A Comprehensive Survey of Metaheuristic Algorithms Applying in Mechanical Design Optimization Problems

Hsu-Hsing Chen and Feng-Cheng Yang

October 31, 2022

A Comprehensive Survey of Metaheuristic Algorithms Applying in Mechanical Design Optimization Problems

Hsu-Hsing Chen
Institute of Industrial Engineering
National Taiwan University
Taipei, Taiwan
hsuhsingchen@ntu.edu.tw

Feng-Cheng Yang
Institute of Industrial Engineering
National Taiwan University
Taipei, Taiwan
iefcyang@ntu.edu.tw

***Abstract*—Metaheuristic algorithms have received great popularity since the first metaheuristic algorithm—Genetic Algorithm—was proposed in 1975. However, conventionally the performances of metaheuristic algorithms are evaluated by a series of unconstrained mathematical functions. Unfortunately, in real-world problems, for example, mechanical design optimization problems, the landscapes of the search area are far different from mathematical ones, and most importantly, the constraints are almost always imposed. Therefore, in this work, 19 metaheuristic algorithms have been implemented to solve three mechanical design optimization problems and evaluate their performance. Results show that the performances of metaheuristic algorithms could be determined by the initial convergence speed.**

***Index Terms*—Metaheuristic algorithm, Evolutionary algorithm, optimization, mechanical design**

I. Introduction

Design problems have always been of great concern in mechanical engineering. However, delivering a feasible design is not the only interest. Instead, an engineer may want to find the best design, and the process of finding this best design is called optimization [1]. Nowadays, some analytic methods are widely used to help engineering design. However, these methods are usually restricted to problems having exact solutions. For example, numerical solutions for the two-position double-rocker design [2]; if there's no exact solution, some proposed guidelines may be considered, but the result is often far from the optimum—for example, the Ziegler-Nichols method for PID controller tuning [3]. Therefore, more advanced and versatile methods should be employed. In this project, the approach is called the **metaheuristic algorithm**.

A. Metaheuristic algorithms

In optimization problems, the goal is to find the best solution among all feasible solutions [4]. However, the optimization problem one might face is often non-linear and with many local optima, where gradient-based methods would often fail to find the best solution. To solve this problem, scientists have turned to seek better solutions from animals or natural phenomena, which is so-called metaheuristic algorithms.

In literature, dozens of metaheuristic algorithms have been proposed. Some of the popular algorithms are Genetic Algorithm (GA)[5], Particle Swarm Optimization (PSO)[6], Simulated Annealing (SA)[7], Differential Evolution (DE)[8], and Imperialist Competitive Algorithm (ICA)[9].

However, comprehensive surveys on the performance of metaheuristic algorithms in real-life optimization problems, especially in mechanical design optimization problems, are still lacking. Even though, as suggested by the no free lunch theorem [10], the performance of any algorithm is equivalent when it is averaged out across all optimization problems. But when we focus on only a particular optimization problem, an algorithm may significantly outperform others. That is why selecting a promising algorithm is vital for a specific problem.

Unfortunately, the suggestion of algorithm selection is hardly found in the literature for mechanical design optimization problems. Most of the papers either merely compare the performance of a few types of algorithms on a single mechanical design optimization problem or apply only one particular algorithm to solving several problems. No thorough comparison had proposed.

Therefore, we investigated **19 different metaheuristic algorithms** and compared their performances in **three classic mechanical engineering design problems**. We hope this research results will help engineers select robust and efficient algorithms for their design optimization problems.

II. Related Works

Wu & Chow [11] reviewed four mechanical design optimization problems [12] to test the performance of GA. Using binary encoding, no specially designed crossover or mutation was needed to deal with a mixed (discrete and continuous) optimization problem.

Coello [13] provided a comprehensive review of constrained-handling techniques for evolutionary algorithms (EAs). In his work, solving techniques were categorized by five approaches: 1) penalty functions, 2) dedicated representations and operators, 3) repair algorithms, 4) separation of objectives and constraints, and 5) hybrid methods. By evaluating two mechanical design optimization problems, the results suggested no significant difference between these techniques. A more complicated constraint-handling technique would only improve the result slightly. Despite delivering comparisons of a wide variety of constraint-handling techniques, his conclusion suggested that when facing an unknown problem, the penalty-based approaches are always the first choice due to their simplicity and efficiency. Once the problem is identified, other techniques should be explored and applied to cope with the characteristics of the problem. For example, for combinatorial optimization problems, the approach of repair algorithms should be promising, or when dealing with linear constraints, the special representation approach should be convenient.

