Self Adaptive Cluster Based and Weed Inspired Differential Evolution Algorithm For Real World Optimization

Udit Halder[1], Swagatam Das[1], Dipankar Maity[1], Ajith Abraham[2, 3] and Preetam Dasgupta[1]

[1]Department of Electronics and Telecommunication Engineering, Jadavpur University, Kolkata 700 032, India
[2]Faculty of Computer Science and Electrical Engineering, VSB – Technical University of Ostrava, Czech Republic
[3]Machine Intelligence Research Labs (MIR Labs), Seattle, WA, USA
ajith.abraham@ieee.org

Abstract— **In this paper we propose a Self Adaptive Cluster based and Weed Inspired Differential Evolution algorithm (SACWIDE), the total population is divided into several clusters based on the positions of the individuals and the cluster number is dynamically changed by the suitable learning strategy during evolution. Here we incorporate a modified version of the Invasive Weed Optimization (IWO) algorithm as a local search technique. The algorithm strategically determines whether a particular cluster will perform Differential Evolution (DE) or the IWO algorithm (modified). The number of clusters in a particular iteration is set by the algorithm itself self-adaptively. The performance of SACWIDE is reported on the set of 22 benchmark problems of CEC-2011.**

Keywords - Differential Evolution; Evolutionary Algorithm; real world optimization; weed colony optimization; self-adaptive optimization algorithm.

I. Introduction

Scientist and engineers from all branches have to deal with the global optimization problem where the main target is to find a set of model parameters or state-variables, which will provide a globally maximum or minimum value for a specified objective or cost function. The Differential Evolution (DE) Algorithm proposed by Storn and Price [1] is a simple but very effective algorithm for global optimization problem. In many applications such as Pattern Recognition [1], Communication [4], mechanical engineering [5] the effectiveness and efficiency has been successfully demonstrated. The Invasive Weed Optimization (IWO) algorithm [6], is also a very effective algorithm which has found successful in some application like design of E-shaped MIMO Antenna [7], design of compact U-array MIMO antenna [8], encoding sequences for DNA computing [9], and design of aperiodic thinned array antennas [10].

In our proposed algorithm, we have used a modified, cluster based, self-adaptive Differential Evolution (DE) algorithm, with a weed inspired technique. In this algorithm, the total population is divided into some clusters based on their spatial positions and a modified version of DE is used. In each generation, the clusters do not share information among them. The cluster modification technique is totally self adaptive, i.e. the cluster number and creation of new cluster is determined by the learning strategy of the algorithm itself. We have used cluster based technique so that the total population does not get stuck to a local optimum. When a cluster tends to converge to a point or any local optimum, a weed inspired technique is adapted on that cluster. The weed inspired technique helps us in local searching in the vicinity of each particle. Thus, there is a chance to find a better position for each particle by its weeds. Thus the population is improved by the self-adaptive behavior of the algorithm itself and the simultaneous use of DE and weed inspired technique.

II. Differential Evolution Algorithm

Differential Evolution (DE) is a very simple but a very powerful algorithm for optimization problem. Let $S \in R^n$ be the search space of the problem under consideration. DE algorithm starts with an initial population of *NP, n* dimensional solution particles. These particles (solution vectors) are initially covering the search space (*S*) as much as possible by randomly initializing them through the search space. The particles are of the form $\vec{X}_i = (x_{i1}, x_{i2}, x_{i3}, ..., x_{in}) \in S$, where *i=1,2,….,NP* and are upgraded from one generation to next generation, where $x_{i1}, x_{i2}, ..., x_{in}$ are in between their respective upper and lower bounds x_j^{upper}, x_j^{lower} respectively. The population undergoes through Crossover, Mutation at each generation t and produces a new solution vector $U_{i,t}$ for each vector $\vec{X}_{i,t}$.

