Natural Computing

Assignment

Rodrigo Dominguez (s2450044)

November 16, 2022

General Introduction

The Sphere and Rastrigin functions are

$$f(\vec{x}) := ||\vec{x}||^2$$
$$h(\vec{x}) := 10d + \sum_{i=1}^{d} x_i^2 - 10\cos(2\pi x_i)$$

respectively. In the following figure we can see a plot of them:

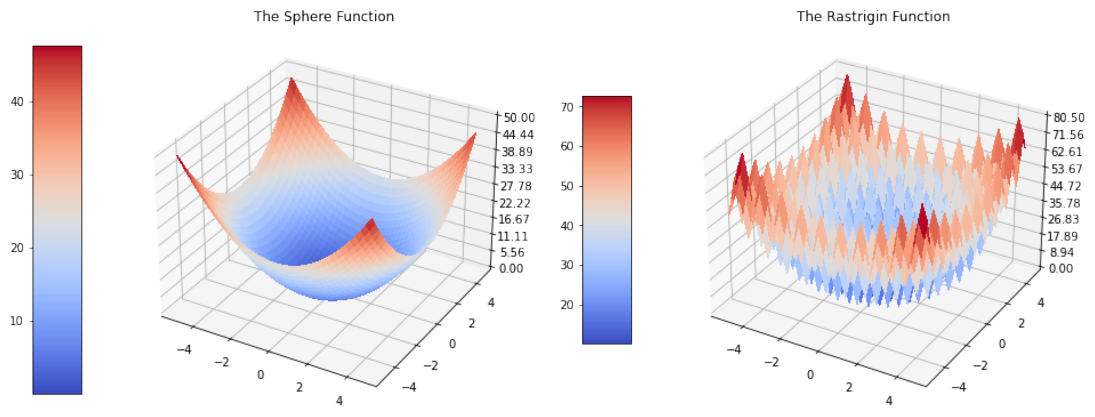


Figure 1: Colour Maps of the Sphere and Rastrigin Functions.

Both of these functions have $\vec{0}$ as global minimum. For the sphere, this is due to the fact that norms are positive-definite in the sense that they are positive unless the argument of the norm is $\vec{0}$. For the Rastrigin function, it suffices to look at the literature (see page 6 of [1]). Notice that $f(\vec{0}) = 0$ and $h(\vec{0}) = 0$.
It is important to realize that the sphere function is the only convex (see Def. 1 in the Appendix) function out of the two. In Figure 1 we can easily see that the Rastrigin function is not convex. This fact will have a lot of weight in how the implemented algorithms throughout the coursework need to be set in order to achieve good performances. We have used the following files:

- Introduction Figures, and Exercises 1 and 2: Intro_EX1_EX2.ipynb
- Exercise 3: heterogeneous swarms.ipynb
- Exercise 4:
- Exercise 5: GP.py

Problem 1: Analysis of Particle Swarm Optimisation

In this first task, we are asked to analyse the PSO algorithm when applied to the minimisation of the Sphere and Rastrigin functions. We start by defining a commom termination criterion. The termination criteria that we give below for stopping our PSO algorithm is based on stopping when the cost value is "sufficiently" close to that of the global optimum:

- For the both functions, we will use the criterion of stopping our iterations when the best fitness along the swarm, f_{best}, satisfies that $f_{best} < \epsilon$, where $\epsilon := 0.00001$ by choice.

