Using Genetic Algorithms to Solve the Travelling Salesman Problem

Helen Harman,
Aberystwyth University,
heh14@aber.ac.uk,
Student Number: 110007212

December 7, 2015

1 Introduction

The aim of the travelling salesman problem (TSP) is to find the shortest possible tour between cities. The tour must pass through all cities before returning to the starting city. The number of possible solutions is the factorial of the number of cities. For example 5 cities has 120 possible solutions, whereas 6 cities has 720 solutions. The TSP is NP-hard, and no general solution to solving it has been found. [16]

The travelling salesman problem has been around since the 1930s. Initially it was known as the "48 state problem". Hassler Whitney wanted to find the shortest school bus route between 49 cities across all 48 US states. By the mid 1940s the problem had been renamed to the travelling salesman problem. [17] Though it is unclear as to who came up with the name.

In 1954 G. Dantzig et al. [7] came up with a solution involving 49 cities. Since then solutions involving many more cities have been found. In 2004 D. Applegate et al. came up with a solution to 24,978 cities.[6]

For many of the solutions found artificial intelligence has been used. These methods include swam intelligence, neural networks, and Genetic Algorithms (GA). This report will look at solving the TSP using GAs.

GAs can be applied to large search problems. A initial population is generated that covers a large distribution of the search space. Based on how well these solutions, called individuals, perform the population is evolved in the hope that the individuals will converge on the optimal solution. The population is evolved using selection, crossover and mutation operations, these come under the umbrella term genetic operations.

There has been a lot of previous work in applying GAs to the TSP. This includes looking at the affect of using different crossover functions. Ahmed [2] introduces a new crossover operator and compares this to previously used crossover operators. Their selection operator keeps the top individuals and is performed after the crossover operation. Abdoun et al. [1] compared the affect of different mutation operations; and uses roulette wheel selection. Both of these use a array containing a list of cities to represent the tour.

This report will compare the use of two different representations, and two selection operators. In section 2 the design decisions will be discussed; and section 3 will discuss the different experiments that have been performed and their results.

2 Design

2.1 System Design

The system has been designed to allow different genetic operations and representations to be easily changed. This is done through the use of abstract classes which take a generic type. An abstract class has been created for representations, the population, mutation operations, selection operations and crossover operations. An example design for this is shown in figure 1. This allows different genetic operators and representations to be used without repeated code.

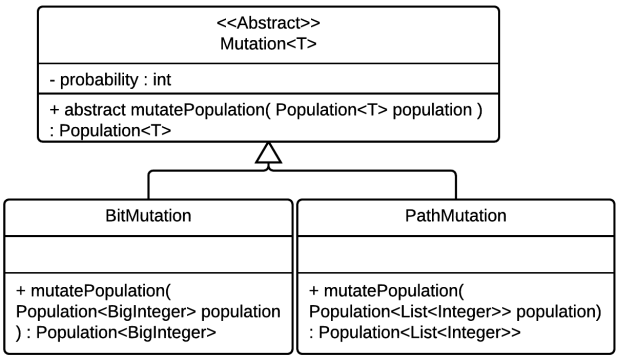


Figure 1: UML class diagram for the mutation operators

Due to the object orientated nature of the design and the use of generic types I chose to use Java. A basic command line interface will be provided to allow the different parameters to be selected. An image showing the shortest found route will be produced. MLPlot [9] will be used to produce graphs showing how the fitness varies between generations.

I chose to use binary representation (the details of this will be discussed in section 2.2). The initial plan was to use Java's BitSet class to hold hold this representation. To do this a long value would be used and converted into the BitSet. Each value represents a different solution,

therefore 21 cities would cause a the long to overflow. The maximum value for a long is $2^{63} - 1$ and there are $5.1 * 10^{19}$ possible solutions. Therefore, Java's BigInteger class is used. The BitInteger class allows for bitwise operations and can be treated as I binary or denary value.

The stopping criteria is based on how the average fitness changes across the generations. If the average fitness does not decrease for N generations the GA will stop. N will be set to 10 for the experiments. It is expected that the average fitness may rise between some generations, therefore using 10 should give enough leeway for this to happen. I considered using the fittest individual within the stopping criteria, however it is likely that this may not change for several generations, which could cause the GA to converge prematurely.

2.2 Representations

A solution to the TSP can be encoded in many different ways. This includes as an ordered list of cities; a matrix of points; or as a binary value. There are advantages and disadvantages for each type of representation.

Binary representation was chosen as often this makes the mutation and crossover functions simpler and in other situations has led to better results than alternative representations. This will be compared to path representation, due to path representation being commonly used for the TSP.

Path Representation

The TSP can be represented as a list of cities in the order that they are visited. For example the list A,B,C,D,E means that city A is visited first followed by B then C and so on. The tour will start and end at city A. This is known as path representation.[1]

Binary Representation

Each binary value represents a different tour. As shown in table 1, the tours are numbered in alphabetical order. The algorithm for translating between the representation is described by A. Mohebifar [13].

