Scientific Programming in R Assignment 3

Student 577747

1. Abstract

Three methods of solving the travelling salesman problem (simulated annealing, elastic net and genetic algorithms) were implemented and comprehensively evaluated on artificially generated and ‘real-life’ datasets (10 different datasets total). The performance of these algorithms on the datasets were used to compare parameter combinations for the algorithms themselves, and later to draw comparisons between the three algorithms. Elastic net and simulated annealing produced the best general performances (*$p=0.003$*), but inefficiency on the part of elastic net resulted in the conclusion that simulated annealing was the best method overall in terms of accuracy, efficiency and ease of implementation/understanding.

2. Introduction

The Travelling Salesman Problem (TSP) is a classic exercise in computation which encompasses key concepts of mathematics including gradient descent and energy minimisation. In this report, three methods of solving this problem making use of the programming language R were demonstrated: Elastic Net, Simulated Annealing, and Genetic Algorithms. The algorithms were built and tested on datasets of randomly generated coordinates to understand how their parameters affected the outcome. When optimal parameters were obtained for each algorithm, datasets of solved TSPs were obtained from the Heidelberg University web repository[1] and solved using each method, with the solutions obtained compared to the optimal paths in terms of distance. Finally, the performance of each method was compared to each other and the optimal solutions to gain insights as to the suitability of each method for a given dataset based on parameters such as dataset size, time constraints etc. For the purposes of this report, the term ‘performance’ is used as a moniker for the ability of a given method to minimise path distance through the cities of a map.

As stated above, to tune the algorithms, three random datasets (map, map2, map3) of 20 ‘cities’ with xy coordinates uniformly distributed were created and used during development and testing of all methods. While the ‘optimum path’ was unknown due to the de novo creation of the datasets, it was not explicitly required during parameter iteration, as the general aim is straightforward distance minimisation. The distances recorded for parameter iterations were standardised against distances obtained from the same map to account for inherent variation in the distances between the maps, as this would affect the outcome of an ANOVA[2] test for significance by artificially inflating the sum of squares within groups. Following algorithm tuning, the ulysses22 dataset was used as a final test, where a tolerance of 20% of the optimal path length was exercised.

To compare the performance of the three methods, five Heidelberg datasets were selected (ulysses16, ulysses22, att48, eil51 and berlin51). These datasets were chosen for their smaller length in the interests of efficiency as it was found during the individual development phase that the time taken for each algorithm (particularly Elastic Net) increased greatly with number of cities. However, less exhaustive trials were completed with the datasets lin105 and tsp255 to gain a complete picture of algorithm performance even with larger datasets.

It was noted early in the process that dataset normalisation for appeared to result in greater performance. Normalisation is a prerequisite for elastic net, but is not often indicated as a necessary step in the literature for Genetic Algorithms and Simulated Annealing. Despite this, the decision was made to normalise the datasets for each method to minimise variation for the inter-method comparison.

3. Simulated Annealing

3.1 Algorithm Development and Tuning

Simulated annealing, in the context of the TSP, is an algorithm that begins with a random path through the points in the map and iteratively makes adjustments to the path[3], accepting adjustments that result in a smaller path length, and conditionally rejecting adjustments which result in a larger path length. During early iterations, there is a certain probability that 'unsuitable' moves may be accepted to avoid the possibility that the path could fall into a local minimum. This likelihood decreases over time, due to the relationship between path fitness and a temperature T which 'cools' over time, i.e. decreases in value. This method is derived from the use of controlled cooling of molten steel to create particular metal formations, and is somewhat mimicked in nature by differential patterning of igneous rock based on rate of cooling after volcanic events (the phenomenon behind the Giant's Causeway, Northern Ireland).

The final code for the Simulated Annealing method is available in Section 8 in the simulatedannealing() function (and it's associated functions under Housekeeping functions). Three parameters were chosen for the tuning and optimisation of this algorithm: T, cooling rate and N (number of iterations), and these parameters were varied and tested on the three randomly generated maps described in section 2.

Values of T between 50 and 200 separated by intervals of 25 were used to calculate a path for the test maps, and the resulting distances recorded and statistically analysed (Fig.1). Interestingly, no significant correlation was found between the value of T and algorithm performance *(p=0.357)*, however it does appear that beyond T=100 the algorithm consistency destabilises (note, the larger confidence intervals). A similar lack of conclusive significance was found for varying the rate of cooling of T (i.e., the magnitude of decrease after each iteration *(p=0.55)*. This is contrary to what would be expected and will be examined in more detail in the discussion (Section 6).

