Appl Intell
DOI 10.1007/s10489-016-0825-8

Multi-objective ant lion optimizer: a multi-objective optimization algorithm for solving engineering problems

Seyedali Mirjalili[1,2] · **Pradeep Jangir**[3] · **Shahrzad Saremi**[1,2]

Abstract This paper proposes a multi-objective version of the recently proposed Ant Lion Optimizer (ALO) called Multi-Objective Ant Lion Optimizer (MOALO). A repository is first employed to store non-dominated Pareto optimal solutions obtained so far. Solutions are then chosen from this repository using a roulette wheel mechanism based on the coverage of solutions as antlions to guide ants towards promising regions of multi-objective search spaces. To prove the effectiveness of the algorithm proposed, a set of standard unconstrained and constrained test functions is employed. Also, the algorithm is applied to a variety of multi-objective engineering design problems: cantilever beam design, brushless dc wheel motor design, disk brake design, 4-bar truss design, safety isolating transformer design, speed reduced design, and welded beam deign. The results are verified by comparing MOALO against NSGA-II and MOPSO. The results of the proposed algorithm on the test functions show that this algorithm benefits from high convergence and coverage. The results of the algorithm on the engineering design problems demonstrate its applicability is solving challenging real-world problems as well.

✉ Seyedali Mirjalili
ali.mirjalili@gmail.com

[1] School of Information and Communication Technology, Griffith University, Nathan Campus, Brisbane, QLD 4111, Australia

[2] Griffith College, Mt Gravatt, Brisbane, QLD 4122, Australia

[3] Lukhdhirji Engineering College, Morbi-Rajkot, Gujarat, India

Keywords Ant lion optimizer · Multi-objective optimization · Optimization · Evolutionary algorithm · Multi-criterion optimization · Heuristic · Algorithm · Meta-heuristic · Engineering optimization

1 Introduction

In recent years, computers have become very popular in different fields for solving challenging problems. Computer-aided design is a field that emphasizes the use of computers in solving problems and designing systems. In the past, the design process of a system would have required direct human involvements. For instance, if a designer wanted to find an optimal shape for a rocket, he would have to first create a prototype and then use a wind tunnel to test. Obviously, such a design approach was very expensive and time consuming. The more complex the system was, the more time and cost the entire project required.

The invention of computers speeded up the design process significantly a couple of decades ago. This means that people are now able to use computers to design a system without even the need for a single prototype. As a result, not only the cost but also the time of the design process is substantially less than before. In spite of the fact that the machine is now a great assistance, designing a system this way still requires direct human involvements. This results in a series of trial and errors where the designer tries to design an efficient system. It is undeniable that a designer is prone to mistakes, which makes the design process unreliable.

Another revolutionary approach was the use of machine to not only simulate a system but also design it. In this case, a designer mostly set up the system and utilize a computer program to find the optimal designs. This automatic design process is still the current approach in science and industry. The main advantages are high speed, low cost, and high reliability. However, the main drawback is the complexity of the design process and the need for finding a suitable approach for designing a system using computers.

Optimization techniques are considered as one of the best techniques for finding optimal designs using machines. Conventional optimization algorithms are mostly depreciated because of their main drawback: local optima stagnation [1, 2]. The main alternatives for designers are stochastic optimization techniques. Such approaches consider problems as black boxes and approximate the optimal designs. They initialize the optimization process with a set of random candidate solutions for a given problem and improve them over a pre-defined number of steps. Despite the advantages of these methods, optimization of real problems require addressing various difficulties: multiple-objectives [3], constraints [4], uncertainties [5], local solutions [6], deceptive global solutions [7], etc. To address these issues, optimization algorithms should be equipped with different operators.

Multi-objective optimization [8], which is the main focus of this work, deals with finding solutions for problems with more than one objective. There is more than one solution for a multi-objective problem due to the nature of such problems [9]. By contrast, a single-objective problem has only one global optimum. Addressing multiple objectives, which are often in conflict, is the main challenge in multi-objective optimization. The duty of a stochastic multi-objective optimization algorithm is to determine the set of best trade-offs between the objectives, the so called Pareto optimal set.

There are two main approaches in the literature of multi-objective optimization using stochastic optimization techniques: *a posteriori* versus *a priori* [3, 10]. For *a priori* approaches, a multi-objective optimization problem is converted to a single-objective one by aggregating the objectives. A set of weights defines how important the objectives are and is normally provided by an expert in the problem domain. After the objective aggregation, a single-objective optimization algorithm is able to readily solve the problem. The main drawbacks of such methods is that an algorithm should be run multiple times to determine the Pareto optimal set. In addition, there is a need to consult with an expert and some special Pareto optimal fronts cannot be determined with this approach [11–13].

A *posterior* approaches benefit from maintaining multi-objective formulation of a multi-objective problems and finding the Pareto optimal set in just one run. Also, another advantage is that any kind of Pareto front can be determined with these algorithms. However, they require higher computational cost and addressing multiple objectives at the same time. The literature shows that such methods have been widely used since the invention and are able to solve real-world problems. The most popular algorithms in the literature are: Non-dominated Sorting Genetic Algorithm (NSGA) [14–16] and Multi-objective Particle Swarm Optimization (MOPSO) [17, 18]. The application of these two algorithms can be found in different fields as the literature shows [3].

Most of the recently proposed single-objective algorithms have been equipped with operators to solve multi-objective problems as well. Some of the most recent ones are Multi-objective Bee Algorithm [19], Multi-objective Bat Algorithm [20], Multi-objective Grey Wolf Optimizer (MOGWO) [21], etc.

The No-Free Lunch [22] theorem for optimization allows researchers to propose new algorithms or improve the current ones because it logically proves that there is no optimization algorithm for solving all optimization problems. This applies to both single- and multi-objective optimization techniques. In an effort to solve optimization problems with multiple objectives, this work proposes a multi-objective version of the recently proposed Ant Lion Optimizer (ALO). Although the current algorithms in the literature are able to solve a variety of problems, according to the NFL theorem, they are not able to solve all optimization problems. This work proposes the multi-objective ALO with the hope to better solve some or new problems. The rest of the paper is organized as follows.

Section 2 provides the literature review. Section 3 proposes the Multi-objective Ant Lion Optimizer. Section 4 presents, discusses, and analyses the results on the test and engineering problems employed. Finally, Section 5 concludes the work and suggests future works.

2 Literature review

In single-objective optimization, there is only one solution as the global optimum. This is because of the unary objective in single-objective problems and the existence of one best solution. Comparison of solutions is easy when considering one objective and is done by the relational operators: $>$, $\geq$, $<$, $\leq$, or $=$. The nature of such problems allows optimization problems to conveniently compare the candidate solutions and eventually find the best one. In multi-objective problems, however, solutions should be compared with more than one objective (criterion). Multi-objective optimization can be formulated as a minimization problem as follows:

$$Minimize: \quad F(\vec{x}) = \{f_1(\vec{x}), f_2(\vec{x}), \ldots, f_o(\vec{x})\} \tag{2.1}$$

$$Subject\ to: \quad g_i(\vec{x}) \geq 0, \qquad i = 1, 2, \ldots, m \tag{2.2}$$

$$h_i(\vec{x}) = 0, \qquad i = 1, 2, \ldots, p \tag{2.3}$$

$$L_i \leq x_i \leq U_i, \quad i = 1, 2, \ldots, n \tag{2.4}$$

where $n$is the number of variables, o is the number of objective functions, m is the number of inequality constraints, p is the number of equality constraints, g_i is the *i-th* inequality constraints, h_i indicates the *i-th* equality constraints, and *[Li,Ui]* are the boundaries of *i-th* variable.

Obviously, relational operators are no longer effective for comparing solutions of a problem with multiple objectives. There should be other operators in this case. Without the loss of generality, the four main definitions in multi-objective optimization (minimization) are as follows:

Definition 1 (Pareto Dominance) Assuming two vectors such as: $\vec{x} = (x_1, x_2, \ldots, x_k)$ and $\vec{y} = (y_1, y_2, \ldots, y_k)$. Vector $\vec{x}$ is said to dominate vector $\vec{y}$ (denote as $\vec{x} \prec \vec{y}$) if and only if:

$$\forall i \in \{1, 2, \ldots, k\} : f_i(\vec{x}) \le f_i(\vec{y}) \wedge \exists i \in \{1, 2, \ldots, k\} : f_i(\vec{x}) < f_i(\vec{y}) \tag{2.5}$$

The definition of Pareto optimality is as follows [23–25]:

Definition 2 (Pareto Optimality [23]) A solution $\vec{x} \in X$ is called Pareto-optimal if and only if:

$$\{\nexists \vec{y} \in X | \vec{y} \prec \vec{x}\} \tag{2.6}$$

Definition 3 (Pareto optimal set) The set all Pareto-optimal solutions is called Pareto set as follows:

$$P_s := \{\vec{x}, \vec{y} \in X | \nexists \vec{y} \prec \vec{x}\} \tag{2.7}$$

Definition 4 (Pareto optimal front) A set containing the value of objective functions for Pareto solutions set:

$$P_f := \{f(\vec{x}) | \vec{x} \in P_s\} \tag{2.8}$$

For solving a multi-objective problem, we have to find the Pareto optimal set, which is the set of solutions representing the best trade-offs between objectives.

