Analysis of the "Travelling Salesman Problem" and an Application of Heuristic Techniques for Finding a New Solution

Mateusz Pacha-Sucharzewski

Analysis of the "Travelling Salesman Problem" and an Application of Heuristic Techniques for Finding a New Solution

MATEUSZ PACHA-SUCHARZEWSKI

Matt is a junior majoring in mathematics with minors in computer science and physics. This project was mentored by Dr. Ward Heilman and funded by the Adrian Tinsley Program for Undergraduate Research (ATP) in the summer of 2010. Currently, under the guidance of Dr. Heilman, Matt is continuing his research, working on an Honors Thesis. Matt plans graduate work in math and computer science.

In 1832, a German travelling salesman published a handbook describing his profession. Sadly, his name is unknown; he only stated that the book was written by "one old travelling salesman." However, he has come down in history thanks to a rather simple and quite obvious observation. He pointed out that when one goes on a business trip, one should plan it carefully; by doing so, one can "win" a great deal of time and increase the trip's "economy." Two centuries later, mathematicians and scientists are still struggling with what is now known as the "Travelling Salesman Problem" (TSP).

The Problem

The authors of *The Traveling Salesman Problem: The Computational Study* define the TSP as follows: "Given a set of cities along with the cost of travel between each pair of them," the problem "is to find the cheapest way of visiting all the cities and returning to the starting point" (1). In other words, the shortest possible route between a number of cities has to be found. The task does not seem to be difficult -- all of the possibilities have to be checked. Unfortunately, that is where problems start….

For small examples of the TSP (often called "instances" of the TSP), say four or five cities, optimal routes can be identified fairly quickly. The only "advanced" math that has to be applied is addition (adding the distances of used roads produces the tour's length) and a comparison of the results for each possibility; however, with additional cities the number of possibilities for tours grows rapidly, making the TSP unsolvable within a reasonable time frame. Table 1 illustrates the issue.

Table 1

Number of Cities		Number of Possibilities
5	5! =	5*4*3*2*1 = 120
25	25! =	25*24*23*...*3*2*1 = 15,511,210,043,330,985,984,000,000
100	100! =	93,326,215,443,944,152,681,699,238,856,266,700,490,715,968,264,381,621,468,592,963,895,217,599,993,229,915,608,941,463,976,156,518,286,253,697,920,827,223,758,251,185,210,916,864,000,000,000,000,000,000,000,000

The large numbers shown in the Table 1 are truly horrifying; they are impossible to be named and even worse to be grasped. Clearly, larger instances of the TSP cannot be solved by a human with pencil and paper. Luckily, with modern computer technology those numbers probably should not scare us. For instance, assuming that we have a supercomputer capable of checking a billion possibilities every second, 25-city example should be solved fairly quickly. Unfortunately, after a thousand years it would have checked only 31,557,600,000,000,000,00 0 possibilities, and that is not even close!

NP-Completeness

Millions of years of supercomputer computations would bring solutions even to some of the large instances. Therefore, the TSP is, in fact, solvable; however it is not solvable in polynomial time. That means that the TSP is an NP-complete problem. NP-complete problems (NP is an abbreviation for "nondeterministic polynomial time") cannot be solved in a reasonable amount of time due to the extreme number of cases. However, if a perfect solution is known, a test of any given solution would not take long. It is best described by the statement that "if only we could guess the right solution, we could then quickly test it" (Eppstein). Math is not about guessing, though: It is about solving and proving. As David Eppstein very accurately pointed out, "NP-completeness is a form of bad news: evidence that many important problems can't be solved quickly."

History

In the mid-nineteenth century, mathematicians were already looking at the TSP. The best known among them was Sir William Rowan Hamilton, an Irish scientist who is famous for his contributions to mathematics (especially graph theory) and physics. However, the TSP was not officially recognized to be a problem in mathematics until the 1930s when Dr. Karl Menger, Professor of Mathematics at the Illinois Institute of Technology, shaped the definition of the TSP.

The interest in the problem seemed to have its peak at the beginning of the twenty-first century. The Clay Mathematics Institute, a mathematical non-profit foundation established in 1998 by London T. Clay, a Boston businessman, to "increase and disseminate mathematical knowledge," created a list of Millennium Prize Problems and the TSP is one of them. The Clay Institute offers $1,000,000 for anyone whose TSP solution succeeds perfectly.

