Journal of Zhejiang University-SCIENCE C (Computers & Electronics)
ISSN 1869-1951 (Print); ISSN 1869-196X (Online)
www.zju.edu.cn/jzus; www.springerlink.com
E-mail: jzus@zju.edu.cn

# FICA: fuzzy imperialist competitive algorithm

Saeid ARISH, Ali AMIRI‡, Khadije NOORI

(*Department of Computer Engineering, University of Zanjan, Zanjan, Iran*)

E-mail: saeed.aiproject@gmail.com; a_amiri@znu.ac.ir; kh.noori90@gmail.com

Received Apr. 11, 2013; Revision accepted Dec. 6, 2013; Crosschecked Apr. 11, 2014

**Abstract:** Despite the success of the imperialist competitive algorithm (ICA) in solving optimization problems, it still suffers from frequently falling into local minima and low convergence speed. In this paper, a fuzzy version of this algorithm is proposed to address these issues. In contrast to the standard version of ICA, in the proposed algorithm, powerful countries are chosen as imperialists in each step; according to a fuzzy membership function, other countries become colonies of all the empires. In absorption policy, based on the fuzzy membership function, colonies move toward the resulting vector of all imperialists. In this algorithm, no empire will be eliminated; instead, during the execution of the algorithm, empires move toward one point. Other steps of the algorithm are similar to the standard ICA. In experiments, the proposed algorithm has been used to solve the real world optimization problems presented for IEEE-CEC 2011 evolutionary algorithm competition. Results of experiments confirm the performance of the algorithm.

**Key words:** Optimization problem, Imperialist competitive algorithm (ICA), Fuzzy ICA.
**doi:**10.1631/jzus.C1300088 **Document code:** A **CLC number:** TP301.6

## 1 Introduction

A wide variety of problems in different sciences can be formulated as an optimization problem. Consequently, finding an appropriate solution for optimization problems has great importance in all sciences, especially the engineering fields.

In general, optimization problem solutions can be divided into two categories: classical and heuristic. Some instances of classical methods include analytical optimization, the Lagrangian method (Andreani *et al.*, 2009), the Nelder-Mead method (Nelder and Mead, 1965), and linear and nonlinear optimization (Golban and Nedevschi, 2011). Most of the classical methods are based on gradient zero-crossing of the fitness function (Bertsekas and Gafni, 1983). In complex functions, these methods may fall into local minima. Also, sometimes the fitness function may not admit derivatives. These drawbacks limit the utilization of the classical methods.

Heuristic methods are used in solving common problems of different sciences. These methods attempt to discover a solution without a deep understanding of the problem structure. In other words, the problem seems like a black box for the algorithm. These methods usually act randomly using statistics gained by the samples of the search space or they are based on a model from some natural phenomena or physical processes. These methods include the genetic algorithm (GA) (Guo *et al.*, 2010), simulated annealing (Davis, 1987), particle swarm optimization (PSO) (Kennedy and Eberhart, 1995), ant colony optimization (Brownlee, 2011), hill climbing (Brownlee, 2011), the Monte-Carlo method (Fishman, 1996), the cuckoo algorithm (Rajabioun, 2011), and so on.

ICA, one of the heuristic methods for solving optimization problems, has achieved higher convergence speed in solving problems with many independent variables, compared to PSO and GA (Gargari and Lucas, 2007; Kaveh and Talatahari, 2010). In recent years, several studies have been done to improve the performance of ICA in solving optimization problems. Kaveh and Talatahari (2010) improved ICA

‡ Corresponding author

by adding a random 'perpendicular vector' between the imperialist and the colony in the absorption step. Talatahari *et al.* (2012) improved the convergence speed of ICA using chaotic functions instead of random functions in the absorption step.

Despite all these successful studies in improving ICA, some challenging issues still exist, including (1) falling into local optima, (2) low convergence speed in some problems, and (3) extremely high computational complexity for a large number of independent variables of the function.

To deal with the above challenges, fuzzy ICA (FICA) is proposed. Using this algorithm, each colony is considered a colony of all imperialists with a membership degree. The colonies move in the direction of the resulting vector from the weighted summation of the imperialists' vectors, proportional to the membership function. The weights of these vectors depend on the power of the imperialists. In other words, the imperialist that has more power will absorb colonies with a higher membership degree and has a greater effect on the moving direction of the colonies. This makes all colonies move coordinately toward the current most optimal direction and increases their focus on searching in one direction as the most optimal direction (instead of focusing on several directions in ICA). Therefore, the convergence speed increases and the optimal solution is reached sooner. Since moving toward one direction increases the possibility of falling into local optima, in each time period of the fuzzy method, the colony and the imperialist among the countries are selected from the very first. Moreover, the bias of the algorithm is reduced to the initial talented points because all initial points or initial countries have been selected randomly, so the probability that their talented points are local minima (especially in complex functions with a lot of local minima) is higher than the probability that the points are selected as talented points in further steps.