Over the course of past decade, a significant number of multi-objective algorithms has been developed. Between the stochastic population-based algorithms, which is the focus of this work, the most well-regarded ones are: Strength-Pareto Evolutionary Algorithm (SPEA) [26, 27], Non-dominated Sorting Genetic Algorithm [28], Non-dominated sorting Genetic Algorithm version 2 (NSGA-II) [16], Multi-Objective Particle Swarm Optimization (MOPSO) [18], Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D) [29], Pareto Archived Evolution Strategy (PAES) [30], and Pareto-frontier Differential Evolution (PDE) [31].

The general frameworks of all population-based multi-objective algorithms are almost identical. They start the optimization process with multiple candidate solutions. Such solutions are compared using the Pareto dominance operator. In each step of optimization, the non-dominated solutions are stored in a repository and the algorithm tries to improve them in the next iteration(s). What make an algorithm different from another is the use of different methods to enhance the non-dominated solutions.

Improving the non-dominated solutions using stochastic algorithms should be done in terms of two perspectives: convergence (accuracy) and coverage (distribution) [32]. The former refers to the process of improving the accuracy of the non-dominated solutions. The ultimate goal is to find approximations very close to the true Pareto optimal solutions. In the latter case, an algorithm should try to improve the distribution of the non-dominated solutions to cover the entire true Pareto optimal front. This is a very important factor in *a posteriori* approaches, in which a wide range of solutions should be found for decision making.

The main challenge in multi-objective optimization using stochastic algorithms is that the convergence and coverage are in conflict. If an algorithm only concentrates on improving the accuracy of non-dominated solutions, the coverage will be poor. By contrast, a mere consideration of the coverage negatively impacts the accuracy of the non-dominated solutions. Most of the current algorithms periodically balance convergence and coverage to find very accurate approximation of the Pareto optimal solutions with uniform distribution along all objectives.

For convergence, normally, the main mechanism of convergence in the single-objective version of an algorithm is sufficient. For instance, in Particle Swarm Optimization (PSO) [33, 34], the solutions tends towards the global best. If the global best be replaced with one non-dominated solution, the particles will be able to improve its accuracy as they do in a single-objective search space. For improving coverage, however, the search should be guided towards different solutions. For instance, the *gbet* in PSO can be replaced with a random non-dominated solution so that particles improve different regions of the Pareto optimal front obtained. The main challenge here is the selection of non-dominated solutions to guarantee improving the distribution of Pareto optimal solutions.

There are different approaches in the literature for improving the coverage of an algorithm. Archive and leader selection in MOPSO, non-dominated sorting mechanism in NSGA, and niching [35–37] are the most popular approaches. In the next section, the multi-objective version of the recently proposed ALO is proposed as an alternative approach for finding Pareto optimal solutions of multi-objective problems.

3 Multi-objective ant lion optimizer (MOALO)

In order to propose the multi-objective version of the ALO algorithm [38], the fundamentals of this algorithm should be discussed first. An algorithm should follow the same search behaviour to be considered as an extended version of the same algorithm. The ALO algorithm mimics the hunting mechanism of antlions and the interaction of their favourite prey, ants, with them.

Similarly to other population-based algorithms, ALO approximates the optimal solutions for optimization problems with employing a set of random solutions. This set is improved based on the principles inspired from the interaction between antlions and ants. There are two populations in the ALO algorithm: set of ants and set of antlions. The general steps of ALO to change these two sets and eventually estimate the global optimum for a given optimization problem are as follows:

a) The ant set is initialized with random values and are the main search agents in the ALO.
b) The fitness value of each ant is evaluated using an objective function in each iteration.
c) Ants move over the search space using random walks around the antlions.
d) The population of antlions is never evaluated. In fact, antlions assumed to be on the location of ants in the first iteration and relocate to the new positions of ants in the rest of iterations if the ants become better.
e) There is one antlion assigned to each ant and updates its position if the ant becomes fitter.
f) There is also an elite antlion which impacts the movement of ants regardless of their distance.
g) If any antlion becomes better than the elite, it will be replaced with the elite.
h) Steps *b* to *g* are repeatedly executed until the satisfaction of an end criterion.
i) The position and fitness value of the elite antlion are returned as the best estimation for the global optimum.

The main responsibility of ants is to explore the search space. They are required to move around the search space using a random walk. The antlions maintain the best position obtained by the ants and guide the search of ants towards the promising regions of the search space. In order to solve optimization problems, the ALO algorithm mimics random walk of ants, entrapment in an antlion pit, constructing a pit, sliding ant towards antlions, catching prey and re-constructing the pit, and elitism. The mathematical model and programming modules proposed for each of these steps are presented in the following paragraphs.

The original random walk utilized in the ALO algorithm to simulate the random walk of ants is as follows:

$$X(t) = [0, cumsum(2r(t_1) - 1), cumsum(2r(t_2) - 1), \ldots, cumsum(2r(t_n) - 1)] \tag{3.1}$$

where *cumsum* calculates the cumulative sum, n is the maximum number of iteration, $t$shows the step of random walk (iteration in this study), and $r(t) = \begin{cases} 1 \; if \; rand > 0.5 \\ 0 \; if \; rand \leq 0.5 \end{cases}$ is a stochastic function where t shows the step of random walk (iteration in this study) and *rand* is a random number generated with uniform distribution in the interval of [0,1].

In order to keep the random walk in the boundaries of the search space and prevent the ants from overshooting, the random walks should be normalized using the following equation:

$$X_i^t = \frac{\left(X_i^t - a_i\right) \times \left(d_i^t - c_i^t\right)}{\left(b_i - a_i\right)} + c_i^t \tag{3.2}$$

where c_i^t is the minimum of *i-th* variable at *t-th* iteration, d_i^t indicates the maximum of *i-th* variable at *t-th* iteration, a_i is the minimum of random walk of *i-th* variable, and b_i is the maximum of random walk in *i-th* variable.

ALO simulates the entrapment of ants in antlions pits by changing the random walks around antlions. The following equations have been proposed in this regard:

$$c_i^t = Antlion_j^t + c^t \tag{3.3}$$

$$d_i^t = Antlion_j^t + d^t \tag{3.4}$$

where c^t is the minimum of all variables at *t-th* iteration, d^t indicates the vector including the maximum of all variables at *t-th* iteration, c_i^t is the minimum of all variables for *i-th* ant, d_i^t is the maximum of all variables for *i-th* ant, and $Antlion_j^t$ shows the position of the selected *j-th* antlion at *t-th* iteration.

In nature, bigger antlions construct bigger pits to increase their chance of survival. In order to simulate this, ALO utilizes a roulette wheel operator that selects antlions based on their fitness value. The roulette wheel assists fitter antlions to attract more ants.

For mimicking the sliding ants towards antlions, the boundaries of random walks should be decreased adaptively as follows:

$$c^t = \frac{c^t}{I} \tag{3.5}$$

$$d^t = \frac{d^t}{I} \tag{3.6}$$

where I is a ratio, c^t is the minimum of all variables at *t-th* iteration, d^t indicates the vector including the maximum of all variables at *t-th* iteration.

In the above equations, $I = 1 + 10^w \frac{t}{T}$ where t is the current iteration, T is the maximum number of iterations, and w is defined based on the current iteration ($w = 2$ when $t > 0.1T$, $w = 3$ when $t > 0.5T$, $w = 4$ when $t > 0.75T$, $w = 5$ when $t > 0.9T$, and $w = 6$ when $t > 0.95T$). The parameter w in the equation for I is able to adjust the accuracy level of exploitation.

The second to last step in ALO is catching the ant and reconstructing the pit. The following equation simulates this:

$$Antlion_j^t = Ant_i^t \qquad if\ f\left(Ant_i^t\right) < f\left(Antlion_j^t\right) \tag{3.7}$$

where t shows the current iteration, $Antlion_j^t$ shows the position of selected *j-th* antlion at *t-th* iteration, and Ant_i^t indicates the position of *i-th* ant at *t-th* iteration.

The last operator in ALO is elitism, in which the fittest antlion formed during optimization is stored. This is the only antlion that is able to have an impact on all ants. This means that the random walks on antlions gravitates toward a selected antlion (chosen using the roulette wheel) and the elite antlion. The equation to consider both of them is as follows:

$$Ant_i^t = \frac{R_A^t + R_E^t}{2} \tag{3.8}$$

where Ant_i^t indicates the position of *i-th* ant at *t-th* iteration, R_A^t is the random walk around the antlion selected by the roulette wheel at *t-th* iteration, and R_E^t is the random walk around the elite at *t-th* iteration.

As mentioned in the literature review, there are different approaches for finding and storing Pareto optimal solutions using heuristic algorithms. In this work, we employ an archive to store Pareto optimal solutions. Obviously, the convergence of the MOALO algorithm inherits from the ALO algorithm. If we pick one solution from the archive, the ALO algorithm will be able to improve its quality. However, finding the Pareto optimal solutions set with a high diversity is challenging.

To overcome this challenge, we have inspired from the MOPSO algorithm and utilized the leader selection and archive maintenance. Obviously, there should be a limit for the archive and solutions should be chosen from the archive in a way to improve the distribution. For measuring the distribution of the solutions in the archive, we use niching. In this approach, the vicinity of each solution is investigated considering a pre-defined radius. The number of solutions in the vicinity is then counted and considered as the measure of distribution. To improve the distribution of the solutions in the archive, we considered two mechanisms similarly to those in MOPSO. Firstly, the antlions are selected from the solutions with the least populated neighbourhood. The following equation is used in this regard that defines the probability of choosing a solution in the archive:

$$P_i = \frac{c}{N_i} \tag{3.9}$$

where c is a constant and should be greater than 1 and N_i is the number of solutions in the vicinity of the *i-th* solution.