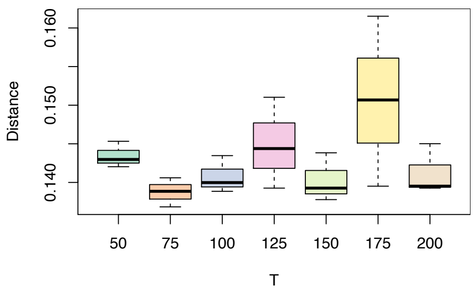

Fig.1 Standardised path distance of maps generated by simulated annealing with differing values of T where all other variables are held constant.

However, examination of the impact of N on the performance proved more fruitful, as a visible improvement in performance was noted from 100 to 1,000 iterations *(p<0.01)* (Fig.2). Interestingly, only 10 iterations resulted in a shorter path than 100, however this is more than likely due to the fact that T would still be reasonably high during the first 100 iterations with longer paths chosen as a result. Essentially, this algorithm results in the path distances initially getting longer before eventually shortening and settling upon an optimum, thus explaining the longest path lengths being after a short, but not miniscule number of iterations.

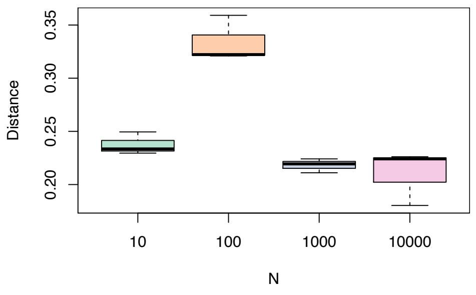

Fig.2 Standardised path distance of maps generated by simulated annealing with differing values of N.

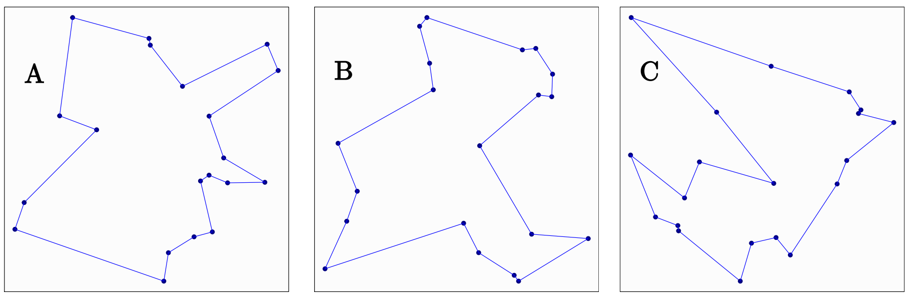

Fig.3 Shortest path for maps 1 **(a)**, 2 **(b)**, and 3 **(c)** found by the simulated annealing method.

3.2 Case Study - Ulysses 22

Upon completion of the algorithm building and tuning with artificially generated maps, the resulting simulated annealing function was used to generate a solution for the real-world problem Ulysses22, for which the optimal path is known. The results are discussed below.

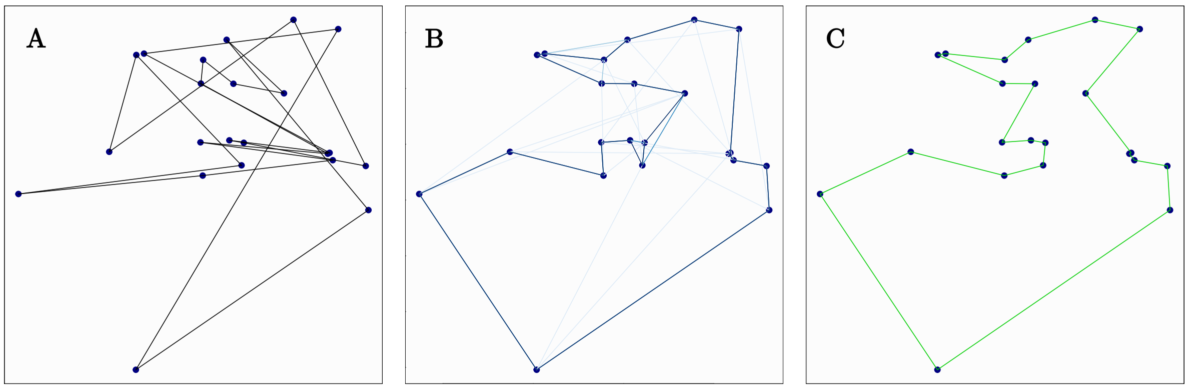

Fig.4 Random **(a)**, Simulated Annealing **(b)**, and Optimum **(c)** paths through the Ulysses 22 dataset, coloured black, blue and green respectively (this colour scheme will be maintained for the remainder of the report). Lighter blue lines in **4(b)** show the iteration process of the algorithm, i.e. the path at earlier iterations during the process.

