**TEAM #3 PROJECT REPORT**

**TRAVELING SALESMAN AND THE GENETIC ALGORITHM**

**Submitted To**

**The RET Site**

**For**

**“Challenge-Based Learning and Engineering Design Process Enhanced Research Experiences for Secondary School and 2 and 4 Year Teachers”**

**Sponsored By**

**The National Science Foundation**

**Grant ID No.: EEC-1710826**

**College of Engineering and Applied Science**

**University of Cincinnati, Cincinnati, Ohio**

**Prepared By**

**Matthew Bagazinski, Physics, Detroit Crysto-Rey High School, Detroit, Michigan**

 **Peter Szyjka, Mathematics, Cincinnati State University, Cincinnati, Ohio**

**Approved By**

 **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Dr. Manish Kumar Dr. Jeffery Kastner**

**College of Engineering and Applied Science**

**University of Cincinnati**

**Reporting Period: June 10 – July 26, 2019**

**ABSTRACT**

The Traveling Salesman Problem (TSP) has become a useful medium to improve the efficiency and cost-effectiveness of commercial delivery applications. Given a set of multiple locations, it poses the problem of determining which path is more efficient, shorter or least expensive. More recently, with the increasing variety of delivery methods and increasing number of destinations, computation of routing selections number on the order of the factorial number of destinations. Computational restrictions therefore accompany solutions as the number of delivery destinations increase. This work was an effort to improve speed and effectiveness of TSPs using genetically inspired computational methods by mutating route populations and selecting the fittest ones for repopulation. Our research, as has others, shows that as the number of iterations of the GA increase, an asymptote of best routes, or shorter distance is reached. We edited a benchmarked GA to improve the speed and efficiency of those algorithms. Our alternative GA reached an asymptote of convergence with a steeper slope and less time than the benchmarked GA. Experimental techniques resulted in an improved genetic algorithm.

**KEY WORDS** Travelling Salesman Problem, Genetic Algorithm, random

1. **INTRODUCTION**

Travelling salesmen want to visit all destinations only once and return to the starting point while covering the least distance. In minimizing distances travelled by human, manned- or automated-delivery systems, commercial organizations may exploit TSP solutions to minimize cost and maximize efficiency. Research into various methods to most effectively determine the best possible TSP routes are frequently found to be ‘good enough’ rather than the shortest or fastest.

One of the most effective TSP algorithms are known as genetic algorithms (GA) and have shown considerable promise as efficient, cost-effective solutions. A variety of methods used in GAs, such as Nearest Neighbor and 2-Opt were explored. They use random numbers to initialize a set of routes, finding the ‘most fit’ of the candidate solutions and mutating those, giving rise to the genetically-inspired termed algorithm.

Optimizing solutions to the Travelling Salesman Problem (TSP) has applications in commercial delivery systems and other scenarios, such as a trucking company’s multiple destinations or recent unmanned aerial vehicle delivery proposals accompanied by the problem of integration of those systems into the national airspace.

There are several approaches to the TSP with names synonymous with genetic methods, such as Nearest Neighbor and 2-Opt. Our goal was to combine them with the TSP and improve TSP solutions using more efficient code. Genetic algorithms (GA) are one of the most promising TSP solution methods that employ random selections of a population of candidate routings, finding the best results, and subsequently reusing those results in an iterative process. This process continues until a ‘good-enough’ solution is found. The primary limitation with fundamental TSP problems lies in the increasing number of computations required as the number of destinations increase; a method known as “brute force.” This limitation has led to a variety of ‘good-enough’ genetic and evolutionary algorithms.

1. **LITERATURE REVIEW**
2. Of the many instructive texts on genetic algorithms and evolutionary modeling, authors1 (Krzanowski and Raper, 2001), and Dianati2 et al, helped this team summarize the simple steps germane to these GA techniques. Dianati, however outlined a more detailed series of algortihm instructions (Danati, et al, 2012)2. The instructions these authors offer also follow a similarly structured theme displayed in the flow chart in Figure 1, TSP Flow and First Iteration. Practical applications of this technology are expected to apply over multiple layers of commercial industry to include unmanned systems (UAV). These unmanned systems are predicted to evolve into the U.S. national airspace as well as become a delivery medium. Anoop3 et.al., discusses the TSP and genetic algorithms in a comparison of different approaches to solve the TSP and its application towards swarming of UAVs,[(Anoop](https://search.proquest.com/indexinglinkhandler/sng/au/Sathyan%2C%2BAnoop/%24N?accountid=2909), S; [Boone, N](https://search.proquest.com/indexinglinkhandler/sng/au/Boone%2C%2BNathan/%24N?accountid=2909)., [Cohen,](https://search.proquest.com/indexinglinkhandler/sng/au/Cohen%2C%2BKelly/%24N?accountid=2909) K., 2015) an issue expected to become problematic in airspace operations. Shiffman4, provided an additional, helpful chapter on genetic algorithms in his The Nature of Code. This team began it’s initial study of the travelling salesman problem with The Traveling Salesman Problem5, ( Applegate, D. et al., 2007).
3. **GOALS AND OBJECTIVES**

