An Effective Improved Multi-objective Evolutionary Algorithm (IMOEA) for Solving Constraint Civil Engineering Optimization Problems

Ali MAHALLATI RAYENI[1]
Hamed GHOHANI ARAB[2]
Mohammad Reza GHASEMI[3]

ABSTRACT

This paper introduces a new metaheuristic optimization method based on evolutionary algorithms to solve single-objective engineering optimization problems faster and more efficient. By considering constraints as a new objective function, problems turned to multi objective optimization problems. To avoid regular local optimum, different mutations and crossovers are studied and the best operators due their performances are selected as main operators of algorithm. Moreover, certain infeasible solutions can provide useful information about the direction which lead to best solution, so these infeasible solutions are defined on basic concepts of optimization and uses their feature to guide convergence of algorithm to global optimum. Dynamic interference of mutation and crossover are considered to prevent unnecessary calculation and also a selection strategy for choosing optimal solution is introduced. To verify the performance of the proposed algorithm, some CEC 2006 optimization problems which prevalently used in the literatures, are inspected. After satisfaction of acquired result by proposed algorithm on mathematical problems, four popular engineering optimization problems are solved. Comparison of results obtained by proposed algorithm with other optimization algorithms show that the suggested method has a powerful approach in finding the optimal solutions and exhibits significance accuracy and appropriate convergence in reaching the global optimum.

Keywords: Evolutionary algorithm, single objective optimization problem, multi objective optimization algorithm, constraint handling, constraint optimization, civil optimization problem.

Note:
- This paper has been received on March 18, 2019 and accepted for publication by the Editorial Board on October 22, 2019.
- Discussions on this paper will be accepted by May 31, 2021.

- https://dx.doi.org/10.18400/tekderg.541640

1 University of Sistan and Baluchestan, Civil Engineering Department, Zahedan, Iran - a.mahallati.r@pgs.usb.ac.ir - https://orcid.org/0000-0002-0259-8849

2 University of Sistan and Baluchestan, Civil Engineering Department, Zahedan, Iran - ghohani@eng.usb.ac.ir - https://orcid.org/0000-0001-9808-4596

3 University of Sistan and Baluchestan, Civil Engineering Department, Zahedan, Iran - mrghasemi@eng.usb.ac.ir - https://orcid.org/0000-0002-7014-6668

1. INTRODUCTION

Optimization is one of the most practical mathematical methods in all fields of science, especially engineering and industrial problems. Regarding the wide range of optimization issues, various categories have been made to divide them into certain divisions, linear and nonlinear, constrained and unconstrained, convex and concave, single-objective and multi-objective categories and so on.[1]. Among them, nonlinear constrained problems are the most difficult issues to tackle and several methods have been proposed for dealing with them. In single-objective optimization problems, designers' goal is finding a vector of design variables that will provide the best possible design. This vector is meant to give global minimum or maximum response to the designer's problem.

Researchers always have been curios to find an appropriate method to detect the best solutions for optimization problems, an accurate method with an acceptable speed. Although gradient-based methods often provide acceptable responses, they may easily trapped in a local optimum point [2]. Thus, after introducing heuristic and metaheuristic methods by researchers, they were greatly welcomed by scientists due to their high speed and convenient accuracy in finding optimal solutions against gradient methods. Extensive activities have been undertaken to solve optimization problems by metaheuristic methods, and so, various methods have been proposed. After the introduction of the genetic algorithm by Holland in 60's, the use of heuristic and metaheuristic methods for solving optimization problems flourished [3]. Kennedy and Eberhart in 1995, by proposing Particle Swarm Optimization (PSO) algorithms, solved some continuous optimization problems [4]. Atashpaz and Lucas also introduced the Imperialist Competitive Algorithm in 2007 by modeling the performance of the imperialists against their Colonies in real world [5]. Among recent optimization algorithms introduced, the Teaching-Learning Based Optimization Algorithm, built bassed on teacher and student behavior in classroom, was presented by Rao and Patel in 2012 [6]. One can also refer to the Ghaemi and Feizi researches that introduced the Forest Optimization Algorithm to solve continuous nonlinear optimization problems [7]. In 2015, Rezaee proposed the Brainstorm optimization algorithm based on the human brain's ability to search for solutions for everyday life issues [8]. Also in the same year, Dai et al. proposed a modified genetic algorithm to optimize Stiffness optimization of coupled shear wall structure[9]. Mirjalili and Lewis adapted Whale Optimization Algorithm in 2016 inspired by the social behavior of the humpback whales and solved various optimization problems [10]. Varaee and Ghasemi introduced a new algorithm based on ideal gas molecular movement[11] .Tabari and Ahmad also proposed the Electro-Search Algorithm in 2017 based on the movement of electrons through the orbits around the nucleus of an atom [12]. And many other researchers focused their attention to introduce new algorithms[13-15].

Many scholars have focused their research on improving the performance of heuristic and metaheuristic algorithm methods in dealing with constrained problems. generally, modification of optimization methods are divided into five main categories, including the penalty function, combining of algorithms, separation of constraints and goals, using special operators, and repairing the algorithm. For modification by combination of many algorithms, researchers attempted to combine different features of existing algorithms and use the strengths of each one to improve the overall performance of the method [16]. The proposed technique was then used in a research by the Zhang and Kucukkoc combines Artificial Intelligence and parallel computing [17]. Also research by Chou et al., Tosta et al., Araghi et al., all utilizing the combination of genetic algorithm and fuzzy logic [18-20]. As well as Li et al. used hybrid algorithm which is achieved by combination of genetic algorithm and particle swarm optimization. And many other researchers modified existed algorithms by this technique.

The second popular task which extensively used to enhance the performance of heuristic and metaheuristic algorithms is the Penalty Functions. It is claimed that by applying various methods of penalizing such as static, dynamic or adaptive penalties and etc., this technique eliminates constraint violated samples from the procedure, thereby increases the efficiency of the method by reducing the computational cost [16]. Yang et al., embedded a weighted penalty function to sequential optimization method in order to improve the performance of the algorithm in the estimation of set density [21]. Dong and Zhu optimize the search pattern in an algorithm in a scattered space by combining an accelerated iteration hard thresholding (AIHT) with analytical penalty methodology[22]. In 2016, Kia used e-exact penalty function to solve problems with inequality constraints [23].

One of the most practical techniques been used for improving the performance of optimization algorithms predominantly, is the separation of objectives and constraints, in which the problem is solved by examining the constraints situation and also limitations of the problem. Due to the particular action given to equations and constraints, this method is also called constraint control, has a huge impact on the performance of the algorithm, especially increasing the speed of computing and reducing its volume [16]. Tang et al. at the selection stage in the Genetic Algorithm, with the ranking of individuals based on the frequency of violations in constraint problems, succeeded in increasing the efficiency of the genetic algorithm [24]. Long in order to solve multi-objective constraint problems, at first divided them into several sub-problems, then prioritized them with a multi-objective genetic algorithm [25]. Garcia et al. investigated the effects of conventional constraints control on a genetic algorithm [26].

With the progress of technology and exploration of new fields in science, scientists also encountered new optimization problems in which they had more than one objective function. Therefore, in order to optimize these problems which contain several objective functions; multi-objective optimization algorithms were developed. Multi-objective optimization methods trying to find values of design variables that are applicable in the constraints and optimize the objective function at the same time. Generally, it is not possible to obtain the best value for all objectives simultaneously, and there will be no definite answer which all the objective functions be satisfied. In multi-objective optimization problems, unlike single-objective problems, generally a unique vector of design variables cannot be suggested because an absolute optimal solution does not exist in such issues. In some cases, these objective functions are opposite to each other, so by minimizing one of them, the other one will deteriorate. Therefore, there is no certain optimal answer that optimizes all the objective functions simultaneously, and generally facing many optimal responses which satisfy problem constraints with no superiority against each other. In multi-objective problems, always there are a number of optimal answers that are known as Pareto optimal solutions or Pareto Front, which are the main difference of single-objective and multi-objective optimization problems. Once the Pareto answers is determined for a problem, the user is able to decide on choosing the best answer based on his preferences and needs. Meanwhile, evolutionary algorithms with their proper function in finding solution of optimization problem were considered more than other methods. Many of the optimization problems in the engineering field are multi-objective optimization problems in which there are several objective functions that needed to be optimized simultaneously[27].

Over the past decades, several heuristic and metaheuristic methods have been proposed for solving multi-objective optimization problems. The primary reason, is to find better Pareto-optimal solutions with least runs. Using stochastic techniques made it much easier to find local optima avoidance and gradient-free mechanism that made them applicable to real world problems[28]. Some of the well-known optimization techniques are: Non-dominated Sorting Genetic Algorithm 2 (NSGA-2) [29-32], Multi-objective Particle Swarm Optimization (MOPSO)[33, 34]., and Multi-objective Evolutionary Algorithm based on Decomposition (MOEA/D)[35].

As mentioned, constraint handling method is one of the applicable methods for improving the performance of evolutionary algorithms. In 2000, Coello Coello by considering the constraints of constraint optimization problem as an objective function, solved the optimal solution using multi-objective evolutionary algorithms [36]. In 2016, Segura et al. also categorized two-objective optimization methods for solving single-objective problems [37].