The simulated annealing method found a path through Ulysses22 that produced a distance of 79.67 units, only 1.05x that of the optimal path (75.67). Given that the length of a randomly generated path through the map is 177.91, 2.35x that of the optimal path, it can be assumed that the simulated annealing method makes a respectable attempt at solving the problem, and performance could increase with further parameter iteration.

4. Elastic Net

4.1 Algorithm Development and Tuning

Of the algorithms discussed in this response, the Elastic Net method is somewhat unique in a number of ways. Rather than starting with a random path and making adjustments until the distance converges to a minimum, Elastic Net utilises a 'net' of points (the number of which are greater than the number of cities) initiated in a small circle that are attracted to the cities on the map and each other by magnitudes controlled by the coefficients alpha α and beta β respectively[4]. With each iteration, the coordinates of points in the net are moved according to weights calculated using these forces, and a distance coefficient K (which deprecates over time), until the net converges to a path through the cities that is kept to a minimum by the force of attraction between the points in the net. In this way, the algorithm is structured more like a neural network.

Three parameters were examined during the tuning of the Elastic Net algorithm (elasticnet() function in Section 8): K; the rate of deprecation of K (cooling); and M (the factor by which the number of points exceeded the number of cities). Initially, four values of K were chosen for investigation (0.1-0.4 by increments of 0.1). However, it was found that when K increased beyond 0.25, the function resulted in an error. This is possibly as an increase in K, which is an argument to the Φ function for calculation of weights w_{ij} but also a coefficient of the force between points, results in the points of the net being too loose due to large, less specific weights, and overinfluence by the coordinates of the cities. While the values for the 'cooling' of K (0.99, 0.9, 0.75) and the values of M (2N, 3N, 4N where N = number of cities) did not affect the final path distance, the time taken to reach this path did vary. For example, higher values for M resulted in exponentially longer run times for the algorithm.

The lack of notable change in path length for this algorithm with varying parameters suggest a number of possibilities: Elastic Net may be more robust to parameter variation than other algorithms, with this consistency being a virtue given that the distance calculated was shorter than that from simulated annealing (section 3) on the same maps (Figure 5).

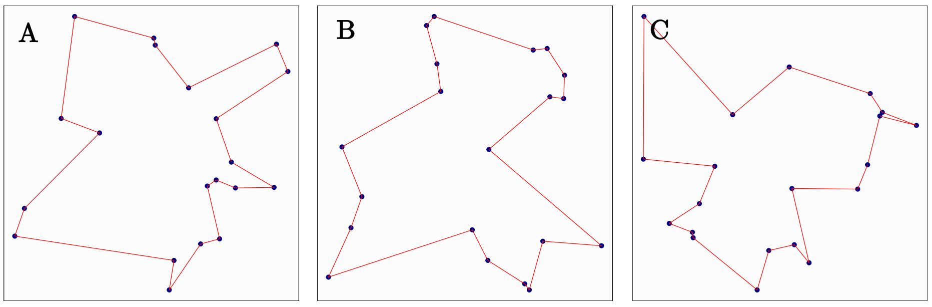

Fig.5 Shortest path for maps 1 **(a)**, 2 **(b)**, and 3 **(c)** found by the elastic net method.

4.2 Case Study - Ulysses22

Ulysses22 was again used as a test of the optimised algorithm, where it was found that the Elastic Net method resulted in a path length of 86.37 versus the optimum of 75.67 (1.14x). Interestingly, the path found by elastic net is notably different to the optimal (Fig. 6b vs 6c), despite the relatively close distances. Different parameter iterations of the elastic net method varied the processing time but did not change the general shape from that seen in Figure 6b. It is possible that the center point of the opening circle (pictured) places the system at a local minimum which the force between the points in the net are unable to overcome. It was not feasible to do a comprehensive analysis of how adjusting alpha and beta could affect the outcome beyond some minor changes – this is a possible future line of inquiry which may result in the system overcoming the local minimum and assuming the shape seen in the optimum path (Fig 6b).

