Report on the optimization of the genetic and particle swarm optimisation algorithms

Genetic algorithm

The genetic algorithm is based on Darwin's theory of evolution. It is a gradual process; the best individuals are successively selected and reproduced until the best generation is reached by using a randomized yet structured information exchange.

The main operators of the genetic algorithms are reproduction, crossover, and mutation.

Reproduction is a process based on the objective function (fitness function) of each string. This objective function identifies how genetically fit a string is. Thus, strings with higher fitness value have bigger probability of contributing offspring to the next generation.

Crossover is a process in which members of the last population are coupled randomly in the mating pool. So, a pair of offspring is created, combining elements from two members, which have improved fitness values.

Mutation is the occasional random alteration of the value of a string position. This process leads to the change in chromosomes for a single individual. Mutation prevents the algorithm from getting stuck at a particular point.

Particle swarm optimization algorithm

PSO is a population-based optimization technique inspired by the motion of bird flocks and schooling fish. The fundamental idea in PSO is that each particle represents a potential solution which it updates according to two important kinds of information available in decision process. The first one (cognitive behaviour) is gained by its own experience, and the second one (social behaviour) is the experience gained from the neighbours, that is, they tried the choices itself and have the knowledge which choices their neighbours have outstand so far and how positive the best pattern of choices was. PSO has been used increasingly due to its several advantages like robustness, efficiency, and simplicity. When compared with other stochastic algorithms it has been found that PSO requires less computational effort.

The algorithm calibration process for scenario 1

Genetic algorithm

The calibration process consisted of testing each parameter, using a value range respective to the parameter being tested, while the other parameters were being set at their base values. Once all the best values for each parameter were found, they were all computed together. Lastly, further optimisation tests were made regarding how the parameters related to each other and adjustments were made accordingly.

Different population to iteration ratio analysis

Five different ratios that achieved the one million calls to the fitness function required for scenario 1 were chosen appropriately.

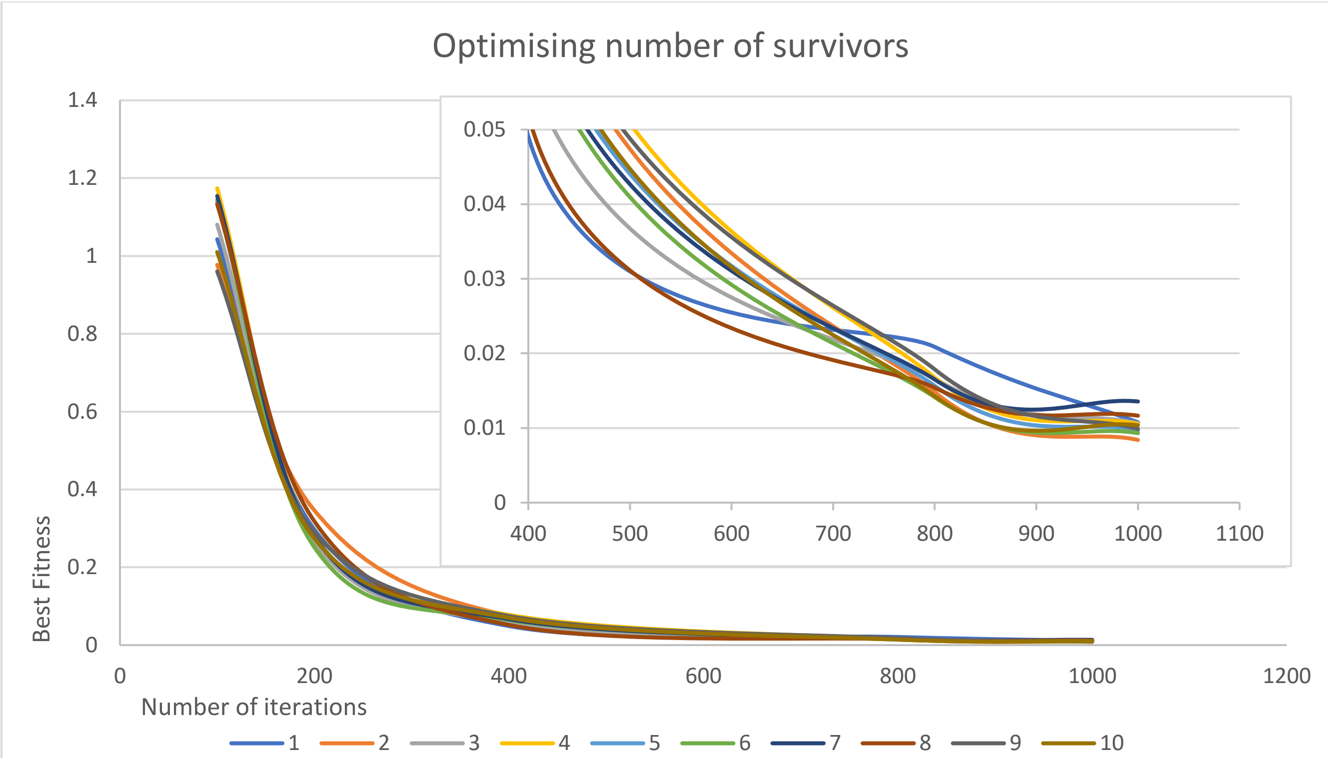
Number of iterations	Population number	Best Fitness
10	100000	15.86015235
100	10000	0.023123956
1000	1000	0.009632744
10000	100	0.009418946
100000	10	0.01150405

The notably smaller Best Fitness value resulted from the 10000 to 100 ratio. As the other parameters' performance was discovered to be wildly dependent on the iteration to population ratio, any further tests were run using the 10000 to 100 ratio, rather than the base one as the calibration method initially suggested.

Calibration of the number of survivors

The numSurvivors parameter was tested using values in a range of 1 through 10, while all other parameters were set at their base values. The value of 2 yielded the best results.

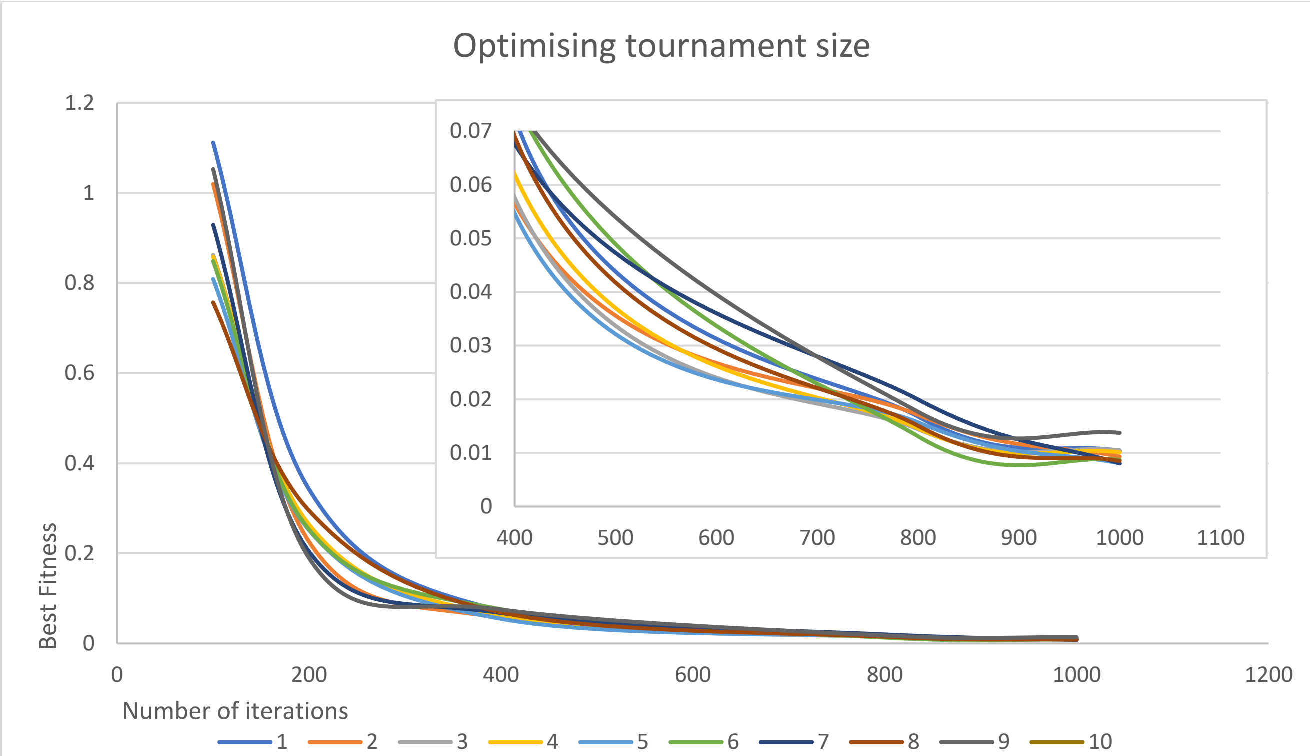
Number of survivors	Best Fitness
1	0.010767459
2	0.008391807
3	0.010651404
4	0.010537638
5	0.009727527
6	0.009311477
7	0.013552844
8	0.011654713
9	0.009853774
10	0.010353932



