Evolutionary Computing - Practical Assignment

Vrije Universiteit Amsterdam

Sophie Klumper (2542508), Thomas Weelinck (2622721),
Robbert Hendriks (2538447) & Noah van Grinsven (UvA: 10501916)

October 2018

1 Introduction

In this paper, we will optimize three 10-dimensional functions with a self-designed evolutionary algorithm (EA) and a limited computational budget. The three functions that are optimized are the Bent Cigar, Schaffers F7 and Katsuura function. In trying to find the optimum, we will use basic approaches of initialization and parameter tuning and control.

Our research goal will be twofold. Our first research goal is to develop our own initialization-based memetic differential evolution (DE) instances, that each perform well on their corresponding function via numerical parameter tuning and control. The performance will be based on the mean best fitness (MBF) and the mean moment of convergence (MMOC). Our second research goal is to investigate the effect of the crossover and mutation parameters on the performance of our EA instances.

Our hypothesis for our first goal is that we can get very close to the global optima for all three functions, as we are using three performance improving methods on DE. As these methods are not that complex, we may not reach the optimum itself. For our second goal we think that both crossover and mutation parameters have a significant effect on the performance measures. We think the crossover parameter has a positive influence on the performance when more diverse individuals are created (high values) and the mutation parameter has a positive influence on the performance when the children created do not differ a lot from their parents (low values).

2 Methods

2.1 Differential evolution

An adjusted form of basic DE will be used[1], because we were intrigued by its special mutation operator and it was mentioned as a promising EA in Eiben et al. (2003). In the original problem context a candidate solution (phenotype) is represented by a 10-dimensional real-valued vector. In this setting it is straightforward to use an identity mapping between the phenotypes and genotypes. The predefined fitness function scales the performance of the EA between 0 ('as good as random search') and 10 ('global optimum found'). The population size μ is constant. We slightly altered the parent selection mechanism of basic DE, the necessary vectors are selected by uniform random selection over the N best individuals. The crossover and mutation operator remained the same compared to basic DE and an explanation can be found in Paragraph 6.6 of Eiben et al. (2003). We also make sure that the genotype of the child is in the genotype space $[-5, 5]^{10}$, by letting any allele that violates this constraint bounce back inside. For example, an allele of 7 bounces back to 3. We also slightly altered the survival selection mechanism, so that the EA can escape local optima more easily. The fitness of the child only has to be greater than R times the fitness of the parent to replace the parent, with $0 < R \leq 1$. An overview of the DE algorithm can be found in Table 1. Note that by setting $N = \mu$ and $R = 1$, a basis DE algorithm is obtained.

[1]Code can be found on `https://github.com/Noahprog/Evolutionary_Computing`

Table 1: Overview of the components of DE.

Representation	10-dimensional real-valued vectors
Recombination	Uniform crossover with crossover probability C_r
Mutation	Differential mutation (Rand/1)
Parent selection	Three parents by uniform random selection over the N best individuals
Survival selection	Replace parent with child when $child_{fitness} \geq R \times parent_{fitness}$

2.2 Initialisation procedure and termination condition

A basic DE algorithm initializes each individual randomly in the search space, which in this case creates a great risk of initializing individuals with low or even non-positive fitness values. This could lead to a malfunctioning DE 'not taking off' by recombining low fitness individuals. To get ahead of this problem we could extend our DE to make it memetic on the initialization, by performing a simple local search heuristic to find a better starting point for each individual.

In the first few iterations, the heuristic searches randomly in the search space until an individual with a positive fitness is found. Its location is saved and used to narrow down the search space and the search criteria are altered to only find individuals with a higher fitness. This process is repeated until a 'sufficient' solution is found. These will be cloned and mass mutation will be performed to initialize our population.

Because only a limited computational budget is available, the EA will terminate if K iterations have been performed, with K equal to the number of evaluations available divided by the population size.

3 Performance measures

We define a run of a DE instance as converged if the best found individual does not improve anymore, and the MMOC expresses this as a percentage of the total number of iterations performed. In this way the MMOC can be used to examine if a DE instance has premature convergence or is still improving. We choose the MBF as performance measure because our EA will also eventually be evaluated on this performance measure. It should be noted that we will always refer to the performance of our EA with respect to these performance measures.

The performance measures are averaged over runs, due to the stochastic character of DE. A small experiment is conducted to investigate how many runs are necessary. Multiple runs were executed with fixed parameters for the Bent Cigar Function, and this was repeated for different parameter settings. In Table 2, a result of this experiment is shown from which, without statistical proof, we assumed that 10 runs is enough to find performance measures with sufficient certainty. This decision was also based on the limited time available for this research.

Table 2: Experiment for finding convergence.