Secondly, when the archive is full, the solutions with most populated neighbourhood are removed from the archive to accommodate new solutions. The following equation is used in this regard that defines the probability of removing a solution from the archive:

$$P_i = \frac{N_i}{c} \tag{3.10}$$

where c is a constant and should be greater than 1 and N_i is the number of solutions in the vicinity of the *i-th* solution.

In order to require ALO to solve multi-objective problems, (3.7) should be modified due to the nature of multi-objective problems.

$$Antlion_j^t = Ant_i^t \qquad if\ f\left(Ant_i^t\right) \prec f\left(Antlion_j^t\right) \tag{3.11}$$

where t shows the current iteration, $Antlion_j^t$ shows the position of selected *j-th* antlion at *t-th* iteration, and Ant_i^t indicates the position of *i-th* ant at *t-th* iteration.

Another modification is for the selection of random antlions and elite in (3.8). We utilize a roulette wheel and (3.8) to select a non-dominated solution from the archive. The rest of the operators in MOALO are identical to those in ALO. After all, the pseudocodes of the MOALO algorithm are shown in Fig. 1.

The computational complexity of the proposed MOALO algorithm is of $O(mn^2)$ where m is the number of objectives and n is the number of individuals. This is identical to the computational complexity of the well-known multi-objective algorithms such as MOPSO, NSGA-II, PAES, and SPEA2. However, the computational complexity of SPEA and NSGA is of $O(mn^3)$, which is worse than that of MOALO. Regarding the space required for the MOALO, it needs the same amount of memory compared to MOPSO. However, both MOALO and MOPSO require more memory

```
while the end condition is not met
    for every ant
        Select a random antlion from the archive
        Select the elite using Roulette wheel from the archive
        Update c and d using equations Eqs. (3.5) and (3.6)
        Create a random walk and normalize it using Eq. (3.1) and Eq. (3.2)
        Update the position of ant using (3.8)
    end for
    Calculate the objective values of all ants
    Update the archive
    if the archive is full
        Delete some solutions using Roulette wheel and Eq. (3.10) from the archive
        to accommodate new solutions.
    end
end while
return archive
```

Fig. 1 Pseudocodes of the MOALO algorithm

compared to NSGA-II due the use of archive to store the best non-dominated solutions obtained so far.

4 Results on test functions

This section benchmarks the performance of the proposed algorithm on 17 case studies including 5 unconstrained functions, 5 constrained functions, 7 and engineering design problems. The details of case study can be found in Appendices A, B, and C. It may be observed that test functions with diverse characteristics (especially different Pareto optimal front) are chosen to test the performance of MOALO from different perspectives. Although test functions can be very beneficial in examining an algorithm, solving real problems is always more challenging. This is why we have chosen a set of 7 multi-objective engineering design problems to confirm the applicability of the MOALO algorithm.

For results verification, two of the most well-regarded algorithms, such as MOPSO and NSGA-II, are employed. The results are collected and presented qualitatively and quantitatively in this section. The MOALO is run 10 times and the statistical results are reported in the tables below. The results are collected qualitatively and quantitatively. Note that we have utilized 100 iterations, 50 search agents, and an archive size of 100 in the experiments. The results of the comparative algorithms on some of the test functions are taken from [43, 45]. For the qualitative results, the best Pareto optimal front obtained by the algorithms are shown in

Table 1 Results of the multi-objective algorithms (using IGD) on the unconstrained test functions employed

Algorithm	ZDT1					ZDT2				
	Ave	Std.	Median	Best	Worst	Ave	Std.	Median	Best	Worst
MOALO	0.01524	0.005022	0.0166	0.0061	0.0209	0.01751	0.010977	0.0165	0.0050	0.0377
MOPSO	0.00422	0.003103	0.0037	0.0015	0.0101	0.00156	0.000174	0.0017	0.0013	0.0017
NSGA-II	0.05988	0.005436	0.0574	0.0546	0.0702	0.13972	0.026263	0.1258	0.1148	0.1834
Algorithm	ZDT3					ZDT1 with linear front				
	Ave	Std.	Median	Best	Worst	Ave	Std.	Median	Best	Worst
MOALO	0.03032	0.000969	0.0323	0.0303	0.0330	0.01982	0.007545	0.0196	0.0106	0.0330
MOPSO	0.03782	0.006297	0.0362	0.0308	0.0497	0.00922	0.005531	0.0098	0.0012	0.0165
NSGA-II	0.04166	0.008073	0.0403	0.0315	0.0557	0.08274	0.005422	0.0804	0.0773	0.0924
Algorithm	ZDT2 with 3 objectives									
	Ave	Std.	Median	Best	Worst					
MOALO	0.02629	0.004451	0.0288	0.0191	0.0315					
MOPSO	0.02032	0.001278	0.0203	0.0189	0.0225					
NSGA-II	0.0626	0.017888	0.0584	0.0371	0.0847					

the following figures. For the quantitative results, it should be noted that we have employed a wide range of performance metrics to quantify the performance of algorithm: Generational Distance (GD) [39], Inverted Generational Distance (IGD) [40], metric of spread [8], and metric of spacing [41]. The results of each set of test functions are presented and discussed in the following subsections.

4.1 Results on unconstrained test problems

As mentioned above, the first set of test problem consists of unconstrained test functions. Appendix A shows that the well-known ZDT test suite is employed [42]. The first three test functions in this work are identical to those in the original ZDT suite, but the last two test functions are slightly different in a same manner similar to [43]. We have deliberately modified ZDT1 and ZDT2 to create a linear and 3D front for benchmarking the performance of the MOALO algorithm proposed. After all, the results are presented in Table 1, Figs. 2 and 3.

Table 1 shows that the MOALO algorithm managed to outperform the NSGA-II algorithm significantly on all unconstrained test functions. The superiority can be seen in all the columns, showing a higher accuracy and better robustness of MOALO compared to NSGA-II. The MOALO algorithm, however, shows very competitive results in comparison with the MOPSO algorithm and occasionally outperforms it.

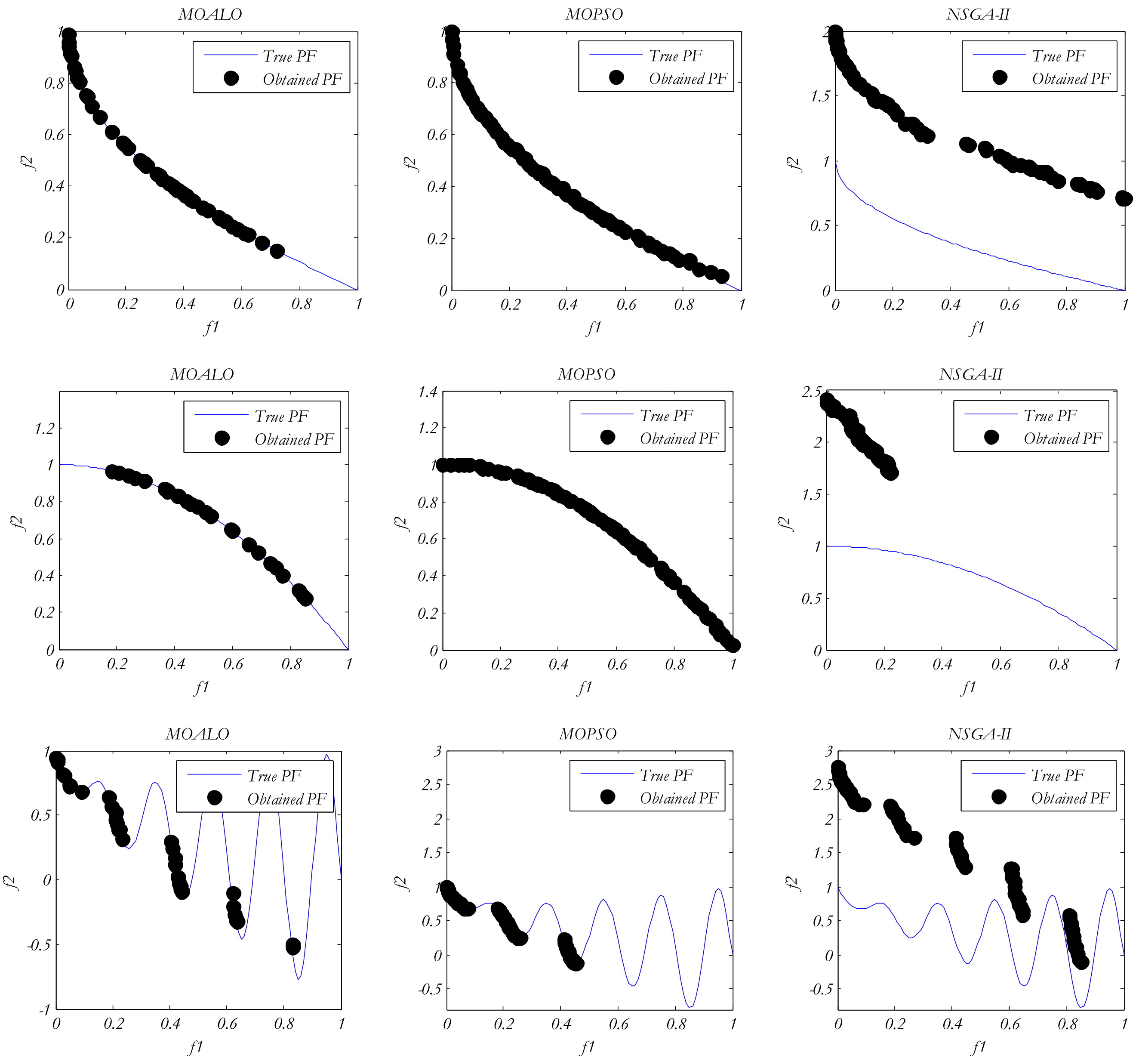

Fig. 2 Best Pareto optimal front obtained by the multi-objective algorithms on ZDT1, ZDT2, and ZDT3