Acharyya & Mandal [14] gave a comparative report on applying three evolutionary algorithms (EAs), Genetic Algorithm (GA), Particle Swarm Optimization (PSO), and Differential Evolution (DE), to four-bar linkage path generating problems. By setting some prescribed points and corresponding angles of the input member, the desired path is generated, and the point of interest on the linkage can traverse the path approximately. Once the motion equations are formulated, the position of the point can be calculated from the given angle of the input. Therefore, the deviation from the generated path of the point can be calculated. The result suggested that DE outperformed the other two EAs with quite a margin in the solution quality and convergence speed.

III. Methodology

In this work, a python library was implemented[1] to perform the comparison of the 19 metaheuristic algorithms (brief introduction to each algorithm is listed in the subsequent section) on solving three mechanical design optimization problems–**Pressure Vessel Design**, **Spring Design**, and **Gear Train Design**. In solving a problem, an initial population was established and used in all algorithms, and each algorithm was independently executed for 30 runs to obtain the best objective value, the objective average, and the standard deviation among the 30 runs.

TABLE I
Full list of the 19 Metaheursitic Algorithms

Ant Lion Optimizer (ALO)	Harmony Search (HS)
Bat Algorithm (BA)	Memetic Algorithm (MA)
Cuckoo Search (CS)	Particle Swarm Optimization (PSO)
Differential Evolution (DE)	Simulated Annealing (SA)
Firefly Algorithm (FA)	Sine Cosine Algorithm (SCA)
Genetic Algorithm (GA)	Grey Wolf Optimizer (GWO)
Artificial Bee Colony Algorithm (ABC)	
Flower Pollination Algorithm (FPA)	
Gravitational Search Algorithm (GSA)	
Imperialist Competitive Algorithm (ICA)	
Shuffled Frog-Leaping Algorithm (SFLA)	
Teaching-Learning-based Optimization (TLBO)	
Whale Optimization Algorithm (WOA)	

[1]https://github.com/markmarkchen/Metaheuristic-Algorithms

A. Brief Introduction to the 19 Metaheuristic Algorithms

The algorithms presented below are organized by their similarities.

*1) **Differential evolution (DE):*** First introduced by Storn & Price [8], Differential evolution (DE) has enjoyed great success in the optimization field due to its simplicity and robustness. Even though several variants of DE have later been proposed, the underlying design paradigm remains the same: improve each individual by comparing with a mutant solution and accept the fitter one. In each variant of DE, the method of producing a mutant is slightly different, but all follow the following steps.

(i) Select an individual as the base vector.
(ii) Select a pair(s) of individuals and calculate their differences.
(iii) Conduct a weighted sum over the result from the step (ii) and add it to the base vector to produce a temporary mutant. (The weights are parameters and are sometimes called the amplification factors.)
(iv) Apply a crossover between the temporary mutant and the to-be-improved individual to produce the mutant.

*2) **Ant Lion Optimizer (ALO):*** In ALO [15], the whole population is formed by antlions, and the goal is to improve each antlion. To achieve that, in each iteration, ants (being the preys) are sent out to perform random walks in the vicinity of the assigned antlion. Once an ant finds a better solution, the antlion would move to that place (improvement) to mimic the hunting behavior. In addition, ALO also implements the transition from exploration to exploitation by shrinking the search radius of random walks during the process.

*3) **Artificial Bee Colony (ABC):*** The idea behind Artificial Bee Colony [16] is to form a simplified model to simulate the swarm intelligence of honey bees in maintaining the ecosystem of a honeycomb. In the model, the to-be-improved individuals (solutions) are referred to as food sources, and an employed bee is assigned to each food source. During the optimization process, the employed bee would first try to search for another food source to find a better food location. Afterward, bees of another type, called onlookers, are sent out to exploit each food source further (a local search). If the food source does not improve after a prescribed limit of iterations, the food source is abandoned, and a new one is established randomly in the search space.

*4) **Cuckoo Search(CS):*** The essential concept of Cuckoo Search [17] is to simulate the unique hatching strategy of cuckoo birds. Unlike other birds, some cuckoo species lay their eggs in other birds' nests called brood parasites. CS performs a different updating strategy on each to-be-improved individual—the host nest to simulate this behavior. In each iteration, a cuckoo would randomly move to a location around the assigned host nest by a particular method called Levy flight. After evaluating the location, another host nest is randomly picked; if the location is better, the newly picked nest will move on to that location. At the end of each iteration, a specified portion of nests would be abandoned, and new ones would be randomly located in the search space.

