Fourth International Conference
Modelling and Development of Intelligent Systems
October 28 - November 1, 2015
"Lucian Blaga" University
Sibiu - Romania

Imperialist Competitive Algorithm with Variable Parameters to Determine the Global Minimum of Functions with Several Arguments

Stelian Ciurea

Abstract

We have implemented an Imperialist Competitive Algorithm (ICA) to a determine the global minimum for nine functions. Some of these represent benchmarks of the problem, while others have expressions that we have defined. For a start, we used three of these functions to determine a set of optimal parameters for the ICA. Then we studied the way in which the behaviour of the ICA is affected by these parameters in order to solve the problem put forth for the nine functions. Finally, we studied the behaviour of the ICA as affected by variable parameters for the most difficult of the nine functions. In our algorithm, the values of some of the parameters change dynamically. The results indicate a better behaviour of the solutions provided by this method.

1 The Imperialist Competitive Algorithm: Structure and Parameters

At the "IEEE Congress on Evolutionary Computation" held on September 27-28, in 2007, Esmaeil Atashpaz Gargari and Carlos Lucas presented a paper describing a new type of evolutionary algorithm inspired by history [1]. Called the "Imperialist Competitive Algorithm" (ICA), it is modelled on the political and historical events from the 17th, 18th and 19th centuries. The ICA belongs to the category of meta-heuristic algorithms based on sets of candidate solutions called "populations" (along with genetic algorithms, algorithms of the "swarm of particles" type, GSA gravitational search algorithms, etc.). The standard structure of an ICA as presented in [1] is the following:

1. Generating an initial set of countries;
2. Initializing the imperialist countries;
3. Occupying the colonies;
4. Assimilating the colonies;
5. If a colony has better results than the imperialist country then
    a. Interchanging the colony with the imperialist country
6. The imperialist competition:
    a. Computing the results of the empires
    b. Occupying the weakest colony of the weakest empire by another empire
    c. If the weakest empire has no colonies left then
        i. Removing this empire

7. If the stopping requirements are met then Stop
   Otherwise Repeat the algorithm from step 3.

The imperialist country that has the best results in the last iteration is the solution to the problem. Below is a brief description of the steps of an ICA.

A. Generating an initial set of countries

In the ICA, a country is a possible solution to the problem studied. When determining the minimum for real functions of known expression with n variables (n=2,3,4, ...), a country is represented by a set of n real values. The number of countries is a first parameter of the ICA. As with other evolutionary algorithms, the considerable size of the initial set leads to a higher probability of finding combinations with higher performance. On the other hand, given the complexity of the algorithm, raising the number of countries causes a linear increase in the runtime. In the completed applications, different values were chosen for this parameter: 55, 108 and 210. For the proper generation, the function in the C standard library has been used, as it provides quasi-random values.

B. Assessment of the initial set of countries and their revaluation

The existence of a function that allows the evaluation of the performance of potential solutions, in our case the performance of a country, is a prerequisite to solve a problem by means of this algorithm. In the problem studied, the evaluation function is precisely the function value calculated for the values representing a country. Since we aimed at determining a minimum, the evaluation function was a penalty: the lower the value of the evaluation function, the more efficient a country.

C. Initialization of imperialist countries

The initial number of "imperialist countries" (or original number of empires) is another parameter characteristic of this algorithm. In the applications that we ran, this number was 5%, 10% or 15% of the original number of countries. Following the initial assessment, the countries with the best values of the evaluation function became imperialist countries.

D. Occupying the colonies

In this algorithm, the countries that do not become imperialist countries turn into colonies and "fall within the scope" of imperialist countries. The lower the value of the evaluation function of that imperialist country, the higher the number of colonies assigned to an imperialist country. The ensemble comprising the imperialist country and the occupied colonies form an empire. The formulae for determining the number of colonies belonging to each imperialist country, designated $nrcol_i$, are given below.