A. *Mutation:*

After initialization for each solution vectors $\vec{X}_{i,t}$, a new vector $\vec{Y}_{i,t}$ is generated at each generation *t*. The vector $\vec{Y}_{i,t}$ can be generated by any one of the following five methods:

"DE/rand/1": $\vec{Y}_{i,t} = \vec{X}_{r_1^i,t} + F.(\vec{X}_{r_2^i,t} - \vec{X}_{r_3^i,t})$ (1)

"DE/best/1": $\vec{Y}_{i,t} = \vec{X}_{best,t} + F.(\vec{X}_{r_1^i,t} - \vec{X}_{r_2^i,t})$ (2)

"DE/target-to-best/1"

$$\vec{Y}_{i,t} = \vec{X}_{i,t} + F.(\vec{X}_{best,t} - \vec{X}_{i,t}) + F.(\vec{X}_{r_1^i,t} - \vec{X}_{r_2^i,t}) \quad (3)$$

"DE/best/2":

$$\vec{Y}_{i,t} = \vec{X}_{best,t} + F.(\vec{X}_{r_1^i,t} - \vec{X}_{r_2^i,t}) + F.(\vec{X}_{r_3^i,t} - \vec{X}_{r_4^i,t}) \quad (4)$$

"DE/rand/2":

$$\vec{Y}_{i,t} = \vec{X}_{r_1^i,t} + F.(\vec{X}_{r_2^i,t} - \vec{X}_{r_3^i,t}) + F.(\vec{X}_{r_4^i,t} - \vec{X}_{r_5^i,t}) \quad (5)$$

Where The indices $r_1^i, r_2^i, r_3^i, r_4^i$ and r_5^i are mutually exclusive integers randomly chosen from the range [1, *NP*], and all are different from the base index *i*. F is a scaling factor for the differential vectors and $\vec{X}_{best,t}$ is the vector with best fitness in the generation *t*. The general convention used for naming the various mutation strategies is DE/a/b/c, where DE stands for Differential Evolution, a represents a string denoting the vector to be perturbed and b is the number of difference vectors considered for perturbation of x. c is the type of crossover being used.

B. *Cross-over:*

After the phase of mutation the crossover phase comes, which plays a major role to enhance the diversity of the population. In this phase the generated vector $\vec{Y}_{i,t}$ exchanges its component with its parent vector $\vec{X}_{i,t}$ to generate a new vector $\vec{U}_{i,t} = (u_{1i,t}, u_{2i,t}, u_{3i,t}, ..., u_{ni,t})$ Where $u_{j,i,t}$ is found by the following procedure:

$$u_{j,i,t} = \begin{cases} Y_{j,i,t}\ , & if(rand_{i,j}[0,1) < Cr \quad or \quad j = j_{rand} \\ X_{j,i,t}\ , & otherwise \end{cases}$$

Where $rand_{i,j}[0,1)$ is a uniformly distributed random number which is called anew for each j^{th} component of the i^{th} parameter vector. $j_{rand} \in [1,2,....,n]$ is a randomly chosen index, which ensures that $\vec{U}_{i,t}$ gets at least one component from $Y_{i,t}$, and *Cr* is a user defined constant in the range [0,1).

C. *Selection Operation:*

In the selection phase, the algorithm determines that which one between the target vector and generated vector will be kept for next generation and which one should be deleted from the population so that the size of the population (no of the solution vectors) remains unchanged down the generation. The selection procedure is done by the following way:

$$\vec{X}_{i,t+1} = \begin{cases} \vec{U}_{i,t} & if\ f(\vec{U}_{i,t}) < f(\vec{X}_{i,t}) \\ \vec{X}_{i,t} & if\ f(\vec{U}_{i,t}) > f(\vec{X}_{i,t}) \end{cases}$$

for minimization problem. Thus after every generation we find either a new solution which has better fitness (here for minimization problem) or the previous vector is kept. So after each generation the population gets better or remains unchanged but never deteriorates.

III. The Invasive Weed Optimization (IWO)

IWO is a population-based algorithm that replicates the colonizing behavior of weeds. The basic characteristic of a weed is that it grows its population entirely or predominantly in a geographically specified area, which can be substantially large or small. Initially a certain number of weeds are randomly spread over the entire search space. These weeds will eventually grow up and execute the following four steps as the algorithm proceeds. There are four steps of the algorithm as described below:

1) Initialization: A certain number m of weeds are randomly spread over the entire *D*-dimensional search space. This initial population of each generation will be termed as $X = \{\vec{X}_1, \vec{X}_{2,.........,}\vec{X}_m\}$.

