Title: Genetic Algorithms and the Travelling Salesman Problem

Research Question: How is an optimal found solution for the Travelling Salesman Problem using a Genetic Algorithm affected by the use of Two Point, Edge Recombination, and Sequential Constructive Crossover Operators?

Subject: Computer Science

Word Count: 3974

Table of Contents

1 Introduction

The nature of computer science as a subject has its roots in optimization, as well as finding simpler and more time and space-efficient methods to solve certain problems. Despite this, various problems continue to exist in today's world whose solutions would pave the way for more efficient results for businesses, who don't have any exact solution that can be solved in an efficient amount of time (Rego, 427), one of which includes the Travelling Salesman Problem. Finding an efficient way to solve vehicle routing problems such as the above mentioned can be extremely beneficial to delivering companies to determine the route with the most amount of mileage in the least amount of time. Businesses that have invested in route saving programs have reported up to 40% savings when it comes to driving times and fuel costs (Ma). One such method is the use of a genetic evolutionary algorithm – an algorithm that uses randomly created suggestions in order to combine them, applies a certain mutation, and repeats the process for a certain number of generations until an approximate solution can be determined. Through this, different methods of combining two solutions exist, known as crossover operators. For this reason, the question *'How is an optimal found solution for the Travelling Salesman Problem using a Genetic Algorithm affected by the use of Two Point, Edge Recombination, and Sequential Constructive Crossover Operators?'* will be assessed, as well as to evaluate the need for a genetic algorithm in such cases and the best crossover operator for this specific problem.

2 Travelling Salesman Problem

Formulated in 1930, the Travelling Salesman Problem is a widely popular problem in the field of computer science and optimization mathematics. It imagines a travelling salesman and a list of cities, and poses the question of finding the shortest possible route (calculated through the shortest possible distance) for them to visit each city on the list exactly once and return to their city of origin (Cook).

There are two broad categories of the problem: symmetric and asymmetric. A symmetric TSP dictates that the distance between cities x and y is the same, forming a direct graph, and allowing the time taken for a problem to be solved to reduce by half (Rodriguez and Ruiz, 2). On the other hand, the distance between cities x and y may not be the same in an asymmetric TSP.

The most common solution of the problem involves using a Brute-force algorithm to generate all possible tours that can be taken, calculating the distances of each tour, and choosing the tour with the shortest total distance. Since this would require checking each and every possible solution to find the shortest distance, the relation between size and permutations is factorial. This can also be expressed as *O(n!)*, in the form of Big-O notation, which describes the worst-case scenario of the algorithm.

No. of cities	Possible tours
5	120
7	5,040
10	3,628,000
15	1,307,674,368,000
20	2,432,902,008,176,640,000

TABLE 1: Relation between the number of cities and possible tours

Even with a computer that could check 1 million possible orderings per second, calculating the shortest tour for 20 cities would take 77,000 years. As size increases, the brute-force algorithm proves to be extremely inefficient at solving this problem.

Currently, there is no efficient algorithm that can be used to find an exact solution for a large number of cities (Rego, 427). In the field of computational complexity, the problem falls in the NP, or Non-Deterministic Polynomial time class, meaning the fastest found process to derive an exact solution can only be solved as a polynomial time function, either being exponential, or factorial. However, various heuristics have been suggested in order to find an approximate solution. Sets containing millions of cities can be within a fraction of the time of a brute-force algorithm, with a percentage error of just 2-3% (Rego, 211).

3 Existing Heuristics

A heuristic is an algorithmic technique used to find an almost approximate solution to a problem using efficient methods, especially when traditional methods do not yield effective results. The value of a heuristic lies in the time taken to solve a problem, especially when there is no known solution to the algorithm (Apter, 83).

