Optimization
Iranian Journal of Optimization
Volume 9, Issue1, 2017, 21-37
Research Paper
Online version is available on: www.ijo.iaurasht.ac.ir

Islamic Azad University
Rasht Branch
E-ISSN:2008-5427

MOEICA: Enhanced Multi-Objective Optimization based on Imperialist Competitive Algorithm

AmirAli Nazari* and Ali Deihimi

Bu-Ali Sina University, Department of Electrical Engineering, Hamedan, Iran

Received: 28 January 2017
Accepted: 14 April 2017

Abstract

In this paper, a multi-objective enhanced imperialist competitive algorithm (MOEICA) is presented. The main structures of the original ICA are employed while some novel approaches are also developed. Other than the non-dominated sorting and crowding distance methods which are used as the main tools for comparing and ranking solutions, an auxiliary comparison approach called fuzzy possession is also incorporated. This new provision enables more countries to participate in guiding the population towards different searching routs. Moreover the computational burden of the algorithm is abated by carrying out the hefty sorting process not at each iteration but at some predefined intervals. The frequency of which is controlled by on optional parameter. Furthermore, the recreation of empires and imperialists several times during the optimization progress, encourages better exploration and less chance to get trapped in local optima. The eligibility of the algorithm is tested on fifteen benchmark functions in terms of different performance metrics. The results through the comparison with NSGA-II and MOPSO shows that the MOEICA is a more effective and reliable multi-objective solver with being able to largely cover the true Pareto fronts (PFs) for the test functions applied in this article.

Keywords:
multi-objective ICA
Pareto front coverage
performance metrics
benchmark functions

***Correspondence E-mail**: aa.nazari.eng@.com

INTRODUCTION

In many scientific researches and engineering applications, optimization is an essential part (Lin et al., 2015, Fattahi et al., 2014). If an optimization problem comprises M objective functions, N_{dim} variables, N_e equality and N_i inequality constraints, it can be formulated according to following terms:

$$\text{Minimize/Maximize } F(X)$$
$$F(X)=F(f_1(X),f_2(X),\ldots,f_M(X)) \quad (1)$$

$$\text{Where } X=\{x_1,x_2,..,x_{NDim}\} \quad (2)$$

$$\text{subject to } \begin{cases} g_i(X) = 0 & i = 1,\ldots,N_e \\ h_j(X) \le 0 & j = 1,\ldots,N_i \end{cases} \quad (3)$$

In the general form, if the goal is to optimize only one objective then the problem is categorized into the single objective problems (SOPs). In contrast, multi-objective problems (MOPs) aim at optimizing two or more objectives simultaneously. In a MOP, rather than locating a global optimal value, a set of optimal solutions are determined. Generally these points are non-dominated solutions, meaning that, no other point in the discovered space can be found to improve the quality of any objective without degrading the performance of at least another one (Xiang, 2015). The set of all non-dominated parameter vectors is called the Pareto optimal set and the corresponding set of objective vectors is the Pareto optimal front or the PF. The Pareto optimal set is a subset of the search space, whereas the PF is a subset of the objective space. The PF forms a curve or surface presenting all possible trade-offs between the objectives. Hence it is up to the decision maker to choose between intermediate or extreme points as the final solution to the problem. Traditional deterministic approaches have a hard time to travel the entire solution space and find a satisfactory result without a huge computational effort. Therefore many scientific researches incorporate meta-heuristics for exploring multi-dimensional search space. A specification of these approaches is that there is no guaranty to find the true PF if it is not already available as in the case of real world problems. Non-dominated solutions are claimed to be optimal only compared to the solutions discovered so far. Thus seeking the true PF is an infinite process in which the solutions are improved through time progress. Multi-objective evolutionary algorithms (MOEAs) are the most commonplace meta-heuristic tools for dealing with MOPs (Durillo et al., 2010 ؛Coello et al., 2007). They are suitable for complex problems which include a large number of variables and objectives (Fan, 2015). Moreover, when the number of required Pareto solutions is increased, a MOEA can perform better than a classical optimization method (Ghiasi et al., 2011). Genetic algorithm (GA) has been utilized in many MOEAs including non-dominated sorting genetic algorithm (NSGA) (Srinivas & Deb, 1994) and fast elitist non-dominated sorting genetic algorithm (NSGA-II) (Deb et al., 2002). Particle swarm optimization (PSO) has also attracted attention due its simplicity and flexibility. Its basic multi-objective variant, MOPSO (Coello & Lechuga, 2002) has been implemented successfully on many engineering applications (Sivasubramani & Swarup, 2009؛ Chandrasekaran et al., 2007). A relatively new evolutionary algorithm called imperialist competition algorithm (ICA), has been proposed by Atashpaz and Lucas in 2007. The algorithm benefits from the capability to escape local minima by creating several search spaces thanks to the idea of empires. In contrast with the PSO, the individuals in the population do not move toward a single global optima, but rather a number of elites called imperialists. This approach removes the concentration on a specific region of the search space and helps for a better exploration. ICA has been widely implemented and improved in single objective optimizations (Khabbazi et al., 2009؛ Shokrollahpour et al., 2011) , but few attempts have been made to utilize it in multi-objective optimization area. Enayatifar et al. (2013), developed a MOEA based on the ICA and demonstrated its effectiveness on several benchmark functions. The results were compared with those of the NSGA-II and MOPSO and proved to be better or at least on par. The authors did not use any external archive and only relied on the non-dominated solutions from the population. Idoumghar et al. (2013), proposed a hybrid ICA-PSO algorithm by incor-

Algorithm 1. The original ICA

1- Initializing a population consisting of all the countries
2- Evaluating costs of all countries
3- Choosing the N_{Imp} better countries as imperialists
4- Allocating colonies to empires based on empires' powers.
5- **While** an ending criterion is not satisfied
6- Moving colonies towards their corresponding imperialists (assimilation)
7- Evaluating costs of all colonies in empires
8- **If** a colony has a better objective function than its corresponding imperialist
9- the positions of the colony and the imperialist are exchanged (possession)
10- **End If**
11- Replacing certain percent of colonies in each empire by random points (revolution)
12- Taking one colony from the weakest empire and giving it a stronger one (imperialistic competition)
13- **If** an empire has no colonies
14- Eliminate that empire
15- **End If**
16- **End While**

porating important aspects of both algorithms. They used a fixed-size external archive to store non-dominated solutions in multi-objective realization of their algorithm. In their proposed structure, the crowding distance method was used to keep the number of archive members constant. Though two former studies are proper enough to be implemented in any practical optimization area, for the development of the ICA as a MOEA, further ideas and improvements seem to be welcome. Hence the main concern of this paper is to present a multi-objective version of the ICA, named MOEICA and validate its advantageous in comparison with other MOEAs. In order to enhance its performance in exploration, exploitation and speed, some strategies are offered within the main structure and others are devised as totally new extensions. The rest of the paper is divided into the following parts. In Section 2 the very original ICA is described. In sections 3, the proposed MOEICA is presented. The benchmark functions and performance metrics which are used for the assessment of the algorithms are explained in Section 4. The numerical results are presented and discussed in section 5 and finally the conclusions are made in section 6.

ORJINAL ICA

ICA is an example of population based algorithms. The algorithm has been inspired by the idea of dividing countries into imperialists and colonies. Individuals in ICA are called countries. Countries are within different parts of the problem space which are called empires. Strongest country in an empire is called the imperialist and other countries within that empire are named colonies. Iterations are called decades. Imperialists tend to attract colonies towards themselves in every decade. The procedure of the original ICA is shown in Algorithm 1.

