ISSN-L: 2223-9553, ISSN: 2223-9944
Vol. 3, No. 2, September 2012

COMPARISON OF THE PERFORMANCES OF DIFFERENTIAL EVOLUTION, PARTICLE SWARM OPTIMIZATION AND HARMONY SEARCH ALGORITHMS ON BENCHMARK FUNCTIONS

Ezgi Deniz ÜLKER
Computer Engineering Department,
Girne American University,
TURKEY.
ezgideniz@gau.edu.tr

Ali HAYDAR
Computer Engineering Department,
Girne American University,
TURKEY.
ahaydar@gau.edu.tr

ABSTRACT

In the literature, there are many efficient metaheuristic algorithms for optimizing real-valued objective functions. In this work, Particle Swarm Optimization (PSO) algorithm, Differential Evolution (DE) algorithm and Harmony Search (HS) algorithm are compared on the basis of speed of convergence of the algorithms. In order to achieve this goal, a set of benchmark functions that have different characteristics are used.

Keywords: Optimization Algorithm, Differential Evolution Algorithm, Particle Swarm Optimization, Harmony Search Algorithm.

INTRODUCTION

Many different metaheuristic optimization algorithms have been developed for the optimization of real-valued problems. These algorithms have attracted the interest of researchers because of their ability to reach to the global optimum without finding a good starting point or without requiring gradient information. In order to test the performance of these algorithms, many different bechmark functions are tested in the literature (Karaboga & Akay, 2009). These functions are classified as unimodal (U) or multimodal (M) and also they can be separable (S) or non-separable (N) functions.

In this paper, 22 benchmark functions are used in order to compare the speed of convergence of the PSO, DE and HS algorithms. These functions are selected in such a way that they are mainly forming four different characteristic groups which are Unimodal and Separable (US), Unimodal and Non-separable (UN), Multimodal and Separable (MS), and lastly Multimodal and Non-separable (MN).

The origin of Particle Swarm Optimization has involved anologues of birds flocks searching for corn and it has been developed to be a powerful optimization algorithm (Kennedy & Eberhart, 1995). In this algorithm, the entities, which are called particles, are placed in the search space and by the movement of these particles, according to some aspect of history information, the aim is to move close to an optimum of the function.

The Harmony Search algorithm is based on the usage of natural musical performance processes in order to search for a perfect state of harmony (Lee & Geem, 2005). This idea is very similar to the optimization process that aims to find a global solution with several same processes.

Differential Evolution algorithm is a population-based algorithm that has mutation, crossover and selection operators. This algorithm has been shown to be one of the very efficient methods of finding the global optimum of real-valued functions (Storn & Price, 1997).

In the next section, we briefly explain the 3 algorithms used for optimizing the selected benchmark functions.

DIFFERENTIAL EVOLUTION ALGORITHM

Differential Evolution (DE) algorithm is a metaheuristic method which converges to the global optimum fast. It was proposed by Storn (Storn, 1996) and became popular by many researches in various fields for solving optimization problems. DE algorithm is a population based algorithm and in despite of the initial values of the population it can approach to the optimal solution with the help of few parameters. Parameters are used to control the search space.

DE algorithm's steps are similar with the steps of Genetic Algorithm (GA). *Mutation, crossover* and *selection* are the similar operators that are used in DE algorithm and GA. The population in DE is developed by applying the above operators.

The general steps of DE is explained as follows:

```
Initialize the candidate solutions $\chi_i$ with random places in search space.
Define the objective function $f$ which is needed to be optimized.
Repeat
    Pick randomly three values called $a, b$ and $c$ in which they are distinct from the
    current candidate.
    Mutant vector $v_i$ is generated using the $a, b, c$ and $F$ parameters.                  } mutation
    For each candidate $\chi_i$, pick a uniformly distributed number $r$.
      Candidate solutions and mutant vector $v_i$ is used to perform a trial vector $U_i^{'}$.
          If; $r \leq CR$  $U_i^{'} = v_i$                                                  } crossover
          Otherwise; $U_i^{'} = \chi_i$
              If $f(U^{'}) < f(\chi)$; replace the current candidates with the new positions of
              candidates.                                                                   } selection
Until termination criteria is met
`
```

F is a random number in the range (0,2) which is called *'differential weight'* and *CR* is a random number in the range (0,1) and it is called '*crossover probability*'. The selection of *F, CR* and the dimension of the function effects the performance of DE.