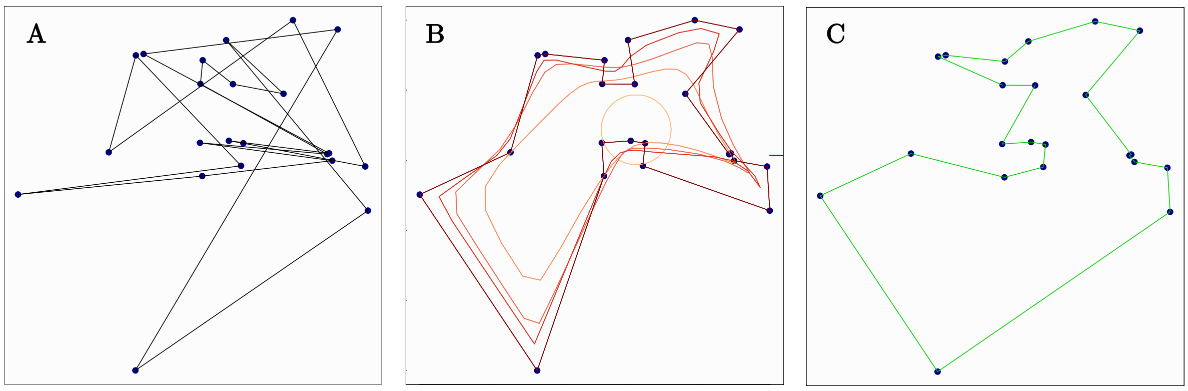

Fig.6 Random **(a)**, Elastic Net **(b)**, and Optimum **(c)** paths through the Ulysses 22 dataset, coloured black, red and green respectively. Lighter red lines in **4(b)** show the iteration process of the algorithm, i.e. the path at earlier iterations during the process.

5. Genetic Algorithms

5.1 Algorithm Development and Tuning

The Genetic Algorithm function is modelled from the positive selection of advantageous alleles in a biological population[5]. A 'population' of paths is maintained at all times, and replaced at each generation with a new population reflective of the 'fittest' members of the previous population, with some mutations. In this instance, fitness is a function of distance, where smaller distance = higher fitness. An additional feature that can be included in this algorithm is 'crossover' where two random members of the previous population (selected proportional to fitness) are combined to produce two members of the new population with inverse combinations of their ancestors, similar to the crossover events that take place during gametogenesis.

For this method, the parameters of N (number of iterations/generations), population size and mutation rate were varied and examined for how they affected performance. Larger population size did see a moderate increase in performance, however this was punished by a sharp increase in time elapsed per iteration. Initially, a varying mutation rate was attempted, where mutation rate would decrease with each generation. However, this did not see the performance increase that was expected, and instead a nominal rate of 6 mutations per individual was chosen.

Interestingly, the method was demonstrated to improve in performance proportional to the logarithmic increase of generations N *($p<0.003$, ANOVA & TukeyHSD)*. This differs from what was seen in Elastic Net and Simulated Annealing, where increases of N beyond where the system reaches a minimum did not see any improvement. The genetic algorithm method appeared to be slower at finding the optimum path: at 10,000 iterations, the path distance was still almost 2x the solution found by simulated annealing after 2,000 iterations. The genetic algorithm method may be more reflective of nature in this way: gradually evolving and improving at a relatively constant rate over time[6].

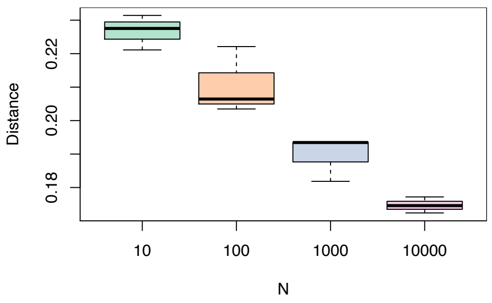

Fig.7 Standardised path distance of maps generated by genetic algorithms with differing values of N.

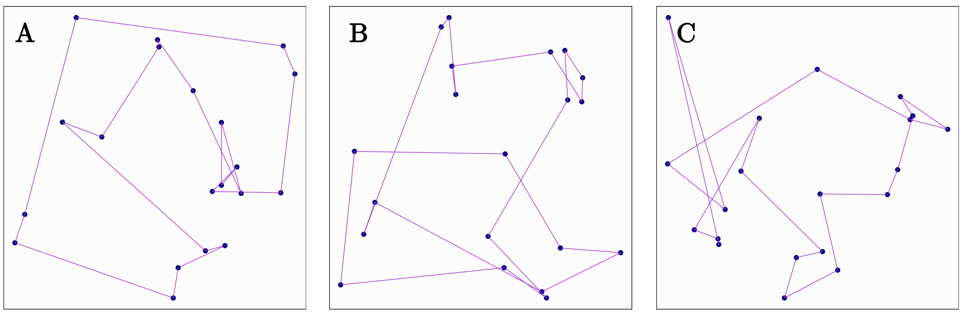

Fig.8 Shortest path for maps 1 **(a)**, 2 **(b)**, and 3 **(c)** found by the genetic algorithms method. The high volume of path crossings indicate that these are evidently not the optimum solutions, but are shorter than a completely random path through the same maps