Creating empires

After the initialization, according to the procedure of Algorithm 1, the N_{Imp} best countries are chosen as imperialists. Thereafter other countries must be allocated to empires in the name of colonies. It is necessary for an empire whose imperialist has a better fitness value to collect more colonies. In the original ICA, the power of each empire is calculated by the following equations:

$$P_i = \frac{C_i}{\sum_j^{N_{Imp}} C_j} \tag{4}$$

$$C_i = c_i - max(c_k) \tag{5}$$

Where c_i is the ith imperialist cost and C_i is its normalized cost. Having the power of all empires, the number of allocated colonies to each empire is calculated by

Eq .6

$$N_{Col,i} = round[(n_{Pop} - N_{Imp}) \times P_i] \tag{6}$$

Assimilation

In the assimilation stage, colonies move to-

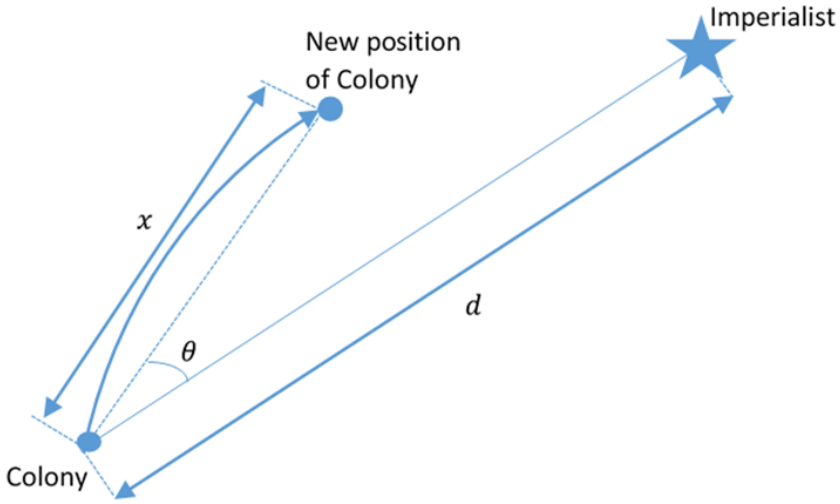

Fig. 1. Movement of a colony in the assimilation stage

wards their corresponding imperialists as shown in Fig. 1. The length of the movement is equal to x which is a random number between 0 and β×d. The quantity of β which is the movement coefficient is arbitrary but values greater than 1, causes the colonies to get closer to the imperialists from all sides (dimensions) (Atashpaz-Gargari & Lucas, 2007, Idoumghar et al., 2013) . The θ parameter also acts as a deviation angle. In that way the colony is able search different points around the imperialist.

Possession

After the assimilation stage, colonies acquire new positions in the problem space. Hence their cost functions must be evaluated again. Then if a colony is proved to have a lower cost than its relevant imperialist, their positions must be exchanged. Thus in the next decade, algorithm will continue by the new imperialist and then colonies start moving toward its position.

Imperialistic competition

It is intended in the procedure of the original ICA to gradually weaken the relatively powerless empires in favor of more powerful ones. In each decade, the weakest colony in the weakest empire is removed and given to an empire which has the most probability to possess it. To carry out the competition, the two following vectors are created:

$$P=[P_1,P_2,\ldots,P_{Nimp}] \quad (7)$$

$$R=[r_1,r_2,\ldots,r_{Nimp}] \quad (8)$$

P is a vector including the powers of all empires. R is a vector with the same size as P, created by random numbers between 0 and 1. Finally vector D is created by subtracting R from P.

$$D=[p_1-r_1,p_2-r_2,\ldots,p_{Nimp})-r_{Nimp}] \quad (9)$$

The empire whose relevant index in D is maximum will win the colony in the competition. The algorithm continues until all empires but one have eliminated. Then the imperialist in the remaining empire will be the founded global optimum.

MOEICA

The aim here is to present an MOEA based on the single objective ICA. Some additional techniques are required in this area to make the structure compatible with multi-objective problems. The first challenge is to devise a method for creating the empires after the initialization. Rather than finding the lowest cost associated individuals as in the original algorithm, here the non-dominated solutions are to be found. Then the non-dominated sorting method, of which the procedure is illustrated in Algorithm 2 is employed here. This method allocates a rank to every country in the population. The countries within the first rank form the non-dominated solutions. In order to discriminate between the countries within the same rank, the crowding distance method is also incorporated. Its implementation is described in Algorithm 3. This method calculates the distance of a country from its neighbors in the objective space. In this approach, countries possessing higher crowding distance values are more desirable, because it means that they are located in a less crowded area. There-

after similar to the original version of the algorithm, the best N_{Imp} countries will be chosen as imperialists. It will include countries with lower ranks and higher crowding distance values. The rank parameter has the prominent role in the sorting. In other words, worst country of a lower rank, has the priority over the best country in a higher rank (Algorithm 2 and Algorithm 3).

Empires and external archive

In the next step, powers of all the empires should be determined. Here we solely rely on the imperialist place in the sorted population to calculate the power of its corresponding empire.

$$P_i = \frac{exp\left[-k_{Power}\left(\frac{SPlace_i}{N_{Imp}}\right)\right]}{\sum_{j=1}^{N_{Imp}} exp\left[-k_{Power}\left(\frac{SPlace_j}{N_{Imp}}\right)\right]} \quad (10)$$

Algorithm 2. The non-dominated sorting

```
For i ϵ N_Pop
    Si=Ø
    Ni=0
    F1=Ø
    For j ϵ NPop
        If Countryi dominates Countryj
            Si=Si∪ {j}
        Else If Countryj dominates Countryi
            Ni=Ni+1
        End If
        If Ni=0
            Countryi.Rank=1
            F1=F1∪ {i}
    End If
  End For
End For
r=1
While Fr≠0
    Q=Ø
        For i ϵ Fr
          For j ϵ Si
            Nj=Nj-1
            If Nj==0
                Countryj.Rank=r+1
                Q=Q∪ {j}
              End If
          End For
      End For
      r=r+1
      Fr=Q
End While
```

$SPlace_i$ is the ith imperialist place after sorting. k_{power} is a coefficient determining the effect of the rank and crowing distance of an imperialist on the power of its corresponding empire. After calculating the powers of all empires, allocation of colonies to the empires is the same process as in section 2.1 and is done by Eq.6. Since imperialists in all empires undergo changes if been dominated by some colonies, in order to preserve the elitism in the algorithm an external archive is required. Hence an archive with dynamic size is used to procure best solutions in the whole optimization process. As Deb et al. (2002) have discussed in their paper, the non-dominated sorting process is the main computational effort in algorithms which use it. This complexity is related to the one by one dominance comparisons in the sorting process. In NSGA-II the act of the sorting is carried out twice in each iteration. Then it makes the algorithm computationally expensive. To alleviate this problem, in our proposed multi-objective algorithm, there is this flexibility to do the sorting process after each optional predefined number of decades (N_{RC}). This option lowers the runtime of the algorithm drastically and does not disintegrate the non-dominated solutions thanks to the presence of the external archive. But still increasing the N_{RC} too much can deteriorate the quality of solutions because the diversity of the imperialists is checked less frequently in subsequent decades. Then an intermediate value should be chosen for N_{RC} to establish a trade-off between speed and accuracy of the algorithm. It must be noted that by reaching each multiple of N_{RC}, the external archive is updated. The positions of all countries within the archive are overridden by those countries that are in the first rank after sorting. The empires are also reset. Then the imperialistic competition stage is an unnecessary effort to be implemented here. The constituted structure so far, is able to evaluate solution candidates in a multi-objective environment and store non-dominated solutions in subsequent iterations. Other stages of the original ICA can also be imported here unchanged. But in this paper, some strategies are also suggested to enhance the performance of the algorithm relating to those specific stages.