The difference between the positions of the cities in an initial tour (e.g. ABCDE) and another tour can be calculated and transformed into binary. The inverse of this can be used to translate from binary to the order list. The algorithms for these are shown in figure 2 and 3 respectively.

```
public void produceEncodedIndividual()
{
  BigInteger binaryTour = BigInteger.ZERO;
  List<Integer> placesToVisit = new ArrayList<Integer>
                                  (Individual.initialList);

  for (int i = 1; i < this.individual.size(); i++)
  {
    int index = placesToVisit.indexOf(individual.get(i-1));
    binaryTour = binaryTour.add( Individual.getFactor(
                                  this.individual.size() - i)
                                  .multiply(BigInteger.valueOf(index))
                                );
    placesToVisit.remove(index);
  }
  this.encodedIndividual = binaryTour;
}
```

Figure 2: How the binary representation of the individual is created from an ordered list of cities.

```
void produceDencodedIndividual(BigInteger binaryTour)
{
  List<Integer> placesToVisit = new ArrayList<Integer>
                                  (Individual.initialList);
  int numberOfNodes  = Individual.initialList.size();
  this.individual = new ArrayList<Integer>();
  for (int i = 1; i < numberOfNodes; i++)
  {
    Integer node = placesToVisit.get((binaryTour.divide
                                  (Individual.getFactor(numberOfNodes - i)
                                  ).intValue()));
    placesToVisit.remove(node);
    individual.add(node);
    binaryTour = binaryTour.mod(Individual.getFactor(numberOfNodes - i));
  }
  individual.add(placesToVisit.get(0));
}
```

Figure 3: How the order list of cities is created from the binary value.

	Ordered tour	Binary representation
1	ABCDE	0000000
2	ABCED	0000001
3	ABDCE	0000010
4	ABDEC	0000011
...	...	...
120	EDCBA	1111000

Table 1: Binary representation of tours through cities A, B, C, D and E.

The use of binary allows simple genetic operations to be used. However, the additional calculations needed to translate between binary and path representations may cause the GA to be computationally slower than using path representation.

The results will discus and compare the fitness achieved by using binary and path representation. The genetic operators chosen must differ slightly between the two representations; but will be kept as similar as possible to allow a fair comparison to be performed.

2.3 Genetic Operators

2.3.1 Selection

Multi-reserved strategy

Often GAs will converge on a local minima causing them to return suboptimal results. The multi-reserved strategy

tries to prevent this by taking the distance between the individuals into account, as well as the fitness, when performing selection. I have used the ideas presented in [10] and [5] to perform a multi-reserved strategy to solve the TSP. The algorithm can be described as :

- Selected the top k% (eg. 10%) of the population. (potential_reserved_list)
  - Stored the rest of the population in an unreserved_list
- Add the fittest individual (B) to a reserved_list
- For each individual (I) in the potential_reserved_list:
  - if $fitness(B) - fitness(I) < a$ then add I to the reserved_list
  - else if $fitness(B) - fitness(I) > b$ then add I to the unreserved_list
  - else if $dis(B, I) > c$ then add I to the reserved_list
  - else add I to the unreserved_list
- Crossover and mutation will be applied to the two lists. The next generation will be formed of 50% from the reserved_list and 50% from the unreserved_list.
- k, a, b and c are constants defined by the user.

Setting k too small could cause the population to not retain enough variation, but setting it to too high will allow more weaker individuals to be kept, which could cause the fitness to increase. These constants could have a large effect on the performance of the GA.

Tanimoto distance will be used for the binary representation. For path representation the distance will be calculated based on the difference between the indexes the cities appear in the two tours.

This method should allow the fittest individuals that a distributed across a large search space to be kept. The multi-reserved strategy will be compared to rank selection, which is more commonly used in GAs.

Rank selection

In rank selection the fittest individual is the most likely to be picked, the next fittest is the second most likely to be picked and so on. Though it is not guaranteed, this should allow the next generation to be formed mostly from the fittest individuals and a few less fit individuals. Picking less fit individuals should allow a larger area of the search space to be covered. [3]

Table 2 shows the probability of an individual being picked when the population contains 6 individuals.

Index	Probability of being picked
0	11 / 35
1	9 / 35
2	7 / 35
3	5 / 35
4	2 / 35
5	1 / 35

Table 2: Rank selection : probability of an individual being picked. The individual at index 0 is the fittest indivdual and index 5 contains the least fit individual.

Using rank selection may cause the next generation to only contain combinations of the fittest individuals, of the current population. This would lead to only a small area of the search space being evaluated, which is likely to result in a local minim being discovered.

However, preforming a random amount of crossover and mutation should allow a diverse population to be created. The next sections will introduce the crossover and mutation operations that are being used.

2.3.2 Crossover

Consideration was also given to many types of crossover operations, this includes N-Point crossover, edge recombination crossover and sequential constructive crossover.[2] As the focus of the experiments will be on the selection and representation used, a standard one-point crossover is applied. [14] The following algorithm describes how crossing individual X with individual Y works :

- Generate a random number (r) between 0 and probability (c) $*$ length of individual (l)
- if $r < l$ then crossover is performed :
  - swap each item in X, with each item in Y: who's position is less than r

Binary representation

For binary representation crossover is performed by swapping the bits and checking that the binary value produced is less than the maximum number of solutions.

Figure 4: Example of binary crossover

Path Representation

Performing crossover on order lists of cities may cause cities to be visited multiple times. Thus producing on invalid solution. A check is performed to find any cities that a missing from the list. Any repeated cities are replaced by missing cities.

