

## 0.1 Demo Analysis

Table 1: Parameter settings

Parameter	$\omega_{min}$	$\omega_{max}$	$c_1 = c_2$	steps	clamp_pos	nhood_size
Value	0.7	0.3	1.496	100,000	periodic	5

The present value for  $\omega_{min}$  is set as `PSO_INERTIA` or 0.7. The strategy implemented is `PSO_W_LIN_DEC`, which decreases the inertia weight. The settings control the degree of descent. `nhood_size` denotes the number of informers for each particle, which in this case is 5.

The demo was primarily analysed to examine the degree of premature convergence present in the algorithm itself. The tests were informed by the work on previous examinations of these specific benchmark functions [?]. The program was executed for the purpose of testing using the `test.sh` bash file. Ten runs of each algorithm were performed for data collection. The `output.dat` files obtained were converted to `*.csv` for analysis.

### 0.1.1 Search ranges

Table 2: Search ranges for each function

Function	Range
Ackley	$-32.8, 32.8$
Sphere	$-100, 100$
Rosenbrock	$-2.048, 2.048$
Griewank	$-600, 600$

The Ackley function has a global minimum of  $f = 0$  where  $x = (0, 0, \dots, 0)$  though it has many minor local minima.

### 0.1.2 Analysis

The first element will be to check the error achieved against the number of iterations of the algorithm required to achieve the desired margin of error. The second element of this analysis is to check the desired fitness level against the different levels actually achieved across different dimensions, `dim`. The mean fitness is the figure to look for. Note, mean time is quoted out of 25 runs. The standard deviation is the standard deviation between the mean values for each run, as explained below.

- **Best fitness:** This is the best fitness solution achieved by each benchmark function as represented by the minimum error level across all runs.

- **Mean fitness:** This is the average figure for the fitness achieved over all runs by each benchmark function.
- **Poorest fitness:** The least best fitness level achieved across all runs by each benchmark function.
- **Standard deviation:** As mentioned, this is the standard deviation between the mean levels of fitness achieved in each individual run, aggregated for all runs.
- **Mean time:** The average time taken for each run by each benchmark function.

Table 3: Results for benchmark functions with dimensions  $N = 20$ , size = 18.

Function	Ackley	Sphere	Rosenbrock	Griewank
Best fitness	<b>2.6375E+00</b>	<b>1.2000E+01</b>	<b>0.0000E+00</b>	<b>4.1120E-01</b>
Mean fitness	<b>5.3458E+00</b>	1.2620E+03	<b>7.0764E+02</b>	<b>1.0834E+01</b>
Poorest fitness	<b>1.9595E+01</b>	4.4634E+04	<b>5.3914E+03</b>	<b>4.3180E+02</b>
Standard deviation	<b>1.1635E-01</b>	<b>4.0254E+01</b>	1.5574E+02	<b>1.022E+00</b>
Mean time (ms)	120.40	71.20	108.40	130.80

Table 4: Results for benchmark functions with dimensions  $N = 30$ , size = 20.

Function	Ackley	Sphere	Rosenbrock	Griewank
Best fitness	<b>3.276E+00</b>	2.6000E+01	<b>0.0000E+00</b>	<b>6.573E-01</b>
Mean fitness	<b>5.8233E+00</b>	<b>1.6646E+03</b>	3.1518E+03	<b>1.4016E+01</b>
Poorest fitness	<b>1.9640E+01</b>	7.9735E+04	<b>1.0195E+04</b>	<b>7.1847E+02</b>
Standard deviation	<b>1.7158E-01</b>	<b>1.4613E+02</b>	3.2299E+02	<b>7.6188E-01</b>
Mean time (ms)	270.40	160.00	240.80	290.40

Table 5: Results for benchmark functions with dimensions  $N = 50$ , size = 24.

Function	Ackley	Sphere	Rosenbrock	Griewank
Best fitness	<b>4.0217E+00</b>	<b>7.8000E+01</b>	3.8887E+03	<b>8.9071E-01</b>
Mean fitness	<b>6.931E+00</b>	<b>2.6504E+03</b>	1.4254E+04	<b>1.8769E+01</b>
Poorest fitness	<b>1.9800E+01</b>	1.4625E+05	<b>1.8408E+04</b>	<b>1.245E+03</b>
Standard deviation	<b>2.9957E-01</b>	<b>1.4614E+02</b>	1.9429E+03	<b>1.1489E+00</b>
Mean time (ms)	741.20	434.40	696.40	785.60

Table 6: Results for benchmark functions with dimensions  $N = 70$ , size = 26.