3.1 Nearest Neighbour

One of the simplest heuristics of the Travelling Salesman Problem is the nearest neighbour algorithm. It functions as a greedy algorithm, taking the lowest local optimum step in the hopes of finding the lowest global optimum solution. The steps for the algorithm are as follows:

1. Begin at a city C
2. Find all connecting edges to C and select the one with the shortest distance
3. Mark the new city as C
4. Mark the previous city as visited
5. If all cities have been searched, terminate the program
6. Repeat from step 2

While easy to implement and being very fast, the Nearest Neighbour heuristic suffers from its "greedy" nature, as the solution resulting in the lowest local optimum may not always be the same as a global optimum.

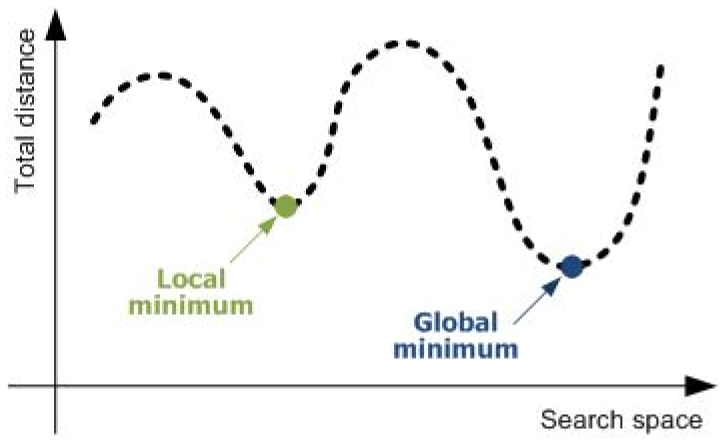

FIGURE 1: Difference between local and global optimum in a search space

As shown by figure 1, if found at the local minimum, the Nearest Neighbour algorithm would not do anything, since any "nearest" choice found both on its left and right yield a higher result. Therefore, the program would stop, even though the exact solution (global minimum) hasn't been found.

The nearest neighbour heuristic also only performs well with less than 50 cities. As the number of cities supersedes that limit, the quality of the performance and solution worsens (AlSalibi et al). Since real-life applications of the problem require a number of nodes much larger than 50, it is not an optimal heuristic for all cases of the Travelling Salesman Problem.

The Nearest Neighbour represents an important idea with the Travelling Salesman Problem and other optimization problems: the balance between exploitation versus exploration. Exploitation refers to the idea of choosing the best possible value from a limited search space with the hopes of reaching a global optimum. However, only a local optimum is often reached. Exploration, on the other hand, consists of diversifying the search by picking choices that may result in a better solution. This prevents the problem of being trapped in a local optimum and allows for more choices to be explored, possibly resulting in the global optimum. One such example of exploration is a genetic algorithm.

4 Genetic Algorithm

A genetic, or evolutionary algorithm, is a search heuristic that mimics the process of natural evolution, including elements of hereditary, variation, and selection, with the intent of reaching an optimal solution through repetition across a number of generations. Developed by John Holland in 1975, the main reason for their popularity is robustness, or as explained by researcher David E. Goldberg, their "balance between efficiency and efficacy" that allows them to be universal and thrive in different environments and problems. (Goldberg, 89). This demonstrates exploration since various solutions of differing quality must be taken into account for the sake of finding a value as close to the global minimum as possible.

Genetic algorithms function by creating a population of certain solutions, or chromosomes, selecting the best-fit solutions, combining and mutating them slightly for the purpose of diversification, and repeating the process over a number of generations. In the context of the Travelling Salesman Problem, a chromosome is a string of genes, or cities, that forms a solution. A population refers to a collection of all chromosomes that are involved in the search (Fischer). These chromosomes are then selected and modified through the use of operators.

4.1 Operators

The main process of selecting the fittest solutions and modified their contents for the purpose of reaching an optimal solution is done through the use of operators. Generally, three main operators known as selection, mutation, and crossover are used to execute a genetic algorithm.