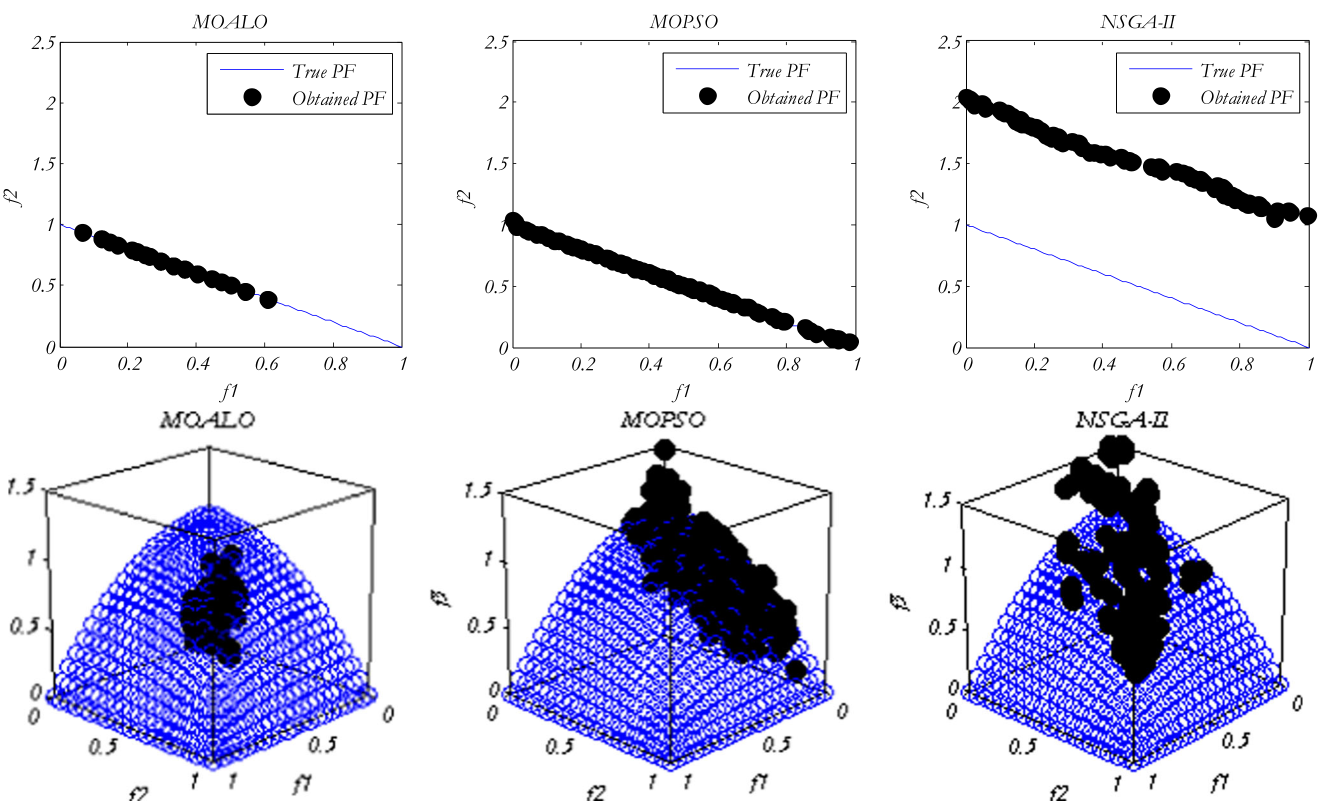

Fig. 3 Best Pareto optimal front obtained by the multi-objective algorithms on ZDT1 with linear front and ZDT2 with 3 objectives

Table 2 Results of the multi-objective algorithms on CONSTR, TNK, SRN, BNH, and OSY constrained test problem

Algorithm	GD		Metric of spread		Metric of spacing		IGD	
	Ave	Std.	Ave	Std.	Ave	Std.	Ave	Std.
CONSTR test problem								
MOALO	1.7021E-04	4.6424E-05	0.34585	1.0425E-02	0.0214	0.0027	1.4E-04	2.42E-05
MOPSO	4.5437E−03	6.8558E−04	0.94312	3.6719E−01	N/A	N/A	N/A	N/A
NSGA-II	5.1349E−03	2.4753E−04	0.54863	2.7171E−02	0.0437	0.0041	N/A	N/A
TNK test problem								
MOALO	7.9693E-04	5.4324E-05	0.64273	1.1525E-02	0.002	0.0001	6.2E-04	7.64E-05
MOPSO	5.0877E−03	4.5564E−04	0.79363	5.1029E−02	N/A	N/A	N/A	N/A
NSGA-II	4.0488E−03	4.3465E−04	0.82286	2.8678E−04	N/A	N/A	N/A	N/A
SRN test problem								
MOALO	6.8937E-05	3.4958E-06	0.3859	2.5242E-02	0.7030	0.102	0.3E-04	6.69E-06
MOPSO	2.7623E−03	2.0794E−04	0.6655	7.2196E−02	N/A	N/A	N/A	N/A
NSGA-II	3.7069E−03	5.1034E−04	0.3869	2.5115E−02	1.586	0.133	N/A	N/A
BNH test problem								
MOALO	2.0466E-04	5.6982E-05	0.3716	2.6356E-02	0.3357	0.024	2.4E-04	5.56E-05
MOPSO	N/A	N/A	N/A	N/A	0.6941	0.038	N/A	N/A
NSGA-II	N/A	N/A	N/A	N/A	0.7756	0.072	N/A	N/A
OSY test problem								
MOALO	3.12E-02	2.58E-02	0.3716	2.6356E-02	0.4959	0.076	2.9E-02	1.17E-02
MOPSO	9.68E−02	7.18e−02	N/A	N/A	0.522	0.095	N/A	N/A
NSGA-II	9.89E−01	9.78E−01	N/A	N/A	1.14	0.275	N/A	N/A

The shape of the best Pareto optimal front obtained by the three algorithms on ZDT1, ZDT2, and ZDT3 are illustrated in Fig. 2. Inspecting this figure, it may be observed that NSGA-II shows the poorest convergence despite its good coverage in ZDT1 and ZDT3. However, the MOPSO and MOALO both provide a very good convergence toward all the true Pareto optimal fronts. The most interesting pattern is that the Pareto optimal solutions obtained by MOPSO show higher coverage than MOALO on ZDT1 and ZDT2. However, the coverage of MOALO on ZDT3 is better than MOPSO. This shows that MOALO has the potential to outperform MOPSO in finding Pareto optimal front with separated regions.

Figure 3 shows the best Pareto optimal front obtained by the algorithms on the last two unconstrained benchmark functions, in which the shape of the fronts are linear and convex. It is interesting the results are consistent with those of the first three test functions where the NSGA-II

