sustainability

MDPI

Article

Improved Dual-Population Genetic Algorithm: A Straightforward Optimizer Applied to Engineering Optimization

Zhihua Chen [1,2], Xuchen Xu [2] and Hongbo Liu [1,3,*]

[1] State Key Laboratory of Hydraulic Engineering Simulation and Safety, Tianjin University, Tianjin 300072, China; zhchen@tju.edu.cn
[2] Department of Civil Engineering, Tianjin University, Tianjin 300072, China; xcxu_2015@163.com
[3] Department of Civil Engineering, Hebei University of Engineering, Handan 056038, China
* Correspondence: hbliu@tju.edu.cn

Abstract: Aiming at the current limitations of the dual-population genetic algorithm, an improved dual-population genetic algorithm (IDPGA) for solving multi-constrained optimization problems is proposed by introducing a series of strategies, such as remaining elite individuals, a dynamic immigration operator, separating the objective and constraints, normalized constraints, etc. We selected 14 standard mathematical benchmarks to check the performance of IDPGA, and the results were compared with the theoretical value of CEC 2006. The results show that IDPGA with the current parameters obtains good solutions for most problems. Then 6 well-known engineering optimization problems were solved and compared with other algorithms. The results show that all of the solutions are feasible, the solution precision of IDPGA is better than other algorithms, and IDPGA performs with good efficiency and robustness. Meanwhile, no parameters need to be ignored when IDPGA is applied to solving engineering problems, which is enough to prove that IDPGA is suitable for solving engineering optimization. A Friedman test showed no significant difference between IDPGA and six algorithms, but significant differences between IDPGA and seven other algorithms; thus, a larger number of evaluators will be needed in the future. In addition, further research is still needed about the performance of IDPGA for solving practical large-scale engineering problems.

Keywords: remaining elite individuals; dynamic immigration operator; dual-population; genetic algorithm; IDPGA; engineering optimization

check for updates

Citation: Chen, Z.; Xu, X.; Liu, H. Improved Dual-Population Genetic Algorithm: A Straightforward Optimizer Applied to Engineering Optimization. *Sustainability* **2023**, *15*, 14821. https://doi.org/10.3390/su152014821

Academic Editor: Gianluca Mazzucco

Received: 19 August 2023
Revised: 1 October 2023
Accepted: 10 October 2023
Published: 12 October 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

In recent years, metaheuristic algorithms have developed rapidly, including genetic algorithms [1], simulated annealing algorithms [2], particle swarm algorithms [3], neural networks [4], ant colony algorithms [5], artificial fish-swarm algorithms [6], firefly algorithms [7], cuckoo search [8], tree growth algorithms [9], interactive fuzzy algorithms [10], chaos game algorithms [11], aquila optimizer [12], giant trevally optimizer [13] and termite life cycle optimizer [14]. Metaheuristic algorithms do not need information about first-order Jacobian matrix or second-order Hessian matrix and break through the traditional mathematical programming algorithms based on gradients.

With the improvement of the computing power of computers, metaheuristic algorithms are more and more widely used in engineering optimization. Hasançebi et al. [15] evaluated the performance of seven algorithms in optimum design of pin-jointed structures. The results revealed that simulated annealing and evolution strategies are most powerful. Gandomi et al. [16] utilized a cuckoo search algorithm to solve 13 design problems and obtained better results. Kazemzadeh Azad et al. [17] presented a guided stochastic search technique for discrete sizing optimization of steel trusses and used less computational effort to solve the problem. Gholizadeh and Milany [18] improved a fireworks algorithm

and solved the discrete structural optimization problems of steel trusses and frames. The results demonstrated that the algorithm performed well in terms of the optimum solutions and the convergence rate. Savvides and Papadopoulos [19] formed a Feed Forward Neural Network for the estimation of stresses and strains at failure for shallow foundations in cohesive soils, which contributes to a more efficient and reliable foundation design. Degertekin et al. [20] proposed a novel Jaya (a Sanskrit word meaning victory) formulation for discrete optimization of truss structures including discrete sizing, layout and topology optimization. Kaveh and Javadi [21] presented two chaotic firefly algorithms and applied them to multiple frequency constraints optimization of some large-scale domes. The results illustrated a desirable performance. Lee and Hyun [22] used a genetic algorithm to schedule modular construction projects and helped managers allocate resources efficiently. Adil and Cengiz [23] developed weighted superposition attraction and implemented it for sizing optimization of truss structures. Degertekin et al. [24] developed a parameter-free Jaya algorithm for sizing and layout optimization of truss structures and obtained superior results. Inspired by eagles' (genus *Aquila*) behaviors in catching the prey, Abualigah et al. [12] proposed the Aquila Optimizer (AO). A set of mathematical and engineering problems was then selected to validate AO's ability to find the optimal solution, and the results show AO has great superiority. Finally, AO was suggested to be applied to other real-world problems. Inspired by the feeding behavior of giant trevally, Sadeeq and Abdulazeez [13] proposed the Giant Trevally Optimizer (GTO). A set of problems was used to evaluate GTO's performance and a Wilcoxon rank sum test was used to compare GTO with other algorithms. The results show that GTO is reliable. Based on both the life cycle of a termite colony and the modulation of movement strategies, Minh et al. [14] proposed the termite life cycle optimizer (TLCO) which mimics the activities of termites. TLCO was used to find the global optimum in classic optimization problems. Compared with other algorithms, the results demonstrate that TLCO is effective and reliable.

Genetic algorithms are the simplest metaheuristic algorithms, but the simple genetic algorithm (SGA) has some defects such as easily falling into local optimum and premature convergence. In order to overcome these shortcomings as much as possible, the idea of dual-population is a feasible way to a certain extent. Two subpopulations evolve separately; one is called the developing subpopulation, with small probabilities of recombination and mutation, and the other is called the detecting subpopulation, with large probabilities of recombination and mutation. The two subpopulations are used to search for local and global optimum. After each or several iterations, two subpopulations exchange their own immigration operators to break the balance of single population evolution and maintain the diversity of population.

In the past, some scholars have studied the idea of dual-population genetic algorithms (DPGAs) from both theory and application. At the level of the algorithm, Srinivasa et al. [25] proposed a self-adaptive immigration model, where parameters of population size, the number of points of crossover and mutation rate for each population were fixed adaptively. Further, the immigration of individuals between populations was decided dynamically. Li et al. [26] proposed one novel agent genetic algorithm, the multi-population agent genetic algorithm for feature selection (MPAGAFS), by constructing a double chain-like agent structure and with improved genetic operators. Park et al. [27] proposed a dual-population GA which used an additional population as a reservoir of diversity and adjusted the distance dynamically to achieve an appropriate balance between exploration and exploitation. Umbarkar et al. [28] proposed the binary encoded multithreaded parallel DPGA to solve the problems of population diversity and premature convergence. Liu et al. [29] proposed a multipopulation parallel genetic algorithm to solve multimodel functions. Several subspaces which had no overlaps and reflect whole characters of the functions were gained through even partition. A parallel optimum search was realized in each subspace and excellent individuals were collected after every evolution, and then the optimum solution was searched in whole space. Li et al. [30] improved the DPGA, such that the parents with higher similarity were crossed at a higher rate and mutated with a general operator.

The new individual replaced the worst individual in one subpopulation, and parents with less similarity were crossed at a higher rate, mutated with a large operator, and the new individual replaced the most similar individual of the old population in the other subpopulation. Pourvaziri et al. [31] developed an effective novel hybrid multi-population genetic algorithm in which solution space was separated into different parts, and each subpopulation represented a separate part. A powerful local search mechanism based on simulated annealing was developed, and the operators were designed to search only the feasible space. Fang and Yu [32,33] introduced the "war" model to try to avoid premature convergence, and solved the Schaffer function, Banana function, Mexican Hat function, Himmelblau function and Camel function. Lei and Jiang [34] formed a DPGA based on autonomic computing, and solved the Himmelblau function, Shekel's Foxholes function and Mexican Hat function. Tan et al. [35] proposed a DPGA with chaotic local search strategy, and solved the Sphere function, Rastrigin function, Ackley function, Griewank function and Generalized Penalized function. Tian et al. [36] introduced individual similarity into DPGA and solved the Camel function, Schaffer function and Banana function. Guo et al. [37] used bi-group random variables as the mutation operator in DPGA, and solved the Sphere function, Rosenbrock function, Rastrigin function, Ackley function and Griewank function.

Although scholars have carried out studies on DPGA, on the one hand, most of these problems are unconstrained optimization problems, and on the other hand, the number of problems is small. In other words, the universality of the algorithm needs to be verified, and further research on DPGA that can solve constrained optimization problems is needed.

At the level of application, Ji et al. [38] developed an adaptive genetic algorithm for optimal path planning, which introduces a penalty function, calculates the fitness of each individual, and adopts roulette selection. The population is divided into three parts, including one part directly remaining and two parts participating in internal evolution and external evolution. However, only one example verifies that the algorithm is effective in solving path planning. Obviously, whether the algorithm is effective for other problems in spaceflight and other engineering fields still needs to be explored; meanwhile, introducing a penalty function takes more time, and dividing the population into more parts may lead to cumbersome operations. In addition, the idea of dual-population has rarely been applied to the field of structural engineering. Li and Wang [39] proposed a novel strategy for crossarm length optimization of prestressed stayed steel columns (PSSCs) based on multi-dimensional global optimization algorithms. Wang et al. [40] obtained the directional design wind speeds for a given city via multi-parameter optimization, which was more suitable in the commonly used simplified method for estimating the wind loads. The applicability, economy and safety of the sector-by-sector method were discussed. Gao et al. [41] proposed a multi-population genetic algorithm program for wind turbine layout optimization in a wind farm. Zheng and Xie [42] introduced an adaptive operator into DPGA, and conducted optimal design of composite laminates. This problem is relatively simple, and DPGA was simply applied to engineering problems. In other words, there is a lack of systematic research on the application of DPGA in structural engineering.

DPGA has also seldom been applied in the optimization design of grid structures. According to the existing literature, there is little research on SGA in solving such optimization problems. Gong and Liu [43] directly applied SGA to optimize a rectangular square pyramid space truss, and a satisfactory result was obtained, but the algorithm was not stable and did not converge in the end. Chen et al. [44] applied SGA to optimize a 10-bar planar truss. SGA was run twice; firstly, a more accurate variable limit was preliminarily determined, and secondly, the variable limit was changed so that the algorithm can jump out of the local optimal. However, the result was very imprecise. Mu et al. [45,46] directly applied SGA to optimize a three-bar planar truss, and introduced niche technique and fuzzy control into SGA and solved an optimization design problem of a single-layer dome structure. Tan et al. [47] directly applied SGA to optimize a 10-bar planar truss and a rectangular square pyramid space truss, but the results were rough. Zhang et al. [48]

adopted a fitness transformation strategy in SGA and optimized a prestressed space lattice structure, but the algorithm was greatly affected by the number of decision variables and algorithm parameters.

When SGA is used to solve the optimization problem of grid structures, on the one hand, the solution accuracy is not high or the algorithm is not convergent; on the other hand, there are few examples to be solved, so the general adaptation of SGA needs further research.

Combined with DPGA and its application in grid structures, due to unfamiliarity of most engineers with theoretical knowledge of algorithms, from the few applications, it can be seen that there is a lack of systematic research on the dual-population algorithm itself to solve grid structure optimization problems. At present, most applications simply transplant SGA into certain problems, which does not completely solve the target problem, because the algorithm itself is not perfect. Meanwhile, considering the simple application of DPGA in other structural engineering problems, scholars often need to ignore some parameters to reduce the complexity of the problem, so that the simple dual-population algorithm can solve the target problem. In this way, the results are not necessarily reliable. For instance, Li and Wang [39] only considered the crossarm length, but the initial pretension, cable diameter, number of bays, etc. were still important parameters affecting the behavior of PSSCs. At the same time, such algorithms are not universal; in other words, the algorithms may only be suitable for solving a certain problems.

To sum up, it is necessary to carry out systematic research on the application of DPGA in the optimization design of grid structure from the algorithm level. In this paper, the improved dual-population genetic algorithm (IDPGA) is described by introducing some strategies, and the composition of the algorithm is explained in detail. Then, a series of mathematical optimization problems and small-scale engineering optimization problems are used to verify its reliability. On the one hand, the application range of DPGA is further expanded, and on the other hand, the core algorithm conditions are created for the joint optimization frame between IDPGA and finite element analysis software in a follow-up study; this frame will be utilized for grid structure optimization design problems. In fact, no single algorithm exists to solve all problems [49–53], so it is always necessary to develop new algorithms.

2. Construction of the Improved Dual-Population Genetic Algorithm

An engineering optimization problem can be abstracted as a mathematical optimization problem. The basic mathematical form is shown in Equation (1), where $f(\mathbf{x})$ is objective, and $g_i(\mathbf{x})$ and $h_i(\mathbf{x})$ are inequality and equality constraints, respectively:

$$\begin{aligned} &\min f(\mathbf{x}) \\ &s.t. \begin{cases} g_i(\mathbf{x}) \le 0,\ i = 1,\ 2,\ \ldots,\ m \\ h_i(\mathbf{x}) = 0,\ i = m+1,\ m+2,\ \ldots,\ n \end{cases} \end{aligned} \tag{1}$$

Based on the idea of two-population evolution, the improved dual-population genetic algorithm (IDPGA) is presented by introducing a series of strategies, including remaining elite individuals, a dynamic immigration operator, separating the objective and constraints, normalized constraints, etc.

2.1. Remaining Elite Individuals Strategy and a Dynamic Immigration Operator

To avoid the loss of the best individual in the subpopulation, the best individual of each generation is preserved and does not participate in the recombination and mutation of the next iteration; this is termed the 'elite individual remaining' strategy. Simultaneously, a fixed number of immigration operators only breaks the opposite evolution balance to a limited and fixed extent. On the one hand, the number of immigrating individuals should be small to break the balance, because exchanging too many individuals may lead to too discrete distribution of individuals, which destroys the evolution process and even causes difficultly for the algorithm to converge. On the other hand, it is more suitable for the laws

of nature that the number of immigration operators changes dynamically. Therefore, a dynamic immigration operator was introduced, and the number was recommended to take a random integer between 5 and 1/10 popsize.

2.2. Constraint Handling Mechanism

Most engineering optimization problems are nonlinear constrained optimization problems. The key to solving constraint optimization problems lies in how to deal with constraints. There have been a series of methods in the past, mainly including combining constraints with objectives, considering constraints and objectives separately, and so on. Applying penalty functions is a typical method of handling constraints, such as static penalty functions [54–56], dynamic penalty functions [57,58], annealing penalty functions [57,59,60], self-adaptive penalty functions [61–63], etc. Penalty functions convert the constraints to objectives by a penalty function as shown in Equation (2). $G(\mathbf{x})$ is the objective after conversion, C is a penalty factor, $\varphi(\mathbf{x})$ is a combined penalty item, and ε is the tolerance error of equality constraints:

$$\begin{aligned} G(\mathbf{x}) &= f(\mathbf{x}) + C\varphi(\mathbf{x}) \\ \varphi(\mathbf{x}) &= \sum_{\mathrm{i}=1}^{n} \varphi_i(\mathbf{x}) \\ \varphi_i(\mathbf{x}) &= \begin{cases} \max\{0,\ g_i(\mathbf{x})\}, & i = 1,\ 2,\ \ldots,\ m \\ \max\{0,\ |h_i(\mathbf{x})| - \varepsilon\}, & i = m+1,\ m+2,\ \ldots,\ n \end{cases} \end{aligned} \tag{2}$$

Thus, the original problem is converted to the unconstrained problem as shown in Equation (3):

$$\min G(\mathbf{x}) \tag{3}$$

There are also some other constraint-handling strategies. Deb [64] exploited an approach and made pairwise comparisons in tournament selection operator to devise a penalty function. Feasible and infeasible solutions were compared carefully, and a niching method was used to maintain diversity. Li and Du [65] proposed a boundary simulation method to address inequality constraints for GA. Liang et al. [66] proposed a genetic algorithm which searches the decision space through the arithmetic crossover, and performs a selection on feasible and infeasible populations according to fitness and constraint violations, respectively. Garcia et al. [67] presented a constraint handling technique called the Multiple Constraint Ranking. Chootinan and Chen [68] proposed a repair procedure, and used gradient information derived from the constraint set to repair infeasible solutions.

This paper separated the objective and constraints without considering penalty function as was done by Deb [64], which can avoid solving the fitness of individuals. Individuals were directly compared by objectives and constraint violation.

For multi-constrained optimization problems, each constraint was normalized to eliminate the difference of orders of magnitude between different constraints, so each constraint contributes similarly to the constraint violation. The original constraints were converted to Equation (4):

$$\begin{cases} \frac{g_i(\mathbf{x})}{G_i} \le 0, & i = 1,\ 2,\ \ldots,\ m \\ \frac{h_i(\mathbf{x})}{H_i} = 0, & i = m+1,\ m+2,\ \ldots,\ n \end{cases} \tag{4}$$

where G_i and H_i are larger values corresponding to the ith constraint.

2.3. Selection, Recombination and Mutation Operaters

Corresponding to separating the objective and constraints, a tournament selection mechanism [64] was adopted in the paper which selects individuals by objective and constraint violation and avoids introducing penalty factors and calculating fitness. For the two individuals selected each time, (a) if they are feasible, the one with the smaller

objective wins; (b) if they are infeasible, the one with the smaller constraint violation wins, and (c) if one is feasible and one is infeasible, the feasible one wins.

