**TEAM PROJECT REPORT**

**“Bio-Inspired Optimization of the Traveling Salesman Problem”**

**Submitted To**

**The University of Cincinnati**

**For**

**“Challenge-Based Learning and Engineering Design Process Enhanced Research Experiences for Middle and High School In-Service Teachers”**

**Sponsored By**

**The National Science Foundation**

**Grant ID No.: EEC-1404766**

**College of Engineering and Applied Science**

**University of Cincinnati, Cincinnati, Ohio**

**Prepared By**

**Ryan Wright, HS Biology, 9th and 10th grades, Ryle High School, Union, KY**

**John McClellan, Geometry and Algebra II, 9th - 11th grades, Reading High School, Reading, OH**

**Approved By**

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

**Dr. Jeffrey Kastner**

**Department of Engineering Education**

**College of Engineering and Applied Science**

**University of Cincinnati**

**Reporting Period: June 12 – July27 2017**

**1. ABSTRACT**

Genetic algorithms apply biological concepts to a guided programming search technique for intelligence systems. One such problem that can make use of such a tool is the Travelling Salesman Problem (TSP). By combining a 2-opt approach with the genetic algorithms, new, more optimal ways to tackle this problem may become more evident. Methods for combining these two separate strategies have been tested over a course of several weeks in a lab setting to find an ideal combination. Using MATLAB software, several iterations of these combinations were tested for their ability to find an optimal solution to the TSP and narrowed down over time. The program developed provides an optimal solution based on given criteria, and fitness levels have been compared to determine which program presented the best results. While it is not expected to find the absolute optimal solution, the presented research aims at providing a heuristically plausible solution that can be efficiently used. This in turn can then be used in other optimization-based settings such as transportation, logistics, and delivery.

**2. INTRODUCTION**

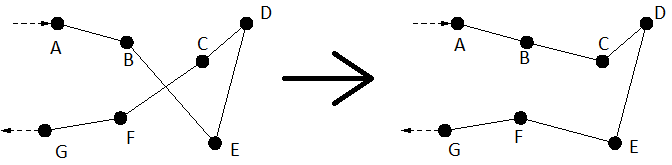
Our world is becoming more and more complex every day. As technology and connectivity advance with each passing year, the problems that arise in relation to these advances also become more and more complex. One such example of this is the Traveling Salesman Problem (TSP). The TSP asks someone to put themselves in the shoes of a salesman who has to travel through several different cities - he/she would like to plan the shortest path possible through these points in order to save themselves time and money. Advances in technology have made the problem even more relevant to today’s society with the advent of drones, worldwide deliveries, global flight connections, and more. Multi-million dollar companies are interested in figuring out the most efficient way to make these connections to save themselves resources. As the problem becomes more applicable on a global scale rather than a local one, the challenge of solving it becomes much more difficult. The number of possible pathways, P, the salesman can take is expressed by the formula,

,

where *n* represents the number of cities to visit. A salesman with only 4 cities to visit only has 3 different routes to choose from, but if that same salesman has to travel between 10 different cities, he now has over 1.8 million routes to choose from! The problem quickly becomes too large for a person to solve.

One method for finding a solution to the TSP is the use of artificial intelligence (AI) systems. AI can test many possible solutions faster and can find a better answer quicker than a human can, making it a very attractive method for solving the TSP. There are several different approaches to AI systems currently being explored.

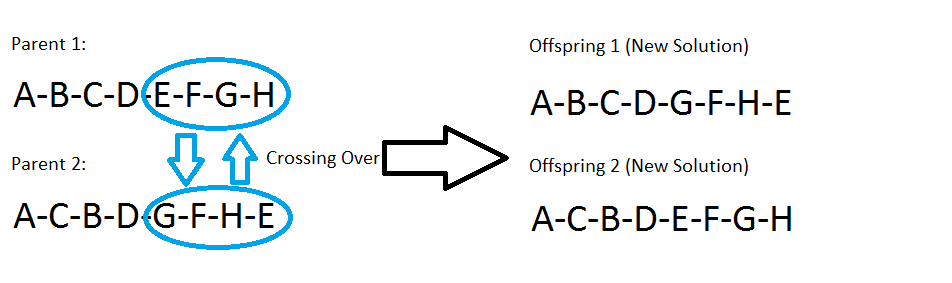
A 2-opt AI approach works to optimize a route by essentially eliminating instances where the path crosses itself. The system works by swapping nodes in a pair of edges along a given path and comparing the distances before the swap and after. A swap that decreases distance is kept while ones that increase the distance are ignored.



**Fig. 1. An example of 2-opt logic.**

For example, in Figure 1, the 2-opt system identifies that the pathway crosses in the *BCEF* region in Figure 1. The 2-opt system will eliminate this crossover by switching the route so that *BC* and *EF* are connected as well, which lowers the total distance.

Genetic Algorithms (GA) are another type of system that can be used to solve the TSP. GA is a programming concept that utilizes evolutionary principles (such as survival of the fittest and crossing over) in a non-biological setting. In GA, a possible solution to the TSP (in this case one of the possible routes) is expressed as a “chromosome”, with each individual point representing a “gene”. New possible solutions are made when 2 solutions “reproduce”, combining their chromosomes to make a new solution. The genetic process of crossing-over combines the two chromosomes of the parents to give the offspring a new combination in its chromosome, thus giving a new possible solution to the problem as seen in Figure 2.



**Fig. 2. Using Crossing Over to Produce New Solutions**

Survival of the fittest is then applied to these solution pathways to determine the most optimal path of the group; in this case, the “fitness” of chromosomes is the total distance of the path. The AI will compare the fitness of each of the children to each other and to the parents. As in nature, the most fit (shortest distance) survive to the next generation, while the less fit (longer distance) are eliminated (in nature, it fails to reproduce/propagate). Over time, the population of solutions “evolves” towards a more and more optimal path.

Both the 2-opt and GA offer particular strengths and weaknesses. The 2-opt system works very quickly and will provide an answer much faster than the GA. However, the solution provided may not be as optimal. GA takes longer to come up with an optimal solution, but the result tends to be better than the 2-OPT. This research attempts to merge the two systems to create a new hybrid system that will provide the best of both - a system that works faster than the typical GA system, while giving a more optimal solution than the typical 2-opt system would produce.

**3. LITERATURE REVIEW**

The Traveling Salesman Problem (TSP) was formalized first in 1832 (Applegate 2006). With various applications such as circuitry, transportation, and sequencing of jobs, heuristic solutions have become popular. In 2004, an optimal tour was found for slightly under 25,000 Swedish cities (Cook 2015).  Since this problem is considered to be NP-hard, an exact solution to the TSP is currently considered to be impossible. The Clay Mathematics Institute has offered a $1,000,000 prize for a proof or disproof of the feasibility of the TSP, as it would solve the standing P vs. NP problem (Cook 2015). The heuristic approach produces algorithms that focus on manipulating possible paths to find a more optimal route.

The *k*-opt approach attempts to find shorter paths by taking *k* routes and rearranging them in a defined order and then checking if that result is shorter than the previous path. In the case of two edges and their end points (usually referred to as “nodes”), the case is simplified to the 2-opt. In comparison, “the 2-opt and 3-opt are much better than the construction heuristics” that work from the ground up (Extreme Algorithms 2015).  The algorithm can check to determine if a swap of 2 edges gives a shorter path using the equation

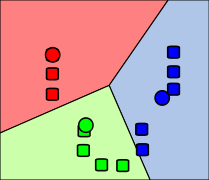
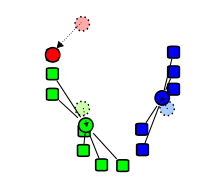
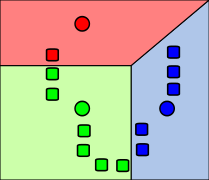
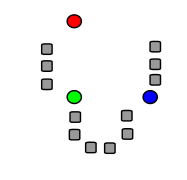
If the inequality is true, then a shorter path has been produced and *i+1* is swapped with *j*. Otherwise, the swap does not take place and the next set of edges is examined (Sathyan 2015). A 2-opt approach is typically able to reduce the total distance of the path for a given route, though other algorithms can produce better results at the cost of a higher time investment (Sabha 2012). Since the 2-opt takes into account all possible combinations of edge-swaps, it follows that the formula for *n* cities is

Comparing this with the brute force problem of significantly reduces the number the number of iterations the computer must check (Kuang 2012).  However, it is important to re-emphasize that the 2-opt strategy is a *heuristic* method that could potentially miss the most optimal route if the right initial given population is given.

Another non-construction approach is that of genetic algorithms. Established in the 1970s by John Holland at the University of Michigan, the approach uses mechanics of biological evolution to optimize solutions over a number of iterations. The strategy focuses on a large search space, but is not considered to be very quick (Peshko 2007). GAs have been used to solve several different types of engineering problems - manufacturing scheduling, image processing, signal process, robotics, protein folding, macromolecule analysis, path planning, and more. GA is often used in path planning in robotics as it provides a robust search and is less costly than other search algorithms (Dianati 2002). GAs can be divided into 2 groups: pure GAs and heuristic GAs. Pure GAs apply recombination to arbitrary permutations, while heuristic GAs incorporate heuristic information about a problem in choosing recombinations. When analyzing the TSP, a heuristic GA will take into account the observation that solutions to the TSP contain short edges. A heuristic GA will choose the shortest edges when an opportunity arises to select between edges. A pure GA is much more random in how it attempts to solve the TSP. Heuristic GAs outperform pure GAs (Jog 1990). GAs are effective because “short, above-average schemata receive exponentially increasing trials in subsequent generations” (Zaritsky 2015). These become building blocks towards building a near-optimal solution.

When the problem’s solution is an ordered list as in the TSP, simple crossing-over algorithms do not work due to nodes potentially being repeated multiple times (Larranaga 1996). There are several examples of crossover operators that do account for this, such as Partially Matched Crossover (PMX), Cycle Crossover (CX), Order Crossover (OX), and Edge Recombination Crossover (ERX). The specific of these algorithms are beyond the scope of this research, but OX is typically the fastest method, while ERX tends to give the best solutions.

An additional approach for solving the TSP involves clustering the cities into smaller groups, then determining an optimal path for each small cluster, essentially dividing one large problem into several smaller ones. *K-means clustering* is an effective method for creating these clusters from a given set of cities (Boone 2015). K-means clustering works to divide a group of *n* number of points into *k* number of clusters (Fig. 3). To do this, *k* number of random “means” are randomly selected within the data’s domain. Each individual point is then clustered with the nearest of these “k-means”, dividing the data domain into *k* clusters. For each cluster, a centroid point is calculated, and each point is then reexamined to determine which centroid it is now closest to. This forms the new clusters, and the whole process repeats until the clusters are stable and the points are no longer moving (Hamerly 2002). This random partition system is generally preferable to other k-means algorithms (Hamerly 2002). MATLAB has an in-built *kmeans* function that makes it easy to implement into any created algorithm.



*Random “means” Initial Clusters   Centroid Comparison New Clusters*

**Fig. 3. K-Means Clustering, Random Partition Algorithm**

**4. GOALS AND OBJECTIVES**

The main goal of this project is to develop an algorithm that can produce approximate solutions to the TSP efficiently. The closed-TSP requires the determination of the shortest route through a set of cities such that each city is visited exactly once and the salesman returns to the original starting city at the end. The objectives of this project are:

1. To learn to understand and write code in the MATLAB program.
2. To learn two popular algorithms that have been used to solve the TSP: Genetic Algorithms (GA) and 2-OPT.
3. To create a hybrid code using both GA and 2-opt methods together.
4. To test these hybrid codes for various TSP scenarios.
5. To analyze the results obtained.