In FICA, no imperialist is eliminated as time increases; all imperialists move toward an optimum imperialist and finally converge at one imperialist, which indicates the optimum solution.

The proposed algorithm is assessed on solving the real world optimization problems presented for the IEEE-CEC 2011 evolutionary algorithm competition. Experimental results confirm the performance of the algorithm.

## 2 ICA and its extensions

In this section, the original ICA and some of its extensions are studied. It is assumed that an arbitrary optimization function with $n$ independent variables is given and its output represents the amount of cost. The goal is to find a point in an $n$-dimensional space such that by giving it to the optimization function, the minimum cost will be achieved.

### 2.1 Original ICA

In ICA (Gargari and Lucas, 2007), to solve the assumed optimization problem, $N$ countries are considered. Each country is represented by a vector which indicates a spot in the $n$-dimensional solution space. Among these spots, those having the minimum cost according to the fitness function are considered the imperialists and the rest are colonies.

For each imperialist, first the normalized cost is calculated:

$$\mathrm{NOC}_n = \max_{i\in\{1,2,\ldots,N_{\mathrm{imp}}\}}(c_i) - c_n, \tag{1}$$

where $c_i$ is the cost of the $i$th imperialist and $C_n$ is the normalized cost of the $n$th imperialist. After this step, colonies should be selected. The imperialist that has more power attracts more colonies. Eq. (2) describes this operation:

$$\mathrm{NC}_n = \mathrm{Round}\left(\left|\mathrm{NOC}_n \Big/ \sum_{j=1}^{N_{\mathrm{imp}}} \mathrm{NOC}_j\right| \cdot N_{\mathrm{col}}\right), \tag{2}$$

where $\mathrm{NC}_n$ represents the number of colonies for the $n$th imperialist, $N_{\mathrm{imp}}$ shows the number of imperialists, and $N_{\mathrm{col}}$ is the number of all colonies.

Next is the absorption step, in which each imperialist tries to absorb its own colonies. Colonies move in the direction of their imperialist's vectors with a random deviation. The following formula describes this movement:

$$\boldsymbol{x} = U(0, \beta d)\cdot \boldsymbol{V}_1, \quad \mathbf{col}(t+1) = \mathbf{col}(t) + \boldsymbol{x}, \tag{3}$$

where $U(0, \beta d)$ is a function that generates a random variable with a uniform distribution in $[0, \beta d]$. $\boldsymbol{V}_1$ is the vector between the imperialist and the colony, $d$ is the distance between the imperialist and the colony,

and $\beta$ is the control factor which usually has a value of 2 (Fig. 1).

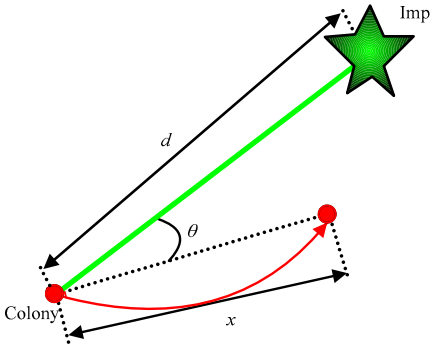


**Fig. 1 The way colonies move toward imperialists in ICA (Gargari and Lucas, 2007)**

To expand the search region around the imperialist, a deviation angle $\theta$ which follows a random number generator with a uniform distribution is added to the main vector:

$$\theta \sim U(-\gamma, \gamma), \tag{4}$$

where $\gamma$ is the controlling limit of the angle deviation. In experiments, it is considered in the limit of $\pi/4$ (Gargari and Lucas, 2007).

The next step is imperialist competition. In this step, a colony from the weakest empire is selected and given to a powerful empire (not necessarily the most powerful one). The more powerful an empire, the higher its probability of being selected.

To model this truth in ICA, first of all, a total power parameter is calculated for each empire. This parameter is more dependent on the power of the imperialist of the empire considered, and power of the colonies has little effect on it. To calculate the total power of the empire, the total cost of the empire is calculated:

$$\mathrm{TC}_n = c_n + \xi \cdot \mu_n, \tag{5}$$

where $\mathrm{TC}_n$ is the total cost of the $n$th empire, $\xi$ is a number less than 1, and $\mu_n$ is the mean of all colonies' costs of the $n$th empire. The cost function calculates the amount of the evaluation function. After computing the total cost, the normalized power of the $n$th empire is calculated:

$$\mathrm{NTC}_n = \mathrm{TC}_n - \max_{i\in\{1,2,\ldots,N_{\mathrm{imp}}\}}\{\mathrm{TC}_i\}. \tag{6}$$

After calculating the normalized power, the weakest colony from the empire that has the least total power ($\mathrm{NTC}_n$) is selected, called $C_{\mathrm{weak}}$. This colony is given to one of the empires based on a probability proportional to the power of the empires. The probability of owing $C_{\mathrm{weak}}$ by each empire is calculated as follows:

$$P_{p_n} = \left| \mathrm{NTC}_n \Big/ \sum_{i=1}^{N_{\mathrm{imp}}} \mathrm{NTC}_i \right|. \tag{7}$$

Finally, when an empire loses all its colonies, it is removed from the empire list and in an imperialist competition will be given to other empires as a colony.

