ISBN 978-987-4417-90-9

Smart Grids Challenge: A competitive variant for Single Objective Numerical Optimization.

Fabricio Loor[1], M. Guillermo Leguizamón[1], Efrén Mezura-Montes[2]

[1] Laboratorio de Investigación y Desarrollo en Inteligencia Computacional (LIDIC)
Universidad Nacional de San Luis, San Luis D 5700 BPB, Argentina
faloor,legui@unsl.edu.ar

[2] Artificial Intelligence Research Center, Universidad Veracruzana, Xalapa, Veracruz, 91000,Mexico.
emezura@uv.mx

Abstract. In this work, we present a new algorithm (AJSO) for high-dimensional single objective problems. It is well known that finding high quality solutions is still a challenge for complex problems like those found in the literature as well as in real world concerning Smart Grids scenarios. Our proposal AJSO is an improvement on a state-of-the-art differential Evolution (DE) based algorithm known as SHADE. More specifically, AJSO implements two novel mutation strategies and also incorporates a mechanism for mantaining and taking good solutions from a special archive when a particular condition during the exploration process is detected. To compare the performance of AJSO, the benchmark given in the WCCI/GECCO 2020 [4] is used. This challenge consisted of optimization problems represented in two testbeds of Smart Grids problems. In this paper we adopted the guidelines given in the WCCI/GECCO 2020 competition. Experimental results show that AJSO outperforms SHADE in the two studied testbeds.

Keywords: Optimization, Smart Grids, Metaheuristics, Differential Evolution

1 Introduction

Optimization problems require minimizing or maximizing a measure according to a given configuration of values. If the possible permutations or combinations in those configurations are really vast, thinking of exhaustive methods or brute force becomes an impractical alternative. Despite this, stochastic algorithms such as metaheuristics have been widely used to solve these situations. In particular, Differential Evolution (DE) [6] is a stochastic algorithm that has proven to be the basis for many improvements and advances, since it is simple and easy to program. For its operation, DE needs to be adjusted by two parameters: a mutation factor (F) and a probability of crossover (CR). Currently, DE-based algorithms are widely competitive with other efficient and state-of-the-art optimization algorithms [11], [13], [12].

ISBN 978-987-4417-90-9

In order to understand the work in this area, we will briefly describe some representative modifications and improvements made throughout the history of DE:
- 2005, FADE [7], Fuzzy Adaptive Differential Evolution. Fuzzy logic controllers are applied for parameters F and CR.
- 2005, SaDE [8], Self adaptive differential Evolution. Parameters F and CR are updated in each generation by normal distributions.
- 2006, jDE [2]. The values of F and CR are given by uniform distributions.
- 2009, JADE [3], introduces a new mutation strategy "*current-to-pbest/1*" by adding an archive of less efficient solutions.
- 2013, SHADE, [4], which is based on JADE and proposes a memory for the satisfactory values of F and CR. This algorithm was placed in third place in the CEC 2013 competition.
- 2014, L-SHADE [5], uses SHADE and introduces a population decrease that behaves linearly. It was the winner in the CEC 2014 competition.
- 2016, iL-SHADE [9] is an improvement over L-SHADE. This algorithm introduces changes in initialization and memory update. It also features a new mechanism to update the F and CR parameters, varying between the current and maximum generation. It was ranked fourth in the CEC 2016 competition.
- 2017, jSO [10], uses the "*current-to-pbest-w/1*" strategy and introduces modifications according to the search period on the mutation factor. It earned second place in the CEC 2017 restricted limits competition.
- 2019, jDE100 [1] is a modification over jDE that adds two populations, one old and one new. It also adds random restarts when search deadlocks are detected. It was the winner of the CEC 2019 100 Digits competition.
In this work, we use one of these improvements and propose changes to solve a high-dimensional problem applied to intelligent energy distribution.

The rest of the paper is organized as follows. In Section II, we present AJSO by describing its main features and details. In Section III, we describe the experimental design and the framework where the algorithms have been tested. Then, in Section IV, the achieved results are shown and analyzed. Finally, in Section V, the conclusions reached and some future works research lines are summarized.

2 Our approach: AJSO (Advanced JSO)

In this section, a new approach (AJSO) for solving single-objective real parameter optimization is presented and its pseudo-code is shown in Algorithm 1. AJSO is an enhanced version of SHADE [4] and jSO [10], two well-known algorithms based on DE. SHADE preserves two population structures, the first one keeps the current population and the other one (an archive) contains the replaced individuals throughout the search process. It also incorporates a recently successful historical parameter memory to guide the generation of parameter control values, for both crossover and mutation operators.