Calibration of the tournament size

The tournamentSize parameter was tested using values in a range of 2 through 10, while all other parameters were set at their base values. The value of 8 produced the best results.

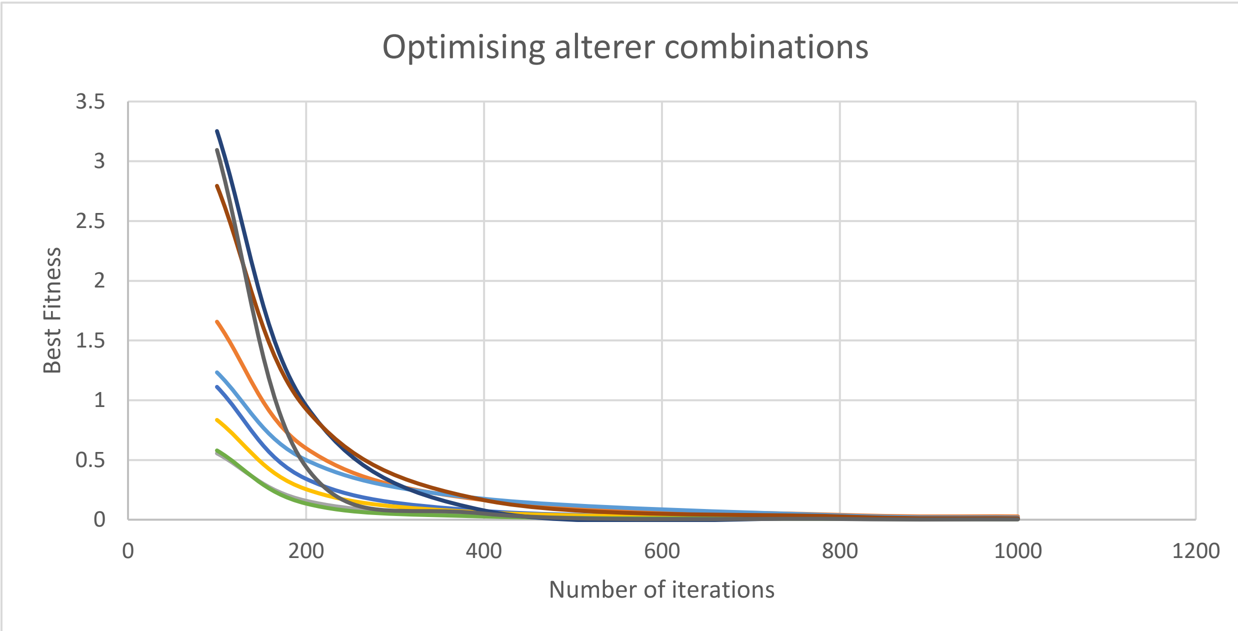
Tournament size	Best Fitness
2	0.010382267
3	0.009315359
4	0.010488771
5	0.010124565
6	0.008066977
7	0.008710772
8	0.008038372
9	0.008503321
10	0.013734796

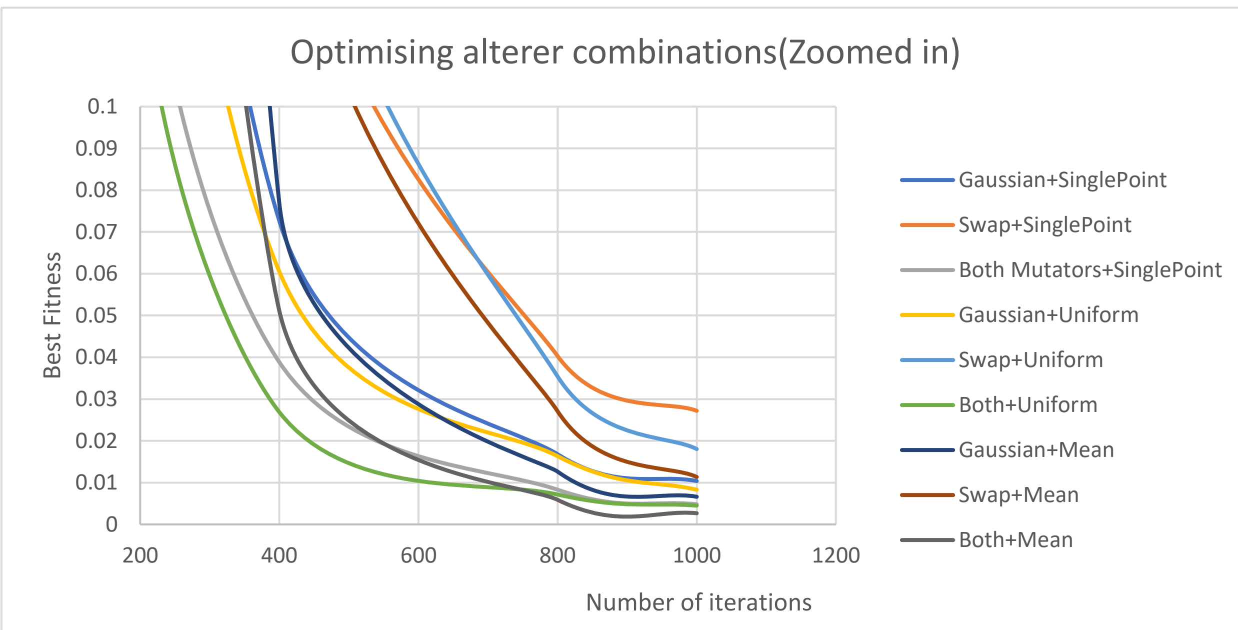


Analysis of the performance of alterers being used

All possible alterer combinations were tested against base parameter values. Using both mutators with the Mean Crossover yielded the smallest Best Fitness value. This combination was used when optimising the mutator probability and crossover probability values.