In this paper, by using the concepts of objectives and constraints separation, multi-objective evolutionary algorithms, efficient selection strategy and elitism, an effective improved multi-objective evolutionary algorithm (IMOEA) is introduced for solving constraint industrial single-objective problems. In order to evaluate the performance of this method, some benchmark problems of the CEC2006 were solved and compared with other methods. Also, in order to innovate and enhance the function of the proposed method, normal mutation and uniform crossover operators are selected for producing the next generation and their performance is also examined by other known methods. Due to the direct effect of mutation and crossover operators on the performance of evolutionary algorithms, the percentage of interference of these two operators is considered dynamically according to the number of iterations carried. Thus, regarding the feasible solutions, a certain percentage of those individuals (elites) are also stored and transferred to the next iteration. After achieving acceptable result for mathematical problems, some popular engineering benchmark problems which is the main contribution of this research is solved and compare to previous studies.

2. BASIC DEFINITIONS IN MULTI-OBJECTIVE OPTIMIZATION

In multi-objective optimization, several objective functions are treated simultaneously and defined in the general model as follows:

$$\text{Minimize } \mathrm{F}(X) = (\mathrm{f}_1(X), \mathrm{f}_2(X), \mathrm{f}_3(X), \dots)$$
$$\text{Subject to: } \begin{cases} \mathrm{h_i}(X) = 0, & \mathrm{I} = 1,2,\dots,\mathrm{m}_1 \\ \mathrm{g_j}(X) \le 0, & \mathrm{j} = 1,2,\dots,\mathrm{m}_2 \end{cases} \tag{1}$$

where $\boldsymbol{X}$ is an n-dimensional vector of design variables and $f_i(X)$ are the objective functions. $g_j(X)$ and $h_i(X)$ are the design constraints known as the inequality and equality constraints, respectively. m_1 and m_2 are, in order, the number of equality and inequality constraints.

Generally, multi-objective optimization problems are converted to a single-objective optimization problem by a scalar function, in particular when objectives are in conflict with each other. They will be then solved by single-objective optimization algorithms. After introducing of Multi-Objective Genetic Algorithm (MOGA) in 1993 by Fonseca and Fleming, based on genetic algorithm, multi-objective optimization algorithms were used for solving multi-objective problems. This algorithm gains premature convergence due to the adoption of a very large answer space in its process [38]. So, in 1994, Srinivas and Deb improved the MOGA algorithm by ranking the answers and introduced the Non-dominated Sorting Genetic algorithm (NSGA) [39]. Since then, Deb et al. in 2000, introduced the second version of the NSGA, which obviated the problems in the first version (NSGA-I) which was including high computational complexity, lack of elitism, and the need to specify the subscription parameter [40]. In each multi-objective optimization problem, finding a set of trade-offs optimal solutions is very important. These solutions practically are the answer to the multi-objective optimization in a variety of situations called the Pareto solution set or Pareto Front, named after Vilferedo Pareto[41]. The goal of multi-objective optimization algorithms is to find the Pareto Front, which enables user to extract the optimal answer according to existing situation. The main feature of The Pareto Front is that there is no better answer than this Pareto solution points set in the problem and in other word Pareto Front solutions are not dominated by any other answer. Thus, the Pareto front uses the non-dominated concept to form the solutions of any problem. In other word, A dominates B, if A has at least one objective function better than B and is not worse in other objectives. The non-dominant answer is called Pareto-optimal.

3. METHOD

Before explaining the method outlined in this paper, several basic descriptions that have been used to solve problems are introduced.

3.1. Basic Definitions

The general form of the single-objective optimization problem with equality and inequality constraints is shown as:

$$\begin{aligned} &Minimize\ f(X) \\ &Subject\ to\colon \begin{cases} h_i(X) = 0, & i = 1,2,\dots,m_1 \\ g_j(X) \le 0, & j = 1,2,\dots,m_2 \\ L_k \le x_k \le U_k, & k = 1,2,\dots m_3 \end{cases} \end{aligned} \tag{2}$$

where X is a n-dimensional vector of design variables and $f(X\ )$ is the objective function which in this case minimization of $f(X)$ is the objective of optimization. $g_j(X)$ and $h_i(X)$ are constraints of optimization problem and also known as inequality and equality constraints, respectively.

To reduce the complexity for solving single-objective problems, due to increase the accuracy and speed of solving such problems, following equation for transforming constraints to objectives are considered.

For equality constraints:

$$v_{1i}(x) = max(|h_i(x) - \sigma|, 0) \quad i = 1,2,3 \dots, m_1 \tag{3}$$

And for inequality constraints:

$$v_{2i}(x) = max(g_i(x), 0) \quad i = 1,2,3 \dots, m_2 \tag{4}$$

Finally, the objective function derived from the constraints is the sum of above-mentioned objectives:

$$v = \sum_{1}^{m_1} v_{1i}(x) + \sum_{1}^{m_2} v_{2i}(x) \tag{5}$$

where, σ is a positive value used for equal constraints in order to convert them to inequality constraints. Now, any constrained single-objective problem can be easily turned into an unconstrained bi-objective optimization.

$$F(X) = \big(f(x), v(x)\big) \tag{6}$$

Now, by above definitions a Pareto Front can be depicted and optimum answer can be easily chosen by this diagram. This procedure will explain in 3.2.2.

3.2. Design of the Proposed Algorithm

The proposed algorithm (IMOEA) is categorized as an evolutionary algorithm. As well as a major novelty in answer selection, it includes effective crossover and mutation which are the two main operators of evolutionary algorithms. In addition to the aforementioned features, further steps are proposed in detail as follows.

3.2.1. Pseudocode of Algorithm

In table 1, the pseudocode of the introduced algorithm is presented.

Table 1 - Pseudo code of the IMOEA

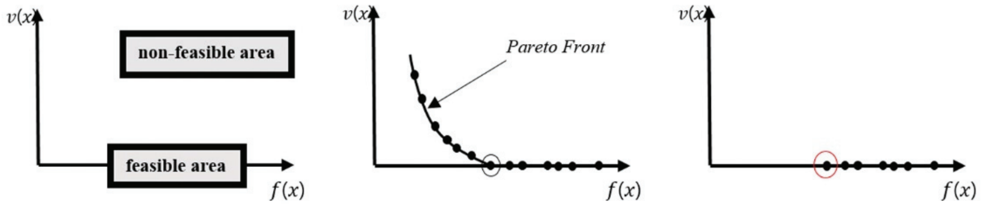
```
Initialize the first population randomly and set the initial parameters
Calculate the objective functions of individuals ($f(x), v(x)$)

Sort non-feasible individuals by $v(x)$ and feasible individuals by $f(x)$
Find the best individuals and choose determined elites of feasible and non-feasible solution

Create the next generation by crossover and mutation
Until reach the convergence or the end of maximum iterations set
Update the position of individuals
Calculate the fitness of individuals
Sort non-feasible individuals by $v(x)$ and feasible individuals by $f(x)$
Find the best individuals and choose determined elites of feasible and non-feasible solution
Update elite if individuals become fitter than the elite
Return
```

3.2.2. Optimal Answer Selection Method

During optimization process and drawing Pareto Front, the Pareto answers will be sorted according to the second objective function (constraints ($v(x)$) which defied in 3.1.). Naturally, the best answer will be in the possible minimum violation (without any violation ($v(x) = 0$)), which is the global answer of optimization problem. By this way, when the obtained values arranged by $v(x)$, it is completely obvious that the values with $v(x) = 0$ are placed at the beginning of these ordered answers. Therefore, according to the mentioned procedure, answers in which the value of the second objective function is zero ($v(x) = 0$) are located in the feasible area and acquired needed qualification to be an acceptable answer for the problem. While the other solutions with the second objective function (constraint) value is not equal to zero (definitely a positive number due to the mentioned definition ($v(x) > 0$)), has violated the initial constraints of the main problem and are not in feasible space. Thus, by selecting an appropriate answer, the speed of algorithm is guaranteed by limiting the search space, due to removing many individuals in the search space and also local optimums which can mislead the optimization process to reach the global optimum. Generally, the researchers are trying to draw the Pareto Front to introduce the existing answers and provide the possibility of choosing appropriate answer by the user. But in this method, depending on the particular approach, which is applied to the constraints, the selection of the answer is already done ($v(x) = 0$) and there is no need to draw other answers or Pareto Front anymore. Therefore, the Pareto front drawing is luxury in this algorithm, and according to the explanation, the answer is a point on the $f(x)$ axis and has the lowest possible value of $f(x)$. In this process, feasible individuals are in the first priority of selection and then for reaching the desired number of individuals of each iteration, remaining individuals are selected between infeasible individuals. This concept is shown in Fig. 1.



