Parameter control for the Plant Propagation Algorithm

Conference Paper · March 2021

CITATIONS
9

READS
275

2 authors:

Wouter Vrielink
University of Amsterdam
3 PUBLICATIONS **36** CITATIONS
SEE PROFILE

Daan van den Berg
VU University UvA University Yamasan
44 PUBLICATIONS **320** CITATIONS
SEE PROFILE

Parameter control for the Plant Propagation Algorithm*

Wouter Vrielink[1][0000−0002−2508−017X]
Daan van den Berg[2][0000−0001−5060−3342]

[1] Informatics Institute
Universiteit van Amsterdam
WLJ.Vrielink@uva.nl
[2] Yamasan Science & Education
daan@yamasan.nl

Abstract. The plant propagation algorithm, a crossoverless population-based metaheuristic, performs significantly better when its fitness function is deterministically adjusted throughout the optimization process.

Keywords: Evolutionary Algorithms · Plant Propagation Algorithm · Metaheuristics · Parameter Control

1 PPA & Parameters

Nearly celebrating its tenth anniversary, the plant propagation algorithm (PPA) is still a relative newcomer in the realm of metaheuristic optimization methods. It revolves around the idea that fitter individuals in its population produce many offspring with small mutations, whereas unfitter individuals produce fewer offspring with larger mutations. Since its inception by Abdellah Salhi and Eric Fraga [9], the paradigm has seen a number of applications [14][11][12][15][4][8], as well as some spinoffs [13][7][10][5]. In this paper, we'll use its seminal form, which iterates through the following routine:

1. Initialize $popSize$ individuals on the problem's domain with a uniform distribution between the bounds.
2. Normalize each individual's objective value $f(x_i)$ to the interval [0,1] as $z(x_i) = \frac{f(x_{max})-f(x_i)}{f(x_{max})-f(x_{min})}$.
3. Assign fitnesses to individuals x_i as $F(x_i) = \frac{1}{2}(tanh(4 \cdot z(x_i) - 2) + 1)$.
4. Assign the number of offspring for each individual x_i as $n(x_i) = \lceil n_{max} F(x_i) r \rceil$, where r is a random value in [0,1) and n_{max} is a parameter determining the number of offspring a population produces.
5. The mutation on dimension j is $(b_j - a_j)d_j(x_i)$, with $d_j(x_i) = 2(r - 0.5)(1 - F(x_i))$, in which r is a random number in [0,1) and b_j and a_j are the upper and lower bounds of the j^{th} dimension. Any individual exceeding a dimension's maximum or minimum bound is corrected back to that bound.
6. The $popSize$ best individuals are selected for the next generation. If the predetermined number of evaluations is not met, go back to 2.