5.2 Case Study – Ulysses 22

Pursuant to tuning of the parameters, the algorithm was tested on the Ulysses 22 dataset. The results obtained were adequate, but far from the optimum solution (path distance = 1.38x optimum versus random path distance = 2.35x optimum solution). It is possible that a greater number of iterations would result in a better outcome (results below obtained at 10000N). To carry out longer runs, certain steps would need to be taken to optimise the runtime in the interests of efficiency.

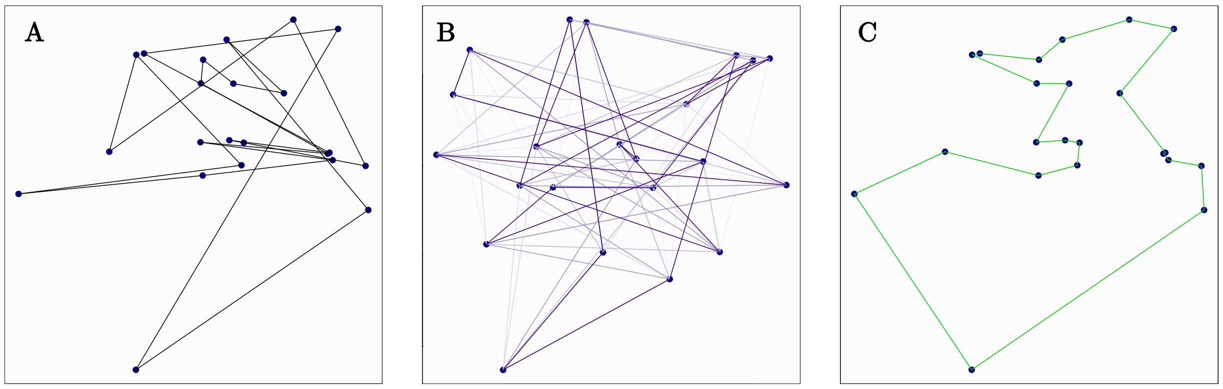

Fig.9 Random **(a)**, Genetic Algorithms **(b)**, and Optimum **(c)** paths through the Ulysses 22 dataset, coloured black, purple and green respectively. Lighter purple lines in **9(b)** show the iteration process of the algorithm, i.e. the path at earlier iterations during the process.

6. Inter-Method Comparison

6.1 Accuracy

Figure 10 displays a comprehensive testing of the three methods (SA - Simulated Annealing, EN – Elastic Net, GA – Genetic Algorithms) on 5 datasets obtained from the Heidelberg Repository. Ulysses 22 was also tested as mentioned in the earlier sections, but in the interests of space is not included in the figure as the visualisations are available earlier in this report.