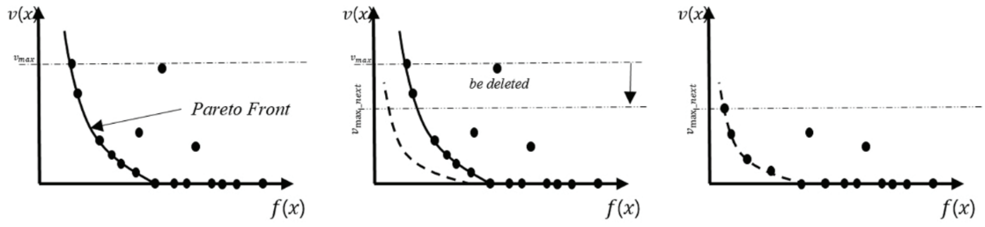
(a) feasible and non-feasible area *(b) Pareto Front* *(c) place of global optimum*

Fig. 1 - Global optimum position in this method

3.2.3. Infeasible Space Controller

To avoid any additional searches in the infeasible space, as well as increasing the efficiency of the algorithm and make faster convergence, some regulation for constraints is employed. Owe to the existence of multiple constraints in constraint optimization problems, and since it is obvious that even if one of these constraints violated, then the answer is not acceptable because this answer is not in the feasible space, so by specifying an allowed value for violation of constraints, producing new generations are focused on the immediate vicinity of the optimum answer which are definitely without any violations. The choice of this value is also considered rationally, and the main definitions of the problem are used in its selection. This value is the maximum violation among all the constraints of the problem. Clearly, in each iteration, this amount can change and reduce the production interval for offspring so that it makes it easier for algorithm to reach the global response of the problem faster and with no violation. It should be noted that the value of v_{max} is specified in the ith iteration and bounds the scope of the second-objective function in the $i^{th} + 1^{st}$ iteration.

$$v_{max} = \max(v_{1i}(x), v_{2i}(x)) \tag{7}$$



(a) whole infeasible answer space *(b) reducing infeasible space* *(c) reduced infeasible space*

Fig. 2 - Procedure of Infeasible space controller. v_{max} is the maximum value of $v(x)$ in the current population. v_{max_next} is the maximum violation for the next population

3.2.4. Mutation Operator

The mutation operator has always been one of the most influential operators in evolutionary algorithms. Many methods are developed to achieve better Mutation, including basic methods such as swap, inversion and etc., as well as more sophisticated methods such as uniform, normal [42] and so on. Swap mutation attempts to select two genes from a chromosome and replaces them with each other [43]. In the inversion mutation, several genes are selected from the chromosome and the order of the selected genes is reversed inversion [44]. For continuous problems, the use of more complicated mutations is more prevalent. The general form of the mutation operator in continuous problems is in the form of the given equation:

$$x_i' = x_i + (u_i - l_i)\overline{\delta_i} \tag{8}$$

where x_i' is the mutated gene (child), x_i is the primary gene (parent), u_i is the upper limit, l_i is lower limit and $\overline{\delta_i}$ is the distribution function. Now, if the uniform distribution function is chosen, that is, random numbers in the range of -1 to 1 is generated as new generation, then the uniform mutation is formed [45]. In the case of the use of a standard normal distribution that normally distributes a set of random numbers, in $\overline{\delta_i}$, the mutation is called normal [46]. In this research, the performance of these four types of mutations was investigated on all test subject on 50 independent runs and 100 iterations. Since G02 is a harder problem against other problems due to its many variables, the average of 50 result is depict in Fig. 3. According to this figure, normal distribution shows proper function to find the optimal point, and this mutation was used as the mutation of the algorithm. It is important to note that this comparison was carried out for problems selected from CEC2006 [47].

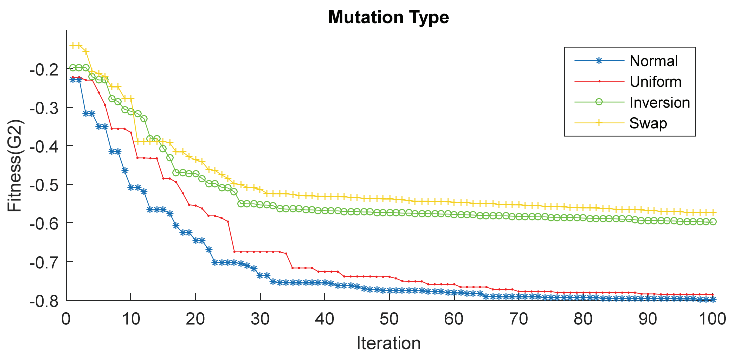


Fig. 3 - Comparing role of mutation operators

3.2.5. Crossover Operator

Another influential operator in the evolutionary algorithms is the crossover operator. This operator, like the mutation operator, includes many methods like single-point, arithmetic, Uniform, heuristic and etc. A single-point crossover is that two parent genes are cut from a random location and then two genes are combined, and children are created [48-50]. In Uniform crossover also a completely randomized gene is produced as the size of parents' genes. In case of having a value of 0 in a randomized gene, the corresponding strand should be found in the first parent and poured into a new gene. In case of having the value of 1 in a randomized gene, corresponding value of the second gene will assign to the new gene. The Uniform crossover is also known as Mask crossover. The schematic of Uniform crossover is shown in Fig. 4. The Uniform function in this research provided good results.

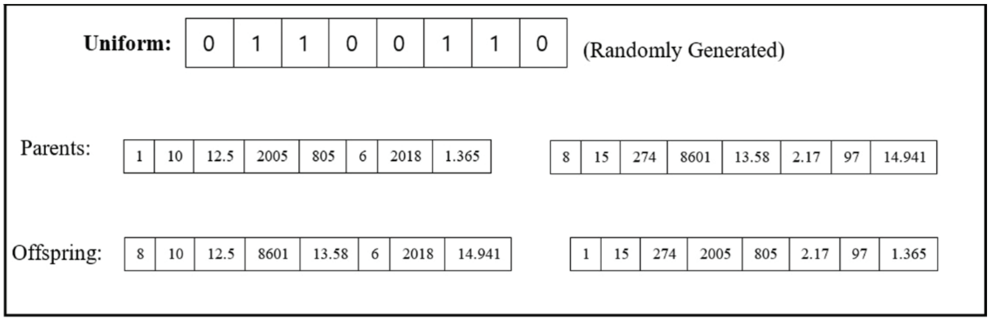

Uniform: 0 1 1 0 0 1 1 0 (Randomly Generated)
Parents: 1 10 12.5 2005 805 6 2018 1.365
8 15 274 8601 13.58 2.17 97 14.941
Offspring: 8 10 12.5 8601 13.58 6 2018 14.941
1 15 274 2005 805 2.17 97 1.365

Fig. 4 - Schematic of Uniform crossover

In the arithmetic crossover, a weighted average between the two parents also produces a new generation. Thus, between the two parents, the one which has a better value is selected as an observer, and a random number is multiplied by the difference between the other two parents, adding to the observer's value, and the new generation is generated by the heuristic crossover. Also, the performance of these four types of crossover was investigated on all test subject on 50 independent runs. Since G02 is a harder problem against other problems, the average of 50 result is depict in Fig. 5, the Uniform crossover was selected as the main crossover of the algorithm, because of its relatively better performance. This comparison was carried out for the problems selected from CEC2006 [47].

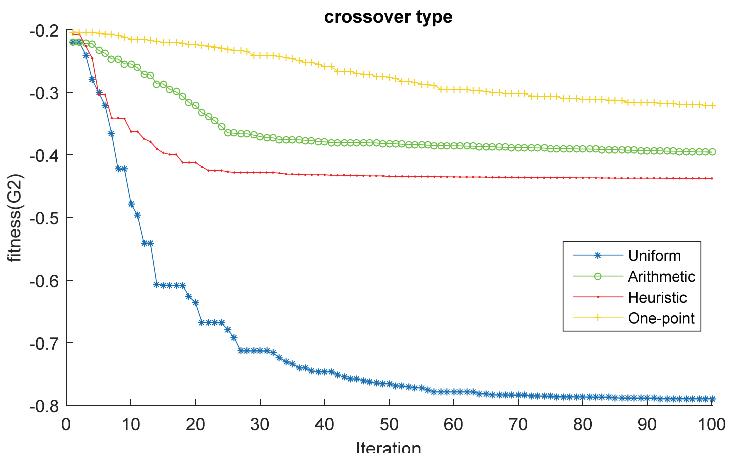


Fig. 5 - comparison function of crossover operators

3.2.6. Interference of Operators

As mentioned in other studies, evolutionary algorithms are used to find the optimal solution by mutation and crossover operator functions [51, 52]. For a reasonable moving toward the global optimum, the impact percentage of these two operators is very important for the production of each new generations. Generally, in the initial iterations, the crossover operator makes a better convergence of the algorithm, and in the final iterations, the Mutation operator prevents individuals from being captured at local optima. Therefore, in order to consider this feature in the proposed algorithm and also to avoid additional calculations, the coefficients of impact of these two operators are considered dynamically and it is related to momentary iteration of the procedure. In Crossover operator it is linearly reduced and for the Mutation operator is linearly increased. This means that in the initial iterations, the Crossover operator has its maximum value and in the final iterations the Mutation operator produce majority of breeds than Crossover. The formulation of this feature of IMOEA shown in equation 9 and 10.

For mutation:

$$M_p = \frac{M_{max} - M_{min}}{it_{max} - 1} \times (it - 1) + M_{min} \tag{9}$$

and for crossover:

$$C_p = \frac{C_{max} - C_{min}}{1 - it_{max}} \times (it - 1) + C_{min} \tag{10}$$

where M_p and C_p are mutation and crossover percentage of current iteration, respectively. M_{max} is maximum allowable mutation, M_{min} is minimum allowable mutation, C_{max} is maximum allowable crossover, C_{min} is minimum allowable crossover and it_{max} is maximum allowable iteration which all of these parameters are defined in the beginning of optimization process. it represent current iteration of algorithm. Due to the above definitions M_p and C_p update in every iteration and their values are completely depends on current iteration of process.