Applications

The incredible amount of money offered by the Clay Institute may be surprising. However, when taking into consideration how much money and possibly time various government and non-government institutions and companies may save by having the ability of finding a perfect solution to any TSP example, the reward value becomes understandable. For example, according to the United States Postal Service, their fleet travels about one and a quarter billion miles every year. More efficient routes could result in millions of dollars in savings on fuel then. And that applies to basically any organization connected to delivering goods or providing on-site services. A perfect TSP solution could also be helpful with machine sequencing in industry and genome sequencing problems in genetics (Chen, 139).

The TSP in Mathematics

In search of a perfect solution for the TSP, mathematicians were and are trying different approaches, quite often using geometry and linear algebra. However, over time, the problem is best settled in graph theory. Graph theory allows visual representations of mathematical problems. Its basic element, a graph, is a set of vertices connected by edges. In other words, it is a collection of dots connected with lines. If all of the vertices are directly connected with each other, then the graph is complete.

The TSP instances can be well transformed into (usually complete) graphs; every vertex can have a city assigned to it while edges may carry the values of distances between the cities.

Human Performance

Psychologists are not surprised that graph theory overpowered other areas of mathematics, claiming the TSP as its own. The key to the mystery seems to lie in how we comprehend things. Once a TSP example is represented visually, the human brain is capable of quickly finding good solutions. Dr. Iris van Rooij conducted a study where the participants were given sheets of paper with sets of dots on them and were asked to connect them as efficiently as they can. Table 2 (source: *The Traveling Salesman Problem: A Computational Study*) shows the average results achieved by the participants; the percentage indicates additional tour length in comparison to the optimal tours.

Table 2

Number of Cities	7-Year-Olds	12-Year-Olds	Adults
5	3.8%	2.5%	1.7%
10	5.8%	3.4%	1.7%
15	9.4%	5.0%	2.7%

Results of the study by Dr. Iris van Rooij

Source: The Traveling Salesman Problem: A Computational Study

Interestingly, age greatly contributes to better results. According to the researchers, when the brain develops, "both perception and cognition" improve (Applegate, 35); there also is "a modest correlation between TSP performance and the scores of individuals on a standard nonverbal intelligence test" (Applegate, 36), and they all play a key role in identifying more efficient solutions. Psychologists also point out "a human desire for minimal structures" and an appreciation of aesthetics (Applegate, 32). Dr. Douglas Vickers, an Australian psychologist, pointed out that: "the link with intelligence, and the occurrence of optimal structure in the natural world, suggest that the perception of optimal structure may have some adaptive utility" (qtd. by Applegate, 35).

Important Observation

The appreciation for tour aesthetics combined with graphical representations of the TSP resulted in an important discovery. Namely, no optimal TSP solution has roads that cross each other. Allow me to demonstrate:

Let four random cities of a TSP tour be named A, B, C and D. City A is connected to D with edge AD and B is connected to C with edge BC, and the cities are positioned such that those edges cross each other at point M (Figure 1 illustrates the situation). So, $|AD| = |AM| + |DM|$ and $|BC| = |BM| + |CM|$. There are also unused alternative edges AB and CD that connect cities A with B and C with D (where AB and CD do not cross each other). The edges create two triangles: ΔABM and ΔCDM, with M being a common vertex. In order to have a triangle, the sum of any two of its edges must be greater than its third edge. So, $|AM| + |BM| > |AB|$ and $|CM| + |DM| > |CD|$. As a result, $|AM| + |BM| + |CM| + |DM| > |AB| + |CD|$, which can be rewritten as $|AD| + |BC| > |AB| + |CD|$. Therefore, uncrossing edges makes the tour shorter. Thus, no optimal TSP tour contains roads that cross each other. Q.E.D.

