Computational Intelligence Solution for Market Pricing

This report outlines two computational intelligence approaches to help solve the supermarket pricing problem. Both approaches are outlined and critically compared before proposing a winning solution.

The application developed for the purpose of this investigation uses settings provided in the configuration files to run each algorithm 5 times and then runs both once before generating a total of 3 line graphs in the 'Results' directory using the JFreeChart library. These graphs allow a visualisation of the results for better analysis. Appendix 1 provides details on how to run the application.

For the problem at hand, the representation of candidate solutions are an Array of double values reflecting the prices for a certain number of items. These are described as strategies. The constraints of the these are dependent on the supermarket but for the purpose of this investigation, the strategy will be restricted to 20 items each priced between £0.01 and £10.00.

The end goal of each of the proposed algorithms is to find a strategy that generates the highest revenue. The provided fitness evaluation function takes a valid strategy and calculates the total revenue that it generates. This provides a measure of quality that will be used to maximise the total revenue. The calculation for the fitness evaluation is predetermined based on various factors that are usually considered when supermarket items are priced which will not be detailed in this report.

The resources for the proposed algorithms have been restricted where each run is limited to a certain amount of fitness evaluation calls. This ensures a common termination condition to allow for a fair comparison and evaluation.

Particle Swarm Optimisation Algorithm (PSO) for Supermarket Pricing Problem

For initialisation in this implementation of PSO, a set number of particles are generated to populate the swarm. For each particle in this swarm, the initial position is set as a randomly generated valid strategy, the personal best position is set to the initial position and finally the initial velocity is equal to half of the difference between the initial position and another randomly generated valid strategy. The swarm's global best position is set to the best strategy within this initial population.

Iterations then begin where the particle's position is updated by adding the velocity to it's current position:

$$\vec{x_i}(t+1) = \vec{x_i}(t) + \vec{v_i}(t)$$

As the current positions are updated, the global position is also being updated. The velocity is then updated using uniform random strategies and the inertial, cognitive and social coefficients provided in the parameter settings:

$$\vec{v_i}(\mathrm{t}+1) = \eta \cdot \vec{v_i}(\mathrm{t}) + \phi_1 \cdot \vec{r_1} \cdot (\vec{p_i}(t) - \vec{x_i}(t)) + \phi_2 \cdot \vec{r_2} \cdot (\vec{g}(t) - \vec{x_i}(t))$$

The inertial (n), cognitive ($\varnothing_1$), and social ($\varnothing_2$) coefficients are taken from the *parameterSettings.txt* configuration file.

For constraint handling the invisible wall method is used which only evaluates the strategy if it is within the bounds. The attraction to personal and global best causes particles to return to the feasible region.

Evolutionary Algorithm (EA) for Supermarket Pricing Problem

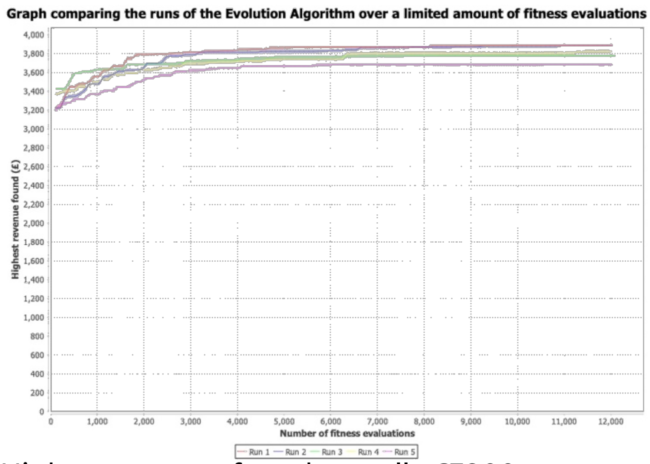
For initialisation in this implementation of EA, the initial population is populated with random valid strategies.

The selection operator is based on Charles Darwin's 'survival of the fittest' using tournament selection where a set amount of random strategies from the population are ranked. The two fittest strategies are selected to produce offspring for the next generation. Once selected, they are recombined. The recombination operator used is a slight variation of the order 1 crossover operator. This is due to the fact that order 1 crossover is problematic for the problem at hand. With each strategy having any value within the price range, the values that would be selected from one parent may not exist or exist multiple times in the other parent resulting in a strategy with more or less values than there is room. This implementation alternatively selects a sequence from the first parent and copies it over to the child in the same position as usual. It then fills the remaining parts with the values from the other parent without checking for identical values. After recombination, based on the mutation probability, the offspring may mutate. The operator for mutation is swap where the offspring is replaced by a random selection from its 2-opt neighbourhood. This process is repeated until the size of the new generation is equal to the population limit. The survival per generation is generational. This is repeated until the termination condition is met. Over time, natural selection causes a rise in the fitness of the population.