Figure 5: Example of crossover for path representation

2.3.3 Mutation

In order to stop all individuals from converging to the same value; and to widen the covered search area mutation is applied. Mutation that has a small effect on the individual will allow for local search; and large effects will allow individuals to escape local minimas.

For both binary and path representation a low mutation probability is used. This will cause a small percentage of the individuals to change. The amount of change the individual undergoes will be discussed separately for the two types of representation.

Binary Representation

Commonly when mutating binary values a single probability is used to determine if an a independent bit within the individual should be flipped. However, in this situation the more significant a bit is the larger effect it has on the tour.

Therefore, the least significant bit has a $1/(p * 1)$ probability of being mutated and the n-th bit has a $1/(p * n)$ chance of mutation. This will still allow large changes in the tour to take place, but this will rarely occur.

Consideration was given to just adding 1 to the binary value. This would allow the tour to change by a very small amount. However, including larger changes allows local minimals to be escaped.

Path Representation

Mutating the order list of cities is performed using a much simpler method. Two cities within the individual are swapped. This causes a significant change in the tour being performed and should allow the exploration of a large distribution of the search space. Their is a $1/p$ probability of an individual being mutated. This will allow the majority of individuals to not be mutated, therefore information from previous generations will be retained.

3 Experiments

The section will explain the different experiments performed. The results will be presented and the differences between the results for the two representations and two selection operation will be discussed.

3.1 Set-up

The randomly generated cities will be spread across a 200 by 200 pixel sized area. The initial population will also be randomly generated. The initial population will remain the same for all experiments involving a number of cities.

Due to the indeterministic aspects of the GAs each experiment will be run 10 times and the average fitness will be calculated. This coupled with the use of the same population for each experiment should allow for a fair evaluation of the different approaches.

3.2 Results

The result tables contain :

- Representation(Rep.) of the individuals either binary representation (B.) or path representation (P.)
- Selection (Sel.) used will either be Rank selection or the multi-reserved strategy (MRS).
- Number of generations(Gen.)
- The average (Avg.) fitness of the final generation
- The fitness of the least fittest individual from the finial generation (least)
- The fitness of the fittest (F.) individual from the finial generation
- The fitness of the fittest (F.) individual over all generations

3.2.1 Initial Experiments

Some initial experiments were performed to determine what mutation and crossover probability should be used when comparing the representations, and selection operators.

Crossover probability

Crossover	Rep.	Sel.	Fittest
5	B.	Rank	973
		MRS	1035
	P.	Rank	765
		MRS	754
10	B.	Rank	948
		MRS	1021
	P.	Rank	748
		MRS	782
15	B.	Rank	986
		MRS	1055
	P.	Rank	756
		MRS	772
20	B.	Rank	1011
		MRS	1026
	P.	Rank	751
		MRS	762

Table 3: Fittest tour found when using different crossover probabilities. The meaning of these crossover values have been described in section 2.3.2.

After running the GAs on different crossover probabilities, it was decided that a probability of 10 will be used during the rest of the experiments. The fitness of the fittest individual starts to rise when a value greater than 10 is used. (Shown in table 3)

Mutation probability

500 will be used for the mutation probability. A greater or lesser value during the initial experiments caused, on average, a longer tour to be found. Two much mutation causes the generations to lose good information, whereas, too little mutation and individuals will not be able to escape local minimas.

Mutation	Rep.	Sel.	Fittest
400	B.	Rank	1132
		MRS	1201
	P.	Rank	830
		MRS	845
450	B.	Rank	1155
		MRS	1213
	P.	Rank	817
		MRS	855
500	B.	Rank	1103
		MRS	1182
	P.	Rank	800
		MRS	853
550	B.	Rank	1135
		MRS	1140
	P.	Rank	868
		MRS	897
600	B.	Rank	1142
		MRS	1190
	P.	Rank	864
		MRS	894

Table 4: Fittest tour found when using different mutation probabilities. The meaning of these values have been described in section 2.3.3.

3.2.2 Representation and Selection Experiments

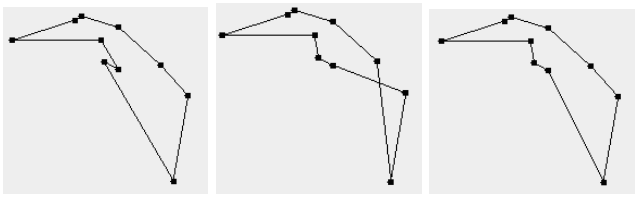
The experiments comparing binary representation to path representation, and multi-reserved strategy to rank selection, were performed using a range of different amounts of cities. The first of these will look at 10 cities being used.

10 cities

When the multi-reserved strategy is used, with either representations, a much higher least fittest value is seen for the final generation. Due to the genetic operations the least fittest individual changes considerably between generations. (Figures 7 to 10). Therefore, it is likely that these experiments have coincidently converged on a higher least fittest, in comparison to rank selection.

Rep.	Sel.	Gen.	In final generation			F.
			Avg.	Least	F.	
B.	Rank	513	544	637	541	541
	MRS	694	559	787	533	533
P.	Rank	54	534	534	534	633
	MRS	62	534	607	531	531

Table 5: Number of cities : 10, Population size : 100



(a) Solution found during many of the experiments. (b) Path representation using rank selection (c) Path representation using MRS

Figure 6: A selection of different best tours found from when using 10 cities.