3.2.7. The Importance of Individuals without Violation (Feasible Elites)

Crossover and Mutation operators use individuals of each iteration to search the solution space to achieve optimal points. If they are chosen among violated individuals (parents), there would be very little chance to find a point in the feasible area. So due to the specific performance of the proposed algorithm and also many points that could be in the infeasible area, a certain percentage of the feasible solution in each iteration are selected called feasible elites. Now crossover and Mutation certainly can select individuals in the feasible area and combine them with other individuals which will increase the probability of finding global answer. There is also a special way to prioritize these individuals. Since these individuals have no constraints violation ($v(x) = 0$), these individuals are arranged according to the value of the main objective function of the problem ($f(x)$) in an ascending order, which is clear that the smallest value of $f(x)$ of these individuals is the optimal answer of the problem, and then the rest of the individuals with higher $f(x)$ are arranged. It should be noted that if the number of existed individuals without any violation is less than a selected specified percentage which set at the beginning of optimization process, the algorithm fills the population with other individuals without violation, and if the present individuals with no violation is greater than is needed, the algorithm chooses the individuals as much as the elite number and does not consider the rest.

4. NUMERICAL VERIFICATION

In order to evaluate the performance of the proposed algorithm, ten mathematical benchmark functions were solved and compared to the some existing algorithms to demonstrate the efficiency of IMOEA. After that by using IMOEA features four benchmark engineering problems have been solved.

4.1. Mathematical Constraint Problems

In this subsection, benchmark problems from the CEC2006 competitions are chosen to test the performance of IMOEA. The solutions and properties of the benchmark functions to the following constrained functional problems can be found in [47] and also demonstrated in Table 2. ρ is the ratio of feasible region to decision region and a is the number of active constraints. Li represents the number of linear inequalities constraints. NI is the number of nonlinear inequalities constraints. LE is the number of linear equality constraints. NE is the number of nonlinear equality constraints.

Table 2 - Details of the test benchmark problems (Liang, et al., 2006)

Function	**Optimal value**	**n**	**Type of function**	**(%)ρ**	Li	NI	LE	NE	**α**
G01	15.0000000000	13	Quadratic	0.0111	9	0	0	0	6
G02	-0.8036191042	20	Nonlinear	99.9971	1	1	0	0	1
G03	-1.0005001000	10	Nonlinear	0.0000	0	0	0	1	1
G04	-30665.53867178	5	Quadratic	52.1230	0	6	0	0	2
G05	5126.4967140071	4	Cubic	0.0000	2	2	0	3	3
G06	-6961.813875580	2	Cubic	0.0066	0	5	0	0	2
G07	24.3062090681	10	Quadratic	0.0003	3	2	0	0	6
G08	-0.0958250414	2	Nonlinear	0.8560	0	4	0	0	0
G09	680.6300573744	7	Nonlinear	0.5121	0	3	0	0	2
G10	7046.2480205287	8	Linear	0.0010	3	0	0	0	3

Table 3 - Robustness of IMOEA

No. samples	Percent of Elites	No. Elites	Best answer
60	10	6	-0.2382
	20	12	-0.3699
	30	18	-0.4098
110	10	11	-0.3756
	20	22	-0.7989
	30	33	-0.4376
160	10	16	-0.6174
	20	32	-0.4609
	30	48	-0.0465

Table 3 represent the robustness of IMOEA and sensitivity to its parameters for G02. G02 selected because is a harder problem against other problems due to its many variables. As it can be seen in the table, with increasing population better answer is acquired. But number of elites have a crucial role. If number of elites chosen is low, IMOEA may stuck in local optimum. And if this number acquired is too high, IMOEA lose its way toward global optimum due to the large number of elites. So, number of elites should be chosen as a moderate number according to population. Also, maximum and minimum crossover and mutation selected based on their abilities to combine answers to find better solution and escape from local optimums after some experiment. The maximum possible Crossover and Mutation are 0.9 and minimum are 0.3 of the whole population.

Table 4 shows the performance of the proposed algorithm in finding the optimal solution. The number of individuals is kept constant as 110 for all problems. The maximum possible Crossover and Mutation are 0.9 and their minimum are 0.3 of population. Also, if there are more than one individual without violation, 20% of all individuals can be allocated to feasible elites. For each problem, 50 independent runs have taken place and the values of the best, average, and worst answers have been reported. It is important to note that the maximum allowable iteration for all problems is considered 100 iterations and final answer is reported in Table 4.

Table 4 – Comparison of optimum results for the proposed algorithm with literature

function	metrics	PSO[55]	ASCHEA[54]	BBO[53]	IMOEA
G1	Best	-15	-15	-14.97	**-15**
	Mean	-13	-14.84	-14.58	**-14.8418**
	Worst	-14.71	-14.555	-14.67	**-14.7207**
	Std.	N/A	N/A	N/A	0.14006
	No. Analyses	350000	155000	240000	11000
G2	Best	-0.66	-0.785	-0.78	**-0.7989**
	Mean	-0.29	-0.59	-0.73	**-0.7794**
	Worst	-0.41	-0.0792412	-0.76	-0.69522
	Std.	N/A	N/A	N/A	0.04497
	No. Analyses	350000	155000	240000	11000
G3	Best	-0.99	**-1**	**-1**	**-1**
	Mean	-0.76	**-1**	-0.04	**-0.99336**
	Worst	0.46	**-1**	-0.39	-0.97104
	Std.	N/A	N/A	N/A	0.01237
	No. Analyses	350000	155000	240000	11000
G4	Best	**-30665.539**	**-30665.539**	**-30665.539**	**-30665.539**
	Mean	-30665.539	-30665.539	-30411.86	**-30665.539**
	Worst	-30665.539	-30665.539	-29942.3	**-30665.539**
	Std.	N/A	N/A	N/A	0
	No. Analyses	350000	155000	240000	11000
G5	Best	5126.5	**5126.484**	5134.2749	5130.3417
	Mean	5249.825	5185.714	7899.2756	**5185.648**
	Worst	**5135.973**	5438.387	6130.5289	5198.481
	Std.	N/A	N/A	N/A	29.5642
	No. Analyses	350000	155000	240000	11000

Table 4 – Comparison of optimum results for the proposed algorithm with literature (continue)

G6	Best	**-6961.814**	**-6961.814**	**-6961.8139**	**-6961.4498**
	Mean	**-6961.814**	-6961.805	-5404.4941	-6716.7997
	Worst	**-6961.814**	-6961.813	-6181.7461	-6412.0801
	Std.	N/A	N/A	N/A	224.7254
	No. Analyses	350000	155000	240000	11000
G7	Best	24.37	**24.33**	25.6645	24.3594
	Mean	32.407	**24.66**	29.829	26.0917
	Worst	56.055	**25.19**	37.6912	30.3007
	Std.	N/A	N/A	N/A	2.4944
	No. Analyses	350000	155000	240000	11000
G8	Best	**-0.095825**	**-0.095825**	**-0.095825**	**-0.095825**
	Mean	-0.095825	**-0.095825**	-0.095817	-0.075487
	Worst	-0.095825	**-0.095825**	-0.095824	-0.069144
	Std.	N/A	N/A	N/A	0.01134
	No. Analyses	350000	155000	240000	11000
G9	Best	680.63	680.63	**680.63**	681.6552
	Mean	**680.33**	680.64	692.7162	687.739
	Worst	**680.33**	680.653	721.0795	690.3995
	Std.	N/A	N/A	N/A	3.6598
	No. Analyses	350000	155000	240000	11000
G10	Best	**7049.548**	7061.13	7679.0681	7052.0911
	Mean	7147.334	7497.434	9570.5714	**7136.2836**
	Worst	9264.886	7224.407	8764.9864	**7164.5763**
	Std.	N/A	N/A	N/A	47.7742
	No. Analyses	350000	155000	240000	11000

As it can be seen, the proposed algorithm in most of cases finds better answer rather than other algorithms with lower number of calculations which shows fast convergence of IMOEA. The main superiority of IMOEA is that in most of the cases, the standard deviation of answer is very low which guaranteed the acceptable performance and obtaining good result of algorithm in each run. It is obvious that if the number of iterations or individuals increases, IMOEA is completely able to acquire better result and find global optimum.

4.2. Real Engineering Design Problems

Since evolutionary algorithms are very popular for solving engineering problems [56-65], in this section, 4 well-known engineering benchmark problems have been solved and compared with the literature. Number of individuals is equal to 120 and the number of permitted iterations is 100. So the maximum amount of required processing is 12,000. Also, the maximum crossover and mutation are 0.9 and their minimum are 0.3. Also, if there are more than one individual without violation, 30% of all individuals can be allocated to feasible elites. For each problem, 50 independent runs have been taken, the values of the best, average, worst responses and standard deviation and also, the number of analyzes needed to reach the final answer have been reported.