*5) **Shuffled Frog-Leaping Algorithm (SFLA)**:* In Shuffled Frog-Leaping Algorithm [18], the population is first divided into sub-populations called memeplexes. In an iteration, the goal is to improve the worst individual in each memeplex by moving it toward the best individual of the memeplex. If not improved, it would then try to move toward the best individual of the entire population. If still not improved, generate a random solution to replace it. After several such iterations, all memeplexes would reunite, and new memeplexes would be formed.

*6) **Imperialist Competitive Algorithm (ICA)**:* As the name suggests, Imperialist Competitive Algorithm [9] is inspired by the conquer or be conquered operations in the 19^{th} and the 20^{th} century, when western countries were trying to expand their power and spread their culture. Following this concept, ICA refers to each solution agent as a country. The best countries are assigned as the empires (or mother countries to prevent confusion), while others are colonies. In the initialization stage, colonies are partitioned by mother countries to form empires. During the optimization process, each colony in an empire would move toward the mother country, and once a better solution is found, the mother country would move on to that place. After updating the locations of colonies, competition between each empire would start, and the weakest empire, calculated by the total fitness of the empire, should give up one of its colonies to the most substantial empire.

*7) **Particle Swarm Optimization (PSO)**:* Particle swarm optimization [6] is a population-based algorithm using swarm intelligence. Each individual is analogous to a particle in PSO. In each iteration, each particle would move toward the best individual so far and its own previously best location. The strengths of the two factors are controlled by two parameters called learning parameters or acceleration constants.

*8) **Flower Pollination Algorithm (FPA)**:* Similar to Particle Swarm Optimization, each individual in Flower Pollination Algorithm [19] either moves toward the best individual so far by Levy flight or conducts a local search around itself. The selection between the two methods is determined by chance.

*9) **Bat Algorithm (BA)**:* With an almost identical approach as Flower Pollination Algorithm, the only differences between Bat Algorithm [20] and Flower Pollination Algorithm are that the chance level between the two methods is adaptive during the process and the Levy flight is replaced.

*10) **Teaching-Learning-based Optimization (TLBO)**:* In each iteration, there are two stages: the teaching stage and the learning (competition) stage [21]. In the learning stage, the difference between the mean of all individuals and the teacher (the best individual) is calculated, and all individuals are expected to move closer to the teacher. After updating the location, all individuals would start competing with others by randomly selecting another individual and deciding whether to move toward or away from it by comparing their fitness.

*11) **Grey Wolf Optimizer (GWO)**:* In Grey Wolf Optimizer [22], the population is modeled as a wolf herd. In the herd, the three leading wolves are called alpha, beta, and delta, respectively. During the optimization process, these three leading wolves are the top three individuals and would significantly affect the searching direction of the entire herd. Therefore, in the algorithm, all other wolves will simultaneously move toward alpha, beta, and delta to locate in the promising search region.

*12) **Sine Cosine Algorithm (SCA)**:* Unlike Grey Wolf Optimizer, Sine Cosine Algorithm [23] only leverages the best individual as the moving direction. By coupling the sine and the cosine terms in the location updating function, the algorithm is thus called by the name.

*13) **Whale Optimization Algorithm (WOA)**:* In contrast to GWO and SCA, Whale Optimization Algorithm [24] adopts more conditions when updating the location. First, each individual would either round in the best individual or move toward others, which is determined stochastically. If the latter move is selected, it has to determine whether apply the exploration operation or the exploitation operation with the magnitude of a parameter indicating the progress.

*14) **Genetic Algorithm (GA)**:* Developed by Holland in 1975 [5], the Genetic Algorithm (GA) has been one of the most popular metaheuristic algorithms. Inspired by the concept of natural selection proposed by C. Darwin, GA intends to find the optimal solution to the problem through a series of processes that mimic the evolution of creatures under excessive fertility and limited resources. In GA, the goal is to find the genotype (solution) with the best phenotype (objective value) in terms of its fitness (optimality) through gene manipulations and competition among all chromosomes (explained later). The solution agent of GA is called the chromosome, derived from biological terminology that organelle consists of genes. The gene encodings may vary for different optimization problems, yet the gene value manipulations are categorized into crossover, mutation, and selection. In each generation, initially, there are parent chromosomes with the size of N_p. With two user-specified parameters: the crossover rate (Cr) and the mutation rate (Mr), one can control the extent of crossover and mutation. The typical setting of CR is [0.5, 0.95], and MR is [0.01, 0.3] (refer to [25] for a comprehensive discussion about setting crossover and mutation rates). After crossover and mutation operations, a new population of chromosomes has resulted. The selection operation is adopted to choose the elite chromosomes as new parents for the next generation, and the remaining chromosomes are discarded.