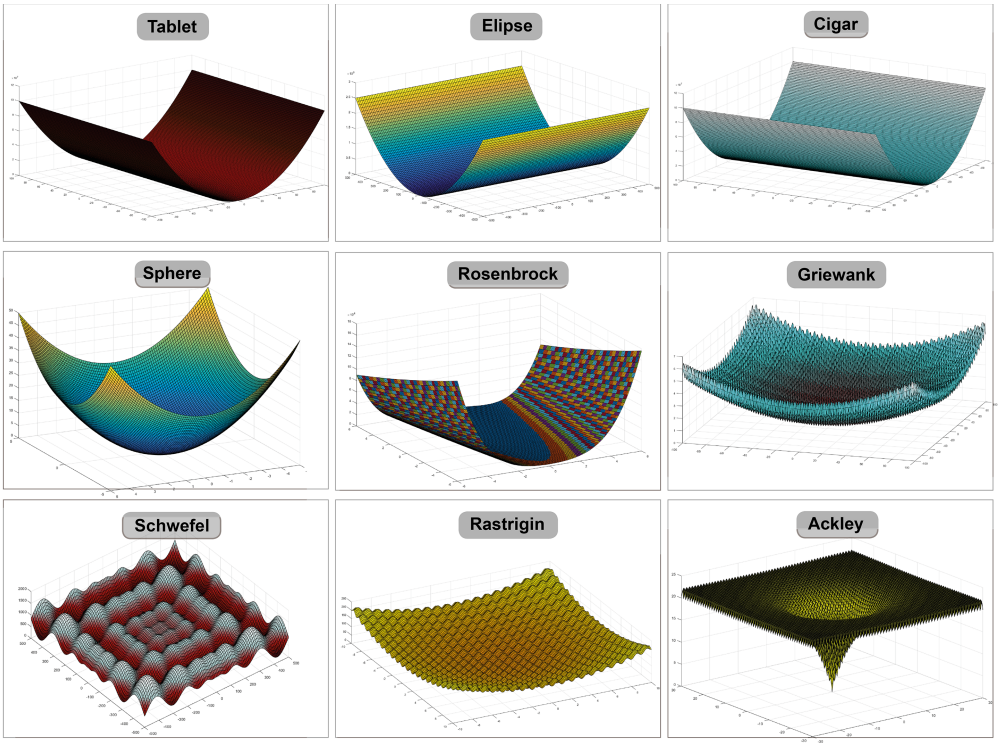

Fig. 1. The nine n-dimensional benchmark test functions used in this experiment. For all functions in this experiment $n = 2$, and our modified PPA algorithm ran for 10,000 function evaluations.

For this preliminary report, we'll slightly adapt point 3 of the algorithm, the nonlinearization step, which assigns fitness to individuals from the normalized objective values, is adapted to

$$F(x_i) = \frac{1}{2}(tanh(4s \cdot z(x_i) - 2s) + 1) \tag{1}$$

in which $s = \frac{evals}{factor} + 1$. By applying equation this change, $F(x_i)$ starts off as a normally shaped sigmoid, but gradually increases its suddenness, and eventually steepens almost to a step function. Two things are important here. First, $factor$ is a constant $\in \{100, 200, ..., 4000\}$ which determines the slowness of the bending process. Second, *evals* is the number of completed *function evaluations*, not the number of generations, so the proposed method does not depend on the parameters *popSize* or n_{max}. Even though the algorithm is largely unsensitive to these parameters in isolation [6][2], any interaction with the modification at hand (Eq. 1) is undocumented as yet.

⋆ This is the first half of a twin submission twin to EvoStar2021, and has an antagonist with identical methods, but different input functions and results [16].

In the Eibenistic classification system, a non-stochastic change in parameter during a run such as our modification qualifies as 'deterministic parameter control' [3], but opinions might differ. Its closest of kin could be the cooling schedule in the simulated annealing algorithm, which is shown to exert a great influence on the quality of its end result [1]. In that case, a usually decreasing parameter 'temperature' ascertains the decline in probability that a deteriorative mutation still gets accepted. Like our s-parameter, the change of that temperature parameter is typically functionally predetermined through the evaluation number, and would thereby also qualify as 'deterministic parameter control'. The difference of course, is that our parameter control affects the relative fitness within the population, and not some acceptance rate.

2 Experiment & Results

We performed a minimization task on nine n-dimensional benchmark test functions, with $n = 2$ (Fig. 1). Besides fixating PPA's default parameterizations ($popSize = 30$ and $n_{max} = 5$) throughout the entire experiment, the possible values for $factor \in \{100, 200, 300...4000\}$ accounted for 40 subexperiments on each of the nine benchmark test functions, 360 in total. For each subexperiment, 10 runs of 10^4 function evaluations was done, and the median end value of a cell's ten runs was recorded (Fig.2).

