Evolutionary Computing - Project Report

Pere-Lluís Huguet Cabot 12345466
Ioanna Sanida 10876812
Manon Schriever 10762418
Fiorella Wever 12101745

Abstract

We investigate the different performance levels for the optimization of three functions in case we initiate the populations with different methods. The intuition behind this idea is that advanced initialization techniques can increase the probability of finding global optima, reduce the variation of the final results, decrease the computational costs and improve the solution's quality. In addition to the uniformly random probability distribution, this paper examines the ideas of using a normal distribution centred in 0 and with mean 2.5, adaptive randomness, and opposite selection. We found that uniform random initialization has the best and fastest performance.

1 Introduction

Evolutionary Algorithms are mainly a population-based stochastic search technique, that shares one common algorithmic step, and that is population initialization. The functionality of this aspect is nothing more than providing an initial guess of solutions, which will be iteratively improved until they reach an optimal solution. If the initial guess is bad, then that may prevent us from locating the optima.

Despite the fact that many population initialization techniques have been employed in evolutionary algorithms, there is not enough comprehensive survey on this particular research topic [1]. To address this issue, this paper examines four initialization techniques for the given functions. These are Random Uniform; Gaussian; Opposition-based; and Adaptive Randomness. The evaluation functions that are given are the Bent Cigar function [2], the Katsuura function [3] and the Schaffer function [4]. We investigate if and how these four different initialization methods influence the performance and speed of convergence. We expect the Opposition-based and Adaptive Randomness methods will be slower than the simpler uniform and Gaussian initializations, but will also compensate for this decrease in speed with better results.

2 Research

2.1 Models

Our final EA for the Schaffers and Katuura Function are described in table 1 and 2.
Differential Crossover seemed to be crucial for both functions in order to achieve a good fitness, so both EA are designed around that. Different mutation methods were tested, which did not improve the run much but helped achieve the best fitness faster. In the final version of the algorithms, a population change was implemented in the middle of the run in order to decrease the exploration and increase the number of available generations to increase exploitation, since the population was already close to the maximum. For this paper, we decided to focus on the two Multimodal functions, since they give more insight into the differences between the 4 initialization methods, as they use bigger populations and longer runs.

Representation	List of 10 dimensional double values
Recombination	Differential Crossover
Recombination probability	0.5
Scaling factor	0.7
Mutation	Single Non Uniform
Mutation probability pm	0.9
Parent selection	Tournament
Survival selection	$\mu+\lambda$
Population size	600
Number of offspring	60
Initialization	Random Uniform, Gaussian, Opposite, Adaptive Randomness
Termination condition	Evaluation Limit

Table 1: Technical summary tableau for Schaffers

Representation	List of 10 dimensional double values
Recombination	Differential Crossover
Recombination probability	0.5
Scaling factor	0.8
Mutation	Cauchy mutation
Mutation probability pm	0.3
Parent selection	Tournament
Survival selection	$\mu+\lambda$
Population size	1200
Number of offspring	40
Initialization	Random Uniform, Gaussian, Opposite, Adaptive Randomness
Termination condition	Evaluation Limit

Table 2: Technical summary tableau for Katsuura

2.2 Initialization

With this EA, the different population initializations were tested. **Random Uniform** consists in selecting the values for each allele randomly uniform within the space between -5 and 5. It is usually the method used in most EA, because you want your initial population to be spread along your space since there is no prior knowledge in the structure of your fitness. A **Gaussian** initialization consisted in sampling each allele value from a Gaussian distribution $\mathcal{N}(0, 2.5)$. Intuition says that this is not a good initialization since values in the centre are predominant, and your search will be biased towards it, having fewer individuals in your edges, where there could be the maximum. **Opposition-based** initialization attempts to force the population spread even more. It is one of the most popular multi-step techniques [5]. Our approach differs slightly from the original: half of the population is uniform randomly generated, the other half of the population consists of the opposites of the first half. No further pre-selection is done. **Adaptive Randomness** initialization uses the difference between individuals to make sure they are as spaced out as possible across the population [6]. When looking at a set of trial individuals that could be selected, AR calculates the Euclidean distance between every pair of trial individual and the individuals already in the population. Finally, the AR selects the trial individual that maximizes the minimum distance, to make sure the search space is evenly spread out.

Figure 1 shows what the starting population looks like when initialized with the four different methods.

