Journal of Systems Science and Information, Volume 7, Issue 6, 2019, pp. 532-549
DOI: 10.21078/JSSI-2019-532-18

Multiobjective Imperialist Competitive Algorithm for Solving Nonlinear Constrained Optimization Problems

Chun-an LIU

School of Mathematics and Information Science, Baoji University of Arts and Sciences, Baoji 721013*, China*

*E-mail: liu*2006@126*.com*

Huamin JIA

*School of Engineering, Cranfield University, England, MK*43 0*AL, UK*

E-mail: hum.jia@cranfield.ac.uk

Abstract Nonlinear constrained optimization problem (NCOP) has been arisen in a diverse range of sciences such as portfolio, economic management, airspace engineering and intelligence system etc. In this paper, a new multiobjective imperialist competitive algorithm for solving NCOP is proposed. First, we review some existing excellent algorithms for solving NOCP; then, the nonlinear constrained optimization problem is transformed into a biobjective optimization problem. Second, in order to improve the diversity of evolution country swarm, and help the evolution country swarm to approach or land into the feasible region of the search space, three kinds of different methods of colony moving toward their relevant imperialist are given. Thirdly, the new operator for exchanging position of the imperialist and colony is given similar as a recombination operator in genetic algorithm to enrich the exploration and exploitation abilities of the proposed algorithm. Fourth, a local search method is also presented in order to accelerate the convergence speed. At last, the new approach is tested on thirteen well-known NP-hard nonlinear constrained optimization functions, and the experiment evidences suggest that the proposed method is robust, efficient, and generic when solving nonlinear constrained optimization problem. Compared with some other state-of-the-art algorithms, the proposed algorithm has remarkable advantages in terms of the best, mean, and worst objective function value and the standard deviations.

Keywords multiobjective optimization; imperialist competitive algorithm; constrained optimization; local search

1 Introduction

In science and engineering fields, many complex optimization problems involve in constraint conditions[1−3]. That's to say, the optimal solution of those practical problems are restricted to the problem's constraint conditions. For example [4], valves in chemical process control need a maximum and a minimum displacement. Also, for safety or other operational reason, it is

Received March 19, 2019, accepted April 26, 2019
Supported by the Planning Fund for the Humanities and Social Sciences of the Ministry of Education (18YJA790053) and the National Scholarship Fund in China, the Project Sponsored by the Scientific Research Foundation for the Returned Overseas Chinese Scholars

usual to impose some limits on allowable temperatures, levels and pressures. When solving these optimization problems, it is difficult to deal with the constraints and find the optimal solution of the nonlinear constrained problem.

Mostly often, constraint handling methods used in classical optimization algorithm can be classified into two types: One is generic methods that do not exploit the mathematical structure of the constraint, such as the penalty function method[5], Lagrange multiple method[6], and some intelligence optimization search heuristic methods[7−11], and the other is special methods that used to solve these problems with specific types of constraints, such as the cutting place method[12], the gradient projection method[13], the quasi-Newton method[14] and the steepest descent method[15], etc.

As far as generic methods are concerned, since these algorithms are generic, some performances of them in some case can't be fully satisfied. However, these special methods are applicable either to these optimization problems having convex search region only or to these optimization problem whose objective and constraint functions are differentiable. In fact, among the generic methods, the most popular approach in real optimization fields to deal with the constraint of an optimization problem is the penalty function method, which involves a number of penalty parameters and we must to set right in any algorithms in order to obtain the optimal solution, and this performance on penalty parameter has led many researches to devise the sophisticated penalty function method. These methods mainly can be divided three categories: a) multi-level penalty functions[16]; b) dynamic penalty functions based on adaptive and co-evolutionary penalty approaches[17]; and c) hybrid penalty functions combined with the advantages of evolutionary computation, such as [18, 19].

Evolutionary algorithm is generally inspired by the modelling of the natural processes, especially human evolution. Genetic algorithm lies in the category of evolutionary algorithms. However, imperialist competitive algorithm (ICA) uses socio-political evolution of human as a source of inspiration for developing a strong optimization strategy proposed by Atashpaz-Gargari and Lucas[20] in 2007. ICA has been succeeded widely to solve many real world optimization problems in recent years, e.g., in [21], Mahdi, Ayaz, and Davoud introduced an imperialist competitive algorithm for solving systems of nonlinear equations; in [22], Mohammadi, Tavakkoli-Moghaddam, and Rostami designed a multi-objective imperialist competitive algorithm to solve a capacitated hub covering location problem; Shokrollahpour, Zandieh and Dorri proposed a novel imperialist competitive algorithm for solving bi-criteria scheduling of the assembly flow-shop problem[23], etc.

Moreover, how to find a balance between exploration and exploitation for an excellent generic algorithm is very important. In [11], the authors proposed an enhanced grey wolf optimization (EGWO) algorithm with a better hunting mechanism, which focuses on the proper balance between exploration and exploitation that leads to an optimal performance of the algorithm and hence promising candidate solutions are generated. And in [24], the authors introduced a nonlinear control parameter strategy and a new position-updated equation in order to balance the exploration and exploitation of the algorithm, etc.

In this paper, we proposed a new multiobjective optimization method based on ICA to solve nonlinear constrained optimization problem. Firstly, the nonlinear constrained optimization

problem concerned is transformed into a bi-objective unconstrained optimization problem, so that no penalty function or other mechanism to deal with the constrained are introduced. Then, in order to improve the diversity of evolution country swarm, and help the evolution country swarm to approach or land into the feasible region, three kinds of different methods of colonies moving toward their relevant imperialist are presented. Also, the new operator for exchanging position of the imperialist and colony is given as a recombination operator in genetic algorithm to achieve a better balance of the explorative and exploitative behaviors of the proposed algorithm. Moreover, a new local search method is also integrated in order to increase the convergence speed of the proposed algorithm. At last, the new method is tested on 13 well-known NP-hard nonlinear constrained optimization functions, and the experiment results suggest that it is robust, efficient, and generic when solving nonlinear constrained optimization problem. Compared with some other state-of-the-art algorithms, the proposed algorithm has remarkably advantage in terms of the best, mean, and worst objective function value and the standard deviation, i.e., it is indicated that the proposed algorithm can effectively solve the nonlinear constrained optimization problem.

The paper is organized as follows. In Section 2, the related concepts of nonlinear constrained optimization problem are given. The main steps of the proposed imperialist competitive algorithm for solving the nonlinear constrained optimization problem are designed in Section 3. The flowchart of the proposed algorithm is described in Section 4. After simulation results are shown in Section 5, the conclusion and acknowledgment are made in Section 6 and Section 7, respectively.

