Soft Computing (2021) 25:389–405
https://doi.org/10.1007/s00500-020-05149-3

METHODOLOGIES AND APPLICATION

An approach for optimizing multi-objective problems using hybrid genetic algorithms

Ahmed Maghawry[1] · Rania Hodhod[2] · Yasser Omar[1] · Mohamed Kholief[1]

Published online: 4 July 2020

Abstract
Optimization problems can be found in many aspects of our lives. An optimization problem can be approached as searching problem where an algorithm is proposed to search for the value of one or more variables that minimizes or maximizes an optimization function depending on an optimization goal. Multi-objective optimization problems are also abundant in many aspects of our lives with various applications in different fields in applied science. To solve such problems, evolutionary algorithms have been utilized including genetic algorithms that can achieve decent search space exploration. Things became even harder for multi-objective optimization problems when the algorithm attempts to optimize more than one objective function. In this paper, we propose a hybrid genetic algorithm (HGA) that utilizes a genetic algorithm (GA) to perform a global search supported by the particle swarm optimization algorithm (PSO) to perform a local search. The proposed HGA achieved the concept of rehabilitation of rejected individuals. The proposed HGA was supported by a modified selection mechanism based on the K-means clustering algorithm that succeeded to restrict the selection process to promising solutions only and assured a balanced distribution of both the selected to survive and selected for rehabilitation individuals. The proposed algorithm was tested against 4 benchmark multi-objective optimization functions where it succeeded to achieve maximum balance between search space exploration and search space exploitation. The algorithm also succeeded in improving the HGA's overall performance by limiting the average number of iterations until convergence.

Keywords Genetic algorithms · Particle swarm optimization · Hybrid genetic algorithm · Multi-objective optimization

1 Introduction

1.1 Evolutionary-based algorithms

In computer science, an evolutionary-based algorithm (EA) is an artificial intelligence technique that targets global optimization by mimicking the biological process of evolution. EAs operate by utilizing operators driven from biological evolution such as breeding, crossover, mutation, and selection (Li et al. 2020; Nopiah et al. 2010). EAs are population based where each individual in an EA's population represents a possible solution to the optimization problem. The quality of a possible solution is determined by a fitness function that measures how good a candidate as a solution to the optimization problem. The evolution process in an EA commences by repeating the evolution operators mentioned above (Luo et al. 2020).

Communicated by V. Loia.

✉ Ahmed Maghawry
ahmed.mg.mohamed@gmail.com

Rania Hodhod
hodhod_rania@columbusstate.edu

Yasser Omar
dr_yaser_omar@yahoo.com

Mohamed Kholief
kholief@gmail.com

[1] College of Computing and Information Technology, Arab Academy for Science, Technology and Maritime Transport (AASTMT), Alexandria, Egypt

[2] TSYS School of Computer Science, Columbus State University, Columbus, GA, USA

1.2 Single- versus multi-objective optimization problems

In computer science, an optimization problem is a problem where the target is to find the best possible solution among all available solutions. In these problems, an algorithm traverses a search space to find the best possible solution. A single-objective optimization problem is a problem that contains one and only one optimization function. In such problems, an algorithm needs to focus only on this function and attempts to find the global minimum/maximum according to the target of optimization and the nature of the problem. On the other hand, a multi-objective optimization problem is a problem that contains more than one optimization function. In such problems, the algorithm needs to focus on more than one optimization function and traverses through the search space to find a solution or a set of solutions that achieve the optimization goal considering all the given optimization functions (Li et al. 2019; Luo et al. 2020).

There exist various single- and multi-objective mathematics-based optimization functions for testing; however, a real-life example of a single-objective optimization problem would be "an attempt to find the best car design that can achieve a very high speed; in this problem, the algorithm will focus only on finding the car design that when manufactured will produce a fast car regardless of any other feature." On the other hand, a real-life example of a multi-objective optimization problem would be "an attempt to find the best car design that when manufactured will produce a car that is fast, cheap, robust, light weight, and with high-quality materials." Obviously from this example, it is notoriously hard to design a car that is fast, robust with high-quality materials and at the same time cheap, which introduces the challenge of optimizing a multi-objective optimization problem especially when there are conflicting objectives.

The rest of the paper includes a background in the coming section, followed by challenges section that discusses the challenges facing this research and then a section for the proposed model architecture followed by the results then a section discussing the results and finally the conclusion.

2 Background

2.1 Genetic algorithms

A genetic algorithm (GA) is a search heuristic that mimics the process of natural evolution. In GAs, the fittest individuals are selected to produce the offspring of the new generation (Durairaj and Dhanavel 2018).

A genetic algorithm represents a mimetic technique that tackles optimization problems. A possible solution is referred to as a chromosome (individual) that consists of genes (features). Each gene describes one feature of the possible solution. The chromosome structure definition is a result of the problem encoding process where the algorithm implementer encodes the targeted problem in such a way that will enable the GA to attempt to solve this problem. A fitness function is a specific function that measures the fitness of a chromosome, in other words, how good a chromosome is as a possible solution to the optimization problem in hand. Once the problem encoding is done and the chromosome is structured, an initial population of randomly generated individuals is created (Kaur and Aggarwal 2013). After then, an iterative process takes place to evaluate the fitness of each chromosome and searches for the best solution (chromosome/individual) that will achieve the fitness function's goal; this is usually to find either a global minimum or a global maximum. Based on the fitness results, one or more individuals will be selected to survive and move to the next generation and the unselected individuals will be dismissed. After then, the survived group of individuals will start the mating process via crossover and mutation. Crossover takes place between two individuals where both individuals will share genes to form new individuals according to a previously defined probability (Pc). Then mutation follows for one or more genes with predefined probability of mutation (Pm). Once fitness evaluation, crossover and mutation steps are completed, a new generation (offspring) is now ready to replace the old population and become the main population where the iterative process will start all over again to create a new generation and so on until a termination criterion is reached

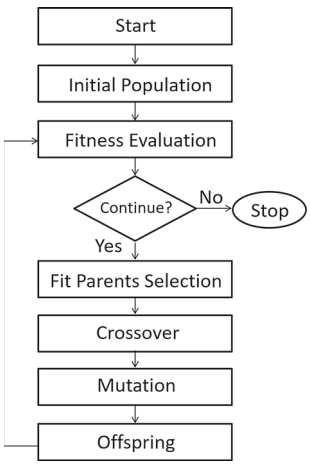

Fig. 1 Basic genetic algorithm flow

(Chen et al. 2018). The basic flow of a genetic algorithm is shown in Fig. 1.

2.2 Particle swarm optimization

In nature, members of bird flocks synchronously and precisely perform intelligent behavior without colliding with each other. Such interesting behavior has been studied in several researches (Heppener and Grenander 1990; Reynolds 1987). In computer science, the particle swarm optimization algorithm (PSO) has been developed as a result of the general belief that information sharing among members of a bird flock creates intelligent behavior. The particle swarm optimization algorithm belongs to the wide category of swarm intelligence techniques (Prado et al. 2010). PSO was proposed in 1995 (Rakitianskaia and Engelbrecht 2014) as an optimization method to simulate social behavior of swarms, since then PSO was successfully applied in a variety of optimization problems such as function optimization and training of neural networks (Rakitianskaia and Engelbrecht 2014). One of the PSO's greatest advantages is being computationally inexpensive as its system requirements are low (Prado et al. 2010). The PSO utilizes a population-based search technique to optimize a targeted objective function. The main component of PSO is the population in which the algorithm searches for the optimal solution. The population consists of particles where each particle is considered a possible solution. Particles in the population are a metaphor of birds in bird flocks or fish in fish pools. In PSO, particles are initialized with random values and can traverse the search space. During operation, each member of the swarm updates its own velocity and position depending on the best result reached so far by this member in addition to the best result reached by the entire swarm. The continuous updating methodology will drive all particles in the swarm toward the area in the search space that have the optimal result that is the global maximum/minimum according to the objective function. Initially, a population of swarm members is generated and randomly and initialized from a permissible range of values. Secondly, the velocity updating process takes place where all velocities of all swarm members are updated according to Eq. 1:

$$\vec{v}_{\mathrm{i}} = w\vec{v}_{\mathrm{i}} + c_1 R_1\left(\vec{p}_{\mathrm{i,best}} - \vec{p}_{\mathrm{i}}\right) + c_2 R_2\left(\vec{g}_{\mathrm{i,best}} - \vec{p}_{\mathrm{i}}\right) \quad (1)$$

where $\vec{p}_{\mathrm{i}}$ and $\vec{v}_{\mathrm{i}}$ represent the position and velocity of a particle i; $\vec{p}_{\mathrm{i,best}}$ represent the personal best of particle I and $\vec{g}_{\mathrm{i,best}}$ represent the best objective function value found so far by entire population; w represents a parameter that dominates the movement dynamics of a particle; R_1 and R_2 both represent random variables with permissible domain of [0, 1]; c_1 and c_2 both represent factors that dominate the weighting of the corresponding term. The existence of random variables grants PSO the ability to perform random searching, while c_1 and c_2 both represent weighting factors that compromise the trade-off between search space exploration and search space exploitation. As the updating process commences, $\vec{v}_{\mathrm{i}}$ is checked and maintained within a predefined domain to prevent stray random walking.

