IJCI V11-1(2024) 44–61
International Journal of Computers and Information
(IJCI)
Available online at *https://ijci.journals.ekb.eg/*

A New Adaptive Cellular Genetic Algorithm for Mixed Variable Optimization Problems

Alaa Fahim
Mathematics Department, Faculty of science, Assiut university, Egypt
alaa@aun.edu.eg, dralaafahim@gmail.com

Abstract

The cellular genetic algorithm (CGA) improves the genetic algorithm for a subclass(GA) with a dispersed population where an exchange of individuals is limited to close neighbors. In this research, the adaptive cellular genetic algorithm (ACG) is applicable to better explore the search space. The ACG algorithm assists in *reaching the optimal solution as quickly as possible. Various problems regarding mixed variable optimization (MVO) problems arise in numerous models and real applications. We applied the ACG algorithm to deal with the MVO problem, which is called the adaptive cellular genetic mixed variable ACGMV method. ACGMV algorithm performance is evaluated using a number of benchmark test issues. The experimental findings indicate that the ACGMV algorithm performs better than other methods. In addition, we applied ACGMV to solve the hard data-clustering problem. which is called ACGMV-HC. The data clustering problem is formulated using the mixed-variable programming methodology. Based on our result, the ACGMV-HC algorithm is more effective than other methods*

Keywords: Mixed variable optimization; Cellular genetic algorithm; hard cluster.

1. Introduction

One of the most widely used metaheuristic algorithms for efficient random search is the genetic algorithm (GA). Metaheuristics algorithms' main benefit is that they provide a balance in finding the global optimum of optimization problems in a reasonable amount of time. The GA operates on a population set and employs stochastic procedures which are known as genetic procedures (selection, crossover, and mutation) [25].

GA includes a cellular genetic algorithm (CGA) with a spreading population whereas the exchange between individuals is limited to close neighbors. To prevent the rapid convergence of GA, we disperse the population which permits the preservation of the population diversity, resulting in better investigation of the search area. Thus, the enactment of the algorithm [13].

In this study, the grid structure of the adaptive cellular genetic algorithm (ACGA) method is applied to achieve convergence in a short time. The ACGA is based on work on the neighborhood selection. The major goal of our algorithm is the selection of neighborhoods from the grid to achieve a more equitable balance between extraction and exploration. The selection of neighborhood members is adaptive to provide diversification in exploring the search space, and intensification, especially on the most effective solutions to find the optimal solution faster.

The MVO has been a subject of interest for many researchers due to its various models and applications. The MVO is used in many areas, such as chemical engineering [27], harvesting machine systems modeling [22], and heat network designing in an energy system [32]. There are several methods to solve MVO problems including filter GA [19], hybrid method according to particle swarm optimization and genetic algorithm [29], multistart Hooke and Jeeves filter method [11], difference-genetic co-evolutionary algorithm [16], heuristic algorithm using line-up competition and pattern search [30], The coordinate search filter algorithm is based on branches and bounds [15].

There are many methods tried to solve mixed variable programming problems [11,16,19,23]. M. Costa[11] introduced a multistart method which utilized an extended iteration of the Hooke and Jeeves algorithm to calculate multiple solutions for mixed variable optimization problems. Y. Gao[16] solved a constraint bi-objective mixed variable optimization issue using the difference co-evolution algorithm., A. Hedar [19] developed a hybrid genetic algorithm and filter method as a local technique to handle mixed variable optimization problems with and without constraints. Y. Lin [23] introduced the hybrid differential evaluation to achieve optimal solutions for mixed variable optimization problems. In this paper, the adaptive cellular genetic for mixed variable (ACGMV) algorithm is used to solve mixed variable optimization (MVO) problems.

In this study, the ACGMV algorithm is applied to solve MVO problems. It provides a superior quality of solutions, which are examined by 9 unconstrained and 12 benchmark test problems. A pattern search technique is included in the ACGMV algorithm to improve the exploitation and obtain the optimal solution faster. The ACGMV^P makes the pattern around the best solutions to make the intensification around them.

Furthermore, data clustering [21,26] means dividing data into groups of like objects. Data clustering is relevant to a wide variety of fields and is critical in a wide variety of applications. Data clustering is typically used to analyze huge datasets and data with a significant number of attributes, such as document extraction, image segmentation, market research, and social network analysis. The majority of methods for data clustering are based on the two most prevalent techniques: hierarchical and partitional. The output is a tree in the hierarchical clustering that shows a sequence of clusters, each cluster containing a partition of a dataset. On the other hand, algorithms for partitional clustering categorize the data set into a predetermined number of groups. Data clustering is classified into two types: fuzzy, and hard clustering. hard clustering is considered in this work.

Many attempts are made to handle the hard clustering problem with GAS. A GA-based clustering technique [26] has been applied to provide an optimal clustering regarding the clustering metric in [21], GAS has a novel sort of crossover operator that has good partitions by exchanges neighboring centers but regrettably does not reach high-dimensional datasets. In [10], GAS is used to study the effectiveness and efficiency of employing GAS.

The clustering issue can be reduced to an optimization issue. However Many optimization issues can be phrased as a clustering problem, which is an interesting observation. In this paper, ACGA is applied to the Hard Clustering problem (denoted by ACGA-HC) using the MVO problem. To our knowledge, just a handful of attempts [14, 28] to solve the difficult clustering problem as MVO formulation. According to our computational findings, the ACGA-HC m technique surpasses other available methods.

The remainder of the paper is arranged accordingly. The concept of a CGA method, the MVO problem, and various mathematical formulas for the hard clustering problem are presented in Section ٢. Section 3 describes ACGMV algorithms and their components. The computational results of the proposed methods are introduced in Section 4. In Section 5, the computational experiments are presented using the ACGA-HC method and the comparison of our method with other methods in three different datasets. Conclusion and future work are discussed in Section ٦.

2. Preliminaries

The notions will be demonstrated to be utilized in the suggested procedure.

2.1. Cellular Genetic Algorithm

The CGA is a type of GA that allows for a more thorough search space exploration and algorithm implementation [13]. CGA's intersecting small neighborhoods help to explore the area of search because they provide a slow distribution of solutions throughout the population. It provides an exploration (diversification) of the population, while exploitation (intensification) happens within each neighborhood by genetic operations [1]. The population in GA could be decentralized in two main ways: cellular GA (CGA) and distributed GA (DGA) [2]. In DGA, the population is divided into various minor subpopulations that exchange some information with one another. Hence, each island's GA investigates a separate search space region, thereby conserving the entire population diversity. In CGA, an individual can relate to the breeding ring. The main difference between DGA and CGA is how their populations are structured.