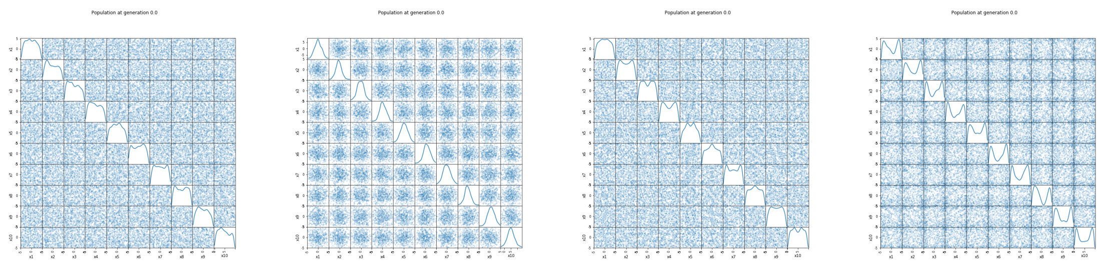

(a) Uniform (b) Gaussian (c) Opposition (d) AR

Figure 1: Starting population with different initializations

3 Results

Figure 2 shows how each initialization ended up after one run of 1600 generations on the Schaffers function [1]. The density plots on the diagonal are scaled to still be visible. All four populations

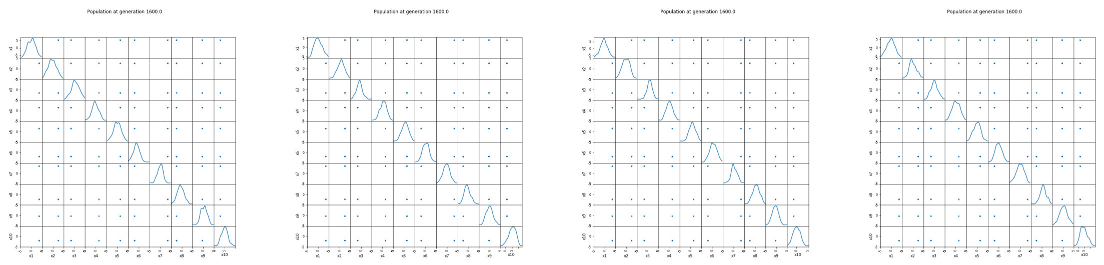

(a) Uniform (b) Gaussian (c) Opposition (d) AR

Figure 2: End population with different initializations

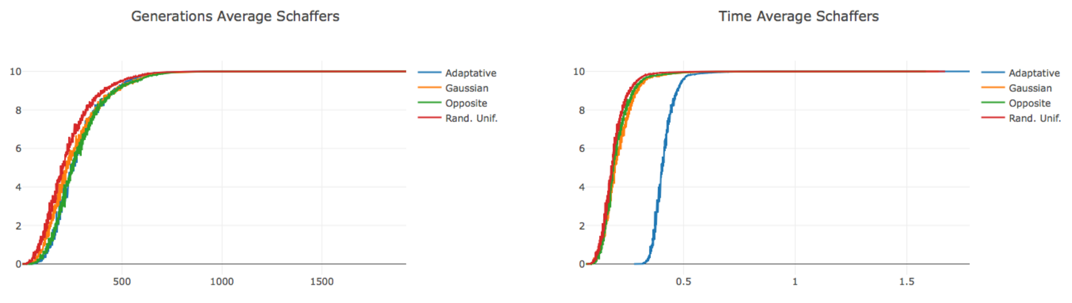
seem to have converged around the same point. The shapes of the density functions do differ between initializations. For example, the Random Uniform initialization (figure 2a) shows two peaks in some dimensions, while the Gaussian (figure 2b) looks much smoother.

Figure 3 also shows this similarity in convergence. All four methods get comparable results in terms of fitness in roughly the same number of generations (figure 3a). However, when considering the wall-time of the algorithms (figure 3b), the Adaptive Randomness performs much slower than the others. All experiments were executed on the same computer.

On the Katsuura function (see figure 4) there is a noticeable difference in fitness. Random Uniform initialization performs best, followed by opposition, Adaptive Randomness and Gaussian. Once again, Adaptive Randomness is the slowest (figure 4b).