2 Related Concepts of NCOP

Without loss of generality, the general nonlinear constrained optimization problem (NCOP) that we are interested in can be formulated as

$$\begin{cases} \min\limits_{x\in D\in[L,U]} f(x) \\ \text{s.t.} \quad g_i(x)\le 0, \qquad i=1,2,\cdots,p \\ \qquad\quad h_j(x)=0, \qquad j=p+1,p+2,\cdots,l, \end{cases} \tag{1}$$

where $x=(x_1,x_2,\cdots,x_n)^{\mathrm{T}}\in \boldsymbol{R}^n$ is n dimension decision vector and $g_i(x)\le 0$ is the inequation constraint for $i=1,2,\cdots,p$, $h_j(x)=0$ is equation constraint for $j=p+1,p+2,\cdots,l$ (in both cases, constraints could be linear or non-linear), and

$$D=\{x|g_i(x)\le 0, i=1,2,\cdots,p; h_j(x)=0, j=p+1,p+2,\cdots,l\} \tag{2}$$

is feasible region,

$$[L,U]=\{[\lambda_1,\mu_1]\times[\lambda_2,\mu_2]\times\cdots\times[\lambda_n,\mu_n]|\lambda_i\le x_i\le\mu_i, i=1,2,\cdots,n\} \tag{3}$$

is the search space, and

Definition 1 For every point $x\in D$, if exists a point $x^*\in D$ such that $f(x^*)\le f(x)$ holds, then the point x^* is called the optimal solution, and $f(x^*)$ is the optimal value for problem (1).

Let $f_1(x) = f(x)$, $f_2(x) = \sum_{i=1}^{p} \max\{0, g_i(x)\}^2 + \sum_{j=p+1}^{l} (h_j(x))^2$, where $f(x)$ is the objective function of problem (1) and $f_2(x)$ is the optimization function defined by the constraint condition of problem (1), then, we can transform the nonlinear constrained optimization problem (1) into the biobjective optimization problem as follows:

$$\min_{x \in [L,U]} \mathcal{F}(x) = (f_1(x), f_2(x)). \tag{4}$$

For the biobjective optimization problem (4), to minimize the first objective function $f_1(x)$ means to find a feasible point so as to become the optimal solution of problem (1), to minimize the second objective function $f_2(x)$ means to search the point in order to meet all the constraints of problem (1). Therefore, when to minimize the two objectives function of problem (4) simultaneously means searching for the point so as to satisfy all the constraints and make the objective function of problem (1) to reach the optimum.

Definition 2 A two dimension vector $u = (u_1, u_2)$ is said to weakly dominate another two dimension vector $v = (v_1, v_2)$, iff $u_i \leq v_i$ for $i = 1, 2$.

Definition 3 A point $x \in [L, U]$ is said to be a weakly Pareto optimal solution for problem (4) if there is not existing another point $y \in [L, U]$ such that $\mathcal{F}(y)$ weakly dominates $\mathcal{F}(x)$. The set of all the weakly Pareto optimal solutions is called the weakly Pareto optimal set and the set of all the weakly Pareto optimal solution's objective vectors is called the weakly Pareto front.

Suppose that $E_w(\mathcal{F}, x)$ is the weakly Pareto optimal solution set of problem (4). Then, the optimal solution of problem (1) and the weakly Pareto optimal solution (4) have the following relation.

Theorem 1 *A solution x^* is the optimal solution of problem* (1)*, iff $x^* \in D \cap E_w(\mathcal{F}, x)$ and $x^* = \arg\min_{x \in D} f_1(x)$.*

Proof Sufficiency is obvious. The necessity proof is given as follows:

Since x^* is the optimal solution of problem (1), then $f(x^*) = f_1(x^*) = \min_{x \in D} f_1(x)$, i.e., $x^* = \arg\min_{x \in D} f_1(x)$. Furthermore, we have $x^* \in D$ and $f_2(x^*) = 0$. If $x^* \notin E_w(\mathcal{F}, x)$, then there at least exists another solution $\tilde{x} \in D$, and makes $f_i(\tilde{x}) < f_i(x^*)$ for $i = 1, 2$ hold, i.e., $f_2(\tilde{x}) < f_2(x^*) = 0$, this is contradiction to the definition of function $f_2(x) \geq 0$ for $\forall x \in D$, so $x^* \in E_w(\mathcal{F}, x)$, i.e., $x^* \in D \cap E_w(\mathcal{F}, x)$.

The conclusion of the Theorem 1 demonstrates that the optimal solution of problem (1) can be obtained from the intersection of the feasible region of problem (1) and the weakly Pareto optimal solution set of problem (4), and the optimal solution makes the first objective function minimum.

3 The Design of the Main Operator's for the Proposed Algorithm

In order to solve the nonlinear constrained optimization problem (NCOP) proposed in Section 2, a new imperialist competitive algorithm is designed in Sections 3 and 4. Firstly, we briefly introduce the idea of the imperialist competitive algorithm (ICA), proposed by the authors Atashpaz-Gargari and Lucas[20]. ICA, similar to the evolutionary algorithm, particle swarm algorithm and so on, is a kind of swarm intelligence algorithm. ICA is inspired by

imperialistic competition. All the countries are divided into two types: Imperialist states and colonies. Imperialistic competition is the main evolution operator and hopefully causes the colonies to approach the global optimal solution. Based on the idea, we design the main operators for the proposed algorithm as follows.

3.1 The Creation of Initial Empires

During the operation process of ICA, the initial evolution country swarm, should be generated firstly. Among the initial country swarm, some of the best countries are selected to form the initial imperialist, and the rest of the countries are divided among the initial imperialists as colonies. In this section, randomly generate $\overline{pop}$ initial countries in search space $[L, U]$, denote them as country $i = (x_1^i, x_2^i, \cdots, x_n^i)^{\mathrm{T}}$ for $i = 1, 2, \cdots, \overline{pop}$, and define the cost of each country i as follows:

$$\text{cost(country } i) = \begin{cases} f_1(\text{country } i), & \text{country } i \in D, \\ f_2(\text{country } i), & \text{country } i \in [L, U] \backslash D, \end{cases} \tag{5}$$

where $f_1(x)$ is the first objective function and $f_2(x)$ is the second objective function of problem (4), respectively. Select N of the most powerful countries to form empires, where the most powerful countries refer to the countries whose cost are relatively small. The rest countries of the initial countries will become colonies of each of empires according to their power. Thus, each empire receives a number of colonies. This process is presented in Figure 1, where the more powerful empires have a greater number of colonies and weaker empires have fewer colonies. At last, these initial countries is divided into two groups: Imperialist and colony (denote in imperialist i and colony j for $i = 1, 2, \cdots, N$, $j = 1, 2, \cdots, \overline{pop} - N$, respectively). In order to form the initial empires, we divide colonies into N imperialists based on their power. Here, we divide these colonies among imperialists according to the method of proportion selection or the roulette wheel selection used in genetic evolution as follows:

Step 1 Suppose the normalized power of each imperialist is defined by

$$p_j = \Big| \frac{C_j}{\sum_{i=1}^{N} C_i} \Big|, \tag{6}$$

where p_j is the normalized power of the j-th imperialist, and $C_j = c_j - \max_{1 \le i \le N}\{c_i\}$ is the normalized cost of the j-th imperialist for $j = 1, 2, \cdots, N$, c_i is the cost of the i-th imperialist for $i = 1, 2, \cdots, N$.

Step 2 Generate the initial number of the colonies belonging to each empire based on the following formula

$$N.C._j = round\{p_j \cdot (\overline{pop} - N)\}, \tag{7}$$

where $N.C._j$ is the number of initial colonies of the j-th empire, and $\overline{pop} - N$ is the total number of all initial colonies.

Step 3 Select $N.C._j$ of the colonies according to the roulette wheel selection and join them to the j-th imperialist. These colonies along with the imperialist together will form the j-th empire (denote empire j, $j = 1, 2, \cdots, N$).

