Adaptive Genetic Algorithms

Bita Alexandru

December 3, 2019

1 Introduction

In this paper, we explore the use of alternative mutation strategies as a means of increasing diversity so that the GA can track the optimum of a changing environment.

This paper contrasts two different strategies: the Standard GA using a constant level of mutation and an adaptive mechanism called Triggered Hypermutation, that increases the mutation rate whenever there is a degradation in the performance of the time-averaged best performance.

The study examines each of these strategies in the context of evaluating several functions.

1.1 Motivation

The goal of this project is to develop the Genetic Algorithms (GA) for solving the Sphere, Sum-Squares, Dixon-Price and Rastrigin functions on a total of 30 runs and compare the results between the two stated strategies.

As stated above, two types of Genetic Algorithms (GA) are presented - Standard GA (SGA) and Hyper-mutation GA (HGA).

2 Method

Both GA implementations start with a population of 100 individuals(chromosomes). The SGA individuals are represented binary while the HGA are encoded using Gray code.

Both algorithms work as described in the previous paper with the following exceptions: HGA uses adaptive probabilities of mutation/crossover and performs a First-Improvement Hill Climbing on the fittest individual at the end of each generation.

As inspired by previous studies [1], the following values probabilities of mutation (p_m) and crossover (p_c) apply:

$$p_m = \begin{cases} 0.5\dfrac{f_{max} - f}{f_{max} - \bar{f}}, & \text{if } f > \bar{f} \\ 0.5, & \text{otherwise} \end{cases}$$

$$p_c = \begin{cases} \dfrac{f_{max} - f'}{f_{max} - \bar{f}}, & \text{if } f' > \bar{f} \\ 1, & \text{otherwise} \end{cases}$$

where f is the fitness value of the individual to be mutated and f' is the larger of the fitness values of the individuals to be crossed. The $p_c = 1, f' \leq f$ and $p_m = 0.5, f \leq \bar{f}$ expressions are to prevent crossover and mutation probabilities from exceeding 1.0 for suboptimal solutions.

3 Experiment

A $\bar{Val}$ score is used to determine the fit of each implementation. Both GA implementations end after 1000 generations, the HC runs for a maximum of 1000 iterations.

In statistics, 30 runs is typically considered acceptable for determining the average of algorithms. As a result, a $\bar{Time}$ value is used to determine the average execution time of each implementation.

4 Results

4.1 Sphere

$f(\mathbf{x}) = f(x_1, x_2, ..., x_n) = \sum_{i=1}^{n} x_i^2, x_i \in [-5.12, 5.12]$

