International Journal of Engineering Research in Computer Science and Engineering (IJERCSE) Vol 3, Issue 4, April 2016

A New Differential Evolutionary Algorithm

[1] Aparna Potbhare, [2] Rutuja Deogade [3] Prof. Amit Khaparde
[1][2][3] Dept. of Information Technology, Rajiv Gandhi College of Engineering and Research, Nagpur, India.
[1]aparnapotbhare1994@gmail.com [2] Rutujadeogade10@gmail.com [3] khaparde.amit@gmail.com

Abstract: **- Differential evolution (DE) has been proven to be one of the most powerful global numerical optimization algorithms in the evolutionary algorithm family. The core operator of DE is the differential mutation operator. Generally, the parents in the mutation operator are randomly chosen from the current population. In nature, good species always contain good information, and hence, they have more chance to be utilized to guide other species. Inspired by this phenomenon, To improve the efficiency of the original differential evolution algorithm, a new differential evolution algorithm was proposed. This paper presents a novel algorithm to accelerate the differential evolution (DE). The proposed opposition-based DE (ODE) employs opposition-based learning (OBL) for population generation jumping. In this work, opposite numbers have been utilized to improve the convergence rate of DE.**

Keywords**- Differential evolution (DE), opposition-based learning (OBL), mutation operator.**

I. INTRODUCTION

EVOLUTIONARY algorithms (EAs) are stochastic search methods that mimic evolutionary processes encountered in nature. The common conceptual base of these methods is to evolve a population of candidate solutions by simulating the main processes involved in the evolution of genetic material of organism populations, such as natural selection and biological evolution. EAs can be characterized as global optimization algorithms.

Their population-based nature allows them to avoid getting trapped in a local optimum and consequently provides a great chance to find global optimal solutions. EAs have been successfully applied to a wide range of optimization problems, such as image processing, pattern recognition, scheduling, and engineering design [1], [2]. The most prominent EAs proposed in the literature are genetic algorithms [1], evolutionary programming [3], evolution strategies [4], genetic programming [5], particle swarm optimization (PSO) [6], and differential evolution [7], [8].In general, every EA starts by initializing a population of candidate solutions (individuals). The quality of each solutions evaluated using a fitness function, which represents the problem at hand. A selection process is applied at each iteration of the EA to produce a new set of solutions (population). The selection process is biased toward the most promising traits of the current population of solutions to increase their chances of being included in the new population. At each iteration (generation), the individuals are evolved through a predefined set of operators, like *mutation* and *recombination*. This procedure is repeated until convergence is reached

The best solution found by this procedure is expected to be a near-optimum solution [2], [9].*Mutation* and *recombination* are the two most frequently used operators and are referred to as *evolutionary* operators. The role of *mutation* is to modify an individual by small random changes to generate a new individual [2], [9]. Its main objective is to increase diversity by introducing new genetic material into the population, and thus avoid local optima. The*re combination* (or crossover) operator combines two, or more, individuals to generate new promising candidate solutions [2],[9]. The main objective of the recombination operator is to explore new areas of the search space [2], [10].

In this paper A new differential evolutionary algorithm with species and best vector selection has been proposed. It uses best determination method to determine the best members in population. Each best member is considered as a niche in population. The species formation takes place around these niches. Once the species get formed then the standard differential evolution algorithm has been used. If species is not performing well, then the opposition learning is used. The scale up study of various parameters is done to get best parameter setting. The performance of newly proposed algorithm is tested on uni-modal and multi-modal test functions. It got success in solving wide range of problems. The results are compared with standard Differential evolution algorithm (SDE) and other state-of-art algorithms. The results are encouraging one.

The rest of this paper is organized as follows. Section II describes the original differential evolution

algorithm. In Section III, describe the opposition based learning. Section IV illustrates the methodology.

II. CLASSICAL DE FAMILY OF ALGORITHMS

DE is a simple real parameter optimization algorithm. It works through a simple cycle of stages of mutation, crossover, and selection, as described in the following.