CGA's purpose is to allow the population structure reshaped as a connected graph, in communication with its closest neighbors. A toroidal mesh is used to map each individual [2]. Here, the grid boundary individuals are connected with individuals in the same row or column on the opposite borders. As a result, a toroidal grid is formed in which each individual has precisely the same number of neighbors. Each grid point has a neighborhood that intersects with surrounding residents' neighborhoods. Each neighborhood is identical in size and shape. Furthermore, for every individual, there are six typical neighborhood structures: L5, L9, C9, C13, C21 and C25 [2].

- L5: This is also called Linear5, NEWS or von Neuman. The four closest members—North, East, West, and South—make up its 4 nearest individuals.
- C9 (Compact9): C9 is also called Moore neighborhood. Here The neighborhood is made up of the central person and the eight others nearby. -L9 (Linear9): It consists of the central individual and its two nearest neighbors along the horizontal and vertical axes.
- C13 (Compact13): Combining L9 and C9 communities created this neighborhood.
- C21 (Compact21): It is composed of the 20 individuals in the horizontal, vertical, and diagonal directions from the one in the center.
- C25 (Compact25): It consists of 24 individuals surrounding the individual under consideration.

When variation operators are utilized, individuals in CGA can only swap with their neighbors during the reproductive cycle. These reproductive processes occur within the selected individual and its neighborhood. The process includes selecting two parents through the current individual and its neighbors, putting the operators for variation to work on them (selection, crossover, and mutation), and substituting the created offspring for the considered individual.

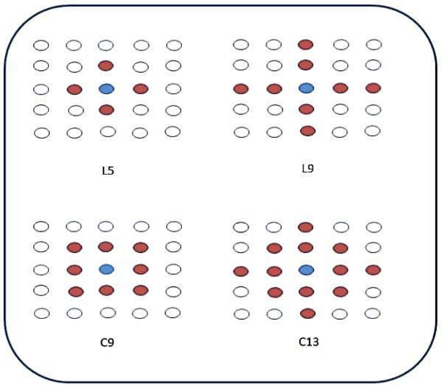

Fig. (1): The block diagram of four common neighborhood structures in CGA.

Moreover, Asynchronously or synchronously updating cells is an option [12]. All the cells in synchronous (parallel) are updated at once, whereas in asynchronous or sequential updates, each cell is updated. Furthermore, we have a better concept of selection behavior in asynchronous CGA than in the synchronous update. In addition, there are four distinct methods for sequentially updating CGA cells: fixed line sweep (FLS), new random sweep (NRS), fixed random sweep (FRS), and uniform choice (UC) [2].

- In the FLS update, the n grid cells are sequentially updated in a row. It is the simplest method.
- For the FRS, with no replacement, A uniform probability is used to select the next cell to be updated, resulting in a specific update sequence. $(\boldsymbol{c}_{\mathbf{1}}^{\boldsymbol{j}}, \boldsymbol{c}_{\mathbf{2}}^{\boldsymbol{k}}, \ldots, \boldsymbol{c}_{\boldsymbol{n}}^{\boldsymbol{m}})$, where $\boldsymbol{c}_{\boldsymbol{q}}^{\boldsymbol{p}}$ indicates that cell number p has been updated at time q and $(j, k, \ldots.., m)$ is a permutation of the n cells. All updated cycles employ the same permutation.
- NRS functions similarly to FRS with the exception that for each sweep of the array, a new random cell permutation is used.
- In the UC updating method, the next cell to be updated is replaced with a uniform probability after a random selection. This pertains to the binomial distribution for the updated probability.

It should be noted that hierarchical CGA depends on how GA relies on how the best individuals are moved to the center of the grid, so that better solutions may be obtained rapidly, while individuals preserve new solutions for exploring the search space. Moreover, CGA is used for manipulating continuous optimization problems [12] and is applied in many areas such as transportation [9].

2.2. Mixed Variable Optimization Problem

The general mathematical form of MVO problems, which contain both integer and continuous variables, is given as the following equation 1:

$$\textbf{Min} \quad f(r,s), \qquad (1)$$

$$\textit{Subject to} \quad \phi i(r,s) \leq 0, \quad i = 1,\dots,n, \qquad (2)$$

$$\psi j(r,s) = 0, \quad j = 1,\dots,m, \qquad (3)$$

$$l_r \leq r \leq u_r, \quad l_s \leq s \leq u_s, \qquad (4)$$

$$r = r_1, r_2, \dots, r_{nr}, \quad s = s_1, s_2, \dots., s_{ns}.$$

Where $r, l_r, u_r \in R^{nr}$, $s, l_s, u_s \in Z^{ns}$, nr and ns are the number of continuous and discrete variables respectively, l_r and u_r are the lower and upper bounds for continuous variables, 1_s and u_s are the vectors of the lower and upper bounds for the integer variables respectively. Assume that the functions f, $\phi_i, i = 1,\dots,n$ *and* $\psi_j, j = 1,\dots,m$ are nonconvex. The penalty approach [24] was used to convert a constrained optimization issue to an unconstrained optimization problem whose the solution must converge to the standard constrained problem solution.

2.3 Hard Clustering Problem

In cluster analysis, it is assumed that X consists of a finite number of *d*-dimensional points R^d, is given as following:

$$X = \{x_1, \dots, x_n\}, \textit{where } x^i \in Rd, i = 1, \dots, n.$$

The set *X* is partitioned into a given number *q* disjoint clusters $C_i, i = 1, \dots, q$.

$$X = \bigcup_{i=1}^{q} c_i \qquad (5)$$

Where sets $C_i, i = 1, \dots, q$. are called clusters. As mentioned before, there are two kind of clustering problems: hard and fuzzy clustering problem.

- Hard clustering problem : Here, Each data point is associated with a single cluster.
- Fuzzy clustering problem: The clusters overlap here, and each member appears to various degrees in each cluster.

For the investigated the hard clustering problem, so we assume that

$$C_i \cap C_k = \varphi, \forall i, k = 1, \dots, q, i \neq k. \quad (6)$$

Further, clustering is reduced to an optimization problem as follows:

$$\boldsymbol{min\ \varphi\ (C, A)} = \frac{1}{n}\sum_{i=1}^{q}\sum_{x \in C_i} \|a^i - x\|^2, \qquad (7)$$

Where $C \in \bar{C}, a_1, \ldots, a_q \in R^d$, $\|.\|$ denote the Euclidean norm, A set of clusters $C = \{C_1, \ldots, C_q\}$, $\bar{C}$ is a set of all the possible *q*-partition of the set *X*, and a^j is the center of cluster C_j defined as follows: $a_j = \frac{1}{|C_j|}\sum_{x \in C_j} x$,

And $|C_j|$ is the cardinality of the set $C_j, j = 1, \ldots, q$. The clustering problem in equation (7) is referred to as the clustering problem with the minimal sum of squares.

