Imperialist Competitive Algorithm Applied to the Optimization of Mathematical Functions: A Parameter Variation Study

Emer Bernal, Oscar Castillo and José Soria

Abstract This paper applies the imperialist competitive algorithm (ICA) to benchmark mathematical functions with the original method to analyze and perform a study of the variation of the results obtained with the ICA algorithm as we vary the parameters manually for 4 mathematical functions. The results demonstrate the efficiency of the algorithm to optimization problems and give us the pattern for future work in dynamically adapting these parameters.

1 Introduction

Swarm Intelligence techniques have become increasingly popular during the last two decades due to their capability to find a relatively optimal solution for complex combinatorial optimization problems. They have been applied in the fields of engineering, economy, management science, industry, etc. Problems that benefit from the application of Swarm Intelligence techniques are generally very hard to solve optimally in the sense that there is no such exact algorithm for solving them in polynomial time. These optimization problems are also known as NP-hard problems [6].

An algorithm that is well recognized in the domain of evolutionary computation is the imperialist competitive algorithm (ICA), which was introduced by Atashpaz-Gargari and Lucas in 2007. ICA has been inspired by the concept of imperialism; where in this case powerful countries attempt to make a colony of other countries. These algorithms have recently been used in several engineering applications [11].

E. Bernal · O. Castillo (✉) · J. Soria
Tijuana Institute of Technology, Tijuana, BC, México
e-mail: ocastillo@tectijuana.mx

P. Melin et al. (eds.), *Design of Intelligent Systems Based on Fuzzy Logic, Neural Networks and Nature-Inspired Optimization*,
Studies in Computational Intelligence 601, DOI 10.1007/978-3-319-17747-2_18

We describe the imperialist competitive algorithm in its original form. The algorithm parameters adjustment is performed manually by varying the parameters to see how the behavior of the algorithm is affected. This algorithm was applied to benchmark mathematical functions, and the results of the ICA algorithm by varying the parameters are presented in Sect. 4.

The study of the algorithm is performed in order to show the effectiveness of the imperialist competitive algorithm (ICA) when applied to optimization problems, and to take this as a basis for future works.

Some of the papers that have applied the imperialist competitive algorithm can be described as follows. In [11] the Imperialist Competitive Algorithm Combined with Refined High-Order Weighted Fuzzy Time Series (RHWFTS–ICA) for short term loads forecasting. In this study, a hybrid algorithm based on a refined high-order weighted fuzzy algorithm and an imperialist competitive algorithm (RHWFTS–ICA) is developed. This method is proposed to perform efficiently under short-term load forecasting (STLF) [11]. In another paper [7] an Imperialist Competitive Algorithm With PROCLUS Classifier For Service Time Optimization In Cloud Computing Service Composition, CSSICA is proposed to make advances toward the lowest possible service time of composite service; in this approach, the PROCLUS classifier is used to divide cloud service providers into three categories based on total service time and assign a probability to each provider. An improved imperialist competitive algorithm is then employed to select more suitable service providers for the required unique services [7].

The paper is organized as follows: in Sect. 2 a description about the imperialist competitive Algorithm ICA is presented, in Sect. 3 a description of the mathematical functions is presented, in Sect. 4 the experiments results are described for we can to appreciate the ICA algorithm behavior by varying the parameters, in Sect. 5 the conclusions obtained after the study of the imperialist competitive algorithm versus mathematical functions are presented.

2 Imperialist Competitive Algorithm

In the field of evolutionary computation, the novel ICA algorithm is based on human social and political advancements [1], unlike other evolutionary algorithms, which are based on the natural behaviors of animals or physical events.

ICA starts with an initial randomly generated population, in which the individuals are known as countries. Some of the best countries are considered imperialists, whereas the other countries represent the imperialist colonies [7].

All the colonies of the initial population are divided among the mentioned imperialists based on their power. The power of an empire which is the counterpart of the fitness value in GA and is inversely proportional to its cost [1].

2.1 Forming Initial Empires (Initialization)

In order to represent an appropriate solution format, a $1 \times N_{var}$ array of variables represents a country, and a country is defined by [6]:

$$Country = [p_1, p_2, \ldots, p_{N_{var}}] \tag{1}$$

where N_{var} is the number of variables to be considered of interest about a country and p_i is the value of *i-th* variable.