$$nrcol_i = v_{nrmet-i-1} \qquad i = 0,1,\ldots,nrmet - 1 \tag{1}$$

where nr_{met} is the initial number of imperialist countries, and values v are calculated by means of the following formula

$$v_i = \left\lfloor \frac{feval(met_i)}{S} nr_{col} \right\rfloor \tag{2}$$

$feval(x)$ is the value of the evaluation function of country x, met_i is the imperialist country with current number i (i=0,1,…,$nrmet$-1), nr_{col} is the initial number of colonies (the difference between the total number of countries and the initial number of imperialist countries) and S is the sum of the values of the evaluation function for all the imperialist countries:

$$S = \sum_{i=0}^{nrmet-1} feval(met_i) \quad (3)$$

E. Assimilation of colonies

By means of this operation characteristic of the ICA, the parameters of the countries of the "colony" type are transformed so that their values can be "attracted" towards the values corresponding to the parameters of the imperialist countries. The way in which this transformation takes place depends on the type of problem that is solved. The formula that we used is the following:

$$paramcol_i = paramcol_{i-1} + p(1 \pm d)(parammet_{i-1} - paramcol_{i-1}) \quad (4)$$

where paramcoli is the value of the colony parameter following iteration i, parammeti is the value of the parameter corresponding to the imperialist country of the colony following i, and d is a random value below par, d∈[-da/2 ; da/2]; da is called assimilation deviation.

F. The revolution operation

Following the historical model, some of the countries of the colony type undergo an operation resulting in modified parameter values. The probability of a country to undergo this operation is called revolution rate. Here is the way in which these values can be modified:

- through a random regeneration: after this operation, all parameters of a country are modified;
- by applying a transformation opposed to that of assimilating some of the parameters ("anti-assimilation"):

$$paramcol_i = paramcol_{i-1} - p(1 \pm d)(parammet_{i-1} - paramcol_{i-1}) \quad (5)$$

$d \in [-d_r/2 ; d_r/2]$, d_r, in this case representing the revolution deviation.

G. The imperialist competition

At this stage of the algorithm, the least powerful empire loses the least efficient colony at the expense of another empire. We start by calculating the performance of each empire using the following formula:

$$performimp_i = preformmet_i + w(\sum performcol_j) \quad (6)$$

where w is the weight that the performance of a colony contributes to the performance of the empire $w \in (0,1)$, $performimp_i$ and $performmet_i$ are the performance of the empire and that of the imperialist country i (i=0,1,…,*nrmet*-1), and $performcol_j$ is the performance of a colony belonging to empire i. To determine the empire that will assimilate the colony, we used the Monte Carlo method. A series of random values a0, a1, … , anrmet-1 / ai∈(0,1) was generated and the colony will belong to the empire for which the value of the expression below is maximum:

$$a_i - \frac{performimp_i}{\sum_{j=0}^{nrmet-1} performimp_j} \quad (7)$$

If, following this operation, an empire loses all the colonies, the imperialist country of that empire also becomes a colony and is assigned to another randomly selected empire.

H. Completion of the algorithm

There are two conditions that determine the completion of the algorithm and that we implemented:

- one empire left (the ideal situation – convergent algorithm);
- the maximum number of iterations was reached.

As with other evolutionary algorithms, there are no mathematical demonstrations to prove that the algorithm converges towards the optimal solution. In fact, the algorithm refines the search in the vicinity of the points where the evaluation functions have good values. The competitive imperialist algorithms have a number of parameters that depend on their structure. For the algorithm structure presented before, the number of parameters is 10.

2 The ICA determining the global minimum of functions with several arguments

A major problem that should be considered when seeking to solve a problem with the help of an algorithm of this type is finding the optimal values for the algorithm parameters. In the specialist literature, we found no recommendations for that purpose. To solve this problem, we considered two categories of parameters: the first category included parameters whose values were combined in the tests observing the principle "with each other":

- Number of countries: 55, 108 and 210;
- The values of the initial set of countries: the algorithm was run for 5,000 initial sets;
- The number of imperialist countries: 5%, 10% and 15% of the total number of countries;
- The method of implementing revolutions: no revolutions, regeneration and anti-assimilation (according to formula 4);
- The maximum number of iterations: 3,000 (a value high enough so that the algorithms end on account of the fact that there is only one empire).