The RET participants wanted to improve the efficiency of GA TSP code written by previous authors. Reducing the computation time of code is considered to be an improvement in the efficiency of an algorithm. As the TSP is tasked with more paths, N, to evaluate, or more cities for the salesman to visit, the computation complexity increases on the order of N factorial. This value is (N-1)! /2 . Extrapolating the TSP idea, clustering of cities provides an opportunity for multiple travelling salesman and an opportunity to evaluate that scenario. Exploring, exploiting and editing various computational methods is a way to discover potential improvements. There are exact and heuristic algorithms. The heuristic algorithms provide the ‘good-enough’ shorter computation time and results. Our efforts centered on editing and potentially improving these methods.

1. **RESEARCH TASKS**

The primary objective of this research was to improve the efficiency of the current benchmark TSP GA algorithm. Efficiency here means to more rapidly, i.e., a steeper slope, converge to a stabilized distance as the number of population generations increase or, to converge to a stabilized distance in less computational time . The team’s tasks and training were similar and included such items as familiarization with simple TSP stand-alone code modules that perform the random number generated, core operations within the overall TSP code. These modules were genetic algorithm type operations. Team tasks concluded with analysis of the results and a contrast of the alternative GA with the benchmarked GA.

1. **RESEARCH STUDY DETAILS AND METHODOLOGY**

The first set of tasks began with understanding the TSP and its various genetic solution methods.

* 1. Genetic Algorithm Fundamentals. Genetics is based on mutations and selection of the fittest in

a population. As the name implies, and as the code iterates, shorter route sequences are chosen as the best, or fittest and mutated. This mutation is pseudo-random in a predetermined way. For example, an algorithm-chosen shorter route sequence of cities may be changed, i.e., mutated, to visit one before another prior to a subsequent iteration. A visual display of the process and an interim result is shown in figure 1, TSP Flow and First Iteration. It depicts the computational flow and display of distances for a simple random selection of eight cities. They are to be portioned into 2 groups of 4 possible routes. The next steps begin with selecting the shortest routes as indicated in the figure, followed by their mutation and selection as the next population.

The top-ranked, i.e., shortest, routes are then reinserted into the ‘population’ of routes as the fittest of the previous population to evaluate their fitness as a shorter route. These methods are termed “flip,” “swap,” and “slide.” That process is shown in figure 2, Flip, Swap and Shift, where the two “No Change” headers indicate the portioning of two top-ranked, shortest route groups out of the eight, selected for next iteration. These are the genetic and evolutionary processes from which the terminology originates.

* 1. Genetic Algorithm Benchmark (MATLAB®, Kirk, J). The TSP GA used as a benchmark to contrast the team’s alternative GA finds a near optimal solution to the TSP. A random selection of x-y city coordinates are generated by the benchmark as input. The GA then searches for the shortest route, i.e., a route with the least distance for the salesman to travel to each city exactly once and return to the starting city. Output is the best (near optimal) route found by the algorithm along with the minimum distance along that path. Our team’s tasks involved editing the benchmark code’s modules to reduce either computation time or some other measure of efficiency.
	2. Genetic Algorithm Specifics. As noted, there exists named methods to conduct genetic

mutations such as the 2-Opt. This procedure is one that randomly selects two cities on a proposed route, swaps paths with another set of cities and compares the distances. If those distances result in longer routes, they are reversed. If shorter, they are used as next-generation route candidates, i.e., a member of a new population. The 2-Opt scheme is the “swap” mentioned above.

Upon the user expanding the number of cities to be evaluated, the benchmark code apportions the selection into groups of four routes, as noted above. For example, if N = 60 cities then 15 groups of four are iterated. The “fittest” route within each group is the shortest route and its sequence of travel is chosen to be mutated, i.e., flipped, swapped or shifted. That iterative process continues until an asymptote of convergence is reached. In these test cases of the algorithm, the shortest distance is known a-priori. That asymptote considers the difference in calculated distances between the current iteration and the previous.

Since the code (as does genetics) requires randomness in its iterations, questions regarding quality of the efficiency of improvements (e.g., reduced computation time) inevitably lead to discussions the pseudo-randomness of MATLAB’s random number generator when compared results are similar. Some initial results comparing benchmark versus alternative code indicated distance differences within 1-standard deviation, resulting in questionable improvements from either the alternative, or as a result of fineness of the random generation process. The decision was made to seed the random number generator prior to each iteration to insure the most random effects possible.