Assigning membership functions

If the colonies in each empire are sorted according to their objective values, it will be beneficial in many areas of the algorithm. It is intended here to give a single fitness value to

Algorithm 3. The crowding distance

```
Let R be the number of available ranks
Let M be the number of objectives
For r ϵ R
    Let Ar be the set of countries in rth rank
    Let Cr be the objective array of the countries in rth rank
    For m ϵ M
        Let SCost be the sorted mth objective values for countries in rth rank
        Let SInd be the sorted indexes regarding to SCost
        d(SInd(1),m)=inf
        d(SInd(end),m)=inf
        For k ϵ {2:size(Ar )-1}
            d(SInd(k),m)=[SCost(k+1)-SCost(k-1)]/[SCost(end)-SCost(1)]
        End For
    End For
    For i ϵ Ar
        Countryi.CD=sum(d(i,:))
    End For
End For
```

each colony in an empire. In this way the colonies in empires can be sorted. Incorporating the non-dominated sorting and crowding distance methods in this area is a hefty procedure and is not recommended. The considered approach is to assign membership functions. For every objective of the problem, a membership function is defined using the fuzzy membership function concept. The membership function changes linearly from zero to one in a predefined interval. The value of one implies that the specific objective has been satisfied completely and the value of zero means that the specific objective has not been satisfied at all. Then to acquire these characteristic points, the problem should be solved for every objective separately. When a solution point, satisfies an objective to some extent, a membership function between zero and one is dedicated to it by the following equation.

$$\mu(f_i) = \begin{cases} \dfrac{f(i)_{worst} - f(i)}{f(i)_{worst} - f(i)_{best}} \\ 0 \\ 1 \end{cases}$$

$$if \;\; f(i)_{best} \le f(i) \le f(i)_{worst}$$

$$if \;\; f(i) > f(i)_{worst}$$

$$if \;\; f(i) < f(i)_{best}$$

$$\forall\, i = 1, \dots, M \qquad (11)$$

$f(i)_{best}$ and $f(i)_{worst}$ are respectively the best and worst possible quantities for ith objective. $\mu(f_i)$ is then the membership value for the ith objective. In order to merge two or more objective functions, there are two possible ways. 1- Calculating the minimum of all membership functions 2- Calculating the mean of all membership functions. The first approach emphasizes on diversity, while the second focuses on convergence. In this stage, one of these two methods is chosen randomly. Then in each empire, to every colony (and the imperialist), a total membership function *(TMF)* is dedicated.

Enhanced assimilation

In this step, only a fraction of colonies in each empire participate in the assimilation which is controlled by the Ass_{Rate} parameter. For the rest of the colonies, another approach is intended, of which the description is proposed later. As shown in section 2.2, the β parameter plays an important role in the assimilation process. The authors of this papers believe that a single constant value for β cannot yield a satisfying performance. Here different movement coefficients are suggested based on the progress of the algorithm and the colony rank within its corresponding empire. Note that the assigned ranks to colonies here are valid only locally in each empire and must not be confused with the ranks given to countries in the creating empires stage. Considering the *TMF* val-

ues assigned to colonies in section 3.2, they are sorted in descending order. Let $\beta(i,j)$ be the movement coefficient of the ith colony in jth empire. Then its value can be formulated by the following equation.

$$\beta(i,j) = \beta_{Min}(Decade) + \frac{[\beta_{Max}(Decade) - \beta_{Min}(Decade)] \times Rank(i,j)}{N_{Col}(j)} \quad (12)$$

$$\beta_{Min}(Decade) = \beta_{Min,first} + (\beta_{Min,last} - \beta_{Min,first}) \times \frac{Decade}{N_{Dec}} \quad (13)$$

$$\beta_{Max}(Decade) = \beta_{Max,first} + (\beta_{Max,last} - \beta_{Max,first}) \times \frac{Decade}{N_{Dec}} \quad (14)$$

$Rank\ (i,j)$ is the rank of the ith colony in jth empire. $N_{Col}\ (j)$ is the number of colonies in jth empire. N_{Dec} is also the total number of decades in a single run.

Creating trial vectors

The approach for those colonies which do not participate in the assimilation process is to incorporate them in creating some trial vectors. The goal here is to improve global search capability of the algorithm. Assuming that the process is being carried out for the ith colony in jth empire. Two random colonies in the same empire are chosen such that $c_1 \neq c_2 \neq i$. Then two new countries are formed according to Eq.15 and 16.

$$X_{Cou1} = X_{c1} + Acc(Decade) \times (-1+2r) \times (X_{imp}\ (j) - X_{c1}) \quad (15)$$

$$X_{Cou2} = X_{c2} + Acc(Decade) \times (-1+2r) \times (X_{imp}\ (k) - X_{c2}) \quad (16)$$

$$Acc(Decade) = Acc_{first} + (Acc_{last} - Acc_{first}) \times Decade/N_{Dec} \quad (17)$$

$X_{imp}\ (k)$ is the position vector of an imperialist in a random empire other than jth. Acc is the acceleration coefficient. Low value of Acc ensures better local search, while by increasing its value, better exploration is concluded. Thereafter, the aim is to create two trial vectors from these two countries to compare with the target vector which is the $X_{Col}\ (i,j)$. To accomplish that, the crossover operator must be used. First of all, three random vectors between zero and one with the length of the problem dimension are created.

$$R_1 = [r_{11},\ldots,r_{1n}] \quad 0 < r_{1d} < 1 \quad (18)$$

$$R_2 = [r_{21},\ldots,r_{2n}] \quad 0 < r_{2d} < 1 \quad (19)$$

$$R_3 = [r_{31},\ldots,r_{3n}] \quad 0 < r_{3d} < 1 \quad (20)$$

$$n = N_{Dim} \quad (21)$$

Then trial vectors are created according to Eq .22 and 23.

$$X^d_{trial1} = \begin{cases} X^d_{Col}(i,j) & r_{1d} \geq r_{2d} \\ X^d_{Cou1} & r_{1d} < r_{2d} \end{cases} \quad \forall d = 1,\ldots,N_{dim} \quad (22)$$

$$X^d_{trial2} = \begin{cases} X^d_{Col}(i,j) & r_{2d} \geq r_{3d} \\ X^d_{Cou2} & r_{2d} < r_{3d} \end{cases} \quad \forall d = 1,\ldots,N_{dim} \quad (23)$$

Finally the trial vectors are compared with $X_{Col}\ (i,j)$. The colony will be replaced in the case of being dominated by any of them.

Enhanced revolution

As mentioned in Algorithm. 1, in the revolution process of the original ICA, a certain percent of colonies in each empire are chosen and their positions are exchanged with random points in the problem space. The modification proposed here is in the creation of the new points. According to the dimension of the problem, the revolving colonies will take some part of their vectors from a random imperialist and the other part is also created randomly. The formulation has been described in Eq.24

$$X^d_{Revolved} = \begin{cases} X^d_{imp}(k) \\ X^d_{lbound} + r \times \left(X^d_{hbound} - x^d_{lbound}\right) \end{cases}$$

$$r \le Rev2ImpRate$$

$$else \qquad (24)$$

$$\forall d = 1, \dots, N_{Dim}$$

$X_{Revolved}$ is the position of a colony which is in the process of revolution. X_{lbound} and X_{hbound} are respectively the lower and upper bound vectors of the decision parameters. *Rev2ImpRate* is the Probability that the revolving colony becomes like the imperialist. *r* is also a random number between zero and one.

Fuzzy possession

In the possession stage of the algorithm, imperialists are replaced by colonies if been dominated by any of them. But there are many occasions that some colonies are mutually non-dominated with the imperialist, meaning that neither the colony nor the imperialist can dominate the other. In this case, the algorithm will enter into another stage called the fuzzy possession. This stage enables the colonies whose *TMF* is higher than that of the imperialist to replace it. The procedure is shown in Algorithm 4.

Considering all the discussed approaches and strategies, the new devised algorithm is called the multi-objective enhanced ICA (MOEICA). The overall procedure of the algorithm is portrayed in the flowchart of Fig. 2.

MOEAs ASSESSMENT

The MOEA community has created various specific benchmark functions which have been employed by other MOEA designers. Some MOPs are intended to examine certain areas of MOEAs. Several appear to be relatively easy in the sense of finding the Pareto optima. This situation implies the identification of appropriate benchmark functions to quantitatively evaluate MOEAs (Coello et al., 200). Although good performance of an algorithm in handling benchmark functions does not guaranty its effectiveness on real-world problems, the benchmarks remain the best tools for comparative studies (Holland, 2000). There are several ways to categorize different MOPs based on their characteristics. For example, scalability, modality, type of constraints and dimensionality alongside with the possible PF geometries like convexity, continuity and bias can be taken into account. Explanation and discussion about the aforementioned traits are provided in Refs (Huband et al., 2006؛ Deb, 1999).