2) Reproduction: Each member of the population ***X*** is allowed to produce seeds within a specified region centered at its own position. The number of seeds produced by $\vec{X}_i,\ i \in \{1,2.....,m\}$ depends on its relative fitness in the

population with respect to the best and worst fitness. The number of seeds produced by any weed varies linearly from $seed_{\min}$ to $change_{\min}$ with $seed_{\min}$ for the worst member and $seed_{\max}$ for the best member in the population.

3) Spatial distribution: The generated seeds are being randomly distributed over the d-dimensional search space by normally distributed random numbers with zero mean and variance σ^2. However, the standard deviation σ is made to decrease over the generations so that the algorithm gradually moves from exploration to exploitation with increasing generations. If $\sigma_{\max}$ and $\sigma_{\min}$ are the maximum and minimum standard deviation, then the standard deviation in a particular generation (or iteration) is given by,

$$\sigma_t = \sigma_{\min} + \left(\frac{t_{\max} - t}{t_{\max}} \right)^{n\_m\_i} .(\sigma_{\max} - \sigma_{\min}), \quad (6)$$

where *n_m_i* represents the non-linear modulation index, *t* is the current iteration number and $t_{\max}$ is the maximum number of iterations allowed. This step ensures that the probability of dropping a seed in a distant area decreases nonlinearly with iterations, which results in grouping fitter plants and elimination of inappropriate plants.

4) Competitive Exclusion: There is a need of some kind of competition between plants to limit the maximum number of plants in a population. Initially, the plants in a colony will reproduce fast and all the produced weeds will be included in the colony, until the number of plants reaches a maximum value of $pop_{\max}$. However, it is expected that by this time the fitter plants have reproduced more than undesirable plants. From then on, only the fittest plants, among the existing ones and the reproduced ones; are taken in the colony and the steps 1 to 4 are repeated until the maximum number of iterations (or function evaluations) have been reached. So, in every generation the population size must be less than or equal to $pop_{\max}$. This method is known as competitive exclusion and is a selection procedure of IWO.

IV. Proposed Algorithm: SACWIDE

In our proposed algorithm we have used a strategy-based clustered DE algorithm, which is self-adaptive with a weed inspired strategy. We initialize the population randomly in the search region and the population is divided into several clusters using the K-means clustering algorithm. If clustering is not done then for multimodal problem there is a high chance to get trapped in a local minimum and the algorithm will not perform better afterwards. That's why the total population is divided into some clusters based on their spatial distribution. The radius of a particular cluster is defined as the mean distance (Euclidean) from the center of the cluster. If the particles of the clusters are $\vec{x}_1, \vec{x}_2 \ldots.., \vec{x}_n$ then the center($\vec{C}$) is determined as

$$\vec{C} = \frac{\sum_{i=1}^{n} \vec{x}_i}{n} \qquad (7)$$

And the radius, $$R = \frac{\sum_{i=1}^{n} \sqrt{\sum_{j=1}^{d} (\vec{x}_{i,j} - \vec{c}_j)}}{n} \qquad (8)$$

After clustering is done, in each cluster modified DE is used separately. In our algorithm we used both the "DE/rand/1" and "DE/best/1" schemes depending on the FE. Between the two schemes "DE/rand/1" usually shows good diversity but slower convergence rate whereas "DE/best/1" shows less diversity but satisfactory convergence rate. In our algorithm, for first 10% of total Function Evaluations the "DE/rand/1" is used and for the remaining FEs "DE/best/1" is used. DE is used for a cluster until its radius becomes very small. If radius of a particular cluster becomes lesser than a pre-assigned value (R_{conv}), then to gain diversity we apply a weed inspired algorithm for that cluster. The value of R_{conv} is calculated as

$$R_{conv} = dist(\vec{X}_{upper,} \vec{X}_{lower}) \cdot 10^{-3} \qquad (9)$$

Where $dist(\vec{a}, \vec{b})$ denotes the Euclidean distance between the vectors $\vec{a} \,\&\, \vec{b}$. $\vec{X}_{upper} \,\&\, \vec{X}_{lower}$ are the upper bound and lower bound vectors in the search region. This expression is used because for all the problems we cannot use a fixed value of R_{conv} . Basically R_{conv} is an indication parameter, which indicates the convergence of a single cluster. It seems obvious that R_{conv} should depend on spread of the upper and lower bounds. Thus instead of assigning a fixed value, we adopted the previous expression for R_{conv} . When the radius of a cluster becomes smaller than R_{conv} a weed inspired algorithm is used instead of DE. We will discuss about the weed inspired algorithm in the next section.

