Journal of AI and Data Mining
Vol 6, No 1, 2018, 59-67

A Hybrid Meta-Heuristic Algorithm based on Imperialist Competition Algorithm

R. Roustaei* and F. Yousefi Fakhr

Department of Computer, Malayer Branch, Islamic Azad University, Malayer, Iran.

Received 04 December 2015; Revised 21 July 2016; Accepted 14 September 2016
**Corresponding author: RassoulRoustaei@gmail.com (R.Roustaei).*

Abstract
The human has always been to up to the best in everything, And this perfectionism has led to the creation of optimization methods. The goal of optimization is to determine the variables to find the best acceptable answer to the limitations in a problem, so that the objective function is a minimum or a maximum. Meta-heuristic algorithms are one of inaccurate optimization methods that inspired by nature. In the recent years, much effort has been made to improve or create metaheuristic algorithms. One of the ways available to make improvements in meta-heuristic methods is to use combination. In this paper, a hybrid optimization algorithm is presented based on the imperialist competitive algorithm (ICA). The ideas used in ICA are an assimilation operation with a variable parameter and a war function that is based upon the mathematical model of a war in the real world. These changes lead to an increase in speed, find a global optimum, and reduce the search steps in contrast with the other meta-heuristic algorithms, so that the evaluations are made for more than 80% of the test cases. The proposed algorithm superior to the imperialist competitive algorithm, social based algorithm, cuckoo optimization algorithm, and genetic algorithm.

Keywords: *Optimization Method, Imperialist Competitive Algorithm(ICA), Meta-heuristic Algorithm, Hybrid Algorithm.*

1. Introduction

Optimization is one of the important branches of engineering, which has a great effect on a structural design. The designers would be able to present better designs only if they could save time and money with the aid of optimization methods. Optimization means finding the best possible answer to an optimization problem [1].

Most optimization problems are more complicated to be solved using the common optimization methods such as math programing. For instance in hybrid optimization problems, the goal is to find the optimum point of the function with discrete variables. In confronting these problems, which are mostly in the NP-Hard group, one of the existing solutions is to use the approximate or heuristic algorithms. These algorithms have the grantee that the given solution is an optimum one, and only after spending a long time we can just find a fairly exact answer; in fact, based on the time it takes, the exactness of the answer will vary [2].

In the past 3 decades, a new type of approximate algorithms has been created, which aims to do the hybrid function in higher levels of basic explorative methods in order to increase the performance and accuracy of a search. Nowadays these algorithms are called meta-heuristic algorithms. The heuristic method is defined as a repetitive production process that guides a subsidiary heuristic function using intelligent composition with different exploration concepts [2]. This process will apply a simple perfect (or imperfect) answer or a set of answers in each iteration. Subsidiary heuristic function could be one of the high-level (or low-level) procedures or a simple local search or just a structural method [3]. There are a variety of hybrid methods available in heuristic algorithms. The first one is to add some components of a heuristic method to another one. The second type consists of a system called collaborative search, in which a variety of algorithms are available for information exchange.

The third method is to merge the approximate and principal methods [3, 4]. In the recent years, we have faced an increase in the optimization researchers' interest to heuristic algorithms, which leads to the achievement of the best results for the optimization problems. Table 1 shows several samples of hybrid heuristic algorithms presented in the recent years.

2. Imperialist competitive algorithm

Generally, the ideas of heuristic and meta-heuristic algorithms are based upon a natural process. The imperialist competitive algorithm (ICA), which was represented in 2007 by Atashpaz Gargary, is the opposite of the other algorithms inspired from a human-social phenomenon. This algorithm particularly looks at a colonial process as steps of a social-political evolution, and with mathematical modeling of this historical phenomenon, uses it as an inspiration source for a strong algorithm in the optimization context.

In a little while past from the introduction of this algorithm, it has been used a lot for solving many problems in the optimization field [5]. ICA[1] starts with some initial populations, like the other evaluation optimization methods. In this algorithm, each population element is called a *country*. In an optimization problem with N dimensions, the specification of *countries* (initial population) is characterized by an array, which is defined as follows:

$$country = [p_1, p_2, p_3, ..., p_{N_{var}}] \quad (1)$$

The variable values are displayed in decimal numbers. From a cultural-historical viewpoint, the components of a *country* can be considered as social-political features such as the culture, language, economy, and structure. Figure 1 demonstrates the social-political components of a *country*.

According to this figure, based on a social-political viewpoint, the passive variables of a cost function are the cultural and historical features, so that a *country* is guided toward a minimum point of the cost function. In fact, the goal of solving an optimization problem by ICA is to find the best *country*, i.e. a *country* with the best cultural-historical features. As a matter of fact, finding this *country* is equal to reach best variables for the problem that produce a minimum value for the cost function.

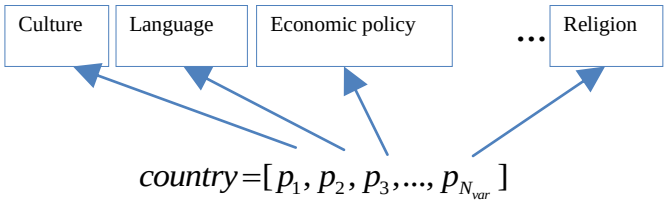

Figure 1. Socio-political components of a *country*.