Equation (7) can be rewritten as mixed variable problem, as shown below [8]:

$$\min \psi\ (A, w) = \frac{1}{n}\sum_{i=1}^{n}\sum_{j=1}^{q} w_{ij}\ \|a^j - x^i\|^2, \qquad (8)$$

Where $\sum_{j=1}^{q} w_{ij} = 1, i = 1, \ldots, n,$ (9)

And $w_{ij} \in \{0, 1\}, i = 1, \ldots, n, j = 1, \ldots, q.$

Here, $a^j = \frac{\sum_{i=1}^{n} w_{ij} x^j}{\sum_{i=1}^{n} w_{ij}}, j = 1, \ldots, q,$

And w_{ij} is the association weight of pattern x_i with cluster C_j, given by

$$wij = \begin{cases} 1, \textit{if the pattern i is allocated to cluster j} \\ \quad \forall i = 1, \ldots, n, j = 1, \ldots, q, \\ 0, \textit{otherwise} \end{cases}$$

The distance between centers and the dataset is calculated using the following equation.

$$D_M = \sum_{i=1}^{n}\sum_{j=1}^{q} w_{ij}\ \|a^j - x^i\|^2, \quad (10)$$

Furthermore, equations (7) and (10) can be reformulated as a mathematical programming problem, as follows.

$$\boldsymbol{min\ f(a^1, \ldots, a^q)}, \qquad (11)$$

Where $A = (a^1, \ldots, a^q) \in R^{d \times q}$, and $f(a^1, \ldots, a^q) = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{min}_{j=1,\ldots,q}\|a^2 - x^i\|^2$

In this paper, Equation (8) is considered as a mixed variable optimization problem. It is shown in [6] that equations (7), (8) and (10) are equivalent. *(n + d)* x *q* is the number of variables in equations (8) as in equation (10). It is *d* x *q* with the number of variables being independent of the number of instances. However, in the hard clustering problem, the coefficients w_{ij} vary form 0-1 variables, which means, equation (8) has both continuous and integer variables.

3. Adaptive Cellular Genetic Algorithm for Mixed Variable Optimization Problem

Individuals produced at random to rearrange on a 2-D toroidal grid constitute the ACGMV algorithm's implementation (achieved by encircling one another in the columns and rows). an asynchronous FLS type are chosen for updating the cells. In addition to, the selection of neighbors is depending to the quality of the individual. The updating in the grid is very important aspect. A small population P are formed from the selected individual and its neighbors in which the genetic operators operate (selection, crossover, and mutation) that allow exploration in every selected individual are used. In order to intensify the search for the best solutions, a pattern search is lastly added to the ACGMV approach to create ACGMV^P. The following are the components of the proposed method:

3.1. Initialization

The initial population is made up of individuals that are distributed uniformly within the search space that is restricted by [1, u].

3.2. Mapping on the Grid

Individuals are mapped from best to worst on a two-dimensional toroidal grid. In the case of a population of size μ, we set up in a $\sqrt{\boldsymbol{\mu}} \times \sqrt{\boldsymbol{\mu}}$ (assuming μ odd). in the case of a population of size μ. An individual is represented on the grid z = ($x_1, x_2, \ldots, x_{nx}, y_1, y_2, \ldots, y_{ny}$), where position (z) is at (i, j) on the toroidal grid[7].

3.3. Neighborhood

In *ACGMV*, Each individual chooses few a neighborhoods using different criteria as shown in Fig.(1). The neighborhoods are chosen in our strategy based on using L5 process for all individuals except for the best individuals, where the selection of neighborhoods is done via C9 process. The individual and its neighborhood together create a small population. Genetic operators are applied to this small population P.

3.4 Selection

The selection technique generates a med population, denoted *P'*, from the existing small population *P*. The more is the number of fit individuals in *P*, the higher is the probability of selection in *P'*. This procedure is repeated until p is complete. The probability for the population size u is determined as follows:

$$Pi = \frac{\boldsymbol{1}}{\boldsymbol{\mu}}\left(\zeta_{\max} - (\zeta_{\max} - (\zeta_{\min})\frac{\boldsymbol{i-1}}{\boldsymbol{\mu-1}}\right)\ i = 1,\ldots,\mu, \qquad (12)$$

Where $\zeta_{\max} + \zeta_{\min} = 2$ and $1 \leq \zeta_{\max} \leq 2$. Individuals in P' are chosen using the linear ranking selection mechanism [5] according to the steps outlined in Procedure 1.

Procedure 1 Linear Ranking Selection

1. *Put* $\boldsymbol{s_0 = 0}$.
2. $\boldsymbol{s_i = s_{i-1} + pi}$ *for all* $\boldsymbol{i = 1,\ldots,\mu}$.
3. $\boldsymbol{i = 1,\ldots,\mu}$ *do steps 4,5.*
4. *Generate a random number* $\gamma \in [0, s_i[$.
5. *Put* P_1 *in* $\boldsymbol{P'_i}$ *where* $\boldsymbol{s_{i-1} \leq \gamma \leq s_i}$.

3.5 Crossover

First, for each individual in P', the crossover procedure generates a number in the interval (0, 1) randomly. The individual is added to the pool known as the parent pool if the generated number is less than the crossover probability Pc. Next, every two parent's p^1 and p^2 are randomly selected from the parent pool. Two offsprings o^1 and o^2 is produced by an arithmetical crossover [20] occurs between the two selected parents using the following procedure 2:

Procedure 2 Arithmetical Crossover(p^1, p^2, o^1,o^2)

```
1. Generate a random number $\gamma \in (0, 1)$.
2. Let $p^1 = (x^1, y^1)$ and $p^2 = (x^2, y^2)$.
3. Calculate the recombined offspring $o^1 = (r^1,s^1)$ and $o^2$ $(r^2, s^2)$, where
       $r^1 = \gamma x^1 + (1 - \gamma) x^2$,
       $s^1 = [\gamma y^1 + (1 - \gamma) y^2]$,
       $r^2 = (1 - \gamma) x^1 + y x^2$,
       $s^2 = [(1 - \gamma)y^1 + \gamma y^2]$.
4. Return.
```

The function ⌈.⌉ represents the upper rounds of the elements, which is the nearest integer available to the discrete variable y.