Then PSO updates the position of its member particles according to Eq. 2:

$$\vec{p}_{\mathrm{i}} = \vec{p}_{\mathrm{i}} + v_{\mathrm{i}} \quad (2)$$

Once the particles position is updated, $\vec{p}_{\mathrm{i}}$ should be checked and constrained to the permissible domain of values. Then the algorithm updates the saved personal best and global best $\vec{p}_{\mathrm{i,best}}$ and $\vec{g}_{\mathrm{i,best}}$ according to Eqs. 3 and 4:

$$\vec{p}_{\mathrm{i,best}} = \vec{p}_{\mathrm{i}} \; if \; f(\vec{p}_{\mathrm{i}}) > f\left(\vec{p}_{\mathrm{i,best}}\right) \quad (3)$$

$$\vec{g}_{\mathrm{i,best}} = \vec{g}_{\mathrm{i}} \; if \; f(\vec{g}_{\mathrm{i}}) > f\left(\vec{g}_{\mathrm{i,best}}\right) \quad (4)$$

where $f(\vec{x})$ represents the objective function targeted for optimization. Finally, the algorithm loops through from the second to the fourth step until a predefined termination condition is reached. For example, a predefined iterations limit or when there are no new results reached by the algorithm for a predefined number of generations. If a termination condition is met, the algorithm presents the values of $\vec{g}_{\mathrm{i,best}}$ and $f\left(\vec{g}_{\mathrm{i,best}}\right)$ as its final solution. Figure 2 presents the basic flow of the particle swarm optimization algorithm.

2.3 Genetic algorithms versus hybrid genetic algorithms

A genetic algorithm (GA) is a population-based metaheuristic search and optimization algorithm. It mimics the process of natural evolution in such a way that it utilizes the concepts of natural selection and genetic dynamics to solve search and optimization problems. The concept of

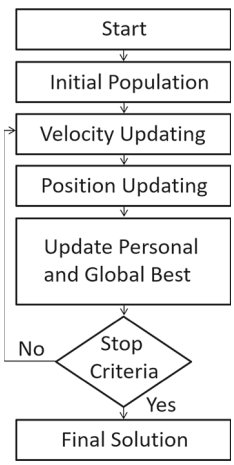

Fig. 2 Particle swarm optimization flow

genetic algorithms was first laid down by Holland (1975) and is discussed further with examples in De Jong (1975) and Goldberg (1989). In theory, GA's performance depends on the ability to optimally balance search space exploration and search space exploitation (Li et al. 2019). Realistically, problems arise because Holland assumed that the population size is infinite, and the fitness function accurately reflects the suitability of a solution and the interactions between the genes are very small (Beasley et al. 1993). In practice, the population size is finite which affects the sampling ability of the GA and its performance. Utilizing a local search method with GA (Hybridization) can help neutralize most of the obstacles that arise as a result of the finite population size, it also accounts for the genetic drift problem (Asoh and Mühlenbein 1994) by introducing new genes. It can also accelerate the search process to reach the global optimum (Hart 1994). The approaches in Goldberg (1999) have shown that hybridization has been one effective way to build competent genetic algorithms.

2.4 K-means clustering

The K-means clustering algorithm belongs to the partitioning-based and non-hierarchical clustering techniques (Abhishekkumar and Sadhana 2017), and it is one of the most used clustering techniques that has been applied in many scientific and technological fields (Xu and Wunschii 2005; Everitt et al. 2011). The k-means clustering algorithm is used commonly because of its applicability on different data types. The algorithm starts with a set of targeted numeric objects X and an integer number k. The algorithm then pursues an effort to partition all members of X into k clusters while minimizing the sum of squared errors (Hamerly and Drake 2014). Initially, the algorithm randomly initializes the k cluster centers; then the algorithm starts to assign each member of X to its closest center according to the square of the Euclidean distance from the cluster (Shrivastava et al. 2016). Consequently, the value of each center is updated by computing the mean value of each cluster; this updating process is a result of the change of membership of the cluster members (Lei 2008). The algorithm then iterates through updating cluster centers to membership reassigning until no more changes in the cluster's membership is achieved. To calculate how near a data vector is to a cluster's center, the following formula is used:

$$d(z_p, a_j) = \sqrt{\sum_{k=1}^{d} (z_{pk} - a_{jk})^2} \tag{5}$$

2.5 Multi-objective optimization test functions

Benchmark problems are usually utilized in order to evaluate the performance of optimization algorithms (Beasley et al. 1993). Using benchmark functions for this purpose facilitates performance comparison between different multi-objective optimization algorithms. In this research, several multi-objective optimization benchmark functions are used to evaluate the proposed algorithm.

(1) Binh and Korn function (1997).
Functions:

$$f_1(x, y) = 4x^2 + 4y^2 \tag{6}$$

$$f_2(x, y) = (x-5)^2 + (y-5)^2 \tag{7}$$

Constrains:

$$g_1(x, y) = (x-5)^2 + y^2 \leq 25 \tag{8}$$

$$g_2(x, y) = (x-8)^2 + (y+3)^2 \geq 7.7 \tag{9}$$

Search domain:

$$0 \leq x \leq 5.0 \leq y \leq 3 \tag{10}$$

(2) Chakong and Haimes function (1983).
Functions:

$$f_1(x, y) = 2 + (x-2)^2 + (y-1)^2 \tag{11}$$

$$f_2(x, y) = 9x + (y-1)^2 \tag{12}$$

Constrains:

$$g_1(x, y) = x^2 + y^2 \leq 225 \tag{13}$$

$$g_2(x, y) = x - 3y + 10 \leq 0 \tag{14}$$

Search domain:

$$-20 \leq x, y \leq 20 \tag{15}$$

(3) Constr-Ex Problem (Han et al. 2019).
Functions:

$$f_1(x, y) = x \tag{16}$$

$$f_2(x) = \frac{1+y}{x} \tag{17}$$

Constrains:

$$g_1(x, y) = y + 9x \geq 6 \tag{18}$$

$$g_2(x, y) = -y + 9x \geq 1 \tag{19}$$

Search domain:

$$0.1 \leq x \leq 1 \tag{20}$$

$$0 \leq y \leq 5 \tag{21}$$

(4) Poloni's two objective (1997).
Functions:

$$f_1(x,y) = \left[1 + (A_1 - B_1(x,y))^2 + (A_2 - B_2(x,y))^2\right] \quad (22)$$

$$f_2(x) = (x+3)^2 + (y+1)^2 \quad (23)$$

$$A_1 = 0.5\sin(1) - 2\cos(1) + \sin(2) - 1.5\cos(2) \quad (24)$$

$$A_2 = 1.5\sin(1) - \cos(1) + 2\sin(1) - 0.5\cos(2) \quad (25)$$

$$B_1(x,y) = 0.5\sin(x) - 2\cos(x) + \sin(y) - 1.5\cos(y) \quad (26)$$

$$B_2(x,y) = 1.5\sin(x) - \cos(x) + 2\sin(y) - 0.5\cos(y) \quad (27)$$

Search domain:

$$-\pi \le x, y \le \pi \quad (28)$$

3 Challenges

3.1 Genetic algorithms challenges

In theory, a genetic algorithm is supposed to achieve the perfect balance between search space exploration and search space exploitation. Search space exploration is a concept where the GA traverses through the search space looking for new solutions, while search space exploitation is another concept where the GA attempts to exploit possible opportunities to get the most out of the searching process. GA is supposed to achieve the perfect balance between search space exploration and search space exploitation as assumed in Beasley et al. (1993), such that "the population size is infinite and the fitness function accurately reflects the suitability of a solution and the gene interactions are minimum." In practice, the population size is finite which affects both the performance and the sampling ability of the genetic algorithm. On the other hand, the GA behavior is highly influenced by the fitness function that selects fit chromosomes to survive to the next generation, while rejecting chromosomes that do not pass the fitness function even if they have good genes. This is simply because the GA searches for good chromosomes not good genes. This behavior may punish individuals that may not pass the fitness function but may possess good genes that can take the search cursor to promising places in the search space.