Function	Ackley	Sphere	Rosenbrock	Griewank
Best fitness	<b>4.9272E+00</b>	<b>1.4500E+02</b>	1.9924E+04	<b>9.9511E-01</b>
Mean fitness	<b>7.9936E+00</b>	<b>3.7159E+03</b>	2.4221E+04	<b>2.5144E+01</b>
Poorest fitness	<b>1.9873E+01</b>	2.0908E+05	<b>2.7458E+04</b>	<b>1.8833E+03</b>
Standard deviation	<b>2.8402E-01</b>	<b>2.4331E+02</b>	1.9516E+03	<b>1.2648E+00</b>
Mean time (ms)	1445.20	851.60	1394.80	1538.80

Table 7: Results for benchmark functions with dimensions  $N = 100$ , size = 30.

Function	Ackley	Sphere	Rosenbrock	Griewank
Best fitness	<b>6.0542E+00</b>	<b>4.2000E+02</b>	<b>2.9383E+04</b>	<b>1.1134E+00</b>
Mean fitness	<b>9.4993E+00</b>	6.0538E+03	3.5019E+04	<b>3.2981E+01</b>
Poorest fitness	<b>1.9894E+01</b>	<b>3.0353E+05</b>	<b>4.5821E+04</b>	<b>2.7054E+03</b>
Standard deviation	<b>4.1190E-01</b>	4.3564E+02	3.6994E+03	<b>1.3156E+00</b>
Mean time (ms)	2908.00	1819.60	2831.20	3194.00

Table 8: Results for benchmark functions with dimensions  $N = 500$ , size = 54.

Function	Ackley	Sphere	Rosenbrock	Griewank
Best fitness	1.4433E+01	5.7015E+04	1.9829E+05	4.8773E+01
Mean fitness	1.6645E+01	1.3874E+05	2.1792E+05	3.0480E+02
Poorest fitness	1.9972E+01	1.5935E+06	2.2998E+05	1.4501E+04
Standard deviation	6.1797E-01	2.7717E+04	6.9748E+03	7.1154E+01
Mean time (ms)	147818.00	95558.40	74689.60	134804.80

Table 9: Results for benchmark functions with dimensions  $N = 1000$ , size = 73.

Function	Ackley	Sphere	Rosenbrock	Griewank
Best fitness	1.7250E+01	3.2513E+05	4.3102E+05	7.3685E+02
Mean fitness	1.8593E+01	5.7175E+05	4.5025E+05	1.6362E+03
Poorest fitness	1.9993E+01	3.2470E+06	4.6860E+05	2.9311E+04
Standard deviation	3.2277E-01	1.1247E+05	1.0289E+04	3.2838E+02
Mean time (ms)	560022.80	419196.80	299853.20	590098.00

Figure 1: Standard deviations of the means illustrate the break in consistency with the Rosenbrock benchmark.