Benchmark functions

There are a wide variety of Benchmark functions proposed in the literature. Many of which are unconstrained problems. Among them Fonseca's (1995) and Kursawe's (1991) functions are of historical importance and hence examined here. The well-known group of ZDT test problems which has been provided by Zitzler et al. (2000) is also employed. They are six different functions of which, two have been chosen for the sake of brevity. ZDT3 has a discontinues Pareto front and ZDT6 features bias towards the first objective. The DTLZ suite of benchmark problems,

Algorithm 4. The fuzzy possession

1- Assume that the process is happening in *j*th empire
2- **For** $i \in N_{Col}(j)$
3- **If** *rand*<.5
4- $TFM_{Col}(i,j)=mean[\mu_{Col,f1}(i,j), \mu_{Col,f2}(i,j), \dots, \mu_{Col,fM}(i,j)]$
5- $TMF_{Imp}(j)=mean[\mu_{Imp,f1}(j), \mu_{Imp,f2}(j), \dots, \mu_{Imp,fM}(j)]$
6- **Else**
7- $TFM_{Col}(i,j)=min[\mu_{Col,f1}(i,j), \mu_{Col,f2}(i,j), \dots, \mu_{Col,fM}(i,j)]$
8- $TMF_{Imp}(j)=min[\mu_{Imp,f1}(j), \mu_{Imp,f2}(j), \dots, \mu_{Imp,fM}(j)]$
9- **End If**
10- **If** $TMF_{Col}(i,j) > TMF_{Imp}(j)$
11- Exchange the position vectors of the imperialist and the colony
12- Exchange the objective vectors of the imperialist and the colony
13- **End For**
14- **End For**

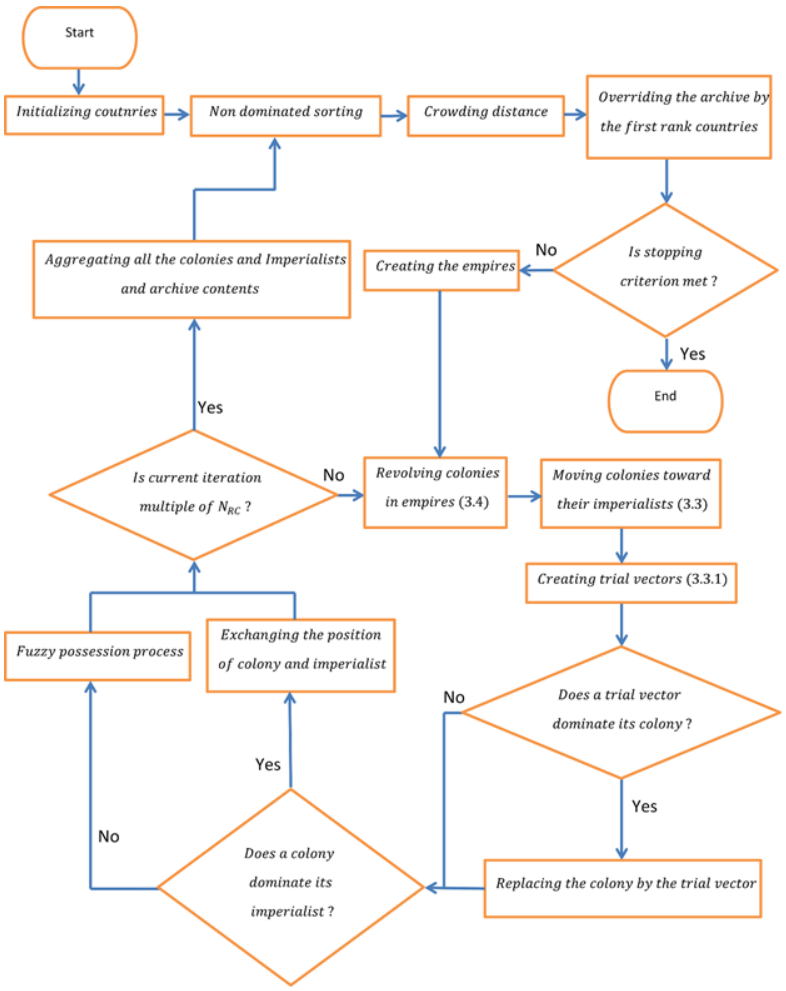

Fig. 2. The flowchart of the proposed MOEICA

created by Deb et al. (2001), is also included. These problems are scalable to any number of objectives. So in this study the dimensionality of three is chosen. WFG Benchmark functions (Huband et al., 2006) are also selected. Similar to the DTLZ family, they are also scalable in terms of number of parameters and objectives. Finally Oka test functions (2004))are also examined. They are very difficult for any MOEA, because of their high modality close to the PF. The specifications of these employed unconstrained benchmark functions are summarized in Table 1. As for the constrained benchmark functions the Osyczka's (1995) and Tanaka's (1995) and Viennet's (1996) functions are incorporated. the specifications of these MOPs are also described in Table 2.

Table 1: The characteristics of unconstrained benchmark functions

Benchmark Function	M	N_{Dim}	Geometry	Modality	Bias	Continuity	Parameter Scalability	Objective Scalability
Fonseca	2	3	Concave	Unimodal	No	Connected	Yes	No
Kursawe	2	3	Mixed	Unimodal	No	Disconnected	Yes	No
ZDT3	2	30	Mixed	Multimodal	No	Disconnected	Yes	No
ZDT6	2	30	Concave	Multimodal	Yes	Connected	Yes	No
DTLZ1	3	7	Linear	Multimodal	No	Connected	Yes	Yes
DTLZ5	3	12	Concave	Unimodal	Yes	Connected	Yes	Yes
DTLZ7	3	22	Mixed	Multimodal	Yes	Disconnected	Yes	Yes
WFG2	3	24	Convex	Unimodal	No	Disconnected	Yes	Yes
WFG6	3	24	Concave	Unimodal	No	Connected	Yes	Yes
WFG8	3	24	Concave	Unimodal	Yes	Connected	Yes	Yes
Oka1	2	2	Convex	Multimodal	No	Connected	No	No
Oka2	2	2	Concave	Multimodal	No	Connected	No	No

Table 2: The characteristics of constrained benchmark functions

Benchmark Function	M	N_{Dim}	Geometry	Modality	Bias	Continuity	Parameter Scalability	Objective Scalability	Objective Scalability
Osyczka2	2	6	Linear	Unimodal	Yes	Connected	No	No	6
Tanaka	2	2	Mixed	Unimodal	No	Disconnected	No	No	1
Viennet4	3	2	Mixed	Unimodal	No	Connected	No	No	3

Performance metrics

The primary purpose of every MOEA is to guide the searching route of the algorithm to the true PF as close as possible, while simultaneously maintain the diversity in the obtained non-dominated solutions or otherwise called the approximate set. Hence in order to evaluate these properties, several assessment metrics have been developed in the literature. The purpose is to quantitatively examine the following criterions: 1) proximity of the approximate set with the true PF (convergence); 2) whether the solutions are uniformly distributed on the obtained approximate set (distribution); and 3) how widespread the solutions are in the objective space (spread). Let P be a set of evenly distributed sample points along the true PF, and S be the obtained approximate set by an algorithm. The following metrics are defined based on the above notations. The inverted generational distance (IGD) developed by Li and Zhang (2009) is used to evaluate the convergence capability of the candidate MOEAs.

$$IGD(P,S) = \frac{\left(\sum_{i=1}^{|P|} d_i^q\right)^{\frac{1}{q}}}{|P|} \quad (25)$$

Where d_i is the distance of $\vec{pi} \in P$ to the closest solution in S. q is taken equal to 2. $|P|$ is the number of points in P. Another convergence metric which is used in this study is the Epsilon indicator which has been introduced by Zitzler et al. (2003). This metric is a measure of the smallest distance one would need to translate every solution in S so that it dominates P.

$$E(S,P) = inf_{\varepsilon \in R^+}\{\forall \vec{p} \in P, \exists \vec{s} \in S : \vec{s} \preceq \vec{p} + \varepsilon\} \quad (26)$$

Where $\vec{s} \preceq \vec{p} + \varepsilon$ means that $\forall j = 1:M, s^j \leq p^j + \varepsilon$. The third metric employed here is the generalized spread (Δ^*) which has been devised by Zhou et al. (2006) and measures the distribution and spread of the solutions found by an algorithm.

$$\Delta^*(S,P) = \frac{\sum_{k=1}^{M} d(e_k,S) + \sum_{i=1}^{|S|}\left|d_i - \bar{d}\right|}{\sum_{k=1}^{M} d(e_k,S) + |S|.\bar{d}} \quad (27)$$

e_k is the extreme solution in P corresponding to the kth objective. d_i is identified as the distance of $\vec{si} \in S$ to its closest solution is S. $\bar{d}$ is the mean of d_i.