3.2 The challenge of multi-objective optimization problems

Optimization algorithms have been used in a variety of fields including image processing (Chen et al. 2019; Zitzler and Kunzli 2004), industry (Li and Mcmahon 2007; Lin et al. 2016; Zhu and Zhou 2006), and manufacturing (Gui and Zhang 2016; Zhang et al. 2016). Optimization algorithms also pose a significant challenge in applied science (Kim et al. 2017; Ni et al. 2016; Tao and Zhang 2013), especially when the optimization algorithms are dealing with a multi-objective optimization problem (MOP) (Bandaru et al. 2014; Coello 2006) that contains two or more optimization objectives. The significant challenge posed to an optimization algorithm in such a case is that the algorithm has to synchronously consider all the objectives in the optimization process. MOP can formally be described as follows:

$$\text{Minimize} \quad F(x) = [f1(x), f2(x), \ldots, fm(x)]^{\mathrm{T}} \quad (29)$$

$$\mathbf{s.t.} \quad \boldsymbol{x} = (\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots, \boldsymbol{X}_n)^{\mathbf{T}} \in \boldsymbol{\Omega} \quad (30)$$

where $\boldsymbol{x}$ represents the decision variable vector and $\boldsymbol{\Omega}$ represents the search space and $\boldsymbol{Rm}$ represents the objective vector space. $\boldsymbol{F}(\boldsymbol{x})$ is the objective vector with $\boldsymbol{m}$ real value objective functions. In a multi-objective optimization problem, there is a relation between the two optimization functions which makes it difficult to a single point in the search space to minimize/maximize both objective functions at the same time. The approach to solve multi-objective optimization problems is to search for several promising points in the search space whose objective functions evaluation achieves a balanced minimum/maximum optimization value. Multi-objective evolutionary algorithms (MOEAs) utilize the evolution process to search for solutions to a multi-objective optimization problem. During this process, the algorithm performs many calculations as all individuals are evaluated in all generations. In some fields, the computation cost of an algorithm is critical. Therefore, it would be optimal to reduce the number of fitness evaluations and maximize the quality of solutions.

4 Proposed model architecture

4.1 Proposed algorithm

The proposed hybrid GA utilizes GA search for a set of optimal solutions that will minimize the objective functions for a given benchmark multi-objective optimization problem. The algorithm will also utilize the K-means clustering algorithm to support the selection process by ensuring a fair feature distribution in both selected to survive (fit) and selected for rehabilitation (non-fit) chromosomes. We assume that the non-fit chromosomes may contain good genes that may take the cursor of the searching process to places in the search space where promising results could be found. Accordingly, the non-fit chromosomes are passed to the particle swarm optimization algorithm for rehabilitation, where all selected for rehabilitation individuals (non-

fit chromosomes) will form the population of the PSO algorithm and communicate with each other to update their velocity and position to reach the best possible outcome from these non-fit individuals. The proposed model is as shown in Fig. 3.

In Fig. 3, condition one evaluates an individual and checks the predefined stopping criterion that is the maximum number of generations; if the maximum number of generations is reached, the algorithm will stop (F); otherwise, it will continue (T). Condition two can be considered as a dual selection mechanism, where on the one hand it selects the fittest individuals to survive to next generation (A). On the other hand, and with the support of the k-means algorithm, it splits the rejected individuals into k clusters where k is the number of optimization functions in a multi-objective optimization problem and fairly transfers a group of rejected individuals to the PSO for rehabilitation (R).

Condition 3 checks for the stopping criteria of the PSO that is a maximum number of iterations, such that, it will either continue looping through the PSO (F) or return the rehabilitated individuals into the GA's population (T). Condition 4 will either inject the incoming individuals into the new population of the GA (T) or force the GA to resume (F) if the GA has already stopped for an additional 5000 iterations.

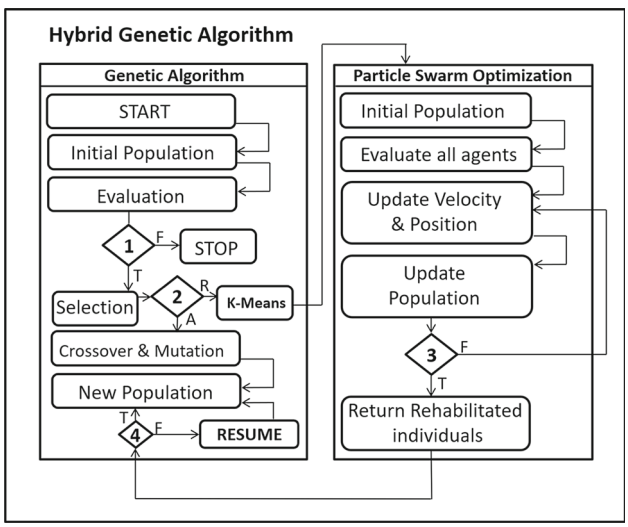

Fig. 3 Proposed model

Proposed algorithm (main part):
Input: BenchMarkFunction(X, Y) : The targeted benchmark function, IterMax: maximum number of iterations.
Output: List<Individual> ParetoSet.
Variables: Individual: Array of X and Y, GAPopulation: List of individual, Pm: probability of mutation, Pc: probability of crossover, i: individual#, j: iteration#.
Process:

```
Begin
    Initialize GAPopulation = GetRandomPopulation()
    Foreach (Individual INDi in GAPopulation)
        Calculate: BenchMarkFunction(INDi.X, INDi.Y)
    End Foreach
    Foreach (Individual INDi in GAPopulation)
        If(GetRandom() >= Pc)
            CrossOver(INDi)
        End If
        Foreach( Gene Gj in INDi)
            If(GetRandom() > Pm)
                Mutate(Gj)
            End If
        End Foreach
    Separate(GAPopulation, out Selected, out Rejected)
    Rehabiliatate = KMeansSelection(Rejected)
    OffSpring = Selected
    If(Online)
        NewThread(PSORehabilitation(Rehabiliatate))
    Else
        PSORehabilitation(Rehabiliatate)
    Output = GetParetoSet(OffSpring)
    End Foreach
    If(MaxGen)
        Return Output
    Else
        Continue
END
```

This is the first and main component of the proposed hybrid algorithm. GetRandomPopulation() is a function used to generate initial chromosomes with random values of X and Y.BenchMarkFunction() is a delegate consuming the targeted benchmak function. GetRandom() is a function used to get a random value to be compared against Pm and Pc. CrossOver(Individual) applies a fixed point crossover on a targeted individual. Mutate(Gene) applies mutation on a targeted gene. Separate(GAPopulation, out Selected, out rejected) scans the current population and outputs the selected individuals and the rejected individuals. KMeansSelection(Rejected) applies clustering where K = the chromosome length to assure fair distribution of the rejected individuals that are selected for rehabilitation. PSORehabilitation(Rehabilitate) applies the rehabilitation process according to the next algorithm either online or offline. GetParetoSet(OffSpring) gets the solution pareto set from the evaluated offspring.

Proposed algorithm (PSO):
Input: List<Individual> : rejected individuals from GA.
Output: Rehabilitated Individual.
Variables: V: Velocity, P: Position, i: individual#, j: iteration#.
GB: global best, PB: personal best.
Process:

```
Begin
        Initialize PSO.Population = Input
         Foreach (Individual INDi  in Input)
            Calculate: V(INDij) &   PB( INDij)
            If(PB(INDij) < GB)
              GB = INDij
            Else
             If(j =Iteration Limit)
              Output =  GB & break
             Else
              Calculate: P(INDij) & V(INDij)
              i++ & j++
              Continue
         End Foreach
        Return Output
END
```

The algorithm above describes the rehabilitation process using the PSO where output is represented as the global best achieved by all particles.

4.1.1 Problem encoding and solution decoding

The proposed algorithm operates on a set of benchmark multi-objective optimization problems with objective functions that require two inputs. Thus, the problem will be encoded in a chromosome structure consisting of two genes (one gene for each input) as shown in Fig. 4.

4.1.1.1 Encoding All four benchmark functions targeted in this research share the same characteristic of having two inputs X and Y and two objective functions F1 and F2. Both X and Y represent coordinates for a point in the search space with constrains, where the proposed algorithm is supposed to find the point with the minimum value of F1 and F2 in the search space of each targeted benchmark function. The proposed algorithm searches for X and Y and for each proposed X and Y we calculate the value of both objective functions F1 and F2. As a result, we encode a possible optimal solution of each targeted benchmark function in the form of a chromosome of X and Y as shown in Fig. 4.