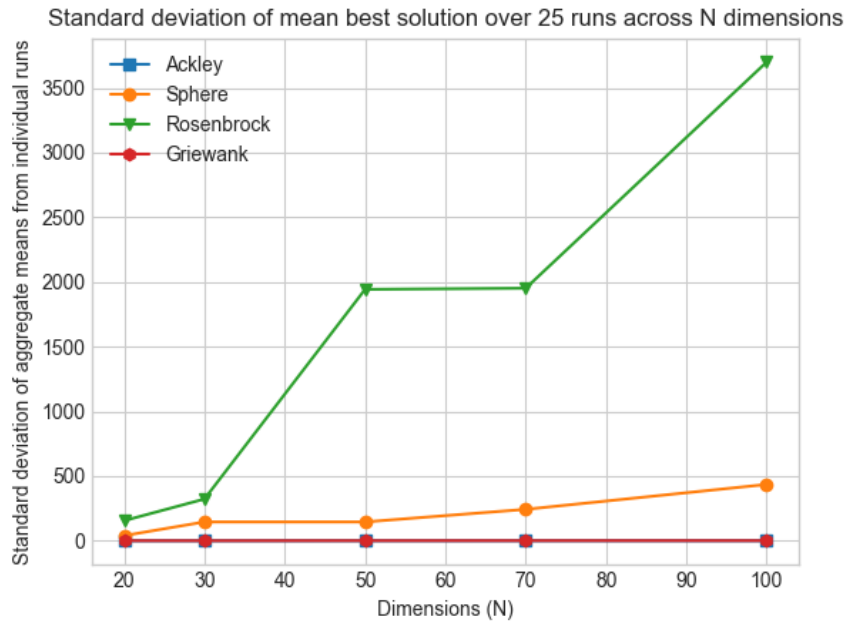
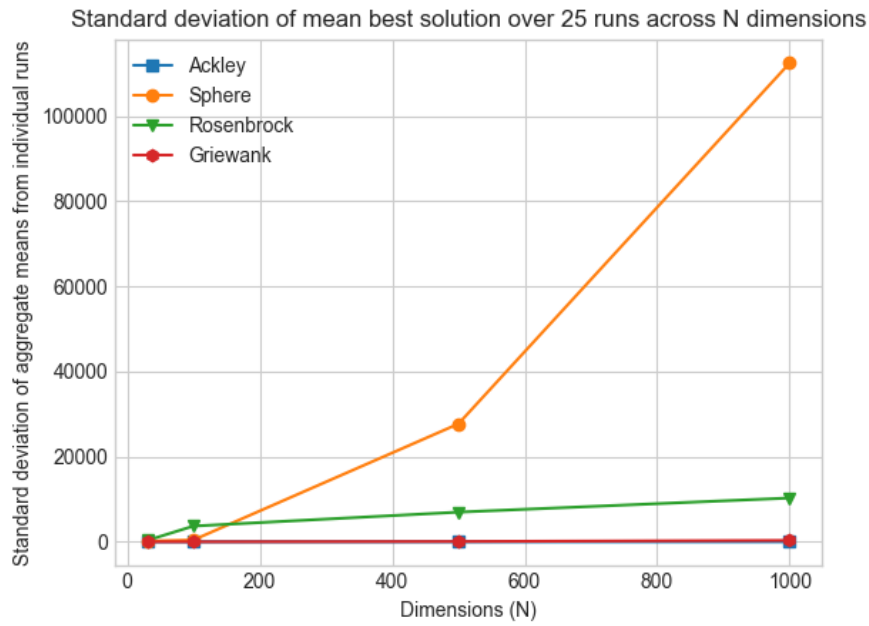
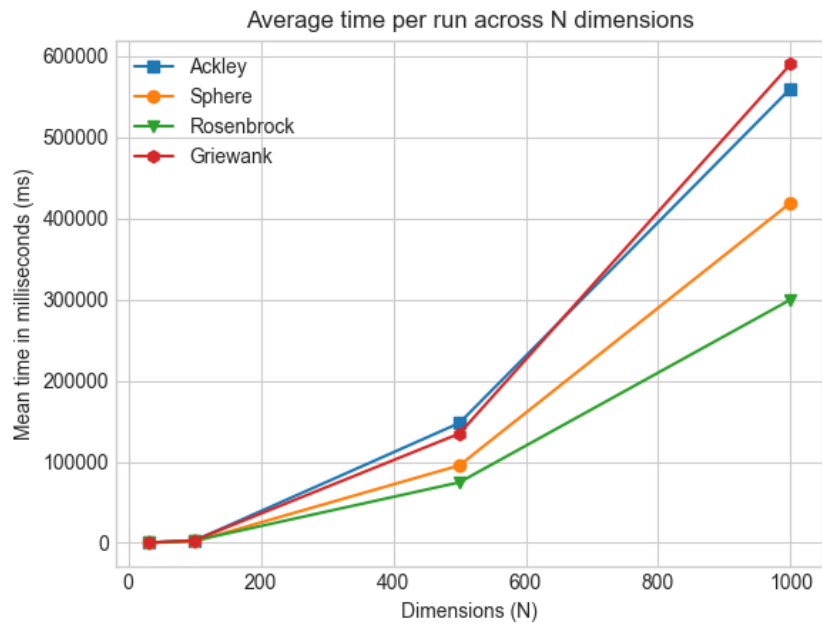


Figure 2: Standard deviations of the means up to  $N = 1000$ .

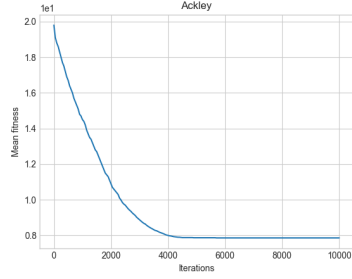


### 0.1.3 Timing Graphs

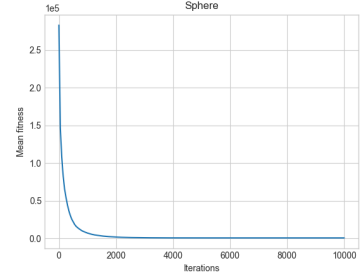
Figure 3: Average time per run for each function.



