THE TRAVELING SALESMAN PROBLEM

by

Corinne Brucato

B.A., Sonoma State University, 2010

M.S., University of Pittsburgh, 2013

Submitted to the Graduate Faculty of
the Department of Mathematics in partial fulfillment
of the requirements for the degree of
Master of Sciences

University of Pittsburgh

2013

UNIVERSITY OF PITTSBURGH

MATHEMATICS DEPARTMENT

This thesis was presented

by

Corinne Brucato

It was defended on

April 16, 2013

and approved by

Dr. Jeffrey Paul Wheeler, University of Pittsburgh, Mathematics

Dr. Beverly Michael, University of Pittsburgh, Mathematics

Dr. Catalin Trenchea, University of Pittsburgh, Mathematics

Dr. Anna Vainchtein, University of Pittsburgh, Mathematics

Thesis Advisor: Dr. Jeffrey Paul Wheeler, University of Pittsburgh, Mathematics

THE TRAVELING SALESMAN PROBLEM

Corinne Brucato, M.S.

University of Pittsburgh, 2013

Although a global solution for the Traveling Salesman Problem does not yet exist, there are algorithms for an existing local solution. There are also necessary and sufficient conditions to determine if a possible solution does exist when one is not given a complete graph. This paper gives an introduction to the Traveling Salesman Problem that includes current research. Additionally, the algorithms are used to find a route traveling through twenty US colleges. As well, we use the Geometric Algorithm to assign scouts for the Pittsburgh Pirates.

TABLE OF CONTENTS

LIST OF FIGURES

ACKNOWLEDGEMENTS

A great number of individuals are owed a debt of gratitude for their contribution to the academic success that I have attained. The following are but a few of those individuals.

First and foremost, I would like to thank Dr. Jeffrey Wheeler of the University of Pittsburgh because absolutely none of this would have been possible without him. Dr. Wheeler walked up to me one day and asked about my future plans as a graduate student, and when he realized that I had no idea he said,"You are going to write a thesis, and I am going to be your advisor." I am so grateful for all of the time that he created out of nowhere in order to be a great advisor. Thank you "all day e'ry day."

Thank you Dr. Beverly Michael, Dr. Catalin Trenchea, and Dr. Anna Vainchtein for going above and beyond the duties of serving on my committee. Thank you for all of the comments on this thesis, and for making my days at the University of Pittsburgh a true pleasure.

I would like to thank my parents for always believing in me, no matter what my dreams are at any given moment. You have shown me what it means to love and have faith, and for that I am grateful.

I would like to thank my many support groups around this country. Specifically in Pittsburgh, I would like to thank Marc, Alex, Nadine, Soheil, and Stuart, all of whom have helped me in countless ways. I am amazed that I get to have all of you in my life and I hope that I can be there for you as you have for me.

My family members around the country all are constantly supporting me. To Lisa, Lenny, Christine, all of my nieces, nephews, and cousins: your love is a blessing. Thank you.

Thank you The Dax, Trevor, Kyle, Adam, Brad, Kerri and Fran for being amazing people in my life that I can always count on although we are far away. I love you all.

Thank you to New Hope Church in Cotati, California. I want you to know that I count you as part of my family. Thank you for being so warm and inviting. A special "thank you" goes out to Fran, for making these relationships possible.

1.0 THE PROBLEM STATED

A traveling salesman wishes to go to a certain number of destinations in order to sell objects. He wants to travel to each destination exactly once and return home taking the shortest total route.

Each voyage can be represented as a graph $G = (V, E)$ where each destination, including his home, is a vertex, and if there is a direct route that connects two distinct destinations then there is an edge between those two vertices. The traveling salesman problem is solved if there exists a shortest route that visits each destination once and permits the salesman to return home. (This route is called a Hamiltonian Cycle and will be explained in Chapter 2.)
The traveling salesman problem can be divided into two types: the problems where there is a path between every pair of distinct vertices (no road blocks), and the ones where there are not (with road blocks). Both of these types of TSP problems are explained in more detail in Chapter 6.

Though we are not all traveling salesman, this problem interests those who want to optimize their routes, either by considering distance, cost, or time. If one has four people in their car to drop off at their respective homes, then one automatically tries to think about the shortest distance possible. In this case, distance is minimized. If one is traveling to different parts of the city using the public transportation system, then minimizing distance might not be the goal, but rather minimizing cost.

In the first case, each vertex would be a person's home, and each edge would be the distance between homes. In the second case, each vertex would be a destination of the city and each edge would be the cost to get from one part of the city to the next. Thus, the Traveling Salesman Problem optimizes routes.

2.0 SOME BASIC GRAPH THEORY

Before presenting algorithms and conditions for when a locally optimal solution exists, some basic graph theory definitions are needed. For an extensive overview of Graph Theory, refer to [1] or [5].

2.1 BASIC DEFINITIONS

The platonic solids in figures 2.1 through 2.7 will be used to explain some basics of Graph Theory.

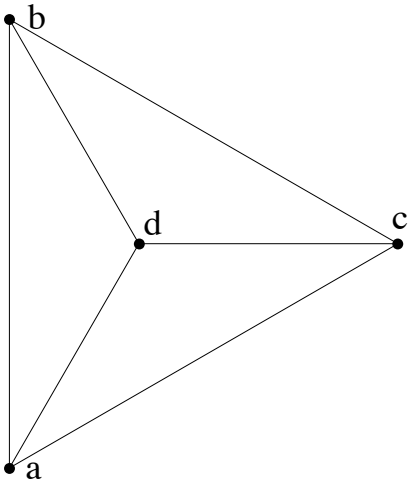

Figure 2.1: Tetrahedral Graph

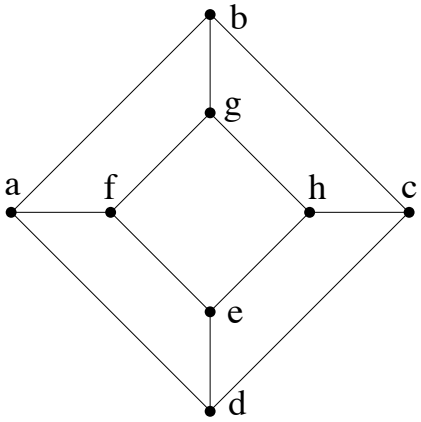

Figure 2.2: Cubical Graph

Definition 1. *[Simple Graph]*

A ***simple graph****,* G = (V,E)*, is a finite nonempty set* V *of objects called vertices (singular vertex) together with a possibly empty set* E *of 2-element subsets of* V *called edges.*

All of the figures in Chapter 2 are examples of simple graphs.

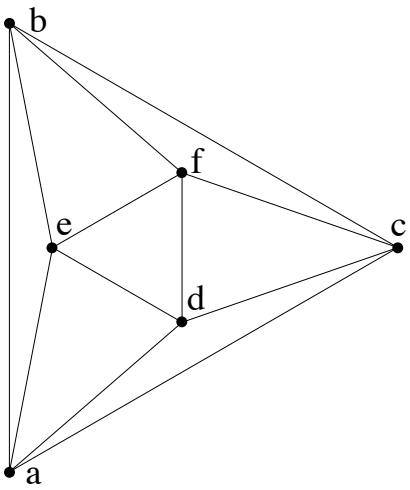

Figure 2.3: Octahedral Graph

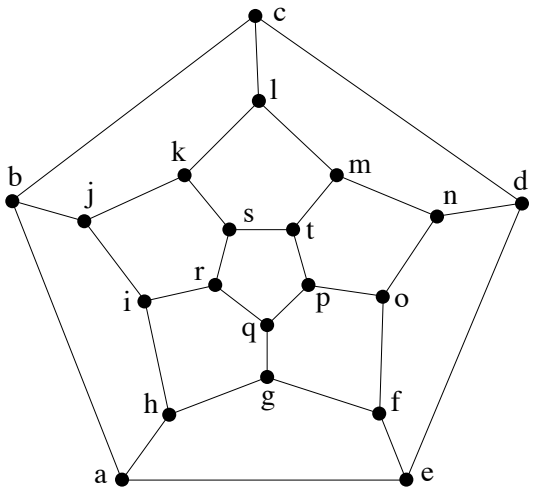

Figure 2.4: Dodecahedral Graph

Definition 2. *[Adjacent]*

Two vertices are ***adjacent*** *if they are connected by an edge.*

Definition 3. *[Incident]*

Vertex u *and the edge* uv *are said to be* ***incident*** *with each other. Similarly,* v *and* uv *are incident.*

Vertices b and c in Figure 2.3 are adjacent, while vertices a and f in the same figure are not. In Figure 2.4, vertex p and edge pq are incident, but vertex p and edge ae are not.

Definition 4. *[Degree of a Vertex]*

The ***degree*** *of a vertex* v (denoted: deg(v)) *is the number of vertices in* G *that are adjacent to* v.

For example, the vertex a in Figure 2.1 has degree three, while the vertex k in Figure 2.5 has degree 5. In a simple graph the maximum degree that any vertex can have is one less than the total number of vertices.

Definition 5. *[Minimum and Maximum Degree of a Graph]*

The ***maximum*** *degree of a graph,* G*, is the greatest degree over all of the vertices in* G*. The* ***minimum*** *degree of a graph is the least degree over all of the vertices in* G*. The maximum degree is denoted* $\Delta(\mathrm{G})$*, while the minimum degree is denoted* $\delta(\mathrm{G})$*.*

For example, let H_1 = Figure 2.5 and H_2 = Figure 2.3. Then, $\Delta(H_1) = 5$, and $\delta(H_1) = 4$, and $\Delta(H_2) = \delta(H_2) = 4$.

Definition 6. *[Order and Size of a Graph]*

The number of vertices in a given graph G*, denoted* $|\mathrm{V(G)}|$*, is called the* ***order*** *of* G *and is denoted* $\mathrm{Ord(G)}$*, or* $|\mathrm{G}|$*. The number of edges in* G*, denoted* $|\mathrm{E(G)}|$*, is called the* ***size*** *of* G*.*

Figures 2.1 through 2.5 have order, 4, 8, 6, 20, and 12, and size 6, 12, 12, 30, and 30, respectively.

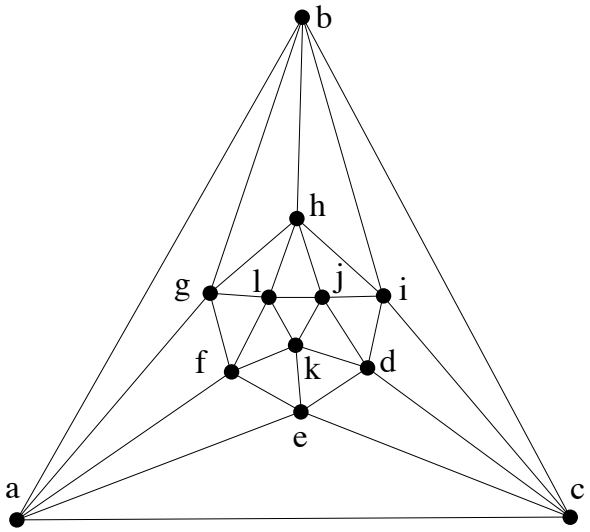

Figure 2.5: Icosahedral Graph

Definition 7. *[Complete Graph]*

A graph in which every pair of distinct vertices is adjacent is called a ***complete graph****. A complete graph of order* n *is denoted* K_n*.*

Up to isomorphism, there is a unique complete graph of order n for each n. Figure 2.6 and Figure 2.7 are the complete graphs of order 2 and 3, respectively. The complete graph of order 4 is represented in Figure 2.1, and the complete graph of order 5 is represented in Figure 2.12.

Definition 8. *[Walk]*

For two, not necessarily distinct, vertices u *and* v *in a graph* G*, a* $\{u \to v\}$ ***walk*** W *in* G *is a sequence of vertices in* G*, beginning with* u *and ending at* v *such that consecutive vertices in* W *are adjacent in* G*.*

In Figure 2.1 an example of a walk from a to d is $\{a \to b \to c \to d\}$, and a walk from d to c could be $\{d \to b \to a \to b \to a \to d \to a \to b \to c\}$. Note that a walk can pass through the same vertex or the same edge more than once.

