Evolutionary Computing Report
Group 24

Francesco Dal Canton, Joseph Groot Kormelink,
Arvid Lindström, Nil Stolt Ansó

University of Amsterdam

The code used in the experiments can be found at `https://github.com/frank/evo_comp`.

Introduction

Evolutionary Algorithms belong to a highly effective class of algorithms modeled after the process of natural evolution. These algorithms mostly serve the purpose of optimisation: just like in nature, they work by taking instances of a problem's solution, filtering out the bad ones, and having the good ones produce new ones that combine their own features. This research focuses on Differential Evolution, an evolutionary algorithm proposed in 1996 by Storn and Price [1].
Differential Evolution uses the difference – or gradient – between individuals to traverse the genetic landscape. Put simply, it creates a perturbation vector and applies crossover between it and the parent. Since the mutation is always based on the previous generation, the mutation is be directed instead of random, which is an improvement over many other evolutionary algorithms. The Differential Evolution algorithm was chosen for the current research because it has been shown to be effective at continuous function optimisation [1].
In the case of this research, three 10-dimensional continuous functions are used to test the efficacy of the algorithm. These are the Bent Cigar function, the Schaffers function, and the Katsuura function.
The goal of this research is to compare and evaluate different methods for setting the parameters *recombination rate* (CR), *scaling factor* (F) and initialisation of the population. In order to do that, a baseline of fixed parameter values was produced for each of the three functions. Then, alternative methods for parameter setting were defined: initialising the population uniformly, having F be sampled from a Gaussian distribution with increasing μ, and having CR increase linearly over time. This experimental setup allows for the exploration of the effects of different parameter setting methods on the mean best fitness for each of the test functions.

Methods

The Differential Evolution algorithm

Assume a population initialised with individuals satisfying the problem constraints $x \in \mathbb{R}^{10}$ with $x_i \in [-5, 5]$. To specify the DE-algorithm we require the following notation. Let $x_{r1}, x_{r2}, x_{r3}, p_i$ be four unique individuals from the population. x_{r1}, x_{r2}, x_{r3} are randomly chosen, p_i is the i'th individual of the population. v_i is the result of adding a weighted

difference, with scaling factor F of x_{r2} and x_{r3}, to x_{r1} as such: $v_i = x_{r1} + F(x_{r2} - x_{r3})$, referred to as the *donor*. A child c_i is defined as:

$$c_{i,j} = \begin{cases} v_{i,j} \text{ if } \epsilon_j \leq CR \text{ or } j = I_{rand} \\ p_{i,j} \text{ if } \epsilon_j > CR \text{ and } j \neq I_{rand} \end{cases}$$

where index j refers to allele$_j$ in the genome, ϵ is a random variable from a uniform distribution $U[0, 1]$, CR is the recombination rate and I_{rand} is a random integer chosen from $[0, 1, ..., 9]$, preventing the direct copying of a parent to a child in the event of a very low recombination rate.
Using these definitions, the technical specification of differential evolution is given in table 1.

Parent Sel.	Parent p_i is the i'th vector in the population
Mutation	Donor v_i is used for mutation of p_i during recombination
Recomb.	A variation of uniform crossover with a guaranteed crossover point at I_{rand}
Population Size	100
F	$\in \mathbb{R}$
CR	$\in [0, 1]$
Survivor Sel.	If $f(c_i) > f(p_i)$, then parent p_i dies and child c_i enters the population.
	If $f(c_i) \leq f(p_i)$, child c_i dies and p_i re-enters the population.