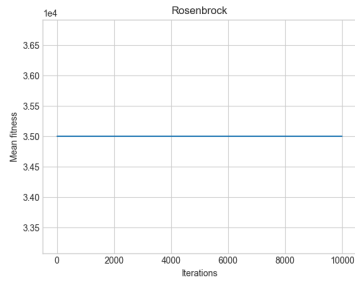
### 0.1.4 Convergence Graphs



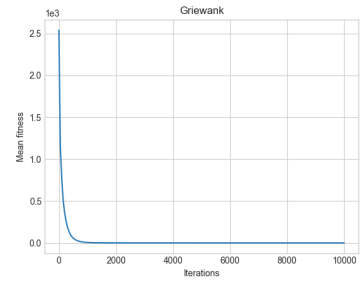
(a) Ackley



(b) Sphere



(c) Rosenbrock



(d) Griewank

Figure 4: Convergence graphs for each function at  $N = 100$  dimensions.

## **0.2 Discussion of observations**

### **0.2.1 Ackley**

In terms of timing, Ackley is fine with 20 and 30 dimensions, but then begins to decay, coming second only to Griewank. That being said, the mean fitness levels achieved are consistently the best out of the benchmark set and even the poorer fitness levels cannot compare with even the better Rosenbrock or Sphere ones in the higher dimensions. The standard deviation between the mean best fitness levels achieved per run (see Figure 1) are also comparatively the lowest.

### **0.2.2 Sphere**

The Sphere function is dependable in lower dimensions and doesn't decay too much in terms of time. This function does however achieve the poorest fitness levels once  $N$  is greater than 30. Whilst the means achieved do not deviate as much as those achieved by the Rosenbrock function, it is nonetheless prone to break in terms of deviations earlier.

### **0.2.3 Rosenbrock**

Rosenbrock is not well behaved in the higher dimensions in spite of its better performance measures at the lower dimensions. A quick median fitness analysis for Rosenbrock at  $N = 30$  is 0.77712, so it converges very quickly in lower dimensions. Time decay takes hold when  $N = 70$ , and so it does not achieve the correct fitness levels afterwards.

### **0.2.4 Griewank**

Griewank decayed the most in terms of timing. Less deviation between the average time of each run for each function due to higher computational load and the sort mechanism in memory is the case here, since the function, whilst able to achieve a consistent fitness level, can also encounter much poorer fitness levels when in the process of evaluating.



### 0.3 Next steps

The next step will be to run these same tests in parallel to see what the performance improvement is by adjusting the number of processors. With the path planning problem, the serial evaluation will look at how increasing the number of particles affects performance, and thus with the parallel evaluation of that application, the same assessment will take place, but with the spread of  $N$  particles across multiple processors. The appendix will be further developed to accommodate more graphical analysis that cannot be included in the main body of the report.

### 0.4 Increment and Problem Dimensionality

```
1 //Calculate swarm size based on dimensionality
2 int pso_calc_swarm_size(int dim) {
3     int size = 10. + 2. * sqrt(dim);
4     return (size > PSO_MAX_SIZE ? PSO_MAX_SIZE : size);
5 }
```

Listing 1: Function for calculating appropriate swarm size

All of the demo runs were performed with swarm sizes that were less than 100, with the highest,  $N = 1000$  containing a swarm size of 73. An automatic calculation of the swarm size was performed by the function shown above, which was used in each case. The classical thinking on the population size is that the best performance typically occurs in populations with smaller sizes, hence the restricted allowance for incrementation with the problem dimensions. However, it has been argued recently that the smaller sizes selected to evaluated different optimisation problems may indeed be too small [?].

The authors of this paper have argued that the best results are typically obtained in the 70-500 size range. This is not a problem for the results obtained above as the range of sizes is incremented sufficiently, and evaluation of the impact of higher dimensions versus the increased population size is beyond the scope of this analysis. Nonetheless, it is evident that the impact of higher dimensions likely outweighs any influence a population size beyond 100 would have, as can be seen by the breakdown in performance of the Rosenbrock benchmark.

### 0.5 Profiling

X

### 0.6 PSO Topologies

X