During a run of the EA, there comes a point where all strategies in the population are either very similar or identical. An alternative to the implementation described above is if the two fittest strategies that are chosen during selection are identical, then one is replaced by a random strategy from of the set of potential parents. This prevents limiting the direction of improvement thus generating solutions of higher quality as shown in figure 1.

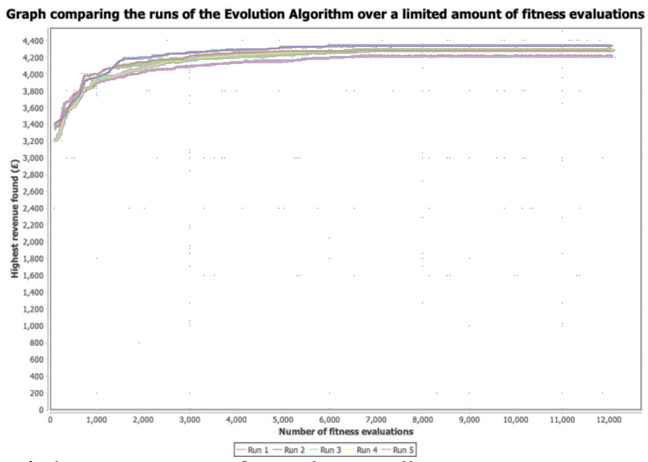
Figure 1 – Graphs showing results of EA before and after implementing an alternate implementation

Previous implementation



Highest revenue found overall ~£3900

Alternative implementation



Highest revenue found overall ~£4350

The alternate implementation improves the exploitation of the algorithm as it finds a better strategy in less fitness evaluations.

<u>Comparison</u>

In order to undertake a fair comparison between the solutions, optimum parameter values for each algorithm have been discovered through trial and error tuning (appendix 2). These values will be used for this evaluation.

Figure 2 – Comparison results with limit of 100,000 fitness evaluations

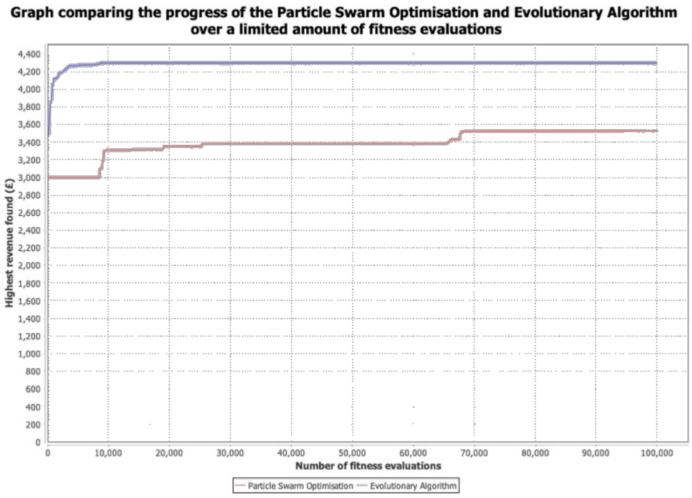
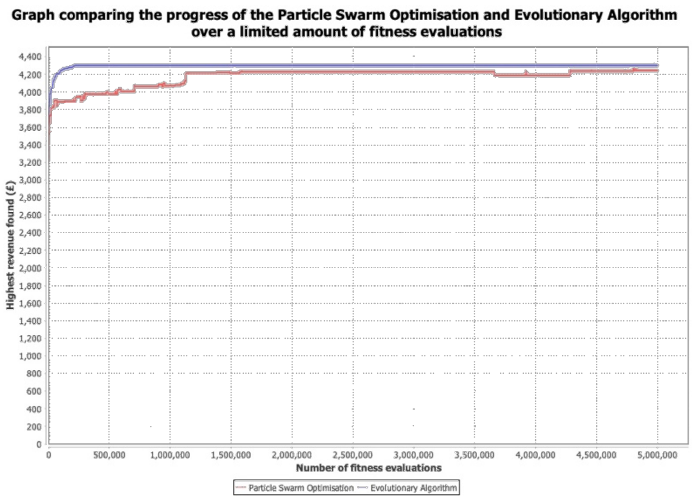


Figure 2 displays the results of the comparison. Both algorithms generate strategies of high quality. With a limit of 100,000 fitness evaluations, the EA comes out on top in finding the highest revenue. It does this within 10,000 fitness evaluations but there is no visible progress after that point. This shows that it prioritises exploitation. The PSO however, makes a gradual improvement over the fitness evaluation budget. This shows that it prioritises exploration.