In this paper, different values of parameters are selected in order to have a good performance. Price and Storn denotes the importance of selection of parameters in performance of DE algorithm (Storn & Price, 1997).

PARTICLE SWARM OPTIMIZATION ALGORITHM

Particle Swarm Optimization (PSO) is a metaheuristic algorithm developed by Kennedy and Eberhart (Kennedy & Eberhart, 1995). It is inspired by the behaviour of bird or fish swarms. The potential candidates are called particles and PSO relies on moving the particles towards the best solutions in the search space (Shi &Eberhart, 1998). PSO can be optimized using the following parameters c_1, c_2, and ω. The term c_1 is called *cognitive parameter*, and c_2 is the *social parameter* and ω is called *inertia weight*. v_{min} and v_{max} can be used as a *constriction parameters* that define the maximum positions in the search space. Combination of these parameters enables to approach the optimal solution rapidly (Shi & Eberhart , 2000).

Each particle in the search space moves with some given parameters and formula. Particles move with respect to their own best position (P_{best}) and the swarm's best position ($global_{best}$). When a better position is being discovered by a particle, all particles try to improve their positions to this better position.

The main steps of the PSO algorithm are given as follows:

a. Initialize the particles χ_i with initial random positions in search space.
b. Initialize the velocities of particles v_i in a given range randomly.
c. Initialize the best known positions (P_{best}) of each particle.
d. Define the objective function f which is need to be optimized.

ISSN-L: 2223-9553, ISSN: 2223-9944
Vol. 3, No. 2, September 2012

Repeat

For each particle calculate the distance to the optimal solution called fitness.

If $f(\chi) < f(P_{best})$; $\chi = P_{best}$

Select $global_{best}$ among the $P_{best.}$

If $f(global_{best})$ reaches to the optimal solution; terminate the algorithm.

Otherwise; update the velocity v_i and particles χ_i according to given formula:

$v_i = \omega + v_i + random[0- c_1]*(P_{best}- \chi_i)+random[0- c_2]*(global_{best}- \chi_i)$

$\chi_i = \chi_i + v_i$

Until termination criteria is met.

HARMONY SEARCH ALGORITHM

Harmony Search (HS) algorithm is a metaheuristic algorithm and it is inspired by the Jazz musicians and developed by Geem *et.al.,* (2001). There is a parallel idea between HS and how Jazz musicians create harmony when they play the music. In other words, the process of Jazz improvisation and the process of reaching the optimal solution is the inspiration of the HS algorithm. Each candidate variable corresponds to a musician and generates a note or value in order to find the best harmony or the optimum solution.

Musicians play the notes randomly or use their previous experiences in order to find the best harmony. The same idea is used in HS algorithm. Each candidate variable is generated in random manner or previously generated good values assigned to candidates in order to find the optimal value.

HS algorithm is based on few parameters; *hmcr, par* and *fw*. The parameter *hmcr* is called *harmony memory considering rate* and it denotes the rate of choosing candidates from the Harmony Memory *(HM)* and generally changes from 0.7 to 0.99. The parameter *par* is called *pitch adjusting rate* and indicates the rate of choosing a neighbouring value and can be selected from 0 to 1. The parameter *fw* is called *fret width* in a musical instrument, but in HS algorithm it denotes the amount of maximum rate of change in the pitch adjustment (Lee & Geem, 2005).

HS algorithm has the following steps:

a. Define harmony memory size *(hms)*. The size of candidates depends on *hms.*

b. Initialize the candidates χ_i randomly and store them in Harmony Memory *(HM).*

c. Define the objective function f which is needed to be optimized.

d. Define the *hmcr, par* and *fw.*

Repeat

With a given probability of *hmcr*, select the value from HM.

With a given probability of 1-*hmcr*, generate a random value in the given range.