*15) **Memetic Algorithm (MA)**:* Basically, Memetic Algorithm [26] is the same as Genetic algorithm. The only difference is that before selection, each chromosome (or meme) would be given the opportunity to further improve itself by performing a local search.

*16) **Simulated Annealing**:* Simulated annealing was introduced by Kirkpatrick et al. in 1983[7], based on the early work by Metropolis et al. [27]. SA mimics the process of cooling materials. During the procedure, due to the high initial temperature, each particle would fluctuate dramatically at first. After the temperature drops, each particle gradually moves to a relatively low energy position and forms the crystalline lattices. Following this concept, one can model the cooling process based on Maxwell–Boltzmann statistics, which describes the distribution of material particles with different energy states under thermal equilibrium [28].

*17) **Harmony Search (HS):*** Harmony Search [29] is inspired by the process of music composing and improvising. The basic concept of Harmony Search is to construct each note (design variable) by three methods: using other existing notes, adjusting the note pitch (local random walk), or choosing randomly within the variable bound.

*18) **Gravitational Search Algorithm (GSA):*** The Gravitational Search Algorithm [30] is established on the concept of celestial mechanics. Based on the assumption that fitter solution agents (or objects) should be endowed with more mass, one can compute the gravitational forces of each object exerted on other objects and update the position following the rule of gravity and object movement with some random deviations. At the end of the optimization process, one should expect to find the optimal solution at the position where all objects are located after convergence.

*19) **Firefly Algorithm (FA):*** With a similar updating mechanism as GSA, Firefly Algorithm [31], instead, is inspired by the mating mechanism of fireflies. In FA, the analogy of mass in GSA is the brightness of the flashing light of each firefly (solution agent) which is proportional to its fitness. After finding the distance between each firefly, one can obtain the light intensity, exponentially decaying with respect to distance, from other fireflies and update its position.

IV. Numerical Tests

A. *Pressure Vessel Design*

The pressure vessel design problem was first proposed in [12]. The goal is to minimize the welding cost, the material cost, and the forming cost, which can be modeled as the following.

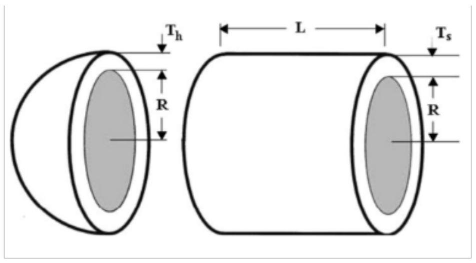

Fig. 1. Pressure Vessel Design Problem [20]

$$x = \begin{bmatrix} T_s & T_h & R & L \end{bmatrix}^T = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \end{bmatrix}^T \tag{1}$$

$$\begin{aligned} f(x) =& 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1611x_1^2x_4 \\ &+ 19.8621x_1^2x_3 \end{aligned} \tag{2}$$

And the constraints can be modeled as below.

$$g_1(x) = 0.0193x_3 - x_1 \le 0 \tag{3}$$
$$g_2(x) = 0.00954x_3 - x_2 \le 0 \tag{4}$$
$$g_3(x) = 1296000 - \pi x_3^2 x_4 - \frac{4}{3}\pi x_3^3 \le 0 \tag{5}$$
$$g_4(x) = x_4 - 240 \le 0 \tag{6}$$
$$x_1 \in [1.125, 99 \times 0.0625] \tag{7}$$
$$x_2 \in [0.625, 99 \times 0.0625] \tag{8}$$
$$x_3 \in [50, 70] \tag{9}$$
$$x_4 \in [30, 50] \tag{10}$$

Where T_s and T_h should be multiples of 0.0625, the nominal thickness of rolled plate. The violation amounts were multiplied by a penalty factor of 1e6 and combined into the objective function.