The variable values in the country are represented as floating point numbers. The cost of a country is found by evaluating the cost function f at the variables (p_1, p_2, ..., p_n) then [1]:

$$Cost = f(Country) = f(p_1, p_2, \ldots, p_n) \tag{2}$$

In the initialization step, we need to generate an initial population size of N_{pop}. Select N_{imp} of the most powerful countries to form empires. The remaining N_{col} population will be the colonies each of which belongs to an empire [1]

$$N_{coI} = N_{pop} - N_{imp} \tag{3}$$

To form empires, the colonies are divided among the imperialist countries according to the power of the imperialists. The normalized cost of each imperialist is determined by [6]

$$c_n = \max_i \{c_i\} - c_n \tag{4}$$

where, c_n is the *n-th* imperialist's cost, and C_n is the normalized cost of *n-th* imperialist.

Therefore, the power of each imperialist is calculated based on the normalized cost [6]:

$$p_n = \frac{C_n}{\sum_{i=1}^{N_{imp}} C_i} \tag{5}$$

where p_n is the power of *n-th* imperialist. The normalized power of *n-th* imperialist is the number of colonies that are possessed by that imperialist, calculated by:

$$NC_n = Round\{p_n N_{col}\} \tag{6}$$

where, NC_n is the number of initial colonies possessed by the *n-th* imperialist; N_{col} is the total number of colonies in the initial population, and round is a function that gives the nearest integer of a fractional number.

2.2 Moving the Colonies of an Empire Toward the Imperialist (Assimilation)

As shown in Fig. 1 the colony moves x distance along with the d direction towards its imperialist. The moving at x distance is a random number generated by random distribution within the interval (0, βd) [6].

$$x \sim U(0, \beta d) \tag{7}$$

where β is a number greater than 1 and d is the distance between the colony and the imperialist.

As shown in Fig. 2, to search for different locations around the imperialist we add a random amount of deviation to the direction of motion, which is given by [1]:

$$\theta \sim U(-\gamma, \gamma) \tag{8}$$

where θ is a random number with uniform distribution and γ is a parameter that adjusts the deviation from the original direction.

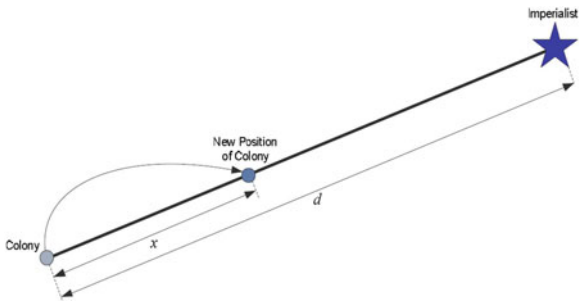

Fig. 1 Movement of the colonies toward the imperialist

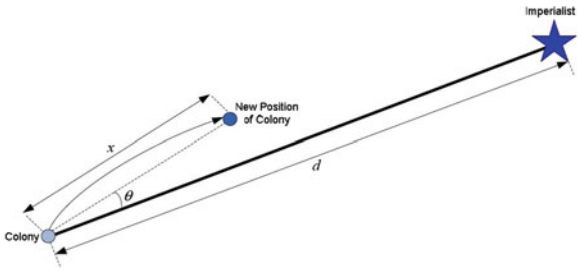

Fig. 2 Movement of the colonies toward their relevant imperialist in a randomly direction deviation

2.3 Exchanging the Positions of a Colony and an Imperialist

While moving towards the imperialist, a colony can reach a position of lower cost than the imperialistic. If so, the imperialist is moved to the position of that colony and vice versa. Then the algorithm will continue with the imperialist in a new position and the colonies begin to move toward this position [1]. In Fig. 3, the best colony of the empire is shown in darker color. This colony has a lower cost than imperialist. Figure 4 shows the whole empire after exchanging the position of the imperialist and the colony.

2.4 Total Power of an Empire

The power of an empire is calculated based on the power of its imperialist and a fraction of the power of its colonies. This fact has been modeled by defining the total Cost given by [6]:

$$TC_n = Cost(Imp) + \xi\, mean\{Cost(Col)\} \tag{9}$$

where TC_n is the total cost of *n-th* Empire and ξ is a positive number between 0 and 1.