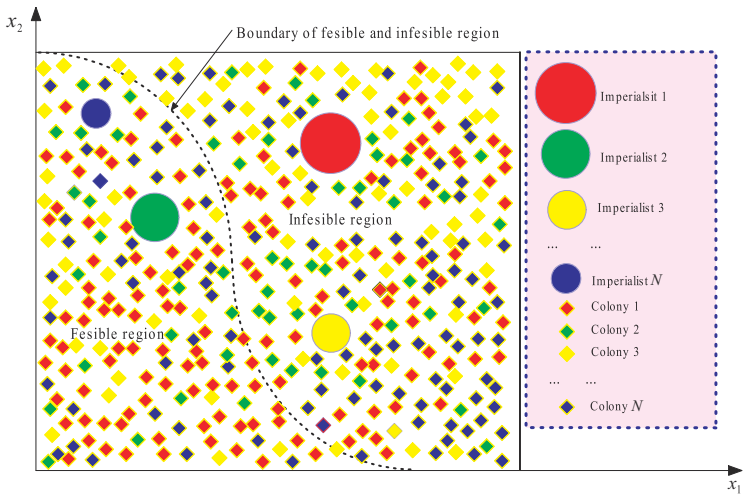

Figure 1 Generation of the initial empire and their initial colony in search space $[L, U] \in \boldsymbol{R}^2$

3.2 Method of Colonies Moving Toward Their Relevant Imperialist

In [20], the authors make each colony to move toward the imperialist by x-units in the direction which is the vector from colony to imperialist. x will be a random variable with uniform distribution, i.e.,

$$x \sim U(0, \beta \times d), \tag{8}$$

where $\beta > 1$ and d is the distance between the colony and imperialist, and parameter β causes the colony to get closer to the imperialist from both sides. However, for constraint optimization problem, it needs the designed algorithm not only to make the infeasible solution approaching the feasible region and satisfying the constraint condition, but also make the objective function minimum. Based on these, we proposed a new method of colonies moving to their relevant imperialist as follows. Suppose that we make the colony j $(j = 1, 2, \cdots, \overline{pop} - N)$ to move the imperialist i $(i = 1, 2, \cdots, N)$, then,

Case 1 If both imperialist i and colony j are feasible, i.e., imperialist i and colony $j \in D$, we generate a circle which the diameter is the straight-line segment d joining the imperialist i and colony j, the new position (denote as colony k) of the j-th colony moved to their relevant imperialist is shown in a gray colour in Figure 2, where A and r are two random numbers with uniform distribution, i.e.,

$$A \sim U(-\pi, \pi), \tag{9}$$

and

$$r \sim U(0, d \cdot \cos A). \tag{10}$$

Parameter r and A can cause the colony j to get closer to the imperialist from its neighbourhood rather than far away from the imperialist i.

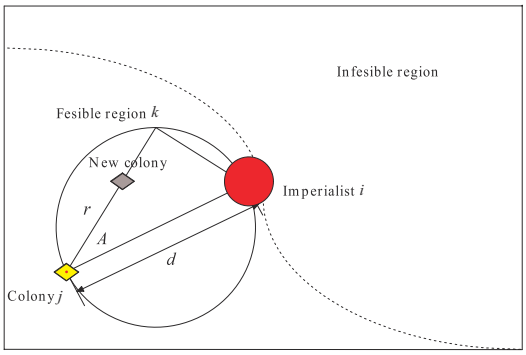

Figure 2 The method of colony moving to imperialist based on both the j-th colony and i-th imperialist are feasible in search space $[L, U] \in \boldsymbol{R}^2$

Case 2 If both imperialist i and colony j are infeasible, i.e., imperialist i, colony $j \in [L, U] \backslash D$, we random select a colony s from feasible region D, compute the barycenter of three countries colony j, imperialist i and colony s, and then, the barycenter (denote by colony k in Figure 3) can be regarded as the new position which colony j move to imperialist i. Using this method, we can make the colony not only to move to the imperialist but approach the feasible region.

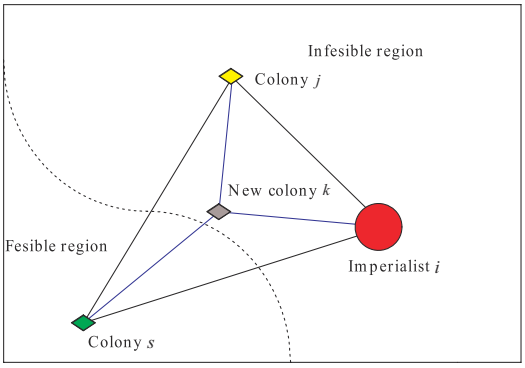

Figure 3 The method of colony moving to imperialist based on the fact that both colony j and imperialist i are infeasible in search space $[L, U] \in \boldsymbol{R}^2$

Case 3 If there exists one feasible country between colony j and imperialist i, and suppose colony j is feasible country and imperialist i is infeasible country and vice versa, i.e., colony $j \in D$, imperialist $i \in [L, U] \backslash D$, then, we generate a circle which the circle's center is colony j and the radius is the straight-line segment L joining the colony j and imperialist i, the new position colony k of colony j moved to imperialist i is shown in a gray colour in Figure 4, where B is a random number with uniform distribution, i.e.,

$$B \sim U(-\varphi, \varphi), \tag{11}$$

where φ is a parameter that adjusts the deviation of direction which is the vector from colony j to imperialist i, $l \sim U(0, \tau \cdot L)$ is a random number with uniform distribution, and τ and

φ are arbitrary. In most of our implementation, the value of $\tau < \frac{1}{2}$ and $\varphi = \frac{\pi}{4}$ have a good convergence to the global minimum and can make the feasible colony j not far away from the feasible region.

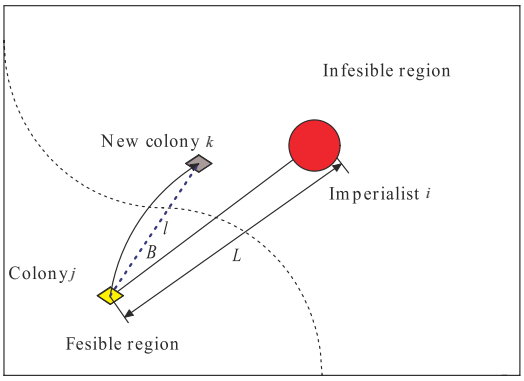

Figure 4 The method of colony moving to imperialist based on the fact that colony j is feasible and imperialist i is infeasible in search space $[L, U] \in \boldsymbol{R}^2$

3.3 Exchanging Position of the Imperialist and Colony

Based on the method of colonies moving toward their relevant imperialist in Subsection 3.2, the operator of exchanging position of the imperialist and the colony can be described as follows:

1) If both colony j and imperialist i are feasible, and suppose that the cost of colony j has lower cost than that the imperialist i does, i.e., $f_1(\text{colony } j) < f_1(\text{imperialist } i)$, then we use the colony j to replace the imperialist i and form the new imperialist, vise versa.

2) If both colony j and imperialist i are unfeasible, then we always choose the one with the smaller cost as the new imperialist i, i.e., if $f_2(\text{colony } j) > f_2(\text{imperialist } i)$, then keep imperialist i invariable; otherwise, if $f_2(\text{colony } j) < f_2(\text{imperialist } i)$, then use the colony j to replace the imperialist i and form new imperialist.