3.6 Mutation

This operator replaces the gene's selected value from a uniform distribution with its lower and upper limits. For every gene on each chromosome (individual) and P', a random number between 0 and 1 is generated. If the produced number is less than the mutation probability Pm, the chromosome is mutated. Let and represent the selected gene and chromosome numbers, correspondingly. The mutated offspring z is then computed based on the following:

Procedure 3 Uniform Mutation

```
1. If the selected gene $\lambda$ is continuous, update $\boldsymbol{z^{\theta}}$ by setting $\boldsymbol{P^{\theta}_{\lambda}} = \boldsymbol{l_{x\lambda}} + \boldsymbol{\gamma}(\boldsymbol{u_{x\lambda}} - \boldsymbol{l_{x\lambda}})$.
2. If the selected gene $\lambda$ is integer, update $\boldsymbol{z^{\theta}}$ by setting $\boldsymbol{y^{\theta}_{\lambda}} = \boldsymbol{l_{y\lambda}} + \boldsymbol{\gamma}(\boldsymbol{u_{y\lambda}} - \boldsymbol{l_{x\lambda}})$,
   where $\gamma \in (0, 1)$.
```

3.7 Replacement (Update the Grid)

After the reproductive cycle, our current individual is derived from the highest quality solution in the small population P. (i.e, the selection, crossover, and mutation procedure).

3.8 Pattern Search

Pattern search method (PSM) is used to enhance the quality of some exceptional offspring. Particularly, some reformed variants of PSM are used to tackle the current problem. PSM's input values and parameters are reset to include both continuous and integer settings. Modifications are made to the PSM to reduce the cost function. The modified PSM then utilizes the set D in a distinct manner.

$$D = \{(\pm 1)^h e_1, \ldots\ldots (\pm 1)^h e_n\},$$

Where h is random number $h \in \{1, 2\}$. Therefore, every iteration of the modified PSM generates only $n + 1$ points.

Procedure 4 Pattern Search Process

1. *Select an x_0 as initial solution, set a step size $\Delta_0 > 0$ for for a positive spanning directions D, and set the counter number $k = 0$.*
2. *The grid G are generated around the initial solution. If there is an improvement in the grid, go to Step 4.*
3. *Compute the poll set PS_k. At each point in PS_k, evaluate the cost function.*
4. *If an improved individual obtained in Step 2 or 3, set x^{k+1} f equal to this improved point, and set $\Delta_{k+1} \geq \Delta_k$. Otherwise, set $x+ = x^k$, and $\Delta_{k+1} < \Delta_k$.*
5. *Stop if the termination conditions are met. Otherwise, set $k = k + 1$, and go to Step 2.*

3.9 ACGMV^P Algorithm

Figure 2 illustrates the ACGMV^P component.

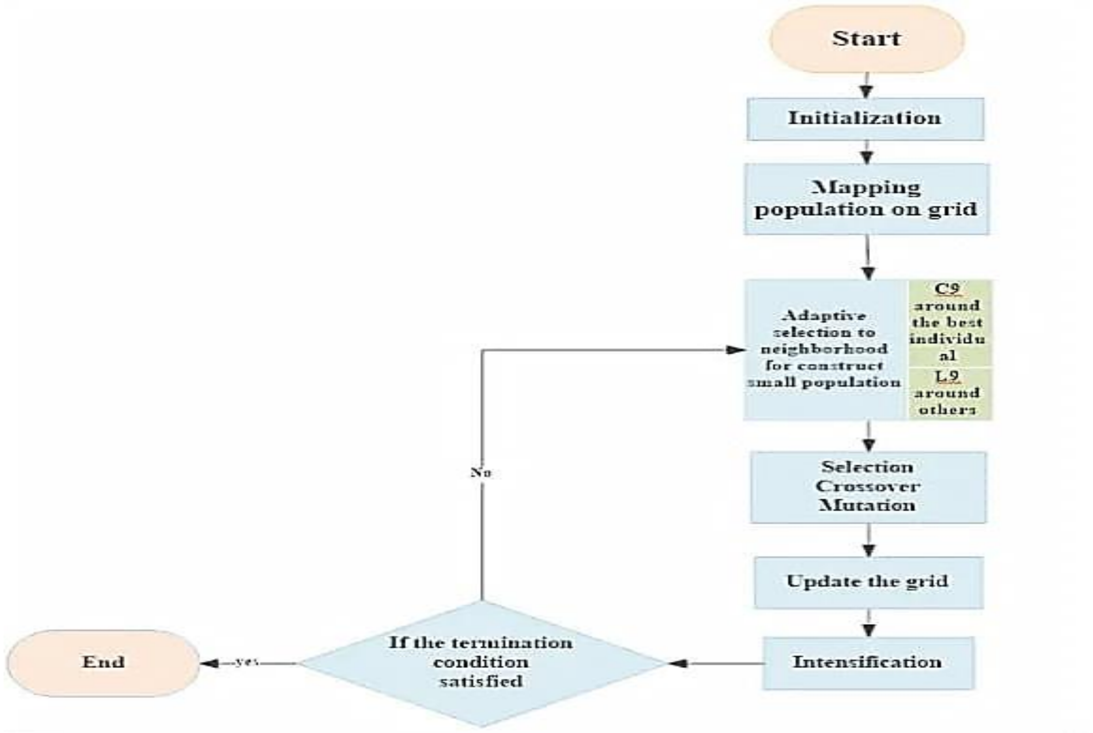
Start
Initialization
Mapping population on grid
Adaptive selection to neighborhood for construct small population
C9 around the best individual
L9 around others
Selection Crossover Mutation
Update the grid
Intensification
If the termination condition satisfied
No
yes
End

Fig.(2) The block diagram of *ACGMV^P* flowchart

4. Result and Discussion

The parameters setting for our proposed method will be discussed. The *ACGMV* and the *ACGMV^P* methods are coded in MATLAB. Our methods are applied to solve 9 benchmark unconstrained problems. f_1 to f_9 and 12 benchmark constrained problems g1 to g12, as shown in [19]. The properties of these problems are sufficiently diverse to encompass numerous categories of difficulty.

The mixed-integer variable in test problems from f_1 to f_9 are defined as $\boldsymbol{z} = (\boldsymbol{x_1}, \dots, \boldsymbol{x_{nx}}, \boldsymbol{y_1}, \dots, \boldsymbol{y_{ny}})$. During the execution, all test problems with 20 real variables ($\boldsymbol{n_x} = \boldsymbol{20}$)

and 20 integer variables ($\boldsymbol{n_y = 20}$) are considered. In addition, The penalization function *u(z, a, k, m)* are used for problems 6 through 9 which is defined accordingly.

$$u(z, a, k, m) = \begin{cases} \boldsymbol{k(z-a)}, & \boldsymbol{z > a}, \\ \boldsymbol{0}, & \boldsymbol{-a \le z \le a}, \\ \boldsymbol{k(-z-a)}, & \boldsymbol{z < a}, \end{cases} \qquad (13)$$

The *ACGMV* and the *ACGMV^P^* parameters are listed in Table 1. It covers the description of our algorithms indicated in the last section. Some parameters are defined in the literature according to their standard values. Others parameter values are obtained through preliminary numerical experiments.