As the sphere function is continuous and convex, the above termination criterion will ensure that ending solutions are, not only close to the optimum in cost value, but also in search space location. Nevertheless, for the Rastrigin function (although it is continuous) this criterion will not ensure search space closeness of ending solutions as the function is not convex. Actually, we will see that the convexity of our functions is decisive in how our PSO parameters need to be set in order to achieve a better minimization performance.
In our code we also end the algorithm's run if it has not yet terminated after 1000 iterations or if the particles are most likely diverging (their distances from the center is too large: 10^{18}). Our code distinguishes whether our algorithm ended because of the termination criterion or because our swarm was most likely behaving undesirably.
Now, we try to find the parameters of the PSO algorithm that minimise the average time (as amount of iterations) until termination while also minimising the proportion of runs in which our swarm will end up behaving undesirably (see Explanation 1 in the Appendix). The experiments have as search space $\mathbb{R}^4$ and swarm size 25. We will assume that $\alpha_1 = \alpha_2$, and do our search for $\omega \in \{0.1, 0.2, 0.3, ..., 0.9\}$ and $\alpha_1 \in \{1, 2, 3, 4\}$ (typical in literature).
The results of our experiments are produced as the two tables in figures 6 and 7 in the Appendix. Given that our priority is in minimizing the possible undesired behaviours of our particle swarms, and then having a minimal time (amount of iterations) until termination, we will conclude that the best parameters setting for the sphere case are $\omega = 0.3$ and $\alpha_1, \alpha_2 = 1$ since it is estimated that the probability of an undesired behaviour is 0, and the expected running time is the lowest among the parameters settings with the estimated probability. Similarly, for the Rastrigin function, we choose $\omega = 0.7$ and $\alpha_1, \alpha_2 = 2$ as optimal since is the setting with smaller estimated probability of having an undesired particle behaviour.
It was previously mentioned that the Sphere function is convex while the Rastrigin function is not. This is determinant in how parameter settings of the PSO algorithm perform in the minimization of them. The reason is that for convex functions we will typically want our algorithm to behave more like a gradient descent algorithm by moderately increasing (exceptions given by too large steps or steps producing swarm constriction) the attraction of particles to the global best in the swarm (done by α_2) and to their own bests (done by α_1), since going along both of these directions usually make improvements. In order to attain a similar behaviour to gradient descent we, intuitively, also prefer to have lower inertia-like parameter (by ω), as this is typically used to get a more spring-like movement of our particles which makes them explore the areas surrounding them more. As a matter of fact we see in Fig. 8 in the appendix that for $\omega = 0$ and $\alpha_1, \alpha_2 = 1$ we get a better average time, although our swarm becomes less stable.
Nevertheless, for non-convex functions we want a compromise between this exploratory spring-like movement of particles and their tendencies towards the global and own bests. This is observed in the resulting parameters for both functions which were mentioned in the third to last paragraph.

Problem 2: Scaling

Now we investigate how the size of the warm, N, alters the performance of the PSO Algorithm in the minimization of the aforementioned functions. For investigating the effects of this parameter, we start by setting the rest of the parameters to the optimal values that we found in the last exercise. Hence, for the Sphere function we will study the effects of N given that $\omega = 0.3$ and $\alpha_1, \alpha_2 = 1$, while for the Rastrigin function we will do this study given that $\omega = 0.7$ and $\alpha_1, \alpha_2 = 2$. We maintain $\mathbb{R}^4$ as search space.
In our experiments, we still measure the average time (in iterations) to termination, given the termination criterion from the last exercise, and measure the proportion of undesired swarm behaviours (taken over 300 runs each time) change as N varies.

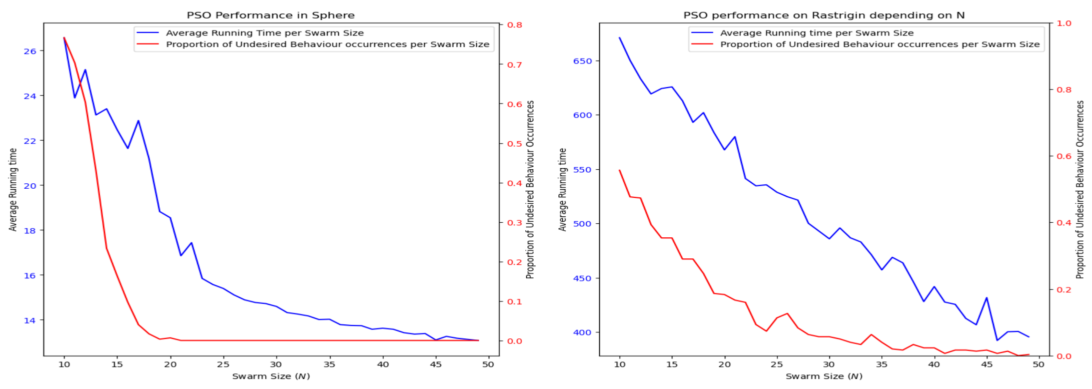


Figure 2: Performance of the PSO Algorithm as N changes.