After a specific number of iterations i.e. generations, the performance of the algorithm is checked. If the performance is satisfactory then cluster numbers are decreased and then again the total population is clustered in new cluster numbers. This is done to reduce the wastage of FEs as the algorithm performed well in those generations. But if the algorithm performs badly, then one new cluster is generated in the search region randomly, without deleting the previous individuals.

The idea of creating clusters from the population comes from the phenomenon of trapping in a local minimum on a multimodal search region, when we apply "DE/best/1" on a single population. For global optimization, it is always preferable that we use a speedy algorithm, which can readily find the optimum. The algorithm, which will be most efficient should have a good convergence rate. In that context it seems that applying DE or some variant of DE, we can find the optimum quickly. But, unfortunately, it has been seen that, those algorithms get trapped in a local minimum and don't improve further. To overcome this fact, we used the cluster based optimization idea. The particles of a population are divided into some clusters and the clusters individually try to find minima. There should not be any information exchange between any clusters and they will work individually. However, in our algorithm, some information is exchanged periodically by redistributing the total population. It is expected that each cluster will find one local minimum, which may or may not be the global minimum. But the problem in this, during optimization, two clusters may be highly overlapped (regarding to spatial distribution) and it may happen that two different clusters are searching for the same optimum. Special care should be taken to avoid this problem, otherwise FEs will be wasted. Again, if we don't know the total number of local minima, we will have to blindly guess the cluster numbers, as here one cluster discovers one minimum. Again if a cluster gets stuck to a local optimum, it is necessary to delete that cluster, saving the optimum value discovered by it, and to reinitialize it to a newer place such that it can again search for another minimum. To deal with the above problems, we introduced a cluster based optimization technique in which the total cluster number at any generation is determined self-adaptively by the algorithm and not by the user. However, the user has to provide the initial cluster numbers (k), which we have taken 5 in this paper. In this algorithm after certain iterations (*time span, TS*) or generations the performance of the clusters is evaluated in terms of percentage change in the value of global best value found so far. Then the individuals of the different clusters are exchanged via another call to K-means such that spatially nearer particles are taken into one cluster. Thus the idea of no exchange of information among the clusters is not followed thoroughly in our algorithm. There will be certain exchange of information periodically during optimization by redistributing the particles. If the performance in previous duration under examination is satisfactory, then some clusters are merged or in other words the total cluster number is decreased. This is done because, as the algorithm is performing well, to reduce the FE cost, cluster number is reduced. But the total number of individuals does not change. If the performance is not satisfactory, then one new cluster is created in the search region randomly, as bad performance indicates that there is a high chance that the global minimum stays in some other region, which is currently not covered by the clusters. Thus, the cluster number is increased then, as if the current situation of the process is demanding it. When we introduce a new cluster, new particles are generated randomly keeping the previous population intact. While redistributing, the total population do not change, just the cluster number and number of individuals in a cluster is changed. We set an upper limit on both the cluster numbers ($Clust_{\max}$) and the total population in a cluster ($pop_{\max}$). In this paper we set, $Clust_{\max} = 2k$, i.e. 10 and $pop_{\max}$ =100. There is also a lower limit of cluster number ($Clust_{\min}$), which is set to 2. If the total population in a cluster exceeds maximum value ($pop_{\max}$), then best $pop_{\max}$ particles are selected and the other particles are deleted. If cluster number is maximum, and the performance evaluation test still indicates a negative result, neither the cluster number is further increased nor is any more clustering done. This belongs to the worst condition in the optimization process. Similar work is done if cluster number is minimum and still tends to reduce its value; we restrict it to do so. A brief description of our algorithm follows:

A. *Initialization:*

A certain number of individuals (*NP*) are spread randomly in the search region. Then they are clustered into *k* clusters using the K-means clustering algorithm.