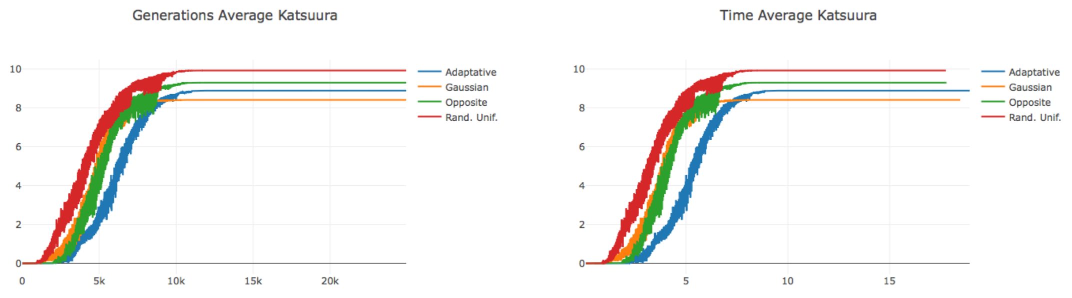
An explanation for the time differences is the difference in computational cost of the initialization (see table 3). Random Uniform, Gaussian and Opposition-based generation methods are within the same time scale, and no difference in both time and computation is worth mentioning. In Adaptive Randomness, there is a search problem to be solved for each individual selected, affecting the computational cost.

[1]See `https://imgur.com/a/yv0uFTr` for gifs showing how the population evolved from start to finish



(a) Fitness by generation (b) Fitness by time

Figure 3: Average highest fitness per generation of ten runs with different seeds on Schaffers



(a) Fitness by generation (b) Fitness by time

Figure 4: Average highest fitness per generation of ten runs with different seeds on Katsuura

In order to check whether the differences in the results were statistically significant and not just due to random chance, we ran an analysis of variance (ANOVA) test. The results were as follows:

- For Schaffer's, there is a statistically significant difference between the time to reach convergence for the different initialization methods, since the P-value for the F-test, ≈ 0.009, is less than 0.05 (even less than 0.01). The results for the maximum fitness scores don't seem significantly different (P-value ≈ 0.057), but that is because they all eventually reach ≈ 10.0 most of the time.

- For Katsuura, there seems to be no statistically significant difference between the time to reach convergence or fitness score for the different initializations, since the $P(> F)$ values for the ANOVA where ≈ 0.056, and ≈ 0.085, respectively. This could be explained by the fact that with Katsuura it is easy to get stuck in a local maximum, and thus there are a

Random Uniform	0.00279 ms
Gaussian	0.0787 ms
Opposition-based	0.00274 ms
Adaptive Randomness	0.00944 ms

Table 3: Average time spend to initialize population over 100 runs

lot of outliers when looking at the distribution of fitness scores (and thus high variance).

- For Bent Cigar, the $P(> F)$ value ≈ 0.041 shows that there is a statistically significant difference between the time to reach convergence for the different initialization methods. This wasn't the case for the maximum fitness scores $(P > F) \approx 0.55$.

4 Conclusion

Our results show that different population initialization can yield different results, and affect both performance and time. In the case of multimodal functions, Random Uniform was the best method, with Gaussian being the worst method (in terms of fitness). Although the literature seems to indicate that smarter methods such as the Opposition-based and Adaptive Randomness should work better, we did not see this result in our experiments. Still, this may be due to the fact that we have no prior knowledge within the space of these functions or any intuition on where the fitness function will result in a better fitness. It is also worth noting that during the tuning of the evolutionary algorithm, uniform initialization was used. This could have positively affected the results of this method. The bad performance of the Gaussian initialization does fit with our expectations.

5 Attachments

Source code: `https://github.com/LittlePea13/evo_comp_43`

References

[1] Shahryar Rahnamayan, Hamid R Tizhoosh, and Magdy MA Salama. A novel population initialization method for accelerating evolutionary algorithms. *Computers & Mathematics with Applications*, 53(10):1605–1614, 2007.

[2] *Bent Cigar Function*, 2015. `http://al-roomi.org/benchmarks/unconstrained/n-dimensions/164-bent-cigar-function`.

[3] *The Katsuura Function. A Continuous Nowhere Differentiable Function*, 2011. `https://caicedoteaching.files.wordpress.com/2012/01/katsuura.pdf`.

[4] *Schaffer N. 1 Function.* `http://benchmarkfcns.xyz/benchmarkfcns/schaffern1fcn.html`.

[5] Borhan Kazimipour, Xiaodong Li, and A Kai Qin. A review of population initialization techniques for evolutionary algorithms. In *Evolutionary Computation (CEC), 2014 IEEE Congress on*, pages 2585–2592. IEEE, 2014.

[6] Weifeng Pan, Kangshun Li, Muchou Wang, Jing Wang, and Bo Jiang. Adaptive randomness: a new population initialization method. *Mathematical Problems in Engineering*, 2014, 2014.