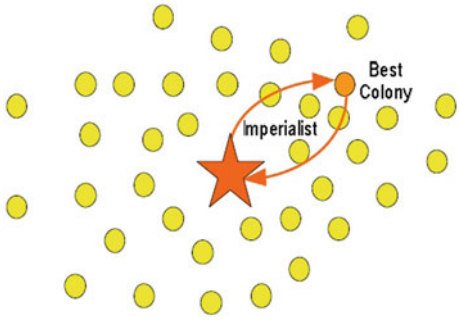

Fig. 3 Position change between the imperialist and a colony

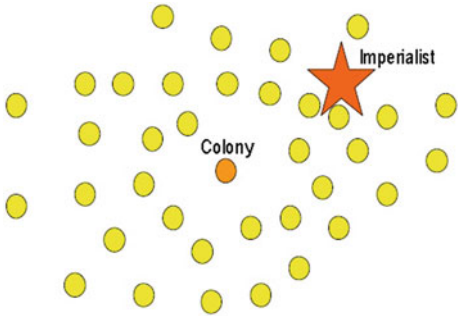

Fig. 4 Position after exchanging empire imperialist and the colony

2.5 Imperialist Competition

To start the competition, first, we find the probability of possession of each empire based on the total power. The normalized total cost is obtained by [1]:

$$NTC_n = \max_i\{TC_i\} - TC_n \tag{10}$$

where NTC_n is the Normalized total cost and TC_n is the total cost of empire. Then the probability of possessing a colony is computed by:

$$p_{p_n} = \frac{NTC_n}{\sum_{i=1}^{N_{imp}} NTC_i} \tag{11}$$

where $\sum_{i=1}^{N_{imp}} p_{p_i} = 1$.

2.6 Elimination of Weaker Empires

Weaker empires lose their colonies gradually to stronger empires, which in turn grow more powerful and cause the weaker empires to collapse over time. In Fig. 5, the weakest empire is eliminated by losing its last colony during the imperialist competition [6].

2.7 Convergence

Similar to other metaheuristic algorithms, ICA continues until a stopping criteria are met, such as predefined running time or a certain number of iterations. The ideal stopping criterion is when all empires have collapsed and only one (grand empire) remains (Fig. 6).

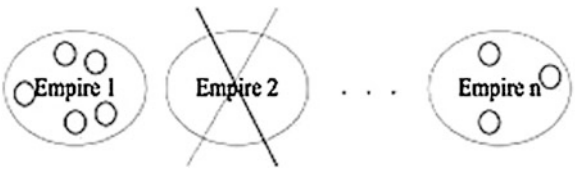

Fig. 5 Elimination of the weakest empire

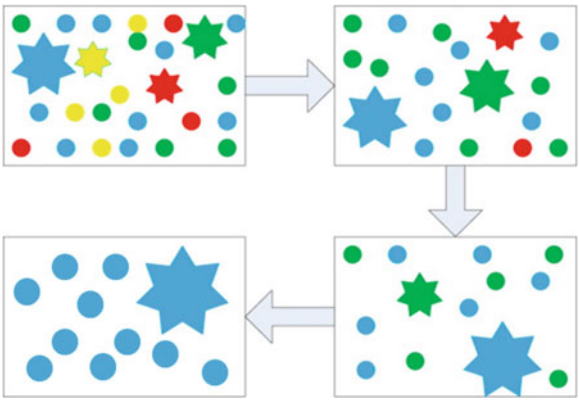

Fig. 6 Representation of convergence in ICA

2.8 Pseudo Code of ICA

The pseudo code it ICA is defined as follows:

1. *Select some random points on the function and initialize the empires.*
2. *Move the colonies toward their relevant imperialist (assimilation).*
3. *If there is a colony in an empire which has lower cost than that of imperialist, exchange the positions of that colony and the imperialist.*
4. *Calculate the total cost of all empires (related to the power of both imperialist and its colonies).*
5. *Pick the weakest colony (colonies) from the weakest empire and give it (them) to the empire that has the most likelihood to possess it (imperialistic competition).*
6. *Eliminate the powerless empires.*
7. *If there is just one empire, stop, if not go to step 2.*

3 Benchmark Mathematical Functions

In this Section, the benchmark functions that are used are listed to evaluate the performance of the ICA algorithm by varying its parameters and to analyze the results obtained.