4.1.1 Selection

The selection operator is necessary for finding and picking the best solutions from a population for the purpose of crossover and mutation. Taking inspiration from Darwin's theory of evolution and natural selection, the majority of genetic algorithms choose chromosomes on the basis of their fitness. The fitness for a chromosome can be evaluated as the inverse of its found distance. Thereby, the higher the fitness value, the more likely the selection operator will choose that chromosome. This form of selection is known as *elitist selection* (AlSalibi et al, 35).

4.1.2 Mutation

Mutation introducing a small change to the resulting chromosome from the crossover with a certain probability. For instance, a common mutation is *flip mutation* (Soni and Kumar, 4520), which involves selecting two random points of a solution and exchanging values. For instance, if flip mutation were to be caused on C_1 with the random points highlighted, the resulting value would be C_m.

C_1 = 1 2 **3** 4 5 **6** 7

C_m = 1 2 6 4 5 3 7

While seemingly small, mutation is vital for the success of a genetic algorithm for the purpose of diversification and preventing stagnation of certain solutions. However, the rate of mutation is kept small (0.01 - 0.05), in order to prevent it from overpowering and worsening good solutions created through crossover.

5 Crossover

Perhaps the most important element of a genetic algorithm, crossover regulates the recombination of the various chromosomes in a population in order to ensure a diverse pool of solutions tending towards a certain value.

In order to examine the characteristics of each crossover operator, the following set of cities will be used in the form of a cost matrix (Ahmed, 99). The distance from a city at one position to a city at another can be found by linking the horizontal versus the vertical node. For instance, the distance between nodes 1 and 7 (c_{17}) will be 8.

Node	1	2	3	4	5	6	7
1	999	75	99	9	35	63	8
2	51	999	86	46	88	29	20
3	100	5	999	16	28	35	28
4	20	45	11	999	59	53	49
5	86	63	33	65	999	76	72
6	36	53	89	31	21	999	52
7	58	31	43	67	52	60	999

TABLE 2: Cost matrix of an example set of cities

Let the pair of parent chromosomes to be used for recombination will be P_1: (1, 4, 7, 5, 2, 6, 3) with distance 391 and P_2: (1, 2, 4, 7, 6, 3, 5) with distance 433.

5.1 Two Point Crossover

Perhaps the simplest crossover operator, Two Point Crossover (TPX) begins by randomly creating two partition points at the same distance from each parent chromosome. The data from each chromosome is then swapped, creating two new child solutions. TPX is implemented using the examples of P_1 and P_2:

P_1: 1 4 | 7 5 2 | 6 3

P_2: 1 2 | 4 7 6 | 3 5

C_1: 1 4 4 7 6 6 3

C_2: 1 2 7 5 2 3 5

When TPX is tried on P_1 and P_2, an error arises that conflicts with the requirements of the Travelling Salesman Problem: each city be visited, and each city be visited only once. In C_1, the cities 4 and 6 are repeated, while in C_2, the cities 2 and 5 are repeated. To stop this problem for occurring, the first occurrence of a repeating city in the first child solution is highlighted, and replaced with the first occurrence in the next solution. For instance:

C_1: 1 $\underline{4}$ 4 7 $\underline{6}$ 6 3

C_2: 1 $\underline{2}$ 7 $\underline{5}$ 2 3 5

C_1: 1 2 4 7 5 6 3

C_2: 1 4 7 6 2 3 5

The resulting values of C_1 and C_2 are 487 and 406 respectively. This example shows the ineffectiveness of TPX to deliver a better solution than P_1 and P_2. Its strength lies in its simplicity, as well as its ease of computation. It has the added benefit of maintaining the fit qualities of a parent chromosome since the first two cities of each path are the same in C_1 and C_2 as they are in P_1 and P_2. However, a lack of diversity and similarity to the parent chromosomes means that stagnation may occur, since not enough of the pattern is being developed to create newer and better solutions (Imtiaz, 20).