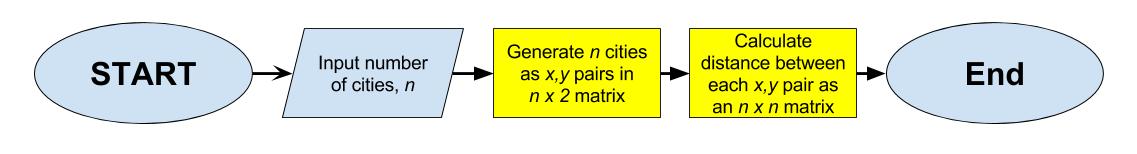
**5. RESEARCH STUDY DETAILS**

The first week of this project was dedicated to training the RET teachers on basic coding procedures, how to use MATLAB, and about the TSP itself. The teachers were asked to complete several practice exercises — both by hand and in the MATLAB program itself — to show that they had gained an understanding in how to do what needed to be done for completion of the project goals. Once the project GRA was satisfied with the progress of the RET teachers, work on the project itself was allowed to begin.

*NOTE: Section of the code have been included for reference in Appendix III. These figures are referred to in the text as “Fig. App1” and can be located at the end of the report if required for elaboration.*

5.1 Problem Formulation: Defining the TSP and Distance Matrix:

Before any algorithms could be applied to the TSP in MATLAB, the TSP had to be coded into the program itself (Fig. 4).



**Fig. 4. TSP Generation Algorithm**

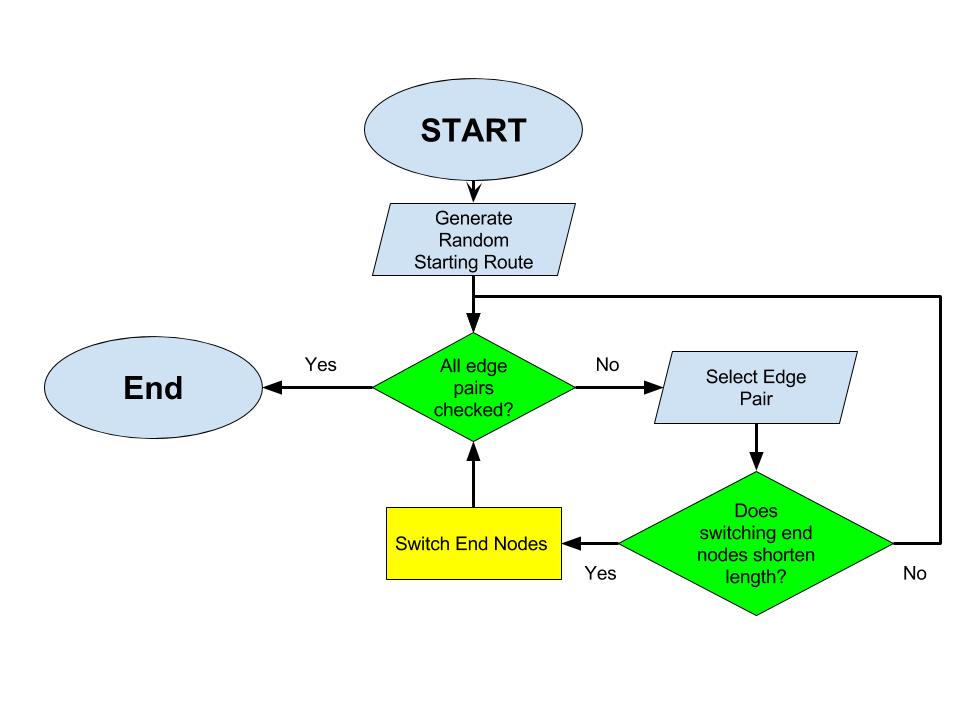
First, the number of cities for the TSP had to be decided. Rather than have a set number programmed in, an input command was used to allow the user to set that particular parameter of the problem (Fig. App1). MATLAB then randomly selects an X and Y coordinate for each city, spreading them across the “map”. The positions are stored in a matrix for later use.

Next, the program calculates a distance matrix (Fig. App2). This *nxn* matrix (where *n* is the number of cities) stores the distance between every pair of points. For example, the row 4, column 5 entry contains the distance between city 4 and city 5. MATLAB uses this distance matrix to calculate the total distance of any given path.

Now that the program has determined all city locations and the distance between each city, MATLAB can work to optimize a path with the selected algorithms. For this project, the TSP was solved using a 2-OPT approach, a GA approach, and a hybrid algorithm combining the two.

5.2 Coding the 2-opt Algorithm:

With the 2-opt approach, the algorithm seeks two edges and compares the sum of the length of the two edges to the sum of the length of the edges when the end nodes are switched (Fig. 5).



**Fig. 5. A 2-opt algorithm**

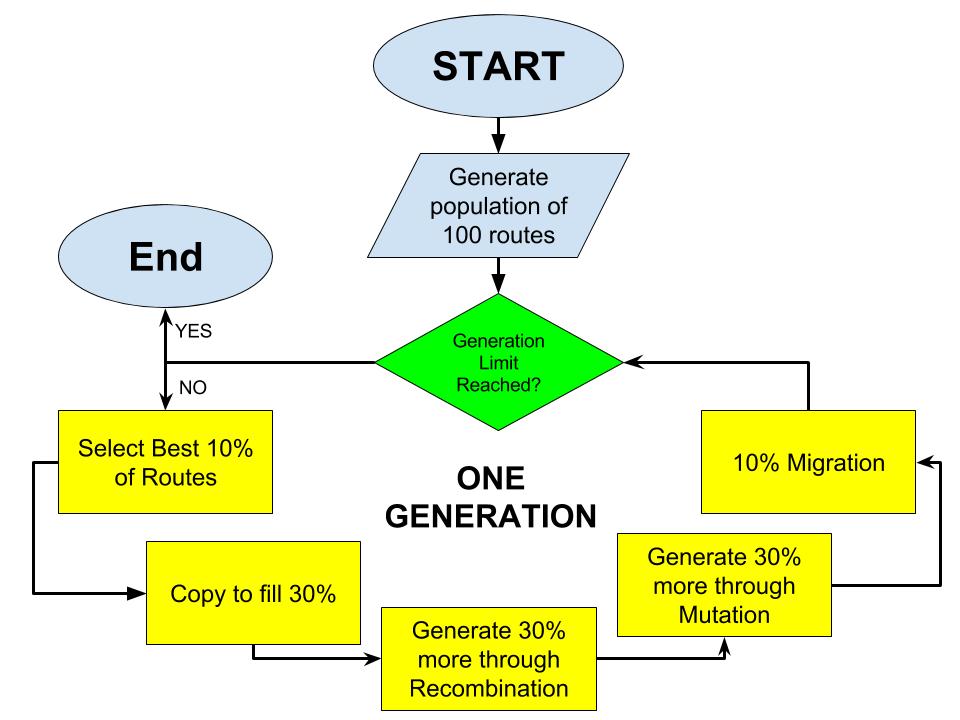
In this algorithm, an “edge” is defined by distance/path between two city locations within a single route. Thus, when comparing edges, taking the *i* and *i+1*th location in the route, plus the *j* and the *j+1*th location in the route, we see if the following inequality holds:

*distance(i,i+1) + distance(j, j+1) > distance(i, j) + distance(i+1,j+1)*

If it does, the algorithm switches *i+1* and *j* locations within the route (Fig. App3).  Usually, this occurs when a crossing of edges occurs, which lengthens the overall route length.  Using two nested for loops, this process works left to right on the route, making switches when the inequality is satisfied.  There is a possibility that two consecutive switches leads to yet another crossing, which is why it is worthwhile to run the 2-opt algorithm a number of times.

5.3 Coding the Genetic Algorithm:

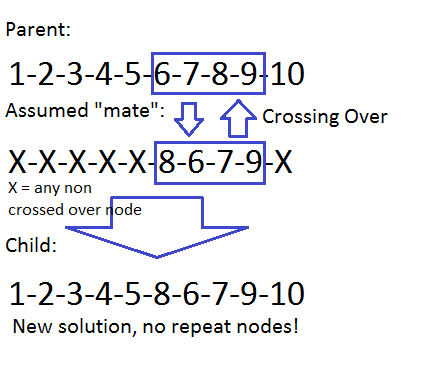
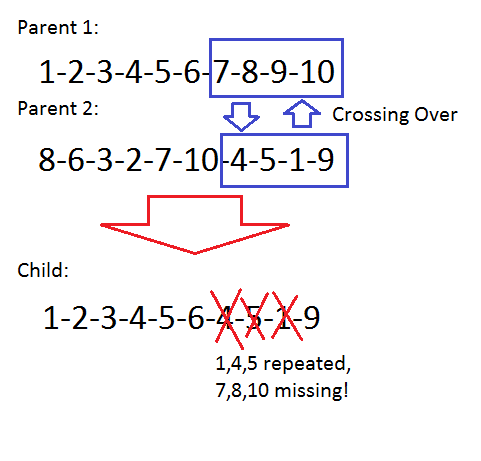
Genetic algorithms typically consist of 3 parts: natural selection, mutations, and recombination. The chosen approach was to make use of all three rather than focus on one in particular (Fig. 6). This GA variant looks at a large population of possible solutions and puts them through these processes to create new ones, refining the path into a more and more “fit” (optimal) solution as generations pass.



**Fig. 6. A Genetic Algorithm**

The GA begins by creating a starting population of 100 random solutions. For each of these solutions, a total distance is also calculated. The solutions are sorted in a matrix based on these distances (their “fitness”), with the shortest path being placed in row 1 and the longest being placed in row 100. Natural selection is simulated by removing the bottom 70% of the solutions - survival of the fittest! (Fig. App4) The remaining solutions are allowed to continue in the algorithm where they will be used to “breed” new solutions.

These selected individuals are used to create offspring solutions to fill in the rest of the population. First, each of the selected solutions is put through a recombination algorithm, simulating the process of crossing-over during reproduction. One big problem with using crossing-over when solving the TSP is that you often end up with new solutions that have the same node on it twice, which doesn’t work within the parameters of the TSP (Fig. 7). This approach had to prevent this from happening and so a compromise was made: Once a section for crossing over was selected, the “mate” solution would be assumed to have the exact same nodes in that exact same section, just in a different order (Fig. 8). This ensures that the new “child” solution only has each node appear once, while still being a combination of two different parent solutions.



**Fig. 7. Standard Crossover Fig. 8. Proposed Crossover Method**

Crossing-over occurs at random locations in nature, and so the algorithm first chooses a random point for the cross over to begin in the parent solution. Starting at this point, a portion of the parent solution of length *x* (*x* is 40% of the solution length) is removed, shuffled, and placed back into the parent solution, producing a new “child” solution (Fig. App5). A total distance is then calculated for this new solution. The new distance and path is stored in the original population matrix, giving us a population of 60 solutions at this point.

For the next 30 solutions in the total population, a mutation algorithm is used. The algorithm takes each of the original selected for solutions (1-30 in the population matrix) and has them go through the process of reproduction/recombination once more. This time, however, the offspring are potentially born with chance mutations, further changing the child solutions that are produced. This produces 30 new child solutions, which are stored in rows 61-90 of the population matrix.

The mutation algorithm itself is based on a randomizer. For each node in the solution, a die is rolled (Fig. App6). If the die lands on one side, a mutation occurs and the algorithm swaps that node with the next node in line. If the die lands on any other side, no mutation occurs at all at that node. This can result in child solutions that are either very similar or very different to the parent.