## 2.2 Orthogonal imperialist competition algorithm (OICA)

OICA (Kaveh and Talatahari, 2010) is an extension of ICA in which the absorption process is modified. In this algorithm, a new vector orthogonal to the perpendicular vector between the imperialist and the colony is used. This vector is summed with the vector between the imperialist and the colony (Fig. 2):

$$\begin{cases} \mathbf{col}(t+1) = \mathbf{col}(t) + U(0, \beta d)\cdot \boldsymbol{V}_1 \\ \qquad\qquad + U(-1,+1)\cdot \tan\theta \cdot d \cdot \boldsymbol{V}_2, \\ \boldsymbol{V}_1 \cdot \bar{\boldsymbol{V}}_2 = 0, \quad \left\|\bar{\boldsymbol{V}}_2\right\| = 1, \end{cases} \tag{8}$$

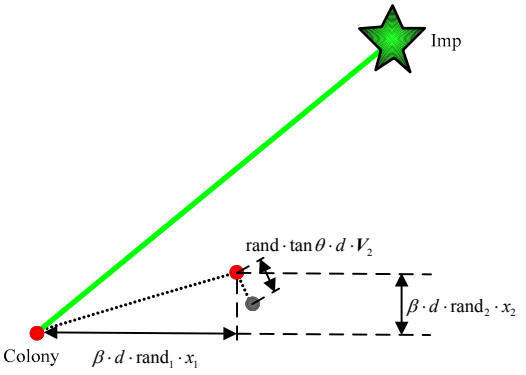


**Fig. 2 The way colonies move toward imperialists in OICA (Kaveh and Talatahari, 2010)**

where $\boldsymbol{V}_1$ is started from the previous position of the colony and points to the position of the imperialist and $\boldsymbol{V}_2$, which is perpendicular to $\boldsymbol{V}_1$, is added to the equation to increase the search region. $\theta$ is a random variable with a uniform distribution $U(-\gamma, \gamma)$, and $\gamma$ determines the limit of the random changes of $\theta$.

### 2.3 Chaotic imperialist competition algorithm (CICA)

An extension of OICA was presented in Talatahari *et al.* (2012), in which chaotic functions, instead of random number generator functions, were used with a uniform probability distribution. Thus, the amount of disorders and oscillations where these functions are needed has been increased. This reduces the probability of falling into local optima and increases the convergence speed because of more disorder in the random functions that expand the search region.

In this algorithm, to add the chaotic functions, a table named cm which contains different values of the chaotic function is created and used instead of the random probability function.

$$\begin{cases}\mathbf{col}(t+1)=\mathbf{col}(t)+\beta\cdot d\cdot \mathrm{cm}\cdot \boldsymbol{V}_1+\mathrm{cm}\cdot\tan\theta\cdot\left|\boldsymbol{V}_2\right|\cdot\boldsymbol{V}_2,\\ \boldsymbol{V}_1\cdot\boldsymbol{V}_2=0,\quad \left\|\boldsymbol{V}_2\right\|=1.\end{cases} \tag{9}$$

## 3 Fuzzy imperialist competition algorithm (FICA)

ICA is based on the belief that each colony has one imperialist and gradually all empires are eliminated until one dominant empire remains in the world. This belief was applicable in the past as the growth of communication was not so fast. Today, with surprising growth of technologies and transformation of the world into a global village, powerful countries do not need military or political dominance to subjugate a weaker country. Rather, by using different tools like satellites and the Internet, they can influence all the countries of the world and apply their own matching policies.

It is shown that all the imperialists, according to their power, dominate weaker countries and apply their matching policies there. Also, the number of imperialists does not decrease; rather, each country that surpasses others in development can take the place of the prior powerful countries.

The other point that should be considered is that as time passes, all countries, even those that influence weaker countries, seem like a country that has the most power.

In this study, this idea is implemented using fuzzy logic. After the selection of imperialists and colonies, colonies move toward the empires according to an absorption operator based on the fuzzy approach. Then, countries that have reached the best optimal properties are considered as imperialists and the others as colonies. In other words, in each step, powerful colonies take the place of weak imperialists. After that, a revolution operator is applied to the colonies. Finally, this change and improvement loop continues till the distance between the imperialists becomes less than a threshold. In this step, the most powerful empire is the solution of the problem. What follows is a detailed description of the algorithm.