The numerical results of our methods are presented in Table 2 through 50 independent runs with termination conditions to find a solution with an error of le-3 or when the maximum number of generations allowed is 200. The result indicates that the *ACGMV* takes more function evolution than *ACGMV^P* to find the optimal solution.

Furthermore, the *ACGMV* and *ACGMV^P* methods solve the unconstrained benchmark test problems as shown in Table 2. The numerical results of our methods obtained through 50 independent runs with the termination conditions to find a solution with an error of l*e*-3 or when the maximum number of generations reached 200, are discussed in Table 3. Moreover, the *ACGMV* method has the success rate 6.6667, 36.6667, 0.0000 and 3.3333 for g_4, g_6, g_9 and g11, respectively. The results of applying ACGMV method in which the success rate extents 100% for all the problems except when g_6 reaches 84%. It is worth to mention that *ACGMV* method is promising in finding the new global optimal values. However, it spends more function evolution to cover all parts of the search space. All other methods that maintain g_4. g_6 and g_9 have global value of 99.245, 4.5796, and 189.234, respectively, but *ACGMV^P* method obtained the global optima as 99.2399, 3.6203 and 0.01183, respectively.

Table (1): The Parameter Setting

Population size P_{size}	***5x5 (25 Individuals)***
The structure of neighborhood	***L5,C9***
Crossover Prob	***0.7***
Mutation	***Uniform***
Mutation. Prob	***0.05***
No of elite solution	***20***
Max-iter	***200***

Table(2): *Outcomes of ACGMV and ACGMV^P techniques for unconstrained* problems

F	*F**	*ACGMV*			*ACGMV^P*		
f	*f**	f_{mean}	*SR*	f_{eval}	f_{mean}	*SR*	f_{eval}
f_1	0	6.34le–4	100	7993	0.436e–4	100	6332
f_2	1	4.2526e6	0	45122	1.0008320	100	26889
f_3	0	168.525	0	46136	4.90e–4	100	2588
f_4	0	1.067	0	52660	0.020	74	46112
f_5	0	0.337	3.334	46111	4.668e–4	100	14845
f_6	0	8.409e–4	100	19590	5.035e–4	100	35166
f_7	0	0.0201	13.334	45600	4.897e–4	100	34655
f_8	0	0.055	0	46115	4.9017e–4	100	10988
f_9	0	0.4832	0	4609	3.5715e–4	100	4687

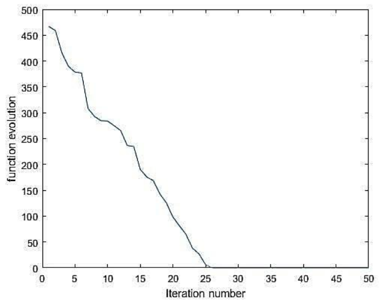

Fig.(3) The block diagram **of** the performance of *ACGMV^P* for the g_1 unconstrained problem

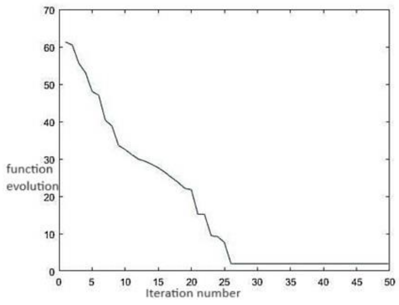

Fig.(4) The block diagram of the performance of *ACGMV^P* for the f_1 constrained problem

The comparisons between the *ACGMV^P* method and some other benchmark methods [23,19] tabulated in Table 4 and 5 along with that of the unconstrained problems.
Tables 4 and 5 present the comparison between the *ACGMV^P* method and the MIHDE/MIDE method [23] using functions from f_1 to f_9. The comparison was done with one run. First, Table 4 shows the comparison with a fixed number of evaluation function. It is clear that the *ACGMV^P* method has better solutions for f_1, f_4, f_5 and f_8 problems, whereas the MIHDE/MIDE method has a better solution for f_3 problem, and both have almost the same values for the remaining problems. Table 5 presents the second comparison under fixed function values. Moreover, the *ACGMV^P* method used a smaller number of evolution functions than MIHDE/MIDE method on f_1, f_3, f_5, f_6 and f_9 problems, whereas the MIHDE/MIDE method used a smaller number of evolution functions for f_4 and f_8 problems, and both have almost similar values for the remaining functions. Finally, the overall results demonstrate that the *ACGMV^P* method outperforms the MIHDE/MIDE method.
Three indicators are used for the comparison, including the mean of the function value, the number of function evaluations and the success rate to reflect the efficiency of the *ACGMV^P*, FGA [19], D-GCE [16] , Ms+m–HJ–f [11], and EFA[15] methods in Table 6 to Table 9. For fair comparisons, the termination criteria are specified to be identical to those described in

the methods. First, Table 6 shows the comparison between the $ACGMV^P$ and FGA methods [19]. In all the problems, the $ACGMV^P$ method provides competitive solutions with less number of evolution functions compared to the FGA method. Further, the FGA method cannot find a feasible solution in g_4 and g_6, but the $ACGMV^P$ method achieves new global optimal solutions for g_4 and g_6.

Table (3): Outcomes of ACGMV and ACGMVP techniques for constrained problems

G	*g**	*ACGMV*			$ACGMV^P$		
		g_{mean}	*SR*	g_{eval}	g_{mean}	*SR*	g_{eval}
g_1	2	2.0005	100	1316	2.000	100	201
g_2	2.124	2.124	100	2385	2.1245	100	1251
g_3	1.076574	1.0772	100	3156	1.0771	100	3151
g_4	99.2396	109.1805	6.6667	19766	**99.2399**	100	671
g_5	–6.66657	–6.666	100	1530	–6.6664	100	665
g_6	3.5578	3.628	36.6667	24222	**3.6203**	84	28340
g_7	–17	–16.999	100	4153	–16.999	100	435
g_8	32217.4	32206	96.6667	22246	32217	100	2597
g_9	0.01	7.8391	0	19755	**0.01183**	100	1088
g_{10}	–2.4444	–2.4443	100	164	–2.4444	100	156
g_{11}	3.2361	5.5790	3.3333	19544	3.2363	100	998
g_{12}	1.125	1.255	100	2635	1.125	100	238