In the area of the metaheuristics for optimization the use of mathematical functions is common, and they are used in this work: consisting of an optimization algorithm based on imperialism in which the variation of its parameters will be analyzed for obtain its optimum values.

The mathematical functions are shown below:

- Sphere

$$f(x) = \sum_{j=1}^{n_x} x_j^2 \tag{12}$$

Witch $x_j \in [-100, 100]$ and $f^*(x) = 0.0$

- Quartic

$$f(x) = \sum_{j=1}^{n_x} x_j^4 + random(0, 1) \tag{13}$$

Witch $x_i \in [-1.28, 1.28]$ and $f^*(x) = 0.0$ + noise

- Rosenbrock

$$f(x) = \sum_{j=1}^{n_z/2} [100(x_{2j} - x_{2j-1}^2)^2 + (1 - x_{2j-1})^2] \tag{14}$$

Witch $x_j \in [-2.048, 2.048]$ and $f^*(x) = 0.0$

- Rastrigin

$$f(x) = \sum_{j=1}^{n_x} (x_j^2 - 10\cos(2\pi x_j) + 10) \tag{15}$$

With $x_j \in [-5.12, 5.12]$ and $f^*(x) = 0.0$.

4 Simulation Results

In this Section the imperialist competitive algorithm (ICA) is implemented with 4 benchmark mathematical functions with 30 variables by varying their parameters and the results obtained by the ICA algorithm are shown in separate tables by parameter.

The parameters used in the imperialist competitive algorithm are:

- Number of variables: 30
- Number of countries: 200
- Number of imperialist: 10
- Number of decades: 3000

Table 1 shows that after executing the ICA Algorithm 10 times, by varying the revolution parameter, we can find the best, average and worst results for the different mathematical functions benchmark.

Table 1 Results by varying the revolution parameter

Function		0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Sphere	Best	7.47E−10	3.24E−09	5.61E−10	2.52E−09	6.23E−09	2.74E−09	5.07E−10	4.34E−09
	Worst	6.69E−08	1.57E−07	1.64E−07	1.51E−07	1.39E−07	7.40E−08	1.11E−07	2.25E−07
	Mean	2.33E−08	2.72E−08	2.54E−08	4.63E−08	3.77E−08	2.65E−08	2.52E−08	3.89E−08
Quartic	Best	2.34E−13	1.91E−13	8.45E−12	9.30E−13	3.58E−14	5.81E−12	3.87E−13	1.87E−13
	Worst	3.22E−09	1.99E−09	2.87E−09	3.10E−08	6.04E−10	3.84E−10	2.88E−09	2.14E−08
	Mean	3.68E−10	4.03E−10	9.12E−10	3.26E−09	1.42E−10	9.76E−11	5.83E−10	2.79E−09
Rosenbrock	Best	2.51	11.74	19.10	1.39E−01	2.51	15.05	6.25	25.54
	Worst	81.26	83.71	80.05	84.89	88.59	85.24	94.01	98.58
	Mean	58.40	56.13	65.16	47.70	47.52	52.81	59.32	65.56
Rastrigin	Best	2.59	1.60E−03	1.12E−02	4.50E−03	3.30E−03	1.00E−03	9.42E−04	2.20E−02
	Worst	72.00	73.59	74.15	74.55	74.27	72.00	73.29	72.78
	Mean	29.84	44.29	37.41	40.96	38.11	26.66	37.38	30.73