Based on this It could be argued that the PSO will eventually find a better strategy if given a large enough budget of fitness evaluations.

Figure 3 backs the claim although requiring a significant budget of fitness evaluations.

Figure 3 – Comparison results with limit of 5 million fitness evaluations



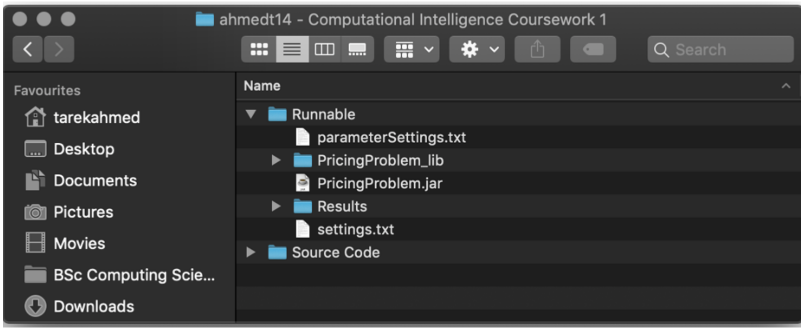
To conclude, the PSO and EA implemented during this investigation have both proven to be quality solutions to the market pricing problem.

The winning solution is simply based on a trade-off between speed of improvement and the chance to find the better strategy in the long term. Therefore, the deciding factor is dependent on the budget of fitness evaluations. With an extremely high budget of fitness evaluations, the PSO will eventually generate the most profitable strategy. With a low budget of fitness evaluations, the EA will generate the most profitable strategy.

Appendices

Appendix 1 - Instructions on running the application.

1. Unzip the compressed submission folder. Everything required to run the application is within the *Runnable* directory.



2. Open the *settings.txt* file and enter the desired settings. By default, the console only prints the relevant test details.
   a. For details on each iteration, set the 'PRINT ALL ITERATIONS TO CONSOLE' setting to 'true'.
   b. For details on the goods setup, set the 'PRINT GOODS SETUP DETAILS TO CONSOLE' setting to 'true'

   Save and close the file.

settings.txt

```
PRINT ALL ITERATIONS TO CONSOLE=false
PRINT GOODS SETUP DETAILS TO CONSOLE=false
```

3. Open the *parameterSettings.txt* file and enter the desired settings (by default, the file contains the best settings found for each algorithm during the investigation). Save and close the file.