Definition 9. *[Closed Walk]*

A walk whose initial and terminal vertices are the same is a ***closed walk****.*

For example, Figure 2.2 we have the following closed walks: $\{a \to b \to c \to d \to e \to d \to a\}$, $\{g \to h \to e \to f \to g\}$.

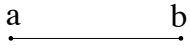

Figure 2.6: K_2

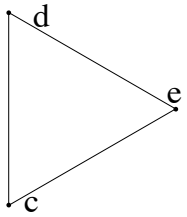

Figure 2.7: K_3

Definition 10. *[Path]*

A walk in a graph G *in which no vertex is repeated is called a* ***path****.*

In Figure 2.3, $\{a \to b \to c \to f\}$ and $\{c \to f \to e \to a \to b\}$ are paths, but $\{a \to b \to e \to b \to c\}$ is not because vertex b is visited more than once. Note that in paths, since no vertex is repeated, it necessarily follows that no edge is repeated.

Definition 11. *[Cycle]*

A walk whose initial and terminal vertices are the same and every other vertex is distinct is called a ***cycle****.*

A cycle can also be thought of as a closed path. For example, in Figure 2.4 we have $\{a \to b \to c \to d \to e \to a\}$ and $\{p \to q \to r \to s \to t \to m \to n \to o \to p\}$.

2.2 ADVANCED DEFINITIONS

Definition 12. *[Hamiltonian Path]*

A path in a given graph G *that contains every vertex of* G *is called a* ***Hamiltonian Path*** *of* G*.*

The following definition will be of special importance to us.

Definition 13. *[Hamiltonian Cycle]*

A cycle in a given graph G *that contains every vertex of* G *is called a* ***Hamiltonian Cycle*** *of* G*.*

In Figure 2.2 $\{e \to f \to g \to h \to c \to d \to a \to b\}$ is an example of a Hamiltonian Path. In Figure 2.5 an example of a Hamiltonian Cycle is $\{a \to b \to c \to e \to d \to k \to f \to l \to j \to i \to h \to g \to a\}$. Note that in Figure 1.8 $\{a \to b \to c \to a\}$ and $\{b \to c \to a \to b\}$ are the same Hamiltonian Cycle.

Definition 14. *[Connected and Disconnected Graphs]*

A graph G *is* ***connected*** *if* G *contains a* $\{u \to v\}$ *walk for every two vertices* u *and* v *of* G*, otherwise* G *is* ***disconnected****.*

Figures 2.1 through 2.8 are all connected graphs, but if we consider Figures 2.6 and 2.7 as one graph, G_1, then G_1 would be a disconnected graph since there does not exist a path from vertex a to vertex d.

Definition 15. *[Component]*

The maximally connected subgraphs of a graph G *are called* ***components*** *of* G*. i.e.* H *is a component of* G *if* H *is not a proper subgraph of any connected subgraph of* G*.*

Definition 16. *[Bridge]*

An edge e = uv *in a connected graph G whose removal results in a disconnected graph is called a* ***bridge****.*

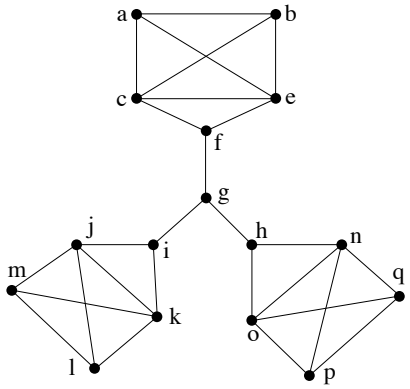

Figure 2.8: Bridge Graph

Figure 2.8 above is a connected graph with three bridges: edges gf, gi, and hi. Figures 2.9 through 2.11 are the disconnected graphs with edges gf, gi, and gh are removed, respectively. Notice, in each case, that once the edge is removed there are two components. For example, in Figure 2.9 one component contains the vertices $\{a, b, c, d, e\}$, and the other contains $\{f, g, h, i, j, k, l, m, n, o, p, q\}$.

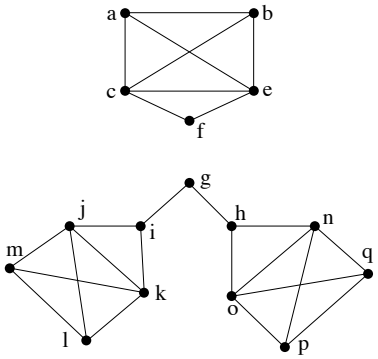

Figure 2.9: Removed edge gf

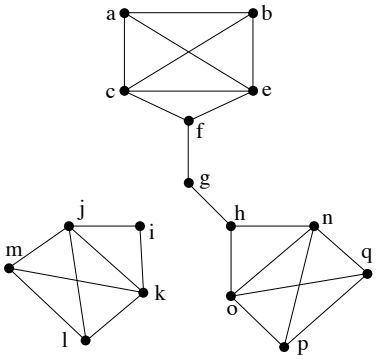

Figure 2.10: Removed edge gi

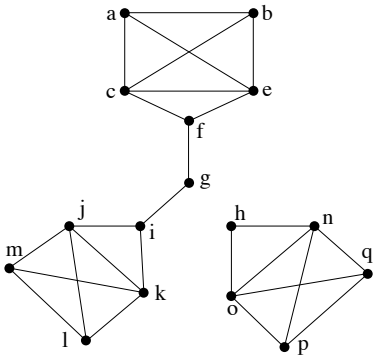

Figure 2.11: Removed edge hi

Definition 17. *[Planar]*

A graph is called ***planar****, provided that it can be represented by a drawing in the plane in such a way that two edges intersect only at vertices. In other words, none of the edges in a planar graph cross over each other.*

Notice that the complete graph of order 5 *(Figure 2.12)* does not have a planar representation. Figure 2.1 through Figure 2.7 are all planar graphs.

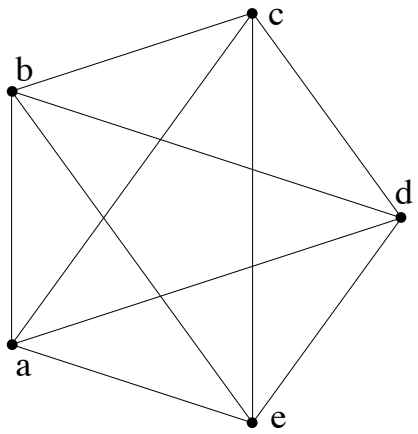

Figure 2.12: K_5

Definition 18. *[Weighted Graph]*

If each edge in a given graph G *is assigned a real number (called the* ***weight*** *of the edge), then* G *is a* ***weighted graph****.*

Definition 19. *[The Join of a Graph]*

The ***join*** $\mathrm{G} = G_1 \wedge G_2$ *of* G_1 *and* G_2 *has vertex set* $V(G) = V(G_1) \cup V(G_2)$ *and edge set*

$$E(G) = E(G_1) \cup E(G_2) \cup \{uv : u \in V(G_1), v \in V(G_2)\}$$

Definition 20. *[Convex Set]*

An object is ***convex*** *if for every pair of points within the object, every point on the line segment that joins them is also within the object.*

Definition 21. *[Convex Hull]*

The intersection of all convex sets containing a set of points, or vertices, is called the ***convex hull.***

Figure 2.13 through 2.16 demonstrate how to create the convex hull given a set of vertices.

Create the complete graph of order n in a graph G, where n is the number of towns in question. Keep all of the outside edges of this graph and destroy all of the other edges.

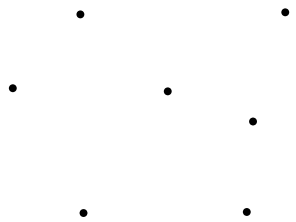

Figure 2.13: Towns

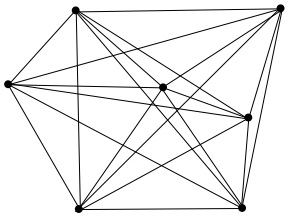

Figure 2.14: Complete Graph

Note that all of the other edges between vertices of the graph are contained inside the convex hull. Another way to imagine the convex hull is if one wraps a rubber band around all of the towns, then pulls the rubber band as tight as possible. This is the convex hull of the set of towns.

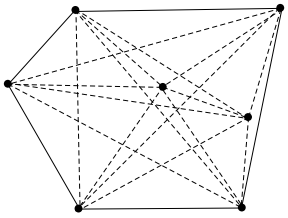

Figure 2.15: Highlight outer edges

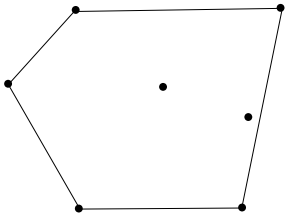

Figure 2.16: Convex Hull of the towns

3.0 THE HISTORY

Although the exact origins of the Traveling Salesman Problem are unclear, the first example of such a problem appeared in the German handbook *Der Handlungsreisende - Von einem alten Commis - Voyageur* for salesman traveling through Germany and Switzerland in 1832 as explained in [3]. This handbook was merely a guide, since it did not contain any mathematical language. People started to realize that the time one could save from creating optimal paths is not to be overlooked, and thus there is an advantage to figuring out how to create such optimal paths.

This idea was turned into a puzzle sometime during the 1800's by W. R. Hamilton and Thomas Kirkman. Hamilton's Icosian Game was a recreational puzzle based on finding a Hamiltonian cycle in the Dodecahedron graph as seen below.

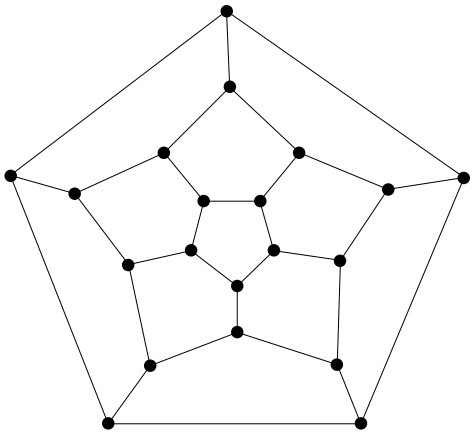

Figure 3.1: Dodecahedron Graph

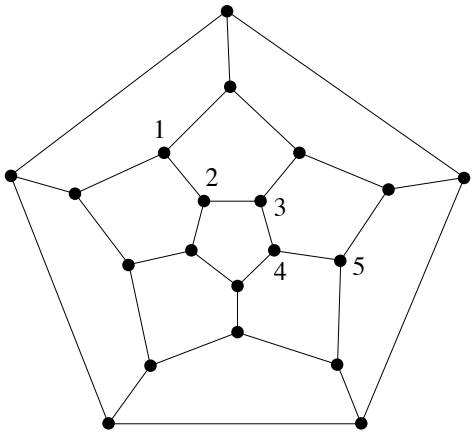

Figure 3.2: Sample Start

One person would put the numbered pegs, 1 through 5, in adjacent vertices on the dodecahedron graph. A second person would use the rest of the numbers, 6 through 20, in adjacent vertices after the vertex with the 5 in order to create a Hamiltonian Cycle that would end at the first peg labeled "1". It is the case that every starting position has a solution. The game was not a big success, for children complained that the

game was too easy. For more information about this and other similar grames, refer to [3].

The Traveling Salesman Problem, as we know and love it, was first studied in the 1930's in Vienna and Harvard as explained in [3]. Richard M. Karp showed in 1972 that the Hamiltonian cycle problem was NP-complete, which implies the NP-hardness of TSP (*see the next section regarding complexity*). This supplied a mathematical explanation for the apparent computational difficulty of finding optimal tours.

The current record for the largest Traveling Salesman Problem including 85,900 cities, was solved in 2006 as explained in [3]. The computers used versions of the branch-and-bound method as well as the cutting planes method (two seemingly elementary integer linear programming methods). The code used in these solutions is called *Concorde* and is available to view for free over the internet.

If one considers every single city in the world, and solve for the shortest Hamiltonian Cycle (hopefully using a computer), then one can win fame, fortune, and a cash prize.

