Introduction

The Bus Routing Problem is a complex NP hard problem. Unlike other NP hard problems such as the Travelling Salesman Problem, the Bus Routing Problem has no strict universal definition, and is often defined according to the needs of the particular investigation being carried out. In essence it is a routing problem, requiring nodes to be connected via weighted vertices in such a way as to minimise a cost value. The analogy of cities being connected by buses is convenient, but we could just as easily imagine servers being connected via network cables, or machines connected via conveyor belts. Whatever analogy we use, it is essential that a model is created that realistically simulates both the demand between the nodes (the passengers), and the flow allowed by a particular set of vertices (the buses). It is crucial that this model is defined in such a way that allows us to make reasonable and realistic comparisons between proposed solutions.

One of the key aims of this project is to investigate an original approach to the Bus Routing Problem that takes its inspiration from the relationship between the 'individual' and 'group' levels in the natural world. I intend to investigate a bottom-up approach that will simulate an environment populated with individuals (in this case the individual buses) competing with one another to survive. From the behaviour and co-evolution of these individuals will emerge an overall solution to the Bus Routing Problem. This approach has parallels with both Co-Evolutionary Algorithms, where the landscape being explored is dynamic and dependent on the current population, and Artificial Life, where solutions emerge from the local interactions of separate agents. Finally, in order to observe and understand these emergent solutions, I will develop a schematic representation that enables a user to visualise a problem and proposed solution.

Hillis' Sorting Networks

I want to start by gaining a better understanding of what distinguishes a co-evolutionary algorithm from more traditional genetic algorithms. To do this I will give an example of a co-evolutionary algorithm developed by Daniel Hillis [1].

Hillis attempts to tackle the problem of creating an optimal sorting network. He defines a sorting network as "a sorting algorithm in which the sequence of comparisons and exchanges of data take place in a predetermined order."[1a] Each network is represented as a series of comparisons and exchanges between pairs of numbers in a given sequence. The quality of such a network can be calculated by applying it to random sequences of numbers of a given length. He states that "A sorting network that correctly sorts all sequences of 1 and 0 will correctly sort any sequence." [1b] Therefore a proposed sorting network can be tested exhaustively with 2^n tests (where n is the length of the sequence to be sorted). He chooses to attempt to solve the particular case of n = 16, as this has been studied intensely by others, using non genetic approaches.

Initially, Hillis adopts a fairly standard evolutionary approach to the problem. He defines a genotype representation for a sorting network, creates a population, and then applies crossover and mutation to create offspring. He defines how a solution is rated as the "percentage of input cases for which the network produces the correct output." [1b] He explains he has chosen this definition as it offers "partial credit for partial solutions." and also that "it can be conveniently approximated by trying out the network on a random sample of the test cases" [1a]. However, it is this approximation that ultimately limits the standard evolutionary approach, as explained by [2] "using a fitness function which tests all possible permutations on each sorting network is impractical. Additionally, a static subset of permutations would clearly encourage solutions that sort only the chosen subset."

This problem of "over fitting" is well known in machine learning. The chosen subset to be tested can be thought of as a multi dimensional landscape on which the algorithm is searching. Each solution could be represented as a point on the landscape. An evolutionary algorithm would evolve a solution that maximised on that particular landscape only, as there is clearly no pressure on the solutions to adapt in any other way. However, we want a solution that is capable of sorting *any* sequence of numbers efficiently, not just the members of the test set.

Hillis' solution to this is ingenious. Instead of using a static test set of sequences, he instead creates a population of sequences to evolve alongside the sorting network population. These sequences are ranked on how poorly the sorting networks perform when attempting to sort them. The harder they are to sort, the higher their ranking. Based on this ranking, selection and crossover is performed, and over multiple

generations the sequences evolve to become better at not being sorted by the current set of sorting networks. Effectively, Hillis has succeeded in creating a virtual "arms race" between the sorting networks and the sequences requiring sorting. Now the solution landscape being explored is constantly shifting as the members of the sequence population adapt and become harder to sort. A selection pressure has been created that forces the sorting networks to generalise their sorting efficiency, and in turn forces the sequences to become harder to sort, driving the sorting networks to an optimum.

In nature we often see characteristics and features in animals that appear to have been either honed to perfection, or exaggerated grossly and seemingly, on first glance, unnecessarily. These features are usually the results of "arms races" with members of rival species or their own. For example, males are in an arms race with each other to "win" females. If the females of the species prefer males with a longer tail, then the males with longer tails will get to have more offspring. Over multiple generations, the average tail length of the population would increase, as generation after generation strive to outdo each other in tail length. This process steadily increases the average tail length of the group. It is a steady increase as the individuals only have to outdo each other by a small amount; as long as they have the longest tail they are at an advantage. There would be no advantage to a individual having a tail a lot longer than its rival, as the extra resources needed to create such a large tail could be used elsewhere. Also note that there is no intrinsic advantage to having a long tail, but there is an advantage to having the longest tail in the population. It's analogous to two countries with nuclear weapons programmes. Each side wants to ensure that it has slightly more weapons than the other, but not too many more as that money could be spent elsewhere. As a result, slowly over time, both sides steadily increase their nuclear arsenal.

Another common example of the arms race phenomenon in the natural world is the predator/prey relationship: the prey population evolves to become better at not being eaten, and the predator population evolves to become better at hunting (and therefore avoiding starvation). An arms race such as this, where each sides has a different goal is known as an *asymmetric* arms race [3]. The alternative to this would be a *symmetric* arms race which occurs when the individuals involved have the same goal as one another (as seen in the previous case of sexual selection by tail length).

Returning to the case of Hillis's Sorting Networks, by modelling an arms race in this way he was able to push his population to create a solution that had just 61 exchanges, 5 better than his previous traditional evolutionary approach [1d].

Co-Evolution Applied to the Bus Routing Problem [4]

To the author's knowledge [4] is the only available example of a co-evolutionary algorithm being applied to the Bus Routing Problem. In this section I will explain and discuss the approach that was used. I will first discuss how the problem itself was defined, followed by the approach used to solve it.

Fitness Measure

The fitness of a solution (the total route reward (TRR)) is defined as follows [4a] :

$$TRR = max(\sum_{r}^{R}\sum_{i=1}^{N}(customers[r_i - 1][r_i] * norm) - distances[r_i - 1][r_i])$$

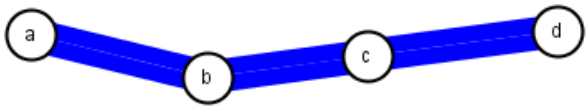
R is the number of routes, r is an individual route, N is the number of pick-up points for route r, r_i is the i^{th} pick-up point for route r, and norm is a normalising constant.

The above formula calculates the difference between the number of passengers wanting to use each route, and the distance they want to travel. This can be better understood if we imagine that each passenger is paying a fixed fare regardless of the distance they wish to travel, and that the bus has to pay a petrol cost that is directly proportional to the distance it has to travel. Each bus's reward can be thought of as its final net profit (the difference between money taken in fares and the cost of petrol). The Total Route Reward (TRR) is simply the summation of all the separate buses' profits.

Approximations

As admitted by the author, his approach to measuring the TRR makes some major simplifications.

Firstly, only customers travelling between neighbouring stops in a route are used in the calculation. For example, if we were to have the following route:



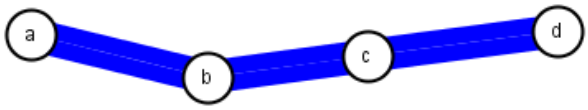
Using the TRR function described above, only passengers travelling from A to B, B to C, and C to D would be taken into account. The author acknowledges that this is naïve but justifies his decision with the following points [4b]:

"1. because the travelling customers are normally distributed and more importantly the main purpose of considering the customers in the TRR, it is as to create route of reasonable size, preventing overloaded route, while creating routes visiting the largest amount of customers wishing to travel short distances. Furthermore, most of the implemented techniques visit all the stops at least once and the route being connected allow any customer to reach its destination in more or less time. The implemented Total Route Reward support and promote (depending on the value of the norm) the routes solutions that visit the largest amount of customers from one stop to another

2. the realistic calculation cost too much time to compute. Indeed, for a route of size N and T being the total number of stops, instead of taking N-1 time steps as for the naive calculation, it will take N*(T-1) to complete the realistic calculation for one route (multiply this number by R routes and we obtain a substantial computational time)."

In the first point he states that it is important to prioritise the passengers wishing to travel short distances, and that he also wants to prevent an overloaded route. However, since the fitness function that is being maximised is only taking into account the demand at a very local level (and only those demands that happen to be connected by an edge) it isn't possible to tell whether the implementation of such a route in real life would be over loaded or not.

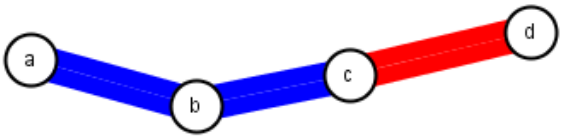
For example if, in the following example, D was a major city centre, and A, B and C were smaller outlying cities, we could imagine a lot of people in A, B and C wanting to travel to D:



The TRR function defined above would only take into account the demand between adjacent cities in the route, meaning only the passengers wishing to travel from A to B, B to C and C to D would be considered. However, in reality this route would also satisfy a lot of passengers travelling from A to D and B to D, but this would be ignored under this model.

The author states in his second point that the "realistic calculation" that considers the demand across the entire route would be highly computationally expensive and therefore difficult to model. Although it is true that a more realistic reward function would require significantly more processing time, it is crucial if we are to accurately rate the quality of a solution.

Another limitation of this approach is that it doesn't consider the interactions between the buses in the group. It is important that we consider how the amount of people wanting to use a bus is affected by other buses within the group. For example if we imagine in the above diagram that the bus route had stopped short of 'D', and that passengers travelling to 'D' were required to take a second bus to complete their journey like so:



Then the amount of people wanting to use the second route would be heavily influenced by the fact

that it is connected to the first route. This concept is made clearer if we imagine what would happen if the first route was removed; the amount of people using the second route would be reduced dramatically. It is perhaps useful to consider the analogy of pipes connecting water sources together. If each city is imagined as a water source, and the routes between them pipes, then the flow of water in one set of pipes is going to affect the flow of water in other sets of pipes connected to it.

As well considering the connections between the routes, we should also consider how routes overlap and conflict. The model described above doesn't take into account how multiple buses travelling along the same path would affect one another. It would be reasonable to assume that two buses travelling along the same path would each have fewer passengers compared to a single bus travelling along that path.

For example, if we imagine two cities A and B, with a demand of 10 between them, then the model in **[**4a**]** would assign each bus travelling between A and B with 10 passengers. This effectively means that the number of passengers travelling between two cities is proportional to the number of routes connecting them. Or to put it another way: under this model there are an unlimited number of passengers to go round. A more realistic model would assume there were a finite amount of passengers wanting to travel between each city, and distribute this demand evenly amongst the buses that are able to provide that service.

The Co-Evolutionary Solution

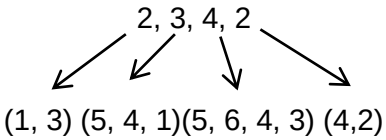
Firstly the author defines 3 populations [4c]. These populations are the routes, the depots, and the stops. Each member of the route population is a list of integers, with each integer defining the length of a route in the solution. Each member of the stop population is also a list of integers (each corresponding to a city on the map) whose order defines the sequence in which each city will be visited. And finally, each member of the depot population represents which stops will be used as depots. A complete solution is then created by taking a member from each population and combining them (each member is chosen using a defined selection process such as tournament selection). For example:

If the following representatives from each population were chosen:

Route Population Representative: 2, 3, 4, 2
Stop Population Representative: 1, 3, 5, 4, 1, 5, 6, 4, 3, 4, 2
Depot Population Representative: 3, 5, 4

They would be combined to create a final solution as follows:

Step 1 - Split Stop Population Representative according to Route Population Representative:



Step 2 – Assign Depots:

(1, **3**) (**5**, **4**, 1)(**5**, 6, **4**, **3**) (**4**,2)

Step 3 – Assign fitnesses: each individual representative is assigned a fitness value proportional to how well their combined final solution performs.

This co-evolutionary approach differs to the co-evolution described previously [1], which considered individuals in conflict with one another. This approach is instead better thought of as a form of cooperative, rather than competitive co-evolution. The author states that competitive co-evolution is less intuitive [4d] for the Bus Routing Problem, and has therefore focused his efforts on this cooperative approach. I would disagree with this, and intend to outline and investigate an approach that aims to harness the power of competitive co-evolution in routing problems such as these.

Conclusion

The function to be maximised as used by [4] doesn't realistically model the reality of a bus system. In particular, it doesn't take into account:

- The flow of passengers along a single route beyond more than one stop (i.e. it only considers the flow between the current stop and the next).
- How separate buses connect and the effect this has on the flow of passengers.
- How the demand between two cities should be spread amongst multiple buses providing the same service.

Aims and objectives

The following are the aims and of objectives of this project:

1. Define and implement a representation of the Bus Routing Problem that overcomes some of the approximations made by the previous co-evolutionary model. In particular, the representation should accurately simulate both the behaviour of passengers and the interactions between individual buses within the service.
2. Investigate a bottom-up, multi-agent co-evolutionary algorithm that takes its inspiration from group behaviour in the natural world, and measure its performance on the Bus Routing Problem. To measure its performance, a range of variations to the approach will be implemented so that comparisons can be made.
3. Develop an effective schematic representation of problems/solutions that allows the user to identify the strengths and weaknesses of the current solution visually.

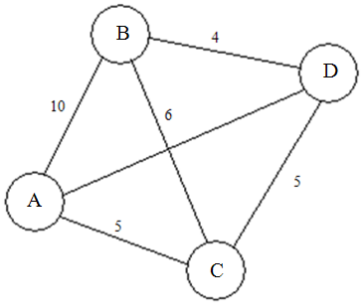
Definition of the Bus Routing Problem and Visualisation

The Bus Routing Problem can be represented as two matrices: the 'Demand Matrix' and the 'Distance Matrix'.

Distance Matrix

The Distance Matrix contains all the costs of travelling between each pair of cities. This cost could be the time it takes to get between each city, or some other cost such as the length of the road directly connecting them. I will imagine it to be the distance in Euclidean space between the two cities, as this can easily be visualised graphically. I will also assume that the distance is equal in both directions i.e. The Distance Matrix is symmetrical. For example:

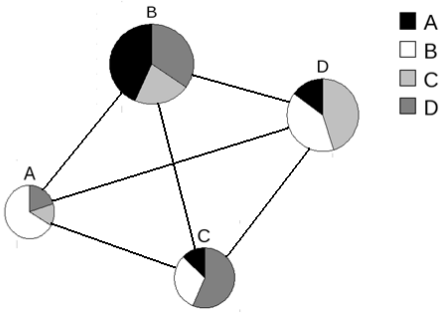
	A	**B**	**C**	**D**
A	0	4	6	10
B	4	0	5	6
C	6	5	0	5
D	10	6	5	0



Demand Matrix

The Demand Matrix represents the amount of people who want to travel between each pair of cities. This is to be thought of as the average amount of people wanting to travel between each pair of cities over the course of a day. It is reasonable to assume that the demand is roughly equal both ways, as presumably the majority of people will be purchasing return tickets, and therefore the Demand matrix is also symmetric.

	A	B	C	D
A	0	10	2	3
B	10	0	5	8
C	2	5	0	9
D	3	8	9	0

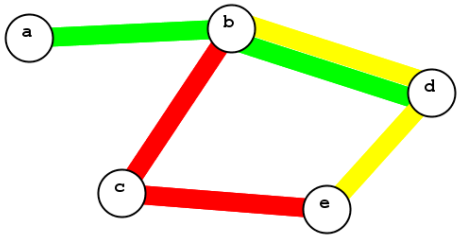


The Demand Matrix can be visualised by depicting each city as a pie chart. Each pie chart's total area is proportional to the total amount of demand at that city (the sum of that city's column in the Demand Matrix). The relative proportions of the destinations of the passengers leaving each particular city are represented as the separate segments. For example, B's pie chart shows that roughly 1/3 of the people requiring a bus service from city B are heading to city D (represented by the dark grey segment).