2.1 Initialization of the Parameter Vectors

DE searches for a global optimum point in a *D*-D real parameter space_*D*. It begins with a randomly initiated population of *Np D*-D real-valued parameter vectors. Each vector, also known as *genome/chromosome*, forms a candidate solution to the multidimensional optimization problem. We shall denote subsequent generations in DE by $G = 0, 1, \ldots, G\max$. Since the parameter vectors are likely to be changed over different generations, we may adopt the following notation for representing the *i*th vector of the population at the current generation:

$$\_Xi,G = [x1,i,G, x2,i,G, x3,i,G, \ldots\ldots\ldots, xD,i,G]. \quad (1)$$

For each parameter of the problem, there may be a certain range within which the value of the parameter should be restricted, often because parameters are related to physical components or measures that have natural bounds (for example, if one parameter is a length or mass, it cannot be negative). The initial population (at $G = 0$) should cover this range as much as possible by uniformly randomizing individuals within the search space constrained by the prescribed minimum and maximum bounds.

$$\_X\min = \{x1,\min, x2,\min, \ldots, xD,\min\}$$

$$\_X\max = \{x1,\max, x2,\max, \ldots, xD,\max\}.$$

Hence, we may initialize the *j*th component of the *i*th vector as

$$xj,i,0 = xj,\min + randi,j[0, 1] \cdot (xj,\max - xj,\min) \quad (2)$$

where $randi,j[0, 1]$ is a uniformly distributed random number lying between 0 and 1 (actually $0 \leq randi,j[0, 1] \leq 1$) and is instantiated independently for each component of the *i*th vector.

2.2 Mutation with Difference Vectors

After initialization, DE creates a *donor* vector $\_Vi,G$*corresponding* to each population member or *target* vector $_Xi,G$in the current generation through mutation. The five most frequently referred mutation strategies are listed in the following:

"DE/rand/1" :

$$\_Vi,G = \_Xri1,G + F \cdot (\_Xri2,G - \_Xri3,G) \quad (3a)$$

"DE/best/1" :

$$\_Vi,G = \_Xbest,G + F \cdot (\_Xri1,G - \_Xri2,G) \quad (3b)$$

"DE/current-to-best/1" :

$$\_Vi,G = \_Xi,G + F \cdot (\_Xbest,G - \_Xi,G) + F \cdot (\_Xri1,G - \_Xri2,G) \quad (3c)$$

"DE/best/2" :

$$\_Vi,G = \_Xbest,G + F \cdot (\_Xri1,G - \_Xri2,G) + F \cdot (\_Xri3,G - \_Xri4,G) \quad (3d)$$

"DE/rand/2" :

$$\_Vi,G = \_Xri1,G + F \cdot (\_Xri2,G - \_Xri3G) + F \cdot (\_Xri4,G - \_Xri5G). \quad (3e)$$

The indices *ri*1, *ri*2, *ri*3, *ri*4, and *ri*5 are mutually exclusive integers randomly chosen from the range $[1,Np]$, and all are different from the index *i*. These indices are randomly generated anew for each donor vector. The scaling factor *F* is a positive control parameter for scaling the difference vectors. $_Xbest,G$is the best individual vector with the best fitness (i.e., lowest objective function value for a minimization problem) in the population at generation *G*. In (3c), $_Xi,G$is known as the target vector, and $_Vi,G$is known as the donor vector. The general convention used for naming the various mutation strategies is DE/*x*/*y*/*z*, where DE stands for differential evolution, *x*represents a string denoting the vector to be perturbed, and *y* is the number of difference vectors considered for the perturbation of*x*. *z* stands for the type of crossover being used (*exp*: exponential and *bin*: binomial). The following section discusses the crossover step in DE.

2.3 Crossover

Through crossover, the donor vector mixes its components with the target vector $_Xi,G$under this operation to form the trial vector $\_Ui,G = [u1,i,G, u2,i,G, u3,i,G, \ldots, uD,i,G]$. The DE family of algorithms uses mainly two kinds of crossover methods—*exponential* (or two-point modulo) and *binomial* (or uniform) [2]. We only elaborate here the binomial crossover as the proposed DE variant uses this scheme. Binomial crossover is performed on each of the *D* variables whenever a randomly generated number between 0 and 1 is less than or equal to the *Cr* value. In this case, the

number of parameters inherited from the donor has a (nearly) binomial distribution. The scheme may be outlined as

$uj,i,G=\{vj,i,G,$ if $(randi,j[0, 1]\leq Cr$ or $j=1rand)$ $xj,i,G,$ otherwise (4)

where, as before, $randi,j[0, 1]$ is a uniformly distributed random number, which is called anew for each jth component of the ith parameter vector. $jrand\in[1, 2, \ldots,D]$ is a randomly chosen index, which ensures that $_Ui,G$gets at least one component from $\_Vi,G$. It is instantiated once for each vector per generation.