Table (4): Outcomes of the comparison between the ACGMVP and MIHDE/MIDE methods

F	$Nf_{\text{-eval}}$	MIHDE/MIDE[23]	$ACGMV^P$
f_1	16591	9.1107e–11	5.3433e–12
f_3	35724	9.3095e–7	9.2068e–5
f_4	17411	8.49815e–7	2.6467e–10
f_5	33136	9.02949e–7	7.3281e–8
f_6	10901	7.667063e–7	9.1124e–7
f_7	10737	9.6735e–7	7.5604e–7
f_8	10764	8.5064e–7	8.2225e–8
f_9	10112	9.1497e–3	4.0210e–7

Table (5): Outcomes of the comparison of the $ACGMV^P$ and MIHDE/MIDE methods

F	*f – value*	MIHDE/MIDE[23]	$ACGMV^P$
f_1	9.1107e–11	16591	16463
f_3	9.3095e–7	35729	29014
f_4	8.49815e–7	17411	18785
f_5	9.02949e–7	33136	18530
f_6	7.667063e–7	10901	10268
f_7	9.6735e–7	10737	10770
f_8	8.5064e–7	10764	14381
f_9	9.1497e–7	10112	9806

Table (6): Outcomes of the comparison between the $ACGMV^P$ and FGA methods

G	*g**	$ACGMV^P$			FGA[19]		
		f_{eval}	Nf_{eval}	*SR*	f_{eval}	Nf_{eval}	*SR*

g_1	2	**2.000**	**202**	100	2.0005	440	100
g_2	2.124	2.1245	**1250**	100	2.1245	1769	100
g_3	1.076574	**1.0771**	**3141**	100	1.0772	3790	100
g_4	99.2396	**99.2399**	670	100	Infeasible	——	——
g_6	3.5578	**3.6203**	28339	84	Infeasible	——	——
g_7	–17	**-16,999**	**436**	100	-16.9995	793	100
g_8	32217.4	32217	**2597**	100	32217	6053	100
g_{10}	–2.4444	-2.4444	**156**	100	-2.4444	231	100
g_{11}	3.2361	**3.2363**	**998**	100	3.5087	2014	74
g_{12}	1.125	**1.125**	**239**	100	1.1256	428	100

Second, Table 7 shows the comparison between the *ACGMV*^{P} and Ms+m–HJ–f methods [11] on problems g_1, g_5, g_6 and g_9. That Ms+m–HJ–f method cannot find a feasible solution in g6 and go, whereas the *ACGMV*^{P} method can extend a global optimal solution in g_6 and g_9. Moreover, the *ACGMV*^{P} method has superior solutions with lower number of evolution functions than the D–GCE in g_1 except in g_5 where the number of evolution function is comparatively higher. Table 8 shows the comparison between Based on the analysis above, the ACGMV method obtains good results with lower number of evolution function than the D-GCE in all the problems except in g_8, where the number of function evaluations is low. The D–GCE methos cannot find a feasible solution in g_4 and g_6 but, the *ACGMV*^{P} method can reach a global solution. Finally, Table 9 presents the comparison between the *ACGMV*^{P} and EFA [15] methods on g_1, g_2, g_3, g_7, g_{10} and g_{11} problems, the ACGMV^P method obtains good results with the lowest number of evolution functions.

Table (7): Outcomes of the comparison between the *ACGMV*^{P} and Ms+m–HJ–f methods

G	G*	*ACGMV*^{P}			Ms[11]		
		g_{eval}	Ng_{eval}	SR	g_{eval}	Ng_{eval}	SR
g_1	2	**2.000**	**202**	100	2.0002495	458	100
g_5	-	**-6.6664**	667	100	6.66639367	**590**	100
g_6	6.66657	**3.6203**	28339	84	Infeasible	——	——
g_9	3.5578	**0.01183**	1088	100	Infeasible	——	——

Table (8): Outcomes of the comparison between the *ACGMV*^{P} and D–GCE methods

G	g*	*ACGMV*^{P}			D-GCE[16]		
		g_{eval}	Ng_{eval}	SR	g_{eval}	Ng_{eval}	SR
g_1	2	2.000	**202**	100	2.0000	3704	100
g_2	2.124	2.1245	**1250**	100	2.124992	1294	100
g_3	1.076574	1.0771	**3141**	100	1.076544	12416	100
g_4	99.2396	**99.2399**	670	100	Infeasible	——	——
g_6	3.5578	**3.6203**	28339	84	Infeasible	——	——
g_8	32217.4	32217	2597	100	32217	**609**	100

Table (9) : Outcomes of the Comparison between the *ACGMV*^{P} and EFA methods

G	g*	*ACGMV*^{P}			EFA[15]		

		g_{eval}	Ng_{eval}	SR	g_{eval}	Ng_{eval}	SR
g_1	2	2.000	**202**	100	2.0000	3409	100
g_2	2.124	2.1245	**1250**	100	2.7149	5253	80
g_3	1.076574	1.0771	**3141**	100	1.0767	5178	100
g_7	-17	**-16.999**	**435**	100	-16.998	3243	100
g_{10}	-2.44	**-2.4444**	**156**	100	-2.4380	3501	98
g_{11}	3.2361	**3.2363**	**998**	100	3.2361	4405	100

5. Adaptive Cellular Genetic Algorithm for Hard Clustering Problem

In this section, methodological components are presented.

5.1 Population Encoding

A population of solutions is composed of μ different weight matrices w^k of size $n \times q$, $\boldsymbol{k} = \mathbf{1}, \ldots, \boldsymbol{\mu}$. Each chromosome w^k can be coded in two different formats as follows.

- w^k is encoded into a vector v^k of size $n \times 1$. The entry j in v^k is set to be equal to the cluster number in which the pattern x^j belongs.
- w^k is also encoded into another array $A^k = (a^{1,k}, \ldots, a^{q,k})$, where $a^{j,k}$ is the center of the cluster C_j, j=1,...q, ie, a is computed using following equation:

$$a^{j,k} = \frac{\sum_{i=1}^{n} w_{ij}^{k} x^{j}}{\sum_{i=1}^{n} w_{ij}^{k}}, j = 1, \ldots, q. \quad (14)$$

It should be noted that the cellular genetic operators use the above mentioned encoding systems for simplicity. The function evolution of our hard cluster problem is presented in equation (8) as a MVO problem. Thus, the *ACGMV-HC* algorithm used all the steps in the ACGMV algorithm with some differences in dealing with the two formulae of population.