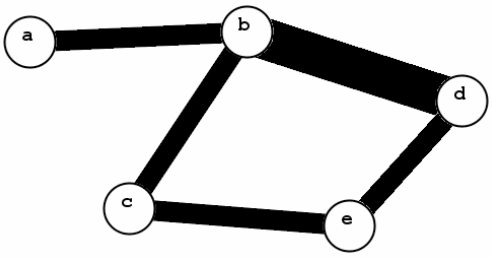
Representing a Solution

A solution to the Bus Routing Problem would consist of a set of routes, with each route having its own dedicated path on the graph. An individual route would consist of an ordered set of edges connecting a subset of the cities together. For example:

Solution A: [$\mathbf{R_1}$[a, b, d], $\mathbf{R_2}$[b, d, e], $\mathbf{R_3}$[b, c, e]]



The diagram above shows 3 routes on a map of 5 cities. The thickness of each route in the diagram can be thought of as the frequency of the bus, and therefore the total area of the route represents how many people can be transported along that route over the course of a day. To better understand this representation it is helpful to think of the routes as pipes, and the cities as sources of water. The amount of water each pipe is capable of transporting is proportional to its width. Using this representation, we can then consider the *combined* width of all the pipes between two particular nodes. This combined thickness represents the total amount of people it is possible to transport between the two cities in a day, or put another way, the overall potential of the *entire* population of routes. This is made clearer by representing the routes in the same colour:



The above diagram is a graphical representation of what I will refer to as the Service Provided Matrix. Each value in this matrix represents how many people it is possible to transport between two cities, which is equal to the combined thickness of all the edges between those two cities. It can be calculated as follows:

```
initialise all values of ServiceMatrix[][] to 0

for every pair of cities a, b
      for every route r whose path includes the edge between a, b
            ServiceMatrix[a][b] = ServiceMatrix[a][b] + r.getFrequency()
```

Modelling the Flow of Passengers

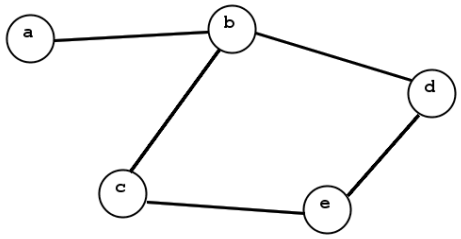
In order to accurately model the Bus Routing Problem we must implement a model of the flow of people between each city, or to carry on with the water analogy: we must determine how much water is flowing through each pipe at any one time. I will do this by defining the Flow Matrix.

The Flow Matrix represents the amount of people needing to travel along each edge of the graph in order to complete their journey. For example, if we imagine in the above example that the demand between city 'a' and 'b' is 10, then we know there are at least 10 people who need to travel along the edge (a – b). If we sum all the demands from 'a', and all the demands leading to 'a' from the other cities, then we will have the total flow of people for the edge a – b. The entire Flow Matrix can be computed as follows:

```
initialise all values of FlowMatrix[][] to 0

for every pair of cities a, b
      R = shortestRoute(a, b)
      for every edge_x,y in R
            FlowMatrix[x][y] = FlowMatrix[x][y] + Demand[a][b]
```

The shortest route method returns the set of edges making up the shortest route between cities 'a' and 'b' using Dijkstra's algorithm. This method doesn't take into account changing buses; it simply considers the shortest route along the 'flattened' graph created by all the routes in the group. The above example 'flattened' would be:



Dijkstra's Algorithm [5] works by constructing the tree of minimum total length between the nodes and selecting the shortest route from there. It is guaranteed to find the shortest route between nodes (if a route exists).

Once this has been computed, we can visualise the flow along each edge in the same way we visualised the potential for transporting people along each edge: plot each edge with a thickness proportional to its flow value.

If we were to then superimpose this flow graph onto the graph showing the routes, we could see how well the routes cope with the demand. For example if we plotted the flow in black and the routes in grey, the following edge would represent an edge between two cities that is being under used:

Using the water analogy, the above representation shows a pipe (grey) transporting water (black). If the demand was greater than the routes along that edge could cope with we would get:

Again to use the water analogy; it is as if the water has burst the pipe. Here is an example of a schematic representation of the Service Provided Matrix superimposed onto the Flow Matrix:

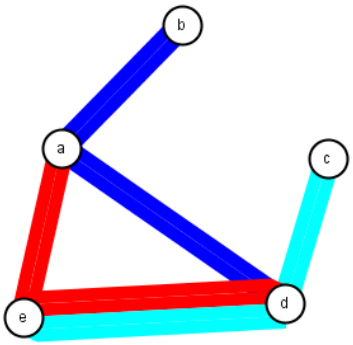


fig.1 Routes

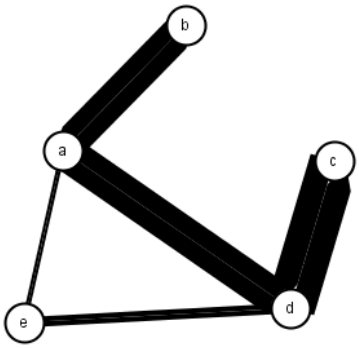


fig.2 Flow

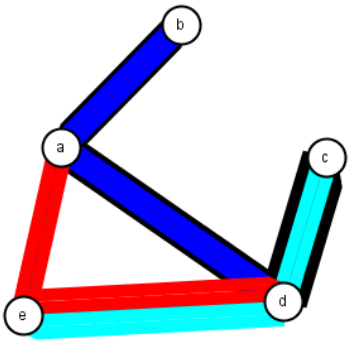


fig.3 Routes superimposed on Flow

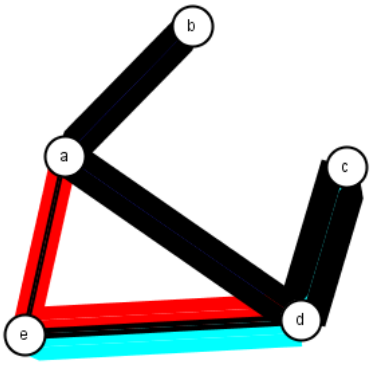


fig.4 Flow superimposed on Routes

We can see in fig.3 above that there is an overflow of passengers along edges b-a, a-d and d-c. This is represented by the black edges showing either side of the routes. Correspondingly, we can see from fig.4 that edges a-e, and e-d are being underused.

Usage Matrix

Once we have the Flow Matrix, the Usage Matrix can be calculated. The Usage Matrix is a numerical representation of the overflow/underflow diagrams shown above. The purpose of the Usage Matrix will become clear in the next section when we look at how to rate the quality of a solution. The Usage Matrix is the ratio of the Flow Matrix and the Service matrix and can be calculated by simply dividing the Flow Matrix by the Service Matrix. A value in the Usage Matrix greater than 1 represents an edge that has an overflow, and a value less than 1 represents an edge that has an underflow.

Rating the Quality of a Solution

I will rate the quality of a complete solution in the following three ways:

- The total profit made.
- The total number of customers satisfied.
- The average journey length of a customer.

In this section I will describe how these values will be calculated.

Profit

The profit of an individual route can be thought of simply as follows:

$$R_{profit} = R_{ticketSales} - R_{petrolCosts}$$

$$R_{ticketSales} = \sum_{\forall edge_{a,b} in R} Usage_{a,b} \times Freq \times Distance_{a,b} \times costPerMeter$$

$$R_{petrolCost} = \sum_{\forall edge_{a,b} in R} Freq \times Distance_{a,b} \times costPerMeter$$

Note that Usage values greater than 1 will simply be rounded down to 1 when calculating Ticket Sales.

It should be noted that although [4] used a set ticket price no matter the length of the journey, I am implementing a system that sets the ticket price directly proportional to the length of the journey, with no maximum value. In reality this is more in accordance with buses travelling on a national scale, where ticket prices often vary significantly depending on the distance of the journey.

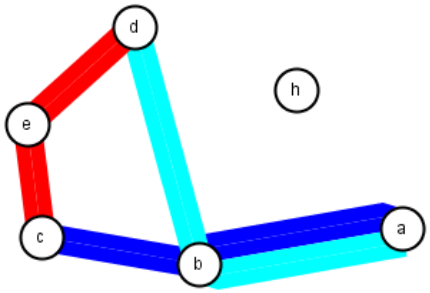
The total profit of the entire solution is simply the summation of the profits of each individual route:

$$TotalProfit = \sum_{R \in Routes} R_{profit}$$

A good solution will maximise on this function.

Total Number of Customers Satisfied

As well as making a profit it is also important that a bus service satisfies the demand of the customers wishing to use it. Previously we calculated the flow of customers by tracing the shortest route around the graph that would get them to their destination. I will define a 'Satisfied Customer' as someone whose route doesn't contain any edges with an overflow of customers. That is to say, any customer whose route doesn't contain an edge with a Usage value greater than 1. Any customer who isn't able to find a route to their desired destination won't be considered as satisfied either. For example, in the following diagram none of the passengers leaving from city 'h' would be classed as satisfied:



The total number of satisfied customers can be calculated using the following algorithm:

```
initialise totalSatisfiedCustomers to 0

for every pair of cities A, B
      R = shortestRoute(A, B)
      if (R != null and satisfiedRoute(R) = true) then
            totalSatisfiedCustomers = totalSatisfiedCustomers + demand[A][B]
end

Sub satisfiedRoute(R) returns boolean
      for every edge_x,y in R
            if (usage[x][y] > 1) then return false
      end

      return true
end sub
```

Average Journey Length

As well as ensuring that each customer is satisfied with the service provided as described above, we can also calculate the average length of journey and use this as a measure of the quality of a solution. The Average Journey Length can be calculated using the following algorithm:

```
initialise totalJourneyLength to 0
initialise totalNumberOfCustomers to 0
initialise journeyLength to 0

for every pair of cities a, b
      R = shortestRoute(a, b)
      journeyLength = 0
      for every edge_x,y in R
```

```
            journeyLength = journeyLength + Distance[x][y]
        end
        totalNumberOfCustomers = totalNumberOfCustomers + Demand[a][b]
        totalJourneyLength = totalJourneyLength + (journeyLength * Demand[a][b])
end

averageJourneyLength = totalJourneyLength / totalNumberOfCustomers
```

An Emergent Co-evolutionary Approach to the Bus Routing Problem

Previously in this report we looked at an approach to the Bus Routing Problem that was inspired by the co-evolution observed in nature [4]. This particular approach created 3 populations, each containing a different element needed for a complete solution. To create a final solution, a representative was selected from each population, and combined. This was described by the author as “collaborative co-evolution” due to the fact that each population was reliant on the others in order to be able to create final solution.

The aim of my project is to implement and investigate a bottom up, competitive co-evolutionary approach to the Bus Routing Problem. The Bus Routing Problem is a complex problem whose solution is made up of many smaller interacting parts (the individual routes). In the previous section I defined how the effectiveness of a complete solution could be assessed. I now want to consider what is going on with each individual route within the group, at what could be called the 'Route level', and discuss how the effectiveness of each individual can be calculated.

An Individual at the Route Level

The distinction I am making between the “Group Level” and the “Route Level” parallels the difference in nature between the herd and the individual. Since Darwin published his theory of evolution by natural selection there has been great debate as to at what level evolution acts. A fundamental problem that needed to be reconciled was that of altruistic behaviour. If, as Darwin proposed, evolution by natural selection was about survival of the fittest, why would altruistic behaviour evolve? Surely any individual who didn't act altruistically would have a higher chance of survival over its altruistic peers, and would therefore prosper within the population. Initially biologists tried to answer this by stepping back and looking at what was going on at the group level. For example, when trying to explain why worker bees ignored their own need to reproduce, and instead dedicated their lives to aiding the queen in reproduction, it was thought that they were acting “for the good of the group”. This “group selection” idea proposed that the reason this altruistic behaviour had survived was because the individuals acting in this way had ensured the survival of the entire species over rival species.

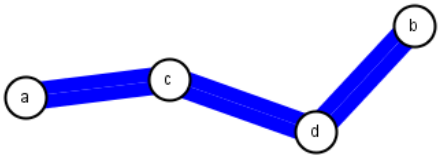
It is now in fact more widely believed that in order to explain altruistic behaviour we must, instead of zooming out and looking at the group level, zoom in past the individual level to the gene level. The model of evolution at the gene level [6] proposes that it is in fact the individual genes that are the fundamental agents of evolution, and that the bodies in which they are contained are simply vehicles to aid their reproduction. Due to the fact that copies of the same genes can exist in separate bodies, at this level it is easy to explain altruistic behaviour. What at the individual level appears to be altruistic is in fact selfish at the gene level. For example, the reason a parent looks after its offspring is because the genes in the parent recognise copies of themselves in the offspring, and have evolved in order to protect those copies.

How does this apply to this project?

A more conventional evolutionary approach to the Bus Routing Problem would create a population of complete solutions to the Bus Routing Problem. These complete solutions would be assigned a fitness value, and then selected for reproduction and deletion by comparison with one and other. In the real world, this is akin to deciding the fate of an entire group based on the combined performance of all the individuals within it, and comparing this to the combined performance of other groups. Instead of this, I want to model a *single* group of individual routes and apply selection and deletion at this individual/route level. I am interested to see whether by ranking and selecting at this individual level, we can observe emergent adaptive behaviour at the group level (i.e. a complete solution to the problem). In this sense, my approach is more akin to the artificial agents often used in Artificial Life, rather than Evolutionary Computing.

Encoding an Individual

An individual route is made up of a path around the graph and a frequency value. A very straight forward approach to encoding the path of a route would be to simply have a list of the cities along the route in the order that they would be visited. I will call this a ***node representation***. For example:



would be encoded
[A, C, D, B]

Note that under this encoding the edges of the route are implicit: that is to say they are a result of the ordering of the list, and if we were to change that order the information would be lost. It will be assumed that at the end of the route (in the example above city 'B') that the bus will simply turn around and go back the other way, meaning that any route that can transport someone from A to B can also do B to A.

A different approach to a node representation would be to explicitly encode the edges using an ***edge representation*** like so:

[(A-C)(C-D)(D-B)]

As we can see, each gene in the encoding is now a pair of cities linked (an edge) and it wouldn't matter what their individual position within the chromosome was, we'd still draw the same path on the graph. (Note that although order doesn't matter, in order to be valid the path must still be unbroken when drawn on the graph of cities).

The frequency of the route can simply be represented as an integer value, where the higher the integer is the more frequent the buses are along that route.

Breeding

Selection

'Simple' Selection

Simple selection sorts the routes into order based on the fitness that they are being measured on, and selects either the best or worst for breeding or removal respectively. For example, if we were to select based on profit then the two routes with the highest profit would be selected for breeding and their offspring would replace the route with the lowest profit. This method is very simple to implement, but gives considerable bias to the high earning routes. For example, a smaller route might be running at full capacity but wouldn't be selected as it shares the environment with much larger routes.

Roulette Selection

Roulette selection works by assigning each route a segment of a “roulette wheel” the size of which is proportional to the fitness of that route: the higher the fitness, the larger the segment of wheel that is assigned. For example is we had a very small population of 3 routes, one making a profit of 100 and the other two of 50, then the first route would have a 50% chance of being selected and other two 25% each.

Reproduction

Cloning

A very straight forward approach to reproduction would be to simply clone the selected parent. If we were to use cloning then it would be crucial to have mutation. Without mutation we would be limited as to what parts of the search space we can reach. It should be noted however that unlike when using traditional evolutionary algorithms, there could still be improvements to the solution using cloning without mutation. It should always be remembered that rather than breeding whole solutions, we are working at the “Route level”, and every route affects every other route, and it is their combined performance that creates the overall solution. However, if we *were* to use cloning without mutation, then only the frequencies of the buses along

each edge could change, meaning only the amount of Satisfied Customers and Profit would adapt. In order for the Average Route Length to adapt we would need a method that also shuffles the genes throughout the generations, in order to fully explore the solution space. This could either be done by mutation or crossover.

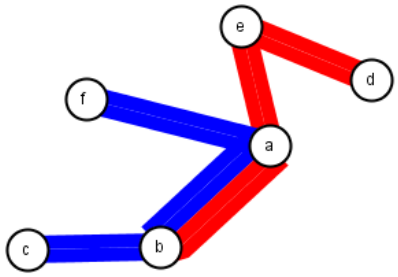
Crossover

Unlike the Travelling Salesmen Problem discussed, as long as the path is unbroken, there are no other restrictions on what is a valid path for a route. It can visit any number of cities, and can visit the same city as many times as it likes. This means using the simple node representation discussed previously there are no invalid encodings, making crossover a lot more straight forward than it would be for a permutation problem**.**

One Point Crossover (1X)

Node Representation

One point crossover works by splitting each parent chromosome in two, and then combining the first half of one with the second half of the other, forming the offspring. The point at which to split the parents is chosen at random. For example, using a node representation we could have:

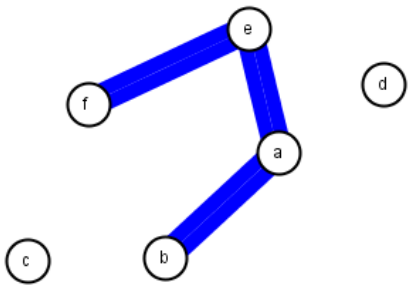


P1: [F, A, B, C] P2: [D, E, A, B]

There are 3 potential crossover points. One of these would be chosen at random. For example, if we chose the first crossover point we would get:

P1: **[F] [A, B, C]** P2: **[D] [E, A, B]**

Offspring1: **[F, E, A, B]**



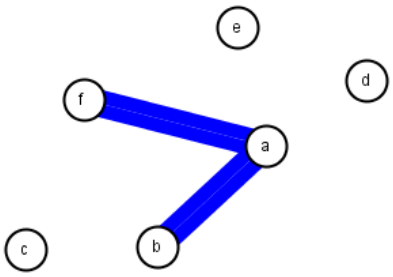
1X has considerable *positional bias* [7]. A positional bias means that the order in which the genes appear in the chromosomes is highly conserved between the parents and the offspring. We can see this in the phenotype of the above example: only one new edge (F-E) appears in the offspring that wasn't in either parent.