The cost of a *country* is obtained by evaluation of function f for the variables $(p_1, p_2, p_3, ..., p_{N_{var}})$.

$$cost_i = f(country_i) = f(p_1, p_2, p_3, ..., p_{N_{var}}) \quad (2)$$

3. Proposed ideas to combine

In this section, the two ideas are considered which has been used in the ICA; then how to combine them with basic algorithms are expressed.

3.1. Fight algorithm

According to the real world that we live in, and knowing that ICA is based on the colonialism among countries, we focused on the point that in the real world, for the colonization of a country, a phenomenon called war may happen between the imperialists. The countries that have more power would attack the other ones and capture them as a part of their empire. In 1832 general Carl Von Clausewitz, a Russian commander and a military theorist, in his thesis called "On war" presented a definition for war: "War is an act of force to compel our enemy to do our will" (Wikipedia).

According to this idea, a war may happen when there are more than one empires. We choose an empire randomly or by using the roulette wheel among stronger empires as the starter of the war, and use the same method to choose a weak empire from the weakest ones to be attacked. In this step, we check to make sure that the attacker empire is stronger than the empire under attack because meanwhile, a weaker empire may attack a stronger one, and it would undoubtedly be defeated. Thus by means of this comparison, we can prevent the occurrence of this war in the beginning. After choosing both the attacker and the immolation countries, if the immolation country has more than one country except the empire, the attacker country will possess one country from the victim country. In this condition, small empires with less population will have more chance to survive. The fight algorithm is shown in figure 2.

[1] Imperialist Competitive Algorithm

Fight Function( )

1- {Select an Imperialist as the Fighter with Roulette Wheel random mechanism

2- Select an Imperialist as Attacked with Roulette Wheel random mechanism

3- If Fighter is Stronger than Attacked

4- {Select a Colony of Attacked as Victim with Roulette Wheel random mechanism

5- Remove Victim of Attack and Update Attacking Empire Add Victim to Fighter and Update Empire of Fighter}}

Figure 2. pseudo-code of Fight algorithm.

The stages of a fight algorithm and the manner to implement them are as follow:

A) Choosing the attacker empire (fighter):
In order to choose the attacker empire, first, we allocate a probability to each empire using the Boltzmann probability distribution according to the cost of each empire, so that the empire with a lower cost will have a more chance to be selected. Using the roulette wheel selection, we choose one of the empires as the attacker country. It is noticeable that in this paper, our assumption is based upon minimization, and therefore, an empire with a lower cost is the stronger.

B) Selection of the empire that comes under attack (The empire under attack):
Again we use the Boltzmann probability distribution to choose the attacked country, by allocating a high probability to the weaker empire, and we give each country a probability. Contrariwise the previous step, here, the stronger empires will have a lower chance to be selected. Now by using the roulette wheel selection, one of the empires will be selected as the attacked empire.

Table 1. Comparison of previous works.

Authors	Algorithm	combined algorithms	Year	Improvement
Nigam , Jain [6]	CCA-GA	GA & CCA	2010	Increasing the convergence speed and accuracy of CCA
Razavi, Khorani, Ghoncheh [7]	R-ICA-GA	GA & ICA	2010	Increasing the convergence speed and accuracy
Abdechiri & Meybodi [8]	ICALR	ICA & LR	2011	High speed than ICA & PSOLR
Abdechiri & Meybodi [8]	HNNICA	ICA & HNN	2011	Increasing the public search functionality of HNN
Taher Niknam et al. [9]	Hybrid K-MICA	K-means & ICA	2011	Greater convergence of hybrid K-MICA algorithm than other evolutionary algorithms
Pooranian. et al. [10]	GA-GELS	Fuzzy logic & Q-learning	2011	This algorithm was presented for task scheduling in grid computing to decrease missed task and make span
Ramezani et al. [11]	HEICA	EA & ICA	2012	Increasing the convergence speed and accuracy of ICA & EA
Ramezani & Lotfi [12]	SBA	EA & ICA	2013	Increasing the convergence speed
Lepagnot. et al. [13]	ICAS	ICA & Nelder-Mead simplex method	2013	Improvements in escape from local optimization
Jalal Nouri. et al. [14]	HYEI	GA & ICA	2014	Better classification of small data sets and large data sets than the last known classification algorithm
Shamshirband et al. [15]	D-FICCA	DBSCAN-based density clustering & fuzzy logic & ICA	2014	This algorithm was presented for sensor network clustering, that leads to increased detection accuracy and clustering quality. The modified ICA-based detection system operates to sense DDoS attacks
Shamshirband. et al. [16]	FQL	Fuzzy & Genetic	2014	this approach was presented to protect wireless nodes from DDoS attacks
Shojafar et al. [17]	FUGE	Fuzzy & Genetic	2015	This algorithm was present for cloud job scheduling and improves on various measures such as execution time, execution cost, and the average degree of imbalance.
Roozbeh Nia et al. [18]	HGA	ICA & GA	2015	find better and nearer optimal solutions.
Mehdinejad et al. [19]	PSO–ICA	ICA & PSO	2016	This algorithm find the solution of optimal reactive power dispatch (ORPD) of power systems. The results show that the proposed hybrid approach is more effective and has a higher capability in finding better solutions
Valipour, Kh., et al.[20]	MHSA	HSA & CLS	2016	This algorithm is a new modified harmony search algorithm to find the solution of optimal reactive power dispatch (ORPD) of power systems.

