

**CS3910**

**Computational Intelligence**

**Comparison of two**

**Computational Intelligence**

**Algorithms on the Pricing Problem**

**Thavi Tennakoon**

**Introduction**

This report will be discussing the comparison of results between the Artificial Immune System (AIS) algorithm and the Particle Swarm Optimisation (PSO) algorithm both running a PricingProblem instance.

**Artificial Immune System**

First, initialise the population ArrayList with a bunch of lists of prices. The amount that’s fill in depends on the parameter we set. Also fill in a separate ArrayList with each list’s cost. The bestCost variable is filled in too so it can be used for normalisation. Using bestCost, it is possible to fill in one more ArrayList with all the normalised costs.

Clones can now be created by performing a hypermutation on each shopping list. The number of clones created for each list is decided by the cloneSizeFactor. During the hypermutation, the length of block will be selected and it will be used to mutate the list. Then those values in the block are given a random valid value.

Now in the selection process the program will go through every clone and keep the best mu routes in the population.

The penultimate step is that, in the metadynamics, the d worst lists will be removed and replace them with random ones.

Finally, evaluate this final list of routes to retrieve the best revenue after this one iteration. The amount of iterations can be manually set (or set a timer) before retrieving the best revenue gained.

**Particle Swarm Optimisation**

The class PSOPriceList.java contains all the properties of a single shopping list object. It stores the position (aka current prices), its personal best list, and its velocity. The velocity in this case will just be altering the prices of each item a little.

The main PSOPricingProblem.java class will create the swarm of PSOPriceList objects that will compare with each other to get the best revenue.

This stores a global best price list that all the other lists will consider when changing their prices.

After a given time limit the global best revenue is returned with its corresponding price list.