Adjusting 1X

A slight variation on 1X would be to choose a different crossover point for each route. For example:

P1: **[F] [A, B, C]** P2: **[D, E] [A, B]**

Offspring1: [F, A, B]



Parent 1 is split at point 1, and Parent 2 at point 2. Although this is a subtle difference, it means that offspring can be produced of varying length (we can see that in the example above the offspring is smaller than either parent). Using the standard 1X crossover described previously, it would be impossible for the offspring to be longer than the longest parent.

Edge Representation

Because of the explicit nature of edge representation applying 1X requires a little more care. Using the previous example but this time with edge representation we would get:

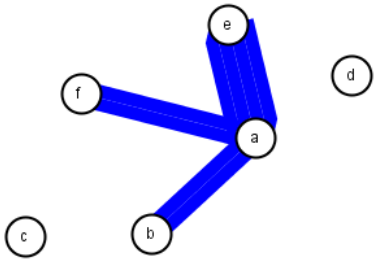
P1: **[(F-A), (A-B), (B-C)]** P2: **[(D-E), (E-A), (A-B)]**

Let us assume the crossover over point is again the first we would have:

P1: **[(F-A)] [(A-B), (B-C)]** P2: **[(D-E)] [(E-A), (A-B)]**

Offspring1: **[(F-A)(E-A)(A-B)]**

This is an invalid route as it is not one continuous unbroken path. To compensate for this, an edge must be added after crossover to ensure the route is valid:

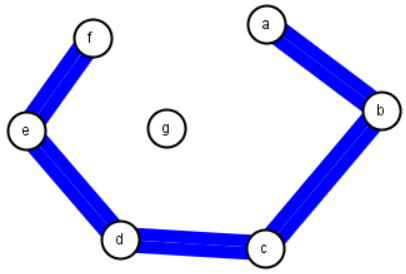


Offspring1: [(F-A)**(A-E)**(E-A)(A-B)]

Mutation

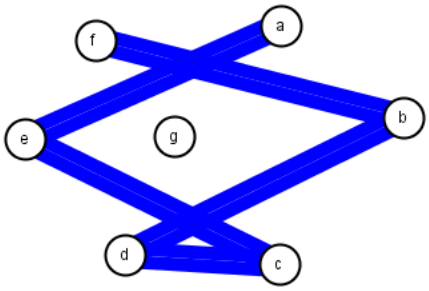
Swap Mutation

Swap mutation is often used in permutation problems where it is essential in order to ensure the mutated chromosome is still valid. It is carried out by selecting two genes, and swapping their position within the chromosome. For example:



[A, B, C, D, E, F]
Before

[A, **B**, C, D, **E**, F]
Swap 2 and 5



[A, E, C, D, B, F]
After

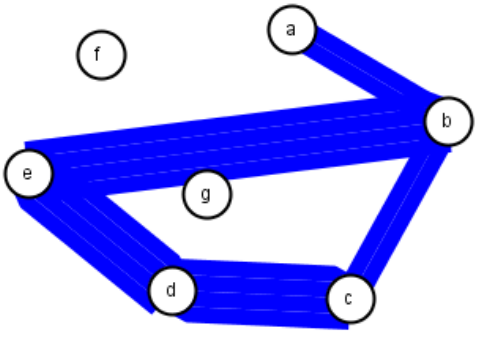
Note that although only two genes have been swapped, four new edges have been introduced into the final phenotype. This is clearer using the more explicit edge representation as discussed previously. The above example using an edge representation would be as follows:

[(A-B), (B-C), (C-D), (D-E), (E-F)]
Before

[(A-B), **(B-C)**, (C-D), (D-E), **(E-F)**]
Swap 2 and 5

[(A-B), **(E-F)**, (C-D), (D-E), **(B-C)]**
After

Currently this mutated chromosome is broken and therefore invalid, it needs new edges to be added to ensure it is valid:



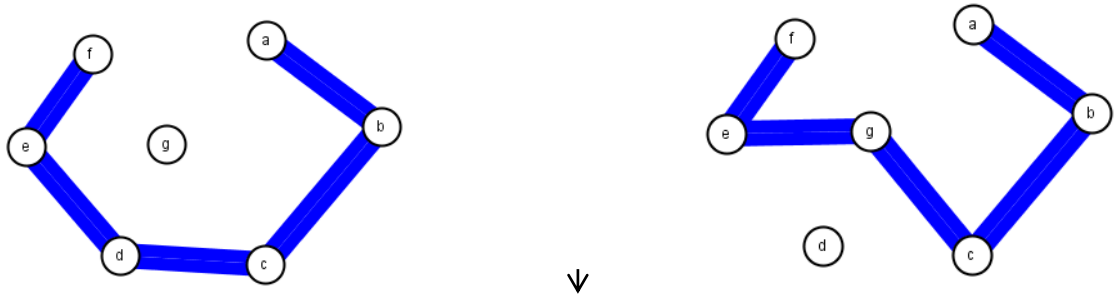
[(A-B), **(B-E),** (E-F), **(F-C)**, (C-D), (D-E), **(E-B)**, (B-C)]
Corrected Chromosome

This correcting process is only necessary because of the explicit nature of the representation, under the node representation this “correction” is covered by the mapping of genotype to phenotype.

Point Mutation

Node Representation

Point mutation simply selects a gene at random within the chromosome and changes it for another valid gene. For example:



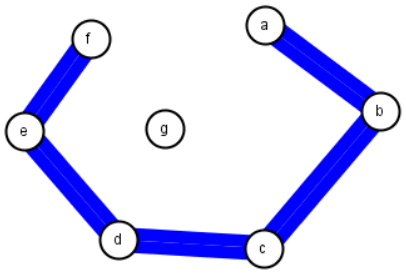
[A, B, C, D, E, F]
Before

[A, B, C, **D**, E, F]
Mutate position 4

[A, B, C, G, E, F]
After

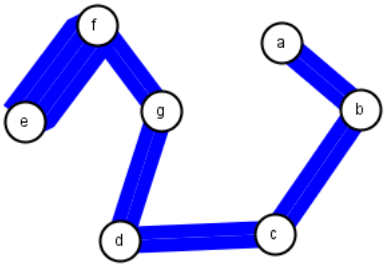
Edge Representation

When using an edge representation, point mutation isn't as straight forward. Each gene in the chromosome is an edge, rather than a city, meaning there are far more alternatives it could be replaced with. If there are n cities then there are n^2 edges to consider. As well as this, in order for the solution to remain valid we have to correct the edges on either side of the mutated chromosome to ensure the path remains unbroken. For example:



[(A-B), (B-C), (C-D), (D-E), (E-F)]
Before

[(A-B), (B-C), (C-D), **(D-E)**, (E-F)]
Mutate position 4



[(A-B), (B-C), (C-D), **(D-G), (G-F), (F-E)**, (E-F)]
After (with correction)

Edge Representation Vs Point Representation

The explicit nature of Edge Representation means that it requires a lot more intervention in the form of corrections than point representation does. We saw above that when performing crossover and mutation, there was a high chance that the offspring would be invalid. In reality there is very little difference between the two representations and there are no routes that can't be represented by both. However, I think the point representation will be simpler to implement, require fewer corrections, and allow much less destructive mutation to occur. For these reasons I will be using Point Representations for my implementation.

Point Mutation vs Swap Mutation

There are some crucial differences between point and swap mutation. In the example given above, city G wasn't in the original chromosome. If we were to purely use swap mutation there would be no way for the 'G' gene (or any other gene that wasn't already contained within an existing chromosome) to appear in the chromosome. If the 'G' gene didn’t appear in any of the routes in the population, then there would be no way for it to appear. All we can hope for under swap mutation is a rearranging of the order of the cities in each chromosome. It should be noted however that by rearranging the cities in this way it is possible to have new *edges* appear in the gene pool, however none of these new edges would contain new cities. This is a major disadvantage, as it means once a gene is lost from the gene pool (i.e. all the routes containing that city die) then there is no way for it to return. This would cause populations to get stuck in parts of the solution landscape that it can't get out of. Point mutation over comes this problem of genes going extinct, by replacing the selected gene with potentially any city on the map.

The other difference that should be considered between the two mutation types is how destructive they are. For example, a single swap mutation in a chromosome of length 6 (as shown in the examples above) changes 5 of the edges, leaving only 1 of the edges from the original non-mutated chromosome. Under point mutation on the other hand, a single point mutation changes only 2 edges. Mutation often has the effect of fine-tuning solutions; if it is too destructive then it will be difficult for this to occur.

Experiments

Avoiding Bias

Averaging Multiple Runs

I intend to run a number of experiments, each with a variation on the methods used for selection, crossover and mutation. Each run will be over 300 generations, and for each generation I will record the Profit, Satisfied Customers and Average Route Length. I will carry out 500 runs for each experiment, and when completed take the average for each generation of the 3 values being recorded. By doing this I will reduce variations caused by chance, improving accuracy.

Creating Random Demand Matrices

I will create each Demand Matrix randomly using an algorithm. The following algorithm does this by taking as its input the total number of passengers to distribute around the matrix, the amount of items to fill in the matrix, and the number of empty ('0') items in the Matrix:

```
CreateRandomDemand( int totalPassengers, int totalItemsToFill, int noEmptyItems)
	maxValue = totalPassengers / ( totalItemsToFill – noEmptyItems )
	for every noEmptyItems
		choose random position in DemandMatrix
		set position to '0'
	end
	for all remaining positions
		choose 2 positions X and Y that have not been chosen before
		set X to a random value between 1 and maxValue
		set Y to (maxValue – X value)
		Set the mirror of X and Y to the same values as X and Y
	end
end
```

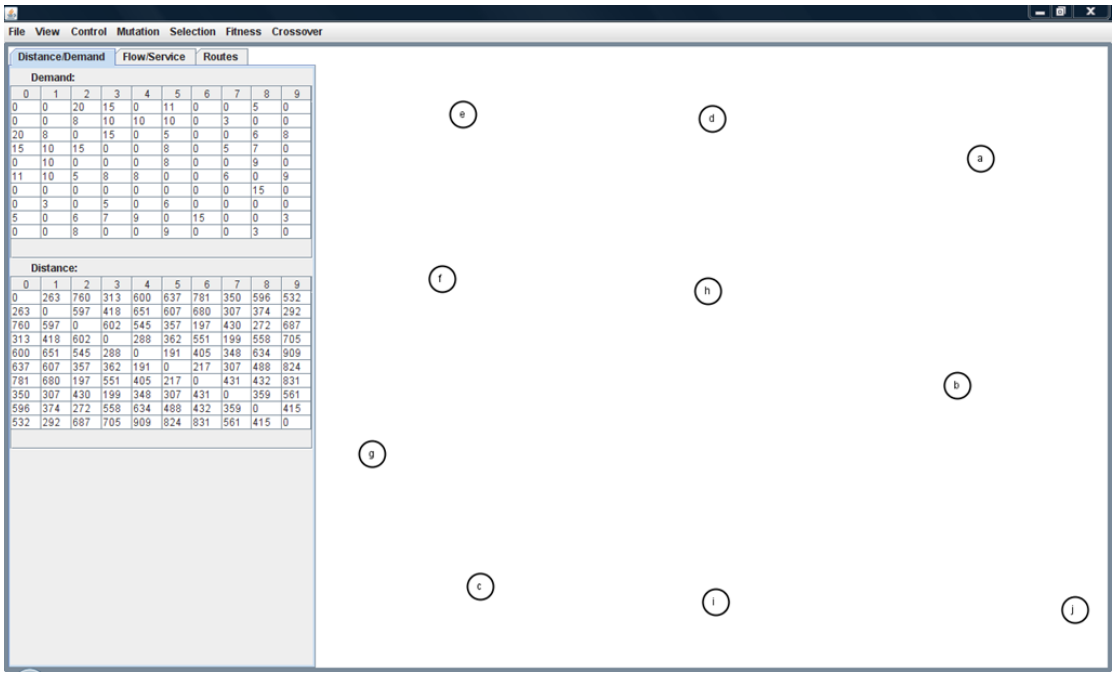
Initially the above algorithm works out the maximum value that a Demand Matrix value can take assuming the passengers are spread equally between the non-empty items in the Demand Matrix. Then, in the first loop, the algorithm sets the required amount of '0' positions. Following this, in the second loop, the algorithm picks pairs of items from the Demand Matrix, assigns the first a value between 1 and the maxValue. The second value is then set as the remaining value between the first items value and the maxValue. To ensure that the matrix is symmetrical the same values are assigned to the corresponding 'mirror values' of X and Y. This approach, of picking pairs and assigning complementary values that sum to a set value, ensures that overall the matrix will contain the right total amount of passengers. For example, if we were to just pick single items and set them a value between 0 and a maximum value, then it is highly unlikely that each randomly created matrix will have the same amount of total passengers.

I will create a new random Demand Matrix for each of the 500 runs (of 300 generations) that make up each experiment. Each new Demand Matrix will contain the same total number of passengers, the same minimum/maximum number of passengers at each city, and the same number of empty (0) items in the Matrix. Only the distribution of the passengers will change as described above. By redistributing the Demand Matrix for each run, I will reduce the bias that each specific Demand Matrix might have for each method.

Previously in this report I discussed the 3 values I would be plotting during each experiment. They were Profit, Satisfied Customers and Average Journey Length. We could imagine that the priority of the bus company is the Profit value (as they want to stay in business), and that the priority of the customers is the Satisfied Customers value (as they want a decent bus service). The Average Journey length would be a priority for both, as it closely linked to both Profit and Customer Satisfaction.

The aim of the following experiments is to investigate what selection and breeding methods at the individual level, best adapt the 3 values at the group level. For each experiment I will be using a map of 10

cities as can be seen in this screen shot:

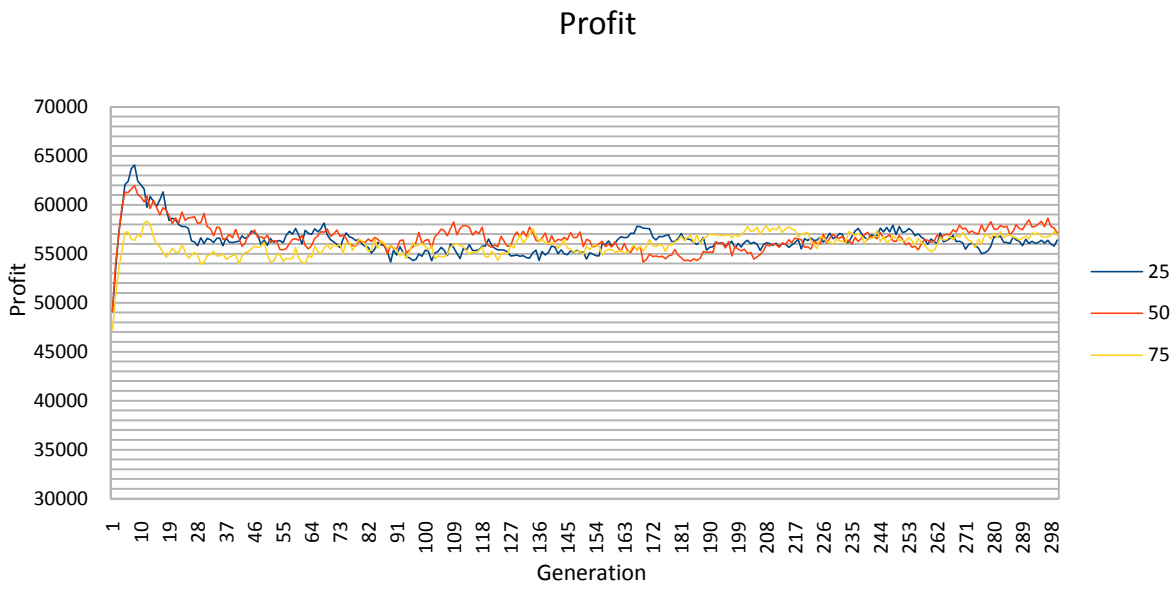


I will be using a fixed group size of ten routes, and a steady state process of removing one route from the group and creating a new route to replace it at each generation. The initial routes will be generated at random, and will have a length between 2 and 4 edges**.** As described previously, each run will be over 300 generations, and will be carried out 500 times and the results averaged. Each run will start with a new random population and a new "redistributed" Demand Matrix. Note however the Distance Matrix will remain constant.