COMPARATIVE SIMULATION

For simulation and comparison, the proposed

Table 3 : The number of function evaluations intended for MOPs

Benchmark Function	Number of Intended Function Evaluations
Fonseca	50,000
Kursawe	
DTLZ7	
WFG2	
WFG6	
WFG8	
Osyczka2	
Viennet4	
ZDT3	100,000
DTLZ5	
Tanaka	
Oka1	
ZDT6	200,000
DTLZ1	
Oka2	

Table 4: The corresponding parameters for MOEICA testing

Parameter	$\beta_{min.first}$	$\beta_{min.last}$	$\beta_{max.first}$	$\beta_{max.last}$	Acc_{first}	Acc_{last}
Value	2	.75	4	1.5	1.5	2.5
Parameter	N_{Imp}	N_{RC}	Ass_{Rate}	*Rev2ImpRate*	-	
Value		3	.5	.25		

MOEICA along with the two well-known algorithms, NSGA-II (Deb et al., 2002) and MOPSO (Coello & Lechuga, 2002) are incorporated. 10 independent runs are carried out for every benchmark MOP. The optimization ending condition is to reach a determined number of function evaluations which has been given for every problem in Table 3. The maximum number of solutions presented in the approximate set is limited to 100 for all methods. To deal with constrained problems, the penalty method is intended, so that sum of all violations multiplied in a fixed large coefficient is added to all objectives. The true PFs which are required for evaluation of performance metrics are available for downloading at: http:// www.cs.cinvestav.mx/~emoobook. All codes have been implemented by Matlab 7.12 software and simulated via an Intel core i5 processor and 4GB of RAM. The chosen parameters associated with the MOEICA for testing are listed in Table 4.

Metrics results

After finishing the simulations, performance metrics are computed. The mean and standard deviation values for IGD, Epsilon, Δ* and run-

Table 5: Mean and standard deviation values for IGD metric

Problem	**NSGA-II**	**MOPSO**	**MOEICA**
Fonseca	2.9539e-4 (2.1635e-5)	4.5784e-4 (3.3951e-5)	**2.6402e-4 (8.3133e-6)**
Kursawe	1.7803e-3 (1.4487e-4)	3.7729e-3 (5.6940e-4)	**9.5241e-4 (1.8178e-4)**
ZDT3	4.4691e-3 (2.9240e-4)	4.0239e-4 (2.6037e-5)	**2.0497e-4 (9.7955e-6)**
ZDT6	2.8129e-2 (9.7979e-4)	2.0214e-3 (4.1438e-3)	**3.3207e-4 (2.1324e-3)**
DTLZ1	2.2175e-2 (1.0549e-2)	1.7929e-1 (8.9960e-2)	**5.4183e-4 (3.0849e-5)**
DTLZ5	7.4553e-5 (1.5403e-5)	7.8666e-4 (3.0379e-5)	**7.1656e-5 (6.0666e-6)**
DTLZ7	1.4092e-2 (1.8501e-3)	**9.9775e-4 (4.1044e-5)**	1.1123e-3 (1.0098e-4)
WFG2	1.4118e-3 (1.1793e-4)	2.8719e-3 (3.7672e-4)	**1.3150e-3 (1.1836e-4)**
WFG6	3.3008e-3 (1.8590e-4)	3.6690e-3 (3.1291e-4)	**3.0397e-3 (9.2205e-4)**
WFG8	4.6618e-3 (1.9773e-4)	6.9185e-3 (2.8333e-4)	**4.3912e-3 (1.1783e-4)**
Oka1	5.2369e-3 (2.2111e-3)	**4.6639e-3 (2.8795e-4)**	5.1135e-3 (3.2387e-4)
Oka2	**4.2369e-2 (1.7211e-3)**	4.8301e-2 (1.4894e-3)	6.9486e-2 (1.5452e-3)
Osyczka2	2.8970e-1 (6.6873e-2)	1.2119e-1 (4.8161e-2)	**8.2187e-2 (6.4158e-3)**
Tanaka	**2.4883e-4 (1.1061e-5)**	4.2180e-4 (1.6020e-5)	2.9307e-4 (1.8769e-5)
Viennet4	1.1772e-3 (6.7711e-5)	**8.0853e-4 (4.3018e-5)**	1.0511e-3 (5.7163e-5)

Table 6 : Mean and standard deviation values for Epsilon metric

Problem	**NSGA-II**	**MOPSO**	**MOEICA**
Fonseca	1.4345e-2 (4.3098e-3)	2.3367e-2 (3.7861e-3)	**1.0856e-2 (1.8341e-3)**
Kursawe	1.3351e-1 (8.0642e-2)	2.2905e-1 (6.2925e-2)	**8.8082e-2 (8.1152e-3)**
ZDT3	2.5294e-1 (3.8942e-2)	2.7577e-2 (5.9607e-3)	**1.0570e-2 (1.6687e-3)**
ZDT6	3.0303 (9.8708e-2)	2.1724e-1 (3.9377e-1)	**2.0879e-1 (5.4252e-2)**
DTLZ1	1.6736 (5.9575e-1)	1.3453e1 (4.8873)	**5.8876e-1 (6.8657e-2)**
DTLZ5	2.3090e-2 (1.1751e-2)	4.6098e-2 (5.8369e-3)	**1.7964e-2 (3.2938e-3)**
DTLZ7	2.8539 (2.7440e-1)	**2.2317e-1 (2.7105e-2)**	2.6971e-1 (6.3449e-2)
WFG2	1.4198 (6.7957e-2)	9.1752e-1 (1.3982e-1)	**4.7969e-1 (3.9354e-2)**
WFG6	1.4403 (1.5491e-1)	9.0753e-1 (1.3556e-1)	**8.8267e-1 (1.6138e-1)**
WFG8	1.2819 (1.6258e-1)	1.5991 (2.1562e-1)	**9.8448e-1 (5.6552e-2)**
Oka1	**1.3614e-1 (7.0550e-2)**	1.4828e-1 (9.4570e-3)	1.8643e-1 (3.6596e-2)
Oka2	5.8778e-1 (4.4253e-2)	**5.5550e-1 (4.9787e-3)**	7.0923e-1 (3.3057e-2)
Osyczka2	1.3390e1 (4.6686e-1)	6.9448 (2.4414)	**1.5711 (3.2575e-1)**
Tanaka	1.2065e-2 (2.6842e-3)	2.1681e-2 (4.2059e-3)	**1.1528e-2 (5.0189e-4)**
Viennet4	4.0430e-1 (1.1403e-1)	**3.1102e-1 (6.9732e-2)**	3.2779e-1 (5.5883e-2)

times are given in Tables 5-8 respectively.

First the algorithms are compared according to the IGD metric. For some problems like Fonseca, Kursawe, ZDT3, ZDT6, DTLZ1, DTLZ5, WFG2, WFG6, WFG8 and Osyczka2 the MOEICA obtains better values which makes it the superior performer among the three algorithms in terms of the IGD metric. MOPSO also scores the best performance in three problems and NSGA-II leads in only two problems. Considering the fact that ZDT3, ZDT6, DTZL1, DTLZ7, Oka1 and Oka2 are multimodal problems, it can be concluded that MOEICA and MOPSO generally acquire better IGD values in problems with high modality, while the NSGA-II deals better with unimodal problems in that respect. Taking the Epsilon metric values into account, the results are more conclusive in favor of the MOEICA. Except in the DTLZ7, Oka1, Oka2 and Viennet4, MOEICA remains the best performer in most of the problems. MOEICA outperforms in three MOPs and NSGA-II leads in only one MOP. Analyzing the Δ* metric calculations indicates that again the MOEICA leads overall by maintaining better values in seven benchmarks. This metric is also the only field that NSGA-II can produce better results than MOPSO, showing its slightly better capability to distribute the population evenly on the approximate set. Considering runtimes of the algorithms is also of great importance. An optimization method which produces same results within a less computational time is claimed to be more efficient. In this aspect the MOEICA and MOPSO are almost on par al-