With a given probability of *par,* update the candidates χ_i by applying the following formula:

$\chi_{i\_new} = \chi_{i\_old} + u(0-1)*fw$
or
$\chi_{i\_new} = \chi_{i\_old} - u(0-1)*fw$

With a given probability of 1-*par,* do not update the candidates χ_i.

If $f(\chi_i) < f(\chi_{worst})$; replace χ_{worst} with χ_i.

Until termination criteria is met.

EXPERIMENTAL RESULTS

In this paper, the performances of the DE algorithm, PSO algorithm and HS algorithm are compared on some well known benchmark functions. Twenty-two bechmark functions are selected as unimodal or multimodal, separable or non-separable. The selected bechmark functions are given in the Table 1.

Table 1. Benchmark functions used in experiments

No	Range	D	Function	Formulation	f_{min}	C
1	[-100, 100]	30	Step	$f(x)=\sum_{i=1}^{n}(\lfloor x_i+0.5\rfloor)^2$	0	US
2	[-100, 100]	30	Sphere	$f(x)=\sum_{i=1}^{n}{x_i}^2$	0	US
3	[-10, 10]	30	SumSquares	$f(x)=\sum_{i=1}^{n}i{x_i}^2$	0	US
4	[-1.28, 1.28]	30	Quartic	$f(x)=\sum_{i=1}^{n}i{x_i}^4+\mathrm{random}[0,1)$	0	US
5	[-D^2, D^2]	6	Trid6	$f(x)=\sum_{i=1}^{n}(x_i-1)^2-\sum_{i=2}^{n}x_ix_{i-1}$	-50	UN
6	[-D^2, D^2]	10	Trid10	$f(x)=\sum_{i=1}^{n}(x_i-1)^2-\sum_{i=2}^{n}x_ix_{i-1}$	-210	UN
7	[-5, 10]	10	Zakharov	$f(x)=\sum_{i=1}^{n}{x_i}^2+\left(\sum_{i=1}^{n}0.5ix_i\right)^2+\left(\sum_{i=1}^{n}0.5ix_i\right)^4$	0	UN
8	[-4, 5]	24	Powell	$f(x)=\sum_{i=1}^{n/k}(x_{4i-3}+10x_{4i-2})^2+5(x_{4i-1}-x_{4i})^2+(x_{4i-2}-x_{4i-1})^4+10(x_{4i-3}-x_{4i})^4$ k=4	0	UN
9	[-10, 10]	30	Schwefel 2.22	$f(x)=\lvert x_i\rvert+\prod_{i=1}^{n}\lvert x_i\rvert$	0	UN
10	[-100, 100]	30	Schwefel 1.2	$f(x)=\sum_{i=1}^{n}\left(\sum_{j=1}^{i}x_j\right)^2$	0	UN
11	[-30, 30]	30	Rosenbrock	$f(x)=\sum_{i=1}^{n-1}[100(x_{i+1}-{x_i}^2)^2+(x_i-1)^2]$	0	UN
12	[-10, 10]	30	Dixon-Price	$f(x)=(x_1-1)^2+\sum_{i=2}^{n}i(2{x_i}^2-x_{i-1})^2$	0	UN
13	[-10, 10]	2	Booth	$f(x)=(x_1+2x_2-7)^2+(2x_1+x_2-5)^2$	0	MS
14	[-5.12, 5.12]	30	Rastrigin	$f(x)=\sum_{i=1}^{n}[{x_i}^2-10\cos(2\pi x_i)+10]$	0	MS