Alterers used	Best Fitness
Gaussian Mutator & Single Point Crossover	0.010382267
Swap Mutator & Single Point Crossover	0.027182957
Swap Mutator & Gaussian Mutator & Single Point Crossover	0.004812659
Gaussian Mutator & Uniform Crossover	0.008288543
Swap Mutator & Uniform Crossover	0.018032949
Swap Mutator & Gaussian Mutator & Uniform Crossover	0.004490749
Gaussian Mutator & Mean Crossover	0.006613809
Swap Mutator & Mean Crossover	0.011349596
Swap Mutator & Gaussian Mutator & Mean Crossover	0.002667814

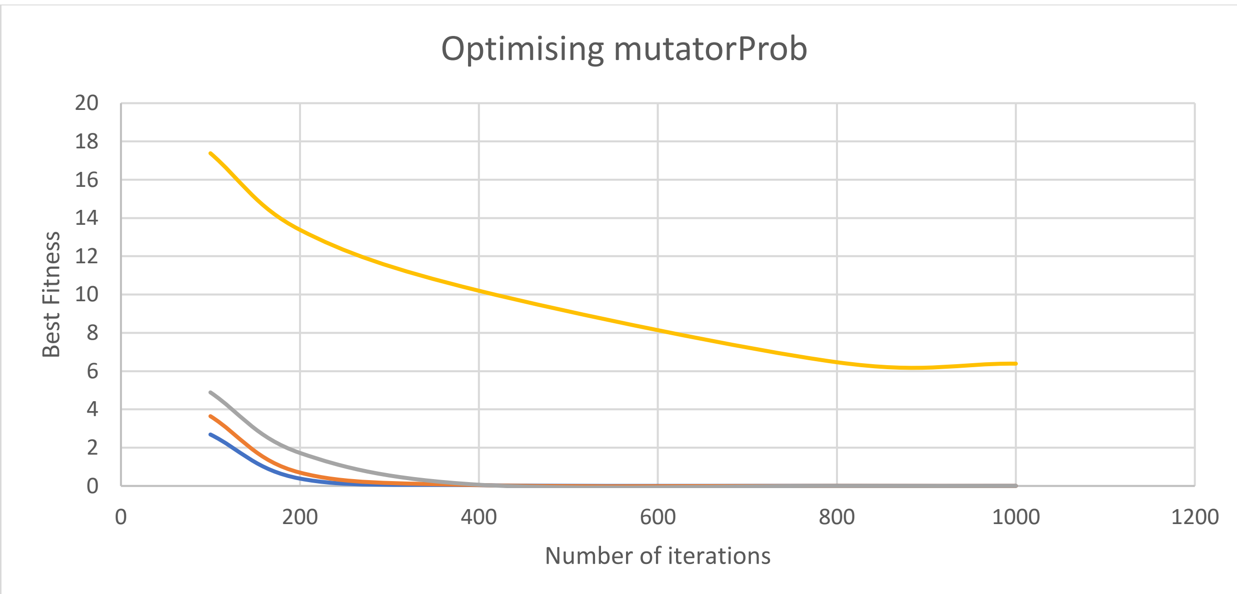

Optimising alterer combinations
Best Fitness
Number of iterations
0
0.5
1
1.5
2
2.5
3
3.5
0
200
400
600
800
1000
1200

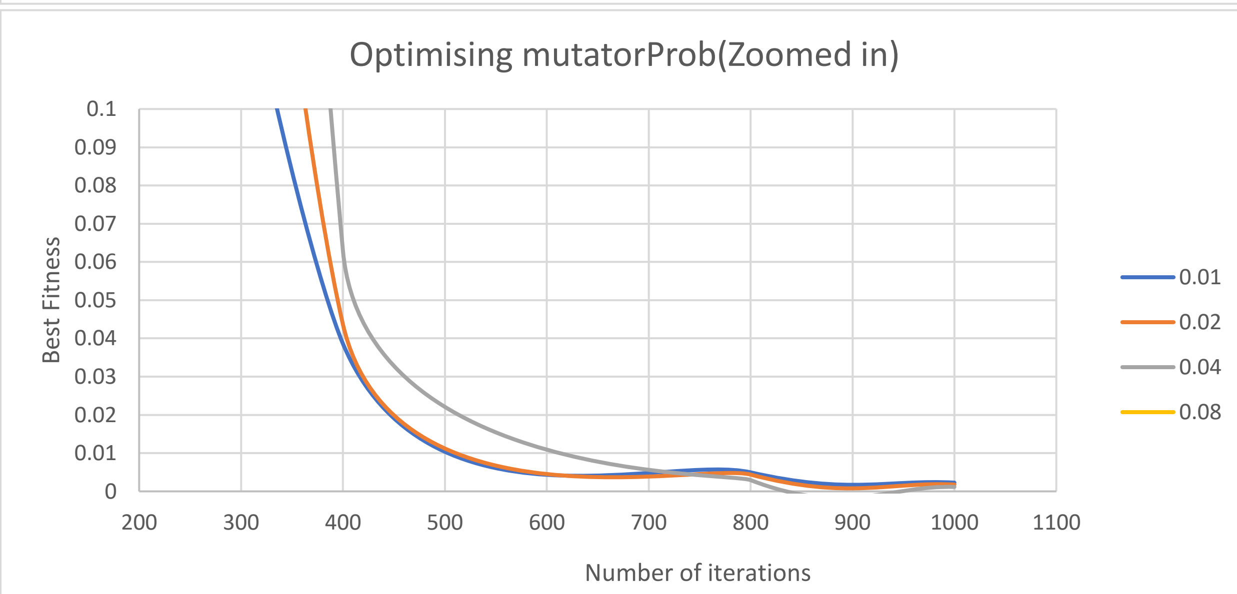

Optimising alterer combinations(Zoomed in)
Best Fitness
Number of iterations
0
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0.1
200
400
600
800
1000
1200
Gaussian+SinglePoint
Swap+SinglePoint
Both Mutators+SinglePoint
Gaussian+Uniform
Swap+Uniform
Both+Uniform
Gaussian+Mean
Swap+Mean
Both+Mean

Calibration of the mutator probability

When testing mutatorProb, the value range was originally meant to be between 0 and 1, however, as Best Fitness shows a sharp increase on a mutation probability of 0.08 no higher values were tested. The best result was recorded when mutationProb was set at 0.04.

Mutation Probability	Best Fitness
0.01	0.002288976
0.02	0.001776241
0.04	0.001156841
0.08	6.39048425

