a density-based clustering algorithm which is appropriate for populations with different sizes and for problems in which the number of clusters is not predetermined. In order to attach two members during the clustering process in CBN, their distance should be less than parameter σ_{dist}. Results showed that this algorithm is highly dependent on σ_{dist} which is a major drawback.

Authors in [13] presented a method in which PSO and cleansing technique were used. In this method, population was divided into different species based on similarity of its members. Each species was then formed around a dominant individual (or the 'specie-seed'). In each phase, particles were selected from the whole population and the species were formed adaptively based on the feedback of fitness space. The method was named as Species based Particle Swarm Optimization (SPSO). Although SPSO was proven to be effective in solving multimodal problems with small dimensions, the dependency on species' radius is among its weaknesses.

Authors in [14] presented an algorithm for solving multimodal optimization problems. This new method was named Multi-Grouped Particle Swarm Optimization (MGPSO) in which, if the number of groups is N, PSO can search N peaks. The efficiency of this method was shown in [14]. Weaknesses of this algorithm are: 1) Determining the number of groups that is determining the number of optimal solutions by user while initializing the algorithm. It is also possible that the function is unknown and no information about the number of optimal points exists. 2) Determining the number of individuals for each group and selecting an appropriate initial value for the radius of each $gbest$.

Authors in [15] used a new algorithm called NGSA for solving multimodal problems. The main idea was that the initial swarm is divided into several sub-swarms. NGSA used following three strategies: i) an elitism strategy, ii) a K-NN strategy, and iii) amendment of active gravitational mass formulation [15]. This algorithm was applied on two important groups of constrained and unconstrained multimodal benchmark functions and obtained good results, but this algorithm suffers from high dependency on two parameters K_i and K_f.

Authors in [16] proposed Multimodal Cuckoo Search (MCS), a modified version of Cuckoo Search (CS) with multimodal capacities provided by the following three mechanisms: (1) incorporating a memory mechanism which efficiently registers potential local optima based on their fitness value and the distance to other potential solutions, (2) modifying of the original CS individual selection strategy for accelerating the detection process of new local minima, and (3) including a depuration procedure for cyclically elimination of duplicated memory elements. Experiments indicated that MCS provides competitive results compared to other algorithms for multimodal optimization.

Author of [17] modified the original PSO by dividing the original population into several subpopulations based on the order of particles. After this, the best particle in each subpopulation was employed in the velocity updating formula instead of the global best particle in PSO. Evaluations showed that after modifying the velocity updating formula, convergence behaviour of particles in terms of the number of iterations, and the local and global solutions was improved.

Authors in [18] combined exploration mechanism of the Gravitational search algorithm with the exploitation mechanism of Cuckoo search and called their method Cuckoo Search-Gravitational Search Algorithm (CS-GSA). Evaluations on standard test functions showed that CS-GSA converges with less number of fitness evaluations than both Cuckoo Search and GSA algorithms.

Authors in [19] proposed a multimodal optimization method based on firefly algorithm. In their method, the optimal points are detected by evolving each sub-population separately. For determining the stability of sub-populations, a stability criterion is used. Based on the criterion, stable sub-populations are found and since they have optimal points, they are stored in the archive. After several iterations, all the optimums are included in the archive. This algorithm also incorporates a simulated annealing local optimization algorithm to enhance search power, accuracy and speed. Experiments show that the proposed algorithm can successfully find optimums in multimodal optimization problems.

Authors in [20] proposed a niching method for Chaos Optimization Algorithm (COA) called NCOA. Their method utilizes a number of techniques including simultaneously contracted multiple search scopes, deterministic crowding, and clearing for niching. Experiments demonstrated that by using niching, NCOA can compete the state-of-the-art multimodal optimization algorithms.

Authors in [21] presented a novel evolutionary algorithm called Negatively Correlated Search (NCS) which parallels multiple individual search and models the behaviour of each individual search as a probability distribution. Experiments showed that NCS provides competitive results to the state-of-the-art multimodal optimization algorithms in the sense that NCS achieved the best overall performance on 20 non-convex benchmark functions.

Author of [22] presented a modified PSO which in the first step randomly divides the original population into two groups with one group focusing on the maximum optimization of the multimodal function and the other on minimization. After this, each group is divided into subgroups for finding optimum points simultaneously. The important point is that subgroups are not related and each one seeks for one optimum individually. Similar to [17], the velocity updating formula is modified in the proposed method by replacing the best particle of each subgroup instead of the global best. Evaluations on different kinds of multimodal

optimization functions and one complex engineering problem demonstrated the applicability of the proposed method.

Authors in [1] incorporated a novel niching method into PSO (named NNGSA) by using Nearest Neighbour (NN) mechanism for forming species inside the population. They also employed the hill valley algorithm without a pairwise comparison between any pair of solutions for detecting niches inside the population. Experiments showed the effectiveness of NNGSA compared to the well-known niching methods.

3 Basic concepts

3.1 Gravitational search algorithm

GSA was presented in [8] based on Newton's law of gravitation. Agents in this algorithm are similar to particles in the universe. The heavier the mass of an agent, the more efficient is that agent. This means that agents with heavier mass have higher attractions and walk more slowly (Figure 1). The location of an agent i in GSA is shown by Equation (1). Considering a system of N agents, the whole system can be formulated as Equation (2).

$$X_i \leftarrow (x_i{}^1,\ x_i{}^2,\ \ldots,\ x_i{}^d, \ldots.., x_i{}^n) \tag{1}$$

$$X \leftarrow (X_1,\ X_2,\ \ldots,\ X_i, \ldots.., X_N) \tag{2}$$

In the above equations, x_i^d is the location of agent i in dimension d, n is the dimension of the search space and N is the number of individuals (or agents). At a given time t, the applied force on agent i by agent j is calculated using Equation (3).

$$F_{ij}^d \leftarrow G(t) \frac{M_{pi}(t) \times M_{aj}(t)}{R_{ij}(t) + \epsilon} (x_j^d(t) - x_i^d(t)) \tag{3}$$

where M_{aj} is the active gravitational mass of agent j, M_{pi} is the passive gravitational mass of agent i, $G(t)$ is gravitational constant at time t, ϵ is a small constant and R_{ij} is the Euclidian distance between the two agents determined by Equation (4).

$$R_{ij}(t) \leftarrow \left\| X_i(t), X_j(t) \right\| \tag{4}$$

Assuming that $Kbest$ is the set of K agents with the best fitness value and thus the biggest mass, the whole applied force to the agent i in dimension d from agents in $Kbest$ is computed by Equation (5).

$$F_i^d(t) \leftarrow \sum_{j \epsilon Kbest,\ j \neq i} rand_j F_{ij}^d \tag{5}$$

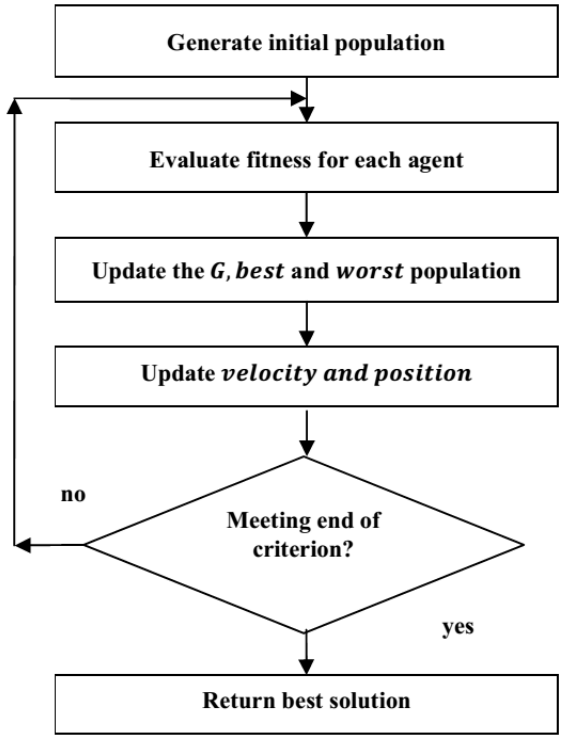
where $rand_j$ is a random number in [0,1]. It should be noted that $Kbest$ changes with time, its initial value is K_0 and as time passes it decreases. The acceleration of the agent i in dimension d at time t is calculated by Equation (6).