 2 Group Route Sequence Total Distance Rank

**8 6 4 5 1 7 3 2 37.3 2**

**2 3 1 6 5 4 7 8 54.6 3**

**5 2 1 7 6 8 3 4 67.8 4**

**7 6 3 8 5 4 1 2 33.4 1**

**4 6 2 7 3 8 5 1 44.7 2**

**4 5 8 6 3 7 1 2 74.1 4**

**5 2 6 4 8 7 1 3 44.3 1**

**6 1 4 3 8 7 2 5 54.6 3**



.

 Figure 1. TSP Flow and First Iteration

 2 Group Route Sequence Mutation Type

**7 6 3 8 5 4 1 2 No Change**

**7 5 8 3 6 4 1 2 Flip**

**7 5 3 8 6 4 1 2 Swap**

**7 3 8 5 6 4 1 2 Shift**

**5 2 6 4 8 7 1 3 No Change**

**5 2 1 7 8 4 6 3 Flip**

**5 2 1 4 8 7 6 3 Swap**

**5 2 4 8 7 1 6 3 Shift**

 Figure 2, Flip, Swap and Shift

1. **RESEARCH RESULTS**

Initial visual comparison between the benchmark GA and the alternative shows marked differences. In the figures to follow, best-route distances are normalized by the known maximum distance both GAs consider. They converge over iterations of population generations for both the benchmarked GA and alternative GA. Both GAs use populations of 20, 40 and 60 destinations. The steeper slope of the alternative GA indicates a more rapid convergence to the route with the shorter, i.e., best route.

In figure 3, Alternative GA and figure 4, Benchmark GA, the most visible difference is the starting slope of the alternative GA. This suggests a faster, more timely convergence of the alternative GA. The horizontal axis, Number of Generations is the number of times a group of four routes is mutated and sent for its next evaluation of fitness, in the benchmark code. The 20, 40 and 60 in the figures are numbers of populations of cities tasked in the algorithm.

For clarity, using the axis format shown in previous figures, Figure 5, Alternative vs. Benchmark Distance Convergence displays only 60 destinations. The alternative GA shows a steeper, more rapid convergence. It further displays a noticeable distance difference between asymptotic convergences of the two methods as the number of generations increase; the constant near-zero slope indicating a most likely true minimum distance.

To improve efficiency, the team wrote an alternative GA code to bypass a four-group segmentation of the population and instead, selected the top three fittest routes from a population, of say, 60 routes, and mutated those with random generated pair-of-dice rolls. The most frequent combination of a pair of dice is seven and this result was selected to generate a flip-type genetic mutation operation. A flip was initially chosen as a suspected most-promising result of a mutation; and that selection was borne out as the quickest asymptotic convergence. The next most probable iteration selected was chosen to be a 3-Opt versus a 2-Opt. This selection is believed to have caused a crossover at approximately 2000 generations, as the 3-Opt is more computationally intense. The benchmark GA then converged closer to the true shortest path. Following 2000 generations, no further improvement toward convergence nor difference is detected. Team analysis and discussion concluded that the small differences in results is offset by the more rapid convergence of the alternative algorithm requiring less computational time. The small difference between an optimal and the true shortest distance is likely to be considered negligible for time-sensitive commercial applications.

An analysis of efficiency in terms of computation time was also conducted. To arrive at 25% of the total route distance calculated by both the benchmark GA and the alternative GA, the alternative GA required 68% less time required by the benchmark.

**Figure 3, Alternative GA**

**Figure 4, Benchmark GA**

**Figure 5, Comparison of Algorithms**

|  |  |  |  |
| --- | --- | --- | --- |
| Fraction of the Initial Route Distance | Benchmark GA Time in Seconds | Alternative GA Time in Seconds | % Decrease in Time from Benchmark |
| 0.50 | 0.030277 | 0.015952 | 47.3% |
| 0.45 | 0.040719 | 0.01778 | 56.3% |
| 0.40 | 0.056924 | 0.023829 | 58.1% |
| 0.35 | 0.071448 | 0.029891 | 58.2% |
| 0.30 | 0.09109 | 0.04545 | 50.1% |
| 0.25 | 0.250238 | 0.079731 | 68.1% |
| 0.24 | 0.335671 | 0.087295 | 73.9% |

**Figure 6, Extended Generations Time Comparison of Algorithms**

1. **CONCLUSIONS**

Results displayed in the figures clearly indicate an improvement in performance of the team’s alternate GA. Both computation time and early generation convergence noted in the slopes are metrics for improved efficiency. The team alternative mutation method of using a two-dice random selection of candidate routes to employ a flip mutation on a full selection of city routes versus the benchmark method of segmenting a collection of routes into four groups, is most likely a key contribution to the improvement. This conclusion is based on the two-dice method being the only change from the benchmark producing a steeper-sloped early convergence. Increased numbers of generation iterations produced no additional significant improvements though the benchmark GA approached the apparent optimal route distance more closely than the team’s GA following additional, time-consuming iterations.