3) If there exists one feasible country between the colony j and imperialist i, we always use the feasible country as the new imperialist in order to make the evolution country swarm approaching the feasible region and fast converging to the minimum.

3.4 Local Search Operator

In order to accelerate the convergence speed, we add a local search operator as follows to the proposed algorithm. Suppose imperialist 1, imperialist 2, $\cdots$, imperialist k are k imperialists obtained by the proposed algorithm in current evolution countries swarm, where k is the number of imperialist and it will become more less with the imperialist competition.

Compute the approximate value of gradient for each imperialist $i \in [L, U]$, i.e., $\nabla y = (\nabla F_1, \nabla F_2, \cdots, \nabla F_n)^{\mathrm{T}}$, where

$$\nabla F_j = \begin{cases} \dfrac{f_1(\text{imperialist } i + \delta e_j) - f_1(\text{imperialist } i)}{\delta}, & \text{imperialist } i \in D, \\ \dfrac{f_2(\text{imperialist } i + \delta e_j) - f_2(\text{imperialist } i)}{\delta}, & \text{imperialist } i \in [L, U] \backslash D, \end{cases} \tag{12}$$

for $i = 1, 2, \cdots, k$, $j = 1, 2, \cdots, n$, $\delta > 0$, and $e = (e_1, e_2, \cdots, e_n)^{\mathrm{T}}$ is a n-dimension unit vector

which the j-th component $e_j = 1$ and the rest of components $e_\iota = 0, \iota \neq j$, then,

Case 1 If $\|\nabla y\| \neq 0$, carry out one-dimensional search along the descent direction $d = -\nabla y$, i.e., obtain α^* and make the following formula

$$\min_{\text{imperialist}i \in D} f_1(\text{imperialist}i + \alpha d) = f_1(\text{imperialist}i + \alpha^* d) \tag{13}$$

or

$$\min_{\text{imperialist}i \in [L,U]\setminus D} f_2(\text{imperialist}i + \alpha d) = f_2(\text{imperialist}i + \alpha^* d) \tag{14}$$

holds, where $(\text{imperialist}i + \alpha^* d) \in [L, U]$, then we use $\text{imperialist}i + \alpha^* d$ to replace $\text{imperialist}i$ and form the new imperialist.

Case 2 If $\|\nabla y\| = 0$, keep $\text{imperialist}i$ no change.

3.5 Imperialistic Competition

As mentioned in [20], during the evolution of countries, all the empires try to possess the colonies of the other empires and control them. As a result, the power of the weaker empires gradually begins to decrease and the power of the more powerful increases. This process can be described in the follows:

1. Compute the total power $T.C._j$ of the j-th empire depending on its own imperialist and colonies as follows:

$$T.C._j = \text{cost}(\text{imperialist}^*) + \sigma \cdot \frac{1}{N.C._j} \sum_{i=1}^{N.C._j} \text{cost}(\text{colony } i), \tag{15}$$

   where $\sigma < 1$ is a positive coefficient, and imperialist^* is the imperialist of the j-th empire, $N.C._j$ is the number of colonies of the j-th empire, $\text{cost}(\cdot)$ is the normalized cost function defined in formula (5).

2. Use the following formula (16) to compute the possession probability $E.P._j$ of each empire$j$ for $j = 1, 2, \cdots, N$, i.e.,

$$E.P._j = \Big| \frac{N.T.C._j}{\sum_{i=1}^{N} N.T.C._i} \Big|, \tag{16}$$

   where $N.T.C._j = T.C._j - \max_{1 \leq i \leq N}\{T.C._i\}$ is the normalized total power and the $T.C._j$ is the total power of the j-th empire, respectively.

3. Randomly select some colonies from current evolution countries swarm, e.g., when select only one and denote in $\overline{\text{colony}}$, let $P = (E.P._1, E.P._2, \cdots, E.P._N)$, and also generate an N-dimension vector V with uniformly distributed elements, i.e.,

$$V = (V_1, V_2, \cdots, V_N)^{\mathrm{T}}, \tag{17}$$

   where $V_i \sim \text{rand}(0, 1)$ for $i = 1, 2, \cdots, N$. Furthermore, let

$$\Theta = (\Theta_1, \Theta_2, \cdots, \Theta_N)^{\mathrm{T}} = (E.P._1 - V_1, E.P._2 - V_2, \cdots, E.P._N - V_N)^{\mathrm{T}}, \tag{18}$$

   then, we divide the $\overline{\text{colony}}$ into the j-th empires, where index $j (j = 1, 2, \cdots, N)$ is subscript of the maximum component in vector Θ.

With the imperialistic competition, powerless empires will collapse in the imperialist competitive, and the number of their colonies become less and less. when an empire loses all of its colonies, we consider that the empire has been collapse and the imperialist become one of the rest colonies.

4 The Flowchart of the Proposed Algorithm

The main difference between the proposed multi-objective imperialist competitive evolutionary algorithm (denote as MICA) and the original imperialist competitive algorithms is the method of colony moving toward their relevant imperialist according to Figures 2∼4. Additionally, in order to make the evolution country swarm to approach or come in the feasible region, three kinds of different methods of colonies moving toward their relevant imperialist are given. In addition, a new operator for exchanging position of the imperialist is also designed to achieve a better balance between the exploration and exploitation behaviors for MICA, and a new local search method is also embedded in order to increase the convergence speed of the proposed algorithm. The flowchart of the proposed algorithm is shown as follows:

Step 1 Choose the proper parameter, initial country size $\overline{pop}$, randomly generate initial country swarm in search space $[L, U]$, and denote them as the set $pop(0)$, find the weakly Pareto optimal countries (i.e., weakly Pareto optimal solutions) in $pop(0)$ according to Definition 3 and denote them as an external set $C(0)$, let $t = 0$.

Step 2 Generate initial empires, i.e., select the most powerful countries N from $pop(t)$ and divide the rest countries to each of them.

Step 3 Make each of colonies to move toward relative imperialist based on the method of colonies moving toward their relevant imperialist in Subsection 3.2, and exchange the position of the imperialist and the colony according to Subsection 3.3.

Step 4 Carry out the local search operator and imperialistic competition, and form the next evolution country swarm $pop(t + 1)$.

Step 5 Find the weakly Pareto optimal countries in the set $C(t) \cup pop(t+1)$ and use them to replace those countries including into set $C(t)$ to form the new external set $C(t + 1)$.

Step 6 If the maximum number of the cycles has been reached, the algorithm stop; output the optimal solution $x^* = \arg\min_{x \in D \cap C(t+1)} f_1(x)$ of problem (1). Otherwise, let $t = t + 1$, go to Step 2.

5 Experimental Results and Discussions

To evaluate the efficiency of the proposed algorithm, thirteen nonlinear constrained optimization test problems g01∼g13 were tested by six optimization evolutionary algorithms: OICA[20], SAEA[25], SMEA[26], RCEA[27], ISEA[28], and the proposed algorithm MICA. These benchmark functions are described in [27]. And they are summarized here for completeness, and the original sources of the functions are also cited. Test functions g02, g03, and g12 are maximization problems, they were transformed into minimization problems using the objective function $\min(-f(x))$.