4.0 COMPLEXITY CLASSES

One would like to believe that the The Traveling Salesman Problem can always be solved by a computer. The maximum number of Hamiltonian Cycles in a given graph with three or more vertices is $\frac{(n-1)!}{2}$. Hence, in theory a computer could consider each of these possible tours. The problem here is that as n gets large, $\frac{(n-1)!}{2}$ gets extremely large, as we can see in the table in Figure 4.1.

n	(n-1)!/2
3	1
4	3
5	12
6	60
7	360
8	2,520
9	20,160
10	181,440
20	6.08226E+16
30	4.42088E+30
40	1.01989E+46

Figure 4.1: Brute Force

As these numbers grow larger, it becomes clear that a computer will not be able to solve such problems in a reasonable amount of time. Solving the Traveling Salesman Problem means searching for an algorithm that will make the number of computations acceptable.

The Traveling Salesman Problem can be split into different, perhaps easier to solve, problems. In this chapter, we will explain these different problems, their complexity classes, and their relation to the full Traveling Salesman Problem.

A complexity class is a set of problems of related complexity. The more simple complexity class are defined by the type of computational problem, the model of computation and the resource (or resources) that are

being bounded.

The Turing Machine was an idea considered before the invention of computers. The machine would be shaped like a box with an infinitely long piece of paper running through it. In each iteration, the Turing Machine reads the entry on the paper, follows a previously determined action (either changing an entry or keeping it the same), and then moves the paper either to the left or the right a certain number of times (could not move at all). These three steps continue until the previously determined algorithm is completed.

There are two different machines to consider, a **deterministic** Turing machine, much like the computers that we use every day, and a **non-deterministic** Turing machine. A deterministic turing machine behaves like a function in the sense that every time the Turing Machine gets to a particular step there is one and only one way to proceed. A non-deterministic Turing Machine may choose different paths when presented with the same set-up. When an algorithm can be split into two parts the non-deterministic computer can continue on both paths at the same time.

A problem that be solved by a deterministic Turing machine in polynomial time, which means that the number of steps to solve the problem can be represented as a polynomial function, is in the complexity class **P**.

For example, if you are given a set of n numbers, $\{3, 7, 5, 1\}$, to put in numerical order, the number of steps that it will take is represented as a polynomial of degree at most 2. For more information about this specific problem, as well as others, refer to [6]. The following is a demonstration of how such a problem would reorder the set $\{3, 7, 5, 1\}$ using the algorithm commonly referred to as bubble sort.

This algorithm looks at two numbers at a time and deciphers if they need to be switched or if they can stay in the same order. In the first case, take 3 and 7. 3 is indeed smaller than 7, so these two numbers do not need to switch places. Next, take 7 and 5. 7 is larger than 5, thus these two must be switched and the result is $\{3, 5, 7, 1\}$. Once the program has reached the end of the row, it will go through again and again until the set of numbers are in increasing order. This specific problem continues as depicted below, with the pair of numbers in question in bold.

$$\{\mathbf{3}, \mathbf{7}, 5, 1\} \rightarrow \{\mathbf{3}, \mathbf{7}, 5, 1\}$$
$$\{3, \mathbf{7}, \mathbf{5}, 1\} \rightarrow \{3, \mathbf{5}, \mathbf{7}, 1\}$$
$$\{3, 5, \mathbf{7}, \mathbf{1}\} \rightarrow \{3, 5, \mathbf{1}, \mathbf{7}\}$$
$$\{\mathbf{3}, \mathbf{5}, 1, 7\} \rightarrow \{\mathbf{3}, \mathbf{5}, 1, 7\}$$
$$\{3, \mathbf{5}, \mathbf{1}, 7\} \rightarrow \{3, \mathbf{1}, \mathbf{5}, 7\}$$

$$\{3, 1, \mathbf{5}, \mathbf{7}\} \rightarrow \{3, 1, \mathbf{5}, \mathbf{7}\}$$
$$\{\mathbf{3}, \mathbf{1}, 5, 7\} \rightarrow \{\mathbf{1}, \mathbf{3}, 5, 7\}$$

A decision problem is a problem whose answer is either "yes" or "no." The set of decision problems that be solved by a non-deterministic Turing machine in polynomial time that can be verified by a deterministic Turing machine is in the complexity class **NP**. A version of the Traveling Salesman Problem, referred to as the *decision version*, asks if given a list of cities and the distances between each city, does there exist a route whose distance is less than k, for some given total distance k? This is a "yes" or "no" question, and thus is in the NP complexity class.

Let Q and R be two decision problems. If an algorithm can reduce a solution for Q to a solution for R in polynomial time, it is said that R is poly-time reducible (or just reducible) to Q. A problem is **NP-complete** if it can be proven that it is in the complexity class NP, and shown that it is reducible to a problem in polynomial time that it is already known to be NP-complete. A simplified version of the Traveling Salesman Problem, classified as NP-Complete, asks if there exists a Hamiltonian Cycle or a Hamiltonian Path in a given graph G.

A problem is **NP-hard** if and only if it is at least as hard as an NP-complete problem. The full Traveling Salesman Problem is at least as hard as the simplified version listed above, since it asks to find all possible Hamiltonian cycles and then find the shortest one. Therefore, the Traveling Salesman Problem is classified as *NP-Hard*.

5.0 SOME KNOWN ALGORITHMS

Since a globally optimal solution has not been found, there are many algorithms that give locally optimal solutions. In this chapter, we will introduce three. For a more extensive overview of the algorithms refer to [1], [4], and [5].

5.1 NEAREST-NEIGHBOR ALGORITHM

The Nearest-Neighbor Algorithm is a type of *greedy* algorithm which means that at every stage of the algorithm we pick the best possible edge to use. In other words, we make a locally optimal decision in every step of the algorithm. Another thing to note about this algorithm is that we need to pick a starting vertex and if we choose different vertices we might get different Hamiltonian Cycles, as illustrated in the following examples. We will think of this starting vertex as the Traveling Salesman's Home.

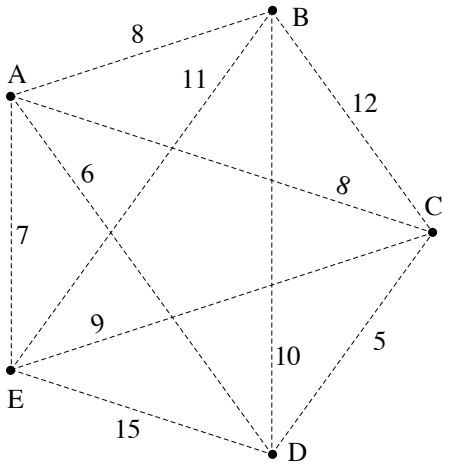

Figure 5.1: Start

We start with a weighted complete graph of order 5 as our example. First, we see what the Nearest-Neighbor

Algorithm gives us if we start with vertex A. In Stage 1 we can see that we have four edges to choose from.

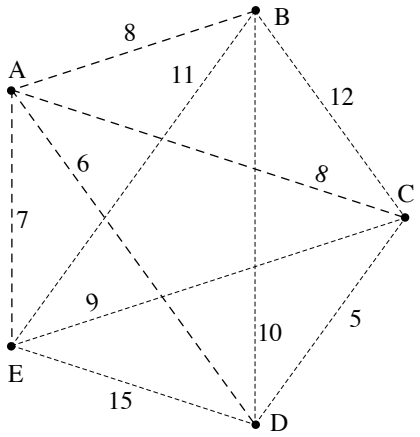

Figure 5.2: Stage 1

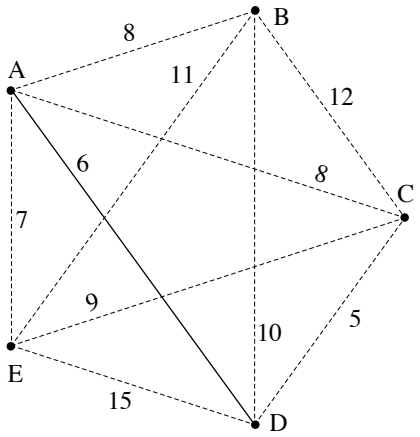

Figure 5.3: Stage 2

Edge AD has the least weight, 6, so we will use that edge, and ignore the others for now. Then, as we can see in Stage 3, we will look at all of the vertices incident with D and again pick the edge with the least weight. In this case, we have edge DC with a weight of 5.

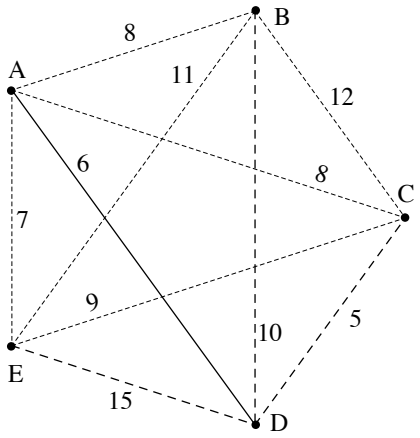

Figure 5.4: Stage 3

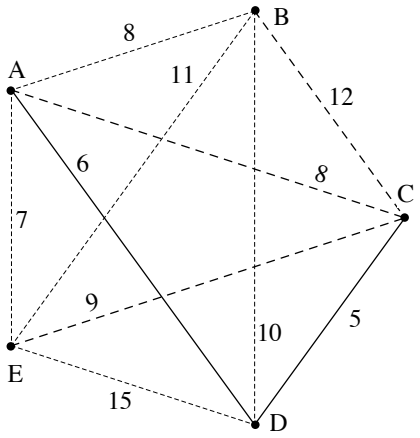

Figure 5.5: Stage 4

In stage 4 we consider the edges incident with vertex C. Note that the edge with the least weight, eight, is edge CA. However, recall that our salesman needs a Hamiltonian Cycle, which means he does not want to travel to any vertex more than once. We ignore edges to vertices that have already been travelled to, until

our salesman can travel home, and choose the edge with the least weight of 9, edge *CE*.

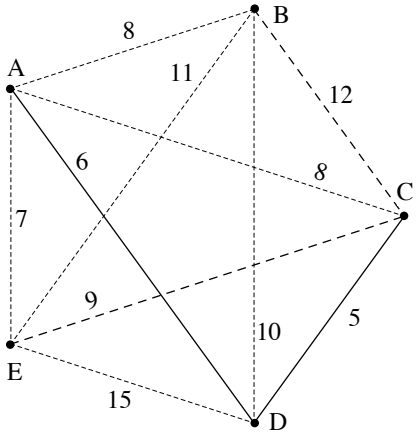

Figure 5.6: Stage 5

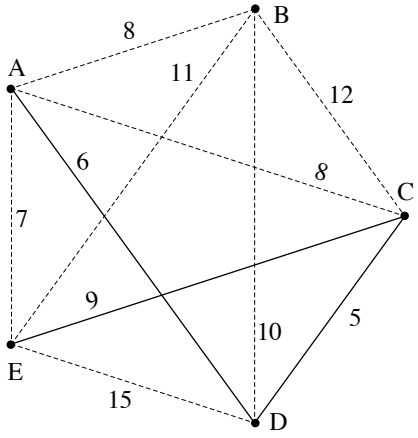

Figure 5.7: Stage 6

At this point we technically have three edges to choose from: *ED*, *EA*, and *EB*. Again, we do not want to travel to vertices that we have already traveled to, unless we are at the end of our Hamiltonian Cycle, and thus we must choose edge *EB*, with a weight of 11. Our last step will be to choose the last edge *BA* that brings our traveling salesman back home. The total trip, when starting at vertex *A*, has a weight of 39.

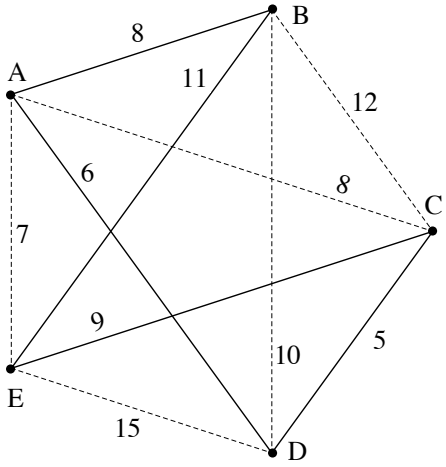

Figure 5.8: Final

We mentioned before that if we start at a different vertex we might get different weights for the same graph. Instead of starting at vertex A, let the traveling salesman start at vertex E.