Finally, to fill out the population of 100 and to add more diverse solution for better chances at optimization, migrants are introduced. The final 10 rows of the population matrix are filled in with random solutions (Fig. App7), introducing new “genetic information” into the original population.

The set of genetic operations are iterated over a set number of generations. After each generation, the GA looks at the best solution produced out of the entire population and compares its distance to the previous best solutions, keeping only the optimal solution.

5.4 Optimizing the GA:

        This genetic algorithm has a lot of variables that can affect the quality of solutions that are produced. When the GA was first coded, arbitrary values were put in for variables as the portion of the population selected for by natural selection, the rate of mutation (default setting was a roll of a die, with a 1/6 chance that each node could become mutated), and the length of the recombinant section (the default setting was 40% of the total solution length). These values needed to be optimized to ensure that the GA was producing the very best solutions possible.

        The 2-opt algorithm was used to establish a “benchmark” for a given TSP. This was run several times for the exact same TSP to produce a series of optimal solutions. The default GA was then used to solve the exact same TSP, and the solutions from each were then compared. Variables within the GA were then manipulated to try to bring the solution from the GA closer to what was produced by the 2-opt solution (if not better). A “sweet spot” was located for each of the three main GA variables, and these values were used in testing the GA from here on out.

Data for the process of optimizing the GA can be seen below (Table 1). Each variable was isolated and tested independently (without manipulating any other variable) to measure its effect on the solution produced. Multiple trials (50) were run for each test and an average distance was calculated. Once a large range of potential values had been observed, the optimal value for each variable was identified. These values are highlighted in Table 1.

**Table 1. Finding Optimal Value for GA Variables**

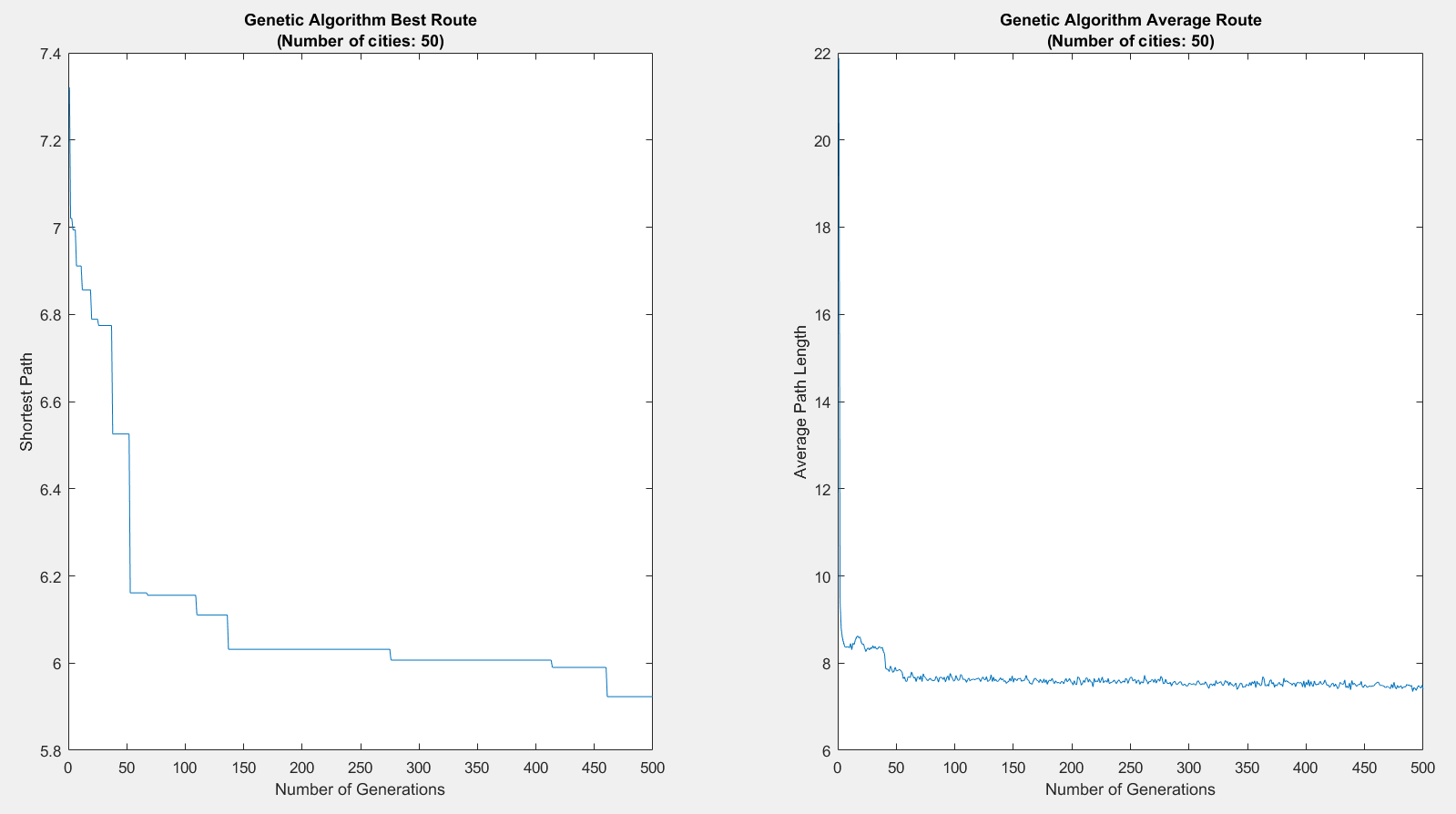
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | Average Distance |  | Average Distance |  | Average Distance |
| Mutation=1/4 | 17.61 | Recomb=50% | 18.02 | Selection=Top 10% | 11.72 |
| Mutation=1/6 | 17.59 | Recomb=40% | 17.29 | Selection=Top 15% | 12.15 |
| Mutation=1/8 | 17.04 | Recomb=30% | 16.86 | Selection=Top 20% | 15.98 |
| Mutation=1/10 | 16.92 | Recomb=20% | 16.11 | Selection=Top 30% | 17.54 |
| Mutation=1/20 | 16.72 | Recomb=15% | 15.53 | Selection=Top 40% | 17.66 |
| Mutation=1/40 | 17.12 | Recomb=10% | 15.94 | Selection=5-35% | 17.61 |
| Mutation=1/60 | 17.27 | Recomb=5% | 16.86 | Selection=10-40% | 18.2 |

Once an optimal value for each of the three main GA variables had been identified, all three of these values were plugged in at once to compare the results to the initial control GA over multiple trials (Table 2). The average distance produced with these variables was much more optimal than those produced by the initial GA.

**Table 2. Comparing Initial GA to Optimized GA**

|  |  |
| --- | --- |
| **GA tested:** | **Average Distance Produced:** |
| Initial GA (Mutation 1/6, Recomb 40%, Select top 30%) | 17.46 |
| Optimized GA (Mutation 1/20, Recomb 15%, Select top 10%) | 11.61 |

Finally, the number of required generations that needed to be run to produce an optimal solution was examined. Intuitively, it was assumed that as more generations were run the GA would continue to refine its answer to produce a better result, but it was observed that results leveled off at a certain point (Fig. 9). After reviewing multiple test runs, results were generally observed to reach a steady point after approximately 200 generations, though occasionally a small drop in distance would be seen after this point. To account for this, all testing was done for over 500 generations.



**Fig. 9. Effect of Generations on Solutions Produced**

5.5 Hybrid Algorithm:

In an effort to gain the advantages of both types of algorithms at the same time, a hybrid algorithm was created. It was decided that the 2-opt algorithm would be implemented in the migration step of the genetic algorithm as a first approach to hybridization. In the base genetic algorithm, each generation has 10 “migrant” solutions added to the population. The thought process behind this was that it would introduce new “genes” into the pool of solutions, hopefully presenting new and beneficial combinations into the mix. One problem with this approach, however, is that these solutions are completely randomized, and so you are just as likely to get very poor solutions as well. By adding a 2-opt algorithm here, this should no longer be an issue as each randomly created “migrant” is first optimized via the 2-opt algorithm, ensuring that solutions presented to the population are much more optimal. Every other aspect besides the migration code was kept untouched from the optimized GA.

5.6 Clustering:

The idea behind researching this approach is to include a “multiple traveling salesmen” applicability. If the total cost to a route for one salesmen can be reduced by delegating *k* available idle resources while reducing the computational and operational time by 1/k, the problem has been further optimized.

Clustering of the population of cities was accomplished by a standard *k-means clustering* algorithm. MATLAB’s inbuilt function *kmeans* was used to generate three clusters for any defined TSP. Each cluster generated was treated as its own independent TSP and the entire aforementioned algorithm was run thrice, albeit for significantly smaller problems.

5.7 Testing the Algorithm:

Four algorithms were tested in total: the 2-opt algorithm, the genetic algorithm, the hybrid algorithm and algorithms applying clustering to 2-opt and hybrid algorithms. Tests were run for a 10-city, 50-city,100-city, and a 1000-city TSP, with the exact same set of city coordinates given to each algorithm. Each algorithm was iterated for 100 trials. The minimum distance, average distance, median distance, and standard deviation of the distances for all trials was recorded. A specific focus was given to the minimum computational time to get results. This has been elaborated upon later on in this report.

**6. RESEARCH RESULTS**

6.1 “Brute Force” and Troubleshooting:

A “brute force” algorithm was created and considered for testing purposes to give another point of comparison in the data. This algorithm tests every single conceivable combination and calculates a distance for each in order to determine what the optimal path is. This “brute force” algorithm first solved the 10-city TSP. This allowed us to identify the exact minimal distance possible for this problem and was an essential step in troubleshooting our algorithms for errors. Once the project algorithms were able to replicate this distance produced by “brute force”, it could be concluded that both were working properly.

The brute-force algorithm, however, was unable to solve a 50-city problem due to the sheer size of the problem – the calculations were too big for the program to run – so this algorithm had to be left out of all future testing. One final point of note is that while the “brute force” algorithm was not able to be fully tested on a wide range of TSPs, we did observe that the time it took to produce these optimal solutions was approximately 3.5 seconds. This time was far longer than the 2-OPT, GA, and hybrid algorithm produced for the same 10-city problem, demonstrating why these algorithms are so desirable for solving the TSP to begin with, even in the presence of supercomputers that can deal with larger data memory

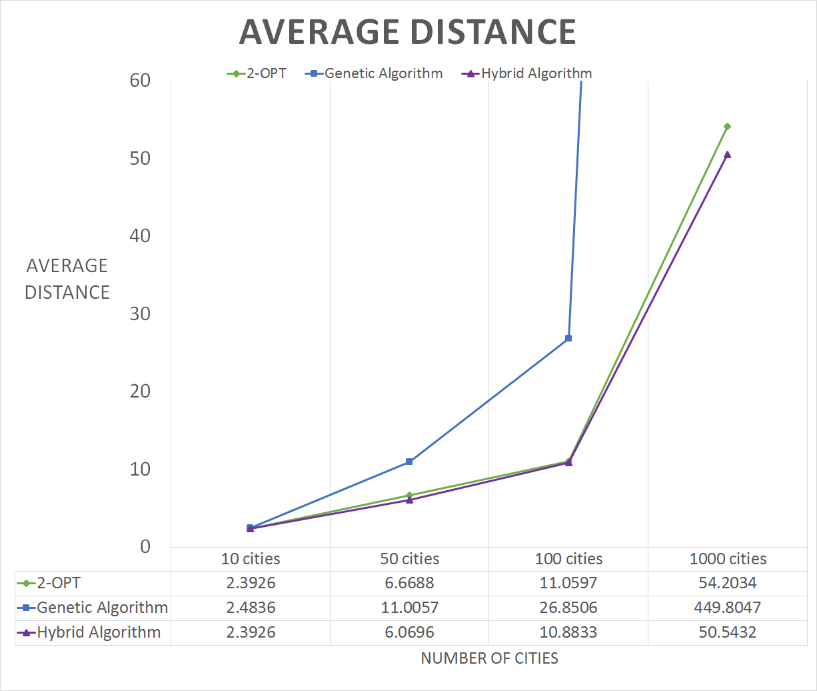
6.2 Comparing Algorithms:

For comparing algorithms, each algorithm was used to run a 10-city, 50-city, 100-city, and 1000-city TSP. Each TSP problem that was given to the algorithms was the exact same problem; that is, each algorithm worked off the exact same set of points to find an optimal route. Each algorithm ran through the same problem 100 times, and the data was collected and aggregated into the table below (Table 3). Optimal values for distances and calculation times have been highlighted.

