Synthesis
Differential Evolution and Fractals

Maurice Coquet Mouad Elbouchattaoui
Emmanuel Kahanam Elie Leroy Capucine Nghiem

September 2019

1 Introduction

Evolutionary algorithms are powerful metaheuristic tools for solving complex optimization problems. They are inspired by natural behaviors such as the evolution of the gene pool of a population. These algorithms themselves are constantly improved by hybridizing them, i.e. mixing them in the hope of overcoming their limitations and gaining superior performance. Indeed those algorithms commonly face stagnation, premature convergence problems as well as dependency on a lot of user-defined parameters.

We will study the prospect of a hybrid of two evolutionary algorithms: Differential Evolution and Stochastic Fractal Search. The latter is an upgrade of the Fractal Search, a method used to explore the solution space of the problem in a smart way, following a fractal behavior. Starting from a set of points, considered as particles, other random particles are generated by the mean of a diffusion process. Stochastic Fractal Search was designed by Salimi [9] to speed up the algorithm and avoid some of the original version's disadvantages.

On the other hand, Differential Evolution is a simple yet powerful algorithm which emulates the evolution of a population of points undergoing the process of mutation, crossover and selection. Awad et al. [4, 3] have worked on implementing Differential Evolution for the diffusion in Stochastic Fractal Search.

2 Differential Evolution

2.1 A brief history

Differential Evolution (DE) is a stochastic, population-based algorithm which tries to give a good approximation for the solution of optimization problems with complex, pathological characteristics. DE was introduced by Storn [10]. It is less time consuming than genetic algorithms for different classical optimization problems. DE can overcome some functional pathologies such as non-differentiability, non-continuity, noise as well as the lack of information over the system parameters [5, 10]. There is no theoretic proof of the convergence of DE but the algorithm has been very efficient on a large class of classic optimization problems. For example, Stone and Price have proved in 1997 [5]that the DE algorithm is far more efficient than the simulated annealing and the genetic algorithm. Compared to controlled random search and to other versions of the genetic

algorithm, Ali and Torn [2] discovered in 2004 that the DE algorithm is more accurate and efficient. In the same year, Lampinen and Storn[6] have proved that the algorithm is more accurate than evolutionary programming and some genetic algorithms. DE also suits very well for discrete optimization, nonlinear constraint optimization including penalty functions and optimization of multimodal search spaces[11, 1].

2.2 Description of the algorithm

Differential Evolution is in some respects similar to the genetic algorithm [8]. The similarity consists in being both based on a population subjected to selection/crossover/mutation processes. However, there are two main differences between the genetic algorithm and the differential evolution:

The mutation scheme: All the solutions have equal chances to be selected as parent in DE because no fitness function is considered.

The selection process: The DE algorithms follow a greedy search process i.e the locally optimal choice is made at each step.

We will describe formally the main steps of the DE algorithm. Suppose we have an optimization problem with D real parameters. Let N be the size of the selected population. Given G, the generation number, the parameter vector is denoted as:

$$x_{i,G} = (x_{1,i,G}, \ldots, x_{D,i,G}), \text{ for } i = 1, 2, \ldots, N$$

Initialization: The parameter vectors are supposed to be bounded. The initialization is thus made by uniformly generating a value for each parameter components. Formally, if for the j-th component $x_j^L \leq x_{j,i,1} \leq x_j^U$ then a value is randomly drawn in $\mathcal{U}([x_j^L, x_j^U])$.

Mutation: Given a parameter vector $x_{i,G}$, we select randomly three other vectors $x_{r_1,G}$, $x_{r_2,G}$, $x_{r_3,G}$ all different from $x_{i,G}$. We then compute the donor vector:

$$v_{i,G+1} = x_{r_1,G} + F(x_{r_2,G} - x_{r_3,G})$$

$F \in [0, 2]$ is called the mutation factor or the weight factor.