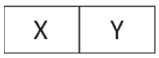

Fig. 4 Chromosome structure

4.1.1.2 Decoding As we target multi-objective optimization benchmark functions, it may not be of an obvious business value to attempt to translate the output of these benchmark functions to assume decoding the X and Y value; in the end, the encoded input of our targeted benchmark functions represents X and Y coordinates where the proposed algorithm is supposed to find the X and Y that will lead to the minimum value of F1 and F2, considering that these multi-objective benchmark functions were designed so that F1 and F2 are conflicting in such a way that generally minimizing F1 will maximize F2 and vice versa. This way, the targeted benchmark functions can test the multi-objective optimization ability of a proposed optimization algorithm. Decoding the chromosome of X and Y will lead to nothing but the X value and Y value that the proposed algorithm is searching for in each search space of the targeted benchmark functions.

4.1.2 Population specifications

A population of individuals is randomly generated to initialize the algorithm with a pre-determined population size n. In the execution phase, the algorithm is tested on different values of n to examine the effect of the population size on the algorithm's performance.

4.1.3 Genetic operators

The genetic operators are an essential part of the proposed algorithm as the algorithm utilizes them to mimic the process of natural evolution. Genetic operators include crossover, mutation, evaluation and selection. As soon as the problem is encoded properly, the algorithm can apply these operators on the individuals in the search for the best possible solution.

4.1.3.1 Crossover The chromosome structure is common in the entire test functions that are used in this research; hence, a single point crossover will be applied in all test cases so that chromosomes may share genes to facilitate the search for the optimal result.

4.1.3.2 Mutation The proposed algorithm can mutate the value of a gene in a chromosome according to the value of the probability of mutation Pm. However, some of the benchmark functions used in this research have a constrained search domain, and so in a test case that is subjected to such a function, the algorithm is permitted to mutate the genes within a range of the permissible values.

4.1.3.3 Evaluation In each generation, the algorithm evaluates all individuals to assign a fitness score for each one. In this research, the focus is on multi-objective

problems. Each of the used benchmark functions have two objective functions, so the fitness of each chromosome will be two values each of them represent one of the fitness functions.

4.1.3.4 Selection The algorithm performs elitism selection where a group of best performing chromosomes is selected to survive to the next generation. Only fitter chromosomes can replace these elite individuals in order; otherwise, this group continues through all generations unchanged. In addition to that, rejected individuals will be clustered by the K-means algorithm according to their fitness values. This step assures a balanced distribution of individuals passed to PSO for rehabilitation.

4.1.4 K-means clustering

The K-means clustering algorithm is utilized to support the individuals filtering process in the selection phase of the proposed algorithms. It can be viewed as a secondary selection technique that operates to gather all individuals that did not pass the fitness function, cluster them according to the values of their fitness functions that will lead to K = 2 clusters, and finally pass a balanced group of rejected individuals to the PSO to assure the existence of all unique individuals without losing an individual in an offspring that had no like in this offspring as shown in Fig. 5.

Figure 5 represents hypothetical individuals with hypothetical values to show the filtration mechanism of the K-means-based selection method. After individuals in a generation are rejected, their evaluation value will be targeted for clustering by the K-means algorithm to produce K clusters where K = the chromosome length. Then, based on the clustering result, the algorithm will select a group of individuals for rehabilitation while making sure to select at least one individual from each cluster, hence asserting the existence of all unique chromosomes. The PSO will then operate on them and return a better individual.

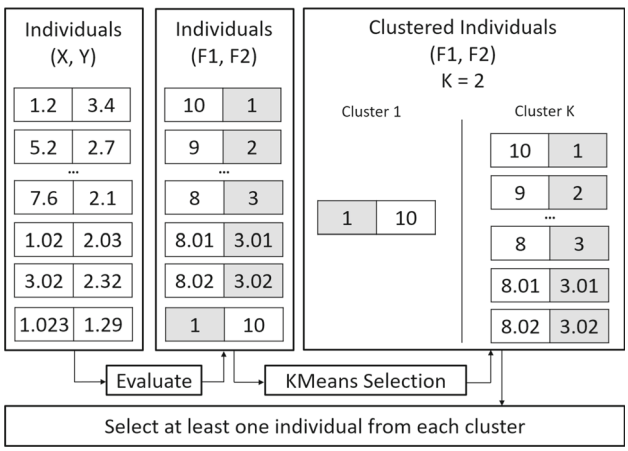

Fig. 5 K-means selection

4.1.5 Stopping criterion

The proposed algorithm's stopping criterion is when the algorithm reaches the maximum number of generations (in this research this number is fixed to 10,000).

4.1.6 PSO Integration

Individuals that did not pass the fitness function are clustered and passed to the PSO algorithm for rehabilitation. These individuals act as particles in a swarm where they all communicate with each other updating their velocity and position continuously. When all particles agree on the best solutions, this individual is passed back to the offspring replacing the least fit offspring in case the algorithm did not terminate. If the algorithm has already terminated, PSO will force it to continue and if no better result is reached for a predefined number of iterations (5000), the algorithm will finally terminate representing a group of solutions called a pareto set.

5 Results

The proposed algorithm is evaluated using four benchmark functions: Binh and Korn (1997), Chakong and Haimes (1983), Constr-Ex Problem (Han et al. 2019), and Poloni's two objective functions (Poloni 1997).

The algorithm was executed under 2 configurable parameters (Algorithm and PS) with two options each which produced 16 different test cases. The 16 test cases were executed on a 2.6 GHz Intel Core i7 vPro machine with 16 GB of RAM and a magnetic HDD on a 64-bit OS.

The test cases conducted in this research are described in Table 1 where the used algorithm is either the genetic algorithm (GA) or the proposed hybrid genetic algorithm (HGA)—PS specifies the used population size.

Test cases in Table 1 have been executed, and the detailed results are noted in Tables 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27 and 28. Each table(s) shows the pareto set extracted from each test case. The pareto set is a group of the best performing solutions to a multi-objective optimization problem. In our research the pareto set size is fixed to 7 solutions.

Figure 6 shows the average value of function 1 for Binh and Korn for the different cases.

Figure 7 shows the average value of function 2 for Binh and Korn for the different cases.

Table 1 Test cases

#	Benchmark Function	Algorithm	PS
1	Binh and Korn	GA	10
2	Binh and Korn	HGA	10
3	Binh and Korn	GA	100
4	Binh and Korn	HGA	100
5	Chakong and Haimes	GA	10
6	Chakong and Haimes	HGA	10
7	Chakong and Haimes	GA	100
8	Chakong and Haimes	HGA	100
9	Constr-Ex Problem	GA	10
10	Constr-Ex Problem	HGA	10
11	Constr-Ex Problem	GA	100
12	Constr-Ex Problem	HGA	100
13	Poloni's Two Objective	GA	10
14	Poloni's Two Objective	HGA	10
15	Poloni's Two Objective	GA	100
16	Poloni's Two Objective	HGA	100

Figure 8 shows the average value of function 1 for Chakong and Haimes for the different cases.

Figure 9 shows the average value of function 2 for Chakong and Haimes for the different cases.

Figure 10 shows the average value of function 1 for Constr-Ex Problem for the different cases.

Figure 11 shows the average value of function 2 for Constr-Ex Problem for the different cases.

Figure 12 shows the average value of function 1 for Poloni's Two Objective Function for the different cases.

Figure 13 shows the average value of function 2 for Poloni's Two Objective Function for the different cases.

Figures 14, 15, 16 and 17 show the average iterations until convergence for each benchmark function on each test case.

6 Discussion of results

The proposed HGA has been tested against four benchmark functions and compared against a normal genetic algorithm (GA) and a hybrid genetic algorithm (HGA) both in different population sizes (10/100) that produced a total of 16 test cases. All 4 multi-objective optimization functions used in this research have 2 objective functions F1 and F2, both have been targeted with a genetic algorithm and a population size of 10, a hybrid genetic algorithm and a population size of 10, a genetic algorithm and a population size of 100 and finally a hybrid genetic algorithm and a population size of 100.

