T2P - Numerical optimization using Optimized Genetic Algorithms

Oloieri Alexandru

December 2, 2019

1 Introduction

This paper presents some optimizations brought to the Genetic Algorithm described in **T2 - Numerical optimization using Genetic Algorithms** paper, and analyzes the effect on the performance of those optimizations, by comparing the results of the two algorithms.

2 Parameters

The parameters of the Genetic Algorithm will have the following values:

- $populationSize = 100$
- $mutationProbability(P_m) = 0.001$
- $crossOverProbability(P_c) = 0.3$
- $k_e = 6$ (best 6 chromosomes will be copied from the current generation to the next one)

3 Optimizations

3.1 The result of the algorithm

Even though elitism is integrated in the selection process, in an iterations all chromosomes can have some bits changed during the mutation process, and that can change their value a lot, and returning in the end the best value throughout all iterations guarantees that the result is the best one found.

3.2 Variable value for generationsCount

In section **4.1** of **T2 - Numerical optimization using Genetic Algorithms** paper, an observation was that in order to get better results with the Genetic Algorithms, the value of *generationsCount* parameter was increased from 1000 to 7500/10000. Most of the times this improves the final result, but also increases the running time of the program, and sometimes the extra generations are for nothing. So an optimization would be to not set a limit for *generationsCount*, but to stop the algorithm when the best value didn't improve in the last *genLimit* generations.

3.3 Converging to the local minima

The value of the best chromosome of all generations can be improved by starting a greedy strategy, like Hill Climbing Best Improvement, from that point. The fact that it will get stuck in a local minima (that sometimes it's also the global minima) is not a problem, as that solution will not be further use for exploration or exploitation, and there is a chance the result will be better.

3.4 Hypermutation

The mutation rate will be increased once to $hypermutationRate = 0.4$ for exactly one generation when the best value did not improve in the last *genLimit* generations. This will force the Genetic Algorithm to explore new points in the function domain, and the final result will not be affected since in the end we return the best value from all generations.

4 Results

dim	generationsCount	best	worst	mean	stDev	time(s)
5	7500	0.00	5.62	1.53	1.68	76.094
10	10000	0.00	8.63	3.86	2.84	197.45
30	10000	11.16	42.19	23.6	7.41	569.203

Figure 1: Rastrigin's function results

dim	genLimit	avg(generationsCount)	best	worst	mean	stDev	time(s)
5	2500	6141.63	0.00	4.92	1.02	1.06	47.705
10	3500	11403.93	0.00	11.3	4.19	2.93	184.029
30	3500	22559.9	8.41	37.51	20.6	7.12	1151.229

Figure 2: Rastrigin's function results - Optimized GA

dim	generationsCount	best	worst	mean	stDev	time(s)
5	7500	0.103	0.51	0.27	0.12	93.81
10	10000	0.106	34.73	2.76	8.53	248.14
30	10000	36.0	727.45	246.18	144.03	723.13

Figure 3: Schwefel's function results

dim	genLimit	avg(generationsCount)	best	worst	mean	stDev	time(s)
5	2500	8790.5	0.103	0.415	0.252	0.102	102.20
10	3500	12428.97	0.312	156.80	29.58	51.79	266.67
30	3500	30989.6	437.9	1578.65	870.82	256.755	1773.59

Figure 4: Schwefel's function results - Optimized GA

dim	generationsCount	best	worst	mean	stDev	time(s)
5	7500	0.007	0.16	0.04	0.034	80.95
10	10000	0.00	0.19	0.07	0.04	212.63
30	10000	0.10	0.25	0.08	0.05	622.93

Figure 5: Griewank's function results

dim	genLimit	avg(generationsCount)	best	worst	mean	stDev	time(s)
5	2500	7942.41	0.00	0.14	0.03	0.024	107.04
10	3500	11133.11	0.019	0.35	0.11	0.06	248.20
30	3500	16575.27	0.00	0.22	0.07	0.06	1086.15

Figure 6: Griewank's function results - Optimized GA

4.1 Interpretation

Even though for some functions the results were a little bit better, the differences are not as big as the ones obtained by changing the values of the suggested parameters. Also, in some cases the performance of the optimized algorithm was much worse compared to the standard GA (both the values and the running time), and I think a reason could be the combination of the optimizations: sometimes the hypermutation together with the termination condition (the algorithm is stopped when the best value didn't improve in the last *genLimit* generations) increase the number of generations, but a lot of the bits of the best candidates are changed when the mutation probability is increased to 0.4, so the quality of the chromosomes decreases.

4.1.1 Rastrigin's function

The results of the optimized algorithm are better for 5 and 30 dimensions, but for 30 dimensions the running time increased a lot, so if time can be traded for better results, the optimizations work for this function.

4.1.2 Schwefel's function

Of all the functions, for this one the optimized algorithm produced the worst results and this confirms the fact that integrating more optimization ideas does not necessarily mean getting better results, as sometimes those optimizations work on their own, but when combined the results are unpredictable. For example, if the only added optimization was running a Hill Climbing from the best point found, then for sure the solution could not be worse, but by also integrating the ones described at sections **3.1** and **3.4**, it can be observed that the results are much worse and the algorithm had a higher running time.

4.1.3 Griewank's function

The obtained results are similar, as the GA already produced close to optimal results, but because of the new termination condition, the running time of the optimized algorithm is higher.

5 Conclusions

In order to get the best possible results, the functions must be analyzed and the algorithms must be run with different values for parameters and different optimizations, as the problems are not the same and require different strategies to be solved. The optimizations and parameters that worked for a specific function with a fixed number of dimensions are not guaranteed to find close to optimal results for all the functions (as seen in the **4 Results** section of this paper).

References

[1] The statement of the assignment: `https://profs.info.uaic.ro/~pmihaela/GA/index.html`

[2] More about GAs in computer science: `https://en.wikipedia.org/wiki/Genetic_algorithm`

[3] Information about functions used in this experiment: `https://www.sfu.ca/~ssurjano/optimization.html`

[4] The github repository of the application used in this experiment: `https://github.com/OloieriAlexandru/Numerical-Optimization-Platform/tree/T2P`