Profit

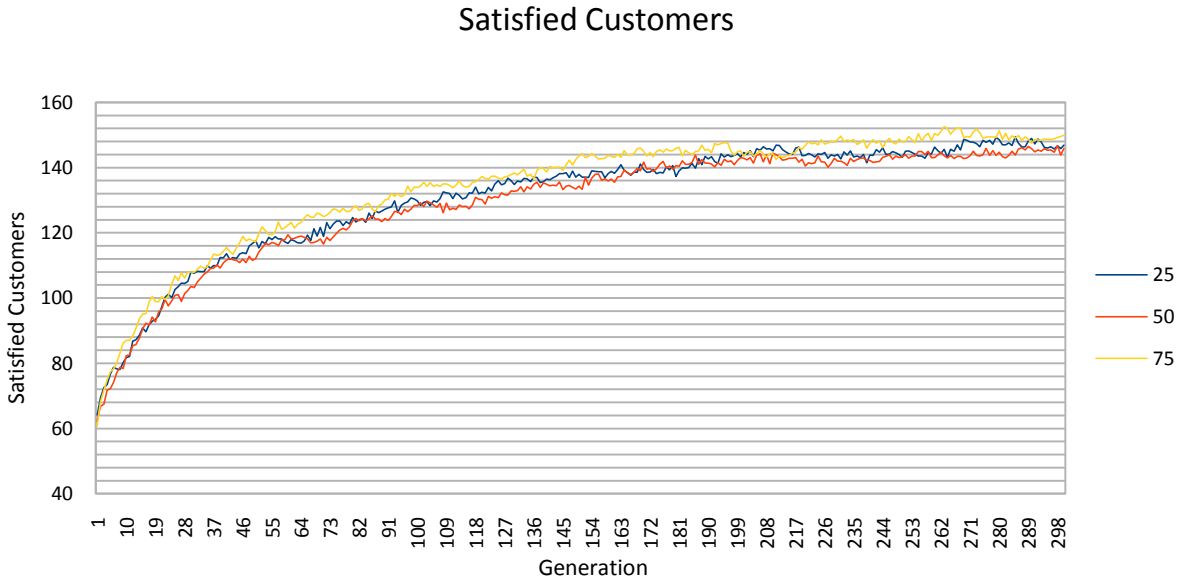
Experiment 1 – *Cloning using simple selection based on profit.*

For the first experiment I am going to select based on the profit using Simple Selection. I will remove the route in the group with the lowest profit and clone the route with the highest profit. I will do the experiment 3 times with varying chances of Point Mutation occurring: 25%, 50% and 75%.

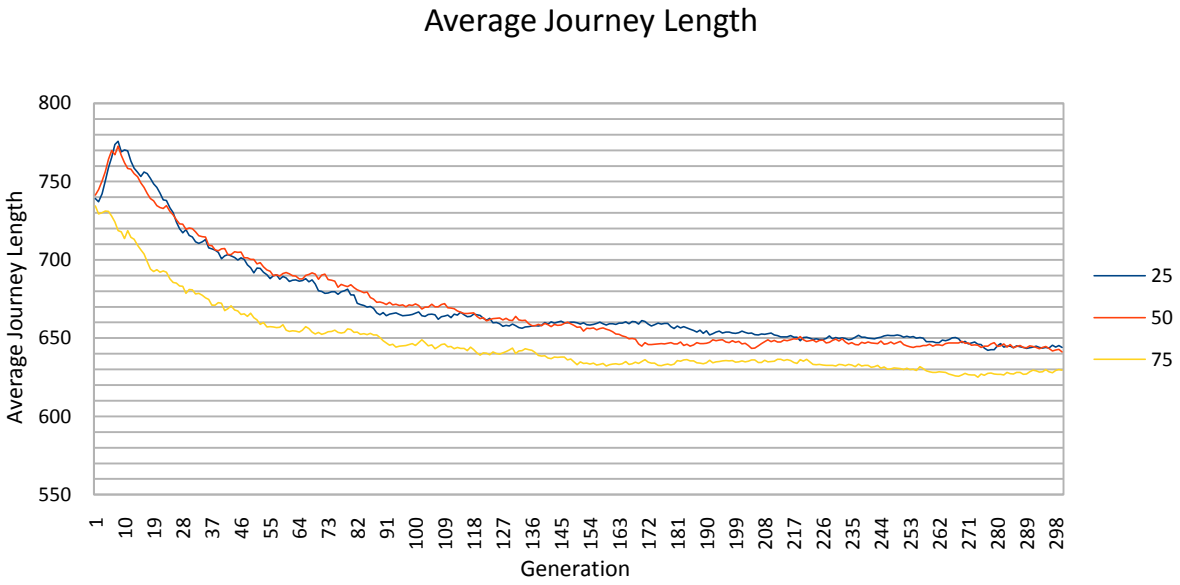


The results were as follows:

We can see in the graph above the overall profit of the group starts at just below 50,000 before quickly increasing to over 60,000 and then slowly declining and settling just over the 55,000 mark. There is very little difference between the 3 levels of mutation, apart from the peak at the start where 25% appears to peak slightly higher than the other two.



We can see from the graph above that the amount of Satisfied Customers steadily increases over the generations before settling at the 140-150 range. There is again very little difference between the mutation levels.



Overall the Average Journey Length decreases across the generations. However, initially there is a small increase. Interestingly this peak doesn't occur when there is a higher chance of mutation (75%).

Analysis of Experiment 1

The initial peak in profit is due to the population maximising on the initial excess of resources (an overflow of passengers). Remember that the passengers are taking the shortest routes they can to their destination, so it is likely that any route that is making a profit is a route that is in high demand, and is therefore likely to have an overflow. In the early generations, when the most profitable route is cloned, it is likely that the offspring will be able to use enough of its parent's overflow to also make a profit. However, this excess of resources is always going to be short lived as there are only a finite number of passengers. This then accounts for the slow decline in profit. In effect, by cloning the most profitable routes we are doubling the overheads for that path in the graph. Unless there is enough of an overflow along that path to justify this increase, then the offspring will have the effect of reducing its parent's profit. To use an analogy from the real world, it is as if there isn't enough food to feed both the offspring and the parents, and so by having offspring the parents are effectively reducing their own chances of survival.

If we were to imagine the fitness landscape with each individual route plotted on it, then the route with the highest profit would be on the highest peak. When this route is cloned, its offspring is likely to be plotted very nearby (or possibly in exactly the same position if mutation didn't occur). If the offspring *is* identical, then the passengers that were previously all using the parent route are now going to be split between the parent and the offspring, drastically reducing the profit of the parent. In terms of the fitness landscape, it is as if the high peak that the parent was initially on has slumped under the weight of both the parent and its offspring. This can be thought of as a form of "overcrowding".

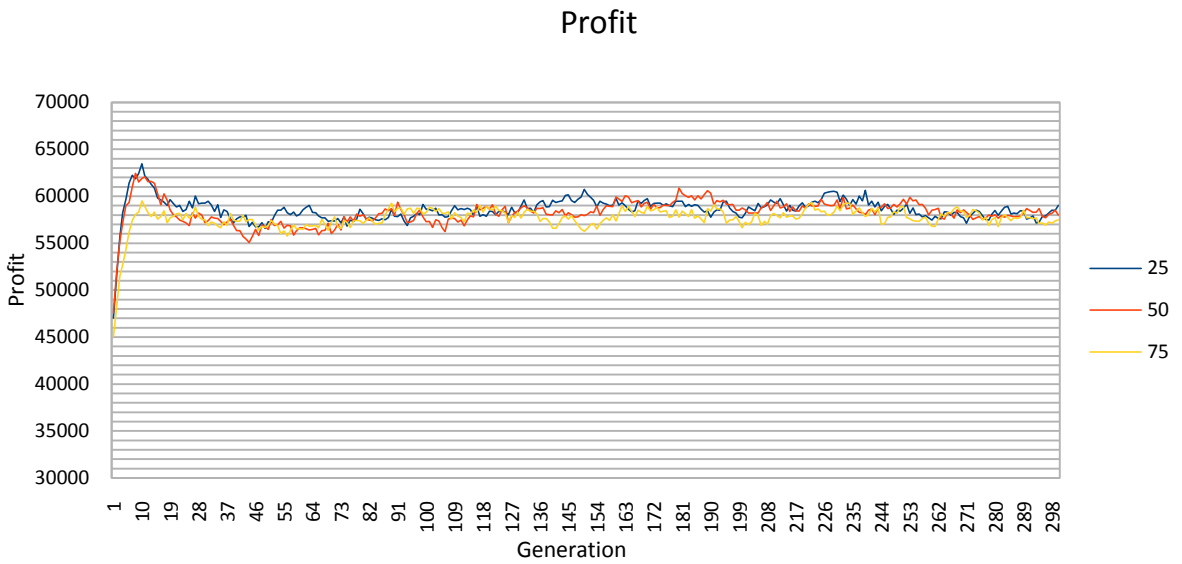
This phenomenon of overcrowding is reflected in the Satisfied Customers graph. To recap, a Satisfied Customer is one who has no over flow on the route he is travelling along. Initially there is a large amount of overflow in the population (an excess of resources) and therefore the number of Satisfied Customers is low. As the profit increases, and the amount of overflow decreases, in turn the amount of Satisfied Customers also increases.

The Average Journey Length can be seen to very slowly decrease by about 100 over the 300 generations. The initial increase can be explained as a result of how the calculation is carried out. Only the customers who have a complete route are considered under the Average Journey Length calculation. Therefore any cities that initially aren't linked into the Bus Service are ignored, giving a low Average Journey Value. Very quickly these cities are "discovered" by the new routes being created, and therefore become included in the calculation, creating a peak in the graph. The slow decline in the Average Journey Length after the peak, is a result of the selection pressure being applied by the passengers. The passengers are selecting the bus routes that provide the shortest path to their destination, and therefore any offspring that "discovers" a shorter path via mutation will be more successful than its parent. This explains why the highest level of mutation (75%) has a quicker descent than the lower levels of mutation. The higher mutation rate is adapting (and therefore exploring the route landscape quicker) than the others, and hence it finds the shorter

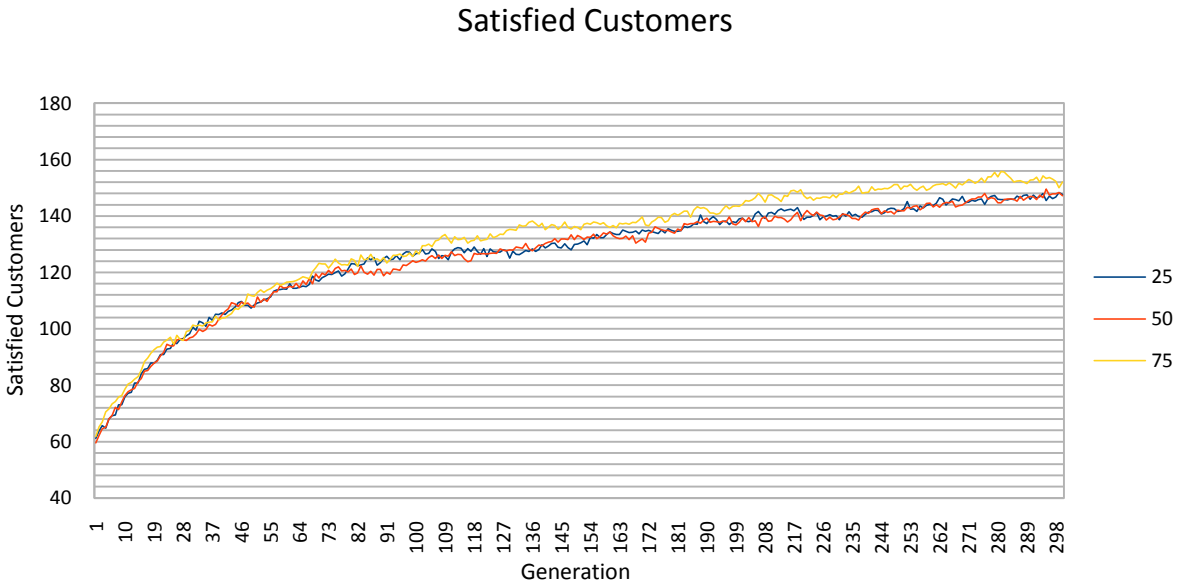
routes faster.

Experiment 2 – *Cloning using roulette selection based on profit.*

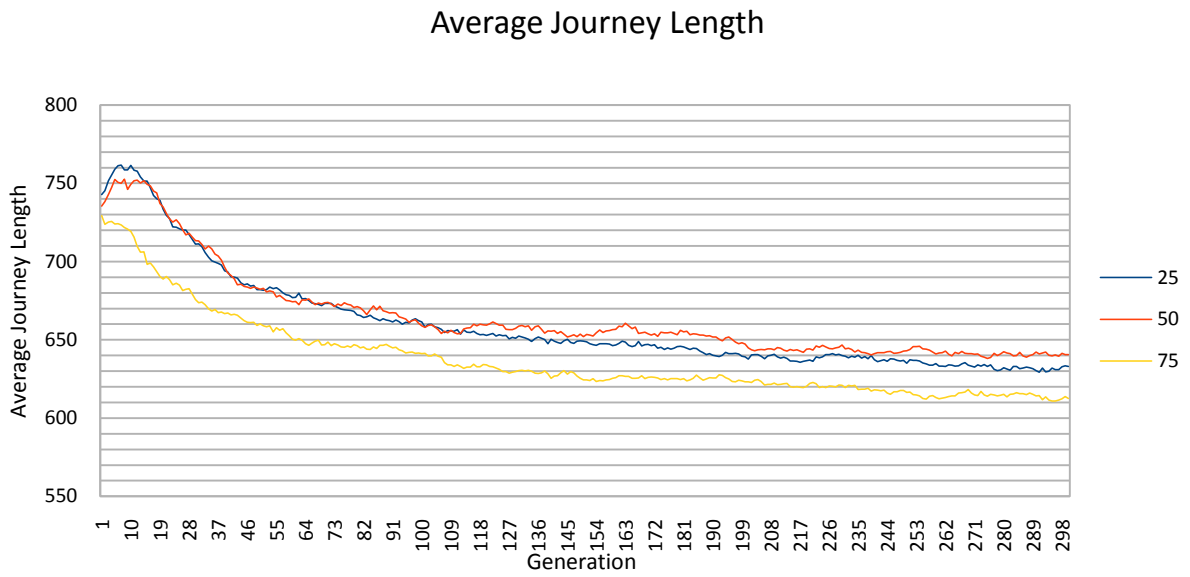
In the previous experiment I used a simple selection method that simply chose the route that was making the most profit. I will now see what occurs when we apply roulette selection proportional to the profit of each route. I will only apply roulette selection to the breeding, and will carry on simply removing the lowest profit route in the group.



We can see that there is an initial peak in profit and then a slow decline to just below 60,000. There is very little difference between the different mutation rates.



We can see that the amount of Satisfied Customers steadily increases from 60 to roughly 150. There is little difference between the 25% and 50% mutation rates, however the 75% mutation rate reaches a slightly higher value, though not enough to be significant.