Each individual was real-value encoded in this paper, and simulated binary crossover (SBX) and polynomial mutation (PM) [69] were adopted as shown in Equations (5) and (6):

$$\begin{cases} \widetilde{x}_{1j} = 0.5 \times \left[(1+\gamma_j) x_{1j} + (1-\gamma_j) x_{2j} \right] \\ \widetilde{x}_{2j} = 0.5 \times \left[(1-\gamma_j) x_{1j} + (1+\gamma_j) x_{2j} \right] \end{cases}$$
$$\gamma_j = \begin{cases} (2\mu_j)^{\frac{1}{\eta+1}} & \mu_j < 0.5 \\ \left[\frac{1}{2(1-\mu_j)} \right]^{\frac{1}{\eta+1}} & else \end{cases} \quad (5)$$

where j is the jth component of the variable, $\widetilde{x}_{1j}$ and $\widetilde{x}_{2j}$ are the new individuals after SBX, x_{1j} and x_{2j} are the old individuals, γ_j is determined by the mathematical distribution, μ_j is a distribution index, and $\mu_j \in (0, 1)$ is suggested as 1 by Deb and Agrawal [69]:

$$\widetilde{\widetilde{x}}_{1j} = x_{1j} + \Delta_j$$
$$\Delta_j = \begin{cases} (2\mu_j)^{\frac{1}{\eta+1}} - 1 & \mu_j < 0.5 \\ 1 - \left[2(1-\mu_j) \right]^{\frac{1}{\eta+1}} & else \end{cases} \quad (6)$$

where $\widetilde{\widetilde{x}}_{1j}$ is the new individual after PM, Δ_j is determined by the mathematical distribution, and the other parameters are the same as above.

2.4. Advantages of IDPGA over SGA

Most metaheuristic algorithms need to achieve a balance between exploration and exploitation in the process of optimization. Simple genetic algorithms have a significant advantage in maintaining population diversity, but a single GA often cannot find an exact solution or even has difficulty in convergence.

In the process of GA optimization, the most critical steps are recombination and mutation, which will produce a large number of new individuals. Obviously, it is very important to balance the developing new individuals in a local region and detecting new individuals in a large region. In SGA, the two processes cannot be clearly differentiated, and the individuals are simply evaluated according to fitness. Once a relatively good individual is found, the population quickly gathers nearby to continue searching, which easily leads the algorithm to fall into local optimal or wrong solutions.

IDPGA has several advantages, as follows:

(1) The developing subpopulation has small probabilities of recombination and mutation, which means individuals do not readily participate in recombination and mutation. Once a relatively good individual is found, the algorithm will continue to search near this individual, and this process is called developing.

(2) The detecting subpopulation has high probabilities of recombination and mutation, which means individuals much more readily participate in recombination and mutation, and a large number of new individuals will be produced. This operation makes it possible for the population to search for better solutions in a larger range, and this advantage exists throughout the whole evolution.

(3) Immigration operators can break the balance of single population evolution. For example, the developing subpopulation is continuing to search around a relatively good individual A; at this time, the detecting subpopulation finds a better individual, B, in a large region. By exchanging A and B, the immigration operator will drag the developing subpopulation from A to B, making it continue to search around B, thus improving the solution accuracy and preventing the algorithm from falling into local optimum. In turn, once the developing subpopulation finds a better individual C, the immigration operator will drag the detecting subpopulation to C, making its search more purposeful, thus improving the optimization efficiency of the algorithm.

(4) The number of immigration operators adopts a dynamic random number, which is more in line with the natural law.

To sum up, IDPGA has significant advantages over SGA and is reliable and effective.

2.5. The Whole Process of the IDPGA

Step 1: input arguments of problems and algorithm.

Step 2: initialize two subpopulations; calculate objective and constraint violation.

Step 3: take out and save the best individual of each subpopulation.

Step 4: judge whether the termination condition is met. If YES, go to Step 10, if NO, go to Step 5.

Step 5: generate the number of immigrants.

Step 6: in each subpopulation, remaining individuals of each subpopulation take part in tournament selection, SBX and PM; calculate the objective and constraint violation of these new individuals; remove the worst individual of each new subpopulation; select the immigrants randomly from the new generation.

Step 7: exchange the immigrants and the best individuals preserved earlier between two subpopulations.

Step 8: repeat Step 3.

Step 9: go to Step 4 and loop.

Step 10: output the best individual.

The complete process of IDPGA is shown in Figure 1.

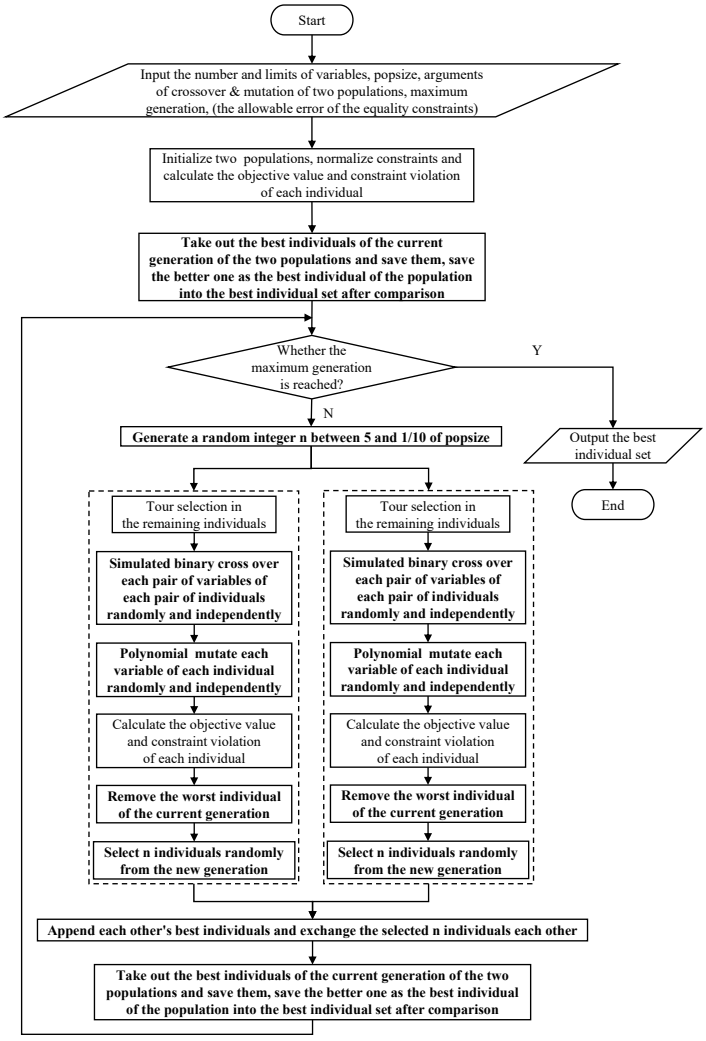


Figure 1. The flowchart of IDPGA.

3. Mathematical Tests and Results

Engineering optimization problems are always constrained optimization problems, so this paper mainly checks the performance of IDPGA for solving constrained optimization problems. As standard mathematical benchmarks are often more complex than actual engineering optimization problems, 14 classical nonlinear optimization benchmarks with multi constraints were utilized to evaluate the IDPGA. All of the scripts were written in Python, and all tests were done on an Intel Core i7-11370H 3.30 GHz machine with 16 GB RAM and single thread under Win10 platform. The timing rule of the script is that it starts when the main program runs, in other words, when the population is randomly initialized. It ends when all generations have finished evolution and the best individual has been saved.

3.1. Benchmarks

The 14 mathematical benchmarks with equality or inequality constraints were selected from the CEC 2006 [70] and marked as f1 to f14. The mathematical description and optimums are shown in Table 1. The number of constraints and the scales of decision variables were different, and all problems were requested to find the minimum.

Table 1. The mathematical expressions of 14 benchmarks.

Benchmark	Mathematical Form & Optimum
f1	$\min f(\mathbf{x}) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i$ $s.t.\begin{cases} g_1(\mathbf{x}) = 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \le 0 \\ g_2(\mathbf{x}) = 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \le 0 \\ g_3(\mathbf{x}) = 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \le 0 \\ g_4(\mathbf{x}) = -8x_1 + x_{10} \le 0 \\ g_5(\mathbf{x}) = -8x_2 + x_{11} \le 0 \\ g_6(\mathbf{x}) = -8x_3 + x_{12} \le 0 \\ g_7(\mathbf{x}) = -2x_4 - x_5 + x_{10} \le 0 \\ g_8(\mathbf{x}) = -2x_6 - x_7 + x_{11} \le 0 \\ g_9(\mathbf{x}) = -2x_8 - x_9 + x_{12} \le 0 \\ 0 \le x_i \le 1 \quad (i = 1,\ 2,\ \ldots,\ 9) \\ 0 \le x_i \le 100 \quad (i = 10,\ 11,\ 12) \\ 0 \le x_{13} \le 1 \end{cases}$ $f_{opt}(1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 1,\ 3,\ 3,\ 3,\ 1) = -15$
f2	$\min f(\mathbf{x}) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141$ $s.t.\begin{cases} g_1(\mathbf{x}) = 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 - 92 \le 0 \\ g_2(\mathbf{x}) = -85.334407 - 0.0056858x_2x_5 - 0.0006262x_1x_4 + 0.0022053x_3x_5 \le 0 \\ g_3(\mathbf{x}) = 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 - 110 \le 0 \\ g_4(\mathbf{x}) = -80.51249 - 0.0071317x_2x_5 - 0.0029955x_1x_2 - 0.0021813x_3^2 + 90 \le 0 \\ g_5(\mathbf{x}) = 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 - 25 \le 0 \\ g_6(\mathbf{x}) = -9.300961 - 0.0047026x_3x_5 - 0.0012547x_1x_3 - 0.0019085x_3x_4 + 20 \le 0 \\ 78 \le x_1 \le 102 \\ 33 \le x_2 \le 45 \\ 27 \le x_i \le 45\ (i = 3,\ 4,\ 5) \end{cases}$ $f_{opt}(78,\ 33,\ 29.9953,\ 45,\ 36.7758) = -30665.5387$
f3	$\min f(\mathbf{x}) = 3x_1 + 0.000001x_1^3 + 2x_2 + \frac{0.000002x_2^3}{3}$ $s.t.\begin{cases} g_1(\mathbf{x}) = x_3 - x_4 - 0.55 \le 0 \\ g_2(\mathbf{x}) = -x_3 + x_4 - 0.55 \le 0 \\ h_1(\mathbf{x}) = 1000\sin(-x_3 - 0.25) + 1000\sin(-x_4 - 0.25) + 894.8 - x_1 = 0 \\ h_2(\mathbf{x}) = 1000\sin(x_3 - 0.25) + 1000\sin(x_3 - x_4 - 0.25) + 894.8 - x_2 = 0 \\ h_3(\mathbf{x}) = 1000\sin(x_4 - 0.25) + 1000\sin(-x_3 + x_4 - 0.25) + 1294.8 = 0 \\ 0 \le x_i \le 1200\ (i = 1,\ 2) \\ -0.55 \le x_i \le 0.55\ (i = 3,\ 4) \end{cases}$ $f_{opt}(679.9451,\ 1026.0670,\ 0.1189,\ -0.3962) = 5126.4967$

Table 1. *Cont.*

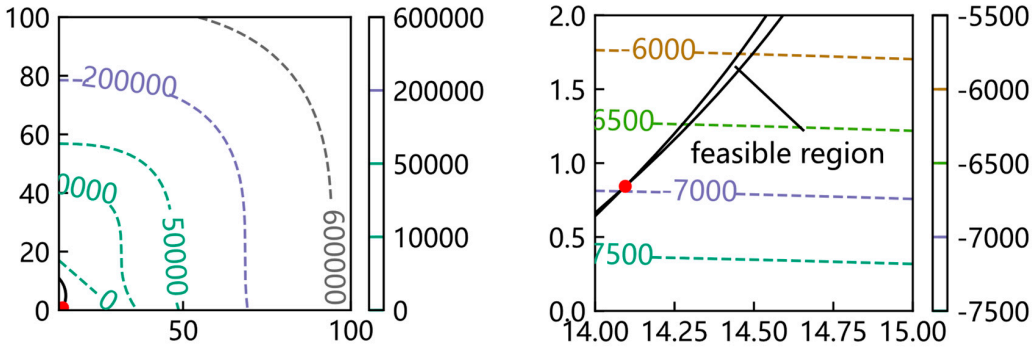
Benchmark	Mathematical Form & Optimum
f4	$\min f(\mathbf{x}) = (x_1-10)^3 + (x_2-20)^3$ $s.t.\begin{cases} g_1(\mathbf{x}) = -(x_1-5)^2 - (x_2-5)^2 + 100 \le 0 \\ g_2(\mathbf{x}) = (x_1-6)^2 + (x_2-5)^2 - 82.81 \le 0 \\ 13 \le x_1 \le 100 \\ 0 \le x_2 \le 100 \end{cases}$ $f_{opt}(14.0950,\ 0.8430) = -6961.8139$
f5	$\min f(\mathbf{x}) = x_1^2 + x_2^2 + x_1x_2 - 14x_1 - 16x_2 + (x_3-10)^2 + 4(x_4-5)^2 + (x_5-3)^2$ $+2(x_6-1)^2 + 5x_7^2 + 7(x_8-11)^2 + 2(x_9-10)^2 + (x_{10}-7)^2 + 45$ $s.t.\begin{cases} g_1(\mathbf{x}) = -105 + 4x_1 + 5x_2 - 3x_7 + 9x_8 \le 0 \\ g_2(\mathbf{x}) = 10x_1 - 8x_2 - 17x_7 + 2x_8 \le 0 \\ g_3(\mathbf{x}) = -8x_1 + 2x_2 + 5x_9 - 2x_{10} - 12 \le 0 \\ g_4(\mathbf{x}) = 3(x_1-2)^2 + 4(x_2-3)^2 + 2x_3^2 - 7x_4 - 120 \le 0 \\ g_5(\mathbf{x}) = 5x_1^2 + 8x_2 + (x_3-6)^2 - 2x_4 - 40 \le 0 \\ g_6(\mathbf{x}) = x_1^2 + 2(x_2-2)^2 - 2x_1x_2 + 14x_5 - 6x_6 \le 0 \\ g_7(\mathbf{x}) = 0.5(x_1-8)^2 + 2(x_2-4)^2 + 3x_5^2 - x_6 - 30 \le 0 \\ g_8(\mathbf{x}) = -3x_1 + 6x_2 + 12(x_9-8)^2 - 7x_{10} \le 0 \\ -10 \le x_i \le 10\ (i = 1,\ 2,\ \ldots,\ 10) \end{cases}$ $f_{opt}\begin{pmatrix} 2.1720,\ 2.3637,\ 8.7739,\ 5.0960,\ 0.9907, \\ 1.4306, 1.3216,\ 9.8287,\ 8.2801,\ 8.3759 \end{pmatrix} = 24.3062$
f6	$\min f(\mathbf{x}) = -\frac{\sin^3(2\pi x_1)\sin(2\pi x_2)}{x_1^3(x_1+x_2)}$ $s.t.\begin{cases} g_1(\mathbf{x}) = x_1^2 - x_2 + 1 \le 0 \\ g_2(\mathbf{x}) = -x_1 + (x_2-4)^2 + 1 \le 0 \\ 0 \le x_i \le 10\ (i = 1,\ 2) \end{cases}$ $f_{opt}(1.2280,\ 4.2454) = -0.0958$
f7	$\min f(\mathbf{x}) = (x_1-10)^2 + 5(x_2-12)^2 + x_3^4 + 3(x_4-11)^2 + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7$ $s.t.\begin{cases} g_1(\mathbf{x}) = -127 + 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 \le 0 \\ g_2(\mathbf{x}) = -282 + 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 \le 0 \\ g_3(\mathbf{x}) = -196 + 23x_1 + x_2^2 + 6x_6^2 - 8x_7 \le 0 \\ g_4(\mathbf{x}) = 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \le 0 \\ -10 \le x_i \le 10\ (i = 1,\ 2,\ \ldots,\ 7) \end{cases}$ $f_{opt}(2.3305,\ 1.9514,\ -0.4775,\ 4.3657,\ -0.6245,\ 1.0381,\ 1.5942) = 680.6301$
f8	$\min f(\mathbf{x}) = e^{x_1x_2x_3x_4x_5}$ $s.t.\begin{cases} h_1(\mathbf{x}) = \sum_{i=1}^{5} x_i^2 - 10 = 0 \\ h_2(\mathbf{x}) = x_2x_3 - 5x_4x_5 = 0 \\ h_3(\mathbf{x}) = x_1^3 + x_2^3 + 1 = 0 \\ -2.3 \le x_i \le 2.3, \quad i = 1,\ 2 \\ -3.2 \le x_i \le 3.2, \quad i = 3,\ 4,\ 5 \end{cases}$ $f_{opt}(-1.7171,\ 1.5957,\ 1.8273,\ -0.7637,\ -0.7637) = 0.0539$
f9	$\min f(\mathbf{x}) = \sum_{i=1}^{10} x_i \left(c_i + \ln \frac{x_i}{\sum_{j=1}^{10} x_j} \right)$ $s.t.\begin{cases} h_1(\mathbf{x}) = x_1 + 2x_2 + 2x_3 + x_6 + x_{10} - 2 = 0 \\ h_2(\mathbf{x}) = x_4 + 2x_5 + x_6 + x_7 - 1 = 0 \\ h_3(\mathbf{x}) = x_3 + x_7 + x_8 + 2x_9 + x_{10} - 1 = 0 \\ 0 < x_i \le 10 \quad (i = 1,\ 2, \ldots, 10) \end{cases}$ $c_1 = -6.089, c_2 = -17.164, c_3 = -34.054, c_4 = -5.914,$ $c_5 = -24.721, c_6 = -14.986, c_7 = -24.1, c_8 = -10.708,$ $c_9 = -26.662, c_{10} = -22.179$ $f_{opt}\begin{pmatrix} 0.0407, 0.1477, 0.7832, 0.0014, 0.4853, \\ 0.0007, 0.0274, 0.0180, 0.0373, 0.0969 \end{pmatrix} = -47.7649$