In our proposal, three structures are used to maintain different type of solutions. The first one is the main population P_g, the second one maintains an

ISBN 978-987-4417-90-9

Archive of useful recent generated solutions, and the last one preserves the history of best individuals found at each generation (A_g). The Population and Archive structures have a fixed N size, while the third one grows dynamically at each generation. The process to update the control parameters is different than in SHADE and will be described later.

Before entering the main loop, the population is initialized, and the archive equals the population. In this part of the code, the memory is given random values that follow a normal distribution. These values in turn will be the mean for future calculations of the respective values of parameters CR and F. After initialization, the algorithm enters the evolutionary stage. The variable sF is described in [4] This stage continues until the stop criterion is reached, which in our case is given by a maximum number of evaluations completed. In the body of the main loop, a population is evaluated with two different mutation strategies, to subsequently carry out eight evaluations with the best strategy.

Algorithm 1: AJSO

```
1  //Initialization phase
2  g = 1, N_g = N_init;
3  Initialize population P_g = (x_{1,g}, ..., x_{N,g}) randomly;
4  Set all values in M_CR, M_F following a normal distribution with
     μ_CR = I_CR and σ_CR = 0.05 ; μ_F = I_F and σ_F = 0.1;
5  Archive = P_g;
6  // Main loop
7  while The termination criterion is not met do
8  |  Sort and select the best B individuals;
9  |  Obtain a vector of values with CR and sF;
10 |  From sF get jF;
11 |  Generate two new populations with two different strategies;
12 |  Control limits and evaluate;
13 |  Select the best strategy S;
14 |  // Exploitation phase
15 |  for i = 1 to 8 do
16 |  |  Sort and select the best B individuals;
17 |  |  Obtain a vector of values with CR and sF;
18 |  |  From sF get jF;
19 |  |  Generate a new population with the best strategy from 13;
20 |  |  Control limits and evaluate;
21 |  |  Update Archive;
22 |  |  Update Memory Parameters;
23 |  |  if nsp = 0 and B_g = BSF and nfes > I_evalmax * 0.4 then
24 |  |  |  add in the Archive one of the best global past (A_g);
25 |  |  end
26 |  end
27 end
```

ISBN 978-987-4417-90-9

2.1 Mutation and crossing process

The SHADE algorithm [4] builds a mutant vector $\vec{v}_i$, according to the strategy "*current-to-pbest/1*" as shown in the next equation:

$$\vec{v}_{i,g+1} = \vec{v}_{i,g} + sF * (\vec{v}_{pbest} - \vec{v}_{i,g} + \vec{v}_{r1} - \vec{v}_{r2}),$$

where individual v_{pbest} is randomly selected from the top $N * p$ ($p \in [0, 1]$) members in generation g. The indices i, $r1$ and $r2$ are randomly selected from [1,N] such that they differ from each other.

AJSO (our algorithm) incorporates two ways to mutate solutions by following two different strategies. In order to this, element of the population or elements of the archive are part of the strategies according to the need of the moment. The first strategy uses elements of the population to generate individual i of generation $g + 1$:

$$\vec{v}_{i,g+1} = \begin{cases} \vec{v}_{pr} + sF * (\vec{v}_{r1} - \vec{v}_{i,g}) + jF * (\vec{v}_{r1} - \vec{v}_{r2}), & \text{if } rand(0,1) < C\vec{R}_i. \\ \vec{v}_{i,g}, & \text{otherwise,} \end{cases}$$

where sF is a scalar obtained from applying a normal distribution on a random value in memory. jF is a proportion less than sF and, v_{pr} is a random vector among the best B elements of the previous generation. The second strategy is stated as follows:

$$\vec{v}_{i,g+1} = \begin{cases} \vec{v}_{pr} + sF * (\vec{v}_{r1} - \vec{v}_{i,g}) + jF * (\vec{v}_{r3} - \vec{v}_{r4}), & \text{if } rand(0,1) < CR_i. \\ v_{i,g}, & \text{otherwise.} \end{cases}$$

Here, vectors $\vec{v}_{r3}$, $\vec{v}_{r4}$ are element randomly selected from *Archive* and different from each other.

2.2 Selecting the best strategy

After checking that the modified solutions are within the allowed limits of the decision variables, the algorithm proceeds to evaluate these solutions. When the evaluation concludes, a selection process is carried out to obtain the best strategy between the two used in the previous step. The selection criterion is that which produced a new population with more improvements with respect to the old one.

2.3 Updating the historical memory

In this section we describe the update of the values for the control memories M_F and M_{CR}. The elements of historical memory, similarly used in SHADE, are updated in the exploitation stage (lines 14 to 26 of Algorithm 1). For this purpose, the evaluation of each population also counts the numbers of individuals that improve compared with the respective old values. That value is called nsp for the Number of Successful Parameters, its complement respect to the total number of new individuals generated is called Number of Fail Parameters (nfp).