C) Selection of victim country:

In order to capture a country, the attacker country should observe two conditions. First, the power of the attacker empire must be more than that of the attacked empire (be more qualified).

Secondly, the attacked empire should have 2 other countries except the emperor. This causes that the attacked empire still has another country after the war, otherwise the war could make an empire to be destroyed and in the imperialist competition algorithm the act of eliminating one empire could happen in the competition among countries but we are preventing this in the war. Therefore, the attacked country could still survive, and this prevents the algorithm from a premature termination. Now we allocate a probability to each exciting country with a high priority for the weaker colony using the Boltzmann distribution and then with the roulette wheel selection, we choose a country. The selected country will be labeled as "Victim".

D) Remove the victim country from its empire (empire under attack) and add it to the attacker empire (fighter).

Now we should separate the victim country from the owner empire and join it to the attacker country. Finally, both countries should be updated (number of member countries, empire cost).

3-2. Policy of variable beta assimilation idea (second idea)

According to [5], the policy of assimilation (absorption) is applied to the culture of the central government, and the aim is to analyze the culture and social structure of the colonies. In fact, in this algorithm, the goal of assimilation policy is the same in ICA; it means that the colony country (country T) has moved as long as x units along with the connection line between the colony and imperialist and will be drawn in the new location. Figure 3 shows the assimilation policy of a colony toward an imperialist.

To move a colony toward an imperialist we calculate the sum of the locations of the colony with an n dimension array of *x*-, where *x* is a random number with uniform distribution or any other distribution that can be obtained by Formula 3.

$$x \in U(0, \beta \times d) \quad (3)$$

In this equation, d is the distance between the colony and the imperialist, and β is a number more than one and close to 2. Also, the possible deviations are applied by adding a random angle to the assimilation path. This deviation angle is called θ.

$$\theta \in U(-\gamma, \gamma) \quad (4)$$

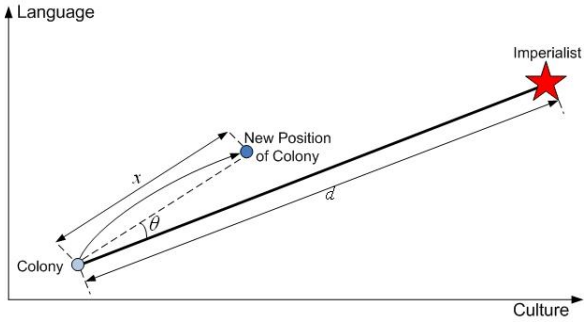

Figure 3. Assimilation policy, moving colony toward imperialist.

In this equation, γ is an arbitrary variable, the increase in which causes an increase in the search around the empire, and also decreasing it causes the colonies to move closer to the vector connecting the colony to the imperialist as much as possible. Considering the unit radians for γ, a number close to $\pi/4$ is a good choice in most implementations.

The difference between the assimilation policy in this algorithm and the one in ICA uses 4 different values for β instead of using one constant assimilation parameter of β, and the method of allocating β will be defined in each iteration by inspiration from the Tabu search. It means that in each iteration, one of the β-s is selected randomly but there is restriction applied to this random selection. If β-$_j$ is used in the i^{th} iteration of the algorithm, using β-$_j$ is forbidden in the $(i+1)^{th}$ iteration, and another coefficient must be used. This will lead to a different space to search around the solution, and also will prevent being trapped in the local minimum because by choosing different and non-repetitive coefficients, the assimilation steps have a wider variety of values.

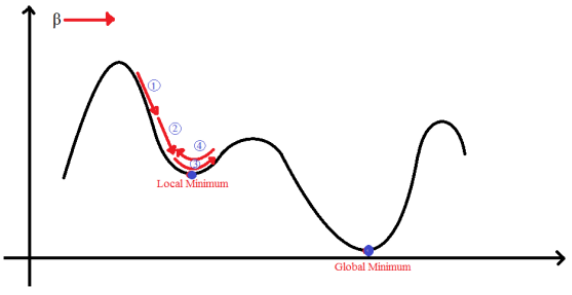

Figure 4. Assimilation with constant β and trapping in a local minimum.

Figure 4 shows the state that colonies move toward the empire by a statistic β, which means that the assimilation parameter is constant.

Assume that the red arrow is the value for β (absorption parameter); with two steps, it could reach the local minimum, and with the third step, it would pass it, although as the slope is toward the local minimum, it will return to the local minimum with that step, and this sweep will be repeated. It is where we say the algorithm is trapped in the local minimum. Now suppose the state in which the values for β are different; for instance, in figure 5, assume the assimilation parameter with 4 different values and different steps in each iteration. If we use the non-repeated coefficients β_1 and β_3, respectively, in the moving steps, the algorithm gets closer to the local minimum; in the next step, the algorithm could not use β_3. Assume that the algorithm randomly uses β_4. Therefore, the algorithm will pass the local minimum, and will continue its path toward the global minimum. Thus this will prevent being trapped in the local minimum.