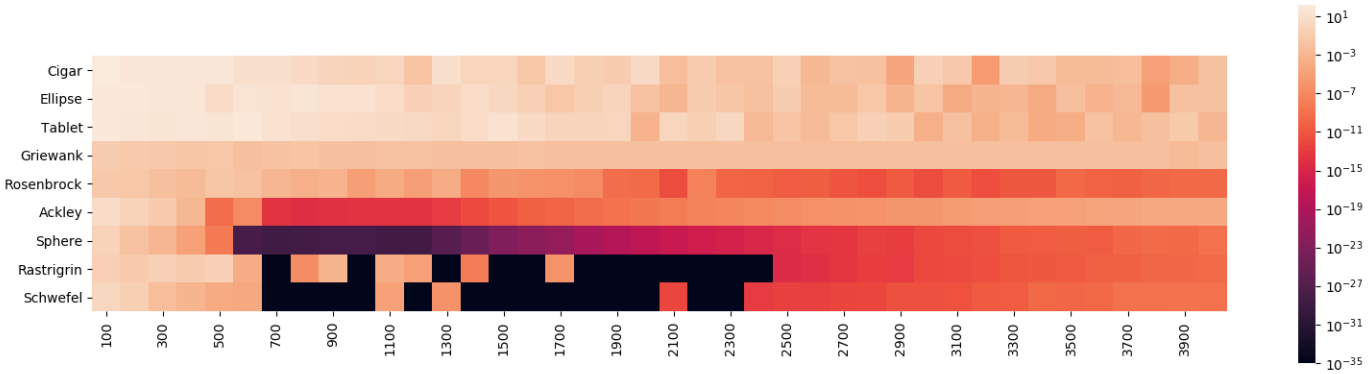

Fig. 2. The plant propagation algorithm can improve a lot from incrementing the steepness of its sigmoidal fitness function. The exact improvements seem to depend on *how slow* (horizontally) the fitness function bends, and differs per function.

Three very distinct patterns can be seen. For Cigar, Ellipse, Tablet, Griewank, Rosenbrock and Ackley, our modification results in mild to medium improvements. For the Sphere function, in many ways the simplest in the test suite,

there is a gentle error decrease leading to an optimum somewhere in the region $600 \leq factor \leq 1200$. For Rastrigin and especially Schwefel, there are steep and deep 'black holes', all located in the range $700 \leq factor \leq 2400$, in which the algorithm actually finds the function's global minimum, greatly outperforming all earlier results [15].

These results show that it's possible, with negligible computational overhead, to significantly improve PPA's performance by just adding a steepening parameter to its fitness function. How robust and widely applicable these results are however, and if further improvements are possible, still remains to be seen.

References

1. Dahmani, R., Boogmans, S., Meijs, A., Van den Berg, D.: Paintings-from-polygons: Simulated annealing. In: ICCC (2020)
2. De Jonge, M., van den Berg, D.: Parameter sensitivity patterns in the plant propagation algorithm. In: IJCCI. p. 92–99 (2020)
3. Eiben, A.E., Smith, J.E.: Introduction to evolutionary computing. Springer (2015)
4. Fraga, E.S.: An example of multi-objective optimization for dynamic processes. Chemical Engineering Transactions **74**, 601–606 (2019)
5. Haddadi, S.: Plant propagation algorithm for nurse rostering. International Journal of Innovative Computing and Applications **11**(4), 204–215 (2020)
6. de Jonge, M., van den Berg, D.: Plant propagation parameterization: Offspring & population size. Evo* 2020 p. 19 (2020)
7. Paauw, M., Van den Berg, D.: Paintings, polygons and plant propagation. In: International Conference on Computational Intelligence in Music, Sound, Art and Design (Part of EvoStar). pp. 84–97. Springer (2019)
8. Rodman, A.D., Fraga, E.S., Gerogiorgis, D.: On the application of a nature-inspired stochastic evolutionary algorithm to constrained multi-objective beer fermentation optimisation. Computers & Chemical Engineering **108**, 448–459 (2018)
9. Salhi, A., Fraga, E.S.: Nature-inspired optimisation approaches and the new plant propagation algorithm (2011)
10. Selamoğlu, B.İ., Salhi, A.: The plant propagation algorithm for discrete optimisation: The case of the travelling salesman problem. In: Nature-inspired computation in engineering, pp. 43–61. Springer (2016)
11. Sleegers, J., van den Berg, D.: Looking for the hardest hamiltonian cycle problem instances. In: IJCCI. p. 40–48 (2020), ECTA2020 Best Paper Award.
12. Sleegers, J., van den Berg, D.: Plant propagation & hard hamiltonian graphs. Evo* 2020 p. 10 (2020)
13. Sulaiman, M., Salhi, A., Fraga, E.S., Mashwani, W.K., Rashidi, M.M.: A novel plant propagation algorithm: modifications and implementation. Science International **28**(1), 201–209 (2016)
14. Sulaiman, M., Salhi, A., Khan, A., Muhammad, S., Khan, W.: On the theoretical analysis of the plant propagation algorithms. Mathematical Problems in Engineering **2018** (2018)
15. Vrielink, W., van den Berg, D.: Fireworks algorithm versus plant propagation algorithm. In: IJCCI. pp. 101–112 (2019)
16. Vrielink, W., van den Berg, D.: A dynamic parameter for the plant propagation algorithm. Evo* 2021 pp. 5–9 (2021)