**Table 3. Aggregate Data Over 100 Trials**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  |  | 10 cities | 50 cities | 100 cities | 1000 cities |
| 2 OPT | Minimum Distance | 2.3926 | 6.2547 | 10.2421 | 51.4687 |
| Average Distance | 2.3926 | 6.6688 | 11.0597 | 54.2034 |
| Median Distance | 2.3926 | 6.6780 | 11.0576 | 54.2668 |
| Standard Deviation | 4.65E-15 | 0.1322 | 0.3298 | 1.1012 |
| Minimum Time | 0.0362 | 0.1838 | 0.4151 | 26.3521 |
| Proposed GA | Minimum Distance | 2.3926 | 8.3567 | 23.1107 | 441.2372 |
| Average Distance | 2.4836 | 11.0057 | 26.8506 | 449.8047 |
| Median Distance | 2.3926 | 10.9368 | 27.0658 | 450.2021 |
| Standard Deviation | 0.1046 | 0.9366 | 1.2271 | 3.1570 |
| Minimum Time | 1.2441 | 3.7929 | 6.8581 | 62.5668 |
| Hybrid Algorithm | Minimum Distance | 2.3926 | 5.7500 | 9.8819 | 50.5432 |
| Average Distance | 2.3926 | 6.0696 | 10.8833 | 51.4909 |
| Median Distance | 2.3926 | 6.0511 | 10.4003 | 51.4332 |
| Standard Deviation | 5.34E-15 | 0.1289 | 0.1766 | 0.6226 |
| Minimum Time | 1.5696 | 12.2635 | 36.6365 | 2197.5000 |

When examining the algorithms for the minimum distance obtained, it was observed that the hybrid algorithm gave the most optimal solutions in terms of distance (Fig. 10). However, there was a large tradeoff, as the hybrid algorithm also required the most time to produce that solution. The 2-opt approach did not give as optimal of a solution, but does so in a far more timely manner. The GA produced by far the least optimal distances and did not seem to be doing that well at solving the TSP by itself. With the hybrid doing so well, however, it can be argued that the GA provided a foundation to optimize solutions over the 2-opt approach itself.

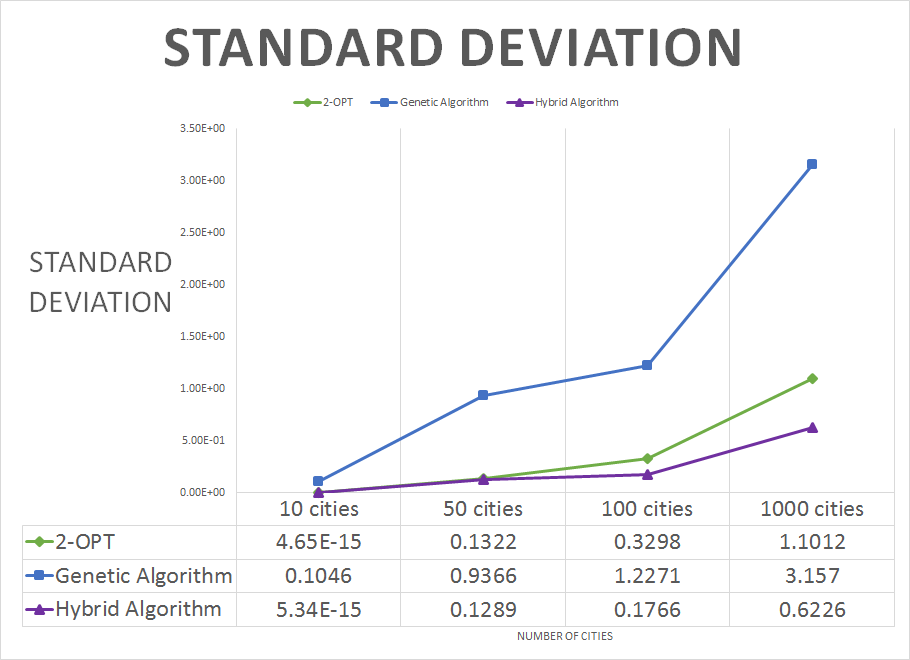


**Fig. 10. Average Distances Produced for a 10, 50, and 100 City TSP**

For the 10-city TSP problem, we were able to use a “brute force” algorithm to determine exactly what the minimum possible distance was – 2.3926. Both the 2-opt and the hybrid were able to produce this exact result, demonstrating that they were performing optimally. The GA did not produce 2.3926 as its average distance, but was able to produce the value during at least one of its trials.

The GA distances fall well behind the other two algorithms as the problem gets larger. It was thought that maybe this had to do with the number of generations being ran – perhaps a larger problem would require more generations to allow selection to work towards a more optimal answer. However, repeating the exact same 100-city TSP over twice as many generations (1000 as opposed to 500) still produced approximately the same answer. Increasing the size of the population in this same manner also did not affect the results produced.

The hybrid algorithm also performs best in terms of standard deviation (Fig. 11) for each of the TSPs tested, producing more precise data than the 2-opt and substantially better than the GA. The higher standard deviations seen in the genetic algorithm explain why the genetic algorithm produced a higher than optimal average distance despite occasionally producing the most optimal distance possible.



**Fig. 11. Standard Deviations for a 10, 50, and 100 City TSP**

One important thing to note here is that these distances produced are unitless and are relative to scale, that is, the size of the area these cities are spread over. For example, when looking at a 100-city TSP the 2-opt algorithm produces an optimal distance of 6.6688, while the hybrid algorithm produces an optimal distance of 6.096. The difference between the two is 0.5728, which does not seem like a large distance. If, however, these 100 points were spread over an entire country as opposed to a coordinate plane, that difference of 0.5728 might equate to thousands of miles, making that quite a large difference indeed. The large the area that these points are spread over, the larger the distance differences, making the hybrid algorithm much more attractive as the area of the problem increases.

Clustering:

The primary motivation behind the idea of clustering with *k*-clusters is that it reduces a *(n-1)!* problem into a *(n-k-1)!* problem, which would lead to a significant change in computing time. To first test this theory, a standard 2-OPT, a 2-opt with 2 clusters, and a 2-opt with 3 clusters were all compared over 100 trials of the same problem, each with a different starting solution permutation to work off of. There was also a comparison of data for a random starting permutation versus a nearest neighbor starting permutation for all three trials for the sake of variability. The clustering algorithm utilized the 2-opt in each cluster, and ran the algorithm through each cluster consecutively. For initial testing, the city limit was restricted to 50 cities. Clustering the cities reduced the distance, but increased the time (Table 4). Likewise, adding a nearest neighbor initial route increased time but reduced overall distance uniformly about each algorithm.

**Table 4. Effect of Clustering on Algorithm Performance**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | 2-OPT (Random) | 2-OPT  (NN) | 2 Clusters (Random) | 2 Clusters (NN) | 3 Clusters (Random) | 3 Clusters (NN) |
| Time (secs) | 0.1599 | 0.1279 | 0.5643 | 0.6047 | 0.3542 | 0.5834 |
| Min Distance | 6.7347 | 5.9152 | 6.3028 | 6.1582 | 6.0155 | 5.7115 |

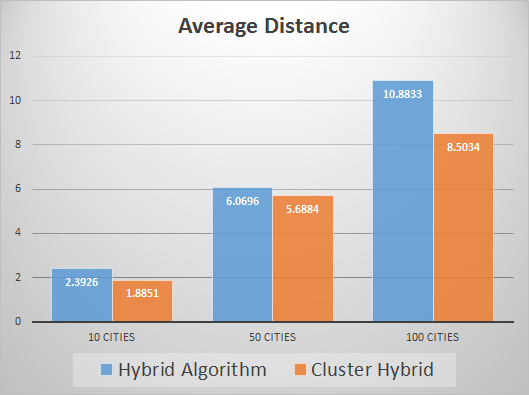
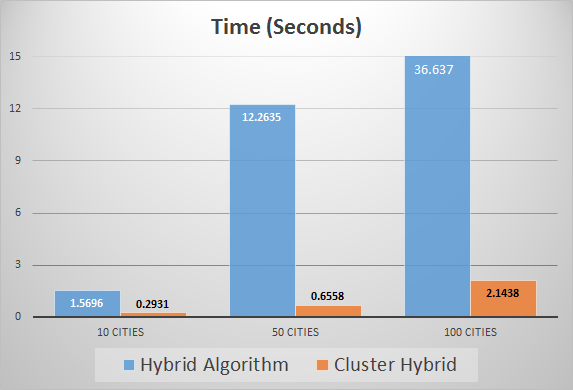
It is important to note that these times were total computing time where the cluster paths were run consecutively, not concurrently. Thus, if these clusters were running parallel, the maximum time for one cluster would represent the computational time. This could be a further research study topic.

The idea that the cluster algorithm might speed up the hybrid algorithm was not diminished because of the results of the 2-opt comparison. Based on the dramatic growth in time for the hybrid between trials with more cities, it could be expected that the initial idea of breaking a (n-1)! problem into a (n-k-1)! problem here might have an even larger effect. The clustering method improved the overall distances for each added cluster in the 2-opt test; thus, although the hybrid already produced the best distances, there was a chance the clustering method applied to the hybrid might produce even more optimal routes.

The clustering function was implemented into the hybrid algorithm. The clusters were re-established for each of the 100 trials, and the same data collection that was gathered for the previous algorithms were gathered for a 3-cluster hybrid algorithm. The results (Table 5) show a dramatic increase in speed (Fig.12) compared to the standard hybrid genetic algorithm, and the distances dropped as well (Fig. 13).

**Table 5. Hybrid Algorithm w/ Clustering Results**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | 10 cities | 50 cities | 100 cities |
| Cluster Hybrid | Minimum Distance | 1.7741 | 5.4006 | 8.1810 |
| Average Distance | 1.8851 | 5.6884 | 8.5034 |
| Median Distance | 1.9160 | 5.6856 | 8.4700 |
| Standard Deviation | 0.0682 | 0.1774 | 0.1657 |
| Minimum Time | 0.2931 | 0.6558 | 2.1438 |



**Fig. 12. Nonclustered vs. Clustered Times Fig. 13. Nonclustered vs. Clustered Dist.**

There is an expectation that the geometry of clustering lends itself to a lower distance by its very nature. The very premise of the problem potentially changes here, as this can be viewed as more of a multiple TSP than the traditional TSP. However, even when viewed as a traditional TSP the consecutive running of the hybrid’s clusters still reduced the overall time by more than a factor of 10. Thus, even if the salesman in question were to treat these as three separate open-routes that he or she would take consecutively, the time spent calculating is still dramatically lower. There is also room for further interpretation while in converting this multiple salesman problem into a single route for a single traveling salesman, with minor fixes in the algorithm.