Table 7: Mean and standard deviation values for Δ* metric

Problem	NSGA-II	MOPSO	MOEICA
Fonseca	7.6637e-1 (4.8241e-2)	9.8211e-1 (4.6348e-2)	**5.1579e-1 (3.1410e-2)**
Kursawe	8.1921e-1 (1.6143e-1)	1.3312 (4.9927e-2)	1.0657 (4.9627e-2)
ZDT3	9.7021e-1 (9.4878e-3)	1.0888 (9.0781e-2)	**6.7337e-1 (2.3302e-2)**
ZDT6	9.9971e-1 (8.9304e-5)	1.1524 (2.3558e-1)	**9.8914e-1 (4.0307e-3)**
DTLZ1	9.2682e-1 (4.9958e-2)	8.9885e-1 (2.0942e-1)	**7.1036e-1 (5.6861e-2)**
DTLZ5	7.9569e-1 (7.9103e-2)	1.1099 (9.7220e-2)	**7.4332e-1 (1.0241e-1)**
DTLZ7	9.4490e-1 (3.4960e-2)	**8.7033e-1 (5.5228e-2)**	9.1171e-1 (1.0314e-1)
WFG2	9.3598e-1 (3.1950e-2)	**9.2760e-1 (1.1658e-1)**	1.0476 (1.3839e-1)
WFG6	8.3210e-1 (3.6395e-2)	**7.6312e-1 (3.8021e-2)**	7.8083e-1 (9.2469e-2)
WFG8	8.2254e-1 (3.0113e-2)	7.3483e-1 (4.7068e-2)	**7.2041e-1 (5.3433e-2)**
Oka1	**9.6469e-1 (9.5404e-1)**	1.3176 (6.6842e-2)	9.8286e-1 (7.5678e-2)
Oka2	**1.0610 (1.0554)**	1.8332 (9.5939e-2)	1.7711 (6.5634e-2)
Osyczka2	**8.8798e-1 (3.5563e-2)**	1.2588 (1.0660e-1)	1.1959 (4.9600e-2)
Tanaka	**7.4325e-1 (2.6707e-2)**	1.0796 (4.5909e-2)	8.7108e-1 (4.1305e-2)
Viennet4	9.7538e-1 (7.3219e-2)	1.0727 (7.8083e-2)	**9.6566e-1 (5.6750e-2)**

Table 8: Mean and standard deviation values for Runtime

Problem	NSGA-II	MOPSO	MOEICA
Fonseca	4.1262e2 (2.7211e-1)	7.7624e1 (5.0013e-1)	**7.2240+1 (3.5989e-1)**
Kursawe	4.1301e2 (3.0627e-1)	**6.8876e1 (6.6814e-1)**	7.0861e1 (7.0064e-1)
ZDT3	1.6416e3 (4.9241)	**1.7603e2 (5.9055)**	1.9131e2 (2.4125)
ZDT6	3.2496e3 (8.0442e1)	5.3721e2 (1.6691e2)	**2.6668e2 (8.3531)**
DTLZ1	2.3172e3 (1.1685e2)	2.5022e2 (5.6093)	**2.4323e2 (1.2916e1)**
DTLZ5	1.0914e3 (7.0054e-1)	**1.7612e2 (8.3517)**	2.0821e2 (1.3558)
DTLZ7	5.4776e2 (2.0704)	9.6007e1 (5.6179e-1)	**9.3428e1 (1.6188)**
WFG2	5.4826e2 (1.1919)	1.1419e2 (1.2774)	**1.1056e2 (1.1679)**
WFG6	5.4845e2 (2.1235)	1.4261e2 (2.1511)	**1.1341e2 (4.5048e-1)**
WFG8	5.4647e2 (1.07580	2.1326e2 (4.6218e-1)	**1.2099e2 (4.7648e-1)**
Oka1	1.3695e3 (1.1598e1)	1.0733e2 (8.1153e-1)	**1.0176e2 (1.1537)**
Oka2	2.3290e3 (2.7604e-1)	**1.4983e2 (9.5205e-1)**	1.6962e2 (5.4638e-1)
Osyczka2	4.1907e2 (5.3328e-1)	**2.8827e2 (7.4869e-1)**	3.2199e2 (1.5575)
Tanaka	8.4426e2 (4.2787e-1)	**1.4385e2 (5.5591e-1)**	2.0031e2 (3.0977)
Viennet4	4.1482e2 (3.4373e-1)	**9.2547e1 (1.0944)**	1.1706e2 (3.9471e-1)

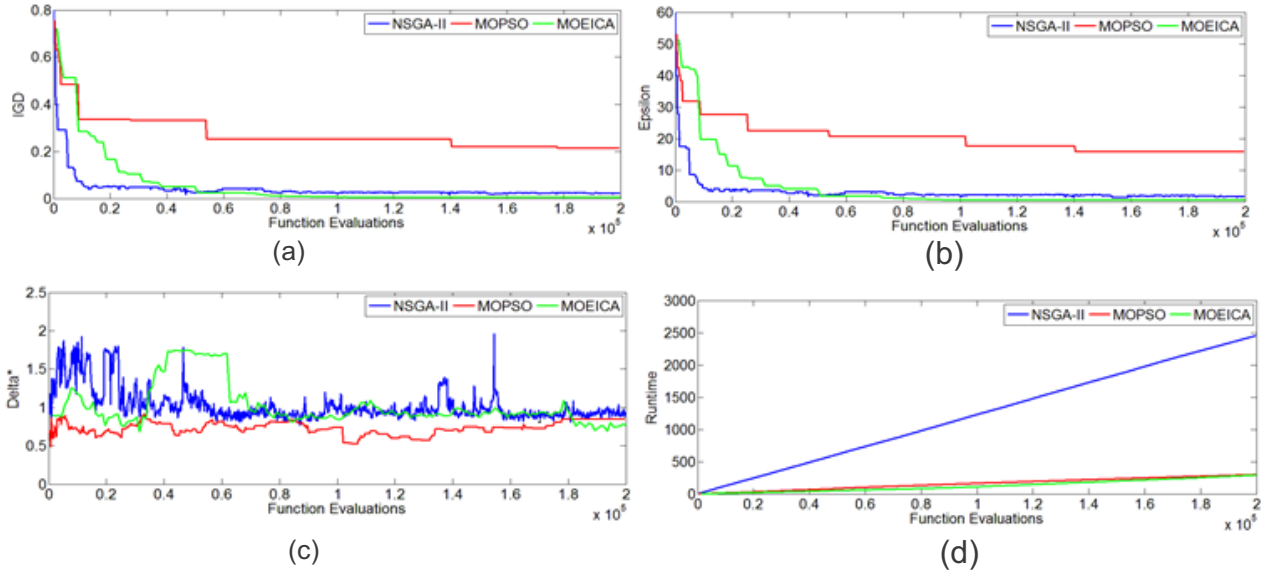

Fig. 5. Variations of different metrics obtained from DTLZ1 problem.
(a) IGD. (b) Epsilon. (c) Δ*. (d) Runtime.

though it seems that the MOEICA leads trivially. NSGA-II has its worst performance in this area where in some problems its computational time is above ten times higher than the other two algorithms. Statistical comparison of the performances in all four categories will be discussed in the next section. It is also beneficial to compare how algorithms guide the population toward the true PF during the optimization. The DTLZ1 problem is selected as an instance. The variations of all metrics for the three algorithms are portrayed in Fig. 5. Fig. 6 also shows obtained solutions in objectives space after 200,000 function evaluations.