In this algorithm, each imperialist is considered a fuzzy set and the colonies belong to these fuzzy sets as members by a specific membership function. Unlike the standard ICA, in each iteration of the algorithm, without considering the previous imperialist, the most powerful imperialists are selected as new imperialists and take the place of the previous ones. Then, based on the new imperialists, colonies are formed. More precisely, assume that $\mathrm{Imp}_1$, $\mathrm{Imp}_2$, …, $\mathrm{Imp}_{N_{\mathrm{imp}}}$ are the imperialists and $\mathrm{Col}_1$, $\mathrm{Col}_2$, …, $\mathrm{Col}_m$ are the colonies. First of all, in the fuzzification step, the membership degree of the colonies to imperialists is determined. To do so, the membership function that is in an S-shaped form is used, defined as

$$\mu_j(\mathrm{col}_i)=\begin{cases}0, & f_j(\mathrm{col}_i)\le\alpha,\\ 2\left(\dfrac{f_j(\mathrm{col}_i)-\alpha}{\gamma-\alpha}\right)^2, & \alpha<f_j(\mathrm{col}_i)\le\beta,\\ 1-2\left(\dfrac{f_j(\mathrm{col}_i)-\alpha}{\gamma-\alpha}\right)^2, & \beta<f_j(\mathrm{col}_i)\le\gamma,\\ 1, & f_j(\mathrm{col}_i)>\gamma.\end{cases} \tag{10}$$

This function is plotted in Fig. 3, where the effects of different parameters $\beta$, $\alpha$, and $\gamma$ on the function are shown.

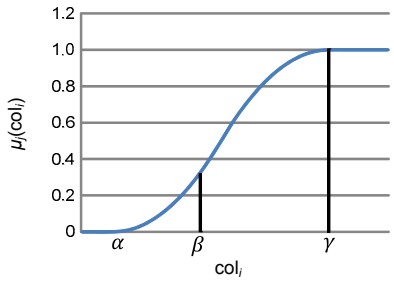


**Fig. 3 The effects of parameters $\alpha$, $\beta$, and $\gamma$ on the S-shaped function**

In Eq. (10), function $f_j$ is defined as follows:

$$f_j(\mathrm{col}_i)=\frac{\mathrm{NOC}_j}{\sum_{k=1}^{N_{\mathrm{imp}}}\mathrm{NOC}_k}\left(1-\frac{\left|\mathrm{col}_i(t)-\mathrm{Imp}_j\right|}{\sum_{k=1}^{N_{\mathrm{imp}}}\left|\mathrm{col}_i(t)-\mathrm{Imp}_k\right|}\right), \quad (11)$$

where $\mathrm{NOC}_j$ is the normalized power of the imperialist $\mathrm{Imp}_j$ which is calculated using Eq. (1). According to this equation, the less the $j$th imperialist's cost and the less the distance between the $i$th colony and the $j$th imperialist, the greater the amount of the function will be. To obtain the direction of each colony, the defuzzification step is used:

$$\boldsymbol{v}_i^*=\sum_{j=1}^{N_{\mathrm{imp}}}\left(\mu_j(\mathrm{col}_i)\cdot(\mathrm{col}_i-\mathrm{Imp}_j)\right)\Big/\sum_{j=1}^{N_{\mathrm{imp}}}\mu_j(\mathrm{col}_i), \quad (12)$$

where $\boldsymbol{v}_i^*$ is the vector to which the $i$th colony intends to move. Therefore, the next position of the $i$th colony is calculated as follows:

$$\mathrm{col}_i(t+1)=\mathrm{col}_i(t)+\beta\cdot\boldsymbol{U}\cdot\boldsymbol{v}_i^*, \quad (13)$$

where $\boldsymbol{U}$=diag$\{u(0, 1), u(0, 1), \ldots, u(0, 1)\}$ is a $d\times d$ matrix, $u(0, 1)$ is a function that generates a random number with a uniform distribution, $\beta$ is a constant coefficient that has been considered to be 2 in experiments, and $d$ is the number of independent variables of the function. Vector $\boldsymbol{v}_i^*$ is thus a $d\times 1$ matrix. Multiplying matrix $\boldsymbol{U}$ in vector $\boldsymbol{v}_i^*$, each element of this vector is multiplied in a random number between 0 and 1 and as a result, this multiplication will generate a random deviation of $\theta$. This random deviation is used to expand the search region. Fig. 4 shows this random deviation.

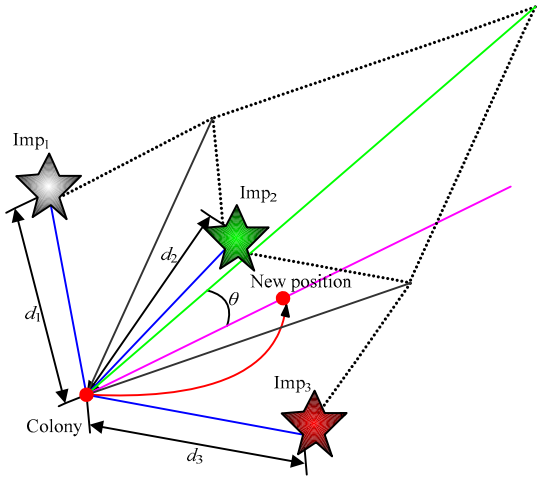


**Fig. 4 The absorption step and colony movement in FICA**

Like ICA, the revolution operator is used to prevent the algorithm from falling into local minima.