In order to estimate how difficult it is to generate feasible countries through a purely random process, we use the ρ-metric[29] which can measure the ratio between the size of the feasible

search space and that of the entire search space, i.e.,

$$\rho = |\Omega|/|S|, \tag{19}$$

where $|S|$ is the number of countries randomly generated from search space $[L, U]$, and $|\Omega|$ is the number of feasible countries found by each algorithm (out of these $|S|$ countries). In our algorithm, $|S|$ is the initial country size N_{pop}.

Each algorithm was implemented by using MATLAB 7.0 on an Intel Pentium IV 2.8-GHz personal computer, and was executed 30 independent runs for each test problem. In the simulation, the initial country size $N_{pop} = 500$, the ratio of the most powerful countries is $N_{pop} \times 5\%$, and the maximum number of cycles is 1500.

Table 1 summarizes the average percentage of the feasible countries in the final evolution country swarm in 30 independent runs for each test problem. Moreover, In order to illustrate the rate of the convergence for the proposed algorithm, we record the average distance from the best individual of the imperialist swarm to the boundaries of the feasible region at every 1500 generations in 30 runs. The results are presented in Table 2. Also, we list the known optimal solution and the best, mean, and the std. for the objective function value in Table 3, and the standard deviation (std.) after 30 independent runs by MICA and the original imperialist competitive algorithm (denote as OICA) is also given. These results provided by four compared algorithms SAEA, SMEA, RCEA and ISEA were taken from the original references. In Table 2, "I.N." represents the iteration number, and in Table 3, "NA" presents no results in the reference.

g01[30]

$$\begin{aligned}
\min\ & f(x) = 5\sum_{i=1}^{4} x_i - 5\sum_{i=1}^{4} x_i^2 - \sum_{i=5}^{13} x_i \\
\text{s.t.}\ & g_1(x) = 2x_1 + 2x_2 + x_{10} + x_{11} - 10 \le 0, \\
& g_2(x) = 2x_1 + 2x_3 + x_{10} + x_{12} - 10 \le 0, \\
& g_3(x) = 2x_2 + 2x_3 + x_{11} + x_{12} - 10 \le 0, \\
& g_4(x) = -8x_1 + x_{10} \le 0, \\
& g_5(x) = -8x_2 + x_{11} \le 0, \\
& g_6(x) = -8x_3 + x_{12} \le 0, \\
& g_7(x) = -2x_4 - x_5 + x_{10} \le 0, \\
& g_8(x) = -2x_6 - x_7 + x_{10} \le 0, \\
& g_9(x) = -2x_8 - x_9 + x_{12} \le 0,
\end{aligned}$$

where $0 \le x_i \le 1$ for $i = 1 \sim 9$, $0 \le x_i \le 100$ $(i = 10 \sim 12)$, and $0 \le x_{13} \le 1$. The global minimum is $x^* = (1, 1, 1, 1, 1, 1, 1, 1, 1, 3, 3, 3, 1)$ and the optimum value $f(x^*) = -15$. Moreover, the constraints g_1, g_2, g_3, g_7 and g_8 are active.

g02[31]

$$\max\ f(x) = \left| \frac{\sum_{i=1}^{n} \cos^4 x_i - 2\prod_{i=1}^{n} \cos^2 x_i}{\sqrt{\sum_{i=1}^{n} i x_i^2}} \right|$$

$$\text{s.t.} \quad g_1(x) = 0.75 - \prod_{i=1}^{n} x_i \leq 0,$$
$$g_2(x) = \sum_{i=1}^{n} x_i - 7.5n \leq 0,$$

where $n = 20$, $0 \leq x_i \leq 10$ for $i = 1 \sim n$, the global maximum is unknown, the best we found is $f(x^*) = 0.803619$, which is better than any reported value up to the best of our knowledge, and the constraint g_1 is active.

g03[32]

$$\max \ f(x) = (\sqrt{n})^n \prod_{i=1}^{n} x_i$$
$$\text{s.t.} \quad h_1(x) = \sum_{i=1}^{n} x_i^2 = 0,$$

where $n = 10$, $0 \leq x_i \leq 1$ for $i = 1 \sim n$, the global maximum is $x_i^* = \frac{1}{\sqrt{n}}$, and $f(x^*) = 1$.

g04[33]

$$\begin{aligned}
\min \ & f(x) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141 \\
\text{s.t.} \ & g_1(x) = 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 - 92 \leq 0, \\
& g_2(x) = -85.334407 - 0.0056858x_2x_5 - 0.0006262x_1x_4 + 0.0022053x_3x_5 \leq 0, \\
& g_3(x) = 80.51249 + 0.0071731x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 - 110 \leq 0, \\
& g_4(x) = -80.51249 - 0.0071731x_2x_5 - 0.0029955x_1x_2 - 0.0021813x_3^2 + 90 \leq 0, \\
& g_5(x) = 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 - 25 \leq 0, \\
& g_6(x) = -9.300961 - 0.0047026x_3x_5 - 0.0012547x_1x_3 - 0.0019085x_3x_4 + 20 \leq 0,
\end{aligned}$$

where $78 \leq x_1 \leq 102$, $33 \leq x_2 \leq 45$, $27 \leq x_i \leq 45$ for $i = 3, 4, 5$. The optimum is $x^* = (78, 33, 29.995256025682, 45, 36.775812905788)$, and $f(x^*) = -30665.539$.

g05[34]

$$\begin{aligned}
\min \ & f(x) = 3x_1 + 0.000001x_1^3 + 2x_2 + \frac{0.000002}{3}x_2^3 \\
\text{s.t.} \ & g_1(x) = -x_2x_3 - 0.55 \leq 0, \\
& g_2(x) = -x_3 + x_4 - 0.55 \leq 0, \\
& h_3(x) = 1000\sin(-x_3 - 0.25) + 1000\sin(-x_4 - 0.25) + 894.8 - x_1 = 0, \\
& h_4(x) = 1000\sin(x_3 - 0.25) + 1000\sin(x_3 - x_4 - 0.25) + 894.5 - x_2 = 0, \\
& h_5(x) = 1000\sin(x_4 - 0.25) + 1000\sin(x_4 - x_3 - 0.25) + 1294.8 = 0,
\end{aligned}$$

where $0 \leq x_1, x_2 \leq 1200$, $-0.55 \leq x_3, x_4 \leq 0.55$. The best known solution $x^* = (679.9452, 1026.067, 0.1188764, -0.3962336)$, and $f(x^*) = 5126.4981$.

g06[30]

$$\min \ f(x) = (x_1 - 10)^3 + (x_2 - 20)^3$$

$$\begin{aligned} \text{s.t.} \quad & g_1(x) = -(x_1-5)^2-(x_2-5)^2+100 \le 0, \\ & g_2(x) = (x_1-6)^2-(x_2-5)^2-82.81 \le 0, \end{aligned}$$

where $13 \le x_1 \le 100$, $0 \le x_2 \le 100$. The best known solution $x^* = (14.095, 0.84296)$, and $f(x^*) = 6961.81388$.