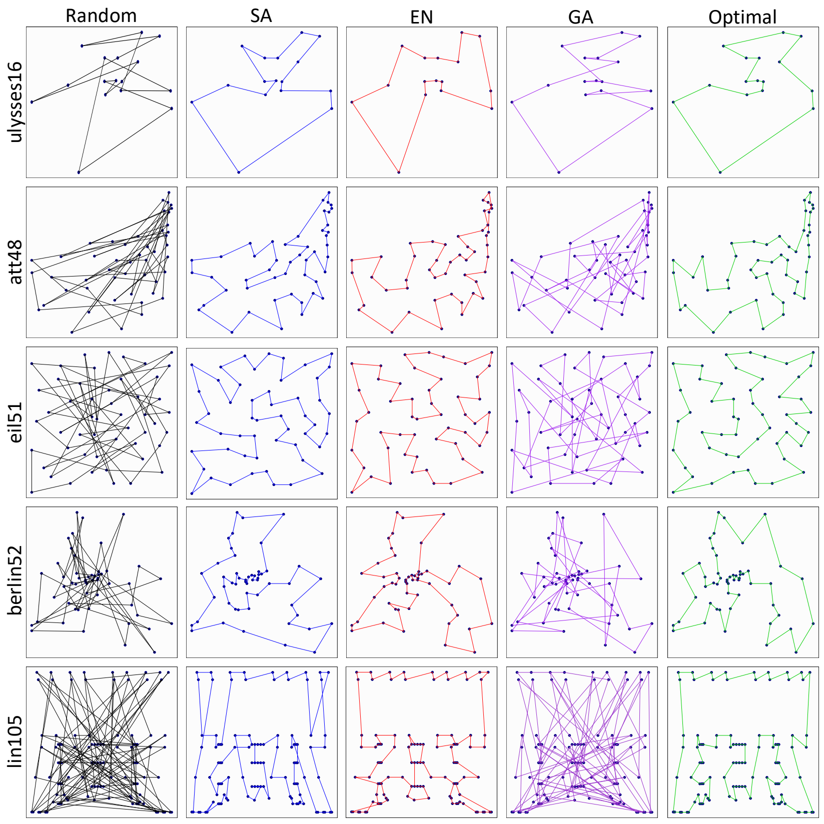

Fig.10 Random (black), Simulated Annealing (blue), Elastic Net(red), Genetic Algorithm (purple) and Optimal (green) paths through five maps of varying city number.

Anecdotally, it is evident from the graphs that each method resulted in a better path than a simply random selection. The degree of this improvement appears to depend on different factors for each method. For example, the genetic algorithm (purple) method accuracy decreases drastically and the number of cities in a dataset increases. Simulated annealing and elastic net proved to be more consistent, although the outcome of elastic net was at time affected by the general shape of a given map, with clusters of cities increasing likelihood of a path crossover like that visible in the top right corner of image att48xEN in Figure 10.

Table 1 (below) displays the distances obtained from each method for each dataset in Fig. 10. These distances were normalised by finding the foldchange vs the optimal distance (for example: ulysses16xSA = 80.33 / 74.11 = 1.08). These data points were then used to conduct a number of statistical tests to evaluate the performances of each network against one another and against the positive and negative controls of the random and optimal paths.

Dataset	**Random**		**SA**		**EN**		**GA**		**Optimal**
	Distance	*Std*	*Distance*	*Std*	*Distance*	*Std*	*Distance*	*Std*	
ulysses16	156.63	2.11	80.33	1.08	86.71	1.17	89.51	1.21	74.11
ulysses22	177.91	2.35	79.67	1.05	86.37	1.14	104.29	1.37	75.67
att48	168435.30	5.02	37510.50	1.11	35211.01	1.05	99758.78	2.97	33523.71
eil51	1707.78	3.97	450.18	1.04	438.35	1.01	1251.84	2.91	429.98
berlin52	29849.31	3.96	8243.28	1.09	8269.89	1.09	21289.62	2.82	7544.37
lin105	121544.30	8.45	17867.00	1.24	17312.48	1.2	97138.41	6.75	14383.00

Table 1 Path distances of the maps displayed in Fig. 10. Both the actual distance and standardised distance (distance/optimal distance) are provided.

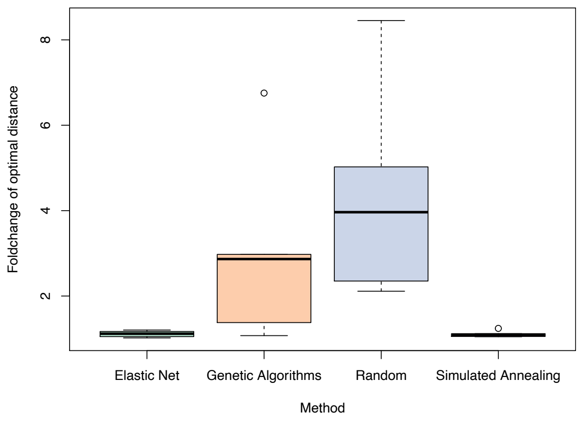

Fig.11 Boxplot of distance foldchange vs optimal for three algorithms and a random path selection (based on data from Table1)

Figure 11 visualises the differences between algorithm performance, which were also analysed for significance with ANOVA and TukeyHSD post-hoc tests. Elastic net and simulated annealing are significantly more accurate than random path generation *(p=0.003)* with genetic algorithms appearing to provide some improvement in path distance but not a statistically significant one *(p=0.47)*. This is likely due to the larger datasets, which genetic algorithm appears to struggle with, elevating the group mean and standard deviation. For each of the datasets used in this analysis and visualised in Fig. 10, a uniform number of iterations of 10,000 was used for genetic algorithms, so if N was to be adjusted to reflect dataset size, this could result in a better outcome for the algorithm.

To further understand how dataset size affects the algorithm performance, see Figure 12. Performance, calculated as 1/normalised distance so that less distance = greater performance are plotted for each algorithm by dataset size. While simulated annealing and elastic net remain consistent as dataset size increases, genetic algorithm demonstrates a sharp decrease, becoming little better than random as the dataset approaches 100 cities in size.