B. *Cluster Improvements:*

As it is stated above that DE performs well with respect to other Evolutionary Algorithms (EAs), we primarily use DE as our main optimization algorithm. It is also sated that, "DE/best/1" has better convergence rate, but after experiments we come to a conclusion that, for 1st 10% of total FE ($FE_{\max}$) if we use "DE/rand/1" with a probability 0.5 (i.e. rand(0,1)>.5), it performs better than using only "DE/best/1". For both the schemes, we used a dynamic value of the scaling factor (*F*). Instead of using a fixed value of *F*, we have used the following formula for deriving *F*.

$$F = (rand+1)*0.5 \tag{10}$$

Here *rand* denotes a random number between 0 and 1. It is seen from the above expression that *F* varies from 0.5 and 1. DE will be applied on a particular cluster until the radius (7) of that cluster becomes lower than a predefined value (R_{conv}). If so, then it can be said that the cluster is highly converged around a point, which may or may not be a minimum even. In this case, we apply the weed inspired technique on that cluster. We have used a modified version of IWO. Some of the parameters are changed and the selection technique is improved. The standard deviation (σ) is defined as

$$\sigma_t = \sigma_{\min} + \left(\frac{FE_{\max} - FE}{FE_{\max}}\right)^2 .(\sigma_{\max} - \sigma_{\min}) \tag{11}$$

$\sigma_{\max}$ is taken as $dist(\vec{X}_{upper,}\vec{X}_{lower}).10^{-1}$, $\sigma_{\min}$ is taken as $dist(\vec{X}_{upper,}\vec{X}_{lower})\cdot 10^{-8}$ .The definitions of *dist,* $\vec{X}_{upper}$ and $\vec{X}_{lower}$ are similar as stated in preceding sections. For the selection part, without selecting all the weeds until population cross $pop_{\max}$, we applied a modified selection criteria. A weed is selected to next generation if it is better than its parent or the mean of its fitness and its parents' fitness is better than the average fitness of best and worst plant in the population. In a cluster, whether it is performing DE or weed inspired algorithm, whenever the population number exceeds $pop_{\max}$, the best $pop_{\max}$ particles are survived and the others are deleted. If a cluster performs DE, population of that cluster will not increase, but while redistributing, it may happen that the population exceeds $pop_{\max}$. This will happen when the algorithm is performing well and the total number of clusters is needed to be reduced. Then, the total population (combining the population from different clusters) is divided into less number of clusters, thus the new clusters will have more number of individuals than that of previous iteration. But, the reduction of cluster numbers is done in the context that the algorithm is performing well so less effort may be given on that time of optimization or in other words we try to save FE. But, if the poorer particles are not deleted, the FE might be wasted.

C. *Performance Evaluation & Redistribution:*

After a *time span* (*TS*) or generations the performance of the algorithm is evaluated and accordingly the cluster numbers are updated. In each iteration, the change in global best value is calculated. The total number of changes over a time span is also calculated. Primarily the importance is given to the number of changes, rather than average change over that period. So, we are giving stress on consistency of change, not on quality of change. But quality affects when the change is very low, but we get change in almost every time. In that case, our desire will not be fulfilled. So, it is checked that if the average change is lower than a value, say, $change_{\min}$, then instead of giving positive result, it will indicate a negative state. After experiments, the optimum value of $seed_{\max}$ is found, and it is taken to be, $seed_{\max}$ =1e-3.The detail of this algorithm is given on the pseudo code.

After evaluating the performance of the algorithm in a *time span* (*TS*), the total population is redistributed. The performance evaluation test gives us the change in cluster number, which is to be made. If the change is positive, then the cluster number is reduced and the total population is redistributed in reduced number of clusters. If the change is negative, then a new cluster is introduced in the search space. The individuals of this new cluster are generated randomly in the entire search region. As stated earlier, cluster number is not updated every time. There are two limits, namely $Clust_{\max}$ and $Clust_{\min}$. When cluster number is equal to $Clust_{\max}$ and still tends to increase, then the increment is restricted. Similarly when cluster number is equal to $Clust_{\min}$ and trying to decrease, we restrict it to decrease further.