g07[34]

$$\begin{aligned} \min \quad & f(x) = x_1^2+x_2^2+x_1x_2-14x_1-16x_2+(x_3-10)^2+4(x_4-5)^2+(x_5-3)^2 \\ & \quad +2(x_6-1)^2+5x_7^2+7(x_8-11)^2+2(x_9-10)^2+(x_{10}-7)^2+45 \\ \text{s.t.} \quad & g_1(x) = -105+4x_1+5x_2-3x_7+9x_8 \le 0, \\ & g_2(x) = 10x_1-8x_2-17x_7+2x_8 \le 0, \\ & g_3(x) = -8x_1+2x_2+5x_9-2x_{10}-12 \le 0, \\ & g_4(x) = 3(x_1-2)^2+4(x_2-3)^2+2x_3^2-7x_4-120 \le 0, \\ & g_5(x) = 5x_1^2+8x_2+(x_3-6)^2-2x_4-40 \le 0, \\ & g_6(x) = x_1^2+2(x_2-2)^2-2x_1x_2+14x_5-6x_6 \le 0, \\ & g_7(x) = 0.5(x_1-8)^2+2(x_2-4)^2+3x_5^2-x_6-30 \le 0, \\ & g_8(x) = -3x_1+6x_2+12(x_9-8)^2-7x_{10} \le 0, \end{aligned}$$

where $-10 \le x_i \le 10$ for $i = 1, 2, \cdots, 10$. The global optimum is $x^* = (2.171996, 2.363683, 8.773926, 5.095984, 0.9906548, 1.430574, 1.321644, 9.828726, 8.280092, 8.375927)$ and $f(x^*) = 24.3062091$.

g08[31]

$$\begin{aligned} \min \quad & f(x) = \frac{\sin^3(2\pi x_1)\sin(2\pi x_2)}{x_1^3(x_1+x_2)} \\ \text{s.t.} \quad & g_1(x) = x_1^2-x_2+1 \le 0, \\ & g_2(x) = 1-x_1+(x_2-4)^2 \le 0, \end{aligned}$$

where $0 \le x_1, x_2 \le 10$, $0 \le x_2 \le 100$. The best solution is $x^* = (1.2279713, 4.2453733)$ and $f(x^*) = 0.095825$.

g09[34]

$$\begin{aligned} \min \quad & f(x) = (x_1-10)^2+5(x_2-12)^2+x_3^4+3(x_4-11)^2+10x_5^6 \\ & \quad +7x_6^2+x_7^4-4x_6x_7-10x_6-8x_7 \\ \text{s.t.} \quad & g_1(x) = -127+2x_1^2+3x_2^4+x_3+4x_4^2-15x_5 \le 0, \\ & g_2(x) = -282+7x_1+3x_2+10x_3^2+x_4-x_5 \le 0, \\ & g_3(x) = -196+23x_1+x_2^2+6x_6^2-8x_7 \le 0, \\ & g_4(x) = 4x_1^2+x_2^2-3x_1x_2+2x_3^2+5x_6-11x_7 \le 0, \end{aligned}$$

where $-10 \le x_i \le 10$ for $i = 1 \sim 7$. The optimum is $x^* = (2.330499, 1.951372, -0.4775414, 4.365726, -0.6244870, 1.038131, 1.594227)$, and $f(x^*) = 680.6300573$.

g10[34]

$$\min \quad f(x) = \sum_{i=1}^{3} x_i$$

$$\begin{aligned}
\text{s.t.} \quad & g_1(x) = -1 + 0.0025(x_4 + x_6) \leq 0, \\
& g_2(x) = -1 + 0.0025(x_5 + x_7 - x_4) \leq 0, \\
& g_3(x) = -1 + 0.01(x_8 - x_5) \leq 0, \\
& g_4(x) = -x_1 x_6 + 833.33252 x_4 + 100 x_1 - 83333.333 \leq 0, \\
& g_5(x) = -x_2 x_7 + 1250 x_5 + x_2 x_4 - 1250 x_4 \leq 0, \\
& g_6(x) = -x_3 x_8 + 1250000 + x_3 x_5 - 2500 x_5 \leq 0,
\end{aligned}$$

where $100 \leq x_1 \leq 10000$, $1000 \leq x_i \leq 100000$ $(i = 2, 3)$, and $10 \leq x_i \leq 1000$ $(i = 4 \sim 8)$, The optimum is $x^* = (579.3167, 1359.943, 5110.071, 182.0174, 295.5985, 217.9799, 286.4162, 395.5979)$, and $f(x^*) = 7049.3307$.

g11[31]

$$\begin{aligned}
\min \quad & f(x) = x_1^2 + (x_2 - 1)^2 \\
\text{s.t.} \quad & h(x) = x_2 - x - 1^2 = 0,
\end{aligned}$$

where $-1 \leq x_1, x_2 \leq 1$. The optimum is $x^* = (\pm\frac{1}{\sqrt{2}}, \frac{1}{2})$, and $f(x^*) = 0.75$.

g12[31]

$$\begin{aligned}
\max \quad & f(x) = \frac{100 - (x_1 - 5)^2 - (x_2 - 5)^2 - (x_3 - 5)^2}{100} \\
\text{s.t.} \quad & g(x) = (x_1 - p)^2 + (x_2 - q)^2 + (x_3 - r)^2 - 0.0625 \leq 0,
\end{aligned}$$

where $0 \leq x_i \leq 10$ for $i = 1, 2, 3$ and $p, q, r = 1, 2, \cdots, 9$. The feasible region of the search space consists of 9^3 disjointed spheres. A point (x_1, x_2, x_3) is feasible iff there exist such that the above inequality holds. The optimum is located at $x^* = (5, 5, 5)$ within the feasible region and $f(x^*) = 1$.

g13[34]

$$\begin{aligned}
\min \quad & f(x) = \mathrm{e}^{x_1 x_2 \cdots x_5} \\
\text{s.t.} \quad & h_1(x) = x_1^2 + x_2^2 + x_3^2 + x_4^2 + x_5^2 - 10 \leq 0, \\
& h_2(x) = x_2 x_3 - 5 x_4 x_5 = 0, \\
& h_3(x) = x_1^3 + x_2^3 + 1 = 0,
\end{aligned}$$

where $-2.3 \leq x_i \leq 2.3$ for $i = 1, 2$ and $-3.2 \leq x_i \leq 3.2$ for $i = 3, 4, 5$. The optimum is located at $x^* = (-1.717143, 1.595709, 1.827247, 0.7636413, 0.763645)$ and $f(x^*) = 0.0539498$.

As can be seen from Table 2, for the test problems without equality constraints (g01, g02, g04, g06, g07, g08, g09, g10, and g12), the proposed algorithm MICA can enter the feasible region within 1500 generations; for test functions g03 and g11, and the proposed algorithm can enter the feasible region within 6000 generations. Although for functions g03 and g13, MICA can enter the feasible region within 6000 and 9000 generations, respectively; however, after 3000 generations, the best individual of the imperialist swarm has had very little distance to the boundaries of the feasible region.