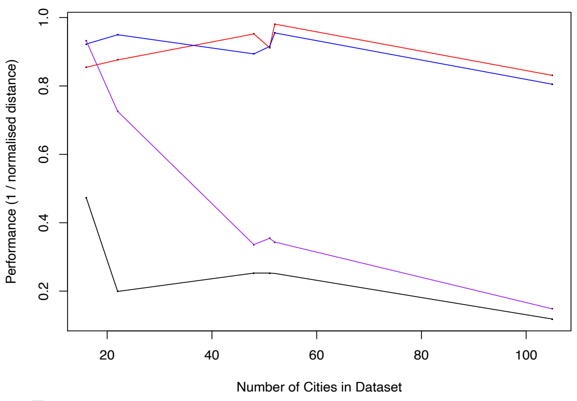

Fig.12 Line chart of performance by dataset size for simulated annealing (blue), elastic net (red), genetic algorithm (purple) and random (black).

6.2 Speed

With an understanding obtained as to the capabilities of each algorithm in terms of accuracy, the next factor to be taken into account is efficiency. Informally it was noted throughout the algorithm development and testing that simulated annealing was the fastest of the three methods, followed by genetic algorithms, with elastic net taking the longest amount of time. It was also found that while the time taken for simulated annealing and genetic algorithm increased incrementally as dataset size increased, time elapsed between each iteration of the elastic net algorithm increased exponentially with larger datasets. For example, 20,000 iterations of the lin105 dataset took the algorithm 12 hours, even when the 'for loops' of the algorithm had been optimised with sapply().

This increase is demonstrated on the line chart in Figure 13. Such was the difference in time elapsed that a logarithmic scale had to be applied to the x axis.

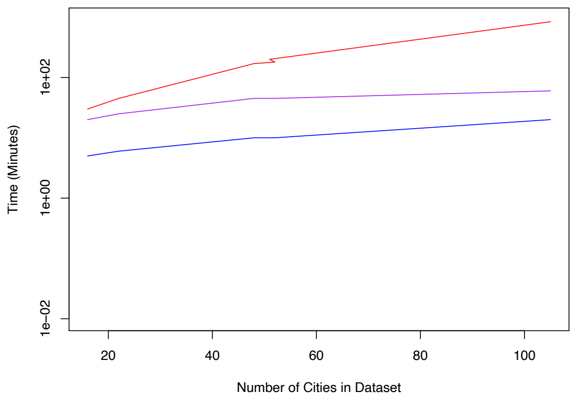

Fig.13 Line chart of time taken by dataset size for simulated annealing (blue), elastic net (red), and genetic algorithm (purple).

7. Discussion

The above report describes the development and thorough assessment of simulated annealing, elastic net and genetic algorithm methods to solve the travelling salesman problem. Throughout the process the performances of these algorithms were contextualised by comparison to each other and to positive and negative controls (optimal path distance where available and random path distance, respectively).

During the development stage, notice was taken of the ease of implementation of the algorithms from a mathematical and programming perspective. All three algorithms were relatively straightforward to implement, with the possible exception of elastic net. The elastic net is quite alternative in its approach to the TSP and uses more advanced mathematical concepts. In addition, there appears to be a smaller volume of peer-reviewed literature referencing the elastic net approach, and a definite lack of general online discourse in comparison to the other two methods chosen. However, when the general logic and mathematical concepts were understood, the algorithm was simple to construct in R, and well suited to optimisation using sapply() in place of for loops. In contrast, the genetic algorithm and simulated annealing functions were straightforward to grasp, but quite dependent as methods on loops through long vectors (such as the population members for genetic algorithm).

During the tuning stage, where algorithms were tested against three randomly generated maps of 20 cities, the effect of certain parameter changes were sometimes less significant than hypothesised. Given that the map sizes were set at 20 cities, it may have been advisable to run replicates of the tuning process with groups of maps varying in city number, in order to investigate the impact of dataset size on parameter suitability. This is a good recommendation moving forward.

While the algorithms were initially tuned to produce a respectable performance on the three randomly generated maps, during the inter-method testing on 'real' datasets some parameters were adjusted slightly to get the best performance for the dataset in question given its size, shape, etc. While this could be considered as introducing potential sources of variation, the position was taken that the inter-test comparisons should consist of paths generated from the best possible performance of each algorithm, which would result in a 'level playing field' of sorts.