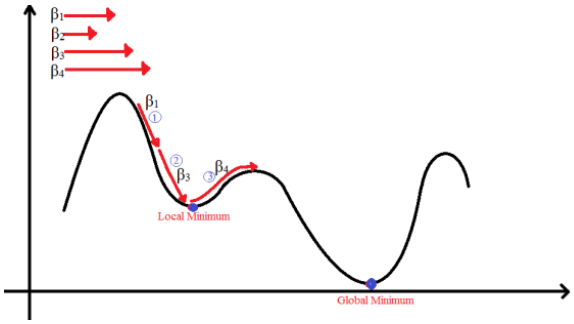

Figure 5. Assimilation with variable β and escape from the local minimum.

As shown in the figures, by using this method, we can avoid getting trapped in the local minimum in many cases. This method also searches in a bigger search space. This will make the algorithm to find the solution faster than the assimilation policy in ICA. In figure 6, you can see the uncertain assimilation algorithm.

```
1.   Beta=[0.5 , 1 , 1.3 , 2.9]
2.   For each colony C
3.   C.Beta = 0;
4.   Variable Beta Assimilation( )
5.   {  for each Empire
6.        for each colony as C  in empire
7.          index = random(1,4)
8.          while (C.Beta == index)
9.            index = random(1,4)
10.           C.Beta = index
       Assimilate C into Imp(Empire) with C.Beta parameter }
```

Figure 6. Assimilation algorithm with variable values for beta.

3.3. Hybrid algorithm (ICA-Fight-β)

Now you can see in figure 7 the new hybrid algorithm. This algorithm is used instead of the original assimilation of the new assimilation function, The FIGHT function is also added to the collection of operations inside the main loop. The changes in the algorithm are shown below in bold.

```
1. Initialization;                    // Create Initial Empires
2. {                                  // The beginning of the loop iterations
3. Variable Beta Assimilation ( );    //Assimilation with Variable Beta
4. Do Revolution ( );                 // Revolution
5. Intra Empire Competition ( );      // Intra-Empire Competition
6. Fight Function ( );                // Fight
7. Update Total Cost ( );             // Update Total Cost of Empires
8. Inter Empire Competition ( );      // Inter-Empire Competition
9. Update  Best Solution ( );         //Update Best Solution Ever Round
If the stop condition is satisfied, stop; if not, go to 2.}
```

Figure 7. ICA-Fight-β algorithm.

You can also view the new algorithm flowchart in figure 8; two new parts are different in the flowchart.

4. System simulator characteristics

To implement the algorithm, its simulation was run in MATLAB on a system with the following hardware and software characteristics:
CPU: AMD Turion™ 64 X2 Mobile Technology TL-64 2.2 GHz
RAM: 2 GB, OS: Windows 7 (32 bit), Version of MATLAB: 7.8.0.347 (2009)

5. Benchmark functions

In the recent years the growing need to optimize the cost in many engineering problems, has caused many optimization algorithms to be introduced by the researchers, but the more important issue is the evaluation method of the algorithm. One of the helpful methods used for analyzing the performance of these algorithms is to test them on different functions called test functions or benchmark functions. In applied mathematics, the evaluation functions are known as an artificial view which is useful for evaluating the features of the optimization algorithms such as the speed of convergence, accuracy, stability, and general performance. Here, we introduce some test functions that could be used as a goal function in the optimization case with a single goal [21-25].

Some test functions are shown in table 2. Also, you can see in figure 9 the graph for a bumpy function.
As you can see in figure 9, some of these benchmark functions contain too many minimal and maximal points, that it is a suitable criterion for evaluating the optimization algorithm. Since these functions simulate a state space with multiple local optimal, so they can evaluate algorithm performance.