2.4 Selection

Selection determines whether the target or the trial vector survives to the next generation, i.e.,

at$G = G+ 1$. The selection operation is described as

$\_Xi,G+1 = \_Ui,G,$ If $f(\_Ui,G) \leq f(\_Xi,G) = \_Xi,G,$ if $f(\_Ui,G) > f(\_Xi,G)$ (5)
where$f(_X)$ is the objective function to be minimized. Therefore, if the new trial vector yields an equal or lower value of the objective function, it replaces the corresponding target vector in the next generation; otherwise, the target is retained in the population.

The original DE algorithm (DE/rand/1/bin) is illustrated in Algorithm 1.

Algorithm 1 Algorithmic scheme for the original Differential Evolution algorithm (DE/rand/1/bin)

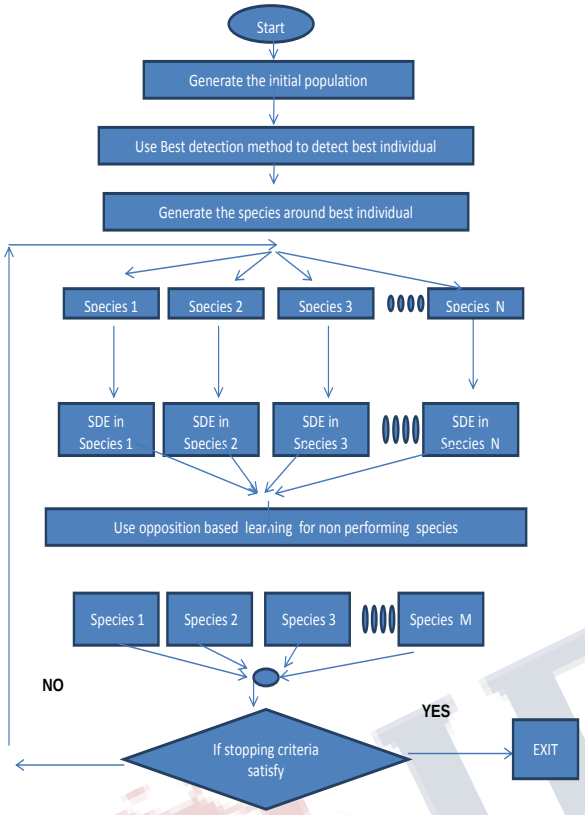
Set the generation counter $g = 0$
/* **Initialize the population** of NP individuals: $Pg=\{x1g, x2g, \ldots, xNPg\}$, with $xig=\{xi1,g, xi2,g, \ldots, xiD,g\}$ for $i =1, 2, \ldots, NP$ uniformly in the optimization search hyperrectangle$[L,U]$.*/
for$i = 1$ to NP **do**
for$j = 1$ to D **do**
$xi0,j= Lj+ randj(0, 1) \cdot(Uj-Lj)$
end for
Evaluate individual $xi0$
end for
whiletermination criteria are not satisfied **do**
Set the generation counter $g = g + 1$
for$i = 1$ to NP **do**
/* **Mutation step** */
Select uniformly random integers $r1, r2, r3 \in Sr=\{1, 2, \ldots, NP\} \setminus \{i\}$
/* For each target vector xiggenerate the correspondingmutant vector vigusing (3) */
for$j = 1$ to D **do**
$vij,g= xr1j,g + F(xr2j,g-xr3j,g)$
end for
/* **Crossover step:** For each target vector xiggenerate the corresponding trial vector uigthrough the **BinomialCrossover** scheme.*/
jrand= a uniformly distributed random integer $\in$ $\{1, 2, \ldots, D\}$
for$j = 1$ to D **do**
$uig,j=\{ vig,j,$ if $(randi,j(0, 1) \_ CR$ or $j = jrand),$
$xig,j,$ otherwise,
end for
/* **Selection step** */
if$f(uig) < f(xig)$ **then**
$xig+1 = uig$
if$f(uig) < f(xbestg)$ **then**
$xbestg= uig$and $f(xbestg) = f(uig)$
end if
else
$xig+1 = xig$
end if
end for
end while