Table 2 Results by varying the β (Beta) parameter

Function		2	1.8	1.6	1.5	1.4	1.3	1.2	1
Sphere	Best	8.12E−09	1.45E−28	5.03E−61	4.60E−83	2.03E−86	3.60E−25	6.16E−19	7.02E−18
	Worst	1.02E−08	3.97E−25	1.32E−58	1.46E−72	6.51E−73	2.08E−17	1.86E−15	1.54E−17
	Mean	6.83E−09	1.20E−25	1.29E−56	1.46E−73	1.30E−73	1.01E−17	3.81E−16	3.30E−17
Quartic	Best	2.34E−13	4.86E−41	8.15E−90	7.76E−132	8.03E−156	5.67E−37	9.75E−27	1.99E−23
	Worst	3.22E−09	8.48E−34	5.58E−85	1.13E−117	4.48E−143	1.79E−27	4.56E−22	2.40E−20
	Mean	3.68E−10	1.70E−34	1.12E−85	1.13E−118	4.48E−144	6.49E−28	9.16E−23	4.92E−21
Rosenbrock	Best	2.52	2.50	3.10E−02	7.67E−04	4.88E−04	6.68	12.36	19.06
	Worst	83.13	73.80	13.32	3.99	9.37	74.84	28.35	73.05
	Mean	59.48	24.15	5.28	1.95	2.11	28.14	22.29	32.60
Rastrigin	Best	1.60E−03	0.00	3.98	6.96	47.76	90.54	110.44	100.50
	Worst	73.59	36.00	80.95	74.80	120.46	120.39	163.54	138.30
	Mean	44.29	18.00	42.91	44.35	90.15	97.73	139.94	121.84

Table 3 Results by varying the γ (Gama) parameter

Function		1	0.9	0.7	0.6	0.5	0.4	0.2	0.1
Sphere	Best	6.47E−09	4.01E−09	2.95E−09	5.51E−09	5.51E−09	2.05E−09	2.06E−09	6.15E−09
	Worst	1.40E−07	2.68E−08	1.63E−08	1.21E−07	1.21E−07	7.63E−08	5.21E−08	5.13E−08
	Mean	2.96E−08	1.70E−08	8.91E−09	3.71E−08	3.71E−08	2.37E−08	1.39E−08	2.06E−08
Quartic	Best	1.46E−13	1.10E−13	3.06E−14	1.19E−12	2.34E−13	2.10E−13	1.86E−12	2.06E−14
	Worst	1.33E−10	1.49E−11	5.42E−11	1.61E−09	3.22E−09	5.64E−10	5.28E−11	2.84E−08
	Mean	3.10E−11	9.46E−12	1.37E−11	3.51E−10	3.68E−10	1.17E−10	2.48E−11	5.78E−09
Rosenbrock	Best	8.45	36.00	36.01	1.46E−01	6.50E−03	6.50E−03	3.95	25.02
	Worst	38.05	72.00	72.00	72.00	72.00	36.00	72.00	72.00
	Mean	28.35	54.07	54.00	46.82	29.23	27.70	33.19	55.41
Rastrigin	Best	2.73	2.71	36.00	9.99E−01	9.70E−03	4.79E−04	6.20E−04	3.95
	Worst	72.00	72.00	72.00	72.00	72.00	72.00	72.00	36.00
	Mean	40.58	27.74	46.82	46.90	36.28	25.44	32.38	31.70

Table 4 Results by varying the ξ (Eta) parameter

Function		0.02	0.1	0.2	0.3	0.5	0.7	0.8	0.9
Sphere	Best	1.40E−09	3.49E−09	7.79E−09	4.54E−09	2.00E−09	4.90E−09	2.73E−09	3.53E−09
	Worst	4.71E−08	3.00E−08	3.82E−08	1.74E−08	1.68E−08	1.66E−08	1.98E−08	4.49E−08
	Mean	1.80E−08	1.32E−08	2.26E−08	1.01E−08	8.46E−09	1.12E−08	9.18E−09	1.67E−08
Quartic	Best	2.34E−13	6.15E−13	9.42E−14	7.89E−14	2.25E−13	1.51E−13	2.65E−13	2.06E−13
	Worst	3.22E−09	1.08E−11	2.37E−12	2.54E−11	3.69E−11	1.12E−08	2.97E−11	3.94E−09
	Mean	3.68E−10	5.72E−12	1.35E−12	3.13E−12	6.35E−12	1.58E−09	5.68E−12	5.34E−10
Rosenbrock	Best	6.50E−03	1.17E−01	2.85E−04	3.60E + 01	4.30E−03	4.30E−04	36.00	2.81
	Worst	72.00	72.00	108.00	72.00	108.00	72.00	72.00	71.82
	Mean	29.23	46.82	47.11	47.35	36.88	34.22	47.06	47.06
Rastrigin	Best	8.45	8.92E−02	1.17E−01	1.07	2.40E−03	18.12	5.20E−03	1.70E−03
	Worst	38.05	72.00	72.00	71.82	72.00	72.00	71.82	72.00
	Mean	28.17	36.06	39.79	47.13	26.01	40.78	36.93	32.40

Table 2 shows that after executing the ICA Algorithm 10 times, by varying the β (Beta) parameter, we can find the best, average and worst results for the different mathematical functions benchmark.