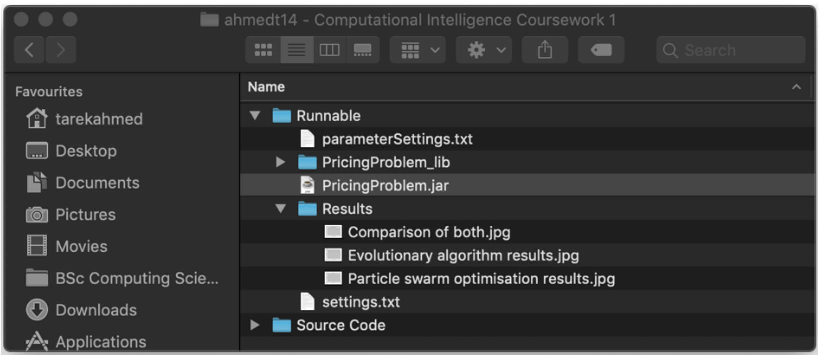
parameterSettings.txt

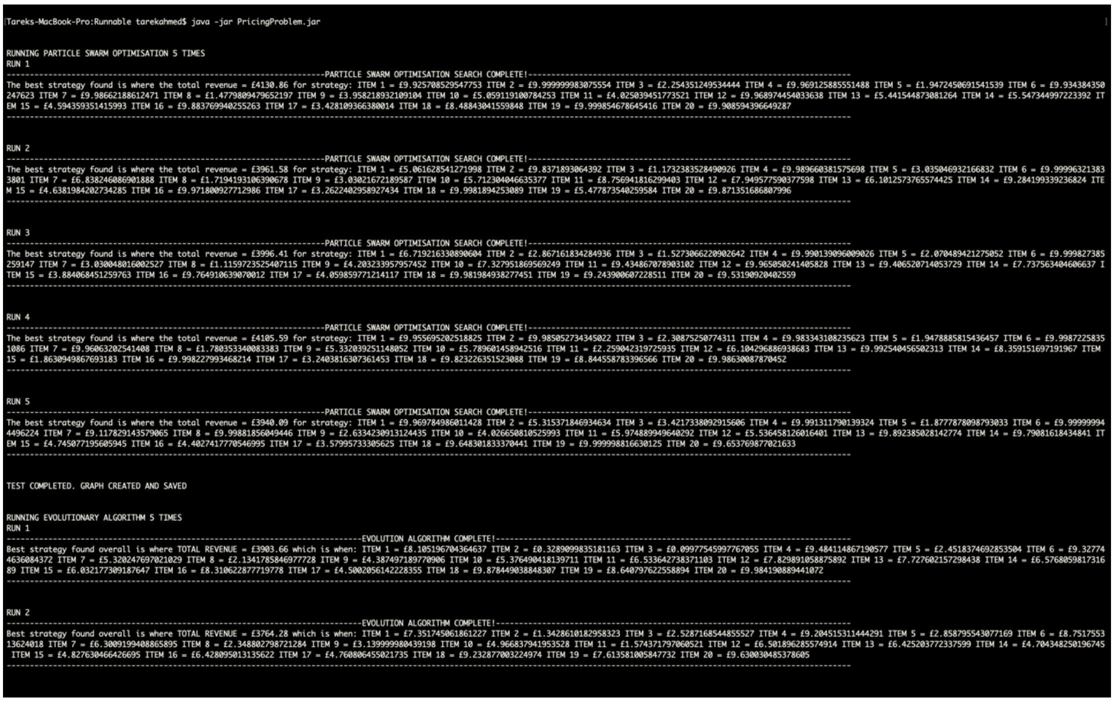
```
NUMBER OF ITEMS IN STRATEGY=20
MINIMUM PRICE OF PRODUCTS=0.01
MAXIMUM PRICE OF PRODUCTS=10.00

FITNESS EVALUATIONS LIMIT=100000
POPULATION SIZE=100

Particle Swarm Optimisation Algorithm Parameters
THETA 1=1.0
THETA 2=0.2
N=0.2

Evolutionary Algorithm Parameters
MUTATION PROBABILITY=0.8
CROSSOVER SET SIZE=12
PARENT SELECTION TOURNAMENT SIZE=20
```

4. To run the application, double click on *PricingProblem.jar*. To run on a Terminal, navigate to the *Runnable* directory in a Terminal window and type the command:

   `java -jar PricingProblem.jar`

   Once the application completes running, the result graphs will be generated in the *Results* directory.


ahmedt14 - Computational Intelligence Coursework 1
Search
Favourites
tarekahmed
Desktop
Documents
Pictures
Movies
BSc Computing Scie...
Downloads
Applications
Name
Runnable
parameterSettings.txt
PricingProblem_lib
PricingProblem.jar
Results
Comparison of both.jpg
Evolutionary algorithm results.jpg
Particle swarm optimisation results.jpg
settings.txt
Source Code