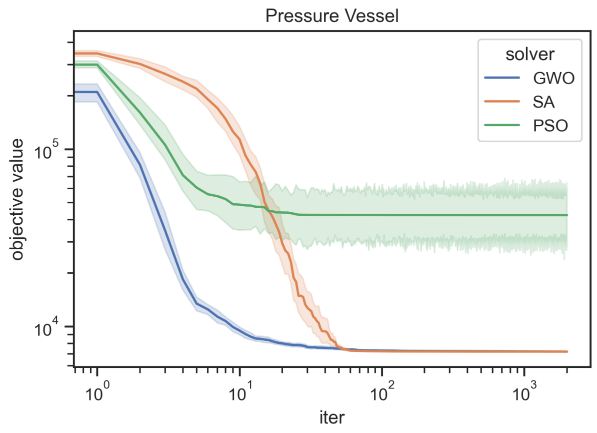

Fig. 2. Convergence curve of PSO, SA, and GWO in pressure vessel design problem

Table II and Fig. 2 shows that GWO and SA have the most robust performance in terms of the best-case scenario and the mean performance. While PSO has the potential ability to acquire promising results, but yet it is easier to be trapped in local minimum.

B. *Tension/compression Spring Design*

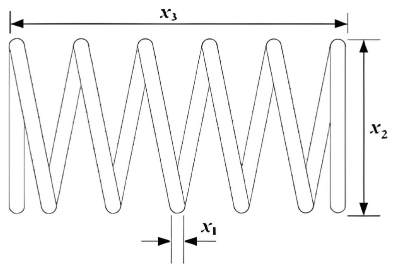

Fig. 3. Tension/compression Spring Design Problem[20]

The goal of the design problem is to minimize the total weight (volume) by changing the configurations of the spring.

TABLE II
RESULT OF PRESSURE VESSEL DESIGN

Solver	Best	Mean	STD
GSA	43086.94	317658.8	104803.8
SFLA	63847.4	177825.4	50083.42
PSO	7310.856	42388.23	43484.35
BA	7263.999	8613.832	1073.165
GA	7304.813	7638.339	617.9234
CS	7202.354	7475.772	202.854
WOA	7235.009	7386.61	126.6179
MA	**7198.005**	7365.327	160.1555
FA	7310.856	7310.858	**0.002448**
FPA	**7198.005**	7246.907	56.8776
TLBO	**7198.005**	7228.099	50.75764
ICA	**7198.005**	7220.576	45.91202
SCA	7199.279	7216.108	10.36912
ALO	**7198.005**	7215.008	23.0375
HS	7199.062	7210.739	10.48913
ABC	**7198.005**	7210.049	34.24981
DE	**7198.005**	7203.884	18.24369
SA	7198.179	7199.674	0.82635
GWO	7198.282	**7199.526**	0.765927

The design variables are: x_1 (wire diameter), x_2 (mean coil diameter), and x_3 (number of active coils). And the objective function and constraints can be modeled as follows.

$$f(x) = (x_3+2)x_2x_1^2 \tag{11}$$

$$g_1(x) = 1 - \frac{x_2^3 x_3}{71875x_1^4} \leq 0 \tag{12}$$

(Deflection constraint)

$$g_2(x) = \frac{4x_2^2 - x_1x_2}{4000\pi(x_1^3x_2 - x_1^4)} + \frac{0.615}{1000\pi x_x^2} - 1 \leq 0 \tag{13}$$

(Maximum shear stress)

$$g_3(x) = 1 - \frac{\sqrt{\frac{1.15\times10^{11}}{14.76684}}}{200\pi x_1 x_2^2 x_3} \leq 0 \tag{14}$$

(Frequency of surge wave)

$$g_4(x) = \frac{x_1+x_2}{1.5} - 1 \leq 0 \tag{15}$$

(Diameter constraint)

$$x_1 \in [0.05, 0.2] \tag{16}$$

$$x_2 \in [0.25, 1.3] \tag{17}$$

$$x_3 \in [2, 15] \tag{18}$$

In Table III, the overall performances of all algorithms are not significantly different. After examining the convergence curves, it is obvious that DE and FPA show a significant convergent ability compared to GA and GSA.

C. Gear Train Design

The problem was originally model by [12], and the goal is to find the best teeth number of each gear that would produce a gear ratio as close as possible to $\frac{1}{6.931}$.