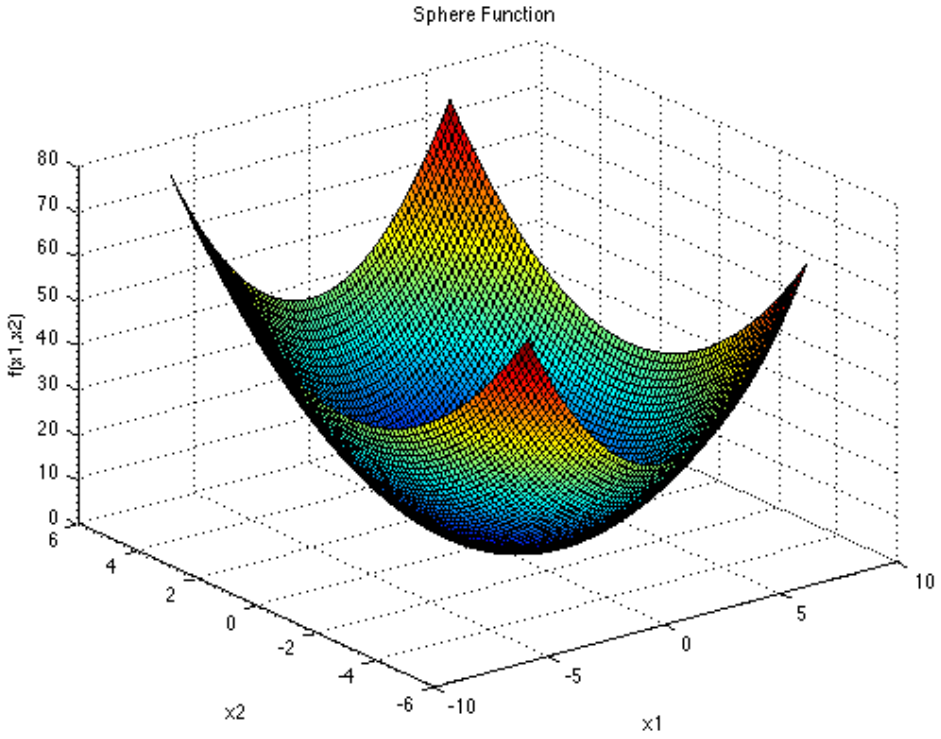

Figure 1: Sphere function [2] for 3 dimensions

SGA	$\bar{Val}$	σ	$\bar{Time}$	**HGA**	$\bar{Val}$	σ	$\bar{Time}$
	0.0000	0.0000	1.0527		0.1052	0.1187	1.0508

Table 1: Results on Sphere function for 2 dimensions

SGA	$\bar{Val}$	σ	$\bar{Time}$	**HGA**	$\bar{Val}$	σ	$\bar{Time}$
	0.3170	0.3971	4.6746		20.8484	4.5271	5.0593

Table 2: Results on Sphere function for 10 dimensions

SGA	$\bar{Val}$	σ	$\bar{Time}$	**HGA**	$\bar{Val}$	σ	$\bar{Time}$
	295.0917	24.3748	13.8355		126.3080	14.0568	17.7432

Table 3: Results on Sphere function for 30 dimensions

4.2 Sum-Squares

$f(\mathbf{x}) = f(x_1,...,x_n) = \sum_{i=1}^{n} ix_i^2, x_i \in [-5.12, 5.12]$

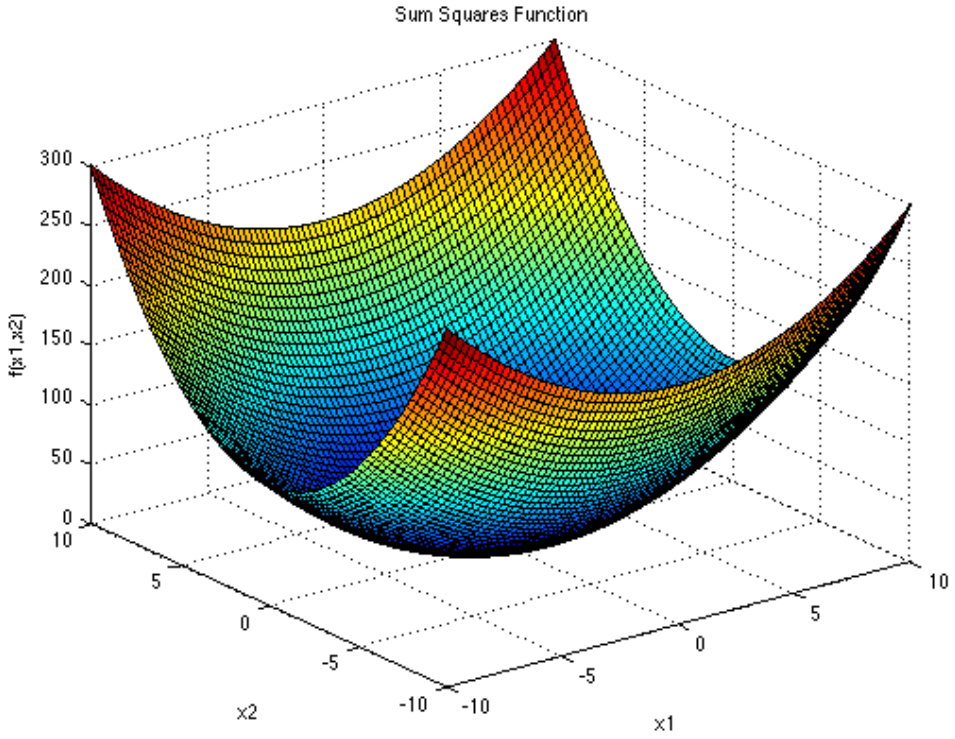

Figure 2: Sum-Squares function [3] for 3 dimensions

SGA	$\bar{Val}$	σ	$\bar{Time}$	**HGA**	$\bar{Val}$	σ	$\bar{Time}$
	0.0000	0.0000	1.0595		0.1409	0.1285	1.0549

Table 4: Results on Sum-Squares function for 2 dimensions

SGA	$\bar{Val}$	σ	$\bar{Time}$	**HGA**	$\bar{Val}$	σ	$\bar{Time}$
	3.1867	9.9838	4.7101		90.9549	28.1496	5.0297