Tareks-MacBook-Pro:Runnable tarekahmed$ java -jar PricingProblem.jar
RUNNING PARTICLE SWARM OPTIMISATION 5 TIMES
RUN 1
------PARTICLE SWARM OPTIMISATION SEARCH COMPLETE!------
The best strategy found is where the total revenue = £4130.86 for strategy: ITEM 1 = £9.925708529547753 ITEM 2 = £9.999999983075554 ITEM 3 = £2.254351249534444 ITEM 4 = £9.969125885551488 ITEM 5 = £1.9472450691541539 ITEM 6 = £9.934384350
247623 ITEM 7 = £9.98662188612471 ITEM 8 = £1.4779809479652197 ITEM 9 = £3.958218932109104 ITEM 10 = £5.059119100784253 ITEM 11 = £4.025039451773521 ITEM 12 = £9.968974454033638 ITEM 13 = £5.441544873081264 ITEM 14 = £5.547344997223392 IT
EM 15 = £4.594359351415993 ITEM 16 = £9.883769940255263 ITEM 17 = £3.428109366380014 ITEM 18 = £8.48843041559848 ITEM 19 = £9.999854678645416 ITEM 20 = £9.908594396649287
RUN 2
------PARTICLE SWARM OPTIMISATION SEARCH COMPLETE!------
The best strategy found is where the total revenue = £3961.58 for strategy: ITEM 1 = £5.061628541271998 ITEM 2 = £9.8371893064392 ITEM 3 = £1.1732383528490926 ITEM 4 = £9.989660381575698 ITEM 5 = £3.035046932166832 ITEM 6 = £9.99996321383
3801 ITEM 7 = £6.838246086901888 ITEM 8 = £1.7194193106390678 ITEM 9 = £3.03021672189587 ITEM 10 = £5.712304046635377 ITEM 11 = £8.756941816299403 ITEM 12 = £7.949577590377598 ITEM 13 = £6.1012573765574425 ITEM 14 = £9.284199339236824 ITE
M 15 = £4.6381984202734285 ITEM 16 = £9.971800927712986 ITEM 17 = £3.2622402958927434 ITEM 18 = £9.9981894253089 ITEM 19 = £5.477873540259584 ITEM 20 = £9.871351686807996
RUN 3
------PARTICLE SWARM OPTIMISATION SEARCH COMPLETE!------
The best strategy found is where the total revenue = £3996.41 for strategy: ITEM 1 = £6.719216330890604 ITEM 2 = £2.867161834284936 ITEM 3 = £1.5273066220902642 ITEM 4 = £9.990139096009026 ITEM 5 = £2.070489421275052 ITEM 6 = £9.999827385
259147 ITEM 7 = £3.030048016002527 ITEM 8 = £1.1159723525407115 ITEM 9 = £4.203233957957452 ITEM 10 = £7.327951869569249 ITEM 11 = £9.434867078903102 ITEM 12 = £9.965050241405828 ITEM 13 = £9.406520714053729 ITEM 14 = £7.737563404606637 I
TEM 15 = £3.884068451259763 ITEM 16 = £9.764910639070012 ITEM 17 = £4.059859771214117 ITEM 18 = £9.981984938277451 ITEM 19 = £9.243900607228511 ITEM 20 = £9.53190920402559
RUN 4
------PARTICLE SWARM OPTIMISATION SEARCH COMPLETE!------
The best strategy found is where the total revenue = £4105.59 for strategy: ITEM 1 = £9.955695202518825 ITEM 2 = £9.985052734345022 ITEM 3 = £2.30875250774311 ITEM 4 = £9.983343108235623 ITEM 5 = £1.9478885815436457 ITEM 6 = £9.9987225835
1086 ITEM 7 = £9.96063202541408 ITEM 8 = £1.780353340083383 ITEM 9 = £5.332039251148052 ITEM 10 = £5.789601458942516 ITEM 11 = £2.259042319725935 ITEM 12 = £6.104296886938683 ITEM 13 = £9.992540456502313 ITEM 14 = £8.359151697191967 ITEM
15 = £1.8630949867693183 ITEM 16 = £9.998227993468214 ITEM 17 = £3.2403816307361453 ITEM 18 = £9.823226351523088 ITEM 19 = £8.844558783396566 ITEM 20 = £9.98630087870452
RUN 5
------PARTICLE SWARM OPTIMISATION SEARCH COMPLETE!------
The best strategy found is where the total revenue = £3940.09 for strategy: ITEM 1 = £9.969784986011428 ITEM 2 = £5.315371846934634 ITEM 3 = £3.4217338092915606 ITEM 4 = £9.991311790139324 ITEM 5 = £1.8777878098793033 ITEM 6 = £9.99999994
4496224 ITEM 7 = £9.117829143579065 ITEM 8 = £9.99881856049446 ITEM 9 = £2.6334230913124435 ITEM 10 = £4.026650810525993 ITEM 11 = £5.974889949640292 ITEM 12 = £5.536458126016401 ITEM 13 = £9.892385028142774 ITEM 14 = £9.79081618434841 IT
EM 15 = £4.745077195605945 ITEM 16 = £4.4027417770546995 ITEM 17 = £3.57995733305625 ITEM 18 = £9.648301833370441 ITEM 19 = £9.999998816630125 ITEM 20 = £9.653769877021633
TEST COMPLETED. GRAPH CREATED AND SAVED
RUNNING EVOLUTIONARY ALGORITHM 5 TIMES
RUN 1
------EVOLUTION ALGORITHM COMPLETE!------
Best strategy found overall is where TOTAL REVENUE = £3903.66 which is when: ITEM 1 = £8.105196704364637 ITEM 2 = £0.3289099835181163 ITEM 3 = £0.09977545997767055 ITEM 4 = £9.484114867190577 ITEM 5 = £2.4518374692853504 ITEM 6 = £9.32774
4636084372 ITEM 7 = £5.320247697021029 ITEM 8 = £2.1341785846977728 ITEM 9 = £4.387497189770906 ITEM 10 = £5.376490418139711 ITEM 11 = £6.533642738371103 ITEM 12 = £7.829891058875892 ITEM 13 = £7.727602157298438 ITEM 14 = £6.5768059817316
89 ITEM 15 = £6.032177309187647 ITEM 16 = £8.310622877719778 ITEM 17 = £4.5002056142228355 ITEM 18 = £9.878449038848307 ITEM 19 = £8.6407976622558894 ITEM 20 = £9.984190889441072
RUN 2
------EVOLUTION ALGORITHM COMPLETE!------
Best strategy found overall is where TOTAL REVENUE = £3764.28 which is when: ITEM 1 = £7.351745061861227 ITEM 2 = £1.3428610182958323 ITEM 3 = £2.5287168544855527 ITEM 4 = £9.204515311444291 ITEM 5 = £2.858795543077169 ITEM 6 = £8.7517553
13624018 ITEM 7 = £6.3009199408865895 ITEM 8 = £2.348802798721284 ITEM 9 = £3.139999980439198 ITEM 10 = £4.966837941953528 ITEM 11 = £1.574371797060521 ITEM 12 = £6.501896285574914 ITEM 13 = £6.425203772337599 ITEM 14 = £4.704348250196745
ITEM 15 = £4.827630466426695 ITEM 16 = £6.428095013135622 ITEM 17 = £4.760806455021735 ITEM 18 = £9.232877003224974 ITEM 19 = £7.613581005847732 ITEM 20 = £9.630030485378605