$$a_i^d(t) \leftarrow \frac{F_i^d(t)}{M_{ii}(t)} \tag{6}$$

where M_{ii} is the inertia mass of the agent i. In addition, the velocity and position of the agent i in dimension d at time $t+1$ are calculated by Equations (7) and (8) respectively. In equation (7), $rand_i$ is a random number in [0,1].

$$v_i^d(t+1) \leftarrow rand_i \times v_i^d(t) + a_i^d(t) \tag{7}$$

Figure 1: General principle of GSA [8]



$$x_i^d(t+1) \leftarrow x_i^d(t) + v_i^d(t) \tag{8}$$

Gravitational fixed G is a function of time that is started with the initial value G_0 and as time passes, it decreases to control the accuracy of search. The value of this function is calculated by Equations (9) and (10).

$$G(t) \leftarrow (G_0,\ t) \tag{9}$$

$$G(t) \leftarrow G_0 e^{\frac{-\alpha t}{T}} \tag{10}$$

where α and G_0 are fixed values and T indicates all iterations. In addition, inertia and gravitational masses are updated using Equations (11)–(13)

$$M_{ai} = M_{pi} = M_{ii} = M_i \qquad i = 1,\ 2,\ \ldots,\ N \tag{11}$$

$$m_i(t) \leftarrow \frac{fit_i(t) - worst(t)}{best(t) - worst(t)} \tag{12}$$

$$M_i(t) \leftarrow \frac{m_i(t)}{\sum_{j=1}^{N} m_j(t)} \tag{13}$$

where $fit_i(t)$ shows fitness value of the agent i at time t and $worst(t)$ and $best(t)$ are calculated using Equations (14) and (15).

$$best\ (t) \leftarrow \max_{j \in \{1,\ldots..,\ N\}} fit_j(t) \tag{14}$$

$$worst\ (t) \leftarrow \min_{j \in \{1,\ldots..,\ N\}} fit_j(t) \tag{15}$$

3.2 Niching

As mentioned previously, niching is the concept of dividing the search space into different areas or niches in a way that these areas are not overlapped. Evolutionary algorithm with niching technique search for optimal solutions in each separated area to efficiently explore the search space for finding global optimal solutions. Some of the well-known niching techniques are: fitness sharing [23], clearing [24], crowding [25], deterministic crowding [26–28] and probabilistic crowding [29].

3.3 K-means algorithm

K-means is employed as the niching technique in KGSA. The most important parameter in K-means is the number of clusters which is manually determined by the user. Clusters are represented by their centers which are randomly selected at the beginning of K-means. Then, K-means works as follows: each point (or agent) is assigned to the closest center, and in this way members of each cluster are determined. The mean of members in each cluster is calculated as the new center of that cluster (Equation 16). This process continues until the maximum number of iterations is reached or the members of clusters do not change.

$$c_k \leftarrow \frac{1}{|S_k|} \sum_{\forall X_i \in S_k} X_i \tag{16}$$

In the above equation, X_i is agent i which belongs to cluster S_k, c_k is the center of S_k and $|S_k|$ is the number of members (agents) in S_k. A pseudocode of K-means algorithm is shown in Figure 2. Simplicity, flexibility and being easy to understand are among the advantages of this algorithm, but determining the number of clusters at the beginning of the algorithm is a weakness. In addition, since initial centers are randomly selected, results are different in different runs.

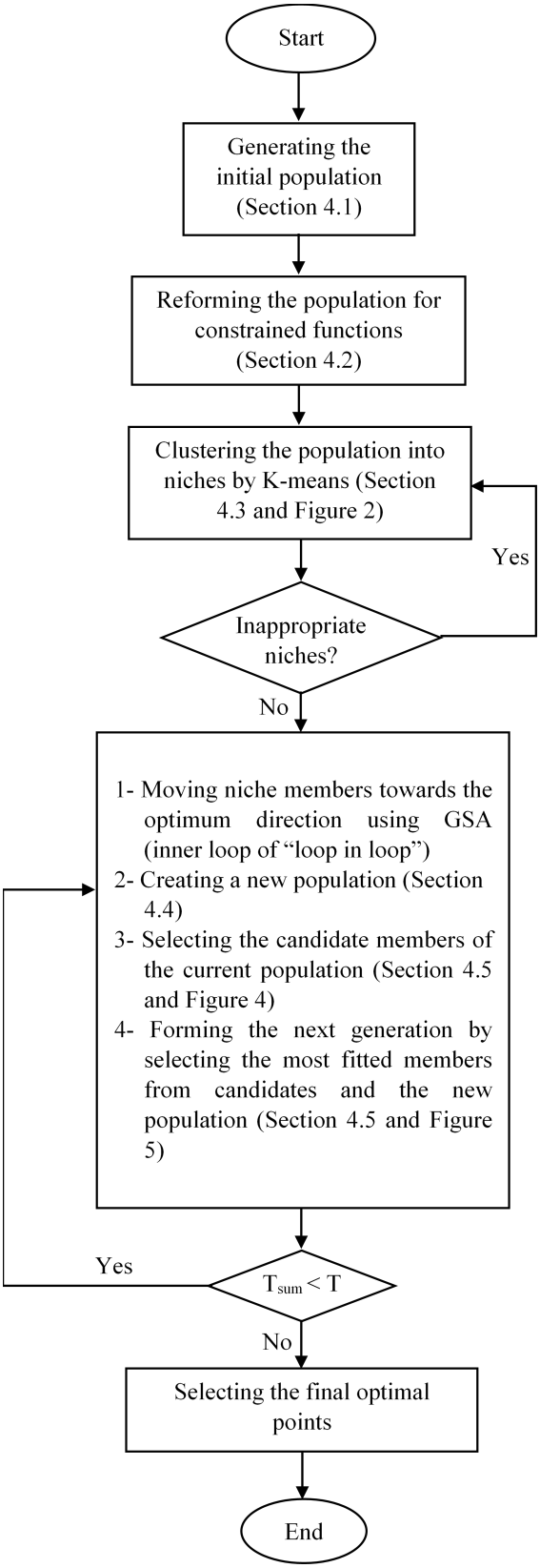
Figure 2: Pseudocode of K-means algorithm for clustering population members

```
Input:
    D = {d1, d2, ….. , dn} //set of n data items.
    k  //Number of desired clusters
Output:
    A set of k clusters.
Steps:
    1. Arbitrarily choose k data-items from D as initial centroids;
    2. Repeat
        Assign each item di to the cluster which has the closest centroid;
        Calculate convergence criteria is met.
      Until convergence criteria is met.
```

4 Proposed algorithm

GSA, K-means and a new elitism strategy called “loop in loop” are used to design the suggestive algorithm. This algorithm is called KGSA (K-means gravitational search algorithm) owing to the use of K-means and GSA. In this algorithm, firstly, the members of the initial population are clustered by K-means. Afterwards, the population is created and divided into different niches or clusters, the reinforced GSA with “loop in loop” technique guide the members of each niche to the optimum direction according to their clusters. More specifically, members of each cluster apply force to each other so that lighter members move towards the optimum direction of each cluster with higher velocity and heavier ones move to the direction slower. The principle of KGSA is shown in Figure 3. Different parts of the KGSA are separately described in order to explain more details.