For many of the runs, the different experiments resulted in the same tour being found (figure 6a). However, this is not the shortest tour possible. From figure 7 and 8 it is clear that the average and fittest fitnesses have decreased before then levelling off. Which provides evidence that a local minim has been reached.

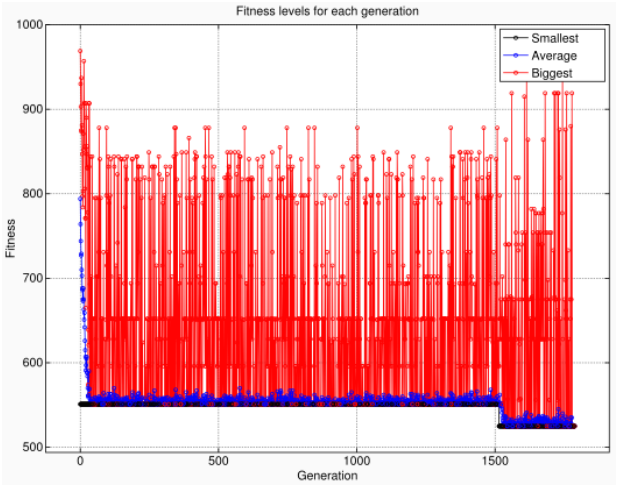


Figure 7: Binary representation using rank selection. This experiment resulted in the tour shown in figure 6a.

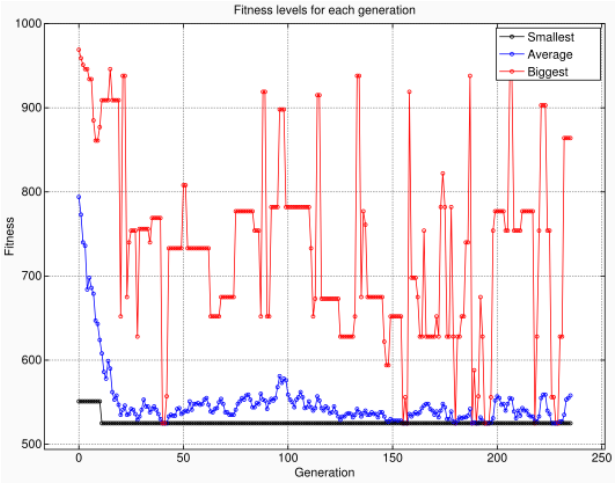


Figure 8: Binary representation using MRS. This experiment resulted in the tour shown in figure 6a.

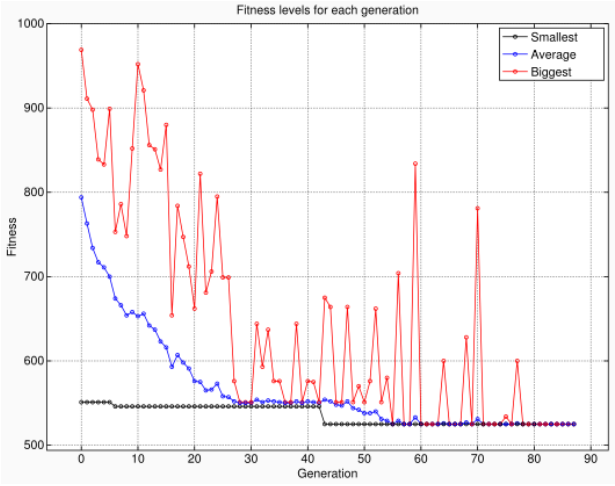


Figure 9: Path representation using rank selection.

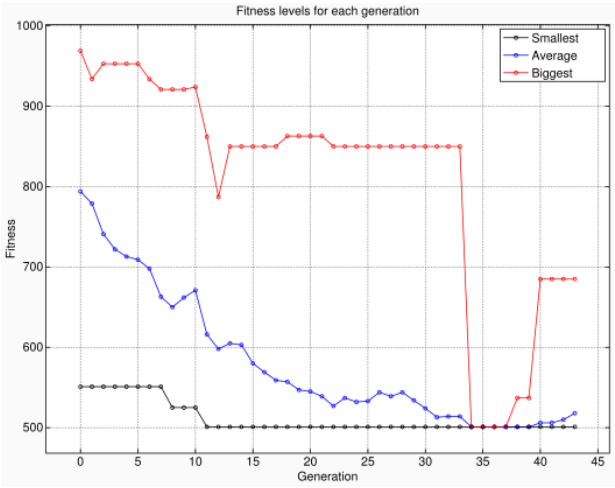
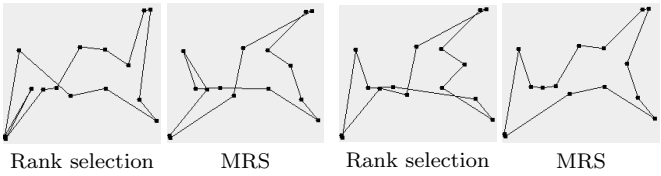


Figure 10: Path representation using MRS

The path representation with rank selection has converged to a longer tour than the other experiments. As the average fitness has levelled off (figure 9) it is likely that a local minima has been researched. A possible reasons for this is that not enough variation and mutation occurred in order to search a wide distribution of the search space. With an average of only 54 generations, this experiment converged much quicker than the other experiments; potentially causing a smaller area of the search space to be covered.