**Results**

**Artificial Immune System**

Figure 2: Best Revenue = 4058.83

Figure 1: Best Revenue = 4103.5

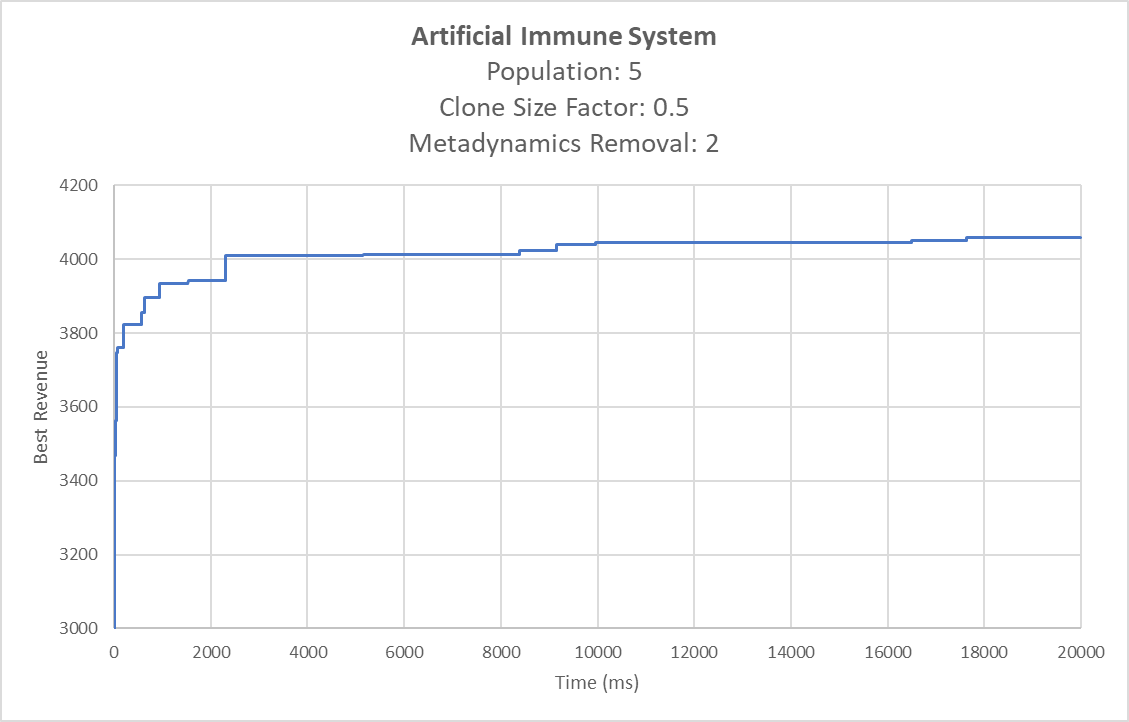
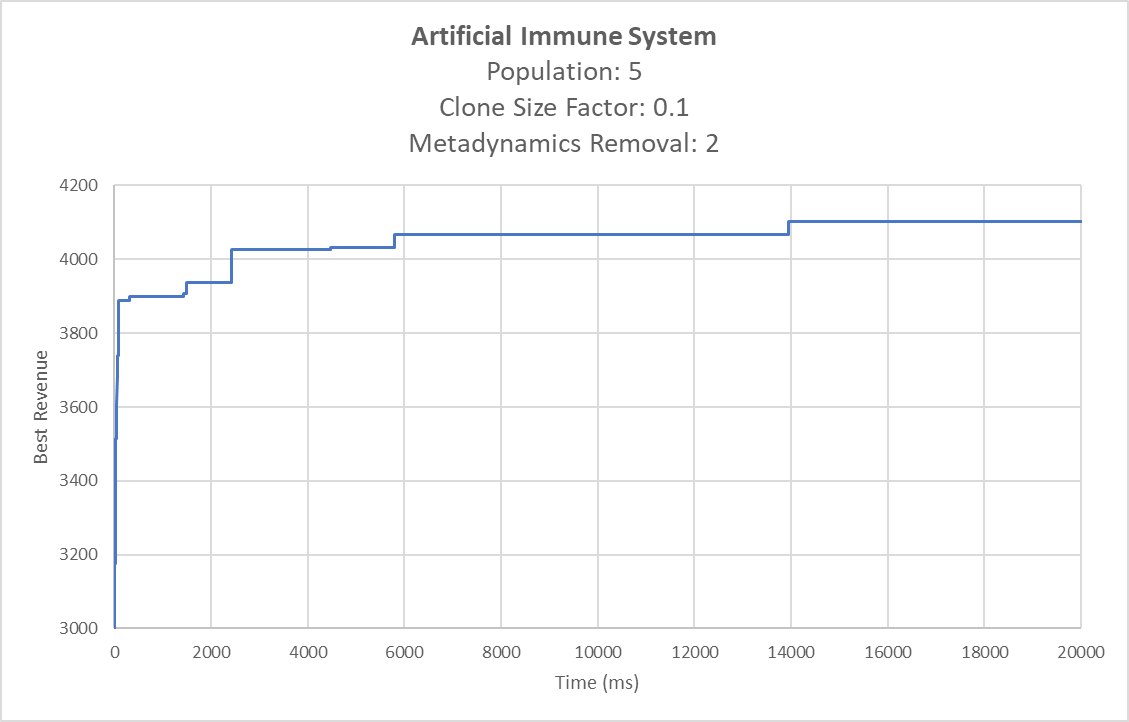
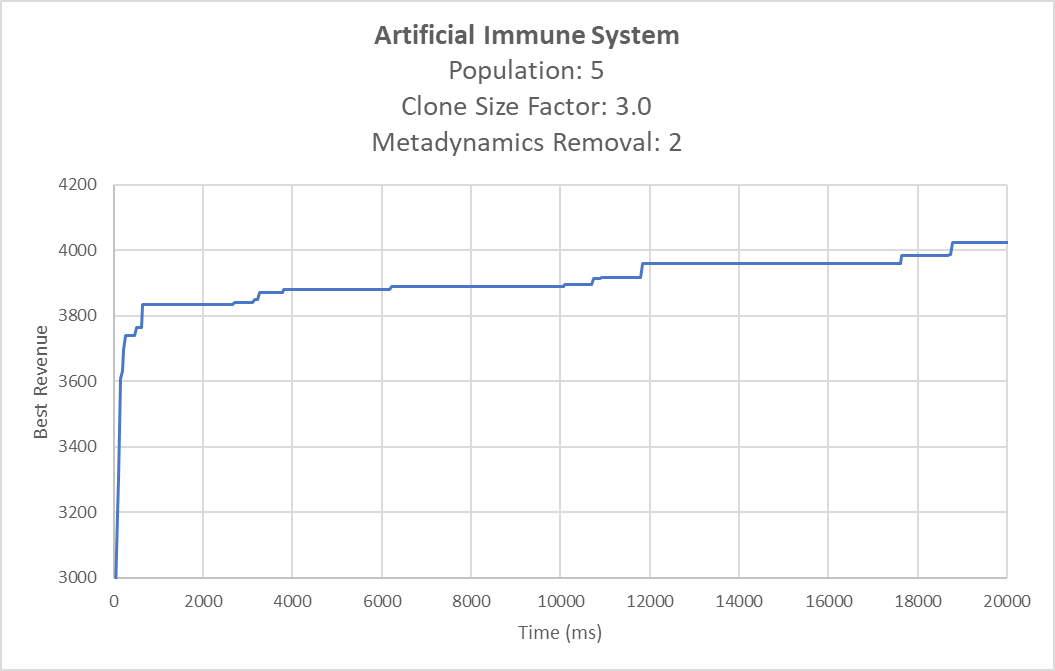