Table 1. *Cont.*

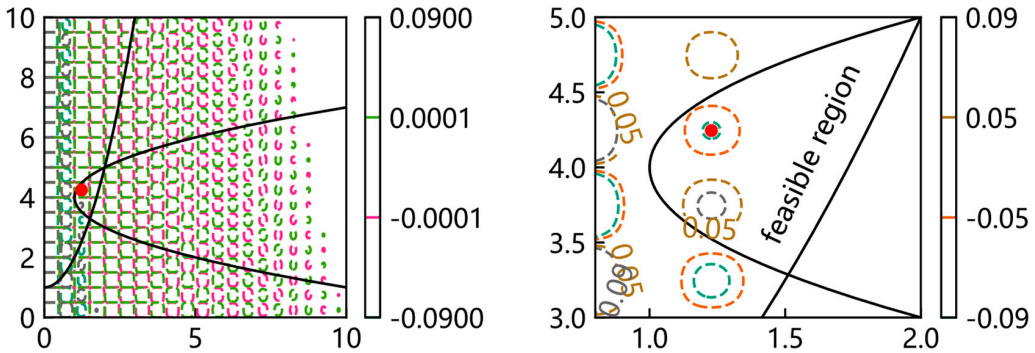
Benchmark	Mathematical Form & Optimum
f10	$\min f(\mathbf{x}) = 1000 - x_1^2 - 2x_2^2 - x_3^2 - x_1x_2 - x_1x_3$ $s.t. \begin{cases} h_1(\mathbf{x}) = x_1^2 + x_2^2 + +x_3^2 - 25 = 0 \\ h_2(\mathbf{x}) = 8x_1 + 14x_2 + 7x_3 - 56 = 0 \\ 0 \le x_i \le 10 \quad (i = 1,\ 2, 3) \end{cases}$ $f_{opt}(3.5121, 0.2170, 3.5522) = 961.7150$
f11	$\min f(\mathbf{x}) = -0.5(x_1x_4 - x_2x_3 + x_3x_9 - x_5x_9 + x_5x_8 - x_6x_7)$ $s.t. \begin{cases} g_1(\mathbf{x}) = x_3^2 + x_4^2 - 1 \le 0 \\ g_2(\mathbf{x}) = x_9^2 - 1 \le 0 \\ g_3(\mathbf{x}) = x_5^2 + x_6^2 - 1 \le 0 \\ g_4(\mathbf{x}) = x_1^2 + (x_2 - x_9)^2 - 1 \le 0 \\ g_5(\mathbf{x}) = (x_1 - x_5)^2 + (x_2 - x_6)^2 - 1 \le 0 \\ g_6(\mathbf{x}) = (x_1 - x_7)^2 + (x_2 - x_8)^2 - 1 \le 0 \\ g_7(\mathbf{x}) = (x_3 - x_5)^2 + (x_4 - x_6)^2 - 1 \le 0 \\ g_8(\mathbf{x}) = (x_3 - x_7)^2 + (x_4 - x_8)^2 - 1 \le 0 \\ g_9(\mathbf{x}) = x_7^2 + (x_8 - x_9)^2 - 1 \le 0 \\ g_{10}(\mathbf{x}) = x_2x_3 - x_1x_4 \le 0 \\ g_{11}(\mathbf{x}) = -x_3x_9 \le 0 \\ g_{12}(\mathbf{x}) = x_5x_9 \le 0 \\ g_{13}(\mathbf{x}) = x_6x_7 - x_5x_8 \le 0 \\ -10 \le x_i \le 10\ (i = 1,\ 2, \ldots, 8) \\ 0 \le x_9 \le 20 \end{cases}$ $f_{opt}\begin{pmatrix} -0.6578, -0.1534, 0.3234, -0.9463, -0.6578, \\ -0.7532, 0.3234, -0.3465, 0.5998 \end{pmatrix} = -0.8660$
f12	$\min f(\mathbf{x}) = x_1$ $s.t. \begin{cases} g_1(\mathbf{x}) = -x_1 + 35x_2^{0.6} + 35x_3^{0.6} \le 0 \\ h_1(\mathbf{x}) = -300x_3 + 7500x_5 - 7500x_6 - 25x_4x_5 + 25x_4x_6 + x_3x_4 = 0 \\ h_2(\mathbf{x}) = 100x_2 + 155.365x_4 + 2500x_7 - x_2x_4 - 25x_4x_7 - 15536.5 = 0 \\ h_3(\mathbf{x}) = -x_5 + \ln(-x_4 + 900) = 0 \\ h_4(\mathbf{x}) = -x_6 + \ln(x_4 + 300) = 0 \\ h_5(\mathbf{x}) = -x_7 + \ln(-2x_4 + 700) = 0 \\ 0 \le x_1 \le 1000 \\ 0 \le x_i \le 40\ (i = 2, 3) \\ 100 \le x_4 \le 300 \\ 6.3 \le x_5 \le 6.7 \\ 5.9 \le x_6 \le 6.4 \\ 4.5 \le x_7 \le 6.25 \end{cases}$ $f_{opt}\begin{pmatrix} 193.7245, 5.5694e-27, 17.3192, \\ 100.0479, 6.6845, 5.9917, 6.2145 \end{pmatrix} = 193.7245$
f13	$\min f(\mathbf{x}) = -x_1 - x_2$ $s.t. \begin{cases} g_1(\mathbf{x}) = -2x_1^4 + 8x_1^3 - 8x_1^2 + x_2 - 2 \le 0 \\ g_2(\mathbf{x}) = -4x_1^4 + 32x_1^3 - 88x_1^2 + 96x_1 + x_2 - 36 \le 0 \\ 0 \le x_1 \le 3 \\ 0 \le x_2 \le 4 \end{cases}$ $f_{opt}(2.3295, 3.1785) = -5.5080$
f14	$\min f(\mathbf{x}) = \left(x_1^2 + x_2 - 11\right)^2 + \left(x_1 + x_2^2 - 7\right)^2$ $s.t. \begin{cases} g_1(\mathbf{x}) = -4.84 + (x_1 - 0.05)^2 + (x_2 - 2.5)^2 \le 0 \\ g_2(\mathbf{x}) = 4.84 - x_1^2 - (x_2 - 2.5)^2 \le 0 \\ 0 \le x_i \le 6\ (i = 1,\ 2) \end{cases}$ $f_{opt}(2.2468,\ 2.3819) = 13.5909$

In order to visualize these problems, some 2D and 3D problems are selected, and the contour lines (surfaces) of objectives and feasible regions of decision variables of f4, f6, f10, f13 and f14 are plotted with Matplotlib and Mayavi in Figure 2; the optimum is marked with a red point. The feasible regions of f4 and f14 are narrow and long, the feasible region of f6 is small and the contour lines are dispersive, and the feasible region of f13 is

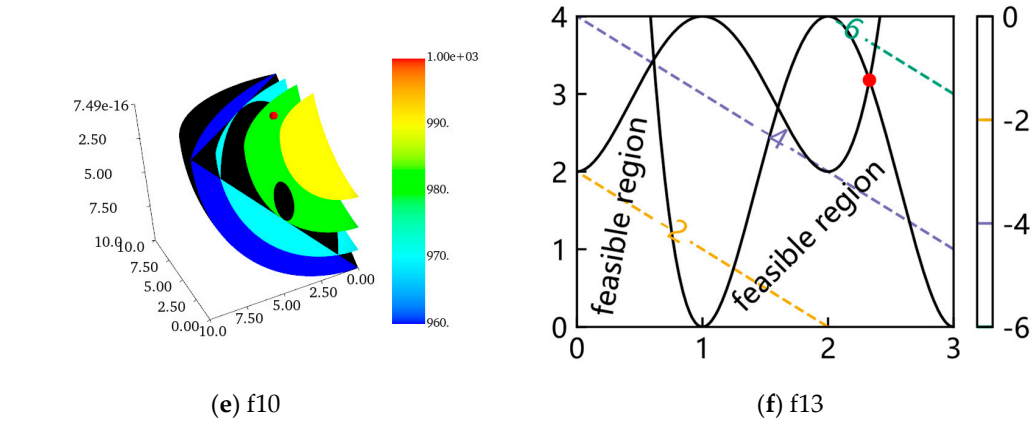
discontinuous. These images demonstrate that these mathematical problems are complex and that it is difficult to find the optimum.



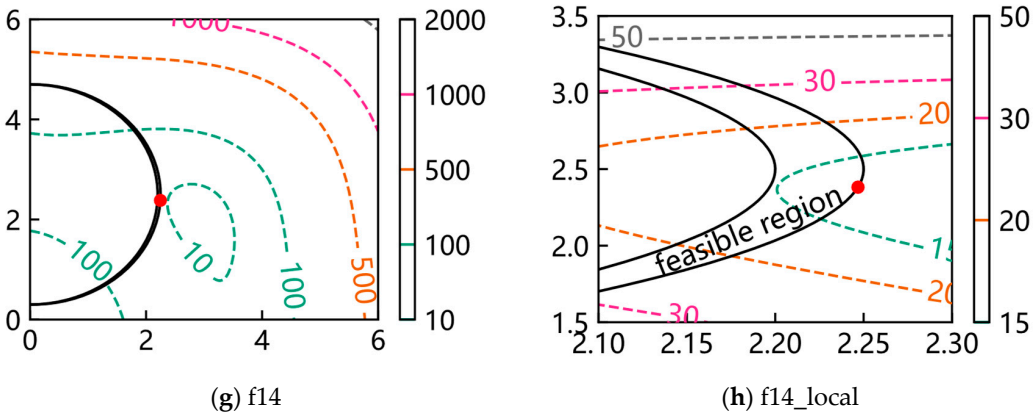
(**a**) f4 (**b**) f4_local



(**c**) f6 (**d**) f6_local



(**e**) f10 (**f**) f13



(**g**) f14 (**h**) f14_local

Figure 2. The contour lines or surfaces and feasible regions of several benchmarks.

3.2. Simulation Settings

After a series of initial tests, the appropriate popsize was taken as 100. The probabilities of recombination and mutation of a developing subpopulation were set as 0.3 and 0.001, and the probabilities of recombination and mutation of a detecting subpopulation were set as 0.9 and 0.05. The arguments η_c and η_m of a developing subpopulation were set as 1 and 5, and the arguments η_c and η_m of a detecting subpopulation were set as 1 and 100. The maximum number of generations was set as 300. The tolerance error ε of equality constraints was set as 0.001 for f3, f9, f10 and f12; meanwhile, the result with tolerance error of 0.001 for f8 was not very satisfactory after initial tests, so the tolerance error for f8 was set as 0.01. The stop criterion was that the maximum number of generations had been reached. All scripts were run 30 times consecutively, and the results were recorded.

3.3. Simulation Results

The best result of 30 runs is defined as follows: if there are feasible solutions, the one with the minimum objective wins; if not, the one with the minimum constraint violation wins. The worst result is similar to the above. The results and time consumption are shown in Tables 2 and 3, including the best, worst, mean, standard deviation (SD) of objectives, the minimum, maximum, mean of constraint violation, the minimum, maximum, mean of errors, and the minimum, maximum, mean of time consumption. It is worth mentioning that the constraint violation here is the sum of each constraint after normalization. Since the ultimate goal of this study is to apply IDPGA to engineering optimization problems, the results of mathematical problems are only compared with the theoretical solutions given by CEC 2006 [54].

Table 2. The objective and constraint violation of 30 runs.

Benchmark	Objective				Constraint Violation		
	Best	Worst	Mean	SD	Min.	Max.	Mean
f1	−14.9992	−12.0206	−14.6135	0.8740	0.000000	0.000000	0.000000
f2	−30,665.5252	−30,350.8881	−30,601.0550	78.3657	0.000000	0.000000	0.000000
f3	5084.7778	5068.7508	5390.8905	321.0657	0.000000	0.009556	0.000319
f4	−6961.7457	−6004.6365	−6668.3022	243.1782	0.008272	0.008958	0.008488
f5	24.4530	38.8329	26.2051	2.4831	0.000000	0.000000	0.000000
f6	−0.0958	−0.0958	−0.0958	0.0000	0.000000	0.000000	0.000000
f7	680.6596	683.2481	681.3567	0.6432	0.000000	0.000000	0.000000
f8	0.0359	0.7473	0.2504	0.1850	0.000000	0.000000	0.000000
f9	−49.4812	−42.2790	−45.6569	1.4903	0.000000	0.000000	0.000000
f10	961.2657	966.5413	962.4657	1.5194	0.000000	0.000000	0.000000
f11	−0.8649	−0.5847	−0.8298	0.0672	0.000000	0.000000	0.000000
f12	189.2239	319.6300	257.7680	45.8913	0.000000	0.000000	0.000000
f13	−5.5080	−5.5080	−5.5080	0.0000	0.000000	0.000000	0.000000
f14	13.5908	13.5912	13.5909	0.0001	0.000000	0.000000	0.000000

Table 3. The time consumption in 30 runs.

Benchmark	Time (s)			Benchmark	Time (s)		
	Min.	Max.	Mean		Min.	Max.	Mean
f1	1.16	1.44	1.23	f8	0.26	0.37	0.29
f2	0.75	1.07	0.83	f9	1.16	1.42	1.25
f3	0.25	0.34	0.28	f10	0.43	0.49	0.44
f4	0.32	0.42	0.33	f11	1.08	1.30	1.12
f5	1.24	2.09	1.44	f12	0.32	0.42	0.35
f6	0.36	0.61	0.44	f13	0.33	0.41	0.36
f7	0.87	1.04	0.92	f14	0.33	0.45	0.36

In general, the final solution found for other problems is a feasible solution with constraint violation of 0.000000 except for f4. In fact, the theoretical decision variable from CEC 2006 [54] is (14.0950, 0.8430), and the corresponding normalized constraint violation is 0.008271919, which illustrates that IDPGA is effective.

Compared with Table 1, the best solution obtained by IDPGA is close to or equal to the theoretical optimum. Sometimes, IDPGA found a smaller objective than the theoretical optimum for f3, f8, f9, f10, f12 and f14, but it is worth noting that the first five problems all contain equality constraints, and a relatively large tolerance error may cause better results. This tolerance error is very small compared to the objective and is acceptable from an engineer's point of view. The standard deviation of the objective is very small except for f2, f3, f4 and f12, for which IDPGA performs with relatively good robustness.

IDPGA takes less than 1 s to find the optimum for most complex problems, which demonstrates that IDPGA is very efficient. This result is due to the fact that IDPGA does not introduce a penalty function or calculate fitness of individuals.

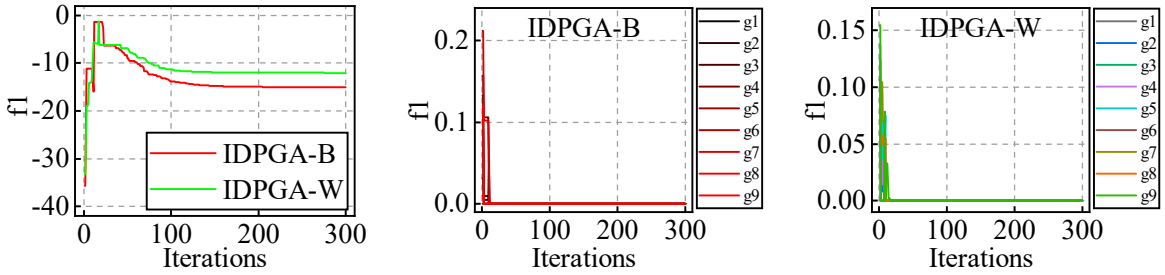
The decision variables corresponding to the best and worst results of 30 runs for each problem are presented in Table 4. These simulated results are very close to the theoretical solution for most problems, which proves that IDPGA is accurate and reliable. From the point of objectives and decision variables, IDPGA works well for most complex mathematical problems.

Table 4. Decision variables corresponding to the best and worst results.