Crossover: In this step we construct a trial vector $u_{i,G+1}$. The elements of the donor vector enter the trial vector with a probability CR in the following way: for the j-th component, we draw $rand_j$ in $\mathcal{U}([0, 1])$. If $rand_j \leq CR$ then $u_{j,i,G+1} = v_{j,i,G+1}$, otherwise $u_{j,i,G+1} = x_{j,i,G}$. To make sure to have some crossover i.e $x_{i,G} \neq u_{i,G}$, one component is firstly selected randomly to be from v_i.

Selection: After the crossover comes the selection step which is based on choosing best solution candidates. Indeed, the trial vector $u_{i,G+1}$ replaces $x_{i,G}$ if it has better fitness, otherwise $x_{i,G}$ is kept for the next generation $G + 1$.

The algorithm repeats this process until a stopping condition is reached.

3 Stochastic Fractal Search

3.1 Fractal search

The Fractal Search algorithm explores the solution space using the fractal property of a diffusion process [9]. At its beginning, a seed x_0 is picked in the solution space to start a process of diffusion known as the Diffusion Limited Aggregation (DLA) [13].

At a given state of the algorithm, with a set $\mathcal{F}$ of points in the solution space belonging to the fractal, new candidates are picked in the neighborhood of a random point $x \in \mathcal{F}$ chosen with a probability w_x weighted by its number of offspring and its fitness. This weighting ensures that the algorithm keeps exploring new solutions and doesn't get stuck in a single branch, loosing its fractal behavior, while growing in the general direction of a minimum of the loss function.

The generation of new candidates from a parent solution is done through a random process, such as a Gaussian walk or a Levy flight. Out of all the generated candidates, only a fraction p, selected as the fittest regarding their loss value, are added to $\mathcal{F}$ to be considered in the following iterations.

The weights of the solutions in $\mathcal{F}$ can be seen as an electrical potential. When a solution $x \in \mathcal{F}$ is used to create a set of new solutions $Y_x = \{y_1, ..., y_q\}$, its potential is split among them giving the highest weight to the fittest. One example of such a weighting consists in expressing the potential of y_j as $w_x^{y_j} = \frac{f(y_j)}{f(x)+\sum_{k=1}^{q} f(y_k)}$.

3.2 Stochastic fractal search

The Stochastic Fractal Search algorithm (SFS) was introduced in 2014 by Hamid Salimi to overcome the shortcomings of the FS algorithm [7]. In fact, the FS algorithm requires many parameters to be carefully selected in order to avoid being trapped in local solutions. Also, the SF algorithm is very time consuming due to the absence of information exchange between individuals within solution groups. SFS deals with these disadvantages by suggesting two processes: the diffusion process and the updating process. The SFS algorithm steps are presented below:

Initialization: Create a set P of N vectors chosen from the solutions space. where the initialization equation of the j-th coordinate of the i-th vector is :

$$X_{i,j} = LB_j + \epsilon(UB_j - LB_j)$$

Where ϵ is uniformly drawn from $[0, 1]$; UB and LB are the upper bound and lower bound of the solutions space respectively.The fitness function is computed for each vector to get the best solution named X_{best}.

Diffusion: This process is a zone search technique based on the natural phenomenon of growth. The algorithm explores the solutions space by using the diffusion property of fractals: the i-th vector X_i from the set P will diffuse into N_i new vectors by using one of the following Gaussian walk models:

$$GW_1 = Gaussian(X_{best}, \sigma) + (\epsilon X + \epsilon' X_i)$$

$$GW_2 = Gaussian(X_i, \sigma)$$

The σ in the Gaussian distribution is given by:

$$\sigma = |\frac{\log(g)}{g}(X_i - X_{best})|$$

Where g is the number of iterations. The quality of generated vectors is measured by using a fitness function. A comparison based on the lowest fitness function value between the new generated vectors and the seed is executed to keep the better solution.