Figure 3: General principle of KGSA



4.1 Population initialization

Before initializing the population, the structure of individuals must be specified. In this research, phenotypic structure is used where each member in each dimension gets a numerical value. To form the initial population, uniform and partition methods adopted from [15] are used. In the uniform method, members are initialized using Equation (17):

$$x_i^d = Low + rand * (High - Low) \tag{17}$$

where x_i^d is the location of agent (member) i in dimension d, Low and $High$ are respectively the lowest and highest possible values in dimension d, and $rand$ is a random number in [0,1]. Although the uniform method is simple, members initialized with this approach may be placed close to each other resulting in poor distribution in the search space.

In the partition method, the legal [Low, $High$] period in each dimension d is first divided into N smaller sub-periods with equal length where N is the size of population. Then for each dimension d, members are assigned to different sub-periods and similar to Equation (17), get a random value in that sub-period. With this approach, the main problem of the uniform method is solved and members are better distributed in the search space.

4.2 Population reformation

Population reformation is used for constrained functions so that infeasible solutions are avoided. It is clear that infeasible solutions may be produced when the initial population or new population are created. In the first case, the infeasible solutions are replaced by new solutions (members) created over and over again using uniform or partition methods. This process continues until possible solutions are found. In the latter case, the infeasible solutions are replaced by the most fitted members of the previous generation [30].

4.3 Production of appropriate clusters from population

Inappropriate clusters when initializing the population are either single-member or empty clusters. In other phases of KGSA, only empty clusters are considered inappropriate. Depending on the type of population initialization especially uniform method, some clusters may be inappropriate after executing the K-means algorithm, resulting in losing niches. To resolve this problem, when a cluster is inappropriate in population initialization, the population is rejected and the initial population is formed and clustered again until some appropriate clusters are created. But in other phases of KGSA, clustering is iterated until none of the clusters are inappropriate.

4.4 Calculation of mass, force, velocity and production of the next generation

Before calculating the mass of each member (agent), its neighbours should be determined. Since members have been clustered before, other members within the cluster of each member are considered as its neighbours. Then, using Equations (11)-(13), the mass of each member is calculated. After the mass of all members were specified, according to the Equation (5), the total force on agent i in dimension d is determined. Applied force to a member is only from its neighbours. When members apply force to each other, each member moves towards a direction with different velocity which can be calculated using Equation (7). As a result of the movement, new population is created. As will be discussed in the next section, this new population competes with the candidate members of the current population to form the next generation.

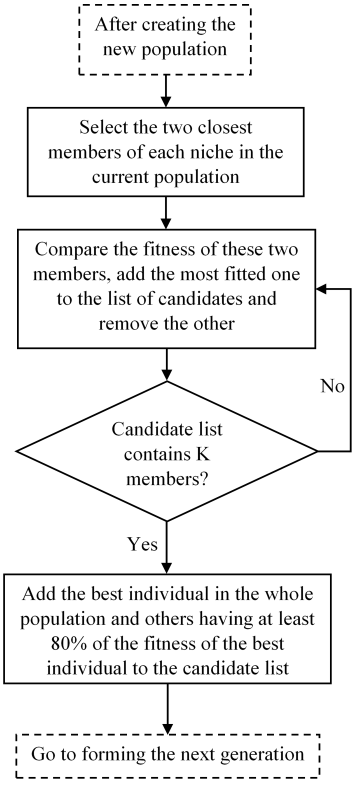
4.5 Use of innovative method "loop in loop" and discovery of optima

As described in the previous section, after the current population was developed by T_l iterations using GSA algorithm, the new population is created (section 4.4). Since this process needs T_l iterations, it is called the first (or inner) loop in the "loop in loop" method. Then, the most fitted members of the current population are selected as candidates (Section 4.5.1) and they compete with members of the new population (Section 4.5.2) to form the initial population for the next generation. This process continues until the maximum number of repetitions or the maximum number of permitted evaluations is reached. This is actually the second (or outer) loop of the "loop in loop".

4.5.1 Selecting the candidate members from the current population

Since after developing the current population via T_l iterations using GSA, members of the current population are closer to the optimal solutions, the two closest members of each cluster (niche) are found, the most fitted one is selected and kept and the other one is removed. This process continues until the number of selected members reaches the number of optimal solutions (niches). These selected members along with the most fitted members of the current population (i.e. the best individual in the whole population and those having at least 80% of the fitness of the best individual) are candidate members for the next generation. The described process is based on the work of [30] and illustrated in Figure 4.

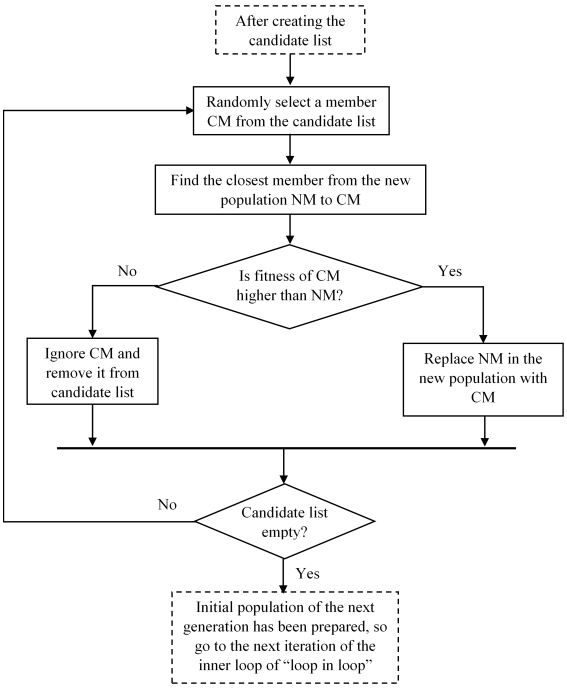
Figure 4: Preparing candidate list at the end of inner loop of "loop in loop"



4.5.2 Forming the next generation by selecting the best members of candidates and the new population

The candidate members of the current population compete with the new population in the following way: for each candidate, the closest member of the new population is found. If the candidate is more fitted, it replaces the closest member of the new population; otherwise, it is ignored (Figure 5). After all the candidates were checked, the initial population for the next generation has been formed, so the next iteration of the inner loop of the "loop in loop" is started. It is worth to mention that when going from one generation to the next, niches are also transferred to the next generation.

Figure 5: Forming the initial population of the next generation by selecting the most fitted members of the candidate list and the new population



4.6 Conditions of ending the "loop in loop" algorithm

As illustrated in Figure 3, the maximum number of generations in "loop in loop" algorithm is fixed number T, but for the K-means algorithm, stopping criteria is when, the center of clusters does not change anymore.

5 Experimental results

To evaluate KGSA, some standard constrained and unconstrained functions are used. These functions are extracted from [15]. Results of benchmarking KGSA on these functions are compared to a number of algorithms including r3PSO [31], r2PSO-lhc [31], r3PSO-lhc [31], SPSO [32], FER-PSO [33], NichePSO [10], deterministic crowding [27], sequential niche [34], NGSA [15], NCOA [20], Firefly [19], and NNGSA [1]. It should be noted that some benchmark functions of this study have not been used in NCOA [20], Firefly [19], and NNGSA [1] and thus, the corresponding cells in tables showing the results are empty.

At the end of section 5, KGSA is statistically compared with other algorithms using Friedman test to show whether this algorithm is statistically superior or not.

5.1 Constrained and unconstrained benchmark functions

Unlike the constrained functions, when the domain is not considered in unconstrained functions, solutions do not follow a specific pattern. Tables 1A and 2A in Appendix show, respectively, unconstrained and constrained benchmark functions used in this study. Some of these benchmark functions along with the position of their optimal solutions are presented in Figures 6-9.