Their is only a small difference in the fitness of the fittest individual for the multi-reserved strategy experiments (table 5). Path representation using the multi-reserved strategy appears to, on average, find a fitter individual. However, as such a small difference is seen further experiments are need to see if this holds true for a greater number of cities.

15 cities



(a) Binary representation (b) Path representation

Figure 11: Selection of tours produced for 15 cities

On average when using 15 cities, path representation has converged on a shorter tour than binary representation. The path representation which uses multi-reserved strategy results in the shortest found tour.

Rep.	Sel.	Gen.	In final generation			F.
			Avg.	Least	F.	
B.	Rank	3426	870	1538	855	838
	MRS	1796	916	1692	842	842
P.	Rank	118	833	982	831	822
	MRS	282	783	1080	771	771

Table 6: Number of cities : 15, Population size : 200

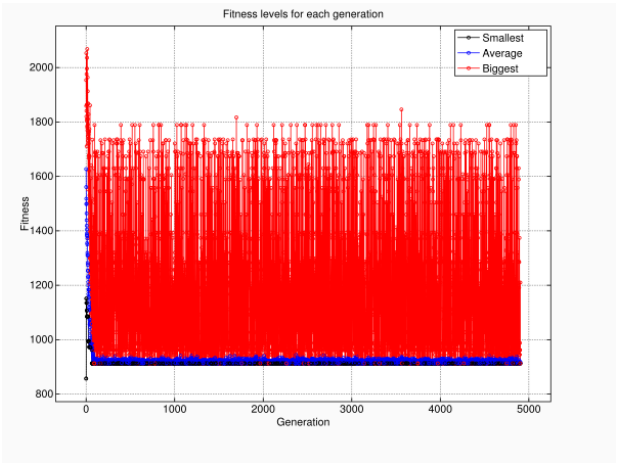


Figure 12: Binary representation using rank selection.

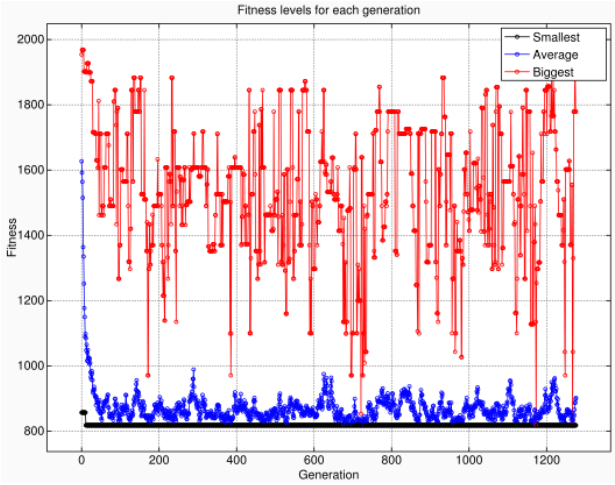


Figure 13: Binary representation using MRS

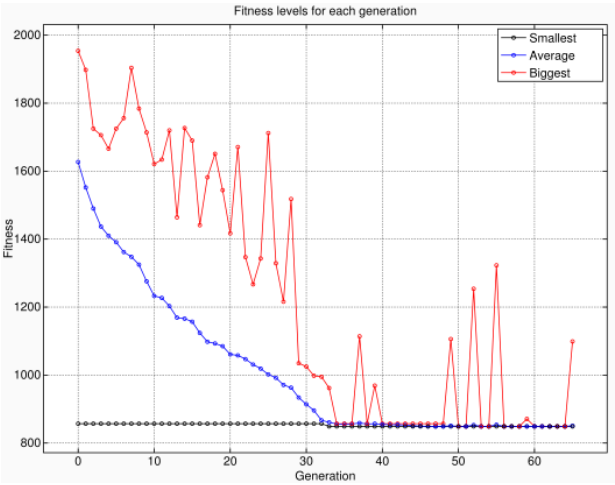


Figure 14: Path representation using rank selection.

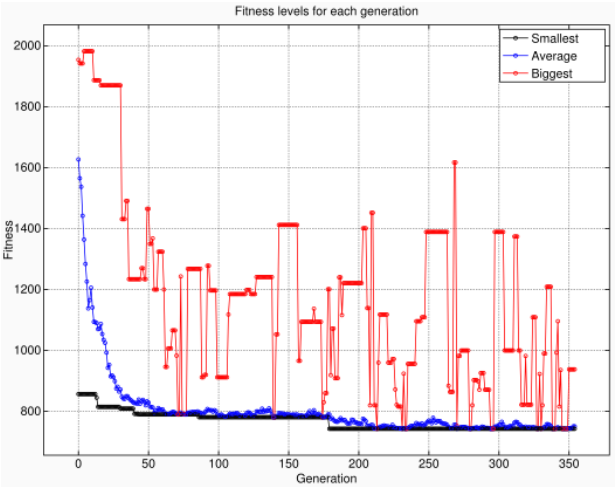


Figure 15: Path representation using MRS

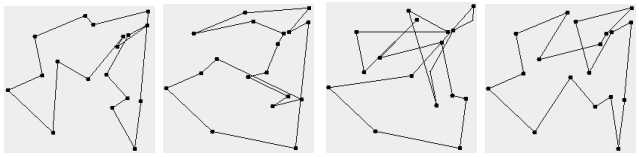
it was predicted that the multi-reserved strategy would take more generations to converge. The distance between individuals is taken into account when performing this selection. Therefore, individuals with a lower fitness are more likely to be kept than when rank selection is used. This should allow a larger search space to be explored. However, binary representation using rank selection takes more generations to converge. Putting an even higher weighting on the elite individuals may allow this to converge using fewer generations.