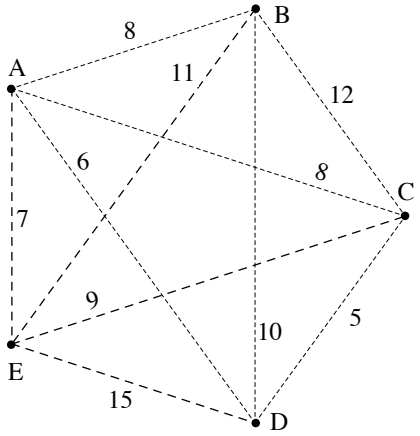

Figure 5.9: Start

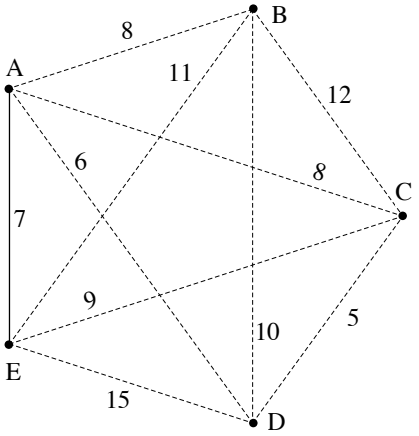

Figure 5.10: Stage 1

First, we choose edge EA with a weight of 7. Then we look at the vertices from vertex A: AB, AC, and AD and choose edge AD with a weight of 6. We continue in the same way that we did before and obtain the graph in figure 5.12 with a total weight or 35, which is a smaller weight than when we started with vertex A.

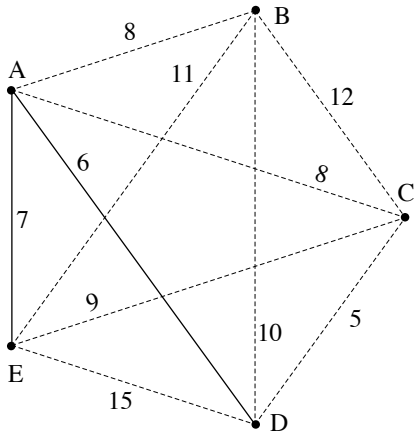

Figure 5.11: Stage 2

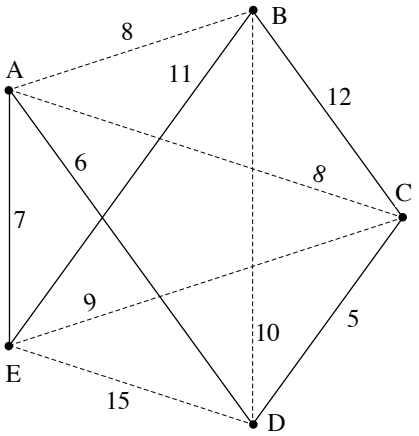

Figure 5.12: Final

Pseudocode

Initialize: Pick a vertex $u_i \in V(G)$

Form a path $P = \{u_i\}$ and put $T = V(P)$.

Put $r = 1$ and $v_r = u_i$

Iteration:

1. Choose $u_j \in V(G) \setminus T$ such that it is the minimum distance from v_r.
   (Ties may be broken arbitrarily.)
2. Put $v_{r+1} = u_j$ and let $P = \{u_i \to u_j\}$
3. Put $r = r + 1$ and repeat until $T = V(G)$

Output: A Hamiltonian Cycle is the path with the last vertex of P made adjacent to the first.

Figure 5.13: Nearest-Neighbor Pseudocode

5.2 CLOSEST INSERTION ALGORITHM

The Closest Insertion algorithm is similar to the Nearest Neighbor Algorithm in the sense that they are both greedy algorithms. We illustrate the Closest Insertion Algorithm with the weighted octahedral graph G below.

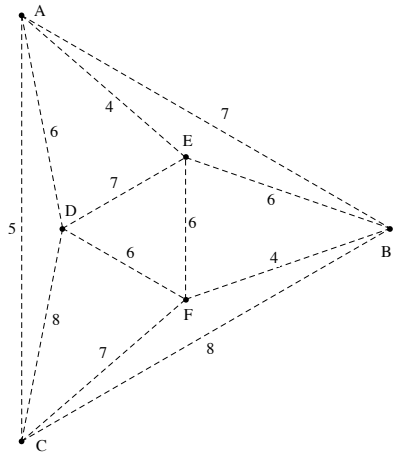

Figure 5.14: Start

We will pick vertex A to be our traveling salesman's home and look at all of the edges incident with vertex A. We collect these edges in a set $T(G)$, such that at stage 1, $T(G) = \{AB, AE, AD, AC\}$. We choose to add edge AE with a weight of 4, since it has the least weight of all of the edges in $T(G)$, and move onto stage 2. In the nearest neighbor algorithm, at this point we would only look at the edges incident with vertex E, but in the Closest Insertion algorithm we will look at the edges that are incident with vertices A and E. Thus, $T(G) = \{AB, AD, AC, EB, EF, ED\}$.

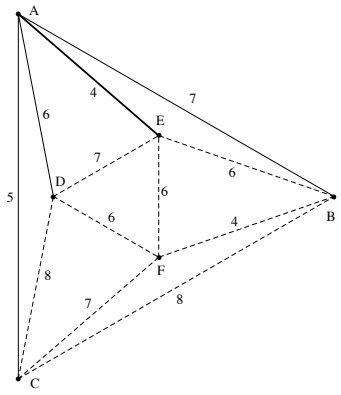

Figure 5.15: Stage 2

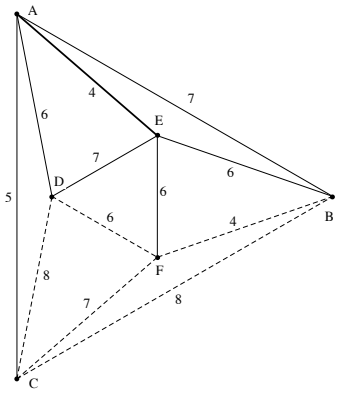

Figure 5.16: Stage 3

We well take the edge with the least weight, which is edge AC. Now we have two edges in our tour as seen in Figure 5.16, and we expand the list of edges again. Note that once a vertex is incident with two edges, we will no longer consider any other edges incident with that vertex, since we do not want to travel to one vertex more than once. If we have two edges incident with each vertex, then we can think of our salesman traveling to that vertex and then away from that vertex.

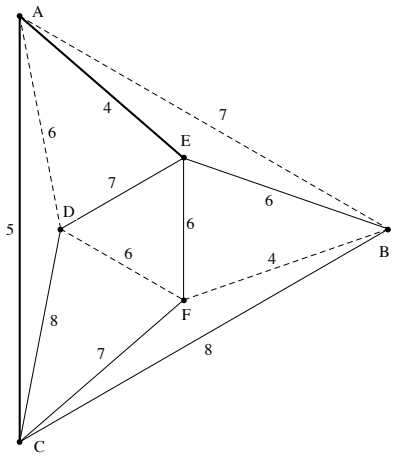

Figure 5.17: Stage 4

Thus, we will remove edges AB, and AD from our vertex set $T(G)$. At the same time we will add the edges incident with our new vertex C. By stage 4, we have that $T(G) = \{EB, EF, ED, CD, CF, CB\}$.

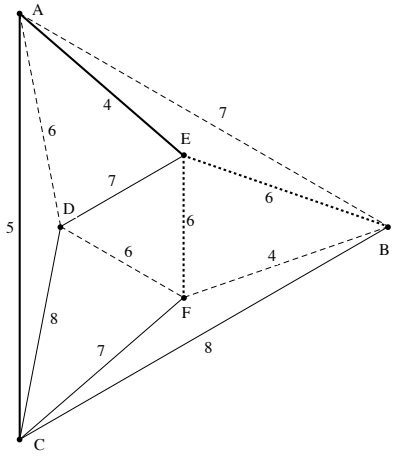

Figure 5.18: Stage 5

Now there is another situation. There are two edges (EB and EF) that have the same weight. In this case, we arbitrarily pick an edge and move on. We shall see what happens in each case. First, we pick edge EB, and at stage 6 we have that $T(G) = \{CD, CF, CB, BF\}$.

If we continue with this algorithm, we next select edge BF, then edge FD, and finally edge CD.

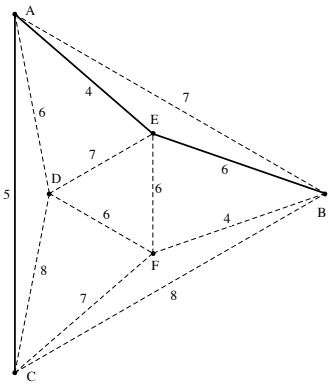

Figure 5.19: Stage 6_a

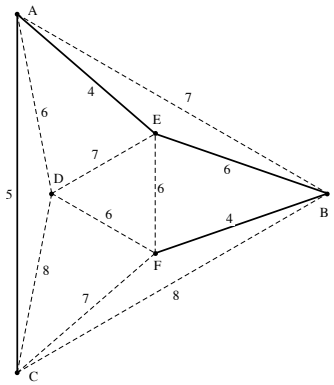

Figure 5.20: Stage 7_a

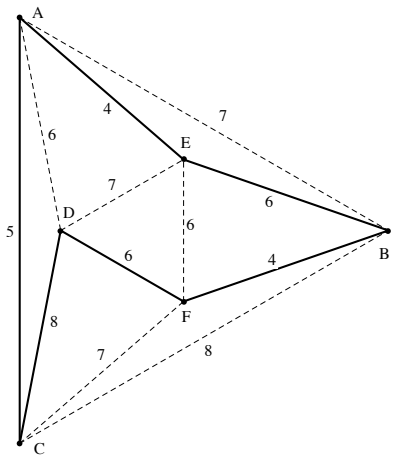

Figure 5.21: Stage 8_a

The final cost for this tour is 33. Note that this is not necessarily the shortest tour, but rather an example of a possible tour. Remember that in stage 5 we made a choice between edges EB and EF.

Next we will see what happens if we had chosen edge EF instead of EB. The algorithm proceeds as follows.

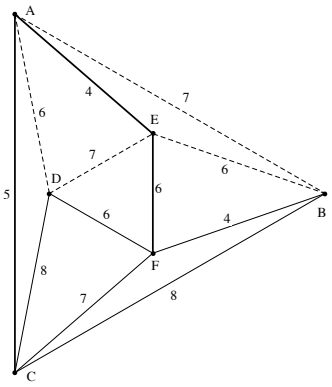

Figure 5.22: Stage 6_b

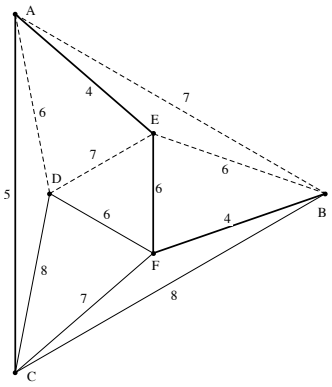

Figure 5.23: Stage 7_b

Now we see that choosing edge EF instead of edge EB has created a problem. Our traveling salesman's tour does not allow him to visit vertex D at all as depicted in Figure 5.23. This brings up another point that we have to consider. The octahedral graph is not a complete graph, and therefore we cannot always find a Hamiltonian Cycle.

In cases such as this, if we had not already found a Hamiltonian Cycle, we would look through the points in the algorithm for which we made a decision between edges and try a different edge. It is important to note that finding a Hamiltonian Cycle is not always possible. We address this issue in Chapter 6 and give some necessary and sufficient conditions for which a Hamiltonian Cycle exists.

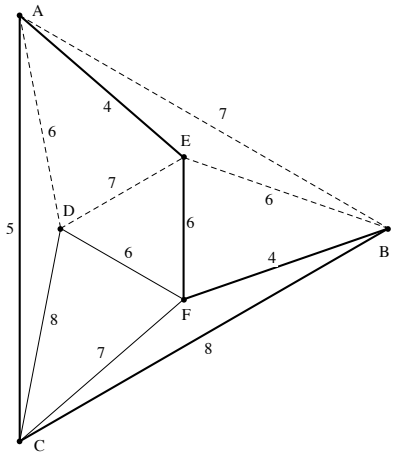

Figure 5.24: Stage 8_b

Pseudocode

Initialize: Pick a vertex $u_i \in V(G)$

Form a path $P = \{u_i\}$ and put $T = V(P)$.