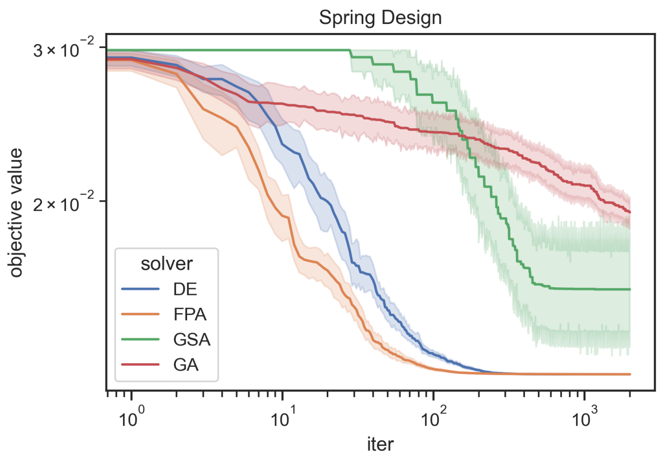

Fig. 4. Convergence curve of the best and the worst algorithms in Tension/compression Spring Design Problem

TABLE III
RESULT OF TENSION/COMPRESSION SPRING DESIGN

Solver	Best	Mean	STD
GA	0.015396	0.019423	0.001896
GSA	0.012792	0.015854	0.006326
MA	0.012817	0.015373	0.00194
SFLA	0.013041	0.015106	0.002156
ICA	0.012683	0.013978	0.001877
HS	0.012739	0.013512	0.000654
PSO	0.012678	0.013321	0.001239
WOA	0.012907	0.013211	0.000507
BA	0.01268	0.013162	0.000899
SCA	0.012777	0.012967	0.000127
SA	0.01274	0.012951	0.000126
ABC	0.012684	0.012898	0.000161
ALO	**0.012678**	0.012866	0.000178
CS	**0.012678**	0.012844	0.000202
FA	0.012763	0.012806	2.76E-05
GWO	0.012683	0.012721	1.93E-05
TLBO	**0.012678**	0.012686	7.80E-06
FPA	**0.012678**	**0.012678**	1.62E-06
DE	**0.012678**	**0.012678**	**1.17E-06**

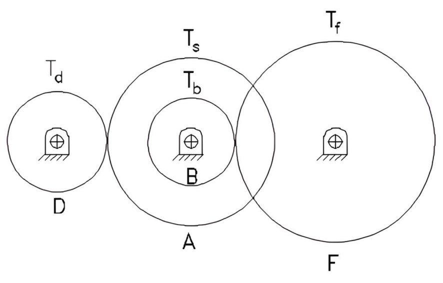

Fig. 5. Gear Train Design Problem[32]

The objective function is modeled as below.

$$x = \begin{bmatrix} T_d & T_b & T_a & T_f \end{bmatrix}^T \tag{19}$$

$$f(x) = (\frac{1}{6.931} - \frac{x_1 x_2}{x_3 x_4})^2 \tag{20}$$

$$x_i \in [12, 60] \text{ and } x_i \in \mathbb{N}, \ i = 1, 2, 3, 4 \tag{21}$$

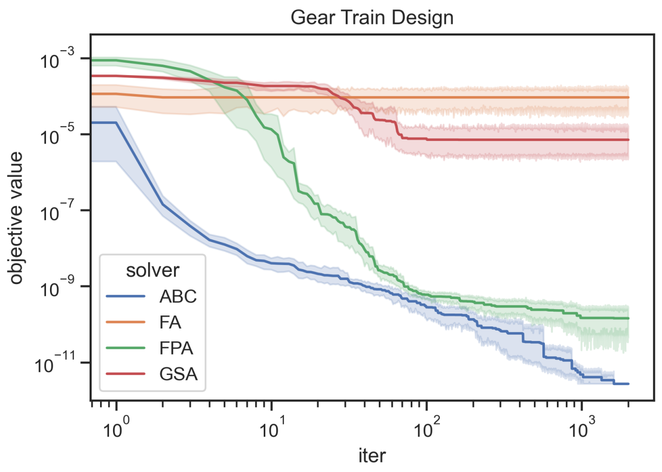

Fig. 6. Convergence curve of ABC, FA, FPA, and GSA in gear train design problem

The results show little information about each algorithm, but in Fig. 6, it is obvious that FA and GSA had prematurely converged while ABC and FPA still kept the improvement tendency toward the end of solution evolution.