However, the above experiments do not account for the following: although bigger swarms might take less amounts of iterations to converge, they will surely require more computations than smaller swarms in each iteration. Therefore, we decide to account this possibility in order to make a more educated choice for the parameter of N. We add another measure: the average time in seconds that our runs last until termination.

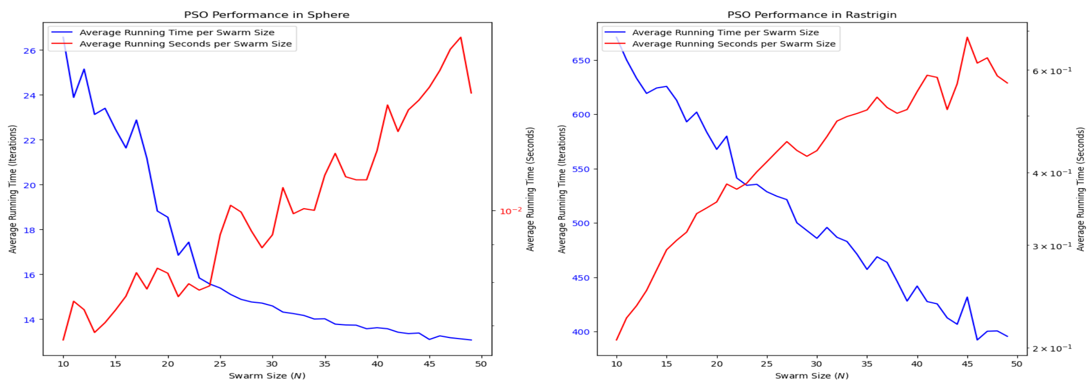


Figure 3: Performance of the PSO Algorithm as N changes.

We see that in both cases as the swarm size increases, both the proportion of undesired events and the average number of iterations until termination decreases. However, in the second round of images, we see that the actual time (in seconds) until our algorithm is terminated increases (notice the logarithmic scale for these measurements). It is noticeable that for the sphere, the number of iterations until termination and the proportion of undesired swarm behaviours decrease more aggressively as N increases. Besides, we observe that for the Rastrigin function the average running time (seconds) increases more aggressively than for the sphere as N increases. Then, the population sizes that are preferable when doing multiple runs of the algorithms are those that are sufficiently large so that the proportion of undesired events is minimal or ”sufficiently” small, but the average running time (seconds) is not too big. That is we want to find the sizes which locate the ”elbows” of our curves. For the Sphere, we would then choose a population size in the range from 20 to 25. Similarly, for the Rastrigin an N between 20 and 30 (approximatelly) would be good. Nevertheless, notice that for the Rastrigin function as the average running times (seconds) are bigger (up to the point of approximately 0.6 seconds), if we were to do a large amount of runs we would maybe prefer to decrease the population size a bit while accepting the corresponding increase of the proportion of undesired swarm behaviours. For instance, we could then choose sizes from 15 to 25.

Problem 3: Heterogeneous Particle Swarm Optimisation

Heterogeneous Particle Swarm Optimisation is the term used to refer to the extension of the Particle Swarm Optimisation Algorithm that uses more than one swarm. In this extension particles only interact (via the velocity update step) with the rest of particles in the same swarm in the sense that the velocity of a particle, v_t in some swarm A, in a step t, is given by:

$$v_t = \omega_A v_{t-1} + \alpha_{A1} r_1 \circ (P_A - x_t) + \alpha_{A2} r_2 \circ (g_A - x_t),$$

where with P_A, g_A, and x_t we are denoting the particle's best position, swarm A's best position, and the particle's current position respectively.
Knowing that effectively Heterogeneous PSO is a PSO algorithm with two swarms moving independently, we choose to grant one of the swarms a more exploratory behaviour while the other one will be given and exploitative tendency. We do this by using the two parameter settings that we obtained from exercise 1 in the two swarms that our population has, where the parameters that resulted in a good performance in the sphere are the ones which give a exploitative behaviour to our swarms (as explained in exercise 1), while those giving good results for the Rastrigin are more explorative (as explained in exercise 1). Furthermore, we will do our experiments for a population sizes of 100 individuals which are equally distributed in both swarms. We choose this size, as the time taken in our experiments is not a big constrain and with this size we completely omit undesired events in our swarms for most of our experiments as we can see in the right hand side image in Fig. 4.