ISBN 978-987-4417-90-9

If more than a half of the individuals in a generation improved the previous population ($nsp > nfp$), the following formula is applied to update $m_i \in Memory$ which are used as explained before for future values of control parameter F:

$$m_i = \begin{cases} 1.5 * avg(goodF) - 0.5 * m_i & \text{within limits (0,1].} \\ m_i, & \text{out of limits (0,1]} \end{cases}$$

where avg calculate the average of a list of numbers and $goodF$ is a list of sF values which have improved the individual in the population.

If the number of individuals in a generation that improved does not exceed the half of the population ($nsp <= nfp$) the following formula is used to update Memory:

$$m_i = \begin{cases} m_i + 0.2 * (0.5 * avg(goodF + BadF) - 0.8 * m_i) & \text{within limits (0,1].} \\ m_i, & \text{out of limits (0,1],} \end{cases}$$

where $BadF$ is a listing of sF values that could not be improved in the population.

The procedure for updating memory elements that store CR values (M_{CR}) is analogous to that described in this subsection.

3 The problem and results

3.1 Problem definition: Optimization applied to energy distribution

The distribution of energy has many stages from its generation to its distribution for consumption. In smart grids, the distribution of energy is optimized for both consumers and suppliers, so it is highly desirable to find a good balance between demand and supply. This balance can be modeled as an objective function where the input variables to the function have certain restrictions. Due to space issues in this link[3] the reader may find a more detailed description of the objective functions.

3.2 The benchmark

We have evaluated the performance of the AJSO with the IEEE WCCI / GECCO 2020 benchmark "*CEC-C4 Evolutionary Computation in the Energy Domain: Smart Grid Applications*". This benchmark has two testbeds. The first one aims at optimizing energy resources management for day-to-day use in smart grids under uncertain environments. 500 scenarios with a high degree of uncertainty are used. The second testbed is concerned with a two-tier optimization of end-user bidding strategies in local energy markets (LM). These two levels represent a

[3] `http://www.gecad.isep.ipp.pt/ERM-competitions/wp-content/uploads/2019/12/WCCI2020_Guidelines.pdf`

ISBN 978-987-4417-90-9

complex problem where competitive agents at the one level try to maximize their profits by modifying a price, on the other hand, an agent tries to minimize costs. The competition proposes 50,000 as a limit for the total number of evaluations for each testbed.

Table 1 and Table 2 contain statistical values and positions for the classification following the criteria of WCCI 2020 for three algorithms studied, DE (a basic version), SHADE, and AJSO. The value that summarizes the respective performances is represented by $RankingIndex$. The calculation of this value can be found in the Guideline of the WCCI Competition.

Alg	RankingIndex	PAvgFit	PstdFit	PminFit	PmaxFit	PvarFit
DE	**442.64**	329.09	23.38	286.53	376.89	546.70
SHADE	**370.81**	267.03	27.63	217.93	316.31	763.87
AJSO	**311.11**	214.68	24.07	182.50	277.26	579.45

Table 1. Results obtained by DE, SHADE and AJSO in the *TestBed* 1

Alg	RankingIndex	AvgFit	StdFit	VarFit	minFit	maxFit	AvgProfit
DE	**3.03**	3.03	0.07	0.0059	2.86	3.20	-4.98
SHADE	**2.49**	2.49	0.09	0.0084	2.35	2.64	-4.04
AJSO	**2.28**	2.28	0.04	0.0020	2.20	2.35	-3.60

Table 2. Results obtained by DE, SHADE and AJSO in the *TestBed 2*

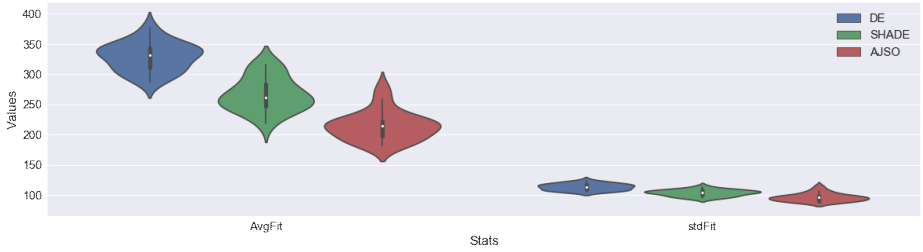

Fig. 1. Comparison between 20 runs of testbed 1 for the three algorithms

Figure 1 and 2 show violin plots to compare the performances of the three algorithms taking into account 20 runs, showing repectively $AvgFit$ and $stdFit$ in the testbed 1 and Fit and $Profit$ in the testbed 2. The figure shows the efficiency of AJSO having a better evaluation and converging much faster than DE and SHADE for the WCCI proficiency test suite. Based on this figure, we