TABLE IV
RESULT OF GEAR TRAIN DESIGN

Solver	**Best**	**Mean**	**STD**
FA	1.83E-08	9.18E-05	0.000173
GSA	1.62E-08	7.06E-06	1.97E-05
GA	2.36E-09	4.90E-06	8.58E-06
MA	2.31E-11	3.35E-06	9.13E-06
BA	1.83E-08	9.02E-07	9.83E-07
PSO	8.70E-09	9.86E-08	1.84E-07
SFLA	**2.70E-12**	4.25E-08	1.32E-07
ALO	**2.70E-12**	7.82E-09	7.61E-09
SA	8.89E-10	6.71E-09	4.68E-09
ICA	2.31E-11	5.60E-09	1.52E-08
CS	2.31E-11	3.24E-09	4.98E-09
DE	**2.70E-12**	2.14E-09	2.92E-09
WOA	**2.70E-12**	2.01E-09	5.80E-09
TLBO	**2.70E-12**	1.30E-09	3.23E-09
SCA	**2.70E-12**	1.01E-09	6.54E-10
HS	**2.70E-12**	3.28E-10	5.27E-10
GWO	**2.70E-12**	3.19E-10	4.52E-10
FPA	**2.70E-12**	1.43E-10	3.19E-10
ABC	**2.70E-12**	**2.70E-12**	**1.23E-27**

V. CONCLUSION AND DISCUSSION

In the numerical tests, different algorithms have shown quite diverged behaviors in their optimizing processes. But by comparing with each other on the same benchmark problem, the convergence curves can be easily observed to come to the conclusion of which algorithm is better in solving that problem. Generally, the one that converges faster in the early stage tends to acquire a better solution at the end. This tendency is repeatedly discovered in nearly every benchmark. Therefore, it is suggested that choosing the one with more convergence ability is always a good start in solving various design optimization problems.

REFERENCES

[1] A. Parkinson, R. Balling, and J. Hedengren, *Optimization Methods for Engineering Design*, 2nd ed. Brigham Young University, 2018. [Online]. Available: http://apmonitor.com/me575/index.php/Main/BookChapters

[2] K. J. Waldron, G. L. Kinzel, and S. K. Agrawal, *Kinematics, Dynamics, and Design of Machinery*. New York, UNITED KINGDOM: John Wiley Sons, Incorporated, 2016. [Online]. Available: http://ebookcentral.proquest.com/lib/ntuedu/detail.action?docID=4526800

[3] M. s. Saad, H. Jamaluddin, and I. Mat Darus, "Pid controller tuning using evolutionary algorithms," *WSEAS Transactions on Systems and Control*, vol. 7, pp. 139–149, 2012.

[4] *More Natural Optimization Algorithms*, 2003, pp. 187–203. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/0471671746.ch7https://onlinelibrary.wiley.com/doi/10.1002/0471671746.ch7

[5] J. H. Holland, *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[6] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4, Conference Proceedings, pp. 1942–1948 vol.4. [Online]. Available: https://ieeexplore.ieee.org/document/488968/

[7] S. Kirkpatrick, J. Gelatt, C. D., and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–80, 1983, kirkpatrick, S Gelatt, C D Jr Vecchi, M P eng Science. 1983 May 13;220(4598):671-80. doi: 10.1126/science.220.4598.671. [Online]. Available: https://www.ncbi.nlm.nih.gov/pubmed/17813860https://science.sciencemag.org/content/sci/220/4598/671.full.pdf

[8] R. Storn and K. Price, "Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997. [Online]. Available: https://doi.org/10.1023/A:1008202821328https://link.springer.com/content/pdf/10.1023/A:1008202821328.pdf

[9] E. Atashpaz-Gargari and C. Lucas, "Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition," in *2007 IEEE Congress*

on Evolutionary Computation, Conference Proceedings, pp. 4661–4667.

[10] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

[11] S.-J. Wu and P.-T. Chow, "Genetic algorithms for solving mixed-discrete optimization problems," *Journal of the Franklin Institute*, vol. 331, no. 4, pp. 381–401, 1994. [Online]. Available: http://www.sciencedirect.com/science/article/pii/0016003294900043

[12] E. Sandgren, "Nonlinear integer and discrete programming in mechanical design optimization," *Journal of Mechanical Design*, vol. 112, no. 2, pp. 223–229, 1990. [Online]. Available: https://doi.org/10.1115/1.2912596https://asmedigitalcollection.asme.org/mechanicaldesign/article-abstract/112/2/223/417355/Nonlinear-Integer-and-Discrete-Programming-in?redirectedFrom=fulltext

[13] C. A. Coello Coello, "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*, vol. 191, no. 11-12, pp. 1245–1287, 2002. [Online]. Available: https://dx.doi.org/10.1016/s0045-7825(01)00323-1

[14] S. K. Acharyya and M. Mandal, "Performance of eas for four-bar linkage synthesis," *Mechanism and Machine Theory*, vol. 44, no. 9, pp. 1784–1794, 2009.