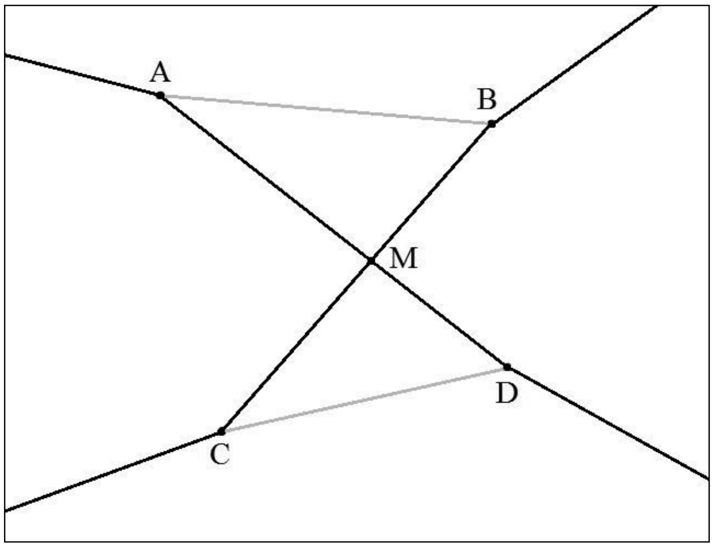

Figure 1

Held-Karp Lower Bound (H-Klb) Estimation

Toward the end of 1970s, an important tool was developed which is used to estimate what the length of an optimal tour should be. Designed by Dr. Martin Held and Dr. Richard M. Karp, it is called the Held-Karp lower bound (H-Klb) estimation. Before explaining its principles, I ought to define a few terms.

In the field of graph theory, if a traveling salesperson starts at a vertex (city), visits at least two other vertices, and finishes at the initial one without repeating any of the edges (roads) or vertices along the way, then his/her tour is a cycle. When there are two or more non-empty sets of vertices (every set has to contain at least one vertex) in a graph and there are no edges between those sets, then the graph is disconnected. Using the terms defined above, Dr. Gary Chartrand and Dr. Ping Zhang describe a tree as "an acyclic connected graph" (87). If vertices of a tree are connected such that the sum of all the edges is the smallest possible, that it is called a minimum tree.

H-Klb is based on a minimum 1-tree, which is a minimum tree containing one cycle. The minimum 1-tree undergoes a series of transformations; subsequently multiple formulas alter a number, which at the end becomes the lower bound. Dr. David S. Johnson and his colleagues established that the H-Klb is "on average within 0.8% of the optimum for (…) instances with many thousands of cities" (Jones, 5) and "the gap is almost always less than 2%" (Johnson, 1). Due to extensive numbers of required operations, only computers are used to estimate the lower bound.

Methods

Years of research and tests have brought some interesting techniques, which allow people who work on TSP instances to either find TSP tours from scratch or improve the ones that are already in existence.

n-Opt Moves:

When trying to improve an existing TSP tour, a method called *n*-Opt Moves offers help. Its rules are quite simple: identify and remove *n* edges and replace them with *n* alternative connections, but only if the modified tour will turn out to be more efficient. For instance, a 2-opt move, meaning find 2 edges, remove both and create 2 alternative connections, can be used to uncross edges.

Cutting Planes

"A breakthrough in solution methods for the *traveling salesman* *problem* (TSP) came in 1954, when George Dantzig, Ray Fulkerson, and Selmer Johnson (…) published a description of a method for solving the TSP and illustrated the power of this method by solving an instance with 49 cities, an impressive size at that time" (Georgia Institute of Technology). This method is called Cutting Planes and is based on identifying groups of clustered cities (or vertices), described as cutting planes, and separating them from the rest of the cities, creating smaller instances. The small cases can be solved fairly quickly and then reconnected with each other. In 1976, Dr. Martin Grötschel adapted the method and found an optimal tour between 120 cities in Germany. Figure 2 (source: *The Traveling Salesman Problem: A Computational Study*) shows Grötschel's hand drawing that identifies cutting planes (Applegate, 111).

Control Zones and Moats:

As I briefly mentioned before, geometry also has a role to play in the problem. The method called Control Zones and Moats is purely geometrical. The technique calls for every city to be surrounded with a circular area with an arbitrary radius, and every area should connect with two other ones, but they cannot overlap with each other. Those areas are called control zones. If there is a gap between some areas that causes some clustered control zones to be separated from the rest, then additional larger circular areas, called moats, are assigned to create the missing connections. Figure 3 (source: Georgia Institute of Technology: http://www.tsp.gatech.edu/) shows control zones marked in light gray and moats in dark gray.