V. Pseudo Code

```
/* Main Algorithm*/
Step 1: %Initialization
Initialize a population of NP individuals in the search region randomly
Divide them into k clusters using K-means
Clusterno ← k , iteration t ← 1
Define $Clust_{\max}$, $Clust_{\min}$ and TS
Set $R_{conv}$ according to (9)

Step 2: %Cluster Improvements

While Termination Criterion is not satisfied
    For i = 1 to clusterno
        Determine Radius $R_i$ according to (8)
        If $R > R_{conv}$
            Improve by DE algorithm
        Else
            Improve by Weed Inspired algorithm
        End If
    End For

Step3: %Performance Evaluation

    If mod(t,TS) = 0
        increase = Determine_increase(changeno,totchange)
        changeno ← 0 & totchange ← 0
    Else
        If $Globalbest_t < Globalbest_{t-1}$
            Calculate percentage change in $Globalbest_{t-1}$
            changeno ← changeno+1
            totchange ← totchange+percentage change
        End If
    End If

Step 4: % Update Cluster number

    If mod(t,TS) = 0
        Create (clusterno - increase) clusters using K-means
        clusterno ← clusterno - increase
    End If
    t ← t+1

End While

/* Function For Determination change in cluster numbers*/
Function Determine_increase(changeno,totchange)

avg_change ← totchange/changeno

If changeno ≥ TS*0.6
    increase ← 2
Else If changeno ≥ TS*0.3
    increase ← 1
Else
    increase ← −1
End If

If changeno ≥ TS*0.3 & avg_change ≤ 1e-03
    increase ← −1
End If

temp ← clusterno - increase

If temp> $Clust_{\max}$ Or temp< $Clust_{\min}$
    increase ← 0      // i.e. no change
End If
```

VI. Solving Real World Optimization Problems

In this paper, 22 benchmark problems for the CEC-2011 competition and special session on "Testing Evolutionary Algorithms on Real World Optimization Problems" are solved.

A. *Test Problems & Experimental Conditions:*

Among the 22 problems, the Dynamic Economic Dispatch (DED) problem has 2 instances; the Static Economic Load Dispatch (ELD) problem has 5 instances and the Hydrothermal Scheduling problem has 3 instances of running. We have performed 25 independent runs on each problem. The maximum function evaluations for each run are 150000. It is asked to report the mean, best and worst objective function values after 50000,100000 and 150000 FEs. The algorithm is tested on the benchmark problems until the FE reaches $FE_{\max}$.

B. *Parameter Settings:*

In this algorithm, parameters for DE are the population size (*NP*), a scaling factor (*F*), a crossover rate (*Cr*). For the weed inspired part, the parameters are maximum number of seeds of the cluster ($seed_{\max}$), minimum seed number ($seed_{\min}$), the standard deviation (σ). The parameters to be adjusted due to self adaptive cluster portion are, initial cluster number (*k*), convergence radius of a cluster (R_{conv}), maximum cluster number ($Clust_{\max}$), minimum cluster number ($Clust_{\min}$), maximum size of a single cluster ($pop_{\max}$) and the time span (*TS*) after which the algorithm self adaptively determines the cluster numbers in the next time span. The actual values of the parameters used are, $NP = 200$, $Cr = 0.9$, $seed_{\max} = 5$, $seed_{\min} = 0$, $k = 5$, $Clust_{\max} = 2k = 10$, $Clust_{\min} = 2$, $pop_{\max} = 100$, $TS = 10$. For the scaling factor (*F*), it is determined according to (10), which is a random number between 0.5 and 1. Experiments show that setting F to a specific value, it degrades the performance of the algorithm when it is applied on various problems. So, randomness is introduced in the value of F. R_{conv} can be calculated from (9). The standard deviation (σ), which determines how far the seeds will be generated around an individual is calculated from (11).