When using the multi-reserved strategy the least fit individual is significantly higher than when rank selection is used. 50% of the population is made from the non-reserved list, therefore there is a very high chance that an unfit individual will make it into the next generation.

When only 15 cities are used there is still very little difference between 3 of the average fittest individual's fitness. However, the path representation using multi-reserved strategy, on average, has given the shortest tour. Adding more cities will hopefully cause more variation within the results.

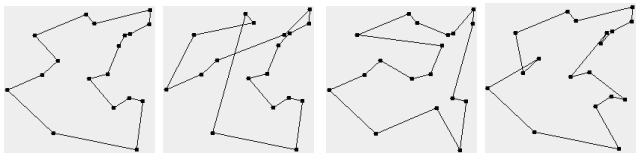
18 cities

As we increase the number of cities to 18 it becomes clearer from both the resulting tours (figures 16 and 17) and the average results (table 7) that the path representation results in a shorter tour being found.



(a) Rank selection. (b) MRS

Figure 16: Binary representation



(a) Rank selection. (b) MRS

Figure 17: Path representation

During the experiments involving 10 and 15 cities the multi-reserved strategy has converged on a shorter tour than the rank selection. When 18 cities are used we see this flip and rank selection starts to find shorter tours. This will be discussed further in the experiments that involve 20 cities.

Rep.	**Sel.**	**Gen.**	**In final generation**			**F.**
			Avg.	**Least**	**F.**	
B.	Rank	4613	1070	1519	1065	1065
	MRS	1961	1231	1941	1179	1179
P.	Rank	318	915	1092	914	913
	MRS	400	1007	1361	997	997

Table 7: Number of cities : 18, Population size : 500

20 cities

The next experiments explore what happens when the number of cities in increased to 20. Just like the experiments with 15 and 18 cities, the path representation has found shorter tours and taken fewer iterations than binary representation.

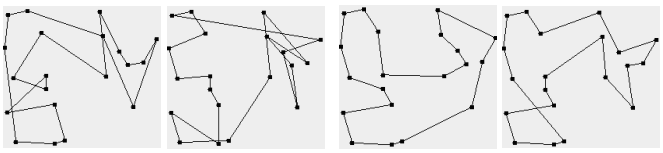
When using binary representation the rank selection reached the maximum number of generations. There has been no change in the fitness of the fittest individual for many generations, but as the least fittest individual changes rapidly so does the average fitness. This causes the GA to converge slowly. (Shown in figure 19.)

Rep.	Sel.	Gen.	In final generation			F.
			Avg.	Least	F.	
B.	Rank	5000	1199	2066	1191	1191
	MRS	1623	1289	2361	1218	1218
P.	Rank	627	935	1297	933	933
	MRS	575	997	1481	985	985

Table 8: Number of cities : 20, Population size : 1000

The mutation and crossover could be having a larger effect on the tours produced by the binary representation than the path representation. Changing the more significant bits of the binary representation can completely change the tour. When fewer cities are used the effect of this is less noticeable. Therefore, in the experiment with just 10 cities the results were very similar, but with 20 cities we see a large difference. These large changes will prevent local area search.

As individuals change a lot this causes the average fitness to remain unstable, so the GA will take longer to converge. This can be see in figure 20. The GA has only converged because the average fitness has risen for more than 10 generations. Though the least fit individual's fitness does fall slightly at the start, in comparison to the path representation (figures 22 and 21) there is a greater difference between this and the average fitness.



(a) Binary representation (b) Path representation

Figure 18: Selection of tours produced for 20 cities

In contrast to previous experiments the rank selection has out performed the multi-reserved strategy, but only a small difference is seen. From one experiment it was unclear if any conclusions could be drawn. Therefore, the experiments with 20 cities were run using several different city positions. There appears to be a 50/50 chance of multi-reserved strategy converging to a shorter tour than rank selection. Therefore, no clear conclusion can be drawn between which selection operator is better at solving a 20 city TSP.

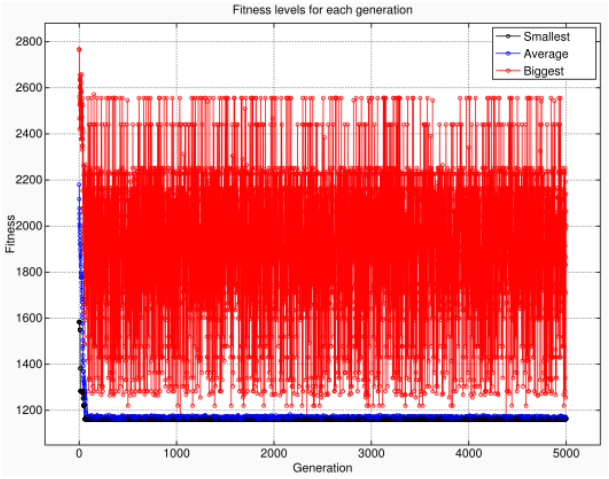


Figure 19: Binary representation using rank selection.