5.2 Edge Recombination Crossover

The Edge Recombination Crossover (ERX) operator, on the other hand, employs a different strategy to the crossover of two parent solutions. Firstly, only a single chromosome is created, as opposed to two children from TPX. A much greater mathematical approach is employed with this crossover, through the use of constructing an edge map.

An edge map lists out the nearest node on the left and right of each node showcased in both parent solutions while ignoring duplicates.

Node	Edge List
1	4, 3, 2, 5
2	5, 6, 1, 4
3	6, 1, 5
4	1, 7, 2
5	2, 7, 3, 1
6	2, 3, 7
7	4, 5, 6

TABLE 3: Edge map that can be constructed using parent nodes P_1 and P_2.

From here on, the chromosome is constructed by taking the edges of the nodes into consideration. The next possible city for a node to choose can be either dictated by the number of edges of the next node, with the one with a lower number of edges being selected, or chosen randomly (Whitley, Starkweather & Shaner). ERX will now be used with P_1 and P_2 in order to generate an expected offspring.

1. Node 1 is the starting point of the tour. The partially mapped chromosome reads {1}.
2. Node 1 has edges 4, 3, 2, 5. Edges 5 and 2 are removed due to them having 4 edges each. Node 3 is chosen at random. The partially mapped chromosome reads {1, 3}.
3. Node 3 has edges 6, 5. Node 6 is chosen since it has less edges than Node 5. The partially mapped chromosome reads {1, 3, 6}.
4. Node 6 has edges 2, 7. Node 7 is chosen since it has less edges than Node 2. The partially mapped chromosome reads {1, 3, 6, 7}.
5. Node 7 has edges 4, 5. Node 4 is chosen since it has less edges than Node 5. The partially mapped chromosome reads {1, 3, 6, 7, 4}.
6. Node 4 has only one remaining edge that is unvisited, which is Node 2. The partially mapped chromosome reads {1, 3, 6, 7, 4, 2}.
7. Node 2 has only one remaining edge that is unvisited, which is Node 5. The partially mapped chromosome is now the full chromosome C_1 which is 1, 3, 6, 7, 4, 2, 5.

The resulting cost of C_1 is 486, which is worse than both parent nodes that it started from. It must also be considered that it is possible value, and often depends on the random choice that is taken at times during the crossover.

5.3 Sequential Constructive

Developed in 2013 by Zakir Ahmed, Sequential Constructive Crossover (SCX) is the newest operator out of the group being tested. His work serves the basis of using the better edges from each parent node, as well as constructing edges that are not present in both parents' structure (Ahmed, 99). It also defines the term 'legitimate node' as the node in each parent that has not been visited yet. By comparing the possible choices of the next node to be taken, SCX also has a quantitative advantage over the previously mentioned operators. Using P_1 and P_2, SCX is conducted using the following steps. The distance from city x to y is transcribed as c_{xy}.

1. Start from Node 1. The 'legitimate' nodes from this point are 4 and 2. Since c_{14} (9) < c_{12} (75), 4 is selected as the next node. The partially mapped chromosome reads {1, 4}.
2. The 'legitimate' node from Node 4 is 7 in both parents, so 7 is selected as the next node. The partially mapped chromosome reads {1, 4, 7}.
3. The 'legitimate' nodes from Node 7 are 5 in P_1 and 6 in P_2. Since c_{75} (52) < c_{76} (60), 5 is selected as the next node. The partially mapped chromosome reads {1, 4, 7, 5}.
4. The 'legitimate' nodes from Node 5 are 2 in P_1 and none in P_2 (1 does not count since that would mean ending the cycle without visiting all nodes). Therefore, the first 'legitimate' node is considered from the set {1, 2, 3, 4, 5, 6, 7}, which is an ascending continuation of all nodes from the last possible point, which was 1. Therefore, the 'legitimate' node for Node 5 is 2 for P_2. Since the values match, 2 is selected as the next node. The partially mapped chromosome reads {1, 4, 7, 5, 2}.