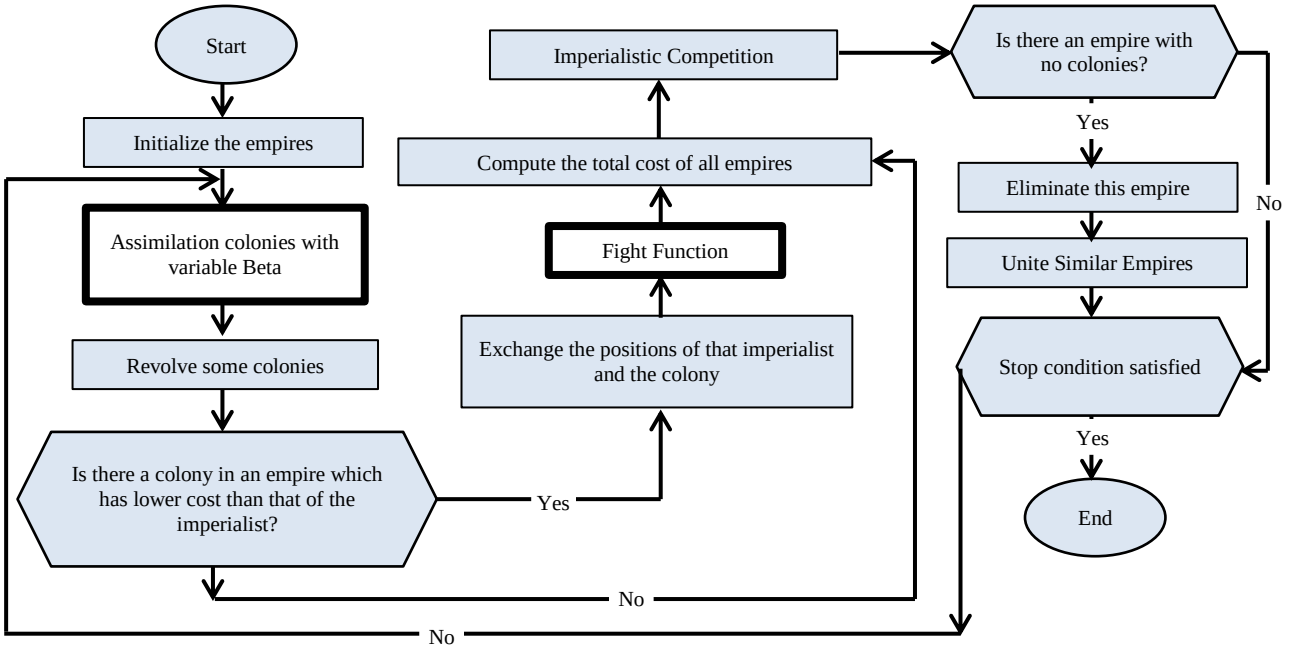

Figure 8. ICA-Fight-β flowchart.

Table 2. Benchmark Functions.

Function name Range Parameters	Formula function
SPHERE	$f(x)=\sum_{i=1}^{d} x_i^2$
SHUBERT	$f(x)=(\sum_{i=1}^{5} i\cos((i+1)x_1+i))(\sum_{i=1}^{5} i\cos((i+1)x_2+i))$
BOOTH	$f(x)=(x_1+2x_2-7)^2+(2x_1+x_2-5)^2$
Sum Squares	$f(x)=\sum_{i=1}^{d} ix_i^2$
Rotated Hyper-Ellipsoid	$f(x)=\sum_{i=1}^{d}\sum_{j=1}^{i} x_i^2$
THREE-HUMP CAMEL	$f(x)=2x_1^2-1.05x_1^4+\frac{x_1^6}{6}+x_1x_2+x_2^2$
RASTRIGIN	$f(x)=10d+\sum_{i=1}^{d}[x_i^2-10\cos(2\pi x_i)]$
De Jong N.3	$f(x)=\sum_{i=1}^{n}\lvert x_i\rvert$
Bumpy	$f(x,y)=x*\sin(4x)+1.1y*\sin(2y)$

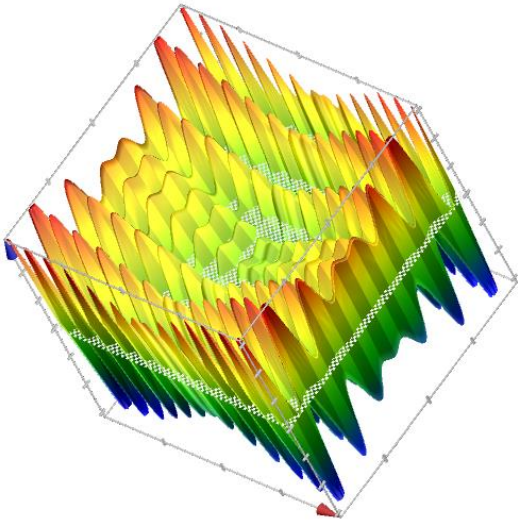

Figure 9. Bumpy function diagram.

6. Evaluation of scenario and evaluation criteria

As we know in order to assess the participant, having suitable criteria is the necessity for comparison. In the optimization algorithms, the goal is to find the optimum solution among a set of candidate solutions. Therefore, in the optimization algorithms, the qualified algorithm is the one that can find the solution faster than the others. However, as there is a possibility of not finding the optimum solution in heuristic algorithms, consider the possibility of not-finding the optimum solution in defining the criteria. Now we introduce the performance assessment criteria for the optimization algorithms. The two criteria

response time and number of iterations are expressed to evaluate the performance of the proposed idea in comparison with the other 4 algorithms ICA, SBA[2], COA[3] and GA[4] as follows:

6.1. Number of iterations for optimization algorithm

In the heuristic optimization algorithms, a special action, the optimization iteration is carried out repeatedly. One of the criteria used to evaluate the performance of the optimization algorithm is to compare the number of repetitions until an optimum global answer is achieved. For example, if algorithm A with *100* iterations and B with *200* ones get the optimum solution, then the number of iterations in algorithm A is better. Table 3 contains the number of iterations required to achieve the optimum solution using the ICA-Fight-β algorithm and the algorithms ICA, SBA, COA, and GA. It is to be noted that each iteration value in table 3 was obtained from an average of *10* tests. Figure 10 represents a comparative diagram of them.