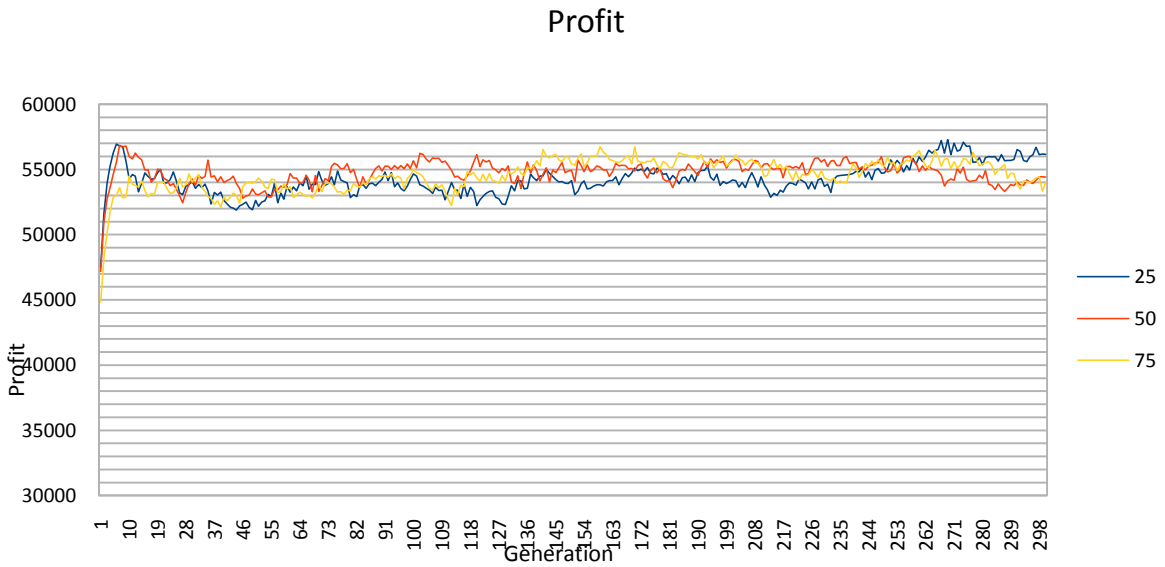
The Average Journey Length steadily decreases from 730 to 610 (in the case of the 75% mutation rate) over the course of the 300 generations.

Analysis of Experiment 2

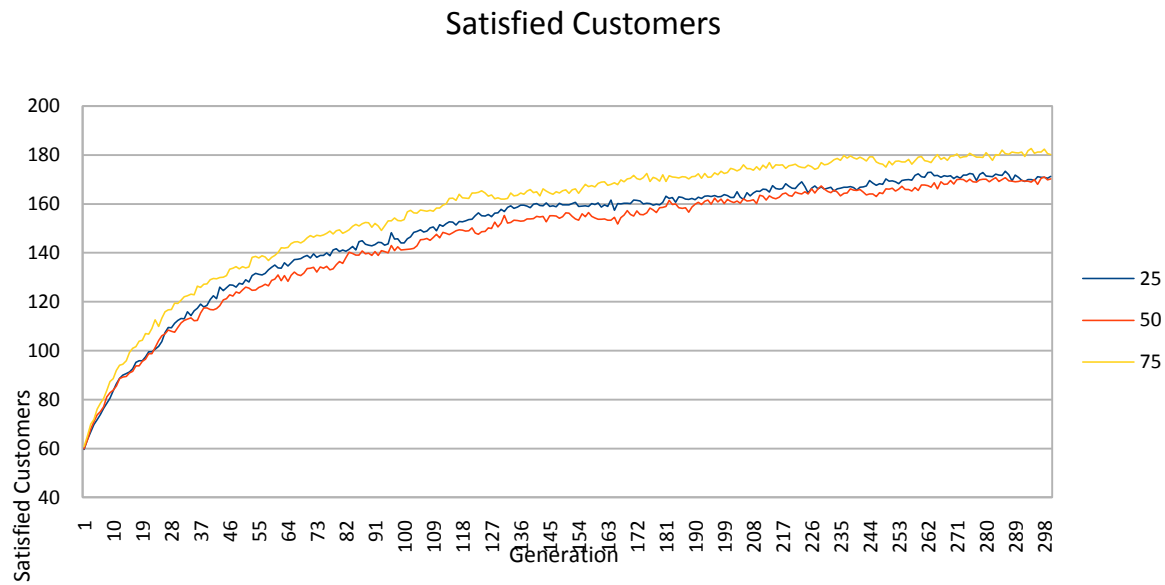
The only difference in approach between this experiment and the previous is in the selection method for breeding. Instead of simply selecting the most profitable route, I instead select using roulette selection based on the profit of each route. As it would be expected the results are therefore very similar. There is no significant difference in the profit values or the amount of Satisfied Customers. However, it does achieve better when it comes to the Average Route Length, with the 75% mutation rate finishing at 610 (20 lower than before). I believe this is due to a more thorough searching of the fitness landscape. Where as before we were only selecting the best route to breed, this time there is a chance that any of the routes could be chosen. This means adaptation, and therefore competition between similar routes, is occurring not just in the best routes, but throughout the population. This results in routes of varying profit being able to optimise, creating the overall result of a lower Average Journey Length.

Experiment 3 – *Crossover using roulette selection based on profit.*

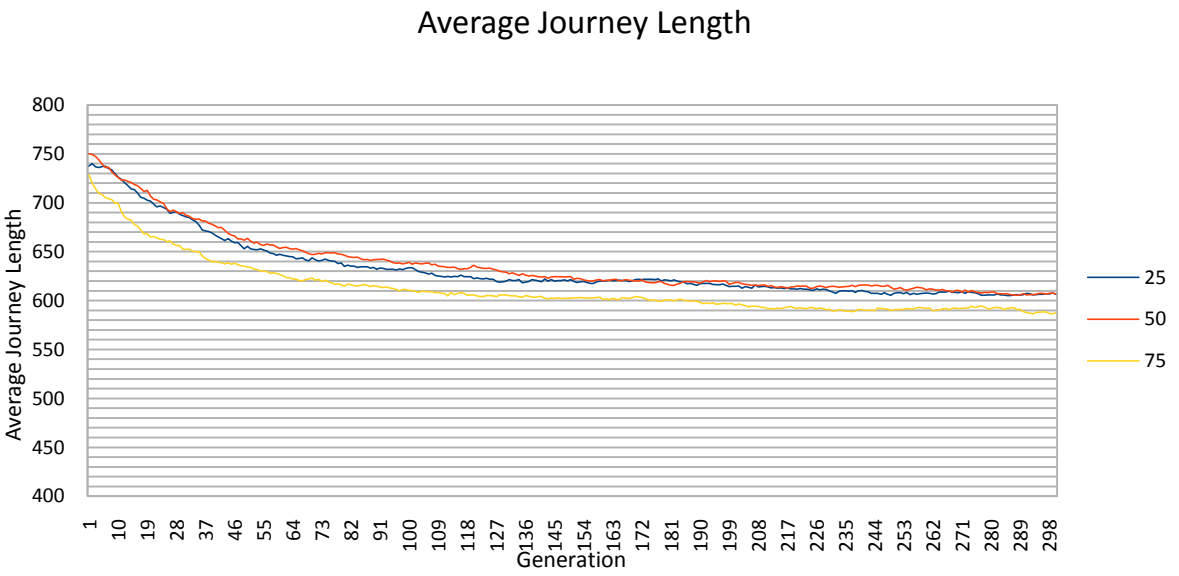
In the previous two experiments I was breeding using a very simple cloning method. This meant that the only way that offspring could diverge from their parents was through mutation. In this experiment I am going apply crossover. As before I will use Roulette Selection based on profit, but instead will select two parents. When a parent is selected it is temporarily removed from the group before the second parent is selected, this ensures that the two parents are different. I will also include the same 3 chances of point mutation (25%, 50% and 75%) and observe the effects.



Above we can see that as before the profit initially increases rapidly but very quickly settles at roughly 55,000. The initial peak is not as pronounced as it has been in the last two experiments.



As before the number of satisfied customers increases gradually over the course of the 300 generations, with the rate of increase slowly reducing. However this time, after 300 generations the 180 mark is reached, whereas before less than 160 was achieved.



We can see above that the average route length slowly decreases from roughly 750 to just below 600. This again is an improvement on the previous two experiments where neither managed to get below the 600 mark.

Analysis of Experiment 3

We can see from the Satisfied Customers and Average Route Length graphs, that using crossover has achieved better results than cloning did. However, the profit value is considerably lower than that achieved by crossover. The lower Average Route Length is achieved as a result of a better exploration of the route landscape. Whereas before, with simple cloning, the system was reliant on mutation alone to explore, (and hence adapted slowly), the population in experiment 3 was able to explore much quicker using crossover.

There at first seems to be a contradiction in the results above. There are more Satisfied Customers yet less overall Profit than the previous experiments. Another way to think of this is that the service being provided is too good; that is to say that a lot of customers are pleased with the service, but a lot of the routes aren't being used fully, and hence the costs are not being covered. I believe this is down to the method of crossover that I am using. As discussed previously in this report the crossover method that I am applying allows for the offspring to be of varying size. This means it is possible that routes are created that are much longer than in previous experiments and therefore require more passengers in order to make a profit. Longer routes are likely to have long offspring, meaning that over the generations the group average route length will steadily increase. If all the routes in the population get longer, then it is easier for longer, non-profitable routes to survive (they only need to make less of a loss than their neighbours). Although it is essential that routes are capable of producing offspring of varying lengths, I need to find a way that will ensure the average profit doesn’t decrease.

Overflow

In the previous three experiments selection was based on the profit that each route was making. Because of the genetic similarity between parent and offspring we observed the phenomenon of overcrowding: by cloning the most profitable routes, they became less profitable. Because of the co-evolutionary nature of the relationships between the routes in the group, to maximise the overall group profit it isn't as simple as just cloning the most profitable routes.

In the next three experiments I want to look at a different fitness function by which to select individual routes. Instead of looking at how much profit a route is making, I will instead select based on the overflow along that route. The overflow of a route can be thought of as how many people are being turned away along that path each day. It seems sensible to say that if a route is turning a lot of people away each day, then we need more routes in the group similar to those routes. Not only do we need more routes like that to satisfy the demand, but we also know that those routes are in a stronger position and are therefore less likely to suffer from overcrowding. To use a real world analogy, it is as if now we are breeding the organisms that have a surplus of food, and therefore at less risk of starvation.

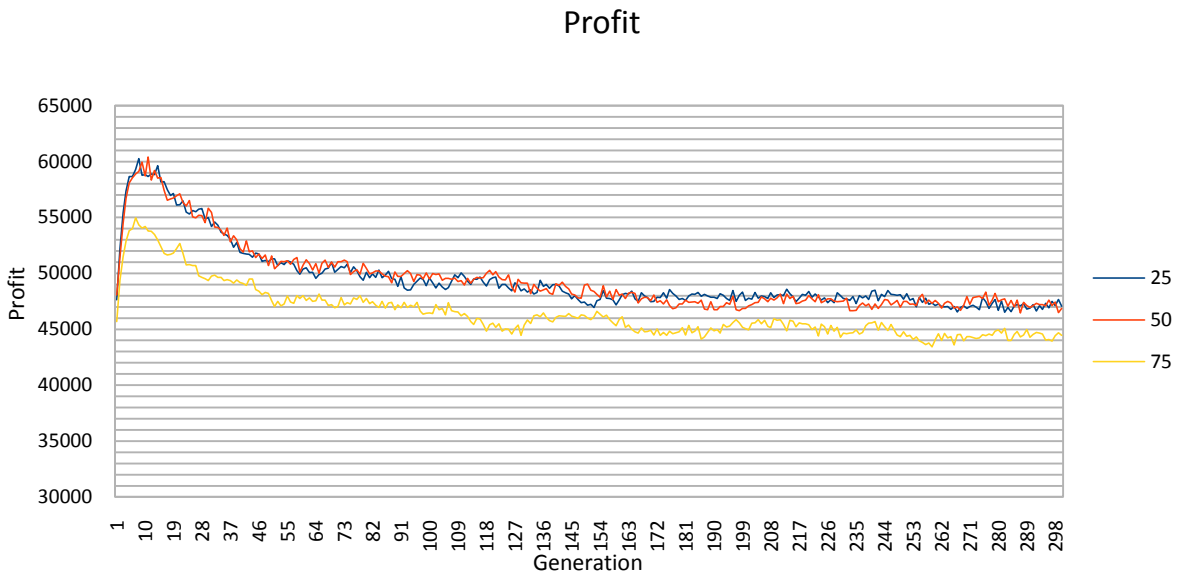
How to measure overflow?

To measure the overflow of each route I will first calculate an Overflow matrix. Very simply this is the Flow Matrix subtract the Service Provided Matrix. A positive value in the Overflow matrix represents an edge on the graph with an overflow. To calculate an overflow value for an individual route I will simply sum the overflow values for each edge along its path:

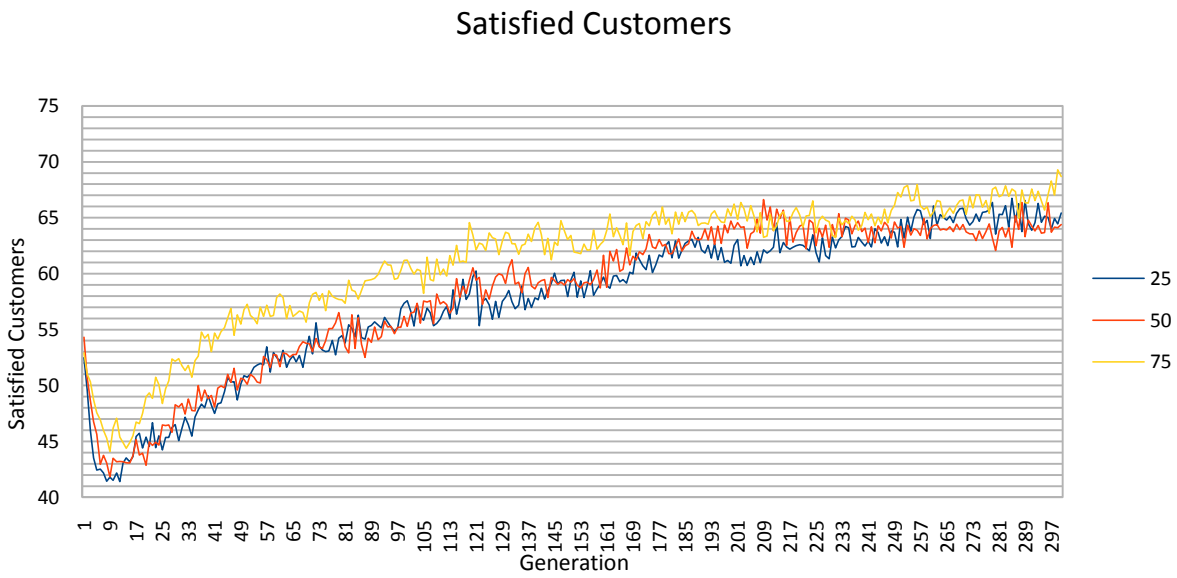
$$R_{Overflow} = \sum_{\forall edge_{a,b} in R} Overflow_{a,b}$$

Experiment 4 – *Cloning using simple selection based on overflow.*

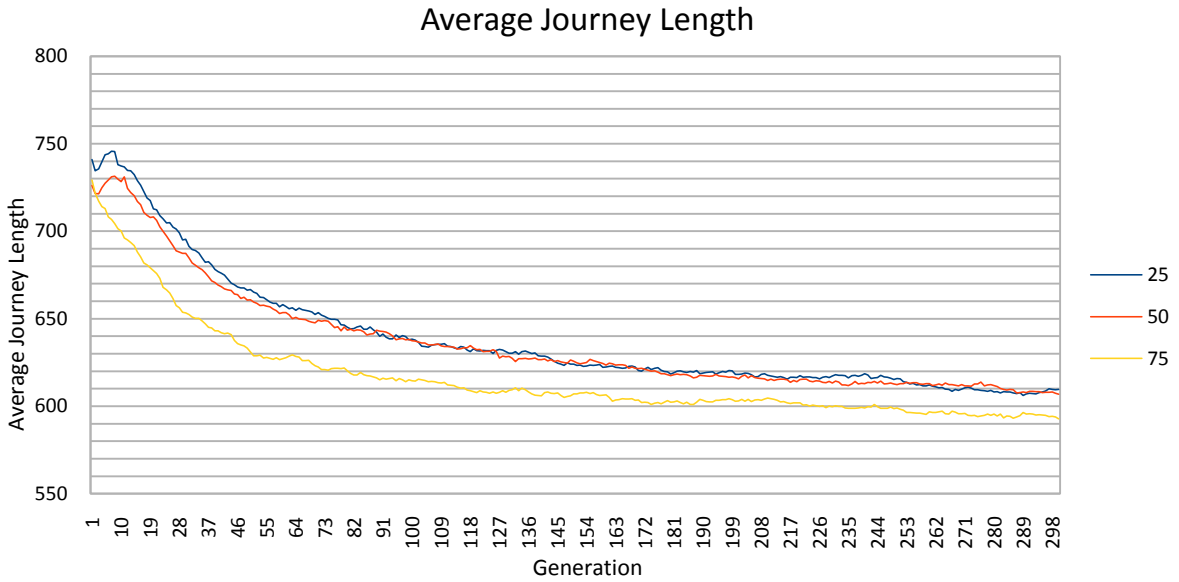
I will start by using cloning and simple selection (cloning the route with the highest overflow, and removing the route with the lowest (in the case of a tie I will select at random between the tied routes).