Iteration:

1. Choose $u_j \in V(G) \setminus T$ such that it is the minimum distance from either the start of P or the end of P. (Ties may be broken arbitrarily.)
2. Put the edge with minimum distance incident with u_j in P
3. Repeat until $T = V(G)$

Output: A Hamiltonian Cycle is the path with the last vertex of P made adjacent to the first.

Figure 5.25: Closest Insertion Algorithm Pseudocode

5.3 GEOMETRIC ALGORITHM

For this example we will start with 12 destinations for our traveling salesman. In the Geometric algorithm it is assumed that we have a complete graph. Recall that a complete graph means that each vertex is adjacent to every other vertex, but note that we will not draw each edge in the figures in order to make the algorithm more clear.

The first step is to create the convex hull around all of the destinations, then draw the edges that connect these surrounding vertices as seen in Stage 2.

We call the vertices that are not on the border of the convex hull *towns*. In Figure 5.25 above, we can see that the towns are {F, I, J, K, L}. We take the first town, vertex *I*, and draw the edges from that vertex to every other vertex in the convex hull, as seen in Stage 3.

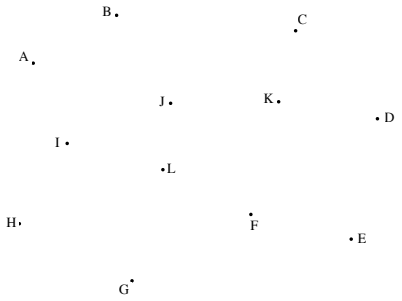

Figure 5.26: Stage 1

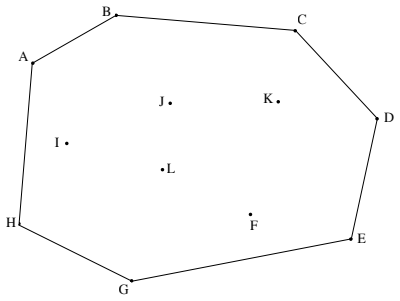

Figure 5.27: Stage 2

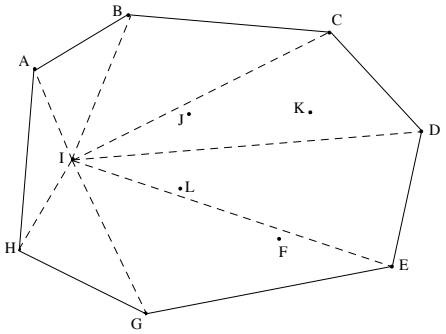

Figure 5.28: Stage 3

Next, we examine all of the angles centered at vertex *I*. We find the largest of these angles, highlight the edges that create that angle, then destroy the edge from the convex hull that is no longer needed as seen in Figure 5.28.

As we destroy the edge that was on the convex hull we also destroy the temporary edges from vertex I to the other vertices as depicted in Stage 6.

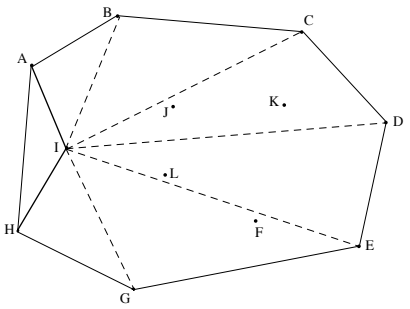

Figure 5.29: Stage 4

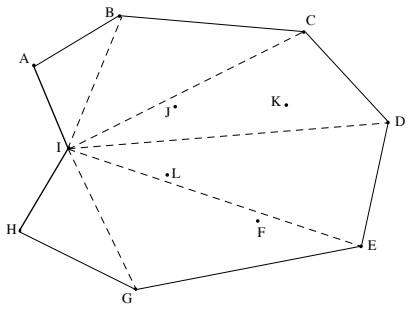

Figure 5.30: Stage 5

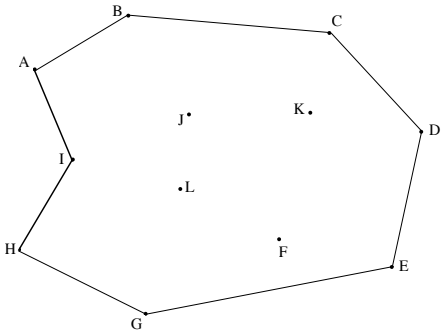

Figure 5.31: Stage 6

Now the border contains the vertices {A, B, C, D, E, G, H, I} and the set of towns is {F, J, K, L}. In the next step we draw all of the edges from town J to the vertices on the border including vertex I. As before, look at the angles created this way, choose largest one, delete the edge on the border, then delete all of the temporary edges incident to J.

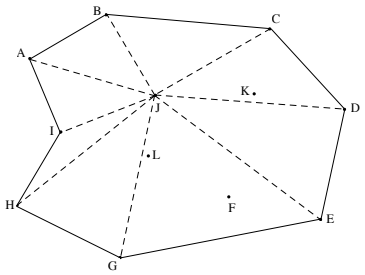

Figure 5.32: Stage 7

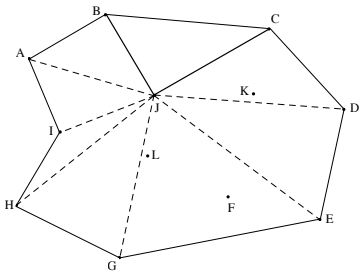

Figure 5.33: Stage 8

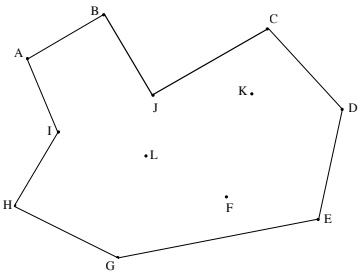

Figure 5.34: Stage 9

We are now left with the graph in Figure 5.32. The algorithm continues in this way until there are no more towns left to put in the tour for our traveling salesman. In our example, we choose to look at vertices *K,* *F*, and *L*, in that order. In the event of a tie between two angles, the programmer is allowed to arbitrarily choose an angle and continue.

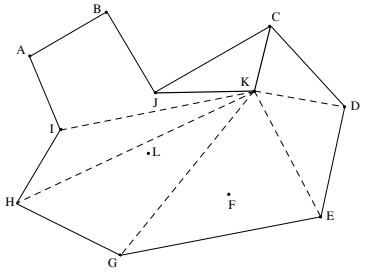

Figure 5.35: Stage 7

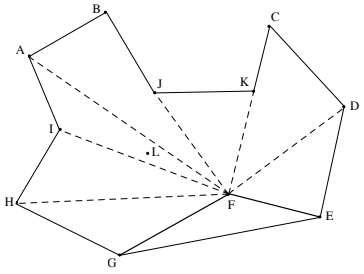

Figure 5.36: Stage 8

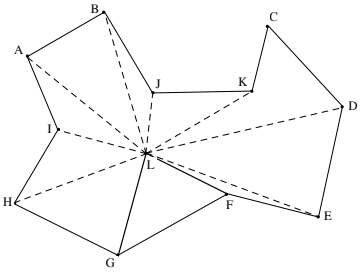

Figure 5.37: Stage 9

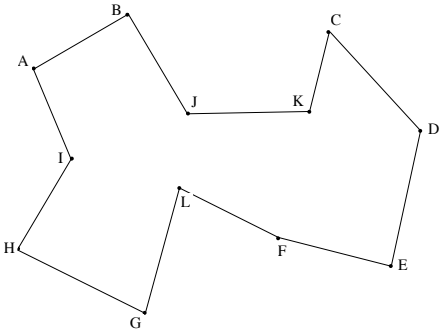

Figure 5.38: Stage 10

Pseudocode

Initialize: Create the convex hull around all of the vertices.

Let A be the set of all vertices that lie on the convex hull.

Let C be the set of edges in the convex hull.

Let T be the set of all vertices that do not lie on the convex hull.

1. Pick the vertex, $v_i \in T$ closest to the convex hull and create all of the edges incident with v_i and the vertices in A.
2. Measure all of the angles created this way. Let v_iu_j and v_iu_{j+1} be the edges that form the largest angle centered at v_i. Keep these edges, and destroy the rest of the edges that are not in C along with the edge u_ju_{j+1}.
3. Now $v_i \in A$.
4. Continue until $T = \emptyset$.

Output: A Hamiltonian Cycle with edges in C.

Figure 5.39: Geometric Algorithm Pseudocode

6.0 EXISTENCE OF HAMILTONIAN PATHS AND CYCLES

Suppose a traveler wants to fly on a Southwest Airlines flight from Pittsburgh, PA to Manchester, NH. The traveler will be disappointed to know that there is not a direct flight. Instead, one must first fly to Baltimore, Maryland in order to eventually reach their destination. We must consider this possibility when searching for routes for the traveling salesman. If such conditions arise, one is no longer guaranteed a Hamiltonian Path nor a Hamiltonian Cycle.

6.1 COMPLETE GRAPHS

If for every pair of towns, there exists a direct road between them, then the voyage can be represented as a complete graph.

Since every destination is connected to every other destination we can always find a Hamiltonian Path, and furthermore, we can determine just how many Hamiltonian Paths there are. There are n choices for the salesman's home, and since there is a route connecting that vertex to every other vertex in the graph there are $(n-1)$ choices for the next destination. Then $(n-2)$ and so on. Thus in a labeled complete graph there are $n!$ Hamiltonian Paths in any voyage without road blocks.

Hamiltonian Cycles can be thought of as complete circular permutations. We no longer worry about choosing a home for your traveling salesman, for each vertex in the Hamiltonian Cycle will have degree 2. One edge incident with each vertex represents arriving at this destination, while the other edge represents leaving that destination, thus each vertex could be thought of as the salesman's home. Once we are at a specific vertex we have $(n-1)$ choices for the next vertex, then $(n-2)$, and so on which will give a total of $(n-1)!$ voyages. However, if there exists a cycle $\{a \to b \to ... \ y \to z \to a\}$, it will be the same as the cycle $\{a \to z \to y \to ... \ b \to a\}$. Thus, it would have been counted twice. Since this will happen with every cycle in the complete graph, there are $\frac{(n-1)!}{2}$ Hamiltonian Cycles in any voyage that can be represented as a complete graph.

6.2 NOT COMPLETE GRAPHS

If the voyage cannot be represented as a complete graph, then there is no longer a guarantee that a Hamiltonian Cycle is possible.

We now discuss known conditions associated with the existence of Hamiltonian Cycles.

Theorem 1. *Bridge Theorem*
If a graph G contains a bridge, then G ***does not*** *contain a Hamiltonian Cycle.*

Proof. Let uv be a bridge in a graph G and call the components of $G-uv$ A and B. Suppose, on the contrary, that G contains a Hamiltonian Cycle, $H(G)$. Then $H(G)$ must contain the edge uv, since it is the only edge that links components A and B. Without loss of generality, if the Hamiltonian Cycle starts on some vertex in component A, then $H(G)$ includes some path through A and the edge uv then into component B. Since edge uv has already been traversed it is impossible to leave component B and return to A. □

Note: If a graph G contains a bridge, then though G has no Hamiltonian cycle, it still may have a Hamiltonian Path.

Property 1. *Ore's Property*
Let G be a graph of order $n \geq 3$. If

$$deg\ u\ +\ deg\ v \geq n$$

for each pair u, v of nonadjacent vertices of G, then G satisfies the Ore Property.

Theorem 2. *Ore's Theorem*
If a graph G satisfies the Ore Property*, then G contains a Hamiltonian Cycle.*

Proof. Assume, on the contrary, that for some integer $n \geq 3$, there exists a graph H of order n such that for each pair of nonadjacent vertices u and v, deg u + deg $v \geq n$ and yet H is not Hamiltonian.
Add as many edges to the graph of H as possible between pair of nonadjacent vertices so that the resulting graph G is still not Hamiltonian. (If we add any other edge, G will be Hamiltonian. In other words, G is a maximal non-Hamiltonian graph.) By the assumption that we made about H, we know that for every pair of nonadjacent vertices u and v in G, $deg_G u + deg_G v \geq n$. Also note that G is not a complete graph.