Figure 6: Shekel's Foxholes function

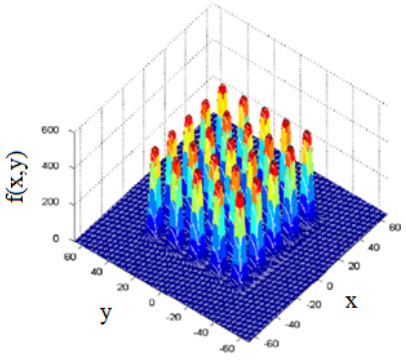
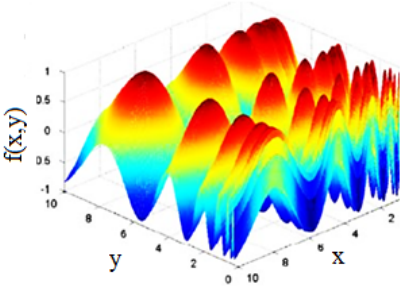


Figure 7: Inverted Vincent function



5.2 Evaluation Criteria of Algorithms

The KGSA algorithm is run multiple times and results of these independent runs are averaged. The evaluation criteria are success rate, error rate, and the number of fitness evaluations.

5.2.1 Success rate

In each run of the algorithm, if all peaks are found, the run is successful. When an agent reaches 99% of its highest value, it is considered as to have reached the peak value [15]. In addition, for a fair comparison, in

Figure 8: Inverted Shubert function

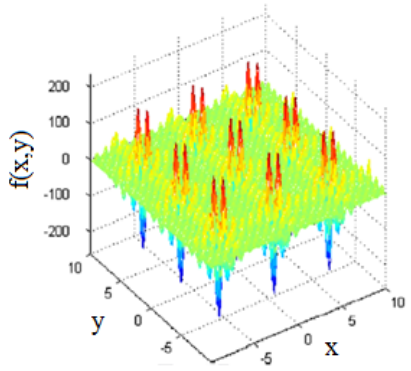
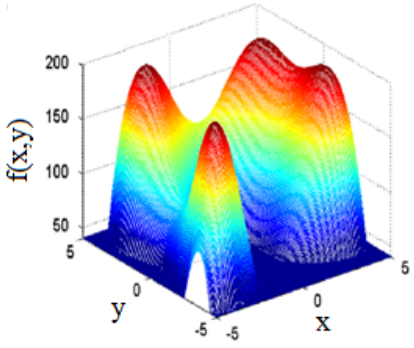


Figure 9: Himmelblau's function



cases that the error rate is reported, they are taken into account in KGSA. The success rate or the Average Discovering Rate (ADR) is calculated by averaging the results of different runs.

5.2.2 Error rate

Suppose that an algorithm finds a number of optimal solutions in an n dimensional space. The error rate is the mean of the Euclidian distance of the position of the discovered solutions from the position of the real optimum and is calculated using Equation (18).

$$\zeta \leftarrow \frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{n}\left[\left(s_i^j - \varphi_i^j\right)^2\right]^{\frac{1}{2}} \tag{18}$$

where m is the number of the optimal solutions found by the algorithm and n is the dimension of the problem. In addition, s indicates the position of the optimum found by the algorithm and φ is the position of the real optimum of the problem. Similar to the ADR, error rate for different runs are averaged.

5.2.3 Number of fitness evaluations

In each generation, when the algorithm finds the niches, number of times that the evaluation function has been called is reported as the number of fitness evaluation. In another words, this number is the number of times that fitness agent has been calculated from the beginning. Similar to success and error rates, this value is also averaged for different runs.

5.3 Parameters and results of other algorithms

The results and parameters of other algorithms including the number of population members or N, number of iteration and maximum number of evaluations can be obtained from [15].

In KGSA, K is the number of clusters and is considered to be the total number of optimal solutions. In cases that the number of optimal solutions is higher than 25, K is equal to the number of the optima that must be discovered. G_0 is set to 0.1 of the domain value and α is equal to 8. In both iterations, K-means algorithm is executed once. In each generation, 70% of the best individuals of each cluster apply force to co-cluster members. Other values from 20% to 95% (with step size 5%) were also tested for this. However, experiments showed that values lower than 70% result in virtual (incorrect) niches and values higher than 85% lead to losing some actual niches. Thus, the least appropriate value (70%) was selected since in addition to providing the desired results, it has less computational cost.

5.4 Finding optimal solutions

5.4.1 Finding global optimum in unconstrained functions

For finding global optima, KGSA was executed on functions F_1 to F_{12} with partitioning initialization. Success rate is reported in Table 1 and the number of fitness evaluations for discovering the global optima are reported in Tables 3A and 4A in Appendix. Table 3A in Appendix compares KGSA with the well-known swarm-based methods for multimodal optimization and Table 4A in Appendix presents the result of comparison with NNGSA and some of the other evolutionary approaches. It should be mentioned that results are averaged over 50 runs. In Table 2, the required parameters for algorithms NGSA and KGSA are shown. Unlike the other algorithms, when executing NGSA on functions F_6 to F_9, the number of the initial population was set to 100 instead of 50, but the maximum number of iterations was decreased such that the maximum number of all evaluations is not exceeded.

As illustrated in Table 1, the KGSA has performed much better than the well-known swarm-based algorithms in discovering all global optima for all functions. Table 3A in Appendix shows that the results of Table 1 were obtained with less number of evaluations in most cases. Furthermore, as can be seen in Table 2, the initial population size for KGSA was lower in all cases, except for function F_{12}.

5.4.2 Finding local and global optima in unconstrained functions

In order to find global and local optima, KGSA is developed with different population sizes in at most 120 generations. Results of this evaluation is compared with that of NGSA algorithm with two different initializations: 1) uniform initialization where the comparative results are presented in Table 3 and 2) partitioning initialization where the results are shown in Table 4. Table 3 shows that the proposed algorithm performed much better when the initialization is uniform. However, with partitioning initialization, both KGSA and NGSA algorithms were equally successful in finding the global and local optima. In order to fine-tune the population size, it was set to 20 and the error rates of KGSA and NGSA were recorded in Table 5. As can be seen from this table, KGSA had lower error rate than NGSA for both initialization methods in this setting. The results were averaged for 30 independent runs of both algorithms. It should be noted that T_l for all experiments was set to 15.

Considering the results presented in Tables 3-5, one can see that KGSA outperforms NGSA in terms of finding global and local optima and lower error rate. In addition, when reducing the population size, NGSA was sensitive to the initialization type but not the KGSA algorithm since it uses the “loop in loop” method resulting in a larger search space.

In order to evaluate the algorithms in other settings and on more functions, the number of members was set to 20, maximum numbers of iterations was set to 2000, initialization method was partitioning and

Table 1: Comparison of found global maximum (%ADR) obtained with KGSA, NGSA, r2PSO, r3PSO, r2PSO-lhc, r3PSO-lhc, FER-PSO and SPSO using functions F_1 to F_{12}. The results were averaged after fifty independent runs of algorithms.