The above graph shows an initial peak in profit and then a steady decline to below 50,000. There is very little difference between the 25% and 50% mutation rates, though the 75% mutation performs worse.



We can see above there is initially a sharp deep in the number of satisfied customers, before a steady increase up to approximately 65. This is significantly lower than any of the experiments done selecting on profit.



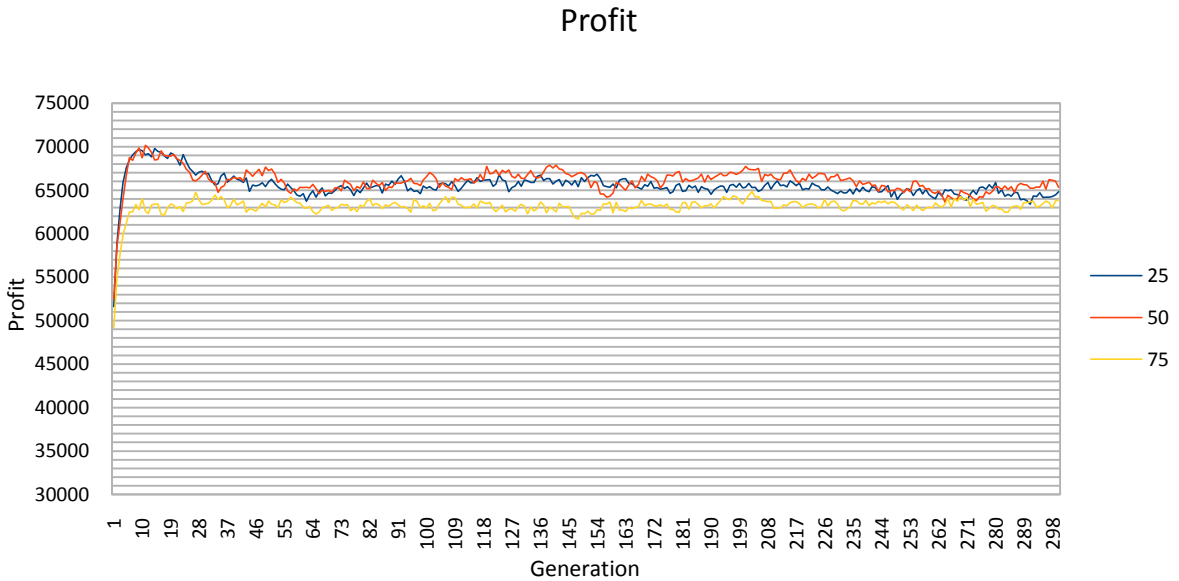
Above we can see the Average Route Length start at approximately 730 before declining steadily to 600.

Analysis of Experiment 4

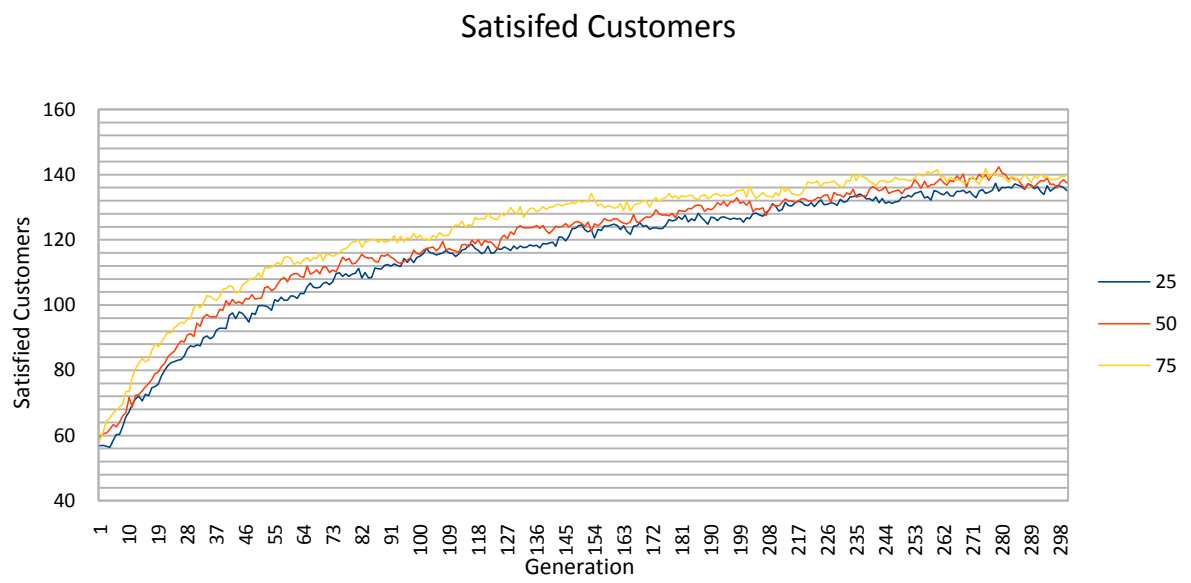
On first inspection it appears that selecting by overflow perform drastically worse than when selecting on profit. The profit declines to below 50,000 (lower than any of the profit experiments) and the Satisfied Customers never gets above 70. The Average Journey Length however is very similar to the previous experiments. However, I believe the reason for this poor performance isn't due to how selection for breeding is occurring, but is due to the selection for removing routes. The system was selecting routes to remove based on the route with the lowest overflow, and in case of a tie randomly picking between the tied routes.

On reflection this is a very poor way to choose routes to remove. It is as if the routes that aren't turning people away each day are being punished. A route could be highly profitable, be exactly full all day with no overflow, and yet still be in the firing line to be removed. I will repeat the experiment, again with the routes with the highest overflow being selected for cloning, however this time I will remove based on the lowest profit as before.

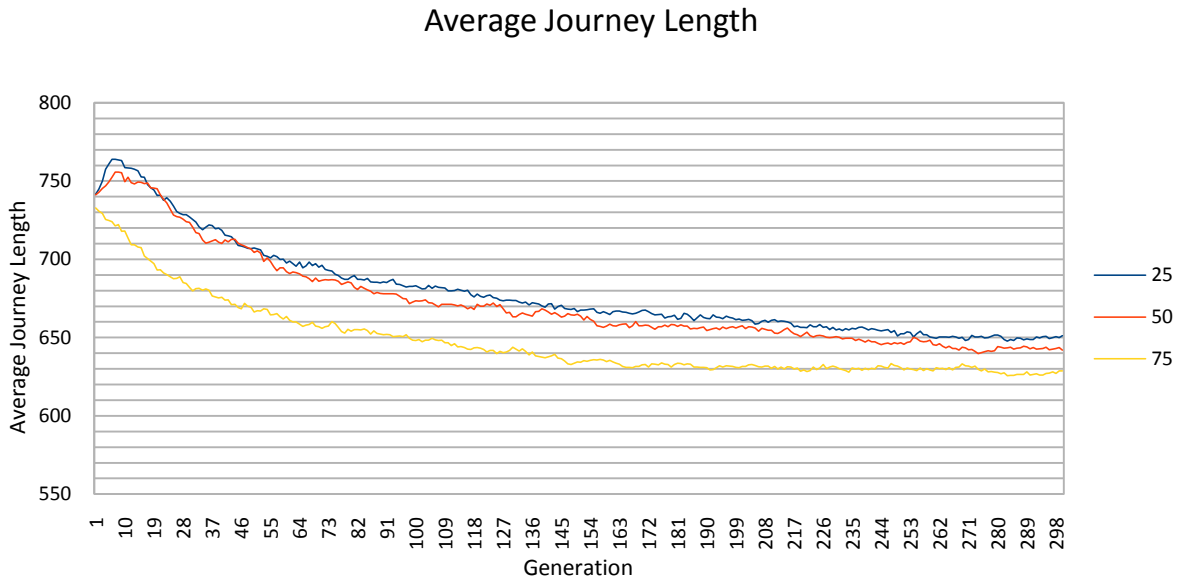
Experiment 5 – *Cloning using simple selection based on overflow, with removal based on profit.*



We can see from the above graph that the profit initially peaks at 70,000 before quickly settling around 65,000. There is very little difference between the mutation rates, apart from 75% which performs significantly worse than the others.



In the above graph we can see that the amount of Satisfied Customers increases steadily from just below 60 to 140.



The above graph shows the Average Route Length decline from around 730 to 650.

Analysis of Experiment 5

We can see immediately that by changing the selection method for removing routes from the group the results are drastically improved. The profit value settles at approximately 65,000, which is higher than any of the previous experiments settled at. I believe this is due to the fact that the offspring were more likely to be profitable because their parents were selected by how many passengers they had to spare, whereas before this wasn't the case.

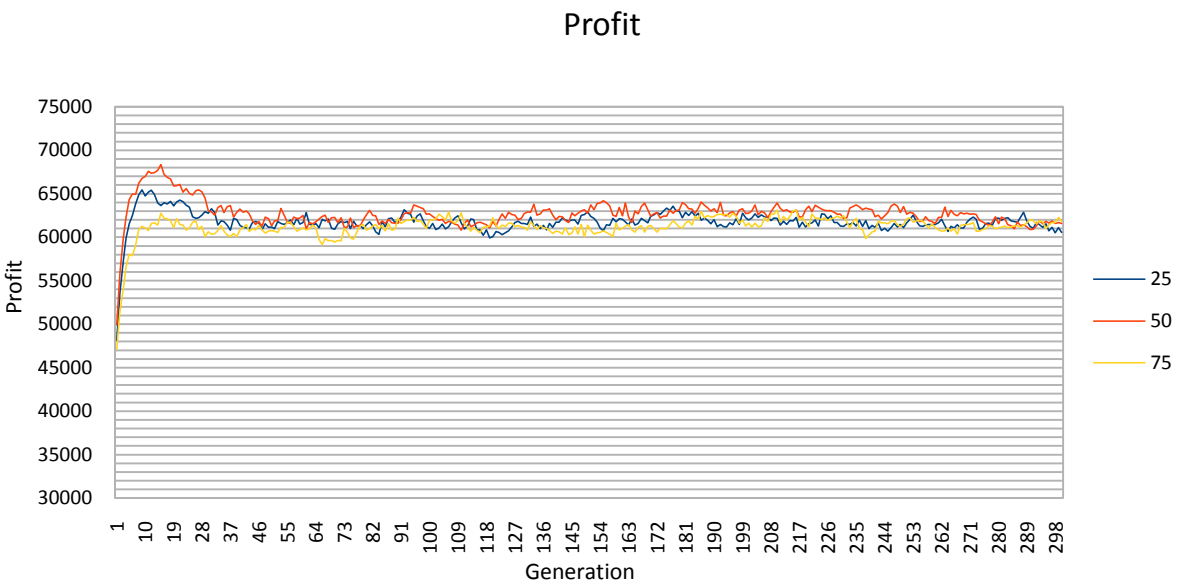
The amount of Satisfied Customers however is slightly lower than all the previous “profit based” experiments, reaching a peak of just below 140 (all the profit experiments went over the 150 mark). I find this surprising as I expected that by selecting based on overflow, the overall overflow of the group would be reduced (just as the profit was reduced when we selected by profit) and a reduction in overflow should mean an increase in Satisfied Customers.

In hindsight the measure of overflow that I am employing is naïve, and puts a bias on the longer routes. This bias is due to the fact that the more edges a route has, the more likely it is to have a higher

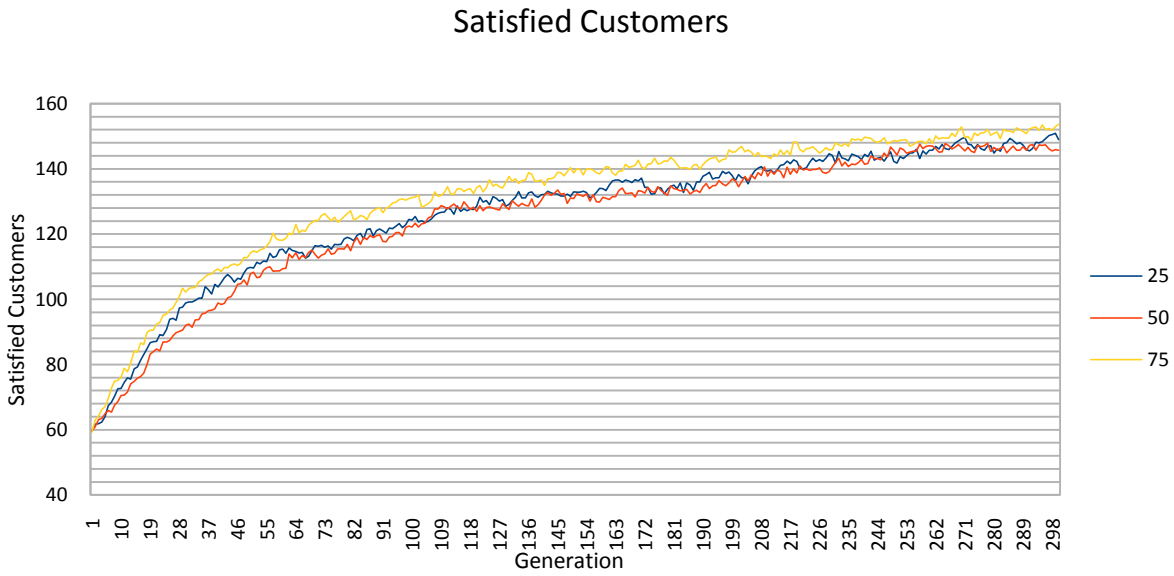
overflow. One solution to this could be to calculate some kind of average overflow, or some other measure that would take the size of the route into account. However, I think there is some merit in having a bias towards the larger routes with the largest overflow, as regardless of the bias, the solution would still be vastly improved if that larger overflow was dealt with. I think therefore the fault lies with the selection method, and a method that gives the routes with lower overflows the chance to get selected (such as roulette) would correct this.

The final Average Route Length also doesn't match any of the "profit based" experiments. Again I believe this is down the bias being placed on the larger routes, meaning the smaller routes rarely got chance to reproduce meaning they couldn't adapt. For the next experiment I will apply roulette selection as this should allow for adaptation at all levels within the group.

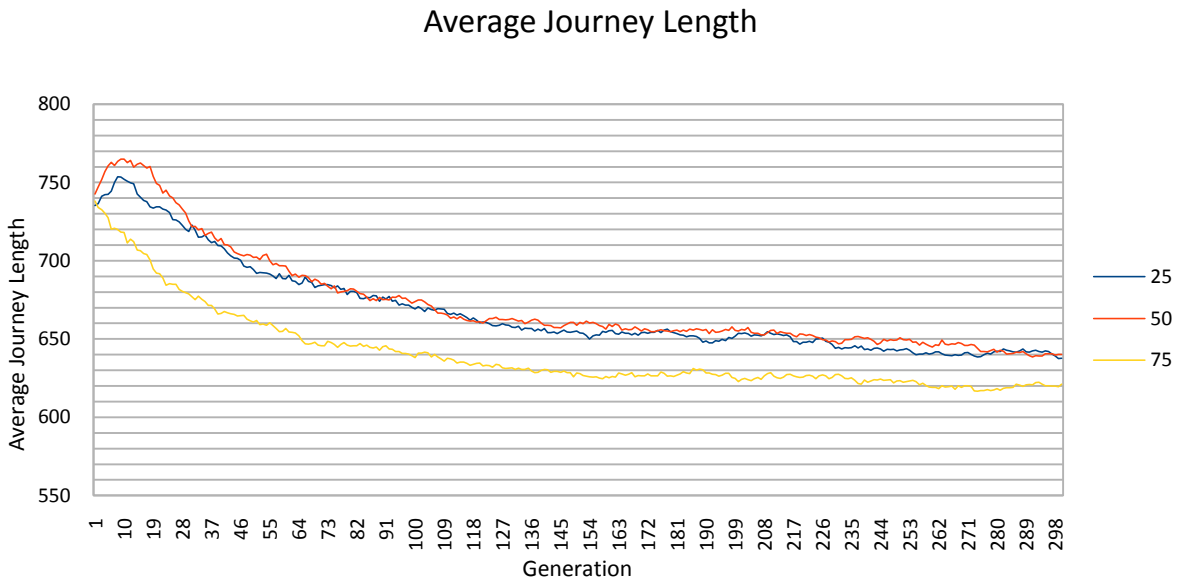
Experiment 6 – *Cloning using roulette selection based on overflow.*



We can see from the above graph that after the initial peak the profit values settle at just over 60,000.



Above we can see as before the amount of satisfied customers gradually increases, however this time the 140 mark is passed, whereas before it finished just below.