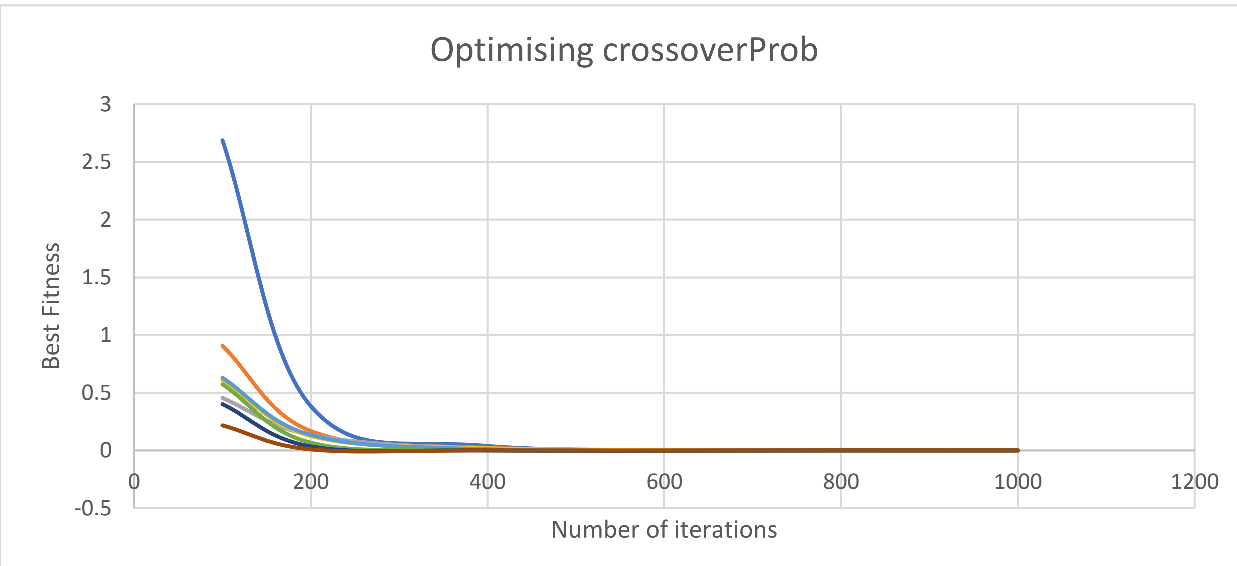


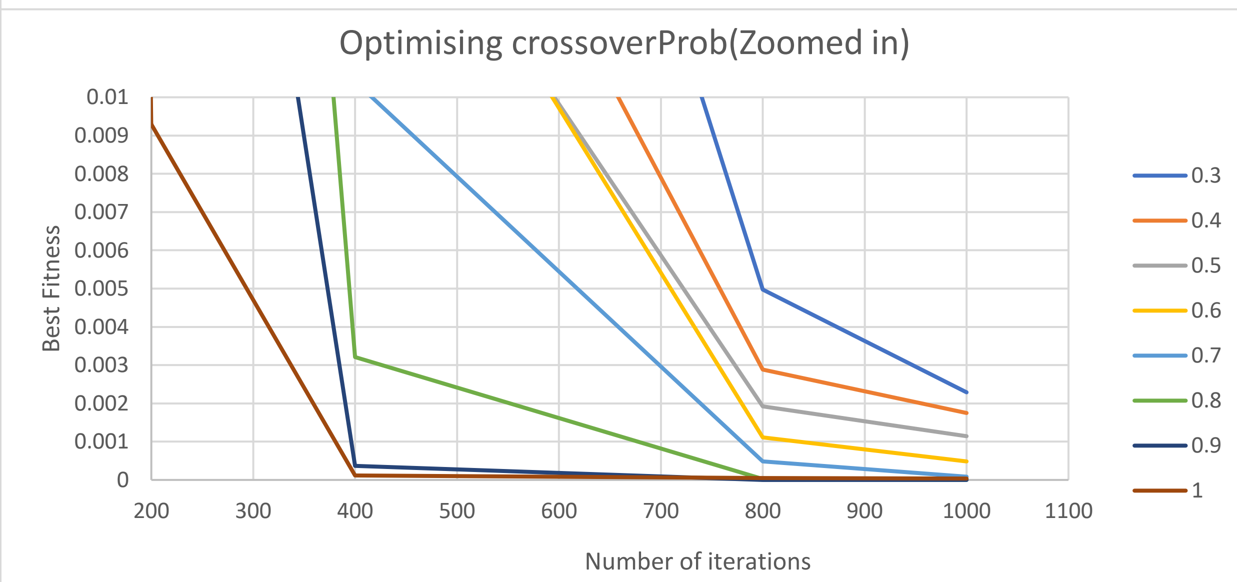


Calibration of the crossover probability

The crossover probability parameter was tested on a value range of 0 to 1. It was observed that increasing the value generally resulted in a smaller best fitness, the best values being recorded at a crossover probability of 0.9.

CrossoverProb	Best Fitness
0.3	0.002288976
0.4	0.001749403
0.5	0.001142942
0.6	4.84E-04
0.7	8.52E-05
0.8	2.60E-06
0.9	4.15E-07
1	3.67E-05





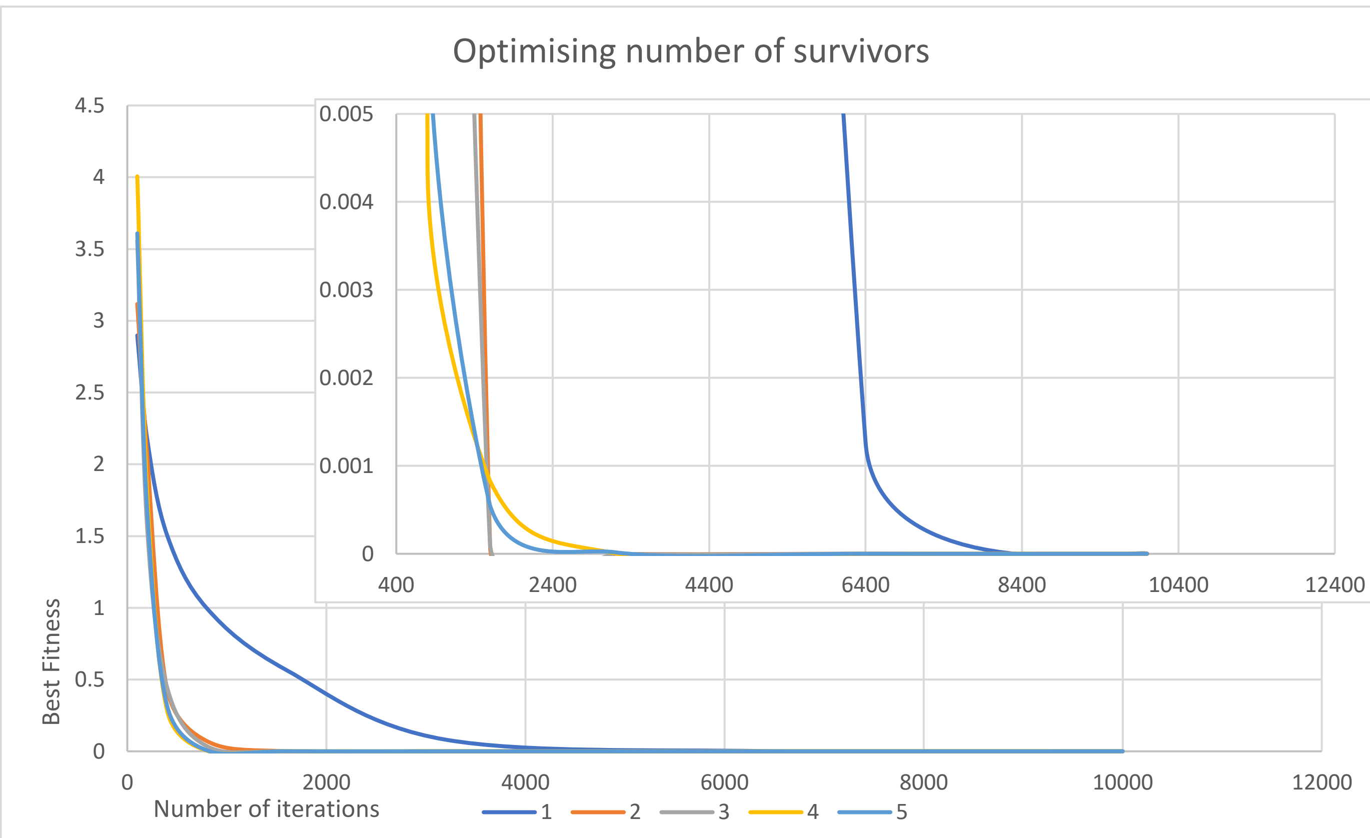
Computation of all the best values found

To ensure the best possible optimisation was being achieved, parameters that were originally tested against the base parameter values were tested again while the other parameters were set at their optimised values.

Further optimisation of the number of survivors

Similarly to the previous test, a smaller number of survivors seemed to positively influence best fitness. Unlike the first test, however, the best result was recorded for 3 survivors and not 2.

Number of survivors	Best Fitness
1	1.72E-06
2	2.17E-08
3	1.93E-09
4	9.80E-09
5	7.20E-08



Further optimisation of the tournament size

Second tests differed wildly from the first attempt. While using the best values for all the other parameters, the smallest Best Fitness was recorded for a tournament size of 4, a value two times as small as the one resulted from the first test.

Final optimised parameters

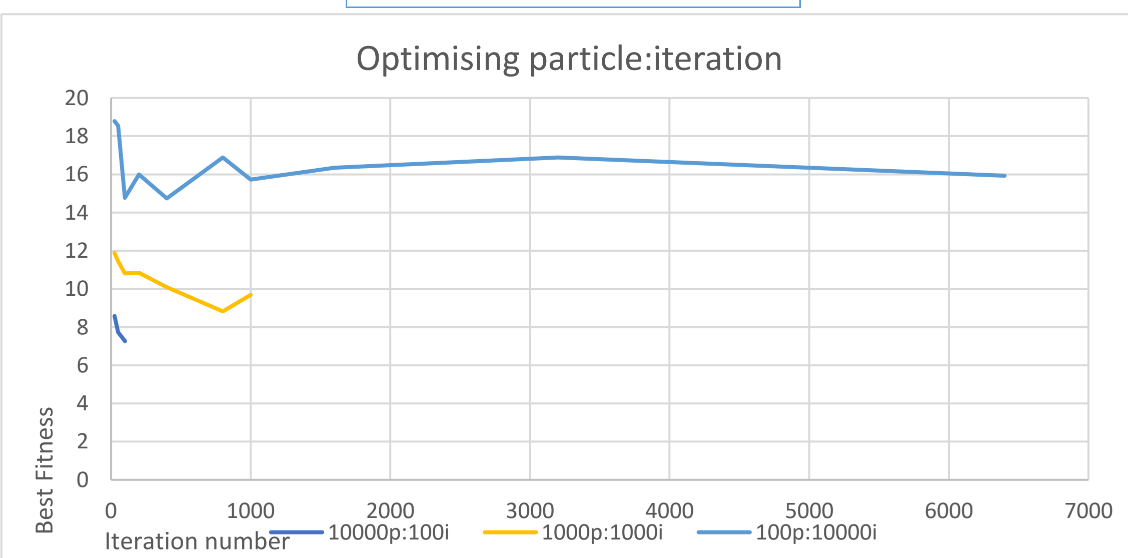
The smallest overall Best Fitness was recorded to be 1.93E-09 while using the following parameters:

Population:Iteration	100:10000
numSurvivors	3
tournamentSize	4
Alterers	Both mutators + Mean crossover
mutationProb	0.04
crossoverProb	1

Particle swarm optimisation algorithm

Optimisation of the particle to iteration ratio

Particle : Iteration	Best Fitness
10000p:100i	7.262585859
1000p:1000i	9.684261025
100p:10000	15.25601473



Optimisation of the weight values

As the weight parameters are largely dependent of each other, the previously employed optimisation strategy of testing each parameter by itself was deemed to be ineffective. Instead, the weight parameters were tested as combinations. Each of them was set with values ranging from 0 to 4. For the sake of finding the best possible outcome, all possible combinations were tested. This, however, proved to be very time demanding. There were 625 possible combinations with each of them having to be run 30 times to provide a reliable result, each run, the fitness function had to be called a million times. The whole data gathering process was automatised, but the code ran for close to 2 hours. A more organic approach of manually trying and adjusting the weight parameters might have been more time efficient but finding the absolute best combination was deemed unlikely.

Some notable combinations will be presented.

neigh	inertia	personal	global	Best Fitness
1	2	1	2	1.355264
1	2	2	0	0.755344
1	2	3	0	1.173982
1	3	1	2	1.977297
1	3	2	0	0.479293
1	4	1	2	1.546134
1	4	2	0	0.191881
2	0	1	1	0.827267
2	1	1	1	1.150824
2	2	1	1	0.900186
2	3	1	1	1.536876
2	4	1	1	1.416921

Further optimisation of the weight values

Once the best combination of values in the range of 0 to 4 had been determined, the values were incremented by 0.5 to further fine tune Best Fitness.

neigh	inertia	personal	global	Best Fitness
0.5	3.5	1.5	0	1.161224222
0.5	3.5	2	0	0.641514991
0.5	3.5	2.5	0	0.192659504
0.5	4	1.5	0	1.250262673
0.5	4	2	0	0.570848695
0.5	4	2.5	0	0.39047425
1	3.5	1.5	0	1.678645011
1	3.5	2	0	0.330595582
1	3.5	2.5	0	0.371803792
1	4	1.5	0	1.008565899
1	4	2	0	0.354126869
1	4	2.5	0	0.480681159
1.5	3.5	1.5	0	1.807016559
1.5	3.5	2	0	2.107589953
1.5	3.5	2.5	0	3.402439518
1.5	4	1.5	0	2.024281381
1.5	4	2	0	1.687053926
1.5	4	2.5	0	2.811385517

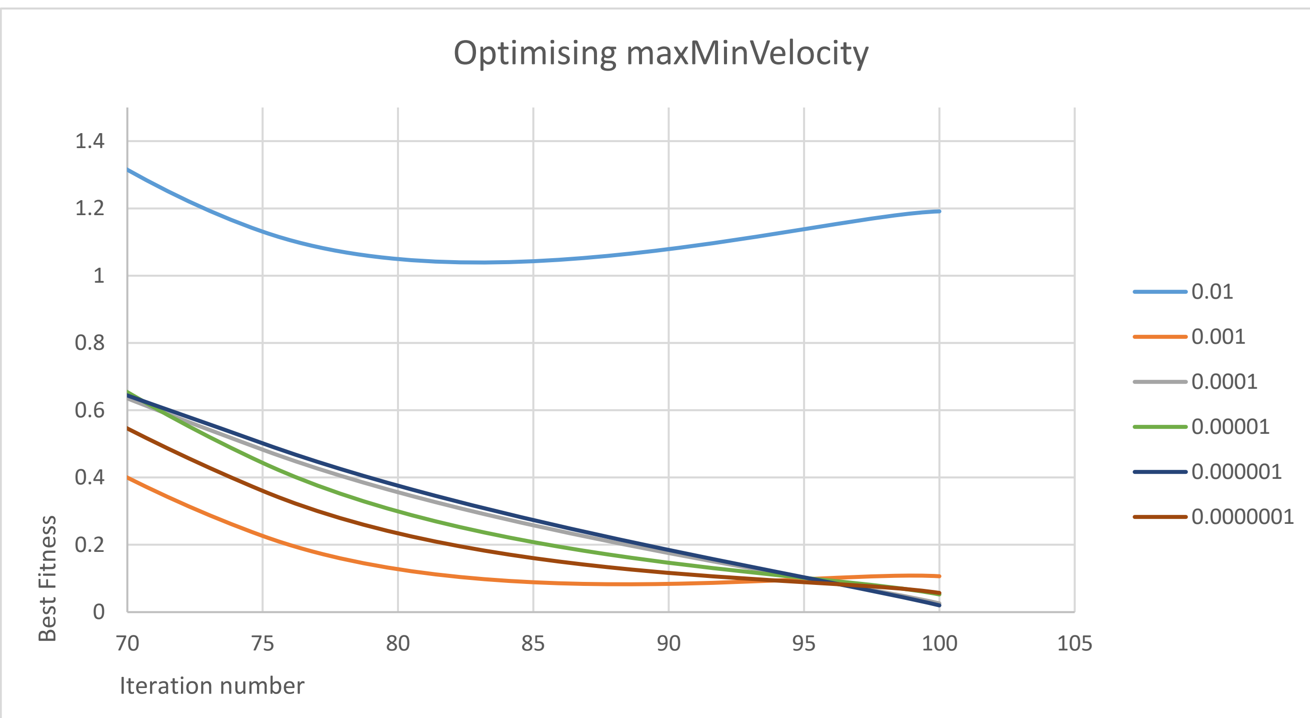
The new values were finally incremented by 0.1 to find the best possible weight combination.

neigh	inertia	personal	global	Best Fitness
0.5	3.5	2.5	0.1	0.094480854
0.5	3.5	2.5	0.2	0.352813167
0.6	3.5	2.5	0.1	0.054690175
0.7	3.5	2.5	0.1	0.120257832
0.6	3.4	2.5	0.1	0.058941655
0.6	3.6	2.5	0.1	0.047644424
0.6	3.7	2.5	0.1	0.058972896
0.6	3.7	2.4	0.1	0.049081084
0.6	3.7	2.3	0.1	0.060770375
0.5	3.6	2.5	0.2	0.020261638
0.5	3.6	2.6	0.2	0.014965103
0.6	3.5	2.7	0.1	0.005837617

Optimisation of the velocity value

Once the combination of weight values that produced the best result was found, it was used when testing values for the maxMinVelocity parameter.