5.2 Computational Results

The *ACGMV-HC* algorithm is programmed in MATLAB and applied to solve three datasets;

- The first dataset of Bavarian postal zones includes 89 records with three attributes.
- The second Bavarian postal zones dataset is comparable to the first but includes four additional attributes: the number of self-employed individuals, civil servants, clerks, and manual laborers. Furthermore, there are 89 instances.
- The German towns database, which uses the Cartesian coordinates of 59 towns, has 59 records with two properties.

Table 9 presents the best-known global values. These values are given as $nf(x^*)$ where n is the number of instances and x^* is the global minimum point. These values are donated as f_{opt} and the solution found by the algorithm as f. The following formula is used to determine the error E reported for each algorithm:

$$E = \frac{\bar{f} - f_{opt}}{f_{opt}} \times 100 \quad (15)$$

The results in Tables 10 and 11 show the comparison between our *ACGMV-HC* method with other methods, like the k-means algorithm (K-M), the simulated annealing (SA), a genetic algorithm (GA), the tabu search (TS), and an optimization-clustering algorithm (Algorithm1). The results of the compared methods are taken from [4]. TS, GA, and SA have been applied to equation (8), which is equivalent to a clustering problem's nonsmooth optimization formulation. Moreover, Algorithm1 has been applied to equation (10).

The result for the first dataset of Bavarian postal zones is shown in Table 10. The result shows that the Algorithm1 method has the best result, whereas, the *ACGMV-HC* and GA methods have the same result. Table 11 presents the result for the second Bavarian postal zones dataset. The *ACGMV-HC* and algorithm methods have the same result. Table 12 presents the result for the German town's dataset, Based on the result, the *ACGMV-HC* method has the best result followed by TS and GA methods.

5.3.1 Wilcoxon signed-ranks test

A non-parametric technique called the Wilcoxon test is applied when testing a hypothesis using a two-sample design [17,18]. It is a comparative test that attempts to determine if there are significant differences in how two algorithms operate.

Table (10): Outcomes of the result for the first Bavarian postal zones dataset

The first Bavarian postal zone dataset				
Q f_{opt}	2 0.60255e12	3 0.29451e12	4 0.10447e12	5 0.59762e11
ACGMV-HC	0.00	23.48	0.00	0.00
The second Bavarion postal zone dataset				
Q f_{opt}	2 0.199080e11	3 0.17387e11	4 0.755908e10	5 0.540379e11
ACGMV-HC	144.28	0.00	0.00	0.00
The German towns dataset				
Q f_{opt}	2 0.12142e6	3 0.77009e5	4 0.49601e5	5 0.3953e5
ACGMV-HC	0.00	0.00	0.00	0.00

Table (11): Outcomes of the comparison between *ACGMV-HC* with other methods for first Bavarion postal zones dataset

Q f_{opt}	**2** **0.60255e12**	**3** **0.29451e12**	**4** **0.10447e12**	**5** **0.59762e11**
K-M	7.75	23.48	166.88	335.32
TS	0.00	23.48	18.14	33.35
GA	0.00	23.48	0.00	0.00
SA	0.00	23.48	0.39	40.32
Algorithm1	0.00	0.00	0.00	0.00
ACGMV-HC	0.00	23.48	0.00	0.00

Table (12): Outcomes of the comparison between *ACGMV-HC* and other methods for second Bavarion postal zones

Q / f_{opt}	**2** / **0.199080e11**	**3** / **0.29451e12**	**4** / **0.10447e12**	**5** / **0.59762e11**
K-M	144.25	106.79	303.67	446.13
TS	0.00	0.00	0.00	15.76
GA	144.25	0.00	0.00	15.76
SA	144.25	77.77	9.13	18.72
Algorithm1	144.25	0.00	0.00	0.00
ACGMV-HC	144.28	0.00	0.00	0.00

Table (13): Outcomes of the comparison between ACGMV-HC with other methods for German towns

Q / f_{opt}	**2** / **0.12142e6**	**3** / **0.77009e5**	**4** / **0.49601e5**	**5** / **0.3953e5**
K-M	0.00	1.45	0.55	2.75
TS	0.00	0.00	0.00	0.00
GA	0.00	0.00	0.00	0.00
SA	0.00	0.29	0.00	0.15
Algorithm1	0.00	0.29	0.00	0.15
ACGMV-HC	0.00	0.00	0.00	0.00

For tests computations, Let d_i be the comparison of two algorithms' performance scores on the *i*-th position of the *N* results. The distinctions are arranged in order of their absolute values; in the event of a tie, average ranks are assigned. Suppose *R*+ is the sum of ranks for functions on which the first algorithm outperforms the second, and *R*– is the sum of ranks in opposite to *R*+. The ranks of *di* = 0 are evenly split among the sums. If there is an odd number of them, one is ignored: The *p*-value associated with the comparison was determined using the Wilcoxon *T* statistic's normal T= min(R^+, R^-) is the smallest of the sums. (Table B.12 in [31]). The equal-means null hypothesis is rejected. approximation. (Section 6, Test 18 in [18]).

Table 14 presents Wilcoxon test results between our *ACGMV-HC* method and other methods K-M, TS, GA, SA and Algorithm1. The result shows that the *ACGMV-HC* method is better than the TS and SA methods.

Table (14): Wilcoxon test results

Compared Methods			**Solution Qualities**			
Method 1	*Method 2*		R^-	R^+	*p*-value	Best Method
C	K-M		73.5	4.5	0.0013	*ACGMV-HC*
ACGMV-HC	TS		48	30	0.4944	-
ACGMV-HC	GA		39.5	36.5	0.7449	-
ACGMV-HC	SA		68	10	0.0234	*ACGMV-HC*
ACGMV-HC	Algorithm 1		39	39	0.7760	-

6. Conclusions and Future work

ACGAMV^P method has been proposed to solve MVO problems. However, The *ACGAMV^P* method's adaptive neighborhood construction process concentrates more emphasis on the most successful individual. The computational results for 9 unconstrained and 12 constrained benchmark test problems show the suggested method outperforms the other existing methods. The suggested method achieved a new global point. Further, *ACGMV-HC* is used to solve hard clustering problems as an MVO problem. Based on our simulation using three datasets, the suggested method is more efficient. In future work, We can use our proposed method to solve real problems as MVO problems such as scheduling problems. Also, *ACGAMV^P* can be modified to deal with MVO with high dimensions and used to solve multi-objective functions.