4.2.1. Optimum Design of a Pressure Vessel

The objective is to minimize the total fabrication cost of a cylindrical pressure vessel, as shown in Fig. 6

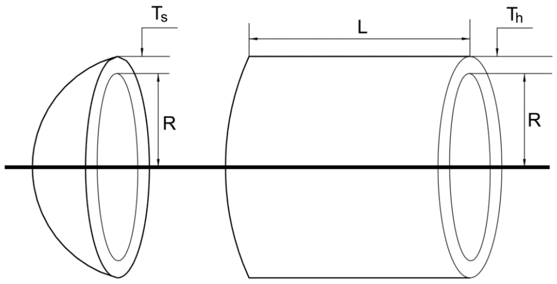


Fig. 6 - Pressure vessel design problem with four design variables

This problem has also been popular among researchers. Many heuristic techniques have been used to optimize this problem, such as GA [66], CPSO [67], DE [68], PSO [67], and GWO [69]. The problem involves two discrete and two continuous variables and four inequality constraints. The design variables are the shell thickness ($\boldsymbol{T_s}$), the spherical head thickness ($\boldsymbol{T_h}$), the radius of cylindrical shell (R), and the shell length (L). The shell and head thicknesses should be multiples of 0.0625 in and within the range 1 × 0.0625 and 99 × 0.0625 in. The radius of cylindrical shell and the shell length are limited between 10 and 200 in. The mathematical model for the problem can be stated as follows:

$$\begin{aligned} &\min f(T_s, T_h, R, L) = 0.6224\,T_s R L + 1.7781 T_h R^2 \ + 3.1661 T_s^2 L + 19.84 T_s^2 R. \\ &\text{Subject to:} \\ &g_1 = -T_s + 0.0193R \le 0, \\ &g_2 = -T_h + 0.0095R \le 0, \\ &g_3 = -\pi R^2 L + \frac{4}{3}\pi R^3 + 1296000 \le 0, \\ &g_4 = L - 240 \le 0. \end{aligned} \tag{11}$$

The optimal solution and constraint values obtained by the suggested method were compared with those in the literature and are presented in Table 5. As shown in Table 5, IMOEA achieved better results for the pressure vessel design problem compared to those based on GA, CPSO, DE, and PSO. The statistical data on various independent runs compared with the results by others in the literature are listed in Table 6.

Table 5 - Comparison of the best solution for pressure vessel design problem

	GA[66]	CPSO[67]	DE[68]	PSO[67]	GWO[69]	IMOEA
T_s	0.812	0.812	0.812	0.812	0.812	0.812
T_h	0.4375	0.4375	0.4375	0.4375	0.4345	0.4345
R	42.0974	42.0913	40.3239	42.0984	42.0892	42.0509
L	176.6540	176.7465	200.00	176.6366	176.759	177.2305
g_1	-0.00002	-1.37E−06	-0.03425	-8.80E−07	-1.788E−04	-9.1666499e-04
g_2	-0.035	-.0036	-0.054	-0.036	-0.038	-0.035
g_3	-27.886	-118.768	-304.4	*3.122*	-40.616	-24.819
g_4	-63.346	-63.253	-40	-63.363	-63.241	-62.769
f_{min}	6059.9463	6061.077	6288.744	6059.714	6051.5639	6056.179
No. Analyses	80,000	240,000	900000	60,000	NA	12000

Table 6 - Statistical results of different approaches for pressure vessel design problem

	Best	Mean	Worst	Std.	No. Analyses
Coello and Montes [68]	6059.946	6177.253	6469.322	130.9267	80,000
He and Wang [67]	6061.078	6147.133	6363.804	86.4545	240,000
Mirjalili et al. [69]	6051.5939	NA	NA	NA	NA
Kaveh and Talatahari [70]	6059.73	6081.78	6150.13	67.2418	NA
IMOEA	6056.179	6060.769	6069.978	4.348555	12000

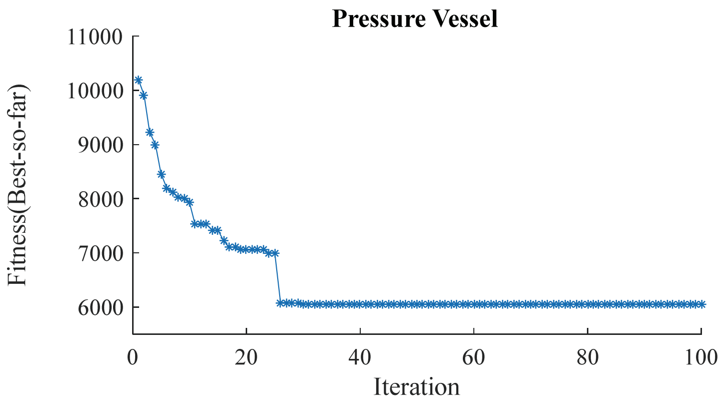


Fig. 7 - Convergence history for the pressure vessel design problem

Fig. 7 illustrates the convergence history for the pressure vessel design problem using proposed algorithm. As is clear from Fig.7, after almost 25 iterations the global optimum is found but since the maximum searching iteration is meant 100, it continuous to that end.

4.2.2. Optimum Design of a Tension/Compression Spring

The tension/compression spring problem was first introduced by Belegundu [71] and Arora [72] as shown in Fig. 7. The aim for this problem is to minimize the weight of the tension/compression spring subject to surge frequency, shear stress, and minimum deflection. There are three design variables in this problem: wire diameter (d), mean coil diameter (w), and the number of active coils (N). The problem can be stated as follows:

$$\min f(w,d,N) = (N+2)w^2 d.$$

Subject to:

$$\begin{aligned} g_1 &= 1 - \frac{d^3 N}{71785 w^4} \leq 0, \\ g_2 &= \frac{d(4d-w)}{12566 w^3 (d-w)} + \frac{1}{5108 w^2} - 1 \leq 0, \\ g_3 &= 1 - \frac{140.45 w}{d^2 N} \leq 0, \\ g_4 &= \frac{2(w+d)}{3} - 1 \leq 0. \end{aligned} \tag{12}$$

The bounds on the design variables are:

$$0.05 \leq w \leq 2.0, 0.25 \leq d \leq 1.3, 2.0 \leq N \leq 15.0. \tag{13}$$

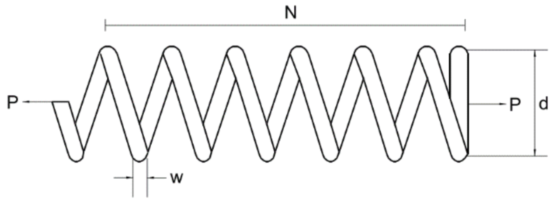


Fig. 8 - Tension/compression spring design problem indicating also the design variables

Belegundu [71] attempted the problem using eight different mathematical optimization methods. Arora [72] also found the optimum results using a numerical optimization approach with a constraint correction at the constant cost. Coello and Montes [68] utilized a GA-based method. In addition, He and Wang [67] solved this problem using a co-evolutionary particle swarm optimization (CPSO). Recently, Eskandar et al. [73] and Kaveh and Talatahari [74] used water iteration algorithm (WCA) and the charged system search (CSS) to find the optimum results. Tables 7 and 8 compare the best results obtained by proposed method with those recorded by others. The standard deviations of optimum costs reported in Table 8 prove the consistency of the proposed method with those in the literature.

Table 7 - Comparing the best solutions for the tension/compression spring design problem using different approaches

	MPM [71]	GA [68]	WCA [73]	CC [72]	CPSO [67]	IMOEA
w	0.50	0.051989	0.051689	0.053396	0.051728	0.0518449
d	0.3159	0.363965	0.356522	0.399180	0.357644	0.360472
L	14.25	10.890522	11.30041	9.185400	11.244543	11.0726
g_1	- 0.000014	-0.000013	-1.65E−13	-0.053396	-0.000845	-1.462237433758062e-05
g_2	- 0.003782	-0.000021	-7.90E−14	-0.000018	-1.26E−05	-1.4317270180419e-05
g_3	- 3.938302	-1.061338	-4.053399	-4.123832	-4.051300	-4.060985594960838
g_4	- 0.756067	-0.722698	-0.727864	-0.698283	-0.727090	-0.725122066666667
f_{min}	0.0128334	0.0126810	0.012665	0.0127303	0.0126747	0.012666178120783
No. Analyses	NA	80,000	11,750	NA	NA	12000

Table 8 - Statistical results of different methods for tension/compression spring optimum design problem

	Best	Mean	Worst	Std.	No. Analyses
Belegundu and Arora [71]	0.0128334	NA	NA	NA	NA
Coello and Montes [68]	0.012681	0.012742	0.012973	5.90E−05	80,000
Eskandar et al. [73]	0.012665	0.012746	0.012952	8.06E−05	11,750
Arora [72]	0.0127303	NA	NA	NA	NA
He and Wang[67]	0.0126747	0.01273	0.012924	5.20E−05	200,000
Kaveh and Mahdavi [75]	0.0126697	0.0127296	0.128808	5.0038E−05	4000
IMOEA	0.012666178120783	0.012732	0.012986	0.000102	12000

Fig. 9 illustrates the convergence history for the tension/compression spring design problem using the new approach. As is clear from Fig. 9, after almost 60 iteration the global optimum is found but since the maximum iteration of search is 100, searching continuous to reach this number. It is important to note that in the first iteration of this problem, proposed algorithm could not find any individuals in feasible area but its procedure did not stop and continue processing until find an answer which suits problem constraints.