MaxMinVelocity	Best Fitness
1	54.97117005
0.1	17.06325899
0.01	1.191202843
0.001	0.106560378
0.0001	0.025965979
0.00001	0.052568894
0.000001	0.02003701
0.0000001	0.05699948
0.00000001	0.085437795
0.000000001	0.101564171
1E-10	0.2340393



The algorithm calibration process for scenario 2: 10000 calls

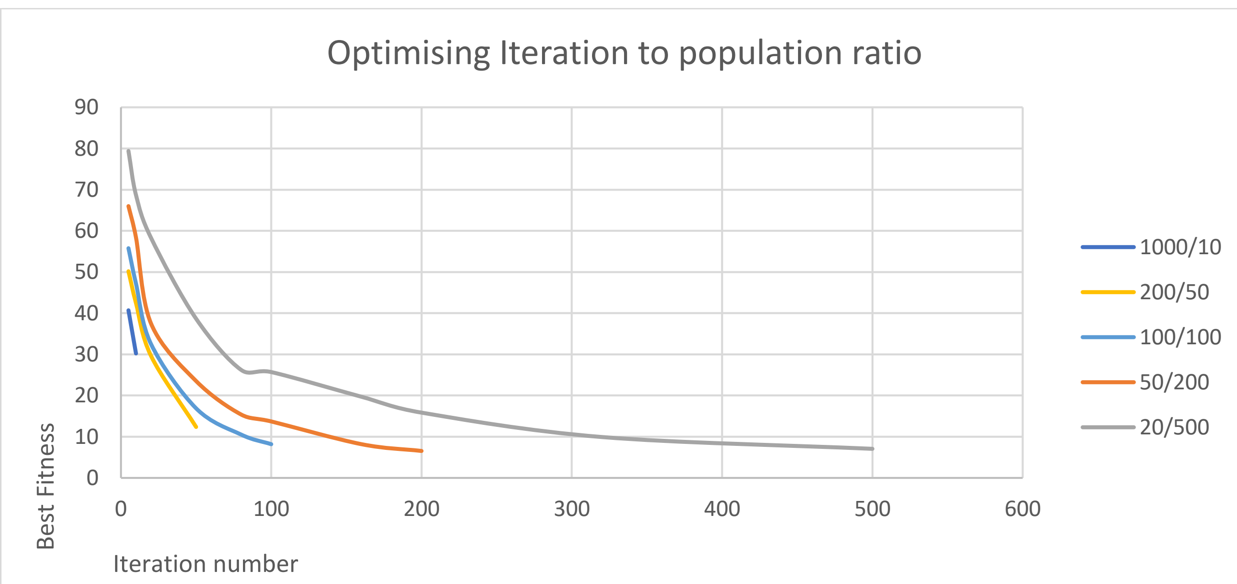
Genetic algorithm

The calibration process was approached in a slightly different manner than the 1M call scenario had been. Rather than testing each parameter individually, the best results obtained from testing were integrated iteratively when testing any new parameter. This was found to be a more reliable way of calibration, as parameters were found to be reliant on each other.

Different population to iteration ratio analysis

Five iteration ratios were tested. The lowest Best Fitness was found when using a ratio of 50 population to 200 iterations. This ratio was used for any further testing.

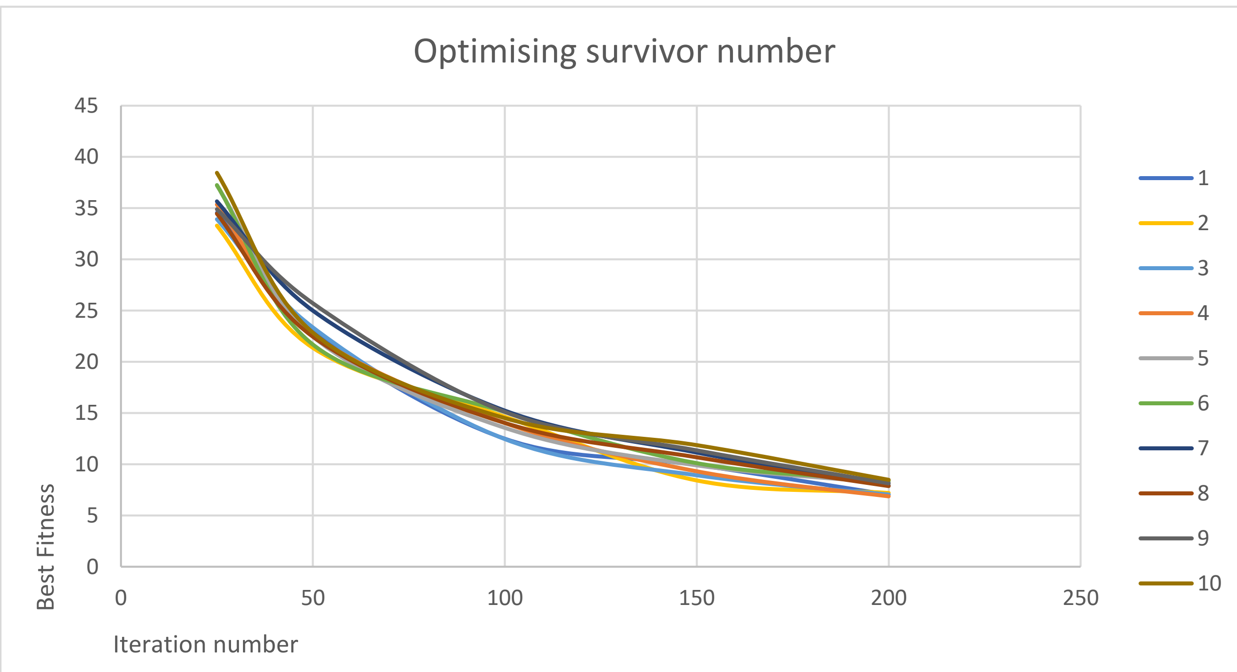
Population to Iteration	Best Fitness
1000p:10i	30.22452379
200p:50i	12.37238247
100p:100i	8.18800154
50p:200i	6.546588555
20p:500i	7.067131381



Calibration of the number of survivors

The numSurvivors variable was tested the same manner as the previous scenario. Unlike the 1M calls scenario, where the best result was obtained when using a value of 2 survivors, the lowest best fitness resulted when numSurvivors was 4.

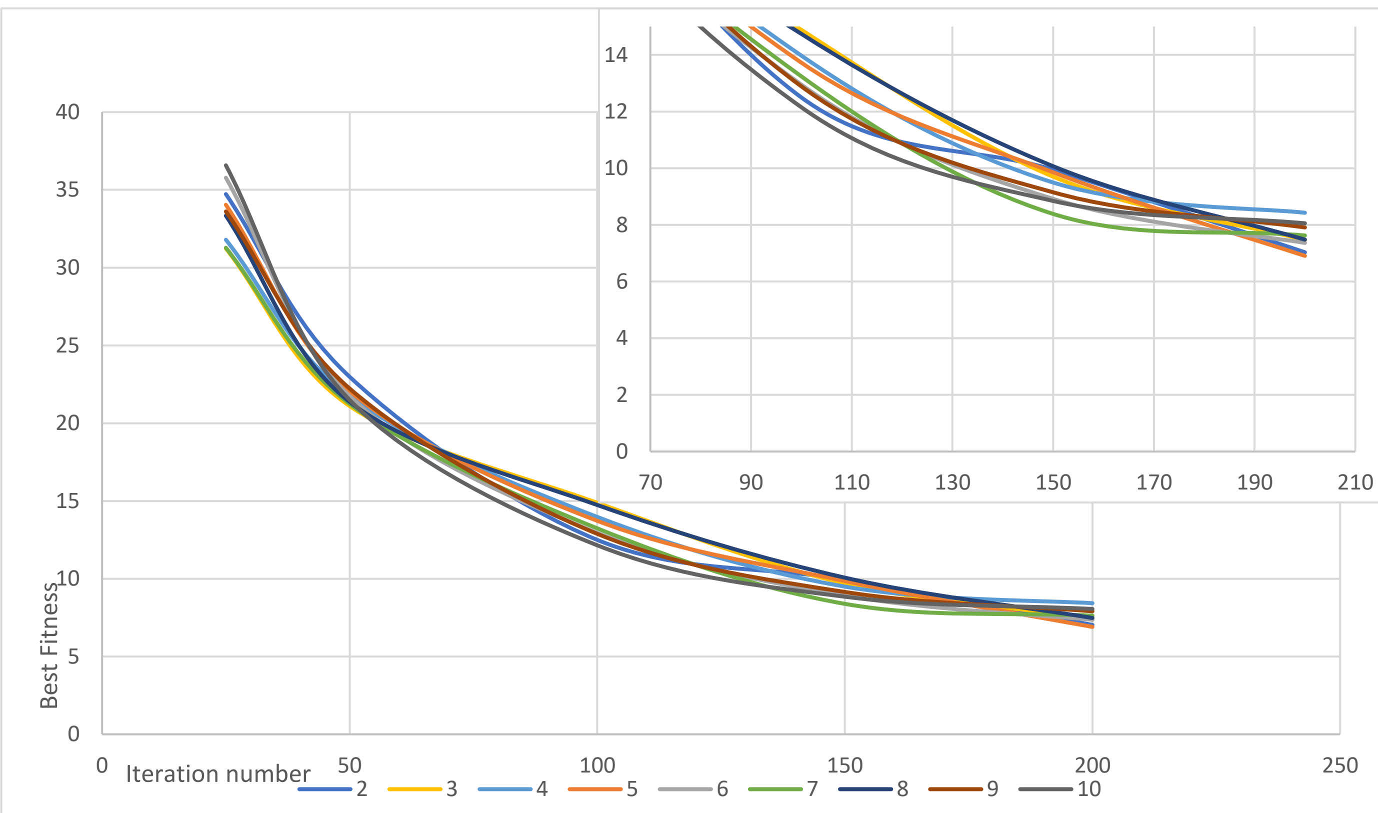
Survivors	Best Fitness
1	7.031709382
2	7.198851515
3	7.047557466
4	6.872804488
5	8.196723398
6	8.389127378
7	7.912807166
8	7.872578617
9	8.14167152
10	8.483882318