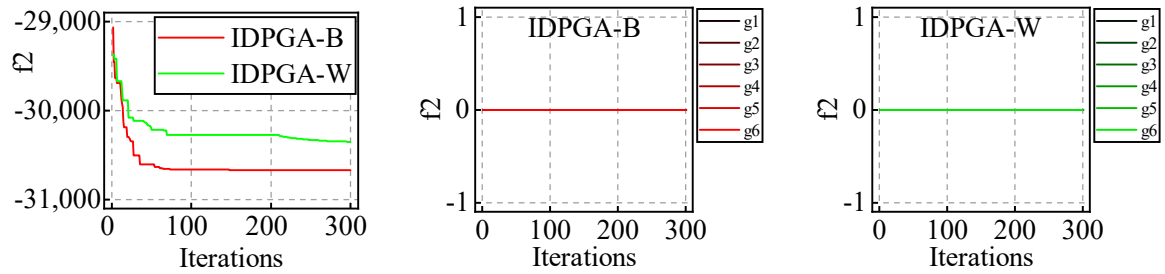
Benchmark	Theoretical	Best	Worst
f1	(1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 1)	(1, 1, 1, 1, 1, 1, 1, 1, 1, 2.9999, 2.9995, 2.9999, 1)	(0.0074, 1, 1, 0, 1, 1, 1, 1, 1, 0.0590, 2.9993, 2.9998, 0.9998)
f2	(78, 33, 29.9953, 45, 36.7758)	(78, 33, 29.9953, 45, 36.7756)	(78.0005, 34.2139, 31.7824, 44.9966, 32.5275)
f3	$\begin{pmatrix} 679.9451, & 1026.0670, \\ 0.1189, & -0.3962 \end{pmatrix}$	(672.1696, 1024.1822, 0.1209, −0.3933)	(515.8793, 1164.9384, 0.2227, −0.3273)
f4	(14.0950, 0.8430)	(14.0950, 0.8430)	(14.4780, 1.7340)
f5	$\begin{pmatrix} 2.1720, & 2.3637, & 8.7739, & 5.0960, \\ 0.9907, & 1.4306, & 1.3216, & 9.8287, \\ 8.2801, & 8.3759 \end{pmatrix}$	(2.1429, 2.4345, 8.7592, 5.0397, 0.9718, 1.3602, 1.2713, 9.7854, 8.2413, 8.4688)	(1.8368, 3.6136, 8.1787, 5.2668, 0.9142, 1.3528, 0.4458, 8.9912, 7.3098, 8.5425)
f6	(1.2280, 4.2454)	(1.2280, 4.2454)	(1.2280, 4.2454)
f7	$\begin{pmatrix} 2.3305, & 1.9514, & -0.4775, & 4.3657, \\ -0.6245, & 1.0381, & 1.5942 \end{pmatrix}$	(2.3110, 1.9561, −0.5165, 4.3633, −0.6493, 1.0501, 1.5833)	(2.2938, 2.0774, −0.5292, 4.0062, −0.6374, 1.0317, 1.5391)
f8	$\begin{pmatrix} -1.7171, & 1.5957, & 1.8273, \\ -0.7637, & -0.7637 \end{pmatrix}$	(−1.5706, 1.4986, 2.0115, 0.5973, 1.1767)	(−0.2512, −0.9421, −2.8240, −0.4852, −0.8985)
f9	$\begin{pmatrix} 0.0407, 0.1477, 0.7832, 0.0014, \\ 0.4853, 0.0007, 0.0274, 0.0180, \\ 0.0373, 0.0969 \end{pmatrix}$	(0.1940, 0.0986, 0.7763, 0.0562, 0.4481, 0.0385, 0.0578, 0.0079, 0.0632, 0.0875)	(1.1466, 0.1982, 0.0051, 0.2516, 0.1453, 0.4660, 0.0419, 0.0096, 0.4763, 0.0508)
f10	(3.5121, 0.2170, 3.5522)	(3.5468, 0.2146, 3.5601)	(4.8536, 0.8724, 0.7408)
f11	$\begin{pmatrix} -0.6578, -0.1534, 0.3234, \\ -0.9463, -0.6578, -0.7532, \\ 0.3234, -0.3465, 0.5998 \end{pmatrix}$	(−0.8311, −0.2700, 0.0890, −0.9953, −0.8176, −0.5733, 0.0659, −0.7113, 0.2859)	(0.5054, −0.1218, 0.7108, 0.6655, −0.2739, 0.4938, −0.2180, 0.3005, 0.7323)
f12	$\begin{pmatrix} 193.7245, 5.5694e-27, 17.3192, \\ 100.0479, 6.6845, 5.9917, 6.2145 \end{pmatrix}$	(189.2239, 7.8998×10^{-6}, 16.6459, 104.5904, 6.6785, 6.0026, 6.1963)	(319.6300, 39.8979, 5.7438×10^{-9}, 299.7904, 6.3971, 6.3966, 4.6108)
f13	(2.3295, 3.1785)	(2.3295, 3.1785)	(2.3295, 3.1785)
f14	(2.2468, 2.3819)	(2.2468, 2.3819)	(2.2467, 2.3790)

3.4. Evolution Process Analysis

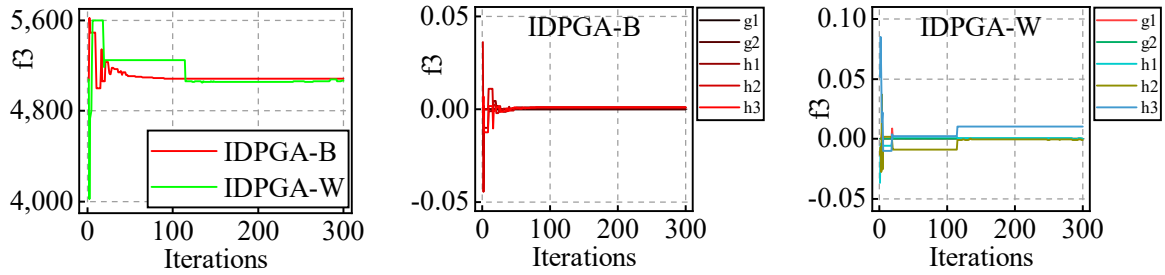
In order to display the searching process of the algorithm more intuitively, the whole evolution processes of the best and worst results of 30 runs are plotted as shown in Figure 3, IDPGA-B is the best result and IDPGA-W is the worst result.



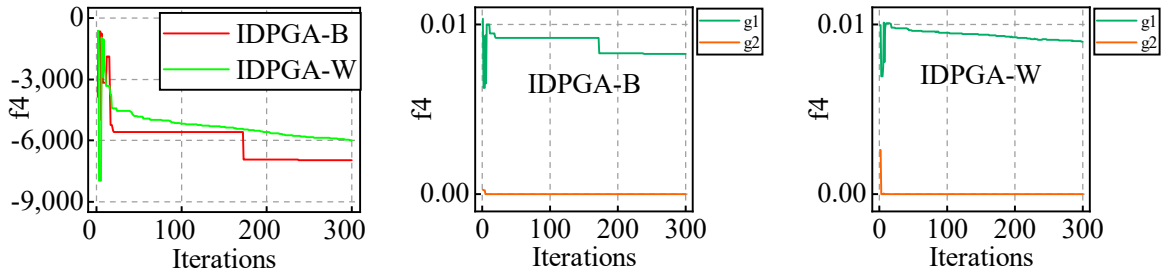
(**i-a**) objective of f1 (**i-b**) constraint violation of f1



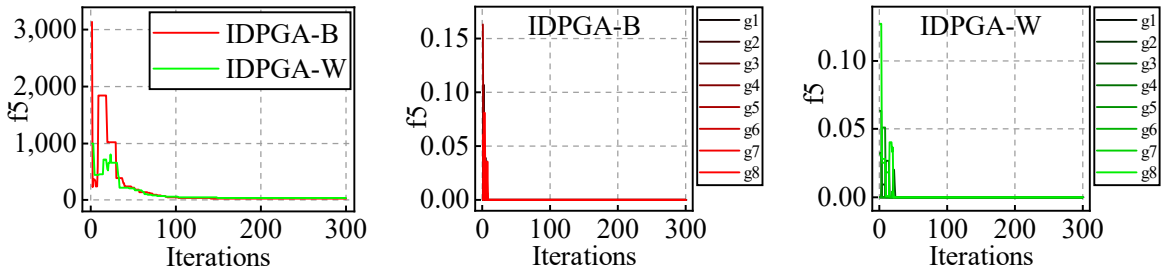
(**ii-a**) objective of f2 (**ii-b**) constraint violation of f2



(**iii-a**) objective of f3 (**iii-b**) constraint violation of f3

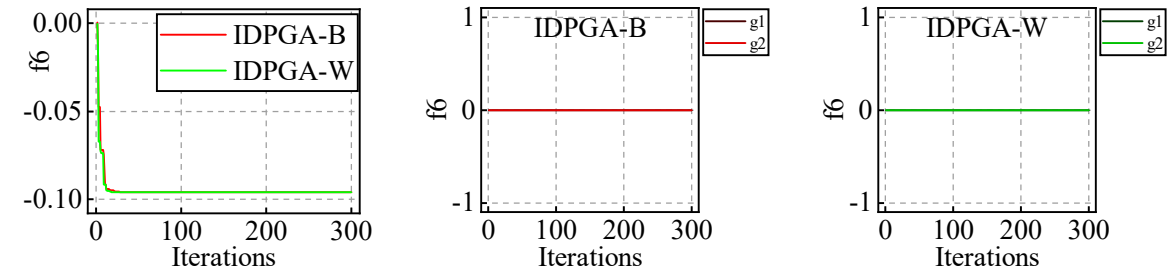


(**iv-a**) objective of f4 (**iv-b**) constraint violation of f4

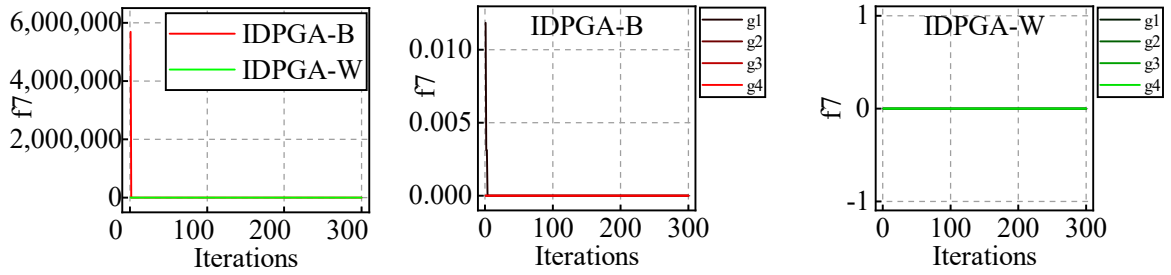


(**v-a**) objective of f5 (**v-b**) constraint violation of f5

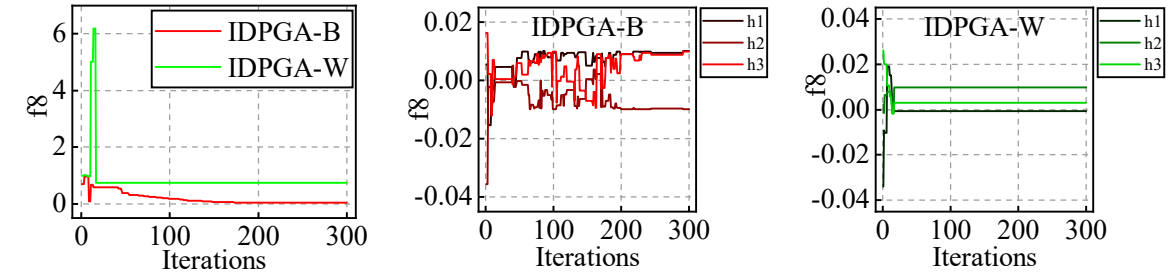
Figure 3. *Cont.*



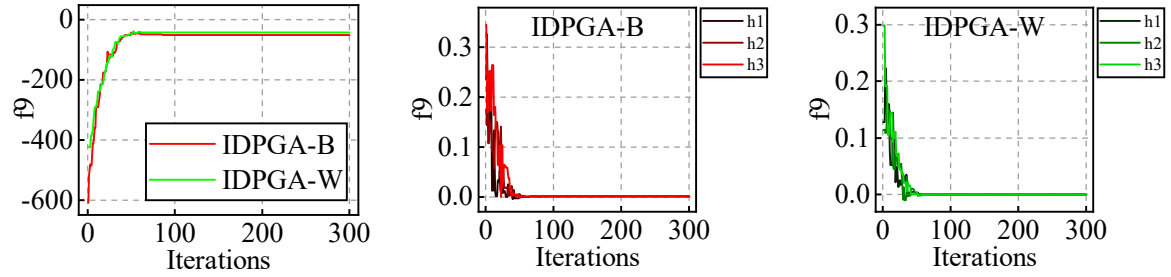
(**vi-a**) objective of f6 (**vi-b**) constraint violation of f6



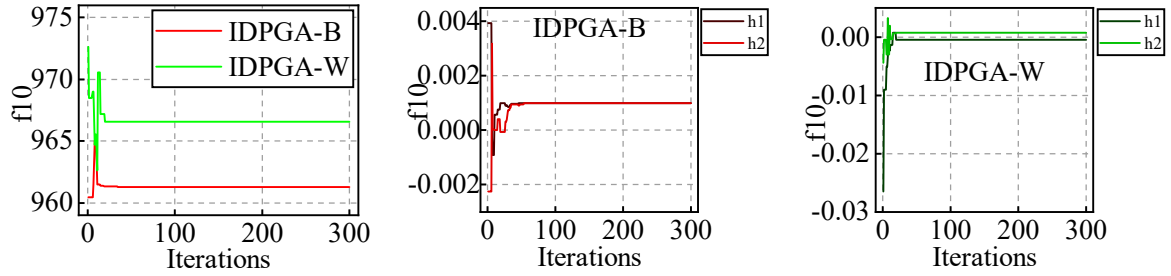
(**vii-a**) objective of f7 (**vii-b**) constraint violation of f7



(**viii-a**) objective of f8 (**viii-b**) constraint violation of f8



(**ix-a**) objective of f9 (**ix-b**) constraint violation of f9



(**x-a**) objective of f10 (**x-b**) constraint violation of f10

Figure 3. *Cont.*

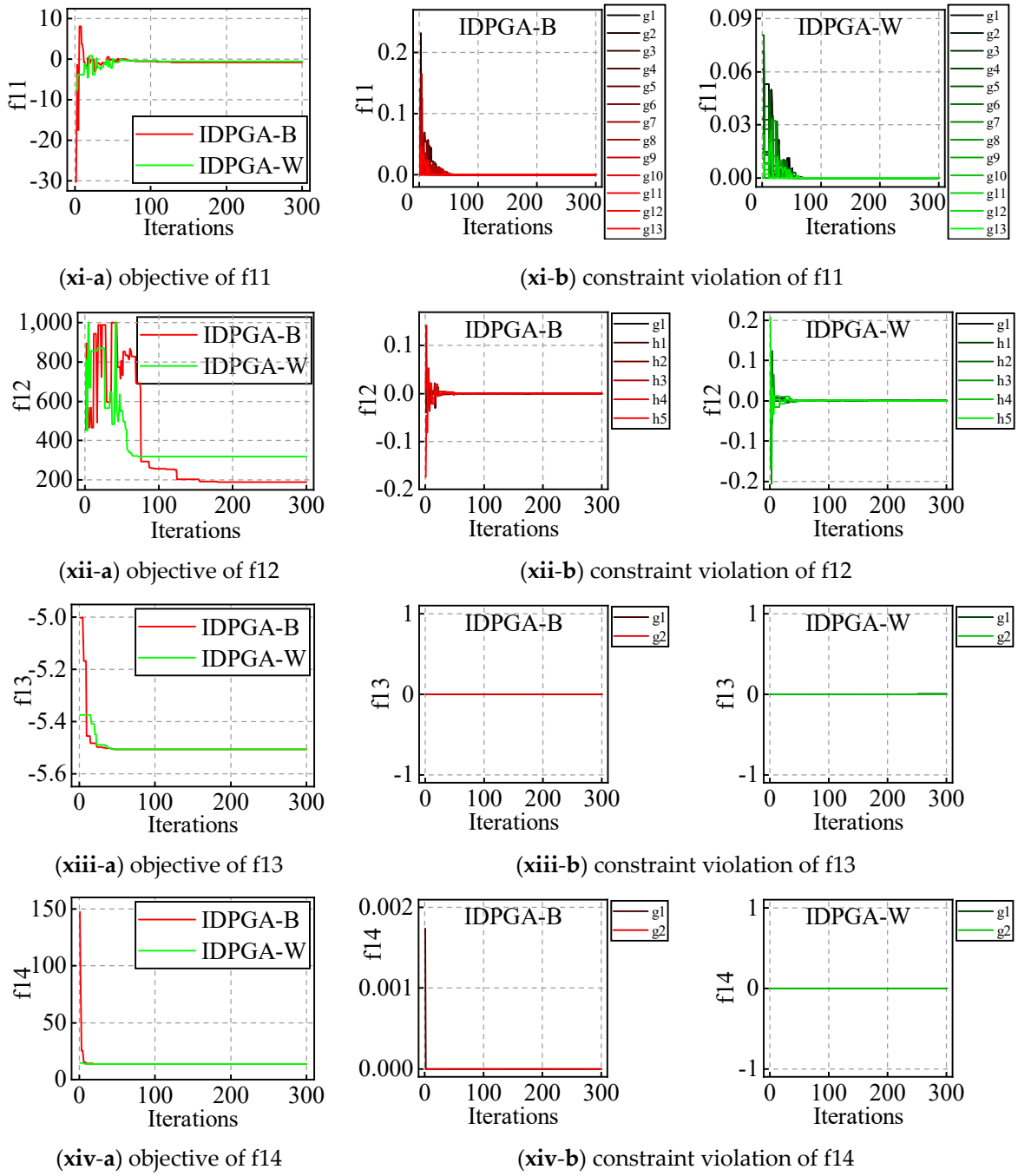


Figure 3. The evolution figures for f1 to f14.