Number of runs	1	2	3	...	8	9	10	...	19	20
Best found fitness	7.14	5.5	6.14	...	6.00	6.21	5.47	...	6.16	7.56
Average	7.14	6.32	6.26	...	6.14	6.15	**6.08**	...	**6.00**	**6.07**

4 Parameter Tuning

Because exhaustive search of the parameter space is not practicable, the tuning of the parameters will initially be done by a non-iterative generate and test method. Different parameter vectors will be generated based on grid search and tested. Based on the result, this method or an iterative generate and test method may be implemented to continue the search. For convenience, the parameter N will be expressed as a percentage of μ, from now on referred to as $\%\mu$.

4.1 Bent Cigar

We initialized our parameters based on values found in Pedersen (2010), which states some 'well-known' parameters for the basic DE algorithm. The parameters and corresponding results can be found in Table 3. We decided to set $\%\mu = 100$ and $R = 1$, so it corresponds with a basic DE algorithm for which the parameter values were found. The results in Table 3 were found without making use of our intelligent initialization, but by using randomly created individuals in the search space.

Table 3: Results for initial parameters for Bent Cigar.

	μ	F	C_r	$\%\mu$	R	MBF (STD)	MMOC (STD)
A)	20	0.7	0.8	100	1	9.99995 ($2.7 \cdot 10^{-5}$)	98.45 % (1.22 %)

As the MBF is extremely close to 10, the standard deviation (STD) is extremely small and there is only one local optimum (Bent Cigar is unimodal), we know that in all runs the global optimum was almost found. In addition, there was no premature convergence as the average convergence happened after 98.45 percent of the iterations had been executed. When having a closer look at the improvements of the best individual, there was also a minimal improvement in the last iterations before convergence. Therefore, we were satisfied with the performance of this parameter setting and decided no further tuning was necessary. Next, we investigated the effect of different parameter values, while keeping the values of the other parameters as in Table 3. The results can be found in Table 4.

Table 4: MBF for different values of the crossover probability (C_r) and differential weight (F).

C_r	0.0	...	0.6	0.8	0.9	0.99	1.0					
MBF	9.99...	...	9.99...	9.99...	9.99...	9.99...	6.17					
F	0.2	0.3	0.4	0.5	...	0.8	0.9	1.0	1.1	1.2	1.3	1.4
MBF	3.09	8.64	9.94	9.99...	...	9.99...	9.98	9.88	9.25	7.90	7.78	5.50

Remarkable is that DE is tolerant for almost the whole domain of C_r, the MBF is approximately 9.99 for $C_r \in [0.0; 0.99]$. DE is also tolerant for a big part of the domain of F, the MBF is above 9.9 for $F \in [0.4; 0.9]$. If $C_r = 0$, a child only differs in one variable from its parent and if $C_r = 1$, a child differs in all variables from its parent. The poor performance of $C_r = 1$ may be explained by the fact that there is too much exploration. The poor performance when $F \notin [0.4; 0.9]$ may be explained by the fact that in these cases the population evolves too slow or too fast.

4.2 Schaffers F7

As the initial parameters used for the Bent Cigar function resulted into good performance measures, we decided to use the same initial parameters for the Schaffers F7 function. The parameters and corresponding results can be found in Table 5. Again we decided to set $\%\mu = 100$ and $R = 1$ and used random created individuals as the starting population.

Table 5: Results for initial parameters for Schaffers F7.

	μ	F	C_r	$\%\mu$	R	MBF (STD)	MMOC (STD)
A)	20	0.7	0.8	100	1	9.999996 ($7.8 \cdot 10^{-6}$)	51.79 % (24.48 %)

In 8 out of the 10 runs, the best fitness is equal to 10 and so the global optimum is found. This affects values of the MMOC. If the global optimum is found, the MMOC is between 32 and 50 percent. In the 2 cases the global optimum is not found, the MMOC is equal to 99.2 and 99.8 percent. These two cases increase the MMOC en the STD of the MMOC. In this case, more runs of the algorithm are necessary to make meaningful statements about the MMOC. Again, we are satisfied with the performance of this parameter setting and decided no further tuning was necessary. Again, we investigated the effect of different values of some parameters, while keeping the values of the other parameters as in Table 5. The results can be found in Table 6, where a value between brackets denotes the number of runs in which the global optimum was found.

Table 6: MBF for different values of the crossover probability (C_r) and differential weight (F).

C_r	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.9	0.99	1.0	
MBF	4.68	7.57	9.28	9.92	9.99...	9.99... (2)	10 (10)	9.99... (9)	9.99... (4)	7.39	3.53	
F	0.2	0.3	0.4	0.5	0.6	0.8	0.9	...	1.3	1.4	1.5	1.6
MBF	2.28	7.67	8.54	9.98 (2)	9.99 (5)	9.99... (9)	9.99... (9)	...	9.14	7.78	6.95	5.60

Note that when $C_r = 0.6$, the global optimum was found in all runs. In this setting, DE is very tolerant for the ranges $C_r \in [0.3; 0.9]$ and $F \in [0.5; 1.1]$, as the MBF is above 9.9 in these cases. When C_r and F are only a little outside these intervals, the performance does not decrease dramatically. However, when C_r and F further deviate from these intervals, the performance does decrease drastically.