Examination of the performances of the methods versus random paths and the optimum paths for the given datasets (Fig. 10, Table 1) allows the conclusion to be drawn that each method implemented achieved at least moderate success when presented with TSP problems. There was no statistically significant difference between the performance of elastic net and simulated annealing, but both methods performed significantly better than random path generation, resulting in a path length of at most 1.2x that of the optimal route for each dataset (Fig. 11).

Given the close performance of simulated annealing and elastic net, the deciding factor was efficiency. While elastic net produced a consistently short path, especially as dataset size increased, the time taken to reach this result was significantly larger than that of simulated annealing (Fig. 13). Therefore, based on this evidence, simulated annealing emerges as the best method for solving a given TSP out of each of the methods attempted in this report.

In this spirit, for a final test, the simulated annealing algorithm was tasked with solving a map expected to be quite challenging: the tsp225 dataset from the Heidelberg repository. This dataset consists of 225 points, distributed unusually throughout the space (i.e. not in a strictly uniform pattern). Rather amusingly, the optimal point through the map draws out the letters 'TSP' (Fig. 14(b)).

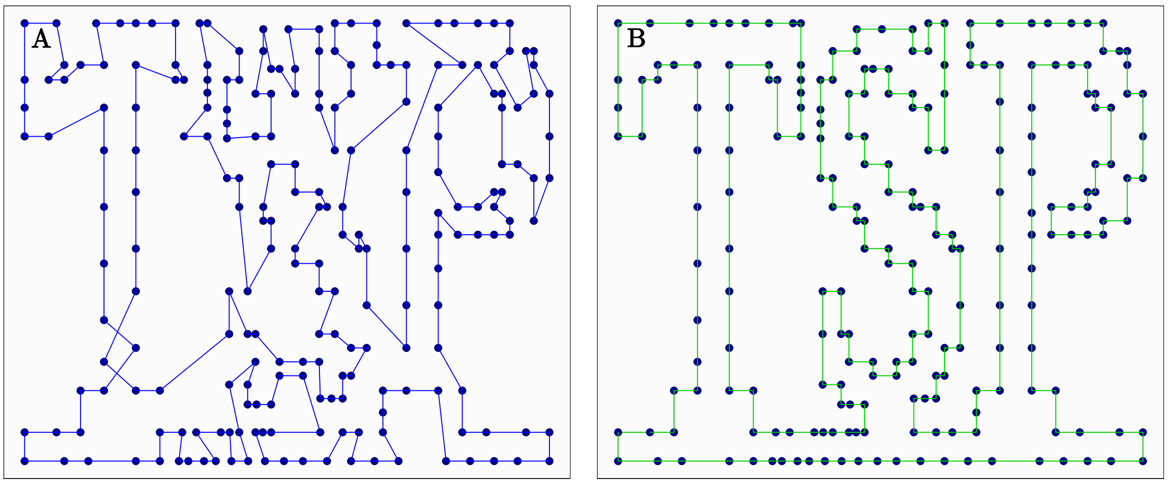

Fig.14 Simulated annealing **(a)**, and optimum **(b)**, paths through the tsp225 dataset.

The simulated annealing algorithm found a path through the map of 4228.01, 1.09x that of the optimal path (3859), an admirable performance for such a large and complex dataset. Some features of the map, specifically the T and P formations, are becoming evident in the simulated annealing route. With more iterations and an adjustment of T (temperature) or the cooling rate, it is possible that the system could approach the optimum solution.

In conclusion, the Travelling Salesman Problem is a superb example of the capabilities of mathematics and computation to solve complex problems efficiently and to a high degree of accuracy. Simulated annealing, elastic net, and genetic algorithms are three possible approaches to the solving of a given TSP, and while each have their merits, simulated annealing is the preferred method following the comparisons performed for this report.

8. Bibliography

1. Teaching. http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/tsp/.
2. aov function | R Documentation. https://www.rdocumentation.org/packages/stats/versions/3.6.2/topics/aov.
3. The Traveling Salesman with Simulated Annealing, R, and Shiny - Todd W. Schneider. https://toddwschneider.com/posts/traveling-salesman-with-simulated-annealing-r-and-shiny/.
4. Durbin, R. & Willshaw, D. An analogue approach to the travelling salesman problem using an elastic net method. *Nature* **326**, 689–691 (1987).
5. Larrañaga, P., Kuijpers, C. M. H., Murga, R. H., Inza, I. & Dizdarevic, S. Genetic algorithms for the travelling salesman problem: A review of representations and operators. *Artif. Intell. Rev.* **13**, 129–170 (1999).
6. Fogel, D. B. An evolutionary approach to the traveling salesman problem. *Biol. Cybern.* **60**, 139–144 (1988).