Function	KGSA	NCOA [20]	FireFly [19]	NGSA	r2 PSO	r3 PSO	r2 PSO –lhc	r3 PSO -lhc	FER-PSO	SPSO
F_1	**100**	**100**	100	100	100	100	100	100	100	100
F_2	**100**	**100**	100	100	98	100	100	100	100	100
F_3	**100**	**100**	100	100	98	98	100	100	100	100
F_4	**100**	**100**	100	100	100	100	100	100	100	100
F_5	**100**	**100**	100	100	92	74	100	98	98	100
F_6	**100**	**100**	100	100	98	100	94	78	88	24
F_7	**100**	**100**	100	100	100	96	98	88	100	22
F_8	**100**	**100**	100	94	100	96	96	96	98	40
F_9	**100**	**100**	100	100	100	100	100	100	100	100
F_{10}	**100**	**90**	100	100	100	100	72	78	100	50
F_{11}(2D)	**100**	**93**	100	100	90	98	98	100	56	49
F_{12}(1D)	**100**	**--**	92	92	94	86	92	90	88	84

Table 2: List of required parameters for discovering global optima in unconstrained functions

	Function	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}
KGSA	***N***	**10**	**10**	**20**	**10**	**20**	**10**	**8**	**30**	**15**	**40**	**100**	500
	Max eval	**800**	**600**	**800**	**750**	**2400**	**1800**	**1280**	**3600**	**1350**	**1000**	**60000**	**90000**
	T_l	20	15	10	15	20	90	80	60	30	50	60	45
NGSA	***N***	50	50	50	50	50	100	100	100	100	500	250	100
	Max eval	100000	100000	100000	100000	100000	100000	100000	100000	100000	100000	100000	100000
	ε	0.01	0.01	0.01	0.01	0.1	0.1	0.1	5	0.01	0.01	0.1	0.01

evaluations were performed on functions F_1 to F_5. Results of this evaluation for KGSA are presented in Tables 6 and 7, and results of other algorithms can be obtained from [15]. Table 6 shows that the proposed algorithm has performed better than other algorithms in terms of finding global and local optima. The number of necessary evaluations for finding optimal solutions is reported in Table 7. This table shows that for 100% discovering the optima, KGSA needed less number of evaluations than the other algorithms. Furthermore,

Table 3: %ADR obtained with the KGSA and NGSA using different population sizes, and uniform initialization for finding all local and global maxima.

Function	N=75		N=50		N=35		N=20	
	KGSA	NGSA	KGSA	NGSA	KGSA	NGSA	KGSA	NGSA
F_1	**100**	100	**100**	100	**100**	93	**100**	80
F_2	**100**	100	**100**	100	**100**	96	**100**	73
F_3	**100**	100	**100**	100	**100**	86	**100**	73
F_4	**100**	100	**100**	96	**100**	90	**100**	66
F_5	**100**	100	**100**	100	**100**	96	**100**	76

Table 4: %ADR obtained with the KGSA and NGSA using different population sizes and partitioning initialization for finding all local and global maxima.

Function	N=75		N=50		N=35		N=20	
	KGSA	NGSA	KGSA	NGSA	KGSA	NGSA	KGSA	NGSA
F_1	**100**	100	**100**	100	**100**	100	**100**	100
F_2	**100**	100	**100**	100	**100**	100	**100**	100
F_3	**100**	100	**100**	100	**100**	100	**100**	100
F_4	**100**	100	**100**	100	**100**	100	**100**	100
F_5	**100**	100	**100**	100	**100**	100	**100**	100

Table 5: The mean error ζ obtained with the KGSA and NGSA using N=20, T=120

Function	Average ζ		
	KGSA uniform initialization	KGSA partitioning initialization	NGSA partitioning initialization
F_1	**1.75e - 6 ± 1.03e - 6**	**1.78e - 6 ± 9.41e - 7**	1.62e - 5 ± 3.47e - 5
F_2	**4.96e - 7 ± 1.54e - 6**	**2.75e - 7 ± 2.94e - 7**	0.001± 1.66e - 4
F_3	**2.41e - 6 ± 2.12e - 6**	**2.35e - 6 ± 1.29e - 6**	0.0012 ± 5.04e - 4
F_4	**6.87e - 7 ± 1.98e - 6**	**5.34e - 7 ± 6.81e - 7**	2.37e - 4 ± 3.38e - 4
F_5	**3.59e - 3 ± 2.13e - 3**	**4.29e - 3 ± 4.22e - 3**	0.0570 ± 0.0262

Table 5A in Appendix shows that KGSA achieved the results presented in Tables 6 and 7 with lower population size and lower "maximum number of iterations" compared to the other algorithms.

Table 6: Comparison of KGSA, NGSA, deterministic crowding, sequential niche and NichePSO algorithms in terms of %ADR for functions F_1 to F_5. The results were averaged after thirty independent runs of algorithms.

Function	KGSA	NGSA	Deterministic Crowding	Sequential Niche	NichePSO
F_1	**100**	100	100	100	100
F_2	**100**	100	93	83	93
F_3	**100**	100	90	100	100
F_4	**100**	100	90	93	93
F_5	**100**	100	90	86	100

Table 7: Average number of fitness function assessments needed to converge for each niching algorithm for functions F_1 to F_5. The results were averaged after thirty independent runs of algorithms.

Function	KGSA	NGSA	Deterministic Crowding	Sequential Niche	NichePSO
F_1	**208 ± 115**	1786 ± 204	14647 ± 4612	4102 ± 577	2372 ± 109
F_2	**211 ± 103**	1892 ± 561	13052 ± 2507	3505 ± 463	2934 ± 475
F_3	**286 ± 136**	1752 ± 273	13930 ± 3284	4141 ± 554	2404 ± 195
F_4	**272 ± 149**	1806 ± 307	13929 ± 2996	3464 ± 287	2820 ± 517
F_5	**897 ± 446**	2033 ± 201	14296 ± 3408	3423 ± 402	2151 ± 200

Also, the KGSA algorithm was applied to functions F_6 to F_{10} using partitioning initialization. Results of this comparison are presented in Tables 8 and 9. The first table shows comparisons with NGSA and the latter presents comparisons with NNGSA. As can be seen from Table 8, KGSA has higher success rate in all situations, and the lower error-rate and the "number of evaluations" in most cases. Moreover, Table 6A in Appendix shows that results presented in Table 8 were achieved with lower population size and lower "number of evaluations" than those of NGSA. All parameters and results for NGSA and KGSA were adopted from [15].

5.4.3 Finding global optima in constrained functions

KGSA was also evaluated on constrained functions F_{13} to F_{15}. Average results of 50 runs using partitioning initialization are shown in Table 10. This table illustrates that KGSA achieved 100% success rate for all the constrained functions in addition to having a lower error-rate than the other algorithms in most cases. Moreover, this algorithm had the lowest "number of evaluations" on two out of three of these functions. In case of function F_{15}, "number of evaluations" for KGSA was only greater than that of NGSA. Table 7A in Appendix shows parameter settings used for these experiments and indicates that results of KGSA were achieved with lower population size and less iterations.

Table 8: Results of KGSA and NGSA in terms of %ADR and the number of fitness assessments needed to converge for each niching algorithm using functions F_6 to F_{10}. The results were averaged after fifty independent runs of algorithms.

	KGSA			NGSA		
Function	**Number of fitness evaluations**	**Average ζ**	**ADR(%)**	**Number of fitness evaluations**	**Average ζ**	**ADR(%)**
F_6	**413 ± 113**	**1.02e – 4 ± 1.38e – 4**	**100**	542 ± 386	4.27e – 4 ± 4.18e – 4	100
F_7	**267 ± 128**	**5.12e – 6 ± 5.66e – 6**	**100**	321 ± 118	7.44e – 6 ± 6.01e – 6	100
F_8	**873 ± 721**	**6.51e – 5 ± 2.08e – 4**	**100**	966 ± 765	2.19e – 3 ± 1.37e – 2	94
F_9	**1097 ± 856**	**7.29e – 5 ± 1.88e – 4**	**100**	1122 ± 353	1.50e – 3 ± 3.33e – 2	98
F_{10}	11318 ± 2861	4.51e – 2 ± 6.73e – 3	**100**	6186 ± 2133	9.66e – 4 ± 1.18e – 3	100

Table 9: Results of KGSA and NNGSA in terms of %ADR and the number of fitness assessments needed to converge for each niching algorithm using functions F_6 to F_{10}. The results were averaged after fifty independent runs of algorithms.