[15] S. Mirjalili, "The ant lion optimizer," *Advances in Engineering Software*, vol. 83, pp. 80–98, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0965997815000113

[16] D. Karaboga and B. Basturk, "On the performance of artificial bee colony (abc) algorithm," *Applied Soft Computing*, vol. 8, no. 1, pp. 687–697, 2008. [Online]. Available: https://dx.doi.org/10.1016/j.asoc.2007.05.007

[17] X.-S. Yang and S. Deb, "Cuckoo search via lévy flights," in *2009 World congress on nature biologically inspired computing (NaBIC)*. Ieee, Conference Proceedings, pp. 210–214.

[18] M. M. Eusuff and K. E. Lansey, "Optimization of water distribution network design using the shuffled frog leaping algorithm," *Journal of Water Resources planning and management*, vol. 129, no. 3, pp. 210–225, 2003.

[19] X.-S. Yang, "Flower pollination algorithm for global optimization," in *International conference on unconventional computing and natural computation*. Springer, Conference Proceedings, pp. 240–249.

[20] A. H. Gandomi, X.-S. Yang, A. H. Alavi, and S. Talatahari, "Bat algorithm for constrained optimization tasks," *Neural Computing and Applications*, vol. 22, no. 6, pp. 1239–1255, 2013. [Online]. Available: https://doi.org/10.1007/s00521-012-1028-9https://link.springer.com/content/pdf/10.1007/s00521-012-1028-9.pdf

[21] R. V. Rao, V. J. Savsani, and D. Vakharia, "Teaching–learning-based optimization: a novel method for constrained mechanical design optimization problems," *Computer-Aided Design*, vol. 43, no. 3, pp. 303–315, 2011.

[22] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Advances in Engineering Software*, vol. 69, pp. 46–61, 2014. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0965997813001853

[23] S. Mirjalili, "Sca: A sine cosine algorithm for solving optimization problems," *Knowledge-Based Systems*, vol. 96, pp. 120–133, 2016. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0950705115005043https://www.sciencedirect.com/science/article/abs/pii/S0950705115005043?via%3Dihub

[24] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Advances in engineering software*, vol. 95, pp. 51–67, 2016.

[25] A. Hassanat, K. Almohammadi, E. Alkafaween, E. Abunawas, A. Hammouri, and V. B. S. Prasath, "Choosing mutation and crossover ratios for genetic algorithms—a review with a new dynamic approach," *Information*, vol. 10, no. 12, p. 390, 2019. [Online]. Available: https://www.mdpi.com/2078-2489/10/12/390

[26] P. Moscato, "On evolution, search, optimization, genetic algorithms and martial arts: Towards memetic algorithms," *Caltech concurrent computation program, C3P Report*, vol. 826, p. 1989, 1989.

[27] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *The Journal of Chemical Physics*, vol. 21, no. 6, pp. 1087–1092, 1953. [Online]. Available: https://dx.doi.org/10.1063/1.1699114https://aip.scitation.org/doi/pdf/10.1063/1.1699114

[28] A. H. Carter, "Classical and statistical thermodynamics," 2000.

[29] Z. W. Geem, J. H. Kim, and G. V. Loganathan, "A new heuristic optimization algorithm: Harmony search," *Simulation*, vol. 76, no. 2, pp. 60–68, 2001, 434mg Times Cited:3246 Cited References Count:19. [Online]. Available: ⟨GotoISI⟩://WOS:000168819000001https://journals.sagepub.com/doi/pdf/10.1177/003754970107600201

[30] E. Rashedi, H. Nezamabadi-Pour, and S. Saryazdi, "Gsa: a gravitational search algorithm," *Information sciences*, vol. 179, no. 13, pp. 2232–2248, 2009.

[31] X.-S. Yang, "Firefly algorithms for multimodal optimization," in *International symposium on stochastic algorithms*. Springer, Conference Proceedings, pp. 169–178.

[32] A. Kaveh and S. Talatahari, "Imperialist competitive algorithm for engineering design problems," *Asian Journal of Civil Engineering (Building and Housing)*, vol. 11, 01 2010.