By two-by-two comparisons between ICA-Fight-B and other algorithms, we will have an interesting result. As it can be seen, ICA-Fight-β reached the optimum solution in all cases, while the classical ICA algorithm in 4 cases and both the COA and GA algorithms in 3 out of *13* evaluation cases received the optimal results in less than *200* iterations (illustrated with a star (*) in Table 3). The rest of the cases did not obtain the optimal solution, even in *200* repetitions.

Table 3. Number of iterations of different algorithms until achieving the optimum or up to 200 steps.

	Evaluation functions	Optimization algorithms				
		ICA	SBA	COA	GA	ICAFightB
1	Sphere	200	200	200	200	99.3
2	Rastrigin	200	200	18.4 *	200	50.1
3	DeJong3	200	200	200	200	95.8
4	DeJong4	200	200	200	200	102.6
5	Sumsqu	200	200	200	200	98.8
6	Booth	200	200	200	200	116.6
7	boha1	58.2 *	111	28 *	72.8 *	34
8	Function9	47.4 *	82.9	189.1 *	157.4 *	41.9
9	Bumpy	15.6 *	22.8	200	200	10.3
10	Shubert	16.3 *	30.6	200	21.6 *	20.3
11	camel3	200	200	200	200	71.2
12	Rothyp	200	200	200	200	99.9
13	Sumpow	200	200	200	200	33.1

[2] Social-Based Algorithm
[3] Cuckoo Optimization Algorithm
[4] Genetic Algorithm

It is noticeable that in the cases where these algorithms reach the optimal answer, the ICA algorithm in one case (shubert function) and the COA algorithm in two cases (rastrigin and boha1 functions) achieve the optimal answer in a less number of iterations rather than the ICA-Fight-β algorithm (illustrated with black square in table 3). In the rest of the cases, the proposed algorithm is conclusively excellent. Also the new algorithm is better than SBA in all cases. This represents an improvement in the proposed method.

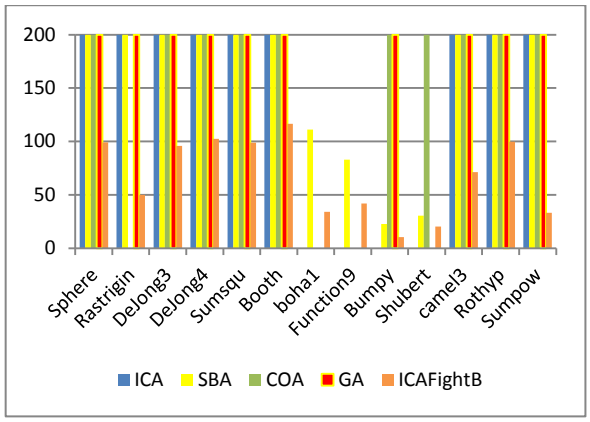

Figure 10. Comparison chart of optimization algorithms in terms of number of iterations to achieve optimal solution.

6.2. Runtime (response time)

Another evaluation parameter for the optimization algorithms is their runtime. To do so, we can choose either the time required to find the optimum answer or the time needed to execute a fixed number of iterations for comparison. It is clear that this time depends on the features of the implementation system of the algorithm. Therefore we must use the same system to compare different algorithms.

Table 4 shows the response time for each algorithm to achieve the best answer. Here, a few required points are being recalled and mentioned. First to calculate the time, the tic and toc functions are used in MATLAB. This time depends on the efficiency of the simulator system. Secondly, like the previous criteria, each time is obtained from an average of *10* tests. Finally, what is being illustrated in table 4 shows the response time of the algorithms in seconds. It should be noted that the calculated values are the time required to reach the optimal solution or the time for *200* iterations. According to the assumption of maximum *200* iterations in the simulation scenarios, if the algorithm does not achieve the optimal solution in *200* iterations, it ends, and this time is considered for the algorithm.

Table 4. Response time of optimization algorithms until achievement of optimal solution.

	Evaluation function	Optimization algorithms				
		ICA	SBA	COA	GA	ICAFightB
1	Sphere	1.6464	27.7335	13.1467	2.7048	11.1724
2	Rastrigin	1.6736	34.5356	0.9266 *	2.8298	6.9239
3	DeJong3	1.4891	28.2778	12.893	2.7969	12.4593
4	DeJong4	1.7244	30.0464	13.272	2.8908	17.7707
5	Sumsqu	2.617	28.5281	12.5234	2.3477	13.9416
6	Booth	2.0865	24.7035	12.477	2.515	3.829
7	boha1	0.9885 *	15.9542	1.2693 *	1.0023 *	1.0209
8	Function9	0.8729 *	12.5166	12.3701 *	2.1185 *	1.6263
9	Bumpy	0.3388 *	3.7347	14.0516	2.5638	0.3196
10	Shubert	0.3332 *	4.5835	14.583	0.3639 *	0.8095
11	camel3	2.346	28.5622	12.3894	2.6	2.4101
12	Rothyp	1.6078	30.197	12.4751	2.7618	17.9215
13	Sumpow	1.68	30.88	12.5814	2.7509	22.0118

Since in most evaluations, the algorithms do not achieve the optimal point in up to 200 iterations (Table 3), the algorithms achieve the optimal points in less than *200* iterations, (illustrated with star (*) in Table 4). The ICA-Fight-β algorithm reached the optimal point in all the investigated evaluation functions, and so the increase in the response time compared with the ICA, SBA, COA, and GA algorithms is not disconcerting because, in fact, the algorithm times must be compared when both algorithms get to the optimal solution. Otherwise, the algorithm that has achieved the optimal solution is superior to the others.