5. The 'legitimate' node from Node 2 is 6 in both parents (4 and 7 have already been visited for P_2, so 6 is selected as the next node. The partially mapped chromosome reads {1, 4, 7, 5, 2, 6}.
6. The 'legitimate' node from Node 6 is 3 in both parents, so 3 is selected at the next node. The partially mapped chromosome, now the completed chromosome C_1, reads {1, 4, 7, 5, 2, 6, 3}

C_1 has resulted in the same value as P_1, which has a value 391. This is expected with a crossover operator, especially when there is a lack of diversity between the two parent solutions, which is usually different when conducting an experiment. However, it was able to determine P_1 as having qualities that were beneficial and necessary to carry onto the next generation due to quantitative comparisons, showing its effectiveness.

These specific crossover operators have been chosen due to the stark differences existing between all of them in terms of their complexity, process of recombination and possible results as shown with the sample Parent genes P_1 and P_2.

6 Investigation

An experiment will be conducted to further investigate the research question. This involves using a program in Matlab-2009 ra (Kirk) to simulate the problem with three separate crossover operators. The program will be run on a Mid-2012 Macbook Pro with a 2.9Ghz quad-core Intel Core i7 processor with 8GB of Ram and Intel HD Graphics 4000. The program will use four known test-cases for the Travelling Salesman Problem: ST 70 (70 cities), FTV 100 (100 cities) , FTV 170 (170 cities) and TSP 225 (225 cities). ST 70 and TSP 225 are symmetric, while FTV100 and FST170 are asymmetric. The best solution for all cases are 675, 3919, 1788, 2755 respectively.

These test cases come from a series available online by author and researcher Gerhard Reinelt (Reinelt). ST70 was chosen as a benchmark, or a way to spot the limits of each crossover operator with a relatively easy problem. TSP 225 was chosen to see if the quality of a solution would diverge drastically over such a difference in size. FTV 100 and FTV 170 were both examples of an asymmetric route, and were used to see if the difference in distance between two cities would affect the quality of a solution as well as the time taken to find the solution. For instance, SCX dictates that the next choice of a 'legitimate' node be made based on a cost analysis, where the order of the cities can affect the total distance more significantly.

Crossover and mutation operators have been kept at 0.8 and 0.01 respectively. The mutation rate is kept at a lower number to ensure the possibility of it occurring, but not cause it to interfere with the actions of the crossover operator. The maximum number of generations will be kept at 50000 to allow for an ample number of turns before a "best-found" solution for each crossover operator is found. The program will deliver an output of the reported "best distance" calculated in each instance, as well as the time taken to solve the problem. Each crossover operator will be tested five times for each case, and the percentage error of the problem will be calculated using the following equation.

$$Percentage\ Error\ = \frac{found\ solution - actual\ solution}{actual\ solution} \times 100$$

Through this, the mean, as well as the lowest percentage error (best) of all five tests will be calculated and reported, along with its standard deviation.

It is hypothesised that the TPX should be able to both find the optimal solution the best (and have the least percentage error), as well as being the fastest in finding the optimal solution due to its simplicity and sheer speed of recombination. ERX will be the slowest due to the steps required to create a combination being lengthy and increasing the time taken to solve a certain tour. SCX will be placed in the middle of both operators, as it produces a recombination that takes into account the distance between the nodes, but requires a lengthy process to do so.

6.1 Results

<u>Example of error calculation:</u>

An example of the equation used to calculate the best found solution of five trials for the FTV100 of ERX is shown.