ISBN 978-987-4417-90-9

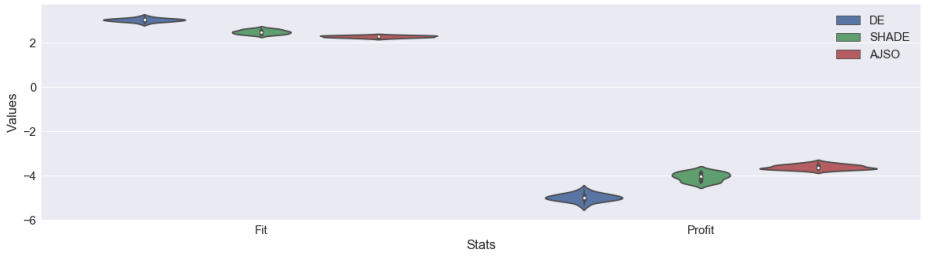

Fig. 2. Comparison between 20 runs of testbed 2 for the three algorithms

observe that AJSO is quicker in finding better values, avoiding the waste of evaluations of the cost function. Therefore, this implies that the exploration phase correctly discards various sectors of search spaces.

4 Conclusions and Future Work

In this article we have proposed AJSO, a new algorithm based on SHADE to solve a Smart Grid application presented at WCCI 2020. AJSO saves the best of each generation (A_g) and makes more efficient use of the Archive to better explore the search space. Both improvements have been introduced as part of the mutation operator and in the updating of the historical memories of the controls parameters. If stagnation is detected, AJSO inserts good solutions in the Archive from A_g structure. In turn, the combination of two mutation strategies allows the algorithm to vary between two abilities, one concerning the amplitude and the other one the depth.

The incorporation of all these characteristics allowed our proposal to obtain the best scores compared to DE and SHADE. In a future work, a deeper analysis can be performed by varying Population and Archive sizes. Currently, AJSO has very few parameters to calibrate, but it may be possible to incorporate values that modify the updating of the memories.

References

1. Brest, J., Maučec, M. S., and Bošković, B. (2019, June). The 100-Digit Challenge: Algorithm jDE100. In 2019 IEEE Congress on Evolutionary Computation (CEC) (pp. 19-26). IEEE.
2. Brest J., Greiner S., Bošković B., Mernik M., Žumer V. (2006). Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems. IEEE Transactions on evolutionary computatio, Vol. 10, No. 6.
3. Zhang, J., and Sanderson, A. C. (2009). JADE: adaptive differential evolution with optional external archive. Evolutionary Computation, IEEE Transactions on, 13(5), (pp. 945-958).

4. R. Tanabe and A. Fukunaga, Success-History Based Parameter Adaptation for Differential Evolution, in IEEE CEC, 2013, pp. 71–78.
5. R. Tanabe and A. S. Fukunaga, "Improving the search performance of SHADE using linear population size reduction," 2014 IEEE Congress on Evolutionary Computation (CEC), Beijing, 2014, pp. 1658-1665, doi: 10.1109/CEC.2014.6900380.
6. Storn, R., and Price, K. (1995). Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces (Vol. 3). Berkeley: ICSI.
7. J. Liu and J. Lampinen, A fuzzy adaptive differential evolution algorithm, Soft Computing—A Fusion of Foundations, Methodologies and Applications, vol. 9, no. 6, pp. 448–462, 2005 [Online]. Available: http://springerlink.metapress.com/index/10.1007/s00500-004-0363-x
8. Brest J., Greiner S., Bošković B., Mernik M. and Žumer V., Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems. IEEE Transactions on evolutionary computation, Vol. 10, no. 6, 2006.
9. J. Brest, M. Sepesy Maučec and B. Bošković, iL-SHADE: Improved L-SHADE algorithm for single objective real-parameter optimization. IEEE CEC, 2016.
10. J. Brest, M. Sepesy Maučec and B. Bošković, Single objective real-parameter optimization: Algorithm jSO. IEEE CEC, 2017.
11. R. Mallipeddi, P. N. Suganthan, Q. K. Pan, and M. F. Tasgetiren,*Differential evolution algorithm with ensemble of parameters andmutation strategies,* Appl. Soft Comput., vol. 11, no. 2, pp. 1679–1696, 2011.
12. J. Brest and M. S. Mau Cec, *Population size reduction for the differential evolution algorithm,* Appl. Intell., Vol: 29, no: 3, pp. 228–247,2008.
13. G. Wu, X. Shen, H. Li,H. Chen,A. Lin and P. N. Suganthan, *Ensemble of differential evolution variants*, Information Sciences Vol: 423 pp: 172-186, 2018.