Calibration of the tournament size

The tournament size was tested using the same parameter range as in the first scenario. However, the best result came from a tournament size of 5, rather than 8.

Tournament size	Best Fitness
2	7.031709382
3	7.467110915
4	8.425800749
5	6.904851126
6	7.362484018
7	7.625213795
8	7.478485107
9	7.908258285
10	8.055749955



Analysis of the performance of alterers being used

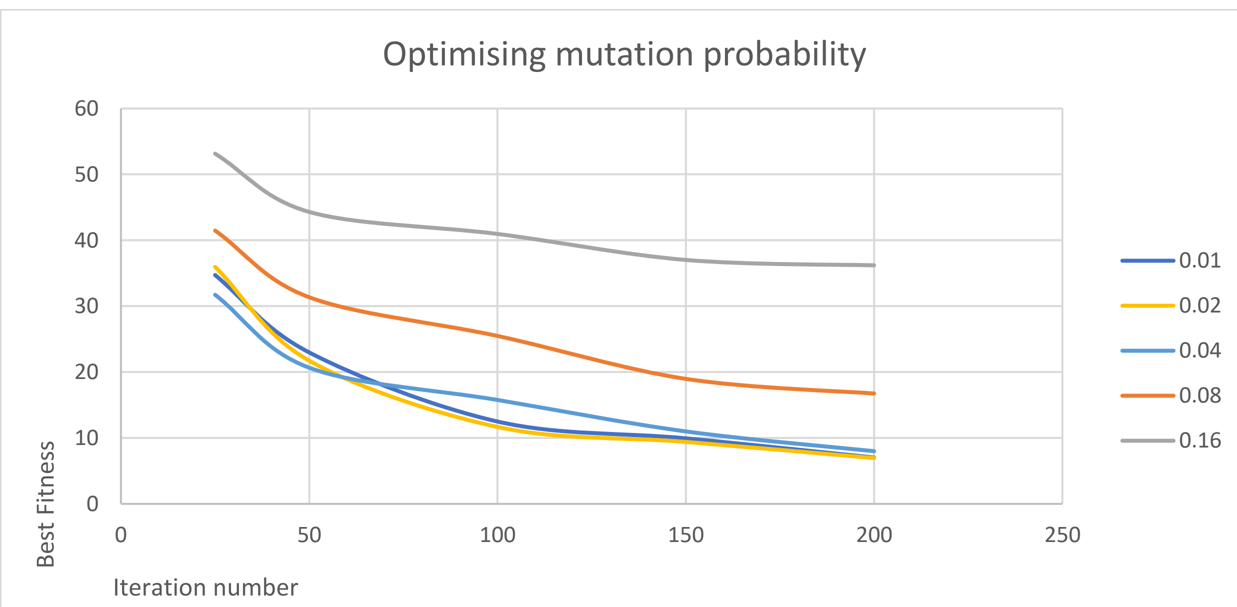
After testing all the altererer combinations possible, it was determined that, similarly to the first 1M calls scenario, the lowest fitness value resulted when using both mutators and the Mean Crossover.

Alterer combinations	Best Fitness
Gaussian Mutator & Single Point Crossover	10.68603637
Swap Mutator & Single Point Crossover	14.85659757
Swap Mutator & Gaussian Mutator & Single Point Crossover	7.891788924
Gaussian Mutator & Uniform Crossover	11.77951629
Swap Mutator & Uniform Crossover	13.5009874
Swap Mutator & Gaussian Mutator & Uniform Crossover	7.583853663
Gaussian Mutator & Mean Crossover	11.46466095
Swap Mutator & Mean Crossover	11.02891004
Swap Mutator & Gaussian Mutator & Mean Crossover	7.183205987

Calibration of the mutator probability

The mutatorProb parameter was tested in a range value of 0.1 to 0.16. The best result was recorded at a value of 0.2.

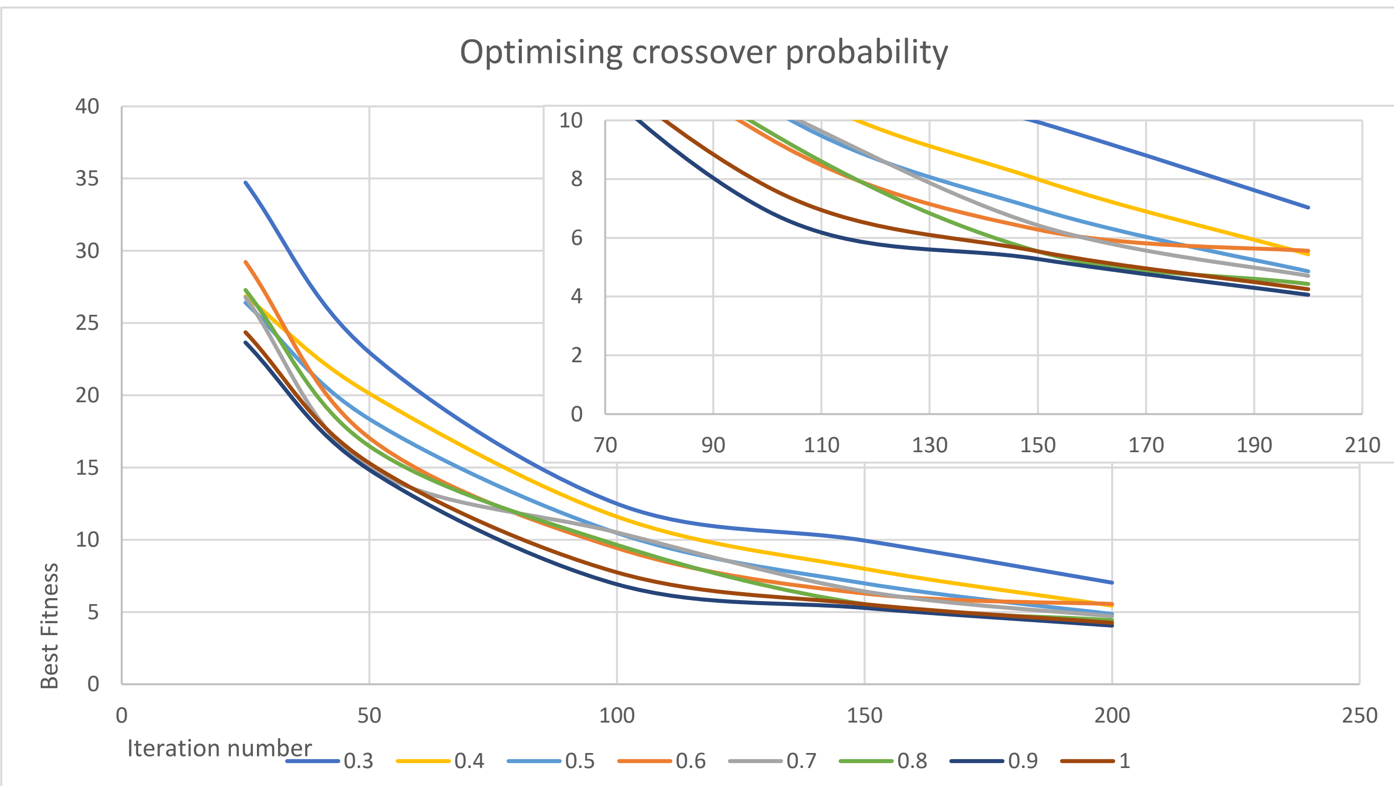
Mutation Probability	Best Fitness
0.1	7.03170938
0.2	6.95486554
0.4	7.99734679
0.8	16.7486637
0.16	36.2023807



Calibration of the crossover probability

The crossover probability parameter was tested using values ranging from 0.3 to 1, the best result being recorded at a crossover probability of 0.9.