The second category included parameters whose values were combined in the tests, following this rule: each of the five values of one of them was combined with the average values of the others. This was the procedure for the following:

- the approach step used in assimilation and revolution operations: 0.1, 0.3, 0.5, 0.7 and 0.9;
- assimilation deviations: 0.2, 0.6, 1.0, 1.4, 1.8;
- revolution deviation: 0.2, 1.0, 2.0, 4.0, 20.0;
- The percentage that colony contributes to the value of the performance of an empire: 0.01, 0.05, 0.1, 0.5, 1.0;
- The rate of revolution: was chosen as the number of countries whose parameters are modified so as to be proportional to the ratio between the number of countries and the originally defined number of empires, but so that the revolutions are initiated after a number of iterations (marked T) equal to 1, 5, 10, 20 and 50;

Thus, we noted how the values of each of these parameters influenced the performance of the algorithm. The behaviour of the ICA was studied for three functions:

$$f_1(x_1, x_2) = 1 - \sin(1 + 3x_1(x_2 - 1))e^{-(x_1-1)^2 - x_2^2} \tag{8}$$

$$f_2(x_1, x_2, x_3) = 7 + \sin(x_1 - 1)e^{\frac{1}{1+x_1^2}} + \sin(x_2)e^{\frac{1}{1+x_2^2}} + \sin(x_3 x_2)e^{\frac{1}{1+x_3^2}} \tag{9}$$

$$f_3(x_1, x_2, x_3, x_4) = 3 + \frac{\sin(x_1 - 0.5x_2 + 1.2x_3 - x_4)}{(x_1 - 1)^2 + (x_2 + 1)^2 + (x_3 + 2)^2 + x_4^2 + 1}(0.5x_1 - x_2 + x_3 - 2x_4 + 4) \tag{10}$$

For each of these three functions were chosen the fields of definition [-10; 10] and [-100, 100]. By default, these fields were also those of searching the minimum and determining to what extent the size of this field affects the results. By joining the values selected for the ICA parameters, a number of 5,400 combinations resulted. Each of these combinations was run for the 5,000 sets of countries and for each of the two areas. The tests conducted resulted in a set of optimal values for the ICA parameters, i.e.:

- The method of implementing the revolution through regeneration;
- The number of countries: 210;
- The initial percentage of empires in the total number of countries: 5%.
- The approach step, p = 0.1;
- The percentage of colonies within the empire performance, w = 0.5;
- The deviation from assimilation, yes = 1.8;
- The number of iterations at which revolutions T = 1 occur;

ICAs with parameters thus determined were used to calculate the minimum of nine reference functions or with an expression determined by us. Starting from the observation that the ICA does not guarantee finding the solution, we ran the algorithm for 100 initial sets. The number of these sets was selected so that the total running time should be under a minute, all tests being run on an Intel Core i3 microprocessor at 2.93 GHz.

Table 1 shows a summary of the results. The values in column nrOk represent the number of sets for which ICA located the global minimum of the function (out of the 100 sets used).

Function	[-10;10][1]	[-100;100]
	nrOk	nrOk
$f_1(x_1, x_2) = 1 - \sin(1 + 3x_1(x_2 - 1))e^{-(x_1-1)^2 - x_2^2}$	100	100
$f_2(x_1, x_2, x_3) = 7 + \sin(x_1 - 1)e^{\frac{1}{1+x_1^2}} + \sin(x_2)e^{\frac{1}{1+x_2^2}} + \sin(x_3 x_2)e^{\frac{1}{1+x_3^2}}$	100	78
$f_3(x_1, x_2, x_3, x_4) = 3 + \frac{\sin(x_1 - 0.5x_2 + 1.2x_3 - x_4)}{(x_1-1)^2 + (x_2+1)^2 + (x_3+2)^2 + x_4^2 + 1}(0.5x_1 - x_2 + x_3 - 2x_4 + 4)$	100	100
$f_4(x_1, x_2) = x_1^2 + (x_2^2 - 2)^2 - 2$ (De Jong)	100	100
$f_5(x_1, x_2) = J0(x_1^2 + x_2^2) + 0.1\lvert 1 - x_1\rvert + 0.1\lvert 1 - x_2\rvert$ (benchmark in [1])	100	100
$f_6(x_1, x_2) = x_1 \sin 4x_1 + 1.1x_2 \sin 2x_2, \quad x, y \in [0,10]$ (benchmark in [8])	100	
$f_7(x_1, x_2) = 2 + \sin(x_1)\frac{x_1 + 2}{1 + x_1^2} + \sin(x_2 + 1)\frac{x_2 + 1.1}{2.1 + x_2^2} + \sin(x_1 x_2)\frac{x_1 + x_2 - 1}{3 + x_1^2 + x_2^2}$	100	100
$f_8(x_i) = \prod_{i=1}^{7} \sqrt{x_i}\sin(x_i)$, $f_8: [0;10]^7 \to R$ (Alpine)	96	
$f_9(x_i) = \sum_{i=1}^{3} (x_i^2)^{x_{i+1}^2+1} + (x_{i+1}^2)^{x_i^2+1}, \quad xi \in [-1,4]$ (Brown)	100	