Table 1: Technical Specification of DE-algorithm

The control conditions

To gather data on the performance of the DE-algorithm on the three functions Bent Cigar, Katsuura, and Schaffers, a cross comparison experiment was conducted. For each optimisation function, a baseline parameter setting was determined through a grid search on all possible combinations of recombination rate $CR \in [0, 1]$ and scaling factor $F \in [0, 1]$. Although there are no restrictions to what values F can take, values larger outside that range gave significantly worse results in preliminary tests. In this search, a search interval for each variable {F, CR} of 0.1 was used, resulting in a total of 100 combinations. The population size of 100 was determined analytically through tuning on all three functions. For a robust comparison across parameter methods for F and CR, the size of the population should remain constant. $p = 100$ was found to yield satisfactory performance on all functions. Finally, the baseline method of population initialisation was a simple uniform-random population initialisation. The grid-search resulted in the following constant values for CR and F, for each function: Katsuura = $\{CR = 0.9, F = 1.0\}$, Schaffers = $\{CR = 0.5, F = 0.3\}$, Bent Cigar = $\{CR = 0.8, F = 0.2\}$. These values resulted in the highest mean best fitness (MBF) for each function and forms the basis for comparison with our methods.

The experimental conditions

For each function, three experimental conditions were tested:

1. Using a uniform initialisation of the population. This method, instead of randomly initialising the population, produced individuals in a uniformly distributed fashion. Using an integer parameter s, it selected s equidistant points across each dimension's range, and produced individuals for all combinations of those values. For $s = 2$, which was the value used for the experiment, the result was $2^{10} = 1024$ individuals, of which the best 100 were kept as the starting population.
2. Using a recombination rate which increases over time. CR would under this condition initialise at 0 and approach 1 bounded by $\frac{E_t}{E_{max}}$ where E_t is the amount of evaluations done at time t. This can be interpreted as increasing mutation (through the donor vector) over time.
3. Using a scaling factor F sampled from a Gaussian distribution that uses standard deviation F_σ and a changing mean, which moves over time from 0 up to F_{end}, which is proportional to $\frac{E_t}{E_{max}}$.

Testing of the three methods for initialising population, adapting F and adapting CR was done one variable at a time. For each function, three experiments were conducted specified as follows:

1. Uniform initialisation: Keep CR and F constant at their baseline values while using the uniform population initialisation method instead of the random (control) initialisation. Record the mean best fitness.
2. Dynamic CR: Keep population initialisation random and set F to the best constant value found in control settings while changing CR over time. Record the mean best fitness.
3. Dynamic F: Keep population initialisation random and set CR to the best constant value found in control settings while changing the mean of the distribution of F over time. Record the mean best fitness.

The parameters F_{end} and F_σ were tuned through a grid search (an F_σ with a value of 0 is equivalent to having a linear increase in F without noise). This was done for each of the four experiments using them – one for Bent Cigar, one for Schaffers, and two for Katsuura, where the last experiment was done in combination with the increasing CR.
All of the code used for the experiments can be found here: `https://github.com/frank/evo_comp`.

Results

Table 2 shows the parameters and results for each experimental setup.
In order to evaluate the effect of each of the experimental setups, we ran unpaired t-tests between the baseline results and the results of each of the other experimental setups.
Since the only two significant improvements occurred when using increasing CR and increasing F on the Katsuura function, it proved interesting to combine those in a last test on the Katsuura function. That meant one last experiment with random population initialisation, increasing CR, and increasing F with $F_{end} = 0.8$ and $F_\sigma = 0.5$. The result was a mean best fitness of 7.885 with $s = 1.971$. A t-test compared to the baseline value gives $t = 27.815$, which is statistically very significant ($<< 0.01$).

		Bent Cigar	**Schaffers**	**Katsuura**
Baseline	Parameters	$CR = 0.8$ $F = 0.2$	$CR = 0.5$ $F = 0.3$	$CR = 0.9$ $F = 1.0$
	MBF	$\bar{x} = 9.999$ $s = 0.001$	$\bar{x} = 10.000$ $s = 0.000$	$\bar{x} = 0.108$ $s = 0.155$
Pop. Uniform	MBF	$\bar{x} = 9.354$ $s = 0.938$	$\bar{x} = 10.000$ $s = 0.000$	$\bar{x} = 0.183$ $s = 0.484$
Increasing CR	MBF	$\bar{x} = 9.996$ $s = 0.003$	$\bar{x} = 10.000$ $s = 0.000$	$\bar{x} = 0.504$ $s = 1.146$
Increasing F	Parameters	$F_{end} = 0.2$ $F_\sigma = 0.4$	$F_{end} = 0.8$ $F_\sigma = 1.0$	$F_{end} = 0.3$ $F_\sigma = 0.8$
	MBF	$\bar{x} = 9.992$ $s = 0.009$	$\bar{x} = 10.000$ $s = 0.000$	$\bar{x} = 0.679$ $s = 1.509$

Table 2: Parameters and mean best fitness with standard deviation for each experimental setup and for each of the three functions. Population size is constant at 100 for all experiments. Increasing variables, unless otherwise specified, go from 0 to 1 during the execution of a search in proportion to $\frac{evals}{evals_{max}}$.