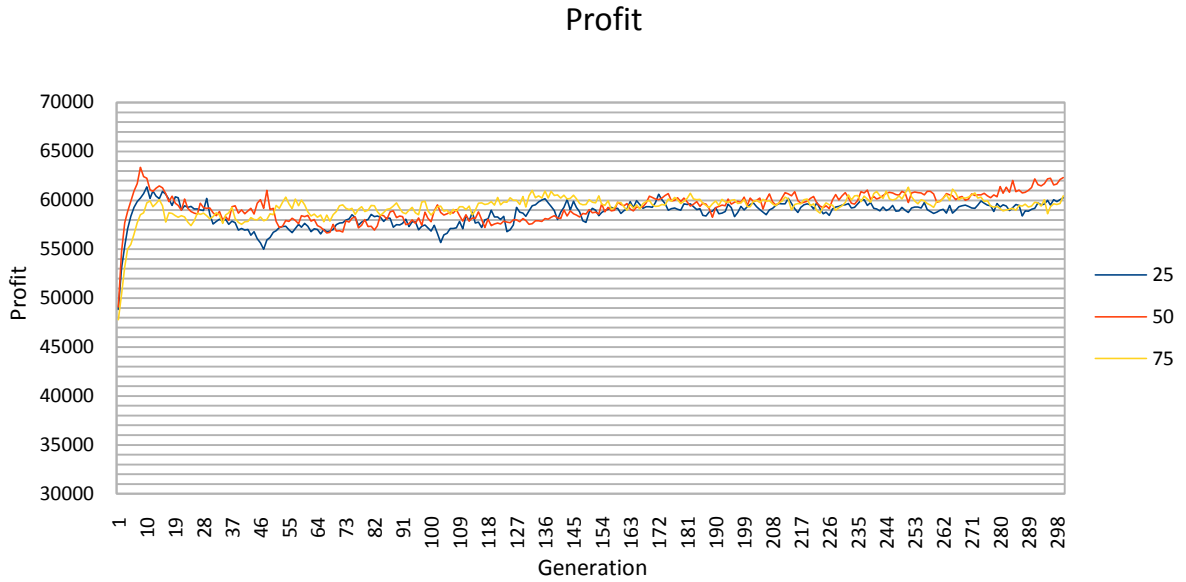
We can see above that the Average Route Length performs in a very similar manner to the previous experiment. The 75% mutation rate performs slightly better than the others, both in the rate at which it decreases and the final result. However, it is not a significant improvement on before.

Analysis of Experiment 6

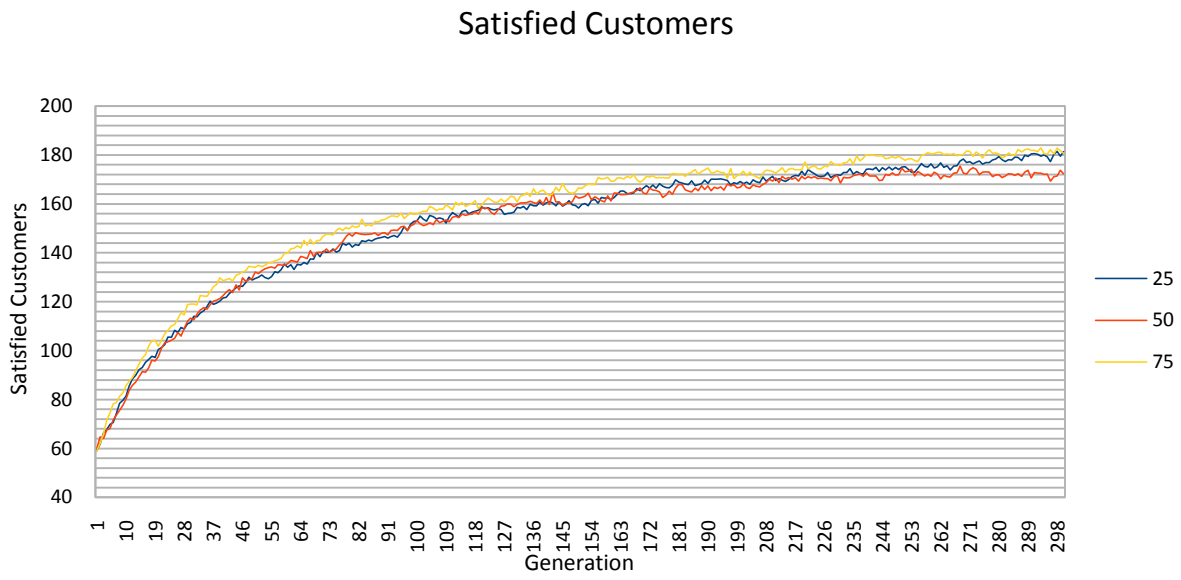
As predicted there has been an improvement in the amount of Satisfied Customers than in the previous experiment. There has also been an improvement in the Average Route Length, with the 75% mutation rate achieving 630. However, it seems the pay-off for this improvement is a lower profit value.

Experiment 7 – *Crossover using roulette selection based on overflow.*

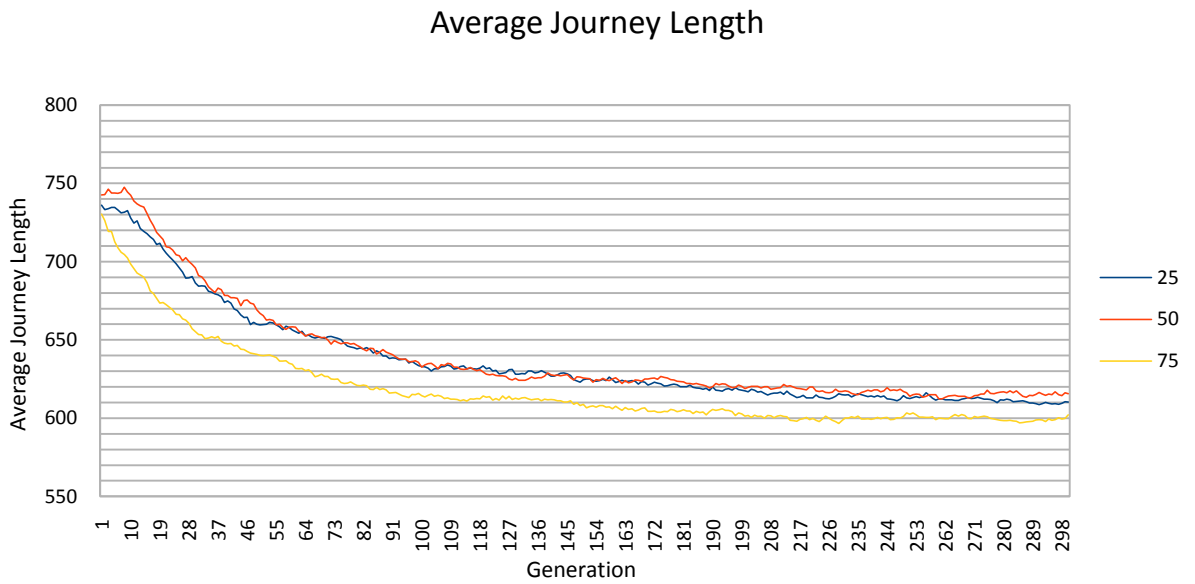
I will now, as with the final profit experiment, use both crossover and roulette selection based on overflow. The route that is being removed at each generation is still being selected based on the lowest profit.



We can see above that the profit initially increases very rapidly before slightly declining and then slowly rising to the 60,000 mark.



As with the previous experiments the amount of satisfied customers slowly increases across the generations reaching a peak of 180. This is significantly higher than in the previous experiment.



We can see that the Average Route Length slowly declines finishing at 600.

Analysis of Experiment 7

This is the best result I have achieved so far. The Satisfied Customer value is 180, just as it was when using crossover and roulette selection based on profit, however this time the profit is 62,000 (2,000 higher than when we were selecting based on profit). Note that this isn't the highest profit value that has been achieved, however when considered with the amount of Satisfied Customers and the Average Route Length, it is the best result so far.

It seems that selecting based on how much unsatisfied demand there is for a route, rather than how much profit the route is making, allows the group to adapt in way that avoids overcrowding. I now want to see what will happen if I consider both the overflow and the profit when selecting.

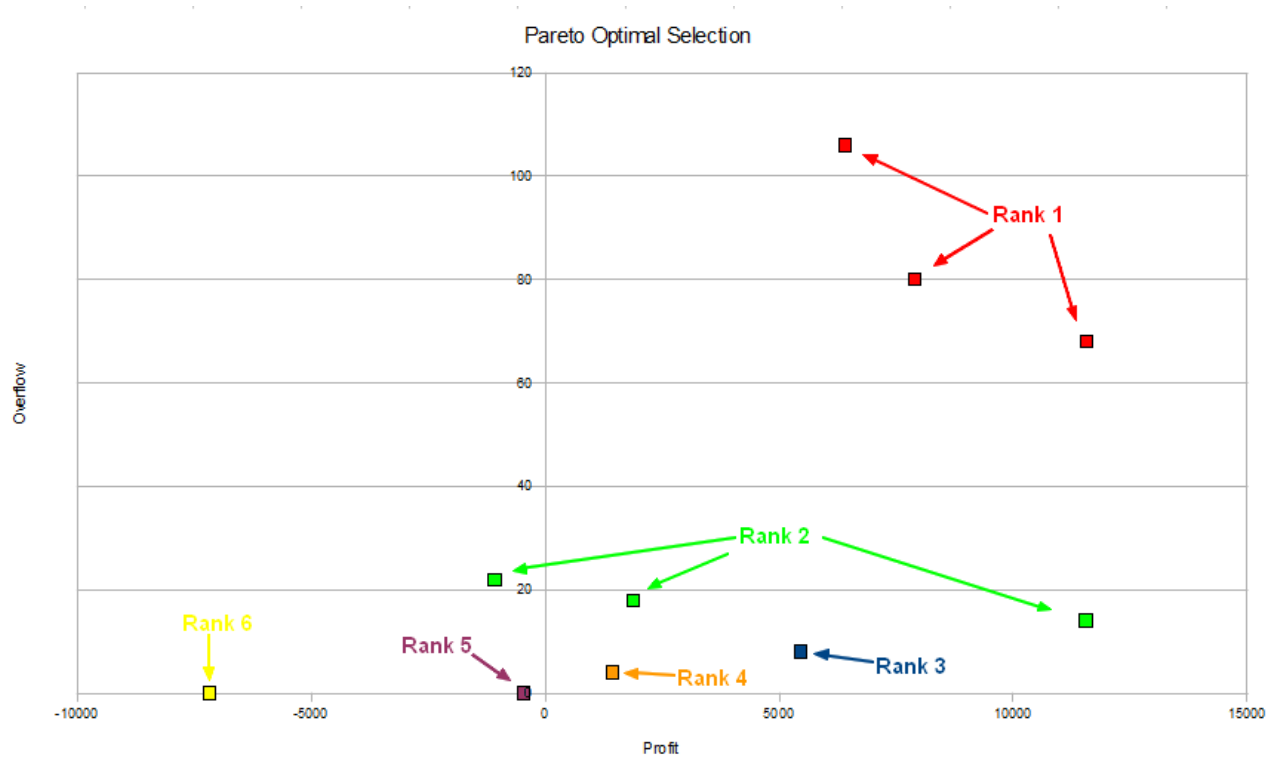
Pareto-Optimal Selection

Domination – Route A is said to dominate Route B, if Route A 'beats' Route B in all measures of fitness.

Above is the definition of domination that I will use for pareto-optimal selection. In this particular case the two measures of fitness that I will be using are profit and overflow. I will plot each route on a graph and rank them using the following algorithm:

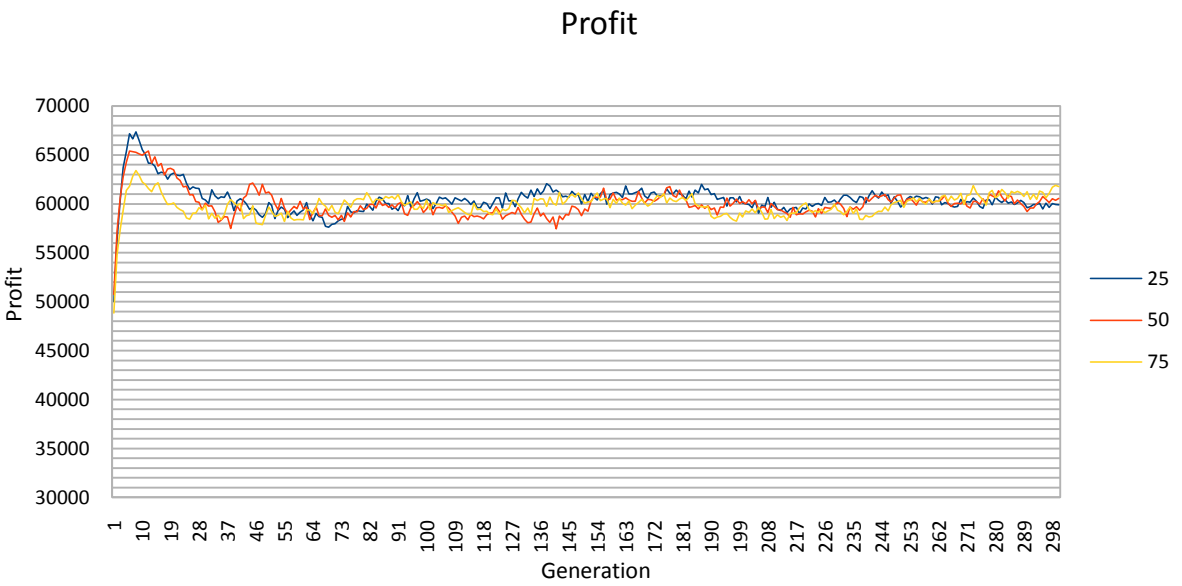
```
Initialise r = 1
While Routes isn't empty
      for each route X in Routes
            if route X isn't dominated then
                  set route X's rank to r
                  remove X from Routes
            end if
      end
      r++
end
```

The above algorithm assigns ranks to each route. It initially finds all the non-dominated solutions, often referred to as the “pareto front”, and assigns these solutions rank 1. It then removes these solutions and starts the process again, only this time assigning rank 2. It does this until there are no more routes left to rank.

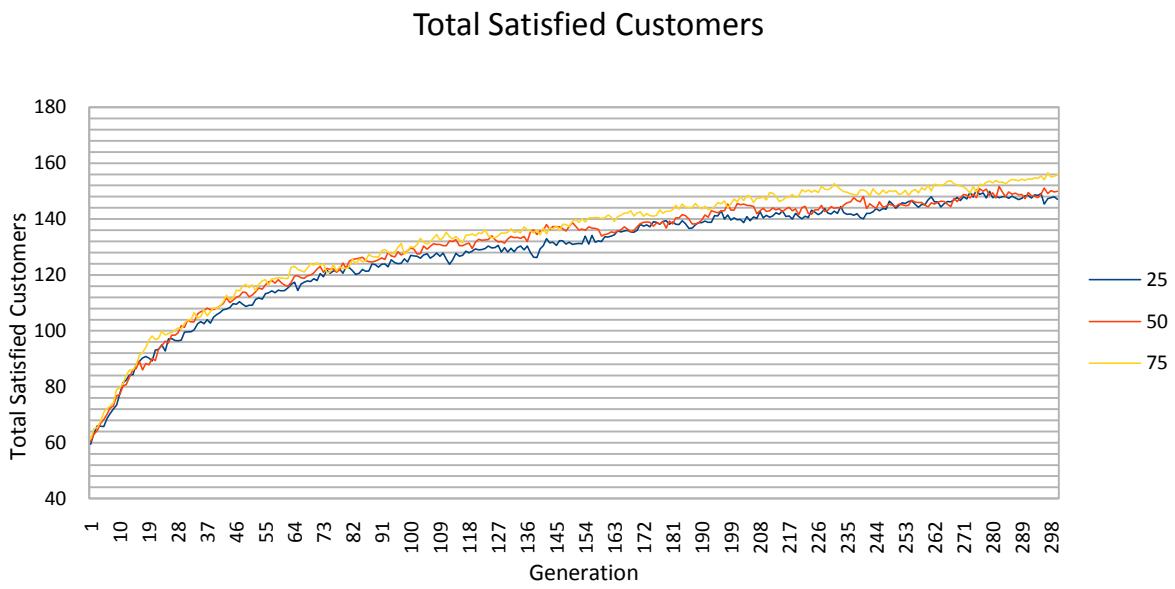


The routes in the first rank have relatively large profit and overflow values when compared to the rest of the group. My theory is that by selecting high ranking routes for reproduction, I can guarantee that the offspring will make a profit (because their parents have a large overflow) and not only that, but that they will make a relatively large amount of profit (because their parents make a large profit).