For most problems, constraint violation approaches zero at the beginning of evolution, which shows that IDPGA has searched for a feasible solution very quickly. Similarly, the objective decreases very quickly at the beginning, and gradually tends to the optimum in one direction in the later stage, which again shows that IDPGA finds the optimum reliably and very quickly.

Moreover, combined with the evolution and constraint violation figures, it is evident that IDPGA has converged to an acceptable solution after 180 iterations. The whole evolution process is relatively stable, which indicates that the algorithm is reliable.

3.5. Errors of 30 Runs

In order to display the results of each run more visually, the errors of the simulated value relative to the theoretical value are calculated and shown in Figure 4. The green columns represent feasible solutions and the red columns represent infeasible solutions.

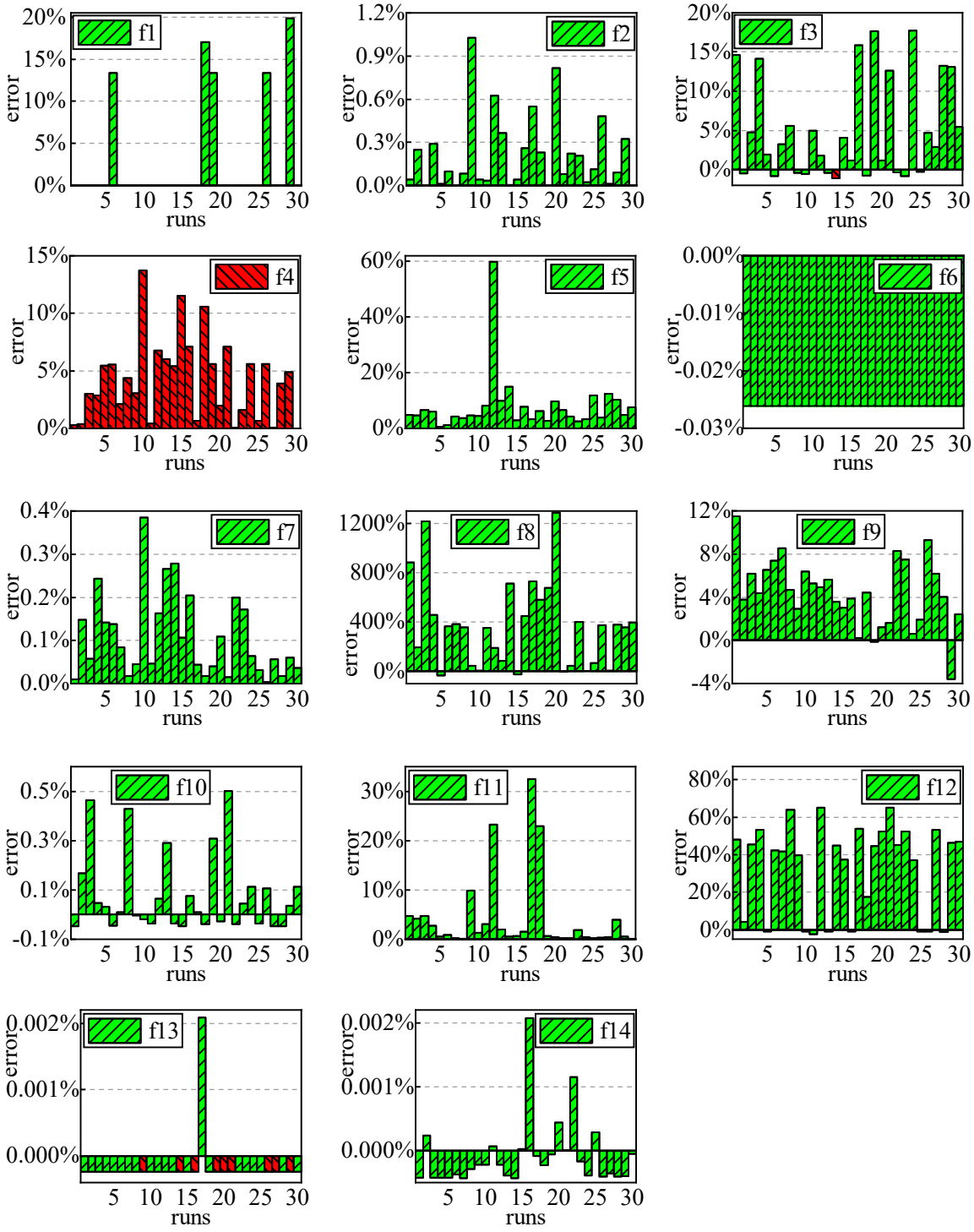


Figure 4. The bar charts of errors for f1 to f14.

From Figure 4, it can be seen that in the face of complex mathematical problems, a few results are unsatisfactory for f8 and f12, which indicates that IDPGA is not 100% reliable for more complex mathematical problems, but the negative errors in f3, f6, f8, f9, f10 and f12 demonstrate that the simulated values are smaller than the existing values; in other

words, the IDPGA finds some better results. Apart from these extreme results, IDPGA is able to obtain acceptable solutions in most cases, even better solutions sometimes, and all of the results are within 20% of the theoretical value, which is acceptable for engineers. Considering that actual engineering problems are often simpler than these benchmarks, IDPGA is still reliable.

4. Engineering Problems and Results

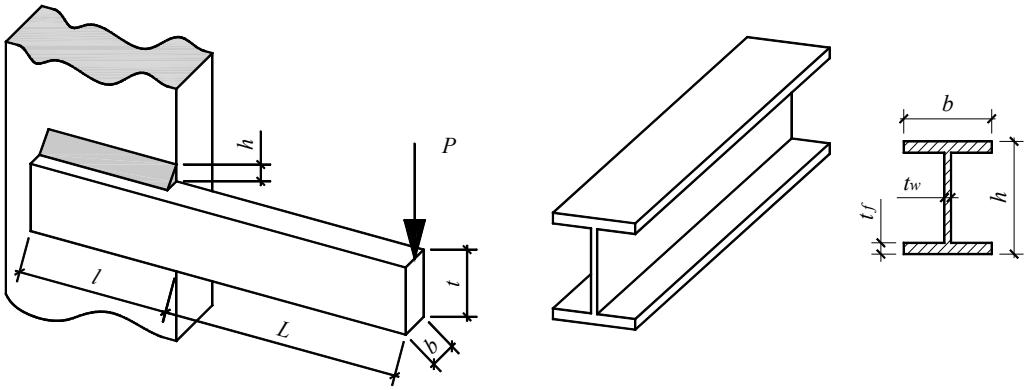
IDPGA behaved well for most of the complex mathematical problems. We next selected six well-known engineering optimization problems to verify the performance of the IDPGA in practical engineering applications: welded beam design, H-shaped steel beam design, three-bar truss design, tabular bar design, tension/compression spring design and Belleville spring design. These problems are small-scale engineering optimization problems, and they are representative of common problems. All arguments of scripts and computing platform are the same as above.

4.1. Description of Engineering Problems

Six engineering optimization problems are described as follows:

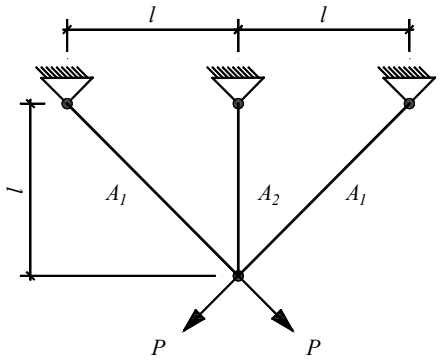
1. Welded beam design: the objective is cost, the constraints include shear stress, bending stress, buckling load, deflection and side constraints, four decision variables.
2. H-shaped steel beam design: the objective is the vertical deflection of the beam, the constraints include cross-sectional area and stress constraints, four decision variables.
3. Three-bar truss design: the objective is the volume of the truss, the constraints include stress constraints, two decision variables.
4. Tabular bar design: the objective is cost, the constraints include buckling stress and yield stress constraints, two decision variables.
5. Tension/compression spring design: the objective is weight, the constraints include deflection, shear stress, surge frequency constraints and outside diameter, three decision variables.
6. Belleville spring design: the objective is weight, the constraints include deflection, stress constraints, overall height and outside diameter, four decision variables.

Six engineering optimization problems are marked as f15 to f20 and the diagrams and mathematical expressions are shown in Figure 5 and Tables 5 and 6.

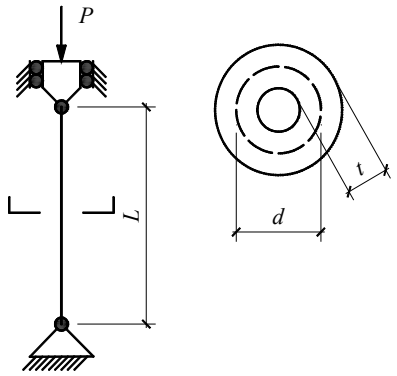


(**a**) Welded beam design (f15) (**b**) H-shaped steel beam design (f16)

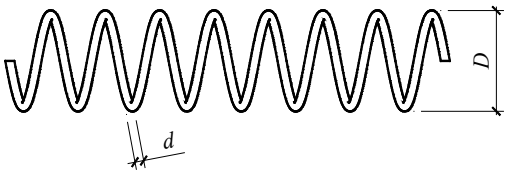
Figure 5. *Cont.*



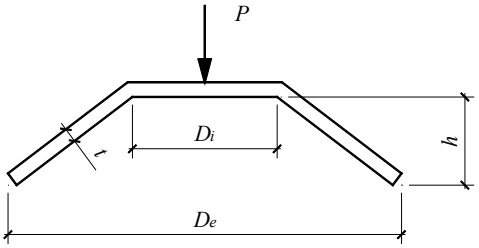
(**c**) Three-bar truss design (f17)



(**d**) Tabular bar design (f18)



(**e**) Tension/compression spring design (f19)



(**f**) Belleville spring design (f20)

Figure 5. Diagrams of six engineering problems.

Table 5. Mathematical expressions of six engineering problems.

Problems	Mathematical Form
f15	$(h,l,t,b) = (x_1,x_2,x_3,x_4)$ $\min f(\mathbf{x}) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14.0+x_2)$ $s.t.\begin{cases} g_1(\mathbf{x}) = \tau(\mathbf{x}) - \tau_{\max} \leqslant 0 \\ g_2(\mathbf{x}) = \sigma(\mathbf{x}) - \sigma_{\max} \leqslant 0 \\ g_3(\mathbf{x}) = \delta(\mathbf{x}) - \delta_{\max} \leqslant 0 \\ g_4(\mathbf{x}) = x_1 - x_4 \leqslant 0 \\ g_5(\mathbf{x}) = P - P_c(\mathbf{x}) \leqslant 0 \\ g_6(\mathbf{x}) = 0.125 - x_1 \leqslant 0 \\ g_7(\mathbf{x}) = 1.10471x_1^2 + 0.04811x_3x_4(14.0+x_2) - 5.0 \leqslant 0 \\ 0.1 \leq x_i \leq 2 \ (i = 1,4) \\ 0.1 \leq x_i \leq 10 \ (i = 2,3) \end{cases}$ $\tau(\mathbf{x}) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2}$ $\tau' = \frac{P}{\sqrt{2}x_1x_2}, \quad \tau'' = \frac{MR}{J}$ $M = P\left(L + \frac{x_2}{2}\right)$ $R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1+x_3}{2}\right)^2}$ $J = 2\sqrt{2}x_1x_2\left[\frac{x_2^2}{12} + \left(\frac{x_1+x_3}{2}\right)^2\right]$ $\sigma(\mathbf{x}) = \frac{6PL}{x_3^2x_4}, \quad \delta(\vec{x}) = \frac{6PL^3}{Ex_3^2x_4}$ $P_C(\mathbf{x}) = \frac{4.013E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right)$ $P = 6000\text{lb}, L = 14\text{in.},$ $\delta_{\max} = 0.25\text{in.}, E = 30 \times 10^6\text{psi},$ $G = 12 \times 10^6\text{psi}, \tau_{\max} = 13600\text{psi},$ $\sigma_{\max} = 30000\text{psi}$
f16	$(h,b,t_w,t_f) = (x_1,x_2,x_3,x_4)$ $\min f(x_1,x_2,x_3,x_4) = \frac{5000}{\frac{x_3(x_1-2x_4)^3}{12} + \frac{x_2x_4^3}{6} + 2x_2x_4\left(\frac{x_1-x_4}{2}\right)^2}$ $s.t.\begin{cases} g_1(\mathbf{x}) = 2x_2x_4 + x_3(x_1-2x_4) \leq 300 \\ g_2(\mathbf{x}) = \frac{18\times10^4x_1}{x_3(x_1-2x_4)^3 + 2x_2x_4(4x_4^2 + 3x_1(x_1-2x_4))} + \frac{15\times10^3x_2}{(x_1-2x_4)x_3^3 + 2x_4x_2^3} \leq 6 \\ 10 \leq x_1 \leq 80 \\ 10 \leq x_2 \leq 50 \\ 0.9 \leq x_3 \leq 5 \\ 0.9 \leq x_4 \leq 5 \end{cases}$	

Table 5. *Cont.*

Problems	Mathematical Form	
f17	$(A_1, A_2, A_3) = (x_1, x_2, x_3)$ $\min f(\mathbf{x}) = \left(2\sqrt{2}x_1 + x_2\right) \times l$ $s.t. \begin{cases} g_1(\mathbf{x}) = \frac{\sqrt{2}x_1 + x_2}{\sqrt{2}x_1^2 + 2x_1x_2} P - \sigma \leqslant 0 \\ g_2(\mathbf{x}) = \frac{x_2}{\sqrt{2}x_1^2 + 2x_1x_2} P - \sigma \leqslant 0 \\ g_3(\mathbf{x}) = \frac{1}{x_1 + \sqrt{2}x_2} P - \sigma \leqslant 0 \\ 0 \leq x_i \leq 1 \ (i = 1, \ 2) \end{cases}$	$l = 100\text{cm}$ $P = 2\ \text{kN/cm}^2$ $\sigma = 2\ \text{kN/cm}^2$
f18	$(d, t) = (x_1, x_2)$ $\min f(\mathbf{x}) = 9.8x_1x_2 + 2x_1$ $s.t. \begin{cases} g_1(\mathbf{x}) = \frac{P}{\pi x_1 x_2 \sigma_y} - 1 \leq 0 \\ g_2(\mathbf{x}) = \frac{8PL^2}{\pi^3 E x_1 x_2 \left(x_1^2 + x_2^2\right)} - 1 \leq 0 \\ g_3(\mathbf{x}) = \frac{2.0}{x_1} - 1 \leq 0 \\ g_4(\mathbf{x}) = \frac{x_1}{14} - 1 \leq 0 \\ g_5(\mathbf{x}) = \frac{0.2}{x_2} - 1 \leq 0 \\ g_6(\mathbf{x}) = \frac{x_2}{0.8} - 1 \leq 0 \end{cases}$	P= 2500 kgf $L = 250$ cm $E = 0.85 \times 10^6\ \text{kgf/cm}^2$ $\sigma_y = 500\ \text{kgf/cm}^2$ $2 \leq x_1 \leq 14$ $0.2 \leq x_2 \leq 0.8$
f19	$(d, D, N) = (x_1, x_2, x_3)$ $\min f(\mathbf{x}) = (x_3 + 2)x_2x_1^2$ $s.t. \begin{cases} g_1(\mathbf{x}) = 1 - \frac{x_2^3 x_3}{71785x_1^4} \leqslant 0 \\ g_2(\mathbf{x}) = \frac{4x_2^2 - x_1x_2}{12566\left(x_2x_1^3 - x_1^4\right)} + \frac{1}{5108x_1^2} - 1 \leq 0 \\ g_3(\mathbf{x}) = 1 - \frac{140.45x_1}{x_2^2x_3} \leqslant 0 \\ g_4(\mathbf{x}) = \frac{x_1 + x_2}{1.5} - 1 \leqslant 0 \\ 0.05 \leqslant x_1 \leqslant 2 \\ 0.25 \leqslant x_2 \leqslant 1.30 \\ 2.00 \leqslant x_3 \leqslant 15 \end{cases}$	
f20	$(t, h, D_i, D_e) = (x_1, x_2, x_3, x_4)$ $\min f(\mathbf{x}) = 0.07075\pi\left(x_4^2 - x_3^2\right)x_1$ $s.t. \begin{cases} g_1(\mathbf{x}) = -S + \frac{4E\delta_{\max}}{\left(1 - \mu^2\right)\alpha x_4^2}\left[\beta\left(x_2 - \frac{\delta_{\max}}{2}\right) + \gamma x_1\right] \leq 0 \\ g_2(\mathbf{x}) = -\frac{4E\delta_{\max}}{\left(1 - \mu^2\right)\alpha x_4^2}\left[\left(x_2 - \frac{\delta_{\max}}{2}\right)(x_2 - \delta_{\max})x_1 + x_1^3\right] + P_{\max} \leq 0 \\ g_3(\mathbf{x}) = -\delta_l + \delta_{\max} \leq 0 \\ g_4(\mathbf{x}) = -H + x_1 + x_2 \leq 0 \\ g_5(\mathbf{x}) = -D_{\max} + x_4 \leq 0, \\ g_6(\mathbf{x}) = -x_4 + x_3 \leq 0, \\ g_7(\mathbf{x}) = -0.3 + \frac{x_2}{x_4 - x_3} \leq 0 \\ 0.01 \leq x_1 \leq 6, \\ 0.05 \leq x_2 \leq 0.5 \\ 5 \leq x_3 \leq 15, \\ 5 \leq x_4 \leq 15 \end{cases}$	$K = x_4/x_3$ $\alpha = \frac{6}{\pi \ln K}\left(\frac{K-1}{K}\right)^2$ $\beta = \frac{6}{\pi \ln K}\left(\frac{K-1}{\ln K} - 1\right)$ $\gamma = \frac{6}{\pi \ln K}\left(\frac{K-1}{2}\right)$ $P_{\max} = 5400\text{lb}$ $\delta_{\max} = 0.2\text{in}$ $S = 200000\text{psi}$ $E = 30 \times 10^6\text{psi}$ $\mu = 0.3$ $H = 2\text{in}$ $D_{\max} = 12.01\text{in}$ $a = x_2/x_1$ $\delta_l = f(a)x_2$

Table 6. Variation of $f(a)$ with a about f20.

a	≤1.4	1.5	1.6	1.7	1.8	1.9	2	2.1
f(a)	1	0.85	0.77	0.71	0.66	0.63	0.6	0.58
a	2.2	2.3	2.4	2.5	2.6	2.7	≥2.8	
f(a)	0.56	0.55	0.53	0.52	0.51	0.51	0.5	

4.2. Simulation Results

The results of six engineering problems are shown in Tables 7 and 8. From the constraint violation, it can be seen that all of the solutions found by IDPGA are feasible, which demonstrates that IDPGA is effective and reliable. From the objective, the worst result is still satisfactory, except for f15 and f20, and the standard deviation is close to zero, which demonstrates that IDPGA performs with great robustness. From the time consumption, all problems are solved in 0.6 s, which demonstrates that IDPGA is efficient.