Table 5: Results on Sum-Squares function for 10 dimensions

SGA	$\bar{Val}$	σ	$\bar{Time}$	**HGA**	$\bar{Val}$	σ	$\bar{Time}$
	4342.0479	653.5685	13.8417		1808.1029	208.8380	17.7305

Table 6: Results on Sum-Squares function for 30 dimensions

4.3 Dixon-Price

$f(\mathbf{x}) = f(x_1,...,x_n) = (x_1-1)^2 + \sum_{i=2}^{n} 2(x_i^2 - x_{i-1})^2, x_i \in [-5.12, 5.12]$

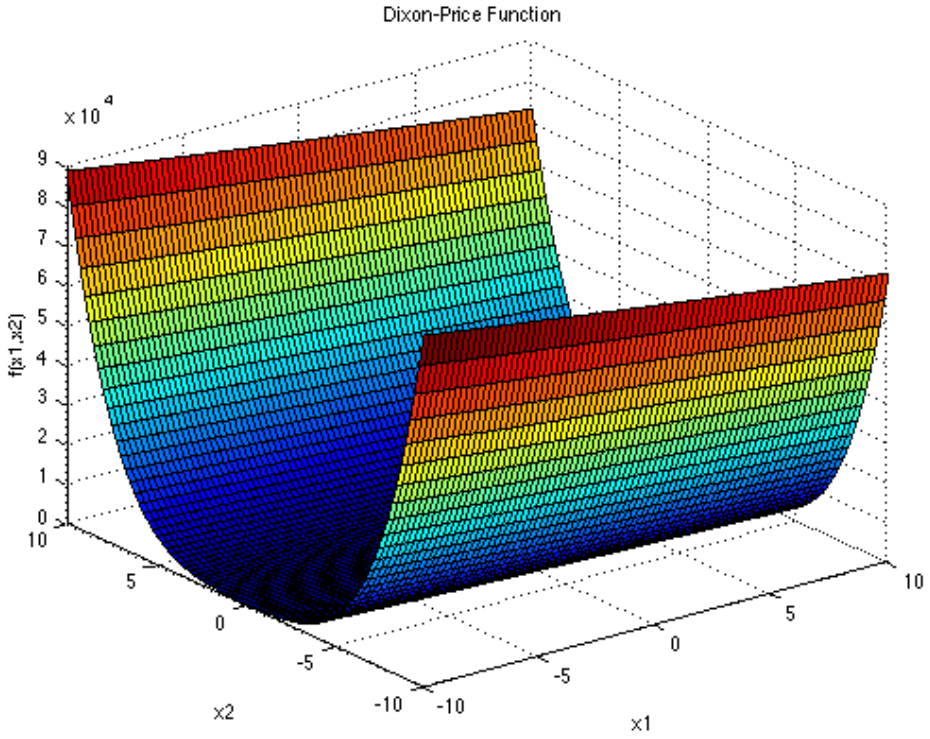

Figure 3: Dixon-Price function [4] for 3 dimensions

SGA	$\bar{Val}$	σ	$\bar{Time}$	**HGA**	$\bar{Val}$	σ	$\bar{Time}$
	0.0220	0.0307	1.1419		0.3676	0.2731	1.1778

Table 7: Results on Dixon-Price function for 2 dimensions

SGA	$\bar{Val}$	σ	$\bar{Time}$	**HGA**	$\bar{Val}$	σ	$\bar{Time}$
	6.4093	18.4152	5.5909		1988.7349	979.7633	6.2190

Table 8: Results on Dixon-Price function for 10 dimensions

SGA	$\bar{Val}$	σ	$\bar{Time}$	**HGA**	$\bar{Val}$	σ	$\bar{Time}$
	246033.7402	66539.8795	13.2411		76297.5198	12942.7652	26.5729