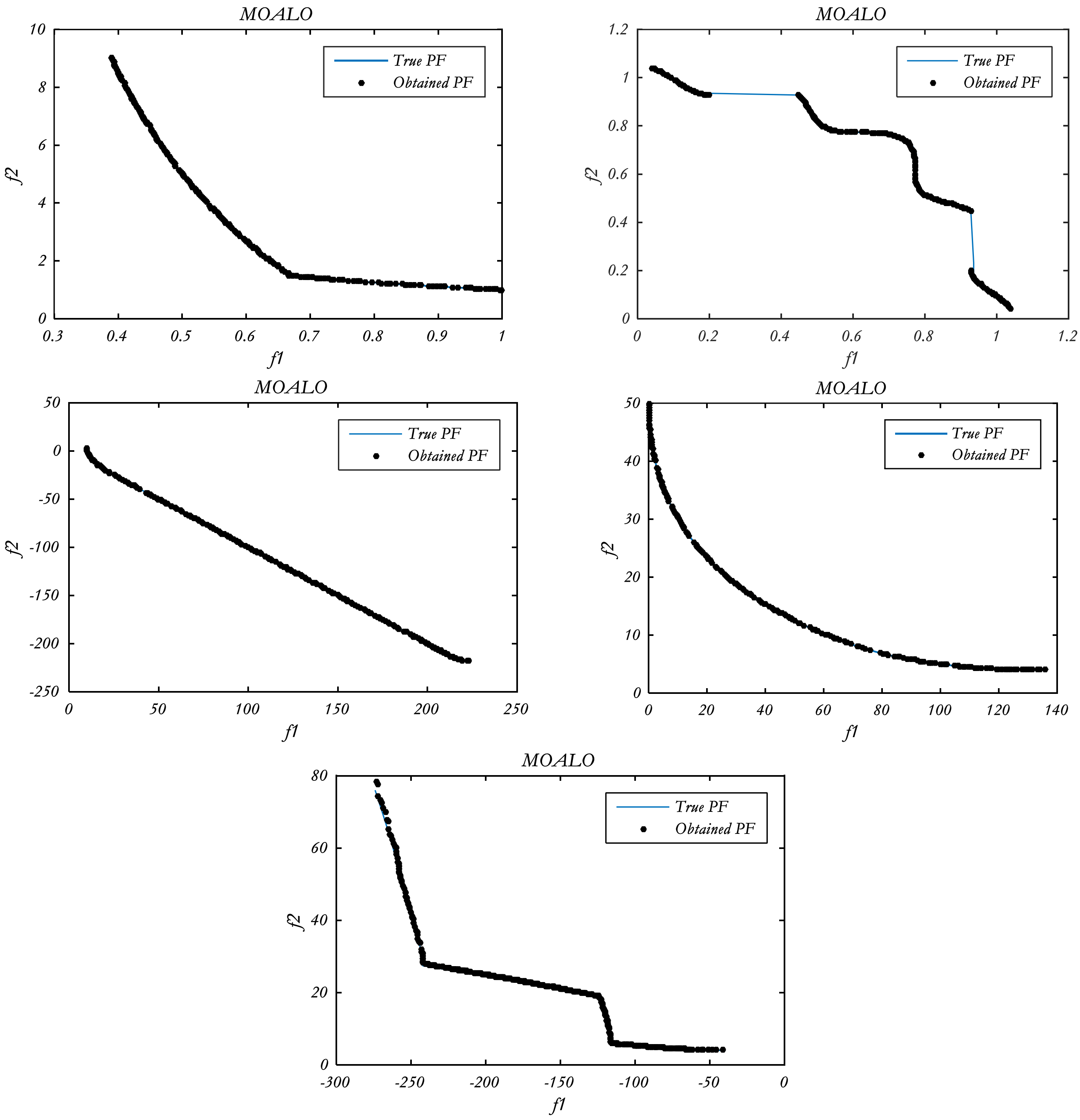

Fig. 4 Best Pareto optimal front obtained by MOALO for CONSTR, TNK, SRN, BNH, and OSY

algorithm shows the worst convergence. Comparing the results of MOPSO and MOALO on these test functions, it may be seen that the convergence of both algorithms are almost similar, but the coverage of the MOPSO is slightly superior.

4.2 Results on constrained test problems

The second set of test function includes constrained benchmark functions. Obviously, we need to equip MOALO with a constraint handling technique to be able to solve such problems. Finding a suitable constraints handling approach is out of the scope of this work, and we employ a death penalty function [44] to penalize search agents that violate any of the constraints at any level. For comparing algorithms, we have utilized four metrics in this experiment: GD, IGD, metric of spread, and metric of space. These performance indicators allow us to quantify and compare algorithms in terms of convergence and coverage. Note that the results of MOPSO and NSGA-II are taken from [45].

Table 2 shows that the MOALO outperforms the other two algorithms on the majority of the constrained test functions employed. The superior convergence can be inferred from the results of GD and IGD. The results collected by the GD performance metric clearly shows that the MOALO algorithm surpasses the MOPSO and NSGA-II algorithm. The best Pareto optimal fronts in Fig. 3 also support this claim since all the Pareto optimal solutions found by MOALO are located on the front.

The results for coverage performance metrics in Table 2 also prove that both MOPSO and MOALO show better results compared to the NSGA-II algorithm. However, they tend to be very competitive in comparison to each other. High coverage of the MOALO algorithm on the constrained test functions can be observed in Fig. 4. This figure illustrates that some of the constrained test functions have very

Table 3 Results of the constrained engineering design problems

Algorithm	GD		Metric of spread		Metric of spacing		IGD	
	Ave	Std.	Ave	Std.	Ave	Std.	Ave	Std.
Four-bar truss design problem								
MOALO	0.1264	0.0327	0.370	0.00251	1.1805	0.144	0.1062	1.52E-02
MOPSO	0.3741	0.0422	N/A	N/A	2.5303	0.227	N/A	N/A
NSGA-II	0.3601	0.0470	N/A	N/A	2.3635	0.255	N/A	N/A
Speed reducer design problem								
MOALO	1.1767	0.2327	0.839	0.1267	1.7706	2.769	0.8672	1.49E-01
MOPSO	0.98831	0.1789	N/A	N/A	16.685	2.696	N/A	N/A
NSGA-II	9.8437	7.0810	N/A	N/A	2.7654	3.534	N/A	N/A
Disk brake design problem								
MOALO	0.0011	0.00245	0.44958	0.05427	0.0421	0.0058	1.94E-2	.78E-3
MOPSO	0.0244	0.12314	0.46041	0.10961	N/A	N/A	N/A	N/A
NSGA-II	3.0771	0.10782	0.79717	0.06608	N/A	N/A	N/A	N/A
Welded beam design problem								
MOALO	0.00665	0.00742	0.19784	0.07962	0.0426	0.0077	1.52E-3	4.65E-3
MOPSO	0.04909	0.02821	0.22478	0.09280	N/A	N/A	N/A	N/A
NSGA-II	0.16875	0.08030	0.88987	0.11976	N/A	N/A	N/A	N/A
Cantilever beam design problem								
MOALO	2.3442E-4	1.6222E-5	0.76731	0.16853	0.00832	0.0029	1.89E-4	6.96E-5
MOPSO	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
NSGA-II	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Brushless DC Wheel Motor design problem								
MOALO	7.75E-3	5.261E-3	0.8211	0.07845	0.00826	0.0007	1.42E-3	1.25E-3
MOPSO	8.78E-2	4.22E-2	N/A	N/A	N/A	N/A	N/A	N/A
NSGA-II	8.51E-2	1.70E-2	N/A	N/A	N/A	N/A	N/A	N/A
Safety Isolating Transformer design problem								
MOALO	1.41720E-3	1.25426E-4	0.40761	0.06802	0.0558	0.0057	1.22E-3	9.87E-4
MOPSO	6.54731E-2	9.85432E-3	N/A	N/A	N/A	N/A	N/A	N/A
NSGA-II	2.65719E-2	7.67234E-3	N/A	N/A	N/A	N/A	N/A	N/A

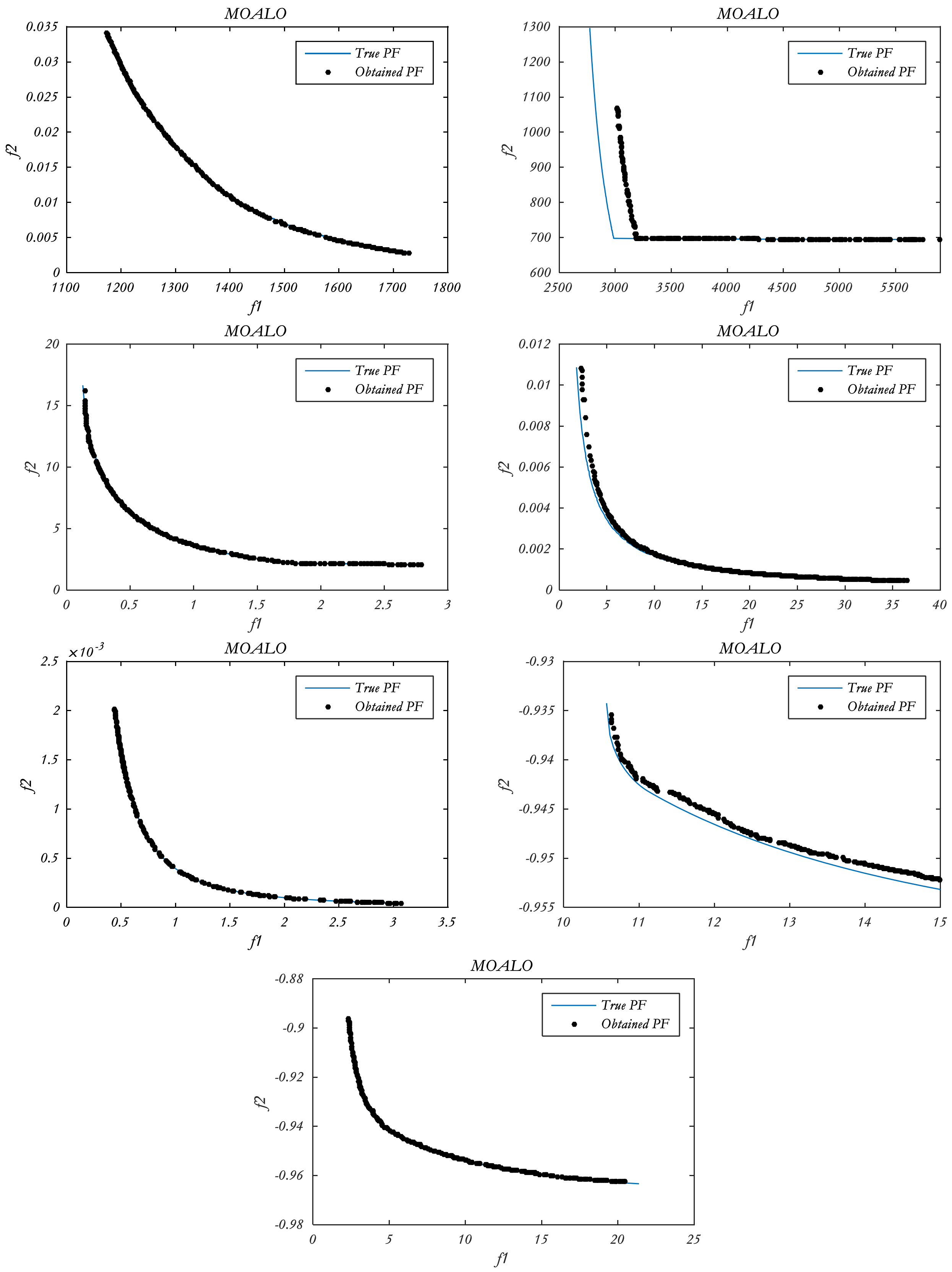

Fig. 5 Best Pareto optimal front obtained by the multi-objective algorithms on the engineering design multiobjective problems: 4-bar truss design, speed reduced design, disk brake design, welded beam deign, cantilever beam design, brushless dc wheel motor design, and safety isolating transformer design

different Pareto fronts compared to the unconstrained test functions utilized in the preceding sub-section, for instance CONSTR, BNH, and OSY. It may be seen that CONSTR has a concave front attached to a linear front. The results show that MOALO managed to approximate both parts successfully. However, the TNK test function has a wave-shaped front, and it was determined completely by the proposed MOALO. The OSY function is slightly similar to CONSTR but with multiple linear regions with different slopes. Again, the results show that these types of fronts are also achievable by the MOALO algorithm.