	Bent Cigar	**Schaffers**	**Katsuura**
Pop. Uniform	$t = 4.862$ < 0.01	$t = 0.000$ > 0.01	$t = 1.0435$ > 0.01
Increasing CR	$t = 6.7082$ < 0.01	$t = 0.000$ > 0.01	$t = 2.242$ < 0.01
Increasing F	$t = 0.611$ > 0.01	$t = 0.000$ > 0.01	$t = 2.661$ < 0.01

Table 3: Parameters and mean best fitness with standard deviation for each experimental setup and for each of the three functions. Population size is constant at 100 for all experiments. Increasing variables, unless otherwise specified, go from 0 to 1 during the execution of a search in proportion to $\frac{evals}{evals_{max}}$.

Discussion

One possible explanation on why an increasing CR gives an increase in performance in Katsuura, is that at the beginning the population is going to be scattered all over the local optima of the function. Towards the start of the simulation, it is going to be more effective to pick the genes from the parent not affected by the difference vector, while towards the end of the simulation, when most of the population is located in the same search-space region, picking the genes of the parent affected by the difference vector is going to be more efficient at hill-climbing. The reason this approach might not work on Bent Cigar is that in a unimodal function, considering the parent affected by the difference vector from the very beginning is going to be a more efficient way to hill-climb.
An explanation that might help explain the higher performance in Katsuura of an increasing F is that at the beginning of a simulation, due to the population being scattered across many local optima, the difference vector might come between two individuals in two

different local optima. Using this difference is thus not beneficial early on, and only becomes relevant later in the simulation when most of the population is very close in the search space.
Although not significant, uniform initialisation gave a higher score than random in Katsuura. It could thus be possible that when combined with other methods, the interaction with uniform initialisation could yield a higher score.
It is hard to evaluate how the investigated methods affect the performance of differential evolution in the Schaffers function, as the baseline parameters already give a score of a 10.0 in (almost) every individual test, with only occasional scores below a 10.0 by a 0.00001 margin. A much larger sample size (probably in the order of $n = 10^3$) could help determine the significance of the tested methods in further research.
Despite extensive tuning, differential evolution does not seem to be able to achieve a single 10.0 in the Bent Cigar function. This could be explained by the algorithm not having any truly random mutation. In differential evolution, the mutation is based on the difference between two selected individuals. In some sense, the applied mutation can be thought of as the direction vector towards the top from those two points. We thus theorise that the reason no 10s are observed in Bent Cigar, is because this algorithm converges towards the top, but it is very hard for it to ever reach it.

Conclusion

In short, our results showed that the uniform initialisation wasn't beneficial when applied to our three test functions. The dynamic strategies for setting CR and F, in turn, were only slightly beneficial when applied to Katsuura, and their combination was extremely effective when applied to that same function.
Given that the main difference between the Katsuura function when compared to the Bent Cigar and Schaffer functions is the high degree of multimodality, with the presence of a large number of local optima, a few more research paths now open up.
The first would be to explore the effect of dynamic CR and F, both separately and in combination, when applied to other functions with similar properties to the Katsuura function. This would allow to determine what properties of the functions may be correlated with the improvement in mean best fitness.
The second research path would involve exploring more complex methods for dynamically regulating CR and F. These could involve a number of changes: having their increase not depend on the maximum number of evaluations, thus removing one degree of arbitrariness; when sampling F, not only having a changing mean for its Gaussian distribution, but also a changing standard deviation; also sampling CR from a changing Gaussian distribution, perhaps with a changing standard deviation.

References

1. Rainer Storn and Kenneth Price. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359, Dec 1997.