Friedman test

In order to determine whether the results of the methods are significantly different from each other, the Friedman statistical test is employed. The principle of the Friedman test is based on a null hypothesis which must be rejected to prove that at least one treatment (method) performs better than at least one of the other treatments (Zimmerman & Zumbo, 1993؛ Villegas, 2011). It is assumed that there are k treatments to be applied on b blocks (problems). The Friedman test is applicable to the observations on the blocks only if they are mutually independent and can be ranked according to each treatment. Considering the aforementioned

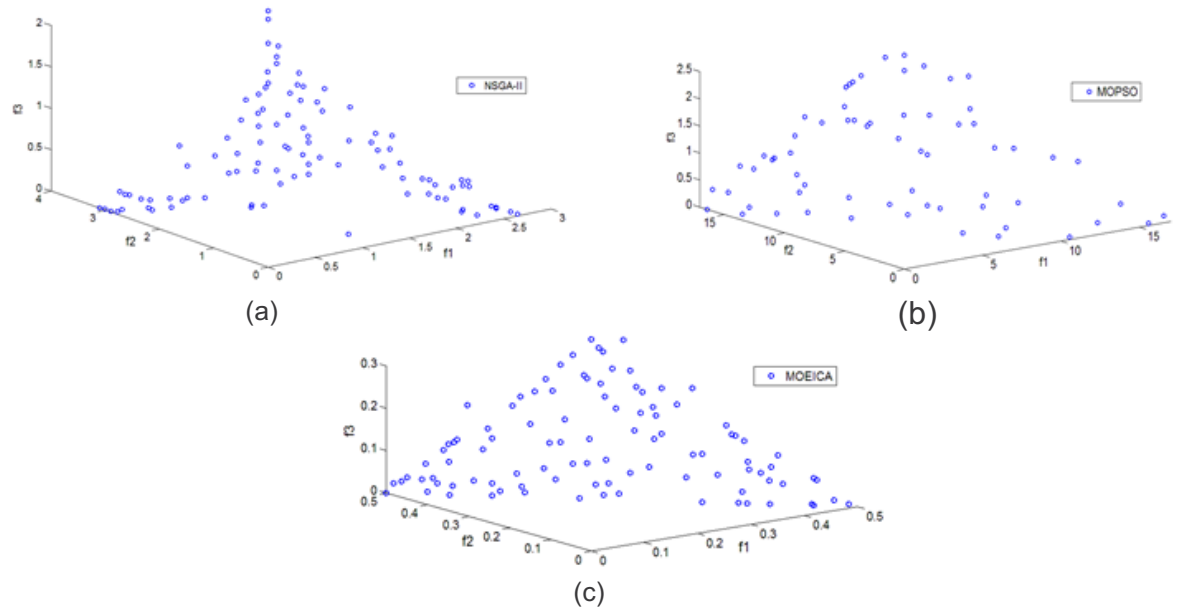

Fig. 6. Approximated sets obtained from DTLZ1 problem after 200,000 function evaluations.
(a) NSGA-II. (b) MOPSO. (c) MOEICA.

assumptions, ranks of the algorithms, for all metrics have been gathered in Tables 9-10.

After the ranking procedure, the following parameters must be calculated:

$$A_2 = \sum_{i=1}^{b}\sum_{j=1}^{k} {R_{ij}}^2 \tag{25}$$

$$B_2 = \frac{1}{b}\sum_{j=1}^{k}\left(\sum_{i=1}^{b} R_{ij}\right)^2 \tag{26}$$

$$T_2 = \frac{(b-1)[B_2 - bk(k+1)^2/4]}{A_2 - B_2} \tag{27}$$

R_{ij} is the rank of jth algorithm on ith problem. T_2 is the statistical test value. If the condition of $T_2 > F_{1-\alpha,k-1,(b-1)(k-1)}$ is satisfied, then the null hypothesis is rejected. α is the level of significance and the function of $F_{x,v1,v2}$ returns the inverse of the F cumulative distribution function (CDF) at the value of x with v_1 and v_2 degrees of freedom. By rejecting the null hypothesis the pairwise comparisons will be possible. Two algorithms are different if $|R_i - R_j|$ surpasses the critical value which is calculated by the following equation.

$$C = T_{1-\alpha/2,(b-1)(k-1)} \cdot \left[\frac{2b(A_2 - B_2)}{(b-1)(k-1)}\right]^2 \tag{28}$$

$T_{x,v}$ also calculates the inverse of CDF for the student's T distribution function. The results of the Friedman tests along with pairwise comparisons are gathered in Table 11. The level of significance for all comparisons have been set to five percent.

What is apparent from the table is that in all four cases the null hypothesis is rejected. Concerning the IGD, the rank differences between the MOEICA with both other algorithms is above the critical value. Then its superior performance in this category is proved. However it is not the case for NSGA-II and MOPSO, concluding that the two methods perform at the same level in terms of IGD metric values. The same conclusions apply for the Epsilon metric, where the MOEICA outperforms the other two by a higher margin than the former case. Again the difference between NSGA-II and MOPSO is negligible. As for the Δ* metric, however still the MOEICA has the lowest overall rank, difference between its rank and that of the NSGA-II is trivial. Then it is not significantly advantageous over NSGA-II regarding Δ* metric performance. Meanwhile the MOPSO has its worst performance here being inferior compared to other two algorithms. Paying attention to runtime ranks provided in the table, it is clear that MOEICA and MOPSO while performing at the same level, have a clear advantage over NSGA-II by a very large margin.

Table 9: Ranks of all algorithms for IGD and Epsilon

Problem	IGD			Epsilon		
	NSGA-II	MOPSO	MOEICA	NSGA-II	MOPSO	MOEICA
Fonseca	2	3	1	2	3	1
Kursawe	2	3	1	2	3	1
ZDT3	3	2	1	2	3	1
ZDT6	3	2	1	3	2	1
DTLZ1	2	3	1	2	3	1
DTLZ5	2	3	1	2	3	1
DTLZ7	3	1	2	3	1	2
WFG2	2	3	1	3	2	1
WFG6	1	3	2	3	2	1
WFG8	2	3	1	2	3	1
Oka1	3	1	2	1	2	3
Oka2	1	2	3	2	1	3
Osyczka2	3	2	1	3	2	1
Tanaka	1	3	2	2	3	1
Viennet4	3	1	2	3	1	2
Sum of Ranks	33	35	22	35	34	21

Table 10 : Ranks of all algorithms for Δ* and runtime

Problem	Δ*			Runtime		
	NSGA-II	**MOPSO**	**MOEICA**	**NSGA-II**	**MOPSO**	**MOEICA**
Fonseca	2	3	1	3	2	1
Kursawe	1	3	2	3	1	2
ZDT3	2	3	1	3	1	2
ZDT6	2	3	1	3	2	1
DTLZ1	3	2	1	3	2	1
DTLZ5	2	3	1	3	1	2
DTLZ7	3	1	2	3	2	1
WFG2	2	1	3	3	2	1
WFG6	3	1	2	3	2	1
WFG8	3	2	1	3	2	1
Oka1	1	3	2	3	2	1
Oka2	1	3	2	3	1	2
Osyczka2	1	3	3	3	1	2
Tanaka	1	3	2	3	1	2
Viennet4	2	3	1	3	1	2
Sum of Ranks	27	37	25	45	23	22

Table 11: Friedman test values and pairwise comparisons

IGD	**Test Values**	**Comparisons**		
	$A_2=210$	$\lvert R_i-R_j\rvert$	MOEICA	MOPSO
	$B_2=186.53$	NSGA-II	**11**	2
	$T_2=3.89$			
	$F=3.34$	MOPSO	**13**	-
	$C=10.27$			
Epsilon	**Test Values**	**Comparisons**		
	$A_2=210$	$\lvert R_i-R_j\rvert$	MOEICA	MOPSO
	$B_2=188.13$	NSGA-II	**14**	1
	$T_2=5.21$			
	$F=3.34$	MOPSO	**13**	-
	$C=9.91$			
Δ*	**Test Values**	**Comparisons**		
	$A_2=215$	$\lvert R_i-R_j\rvert$	MOEICA	MOPSO
	$B_2=189$	NSGA-II	**2**	10
	$T_2=4.85$			
	$F=3.34$	MOPSO	**12**	-
	$C=10.81$			
runtime	**Test Values**	**Comparisons**		
	$A_2=210$	$\lvert R_i-R_j\rvert$	MOEICA	MOPSO
	$B_2=202.53$	NSGA-II	**23**	22
	$T_2=42.25$			
	$F=3.34$	MOPSO	**1**	-
	$C=5.79$			

CONCLUSIONS

In this paper, a MOEA based on the single objective ICA was proposed. In order to deal with the dominance criteria in multi-objective problems, the non-dominated sorting and crowding distance methods were incorporated. An auxiliary comparison approach called fuzzy possession was also presented using the concept of fuzzy membership functions. By doing so it was made possible for all the colonies in each empire to be ranked and compared with the imperialist without a huge computational effort. The mini-

mum and mean calculations in TMF focused on respectively convergence and diversity enhancement. Moreover operating the sorting process at a less frequent manner than NSGA-II paid off by reducing the runtime of the algorithm and making it somewhat between five-to-ten times faster. The satisfying performance of the algorithm in multimodal problems indicates that the recreation of empires at predefined intervals has aided for better exploration. These provisions altogether made it possible for the proposed algorithm to yield an approximated solution set with better performance than similar methods like NSGA-II and MOPSO. To make it evident, the performance of the algorithm was tested on fifteen benchmark MOPs in terms of IGD, Epsilon and Δ^* metrics. The Friedman test study by the five percent level of significance showed that the MOEICA is meaningfully preferable than other methods in convergence metrics including IGD and Epsilon. The diversity of the solutions was much better than MOPSO and slightly better than NSGA-II. Moreover the speed of the algorithm is pretty much the same as the MOPSO and significantly faster than NSGA-II. In our future work, we will focus on the practical applications of the MOEICA. Furthermore, it is also beneficial to test other sorting and archive preserving methods to improve the performance of the algorithm even more.