**7. CONCLUSIONS**

In conclusion, three separate algorithms to solve a generic TSP were developed. The advantages and disadvantages of each algorithm were observed and combinations were developed to optimize a problem. Laying focus on a heuristic approach, this research enabled an intricate understanding off the process of optimization within limiting criteria.

The hybrid algorithm produces the most optimal routes in terms of distance, but requires a larger time investment. The 2-opt algorithm runs the fastest, though the routes it produces will not be the most optimal. Which algorithm is “the best” is dependent on the focus point of the optimization. If the shortened distances provide large enough savings in cost, resources, or travel time, then the extra time invested in planning it out may very well be worth it. If, however, the distance differences are small enough not to create the large savings, then using the 2-opt algorithm to produce a route in a quicker amount of time is more efficient.

Clustering allows these algorithms to break down larger problems into smaller ones. Clustering showed a positive effect on both the 2-opt and hybrid algorithms solution distances and computational times, significantly so for the hybrid algorithm. This suggests that clustering is a great option for continuing to optimize the TSP and warrants further study.

**8. RECOMMENDATIONS**

For continued research on this topic, it would be beneficial to continue to try to monitor and refine the hybrid algorithm in hopes of making it perform much faster than it currently does. The possibility that there are unnecessary tasks or functions that are part of the algorithm that do not affect the quality of the solutions produced but that do slow down the computing time should be looked into. If these could be identified, they could be removed from the code to produce solutions more quickly.

A second recommendation is to continue to manipulate the variables in the GA tested for optimization above. There are infinite combinations of values that can be tested, and perhaps different combinations will be able to produce even better results. It is also likely that the optimal values may vary depending on the size of the TSP tested – the optimization of the GA was done on a 50-city problem, but these optimal values may change as the problem gets larger and larger.

One last recommendation is to look deeper into how clustering can impact the performance of the genetic and hybrid algorithms. Examining this aspect in more detail and continuing to refine it (such as the number of clusters examined, clustering methods, etc.) could produce even better results than what was produced.

**9. CLASSROOM IMPLEMENTATION PLAN**

Ryan’s classroom implementation plan focuses on taking the main goal of this project (optimization) and connecting it to the process of natural selection. Students will focus on the biological ideas of fitness, adaptations, and survival of the fittest throughout this unit. The unit begins with two different activities designed to show students the connection between adaptations and fitness. First, they will play a game that asks them to reflect on their own fitness at different activities and why that may be the case. This leads into a discussion of the connection between adaptations and fitness, and how these adaptations may have come to be in the first place. The second activity continues this line of instruction through a lab activity related to bird beaks. Students perform different tasks using different tools to represent different kinds of beaks, then reflect on the advantages and disadvantages that each beak type has. This allows students to infer things about the niche that these birds inhabit and drives them towards considering how natural selection shaped them the way it did. Students formulate questions about the topic of natural selection through these activities which lead towards a challenge related to the idea of how natural selection works. The unit then focuses on optimization and computational thinking, showing students how to make a flowchart and having them practice this be applying it to different household tasks. Finally, students connect these two main ideas (natural selection and computational thinking) with the unit’s main challenge - creating a flowchart or algorithm showing the process through which nature optimizes different life forms towards their particular niches (in other words, the process of natural selection). This allows students to demonstrate a clear understanding for how natural selection occurs.

John’s classroom implementation plan takes the MATLAB programming aspect of the project and relates it to computer programming in general.  The focus will be a simple introduction to computing language and syntax using a web-based compiler called Scratch.  Students will first explore how computers work and “think” by playing games and writing out common abilities/recognitions of the computer.  The challenge will be introduced in the first activity after much discussion on the global importance of the ability to read and write code.  Students will be designing their own computer program to run an automated function, such as a robot navigating through the maze.  After establishing the challenge, students will run through activities and analogies of computer language, discovering that computers are good at storing memory (variables), running the same function over and over again really quickly (loops), and making a quick yes-or-no choice (Boolean logic).  The activities lead them all the way down pseudo-code writing before first pulling up any tutorials on Scratch.  When the time comes to implement the challenge, students will work in groups and follow the Engineering Design process to refine and communicate their programs to the class.  The goal of the unit is to display to the students the importance of knowing basic computer programming, discover how a computer operates and how it is used in our world, and to build up the 21st century skills in problem-solving, communication, and collaboration.

**10. ACKNOWLEDGEMENTS**

The RET program is funded by the National Science Foundation (NSF), Grant ID# EEC-1404766

Dr. Kelly Cohen and Dr. Jeff Kastner, project faculty mentors

Ms. Jutshi Agarwal, graduate research assistant

Dr. Anant Kukreti, RET project director and principal investigator

Ms. Debbie Liberi, RET resource person and grant coordinator

**11. BIBLIOGRAPHY**

(2015) "The Traveling Salesman Problem (TSP)." Extreme Algorithms. <http://www2.seas.gwu.edu/~simhaweb/champalg/tsp/tsp.html>. (June 23, 2017).

Applegate, D., Bixby, R., Chvatal, V., Cook, W. (2006). *The Traveling Salesman Problem: A Computational Study.* Princeton University Press, Princeton, NJ.

Boone, N., Sathyan, A., and Cohen, K. (2015). “Enhanced Approaches to Solving the Multiple Traveling Salesman Problem” *AIAA SciTech*. Kissimme, Florida.

Cook, W. (2015). "The Problem." University of Waterloo, Ontario, Canada. <http://www.math.uwaterloo.ca/tsp/problem/index.html>. (June 23, 2017).

Dianati, M., Song, I., and Treiber, M. (2002). “An introduction to genetic algorithms and evolutionary strategies.” N2L 3G1, University of Waterloo, Ontario, Canada.

Hamerly, G., and Elkan, C. (2002). ["Alternatives to the k-means algorithm that find better clusterings"](http://people.csail.mit.edu/tieu/notebook/kmeans/15_p600-hamerly.pdf) *Proceedings of the eleventh international conference on information and knowledge management (CIKM),* ACM, 600-607.

Jog, P. Suh, J., and Gucht, D. (1990). “Parallel Genetic Algorithms Applied to the Traveling Salesman Problem.” (1991). *SIAM Journal on Optimization,* 1(4), 515-529.

Kuang, E. (2012). “A 2-OPT-based Heuristic for the Hierarchical Traveling Salesman Problem”.  University of Maryland, College Park, Maryland.

Larrañaga, P., et al. (1996). "Learning Bayesian Network Structures by searching for the best ordering with genetic algorithms". *IEEE Transactions on systems, man and cybernetics*, 26(4).

Peshko, O. (2007). “Global Optimization Genetic Algorithms” *McMaster University Hamilton, Ontario* *ppt presentation, 25*.

Sabha, S., and Chikhi, S. (2012). “Intergrating the Best 2-opt Method to Enhance the Genetic Algorithm Execution Time in Solving the Travelling Salesman Problem.” *Complex Systems and Dependability,* 195-208.

Sathyan, A., Boone, N., and Cohen, K. (2015). “Comparison of Approximate Approaches to Solving the TSP and Its Application to UAV Swarming.” *International Journal of Unmanned Systems Engineering IJUSEng,* 3(1), 1-16.

Zaritsky, A. (2015). “Introduction to Genetic Algorithms.” Ben-Gurion University, Israel. <https://www.cs.bgu.ac.il/~sipper/courses/ecal051/assaf-ga.ppt>. (June 23, 2017).

**12. APPENDIX I: NOMENCLATURE USED**

2-opt – an algorithm that works by swapping edges along a given path and comparing the distances of the path before and after

“brute force” algorithm – an algorithm that attempts to solve a problem by testing every possible solution

crossing-over – a biological mechanism resulting in the exchanging of genetic material between two chromosomes during meiosis

edge – a path between two cities

fitness – the ability of an organism to survive and reproduce

GA – genetic algorithm, an algorithm based on the principles of evolution used to find an optimal solution

k-means clustering – a mathematical method for dividing a large group of points into smaller clusters based on position on a coordinate plane

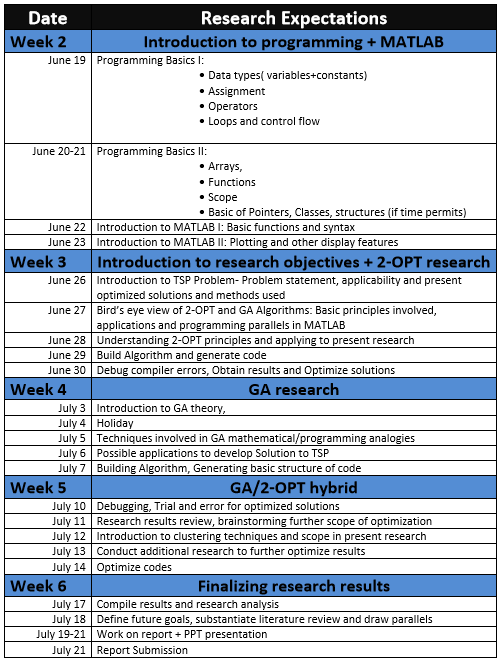
MATLAB – mathematical software used for coding the algorithms in this research

n – number of points/cities

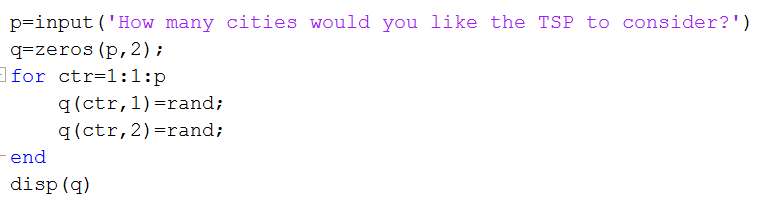
node – a city location on a map

TSP – traveling salesman problem, a problem that asks to find the shortest distance between a given set of points, where each point is only visited once and where the path ends where it began