$found\ solution\ = 5299.27$

$actual\ solution\ = 1788$

$Percentage\ Error\ = \frac{5299.27 - 1788}{1788} \times 100$

$= 196.38\%$

Crossover operator	ST 70		FTV 100		FTV 170		TSP 225	
	Mean (Std. Dev)	Best	Mean (Std. Dev)	Best	Mean (Std. Dev)	Best	Mean (Std. Dev)	Best
TPX	12.59 (1.25)	10.37	62.73 (3.32)	56.14	81.17 (10.94)	78.12	100.06 (14.95)	97.13
ERX	65. 32 (6.93)	61.29	202.14 (12.93)	196.38	257.17 (20.02)	248.89	309.76 (22.14)	306.31
SCX	**2.14 (0.29)**	**1.68**	**13.21 (1.97)**	**10.98**	**35.01 (10.34)**	**28.27**	**44.92 (6.68)**	**43.88**

TABLE 4: Percentage error of best-found solution for each crossover operator in 4 test cases.

Crossover operator	ST 70	FTV 100	FTV 170	TSP 225
TPX	**327.84**	**404.92**	**652.87**	**1179.17**
ERX	7081.23	9397.46	12024.65	15812.44
SCX	1392.88	1905.96	2682.68	4097.29

TABLE 5: Time taken to find solution (seconds)

6.2 Analysis

The experiment conducted shows that none of the crossover operators were able to find the exact solution to each case taken. However, in terms of delivering the optimal solution with the least percentage of error, SCX was the best, followed by TPX, and ERX. In terms of the total time taken to find the optimal solution, TPX was the best, with SCX and ERX following. Based on this, the hypothesis mentioned before was only half correct, since SCX was able to produce a better solution compared to its competitors, but TPX was able to deliver the fastest solution to the problem.

The likely reason for this result is due to the process used by SCX to combine two parent solutions to make a child solution. The main purpose of a crossover operator is to generate a solution that maintains the benefitting characteristics of both parents' structure, while still changing to form to create as high a fitness value as possible for the following generation. SCX is able to accomplish all of this by taking into account the cost of the next possible city to be taken, and choosing the city that results in an overall lower tour length. Despite its effectiveness, the comparison needed between comparing the adjacent nodes scales up quickly as the number of cities increases, resulting in a program that was not the fastest out of the three tested.

TPX on the other hand, is extremely easy to implement and use. Requiring only two points of crossover to create two new solutions is a much faster process compared to constructing the edges of each crossover and comparing similar results. However, it's main fault lies in its ineffectiveness, since the operator is often not able to create solutions that are diverse enough. The software used during the experiment also showed TPX as being the fastest to reach a "best-found" solution, but then staying at that general area and "plateauing". Due to not having diverse enough solutions, or a practical reasoning for the steps required to crossover, TPX resulted in only the second-best solution.

The likely reason for the poor performance of ERX is the steps involved in actually executing the algorithm, and the amount of processing power required to execute it efficiently. ERX requires the construction of an edge map of each of the nodes, which while works fine through the use of the 7-city cost matrix used above, is much more processor intensive as the number of cities increases. Furthermore, even with the comparison of edges, the decision made by the operator was often just a random choice. An improvement on the crossover would be to use the method of comparison from SCX and implement that technique when deciding the next edge to be taken for ERX. This would allow for more quantitative comparisons that would help the performance of the program.

Interestingly, asymmetry had no noticeable effect in the time taken to find an optimal solution, which followed a trend of increasing with the number of nodes. However, the quality of the found solution did suffer from both asymmetry as well as the number of cities. The best percentage error on for TPX on ST 70 was 10.37, and on FTV 170 was 78.12. For a 100 city increase, the percentage error increased by nearly 68 points. Such a harsh increase shows the worsening of the found solution of genetic algorithms with larger sets, especially when the right crossover operator is not used.