Table 1. Results obtained

[1] Valid definition fields except for functions in the expression of which it is explicit: f_6, f_8 and f_9

The following figures shows the behaviour of the algorithm in determining the minimum of function f1: [-100;100]×[-100;100] for one of the 100 initial sets with 108 countries and 5 imperialist countries.

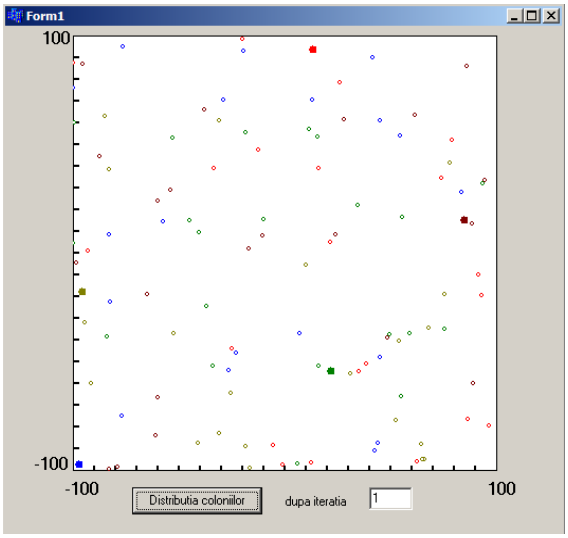

Fig.1 Distribution of colonies following one iteration

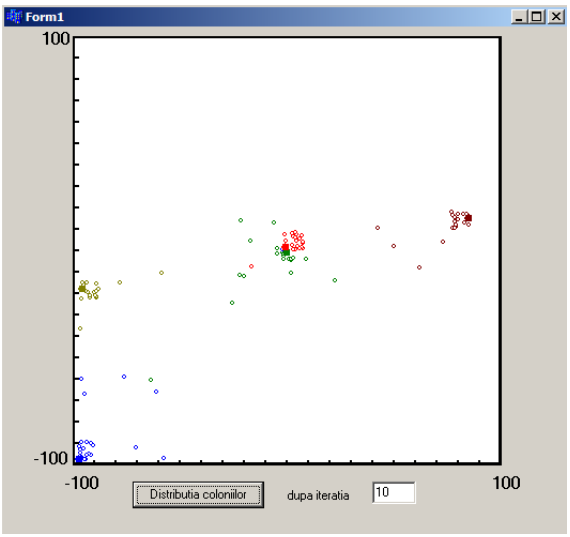

Fig. 2 Distribution of colonies following10 iterations

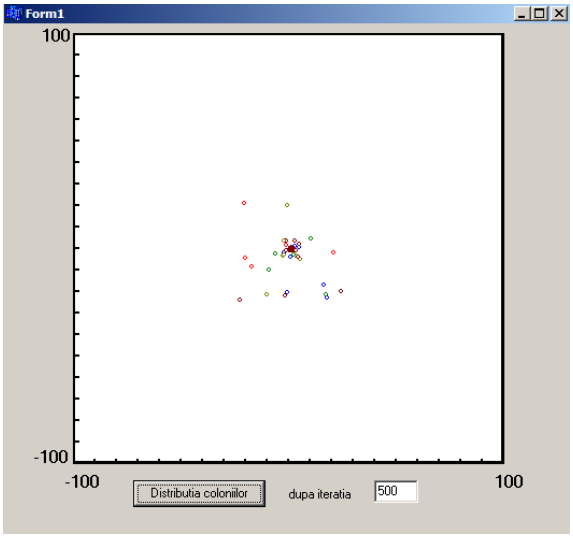

Fig. 3 Distribution of colonies following 500 iterations

The first figure illustrates the initial distribution obtained by random generation. The larger dots represent the location of the imperialist countries. With the same colour are marked the imperialist country and colonies that belong to it. After 10 iterations, we can notice the grouping of colonies around the imperialist country to which they belong as a result of assimilation. We can also see that two potential solutions are closer to the solution sought for. After 500 iterations, all 5 imperialist countries as well as many of the colonies that belong to them are closer to the optimal solution. Figures 4, and 5 illustrate the behaviour of the algorithm in the case of function f2 for one

of the tests in which 108 countries and 5 imperialist countries were used. The first figure illustrates the initial distribution obtained by random generation. The larger dots represent the location of the countries. With the same colour are marked the imperialist country and colonies that belong to it. Following 7 iterations, we can notice the grouping of the colonies around the imperialist country they belong to as a result of assimilation. Following 400 iterations, there is a group consisting of two imperialist countries located very close to each other and a group of three imperialist countries located very closet o them. Many of the colonies that belong to them are located in their vicinity. Following 1283 iterations, three empires were left, their imperialist countries being located very close to them.