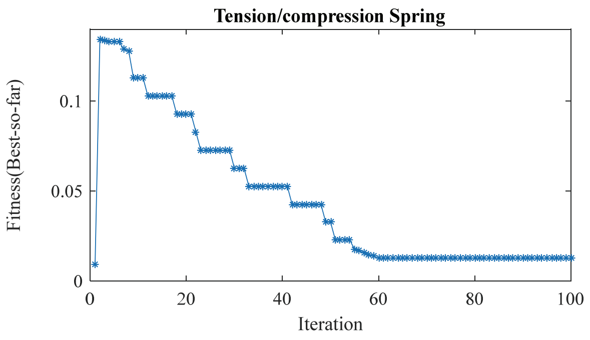


Fig. 9 - Convergence history for the tension/compression spring

4.2.3. Cantilever Beam Design Problem

Fig. 10 shows a cantilever beam consisting of five hollow square blocks. There is also a vertical load applied to the free end of the beam while the other side of the beam is rigidly supported. The aim is to minimize the weight of the beam, while the vertical displacement is defined as a constraint that should not be violated by the final optimal design. The design variables are the heights (or widths) of the different hollow blocks with fixed thickness (t = 2/3).

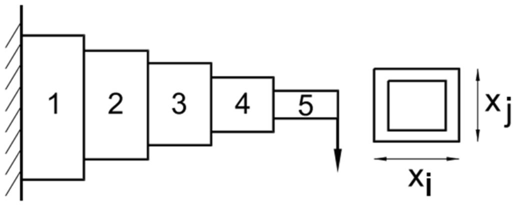


Fig. 10 - Cantilever beam design problem also indicating the design variables

Based on the discretization of five elements, the optimization problem was formulated by Svanberg [76] in a closed form:

$$\min f(x) = 0.0624(x_1 + x_2 + x_3 + x_4 + x_5).$$
$$\text{Subject to:} \tag{14}$$
$$g(x) = \frac{61}{{x_1}^3} + \frac{37}{{x_2}^3} + \frac{19}{{x_3}^3} + \frac{7}{{x_4}^3} + \frac{1}{{x_5}^3} \leq 1.$$

The bounds on the design variables are:

$$0.01 \leq x_1, x_2, x_3, x_4, x_5 \leq 15.0. \tag{15}$$

Having solved the problem using proposed method, the results were recorded in Table 8, where a comparison is made between method of moving asymptotes (MMA) [77], generalized convex approximation (GCA_I) [77], GCA_II [77], moth-flame optimization (MFO) algorithm [78], and symbiotic organisms search (SOS) [79]. It shows that the proposed algorithm exhibits a better performance compared to other algorithms. The statistical data on various independent runs compared with the results obtained by others in the literature are listed in Table 10.

Table 9 - Comparison of results for cantilever beam design problem

	SOS [79]	MMA [77]	GCA-I [77]	GCA-II [78]	MFO [78]	IMOEA
x_1	6.01878	6.0100	6.0100	6.0100	5.98487	6.0367
x_2	5.30344	5.3000	5.3040	5.3000	5.31673	5.2859
x_3	4.49587	4.4900	4.4900	4.4900	4.49733	4.5067
x_4	3.49896	3.4900	3.4980	3.4900	3.51362	3.4858
x_5	2.15564	2.1500	2.1500	2.1500	2.16162	2.1592
g	*0.000139*	NA	NA	NA	NA	-6.251037278692806e-06
f_{min}	1.33996	1.3400	1.3400	1.3400	1.339988	1.339996320000000
No. Analyses	15000	NA	NA	NA	30,000	12000

Table 10 - Statistical results using different approaches for cantilever beam design problem

	Best	Mean	Worst	Std.	No. Analyses
Cheng and Prayogo [79]	1.33996	NA	NA	NA	15,000
Chickermane and Gea [77]	1.34	NA	NA	NA	NA
Mirjalili et al. [78]	1.33998	NA	NA	NA	30,000
Mirjalili et al. [78]	1.33996	NA	NA	NA	15,000
IMOEA	1.339996320000000	1.340907	1.3458	0.001643	12000

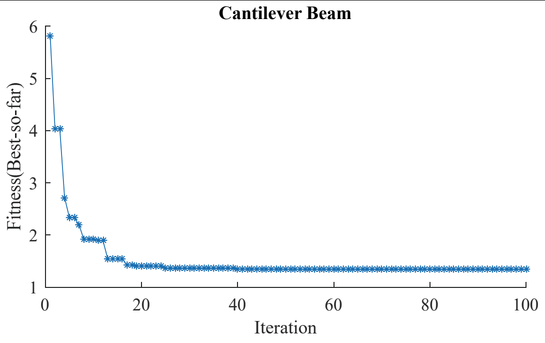


Fig. 11 - Convergence history for the Cantilever beam design problem

Fig. 11 illustrated the convergence history for the cantilever design problem using proposed algorithm. As is clear from Fig. 11, before reaching 20 iteration proposed algorithm almost found the global optimum but the process continuous until reach the maximum iteration of 100.

4.2.4. Three-Bar Truss Design Problems

This case considers a three-bar truss design problem, as shown in Fig. 11. The objective of this case is to minimize the volume of a statistically loaded three-bar truss subject to stress (σ) constraints. The problem involves two decision variables: cross sectional areas x_1 and x_2.

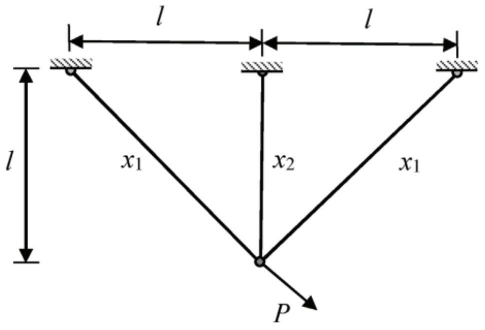


Fig. 12 - Schematic view of the three-bar truss indicating the design variables

The design variables are bounded as:

$$0 \leq x_1 \leq 1, 0 \leq x_2 \leq 1. \tag{16}$$

where $l = 100$ and $P = 2\ KN$ and $\sigma = 2\ {}^{KN}/_{cm^2}$.

This design is a nonlinear fractional programming problem. This problem has been solved by a hybrid method based on PSO and differential evolution (PSO-DE)[80], Society and Civilization optimization algorithm[81] and PSO-SA algorithm[82]. The results are given in Table 11 and 12 showing that the results of the proposed algorithm are very competitive with less function evaluation numbers.

Table 11 - Comparison of results for three-bar truss design problems

	PSO-DE[80]	SAC[81]	PSO-SA[82]	IMOEA
x_1	NA	NA	NA	0.78883
x_2	NA	NA	NA	0.40781
f_{min}	263.895843	263.895846	263.895918	263.8958168813538
No. Analyses	17,600	17,610	12,530	12000

Table 12 - Statistical results using different approaches for three-bar truss design problem

	Best	Mean	Worst	Std.	No. Analyses
Liu et al.[80]	263.89584338	263.89584338	263.89584338	4.5E-10	17,600
Ray & Liew[81]	263.89584654	263.90335672	263.96975638	1.3E-02	17,610
Javidrad &Nazari[82]	263.89591830	263.89656630	263.89699970	9.8E-08	12,530
IMOEA	263.895816881	263.895816881	263.89584338	1.3 E-07	12000

It is noted that although the IMOEA method has better objective function values than the other algorithms with minimum number of iterations required. The stability of solution was 100%. The overall results confirm that the IMOEA can have the substantial capability in handling constrained optimization problems.

Fig. 13 shows the convergence history for the three-bar truss design problem using proposed algorithm. As is clear from Fig. 13, before reaching 10 iteration proposed algorithm almost found the global optimum but the process continuous until reach the maximum iteration of 100. It is important to note that in the first iteration of this problem, proposed algorithm could not find any individuals in feasible area but its procedure continued and until find an answer which suits problem constraints.