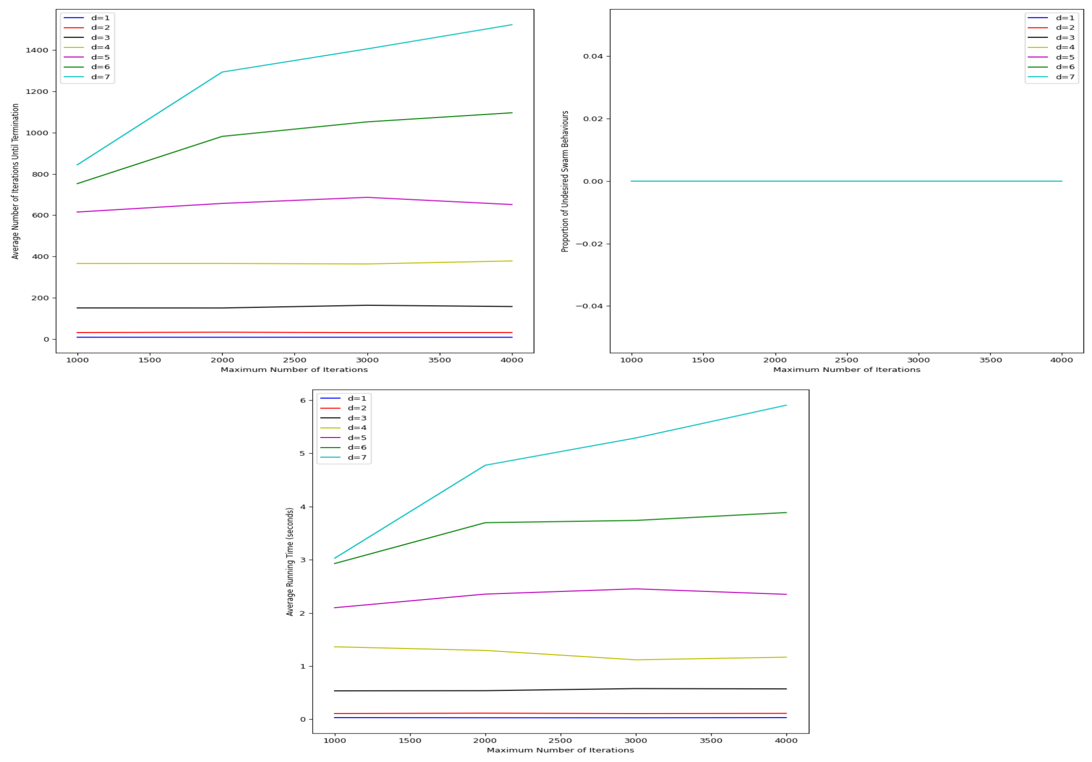


Figure 4: Performance of the Heterogeneous PSO Algorithm as d and the maximum number of fitness evaluations permitted change.

We see that as both the maximum number of iterations (effectively the maximum number of total fitness evaluations) and the dimension of the search space increase, the Heterogeneous PSO algorithm takes, on average, both more iterations, and more seconds to terminate. This is expected, since the number of dimensions of the search space increases the complexity of optimization. Nevertheless, we see that the increase in the maximum number of iterations permitted does not hugely impact our measurements (at least at those scales), since most of the Heterogeneous PSO runs (for all the studied dimensions) take less than 1700 iterations to run.

Problem 4: Differential Evolution

In this section we compare the PSO algorithm that we have seen so far with the Differential Evolution Algorithm, in the task of optimising the Rastrigin Function. We do this by comparing the average running times in both seconds and iterations and the proportion of undesired events for both the DE algorithm and the PSO. The PSO is chosen with the optimal parameter setting for the Rastrigin function that we found in the first problem. The DE algorithm is studied with several combination of parameters where, in this search over the parameter space, the crossover probability p and the population size N are free parameters, while for the constant F is chosen so that the variance of individuals is unchanged as the population evolves. That is, give some F and N, we choose:

$$\sqrt{\frac{1}{N}\left(1-\frac{p}{2}\right)}.$$

In addition, we decide to keep our 4 dimensional search space for this search. Furthermore, following the guidance of [2] on page 3, we choose N to be ten times the number of dimensions. That is, $N = 40$. Therefore, our search will be based on looking over different possibilities for $p \in [0, 1]$. We will look at: $p \in \left\{0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1\right\}$. In the case of Differential Evolution, we defined undesired events as simply having more iterations than the maximum that we have established, 1000 iterations, or if our individuals gets too large. The experience in running the code is that almost all of the times the DE algorithm is terminated because of not satisfying the termination criterion (same as in Problem 1) before the maximum number of iterations. We will see that in the following table next to the other results of our experiments. In the table, as we have done before for the other problems, we measure (over 300 runs) the average running time in seconds and iterations, the proportion of undesired events, and the average best value achieved:

p	Running time (s)	Running time (its)	Prop. of Undesired Events	Best Value achieved
0	1.375625	1000	1	676.896
0.25	1.3884375	1000	1	214.617
0.5	1.3553125	1000	1	65.440
0.75	1.335	1000	1	57.066
1	1.4653125	1000	1	132.887

Table 1: Experiments for DE algorithm.

The best result is then attained with the parameter setting of $p = 0.75$, $N = 40$ and the corresponding F. We see that the DE algorithm reaches its maximum number of iterations even with this setting, producing an ”undesired event”. Furthermore, we can see by looking at the table for Problem 1, and looking at the right image in Figure 3 in Problem 2, that average the running time in seconds and iterations are worse for the DE algorithm than for the PSO. Besides, we see that not only is the PSO algorithm faster, but it also gets better results as it is mostly terminated by meeting the established termination criterion (in Problem 1).
Therefore, although the DE algorithm might be used for an initial search over the search space. To attain better solutions, we will use the PSO algorithm (at least in the Rastrigin function).

Problem 5: Genetic Programming

For this task, we implement Genetic Programming to perform symbolic regression on the 1-dimensional Rastrigin function. This is done, by first sampling data points from the Rastrigin function, which I do by taking 100 points from the domain of the function, $[-5.12, 5.12]$, and the corresponding outputs of the Rastrigin function. Then we define both a set of terminals $\{1, 2, \pi, x\}$ and a set of functions $\{+, -, \cos()\}$ which is, afterwards, fed into the primitive set of our GP algorithm. At last, we define our cost function (in this approach we use a cost function rather than a fitness one) which gives as cost value for a particular program, the mean squared error of the program in the aforementioned data set. The Genetic Programming Algorithm in which our GP implementation (see GP.py) is inspired is the one given by [3] which uses the grow method for initialization, uses point crossover as recombination operator, Subtree Mutation as mutation operator, and tournament selection. Furthermore, we use a population of 100 individuals, we initialize individuals with 3 as depth, and allow a maximum depth of 5 and a maximum number of generations of 5000. In the following figure: we see in the left

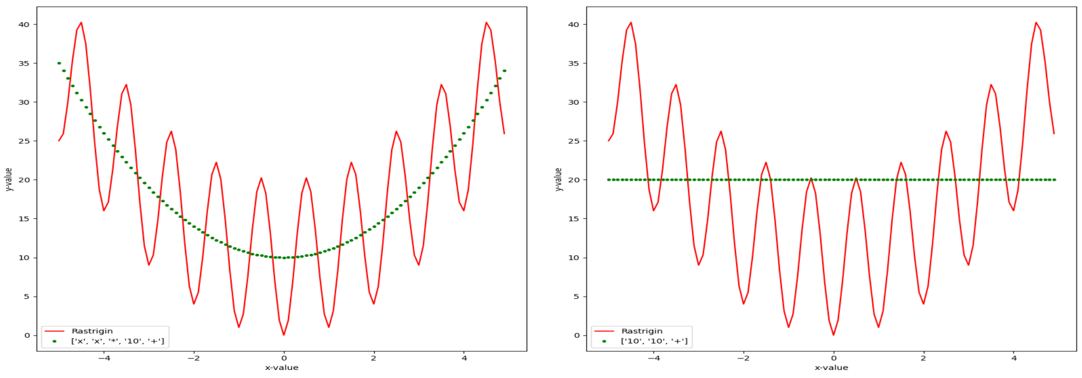


Figure 5: Performance of the PSO Algorithm as N changes.

hand side the best trial that the experiments gave, $x^2 + 10$, while the right hand side was the most common individual that we obtained, 20. Indeed, we see that the left hand side individual seems to fit the Rastrigin in a way that would minimise the mean squared error cost function efficiently.