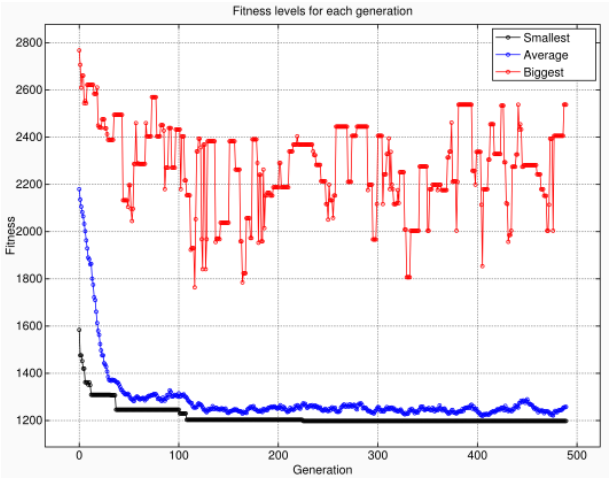


Figure 20: Binary representation using MRS.

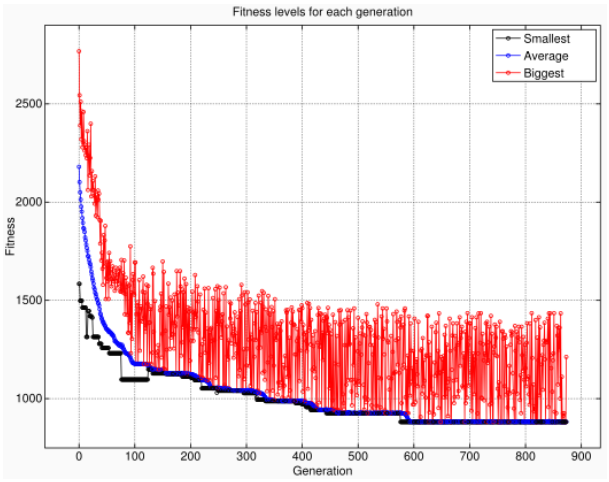


Figure 21: Path representation using rank selection.

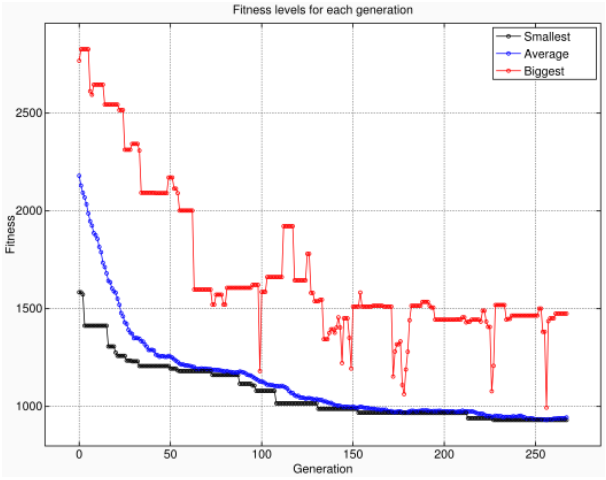


Figure 22: Path representation using MRS.

Overtime the path representation least fittest, fittest and average fitness has fallen steeply before levelling off. The least fittest line shows that the genetic operations have allowed variations in the tours to enter the population. This helps local minims to be escaped. Though the algorithm can still fall into minims, often a correct, or close to correct, tour is discovered.

3.2.3 Experiment Summary

The number of generations needed before the GA converges is greater for binary representation than path representation. When the average fitness of a generation does not fall for 10 generations then the GA stops. The average fitness remains far less stable in the binary representation (figure 20) than in the path representation (figure 22). It is likely that the mutation probability has been set too high causing too much change in the individuals.

Each experiment was run 10 times and the averages of these were used when discussing the results. However, during these 10 runs a wide range in the number of generations was seen. For the experiment on 20 cities using path representation with rank selection the number of generations ranged from 153 to 1035. When the number of generations reached 1035 the last drop in the fittest individual was seen around the 350th generation. Therefore perhaps an alternative stopping criteria should have been used. For example it could be based on the number of generations where the fittest individual has not changed.

As more cities are added the tours discovered by the binary representation quickly become far worse. Whereas, the path representation becomes worse at a much slower rate and can still sometimes find the shortest tour. Experiments with large numbers of cities were performed and these experiments continued to become less likely to find the shortest tour.

The experiments have shown that, for the genetic operations chosen, path representation finds a shorter tour and uses less generations, than binary representation. The multi-reserved strategy takes more generations to converge than the rank selection, however similar tour lengths are found.

3.2.4 Improvements to experiments

Rather than using a value for mutation and crossover probability that is in the form of $1/p$, a value in the range 0 to 1 should have been used. This would make it more user friendly as 0 would cause no mutation/crossover to take place; and 1 would cause every binary value, or ordered list, to mutate/crossover. This should not impact the validity of the results produced.

4 Conclusion

4.1 Future experiments with GAs

Experiments using the binary representation with different types of crossover operations could be performed. One-point crossover causes a large change to the individuals, which prevents local area being searched. Using two-point crossover may allow a shorter tours to be discovered.