6.3 Limitations

Limitations had been apparent after the investigation was completed. One of these was the hardware used to conduct the experiment. The Matlab program was run on a mid-2012 Macbook Pro having a 4th generation Intel Processor with integrated Intel HD 4000 graphics. While fine for normal tasks, the graphics processing unit (GPU) of the computer used was never intended to be used for large scale combinatorial problems with population sizes of 50,000. This increased number of calculations was extremely intensive on the processor and significantly slowed down the process, requiring a time of almost five hours to find an optimal solution for the largest test case (TSP 225) for the ERX operator. Likewise, both ERX and SCX require the calculation of edge maps and comparisons between next possible solutions, and as such, were limited by the low power of the processor. While the resulting calculations serve the purpose of the investigation for the sake of comparison, using a computer with a dedicated GPU would have improved upon the speed of the process, and allowed for experimentation with more test cases.

One such example is NVIDIA's Kepler series of high performance computing architecture, which are designed with the purpose of being able to conduct experiments with large quantities of calculations, such as those in the Travelling Salesman Problem. ("KEPLER THE WORLD'S"). These processors have the benefit of taking advantage of their multiple cores and minimizing on idle time in order to advance efficiency.

7 Conclusion

Concluding the question, *'How is an optimal found solution for the Travelling Salesman Problem using a Genetic Algorithm affected by the use of Two Point, Edge Recombination, and Sequential Constructive Crossover Operators?'* again, shows SCX as being the best crossover operator compared to TPX and ERX in terms of finding an optimal solution to the problem. However, when considering the time taken to find an optimal solution, TPX was found to be the fastest operator, with SCX and ERX following. Deciding on an operator depends upon the number of nodes, amount of time available, and the desired quality of a solution.

The results of this experiment are relevant in the field of the Travelling Salesman Problem, but in the whole general area of combinatorial optimization. For instance, a problem had arisen in the Dutch Town of New Holland where telephone boxes needed to be added by the postal service. With a set route and a maximum working time of 445 min, the problems qualities are extremely similar to that of the Travelling Salesman Problem. Other more diverse applications also exist, such as computer wiring and gene modification, since both those require the use of a path to be taken and finding that shortest path in order to maximize the efficiency of the algorithm being used. (Lenstra)

It is also important to consider that this is in no way an exhaustive solution to the correct way of solving this problem. The number of crossover operators is a list that continues to increase frequently with inspiration from the natural world (k-Point Crossover) or innovation through the combination of the principle of various operators to form one (Modified Sequential Constructive Crossover). This essay serves the purpose of determining the quality of just three crossover operators based on their algorithmic and visual complexities, and viewing how that translates into performance.

Genetic Algorithms have a future in today's society as precursors to Machine Learning and Artificial Intelligence. David. E. Goldberg, an expert in the field who has been studying genetic algorithms for over three decades, has mentioned that these algorithms are the best step for computing in the future, as they are being able to describe innovation in a quantitative rather than qualitative manner. They are able to owe their success to using a principle found in the natural world in an application where the answer might not necessarily be known, helping pave the way for innovative solutions for problems that continue to remain unsolved in the realm of Computer Science.

Works Cited

Ahmed, Zakir H. "Genetic Algorithm for the Traveling Salesman Problem using Sequential Constructive Crossover Operator." *International Journal of Biometrics and Bioinformatics*, vol. 3, no. 6, pp. 96-104. *Al-Imam Muhammad Ibn Saud Islamic University*. Accessed 5 May. 2018.

AlSalibi, Besan A., et al. "A Comparative Study between the Nearest Neighbor and Genetic Algorithms: A revisit to the Traveling Salesman Problem." *International Journal of Computer Science and Electronics Engineering*, vol. 1, no. 1, 2013. *Semantic Scholar*. Accessed 25 May. 2018.

---. "A Comparative Study between the Nearest Neighbor and Genetic Algorithms: A Revisit to the Traveling Salesman Problem." *International Journal of Computer Science and Electronics Engineering*, vol. 1, no. 1, 2013, pp. 34-38. *Universiti Sains Malaysia*. Accessed 5 May. 2018.