III. OPPOSITION BASED LEARNING

Opposition-based learning was first introduced by Tizhoosh [Tizhoosh, 2005a] and later applied to Reinforcement Learning [Tizhoosh, 2005b] [Tizhoosh, 2006], Differential Evolution [Rahnamayan, 2006a] [Rahnamayan, 2006b] [Rahnamayan, 2008] and Particle Swarm Optimization [Han, 2007] [Omran, 2009] [Wang, 2007] [Wu, 2008]. Opposition-based learning is based on the concept of opposite points and opposite numbers. If x is a real number in the range [a, b], i.e. x Є [a, b] then the opposite number x' of x is defined as x' = a + b – x. For example if a = -5 and b = 5, then the opposite of x = -2 will be x' = 2. When working with n dimensional vectors, the definition of opposite numbers can be analogously extended to opposite point in n dimensions. If X(x1, x2, … ,xn) is an n dimensional vector, where xi Є [ai, bi] and i = 1, 2, … , n; then the opposite point of X is X'(x1',
x2', …,xn') where xi' = ai+ bi – xi.

The basic idea of opposition-based learning is that 50% of the time the current solution is further away from the optimum than its opposite solution. By considering both and retaining the fitter of the two we may improve our chances of finding the optimum quickly. This process can be incorporated in the initialization stage as well as during the evolution of the swarm (called generation jumping).

IV. METHODOLOGY



V. CONCLUSION

With the increasing complexity of real world optimization problems, demand for robust, fast, and accurate optimizers is on the rise among researchers from various fields. DE emerged as a simple and efficient scheme for global optimization over continuous spaces.

ACKNOWLEDGMENT

We are extremely thankful to our guide **Prof. Amit R. Khaparde** under whom our project took the shape of reality from mere idea. We are thankful to our guide for enlightening us with his precious guidance and constant encouragement. We thank our guide for providing us with ample support and valuable time. We are indebted to our guide who constantly provided a stimulus to reach our goals.

We are grateful to **Prof. M. M. Goswami**, HOD Information Technology, RGCER, for his kind co-operation and timely help.

We express our gratitude towards Dr**. A. V. Bapat,** Principal RGCER, for his never ending support and motivation.

Lastly we would like to thank all those who were directly or indirectly related to our project and extended their support to make the project successful**.**

REFERENCES

[1] D. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley, 1989.

[2] T. B¨ack, D. B. Fogel, and Z. Michalewicz, Eds., *Handbook of Evolutionary Computation*. Oxford, U.K.: Oxford University Press, 1997.

[3] L. J. Fogel, A. J. Owens, and M. J. Walsh, *Artificial intelligence Through Simulated Evolution*. New York: Wiley, 1966.

[4] H.-G. Beyer and H.-P.Schwefel, "Evolution strategies: A comprehensive introduction," *Natural Comput.*, vol. 1, no. 1, pp. 3–52, May 2002.

[5] J. R. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. Cambridge, MA: MIT Press, 1992.

[6] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw.*, vol. IV. 1995, pp. 1942–1948.

[7] R. Storn and K. Price, "Differential evolution: A simple and efficient adaptive scheme for global optimization over continuous spaces," *J. Global Optimization*, vol. 11, no. 4, pp. 341–359, Dec. 1997.

[8] R. Storn and K. Price, "Differential evolution: A simple and efficient adaptive scheme for global optimization over continuous spaces," Int. Comput.Sci. Instit., Berkeley, CA, Tech. Rep. TR-95-012, 1995 [Online]. Available: ftp://ftp.icsi.berkeley.edu

[9] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, 3rd ed. Berlin, Germany: Springer-Verlag, 1996.

[10] A. A. Salman, "Linkage crossover operator for genetic algorithms," Ph.D. dissertation, Dept. Electric.Eng. Comput.Sci., Syracuse Univ.,Syracuse, NY, 1999.