Pick nonadjacent vertices x, y and add edge xy to the graph G. Then $G + xy$ is necessarily Hamiltonian. Thus, there exists a Hamiltonian Cycle C, and C necessarily contains the edge xy, thus G must contain a Hamiltonian path from x to y, say $(x = v_1, v_2, ..., v_n = y)$.

If $xv_i \in E(G)$, where $2 \leq i \leq (n-1)$, then $yv_{v-1} \notin E(G)$; for otherwise,

$$C' = (x = v_1, v_2, ..., v_{i-1}, y = v_n, v_{n-1}, v_{n-2}, ..., v_i, x)$$

is a Hamiltonian cycle of G, which is impossible. Hence, for each vertex of G adjacent to x, there is a vertex of $V(G) \setminus \{y\}$ not adjacent to to y. However then,

$$deg_G y \leq (n-1) - deg_G x$$
$$deg_G x + deg_G y \leq (n-1)$$

which is a contradiction. □

Note that Ore's Theorem is not a necessary condition for a Hamiltonian Cycle, but a sufficient condition. Figure 6.1 below does not satisfy Ore's Property since the sum of the degrees of every pair of nonadjacent vertices is 4, which is less than 6. However, the graph still contains a Hamiltonian Cycle, namely $\{A \to C \to F \to E \to B \to D \to A\}$.

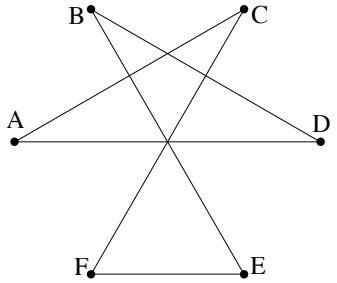

Figure 6.1: Each Degree is 2

A consequence of Ore's Theorem is Dirac's Theorem.

Theorem 3. *Dirac's Theorem*
If G is a graph of order $n \geq 3$ such that

$$deg\ v \geq \frac{n}{2}$$

for each vertex v of G, then G is Hamiltonian.

Proof.

By our assumption, we have that for every vertex v

$$\begin{aligned} deg\ v &\geq \frac{n}{2} \\ deg\ v + deg\ v &\geq \frac{n}{2} + \frac{n}{2} \\ 2\ (deg\ v) &\geq n \\ deg\ v &\geq \frac{n}{2} \end{aligned}$$

Since v is not adjacent to itself, by Ore's Theorem, G contains a Hamiltonian Cycle. □

Lemma 1.

If n-1 ≥ 2k, then

$$2k\ (k\ -\ n\ +\ 1)\ +\ (n^2 - n + 1) \geq \tfrac{n^2+1}{2}$$

Proof.

$$\begin{aligned} 4k(k-n+1) + (n-1)^2 &= 4k^2 - 4k(n-1) + (n-1)(n-1) \\ &= 4k^2 + (n-1)[(n-1) - 4k] \\ &\geq 4k^2 + 2k[2k - 4k] \\ &= 4k^2 + 2k[-2k] \\ &= 0 \\ 4k(k-n+1) + (n-1)^2 &\geq 0 \\ 4k(k-n+1) + (n-1)^2 + n^2 + 1 &\geq n^2 + 1 \\ 4k(k-n+1) + n^2 - 2n + 1 + n^2 + 1 &\geq n^2 + 1 \\ 4k(k-n+1) + 2(n^2 - n + 1) &\geq n^2 + 1 \\ 2k(k-n+1) + (n^2 - n + 1) &\geq \frac{n^2+1}{2} \end{aligned}$$

□

Theorem 4.

Let G be a graph of order $n \geq 3$ and size m. If

$$deg\ u\ +\ deg\ v \geq n$$

for each pair u, v of nonadjacent vertices of G, then $m \geq \frac{n^2}{4}$.

Proof. From Dirac's Theorem above, we can assume that the minimum degree of G is k where $k > \frac{n}{2}$. Thus, we have that $k < \frac{(n-1)}{2}$ and $(n-1) \geq 2k$.

Let S be the set of vertices in G with degree k. Since the sum of the degrees of every pair of nonadjacent vertices in at least n, it follows that $G[S]$ is a complete subgraph of G.

G is Hamiltonian, by Ore's Theorem, and thus is connected. Thus, some vertex in S is adjacent to a vertex not in S. Hence, $|S| \geq k$. Let $u \in S$. Thus, $deg_G u = k$. Therefore, u is adjacent to $k-1$ vertices having degree k or more and adjacent to a vertex of degree $k+1$ or more. Let w be a vertex distinct from u that is not adjacent to u. Then

$$\deg u + \deg w \geq n$$
$$\deg w \geq n - k$$

Therefore,

$$\begin{aligned} 2m &= \sum_{v \in V(G)} degv = degu + \sum_{v \in N(u)} degv + \sum_{w \notin N[u]} degw \\ &\geq k + [(k-1)k + (k+1)] + (n-k-1)(n-k) \\ &= 2k(k-n+1) + (n^2 - n + 1) \\ &\geq \frac{n^2+1}{2} \qquad (6.1) \\ 2m &\geq \frac{n^2+1}{2} \\ m &\geq \frac{n^2+1}{4} \\ &\geq \frac{n^2}{4} \end{aligned}$$

Equation (6.1) is satisfied by Lemma 1 above.

□

Corollary 1.

Let G be a graph of order $n \geq 2$. If

$$deg\ u\ +\ deg\ v \geq (n-1)$$

for each pair u, v of nonadjacent vertices of G, then G contains a Hamiltonian path.

Proof. (Found in [1]) Let $H = G \wedge K_1$ be the join of G and K_1, where w is the vertex of H that does not belong to G. Then

$$\deg u + \deg v \geq (n+1)$$

for each pair u, v of nonadjacent vertices of H. Since the order of H is $(n+1)$, it follows by Ore's Theorem that H is Hamiltonian. Let C be a Hamiltonian cycle of H. Deleting w from C produces a Hamiltonian path in G. □

7.0 APPLICATIONS

7.1 COLLEGES

Although one may not be a traveling salesman, there are other ways that one can use this information. For example, a student wants to drive to twenty colleges around the United States in the shortest way possible. Since the student is driving, he or she wants to minimize the distances between each of the schools. Figure 7.1 shows the twenty schools that the student wants to visit, and Figure 7.2 shows the distance between each of the colleges in alphabetical order.

Figure 7.1: US Colleges (© 2013 Google, © 2013 Tele Atlas)

In this chapter we will use the three algorithms described in Chapter 5 to obtain the Hamiltonian Cycle for the student. The student in this situation could start anywhere, so let's look at Figure 7.2 and find the shortest distance between two colleges and pick one of those to start the journey. The distance between Yale University and Brown University is 103 miles by car, thus we will start at Brown University and head towards Yale University.

Colleges	Arizona State	Brigham Young	Brown	Colorado	Duke	Florida State	Louisiana	Louisville	Michigan	New Mexico	North Dakota	Notre Dame	Ohio	Oklahoma	Oregon	Pitt	Stanford	Texas A&M	Wisconsin	Yale
Arizona State	0	648	2625	549	2185	1898	1458	1752	1963	427	1743	1817	1899	1060	1148	2084	732	1095	1725	2524
Brigham Young	648	0	2363	481	2129	2030	1641	1594	1638	557	1214	1492	1710	1126	825	1861	811	1195	1375	2262
Brown	2625	2362	0	1965	669	1274	1541	920	744	2172	1623	875	720	1595	3085	543	3113	1734	1111	103
Colorado	549	481	1965	0	1667	1605	1194	1132	1242	431	963	1096	1280	664	1249	1464	1276	799	979	1866
Duke	2185	2129	669	1667	0	621	906	541	643	1733	1504	733	459	1187	2880	479	2791	1169	932	566
Florida State	1898	2030	1274	1605	621	0	443	662	978	1482	1669	925	839	1007	2855	929	2541	843	1107	1172
Louisiana	1458	1641	1541	1194	906	443	0	754	1106	1074	1447	976	968	638	2477	1148	2132	435	1027	1442
Louisville	1752	1594	920	1132	541	662	754	0	347	1293	1015	261	209	724	2319	389	2346	841	443	818
Michigan	1963	1638	744	1242	643	978	1106	347	0	1511	961	170	183	970	2361	287	2389	1124	388	688
New Mexico	427	557	2172	431	1733	1482	1074	1293	1511	0	1318	1363	1447	585	1378	1631	1063	641	1275	2071
North Dakota	1743	1214	1623	963	1504	1669	1447	1015	961	1318	0	813	1078	918	1571	1182	1882	1147	582	1583
Notre Dame	1817	1492	875	1096	733	925	976	261	170	1363	813	0	271	826	2215	373	2242	979	241	774
Ohio	1899	1710	720	1280	459	839	968	209	183	1447	1078	271	0	882	2453	192	2408	1049	504	621
Oklahoma	1060	1126	1595	664	1187	1007	638	724	970	585	918	826	882	0	1891	1066	1667	251	798	1495
Oregon	1148	825	3085	1249	2880	2855	2477	2319	2361	1378	1571	2215	2453	1891	0	2583	559	2042	2108	2984
Pitt	2084	1861	543	1464	479	929	1148	389	287	1631	1182	373	192	1066	2583	0	2612	1220	611	442
Stanford	732	811	3113	1276	2791	2541	2132	2346	2389	1063	1882	2242	2408	1667	559	2612	0	1701	2125	3012
Texas A&M	1095	1195	1734	799	1169	843	435	841	1124	641	1147	979	1049	251	2042	1220	1701	0	977	1634
Wisconsin	1725	1375	1111	979	932	1107	1027	443	388	1275	582	241	504	798	2108	611	2125	977	0	1011
Yale	2524	2262	103	1866	566	1172	1442	818	688	2071	1583	774	621	1495	2984	442	3012	1634	1011	0

Figure 7.2: Driving Distances Between Each College (© 2013 Google, © 2013 Tele Atlas)

7.1.1 Nearest Neighbor and Closest Insertion

We follow the Nearest-Neighbor algorithm in the first case. After traveling from Brown University to Yale University, looking at the column under Yale University in Figure 7.2 above, the next smallest distance is to the University of Pittsburgh with a total driving distance of 442 miles. We add this destination to the route then look at the next closest college on our list remembering not to travel to a college that we have already visited.

With the Closest Insertion Algorithm, one does not just look at the distance from that college, but also the distances at the beginning of the route. At the point where Brown University and Yale University are in the route, we look at both of these columns in the table above. The minimum distance from Yale to another college, the University of Pittsburgh, is 442 miles as we mentioned before which is shorter than the distance between Brown University and the University of Pittsburgh. The next smallest distance between Brown University and another college on the list is 720 miles to Ohio State University.

The Closest Insertion Algorithm will continue with the same route as the Nearest Neighbor Algorithm until the distance between the last college put into the route and the next college is greater than 720 miles, or

until Ohio State University is added to the route in another way. If and when Ohio State is added to the route, we look at the next smallest distance between Brown University and any other college on the list.

Figure 7.3 expresses the route that the student would travel to each of the twenty colleges using both the Nearest Neighbor Algorithm and the Closest Insertion Algorithm. Both of the algorithms seem to start off well, in the sense that every distance is below 1000 miles, so the student does not have to travel for too long before they get to the next destination. However, once the student arrives in Oregon, the situation changes. The next destination is 1571 miles away to the University of North Dakota. Then, the trip back to Brown University is 1623 miles which no amount of Taylor Swift CDs will make the 25 hour trip go any faster.

Order	College	Distance
1	Brown University	**-**
2	Yale University	**103**
3	University of Pittsburgh	**442**
4	Ohio State University	**192**
5	University of Michigan	**183**
6	University of Notre Dame	**170**
7	University of Wisconsin	**241**
8	University of Louisville	**443**
9	Duke University	**541**
10	Florida State University	**621**
11	Louisiana State University	**443**
12	Texas A&M University	**435**
13	University of Oklahoma	**251**
14	University of New Mexico	**585**
15	Arizona State University	**427**
16	University of Colorado	**549**
17	Brigham Young University	**481**
18	Stanford University	**811**
19	University of Oregon	**559**
20	University of North Dakota	**1571**
	Brown University	**1623**
TOTAL DISTANCE		**10671**

Figure 7.3: Nearest Neighbor and Closest Insertion Algorithms on Colleges

Figure 7.4 shows the route that the student would take on the map (if the roads were a straight line, of course). If the student wants to map their route using graph theory in order to show his or her parents, he or she might draw this map.