Appendix

Definition 1. *A function $f : \mathbb{R}^d \to \mathbb{R}$ is convex if for all $\vec{x}, \vec{y} \in \mathbb{R}^d$ and $\theta \in [0,1]$:*

$$f(\theta x + (1-\theta)y) \leq \theta f(x) + (1-\theta)f(y).$$

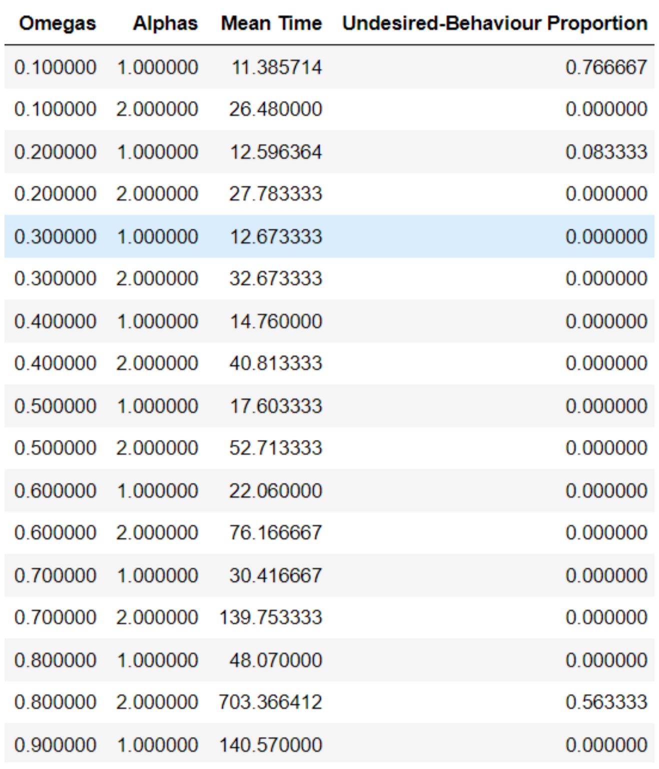
Explanation 1. *For our search we construct one table for each of the functions. In each of them, we have the possible combinations that our ω and α parameters can make. Then, for each of the combinations, we can see the average time (taken over 300 runs) that our algorithm took to get terminated (whenever it managed to satisfy the termination criterion) and the proportion of runs in which our swarm turned out to behave poorly.*

In our table the proportion of runs in which we get an undesirable behaviour of our swarm is a good approximation of the probability of having such a bad behaviour given our parameter setting since:

$$\mathbb{P}(\textit{Undesired Behaviour}) = \mathbb{E}[\mathcal{X}_{(\textit{Undesired Behaviour})}]$$

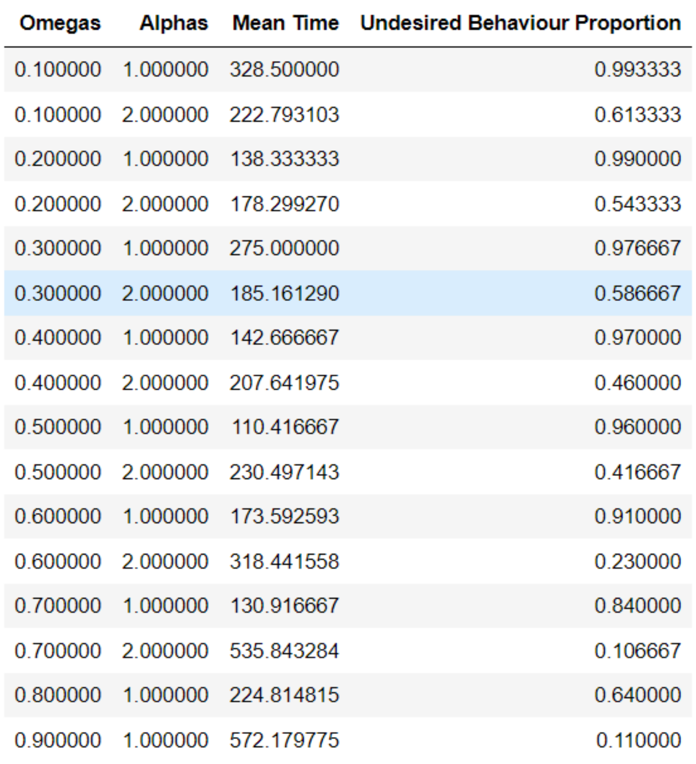
where $\mathcal{X}_{(\textit{Undesired Behaviour})}$ is a random variable which outputs 1 when our run turns into an undesired behaviour and 0 when the behaviour of our run is good. Then (using the fact that this random variable has a finite second moment), we can apply the Law of Large Numbers to say that the proportion that appears in our tables is highly likely to be close to the true probability $\mathbb{P}(\textit{Undesired Behaviour})$ given the setting of our PSO algorithm.