Table 1. Experimental results

FEs		F-1	F-2	F-3	F-4	F-5
5e+04	Best	0.00e+0	-2.64e+1	2.87e-4	1.38e+1	-3.55e+1
	Worst	1.79e+1	-1.15e+1	2.87e-4	1.43e+1	-2.65e+1
	Mean	3.54e+0	-1.83e+1	2.87e-4	1.41e+1	-3.20e+1
	Std	5.93e+0	3.78e+0	0.00e+0	2.87e-1	3.26e+0
1e+05	Best	0.00e+0	-2.84e+1	2.87e-4	1.38e+1	-3.68e+1
	Worst	1.45e+1	-1.91e+1	2.87e-4	1.43e+1	-3.41e+1
	Mean	3.28e+0	-2.35e+1	2.87e-4	1.41e+1	-3.53e+1
	Std	5.42e+0	2.53e+0	0.00e+0	2.72e-1	9.81e-1
1.5e+05	Best	0.00e+0	-2.84e+1	2.87e-4	1.38e+1	-3.68e+1
	Worst	1.45e+1	-2.21e+1	2.87e-4	1.43e+1	-3.42e+1
	Mean	3.28e+0	-2.51e+1	2.87e-4	1.40e+1	-3.56e+1
	Std	5.42e+0	2.05e+0	0.00e+0	2.05e-1	9.04e-1

FEs		F-6	F-7	F-8	F-9	F-10
5e+04	Best	-2.92e+1	5.00e-1	2.20e+2	3.38e+3	-2.18e+1
	Worst	-2.21e+1	1.27e+0	2.20e+2	1.98e+4	-2.14e+1
	Mean	-2.62e+1	1.00e+0	2.20e+2	9.74e+3	-2.16e+1
	Std	2.78e+0	1.73e-1	0.00e+0	5.18e+3	1.94e-1
1e+05	Best	-2.92e+1	5.00e-1	2.20e+2	2.17e+3	-2.18e+1
	Worst	-2.30e+1	1.06e+0	2.20e+2	8.89e+3	-2.14e+1
	Mean	-2.64e+1	7.58e-1	2.20e+2	5.51e+3	-2.17e+1
	Std	2.59e+0	1.40e-1	0.00e+0	1.92e+3	1.51e-1
1.5e+05	Best	-2.92e+1	5.00e-1	2.20e+2	1.97e+3	-2.18e+1
	Worst	-2.30e+1	9.93e-1	2.20e+2	8.04e+3	-2.14e+1
	Mean	-2.65e+1	6.56e-1	2.20e+2	4.55e+3	-2.17e+1
	Std	2.40e+0	1.16e-1	0.00e+0	1.97e+3	1.50e-1

FEs		F-11.1	F-11.2	F-11.3	F-11.4	F-11.5
5e+04	Best	6.50e+4	1.10e+6	1.54e+4	1.82e+4	3.27e+4
	Worst	5.59e+5	1.33e+6	1.54e+4	1.86e+4	3.31e+4
	Mean	2.32e+5	1.19e+6	1.54e+4	1.84e+4	3.28e+4
	Std	1.34e+5	6.06e+4	4.00e-3	1.02e+2	6.23e+1
1e+05	Best	5.15e+4	1.07e+6	1.54e+4	1.81e+4	3.27e+4
	Worst	1.64e+5	1.11e+6	1.54e+4	1.83e+4	3.29e+4
	Mean	6.27e+4	1.08e+6	1.54e+4	1.82e+4	3.28e+4
	Std	2.46e+4	1.14e+4	0.00e+0	6.22e+1	4.54e+1
1.5e+05	Best	5.13e+4	1.07e+6	1.54e+4	1.81e+4	3.27e+4
	Worst	7.15e+4	1.11e+6	1.54e+4	1.83e+4	3.29e+4
	Mean	5.41e+4	1.08e+6	1.54e+4	1.82e+4	3.28e+4
	Std	4.72e+3	9.04e+3	0.00e+0	5.45e+1	4.23e+1

FEs		F-11.6	F-11.7	F-11.8	F-11.9	F-11.10
5e+04	Best	1.32e+5	1.93e+6	1.05e+6	1.04e+6	1.04e+6
	Worst	1.47e+5	3.35e+6	1.56e+6	2.08e+6	1.87e+6
	Mean	1.40e+5	2.32e+6	1.28e+6	1.42e+6	1.29e+6
	Std	3.90e+3	3.71e+5	1.49e+5	2.39e+5	1.78e+5
1e+05	Best	1.31e+5	1.91e+6	9.95e+5	1.01e+6	9.58e+5
	Worst	1.45e+5	3.31e+6	1.30e+6	1.77e+6	1.36e+6
	Mean	1.37e+5	2.19e+6	1.11e+6	1.21e+6	1.14e+6
	Std	3.00e+3	3.13e+5	9.30e+4	1.85e+5	9.45e+4
1.5e+05	Best	1.31e+5	1.91e+6	9.64e+5	9.76e+5	9.55e+5
	Worst	1.41e+5	2.56e+6	1.22e+6	1.51e+6	1.27e+6
	Mean	1.36e+5	2.06e+6	1.06e+6	1.14e+6	1.08e+6
	Std	2.57e+3	1.58e+5	7.53e+4	1.49e+5	7.04e+4