6.1 Results analysis

Results of test cases 1, 2, 3 and 4 targeted the Binh and Korn optimization function F1 showed that the GA with population size 10 achieved good minimization value of F1, however when switched to HGA with population size 10, a better minimization value was achieved. As the population size increased to 100, the GA seemed not to

Table 2 Pareto set of test case 1

#	X	Y	F1	F2	ITR
1	0.134164947	0.01495685	0.072895762	48.52700597	9075
2	2.062855885	2.060158351	33.99850735	17.26948447	3917
3	0.920472136	2.77477923	34.18667491	21.59415507	4955
4	1.507794034	2.522610413	34.54802458	18.33296168	6184
5	2.370373636	1.762134252	34.89515318	17.39870942	0316
6	2.98028398	0.912911017	38.8619965	20.78354916	3903
7	4.960768472	2.522610413	123.8911485	6.138998278	7513

Table 3 Pareto set of test case 2

#	X	Y	F1	F2	ITR
1	0.043222038	0.002652104	0.007500713	49.54313376	531
2	1.210843376	2.872314101	38.8653199	18.8847552	616
3	2.405085765	1.983257128	38.87098549	15.83431744	131
4	3.063461782	0.577933328	38.87522008	23.30485393	307
5	3.074052638	0.521459137	38.88687701	23.7666015	843
6	1.231042953	2.865080728	38.89661731	18.76291752	147
7	4.697224519	2.921041917	122.3856162	4.413739701	356

Table 4 Pareto set of test case 3

#	X	Y	F1	F2	ITR
1	0.100979526	0.099351048	0.080269981	48.01676176	8186
2	0.641492475	1.968689939	17.14901069	28.18542853	9493
3	0.547274449	2.062328761	18.21083697	28.45667714	4770
4	2.425644352	0.205410998	23.70377681	29.6153907	1596
5	2.461141712	0.35049445	24.72025955	28.06370326	9916
6	2.512867457	1.687422256	36.64758689	17.1589996	4678
7	4.943641939	2.959672433	132.7970261	4.166112812	148

Table 5 Pareto set of test case 4 (input)

#	X	Y
1	0.025007743	0.000998159
2	2.429877297	0.247642491
3	2.241525288	1.205268405
4	2.52379106	0.43975363
5	2.714824873	1.806695914
6	2.431597976	2.247782434
7	4.989237397	2.991881041

Table 6 Pareto set of test case 4 (output)

#	F1	F2	ITR
1	0.002505534	49.74056736	906
2	23.86252193	29.1904326	951
3	25.90843018	22.00917062	876
4	26.25161828	26.92745767	292
5	42.53769687	15.41921635	773
6	43.86077836	14.17139048	588
7	135.3753679	4.032657586	212

Table 7 Pareto set of test case 5(input)

#	X	Y
1	0.790735223	3.669498192
2	5.555006836	5.304598778
3	5.512129993	5.781028082
4	5.731513102	6.22127259
5	1.832623739	10.10870305
6	5.37444378	11.65531614
7	0.881684907	14.78213384

take advantage of the large population size; on the other hand the HGA with population size 100 succeeded to achieve the least possible minimization value.

The same test cases also targeted Binh and Korn's F2 where the GA with population size 10 achieved a good minimization value that decreased when switched to HGA. However, in F2, the GA seemed to benefit from the

Table 8 Pareto set of test case 5 (output)

#	F1	F2	ITR
1	10.5885419	0.009603592	5422
2	33.16764425	31.46549089	4588
3	37.19328661	26.75094041	5983
4	43.18587749	24.32193047	7278
5	84.99648613	−66.47485767	5457
6	126.9226329	−65.16576801	5693
7	193.1978417	−182.0120489	8964

Table 9 Pareto set of test case 6(input)

#	X	Y
1	0.790735223	3.669498192
2	5.555006836	5.304598778
3	5.512129993	5.781028082
4	5.731513102	6.22127259
5	1.832623739	10.10870305
6	5.37444378	11.65531614
7	0.881684907	14.78213384

Table 10 Pareto set of test case 6 (output)

#	F1	F2	ITR
1	10.5885419	0.009603592	5422
2	33.16764425	31.46549089	4588
3	37.19328661	26.75094041	5983
4	43.18587749	24.32193047	7278
5	84.99648613	−66.47485767	5457
6	126.9226329	−65.16576801	5693
7	193.1978417	−182.0120489	8964

Table 11 Pareto set of test case 7 (input)

#	X	Y
1	1.132622967	3.728393034
2	0.638726615	5.898149896
3	5.245617938	5.232664531
4	6.356111423	6.522970724
5	8.20155214	7.389476242
6	5.893180657	13.70068807
7	0.476608146	14.98233248

Table 12 Pareto set of test case 7 (output)

#	F1	F2	ITR
1	10.19647146	2.749478156	556
2	27.84493764	−18.24333287	218
3	30.44948483	29.29511241	491
4	51.47891235	26.70179719	2
5	81.28465559	32.98856261	887
6	178.4643332	−108.2688516	627
7	199.8263443	−191.2161482	293

Table 13 Pareto set of test case 8 (input)

#	X	Y
1	1.018223176	3.69941645
2	2.832960716	5.692491758
3	4.586781936	5.059838949
4	5.071999512	5.656455888
5	0.279482226	12.51596106
6	0.391765502	13.55972229
7	0.14079175	14.90785348

Table 14 Pareto set of test case 8 (output)

#	F1	F2	ITR
1	10.2507349	1.877159415	9470
2	24.71330245	3.477167549	7929
3	25.17373308	24.79874513	9968
4	33.11976244	23.96541417	6923
5	137.5775405	−130.1020191	5810
6	162.3330421	−154.2207344	9712
7	198.8850439	−192.1612628	2343

increase in the population size as it decreased the minimization value and finally achieved the best result when using HGA with a population size of 100.

Results of test cases 5, 6, 7 and 8 targeted the Chakong and Haimes optimization function F1 and showed that the GA with population size 10 achieved fair minimization value of F1, switching to HGA with population size of 10 individuals achieved a much better minimization value. Increasing the population size to 100 seemed to disrupt the GA, however switching to HGA with population size 100 succeeded to achieve the least possible minimization value. The same test cases also targeted Chakong and Haimes F2 where the GA with population size 10 achieved a good minimization value that decreased when switched to HGA. In F2, the GA seemed also to benefit from the increase in the population size that decreased the minimization value and finally the best possible minimization result was achieved using HGA with a population size of 100.

Results of test cases 9, 10, 11 and 12 targeted the Constr-Ex Problem optimization function F1 where a serial decrease in the minimization value has been witnessed when switching from GA to HGA and from population size of 10 to population size of 100. The same phenomenon has been also witnessed in F2 with a serial decrease in the minimization value when switching from GA to HGA and from population size 10 to 100.

Results of test cases 13, 14, 15 and 16 targeted the Poloni's Two Objective optimization function F1 and showed a smooth decrease in the minimization value when switching from GA to HGA and from population size 10 to population size 100. The same test cases also targeted Poloni's Two Objective F2 where almost the same phenomenon has been witnessed except for a slight increase in the minimization value when used a GA with population size 100.

All test cases on all multi-objective optimization functions have shared the same phenomenon in terms of average iterations until convergence where GA with population size 10 has consumed the most average iterations until convergence, while the average iterations until convergence have slightly decreased on switching to population size 100. On the other hand, a great decrease in the average iterations until convergence has been witnessed when HGA was used and decreased more on switching population size from 10 to 100.

6.2 Complexity

The proposed hybrid algorithm utilizes two evolutionary algorithms as well as the K-means clustering algorithm with overall 5 loop structures as assumed below:

(n): GA maximum number of iterations.

Table 15 Pareto set of test case 9

#	X	Y	F1	F2	ITR
1	1.141650528	3.714503396	10.1052925	2.906326066	170
2	2.270679391	4.1116808	11.75582473	10.75355712	682
3	4.777027361	4.931482926	25.16843897	27.53668825	271
4	4.037882334	5.780501252	29.00615663	13.48774878	399
5	6.166195882	5.574110591	40.27967582	34.57327524	478
6	0.400150437	12.96051534	147.6134458	−139.4525733	112
7	0.030603712	14.98981249	201.5933752	−195.4394201	381

Table 16 Pareto set of test case 10

#	X	Y	F1	F2	ITR
1	0.480483617	2.52409605	0.480483617	7.334477026	195
2	0.672769379	4.780821276	0.672769379	8.592574893	868
3	0.714293328	0.209769169	0.714293328	1.693658785	100
4	0.831604894	0.201440994	0.831604894	1.444725738	937
5	0.831604894	0.117248981	0.831604894	1.343485338	871
6	0.831604894	1.410850953	0.831604894	2.89903411	522
7	0.884108642	0.179379825	0.884108642	1.33397613	770

Table 17 Pareto set of test case 11

#	X	Y	F1	F2	ITR
1	0.424663285	2.327685259	0.424663285	7.836055946	215
2	0.811333275	4.643209746	0.811333275	6.955476771	826
3	0.65629574	0.487813009	0.65629574	2.266985625	1
4	0.766314925	0.022275313	0.766314925	1.334014619	572
5	0.811333275	0.022275313	0.811333275	1.259994314	621
6	0.965438728	0.148130385	0.965438728	1.189231747	505
7	0.965438728	0.022275313	0.965438728	1.05887125	130

Table 18 Pareto set of test case 12

#	X	Y	F1	F2	ITR
1	0.409157113	2.368311669	0.409157113	8.232318488	930
2	0.543566211	2.830027301	0.543566211	7.04610997	1
3	0.688640024	0.424545282	0.688640024	2.068635618	620
4	0.659018699	0.068861651	0.659018699	1.621898821	630
5	0.739009283	0.154299075	0.739009283	1.56195477	234
6	0.852115546	0.665951977	0.852115546	1.955077554	955
7	0.931696622	0.100354292	0.931696622	1.18102209	868

(x): GA Population size.
(y): K-Means maximum number of iterations.