Table 1 Average percentage of feasible countries in the final country swarm with 30 independent runs

Test problem	g01	g02	g03	g04	g05	g06	g07	g08	g09	g10	g11	g12	g13
Average percentage	100	95	59	89	100	62	78	92	85	78	86	98	57

Table 2 Average distance from the best individual of the imperialist swarm to the boundaries of the feasible region at every 1500 generations for the 30 runs

	I. N.									
Fuc.	1500	3000	4500	6000	7500	9000	10500	12000	13500	15000
g01	0	0	0	0	0	0	0	0	0	0
g02	0	0	0	0	0	0	0	0	0	0
g03	6.21×10^{-3}	4.42×10^{-1}	5.92×10^{-13}	0	0	0	0	0	0	0
g04	0	0	0	0	0	0	0	0	0	0
g05	3.28×10^{-6}	1.12×10^{-4}	0	0	0	0	0	0	0	0
g06	0	0	0	0	0	0	0	0	0	0
g07	0	0	0	0	0	0	0	0	0	0
g08	0	0	0	0	0	0	0	0	0	0
g09	0	0	0	0	0	0	0	0	0	0
g10	0	0	0	0	0	0	0	0	0	0
g11	2.86×10^{-14}	1.85×10^{-5}	2.72×10^{-17}	0	0	0	0	0	0	0
g12	0	0	0	0	0	0	0	0	0	0
g13	3.24×10^{-12}	5.32×10^{-11}	1.54×10^{-7}	2.34×10^{-14}	3.35×10^{-6}	0	0	0	0	0

It can be seen from Table 3, our algorithm MICA can find a better "best" result, compared with the other five algorithms OICA, SAEA, SMEA, RCEA and ISEA, in four functions g02, g07, g10 and g13. In addition, algorithm MICA found a similar best solution in five problems g01, g03, g06, g08, g11, and g12 (ISEA didn't give the results for g12). Slightly better best results were found by MICA in the remaining functions g04, g05, g06, and g09 (in fact, our algorithm obtained a similar best solution in g04 and g06 along with the compared three algorithms SMEA, RCEA and ISEA). Our approach found better mean and worst results in seven test functions g02, g05, g06, g07, g09, g10, and g10 except the compared algorithm ISEA for test g02 does. It also provided similar mean and "worst" results in six functions g01, g03, g04, g08, g11, and g12. Also, the proposed algorithm obtained the slightly "worse" mean in test functions g01, g08, g12 and g13 for RCEA, and in g02 for the compared algorithm SMEA and ISEA.

Table 3 Comparison of the proposed algorithm MICA with respect to OICA[19], SAEA[9], SMES[16], RCEA[17], ISEA[26] on 13 benchmark functions. "NA" presents not results

Function	Optimal	Status	Methods					
			OICA[19]	SAEA[9]	SMEA[16]	RCEA[17]	ISEA[26]	MICA
g01	−15.000	best	−15.000	−15.000	−15.000	−15.000	−15.000	−15.000
		mean	−15.000	−15.000	−15.000	−15.000	−14.494	−15.000
		worst	−15.000	−15.000	−15.000	−15.000	−12.446	−15.000
		std.	−15.000	0.0000	0.0000	0.0	9.3×10^{-1}	1.3×10^{-11}
g02	−0.803619	best	−0.80342	−0.80297	−0.803601	−0.803515	−0.803376	−0.803619
		mean	−0.79212	−0.79010	−0.785238	−0.781975	−0.798231	−0.793421
		worst	−0.76213	−0.76043	−0.751322	−0.726288	−0.768291	−0.783461
		std.	1.5×10^{-3}	1.2×10^{-2}	1.7×10^{-2}	2.0×10^{-2}	9.0×10^{-3}	2.5×10^{-2}
g03	−1.0000	best	−1.0000	−1.0000	−1.0000	−1.0000	−1.0000	−1.0000
		mean	−1.0000	−1.0000	−1.0000	−1.0000	−1.0000	−1.0000
		worst	−1.0000	−1.0000	−1.0000	−1.0000	−1.0000	−1.0000
		std.	6.4×10^{-3}	7.5×10^{-5}	2.1×10^{-4}	1.9×10^{-4}	9.7×10^{-5}	2.3×10^{-12}
g04	−30665.539	best	−30665.405	−30665.500	−30665.539	−30665.539	−30665.539	−30665.539
		mean	−30665.531	−30665.200	−30665.539	−30665.539	−30665.539	−30665.539
		worst	−30665.523	−30665.300	−30665.539	−30665.539	−30665.539	−30665.539
		std.	0.0000	4.9×10^{-1}	0.0000	2.0×10^{-5}	0.0000	7.2×10^{-10}
g05	5126.4981	best	5126.964	5126.989	5126.599	5126.497	NA	5126.4981
		mean	5432.080	5432.080	5174.492	5128.881	NA	5126.4981
		worst	5883.950	6089.430	5304.167	5142.472	NA	5126.4981
		std.	3.3×10^{5}	3.9×10^{3}	5.0×10	3.5	NA	1.512×10^{-10}
g06	−6961.81388	best	−6961.800	−6961.800	−6961.814	−6961.814	−6961.814	−6961.814
		mean	−6961.800	−6961.800	−6961.284	−6875.940	−6961.813	−6961.814
		worst	−6961.800	−6961.800	−6952.482	−6350.262	−6961.812	−6961.814
		std.	0.0000	0.0000	1.9	1.6×10^{2}	8.5×10^{-5}	1.21×10^{-10}
g07	24.3062091	best	24.47	24.48	24.327	24.307	24.338	24.3062
		mean	25.38	26.58	24.475	24.374	24.527	24.3457
		worst	28.32	28.40	24.843	24.642	24.995	24.3812
		std.	1.2×10	1.1	1.3×10^{-1}	6.6×10^{-2}	1.7×10^{-1}	2.53×10^{-9}
g08	0.095825	best	0.095825	0.095825	0.095825	0.095825	0.095825	0.095825
		mean	0.095825	0.095825	0.095825	0.095825	0.095825	0.095825
		worst	0.095825	0.095825	0.095825	0.095825	0.095825	0.095825
		std.	0.0000	0.0000	0.0000	2.6×10^{-17}	0.0000	3.21×10^{-14}
g09	680.6300573	best	680.64	680.64	680.632	680.630	680.630	680.630
		mean	680.70	680.72	680.643	680.656	680.631	680.630
		worst	680.83	680.87	680.719	680.763	680.634	680.630
		std.	5.3	5.9×10^{-2}	1.6×10^{-2}	3.4×10^{-2}	8.1×10^{-4}	4.2×10^{-9}
g10	7049.3307	best	7051.31	7061.34	7051.903	7054.316	7062.019	7049.330
		mean	7625.87	7627.89	7253.047	7559.192	7342.944	7049.330
		worst	8187.54	8288.79	7638.366	8835.655	7588.054	7049.330
		std.	3.4×10	3.7×10^{2}	1.4×10^{2}	5.3×10^{2}	1.4×10^{2}	1.1×10^{-9}
g11	0.7500	best	0.750	0.750	0.750	0.750	0.750	0.750
		mean	0.750	0.750	0.750	0.750	0.750	0.750
		worst	0.750	0.750	0.750	0.750	0.751	0.750
		std.	0.0000	0.0000	1.5×10^{-4}	8.0×10^{-5}	2.6×10^{-4}	5.31×10^{-8}
g12	−1.0000	best	−1.0000	−1.0000	−1.0000	−1.0000	NA	−1.0000
		mean	−1.0000	−1.0000	−1.0000	−1.0000	NA	−1.0000
		worst	−1.0000	−1.0000	−1.0000	−1.0000	NA	−1.0000
		std.	0.0000	0.0000	0.0000	0.0000	NA	6.8×10^{-12}
g13	0.0539498	best	0.053997	NA	0.053986	0.053957	0.05517	0.053949
		mean	0.066531	NA	0.166385	0.067543	0.28184	0.025432
		worst	0.097569	NA	0.468294	0.216915	0.471	0.043957
		std.	1.6×10^{-2}	NA	1.8×10^{-1}	3.1×10^{-2}	1.8×10^{-1}	1.5×10^{-1}

The compared results in Table 2 verifies that MICA has the capability in convergence rate, and the compared results in Table 3 reflects the fact that our algorithm is capable of performing a robust and stable search. Furthermore, feasible solutions are consistently found for all test problems in Table 1. The above observations validate that the proposed algorithm MICA has substantial potential in coping with various nonlinear constrained optimization problems.