	KGSA		NNGSA [1]	
Function	**Number of fitness evaluations**	**Average ζ**	**Number of fitness evaluations**	**Average ζ**
F_6	413 ± 113	1.02e – 4 ± 1.38e – 4	-	-
F_7	267 ± 128	5.12e – 6 ± 5.66e – 6	-	-
F_8	873 ± 721	6.51e – 5 ± 2.08e – 4	-	-
F_9	1,097 ± 856	7.29e – 5 ± 1.88e – 4	200 ± 0	0 ± 0
F_{10}	11,318 ± 2861	4.51e – 2 ± 6.73e – 3	1,5740e + 4 ± 8.313e + 3	3,6015e - 4 ± 0.151 e - 1

5.5 Sensitivity analysis of parameter T_l in KGSA

In this section, sensitivity of KGSA to T_l is evaluated. As stated previously, $T_l$represents the number of repetitions of the inner loop of "loop in loop" method. This parameter can affect the performance of KGSA since loops have a major role in finding optima; therefore, correct setting of T_l is very important. If T_l is too small, few generations will be created in a loop and therefore, the population will not develop adequately and candidates with lower fitness will go to the next generation. On the other hand, if T_l is too large, the number of loops will reduce and consequently, search space will be small. Therefore, it is necessary to select an appropriate value for T_l. For doing this, an experiment was conducted to evaluate KGSA on functions F_1 to

Table 10: Performance comparison when constrained test functions were used.

Function	Methods	Average ζ	Number of fitness evaluations	ADR(%)
F_{13}	r2PSO	3.1586e - 04±0.0080	510±438.1082	100
	r3PSO	6.3425e - 04±0.0105	546±396.0056	100
	r2PSO-lhc	5.5256e - 04±0.0067	474±383.7569	100
	r3PSO-lhc	1.2619e - 03±0.0083	450±292.2498	100
	Deterministic crowding	4.3717e - 04±0.0103	1,098±662.8602	100
	NGSA	3.1255e - 04±0.0107	78±146.0919	100
	NNGSA	2.5352 - 04 ± 6.1e - 03	2,080±19,110	100
	KGSA	**6.04e - 05 ± 5.67e - 05**	92±43	**100**
F_{14}	r2PSO	2.3313e - 02±2.2522	2,396±0.1469	100
	r3PSO	7.1531e - 03±0.9835	2,092±0.5050	100
	r2PSO-lhc	8.9047e - 03±1.8314	2,476±0.5123	100
	r3PSO-lhc	1.3116e - 02±2.1065	2,232±0.5539	100
	Deterministic crowding	1.6902e - 02±0.4956	21,552±1.0056	100
	NGSA	1.9672e - 02±0.3790	1,944 ± 1.2944e + 03	100
	NNGSA	2.0176e-03±0.0365e-02	1,413±1,080	100
	KGSA	**5.79e - 03 ± 0.04098**	**1,395 ± 587**	**100**
F_{15}	r2PSO	0.3966±0.2352	788±0.0849	100
	r3PSO	0.381±0.1997	792±0.0849	100
	r2PSO-lhc	4.9832e - 03±0.2081	812±0.0396	100
	r3PSO-lhc	0.3237±0.1873	796±0.0480	100
	Deterministic crowding	6.2571e - 04±0.0552	6,672±0.0283	100
	NGSA	5.0262e - 04±0.0203	3,116±1.0135	100
	NNGSA	1.4415e-03±3.02e-02	5,183±1,560	100
	KGSA	**3.37e - 04 ± 6.37 e - 04**	**480 ± 103**	**100**

F_5 with different values for T_l. In this experiment, the population number was set to 20, the first population was initialized by the partitioning method, and the maximum number of generations was set to 120. Result of 50 independent runs of KGSA in this setting is shown in Table 8A in Appendix. As can be seen from this table, KGSA success rate on F_1 to F_4 has not been changed for different values of T_l. However, for function F_5, result is different. When T_l is 10, the success rate is 96% which indicates that the number of repetitions for building the population is not enough. On the other hand, when T_l is 40 and 60, the success rate is 98% and 94% respectively. This points out the excessive number of repetitions in a loop for building the population.

This means that the search space has been reduced in this case. Overall, it can be concluded that the success rate is not significantly affected by T_l, showing the low sensitivity of KGSA to T_l.

5.6 Statistical Tests

In the final test, KGSA is statistically compared with other methods. For this purpose, Friedman test has been performed on possible metrics between KGSA and other methods. It is known that Friedman test is based on the chi-square distribution with for analyzing the statistical significant difference between the results of several methods [35]. The Friedman test is a non-parametric statistical test for ranking the algorithms and evaluating whether their results are statistically significantly different or not. It computes a score for each algorithm on a specific criterion and finally ranks them based on the scores. In the Friedman test, the null hypothesis states that the methods are not statistically significantly different. If the p-value is less than a predetermined level, null hypothesis is rejected which shows that results of the methods are significantly different. In this paper, p-value was set to 0.05.

Results of Friedman test are provided in Table 11. In this table, the best method in each column is shown in bold face and the second best is underlined. This table indicates that KGSA and Firefly [19] algorithms are the most successful. It must be noted that since we did not have access to the codes of other algorithms and some of the benchmark functions used in this study were different, all tests could not be performed on all algorithms and thus, some of the cells in Table 11 are empty.

Table 11: Friedman test on KGSA and the other algorithms

	Finding all global maxima	Number of Function Evaluations
r2 PSO	3.77	7.25
r3 PSO	5.32	8.50
r2 PSO lhc	4.73	6.25
r3 PSO lhc	5.09	7.25
FER PSO	4.82	8.75
SPSO	4.77	7.75
NGSA	<u>6.36</u>	6.00
NCOA	6.23	5.25
Firefly	**6.95**	3.25
NNGSA	---	<u>2.75</u>
KGSA	**6.95**	**2.50**
P-value	*0.006*	*0.033*
Degree of Freedom	9	10

6 Conclusions and Future Works

In this paper, KSGA, a novel Gravitational Search Algorithm for multimodal problems was proposed. KSGA incorporated *k-means* and a new elitism strategy called “loop in loop” into the conventional Gravitational Search Algorithm (GSA). First in KGSA, the initial population was clustered by K-means and after that, the first population was created by selecting the members from different clusters (niches). This resulted in a large search space and thus increased the chance of finding local and global optima in KGSA. “loop in loop” technique was used to guide the members of each niche to the optimum direction according to their clusters.

With these modifications, KGSA does not need the following items: the type of population initialization and parameter "radius of niche". Evaluations on different benchmark functions showed that KGSA is superior to other GSA based evolutionary algorithms in finding both local and global optima. We intend to exploit fuzzy methods to determine the number of optima at the beginning of KGSA and incorporating the "loop in loop" technique into unimodal problems for future works.

References

[1] Haghbayan P., Nezamabadi-Pour H., Kamyab S., A niche GSA method with nearest neighbor scheme for multimodal optimization, Swarm and Evolutionary Computation, 35, 2017, 78-92

[2] Subhrajit R., Minhazul I., Das S., Ghosh S., Multimodal optimization by artificial weed colonies enhanced with localized group search optimizers, Applied Soft Computing, 13, 2013, 27-46

[3] Tang K. S., Man K., Kwong S., He Q., Genetic algorithms and their applications, IEEE Signal Processing Magazine, 13, 1996, 6, 22-37

[4] Kirkpatrick S., Vecchi M., Optimization by simmulated annealing, Science, 220, 1983, 4598, 671-680

[5] Farmer J. D., Packard N. H., Perelson A. S., The immune system, adaptation, and machine learning, Physica D: Nonlinear Phenomena, 22, 1986, 1, 187-204

[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, 26, 1996, 1, 29-41

[7] Kenndy J., Particle Swarm Optimization, in "Encyclopedia of machine learning", Sammut C., Webb G. I., eds., Springer, Boston, MA, 2011, 760-766