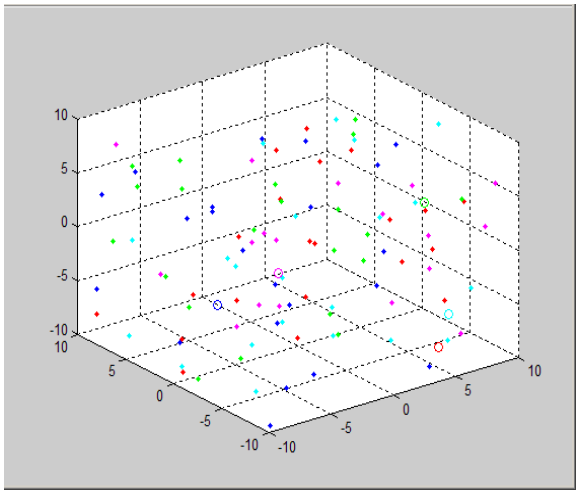

Fig. 4 Distribution of colonies following one iteration

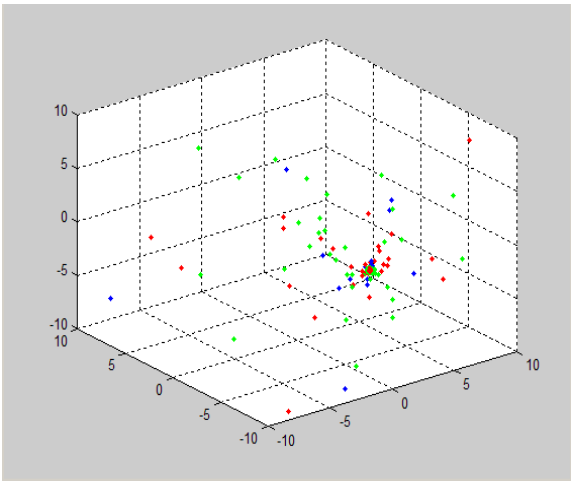

Fig. 5 Distribution of colonies following iteration no. 1283

3 Behaviour of ICA with variable parameters

3.1 ICA with dynamic weight

In the simulations performed, there were many situations in which the standard algorithm converged very slowly or was even stuck because the performance of last two, sometimes even three or four empires left in the algorithm, became very close – differences of the order of 0.1% of the values of the performance. The result was that, from a certain iteration, one and the same colony shifts from one empire to the other. Here is the explanation: when a colony shifts from one empire – be it I1 – to the other – be it I2, (since performance (I1) < performance (I2)), it undergoes an assimilation process by the new imperialist country. The immediate effect is a worsening of the performance of the colony; this value affects the performance of the empire which took over the colony, so that in the next iteration performance (I2) < performance (I1) and the weakest colony is precisely the one which has been taken over. Thus, the colony returns to the first empire, but with a performance value that was worsened by applying a new assimilation operation by I1. Once more, performance (I1) < performance (I2) and the weakest colony that will be taken over by I2 is obviously the one that has just returned to I1. This algorithm follows a loop: one colony shifts between the two empires, the others are assimilated in an increasingly higher percentage, and the chances that colonies with better performance than that of the imperialist country may appear decrease with each iteration. Such blockage has been removed by using weights whose values change dynamically as follows: initially, each colony is assigned the same