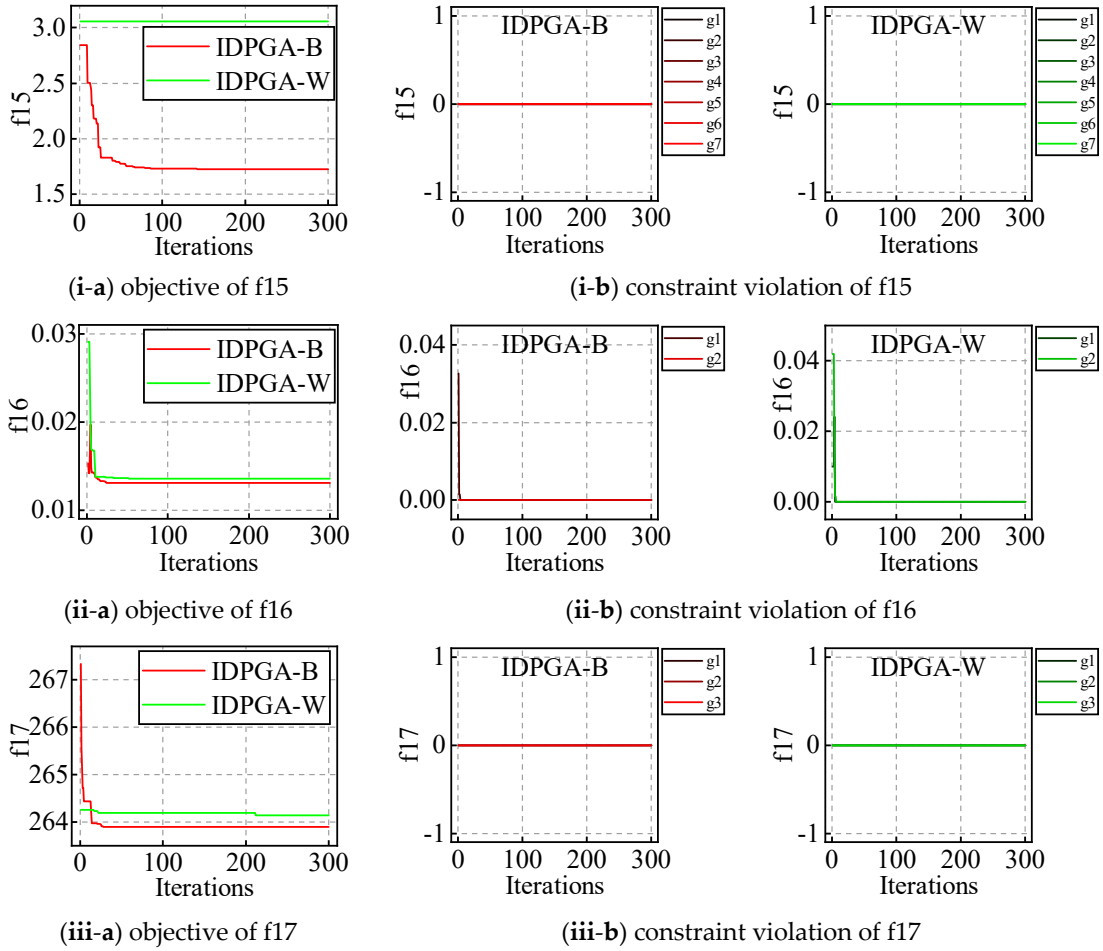
Table 7. The objectives and constraint violations of 30 runs.

Benchmark	Objective				Constraint Violation		
	Best	Worst	Mean	SD	Min.	Max.	Mean
f15	1.7249	3.0597	1.8996	0.2758	0.000000	0.000000	0.000000
f16	0.0131	0.0136	0.0132	0.0001	0.000000	0.000000	0.000000
f17	263.8959	264.1417	263.9386	0.0571	0.000000	0.000000	0.000000
f18	26.5313	26.5313	26.5313	0.0000	0.000000	0.000000	0.000000
f19	0.0127	0.0155	0.0134	0.0007	0.000000	0.000000	0.000000
f20	2.0004	2.5239	2.1438	0.1186	0.000000	0.000000	0.000000

Table 8. The time consumption of 30 runs.

Benchmark	Time (s)			Benchmark	Time (s)		
	Min.	Max.	Mean		Min.	Max.	Mean
f15	0.48	0.60	0.51	f18	0.20	0.23	0.20
f16	0.46	0.54	0.48	f19	0.22	0.26	0.23
f17	0.32	0.35	0.33	f20	0.29	0.34	0.31

The whole evolutionary processes are plotted in Figure 6. IDPGA searches for satisfactory solutions quickly after 100 iterations when solving engineering problems. The constraint violation of both the best and the worst solutions converges to zero at the beginning. Therefore, IDPGA takes only about 0.2 s to obtain a satisfactory solution. The whole search process is smooth and swift.



(**i-a**) objective of f15 (**i-b**) constraint violation of f15

(**ii-a**) objective of f16 (**ii-b**) constraint violation of f16

(**iii-a**) objective of f17 (**iii-b**) constraint violation of f17

Figure 6. *Cont.*

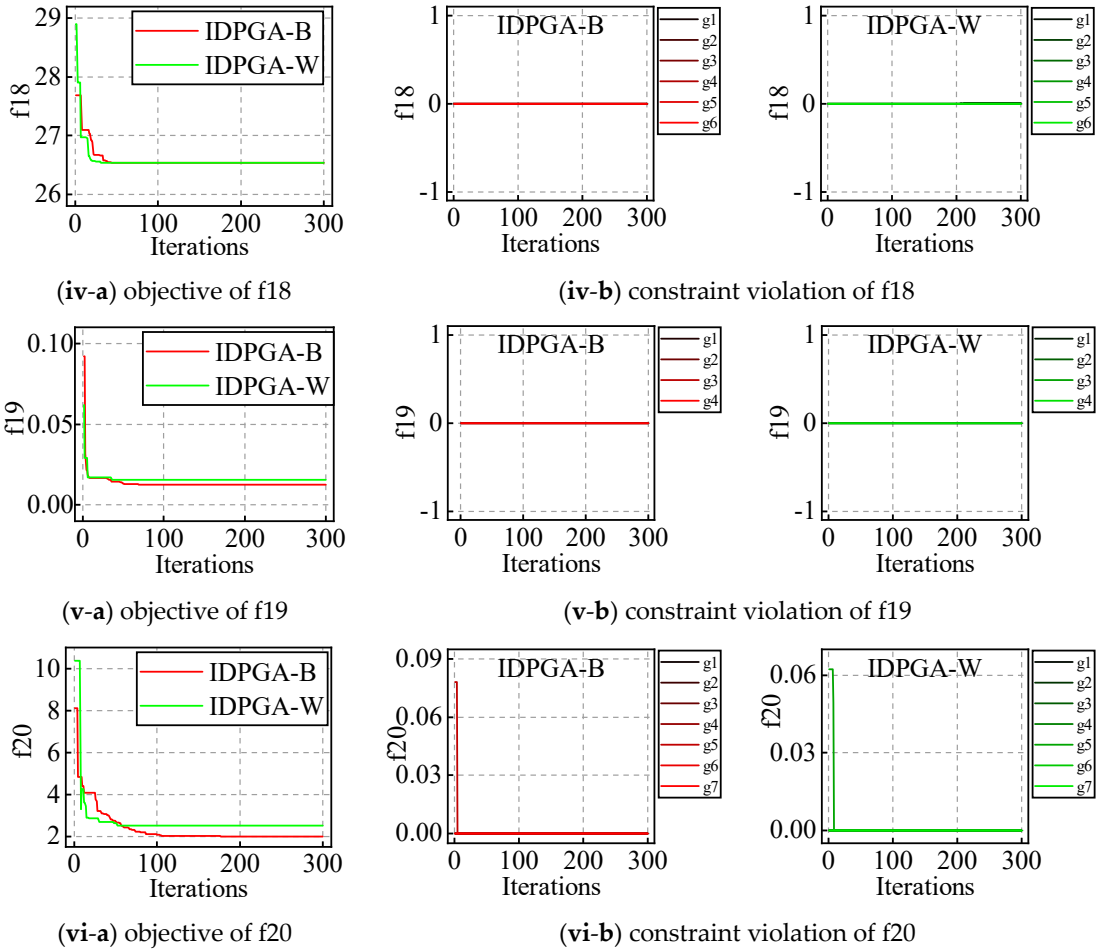


Figure 6. The evolution figures for f15 to f20.

Likewise, the errors of the simulated value relative to the theoretical value are calculated and shown in Figure 7. There seem to be some infeasible solutions of 30 runs for f18. In fact, the largest constraint violation of 30 runs is 1.15×10^{-14} after careful examination, which can be considered as zero in engineering and does not contradict the conclusions mentioned above. Meanwhile, it can be seen that unsatisfactory results account for a very small percentage, which fully proves that IDPGA is very reliable for solving small-scale engineering problems. The first four problems are common in civil engineering. from Figure 7, the results of 30 runs for solving f17 (three-bar truss design) and f18 (a compression tabular design) are very close, which demonstrates that IDPGA may be very stable and effective for solving the design problems of the truss system.

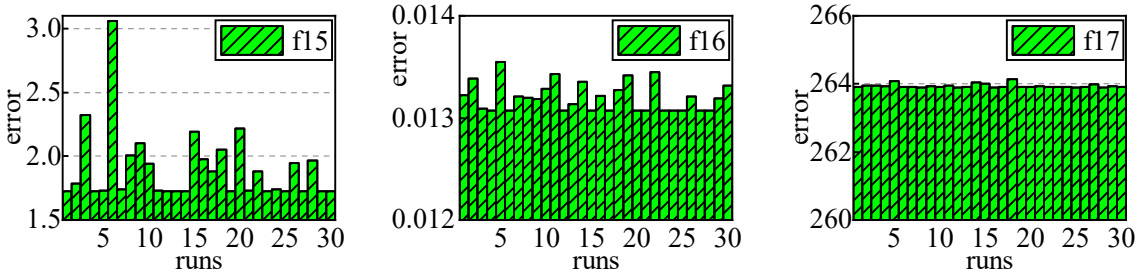


Figure 7. *Cont.*

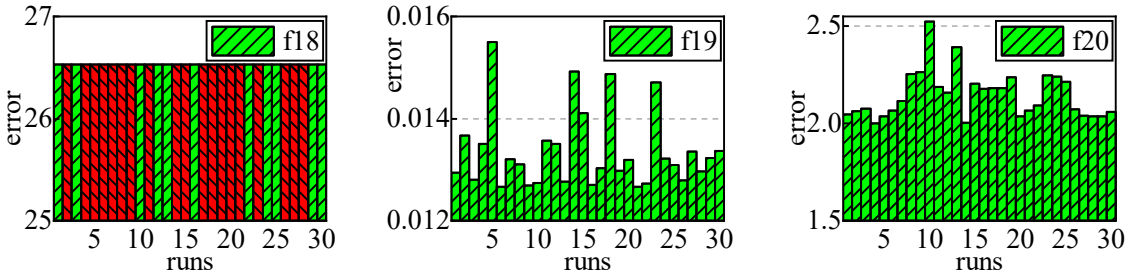


Figure 7. Bar charts of errors for f15 to f20.

4.3. Comparison with Other Algorithms

Finally, some existing algorithms were selected for comparison with IDPGA. These algorithms performed well in solving all or some of these six problems, including SC (society and civilization) [71], PSO-DE (a novel hybrid algorithm) [72], DEDS (differential evolution with dynamic stochastic selection for constrained optimization) [73], HEAA (constrained optimization based on hybrid evolutionary algorithm and adaptive constraint-handling technique) [74], CS (cuckoo search algorithm) [16], AHA (artificial hummingbird algorithm) [75], GA2 (using co-evolution to adapt the penalty factors of a fitness function incorporated in a genetic algorithm) [76], GA3 (a dominance-based selection scheme to incorporate constraints into the fitness function of a genetic algorithm) [77], CA (a cultural algorithm that uses domain knowledge) [78], CPSO (a co-evolutionary particle swarm optimization approach) [79], ABC (artificial bee colony) [80], GA5 (a new approach to handle constraints using evolutionary algorithms in which the new technique treats constraints as objectives and uses a multiobjective optimization approach to solve the re-stated single-objective optimization problem) [81], GeneAS I (genetic adaptive search I) [82], GeneAS II (genetic adaptive search II) [76], DMO (dwarf mongoose optimization algorithm) [83], AOA (the arithmetic optimization algorithm) [84], SSA (squirrel search algorithm) [85], SCA (sine cosine algorithm) [86], GWO (grey wolf optimizer) [87], CPSOGSA (hybrid constriction coefficient based PSO with gravitational search algorithm (GSA)) [83], Hsu and Liu's algorithm [88], Rao's algorithm [89], GSA-GA (a new hybrid GSA-GA algorithm) [90], AO (aquila optimizer) [12], GTO (giant trevally optimizer) [13] and TLCO (termite life cycle optimizer) [14]. The best simulation results of IDPGA and other results reported in the current literature are listed in Table 9.

Combining the results of mathematics benchmarks for six engineering problems, IDPGA obtained satisfactory solutions equal to or close to the existing optimums reported previously. For f15, IDPGA has a higher accuracy of 38% than SC. For other problems, the errors of IDPGA and the best value currently are within 2%. These results illustrate that IDPGA is reliable for solving engineering optimization problems. Similar to the above, IDPGA may be more suitable for solving the optimization problems of truss systems, which is the direction of further research.

For many population-based metaheuristic algorithms, parameter values will affect the performance of the algorithms, including the nature and diversity of the initial population, popsize and iterations [91,92]. Meanwhile, some of the algorithms cited in the paper reached similar conclusions. For example, the convergence of DEDS is related with the crossover rate parameter, and the performance of DMO is affected by popsize and iterations. Varying nutritious food resources and popsize can optimize the problem more accurately for SSA, but there is no rule of thumb for selection of nutritious food resources, and it depends on the nature of problem. In fact, tuning the parameters—finding the best parameter values for an algorithm—is a very specialized problem. Therefore, corresponding to the results in Table 9, according to the literature available, the main parameter values of most algorithms presented in the literature are listed in Table 10.

Table 9. Comparison of the best result with other algorithms.