Table 19 Pareto set of test case 13 (input)

#	X	Y
1	0.399905839	2.520565424
2	0.450174571	2.221508763
3	0.835870807	0.400861474
4	0.807107902	0.400861474
5	0.839081845	0.136104268
6	0.890787528	0.036718799
7	0.991339619	0.005479991

Table 20 Pareto set of test case 13 (output)

#	F1	F2	ITR
1	0.399905839	8.803485919	115
2	0.450174571	7.15613224	775
3	0.835870807	1.675930612	727
4	0.807107902	1.735655755	521
5	0.839081845	1.353985044	25
6	0.890787528	1.163822759	30
7	0.991339619	1.014263903	658

Table 21 Pareto set of test case 14(input)

#	X	Y
1	2.026772379	0.725566987
2	2.026772379	0.404946227
3	1.060018475	1.689872468
4	0.687651732	1.354777902
5	0.026858216	1.115714894
6	0.192344673	0.84673096
7	0.026858216	1.009679782

(z): PSO maximum number of iterations.
(m): PSO population size.

The combined complexity will be as follows:

$$\boldsymbol{O}(\boldsymbol{n}) \times \left(\boldsymbol{O}(\boldsymbol{x}) + \boldsymbol{O}\left(\boldsymbol{y}^2\right)\right) + \boldsymbol{O}(\boldsymbol{z}) \times \boldsymbol{O}(\boldsymbol{m}) \tag{31}$$

That can be further calculated as:

Table 22 Pareto set of test case 14(output)

#	F1	F2	ITR
1	1.0000782200791	28.2460219774499	459
2	1.6804261129277	27.2423144500805	6546
3	1.0731068334506	23.7191639078846	4849
4	3.6149691740080	19.1437542669727	1781
5	12.4096393589383	13.6381201742824	2767
6	15.0963843208604	13.6014797502701	1387
7	14.1033896465813	13.2006834859074	1824

Table 23 Pareto set of test case 15(input)

#	X	Y
1	1.992796167	0.74599444
2	0.431708764	2.560063858
3	0.414587958	1.580209137
4	0.033285138	1.015161588
5	0.244105733	0.286065834
6	0.129004237	0.175458435
7	0.033285138	0.04120158

Table 24 Pareto set of test case 15 (output)

#	F1	F2	ITR
1	1.0008368469679	27.9765101466281	427
2	3.7484427646468	24.4506797097984	431
3	4.3293861683622	18.3168901121083	289
4	13.9461945091865	13.2616949567807	143
5	26.5294904325897	12.1781873364260	6
6	31.3897481027259	11.1723700495869	998
7	36.5414105010677	10.2849194605944	67

Table 25 Pareto set of test case 16 (input)

#	X	Y
1	2.014379037	0.730414493
2	1.516069487	0.983394203
3	0.786375781	0.739979921
4	0.073302244	1.000250057
5	0.107229003	0.730414493
6	0.048561879	0.440015081
7	0.042114537	0.025438154

$$\boldsymbol{O(nx) + O\left(ny^2\right) + O(zm)} \tag{32}$$

where generally $O(nx)$ is observed as the GA's maximum number of iterations by the genetic operators process, while $O(ny^2)$ is observed as the GA's maximum number of

Table 26 pareto set of test case 16 (output)

#	F1	F2	ITR
1	1.00019889940196	28.13833144168340	8422
2	1.23976085602110	24.32873617242680	3219
3	8.59745451825322	17.36417168097560	6913
4	13.78220356963130	13.44618697211410	9068
5	18.48631603556300	12.64920639464410	298
6	26.00059998200200	11.36737295952180	7408
7	36.79061228187800	10.30598426556100	2662

Table 27 Pareto set of test case 17 (input)

#	X	Y
1	0.997302462811257	1.99525802954811
2	0.938561079529375	1.83597606692276
3	0.491012027715804	1.34152916322533
4	0.163117828854880	0.941434032722113
5	0.289071682043873	0.407450404207898
6	0.0954794325379093	0.255250814489671
7	2.86696013196696E3	1.13007891975813E2

Table 28 Pareto set of test case 17 (output)

#	F1	F2	ITR
1	1.00010284546762	24.9499976427693	871
2	1.0911184293893	23.5550236293423	58
3	5.31434717696346	17.6699237998912	479
4	13.7736062389852	13.7744805026313	437
5	22.8491292496355	12.7989091699279	707
6	29.9417735169670	11.1576475245422	1
7	37.8388695531307	10.0399392664838	105