15	[-500, 500]	30	Schwefel	$f(x)=\sum_{i=1}^{n}-x_i\sin(\sqrt{\lvert x_i\rvert})$	-418.9829*D	MS
16	[0, π]	10	Michalewicz10	$f(x)=-\sum_{i=1}^{n}\sin(x_i)(\sin\left(\frac{ix_i^2}{\pi}\right))^{2m}$ m=10	-9.6602	MS
17	[-100, 100]	2	Schaffer	$f(x)=0.5+\frac{\sin^2(\sqrt{x_1^2+x_2^2})-0.5}{(1+0.001(x_1^2+x_2^2))^2}$	0	MN
18	[-5, 5]	2	Six Hump Camel Back	$f(x)=4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-4x_2^2+4x_2^4$	-1.03163	MN
19	[-10, 10]	2	Shubert	$f(x)=\left(\sum_{i=1}^{5}i\cos((i+1)x_1+i\right)\left(\sum_{i=1}^{5}i\cos((i+1)x_2+i\right)$	-186.73	MN
20	[-600, 600]	30	Griewank	$f(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	0	MN
21	[-32, 32]	30	Ackley	$f(x)=-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-\exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i))+20+e$	0	MN
22	[-50, 50]	30	Penalized	$f(x)=\frac{\pi}{n}\left\{10\sin^2(\pi y_1)+\sum_{i=1}^{n-1}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})]+(y_n-1)^2\right\}+\sum_{1}^{n}u(x_i,10,100,4)$ $y_i=1+\frac{1}{4}(x_i+1)$ $u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m, & x_i>a\\ 0, & -a\le x_i\le a\\ k(-x_i-a)^m, & x_i<-a\end{cases}$	0	MN

D: Dimension, C: Characteristic, U: Unimodal, M: Multimodal, S: Separable, N: Non-Separable

For all of the three algorithms used in this study, the number of population is fixed to 100. All results are averaged over 20 runs. The performance of the meta-heuristic algorithms mainly depends on the selection of control parameters correctly. For the DE algorithm, the two control parameters which are *F* and *CR* are selected from the sets {0.3, 0.5, 0.7, 0.8, 0.9, 1.2, 1.4}, {0.1, 0.2, 0.4, 0.6, 0.8, 0.9} respectively. For the PSO algorithm, the number of control parameters are more than the DE algorithm and are selected from the sets as given below;

$c1 \in \{0.3, 0.6, 0.9, 1.2, 1.5, 1.8\}$

$c2 \in \{0.3, 0.6, 0.9, 1.2, 1.5, 1.8\}$

$w \in \{0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$

v_{min} and v_{max} are fixed to the upper and lower bound of each function in PSO. Lastly, for the HS algorithm, the control parameter *hmcr* is selected from the set {0.7, 0.8, 0.9, 0.93, 0.96, 0.98} and the control parameter *par* is selected from the set {0.01, 0.02, 0.05, 0.1, 0.2} . The parameter *fw* is adjusted as 0.01, 0.05, 0.1, and 0.2 of upper bound of each function in HS.

In Table 2 through 5, the performance of these three algorithms over 100000 evaluations is shown. For each algorithm, the best and the average of 20 runs are tabulated for the best combination of control parameters for that function.

Table 2. Performance of the Algorithms on US functions

No	Function	DE		PSO		HSA	
	US	Reached optimal	Average	Reached optimal	Average	Reached optimal	Average
1	Step	0	0	0	0	0	0
2	Sphere	4.13886e-025	1.1e-24	4.20318e-033	2.68281e-030	0.000172047	0.000306177
3	SumSquares	6.56945e-027	2.43e-26	2.29419e-035	1.33534e-033	0.000785468	3.60788e-005
4	Quartic	0.000834919	0.001184	0.00073759	0.00426387	3.42479e-005	6.30626e-005

Table 3. Performance of the Algorithms on UN functions

No	Function UN	DE		PSO		HSA	
		Reached optimal	Average	Reached optimal	Average	Reached optimal	Average
1	Trid6	-50	-50	-50	-50	-50	-50
2	Trid10	-210	-210	-210	-210	-209.737	-209.678
3	Zakharov	1.79694e-66	3.74e-65	1.05622e-128	2.13049e-123	7.71711e-009	1.87231e-005
4	Powell	3.22879e-06	7.8e-06	1.84456e-005	8.84646e-005	0.0287128	0.0152576
5	Schwefel 2.22	1.80172e-14	2.68e-14	3.02974e-013	1.03357e-011	0.0028714	0.00442567
6	Schwefel 1.2	2.54742	19.6012	0.000657878	0.00870464	182.804	365.869
7	Rosenbrock	20.7731	22.3484	1.95745	18.0948	1.6957	53.3728
8	Dixon-Price	3.85044e-08	3.34E-07	2.3402e-021	1.55883e-015	0.926283	1.09333