A similar argument works for explaining why the average time until termination is a good approximation of the expected time to termination (here we actually assumed that the absolute value of the time-to-termination random variable has finite expectation to apply the Law of Large Numbers).



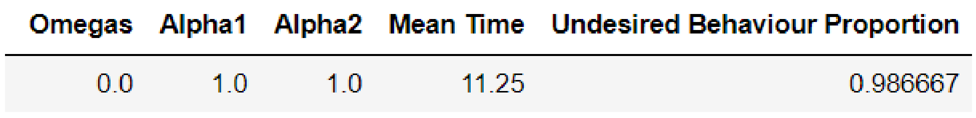
Omegas	Alphas	Mean Time	Undesired-Behaviour Proportion
0.100000	1.000000	11.385714	0.766667
0.100000	2.000000	26.480000	0.000000
0.200000	1.000000	12.596364	0.083333
0.200000	2.000000	27.783333	0.000000
0.300000	1.000000	12.673333	0.000000
0.300000	2.000000	32.673333	0.000000
0.400000	1.000000	14.760000	0.000000
0.400000	2.000000	40.813333	0.000000
0.500000	1.000000	17.603333	0.000000
0.500000	2.000000	52.713333	0.000000
0.600000	1.000000	22.060000	0.000000
0.600000	2.000000	76.166667	0.000000
0.700000	1.000000	30.416667	0.000000
0.700000	2.000000	139.753333	0.000000
0.800000	1.000000	48.070000	0.000000
0.800000	2.000000	703.366412	0.563333
0.900000	1.000000	140.570000	0.000000

Figure 6: Parameter testing for the PSO Algorithm in the minimization of the Sphere function.



Omegas	Alphas	Mean Time	Undesired Behaviour Proportion
0.100000	1.000000	328.500000	0.993333
0.100000	2.000000	222.793103	0.613333
0.200000	1.000000	138.333333	0.990000
0.200000	2.000000	178.299270	0.543333
0.300000	1.000000	275.000000	0.976667
0.300000	2.000000	185.161290	0.586667
0.400000	1.000000	142.666667	0.970000
0.400000	2.000000	207.641975	0.460000
0.500000	1.000000	110.416667	0.960000
0.500000	2.000000	230.497143	0.416667
0.600000	1.000000	173.592593	0.910000
0.600000	2.000000	318.441558	0.230000
0.700000	1.000000	130.916667	0.840000
0.700000	2.000000	535.843284	0.106667
0.800000	1.000000	224.814815	0.640000
0.900000	1.000000	572.179775	0.110000

Figure 7: Parameter testing for the PSO Algorithm in the minimization of the Rastrigin function.



Omegas	Alpha1	Alpha2	Mean Time	Undesired Behaviour Proportion
0.0	1.0	1.0	11.25	0.986667

Figure 8: Performance of the PSO Algorithm in the Sphere for $\omega = 1$ and $\alpha_1, \alpha_2 = 1$.

References

[1] P. BAJPAI and M. Kumar, "Genetic algorithm - an approach to solve global optimization problems," *Indian Journal of Computer Science and Engineering*, vol. 1, pp. 199–206, 10 2010.

[2] R. Storn, "On the usage of differential evolution for function optimization," 07 1996, pp. 519 – 523.

[3] A. Alves, "gp-calc," https://github.com/arturhgca/gp-calc, accessed: 12/11/2022.