[8] Rashedi E., Nezamabadi-Pour H., Saryazdi S., GSA: a Gravitational Search Algorithm, Information sciences, 179, 2009, 13, 2232-2248

[9] Rashedi E., Nezamabadi-Pour H., Saryazdi S., BGSA: binary gravitational search algorithm, Natural Computing, 9, 2010, 3, 727-745

[10] Brits R., Engelbrecht A. P., Van den Bergh F., A niching particle swarm optimizer, In: Proceedings of the 4th Asia-Pacific conference on simulated evolution and learning, Paper Presented at Proceedings of the 4th Asia-Pacific conference on simulated evolution and learning (Orchid Country Club, Singapore), 2002, November 18, Orchid Country Club, 692-696

[11] Thiemard E., Economic generation of low-discrepancy sequences with a b-ary Gray code, Technical Report, 1998, Department of Mathematics, Ecole Polytechnique Fédérale de Lausanne, Lausanne, Switzerland

[12] Streichert F., Stein G., Ulmer H., Zell A., A clustering based niching method for evolutionary algorithms, In: Genetic and Evolutionary Computation Conference, Paper Presented at Genetic and Evolutionary Computation Conference (Chicago, USA), 2003, July 9-11, Springer, 644-645

[13] Li X., Adaptively choosing neighbourhood bests using species in a particle swarm optimizer for multimodal function optimization, In: Genetic and Evolutionary Computation Conference, Paper Presented at Genetic and Evolutionary Computation Conference, (Seattle, USA), 2004, June 26-30, Springer, 105-116

[14] Seo J. H., Im C. H, Heo C. G., Kim J. K., Jung H. K., Lee C. G., Multimodal function optimization based on particle swarm optimization, IEEE Transactions on Magnetics, 42, 2006, 4, 1095-1098

[15] Yazdani S., Nezamabadi-Pour H., Kamyab S., A gravitational search algorithm for multimodal optimization, Swarm and Evolutionary Computation, 14, 2014, 1-14

[16] Cuevas E., Reyna-Orta A., A cuckoo search algorithm for multimodal optimization, 2014, The Scientific World Journal, 2014

[17] Chang W. D., A modified particle swarm optimization with multiple subpopulations for multimodal function optimization problems, Applied Soft Computing, 33, 2014, 170-182

[18] Naik M. K., Panda R., A new hybrid CS-GSA algorithm for function optimization, In: Proceedings of the IEEE International Conference on Electrical, Electronics, Signals, Communication and Optimization, Paper Presented at IEEE International Conference on Electrical, Electronics, Signals, Communication and Optimization, (Visakhapatnam, India), 2015, January 24-25, IEEE, 1-6

[19] Nekouie N., Yaghoobi M., A new method in multimodal optimization based on firefly algorithm, Artificial Intelligence Review, 46, 2016, 2, 267-287

[20] Rim C., Piao S., Li G., Pak U., A niching chaos optimization algorithm for multimodal optimization, Soft Computing, 22, 2016, 2, 621-633

[21] Tang K., Yang P., Yao S., Negatively correlated search, IEEE Journal on Selected Areas in Communications, 34, 2016, 3, 542-550

[22] Chang W. D., Multimodal function optimizations with multiple maximums and multiple minimums using an improved PSO algorithm, Applied Soft Computing, 60, 2017, 60-72

[23] Goldberg D. E., Richardson J., Genetic algorithms with sharing for multimodal function optimization, In: Proceedings of the Second International Conference on Genetic Algorithms and their applications, Paper Presented at the Second International Conference on Genetic Algorithms and their applications, (Cambridge, USA), 1987, Hillsdale: Lawrence Erlbaum, 41-49

[24] Pétrowski A., A clearing procedure as a niching method for genetic algorithms, In: Proceedings of IEEE International Conference on Evolutionary Computation, Paper Presented at Proceedings of IEEE International Conference on Evolutionary Computation,

(Nagoya, Japan), 1996, IEEE, May 20-22, 798-803
[25] De Jong K. A., An Analysis of the Behavior of a Class of Genetic Adaptive Systems, In: Techincal Report, 1975, Computer and Communication Sciences Department, College of Literature, Science, and the Arts, University of Michigan
[26] Mahfoud S. W., Simple Analytical Models of Genetic Algorithms for Multimodal Function Optimization, In: Proceedings of the 5th International Conference on Genetic Algorithms, Paper Presented at Proceedings of the 5th International Conference on Genetic Algorithms (San Fransisco, USA), 1993, San Fransisco: Morgan Kaufman Publishers, 643
[27] Mahfoud S. W., Crossover Interactions Among Niches, In: Proceedings of the First IEEE Conference on Evolutionary Computation, Paper Presented at the First IEEE Conference on Evolutionary Computation, (Orlando, USA), 1994, June 27-29, IEEE, 188-193
[28] Mahfoud S. W., Niching methods for genetic algorithms, Urbana, 51, 1994, 95001, 62-94
[29] Mengshoel O. J., Goldberg D. E., Probabilistic crowding: Deterministic crowding with probabilistic replacement. In: Genetic and Evolutionary Computation Conference, Paper Presented at Genetic and Evolutionary Computation Conference, (San Fransisco, USA), 1999, July 8-12, San Fransisco: Morgan Kaufman Publishers, 409
[30] Kimura S., Matsumura K., Constrained multimodal function optimization using a simple evolutionary algorithm, In: IEEE Congress on Evolutionary Computation, Paper Presented at IEEE Congress on Evolutionary Computation, (New Orleans, USA), 2011, June 5-8, IEEE, 447-454
[31] Li X., Niching without niching parameters: particle swarm optimization using a ring topology, IEEE Transactions on Evolutionary Computation, 14, 2010, 1, 150-169
[32] Parrott D., Li X., Locating and tracking multiple dynamic optima by a particle swarm model using speciation, IEEE Transactions on Evolutionary Computation, 10, 2006, 4, 440-458
[33] Li X., Multimodal function optimization based on fitness-euclidean distance ratio, In: Genetic and Evolutionary Computation Conference, Paper Presented at Genetic and Evolutionary Computation Conference, (Washington, USA), 2007, June 25-29, New York: ACM, 78-85
[34] Zhang J., Zhang J. R., Li K., A sequential niching technique for particle swarm optimization, In: Advances in Intelligent Computing, Paper Presented at Advances in Intelligent Computing, (Hefei, China), 2005, August 23-26, Berlin Heidelberg: Springer-Verlag, 390-399
[35] García S., Fernández A., Luengo J., Herrera F., Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power, Information Sciences, 180, 2010, 10, 2044-2064

Table 1A: Unconstrained test functions in the experiments (see [15])