4.3 Results on constrained engineering design problems

The last set of test functions is the most challenging one and includes 7 real engineering design problems. The Appendix C shows that these problems have diverse characteristics and are all constrained. Therefore, they highly suit benchmarking the performance of the proposed MOALO algorithm. A similar set of performance metrics is employed to compare the results of algorithms quantitativelly, and the results are presented in Table 3 and Fig. 5.

The results in Table 3 are consistent with those in the preceding tables, in which the MOALO algorithm mostly shows better convergence and coverage. Due to the difficulty of these real engineering design problems, the results highly support the superiority of the MOALO algorithm and its applicability. The best Pareto optimal fronts in Fig. 5, however, present different behavior from other test suites. It may be seen in this figure that the convergence of the MOALO algorithm is not 100% close to the true Pareto front in the speed reducer, welded beam, and brushless DC wheel motor design problems. This is due to the multi-modality (multiple local fronts) of the search space and existence of many constraints. In spite of this fact, the convergence is reasonable and coverage is extremely high and almost uniform.

4.4 Discussion

The qualitative and quantitative results showed that the MOALO algorithm benefits from high convergence and coverage. High convergence of MOALO is inherited from the ALO algorithm. The main mechanisms that guarantee convergence in ALO and MOALO are shrinking boundaries of random walks in the movement of ants and elitism. These two mechanisms emphasize exploitation and convergence proportional to the number of iterations. Since we select two solutions from the archive in every iterations and require an ant to move around both of them in MOALO, degraded convergence might be a concern. However, the results prove that the MOALO algorithm does not suffer from slow convergence.

It was also observed and proved that high coverage is another advantage of the MOALO algorithm. The superior coverage originates from the antlion selection and archive maintenance methods. Anlions in the regions of the search space with a less-populated neighbourhood have a lower chance of being chosen from the archive. This requires ants to explore and explore the un-covered or less-covered areas of the search space and front. In addition, the archive maintenance mechanism is regularly triggered when the archive becomes full. Since solutions in the most populated regions have a higher chance to be thrown away, this mechanism again emphasizes improving the coverage of the Pareto optimal front obtained during the optimization process.

5 Conclusion

This paper proposed the multi-objective version of the recently proposed ALO algorithm called MOALO. With maintaining the main search mechanism of ALO, MOALO was designed with equipping ALO with an archive and antlion selection mechanism based on Pareto optimal dominance. The algorithm was tested on 17 case studies including 5 unconstrained functions, 5 constrained functions, 7 and engineering design problems. The quantitative results were collected using four performance indicators: GD, IGD, metric of spread, and metric of spacing. Also, qualitative results were reported as the best Pareto optimal front found in 10 runs. For results verification, the proposed algorithm was compared to the well-regarded algorithms in the field: NSGA-II and MOPSO. The results showed that the MOALO is able to outperform NSGA-II on the majority of the test functions and provide very competitive resulted compared to the MOPSO algorithm. It was observed that MOALO benefits from high convergence and coverage as well. The test functions employed are of different type and have diverse Pareto optimal fronts. The results showed that MOALO can find Pareto optimal front of any shape. Finally, the results of constrained engineering design problems testified that MOALO is capable of solving challenging problems with many constraints and unknown search spaces. Therefore, we conclude that the proposed algorithm has merits among the current multi-objective algorithms and offer it as an alternative for solving multi-objective optimization problems. Another conclusion is made based on the NFL theorem: MOALO outperforms other algorithms on the test functions, so it has the potential to provide superior results on other problems as well. Note that the source codes of the MOALO and ALO algorithms are publicly available at http://alimirjalili.com/ALO.html.

For future works, it is recommended to apply MOALO to other engineering design problems. Also, it is worth to investigate and find the best constrained handling technique for this algorithm.

Appendix A: Unconstrained multi-objective test problems utilised in this work

ZDT1:

Minimize: $f_1(x) = x_1$ (A.1)

Minimize: $f_2(x) = g(x) \times h(f_1(x), g(x))$ (A.2)

Where: $G(x) = 1 + \frac{9}{N-1} \sum_{i=2}^{N} x_i$ (A.3)

$h(f_1(x), g(x)) = 1 - \sqrt{\frac{f_1(x)}{g(x)}}$ (A.4)

$0 \le x_i \le 1, 1 \le i \le 30$

ZDT2:

Minimize: $f_1(x) = x_1$ (A.5)

Minimize: $f_2(x) = g(x) \times h(f_1(x), g(x))$ (A.6)

Where: $G(x) = 1 + \frac{9}{N-1} \sum_{i=2}^{N} x_i$ (A.7)

$h(f_1(x), g(x)) = 1 - \left(\frac{f_1(x)}{g(x)}\right)^2$ (A.8)

$0 \le x_i \le 1, 1 \le i \le 30$

ZDT3:

Minimize: $f_1(x) = x_1$ (A.9)

Minimize: $f_2(x) = g(x) \times h(f_1(x), g(x))$ (A.10)

Where: $G(x) = 1 + \frac{9}{29} \sum_{i=2}^{N} x_i$ (A.11)

$h(f_1(x), g(x)) = 1 - \sqrt{\frac{f_1(x)}{g(x)}} - \left(\frac{f_1(x)}{g(x)}\right) \sin(10\pi f_1(x))$ (A.12)

$0 \le x_i \le 1, 1 \le i \le 30$

ZDT1 with linear PF:

Minimize: $f_1(x) = x_1$ (A.13)

Minimize: $f_2(x) = g(x) \times h(f_1(x), g(x))$ (A.14)

Where: $G(x) = 1 + \frac{9}{N-1} \sum_{i=2}^{N} x_i$ (A.15)

$h(f_1(x), g(x)) = 1 - \frac{f_1(x)}{g(x)}$ (A.16)

$0 \le x_i \le 1, 1 \le i \le 30$

ZDT2 with three objectives:

Minimize: $f_1(x) = x_1$ (A.17)

Minimize: $f_2(x) = x_2$ (A.18)

Minimize: $f_3(x) = g(x) \times h(f_1(x), g(x)) \times h(f_2(x), g(x))$ (A.19)

Where: $G(x) = 1 + \frac{9}{N-1} \sum_{i=2}^{N} x_i$ (A.20)

$h(f_1(x), g(x)) = 1 - \left(\frac{f_1(x)}{g(x)}\right)^2$ (A.21)

$0 \le x_i \le 1, 1 \le i \le 30$

Appendix B: Constrained multi-objective test problems utilised in this work

CONSTR:

This problem has a convex Pareto front, and there are two constraints and two design variables.

Minimize: $f_1(x) = x_1$ (B.1)

Minimize: $f_2(x) = (1 + x_2)/(x_1)$ (B.2)

Where: $g_1(x) = 6 - (x_2 + 9x_1), g_2(x) = 1 + x_2 - 9x_1$

$0.1 \le x_1 \le 1, 0 \le x_2 \le 5$

TNK:

The second problem has a discontinuous Pareto optima front, and there are two constraints and two design variables.

Minimize: $f_1(x) = x_1$ (B.3)

Minimize: $f_2(x) = x_2$ (B.4)

Where: $g_1(x) = -x_1^2 - x_2^2 + 1 + 0.1 Cos\left(16 arctan\left(\frac{x_1}{x_2}\right)\right)$

$g_2(x) = 0.5 - (x_1 - 0.5)^2 - (x_2 - 0.5)^2$

$0.1 \le x_1 \le \pi, 0 \le x_2 \le \pi$

SRN:

The third problem has a continuous Pareto optimal front proposed by Srinivas and Deb [46].