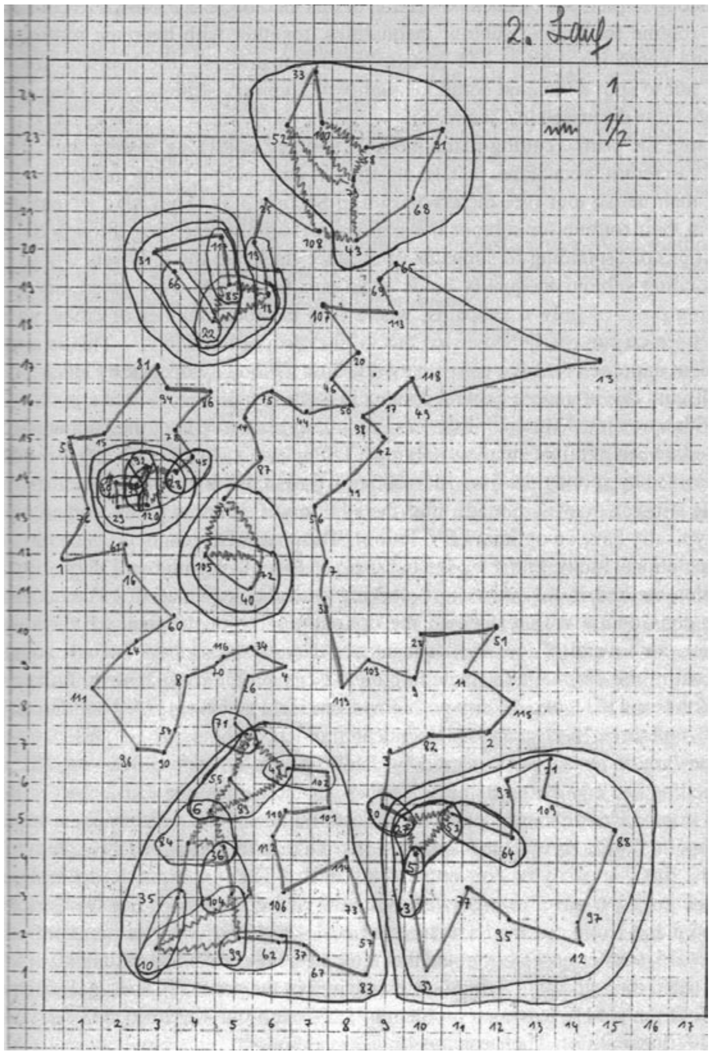

Figure 2

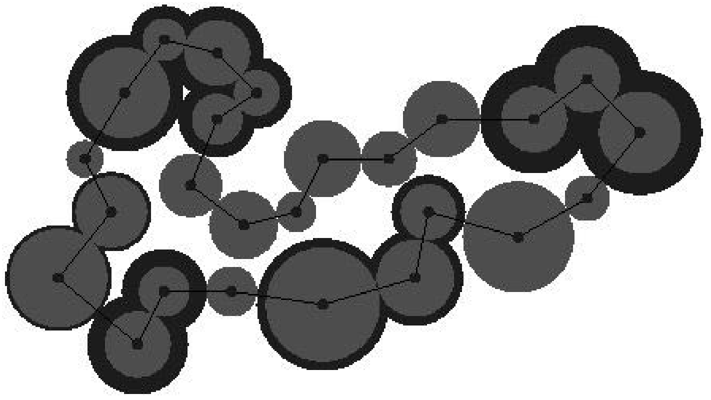

Figure 3

Algorithms

In addition to the above methods for attempting to solve the TSP, there are TSP algorithms. An algorithm is a specific set of rules which establishes how to treat a given problem; usually it defines a step-by-step repetitious procedure which has to be followed. Before I describe some of the TSP algorithms, I should explain that the number of edges incident with (connected to) a vertex is called a degree (Chartrand, 27).

Greedy Algorithm

The Greedy Algorithm is based on building a tour by selecting and using the shortest possible edges. The initial step is to make a list of all edges in order of their increasing length. Once the list is completed, the shortest available edges are repeatedly chosen and included in the tour. While selecting the edges, it is necessary to make sure that the next edge intended to be used will neither close a smaller cycle, excluding some vertices, nor connect to a vertex which already has a degree 2. The algorithm's results exceed optimal tours by 14% on average, judging by the H-Klb.

Vertex Insertion Algorithms

Vertex Insertion Algorithms are based on picking an initial vertex and then including the rest of the vertices into a tour, adding one at a time. There are different ways of choosing which vertex should be included next.