Crossover Probability	Best Fitness
0.3	7.03170938
0.4	5.44142483
0.5	4.85971953
0.6	5.55706342
0.7	4.70700178
0.8	4.43141576
0.9	4.06115027
1	4.25186572



Computation of the best values found

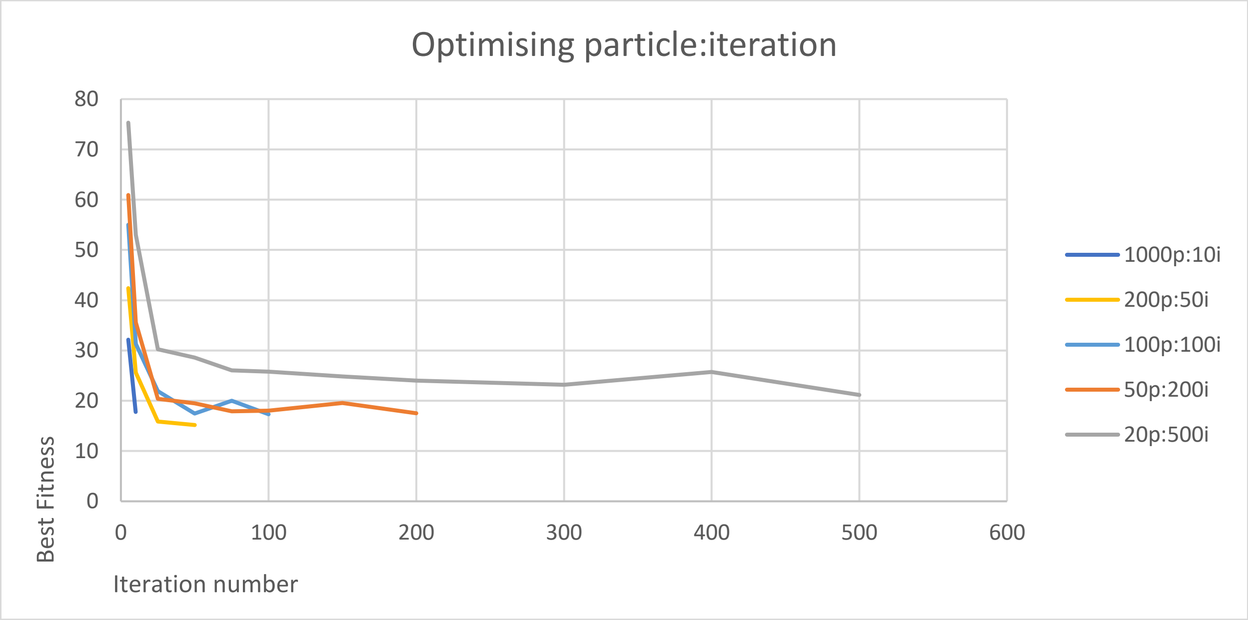
With some further tweaking, an all-time low Best fitness value of 3.294484469 was recorded using the following parameter values.

Population:Iteration	50p:200i
numSurvivors	3
tournamentSize	4
Alterers	Both mutators + Mean crossover
mutationProb	0.04
crossoverProb	1

Particle swarm optimisation algorithm

Optimisation of the particle to iteration ratio

Particle:Iteration	Best Fitness
1000p:10i	17.77806214
200p:50i	15.17152837
100p:100i	17.31635386
50p:200i	17.51588657
20p:500i	21.1550835



Optimisation of the weight values

The weight values were optimised following the same strategy as in the first scenario, the four parameters were tested simultaneously, using smaller value increments as testing progressed.

neigh	inertia	personal	global	Best Fitness
1.5	0.5	1.5	0.5	9.257318
0.5	0	2	0.5	8.362321
0.5	0	2.5	0.5	8.968857
0.5	0.5	2	0.5	7.674177
0.5	0.5	2.5	0.5	6.963428
1	0	2	0.5	7.926418
1	0.5	2	0.5	7.309157

neigh	inertia	personal	global	Best Fitness
0.4	0.4	2.4	0.4	7.530887
0.4	0.4	2.4	0.5	7.767344
0.4	0.4	2.4	0.6	7.000043
0.4	0.4	2.5	0.4	7.101686
0.4	0.5	2.4	0.5	6.532895
0.4	0.5	2.6	0.5	6.388311
0.5	0.4	2.4	0.4	6.538541
0.5	0.4	2.6	0.5	6.417414
0.5	0.5	2.6	0.4	6.348652
0.5	0.6	2.5	0.5	6.544236
0.6	0.5	2.5	0.4	6.379112

Optimisation of the velocity value

The velocity parameter was tested using values ranging from 1 to 1E-10 the best result being noted at a value of 0.000001.

MaxMinVelocity	Best Fitness
1	6.891784985
0.1	6.929522117
0.01	6.93746934
0.001	7.563513366
0.0001	8.217416932
0.00001	7.180797733
0.000001	6.305367331
0.0000001	6.8115441
0.00000001	6.529642148
0.000000001	7.285063088
1E-10	6.799774541

Critical comparison between the two algorithms

PSO and GA share many similarities in the way they operate, such as the fact that the system is initialized with a population of random solutions, and the search for the optimal solution is performed by updating generations.

Unlike GA, PSO has no evolution operators, such as crossover and mutation. The variables in PSO can take any values based on their current position in the particle space and the corresponding velocity vector

In the scenario of 1M calls to the Fitness function, optimisation of the GA resulted in a lowest Best Fitness value of 1.93E-09, while PSO only managed a value of 0.02003701.

Furthermore, GA produced much more cohesive results, while PSO was found to often produce wildly distinct results over the same parameter values. Despite not converging to a solution as accurate as GA, PSO was much more efficient, requiring a lower number of iterations to reach its better solutions.

When calling the Fitness function 10k times, previously made assumptions on the advantages and disadvantages of the tow algorithms were further confirmed.

The optimised GA produced a Best Fitness value of 3.294484469, while the calibrated PSO produced a value of 6.305367331. Once again, running the same number of iterations, GA proved to arrive at a better solution.

Genetic algorithms do not handle complexity in an efficient way, because the number of elements undergoing mutation in very large which causes a considerable increase in the search space. So, in this case PSO is the best alternative as it requires small number of parameters and correspondingly lower number of iterations.

The results of all performed tests favour PSO as being better in terms of speed and computational power required. However, PSO proved to be undeniably worse in terms of accuracy and reliability of results.

References

A. Engelbrecht, —Fundamentals of computational swarm intelligence, 2006, ‖ Hoboken: John Wiley & Sons, Ltd.

N. Nedjah, L. dos Santos Coelho, and L. de Macedo Mourelle, —Multiobjective swarm intelligent systems: theory & experiences. ‖ Springer Science & Business Media, 2009, vol. 261.

R. Singla, S. Shabir, —A Comparative Study of Genetic Algorithm and the Particle Swarm Optimization ‖ International Journal of Electrical Engineering, 2016, vol. 9, Number 2, pp. 215-223

W. Roetzel, X. Luo, and D. Che, —Design and Operation of Heat Exchangers and their Networks,2020 ‖ Elsevier, Inc.