By comparing the ICA-Fight-β algorithm with the other ones in terms of the response time, the ICA algorithm has a lower response time in 3 out of 4 cases which achieved to the optimal solution (illustrated with star (*) in Table 4). In the remainder, it does not reach the optimal solution or its response time is higher, and this is due to the some functions are added to the proposed method compared with ICA. Compared with the COA and GA algorithms, in only one case and two cases, respectively (illustrated with star (*) in Table 4), these algorithms reach the optimum solution faster than ICA-Fight-β algorithms, and in the other cases, the optimal solution is not reached or the ICA-Fight-β algorithm is superior. Also the new algorithm is better than SBA in all cases.

7. Conclusion

After analysis of evaluation results, it was found the proposed algorithm is more efficient than the other algorithms since the ICA-Fight-B reaches to the optimal solution in less iteration compared with the ICA and SBA; this algorithm finds the optimum solution in a shorter response time and it is superior 100%. Also in comparison with COA, in more than 87% cases, the ICA-Fight-B algorithm finds the solution in fewer iterations. In terms of the response time, the presented algorithm has a better response time in 93% of cases in comparison with ICA, GA, and COA, because the ICA-Fight_B achieves to answer faster than other algorithms the exception in 1 case out of 13 cases.

The ICA-Fight-B algorithm is the first version of a hybrid algorithm based on a social-political process and a process inspired by real world war. Thus some changes in the algorithm may lead to an improvement in its performance an application. The proposed algorithm is suitable to solve continuous optimization problems for now. In order to solve the discrete optimization problems, some manipulations are needed in the algorithm. Presenting the discreet version of the algorithm is helpful for solving problems like choosing inputs in systems diagnosing, feature selection in pattern recognition, and traveling salesman. One other task for future works is to use a dynamic parameter β instead of a constant β or β with limited values such that, depending on the distance from the optimum point, the algorithm could use a suitable β to move because even in the real world, assimilation toward the global optimum is dynamic.

References

[1] Weise, T. (2009). Global Optimization Algorithms– Theory and Application. Germany: it-weise.de (self-published).

[2] Voß, S., Martello, S., Osman, I. H. & Roucairol, C. (1998) . Meta-heuristics: Advances and Trends Local Search Paradigms for Optimization. New York: Springer.

[3] Grosan, C. & Abraham, A. (2007). Hybrid Evolutionary Algorithms: Methodologies, Architectures and Reviews. Springer-Verlag, New York, pp. 1–17.

[4] Chan K.Y., et al. (2010) . A new orthogonal array based crossover with analysis of gene interactions for evolutionary algorithms and its application to car door design. Expert Systems with Applications, vol. 37, no. 5, pp 3853-3862.

[5] Atashpaz-Gargari, E. & Lucas, C. (2007). Imperialist Competitive Algorithm:An Algorithm For Optimization Inspired By Imperialistic Competition, 2007 IEEE Congress on Evolutionary Computation , Singapore, 2007.

[6] Jain, T. & Nigam, M.J. (2010). Synergy of evolutionary algorithm and socio-political process for global optimization. Expert Systems with Applications, vol. 37,no. 5, pp. 3706-3713.

[7] Khorani, V., Razavi, F. & Ghoncheh, E. (2010). A New Hybrid Evolutionary Algorithm Based on ICA and GA: Recursive-ICA-GA. World Comp2010, California, USA, 2010.

[8] Abdechiri, M. & Meybodi, M. R. (2011). A Hybrid Hopfield Network-Imperialist Competitive Algorithm for Solving the SAT Problem. 3th International Conference on Signal Acquisition and Processing, Singapore, 2011.

[9] Niknama T., et al. (2011). An efficient hybrid algorithm based on modified imperialist competitive algorithm and K-means for data clustering. Engineering Applications of Artificial Intelligence. vol. 24, no. 2, pp. 306-317.

[10] Pooranian Z., et al. (2011). New hybrid algorithm for task scheduling in grid computing to decrease missed task. World Acad Sci Eng Technol, vol. 55, no. 1, pp. 5-9.

[11] Ramezani F., et al. (2012).A Hybrid Evolutionary Imperialist Competitive Algorithm(HEICA). Computer Science, vol. 1, no.1, pp 359-368.

[12] Ramezani, F. & Lotfi, S. (2013). Social-Based Algorithm (SBA). Applied Soft Computing, vol. 13, no. 5, pp. 2837-2856.

[13] Lepagnot, J., et. al. (2013). Hybrid Imperialist Competitive Algorithm with Simplex Approach: Application to Electric Motor Design. 2013 IEEE International Conference on Systems Man and Cybernetics, Manchester, United Kingdom, 2013.