weight, having a predetermined value; whenever a colony changes the imperialist country, the value of the weight of that colony is reduced by multiplying it with a below par value. We called this value weight contraction (cw). Thus, equation (7) turned into:

$$performimp_i = preformmet_i + \sum w_j \, performcol_j \qquad (11)$$

where wj is the weight associated to colony j.

Another change to the standard algorithm that was performed in order to avoid blockage was the following: the weakest colony is determined having as a criterion the product between the evaluation function and the weight of the colony in that iteration (and not only the value given by the evaluation function for that colony). Figures 10 and 11 illustrate the comparative evolution of empire performances when the ICA is applied to function f2: [-100;100]×[-100;100] for an initial set of 108 countries and 5 imperialist countries. Figures 12 and 13 illustrate the evolution of the number of colonies within each empire under the same conditions. It can be noted that the ICA with variable weight is completed after 253 iterations, but the performance of the algorithm is not affected.

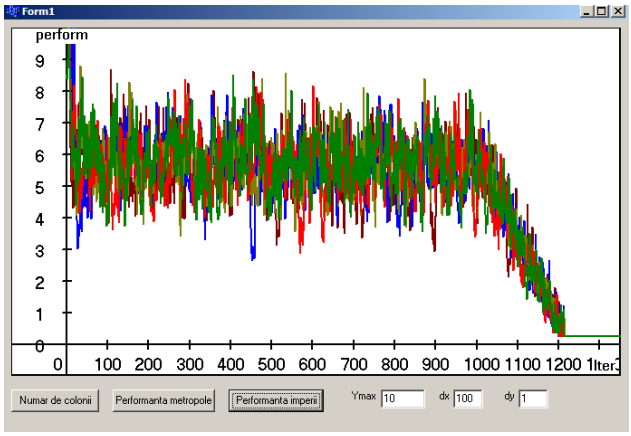

Fig. 6 Empire performances in the case of the ICA with fixed weight

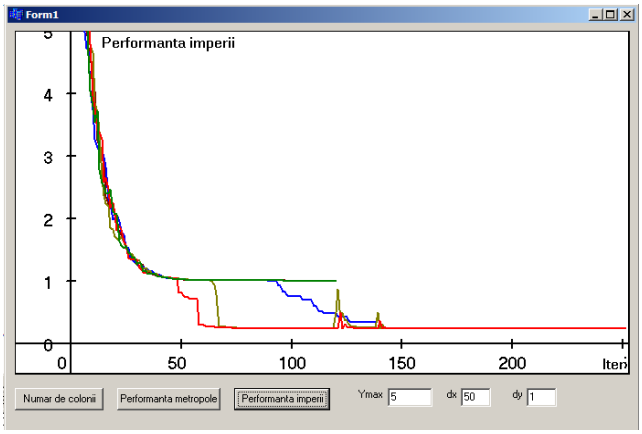

Fig. 7 Empire performances in the case of the ICA with variable weight cw=0.5

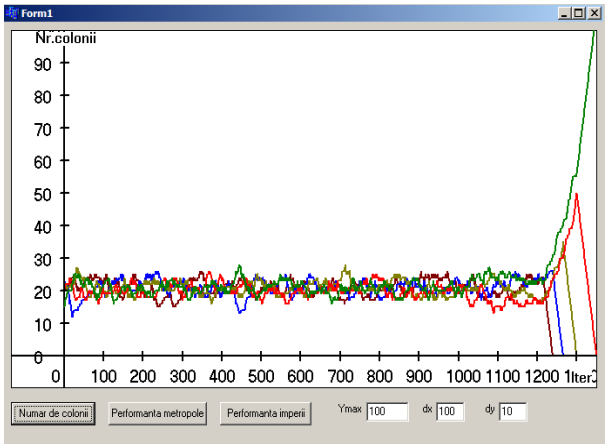

Fig. 8 Evolution of the number of colonies in the case of the ICA with fixed weight