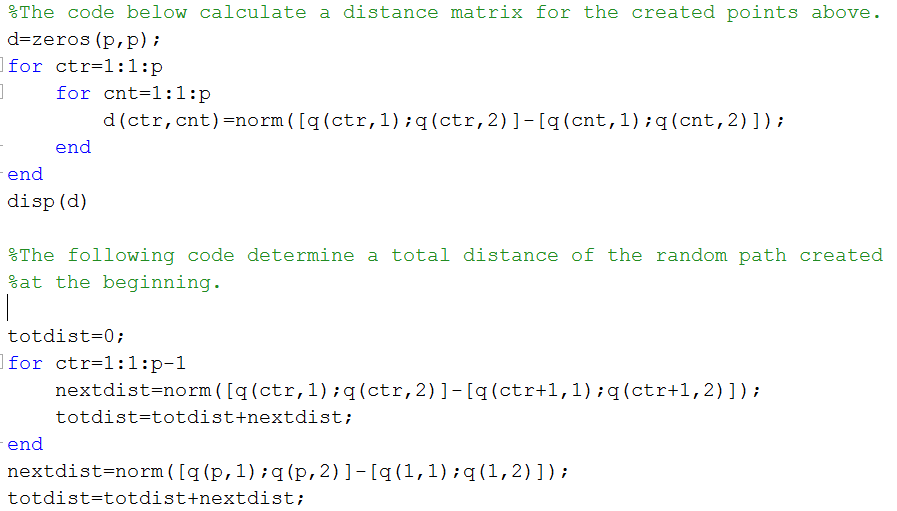
**13. APPENDIX II: RESEARCH SCHEDULE**

****

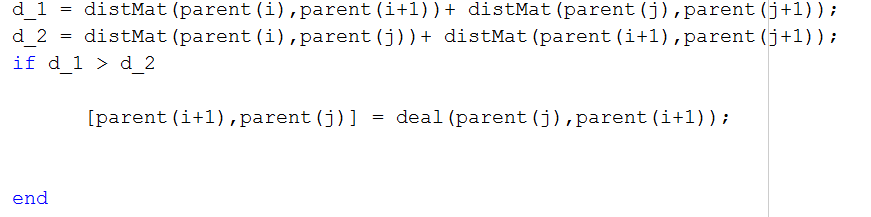
**14. APPENDIX III: SAMPLE CODE**



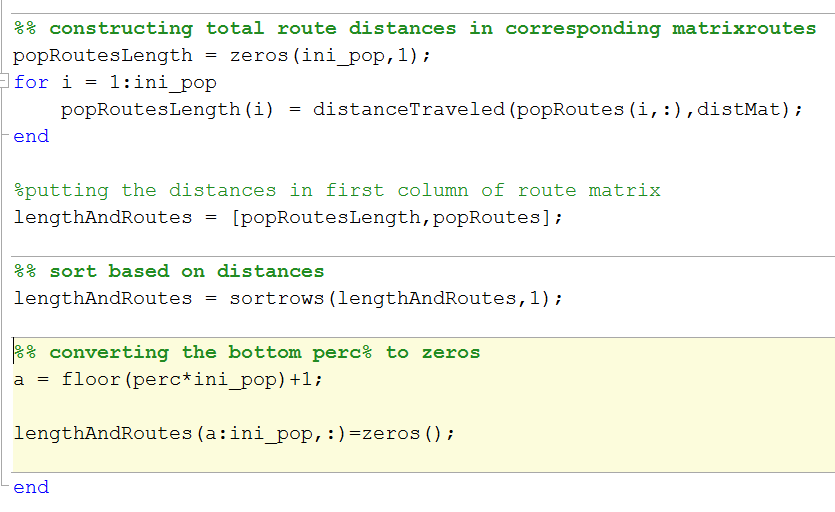
**Fig. App1. Randomizing a Given Number of City Locations**



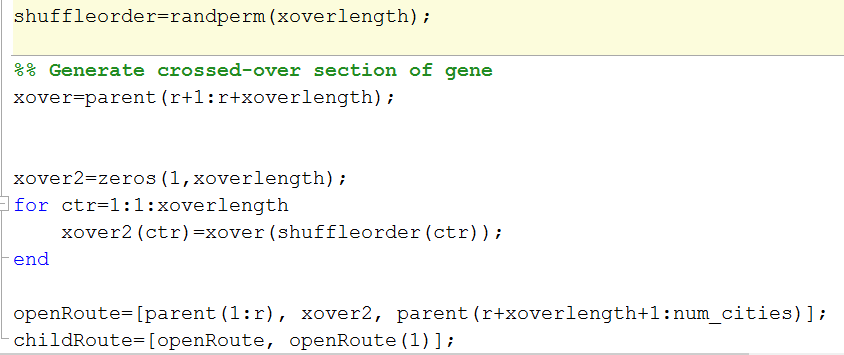
**Fig. App2. Calculating Distances.**



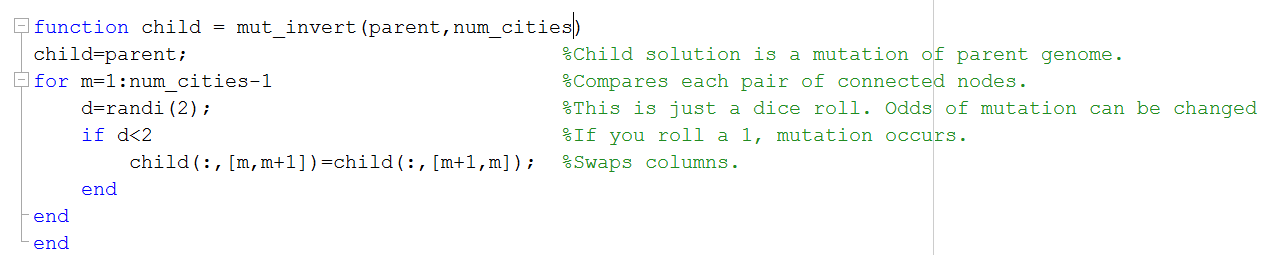
**Fig. App3. Deciding to swap.**



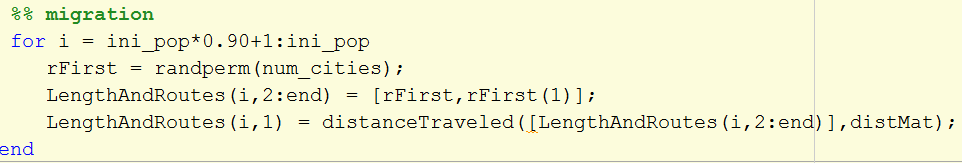
**Fig. App4. Natural Selection Algorithm.**



**Fig. App5. Crossing Over Code.**



**Fig. App6. Chance Mutations in a Sequence.**



**Fig. App7. Introducing Migrant Solutions.**

**15. APPENDIX IV: UNIT TEMPLATE FOR JOHN MCCLELLAN’S PROGRAMMING UNIT**

|  |  |  |
| --- | --- | --- |
| **Name:** John McClellan | **Contact Info:** jmcclellan@readingschools.org | **Date:** 6/21/17 |

|  |
| --- |
| **Unit Number and Title:** Unit 1: Computer Programming with Scratch |

|  |  |
| --- | --- |
| **Grade Level:** | 12 |

|  |  |
| --- | --- |
| **Subject Area:** | Mathematics |

|  |  |
| --- | --- |
| **Total Estimated Duration of Entire Unit:** | 7 Days |

**Part 1: Designing the Unit**

|  |
| --- |
| 1. **Unit Academic Standards (**Identify which standards:NGSS, OLS and/or CCSS.Cut and paste from NGSS, OLS and/or CCSS and be sure to include letter and/or number identifiers.**):** |

**Standard for Mathematical Practice #2 Reason Abstractly and Quantatively; Standard for Mathematical Practice #5 Use Appropriate Tools Strategically; CCSS High School Modeling Domain: *“When making mathematical models, technology is valuable for varying assumptions, exploring consequences, and comparing predictions with data.”***

|  |
| --- |
| 1. **Unit Summary** |

The Big Idea (including global relevance):

*Computer Programming for the Future – In ten years, there will be a 1 million workforce shortfall in the computer science sector. Many companies are looking for some proficiency or at least familiarity with computer programming.*

The (anticipated) Essential Questions: List 3 or more questions your students are likely to generate on their own. (Highlight in yellow the one selected to define the Challenge):

*How is a computer programmed?*

*How does a computer perform functions compared to a human brain?*

*How does a computer think?*

*Why are computers assigned the tasks we assign each day?*

*What is a computer good for?*

|  |
| --- |
| 1. **Unit Context** |

Justification for Selection of Content– Check all that apply:

☐ Students previously scored poorly on standardized tests, end-of term test or any other test given in the school or district on this content.

x

☐ Misconceptions regarding this content are prevalent.

x

☐ Content is suited well for teaching via CBL and EDP pedagogies.

☐ The selected content follows the pacing guide for when this content is scheduled to be taught during the school year. (Unit 1 covers atomic structure because it is taught in October when I should be conducting my first unit.)

x

☐ Other reason(s) \_\_\_Need for computer literacy and programming knowledge for the sake of future careers\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

The Hook: (Describe in a few sentences how you will use a “hook” to introduce the Big Idea in a compelling way that draws students into the topic.)

*Students are to gather into groups or pairs and share their favorite mobile game. A brief survey about the interaction of human and computer is handed out to record and guide their conversation. Students are given time to play various games if they haven’t before, and to write down a few notes based on some questions on the survey. After about 8 minutes of play, we will gather up and discuss each game and what elements of the game made it fun.*

The Challenge and Constraints:

x

☐ Product **or** ☐ Process (Check one)

|  |  |
| --- | --- |
| Description of Challenge (Either Product or Process is clearly explained below): | List the Constraints Applied |
| **Use Scratch to assemble a character to run an automation function (like going the through the maze)**  **Save the electric grid before time is up!** | **Time, Scope, Resources that Scratch can provide** |

Teacher’s Anticipated Guiding Questions (that apply to the Challenge and may change with student input.):

*How can I create pictures on the computer?*

*How do I instruct the pictures to move?*

*How can the computer recognize the mouse movement/click?*

*How can the computer recognize the keyboard strokes?*

*What software do we need?*

*What are the limitations of computers?*

*What are the strengths of computers?*

*How does Scratch compare with other languages?*

*What is in Scratch’s library?*

*What syntax does Scratch use for various functions?*

*How complex can my animation become?*

|  |
| --- |
| **4. EDP: Use the diagram below to help you complete this section.** |

****

How will students test or implement the solution? What is the evidence that the solution worked? Describe how the iterative process from the EDP applies to your Challenge.

*Students will have to draw out their strategies on paper first. Then, once they have everything ready, they can use my computer to type in and structure their program. Once the program runs, we check for any errors, and if it is error free, immediately see what we can do to improve the program. The program will produce error messages if there are any syntax issues, and students will be able to see right away if the program does not perform the intended task.*

How will students present or defend the solution? Describe if any formal training or resource guides will be provided to the students for best practices (e.g., poster, flyer, video, advertisement, etc.) used to present work.

*After finishing their code, students are to present their EDP to the class, including how they refined. They will not need a Powerpoint, but their game/animation will be visible to the class. Students will be given the EDP poster and be asked to identify each step. They will need to know Scratch and the EDP process, but be expected to outline informally.*

What academic content is being taught through this Challenge?