### 3.1 Imperialist's replacement

In ICA, if a colony gains less cost than its own imperialist, the colony and the imperialist will be replaced. Now, imagine in one step $m$ colonies reach the more optimum value than their imperialist. In this situation, only the colony that has the most optimum cost has a chance to become an imperialist and the remaining $m-1$ optimum colonies may be eliminated by the revolution operator in the next time frame. Also, a powerful imperialist may have $m$ colonies that are more powerful than $n$ foreign weaker imperialists. In this case, those $m$ colonies may be eliminated due to applying the revolution operator while weaker imperialists will still exist.

To address this issue, in each iteration of the FICA, instead of comparing the colony with its imperialist, among all countries, including colonies and imperialists, countries that have more optimal costs are chosen as imperialists. Figs. 5a and 5b describe this issue. In these figures, numbers inside the countries indicate the power of countries and the smaller the number, the more powerful the country. The pseudo code presented in Fig. 6 explains the way imperialists are chosen in the exchange step of FICA.

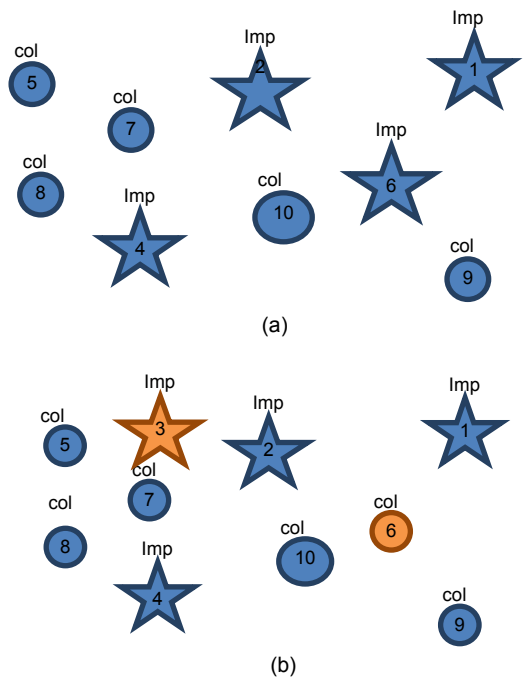


**Fig. 5 Before (a) and after (b) position exchange of powerful colony and weak imperialist**

**Table 1 Problem set definitions**

| Problem | Number of variables | Description |
|---|---|---|
| P01 | 6 | Parameter estimation for frequency-modulated (FM) sound waves |
| P02 | 30 | Lennard-Jones potential problem |
| P03 | 1 | Bifunctional catalyst blend optimal control problem |
| P04 | 1 | Optimal control of a non-linear stirred tank reactor |
| P05 | 30 | Tersoff potential function minimization problem (instance 1) |
| P06 | 30 | Tersoff potential function minimization problem (instance 2) |
| P07 | 20 | Spread spectrum radar polly phase code design |
| P08 | 7 | Transmission network expansion planning problem |
| P09 | 122 | Large scale transmission pricing problem |
| P10 | 12 | Circular antenna array design problem |
| P11.1 | 120 | Dynamic economic dispatch problem, instance 1 |
| P11.2 | 216 | Dynamic economic dispatch problem, instance 2 |
| P11.3 | 6 | Static economic load dispatch problem, instance 1 |
| P11.4 | 13 | Static economic load dispatch problem, instance 2 |
| P11.5 | 15 | Static economic load dispatch problem, instance 3 |
| P11.6 | 40 | Static economic load dispatch problem, instance 4 |
| P11.7 | 140 | Static economic load dispatch problem, instance 5 |
| P11.8 | 96 | Hydrothermal scheduling problem, instance 1 |
| P11.9 | 96 | Hydrothermal scheduling problem, instance 2 |
| P11.10 | 96 | Hydrothermal scheduling problem, instance 3 |
| P12 | 26 | Spacecraft trajectory optimization problem, Messenger |
| P13 | 22 | Spacecraft trajectory optimization problem, Cassini 2 |

```
For i=1 to the number of imperialists
  If the cost of the weakest imperialist is larger than the
    cost of the strongest colony
      Remove the weakest imperialist position from the
        imperialist list and put its position in the colony list
      Remove the strongest colony position from the colony
        list and put its position in the imperialist list
  End If
End For
```