1. **RECOMMENDATIONS**

Additional research my find more efficient methods beyond those found in the alternative GA in this effort. Though randomness is a core operation, it may be exploited. This team did not evaluate multiple, clustered locations with GAs. We did not evaluate any three-dimensional scenarios that may be applicable to aerial vehicles, therefore we recommend effort in these areas.

1. **RESEARCH TRAINING**

RET participants began by familiarizing themselves with a selected variety of literature targeted for those unfamiliar with the TSP problem, followed by a review of the MATLAB programming language to be used in the TSP algorithm research effort. The project’s graduate student research assistant and mentor were guides throughout that process. A variety of TSP-related algorithm modules were provided to each participant and studied with the initial objective of becoming familiar with them. Those related algorithms provided stepping-stones toward enabling participants to improve the efficiency of the current TSP code.

1. **CLASSROOM IMPLEMENTATION PLANS**

Mr. Szyjka’s Classroom Implementation Plan

Mr. Szyjka’s classes are a community college group of students studying algebra to prepare for Federal Aviation Administration tests as certified aircraft mechanics with specialized disciplines in airframes, engines, electronics and digital electronics. Preparation for their unit activities will be through knowledge of plotting data in a scatterplot, evaluation of the fundamental properties of lines, higher polynomials and study of the Design of Experiments (DOE). Students will evaluate the fastest escape route in a simulation of the evacuation from an airliner and building. Simulation results will be used in a DOE, for scatter-plotting and further evaluation of the data for linear, polynomial or exponential fit.

Mr. Bagazinski’s Classroom Implementation Plan

Mr. Bagazinski’s class will use electronic fitness monitors to enable students to study their energy output enroute to their classes. They will evaluate kinetic and potential energies throughout their school day.

1. **BIBLIOGRAPHY**
2. Krzanowski, R., & Raper, J., (2001). “Spatial Evolutionary Modeling,” Oxford University Press.
3. Dianati, M., Song, I., Treiber, M., (2012). “An Introduction to Genetic Algorithms and Evolution Strategies,” Electrical and Computer Engineering and Mechanical Engineering, University of Waterloo, Ontario, N2L 3G1, Canada.
4. Sathyan, A., Boone, N. and Cohen, K., (2015). “Comparison of Approximate Approaches to Solving the Traveling Salesman Problem and its Application to UAV Swarming,” *International Journal of Unmanned Systems Engineering*. Vol. 3, No. 1, pp. 1-16.
5. Shiffman, D., (2012). “The Nature of Code,” Published by the Magic Book Project.
6. Applegate, D. L., Bixby, R. E., Chvátal, V., & Cook, W. J., (2007). The Traveling Salesman Problem. Princeton University Press.
7. **APPENDIX I: NOMENCLATURE USED**

Travelling Salesman Problem (TSP): The concept of determining the shortest route given a selection of possible routes over multiple destinations.

Genetic Algorithm (GA): A computational method based on the biological process of survival of the fittest, random mutations and selection, designed to be applied to more efficient computation of the TSP.

2-Opt: A computational technique frequently applied with GA methods within a TSP algorithm in which potential routes are crossed to determine route distances.

3-Opt: A computational technique frequently applied with GA methods within a TSP algorithm in which randomly selected routes are crossed with additional frequency to determine route distances. Similar to the 2-Opt with an additional route segment mutation.

Unmanned Systems, Unmanned Aerial Vehicle (UAV): The more common term for a single unit is “drone.”

1. **APPENDIX II: RESEARCH SCHEDULE**

Week 1: Travelling Salesman Problem (TSP) Introduction, Begin Professional Development (PD) Seminars

Week 2: Add Detail to Research, e.g., MATLAB code (re)familiarization, Familiarization with Genetic Algorithms, Additional PD Seminar, Project Field Trips/Tours

Week 3: Add Detail to Research, e.g., attempt edits to improve TSP benchmarked code, Project Field Trips/Tours, Continue PD Seminars, e.g., library/literature research methods

Week 4: Continue efforts at edits to improve TSP code, introduce complexity, i.e., Multiple Travelling Salesmen, Construct PowerPoint (PPT) project briefings and Project Posters

Week 5: Compare team genetic algorithm code to benchmark, begin writing report, constructing video documentation

Week 6: Summarize Unit Plans/Activities, complete NSF report, summary, PPT project briefing

1. **APPENDIX III: UNIT PLAN FOR MR. PETE SZYJKA**











1. **APPENDIX IV: UNIT PLAN FOR MR. MATT BAGAZINSKI**