Appendix 2 - Parameter tuning for EA and PSO

EVOLUTIONARY ALGORITHM

	Mutation Probability Tuning		Crossover Set Size Tuning		Parent Selection Tournament Size Tuning	
Mutation Probability	0.2		0.8		0.8	
Crossover Set Size	0	£3395.49	4	£4306.99	12	£3275.99
Parent Selection Tournament	2		2		5	
Mutation Probability	0.4		0.8		0.8	
Crossover Set Size	0	£3397.77	8	£4330.06	12	£4247.05
Parent Selection Tournament	2		2		10	
Mutation Probability	0.6		0.8		0.8	
Crossover Set Size	0	£3448.5	12	**£4342.41**	12	£4266.24
Parent Selection Tournament	2		2		15	
Mutation Probability	0.8		0.8		0.8	
Crossover Set Size	0	**£3525.94**	16	£4289.75	12	**£4279.25**
Parent Selection Tournament	2		2		20	
Mutation Probability	1		0.8		0.8	
Crossover Set Size	0	£3423.11	20	£3699.07	12	£4270.55
Parent Selection Tournament	2		2		25	

PARTICLE SWARM OPTIMISATION

	Theta 1 Tuning		Theta 2 Tuning		N Tuning	
Theta 1	0.2		1		1	
Theta 2	0	£3424.36	0.2	**£4172.88**	0.2	**£3783.78**
N	0		0		0.2	
Theta 1	0.4		1		1	
Theta 2	0	£3483.02	0.4	£3373.23	0.2	£3221.78
N	0		0		0.4	
Theta 1	0.6		1		1	
Theta 2	0	£3269.29	0.6	£3106.29	0.2	£3196.51
N	0		0		0.6	
Theta 1	0.8		1		1	
Theta 2	0	£3333.96	0.8	£3276.12	0.2	£3261.48
N	0		0		0.8	
Theta 1	1		1		1	
Theta 2	0	**£3434.03**	1	£3111.57	0.2	£3207.37
N	0		0		1	