References

1. N. A. AL-Madi, K. A. Maria and M. A. AL-Madi"A structured-population human community based genetic algorithm (HCBGA) in a comparison with both the standard genetic algorithm (SGA) and the cellular genetic algorithm (CGA)". ICIC Express Letters, 12(12), 1267-1275, 2018.
2. E. Alba & B. Dorronsoro. "Cellular genetic algorithms book. Springer Science & Business Media, 2018.
3. P. Alberto, F. Nogueira, H. Rocha &L. Vicente. "Pattern search methods for user-provided points: Application to molecular geometry problems". SIAM Journal on Optimization, 14(4) 1216-1236, 2004.
4. A. Bagirov & J. Yearwood." A new nonsmooth optimization algorithm for minimum sum-of-squares clustering problems". European journal of operational research, 170(2), 578-596, 2006.
5. T. Blickle & L. Thiele. "A comparison of selection schemes used in genetic algorithms", Evolutionary Computation 4.4 (1996): 361-394.
6. HH. Bock, HH. "Co-clustering for object by variable data matrices. Advanced studies in behaviormetrics and data science": essays in Honor of Akinori Okada (2020): 3-17.
7. O. Brudaru, A. Vilcu & D. Popovici. "Cellular Genetic Algorithm with Communicating Grids for a Delivery Problem". Symbolic and Numeric Algorithms for Scientific Computing (SYNASC), 2011 13th International Symposium on, 215-221, 2011.
8. M. Chakraborty & U.K. Chakraborty. "Branching process analysis of linear ranking and binary tournament selection in genetic algorithms". Journal of computing and information technology, 7(2), 107-113,2000.
9. Chebbi, O., Fatnassi, E., Chaouachi, J., & and Nouri, N.. Cellular Genetic Algorithm for Solving a Routing On- Demand Transit Problem. Genetic and Evolutionary Computation Conference, 301- 308, 2016.
10. Y. hiou & L. Lan ." Genetic clustering algorithms". European journal of operational research, 135(2), 413-427, 2001.
11. M. F. P.Costa, E. M. Fernandes and A. M. Rocha." Multiple solutions of mixed variable optimization by multistart Hooke and Jeeves filter method". Applied Mathematical Sciences, 8, 2163-2179, 2014.
12. B. Dorronsoro, & E. Alba. "A simple cellular genetic algorithm for continuous optimization. Evolutionary Computation", CEC 2006. IEEE Congress on, 2838-2844, 2006.
13. B. Dorronsoro, & P.Bouvry, " Adaptive neighborhoods for cellular genetic algorithms. Parallel and Distributed Processing". Workshops and Phd Forum (IPDPSW), IEEE International Symposium, 388-394, 2011.

14. L. Escudero, M. A. Garín, M. Merino & G. Pérez .” On BFC-MSMIP strategies for scenario cluster partitioning, and twin node family branching selection and bounding for multistage stochastic mixed integer programming”. Computers & Operations Research, 37(4) 738-753, 2010.
15. M. F. P. Costa, & F. P.Fernandes, “Extension of the firefly algorithm and preference rules for solving MINLP problems”. AIP Conference Proceedings. Vol. 1863. No. 1. AIP Publishing, 201٨.
16. Y. Gao, Y. Sun & J. Wu. “Difference-genetic co- evolutionary algorithm for nonlinear mixed integer programming problems”. Journal of Nonlinear Science and Its Applications, 9(3), 1261-1284, 2016.
17. S. García, A. Fernández, J. Luengo & F. Herrera. “A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability”. Soft Computing 13,959-977, 2009.
18. A. Hedar, & M. Fukushima. “Minimizing multimodal functions by simplex coding genetic algorithm”. Optimization Methods and Software, 18(3), 265-282, 2003.
19. A. Hedar & A. Fahim. “Filter-based genetic algorithm for mixed variable programming”. Numerical Algebra, Control and Optimization, 1(1), 99-116, 2011.
20. F. Herrera, M. Lozano, & J. L. Verdegay. “Tackling real-coded genetic algorithms: Operators and tools for behavioural analysis”, 12(4), 265-319,1998.
21. M. Laszlo & S. Mukherjee. “A genetic algorithm that exchanges neighboring centers for k-means clustering”. Pattern Recognition Letters, 28(16), 2359-2366, 2007.
22. Ma. Li, Li. Qianting ,Ma. Meiqiong & Lv Sicong. “Optimization and Application of Single- point Crossover and Multi-offspring Genetic Algorithm”. International Journal of Hybrid Information Technology, 9(1), 1-8, 2016.
23. Y. Lin, K. Hwang & F. Wang. “A mixed-coding scheme of evolutionary algorithms to solve mixed-integer nonlinear programming problems”. Soft ComputingComputers and Mathematics with Applications, 47(8), 1295-1307, 2004.
24. J. Liu, K. Teo, X. Wang & Ch. Wu. “An exact penalty function-based differential search algorithm for constrained global optimization”. Soft Computing, 20(4), 1305-1313, 2016.
25. S.Katoch, S.Chauhan & V. Kumar. “A review on genetic algorithm: past, present, and future. Multimedia Tools and Applications”, 80(5), 8091-8126, 2011.
26. P. Raju, Y. Subash, and K. Rishabh. “EEWC. energy-efficient weighted clustering method based on genetic algorithm for HWSNs”. Complex & Intelligent Systems, 6(2), 391-400, 2020.
27. Thomas Pogiatzis. “Application of mixed-integer programming in chemical engineering”. Ph.D. thesis at University of combridge , 2013.
28. B. Sağlam, F. Salman, S. Sayın, S. & M.Türkay.” A mixed-integer programming approach to the clustering problem with an application in customer segmentation”. European Journal of Operational Research, 173(3), 866-879, 2006.
29. L. Sahoo, and A. Banerjee, A. K. Bhunia & S. Chattopadhyay. “An efficient GA-PSO approach for solving mixed-integer nonlinear programming problem in reliability optimization”. Swarm and Evolutionary Computation, 19, 43-51, 2014.
30. B. Shahriari, M. Ravari, S.Yousefi & M. Tajdari,.” A Heuristic Algorithm Based on Line-up Competition and Generalized Pattern Search for Solving Integer and Mixed Integer Non-linear Optimization Problems”. Latin American Journal of Solids and Structures, 13(2), 224-242, 2016.
31. Zar, & JH. “Cyber deception: Virtual networks to defend insider reconnaissance. In Biostatistical Analysis”.. 4th ed.(Prentice-Hall: Englewood Cliffs, NJ.),2016.
32. Jianyun Zhang, Pei Liu, Zhe Zhou, Linwei Ma, Zheng Li, & Weidou Ni. “A mixed-integer nonlinear programming approach to the optimal design of heat network in a polygeneration energy system”. Applied Energy, Elsevier (pp. 146-154), 2014.