FEs		F-12	F-13
5e+04	Best	9.56e+0	1.04e+1
	Worst	1.75e+1	2.24e+1
	Mean	1.42e+1	1.66e+1
	Std	3.01e+0	4.42e+0
1e+05	Best	8.88e+0	1.00e+1
	Worst	1.44e+1	2.12e+1
	Mean	1.26e+1	1.52e+1
	Std	2.17e+0	4.18e+0
1.5e+05	Best	6.78e+0	8.84e+0
	Worst	1.32e+1	1.97e+1
	Mean	1.15e+1	1.28e+1
	Std	2.44e+0	3.65e+0

VII. Conclusions

We have applied our algorithm to 22 real world optimization problems for CEC-2011. The above tables give the best, worst, mean and standard deviation values of 25 independent runs of the algorithm on the problems. The results are taken after 50000, 100000 and 150000 FEs. It has been noted that for functions F-3, F-4, F-6, F-8, F-10, F-11.3, F-11.4, F-11.5 we don't need maximum value of FEs, i.e. 150000, rather, we have reached the optimal values before 50000 FEs. For the other functions such as cassini2, messengerfull, circular antenna problem, Lennard jones potential problem, transmission pricing problem, we have noted that the function value is improving during entire optimization process. It may so happen that, if we increase maximum value of FE, the function value may improve further.

Acknowledgements

This work was supported by the Czech Science Foundation, under the grant no. GA102/09/1494.

References

[1] R. Storn and K. V. Price, “Differential evolution-a simple and efficient heuristic for global optimization over continuous spaces,” Journal of Global Optimization 11:341-359.1997.

[2] K. V. Price, R. Storn, and J. Lampinen, *Differential Evolution - A Practical Approach to Global Optimization*, Springer, Berlin, 2005

[3] S. Das and P. N. Suganthan, “Differential Evolution – a survey of the state-of-the-art”, *IEEE Transactions on Evolutionary Computation*, Vol. 15, No. 1, pp. 4 – 31, Feb. 2011.

[4] J. Ilonen, J.-K. Kamarainen and J. Lampinen, ”Differential Evolution Training Algorithm for Feed-Forward Neural Networks,” In: *Neural Processing Letters* Vol. 7, No. 1 93-105. 2003.

[5] R. Storn, “Differential evolution design of an IIR-filter,” In: *Proceedings of IEEE Int. Conference on Evolutionary Computation ICEC'96*. IEEE Press, New York. 268-273. 1996.

[6] A. R. Mehrabian and C. Lucas, “A novel numerical opmization algorithm inspired from weed colonization,” *Ecological Informatics*, vol. 1, pp. 355–366, 2006.

[7] A. R. Mallahzadeh, S. Es'haghi, and A. Alipour, “Design of an e-shaped mimo antenna using IWO algorithm for wireless application at 5.8 GHz”, *Progress In Electromagnetics Research,* PIER 90, 187 - 203, 2009.

[8] A. R. Mallahzadeh, S. Es'haghi, and H. R. Hassani, Compact U-array MIMO antenna designs using IWO algorithm, *International Journal of RF and Microwave Computer-Aided Engineering*, Wiley-InterSscience, DOI: 10.1002/mmce.20379, Jul, 2009.

[9] Tanaka, F., Kameda, A., Yamamoto, M. and Ohuchi, A., Design of nucleic acid sequences for DNA computing based on a thermodynamic approach. Nucleic Acids Research. v3. 903-911.

[10] S. Karimkashi, and A. A. Kishk, "Invasive weed optimization and its features in electromagnetics," *IEEE Transactions on Antennas and Propagation*, Vol. 58, Issue 4, pp.1269 – 1278, April, 2010.