*Computer Programming, Computer languages, and EDP*

Assessment and EDP:

Using the diagram above, identify any places in the EDP where assessments should take place, as it applies to your Challenge. Describe below what kinds of assessment are most appropriate.

|  |  |
| --- | --- |
| What EDP Processes are ideal for conducting an Assessment? (List ones that apply.) | List the type of Assessment (Rubric, Diagram, Checklist, Model, Q/A etc.) Check box to indicate whether it is formative or summative. |
| \_\_\_\_\_\_**Seek Solutions\_\_\_\_\_\_\_**  **\_\_\_Evaluate Solution\_\_\_**  **\_Communicate Solution**\_  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | x  \_\_\_\_**Write-up**\_\_\_\_\_\_\_\_­­\_\_\_\_\_\_\_\_\_\_\_\_\_ ☐ formative ☐ summative  x  \_\_\_\_\_**Running Simulation**\_\_\_\_\_\_\_\_\_\_\_ x formative ☐ summative  \_\_\_\_**Present Final Program\_**­\_\_\_\_\_\_\_\_\_ ☐ formative x summative  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_­­\_\_\_\_\_\_\_\_\_\_\_\_\_ ☐ formative ☐ summative |

Check below which characteristic(s) of this Challenge will be incorporated in its implementation using EDP. (Check all that apply.)

x

☐ Has clear constraints that limit the solutions

x

☐ Will produce more than one possible solution that works

x

☐ Includes the ability to refine or optimize solutions

x

☐ Assesses science or math content

x

☐ Includes Math applications

☐ Involves use of graphs

x

☐ Requires analysis of data

x

☐ Includes student led communication of findings

|  |
| --- |
| **5. ACS (Real world applications; career connections; societal impact):** |

Place an X on the continuum to indicate where this Challenge belongs in the context of real world applications:

|  |  |  |
| --- | --- | --- |
| **Abstract or Loosely Applies to the Real World** | **|--------------------------------------|-----------------------XX-------------|**  x | **Strongly Applies to the Real World** |

Provide a brief rationale for where you placed the X:­­­­­­­­­­­­­­ **Though game design and coding applies to the real world, the challenge itself focuses on the need for more confidence in programming, not in deep programming skills.**

What activities in this Unit apply to real world context? **Analyze interaction with computer, playing games**

Place an X on the continuum to indicate where this Challenge belongs in the context of societal impact:

|  |  |  |
| --- | --- | --- |
| **Shows Little or No Societal Impact** | **|-------------------------------------|----------------------------------------|**  x | **Strongly Shows Societal Impact** |

Provide a brief rationale for where you placed the X: **­­­­­­­­­­­­­­Most industries need employees who have some ability to read code; even with the basic syntax learned over the course of a couple weeks, students will be better off than they were prior to the class. I am hoping this unit inspires students to pursue further learning in this field.**

What activities in this Unit apply to societal impact?**The Challenge helps understand the importance of learning code, both on a micro scale like job opportunities and a major scale like cyber terrorism.**

Careers: What careers will you introduce (and how) to the students that are related to the Challenge? (Examples: career research assignment, guest speakers, fieldtrips, Skype with a professional, etc.)

**Computer Scientist, Analyst, Computer Engineer, Cyber security. I will play a few select YouTube videos (probably from the ted talks and hour of code selection) to review the need for these careers for the coming generation.**

|  |
| --- |
| **6. Misconceptions:** |

* *Only “smart” people know how to code.*
* *I don’t need to know how to code for my future job.*
* *There aren’t that many jobs out there for coders.*
* *Computer programming is a boring subject.*
* *You need to know binary, 1’s & 0’s, to know even a little bit of programming.*

|  |
| --- |
| **7. Unit Lessons and Activities: (**Provide a tentative timeline with a breakdown for Lessons 1 and 2. Provide the Lesson #’s and Activity #’s for when the Challenge Based Learning (CBL) and Engineering Design Process (EDP) are embedded in the unit.) |

**Unit 1: Computer Programming—Becoming Technologically Literate for the Upcoming Programming Employment Shortfall**

**Lesson 1: How Computers Think –** (4 days)  
*Lesson 1 will focus on teaching students the background on how computer work and what they are good for compared to human abilities. Students will structure and formalize the logic of conditional statements, loops, variable definition, and keyboard/mouse interaction.*

Activity 1: Introduction of the Big Idea, Generating the Essential Question, Challenge & Guiding Questions **(1 day)**  
Activity 2: Introduction to Computer functions **(3 days)**

**Lesson 2: Coding With Scratch –** (3 days)  
*Lesson 2 introduces students to the Scratch online coding software. Working through a number of tutorials, students are expected to apply what they have learned about how computers work to the specific syntax of Scratch. The Challenge will be to create their own (as a team) animation on Scratch that uses all the functions we have outlined.*

Activity 3: “Flappy Bird” or other tutorials on Scratch **(1 day)**

Activity 4: Design a Computer Program on Scratch that combines all the functions we outlined **(2  
days)**

CBL: Lesson 1, Activity 1

EDP: Lesson 2, Activity 4

|  |
| --- |
| **8. Keywords:** |

Scratch, coding, computer programming, functions

|  |
| --- |
| **9. Additional Resources:** |

|  |
| --- |
| **10. Pre-Unit and Post-Unit Assessment Instruments:** |

|  |  |
| --- | --- |
| **11. Poster** | **12. Video (Link here.)** |

**If you are a science teacher, check the boxes below that apply:**

| **Next Generation Science Standards (NGSS)** | |
| --- | --- |
| **Science and Engineering Practices (Check all that apply)** | **Crosscutting Concepts (Check all that apply)** |
| x Asking questions (for science) and defining problems (for engineering) | x Patterns |
| x Developing and using models | x Cause and effect |
| x Planning and carrying out investigations | ☐ Scale, proportion, and quantity |
| ☐ Analyzing and interpreting data | ☐ Systems and system models |
| ☐ Using mathematics and computational thinking | ☐ Energy and matter: Flows, cycles, and conservation |
| x Constructing explanations (for science) and designing solutions (for engineering) | x Structure and function. |
| ☐ Engaging in argument from evidence | ☐ Stability and change. |
| x Obtaining, evaluating, and communicating information |  |

**If you are a science teacher, check the boxes below that apply:**

| **Ohio’s Learning Standards for Science (OLS)** |
| --- |
| **Expectations for Learning - Cognitive Demands (Check all that apply)** |
| x Designing Technological/Engineering Solutions Using Science concepts **(T)** |
| ☐ Demonstrating Science Knowledge **(D)** |
| x Interpreting and Communicating Science Concepts **(C)** |
| ☐ Recalling Accurate Science **(R)** |

**If you are a math teacher, check the boxes below that apply:**

| **Ohio’s Learning Standards for Math (OLS) or**  **Common Core State Standards -- Mathematics (CCSS)** | |
| --- | --- |
| **Standards for Mathematical Practice (Check all that apply)**  x | |
| x Make sense of problems and persevere in solving them  x | ☐ Useappropriate tools strategically |
| ☐ Reason abstractly and quantitatively | ☐ Attendto precision  x |
| ☐ Construct viable arguments and critique the reasoning of others | ☐ Look for and make use of structure |
| ☐ Model with mathematics | ☐ Look for and express regularity in repeated reasoning |

**Part 2: Post Implementation- Reflection on the Unit**

|  |
| --- |
| **Results: Evidence of Growth in Student Learning -** After the Unit has been taught and the Post-Unit Assessment Instrument has been used to assess student growth in learning, the teacher must analyze the data and determine whether or not student growth in learning occurred. Present all documents used to collect and organize Post- Unit evaluation data such as graphs or charts. Provide a written analysis in sentence or paragraph form which provides the evidence that student growth in learning took place. Please present results and, if applicable, student work (as a hyperlink) used as evidence after the Unit has been taught.  **Please include**:   * Any documents used to collect and organize post unit evaluation data. (charts, graphs and /or tables etc.) * An analysis of data used to measure growth in student learning providing evidence that student learning occurred. (Sentence or paragraph form.) * Other forms of assessment that demonstrate evidence of learning. * Anecdotal information from student feedback. |

|  |
| --- |
| **Reflection: Reflections: Reflect upon the successes of teaching in this Unit in 5 or more sentences in the form of a narrative. Consider the following questions:**   1. Why did you select this content for the Unit? 2. Was the purpose for selecting the Unit met? If yes, provide student learning related evidence. If not, provide possible reasons. 3. Did the students find a solution or solutions that resulted in concrete meaningful action for the Unit’s Challenge? Hyperlink examples of student solutions as evidence. 4. What does the data indicate about growth in student learning? 5. What would you change if you re-taught this Unit? 6. Would you teach this Unit again? Why or why not? |

**16. APPENDIX V: UNIT TEMPLATE FOR RYAN WRIGHT’S NATURAL SELECTION UNIT**

|  |  |  |
| --- | --- | --- |
| **Name: Ryan Wright** | **Contact Info: (859)638-9809**  **Ryan.wright@boone.kyschools.us** | **Date: 6/21/17** |

|  |
| --- |
| **Unit Number and Title: Unit 6: Natural Selection – Nature’s Optimization Method** |

|  |  |
| --- | --- |
| **Grade Level:** | 9th-10th |

|  |  |
| --- | --- |
| **Subject Area:** | Biology |

|  |  |
| --- | --- |
| **Total Estimated Duration of Entire Unit:** | 2-3 weeks |

**Part 1: Designing the Unit**

|  |
| --- |
| 1. **Unit Academic Standards (**Identify which standards:NGSS, OLS and/or CCSS.Cut and paste from NGSS, OLS and/or CCSS and be sure to include letter and/or number identifiers.**):** |

**NGSS HS-LS4-2, HS-LS4-3, HS-LS4-4, HS-LS4-5**

|  |
| --- |
| 1. **Unit Summary** |

The Big Idea (including global relevance):

*The big idea of this unit is Natural Selection. NS is the force that shapes all life on this planet – where we are going, and where we come from. As our environment changes over time, natural selection will decide which organisms survive and which will perish; with natural environments changing now more than ever, this is becoming even more important.*

The (anticipated) Essential Questions: List 3 or more questions your students are likely to generate on their own. (Highlight in yellow the one selected to define the Challenge):

*How do organisms evolve (to react to their environments)?*

*Why are some living things better at surviving than others?*

*What determines if an adaptation will be passed to others or not?*

|  |
| --- |
| 1. **Unit Context** |

Justification for Selection of Content– Check all that apply:

☐ Students previously scored poorly on standardized tests, end-of term test or any other test given in the school or district on this content.

☐ Misconceptions regarding this content are prevalent.

☐ Content is suited well for teaching via CBL and EDP pedagogies.

☐ The selected content follows the pacing guide for when this content is scheduled to be taught during the school year. (Unit 1 covers atomic structure because it is taught in October when I should be conducting my first unit.)

☐ Other reason(s) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

The Hook: (Describe in a few sentences how you will use a “hook” to introduce the Big Idea in a compelling way that draws students into the topic.)

*In the past, I have started my natural selection with a simple/fun activity I call the “Caveman Games”. This activity consists of several 10-12 small mini-activity that students compete in. Some of the activities are more athletic in nature, while others are centered around logical skills, or even artistic skills. Students are given a simple task (for example, “jump as high as you can and see if you can reach the banana hanging from the tree”). Students will either fail the event or succeed. They must then think about what traits they might have that helped make them more or less successful at this event. This gets students to start thinking about things like fitness and adaptations, as well as survival of the fittest (ie, natural selection), and it also reinforces that being fit is more than just being \*physically\* fit. This then leads into a discussion about what makes some lifeforms on Earth better at surviving than others, and about how these advantages come to be and/or are propagated.*

*This is usually a pretty good hook in that it grabs students attentions from the get go – they love playing the games, and I usually dress up and do voices as we work through the “Caveman Games”. This gets them engaged and thinking about what I want them to think about.*

The Challenge and Constraints:

☐ Product **or** ☐ Process (Check one)

|  |  |
| --- | --- |
| Description of Challenge (Either Product or Process is clearly explained below): | List the Constraints Applied |
| **Create a system (flow chart or algorithm) that shows the process of how nature optimizes life through natural selection.** | **Time – approximately 1 week.**  **System must have \_\_\_ number of steps.**  **Death must only appear in one spot in the system and be the last possible outcome.**  **System must address certain conditions stressed by the teacher. Examples include:**   * **Food Procurement** * **Climate Tolerance** * **Reproductive Strategies** |

Teacher’s Anticipated Guiding Questions (that apply to the Challenge and may change with student input.):

*What specific challenges do organisms face in their environments?*

*How long does it take for life to change in response to its environment?*

*What options do organisms that are less fit have to try and survive?*

*How quickly does an adaptation spread over time?*

*What causes these changes to occur in the first place?*

*How is fitness measured?*

|  |
| --- |
| **4. EDP: Use the diagram below to help you complete this section.** |

****

How will students test or implement the solution? What is the evidence that the solution worked? Describe how the iterative process from the EDP applies to your Challenge.