4.3 Katsuura

An initial grid search is done based on population size μ (250, 750, 1500), differential weight F (0.01, 0.05, 0.1) and crossover probability C_r (0.25, 0.5, 0.75). In addition, this time the DE instance had to make use of our intelligent initialization to 'take off'.

Table 7: Results for different parameters for Katsuura.

	μ	F	C_r	$\%\mu$	R	MBF (STD)	MMOC (STD)
A)	1500	0.05	0.25	25	0.95	6.22 (1.13)	96.2 % (9.6 %)
B)	1750	0.025	0.75	34	0.95	7.95 (1.10)	83.2 % (18.1 %)
C)	2500	0.025	0.75	34	0.95	8.73 (0.44)	96.1 % (6,5 %)

In Table 7 row A, the best result of the grid search is displayed. For fine tuning the parameters, a second grid search is performed. For the parameters μ, F, C_r and $\%\mu$, a smaller interval as well as stepsize is used. Row B in Table 7 displays the best combination of parameter values based on the second grid search. Noticeable is that the performance is higher when μ increases (higher than the values obtained in the first grid search). Subsequently higher population sizes are tested and the results can be found in row C. This results in a MBF of 8.73 and MMOC of 96.1%, which is higher and more stable compared to case B, where the population size is smaller.

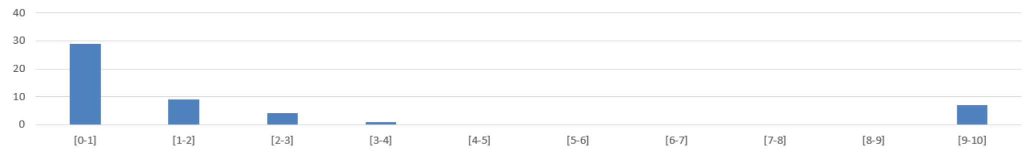
Again, we investigate the effect of different values for some parameters, while keeping the values of the other parameters fixed. For $C_r \in [0.65; 0.85]$, $F \in [0.02; 0.05]$ and $\mu \in [1250; 3000]$, the DE instances are quite stable meaning the MBF does not differ a lot (approximately [7; 8.73]). When the parameters were outside of the previously mentioned intervals, the performance of the DE instance immediately drops.

5 Parameter control

Although the performance of our EA for Katsuura is quite acceptable by using our initialization method and after parameter tuning, there is still room for improvement. Therefore we implemented a simple linear decreasing adaption method for the parameter F, to synchronize it better with the different phases of exploration and exploitation within an EA. The adaption of F in iteration n is given by, $F^n = F - (\sigma * n)$.

Some interesting results were found after a small generate and test procedure for σ with other parameter settings, which led to runs with a best fitness value of 10. The problem was that these settings also led to huge variances between single runs and therefore resulted in a low MBF over multiple runs. The frequency of the best fitness results of an experiment with 50 runs of such a setting can be found in Figure 1.

Figure 1: Frequency of best fitness values on Katsuura ($\mu = 100$, $F = 0.9$, $C_r = 1$, $\%\mu = 100$, $R = 1$, $\sigma = 0.0001$).



This supports the thought mentioned in Eiben et al. (2003) of interpreting academic research as a design problem (focusing on peak performance) instead of as an iterative problem (focusing on average performance). The MBF might not be the best performance measure in this case.

6 Conclusion and discussion

To conclude, three DE instances are found that perform well on their corresponding functions, as we expected. Because the Bent Cigar function is unimodal and the MBF was 9.99, we know the corresponding DE instance finds solutions very close to the global optimum. The DE instance found for the Schaffers F7 function probably even finds the global optimum in every run. For the Katsuura function, a DE instance was found via the initialization method and parameter control that performs well based on peak performance; the global optimum is found in some runs but the MBF is low. For the DE instances corresponding to the Bent Cigar en Schaffers F7 function, we found a high tolerance level with respect to the domain of the crossover probability and differential weight and a threshold close to the global optimum. We did not expect these intervals to be this big, as we expected only crossover parameters that create divers individuals and mutation parameters for which the children do not differ too much from their parents would have a positive effect on the performance. For the Katsuura function, in contrast to the other two functions, the performance of a DE instance immediately drops when the crossover probability and differential weight are only slightly outside the intervals found.

As we already mentioned, our decision to use 10 runs was partially based on time limitations. It is wise to further investigate if the results are comparable when more runs are used. In addition, our decision was based on the Bent Cigar function, however the number of runs necessary may differ between functions. Another idea is to expand the initialization method, by creating more sufficient solutions to which mass mutation can be performed. The current method may not create enough diversity in the initial population. Furthermore, the initialization method may influence the parameter tuning. This interaction should be investigated too.

References

Eiben, A. E., Smith, J. E., et al. (2003). *Introduction to evolutionary computing*, volume 53. Springer.

Pedersen, M. E. H. (2010). Good parameters for differential evolution. *Magnus Erik Hvass Pedersen*, 49.