Figure 3: Best Revenue = 4023.74



These are the results after three different iterations of the Artificial Immune System algorithm running for 20 seconds.

In figure 1, a clone size factor of 0.1 is used. The initial revenue values are around the 3900 mark. The algorithm then slowly crawls its way up to its peak value, 4103.5.

In figure 2, a clone size factor of 0.3 is used. This follows a similar pattern to figure 1 where it slowly crawls to its peak value. However, the peak best revenue of 4058.83 is lower than figure 1.

In figure 3, a clone size factor of 3.0 is used. It takes longer to crawl to its peak value compared to figures 1 and 2. The peak best revenue of 4023.74 is also smaller than the first two figures.

Looking at the results, the lower the clone size factor is, the better the revenue gained is. The algorithm also manages to reach the peak value much more quickly with a smaller clone factor. You can see when the clone size factor is <1 the best revenue is above 4000 after about 2-3 seconds. When the clone size factor was 3 it took almost 19 seconds.

**Particle Swarm Optimisation**

Figure 5: Best Revenue: 4066.27

Figure 4: Best Revenue: 3953.43

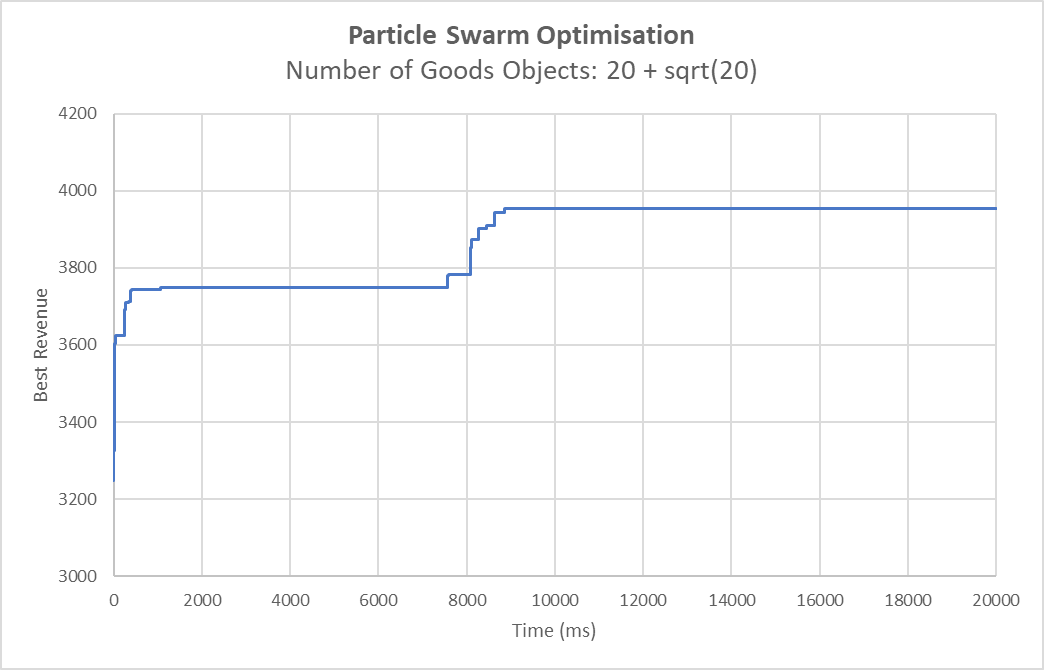
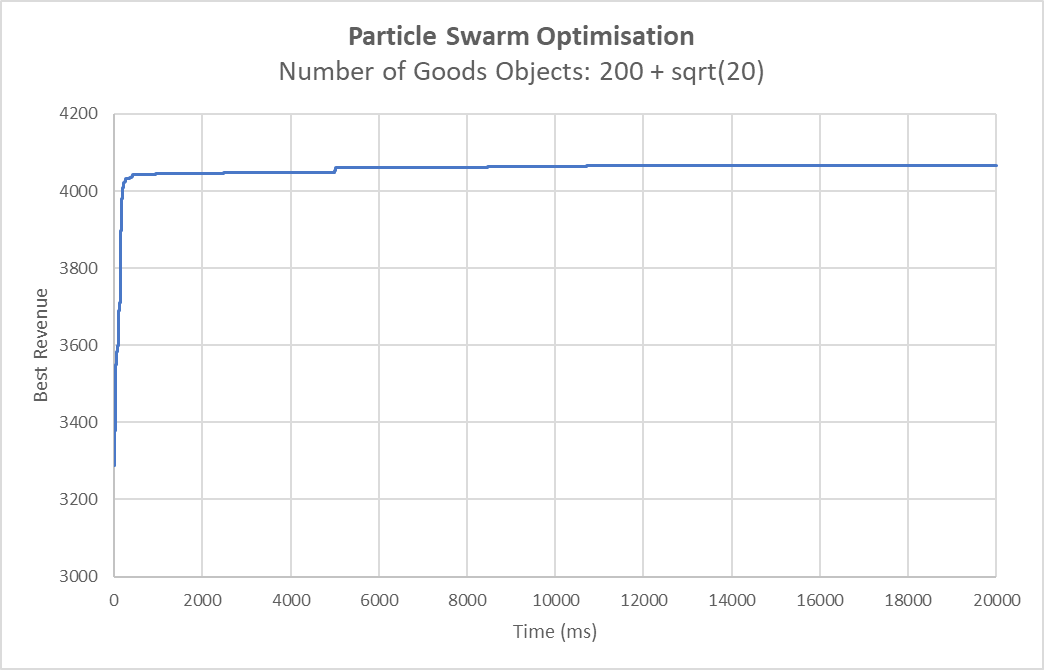
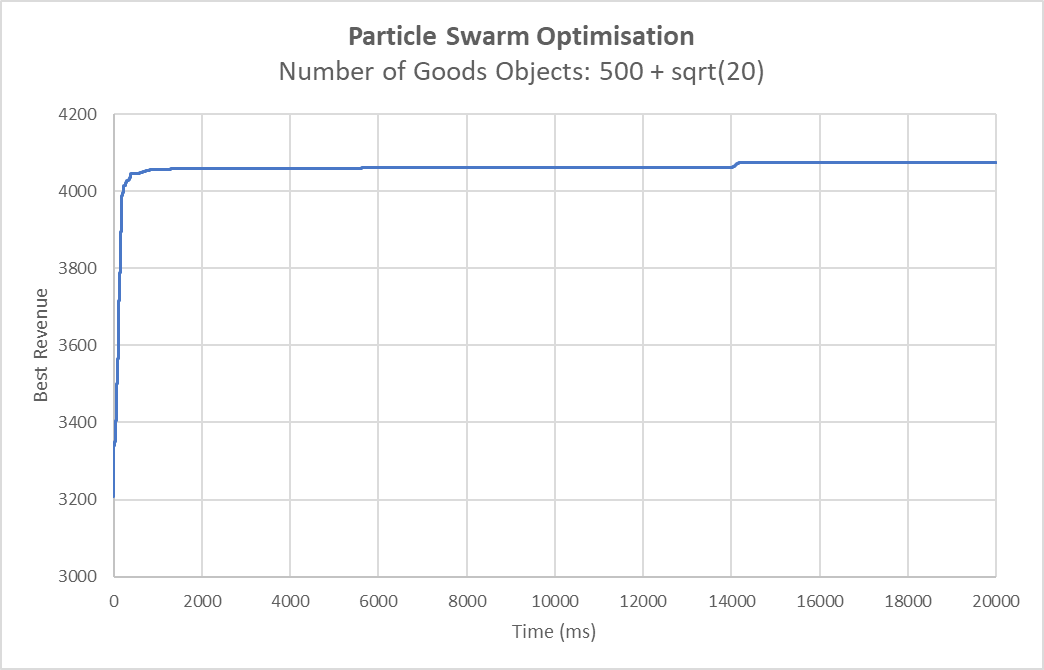