**Fig. 6 Position replacement of colonies and imperialists in FICA**

## 4 Experimental results

The performance of the FICA algorithm was compared with those of different versions of ICA and also GA. The proposed algorithm was implemented in MATLAB on a system with 4 GB RAM, 2.5 GHz Core 5 processor and Windows 7 operating system. For different versions of ICA, the number of countries was considered to be 80 and the number of imperialists was 8. Also, in GA, the number of the initial population was considered to be 100. To assess the efficiency of the algorithm, optimization problems in IEEE-CEC2011 (Das and Suganthan, 2010) were used. All experiments were repeated 25 times. The maximum number of iterations of each algorithm to obtain coverage and obtain the best solution varies from 500 to 5000. Table 1 briefly describes the properties of each problem, Table 2 shows the fitness value of the best solution, and Table 3 gives the mean values from the 25 experiments.

In addition, to assess the results of the experiments, an approach introduced in Das and Suganthan (2010) was used. In this approach, for each experiment, first the mean and minimum cost values of the solutions of the algorithms were ranked from best to worst, and then the sum of the mean and minimum values of the solution of each algorithm was calculated. At last, the averages of the mean and minimum values were calculated for each algorithm and the algorithm that has the minimum average was chosen as the best algorithm and the others were ranked according to their average values.

**Table 2 Best function values achieved**

| Problem | ICA | OICA | COICA | FICA | GA |
|---|---|---|---|---|---|
| P01 | 0.000E+00 | 2.251E+00 | 3.692E-01 | 1.115E-05 | 7.981E+00 |
| P02 | −2.477E+01 | −1.334E+01 | −9.479E+00 | −2.328E+01 | −2.318E+01 |
| P03 | 1.151E-05 | 1.151E-05 | 1.151E-05 | 1.151E-05 | −1.939E-09 |
| P04 | 1.377E+01 | 1.377E+01 | 1.377E+01 | 1.392E+01 | 1.383E+01 |
| P05 | −3.679E+01 | −2.842E+01 | −2.842E+01 | **−3.724E+01** | −7.720E+00 |
| P06 | −2.914E+01 | −1.837E+01 | −1.885E+01 | **−3.034E+01** | 2.395E+01 |
| P07 | 5.899E-01 | 1.372E+00 | 1.565E+00 | 9.719E-01 | 1.079E+00 |
| P08 | 2.200E+02 | 2.200E+02 | 2.200E+02 | 2.200E+02 | 2.200E+02 |
| P09 | 7.253E+04 | 2.716E+05 | 9.274E+05 | **9.813E+03** | 2.538E+06 |
| P10 | −2.146E+01 | −2.003E+01 | −1.775E+01 | **−2.391E+01** | −1.032E+01 |
| P11.1 | 4.743E+05 | 1.955E+08 | 2.203E+08 | **3.444E+05** | 7.851E+06 |
| P11.2 | 2.162E+06 | 1.270E+07 | 1.251E+07 | **3.868E+05** | 7.646E+06 |
| P11.3 | 1.545E+04 | 1.545E+04 | 1.545E+04 | 1.546E+04 | 1.545E+04 |
| P11.4 | 1.877E+04 | 1.951E+04 | 1.943E+04 | **9.627E+03** | 1.881E+04 |
| P11.5 | 3.293E+04 | 4.025E+04 | 1.373E+05 | 3.304E+04 | 5.608E+05 |
| P11.6 | 1.488E+05 | 6.443E+05 | 2.445E+06 | **1.053E+05** | 2.083E+08 |
| P11.7 | 1.937E+06 | 1.257E+08 | 2.489E+08 | **1.192E+06** | 3.410E+09 |
| P11.8 | 9.402E+05 | 1.279E+07 | 2.499E+07 | **1.953E+05** | 6.699E+07 |
| P11.9 | 1.383E+06 | 1.120E+07 | 2.629E+07 | **1.013E+06** | 7.721E+07 |
| P11.10 | 9.482E+05 | 2.644E+06 | 4.401E+06 | **7.237E+05** | 4.225E+08 |
| P12 | 1.826E+01 | 1.469E+01 | 1.959E+01 | 1.457E+01 | 1.551E+02 |
| P13 | 2.037E+01 | 1.539E+01 | 2.908E+01 | 1.245E+01 | 1.204E+02 |