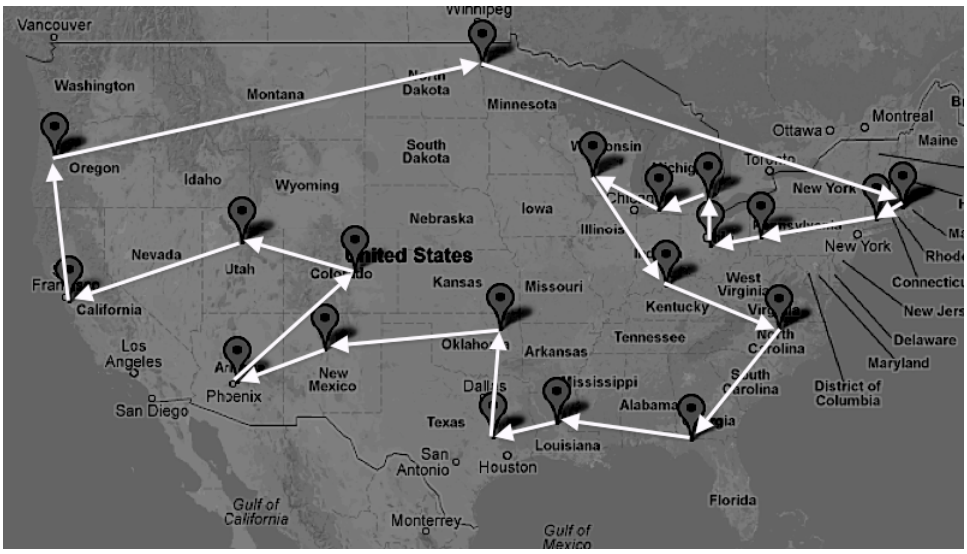

Figure 7.4: Hamiltonian Cycle Using the Nearest Neighbor and Closest Insertion Algorithm (© 2013 Google, © 2013 Tele Atlas)

7.1.2 Geometric Algorithm

We have just seen that if the student chooses to use the Nearest Neighbor algorithm or the Closest Insertion algorithm, they will get the same route. If the student uses the Geometric algorithm, however, the journey is different and actually shorter.

Recall that the geometric algorithm starts with the convex hull of all of the points as seen in Figure 8.6 below.

Figure 7.5: Stage 1

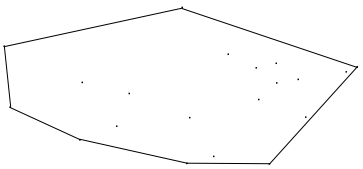

Figure 7.6: Stage 2

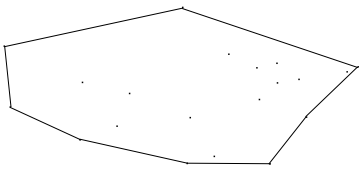

Figure 7.7: Stage 3

We can see the progression of the algorithm in the proceeding figures below. In each step, the college closest to the edge set at the beginning of each step is the town that is selected next in the algorithm.

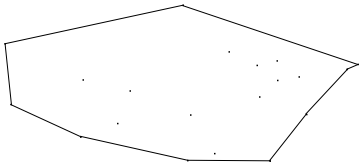

Figure 7.8: Stage 4

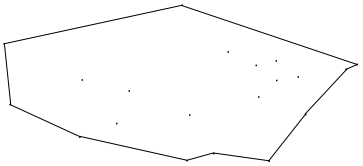

Figure 7.9: Stage 5

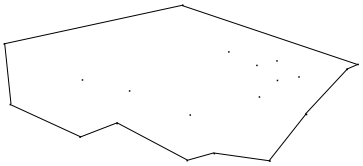

Figure 7.10: Stage 6

At the start of the algorithm it seems as if the total distance will be greater than the distance traveled using the Nearest Neighbor algorithm or the Closest Insertion algorithm. However, look at the list of the distances as a whole. While the Nearest Neighbor and Closest Insertion algorithms seem to have taken the student across the United States in an effective manner , the journey back home was quite awful. The Geometric Algorithm takes the journey, as a whole, into consideration. This is further justified as we look at the chart of distances for the algorithm. Notice that not one of the distances exceeds 1000 miles. This will make both the student and his or her parents very happy.

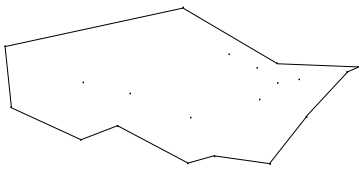

Figure 7.11: Stage 7

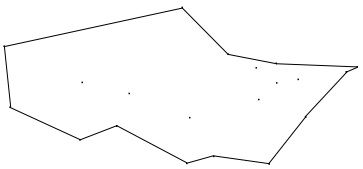

Figure 7.12: Stage 8

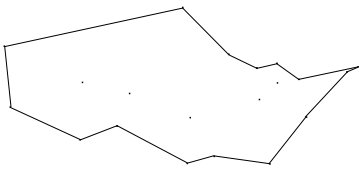

Figure 7.13: Stage 9

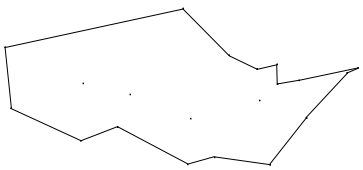

Figure 7.14: Stage 10

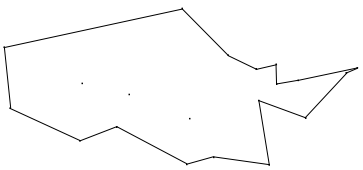

Figure 7.15: Stage 11

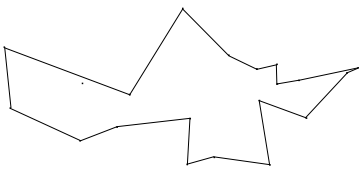

Figure 7.16: Stage 12

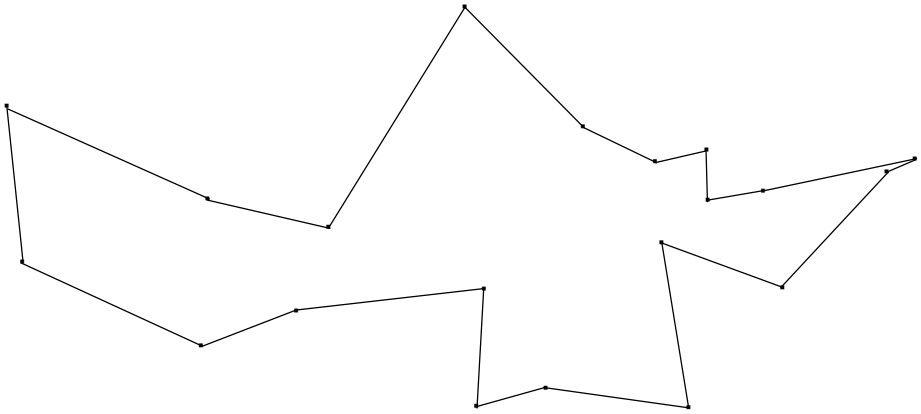

Figure 7.17: Final

Order	College	Distance
1	Brown University	**-**
2	Yale University	**103**
3	Duke University	**566**
4	University of Louisville	**541**
5	Florida State University	**662**
6	Louisiana State University	**443**
7	Texas A&M University	**435**
8	University of Oklahoma	**251**
9	University of New Mexico	**585**
10	Arizona State University	**427**
11	Stanford University	**732**
12	University of Oregon	**559**
13	Brigham Young University	**825**
14	University of Colorado	**481**
15	University of North Dakota	**963**
16	University of Wisconsin	**582**
17	University of Notre Dame	**241**
18	University of Michigan	**170**
19	Ohio State University	**183**
20	University of Pittsburgh	**192**
	Brown University	**543**
TOTAL DISTANCE		**9484**

Figure 7.18: Geometric Algorithm on Colleges

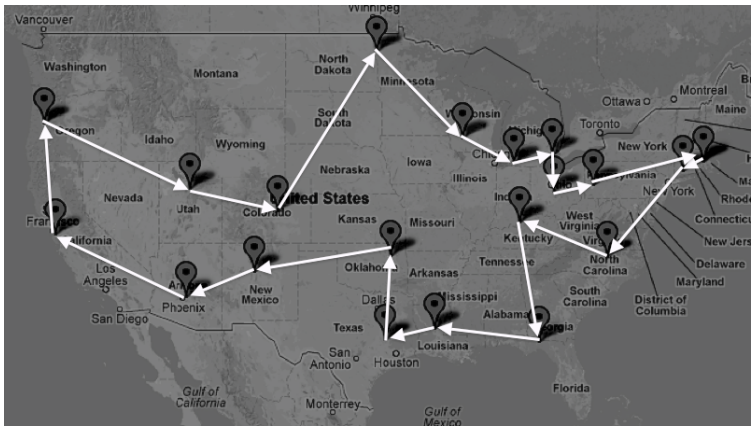

Figure 7.19: Hamiltonian Cycle Using the Geometric Algorithm (© 2013 Google, © 2013 Tele Atlas)

The Nearest-Neighbor and the Closest Insertion Algorithms are both examples of greedy algorithms as previously stated. In that sense, they are blind to the fact that the student in this scenario is looking for the minimum distance through all of these colleges, including the trip home. The Geometric Algorithm takes the entire journey into consideration at the same time. Traveling across the United States went well for both algorithms, however it was the trip home that was unbearable. When traveling across the country, one should have more than one stop. The Geometric Algorithm, on the other hand, takes the convex hull of the destinations and shrinks it until all of the destinations have been added to the journey. The algorithm takes the whole trip into consideration. For this reason, we have found that the Geometric Algorithm gives a shorter journey for the student. Therefore, we will only use the Geometric Algorithm for the following problem.

7.2 PITTSBURGH PIRATES

The following data was provided by the Pittsburgh Pirates of major league baseball.

Every year the Pittsburgh Pirates organization sends scouts all over the United States. Last year twelve groups of scouts traveled to 325 cities. Figure 7.20 is a map of all of the cities in question. (The full list of cities appear in Figures 7.21 through 7.32.) We will use the Geometric Algorithm to find a Hamiltonian

Cycle for the first of the twelve groups of scouts. This will provide a journey through each destination that will minimize their traveling distance. Since there are 12 groups of scouts, we will say that the average number of cities that each group will travel to will be 27.

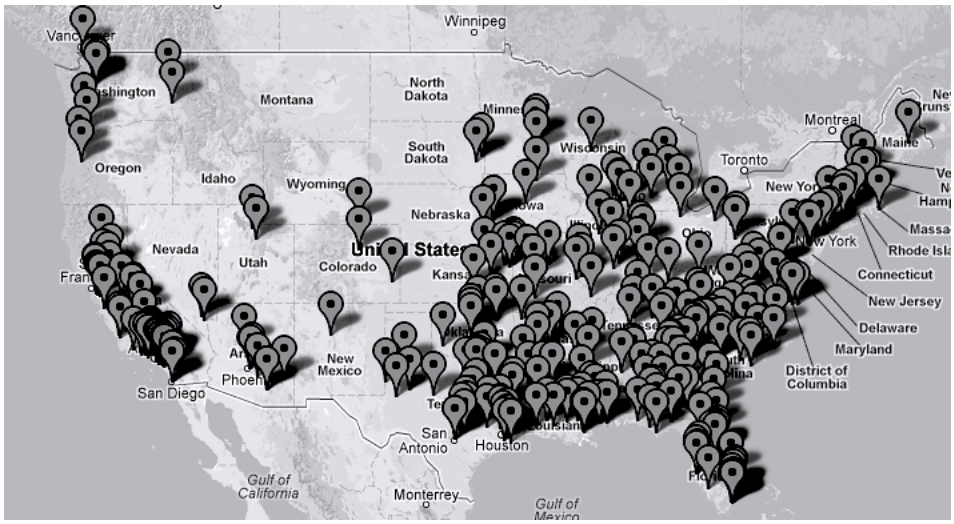

Figure 7.20: NAME (© 2013 Google, © 2013 Tele Atlas)

The first task is to group these 325 cities into 12 groups for each of the scouts as seen in the following figures.