Table 4. Performance of the Algorithms on MS functions

No	Function MS	DE		PSO		HSA	
		Reached optimal	Average	Reached optimal	Average	Reached optimal	Average
1	Booth	0	0	0	0	9.49694e-010	2.21531e-009
2	Rastrigin	0.00589048	2.84916	15.9194	22.6993	4.02884e-005	0.000101882
3	Schwefel	-12569.5	-12569.5	-11740.4	-11740.4	-12569.5	-12569.5
4	Michalewicz10	-9.66015	-9.66015	-9.65277	-9.48277	-9.66015	-9.66015

ISSN-L: 2223-9553, ISSN: 2223-9944
Vol. 3, No. 2, September 2012

Table 5. Performance of the algorithms on MN functions

No	Function MN	DE		PSO		HSA	
		Reached optimal	Average	Reached optimal	Average	Reached optimal	Average
1	Schaffer	0	0	0	0	1.77453e-008	1.21851e-007
2	Six Hump Camel Back	-1.03163	-1.03163	-1.03163	-1.03163	-1.03163	-1.03163
3	Shubert	-186.731	-186.731	-186.731	-186.731	-186.731	-186.731
4	Griewank	0	1.08e-20	5.48962e-011	0.00910358	0.000306415	0.00159353
5	Ackley	4.48102e-008	7.5e-08	2.79584e-008	5.59113e-007	0.00361203	0.00486016
6	Penalized	8.16099e-027	1.34e-25	5.42457e-015	0.0103669	6.55047e-007	1.38665e-006

The experimental results have shown that for US functions the PSO algorithm converges faster than the other two algorithms. For the UN functions, the performance of DE and PSO is comparable in the first five functions. However, the convergence speed of PSO algorithm for the last three functions is much more faster than both DE and HS algorithm.

Unlike the unimodal functions, the convergence speed and performance of DE and HS algorithm is better than the PSO for MS functions. In Table 4, we can easily observe that for MS functions, the convergence speed of DE and HS algorithms are comparable and better than PSO algorithm.

Lastly for the MN functions, the convergence speed of the algorithms are similar for the first three functions. For function 4 and 6 the convergence speed of DE is better than the other two algorithms.

CONCLUSION

From the experimental results, we can summarize our findings as follows;

1 For Unimodal functions, the convergence speed of PSO algorithm is better than the other two algorithms.

2 For Multimodal Separable functions, the DE algorithm is a very efficient method for finding the optimal solution and its convergence speed is much more faster than the PSO algorithm and slightly faster than HS algorithm.

3 For the Multimodal Non-separable functions, when we take into account all 6 functions used, the convergence speed of DE is better than the other two algorithms.

REFERENCES

Geem, Z.W., Kim, J.H. and Loganathan, G.V. (2001). A New Heuristic Optimization Algorithm: Harmony Search. *Simulation*, *Transaction of the Society for Modelling and Simulation International,* pp. 60-68.

Karaboga, D. and Akay, B. (2009). A Comparative Study of Artificial Bee Colony Algorithm. *Applied Mathematics and Computation*, No.214, pp.108-132.

Kennedy, J. and Eberhart, R. (1995). Particle Swarm Optimization. *Proceedings of IEEE International Conference on Neural Networks. IV.,* pp. 1942–1948.

Lee, K.S. and Geem, Z.W. (2005). A New Meta-Heuristic Algorithm for Continuous Engineering Optimization: Harmony Search Theory and Practice. *Computer Methods in Applied Mechanics and Engineering*, pp. 3902-3933.

Shi, Y. and Eberhart, R. (1998). Parameter selection in particle swarm optimization. *Proceedings of Evolutionary Programming*, pp. 591–600.

Shi, Y. and Eberhart, R. (2000). Comparing inertia weights and constriction factors in particle swarm optimization. *Proceedings of the Congress on Evolutionary Computation.*, pp. 84–88.

Storn, R. (1996). On the usage of differential evolution for function optimization. *Biennial Conference of the North American Fuzzy Information Processing Society (NAFIPS)*. pp. 519–523.

Storn, R. and Price K. (1997). Differential Evolution; A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. *Journal of Global Optimization*, vol.11, pp. 341-359.