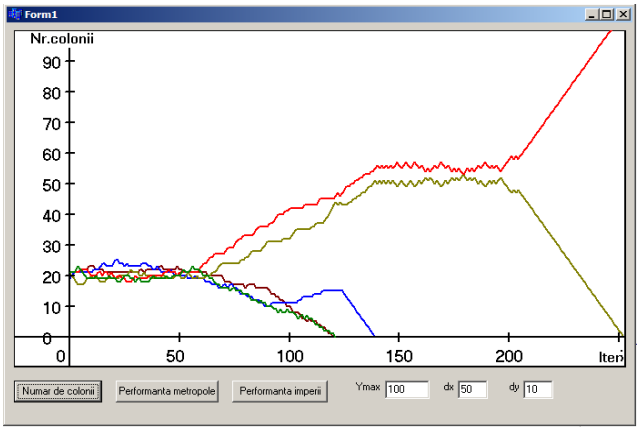

Fig. 9 Evolution of the number of colonies in the case of the ICA with variable weight cw=0.5

In the following, we present the results obtained by applying the ICA with variable weight to determine the minimum of function f2 with the definition field [-100 ; 100]3, practically the most "difficult" of the functions studied. Table 2 shows the behaviour of the ICA in three cases: high fixed weight, low fixed weight and variable weight. The number of sets was 100 for the ICA with w=0.5 and cw=1 (fixed weight). In the other cases, this number was inversely proportional to the average number of iterations. After that, the algorithm stopped (we designated this quantity nrIt) so that the total running time may be the same for each of the three cases analyzed.

	$f_2 : [-100;100]^3 \rightarrow R$								
	55 countries			108 countries			210 countries		
Tip ICA	$\min(f_2)$	nrOk	nrIt	$\min(f_2)$	nrOk	nrIt	$\min(f_2)$	nrOk	nrIt
w=0.5, cw=1	0.844188	51	1150	0.844188	67	1337	0.844188	78	1792
w=0.01, cw=1	0.844188	53	1257	0.844188	58	1593	0.844188	34	1089
w=0.5, cw=0.5	0.844188	3	67	0.844188	47	172	0.844188	**156**	491

Table 2. ICA with fixed weight vs. ICA with variable weight

The most efficient variant of those shown in the table according to the number of cases where the minimum of the function is obtained is that of the algorithm with variable weight and 210 countries in the initial set.

3.2 ICA with variable revolution rate

The ICA behaviour was studied when the number of countries where revolutions occur varies, this number increasing during the sequence of the algorithm. This is because, as the algorithm progresses, the number of colonies that will be in the vicinity of the imperialist countries is increasingly higher, due to the convergent formulae used in the assimilation operation. Hence, the idea of having an increasing number of countries involved in the revolutions.

In the tests conducted, the revolution rate, designated $prob_r$ below, had the following values:

- $$prob_r = \frac{\text{number of countries}}{\text{initial number of empires}} \quad (12)$$
- $$prob_r = \frac{\text{number of countries}}{\text{initial number of empires}} + 20 \quad (13)$$
- $$prob_r = \frac{\text{number of countries}}{\text{initial number of empirea}} + \frac{iteration}{50} \quad (14)$$
- $$prob_r = \frac{\text{number of countries}}{\text{initial number of empires}} + \frac{iteration}{100} \quad (15)$$

As it can be noticed, in the first three formulae, the revolution rate is fixed and, in the last two formulae, it is variable. The other parameters were kept constant at the values determined in section 2. The tests were also carried out in order to determine the minimum of function f2: [-100;100]3→ R. The results are shown in Table 3.