Name	Test function	Range	Number of global peak	Number of all peak
Equal maxima	$F_1(x) = \sin^6(5\pi x)$	$0 \le x \le 1$	5	5
Decreasing maxima	$F_2(x) = e^{-2\log(2)\left(\frac{x-0.1}{0.8}\right)^2}\sin^6(5\pi x)$	$0 \le x \le 1$	1	5
Uneven maxima	$F_3(x) = \sin^6\left(5\pi\left(x^{3/4} - 0.05\right)\right)$	$0 \le x \le 1$	5	5
Uneven decreasing maxima	$F_4(x) = e^{-2\log(2)\left(\frac{x-0.08}{0.854}\right)^2}\sin^6\left(5\pi\left(x^{3/4} - 0.05\right)\right)$	$0 \le x \le 1$	1	5
Himmelblau's function	$F_5(x_1, x_2) = 200 - (x_1^2 + x_2 - 11)^2 - (x_1 + x_2^2 - 7)^2$	$-6 \le x_1, x_2 \le 6$	4	4
Two-peak trap	$F_6(x) = \begin{cases} 160/15\,(15 - x) & \text{for } 0 \le x \le 15 \\ 200/5\,(x - 15) & \text{for } 15 \le x \le 20 \end{cases}$	$0 \le x \le 20$	1	2
Central two-peak trap	$F_7(x) = \begin{cases} 160/10\,x & \text{for } 0 \le x \le 10 \\ 160/5\,(15 - x) & \text{for } 10 \le x \le 15 \\ 200/5\,(x - 15) & \text{for } 15 \le x \le 20 \end{cases}$	$0 \le x \le 20$	1	2
Five-uneven-peak-trap	$F_8(x) = \begin{cases} 80(2.5 - x) & \text{for } 0.0 \le x \le 2.5 \\ 64(x - 2.5) & \text{for } 2.5 \le x \le 5.0 \\ 64(7.5 - x) & \text{for } 5.0 \le x \le 7.5 \\ 28(x - 7.5) & \text{for } 7.5 \le x \le 12.5 \\ 28(17.5 - x) & \text{for } 12.5 \le x \le 17.5 \\ 32(x - 17.5) & \text{for } 17.5 \le x \le 22.5 \\ 32(27.5 - x) & \text{for } 22.5 \le x \le 27.5 \\ 80(x - 27.5) & \text{for } 27.5 \le x \le 30 \end{cases}$	$0 \le x \le 30$	2	5
Six-Hump Camel Back	$F_9(x_1, x_2) = -4\left[\left(4 - 2.1x_1^2 + \frac{x_1^4}{3}\right)x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2\right]$	$-1.9 \le x_1 \le 1.9$ $-1.1 \le x_2 \le 1.1$	2	4
Shekel's Foxholes	$F_{10}(x_1, x_2) = 500 - \left\{0.002 + \sum_{i=0}^{24}\left[1 + i + \left(x_1 - a(i)\right)^6 + \left(x_2 - b(i)\right)^6\right]^{-1}\right\}^{-1}$ where $a(i) = 16\left((i \bmod 5) - 2\right)$, and $b(i) = 16(\lfloor (i/5) \rfloor - 2)$	$-65.536 \le x_1, x_2 \le 65.536$	1	25
Inverted Shubert	$F_{11}(x) = -\prod_{i=1}^{n}\sum_{j=1}^{5} j\cos[(j+1)x_i + j]$	$-10 \le x_i \le 10$	3n	a
Inverted Vincent	$F_{12}(x) = 1/n\sum_{i=1}^{n}\sin(10\log(x_i))$	$0.25 \le x_i \le 10$	6^n	6^n

Table 2A: Constrained test functions in the experiments (see [15])

Name	Test function	Range	Number of global peak	Number of all peak
cRastrigin	$F_{13}(x) = 10n + \sum_{i=1}^{n}[x_i^2 - 10\cos(2nx_i)]$ $\text{s.t. } h(x) = \sum_{i=1}^{n} x_{i=0}$	$-5.12 \le x_i \le 5.12$	2	2
cGriewank	$F_{14}(x) = 1/4000\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$ $\text{s.t. } h(x) = \frac{1}{512(n-1)}\sum_{i=2}^{n} x_i^2 - \lvert x_1\rvert = 0$	$-512 \le x_i \le 512$	4	4
Deb's constrained	$F_{15}(x) = (x_1 + a_1)^2 + (x_2 + a_2)^2 + \ldots + (x_n + a_n)^2$ $\text{s.t. } [(x_1 + b_1)^2 + \ldots (x_n + b_n)^2] \ge n^2,$ $-(n+1) \le x_i \le n+1$	$-3 \le x_i \le 3$	1	4

Table 3A: Mean value of fitness function assessments needed to converge for each PSO based algorithm for the results shown in Table 1. The results were averaged over fifty independent runs of algorithms

Function	KGSA	NGSA	r2 PSO	r3 PSO	r2 PSO -lhc	r3 PSO -lhc	FER-PSO	SPSO
F_1	**214±133**	263±89	376±30	443±51	396±51	447±52	384±29	355±30
F_2	**105±108**	300±107	2120±1999	141±11	143±14	144±13	170±12	127±9
F_3	**263±140**	334±81	2430±1994	2440±1994	456±33	623±273	317±31	343±23
F_4	**130±124**	316±76	175±17	160±20	178±18	162±16	189±20	144±13
F_5	**864±382**	1632±330	7870±2891	21400±5467	1490±138	7380±3347	5070±1945	1250±45
F_6	**304±110**	477±409	3460±197	2620±874	7390±3340	23200±5834	14400±4535	77200±5859
F_7	290±130	234±108	2960±1520	5340±2764	4340±2229	13100±4588	2110±227	78300±5856
F_8	**557±346**	694±853	978±186	4650±2784	4710±2783	6730±3088	2660±1992	63300±6773
F_9	**230±121**	1032±892	619±24	684±30	618±30	650±25	965±53	653±32
F_{10}	**3119±2172**	4164±1768	4360±559	5310±453	29700±6277	24800±5738	3470±336	42800±6968
F_{11}(2D)	33344±8099	5369±1930	55900±2676	39100±1648	37800±1480	32400±581	94900±1261	61600±4463
F_{12}(1D)	12480±18175	2134±430	8310±3371	15400±4906	9600±3824	14700±4344	13000±4601	17000±5162

Table 4A: Average number of fitness function evaluations required to converge for each of the evolutionary algorithms. The results were averaged over fifty independent runs of algorithms

Function	KGSA	NCOA [20]	Firefly [19]	NNGSA [1]
F_1	**214±133**	**607±31**	**182±94**	**-**
F_2	**105±108**	**614±22**	**206±91**	**-**
F_3	**263±140**	**622±34**	**186±101**	**-**
F_4	**130±124**	**620±31**	**180±88**	**-**
F_5	**864±382**	**1180±50**	**1152±274**	**1466±513**
F_6	**304±110**	**498±76**	**460±178**	**-**
F_7	**290±130**	**477±63**	**489±184**	**-**
F_8	**557±346**	**580±39**	**528±215**	**200±0**
F_9	**230±121**	**2196±72**	**692±402**	**200±0**
F_{10}	**3119±2172**	**27539±1431**	**-**	**15740±8313**
F_{11}(2D)	**33344±8099**	**29397±2555**	**1260±394**	**30840±7254**
F_{12}(1D)	**12480±18175**	**-**	**1002±404**	**-**

Table 5A: List of required parameters for discovering local and global optima in Tables 6, 7

	Function	F_1	F_2	F_3	F_4	F_5
KGSA	N	**10**	**10**	**20**	**10**	**20**
	T	**80**	**60**	**40**	**75**	**120**
	T_l	20	15	10	15	20
Other Algorithms	N	20	20	20	20	20
	T	2000	2000	2000	2000	2000

Table 6A: List of required parameters for discovering local and global optima in Table 8

	Function	F_6	F_7	F_8	F_9	F_{10}
KGSA	N	**15**	**8**	**30**	**15**	**80**
	$Max\ eval$	**2700**	**5600**	**3600**	**5250**	**20000**
	T_l	90	70	60	50	50
NGSA	N	100	100	100	100	500
	$Max\ eval$	100000	100000	100000	100000	100000
	ε	0.1	0.1	5	0.01	0.01

Table 7A: List of required parameters for discovering global optima in Table 10

	Function	F_{13}	F_{14}	F_{15}
KGSA	N	**10**	**40**	**25**
	T	**80**	**100**	**45**
	T_l	40	50	45
NGSA	N	300	200	200
	T	200	100	100

Table 8A: Success rate of the KGSA for the different values of T_l

Function	$T_l = 10$	$T_l = 15$	$T_l = 20$	$T_l = 30$	$T_l = 40$	$T_l = 60$
F_1	100	100	100	100	100	100
F_2	100	100	100	100	100	100
F_3	100	100	100	100	100	100
F_4	100	100	100	100	100	100
F_5	96	100	100	100	98	94