Chapel Hill, NC
Cullowhee, NC
Davidson, NC
Elizabethtown, NC
Elon, NC
Greenville, NC
Holly Ridge, NC
Holly Springs, NC
Lexington, NC
Raleigh, NC
Southern Pines, NC
Blythewood, SC
Columbia, SC
Columbia, SC
Conway, SC
Due West, SC
Duncan, SC
Gaffney, SC
Lake City, SC
Murrells Inlet, SC
Rock Hill, SC
Roebuck, SC
Spartanburg, SC
Alexandria, VA
Hampton, VA
Newport News, VA
Norfolk, VA
Virginia Beach, VA

Figure 7.21: Group 1

Alhambra, CA
Arroyo Grande, CA
Bakersfield, CA
Bellflower, CA
Canoga Park, CA
Charleston, CA
Compton, CA
Costa Mesa, CA
Cypress, CA
Fullerton, CA
Huntington Beach, CA
La Mirada, CA
Long Beach, CA
Los Angeles, CA
Malibu, CA
Newbury Park, CA
Newhall, CA
North Hollywood, CA
Northridge, CA
Oxnard, CA
Salinas, CA
San Luis Obispo, CA
Santa Barbara, CA
Santa Clarita, CA
Stevenson Ranch, CA
Stockton, CA
Valencia, CA

Figure 7.22: Group 2

Ladner, BC
Berkeley, CA
Chico, CA
Clovis, CA
Davis, CA
Elk Grove, CA
Fresno, CA
Merced, CA
Mountain View, CA
Oakland, CA
Rocklin, CA
San Jose, CA
Santa Clara, CA
Stanford, CA
Visalia, CA
Honolulu, HI
Beaverton, OR
Corvallis, OR
Eugene, OR
Bellevue, WA
Longview, WA
Lynnwood, WA
Mill Creek, WA
Pullman, WA
Redmond, WA
Shoreline, WA
Spokane, WA

Figure 7.23: Group 3

Boca Raton, FL
Boynton Beach, FL
Bradenton, FL
Buford, FL
Coral Gables, FL
Deltona, FL
Doral, FL
Ft Myers, FL
Ft Pierce, FL
Gainesville, FL
Jacksonville, FL
Lantana, FL
Maitland, FL
Marianna, FL
Miami Shores, FL
Miami, FL
Montverde, FL
Niceville, FL
Orlando, FL
Oveido, FL
Palmetto, FL
Panama City, FL
Plantation, FL
Port St Joe, FL
Sanford, FL
St Augustine, FL
St Petersburg, FL
Tallahassee, FL
Tampa, FL

Figure 7.24: Group 4

Arlington, TX
Austin, TX
Big Spring, TX
Canton, TX
Dallas, TX
Denison, TX
Dripping Springs, TX
Ft Worth, TX
Grand Prairie, TX
Hurst, TX
Irving, TX
Klein, TX
Leander, TX
Lubbock, TX
Odessa, TX
San Antonio TX
San Antonio, TX
San Marcos, TX
Sherman, TX
Southlake, TX
Temple, TX
The Colony, TX
Waco, TX
Weatherford, TX
West, TX

Figure 7.25: Group 5

New Haven, CT
Southington, CT
Storrs, CT
Washington, DC
Groton, MA
Lowell, MA
Worchester, MA
Orono, ME
Hanover, NH
Rindge, NH
Tilton, NH
East Brunswick, NJ
Tabernacle, NJ
Brooklyn, NY
Jamaica, NY
Poughkeepsie, NY
Lancaster, PA
McMurray, PA
Philadelphia, PA
Pittsburgh, PA
Villanova, PA
Blacksburg, VA
Charlottesville, VA
Harrisonburg, VA
Lexington, VA
Lynchburg, VA
Richmond, VA
Huntington, WV

Figure 7.26: Group 6

Fayetteville, AR
Jessieville, AR
Little Rock, AR
Searcy, AR
State University, AR
Texarkana, AR
Emporia, KS
Gardner, KS
Lawrence, KS
Manhattan, KS
Overland Park, KS
Wichita, KS
Cape Girardeau, MO
Columbia, MO
Florissant, MO
Hillsboro, MO
Kansas City, MO
Riverside, MO
Springfield, MO
Warrensburg, MO
Warsaw, MO
Altus, OK
Broken Arrow, OK
Edmond, OK
Norman, OK
Oklahoma City, OK
Owasso, OK
Stillwater, OK

Figure 7.27: Group 7

Boone, IA
Mason City, IA
Champaign, IL
Evanston, IL
Lake Villa, IL
Normal, IL
Olney, IL
Robinson, IL
Rock Falls, IL
Waterloo, IL
Winnetka, IL
Bloomington, IN
Indianapolis, IN
Valparaiso, IN
Zionsville, IN
Allendale, MI
Ann Arbor, MI
East Lansing, MI
Kalamazoo, MI
Mt Pleasant, MI
Circle Pines, MN
Minneapolis, MN
Northfield, MN
Lincoln, NE
Omaha, NE
Bowling Green, OH
Kent, OH
Brookings, SD
Madison, SD
Stevens Point, WI

Figure 7.28: Group 8

Coolidge, AZ
Pheonix, AZ
Prescott, AZ
Scottsdale, AZ
Tempe, AZ
Thatcher, AZ
Tucson, AZ
Corona, CA
El Cajon, CA
Irvine, CA
Laguna Beach, CA
Rancho Cucamonga, CA
Riverside, CA
San Bernadino, CA
San Diego, CA
San Juan Capistrano, CA
San Marcos, CA
Tustin, CA
Aurora, CO
Lamar, CO
Hobbs, NM
Rio Rancho, NM
Henderson, NV
Las Vegas, NV
Provo, UT
Salt Lake City, UT
Cheyenne, WY

Figure 7.29: Group 9

Auburn, AL
Birmingham, AL
Dothan, AL
Hoover, AL
Slocomb, AL
Troy, AL
Tuscaloosa, AL
Wetumpka, AL
Athens, GA
Atlanta, GA
Brunswick, GA
Cochran, GA
Columbus, GA
Cuthbert, GA
Decatur, GA
Locust Grove, GA
Macon, GA
Moultrie, GA
Nashville, GA
Peachtree City, GA
Richmond Hill, GA
Statesboro, GA
Winder, GA
Athens, TN
Cleveland, TN
Johnson City, TN
Kingsport, TN
Knoxville, TN

Figure 7.30: Group 10

Bowling Green, KY
Lexington, KY
Louisville, KY
Biloxi, MS
Clarksdale, MS
Hattiesburg, MS
Jackson, MS
Lucedale, MS
Mississippi State, MS
Pascagoula, MS
Sumrall, MS
University, MS
Wheeler, MS
Winona, MS
Anderson, SC
Clemson, SC
Easley, SC
Columbia, TN
Cookeville, TN
Nashville, TN

Figure 7.31: Group 11

Baton Rouge, LA
Belle Chasse, LA
Bossier City, LA
Corsicana, LA
Eunice, LA
Franklinton, LA
Grambling, LA
Hammond, LA
Lafayette, LA
Lake Charles, LA
Marrero, LA
Monroe, LA
Natchitoches, LA
New Orleans, LA
Ruston, LA
Slidell, LA
Zachary, LA
Alvin, TX
Brenham, TX
College Station, TX
Crosby, TX
Galveston, TX
Houston, TX
Huntsville, TX
League City, TX
Lufkin, TX
Nacogdoches, TX
The Woodlands, TX

Figure 7.32: Group 12

Next, for each group we plot each of the destinations on the same graph and follow through with the Geometric Algorithm as we have done previously. The algorithm for Group 1 expressed in Figure 7.21 is as follows.

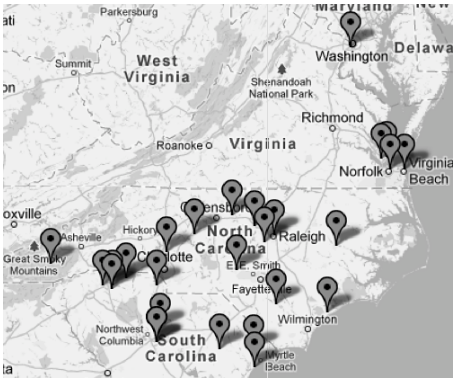

Figure 7.33: Stage 1: (© 2013 Google, © 2013 Tele Atlas)

Figure 7.34: Stage 2

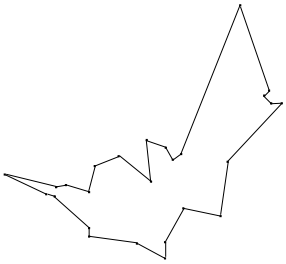

Figure 7.35: Stage $(n-1)$

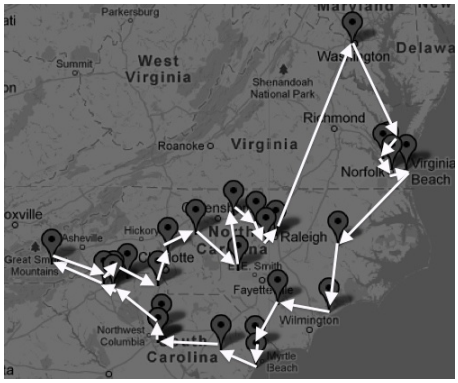

Figure 7.36: Stage n: (© 2013 Google, © 2013 Tele Atlas)

This process is easily repeated for each of the other groups of cities, but is not included in this work.

8.0 CONCLUSION

We have seen for our college visit example that the Geometric Algorithm for the Traveling Salesman Problem produced better results than the Nearest-Neighbor and Closest Insertion Algorithms. This may be true in general, but more research is necessary.

There are many other algorithms for the Traveling Salesman Problem, including the computer program the *Concorde* as explained in Chapter 3. Researchers are constantly trying to improve the results that have been around for years. In 1976, Nicos Christofides [2] created an algorithm that gives a local solution which is at most 1.5 times the optimal global solution. This record was recently surpassed in September of 2012 by András Sebő and Jens Vygen [8] who created an algorithm that gives a local solution which is at most 1.4 times the optimal global solution. Furthermore, it is conjectured that one can find an algorithm that will give a local solution which is at most $\frac{4}{3}$ times the optimal global solution [8].

Whether we travel by a stage coach or a space ship, the Traveling Salesman Problem is relevant today and will continue to be relevant tomorrow.

BIBLIOGRAPHY

[1] Chartrand, Gary and Lesniak, Linda and Zhang, Ping. *Graphs & Digraphs*. Fifth ed. Boca Raton: CRC, 2011. Print.

[2] Christofides, Nicos *Worst-case Analysis of a New Heuristic for the Traveling Salesman Problem*. Technical Report 388, Graduate School of Industrial Administration, Carnegie Melon University 1976. Print.

[3] Cook, William J. *In Pursuit of the Traveling Salesman: Mathematics at the Limit of Computation*. Princeton, NJ: Princeton UP, 2012. Print.

[4] Foulds, L. R. *Combinatorial Optimization for Undergraduates*. New York: Springer-Verlag, 1984. Print.

[5] Gross, Jonathan L., and Yellen, Jay. *Graph Theory and Its Applications*. Second ed. Boca Raton, FL: CRC, 1999. Print.

[6] Knuth, Donald Ervin. "Sorting by Exchanging" *The Art of Computer Programming*. Reading, Mass. [u.a.: Addison-Wesley, 1998. N. pag. Print.

[7] Papadimitriou, Christos H., and Steiglitz, Kenneth. *Combinatorial Optimization: Algorithms and Complexity*. Englewood Cliffs, NJ: Prentice Hall, 1982. Print.

[8] Sebő, András and Vygen, Jens. *Shorter Tours by Nicer Ears*. arXiv:1201.1870v3 2012. Electronic.

INDEX