Table 9: Results on Dixon-Price function for 30 dimensions

4.4 Rastrigin

$f(x,y) = 10n + \sum_{i=1}^{n}(x_i^2 - 10cos(2\pi x_i)), x_i \in [-5.12, 5.12]$

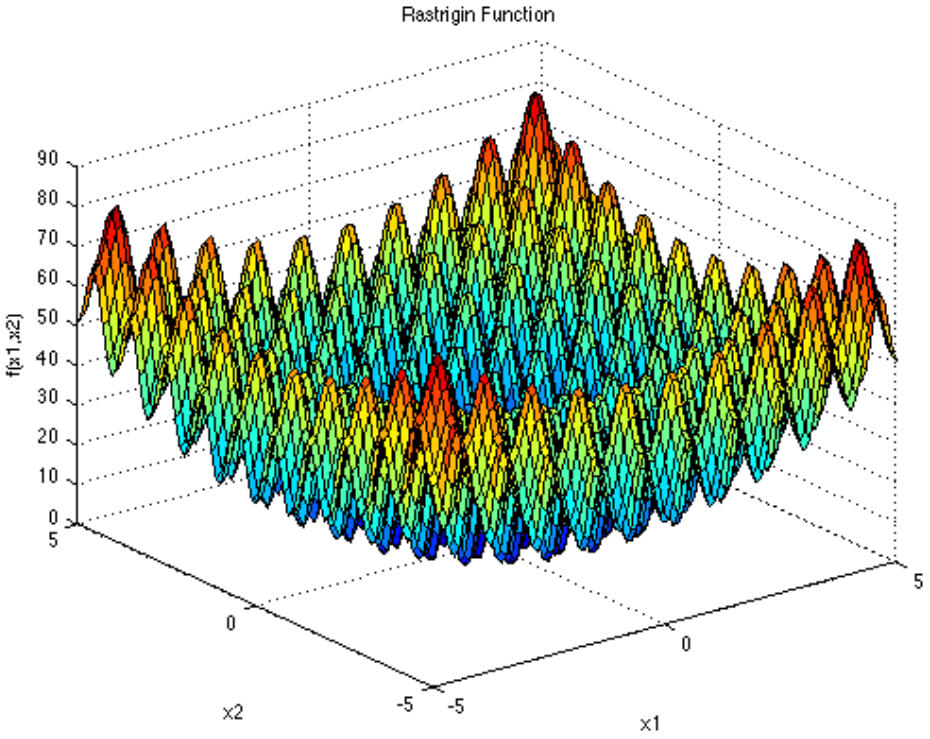

Figure 4: Rastrigin function [5] for 3 dimensions

SGA	$\bar{Val}$	σ	$\bar{Time}$	**HGA**	$\bar{Val}$	σ	$\bar{Time}$
	0.0000	0.0000	1.0844		2.5946	1.4367	1.0689

Table 10: Results on Rastrigin function for 2 dimensions

SGA	$\bar{Val}$	σ	$\bar{Time}$	**HGA**	$\bar{Val}$	σ	$\bar{Time}$
	108.5343	13.4287	4.8346		78.9284	12.7308	5.1453

Table 11: Results on Rastrigin function for 10 dimensions

SGA	$\bar{Val}$	σ	$\bar{Time}$	**HGA**	$\bar{Val}$	σ	$\bar{Time}$
	545.9192	37.0362	14.0657		359.1820	21.8536	18.0593

Table 12: Results on Rastrigin function for 30 dimensions

5 Interpretation

Several experiments have been carried out. Tables [1] to [12] show the comparison in performance of the two stated algorithms.

As shown, while SGA is slightly faster and gives better results overall, HGA achieves better results when running at higher dimensions.

The explanation for this is that HGA adaptively introduces diversity when needed; however, the mechanism sometimes does not perform well in abruptly changing environments (as observed when running at lower dimensions); in other cases, the level of mutation may exceed the amount required.

6 Conclusions

In this paper, we present a different type of genetic algorithm which introduces adaptive parameters and operators.

Compared to a standard GA, this implementation manages to output better results when working on a high number of dimensions at the cost of only a few seconds. However, the standard approach outperforms our proposed algorithm on lower dimensions, which we explain why above.

Our Hypermutation mechanism might be improved by using a different kind of trigger.

References

[1] Fitness-based Adaptive Control of Parameters in Genetic Programming
`https://arxiv.org/ftp/arxiv/papers/1605/1605.01514.pdf`

[2] Sphere function
`https://www.sfu.ca/~ssurjano/spheref.html`

[3] Sum squares function
`https://www.sfu.ca/~ssurjano/sumsqu.html`

[4] Moved axis parallel hyper-ellipsoid function
`https://www.sfu.ca/~ssurjano/dixonpr.html`

[5] Rastrigin's function `https://www.sfu.ca/~ssurjano/rastr.html`