$prob_r$ (formula)	Number of countries					
	55		108		210	
	$min(f_2)$	nrOk	$min(f_2)$	nrOk	$min(f_2)$	nrOk
12	0.844188	51	0.844188	67	0.844188	78
13	0.862990	1	0.844188	52	0.844188	82
14	0.844748	4	0.844188	55	0.844188	**87**
15	0.844201	21	0.844188	62	0.844188	82

Table 3. ICA with fixed revolution rate vs. ICA with variable rate

A better behaviour of the ICA with variable revolution rate can be noticed in tests with a high number of countries (210) according to formula (16). This formula corresponds to the case where there is a more rapid increase in the number of countries subject to the operation of revolution depending on the iteration reached by the algorithm.

4 Conclusions

The ICA proved to be an efficient algorithm in determining the minimum of functions with several arguments. To determine an optimal set of parameters for the algorithm, we ran 27,000,000 tests. The ICA with this optimal set of parameters was used to determine the global minimum for nine functions; for seven of them, we used two fields of definition. In 14 of the 16 cases that resulted, the ICA determined the minimum in 100% of the 100 tests conducted. The lowest probability of obtaining the minimum was 78%.

In section 3, we studied the behaviour of the ICA with two of its characteristic parameters dynamically modified. The studies were conducted for the function considered to be the most difficult. The first study aimed at modifying the parameter that determines the weight with which a colony influences the performance of the whole empire to which it belongs. This parameter was modified with the initial purpose of removing the blockages of the ICA because of very poor performance colonies. Thus modified, the ICA presented superior convergence: if for the ICA with fixed weight the algorithm stopped after 1792 iterations on average, the ICA with variable weight stopped after 491 iterations on average. Thus, within the same time interval, the ICA with fixed weight located the global minimum for the function studied for 78 initial sets and the ICA with variable weight located the minimum in 156 initial sets.

The second study was concerned with the behaviour of the ICA when the number of countries where revolutions take place varies, increasing during the sequence of the algorithm. This is because, as the algorithm progresses, the number of colonies that will be in the vicinity of imperialist countries is increasingly higher, due to the convergent formulae used in the assimilation operation. Hence, the idea of having an increasing number of countries involved in the revolutions. In the tests conducted, we used two formulae for this parameter. For one of them, the ICA located the minimum of function f2 in 86 of the 100 initial sets, which represented an improvement of the performance of the algorithm with 10.25%.

References

[1] Gargari Atashpaz, E., Caro, L.: *"Imperialist Competitive Algorithm: An Algorithm for Optimization Inspired by Imperialistic Competition"*, IEEE Congress on Evolutionary Computation, CEC, 2007.

[2] Beheshti, Y., Shamsuddin, S. M. Hj.: *"A Review of Population-based Meta-Heuristic Algorithms"*, Int. J. Advance. Soft Comput. Appl., Vol. 5, No. 1, March 2013 ISSN 2074-8523.
[3] Forouharfard, S., Zandieh, M.: "*An imperialist competitive algorithm to schedule of receiving and shipping trucks in cross-docking systems*", International Journal of Advanced Manufacturing Technology, 2010
[4] Hojjat, E., Shahriar, L.: "*Graph Colouring Problem Based on Discrete Imperialist Competitive Algorithm*", CoRR, 2013.
[5] Liu, J.Y.-C., Yuan Z., S., Chiang-Tien C.: "*On the Convergence of Imperialist Competitive Algorithm*", 7th Asia Modelling Symposium, 2013.
[6] Soltani-Sarvestani, M. A., Badamchizadeh, M. A., Soltani-Sarvestani, Sh., Javanray, D.: "*Investigation of Revolution Operator in Imperialist Competitive Algorithm*", 4th International Conference on Computer and Electrical Engineering (ICCEE 2011).
[7] Ciurea S., Trifa V.: "*Imperialist Competitive Algorithm with Variable Parameters for the Optimization of a Fuzzy Controller*", International Conference on System Theory, Control and Computing ICSTCC, Sinaia, 2014.
[8] Schwefel, H.-P.: *Numerical optimization of computer models*, Wiley, 1981, ISBN13: 978-0471099888, LC: QA402.5.S3813.

STELIAN CIUREA
"Lucian Blaga" University of Sibiu
Faculty of Engineering, Department of Computer and Electrical Engineering
E. Cioran Str, No. 4, Sibiu-550025, ROMANIA,
E-mail: stelian.ciurea@ulbsibiu.ro