Algorithm	f15	f16	f17	f18	f19	f20
IDPGA	1.7249	0.0131	263.8959	26.5313	0.0127	2.0004
SC	2.3854		263.8958		0.0127	
PSO-DE	1.7249		263.8958		0.0127	1.9883
DEDS	2.3810		263.8958		0.0127	
HEAA	2.3810		263.8958		0.0127	
CS		0.0131	263.9716	26.5322		
AHA	1.7249		263.8958		0.0127	1.9797
GA2	1.7483				0.0127	
GA3	1.7282				0.0127	
CA	1.7249				0.0127	
CPSO	1.7280				0.0127	
ABC	1.7249					1.9883
GA5						2.1220
GeneAS I						2.0181
GeneAS II						2.1626
DMO	1.6953	0.0131		24.6150		
AOA	2.1397	0.0131	263.9154	24.6170	0.0121	
SSA	1.7249	0.0131	263.8958	24.6150	0.0127	
SCA	1.7738	0.0131		24.6150		
GWO	1.7262	0.0131	263.8960	24.6150	0.0127	
CPSOGSA	1.6986	0.0131		24.6150		
Hsu and Liu				25.5316	0.0127	
Rao	2.3860			26.5323		
GSA-GA	1.6952		263.8958	26.5313	0.0127	1.9859
AO	1.6566		263.8684		0.0112	
GTO			263.8958			
TLCO	1.7249				0.0127	

Finally, a Friedman test [93] was adopted to further evaluate the performance of IDPGA. The Friedman test is a non-parametric test method that uses rank and rank sum to sort and compare samples. For the above six engineering problems, many existing algorithms only solve part of them. At the same time, for a certain algorithm, the results obtained after every run of a certain problem are rarely provided in the literature. Therefore, in the non-parametric test, we cannot select multiple results for a certain problem as the number of evaluators. Instead, we select the best simulation results for different problems for a certain algorithm in Table 9, and the number of these results is used as the number of evaluators. In addition, the number of algorithms is the number of samples.

To fully compare the algorithms, the algorithm that has solved as many problems as possible is selected. Firstly, the results of IDPGA, AOA, SSA and GWO are selected as shown in Table 11.

Table 10. The main parameter values of different algorithms.

Algorithm	Parameter Values
IDPGA	popsize: 100; iterations: 300; developing subpopulation: recombination and mutation probabilities: 0.3 and 0.001; detecting subpopulation: recombination and mutation probabilities: 0.9 and 0.05
SC	civilization size: ten times the dimension of variables; time steps: over 1000
PSO-DE	popsize: 100; crossover probability (F): a random number between 0.9 and 1.0; crossover probability (CR): a random number between 0.95 and 1.0

Table 10. *Cont.*

Algorithm	Parameter Values
DEDS	popsize: 50; crossover rate (CR): 0.9; iterations: 900, 1400; tolerances for functions: 0.001
HEAA	initial popsize of $P_{(0)}$: 60; popsize of Q_1 and Q_2: 200 and 60; μ, λ, ε in simplex crossover: 10, 5 and 10; simplex crossover is executed 40 times in each generation, parents of size μ are randomly chosen from the population in each simplex crossover
CS	number of cuckoos: 25; fraction of the nests being replaced by new nests: a random number of 0 and 1; iterations: 200
AHA	number of hummingbirds: 50; migration coefficient: 2 times popsize; guided factor a and territorial factor b are subject to the normal distribution $N(0,1)$
GA2	uniform crossover, crossover probability: 0.8; non-uniform mutation; only count the number of constraints violated but not the magnitude selection ratio S_r: 0.99;
GA3	binary representation, two-point crossover, and uniform mutation; popsize: 200; iterations: 400; crossover rate: 0.6; mutation rate: 0.03; tournament size: 10
CA	the percentage regulating the rate at which the knowledge becomes specialized: 25%; popsize: 20; iterations: 2500; normative part is updated every 20 generations; tournaments consist of 10 encounters per individual; maximum depth of the octree is equal to the dimension of decision variables
CPSO	Swarm1: popsize: 50; a certain number of generations in PSO: 25; Swarm2: popsize: 20; a certain number of generations in PSO: 8; study factor c_1 and c_2: 2.0 and 2.0; weight factor w: linearly decreases from 0.9 to 0.4; maximum and minimum positions of particles: 1000 and 0; maximum and minimum velocities of particles: 0.2 times range of variables and opposite number
ABC	mean number of scouts: 5–10%; popsize: 50; limit value: product of the dimension of the problem and the number of food sources or employed bees
GA5	binary and fixed point encoding; uniform crossover; non-uniform mutation; crossover and mutation rates: iterated from 0.1 to 0.9 in a nested loop; precision of variables: 3-decimal precision; popsize: 160; iterations: 150
DMO	popsize: 50; iterations: 1000; uniform distribution
AOA	popsize: 30; iterations: 500; sensitive parameter defining the exploitation accuracy over the iterations α: 5; control parameter adjusting the search process μ: 0.5

Table 10. *Cont.*

Algorithm	Parameter Values
SSA	popsize: 50; iterations: 500; nutritious food resources N_{fs}: 4, 1 hickory nut tree and 3 acorn nut trees; gliding constant keeping the balance between exploration and exploitation G_c: 1.9; predator presence probability P_{dp}: 0.1
SCA	parameter dictating the next position regions r_1: a random number; parameter defining the movement r_2: a random number; parameter giving random weights for destination r_3: a random number; parameter equally switching between the sine and cosine components r_4: a random number between 0 and 1; number of search agents: 30; iterations: 500
GWO	parameter emphasizing exploration and exploitation a: decreases from 2 to 0;
CPSOGSA	simple, scalar penalty functions and a more complex penalty function control parameter: 2.05
Hsu and Liu	iterations: 225, 53; tolerance of constraints: 0.01% of the initial values
GSA-GA	popsize: 20 times the dimension of problems; randomly chosen control parameters: $\gamma = 2,\ \delta = 15,\ \beta = 15,\ GA_{MinPS} = 10,\ GA_{MinIter} = 10,$ $GA_{NumMin} = 1,\ GA_{NumMax} = 20$
AO	popsize: 30; iterations: 500
GTO	position-change-controlling parameter A: 0.4; parameter specifying the jumping slope function H: decreases from 2 to 0; popsize: 30; iterations: 3000
TLCO	popsize: 50; iterations: 5000; dead penalty function, penalty parameters: $\beta = 2,\ r_i = 10^{10}$

Table 11. The best results of four algorithms.

Evaluator	Sample			
	IDPGA	**AOA**	**SSA**	**GWO**
1 (f15)	1.7249	2.1397	1.7249	1.7262
2 (f16)	0.0131	0.0131	0.0131	0.0131
3 (f17)	263.8959	263.9154	263.8958	263.8960
4 (f18)	26.5313	24.6170	24.6150	24.6150
5 (f19)	0.0127	0.0121	0.0127	0.0127

The value of each evaluator (the best simulation value for each problem) is ordered from smallest to largest; the smaller the value, the smaller the rank, so that the rank of the four algorithms is 1 to 4, and if the best results of several algorithms are equal, the rank is averaged. The rank sum of an algorithm can be calculated by summing all ranks of the algorithm as shown in Table 12.

Table 12. The rank and rank sum of four algorithms.

Evaluator	Sample			
	IDPGA	**AOA**	**SSA**	**GWO**
1 (f15)	1.5	4	1.5	3
2 (f16)	2.5	2.5	2.5	2.5
3 (f17)	2	4	1	3

Table 12. *Cont.*

Evaluator	Sample			
	IDPGA	**AOA**	**SSA**	**GWO**
4 (f18)	4	3	1.5	1.5
5 (f19)	3	1	3	3
Rank sum	13	14.5	9.5	13

Then, the statistic required by the Friedman test is calculated according to Equation (7), where F is the statistic, A is the number of evaluators, S is the number of samples, and L_i is the sum rank of a sample. Meanwhile, if there are multiple identical ranks in an evaluator's values, the statistic needs to be corrected by Equation (8), and F' is the corrected statistic, n_j is the number of samples with the same rank in each evaluator's values:

$$F = \frac{12}{AS(S+1)} \sum_{i=1}^{n} {L_i}^2 - 3A(S+1) \tag{7}$$

$$\begin{cases} F' = \frac{F}{1-\frac{V}{AS(S^2-1)}} \\ V = \sum\limits_{j=1}^{m} \left({n_j}^3 - n_j\right) \end{cases} \tag{8}$$

The critical value F_0 can be obtained from the critical value table of the Friedman test; F_0 follows the $F((S-1),(S-1)(A-1))$ distribution approximately if A is large or S is greater than 5, and F_0 follows $\chi^2(S-1)$ distribution approximately. If $F \geq F_0$ or $F' \geq F_0$, the samples are significantly different on the whole.

According to Equations (7) and (8), it can be found that F is 1.62, and F' is 2.382. A is small and $S < 5$, so we query the critical value table of Friedman test $\alpha = 0.05, F(3,12)$, F_0 is 3.490, $F' < F_0$, so at the 5% significance level, there is no obvious difference among the four algorithms.

Similarly, the results of IDPGA, DMO, SCA and CPSOGSA are selected as shown in Table 13. The rank and rank sum of an algorithm are shown in Table 14.

Table 13. The best results of four other algorithms.

Evaluator	Sample			
	IDPGA	**DMO**	**SCA**	**CPSOGSA**
1 (f15)	1.7249	1.6953	1.7738	1.6986
2 (f16)	0.0131	0.0131	0.0131	0.0131
3 (f18)	26.5313	24.6150	24.6150	24.6150

Then F is 2, F' is 3.750, query the critical value table of Friedman test $\alpha = 0.05, F(3,6)$, F_0 is 4.757, $F' < F_0$, so at the 5% significance level, there is no obvious difference among the four algorithms.

Table 14. The rank and rank sum of four other algorithms.

Evaluator	Sample			
	IDPGA	**DMO**	**SCA**	**CPSOGSA**
1 (f15)	3	1	4	2
2 (f16)	2.5	2.5	2.5	2.5
3 (f18)	4	2	2	2
Rank sum	9.5	5.5	8.5	6.5

Similarly, the results of IDPGA, SC, PSO-DE, DEDS, HEAA, AHA, GSA-GA and AO are selected as shown in Table 15. The rank and rank sum of each algorithm are shown in Table 16.

Table 15. The best results of eight algorithms.

Evaluator	Sample							
	IDPGA	**SC**	**PSO-DE**	**DEDS**	**HEAA**	**AHA**	**GSA-GA**	**AO**
1 (f15)	1.7249	2.3854	1.7249	2.3810	2.3810	1.7249	1.6952	1.6566
2 (f17)	263.8959	263.8958	263.8958	263.8958	263.8958	263.8958	263.8958	263.8684
3 (f19)	0.0127	0.0127	0.0127	0.0127	0.0127	0.0127	0.0127	0.0112

F is 8.611, F' is 13.910; query the critical value table of $\chi^2(7, 0.1)$ distribution, F_0 is 12.017, $F' > F_0$, so at the 10% significance level, there is obvious difference among the eight algorithms combined with rank sum, AO is the best, PSO-DE, AHA and GSA-GA are close, and IDPGA, SC, DEDS and HEAA are close.

Table 16. The rank and rank sum of eight algorithms.

Evaluator	Sample							
	IDPGA	**SC**	**PSO-DE**	**DEDS**	**HEAA**	**AHA**	**GSA-GA**	**AO**
1 (f15)	4	8	4	6.5	6.5	4	2	1
2 (f17)	8	4.5	4.5	4.5	4.5	4.5	4.5	1
3 (f19)	5	5	5	5	5	5	5	1
Rank sum	17	17.5	13.5	16	16	13.5	11.5	3

Although the first two Friedman tests show no significant difference between IDPGA and six algorithms (AOA, SSA, GWO, DMO, SCA and CPSOGSA), the last Friedman test shows there are significant differences between IDPGA and seven algorithms (SC, PSO-DE, DEDS, HEAA, AHA, GSA-GA and AO). It is worth noting that the number of evaluators here is small, in order to comprehensively compare the performance of the algorithms; if possible, for a certain problem, each algorithm should be repeated to solve the problem several times, and multiple results are selected as the number of evaluators. Of course, combined with the above conclusions, the accuracy and robustness of IDPGA can be guaranteed in solving these six problems, especially the three-bar truss design problem.

5. Conclusions

The improved dual-population genetic algorithm (IDPGA) is proposed for solving multi-constrained optimization problems, which can be applied to engineering problems without simplifying the problems. IDPGA is applied to solve a set of mathematical benchmarks and engineering problems. A few results are unsatisfactory for f8 and f12 in mathematical benchmarks in that the error is more than 60%, but most results are within 20% of the theoretical value and few results are better than the existing. For engineering problems, IDPGA has a higher accuracy of 38% than SC for f15, and the errors and the best value currently are within 2% for other problems. In short, most of the results are satisfactory. Friedman test shows no significant difference between IDPGA and six algorithms, but there are significant differences between IDPGA and other seven algorithms; thus, in order to reach more accurate conclusions, a larger number of evaluators will be needed in the future.

Next, IDPGA will be combined with the FEM software ABAQUS 2022 to form a simulation optimization framework, which is being used to solve the frequency and weight optimization problem of trusses and domes, and several preliminary results show that the framework is suitable for solving the optimization problem of truss systems and yields better solutions than reported.

Some aspects need to be improved in the future, as follows. (i) Parameter tuning is a complex process, and the optimal parameter configuration of IDPGA needs to be further determined more scientifically, especially the number of immigrants. (ii) It is still necessary to further improve the searching speed of IDPGA, such as parallel computing, etc. (iii) More large-scale engineering optimization problems need to be used to prove the effectiveness of IDPGA.

Author Contributions: Formal analysis, X.X.; Investigation, H.L.; Methodology, Z.C.; Software, X.X.; Supervision, Z.C.; Validation, H.L.; Visualization, Z.C.; Writing—original draft, X.X.; Writing—review & editing, H.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by the National Natural Science Foundation of China (Grant No. 52378175).

Institutional Review Board Statement: This section is not relevant to our research.

Informed Consent Statement: This section is not relevant to our research.

Data Availability Statement: All scripts of algorithms are completed in Python and all contours are plotted with Matplotlib and Mayavi. Some or all data, models, or code that support the findings of this study are available from the corresponding author upon reasonable request.

Conflicts of Interest: On behalf of all authors, the corresponding author states that there is no conflict of interest.

Nomenclature

Symbols	Description
$f(\mathbf{x})$	obejective
$g_i(\mathbf{x})$	ith inequality constraints
$h_i(\mathbf{x})$	ith equality constraints
n	number of immigrants
$G_i(H_i)$	a larger value corresponding to the ith constraint
$x_{1j}(x_{2j})$	old individuals
$\widetilde{x}_{1j}\left(\widetilde{x}_{2j}\right)$	new individuals after SBX or PM
$\gamma_j, \mu_j, \Delta_j, \eta_c, \eta_m$	arguments of SBX and PM
ε	tolerance error of equality constraints

References

1. Holland, J.H. *Adaptation in Natural and Artificial Systems*; University of Michigan Press: Ann Arbor, MI, USA, 1975.
2. Kirkpatrick, S.; Gelatt, C.D.; Vecchi, M.P. Optimization by simulated annealing. *Science* **1983**, *220*, 671–680. [CrossRef]
3. Kennedy, J.; Eberhart, R. Particle swarm optimization. In Proceedings of the ICNN'95—International Conference on Neural Networks, Perth, WA, Australia, 27 November–1 December 1995; Volume 4, pp. 1942–1948.
4. Hopfield, J.J. Neural networks and physical systems with emergent collective computational abilities. *Proc. Natl. Acad. Sci. USA* **1982**, *79*, 2554–2558. [CrossRef] [PubMed]
5. Colorni, A.; Dorigo, M.; Maniezzo, V. Distributed optimization by ant colonies. In Proceedings of the First European Conference on Artificial Life, Paris, France, 11–13 December 1991; pp. 134–142.
6. Li, X.L.; Shao, Z.J.; Qian, J.X. An optimizing method based on autonomous animats: Fish-swarm algorithm. *Syst. Eng. Theory Pract.* **2002**, *22*, 32–38. (In Chinese)
7. Yang, X.-S. *Nature-Inspired Metaheuristic Algorithms*; Luniver Press: Bristol, UK, 2010.
8. Yang, X.-S.; Deb, S. Cuckoo search via Lévy flights. In Proceedings of the 2009 World Congress on Nature & Biologically Inspired Computing (NaBIC), Coimbatore, India, 9–11 December 2009; pp. 210–214.
9. Cheraghalipour, A.; Hajiaghaei-Keshteli, M.; Paydar, M.M. Tree Growth Algorithm (TGA): A novel approach for solving optimization problems. *Eng. Appl. Artif. Intell.* **2018**, *72*, 393–414. [CrossRef]
10. Mortazavi, A. Interactive fuzzy search algorithm: A new self-adaptive hybrid optimization algorithm. *Eng. Appl. Artif. Intell.* **2019**, *81*, 270–282. [CrossRef]
11. Talatahari, S.; Azizi, M. Optimization of constrained mathematical and engineering design problems using chaos game optimization. *Comput. Ind. Eng.* **2020**, *145*, 106560. [CrossRef]
12. Abualigah, L.; Yousri, D.; Abd Elaziz, M.; Ewees, A.A.; Al-qaness, M.A.A.; Gandomi, A.H. Aquila Optimizer: A novel meta-heuristic optimization algorithm. *Comput. Ind. Eng.* **2021**, *157*, 107250. [CrossRef]