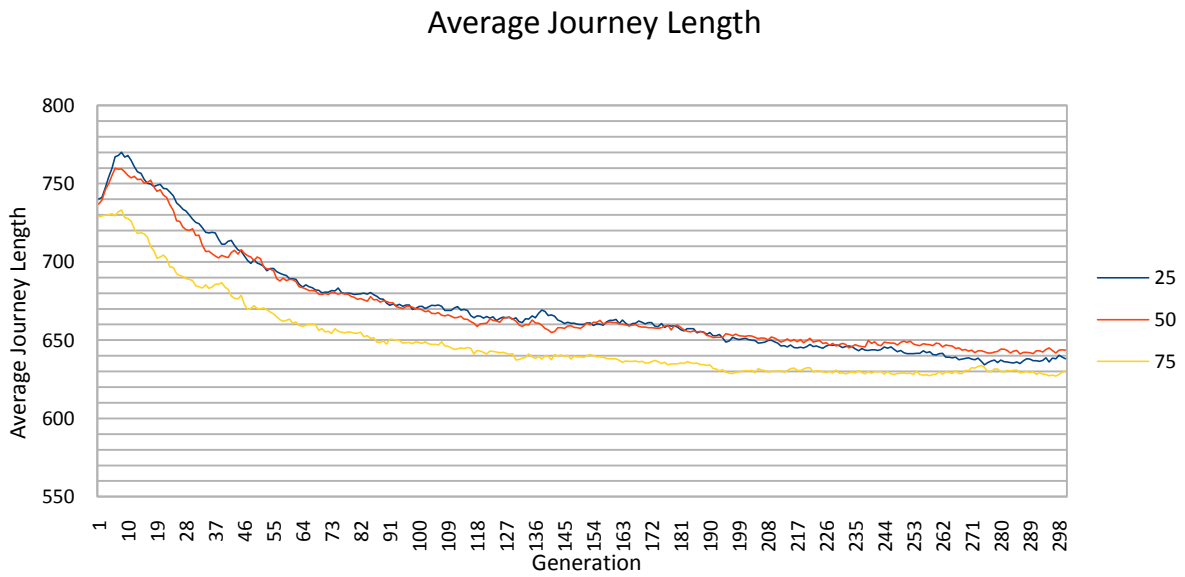
Experiment 8 – *Cloning using pareto-optimal selection based on overflow and profit.*



We can see that initially the profit peaks before slowly declining and settling around the 60000 mark. There is very little difference between the mutation rates.



The amount of Total Satisfied Customers steadily increases finishing around the 150 mark.



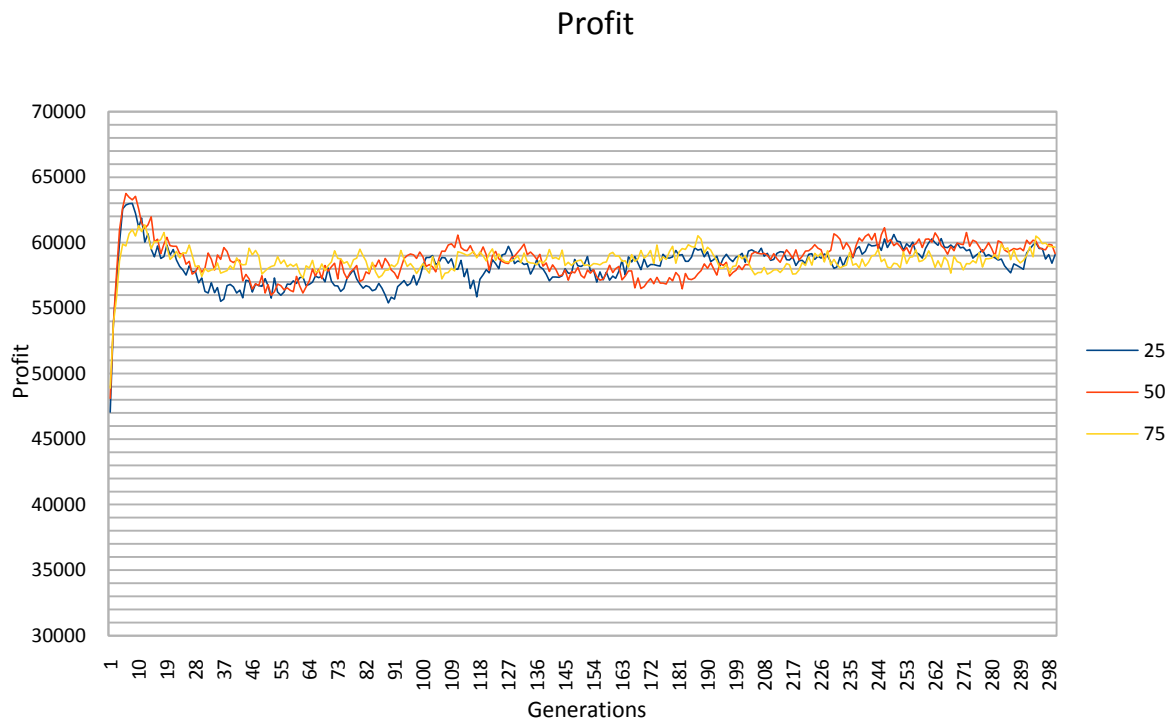
The Average Journey Length steadily declines finishing between 625 and 650.

Analysis of Experiment 8

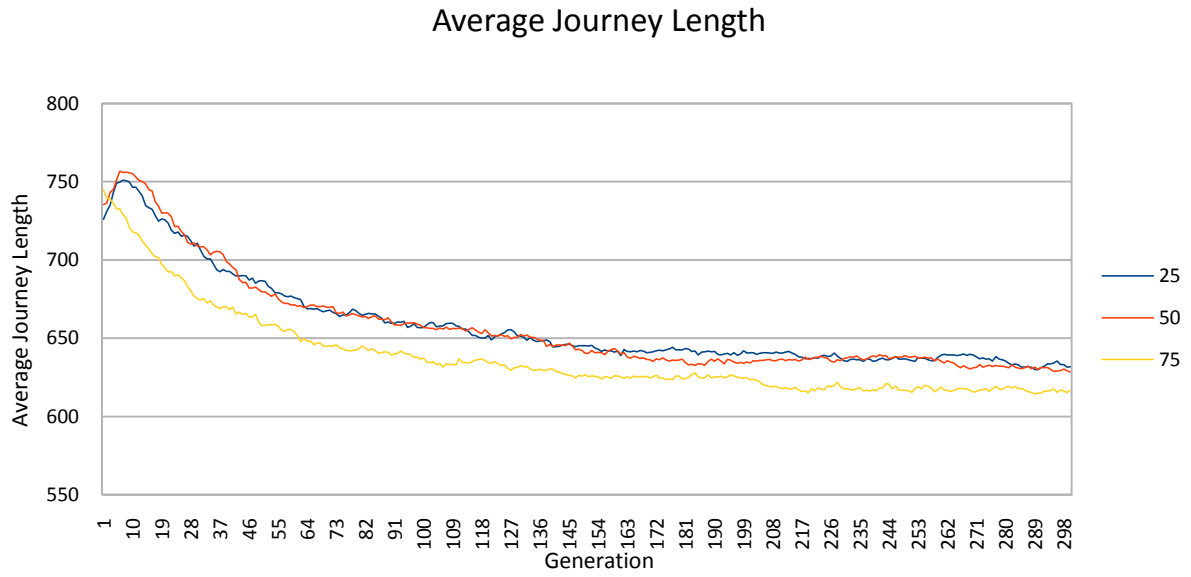
I hypothesised that by using a pareto optimal approach I could achieve better solutions. At the very least I thought that solutions of a similar quality could be reached quicker. However, it is clear from the graphs above that this is not the case. The profit value settles around the 60,000 mark which is neither an improvement on when we selected by profit or overflow individually. The Average Journey Length and Amount of Satisfied Customers do significantly worse. I will now perform pareto-optimal selection with crossover to determine whether this failure is a result of the selection method or the breeding method.

Experiment 9 – *Crossover using pareto-optimal selection based on overflow and profit.*

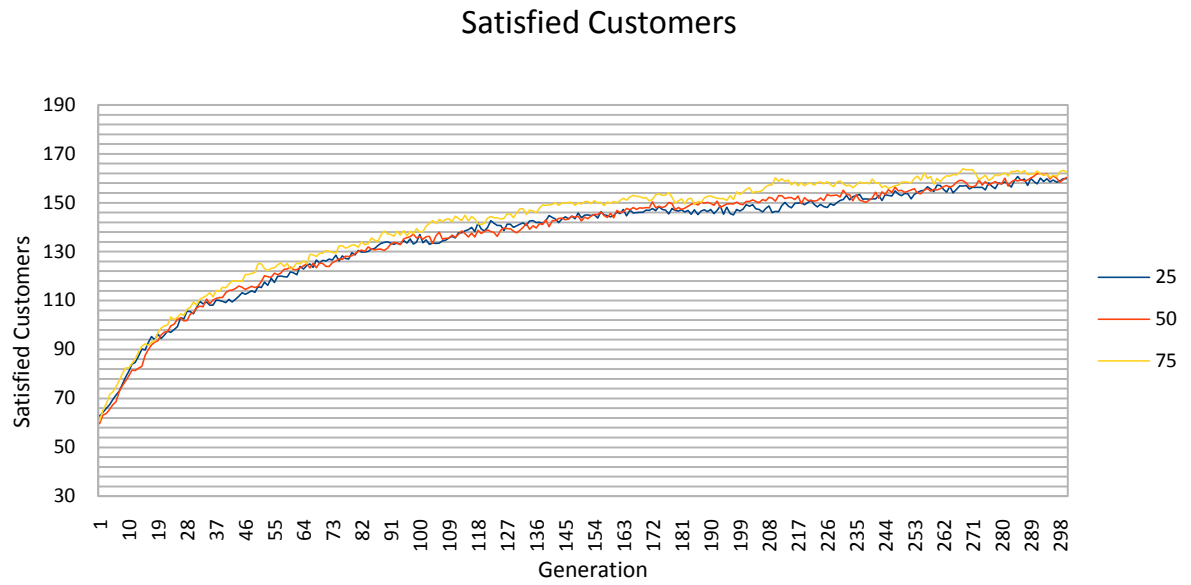
During this experiment I will be selecting two distinct parents from the subset of routes in rank 1. If there is only 1 route in rank 1 I will select the second from the rank 2 subset.



We can see in the graph above that after the initial peak the profit doesn't exceed higher than 60,000.



We can see in the graph above that after an initial peak the Average Journey Length declines slowly finishing in the range between 600 and 650.



The Satisfied Customers steadily increases finishing just below the 160 mark.

Analysis of Experiment 9

Changing the breeding method to crossover has improved the performance of pareto-optimal selection in the Satisfied Customer and Average Journey Length values. However the profit hasn't improved. As in the previous experiments crossover allows a quicker searching of the solution space, which is required to optimise the Satisfied Customer and Average Journey Length values. However, using pareto-optimal selection has failed to better either roulette selection by profit or roulette selection by overflow.

Conclusion and Further Work

Everywhere we look in nature we see organisms that are highly adapted to survive in the environment in which they exist. These adaptations are not simply a result of the physical landscape in which they exist, but also the genetic. An organism's chances of survival are not just affected by which physical environment they inhabit, but also heavily dependent on whom they share that environment with. Over the generations, competition between organisms creates arms races, with each side slowly pushing the other to become more and more refined in order to survive. Paradoxically, often the result of such competition is apparent collaboration. Organism's that adapt to work with one another have a greater chance of survival. Whether this collaboration is between members of the same species (such as the highly complex and sophisticated ant colonies) or between members of species that separated millions of years ago (such as the relationship between the fig and the fig wasp) the end product of such an arms race is a seemingly well designed, highly engineered *system*.

Since the 1950s, aspects of evolution have been modelled by Computer Scientists with great success. The vast majority of these algorithms create populations of solutions that are then tested in isolation from one another. One of the first *co-evolutionary* algorithms, Hillis's Sorting Networks, demonstrated that by creating two populations, one of solutions and one of problems, an arms race can be modelled, just like those observed between rival species in nature. The power of such a model is that the solution population cannot stagnate, as may happen when using a static problem set, meaning they are pushed to be as effective as they possibly can be.

The aim of this project was to investigate how the competition and interactions that occur between members of the *same* species can be modelled in order to solve complex routing problems. In particular I wanted to see how this approach would fare on the Bus Routing Problem.

Firstly I created a model of the Bus Routing Problem. I wanted to create a model that realistically simulated the flow of people around a network. By assuming that passengers would always choose the shortest route (using the available routes) to their destination I was able to use the Demand and Distance Matrices to model the flow of people along the vertices. To measure the success of a particular solution I defined a Profit and Satisfied Customers function (that should be maximised), and an Average Journey Length that should be minimised. Using this model I was able to apply and test my algorithm.

Working at the 'individual level' I was able to observe a bottom-up, emergent behaviour at the 'group level'. Initially, using a fixed population size and a steady state selection process, I selected routes within the system based on how much profit they were making. The most profitable route was cloned, and the least profitable removed. Using cloning meant that routes in future generations were completely reliant on mutation in order to adapt their paths around the graph. As a result, although I observed adaptation in the Average Journey Length, it was slow. Next, still solely looking at profit, I applied roulette selection and crossover and found that this had a significant improvement on the Average Journey Length and Satisfied Customer count. Crossover created a much quicker exploration of the solution landscape allowing the system to find better solutions. However, the profit value hardly adapted at all, very quickly settling at a value not vastly greater than the initial starting value.

Although it initially seemed intuitive to select by profit at the individual level in order to maximise profit at the group level, this clearly wasn't the case. What in fact happened was overcrowding. By reproducing profitable routes, I created organisms very similar to that route, creating competition and therefore reducing the overall success of both. Just because a route made a profit, it didn't guarantee that there would be space for its offspring to also make a profit. My next approach was to instead select routes based on their potential to make *more* profit. Instead of looking at how much profit they were making, I took a measure of how many people they were turning away each day due to being full. I called this overflow.

Initially, I selected the route with the highest overflow in order to breed, and the route with the lowest overflow to remove. This produced terrible results, with the profit plummeting. This was due to how I was choosing routes to remove. A lot of the routes had no overflow, so this removal process effectively became a random choice between a relatively large subset of the population. I altered this so that the routes being removed was based on profit and the routes being bred based on overflow. Now the profit reached a much higher value (10,000 more than the profit experiments). As before, using roulette selection and crossover had the greatest success for the Satisfied Customers and Average Journey Length values. This time however the profit value was also significantly higher. Finally, I applied pareto-optimal selection which

considered both profit and overflow. I hoped that by considering both these values when selecting I would be able to exceed the results of both the previous approaches. Unfortunately this wasn't the case.

I have shown, by modelling multiple agents and selecting them based on their individual successes and failures that adaptation will occur at the group level, with effective solutions emerging as a result. However thought must be given to how these individuals are selected. Because of the effect that each agent in the system has on its neighbours it is not as simple as simply selecting the most successful routes. Thought must be given to which parts of the system are under-catered for, and select based on this. The best results were obtained when I measured the overflow of each route and applied crossover accordingly.

Given extra time, I would be keen to investigate how a similar approach would be affected by using a varying population size. During the experiments in this report I solely used a fixed population size, however this is clearly not the case in nature. If a population was allowed to vary in size, we could imagine solutions being able to become even more profitable due to the fact that the population would be able to adjust in size to fit the particular problem. However, I don't imagine this to be a simple task. Creating a dynamic population that is capable of sustaining itself and not simply going extinct would require a lot of thought and experimentation. In addition to a varying population size, it would also be interesting to investigate the effects of varying frequency rates. In this report, I assumed a fixed frequency rate for all the routes. However, if this rate were capable of adjusting as well, we could imagine the system being capable of a much higher degree of fine-tuning.

References

[1] Co-Evolving Parasites Improve Simulated Evolution As An Optimization Procedrue, W. Daniel Hillis

[1a] p230

[1b] p231

[1c] p232

[1d] p233

[2] Competitive Environments Evolve Better Solutions for Complex Tasks, Peter J. Angeline and Jordan B. Pollack, 1993, p2

[3] The Blind Watchmaker, Richard Dawkins, 1986, p185

[4] Co-evolutionary Algorithms for the Bus Routing Problem, Kevin Nave, 2007

[4a] p32

[4b] p33

[4c] p58

[4d] p50

[5] Note On Two Problems In Connexion with Graphs, Edsger W. Dijkstra, Numerische Mathematik, 269—271, 1959

[6] The Selfish Gene, Richard Dawkins, 1976

[7] Genetic Algorithms – Principles and Perspectives – A Guide to GA Theory, Colin R. Reeves and Jonathon E. Rowe, 2003, p38 - 39