**Table 3 Mean function values achieved**

| Problem | ICA | OICA | COICA | FICA | GA |
|---|---|---|---|---|---|
| P01 | 8.760E+00 | 1.014E+01 | 9.443E+00 | **3.713E+00** | 1.980E+01 |
| P02 | −1.852E+01 | −1.117E+01 | −7.929E+00 | −1.816E+01 | −1.996E+01 |
| P03 | 1.151E-05 | 1.151E-05 | 1.151E-05 | 1.151E-05 | −1.939E-09 |
| P04 | 1.402E+01 | 1.427E+01 | 1.393E+01 | 1.418E+01 | 1.418E+01 |
| P05 | −3.454E+01 | −2.370E+01 | −2.202E+01 | −2.506E+01 | 4.491E+00 |
| P06 | −2.786E+01 | −1.598E+01 | −1.448E+01 | −1.770E+01 | 4.081E+01 |
| P07 | 1.165E+00 | 1.681E+00 | 1.854E+00 | **1.357E+00** | 1.463E+00 |
| P08 | 2.200E+02 | 2.229E+02 | 2.239E+02 | 2.219E+02 | 2.247E+02 |
| P09 | 1.906E+05 | 3.508E+05 | 1.062E+06 | **3.329E+04** | 2.767E+06 |
| P10 | −1.899E+01 | −1.772E+01 | −1.616E+01 | **−2.074E+01** | −1.023E+01 |
| P11.1 | 2.399E+06 | 2.755E+08 | 2.751E+08 | **1.790E+06** | 7.974E+06 |
| P11.2 | 2.945E+06 | 1.381E+07 | 1.395E+07 | **2.092E+06** | 7.752E+06 |
| P11.3 | 1.546E+04 | 1.551E+04 | 1.552E+04 | 1.549E+04 | 1.550E+04 |
| P11.4 | 2.076E+04 | 2.373E+04 | 3.081E+04 | **1.457E+04** | 1.924E+04 |
| P11.5 | 3.306E+04 | 1.428E+05 | 3.138E+05 | 3.322E+04 | 3.159E+06 |
| P11.6 | 3.146E+06 | 5.320E+06 | 5.435E+06 | **3.351E+05** | 2.168E+08 |
| P11.7 | 4.311E+06 | 4.567E+08 | 1.015E+09 | 4.642E+06 | 6.734E+09 |
| P11.8 | 1.966E+06 | 2.506E+07 | 3.484E+07 | 3.102E+06 | 9.707E+07 |
| P11.9 | 4.357E+06 | 2.310E+07 | 3.756E+07 | **3.550E+06** | 1.000E+08 |
| P11.10 | 1.804E+06 | 4.402E+06 | 1.078E+07 | 3.913E+06 | 4.575E+08 |
| P12 | 2.734E+01 | 2.306E+01 | 2.343E+01 | 2.517E+01 | 1.877E+02 |
| P13 | 3.185E+01 | 2.728E+01 | 3.266E+01 | **2.482E+01** | 1.253E+02 |

Table 4 explains the minimum values and the mean values achieved from 25 experiments. Values of the algorithms are numbered from 1 to 5; 1 is the best and 5 is the worst. Table 5 shows a sum of the ranks of the algorithms according to Table 4. It can be seen that FICA has the highest rank among these algorithms.

**Table 4 Ranking the algorithms based on the best solution and the mean of fitness values of solutions achieved from 25 experiments**

| Problem | Rank by minimum | | | | | Rank by mean | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | ICA | OICA | COICA | FICA | GA | ICA | OICA | COICA | FICA | GA |
| P01 | 1 | 4 | 3 | 2 | 5 | 2 | 4 | 3 | 1 | 5 |
| P02 | 1 | 4 | 5 | 2 | 3 | 2 | 4 | 5 | 3 | 1 |
| P03 | 4 | 3 | 5 | 2 | 1 | 3 | 4 | 5 | 2 | 1 |
| P04 | 1 | 2 | 3 | 5 | 4 | 2 | 5 | 1 | 4 | 3 |
| P05 | 2 | 3 | 4 | 1 | 5 | 1 | 3 | 4 | 2 | 5 |
| P06 | 2 | 4 | 3 | 1 | 5 | 1 | 3 | 4 | 2 | 5 |
| P07 | 1 | 4 | 5 | 2 | 3 | 1 | 4 | 5 | 2 | 3 |
| P08 | 1 | 2 | 3 | 4 | 5 | 1 | 3 | 4 | 2 | 5 |
| P09 | 2 | 3 | 4 | 1 | 5 | 2 | 3 | 4 | 1 | 5 |
| P10 | 2 | 3 | 4 | 1 | 5 | 2 | 3 | 4 | 1 | 5 |
| P11.1 | 2 | 4 | 5 | 1 | 3 | 2 | 5 | 4 | 1 | 3 |
| P11.2 | 2 | 5 | 4 | 1 | 3 | 2 | 4 | 5 | 1 | 3 |
| P11.3 | 1 | 4 | 3 | 5 | 2 | 1 | 4 | 5 | 2 | 3 |
| P11.4 | 2 | 5 | 4 | 1 | 3 | 3 | 4 | 5 | 1 | 2 |
| P11.5 | 1 | 3 | 4 | 2 | 5 | 1 | 3 | 4 | 2 | 5 |
| P11.6 | 2 | 3 | 4 | 1 | 5 | 2 | 3 | 4 | 1 | 5 |
| P11.7 | 2 | 3 | 4 | 1 | 5 | 1 | 3 | 4 | 2 | 5 |
| P11.8 | 2 | 3 | 4 | 1 | 5 | 1 | 3 | 4 | 2 | 5 |
| P11.9 | 2 | 3 | 4 | 1 | 5 | 2 | 3 | 4 | 1 | 5 |
| P11.10 | 2 | 3 | 4 | 1 | 5 | 1 | 3 | 4 | 2 | 5 |
| P12 | 3 | 2 | 4 | 1 | 5 | 4 | 1 | 2 | 3 | 5 |
| P13 | 3 | 2 | 4 | 1 | 5 | 3 | 2 | 4 | 1 | 5 |