Table 3 shows that after executing the ICA Algorithm 10 times, by varying the γ (Gama) parameter, we can find the best, average and worst results for the different mathematical functions benchmark.

Table 4 shows that after executing the ICA Algorithm 10 times, by varying the ξ (Eta) parameter, we can find the best, average and worst results for the different mathematical functions benchmark

5 Conclusions

By analyzing the ICA algorithm by varying the parameters, we noticed that the parameter that affected most the operation is the β parameter in the range of 1.4–1.6 good results for the sphere and quartic functions.

For the Rosenbrock and Rastrigin functions the results are not good for 30 variables, but with fewer variables the algorithm performs better.

The remaining parameters though apparently do not affect significantly the operation of the algorithm these can be in the range of 0.2–0.5 in the revolution, from 0.4 to 0.6 for γ and small values of ξ the algorithm behaves better.

Acknowledgments We would like to express our gratitude to the CONACYT and Tijuana Institute of Technology for the facilities and resources granted for the development of this research.

References

1. Atashpaz-Gargari, E., Lucas, Y.C.: Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition. In: Evolutionary Computation, pp. 4661–4667 (2007)
2. Banaei, M., Seyed-Shenava, S., Farahbakhsh, Y.P.: Dynamic stability enhancement of power system based on a typical unified power flow controllers using imperialist competitive algorithm. Ain Shams Eng. J. **5**(3), 691–702 (2014)
3. Dallegrave Afonso, L., Cocco Mariani, V., dos Santos Coelho, Y.L.: Modified imperialist competitive algorithm based on attraction and repulsion concepts for reliability-redundancy optimization. Expert Syst. Appl. **40**(9), 3794–3802 (2013)
4. Duan, H., Huang, Y.L.Z.: Imperialist competitive algorithm optimized artificial neural networks for UCAV global path planning. Neurocomputing **125**, 166–171 (2013)
5. Goldansaz, S.M., Fariborz, J., Zahedi Anaraki, Y.A.H.: A hybrid imperialist competitive algorithm for minimizing makespan in a multi-processor open shop. Appl. Math. Model. **37** (23), 9603–9616 (2013)
6. Hosseini, S.M., Al Khaled, Y.A.: A survey on the imperialist competitive algorithm metaheuristic: implementation in engineering domain and directions for future research. Appl. Soft Comput. J. **24**, 1078–1094 (2014)

7. Jula, A., Othman, Z., Sundararajan, Y.E.: Imperialist competitive algorithm with PROCLUS classifier for service time optimization in cloud computing service composition. Expert Syst. Appl. **42**(1), 135–145 (2014)
8. Melin, P., Olivas, F., Castillo, O., Valdez, F., Soria, J., Valdez, M.: Optimal design of fuzzy classification systems using PSO with dynamic parameter adaptation through fuzzy logic. Expert Syst. Appl. **40**(8), 3196–3206 (2012)
9. Mozafaria, A., Behzad, Y., Ayoba, A.: Optimization of adhesive-bonded fiber glass strip using imperialist competitive algorithm. Proc. Technol. **1**, 194–198 (2012)
10. Nourmohammadia, A., Zandiehb, M., Tavakkoli-Moghaddamca, Y.R.: An imperialist competitive algorithm for multi-objective U-type assembly line design. J. Comput. Sci. **4** (5), 393–400 (2012)
11. Rasul, E., Javedani Sadaei, H., Abdullah, A.H., Gani, Y.A.: Imperialist competitive algorithm combined with refined high-order weighted fuzzy time series (RHWFTS–ICA) for short term load forecasting. Energy Convers. Manage. **76**, 1104–1116 (2013)
12. Shamshirband, S., Amini, A., Nor Badrul, A., Mat Kiah, M.L., Ying, W.T., Furnell, Y.S.: D-FICCA: a density-based fuzzy imperialist competitive clustering algorithm for intrusion detection in wireless sensor networks. J. Int. Measur. Confederation **55**, 212–226 (2014)