*After the first iteration of their system is complete, students will be given an unknown condition – either a brand new organism, or a change to the environment where there system takes place. They will then need to see if their new system accommodates the new condition that was given to them. If not, they will have the chance to add to or refine their system to make sure that it accounts for more than just what they were originally given in the project.*

How will students present or defend the solution? Describe if any formal training or resource guides will be provided to the students for best practices (e.g., poster, flyer, video, advertisement, etc.) used to present work.

*Students will present their systems in the form of a poster and power point. They will then have to present their systems to the class, walking us through the process they created for their organisms.*

What academic content is being taught through this Challenge?

*Natural Selection, populations, Darwin’s principles, ecology, competitive exclusion*

Assessment and EDP:

Using the diagram above, identify any places in the EDP where assessments should take place, as it applies to your Challenge. Describe below what kinds of assessment are most appropriate.

|  |  |
| --- | --- |
| What EDP Processes are ideal for conducting an Assessment? (List ones that apply.) | List the type of Assessment (Rubric, Diagram, Checklist, Model, Q/A etc.) Check box to indicate whether it is formative or summative. |
| \_\_Gather Information\_\_\_  \_\_Evaluate Solution \_\_\_  \_Communicate Solution\_  \_\_\_ALL STEPS\_\_\_\_\_\_\_\_ | \_\_Information Table \_\_­­\_\_\_\_\_\_\_\_\_\_\_\_\_ ☐ formative ☐ summative  \_\_Rubric\_\_\_\_\_\_\_\_\_\_\_\_­­\_\_\_\_\_\_\_\_\_\_\_\_\_ ☐ formative ☐ summative  \_\_Poster Presentation\_­­\_\_\_\_\_\_\_\_\_\_\_\_\_ ☐ formative ☐ summative  \_\_Engineering Journal – PowerPoint\_\_\_ ☐ formative ☐ summative |

Check below which characteristic(s) of this Challenge will be incorporated in its implementation using EDP. (Check all that apply.)

☐ Has clear constraints that limit the solutions

☐ Will produce more than one possible solution that works

☐ Includes the ability to refine or optimize solutions

☐ Assesses science or math content

☐ Includes Math applications

☐ Involves use of graphs

☐ Requires analysis of data

☐ Includes student led communication of findings

|  |
| --- |
| **5. ACS (Real world applications; career connections; societal impact):** |

Place an X on the continuum to indicate where this Challenge belongs in the context of real world applications:

|  |  |  |
| --- | --- | --- |
| **Abstract or Loosely Applies to the Real World** | **|--------------------------------------|---------------------------------------|** | **Strongly Applies to the Real World** |

Provide a brief rationale for where you placed the X**:­­­­­­­ I feel like while this unit is a bit more on the abstract side, there are several real world connections to be made. Students are being asked to use real world programming skills, but they’re applying it to something that they probably wouldn’t normally apply it to. This activity will also easily tie in to global change and the way it stresses organisms, and will also connect to examples of natural selection that greatly influence student’s lives, such as the increase in antibiotic resistance in bacteria due to overuse.**

What activities in this Unit apply to real world context? **Creating their flow chart or algorithm will have kids using real world programming and optimization based skills. We are applying organization to real world problems in this unit.**

Place an X on the continuum to indicate where this Challenge belongs in the context of societal impact:

|  |  |  |
| --- | --- | --- |
| **Shows Little or No Societal Impact** | **|-------------------------------------|----------------------------------------|** | **Strongly Shows Societal Impact** |

Provide a brief rationale for where you placed the X**: ­­­­­­­­­­­­­­I am going to make a real attempt to connect this to changes in biomes being caused by human impact and climate change, both of which have real societal implications.**

What activities in this Unit apply to societal impact? **Discussions on climate change and human caused changes to Earth’s biomes.**

Careers: What careers will you introduce (and how) to the students that are related to the Challenge? (Examples: career research assignment, guest speakers, fieldtrips, Skype with a professional, etc.)

**Zoologists, biologist, and environmental activists would all be great speakers to bring in as a part of this unit.**

|  |
| --- |
| **6. Misconceptions:** |

*Students have a tendency to think of evolution on a much smaller scale than it actually occurs; in other words, they think it occurs at an individual level as opposed to a population level, and think that it takes place within 1-2 generations as opposed to several.*

*We will also need to dress the fact that many scientists believe that due to advances in technology and health care, humans are not really affected by natural selection anymore. Even less fit individuals are typically able to survive with proper care.*

*Students often think that the evolution of a new species results in the replacement of the original. The flow charts they make will help them visualize how this is not the case, but be sure to stress that the loops on these charts represents different species “going their own way”.*

|  |
| --- |
| **7. Unit Lessons and Activities: (**Provide a tentative timeline with a breakdown for Lessons 1 and 2. Provide the Lesson #’s and Activity #’s for when the Challenge Based Learning (CBL) and Engineering Design Process (EDP) are embedded in the unit.) |

*Lesson 1: Fitness and Adaptations – this lesson focuses on teaching the concepts of fitness, adaptations, and natural selection, and emphasizes the connection between the 3. Students work on 2 main activities designed to show them the effects that adaptations have on fitness, then work to unravel how nature may have shaped them to be the way that they are.*

*Activity 1 (Day 1): “Caveman Games” activity*

*Activity 1 (Day 2): Introduce Big Idea, formulate Essential Questions, and come up with Challenge*

*Ideas*

*Activity 2 (Day 3): Bird Beak Lab, Day 1*

*Activity 2 (Day 4): Bird Beak Lab, Day 2*

*Lesson 2: Optimization Systems – this lesson focuses on the process of optimization and connects it to natural selection. Students learn how computers and businesses analyze a problem through computational/logical thinking, and attempt to connect that to what happens for life, first through their own expriences and then through what happens in the wild.*

*Activity 3 (Day 5): Present challenge, formulate guiding questions, begin flow chart activity*

*Activity 3 (Day 6): Finish flow chart activity, share with class*

*Activity 4 (Day 7): Go over challenge instructions, receive group and organism assignments, go over*

*one example (Darwin’s Finches) as a class to demonstrate what students will be doing.*

*Activity 4 (Day 8): Research organisms, complete table, formulate potential if/then statements for*

*separating organisms.*

*Activity 4 (Day 9): Choose if/then statements from yesterday and begin creating rough draft*

*Activity 4 (Day 10): Finish draft, check-in with teacher, receive unknown condition and evaluate*

*system’s ability to handle it. Refine and improve system to do so.*

*Activity 4 (Day 11): Create final poster, finish PowerPoint*

*Activity 4 (Day 12): Present poster and PowerPoint to class.*

*CBL integrated: Day 2, Day 5, Days 7-12*

*EDP integrated: Days 7-12*

|  |
| --- |
| **8. Keywords:** |

Fitness, adaptations, natural selection, logical thinking, algorithm, if/then statements, selecting factor, niche, population

|  |
| --- |
| **9. Additional Resources:** |

Other unit materials:

* Caveman Games Powerpoint
* Caveman Games Grid WS
* Bird Beak Lab Packet
* Flow Chart Activity WS
* Optimizing Life Project Instructions
* Project Rubric
* Organism Research Table WS
* Pre/Post Unit Quiz

Copies of all of these materials can be found at: <https://sites.google.com/site/ryanawright2017/>

|  |
| --- |
| **10. Pre-Unit and Post-Unit Assessment Instruments:** |

Use the “Pre/Post Unit Quiz” attached in the Additional Resources section of this template to assess student growth. Give the quiz at the start of the unit and at the end, then compare results to see if the unit accomplished the goals of the unit. An open response question could be added at the end of the unit to further assess student understanding.

|  |  |
| --- | --- |
| **11. Poster** | **12. Video (Link here.)** |

Link: [Click Here](https://docs.google.com/viewer?a=v&pid=sites&srcid=ZGVmYXVsdGRvbWFpbnxyeWFuYXdyaWdodDIwMTd8Z3g6MWFkZjUwYTkzYTg3MzEzMQ) Link: [Click Here](https://www.youtube.com/watch?v=20pvnr7Dnso&t=3s)

**If you are a science teacher, check the boxes below that apply:**

| **Next Generation Science Standards (NGSS)** | |
| --- | --- |
| **Science and Engineering Practices (Check all that apply)** | **Crosscutting Concepts (Check all that apply)** |
| ☐ Asking questions (for science) and defining problems (for engineering) | ☐ Patterns |
| ☐ Developing and using models | ☐ Cause and effect |
| ☐ Planning and carrying out investigations | ☐ Scale, proportion, and quantity |
| ☐ Analyzing and interpreting data | ☐ Systems and system models |
| ☐ Using mathematics and computational thinking | ☐ Energy and matter: Flows, cycles, and conservation |
| ☐ Constructing explanations (for science) and designing solutions (for engineering) | ☐ Structure and function. |
| ☐ Engaging in argument from evidence | ☐ Stability and change. |
| ☐ Obtaining, evaluating, and communicating information |  |

**If you are a science teacher, check the boxes below that apply:**

| **Ohio’s Learning Standards for Science (OLS)** |
| --- |
| **Expectations for Learning - Cognitive Demands (Check all that apply)** |
| ☐ Designing Technological/Engineering Solutions Using Science concepts **(T)** |
| ☐ Demonstrating Science Knowledge **(D)** |
| ☐ Interpreting and Communicating Science Concepts **(C)** |
| ☐ Recalling Accurate Science **(R)** |

**If you are a math teacher, check the boxes below that apply:**

| **Ohio’s Learning Standards for Math (OLS) or**  **Common Core State Standards -- Mathematics (CCSS)** | |
| --- | --- |
| **Standards for Mathematical Practice (Check all that apply)** | |
| ☐ Make sense of problems and persevere in solving them | ☐ Useappropriate tools strategically |
| ☐ Reason abstractly and quantitatively | ☐ Attendto precision |
| ☐ Construct viable arguments and critique the reasoning of others | ☐ Look for and make use of structure |
| ☐ Model with mathematics | ☐ Look for and express regularity in repeated reasoning |

**Part 2: Post Implementation- Reflection on the Unit**

|  |
| --- |
| **Results: Evidence of Growth in Student Learning -** After the Unit has been taught and the Post-Unit Assessment Instrument has been used to assess student growth in learning, the teacher must analyze the data and determine whether or not student growth in learning occurred. Present all documents used to collect and organize Post- Unit evaluation data such as graphs or charts. Provide a written analysis in sentence or paragraph form which provides the evidence that student growth in learning took place. Please present results and, if applicable, student work (as a hyperlink) used as evidence after the Unit has been taught.  **Please include**:   * Any documents used to collect and organize post unit evaluation data. (charts, graphs and /or tables etc.) * An analysis of data used to measure growth in student learning providing evidence that student learning occurred. (Sentence or paragraph form.) * Other forms of assessment that demonstrate evidence of learning. * Anecdotal information from student feedback. |

|  |
| --- |
| **Reflection: Reflections: Reflect upon the successes of teaching in this Unit in 5 or more sentences in the form of a narrative. Consider the following questions:**   1. Why did you select this content for the Unit? 2. Was the purpose for selecting the Unit met? If yes, provide student learning related evidence. If not, provide possible reasons. 3. Did the students find a solution or solutions that resulted in concrete meaningful action for the Unit’s Challenge? Hyperlink examples of student solutions as evidence. 4. What does the data indicate about growth in student learning? 5. What would you change if you re-taught this Unit? 6. Would you teach this Unit again? Why or why not? |