[14] Jalal Nouri, D., Saniee Abadeh, M. & Ghareh Mohammadi, F. (2014). HYEI: A New Hybrid Evolutionary Imperialist Competitive Algorithm for Fuzzy Knowledge Discovery. Advances in Fuzzy Systems. vol. 14, no. 11, pp.387-395.

[15] Shamshirband S., et al. (2014). D-FICCA: A density-based fuzzy imperialist competitive clustering algorithm for intrusion detection in wireless sensor networks. Measurement, vol. 55, no. 1, pp. 212-226.

[16] Shamshirband S., et al. (2014). Anomaly Detection using Fuzzy Q-learning Algorithm. Acta Polytechnica Hungarica, vol. 11, no. 8, pp.5-28.

[17] Shojafar M., et al. (2015). FUGE: A joint meta-heuristic approach to cloud job scheduling algorithm using fuzzy theory and a genetic method. Cluster Computing, vol. 18, no. 2, pp. 829-837.

[18] Roozbeh Nia, A., Hemmati Far, M. & Akhavan Niaki, S.T. (2015). A hybrid genetic and imperialist competitive algorithm for green vendor managed inventory of multi-item multi-constraint EOQ model under shortage. Applied Soft Computing, vol. 30, no. 1, pp. 353-364.

[19] Mehdinejad M., et al. (2016).Solution of optimal reactive power dispatch of power systems using hybrid particle swarm optimization and imperialist competitive algorithms. International Journal of Electrical Power & Energy Systems, vol. 83, no. 1, pp. 104-116.

[20] Valipour, Kh. & Ghasemi, A. (2016). Using a new modified harmony search algorithm to solve multi-objective reactive power dispatch in deterministic and stochastic models. Journal of AI and Data Mining, vol. 5, no. 1, pp. 89-100.

[21] Li X., et al. (2013). Benchmark functions for the CEC 2013 special session and competition on large-scale global optimization. Gene, vol. 7, no. 33, pp. 87-94.

[22] Molga, M. & Smutnicki,C. (2005). Test functions for optimization needs. Available: http://www. zsd.ict.pwr.wroc.pl/files/docs/functions.pdf

[23] The GEATbx website, (2005), Available: http://www.geatbx.com/download/ GEATbx_ObjFunExpl_v37.pdf.

[24] Adorio, E. P. & Diliman, U. P. (2005). MVF: Multivariate Test Functions Library in C for Unconstrained Global Optimization. Available: http://www.geocities.ws/eadorio/mvf.pdf

[25] SheYang, X. (2010) Test Problems in Optimization in Engineering Optimization: An Introduction with Metaheuristic Applications. New York: John Wiley & Sons.

یک الگوریتم فرا ابتکاری ترکیبی جدید جهت حل مسائل بهینه‌سازی مبتنی بر الگوریتم رقابت استعماری

رسول روستائی* و فاطمه یوسفی فخر

[1]گروه مهندسی کامپیوتر، واحد ملایر، دانشگاه آزاد اسلامی، ملایر، ایران.

گروه مهندسی کامپیوتر، واحد ملایر، دانشگاه آزاد اسلامی، ملایر، ایران.

ارسال ۲۰۱۵/۱۲/۰۴؛ بازنگری ۲۰۱۶/۰۷/۲۱؛ پذیرش ۲۰۱۶/۰۹/۱۴

چکیده:

همواره بشر به دنبال یافتن بهترین‌ها در تمامی امور بوده است. این کمال طلبی منجر به پیدایش روش‌های بهینه‌سازی گردیده است. هدف از بهینه‌سازی تعیین متغیرهای مساله و یافتن بهترین جواب قابل قبول، با توجه به محدودیت‌های مسأله است، به گونه‌ای که تابع هدف کمینه یا بیشینه شود. یکی از روش‌های غیردقیق بهینه‌سازی، روش‌های فرا ابتکاری است که معمولا با الهام گرفتن از طبیعت بدنبال جواب بهینه هستند. درسال‌های اخیر تلاش‌های زیادی برای بهبود و یا ایجاد الگوریتم‌های فرا ابتکاری صورت پذیرفته است. یکی از راه‌های ایجاد بهبود در روش‌های فراابتکاری استفاده از ترکیب است. در این مقاله یک الگوریتم بهینه‌سازی ترکیبی مبتنی بر الگوریتم رقابت استعماری ارائه می‌شود. ایده‌های بکاررفته دراین الگوریتم عبارت است از عملیات جذب با پارامتر متغییر و استفاده از تابعی به نام جنگ که مبتنی بر مدل ریاضی جنگ در دنیای واقعی است. این تغییرات منجر به افزایش سرعت و کاهش مراحل جستجو درمقایسه با دیگر الگوریتم‌های فرا ابتکاری می‌شود، به گونه‌ای که در ارزیابی‌های انجام شده دربیش از ۸۰٪ موارد آزمون، در قیاس با الگوریتم‌های رقابت استعماری، جستجوی فاخته و الگوریتم مبتنی براجتماع برتری با الگوریتم ارائه شده بود.

کلمات کلیدی: بهینه‌سازی ترکیبی، الگوریتم رقابت استعماری، الگوریتم فرا ابتکاری، بهینه سراسری، بهینه محلی.