Minimize: $f_1(x) = 2 + (x_1 - 2)^2 + (x_2 - 1)^2$ (B.5)

Minimize: $f_2(x) = 9x_1 - (x_2 - 1)^2$ (B.6)

Where: $g_1(x) = x_1^2 + x_2^2 - 255$

$g_2(x) = x_1 - 3x_2 + 10$

$-20 \le x_1 \le 20, -20 \le x_2 \le 20$

BNH:

This problem was first proposed by Binh and Korn [47]:

$$\text{Minimize:} \quad f_1(x) = 4x_1^2 + 4x_2^2 \qquad \text{(B.7)}$$

$$\text{Minimize:} \quad f_2(x) = (x_1 - 5)^2 + (x_2 - 5)^2 \qquad \text{(B.8)}$$

$$\text{Where:} \quad g_1(x) = (x_1 - 5)^2 + x_2^2 - 25$$

$$g_2(x) = 7.7 - (x_1 - 8)^2 - (x_2 + 3)^2$$

$$0 \le x_1 \le 5, 0 \le x_2 \le 3$$

OSY:

The OSY test problem has five separated regions proposed by Osyczka and Kundu [48]. Also, there are six constraints and six design variables.

$$\text{Minimize:} \quad f_1(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 + x_6^2 \quad \text{(B.9)}$$

$$\text{Minimize:} \quad f_2(x) = [25(x_1 - 2)^2 + (x_2 - 1)^2 + (x_3 - 1) + (x_4 - 4)^2 + (x_5 - 1)^2] \qquad \text{(B.10)}$$

$$\text{Where:} \quad g_1(x) = 2 - x_1 - x_2$$

$$g_2(x) = -6 + x_1 + x_2$$

$$g_3(x) = -2 - x_1 + x_2$$

$$g_4(x) = -2 + x_1 - 3x_2$$

$$g_5(x) = -4 + x_4 + (x_3 - 3)^2$$

$$g_6(x) = 4 - x_6 - (x_5 - 3)^2$$

$$0 \le x_1 \le 10, 0 \le x_2 \le 10, 1 \le x_3 \le 5, 0 \le x_4 \le 6, 1 \le x_5 \le 5, 0 \le x_6 \le 10$$

Appendix C: Constrained multi-objective engineering problems used in this work

Four-bar truss design problem

The 4-bar truss design problem is a well-known problem in the structural optimization field [49], in which structural volume (f_1) and displacement (f_2) of a 4-bar truss should be minimized. As can be seen in the following equations, there are four design variables (x_1-x_4) related to cross sectional area of members 1, 2, 3, and 4.

$$\text{Minimize:} \quad f_1(x) = 200 * (2 * x(1) + sqrt(2 * x(2)) + sqrt(x(3)) + x(4)) \qquad \text{(C.1)}$$

$$\text{Minimize:} \quad f_2(x) = 0.01 * (\left(\frac{2}{x(1)}\right) + \left(\frac{2 * sqrt(2)}{x(2)}\right) \ldots - ((2 * sqrt(2))/x(3)) + (2/x(1))) \qquad \text{(C.2)}$$

$$1 \le x_1 \le 3, 1.4142 \le x_2 \le 3, 1.4142 \le x_3 \le 3, 1 \le x_4 \le 3$$

Speed reducer design problem

The speed reducer design problem is a well-known problem in the area of mechanical engineering [49, 50], in which the weight (f_1) and stress (f_2) of a speed reducer should be minimized. There are seven design variables: gear face width (x_1), teeth module (x_2), number of teeth of pinion (x_3 integer variable), distance between bearings 1 (x_4), distance between bearings 2 (x_5), diameter of shaft 1 (x_6), and diameter of shaft 2 (x_7) as well as eleven constraints.

$$\text{Minimize:} \quad f_1(x) = 0.7854 * x(1) * x(2)^2 * (3.3333 * x(3)^2 + 14.9334 * x(3)) \ldots - 43.0934) - 1.508 * x(1) * (x(6)^2 + x(7)^2 + 7.4777 * (x(6)^3 + x(7)^3) \ldots + 0.7854 * (x(4) * x(6)^2 + x(5) * x(7)^2) \qquad \text{(C.3)}$$

$$\text{Minimize:} \quad f_2(x) = ((sqrt(((745 * x(4))/x(2) * x(3)))^2 + 19.9e6))/(0.1 * x(6)^3)) \qquad \text{(C.4)}$$

$$\text{Where:} \quad g_1(x) = 27/(x(1) * x(2)^2 * x(3)) - 1$$

$$g_2(x) = 397.5/(x(1) * x(2)^2 * x(3)^2) - 1$$

$$g_3(x) = (1.93 * (x(4)^3)/(x(2) * x(3) * x(6)^4) - 1$$

$$g_4(x) = (1.93 * (x(5)^3)/(x(2) * x(3) * x(7)^4) - 1$$

$$g_5(x) = ((sqrt(((745 * x(4))/(x(2) * x(3)))^2 + 16.9e6))/(110 * x(6)^3)) - 1$$

$$g_6(x) = ((sqrt(((745 * x(5))/(x(2) * x(3)))^2 + 157.5e6))/(85 * x(7)^3)) - 1$$

$$g_7(x) = ((x(2) * x(3))/40) - 1$$

$$g_8(x) = (5 * x(2)/x(1)) - 1$$

$$g_9(x) = (x(1)/12 * x(2)) - 1$$

$$g_{10}(x) = ((1.5 * x(6) + 1.9)/x(4)) - 1$$

$$g_{11}(x) = ((1.1 * x(7) + 1.9)/x(5)) - 1$$

$$2.6 \le x_1 \le 3.6, 0.7 \le x_2 \le 0.8, 17 \le x_3 \le 28, 7.3 \le x_4 \le 8.3, 7.3 \le x_5 \le 8.3, 2.9 \le x_6 \le 3.9$$

$$5 \le x_7 \le 5.5$$

Disk brake design problem

The disk brake design problem has mixed constraints and was proposed by Ray and Liew [51]. The objectives to be minimized are: stopping time (f_1) and mass of a brake (f_2) of a disk brake. As can be seen in following equations, there are four design variables: the inner radius of the disk (x_1), the outer radius of the disk (x_2), the engaging force

(x_3), and the number of friction surfaces (x_4) as well as five constraints.

$$\text{Minimize:}\quad f_1(x) = 4.9*(10^{(-5)})*(x(2)^2-x(1)^2)*(x(4)-1) \tag{C.5}$$

$$\text{Minimize:}\quad f_2(x) = (9.82*(10^{(6)}))*(x(2)^2 - x(1)^2))/((x(2)^3 - x(1)^3)*x(4)*x(3)) \tag{C.6}$$

Where: $g_1(x) = 20 + x(1) - x(2)$

$g_2(x) = 2.5*(x(4)+1) - 30$

$g_3(x) = (x(3))/(3.14*(x(2)^2 - x(1)^2)^2) - 0.4$

$g_4(x) = (2.22*10^{(-3)}*x(3)*(x(2)^3 - x(1)^3))/((x(2)^2 - x(1)^2)^2) - 1$

$g_5(x) = 900 - (2.66*10^{(-2)}*x(3)*x(4)*(x(2)^3 - x(1)^3))/((x(2)^2 - x(1)^2))$

$55 \le x_1 \le 80,\ 75 \le x_2 \le 110,\ 1000 \le x_3 \le 3000,\ 2 \le x_4 \le 20$

Welded beam design problem

The welded beam design problem has four constraints first proposed by Ray and Liew [51]. The fabrication cost (f_1) and deflection of the beam (f_2) of a welded beam should be minimized in this problem. There are four design variables: the thickness of the weld (x_1), the length of the clamped bar (x_2), the height of the bar (x_3) and the thickness of the bar (x_4).

$$\text{Minimize:}\quad f_1(x) = 1.10471*x(1)^2*x(2) + 0.04811*x(3)*x(4)*(14.0+x(2)) \tag{C.7}$$

$$\text{Minimize:}\quad f_2(x) = 65856000/(30*10^6*x(4)*x(3)^3) \tag{C.8}$$

Where: $g_1(x) = \tau - 13600$

$g_2(x) = \sigma - 30000$

$g_3(x) = x(1) - x(4)$

$g_4 = 6000 - P$

$0.125 \le x_1 \le 5,\ 0.1 \le x_2 \le 10,\ 0.1 \le x_3 \le 10,\ 0.125 \le x_4 \le 5$

Where

$$q = 6000*\left(14 + \frac{x(2)}{2}\right);\ D = sqrt\left(\frac{x(2)^2}{4} + \frac{(x(1)+x(3))^2}{4}\right)$$

$$J = 2*\left(x(1)*x(2)*sqrt(2)*\left(\frac{x(2)^2}{12} + \frac{(x(1)+x(3))^2}{4}\right)\right)$$

$$\alpha = \frac{6000}{sqrt(2)*x(1)*x(2)}$$

$$\beta = Q*\frac{D}{J}$$

$$\tau = sqrt\left(\alpha^2 + 2*\alpha*\beta*\frac{x(2)}{2*D} + \beta^2\right)$$

$$\sigma = \frac{504000}{x(4)*x(3)^2}$$

$$tmpf = 4.013*\frac{30*10^6}{196}$$

$$P = tmpf*sqrt\left(x(3)^2*\frac{x(4)^6}{36}\right)*\left(1 - x(3)*\frac{sqrt\left(\frac{30}{48}\right)}{28}\right)$$

Cantilever beam design problem

The cantilever beam design problem is another well-known problem in the field of concrete engineering [8], in which weight (f_1) and end deflection (f_2) of a cantilever beam should be minimized. There are two design variables: diameter (x_1) and length (x_2).

$$\text{Minimize:}\quad f_1(x) = 0.25*\rho*\pi*x(2)*x(1)^2 \tag{C.9}$$

$$\text{Minimize:}\quad f_2(x) = (64*P*x(2)^3)/(3*E*\pi*x(1)^4) \tag{C.10}$$

Where: $g_1(x) = -Sy + (32*P*x(2))/(\pi*x(1)^3)$

$g_2(x) = -\delta_{max} + (64*P*x(2)^3)/(3*E*\pi*x(1)^4)$

$0.01 \le x_1 \le 0.05,\ 0.20 \le x_2 \le 1$

Where

$P=1,\ E=207000000,\ Sy=300000,\ \delta_{max}=0.005;\ \rho=7800$

Brushless DC wheel motor with two objectives

Brushless DC wheel motor design problem is a constrained multi-objective problem in the area of electrical engineering [52]. The objectives are in conflict, and there are five design variables: stator diameter (Ds), magnetic induction in the air gap (Be), current density in the conductors (δ), magnetic induction in the teeth (Bd) and magnetic induction in the stator back iron (Bcs).