13. Sadeeq, H.T.; Abdulazeez, A.M. Giant Trevally Optimizer (GTO): A Novel Metaheuristic Algorithm for Global Optimization and Challenging Engineering Problems. *IEEE Access* **2022**, *10*, 121615–121640. [CrossRef]
14. Minh, H.-L.; Sang-To, T.; Theraulaz, G.; Abdel Wahab, M.; Cuong-Le, T. Termite life cycle optimizer. *Expert Syst. Appl.* **2023**, *213*, 119211. [CrossRef]
15. Hasançebi, O.; Çarbaş, S.; Doğan, E.; Erdal, F.; Saka, M.P. Performance evaluation of metaheuristic search techniques in the optimum design of real size pin jointed structures. *Comput. Struct.* **2009**, *87*, 284–302. [CrossRef]
16. Gandomi, A.H.; Yang, X.-S.; Alavi, A.H. Cuckoo search algorithm: A metaheuristic approach to solve structural optimization problems. *Eng. Comput.* **2013**, *29*, 17–35. [CrossRef]
17. Kazemzadeh Azad, S.; Hasançebi, O.; Saka, M.P. Guided stochastic search technique for discrete sizing optimization of steel trusses: A design-driven heuristic approach. *Comput. Struct.* **2014**, *134*, 62–74. [CrossRef]
18. Gholizadeh, S.; Milany, A. An improved fireworks algorithm for discrete sizing optimization of steel skeletal structures. *Eng. Optim.* **2018**, *50*, 1829–1849. [CrossRef]
19. Savvides, A.-A.; Papadopoulos, L. A Neural Network Model for Estimation of Failure Stresses and Strains in Cohesive Soils. *Geotechnics* **2022**, *2*, 1084–1108. [CrossRef]
20. Degertekin, S.O.; Lamberti, L.; Ugur, I.B. Discrete sizing/layout/topology optimization of truss structures with an advanced Jaya algorithm. *Appl. Soft Comput.* **2019**, *79*, 363–390. [CrossRef]
21. Kaveh, A.; Javadi, S.M. Chaos-based firefly algorithms for optimization of cyclically large-size braced steel domes with multiple frequency constraints. *Comput. Struct.* **2019**, *214*, 28–39. [CrossRef]
22. Lee, J.; Hyun, H. Multiple Modular Building Construction Project Scheduling Using Genetic Algorithms. *J. Constr. Eng. Manag.* **2019**, *145*, 04018116. [CrossRef]
23. Adil, B.; Cengiz, B. Optimal design of truss structures using weighted superposition attraction algorithm. *Eng. Comput.* **2020**, *36*, 965–979. [CrossRef]
24. Degertekin, S.O.; Yalcin Bayar, G.; Lamberti, L. Parameter free Jaya algorithm for truss sizing-layout optimization under natural frequency constraints. *Comput. Struct.* **2021**, *245*, 106461. [CrossRef]
25. Srinivasa, K.G.; Sridharan, K.; Shenoy, P.D.; Venugopal, K.R.; Patnaik, L.M. A Dynamic Migration Model for self-adaptive Genetic Algorithms. In *Intelligent Data Engineering and Automated Learning Ideal 2005*; Springer: Berlin, Germany, 2005; pp. 555–562.
26. Li, Y.M.; Zhang, S.J.; Zeng, X.P. Research of multi-population agent genetic algorithm for feature selection. *Expert Syst. Appl.* **2009**, *36*, 11570–11581. [CrossRef]
27. Park, T.; Ryu, K.R. A Dual-Population Genetic Algorithm for Adaptive Diversity Control. *IEEE Trans. Evol. Comput.* **2010**, *14*, 865–884. [CrossRef]
28. Umbarkar, A.J.; Joshi, M.S.; Hong, W.C. Multithreaded Parallel Dual Population Genetic Algorithm (MPDPGA) for unconstrained function optimizations on multi-core system. *Appl. Math. Comput.* **2014**, *243*, 936–949. [CrossRef]
29. Liu, S.S.; Yu, S.L.; Ding, Y.; Zhong, J. Multipopulation Parallel Genetic Algorithm Based on Even Partition. *J. Data Acquis. Process.* **2003**, *18*, 142–145. (In Chinese)
30. Li, J.H.; Li, M.; Yuan, L.H. Improved Dual Population Genetic Algorithm. *J. Chin. Comput. Syst.* **2008**, *11*, 2099–2102. (In Chinese)
31. Pourvaziri, H.; Naderi, B. A hybrid multi-population genetic algorithm for the dynamic facility layout problem. *Appl. Soft Comput.* **2014**, *24*, 457–469. [CrossRef]
32. Fang, B.H.; Yu, L.L. Dual Population Genetic Algorithm Based on Out Mechanism. *Comput. Technol. Dev.* **2009**, *19*, 101–103, 107. (In Chinese)
33. Yu, L.L. The Improvement and Application Research of Dual Population Genetic Algorithm. Master's Thesis, Hefei University of Technology, Hefei, China, 2009. (In Chinese).
34. Lei, Z.Y.; Jiang, Y.M. Dual Population Genetic Algorithm Based on Autonomic Computing. *Comput. Eng.* **2010**, *36*, 189–191. (In Chinese)
35. Tan, Y.; Tan, G.Z.; Ye, Y.; Wu, X.D. Dual population genetic algorithm with chaotic local search strategy. *Appl. Res. Comput.* **2011**, *28*, 469–471. (In Chinese)
36. Tian, F.; Yao, A.M.; Sun, X.P.; Wang, C.Y.; Fan, L.L. Dual population genetic algorithm based on individual similarity. *Comput. Eng. Des.* **2011**, *32*, 1789–1791, 1848. (In Chinese)
37. Guo, D.L.; Yang, N.; Xie, Z.Z. On the Application of Bi—Group Evolutionary Strategies for Solving Nonlinear Multimodal Function Optimization. *J. Qiannan Norm. Univ. Natl.* **2012**, *32*, 105–108. (In Chinese)
38. Ji, X.T.; Xie, H.B.; Zhou, L.; Jia, S. Flight path planning based on an improved genetic algorithm. In Proceedings of the 2013 Third International Conference on Intelligent System Design and Engineering Applications, Hong Kong, China, 16–18 January 2013; pp. 775–778.
39. Li, P.C.; Wang, H. A novel strategy for the crossarm length optimization of PSSCs based on multi-dimensional global optimization algorithms. *Eng. Struct.* **2021**, *238*, 11. [CrossRef]
40. Wang, J.C.; Quan, Y.; Gu, M. Estimation of directional design wind speeds via multiple population genetic algorithm. *J. Wind Eng. Ind. Aerodyn.* **2021**, *210*, 15. [CrossRef]
41. Gao, X.X.; Yang, H.X.; Lu, L.; Koo, P. Wind turbine layout optimization using multi-population genetic algorithm and a case study in Hong Kong offshore. *J. Wind Eng. Ind. Aerodyn.* **2015**, *139*, 89–99. [CrossRef]

42. Zheng, G.W.; Xie, X.H. Optimization design of composite laminates based on improved double population genetic algorithm. *Compos. Sci. Eng.* **2017**, *6*, 28–32. (In Chinese)
43. Gong, J.H.; Liu, X.L. Grid structure optimization method based on genetic algorithm. *J. Tianjin Univ.* **2000**, *01*, 94–97. (In Chinese)
44. Chen, Y.Z.; Mu, Z.G.; Zhang, J.B. Research on genetic algorithm for optimal design of ten-bar truss. In Proceedings of the 10th Conference on Spatial Structures, Beijing, China, 1 December 2002; pp. 471–478.
45. Mu, Z.G.; Chen, Y.Z.; Xiu, L. Application of genetic algorithm in optimum design of space grid structures. *Spat. Struct.* **2003**, 52–54. (In Chinese)
46. Mu, Z.G.; Liang, J.; Sui, J.; Ning, P.H.; Yan, M. Study of optimum design of single layer dome structures based on niche genetic algorithm. *J. Build. Struct.* **2006**, *27*, 115–119. (In Chinese)
47. Tan, R.M.; Xiao, J.C.; Han, Z.G. Multi variable optimal design of spatial grid structures. *Spat. Struct.* **2009**, *15*, 58–61. (In Chinese)
48. Zhang, C.Y.; Wang, Z.Q.; Liang, W.Y. Improvement and application of genetic algorithm in prestressed space lattice work. *J. Harbin Eng. Univ.* **2010**, *31*, 1317–1322. (In Chinese)
49. David, H.W.; William, G.M. *No Free Lunch Theorems for Search*; Santa Fe Institute: Santa Fe, NM, USA, 1995.
50. Wolpert, D.H. The Lack of A Priori Distinctions Between Learning Algorithms. *Neural Comput.* **1996**, *8*, 1341–1390. [CrossRef]
51. Wolpert, D.H.; Macready, W.G. No free lunch theorems for optimization. *IEEE Trans. Evol. Comput.* **1997**, *1*, 67–82. [CrossRef]
52. Wolpert, D.H. The Supervised Learning No-Free-Lunch Theorems. In *Soft Computing and Industry: Recent Applications*; Roy, R., Köppen, M., Ovaska, S., Furuhashi, T., Hoffmann, F., Eds.; Springer: London, UK, 2002; pp. 25–42.
53. Wolpert, D.H.; Macready, W.G. Coevolutionary free lunches. *IEEE Trans. Evol. Comput.* **2005**, *9*, 721–735. [CrossRef]
54. Homaifar, A.; Qi, C.X.; Lai, S.H. Constrained Optimization Via Genetic Algorithms. *Simulation* **1994**, *62*, 242–253. [CrossRef]
55. Hoffmeister, F.A.; Sprave, J. Problem-Independent Handling of Constraints by Use of Metric Penalty Functions. *Evol. Program.* **1996**, *870*. Available online: www.semanticscholar.org/paper/Problem-Independent-Handling-of-Constraints-by-Use-Hoffmeister-Sprave/137a9c23b07bc3f694fde48db4c0ddbf7fccd079 (accessed on 18 August 2023).
56. Kuri, A.; Quezada, C. A universal eclectic genetic algorithm for constrained optimization. In Proceedings of the 6th European Congress on Intelligent Techniques & Soft Computing EUFIT '98, Aachen, Germany, 7–10 September 1998.
57. Joines, J.A.; Houck, C.R. On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GA's. In Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence, Orlando, FL, USA, 27–29 June 1994; Volume 2, pp. 579–584.
58. Kazarlis, S.; Petridis, V. *Varying Fitness Functions in Genetic Algorithms: Studying the Rate of Increase of the Dynamic Penalty Terms*; Springer: Berlin/Heidelberg, Germany, 1998; pp. 211–220.
59. Michalewicz, Z.; Attia, N. *Evolutionary Optimization of Constrained Problems*; Springer: Berlin/Heidelberg, Germany, 1994.
60. Carlson, S.E.; Shonkwiler, R. Annealing a genetic algorithm over constraints, SMC'98 Conference Proceedings. In Proceedings of the 1998 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No. 98CH36218), San Diego, CA, USA, 14 October 1998; Volume 4, pp. 3931–3936.
61. Smith, A.E.; Tate, D.M. Genetic Optimization Using A Penalty Function. In Proceedings of the 5th International Conference on Genetic Algorithms, Urbana-Champaign, IL, USA, 17–21 July 1993; Morgan Kaufmann Publishers Inc.: Burlington, MA, USA, 1993; pp. 499–505.
62. Coit, D.W.; Smith, A.E. Penalty guided genetic search for reliability design optimization. *Comput. Ind. Eng.* **1996**, *30*, 895–904. [CrossRef]
63. Hadj-Alouane, A.B.; Bean, J.C. A Genetic Algorithm for the Multiple-Choice Integer Program. *Oper. Res.* **1997**, *45*, 92–101. [CrossRef]
64. Deb, K. An efficient constraint handling method for genetic algorithms. *Comput. Methods Appl. Mech. Eng.* **2000**, *186*, 311–338. [CrossRef]
65. Li, X.; Du, G. Inequality constraint handling in genetic algorithms using a boundary simulation method. *Comput. Oper. Res.* **2012**, *39*, 521–540. [CrossRef]
66. Liang, X.M.; Qin, H.Y.; Long, W. Genetic Algorithm for Solving Constrained Optimization Problem. *Comput. Eng.* **2010**, *36*, 147–149. (In Chinese)
67. Garcia, R.D.; de Lima, B.; Lemonge, A.C.D.; Jacob, B.P. A rank-based constraint handling technique for engineering design optimization problems solved by genetic algorithms. *Comput. Struct.* **2017**, *187*, 77–87. [CrossRef]
68. Chootinan, P.; Chen, A. Constraint handling in genetic algorithms using a gradient-based repair method. *Comput. Oper. Res.* **2006**, *33*, 2263–2281. [CrossRef]
69. Deb, K.; Agrawal, R.B. Simulated binary crossover for continuous search space. *Complex Syst.* **1995**, *9*, 115–148.
70. Liang, J.J.; Runarsson, T.P.; Mezura-Montes, E.; Clerc, M.; Suganthan, P.N.; Coello, C.C.; Deb, K. Problem definitions and evaluation criteria for the CEC 2006 special session on constrained real-parameter optimization. *J. Appl. Mech.* **2006**, *41*, 8–31.
71. Ray, T.; Liew, K.M. Society and civilization: An optimization algorithm based on the simulation of social behavior. *IEEE Trans. Evol. Comput.* **2003**, *7*, 386–396. [CrossRef]
72. Liu, H.; Cai, Z.; Wang, Y. Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization. *Appl. Soft Comput.* **2010**, *10*, 629–640. [CrossRef]
73. Zhang, M.; Luo, W.; Wang, X. Differential evolution with dynamic stochastic selection for constrained optimization. *Inf. Sci.* **2008**, *178*, 3043–3074. [CrossRef]

74. Wang, Y.; Cai, Z.; Zhou, Y.; Fan, Z. Constrained optimization based on hybrid evolutionary algorithm and adaptive constraint-handling technique. *Struct. Multidiscip. Optim.* **2009**, *37*, 395–413. [CrossRef]
75. Zhao, W.; Wang, L.; Mirjalili, S. Artificial hummingbird algorithm: A new bio-inspired optimizer with its engineering applications. *Comput. Methods Appl. Mech. Eng.* **2022**, *388*, 114194. [CrossRef]
76. Coello Coello, C.A. Use of a self-adaptive penalty approach for engineering optimization problems. *Comput. Ind.* **2000**, *41*, 113–127. [CrossRef]
77. Coello Coello, C.A.; Mezura Montes, E. Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. *Adv. Eng. Inform.* **2002**, *16*, 193–203. [CrossRef]
78. Coello Coello, C.A.; Becerra, R.L. Efficient evolutionary optimization through the use of a cultural algorithm. *Eng. Optim.* **2004**, *36*, 219–236. [CrossRef]
79. He, Q.; Wang, L. An effective co-evolutionary particle swarm optimization for constrained engineering design problems. *Eng. Appl. Artif. Intell.* **2007**, *20*, 89–99. [CrossRef]
80. Karaboga, D.; Akay, B. A comparative study of Artificial Bee Colony algorithm. *Appl. Math. Comput.* **2009**, *214*, 108–132. [CrossRef]
81. Coello, C.A.C. Treating constraints as objectives for single-objective evolutionary optimization. *Eng. Optim.* **2000**, *32*, 275–308. [CrossRef]
82. Deb, K.; Goyal, M. Optimizing engineering designs using a combined genetic search. *ICGA Citeseer* **1997**, 521–528.
83. Agushaka, J.O.; Ezugwu, A.E.; Abualigah, L. Dwarf Mongoose Optimization Algorithm. *Comput. Methods Appl. Mech. Eng.* **2022**, *391*, 114570. [CrossRef]
84. Abualigah, L.; Diabat, A.; Mirjalili, S.; Abd Elaziz, M.; Gandomi, A.H. The Arithmetic Optimization Algorithm. *Comput. Methods Appl. Mech. Eng.* **2021**, *376*, 113609. [CrossRef]
85. Jain, M.; Singh, V.; Rani, A. A novel nature-inspired algorithm for optimization: Squirrel search algorithm. *Swarm Evol. Comput.* **2019**, *44*, 148–175. [CrossRef]
86. Mirjalili, S. SCA: A Sine Cosine Algorithm for solving optimization problems. *Knowl. Based Syst.* **2016**, *96*, 120–133. [CrossRef]
87. Mirjalili, S.; Mirjalili, S.M.; Lewis, A. Grey Wolf Optimizer. *Adv. Eng. Softw.* **2014**, *69*, 46–61. [CrossRef]
88. Hsu, Y.-L.; Liu, T.-C. Developing a fuzzy proportional–derivative controller optimization engine for engineering design optimization problems. *Eng. Optim.* **2007**, *39*, 679–700. [CrossRef]
89. Rao, S.S. *Engineering Optimization: Theory and Practice*, 3rd ed.; John Wiley Sons: Chichester, UK, 1996.
90. Garg, H. A hybrid GSA-GA algorithm for constrained optimization problems. *Inf. Sci.* **2019**, *478*, 499–523. [CrossRef]
91. Agushaka, O.J.; Ezugwu, A.E.S. Influence of Initializing Krill Herd Algorithm with Low-Discrepancy Sequences. *IEEE Access* **2020**, *8*, 210886–210909. [CrossRef]
92. Li, Q.; Liu, S.-Y.; Yang, X.-S. Influence of initialization on the performance of heuristic optimizers. *Appl. Soft Comput.* **2020**, *91*, 106193. [CrossRef]
93. Beasley, T.M.; Zumbo, B.D. Comparison of aligned Friedman rank and parametric methods for testing interactions in split-plot designs. *Comput. Stat. Data Anal.* **2003**, *42*, 569–593. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.