6 Conclusions

This paper introduces a new imperialist competitive algorithm (MICA) for solving nonlinear constrained optimization problem. The proposed algorithm has three important characterizes: 1) Combining multiobjective optimization with local search models; 2) To achieve a better balance of the exploration and exploitation through the method of exchanging positions of the imperialist and colony; 3) Speeding up the convergence by taking advantage of a new local search method. Based on the comparision between the proposed algorithm and the five compared algorithms, it is concluded our algorithm NICA has shown its potential to handle various nonlinear constrained optimization problems, and its performance is much better than all other state-of-the-art evolutionary algorithms referred in this paper in terms of the selected performance metrics.

An important subject of ongoing work is of applying our approach to the solution of real-world optimization problems. Additionally, try to design different global and local search models since suitable search model can improve the capability of the algorithm remarkably. Last, we aim to explore the possibility of decreasing its computational cost after reaching the feasible region.

References

[1] Homaifar A, Lai S H Y, Qi X. Constrained optimization via genetic algorithms. Simulation, 1994, 62(4): 242–254.

[2] Coello C A C. Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art. Comput. Methods Appl. Mech. Eng., 2002, 191(11–12): 1245–1287.

[3] Hamida S B, Schoenauer M. ASCHEA: New results using adaptive segregational constraint handling. Proc Congr Evol Comput, IEEE Press, 2002.

[4] Graham C G, Mará M S, José A, et al. Constrained control and estimation — An optimisation approach. Springer-Verlag, London, 2005.

[5] Özgür Y. Penalty function methods for constrained optimization with genetic algorithms. Mathematical and Computational Applications, 2005, 10(1): 45–56.

[6] Dimitri B. Constrained optimization and lagrange multiple methods. Academic Press, USA, 1982.

[7] Joshi H, Arora S. Enhanced grey wolf optimization algorithm for global optimization. Fundamenta Informaticae, 2017, 153(3): 235–264.

[8] Miranda-Varela M E, Mezura-Montes E. Constraint-handling techniques in surrogate-assisted evolutionary optimization: An empirical study. Applied Soft Computing Journal, 2018, 73: 215–229.

[9] Arora S, Singh S, Yetilmezsoy K. A modified butterfly optimization algorithm for mechanical design optimization problems. Journal of the Brazilian Society of Mechanical Sciences and Engineering, 2018, 40(21): 173–185.

[10] Kohli M, Arora S. Chaotic grey wolf optimization algorithm for constrained optimization problems. Journal of Computational Design and Engineering, 2018, 5(4): 458–472.

[11] Joshi H, Arora S. Enhanced grey wolf optimisation algorithm for constrained optimisation problems. International Journal of Swarm Intelligence, 2017, 3(2/3): 126–151.

[12] Ion N. A cutting plane method for solving convex optimization problems over the cone of nonnegative polynomials. WSEAS Transactions on Mathematics, 2009, 8(7): 269–279.
[13] Paul H C, Jorge J M. Projected gradient methods for linearly constrained problems. Mathematical Programming, 1987, 39: 93–16.
[14] Philipp H, Martin K. Quasi-Newton methods: A new direction. Journal of Machine Learning Research, 2013, 14: 843–865.
[15] Yuan Y X. A new stepsize for the steepest descent method. Journal of Computational Mathematics, 2006, 24(2): 149–156.
[16] Hovd M. Multi-level programming for designing penalty functions for MPC controllers. Proc of the 18th IFAC World Congress. Milano, Italy: IEEE Press, 2011.
[17] Huang F Z, Wang L, He Q. An effective co-evolutionary differential evolution for constrained optimization. Applied Mathematics and Computation, 2007, 186(1): 240–256.
[18] He Q, Wang L. A hybrid particle swarm optimization with a feasibility-based rule for constrained optimization. Applied Mathematics and Computation, 2007, 186(2): 1407–1422.
[19] Lin Y C, Wang F S, Hwang K S. A hybrid method of evolutionary algorithms for mixed-integer nonlinear optimization problems. Proceedings of the 1999 Congress on Evolutionary Computation, Washington, IEEE Press, 1999.
[20] Atashpaz-Gargari E, Lucas C. Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition. IEEE Congress on Evolutionary Computation, 2007: 4661–4667.
[21] Mahdi A, Ayaz I, Davoud A. Imperialist competitive algorithm for solving systems of nonlinear equations. Computers and Mathematics with Applications, 2013, 65: 1894–1908.
[22] Mohammadi M, Tavakkoli-Moghaddam R, Rostamib H. A multi-objective imperialist competitive algorithm for a capacitated hub covering location problem. International Journal of Industrial Engineering Computations, 2011, 2: 671–688.
[23] Shokrollahpour E, Zandieh M, Dorri B. A novel imperialist competitive algorithm for bi-criteria scheduling of the assembly flowshop problem. International Journal of Production Research, 2011, 49(11): 3087–3103.
[24] Long W, Jiao J J, Liang X M, et al. An exploration-enhanced grey wolf optimizer to solve high-dimensinal numerical optimization. Engineering Applications of Artificial Intelligence, 2018, 68: 63–80.
[25] Farmani R, Wright J A. Self-adaptive fitness formulation for constrained optimization. IEEE Trans Evol Comput, 2003, 7(5): 445–455.
[26] Mezura-Montes E, Coello C A C. A simple multimembered evolution strategy to solve constrained optimization problems. IEEE Trans Evol Comput, 2005, 9(1): 1–17.
[27] Runarsson T P, Yao X. Stochastic ranking for constrained evolutionary optimization. IEEE Trans Evol Comput, 2000, 4(3): 284–294.
[28] Aguirre A H, Rionda S B, Coello C A C, et al. Handling constraints using multiobjective optimization concepts. Int J Numerical Methods in Eng, 2004, 59(15): 1989–2017.
[29] Datta R, Regis R. A surrogate-assisted evolution strategy for constrained multi-objectiveoptimization. Expert Syst Appl, 2016, 57: 270–284.
[30] Floundas C, P. Pardalos. A collection of test problems for constrained global optimization. Lecture Notes in Comput Sci. Berlin, Germany: Springer-Verlag, 1987.
[31] Koziel S, Michalewicz Z. Evolutionary algorithms, homomorphous mappings, and constrained parameter optimization. Evol Comput, 1999, 7(1): 19–44.
[32] Michalewicz Z, Nazhiyath G, Michalewicz M. A note on usefulness of geometrical crossover for numerical optimization problems. Proc 5th Annu Conf Evolutionary Programming. Fogel L J, Angeline P J, Bäck T. Cambridge, MA: MIT Press, 1996.
[33] Himmelblau D. Applied nonlinear programming. New York: McGraw-Hill, IEEE Press, 1972.
[34] Hock W, Schittkowski K. Test examples for nonlinear programming codes. Berlin, Germany: Springer-Verlag, 1981.