$$\text{Maximize:}\quad f_1(x) = \text{Max}\ \eta \tag{C.11}$$

$$\text{Minimize:}\quad f_2(x) = Min\ \ M_{tot} \tag{C.12}$$

$D_{ext} \le 340mm$

Where: $G(x) = D_{\text{int}} \ge 76mm,\ I_{\max} \ge 125A$

$T_a \le 120°,\ discr \ge 0$

$150mm \le D_s \le 330mm,\ 0.5T \le B_e \le 0.76T$

$2A/mm^2 \le \varsigma \le 5A/mm^2,\ 0.9T \le B_d \le 1.8T$

$0.6T \le B_{cs} \le 1.6T$

Safety isolating transformer design with two objectives

$$\text{Maximize:} \quad f_1(x) = \text{Max} \ \ \eta \tag{C.13}$$

$$\text{Maximize:} \quad f_2(x) = Min \ \ M_{tot} \tag{C.14}$$

$$T_{cond} \leq 120°C, T_{iron} \leq 100°C$$

$$\text{Where:} \quad G(x) = \frac{\Delta V_2}{V_{20}} \leq 0.1, \frac{I_{10}}{I_1} \leq 0.1$$

$$f_2 \leq 1, f_1 \leq 1$$

$$residue < 10^{-6}$$

$$3mm \leq a \leq 30mm, 14mm \leq b \leq 95mm$$

$$6mm \leq c \leq 40mm, 10mm \leq d \leq 80mm$$

$$200 \leq n_1 \leq 1200, 0.15mm^2 \leq S_1 \leq 19mm^2$$

$$0.15mm^2 \leq S_2 \leq 19mm^2$$

References

1. Kelley CT (1999) Detection and remediation of stagnation in the Nelder–Mead algorithm using a sufficient decrease condition. SIAM J Optim 10:43–55
2. Vogl TP, Mangis J, Rigler A, Zink W, Alkon D (1988) Accelerating the convergence of the back-propagation method. Biol Cybern 59:257–263
3. Marler RT, Arora JS (2004) Survey of multi-objective optimization methods for engineering. Struct Multidiscip Optim 26:369–395
4. Coello CAC (2002) Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. Comput Methods Appl Mech Eng 191:1245–1287
5. Beyer H.-G., Sendhoff B (2007) Robust optimization–a comprehensive survey. Comput Methods Appl Mech Eng 196:3190–3218
6. Knowles JD, Watson RA, Corne DW (2001) Reducing local optima in single-objective problems by multi-objectivization. In: Evolutionary multi-criterion optimization, pp 269–283
7. Deb K, Goldberg DE (1993) Analyzing deception in trap functions. Found Genet Algoritm 2:93–108
8. Deb K (2001) Multi-objective optimization using evolutionary algorithms, vol 16. Wiley
9. Coello CAC, Lamont GB, Van Veldhuisen DA (2007) Evolutionary algorithms for solving multi-objective problems. Springer
10. Branke J, Deb K, Dierolf H, Osswald M (2004) Finding knees in multi-objective optimization. In: Parallel problem solving from nature-PPSN VIII, pp 722–731
11. Das I, Dennis JE (1998) Normal-boundary intersection: a new method for generating the Pareto surface in nonlinear multicriteria optimization problems. SIAM J Optim 8:631–657
12. Kim IY, De Weck O (2005) Adaptive weighted-sum method for bi-objective optimization: Pareto front generation. Struct Multidiscip Optim 29:149–158
13. Messac A, Mattson CA (2002) Generating well-distributed sets of Pareto points for engineering design using physical programming. Optim Eng 3:431–450
14. Deb K, Agrawal S, Pratap A, Meyarivan T (2000) A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. In: Parallel problem solving from nature PPSN VI, pp 849–858
15. Deb K, Goel T (2001) Controlled elitist non-dominated sorting genetic algorithms for better convergence. In: Evolutionary multi-criterion optimization, pp 67–81
16. Deb K, Pratap A, Agarwal S, Meyarivan T (2002) A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans Evol Comput 6:182–197
17. Coello CAC, Lechuga MS (2002) MOPSO: a proposal for multiple objective particle swarm optimization. In: Proceedings of the 2002 congress on evolutionary computation, 2002. CEC'02, pp 1051–1056
18. Coello CAC, Pulido GT, Lechuga MS (2004) Handling multiple objectives with particle swarm optimization. IEEE Trans Evol Comput 8:256–279
19. Akbari R, Hedayatzadeh R, Ziarati K, Hassanizadeh B (2012) A multi-objective artificial bee colony algorithm. Swarm Evol Comput 2:39–52
20. Yang X-S (2011) Bat algorithm for multi-objective optimisation. Int J Bio-Inspired Comput 3:267–274
21. Mirjalili S, Saremi S, Mirjalili SM, Coelho L. d. S. (2016) Multi-objective grey wolf optimizer: a novel algorithm for multi-criterion optimization. Expert Syst Appl 47:106–119
22. Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1:67–82
23. Ngatchou P, Zarei A, El-Sharkawi M (2005) Pareto multi objective optimization. In: Proceedings of the 13th international conference on intelligent systems application to power systems, 2005, pp 84–91
24. Pareto V (1964) Cours d'economie politique: Librairie Droz
25. Edgeworth FY (1881) Mathematical physics. P. Keagan, London
26. Zitzler E (1999) Evolutionary algorithms for multiobjective optimization: methods and applications, vol 63. Citeseer
27. Zitzler E, Thiele L (1999) Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. IEEE Trans Evol Comput 3:257–271
28. Srinivas N, Deb K (1994) Muiltiobjective optimization using nondominated sorting in genetic algorithms. Evol Comput 2:221–248
29. Zhang Q, Li H (2007) MOEA/D: a multiobjective evolutionary algorithm based on decomposition. IEEE Trans Evol Comput 11:712–731
30. Knowles JD, Corne DW (2000) Approximating the nondominated front using the Pareto archived evolution strategy. Evol Comput 8:149–172
31. Abbass HA, Sarker R, Newton C (2001) PDE: a Pareto-frontier differential evolution approach for multi-objective optimization problems. In: Proceedings of the 2001 congress on evolutionary computation, 2001, pp 971–978
32. Branke J, Kaußler T, Schmeck H (2001) Guidance in evolutionary multi-objective optimization. Adv Eng Softw 32:499–507
33. Eberhart RC, Kennedy J (1995) A new optimizer using particle swarm theory. In: Proceedings of the sixth international symposium on micro machine and human science, pp 39–43
34. Kennedy J (2011) Particle swarm optimization. In: Encyclopedia of machine learning. Springer, pp 760–766
35. Horn J, Nafpliotis N, Goldberg DE (1994) A niched Pareto genetic algorithm for multiobjective optimization. In: Proceedings of the 1st IEEE conference on evolutionary computation, 1994. IEEE world congress on computational intelligence, pp 82–87
36. Mahfoud SW (1995) Niching methods for genetic algorithms. Urbana 51:62–94
37. Horn JR, Nafpliotis N, Goldberg DE (1993) Multiobjective optimization using the niched pareto genetic algorithm. IlliGAL report, pp 61801–2296
38. Mirjalili S (2015) The ant lion optimizer. Adv Eng Softw 83:80–98
39. Van Veldhuizen DA, Lamont GB (1998) Multiobjective evolutionary algorithm research: a history and analysis. Citeseer

40. Sierra MR, Coello CAC (2005) Improving PSO-based multi-objective optimization using crowding, mutation and ∈-dominance. In: Evolutionary multi-criterion optimization, pp 505–519
41. Schott JR (1995) Fault tolerant design using single and multicriteria genetic algorithm optimization, DTIC Document
42. Zitzler E, Deb K, Thiele L (2000) Comparison of multiobjective evolutionary algorithms: Empirical results. Evolutionary computation 8(2):173–195
43. Mirjalili S (2016) Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. Neural Comput & Applic 27:1053–1073
44. Coello CAC (2000) Use of a self-adaptive penalty approach for engineering optimization problems. Comput Ind 41:113–127
45. Sadollah A, Eskandar H, Kim JH (2015) Water cycle algorithm for solving constrained multi-objective optimization problems. Appl Soft Comput 27:279–298
46. Srinivasan N, Deb K (1994) Multi-objective function optimisation using non-dominated sorting genetic algorithm. Evol Comput 2:221–248
47. Binh TT, Korn U (1997) MOBES: A multiobjective evolution strategy for constrained optimization problems. In: The 3rd international conference on genetic algorithms (Mendel 97), p 7
48. Osyczka A, Kundu S (1995) A new method to solve generalized multicriteria optimization problems using the simple genetic algorithm. Struct Optim 10:94–99
49. Coello CC, Pulido GT (2005) Multiobjective structural optimization using a microgenetic algorithm. Struct Multidiscip Optim 30:388–403
50. Kurpati A, Azarm S, Wu J (2002) Constraint handling improvements for multiobjective genetic algorithms. Struct Multidiscip Optim 23:204–213
51. Ray T, Liew KM (2002) A swarm metaphor for multiobjective design optimization. Eng Optim 34:141–153
52. Moussouni F, Brisset S, Brochet P (2007) Some results on the design of brushless DC wheel motor using SQP and GA. Int J Appl Electromagn Mech 26:233–241