Updating: The vectors are ranked based on their fitness function value so that the best solution has the N-th rank and the worst the first rank. A ratio is computed for each vector based on the following equation:

$$Pa_i = \frac{rank(X_i)}{N}$$

This ratio is used to increase the chance of changing the position of vectors which have not obtained a good solution. It also increases the chances of good solutions passing to next generations. If a solution's ratio is lower than a random number ϵ drawn from uniform distribution on $[0, 1]$, it is updated using the equation:

$$X_i^{'} = X_{r1} - \epsilon(X_{r2} - X_i)$$

Where X_{r1} and X_{r2} are randomly selected from P. The $X_i^{'}$ vectors are ranked like before and the ratio is computed for each vector. If a vector's ratio is lower than a random number $\epsilon^{'}$ drawn from uniform distribution $[0, 1]$, it is updated using the equations:

$$X_i^{''} = X_d - \beta(X_{r3} - X_{best})\ if\ \epsilon^{'} \leq 0.5$$

$$X_i^{''} = X_d + \beta(X_{r3} - X_{r4})\ if\ \epsilon^{'} > 0.5$$

Were X_{r3} and X_{r4} are randomly selected from P. β is drawn from a normal distribution. $X_i^{''}$ is replaced by $X_i^{'}$ if its fitness function value is better than $X_i^{'}$.

3.3 Differential Evolution and Stochastic Fractal Search

The introduction of Differential Evolution in Stochastic Fractal Search method has been presented in [4] to combine information from all the initial members of the solution population instead of searching new solutions in the neighborhood of one solution, blind to the others. This proves very effective for many kind of non-convex losses to avoid getting trapped in a sub-optimal region.

The suggested method modifies SFS by introducing the possibility to use DE methods such as Success-History Based Parameter Adaptation for Differential Evolution mutation (SHADE) [12] instead of Gaussian Walk in the diffusion step.

Conclusions

The hybridization of Differential Evolution method and Stochastic Fractal Search may overcome limitations inherent to each method, the risk of getting stuck around a local minimum for SFS and the tedious tuning of parameters in DEs.

References

[1] H. A. Abbass, R. Sarker, and C. Newton, *Pde: a pareto-frontier differential evolution approach for multi-objective optimization problems*, in Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No. 01TH8546), vol. 2, IEEE, 2001, pp. 971–978.

[2] M. M. Ali and A. Törn, *Population set-based global optimization algorithms: some modifications and numerical studies*, Computers & Operations Research, 31 (2004), pp. 1703–1725.

[3] N. H. Awad, M. Z. Ali, P. N. Suganthan, and E. Jaser, *A decremental stochastic fractal differential evolution for global numerical optimization*, Information Sciences, 372 (2016), pp. 470 – 491.

[4] N. H. Awad, M. Z. Ali, P. N. Suganthan, and E. Jaser, *Differential evolution with stochastic fractal search algorithm for global numerical optimization*, in 2016 IEEE Congress on Evolutionary Computation (CEC), IEEE, 2016, pp. 3154–3161.

[5] K.Price, *New ideas in optimization*, McGraw-Hill Publishing Company, 1999.

[6] J. Lampinen and R. Storn, *Differential evolution*, in New optimization techniques in engineering, Springer, 2004, pp. 123–166.

[7] L. H. Pham, M. Q. Duong, V.-D. Phan, T. T. Nguyen, and H.-N. Nguyen, *A high-performance stochastic fractal search algorithm for optimal generation dispatch problem*, Energies, 12 (2019), p. 1796.

[8] K. Price, R. M. Storn, and J. A. Lampinen, *Differential evolution: a practical approach to global optimization*, Springer Science & Business Media, 2006.

[9] H. Salimi, *Stochastic fractal search: A powerful metaheuristic algorithm*, Knowledge-Based Systems, 75 (2015), pp. 1 – 18.

[10] R. Storn and K. Price, *Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces*, Journal of global optimization, 11 (1997), pp. 341–359.

[11] M. J. Strens and A. W. Moore, *Policy search using paired comparisons*, Journal of Machine Learning Research, 3 (2002), pp. 921–950.

[12] R. Tanabe and A. Fukunaga, *Success-history based parameter adaptation for differential evolution*, in 2013 IEEE congress on evolutionary computation, IEEE, 2013, pp. 71–78.

[13] T. A. Witten and L. M. Sander, *Diffusion-limited aggregation*, Physical Review B, 27 (1983), p. 5686.