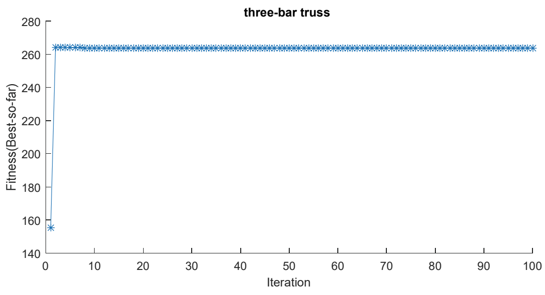


Fig. 13 - Convergence history for the three-bar truss design problem

5. DISCUSSION

In this paper, to evaluate efficiency of IMOEA, different kind of problems, including linear, nonlinear, Quadratic, Cubic, unconstrained, constrained and continuous single objective problems are solved. According to the results in Table 4, IMOEA finds better answer rather than other algorithms in most of cases with lower number of calculations. Also, low standard deviation which is reported in Table 4 almost guarantee the acceptable performance and obtaining good result of algorithm in each run. It is obvious that if the number of iterations or individuals increases, IMOEA has a better chance to acquire better result and find global optimum. In case of engineering design problems, Table 5-12 showing results of IMOEA compare with other studies with reporting important data. In these tables low standard deviation and also finding better solution with less calculation can be observed. In fig. 7,9,11 and 13 convergence history of IMOEA for solving engineering problems is depicted. These figures reflects the fast convergence speed of IMOEA in its process. So, after interpretation of results, acceptable Performance and efficiency of IMOEA is proven. But it should be noted that correct selection of operators is very crucial. Like other optimization algorithms (c_1, c_2 and ω in PSO, α and β in ACO and etc.), IMOEA is very sensitive to its parameters in a way that with incorrect assignment, the algorithm cannot find the global optimum and deviate to a local optimum [16, 83-85]. So, finding appropriate parameters for operators depends on the skills of user of the algorithm. IMOEA will converge to an optimum because of its process in case of large number of design variables, it may face some difficulties in finding the global answer. Though, this problem can easily be solved by selecting appropriate parameters it is still a restriction of IMOEA similar to all other meta-heuristic algorithms [5-7]. On the other hand, in case of enormous search space, finding global optimum is hard and time-consuming for IMOEA. For solving mentioned problems, considering special operators which is designed for these purpose is highly recommended[86, 87].

6. CONCLUSION

To address the issue of the trade-off between the convergence and speed, this paper proposes a method which converts a SOP into an equivalent MOP with two objectives by basic definition which used in optimizations algorithm. Then an effective improved multi-objective evolutionary algorithm (IMOEA) is applied to the MOP, thus the SOP is solved. Comparison between different Mutation and Crossover leads to wise selection of these two important operators. The dynamic environment of gradually increasing mutation and decreasing crossover drives the trade-off between the convergence and the diversity through the whole evolutionary process. Also considering feasible elites for the crossover and mutation operators produced better offspring. Furthermore, new selection strategy enhances the search ability of the algorithm and provide faster convergence by including non-feasible individuals. The proposed method has been validated by optimization of various benchmark functions. The results show that, in general, the proposed method has good stability together with reasonable speed in finding global optimum point. Moreover, the performance of IMOEA is statistically better than that of the other state-of-the-art algorithms. Furthermore, the IMOEA resulted in the best mean solution, solution, and standard deviation and the least function evaluations compared to that of all of the algorithms tested in the civil engineering design problems. In overall, it is concluded that the proposed method can effectively and reliably be used for constraint optimization purposes especially for complex civil-engineering optimization problems

References

[1] Rao, S.S., *Engineering optimization: theory and practice*. 2009: John Wiley & Sons.

[2] Bazaraa, M.S., J.J. Jarvis, and H.D. Sherali, *Linear programming and network flows*. 2011: John Wiley & Sons.

[3] Holland, J.H.J.S.a., *Genetic algorithms*. 1992. **267**(1): p. 66-73.

[4] Eberhart, R. and J. Kennedy. *A new optimizer using particle swarm theory*. in *Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on*. 1995. IEEE.

[5] Atashpaz-Gargari, E. and C. Lucas. *Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition*. in *Evolutionary computation, 2007. CEC 2007. IEEE Congress on*. 2007. IEEE.

[6] Rao, R. and V. Patel, *An elitist teaching-learning-based optimization algorithm for solving complex constrained optimization problems.* International Journal of Industrial Engineering Computations, 2012. **3**(4): p. 535-560.

[7] Ghaemi, M. and M.-R. Feizi-Derakhshi, *Forest optimization algorithm.* Expert Systems with Applications, 2014. **41**(15): p. 6676-6687.

[8] Jordehi, A.R., *Brainstorm optimisation algorithm (BSOA): An efficient algorithm for finding optimal location and setting of FACTS devices in electric power systems.* International Journal of Electrical Power & Energy Systems, 2015. **69**: p. 48-57.

[9] Dai, T., et al., *Stiffness optimisation of coupled shear wall structure by modified genetic algorithm.* 2016. **20**(8): p. 861-876.

[10] Mirjalili, S. and A. Lewis, *The whale optimization algorithm.* Advances in Engineering Software, 2016. **95**: p. 51-67.

[11] Varaee, H. and M.R. Ghasemi, *Engineering optimization based on ideal gas molecular movement algorithm.* Engineering with Computers, 2017. **33**(1): p. 71-93.

[12] Tabari, A. and A. Ahmad, *A new optimization method: Electro-Search algorithm.* Computers & Chemical Engineering, 2017. **103**: p. 1-11.

[13] TOĞAN, V. and M.A.J.T.D. EIRGASH, *Time-Cost Trade-Off Optimization with a New Initial Population Approach.* 2018. **30**(6).

[14] Muhammad, A.A., et al., *Adoption of Virtual Reality (VR) for Site Layout Optimization of Construction Projects.* 2019. **31**(2).

[15] AZAD, S.K. and A.J.T.D. Ebru, *Cost Efficient Design of Mechanically Stabilized Earth Walls Using Adaptive Dimensional Search Algorithm.* **31**(4).

[16] Coello, C.A.C., *Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art.* Computer methods in applied mechanics and engineering, 2002. **191**(11): p. 1245-1287.

[17] Kucukkoc, I. and D.Z. Zhang, *Balancing of parallel U-shaped assembly lines.* Computers & Operations Research, 2015. **64**: p. 233-244.

[18] Chou, C.-H., S.-C. Hsieh, and C.-J. Qiu, *Hybrid genetic algorithm and fuzzy clustering for bankruptcy prediction.* Applied Soft Computing, 2017. **56**: p. 298-316.

[19] Araghi, S., et al., *Influence of meta-heuristic optimization on the performance of adaptive interval type2-fuzzy traffic signal controllers.* Expert Systems with Applications, 2017. **71**: p. 493-503.

[20] Tosta, T.A.A., et al., *Computational method for unsupervised segmentation of lymphoma histological images based on fuzzy 3-partition entropy and genetic algorithm.* Expert Systems with Applications, 2017. **81**: p. 223-243.

[21] Yang, G., Y. Wang, and L. Guo, *A sparser reduced set density estimator by introducing weighted l 1 penalty term.* Pattern Recognition Letters, 2015. **58**: p. 15-22.

[22] Dong, Z. and W. Zhu, *An improvement of the penalty decomposition method for sparse approximation.* Signal Processing, 2015. **113**: p. 52-60.

[23] Kia, S.S., *Distributed optimal resource allocation over networked systems and use of an e-exact penalty function.* IFAC-PapersOnLine, 2016. **49**(4): p. 13-18.

[24] Tang, K.-Z., T.-K. Sun, and J.-Y. Yang, *An improved genetic algorithm based on a novel selection strategy for nonlinear programming problems.* Computers & Chemical Engineering, 2011. **35**(4): p. 615-621.

[25] Long, Q., *A constraint handling technique for constrained multi-objective genetic algorithm.* Swarm and Evolutionary Computation, 2014. **15**: p. 66-79.

[26] de Paula Garcia, R., et al., *A rank-based constraint handling technique for engineering design optimization problems solved by genetic algorithms.* Computers & Structures, 2017. **187**: p. 77-87.

[27] Coello, C.A.C., G.B. Lamont, and D.A. Van Veldhuizen, *Evolutionary algorithms for solving multi-objective problems*. Vol. 5. 2007: Springer.

[28] Dhiman, G. and V.J.K.-B.S. Kumar, *Multi-objective spotted hyena optimizer: A Multi-objective optimization algorithm for engineering problems.* 2018. **150**: p. 175-197.

[29] Deb, K., et al., *A fast and elitist multiobjective genetic algorithm: NSGA-II.* IEEE transactions on evolutionary computation, 2002. **6**(2): p. 182-197.

[30] Monfared, S.A.H., et al., *Water Quality Planning in Rivers: Assimilative Capacity and Dilution Flow.* Bulletin of environmental contamination and toxicology, 2017. **99**(5): p. 531-541.

[31] Hashemi Monfared, S. and M. Dehghani Darmian, *Evaluation of Appropriate Advective Transport Function for One-Dimensional Pollutant Simulation in Rivers.* International Journal of Environmental Research, 2016. **10**(1): p. 77-84.

[32] Noura, A. and F.J.A.M.S. Saljooghi, *Determining feasible solution in imprecise linear inequality systems.* 2008. **2**(36): p. 1789-1797.

[33] Mostaghim, S. and J. Teich. *Strategies for finding good local guides in multi-objective particle swarm optimization (MOPSO).* in *Swarm Intelligence Symposium, 2003. SIS'03. Proceedings of the 2003 IEEE*. 2003. IEEE.

[34] Coello Coello, C. and M. Lechuga. *MOPSO: a proposal for multiple objective particle swarm optimization*. in *Proc., Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on*.

[35] Zhang, Q. and H.J.I.T.o.e.c. Li, *MOEA/D: A multiobjective evolutionary algorithm based on decomposition.* 2007. **11**(6): p. 712-731.

[36] Coello, C.A.C., *Use of a self-adaptive penalty approach for engineering optimization problems.* Computers in Industry, 2000. **41**(2): p. 113-127.

[37] Segura, C., et al., *Using multi-objective evolutionary algorithms for single-objective constrained and unconstrained optimization.* Annals of Operations Research, 2016. **240**(1): p. 217-250.