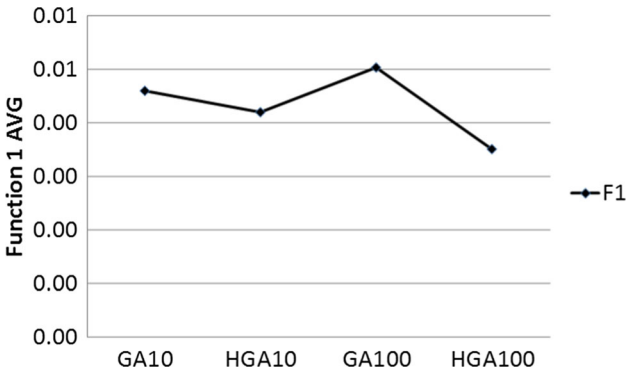

Fig. 6 Binh and Korn F1 average

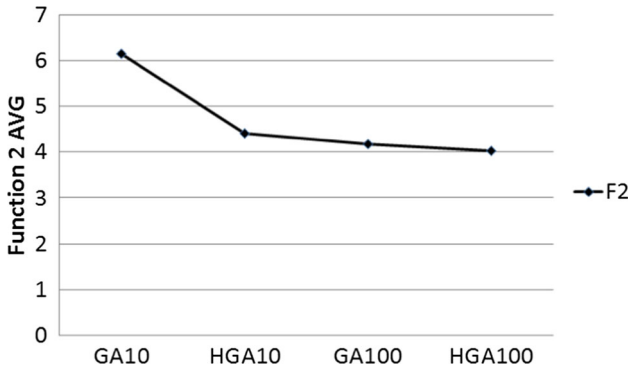

Fig. 7 Binh and Korn F2 average

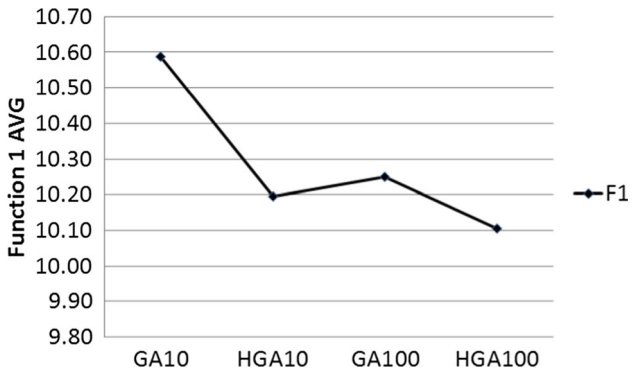

Fig. 8 Chakong and Haimes F1 average

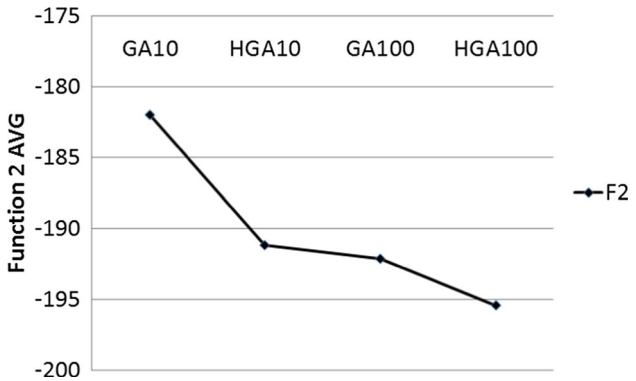

Fig. 9 Chakong and Haimes F2 average

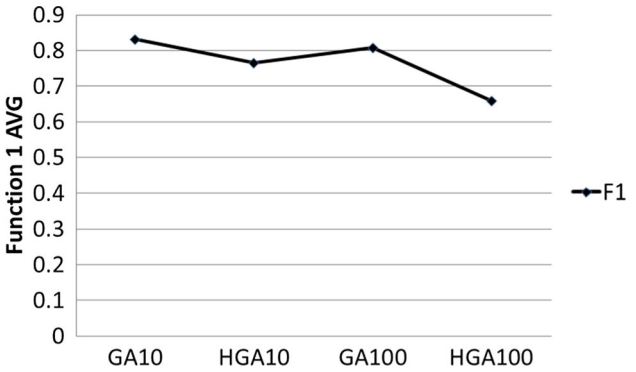

Fig. 10 Constr-Ex F1 average

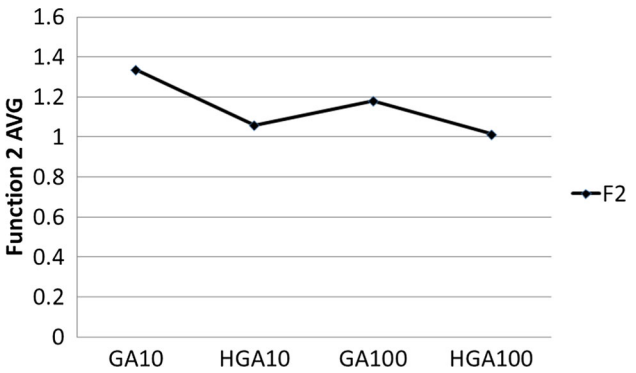

Fig. 11 Constr-Ex F2 average

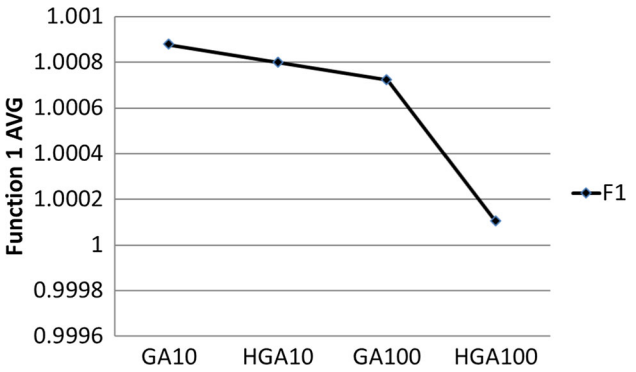

Fig. 12 Poloni's F2 average

generations by the K-means process and finally $O(zm)$ is observed as the PSO's population size by the PSO's velocity and position process. Hence, we can conclude that better optimization was achieved with a trade-off in performance and resource consumption.

6.3 Contributions

The proposed technique utilizes a genetic algorithm that targets search space exploration supported by the K-means algorithm to enhance the selection mechanism as described in Sect. 4.1.4. The particle swarm optimization algorithm was also utilized to target the rejected individuals of each generation to fulfill the concept of the rehabilitation of rejected individuals to maximize the utilization of all individuals in each generation. To test the effect of each component of the proposed hybrid algorithm, the hybrid algorithm was tested against 4 benchmark functions under several configurations that resulted in 16 different test cases where their results were finely analyzed in Sect. 6.1.

From this analysis, it was observed that the proposed K-means-based selection mechanism enhanced the optimization ability of the genetic algorithm, however added

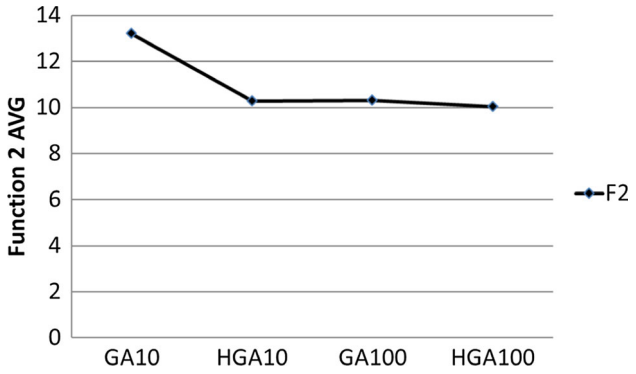

Fig. 13 Poloni's F2 average

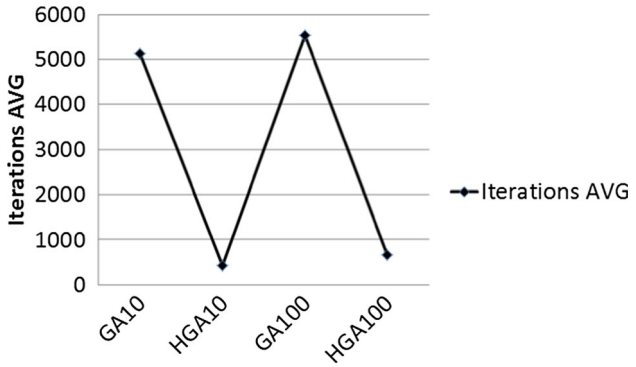

Fig. 14 Binh and Korn Iterations AVG

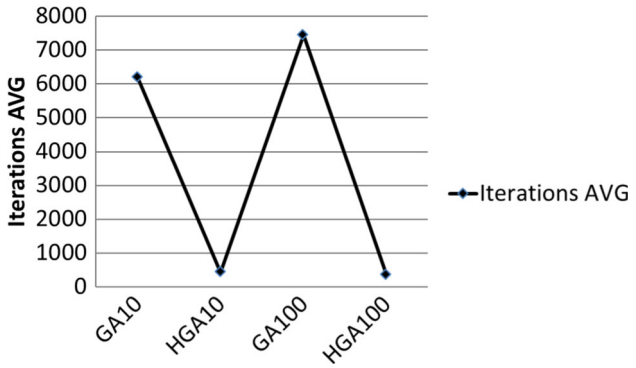

Fig. 15 Chakong and Haimes Iterations AVG

more computational cost to the GA. It was also observed that the PSO further enhanced the optimization ability of the hybrid algorithm, however, also added to the computational complexity of the hybrid algorithm.

From Sect. 6.2 it was also observed that the overall optimization process is enhanced significantly, however, and as a trade-off to this enhancement, more computational cost was added compared to a basic GA.

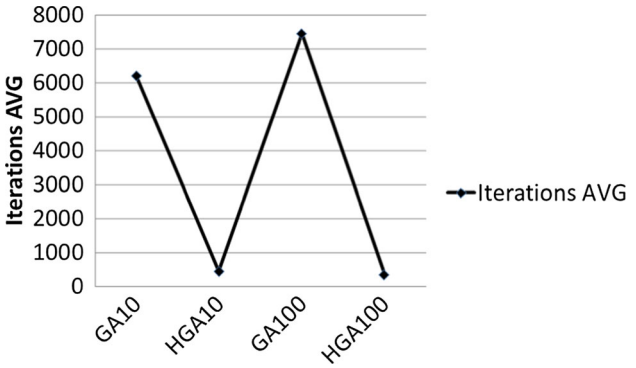

Fig. 16 Constr-Ex Iterations AVG

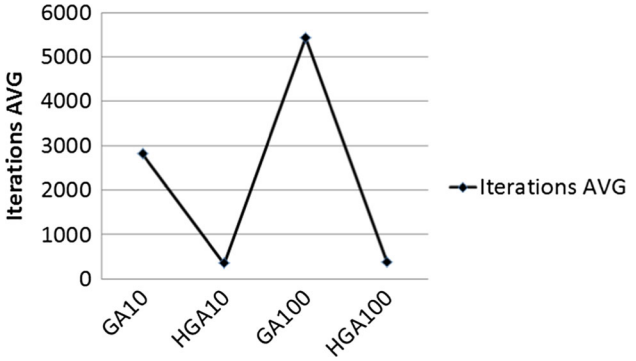

Fig. 17 Poloni's Iterations AVG

7 Conclusion

In this research, a hybrid genetic algorithm was proposed to solve multi-objective optimization problems. The hybrid genetic algorithm utilized the particle swarm optimization (PSO) as well as the K-means algorithm in order to solve multi-objective optimization problems. In this research, four benchmark multi-objective optimization problems have been used to test the proposed hybrid genetic algorithm (HGA). The three main components of the proposed hybrid algorithm (GA, PSO and K-means) have been utilized to achieve better optimization results as well as performance. In concept, the genetic algorithm was used to achieve search space exploration supported by the K-means algorithm to enhance the selection operation of the GA, while the PSO was used to achieve search space exploration.