Figure 6: Best Revenue: 4074.33



These are the results after three different iterations of the Particle Swarm Optimisation algorithm running for 20 seconds. I’ve modified each iteration by mutating the number of goods objects in the swarm.

In figure 4, 20+sqrt(20) goods objects are in the swarm. The graph shows two stagnation points until it reaches the peak value at around 9s into the operation.

In figure 5, 200+sqrt(20) goods objects are in the swarm. Now is takes around 100ms to reach a very high best revenue. The peak best revenue is only a little higher than this.

In figure 6, 500+sqrt(20) goods objects are in the swarm. It follows the same suit as figure 5 where it immediately gets to a high best revenue.

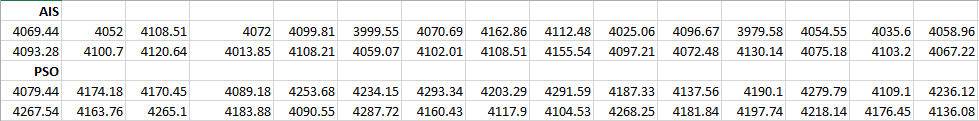
Looking at the results, it shows that the higher the swarm, the quicker the best revenue finds a peak value. However, the difference of best revenue between figure 5 and 6 is very small compared to figure 4 and 5. This means there’s no point to keep adding more particles in the swarm.

**Comparison**

Welch’s T-Test will be performed on the two data sets generated from both algorithms. Each algorithm has generated 30 results each.

Figure 7 shows the data sets generated by both algorithms.

Figure 7: Data Sets



Null Hypothesis: The Artificial Immune System algorithm is, on average, no better than the Particle Swarm Optimisation algorithm on the Pricing Problem after 30 fitness evaluations.

Alternative Hypothesis: The Artificial Immune System algorithm is, on average, better than Particle Swarm Optimisation algorithm on Pricing Problem after 30 fitness evaluations.

After performing Welch’s T-Test, a p-value of 1.65862E-10 is generated. This is less than the significance level (0.05) and therefore, the null hypothesis is rejected.

**Conclusion**

After analysing both algorithms on the PricingProblem instance I will say that the ‘winning’ solution is to use the PSO algorithm. This is because it finds a peak value for the best revenue faster than the AIS algorithm at their most optimal parameters.

It is worth noting that the AIS algorithm is also a completely viable option as it has found a peak value for best revenue that is close to the PSO algorithm. Objectively, the PSO algorithm is better, however depending on the seed you choose, the AIS can do better.

Also, seeing as the PricingProblem is a continuous problem, it is more suited for the PSO algorithm.