[38] Fonseca, C.M. and P.J. Fleming. *Genetic Algorithms for Multiobjective Optimization: FormulationDiscussion and Generalization*. in *Icga*. 1993.

[39] Srinivas, N. and K. Deb, *Muiltiobjective optimization using nondominated sorting in genetic algorithms.* Evolutionary computation, 1994. **2**(3): p. 221-248.

[40] Deb, K., et al. *A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II*. in *International Conference on Parallel Problem Solving From Nature*. 2000. Springer.

[41] Ngatchou, P., A. Zarei, and A. El-Sharkawi. *Pareto multi objective optimization*. in *Intelligent systems application to power systems, 2005. Proceedings of the 13th international conference on*. 2005. IEEE.

[42] Smith, J. and T.C. Fogarty. *Self adaptation of mutation rates in a steady state genetic algorithm*. in *Evolutionary Computation, 1996., Proceedings of IEEE International Conference on*. 1996. IEEE.

[43] Moon, C., et al., *An efficient genetic algorithm for the traveling salesman problem with precedence constraints.* European Journal of Operational Research, 2002. **140**(3): p. 606-617.

[44] Ho, W., et al., *A hybrid genetic algorithm for the multi-depot vehicle routing problem.* Engineering Applications of Artificial Intelligence, 2008. **21**(4): p. 548-557.

[45] Juang, C.-F., *A hybrid of genetic algorithm and particle swarm optimization for recurrent network design.* IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 2004. **34**(2): p. 997-1006.

[46] Mühlenbein, H. and D. Schlierkamp-Voosen, *Predictive models for the breeder genetic algorithm i. continuous parameter optimization.* Evolutionary computation, 1993. **1**(1): p. 25-49.

[47] Liang, J., et al., *Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization.* Journal of Applied Mechanics, 2006. **41**(8).

[48] Horn, J., N. Nafpliotis, and D.E. Goldberg. *A niched Pareto genetic algorithm for multiobjective optimization*. in *Evolutionary Computation, 1994. IEEE World Congress on Computational Intelligence., Proceedings of the First IEEE Conference on*. 1994. Ieee.

[49] Deep, K. and M. Thakur, *A new crossover operator for real coded genetic algorithms.* Applied mathematics and computation, 2007. **188**(1): p. 895-911.

[50] Herrera, F., M. Lozano, and A.M. Sánchez, *A taxonomy for the crossover operator for real-coded genetic algorithms: An experimental study.* International Journal of Intelligent Systems, 2003. **18**(3): p. 309-338.

[51] Weile, D.S. and E. Michielssen, *Genetic algorithm optimization applied to electromagnetics: A review.* IEEE Transactions on Antennas and Propagation, 1997. **45**(3): p. 343-353.

[52] Schott, J.R., *Fault Tolerant Design Using Single and Multicriteria Genetic Algorithm Optimization*. 1995, AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OH.

[53] Patel, V.K. and V.J. Savsani, *Heat transfer search (HTS): a novel optimization algorithm.* Information Sciences, 2015. **324**: p. 217-246.

[54] Hamida, S.B. and M. Schoenauer. *ASCHEA: New results using adaptive segregational constraint handling*. in *Evolutionary Computation, 2002. CEC'02. Proceedings of the 2002 Congress on*. 2002. IEEE.

[55] Karaboga, D. and B. Akay, *A modified artificial bee colony (ABC) algorithm for constrained optimization problems.* Applied soft computing, 2011. **11**(3): p. 3021-3031.

[56] Topal, U., et al., *Buckling load optimization of laminated plates resting on Pasternak foundation using TLBO.* J Structural Engineering and Mechanics, 2018. **67**(6): p. 617-628.

[57] Zhiyi, Y., Z. Kemin, and Q. Shengfang, *Topology optimization of reinforced concrete structure using composite truss-like model.* J Structural Engineering and Mechanics, 2018. **67**(1): p. 79-85.

[58] Noura, A. and F. Saljooghi, *Ranking decision making units in Fuzzy-DEA Using entropy.* Applied Mathematical Sciences, 2009. **3**(6): p. 287-295.

[59] Saljooghi, F.H. and M.M.J.A.J.o.A.S. Rayeni, *Distinguishing congestion and technical inefficiency in presence undesirable output.* 2011. **8**(9): p. 903.

[60] Artar, M. and A.J.T.D. Daloglu, *The Optimization of Multi-Storey Composite Steel Frames with Genetic Algorithm Including Dynamic Constraints.* 2015. **26**(2): p. 7077-7098.

[61] Mustafa, O., et al., *Construction Site Layout Planning: Application of Multi-Objective Particle Swarm Optimization.* **29**(6).

[62] Rayeni, M.M., F.H.J.I.J.o.S. Saljooghi, and O. Management, *Ranking and measuring efficiency using secondary goals of cross-efficiency evaluation–a study of railway efficiency in Iran.* 2014. **17**(1): p. 1-16.

[63] Bulut, B. and M.T.J.T.D. Yilmaz, *Analysis of the 2007 and 2013 Droughts in Turkey by NOAH Hydrological Model.* 2016. **27**(4): p. 7619-7634.

[64] Mahallati, M. and F.J.J.o.A.S. Saljooghi, *Performance assessment of education institutions through interval DEA.* 2010. **10**: p. 2945-2949.

[65] Tözer, K.D., T. Çelik, and G.E.J.T.D. Gürcanlı, *Classification of Construction Accidents in Northern Cyprus.* 2018. **29**(2): p. 8295-8316.

[66] Coello, C.A.C. and C.S.P. Zacatenco, *List of references on constraint-handling techniques used with evolutionary algorithms.* Information Sciences, 2012. **191**: p. 146-168.

[67] He, Q. and L. Wang, *An effective co-evolutionary particle swarm optimization for constrained engineering design problems.* Engineering Applications of Artificial Intelligence, 2007. **20**(1): p. 89-99.

[68] Mezura-Montes, E. and C.A.C. Coello, *An empirical study about the usefulness of evolution strategies to solve constrained optimization problems.* International Journal of General Systems, 2008. **37**(4): p. 443-473.

[69] Mirjalili, S., S.M. Mirjalili, and A. Lewis, *Grey wolf optimizer.* Advances in Engineering Software, 2014. **69**: p. 46-61.

[70] Kaveh, A. and S. Talatahari, *An improved ant colony optimization for constrained engineering design problems.* Engineering Computations, 2010. **27**(1): p. 155-182.

[71] Belegundu, A.D. and J.S. Arora, *A study of mathematical programming methods for structural optimization. Part I: Theory.* International Journal for Numerical Methods in Engineering, 1985. **21**(9): p. 1583-1599.

[72] Arora, J., *Introduction to optimum design*. 2004: Academic Press.

[73] Eskandar, H., et al., *Water cycle algorithm–A novel metaheuristic optimization method for solving constrained engineering optimization problems.* Computers & Structures, 2012. **110**: p. 151-166.

[74] Kaveh, A. and S. Talatahari, *A novel heuristic optimization method: charged system search.* Acta Mechanica, 2010. **213**(3): p. 267-289.

[75] Kaveh, A. and V. Mahdavi, *Colliding bodies optimization: a novel meta-heuristic method.* Computers & Structures, 2014. **139**: p. 18-27.

[76] Svanberg, K., *The method of moving asymptotes—a new method for structural optimization.* International journal for numerical methods in engineering, 1987. **24**(2): p. 359-373.

[77] Chickermane, H. and H. Gea, *Structural optimization using a new local approximation method.* International journal for numerical methods in engineering, 1996. **39**(5): p. 829-846.

[78] Mirjalili, S., *Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm.* Knowledge-Based Systems, 2015. **89**: p. 228-249.

[79] Cheng, M.-Y. and D. Prayogo, *Symbiotic organisms search: a new metaheuristic optimization algorithm.* Computers & Structures, 2014. **139**: p. 98-112.

[80] Liu, H., Z. Cai, and Y. Wang, *Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization.* Applied Soft Computing, 2010. **10**(2): p. 629-640.

[81] Ray, T. and K.M. Liew, *Society and civilization: An optimization algorithm based on the simulation of social behavior.* IEEE Transactions on Evolutionary Computation, 2003. **7**(4): p. 386-396.

[82] Javidrad, F. and M. Nazari, *A new hybrid particle swarm and simulated annealing stochastic optimization method.* Applied Soft Computing, 2017. **60**: p. 634-654.

[83] Yılmaz, M., et al., *Uydu Kaynaklı Yağmur Verilerinin Hata Oranlarının Deniz Kıyılarına Olan Uzaklığa Bağlı Analizi.* 2017. **28**(3): p. 7993-8005.

[84] Dorigo, M. and G. Di Caro. *Ant colony optimization: a new meta-heuristic*. in *Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406)*. 1999. IEEE.

[85] Shafigh, P., S.Y. Hadi, and E.J.I.s. Sohrab, *Gravitation based classification.* 2013. **220**: p. 319-330.

[86] Lin, Y., et al., *A hybrid differential evolution algorithm for mixed-variable optimization problems.* 2018. **466**: p. 170-188.

[87] Deng, H., et al., *Ranking-based biased learning swarm optimizer for large-scale optimization.* 2019. **493**: p. 120-137.