In the experiments phase of this research, these concepts have been put to test on four benchmark multi-objective optimization functions with different settings in terms of population size (10 or 100) and the algorithm mode (GA or HGA) which produced 16 different test cases (four benchmark functions x four different settings) as shown in Table 1. All 16 test cases were executed, and the results were noted in Tables 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14,

15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27 and 28. The results were discussed in detail in the previous section from which we can conclude that the proposed HGA has achieved better optimization results in terms of minimizing the objective functions in all test cases as well as better performance in terms of average iterations until convergence. For each benchmark function, better results in terms of minimizing the objective functions were achieved when the algorithm was switched from GA to HGA.

It was noticed that increasing the population size enhanced the minimization ability of both the GA compared to itself and HGA compared to itself but gave no superiority to GA over HGA even with small population size. Better performance was also achieved as the proposed HGA has significantly decreased the average iterations needed until convergence when compared to GA.

Compliance with ethical standards

Conflict of interest Author Ahmed Maghawry declares that he has no conflict of interest. Author Rania Hodhod declares that she has no conflict of interest. Author Yasser Omar declares that he has no conflict of interest. Author Mohamed Kholief declares that he has no conflict of interest.

Human and animal rights This article does not contain any studies with human participants or animals performed by any of the authors.

References

Abhishekkumar K, Sadhana C (2017) Survey report on K-means clustering algorithm. Int J Mod Trends Eng Res 4:218–221. https://doi.org/10.21884/ijmter.2017.4143.lgjzd

Asoh H, Mühlenbein H (1994) On the mean convergence time of evolutionary algorithms without selection and mutation. In: Davidor Y, Schwefel H-P, Manner R (eds) Parallel problem solving from nature, PPSN III. Springer, Berlin, pp 88–97

Bandaru S, Ng A, Deb K (2014) On the performance of classification algorithms for learning pareto-dominance relations. In: Evolutionary computation (CEC), 2014. IEEE, pp 1139–1146

Beasley D, Bull DR, Martin R (1993) An overview of genetic algorithms: part 1, fundamentals. Univ Comput 15:58–69

Binh T, Korn U (1997) MOBES: a multiobjective evolution strategy for constrained optimization problems. In: Proceedings of the third international conference on genetic algorithms, Czech Republic (1997), pp 176–182

Chankong V, Haimes YY (1983) Multiobjective decision making: theory and methodology. North Holland, New York

Chen J, Zhang D, Liu D, Pan Z (2018) A network selection algorithm based on improved genetic algorithm. In: 2018 IEEE 18th international conference on communication technology (ICCT), 2018

Chen X, He F, Yu H (2019) A matting method based on full feature coverage. Multimed Tools Appl 78(9):11173–11201

Coello C (2006) Twenty years of evolutionary multi-objective optimization: a historical view of the field. IEEE Comput Intell Mag 1(1):28–36

De Jong K (1975) An analysis of the behavior of a class of genetic adaptive systems. Doctoral Dissertation. The University of Michigan, Ann Arbor

Durairaj M, Dhanavel C (2018) A survey on cloud service scheduling using genetic algorithm. Int J Comput Sci Eng 6(6):1201–1207

Everitt BS, Landau S, Leese M, Stahl D (2011) An introduction to classification and clustering. Clust Anal 5:1–3. https://doi.org/10.1002/9780470977811.ch1

Goldberg DE (1989) Genetic algorithms in search, optimization, and machine learning. Addison-Wesley, Reading

Goldberg D (1999) The race, the hurdle, and the sweet spot: Lessons from genetic algorithms for the automation of design innovation and creativity. In: Evolutionary design by computers. Morgan Kaufmann, 1999, pp 105–118

Gui W, Zhang H (2016) Asymptotic properties and expectation-maximization algorithm for maximum likelihood estimates of the parameters from Weibull-Logarithmic model. Appl Math A J Chin Univ 31(4):425–438

Hamerly G, Drake J (2014) Accelerating Lloyd's algorithm for k-means clustering. In: Partitional clustering algorithms. https://doi.org/10.1007/978-3-319-09259-1_2

Han Z, Ning C, Wei Y (2019) MOPSO for BIM: a multi-objective optimization tool using particle swarm optimization algorithm on a BIMbased visual programming platform. In: "Hello, Culture!" 18th international conference, CAAD futures 2019, pp 39–51

Hart WE (1994) Adaptive global optimization with local search, Doctoral Dissertation. University of California, San Diego

Heppener F, Grenander U (1990) A stochastic nonlinear model for coordinate bird flocks. In: Krasner S (ed) The ubiquity of chaos. AAAS Publications, New York

Holland J (1975) Adaptation in natural and artificial systems. The University of Michigan, Michigan

Kaur G, Aggarwal S (2013) A survey of genetic algorithm for association rule mining. Int J Comput Appl 67(20):25–28

Kim W, Xiong S, Liang Z (2017) Eect of loading symbol of online video on perception of waiting time. Int J Hum Comput Interaction 33(5):1001–1009

Lei X-F (2008) An efficient clustering algorithm based on local optimality of K-means. J Softw 19:1683–1692. https://doi.org/10.3724/sp.j.1001.2008.01683

Li W, Mcmahon C (2007) A simulated annealing-based optimization approach for integrated process planning and scheduling. Int J Comput Integr Manuf 20(1):80–95

Li H-R, He F-Z, Yan X-H (2019) IBEA-SVM: an indicator-based evolutionary algorithm based on pre-selection with classification guided by SVM. Appl Math A J Chin Univ 34(1):1–26

Li H, He F, Liang Y et al (2020) A dividing-based many-objective evolutionary algorithm for large-scale feature selection. Soft Comput 24:6851–6870. https://doi.org/10.1007/s00500-019-04324-5

Lin X, Zhang Q, Kwongs S (2016) A decomposition based multiobjective evolutionary algorithm with classification. In: IEEE congress on evolutionary computation, 2016. IEEE, pp 3292–3299

Luo J, He F, Yong J (2020) An efficient and robust bat algorithm with fusion of opposition-based learning and whale optimization algorithm. Intell Data Anal 24:581–606

Ni B, He F, Pan Y et al (2016) Using shapes correlation for active contour segmentation of uterine fibroid ultrasound images in computer-aided therapy. Appl Math A J Chin Univ 31(1):37–52

Nopiah Z, Khairir M, Abdullah S, Baharin M, Arifin A (2010) Time complexity analysis of the genetic algorithm clustering method, pp 171–176

Poloni C (1997) Hybrid GA for multiobjective aerodynamic shape optimization. In: Winter G, Periaux J, Galan M, Cuesta P (eds) Genetic algorithms in engineering and computer science. Wiley, New York, pp 397–414

Prado RP, García-Galán S, Yuste A, Munoz-Expósito JE (2010) Genetic fuzzy rule-based meta-scheduler for grid computing. In: Proceedings of the 4th international workshop on genetic and evolutionary fuzzy systems. IEEE Computational Intelligence Society, Piscataway, NJ

Rakitianskaia A, Engelbrecht A (2014) Weight regularisation in particle swarm optimisation neural network training. In: 2014 IEEE symposium on swarm intelligence, 2014

Reynolds CW (1987) Flocks, herds and schools: a distributed behavioral model. ACM SIGGRAPH Comput Graph 21(4):25–34

Shrivastava P, Kavita P, Singh S, Shukla M (2016) Comparative analysis in between the k-means algorithm, k-means using with Gaussian mixture model and fuzzy c means algorithm. Commun Comput Syst. https://doi.org/10.1201/9781315364094-186

Tao Q, Zhang M (2013) Mathematical theory of signal analysis vs complex analysis method of harmonic analysis. Appl Math A J Chin Univ 28(4):505–530

Xu R, Wunschii D (2005) Survey of clustering algorithms. IEEE Trans Neural Netw 16:645–678. https://doi.org/10.1109/tnn.2005.845141

Zhang DJ, He FZ, Han SH et al (2016) Quantitative optimization of interoperability during feature-based data exchange. Integr Comput Aided Eng 23(1):31–50

Zhu H, Zhou M (2006) Role-based collaboration and its kernel mechanisms. IEEE Trans Syst Man Cybern Part C Appl Rev 36(4):578–589

Zitzler E, Kunzli S (2004) Indicator-based selection in multiobjective search. Lect Notes Comput Sci 3242:832–842