Apter, Michael J. (1970). The Computer Simulation of Behaviour. London: Hutchinson & Co. p. 83

Cook, William J. *In Pursuit of the Traveling Salesman: Mathematics at the Limits of Computation*. Princeton, NJ, Princeton UP, 2012.

Fischer, Dominique. “Evolution with Teleology: The Genetic Programming Heuristic Approach to Modeling.” *Journal of Real Estate Literature*, vol. 16, no. 3, 2008, pp. 347–362. *JSTOR*, JSTOR, www.jstor.org/stable/44103651.

Goldberg, David E. *Genetic Algorithms in Search, Optimization & Machine Learning*. 4th Impression ed., Pearson Education, 1989.

Imtiaz Hussain Khan,"Assessing Different Crossover Operators for Travelling Salesman Problem", International Journal of Intelligent Systems and Applications (IJISA), vol.7, no.11, pp.19-25, 2015. DOI: 10.5815/ijisa.2015.11.03

"KEPLER - THE WORLD'S FASTEST, MOST EFFICIENT HPC ARCHITECTURE." *Nvidia.in*, www.nvidia.in/object/nvidia-kepler-in.html. Accessed 10 Oct. 2018.

Kirk, Joseph (2014). Traveling Salesman Problem - Genetic Algorithm (https://in.mathworks.com/matlabcentral/fileexchange/13680-traveling-salesman-problem-genetic-algorithm), MATLAB Central File Exchange, Accessed 29 June 2018

Lenstra, J. K., and A. H. G. Rinnooy Kan. “Some Simple Applications of the Travelling Salesman Problem.” *Operational Research Quarterly (1970-1977)*, vol. 26, no. 4, 1975, pp. 717–733. *JSTOR*, JSTOR, www.jstor.org/stable/3008306.

Ma, Suzanne. "The Traveling Salesman Problem and Other Classical Algorithmic Tales…." *Routific*, Medium, 25 Oct. 2016, blog.routific.com/travelling-salesman-problem-vehicle-routing-problem-e5834f652a76. Accessed 14 Jan. 2018.

Medium.com. 25 Oct. 2016, blog.routific.com/travelling-salesman-problem-vehicle-routing-problem-e5834f652a76. Accessed 29 June 2018.

Mühlenbein, Heinz. "Genetic Algorithms." *Local Search in Combinatorial Optimization*, edited by Emile Aarts and Jan Karel Lenstra, Princeton UP, 1997, pp. 137-72.

Reinelt, Gerhard. “MP-TESTDATA - The TSPLIB Symmetric Traveling Salesman Problem Instances.” *Zib.de*, 1 June 1995, elib.zib.de/pub/mp-testdata/tsp.

Reinelt, Gerhard. “MP-TESTDATA - The TSPLIB Asymmetric Traveling Salesman Problem Instances.” *Zib.de*, 1 June 1995, elib.zib.de/pub/mp-testdata/atsp.

Rego, César, et al. "Traveling salesman problem heuristics: Leading methods, implementations and latest advances." *European Journal of Optional Research*, 21 Sept. 2011, pp. 427-41. *ScienceDirect*. Accessed 3 July 2018.

Rodriguez, Alejandro, and Rubén Ruiz. "The effect of asymmetry on travelling salesman problems." *Grupo de Sistemas de Optimización*. *Universidad Politécnica de Valencia*. Accessed 3 July 2018.

Smith, George. *In regards to genetic algorithms*. *Stackoverflow.com*, 26 Feb. 2012. Accessed 13 Sept. 2018.

Soni, Nitasha, and Tapas Kumar. "Study of Various Mutation Operators in Genetic Algorithms." *International Journal of Computer Science and Information Technologies*, vol. 5, no. 3, 2014. *IJCSIT*. Accessed 25 May 2018.

Whitley, Darrell & Starkweather, Timothy & Shaner, Daniel. (2002). The Traveling Salesman and Sequence Scheduling: Quality Solutions Using Genetic Edge Recombination.