**Table 5 Comparison of algorithms based on the sum of ranks in Table 4**

| Algorithm | Best | Mean | Overall | Average |
|---|---|---|---|---|
| ICA | 41 | 40 | 81 | 40.5 |
| OICA | 72 | 74 | 146 | 73.0 |
| COICA | 87 | 88 | 175 | 87.5 |
| FICA | **38** | **39** | **77** | **38.5** |
| GA | 92 | 89 | 181 | 90.5 |

## 5 Conclusions

In this paper, a fuzzy ICA (FICA) was proposed to solve ICA problems including frequently falling into local minima and low convergence speed in solving some optimization problems. In each step of FICA, powerful countries are selected as imperialists. Then, other countries become a colony member of all the empires based on a fuzzy membership function. In the absorption policy, based on the fuzzy membership function, colonies move toward the resulting vector of all imperialists. In this algorithm, no empire will be eliminated; instead, during the execution of the algorithm, empires move toward one point. Other steps of

the algorithm are similar to those of the standard ICA.

In experiments, the proposed algorithm has been used to solve the real world optimization problems presented for IEEE-CEC 2011 evolutionary algorithm competition. Results of experiments confirm the performance of the algorithm.

## References


Andreani, R., Birgin, E.G., Martinez, J.M., *et al*., 2009. Augmented Lagrangian methods under the constant positive linear dependence constraint qualification. *Math. Progr.*, **111**(1-2):5-32. [doi:10.1007/s10107-006-0077-1]

Bertsekas, D., Gafni, E., 1983. Projected Newton methods and optimization of multicommodity flows. *IEEE Trans. Autom. Contr.*, **28**(12):1090-1096. [doi:10.1109/TAC.1983.1103183]

Brownlee, J., 2011. Clever Algorithms: Nature-Inspired Programming Recipes (1st Ed.). LuLu Enterprises.

Das, S., Suganthan, P.N., 2010. Problem Definitions and Evaluation Criteria for CEC 2011 Competition on Testing Evolutionary Algorithms on Real World Optimization Problems. Technical Report, Jadavpur University, India, and Nanyang Technological University, Singapore.

Davis, L., 1987. Genetic Algorithms and Simulated Annealing. Morgan Kaufman Publishers, Los Altos, CA.

Fishman, G., 1996. Monte Carlo Concepts, Algorithms and Applications. Chapel-Hill, Springer-Verlag, New York.

Gargari, A., Lucas, E., 2007. Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition. IEEE Congress on Evolutionary Computation, p.4661-4667. [doi:10.1109/CEC.2007.442 5083]

Golban, C., Nedevschi, S., 2011. Linear vs. nonlinear minimization in stereo visual odometry. IEEE Int. Intelligent Vehicles Symp., p.888-894. [doi:10.1109/IVS.2011.5940537]

Guo, P., Wang, X., Han, Y., 2010. The enhanced genetic algorithms for the optimization design. IEEE Conf. on Biomedical Engineering and Informatics, p.2990-2994. [doi:10.1109/BMEI.2010.5639829]

Kaveh, A., Talatahari, S., 2010. Optimum design of skeletal structures using imperialist competitive algorithm. *J. Comput. Struct.*, **88**(21):1220-1229. [doi:10.1016/j.compstruc.2010.06.011]

Kennedy, J., Eberhart, R., 1995. A new optimizer using particle swarm theory. Proceedings 6th International Symposium on Micro Machine and Human Science, p.39-43. [doi:10.1109/MHS.1995.494215]

Nelder, J., Mead, R., 1965. A simplex method for function minimization. *Comput. J.*, **7**(4):308-313. [doi:10.1093/comjnl/7.4.308]

Rajabioun, R., 2011. Cuckoo optimization algorithm. *Appl. Soft Comput.*, **11**(8):5508-5518. [doi:10.1016/j.asoc.2011.05.008]

Talatahari, S., Azar, F., Sheikholeslami, R., *et al*., 2012. Imperialist competitive algorithm combined with chaos for global optimization. *Commun. Nonl. Sci. Numer. Simul.*, **17**(3):1312-1319. [doi:10.1016/j.cnsns.2011.08.021]