The GA produced checks that no individuals are the same within a single generation, but it does not check that the individuals are unique within the life time of the GA. These means that the fitness function is potentially being ran many times using the same input. Future work could consider remembering the fitness. This will add a large memory overhead, but may save on computational time.

The initial population produced may not be distributed across a wide area of the search space. This may cause the population to be trapped within a local search area. Experiments using a Gaussian or Bernoulli distribution could be performed.

The experiments used show that when using path representation there is a good chance that the shortest tour will be found. However as more cities are added this becomes less and less likely. The GA is likely to discover a local minim. Alternative methods for solving TSP should be considered.

4.2 Alternative methods

Swarm intelligence would allow a large area of the search space to be explored and individuals would be able to adapt according to how other individuals perform. This should lead to all individuals converging of the minim value. There are many forms of swarm intelligence including Ant Colony Optimization (ACO) and Particle Swarm Optimization (PSO).

Chen et. al. [15] has compared their method, Gene Regulation Particle Swarm Ant Colony Optimization (GRPSAC), to ACO, PSO and a GA to find out which method is better at solving the TSP. Their results show that for 30 cities ACO achieves the worst solutions, followed by GAs. When using 48 cites the GA, on average, has the worst results.

Chen et. al., and related work, has shown that swarm intelligence can perform much better at solving TSP than GAs. Therefore, further research such look at adapting and improving swarm intelligence methods in order to achieve more optimal solutions for the TSP.

References

[1] Otman Abdoun, Jaafar Abouchabaka, and Chakir Tajani.
Analyzing the performance of mutation operators to solve the travelling salesman problem.
arXiv preprint arXiv:1203.3099, 2012.

[2] Zakir H Ahmed.
Genetic algorithm for the traveling salesman problem using sequential constructive crossover operator.
International Journal of Biometrics & Bioinformatics (IJBB), 3(6):96, 2010.

[3] James Edward Baker.
Adaptive selection methods for genetic algorithms.
In *Proceedings of an International Conference on Genetic Algorithms and their applications*, pages 101–111. Hillsdale, New Jersey, 1985.

[4] Yang Chen, Jinglu Hu, Kotaro Hirasawa, and Songnian Yu.
Gars: An improved genetic algorithm with reserve selection for global optimization.
In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, GECCO '07, pages 1173–1178, New York, NY, USA, 2007. ACM.

[5] Yang Chen, Jinglu Hu, Kotaro Hirasawa, and Songnian Yu.
Solving deceptive problems using a genetic algorithm with reserve selection.
In *Evolutionary Computation, 2008. CEC 2008.(IEEE World Congress on Computational Intelligence). IEEE Congress on*, pages 884–889. IEEE, 2008.

[6] William Cook.
Milestones in the solution of tsp instances.
Avaliable: `http://www.math.uwaterloo.ca/tsp/history/milestone.html`, 2015.
Accessed: [29 November 2015].

[7] S.M.Johnson G.B.Dantzig, D.R.Fulkerson.
Solution of a large scale traveling salesman problem.
In *Technical Report*, page 510. RAND Corporation, Santa Monica, California, USA, 1954.

[8] Khalid Jebari and Mohammed Madiafi.
Selection methods for genetic algorithms.
International Journal of Emerging Sciences, 3(4), 2013.

[9] Pascal Lehwark.
Mlplot.
Avaliable : `http://sourceforge.net/projects/mlplot/`, 2011.
Accessed: [01 December 2015].

[10] Fachao Li, Li Da Xu, Chenxia Jin, and Hong Wang.
Intelligent bionic genetic algorithm (ib-ga) and its convergence.
Expert Systems with Applications, 38(7):8804 – 8811, 2011.

[11] Adam Marczyk.
Genetic algorithms and evolutionary computation.
Avaliable: `http://www.talkorigins.org/faqs/genalg/genalg.html`, 2004.
Accessed: [06 October 2015].

[12] Thomas M. Mitchell.
Machine Learning.
McGraw-Hill, Inc., New York, NY, USA, 1 edition, 1997.

[13] Amin Mohebifar.
New binary representation in genetic algorithms for solving tsp by mapping permutations to a list of ordered numbers.
WSEAS Transactions on Computers Research, 1(2):114–118, 2006.

[14] Riccardo Poli and William B Langdon.
Schema theory for genetic programming with one-point crossover and point mutation.
Evolutionary Computation, 6(3):231–252, 1998.

[15] Yong-Qin Tao, Du-Wu Cui, Xiang-Lin Miao, and Hao Chen.
An improved swarm intelligence algorithm for solving tsp problem.
In De-Shuang Huang, Laurent Heutte, and Marco Loog, editors, *Advanced Intelligent Computing Theories and Applications. With Aspects of Artificial Intelligence*, volume 4682 of *Lecture Notes in Computer Science*, pages 813–822. Springer Berlin Heidelberg, 2007.

[16] Eric Weisstein.
Traveling salesman problem.
From MathWorld–A Wolfram Web Resource : `http://mathworld.wolfram.com/TravelingSalesmanProblem.html`, 2014.
Accessed: [01 December 2015].

[17] David P. Williamson.
The traveling salesman problem: An overview.
Avaliable: `http://people.orie.cornell.edu/dpw/talks/EbayJan2014.pdf`, 2014.
Accessed: [29 November 2015].