REFERENCE

Atashpaz-Gargari, E., & Lucas, C. (2007, September). Imperialist competitive algorithm: an algorithm for optimization inspired by imperialistic competition. In Evolutionary computation, 2007. CEC 2007. IEEE Congress on (pp. 4661-4667). IEEE.

Chandrasekaran, S., Ponnambalam, S.G., Suresh, R. K., & Vijayakumar, N. (2007, September). Multi-objective particle swarm optimization algorithm for scheduling in flowshops to minimize makespan, total flowtime and completion time variance. In Evolutionary Computation, 2007. CEC 2007. IEEE Congress on (pp. 4012-4018). IEEE.

Coello, C. A. C., Lamont, G. B., & Van Veldhuizen, D. A. (2007). Evolutionary algorithms for solving multi-objective problems (Vol. 5). New York: Springer

Coello, C.C., Pulido, G.T., & Lechuga, M.S. (2002). An extension of particle swarm optimization that can handle multiple objectives. In Workshop on Multiple Objective Metaheuristics (pp. 1-5).

Coello Coello, C. A., Lamont, G. B., & Van Veldhuizen, D.A. (2007). Evolutionary algorithms for solving multi-objective problems

Durillo, J.J., Nebro, A.J., & Alba, E. (2010, July). The jmetal framework for multi-objective optimization: Design and architecture. In Evolutionary Computation (CEC), 2010 IEEE Congress on (pp. 1-8). IEEE

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. A. M. T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE transactions on evolutionary computation, 6*(2), 182-197

Deb, K. (1999). Multi-objective genetic algorithms: Problem difficulties and construction of test problems. *Evolutionary computation, 7*(3), 205-230.

Deb, K., Thiele, L., Laumanns, M., & Zitzler, E. (2001). Scalable Test Problems for Evolutionary Multi-Objective Optimization. Zurich. Switzerland, Tech. Rep. 112.

Enayatifar, R., Yousefi, M., Abdullah, A. H., & Darus, A. N. (2013). MOICA: A novel multi-objective approach based on imperialist competitive algorithm. *Applied Mathematics and Computation, 219*(17), 8829-8841.

Fattahi, M., Mahootchi, M., Mosadegh, H., & Fallahi, F. (2014). A new approach for maintenance scheduling of generating units in electrical power systems based on their operational hours. *Computers & Operations Research, 50*, 61-79.

Fan, S.K.S., Chang, J.M., & Chuang, Y.C. (2015). A new multi-objective particle swarm optimizer using empirical movement and diversified search strategies. *Engineering Optimization, 47*(6), 750-770.

Fonseca, C. M., & Fleming, P. J. (1995). Multiobjective genetic algorithms made easy: selection sharing and mating restriction

Ghiasi, H., Pasini, D., & Lessard, L. (2011). A non-dominated sorting hybrid algorithm for multi-objective optimization of engineering problems. *Engineering Optimization, 43*(1), 39-59.

Holland, J. H. (2000). Building blocks, cohort genetic algorithms, and hyperplane-defined

functions. *Evolutionary computation, 8*(4), 373-391.

Huband, S., Hingston, P., Barone, L., & While, L. (2006). A review of multiobjective test problems and a scalable test problem toolkit. *IEEE Transactions on Evolutionary Computation, 10*(5), 477-506.

Idoumghar, L., Chérin, N., Siarry, P., Roche, R., & Miraoui, A. (2013). Hybrid ICA–PSO algorithm for continuous optimization. *Applied Mathematics and Computation, 219*(24), 11149-11170.

Kursawe, F. (1990, October). A variant of evolution strategies for vector optimization. In International Conference on Parallel Problem Solving from Nature (pp. 193-197). Springer Berlin Heidelberg.

Khabbazi, A., Atashpaz-Gargari, E., & Lucas, C. (2009). Imperialist competitive algorithm for minimum bit error rate beamforming. *International Journal of Bio-Inspired Computation, 1*(1-2), 125-133.

Lin, Q., Zhu, Q., Huang, P., Chen, J., Ming, Z., & Yu, J. (2015). A novel hybrid multi-objective immune algorithm with adaptive differential evolution. *Computers & Operations Research, 62*, 95-111.

Li, H., & Zhang, Q. (2009). Multiobjective optimization problems with complicated Pareto sets, MOEA/D and NSGA-II. *IEEE Transactions on Evolutionary Computation, 13*(2), 284-302.

Okabe, T., Jin, Y., Olhofer, M., & Sendhoff, B. (2004). On test functions for evolutionary multi-objective optimization. In Parallel Problem Solving from Nature-PPSN VIII (pp. 792-802). Springer Berlin/Heidelberg.

Osyczka, A., & Kundu, S. (1995). A new method to solve generalized multicriteria optimization problems using the simple genetic algorithm. *Structural and Multidisciplinary Optimization, 10*(2), 94-99.

Shokrollahpour, E., Zandieh, M., & Dorri, B. (2011). A novel imperialist competitive algorithm for bi-criteria scheduling of the assembly flowshop problem. *International Journal of Production Research, 49*(11), 3087-3103.

Sivasubramani, S., & Swarup, K. S. (2009, December). Multiagent based particle swarm optimization approach to economic dispatch with security constraints. In Power Systems, 2009. ICPS'09. International Conference on (pp. 1-6). IEEE.

Srinivas, N., & Deb, K. (1994). Muiltiobjective optimization using nondominated sorting in genetic algorithms. *Evolutionary computation, 2*(3), 221-248.

Tanaka, M., Watanabe, H., Furukawa, Y., & Tanino, T. (1995, October). GA-based decision support system for multicriteria optimization. In Systems, Man and Cybernetics, 1995. Intelligent Systems for the 21st Century., IEEE International Conference on (Vol. 2, pp. 1556-1561). IEEE

Vlennet, R., Fonteix, C., & Marc, I. (1996). Multicriteria optimization using a genetic algorithm for determining a Pareto set. *International Journal of Systems Science, 27*(2), 255-260.

Villegas, J. G. (2011). Using Nonparametric Test to Compare The Performance of Metaheuristics. EU/ME-The metaheuristics community, internet.

Xiang, Y., Zhou, Y., & Liu, H. (2015). An elitism based multi-objective artificial bee colony algorithm. *European Journal of Operational Research, 245*(1), 168-193.

Zitzler, E., Deb, K., & Thiele, L. (2000). Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary computation, 8*(2), 173-195.

Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C. M., & Da Fonseca, V. G. (2003). Performance assessment of multiobjective optimizers: An analysis and review. *IEEE Transactions on evolutionary computation, 7*(2), 117-132.

Zhou, A., Jin, Y., Zhang, Q., Sendhoff, B., & Tsang, E. (2006, July). Combining model-based and genetics-based offspring generation for multi-objective optimization using a convergence criterion. In Evolutionary Computation, 2006. CEC 2006. IEEE Congress on (pp. 892-899). IEEE.

Zimmerman, D. W., & Zumbo, B. D. (1993). Relative power of the Wilcoxon test, the Friedman test, and repeated-measures ANOVA on ranks. *The Journal of Experimental Education, 62*(1), 75-86.