The Cheapest Insertion Algorithm inserts a "free" vertex into a partial tour such that the tour-length increase will be the smallest possible. This variation of Vertex Insertion provides tours that are about 27% longer than suggested by the H-Klb. Nearest Insertion and Farthest Insertion algorithms insert a "free" vertex which is respectively the closest to or the farthest from any of the vertices that are already a part of the tour. The Nearest exceeds the H-Klb by 22% and the Farthest goes over the bound by only 11%. The best among the Vertex Insertion Algorithms turns out to be the Random Insertion Algorithm. Just as the name suggests, a purely random "free" vertex is inserted into the partial tour; the entire tour at the end goes beyond the H-Klb estimation by 9%.

The last but not the least among the Vertex Insertion algorithms is the Angle Selection Algorithm which is based on geometry; therefore it can only be used on geometric graphs of the TSP (where relative positions of every vertex and distances between the vertices correspond adequately to its real life equivalents). This algorithm, which has a similar principle to the Cheapest Insertion (lowest possible increases in the partial tour's length), inserts a "free" vertex into a partial tour such that the angle created by the two new edges connecting the newest city to the rest of the tour will be as close to 180° as possible. Figure 4 shows an example of two possible angles, AMB and ANB, which can be created by including cities M and N into the tour. Visibly, adding city M and the angle AMB, which is closer to 180° than the angle ANB, into the tour will result in a smaller length increase (note that the edge AB will disappear after including city M).

Nearest Neighbor Algorithms

Two Nearest Neighbor Algorithms are named perfectly; they also resemble the Nearest Insertion Algorithm. For both, an initial vertex has to be chosen. At this point, a travelling salesperson has to make a decision on how to build his/her tour. The basic Nearest Neighbor algorithm asks the person to expand the trip in just one direction -- from the initial city the person has to visit its closest neighbor, from the neighbor to the neighbor's closest neighbor, without repeating any cities, and so on; when no more "free" cities are left, the salesperson goes back directly to the initial city, closing a cycle. For the Double-Ended Nearest Neighbor Algorithm, once the salesperson has the first two vertices connected with an edge, he/she has to check which of the two cities has the closest "free" neighbor available and connect it to the trip. In other words, the trip can be expanded in two directions, until a cycle is closed at the very end. Once again, a short cycle cannot be closed, excluding some vertices. Both variations exceed the H-Klb by circa 23%.

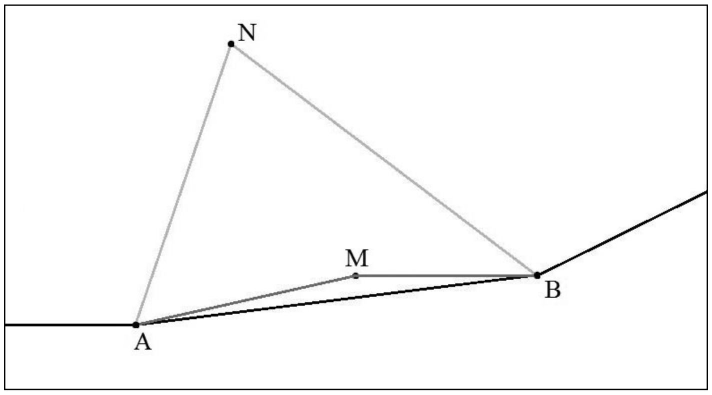

Figure 4

Heavy Edge and Degree (HEaD) Algorithm

The HEaD Algorithm is my own creation. As I discovered during my TSP research, I approached the problem differently than anybody else, in a way creating an opposition to the Greedy Algorithm, actually even before being aware of the existence of the Greedy Algorithm. This is how HEaD works:

The TSP instance has to be presented as a complete graph, unless there are vertices that simply do not have a direct connection between them, and every vertex needs to have a degree of at least 2. First, I start with identifying the degrees of the vertices. Then, I focus only on the vertices that uniquely hold or share the highest degree among all. Following that, I find the longest edge connected to any of those high degree vertices, and remove it. At that point, the two of the vertices which were previously connected by the removed edge go down by a degree. I keep identifying the high degree vertices and removing the longest edges connected to them until every vertex has a degree 2 -- I then have a cycle.

There are a couple of things that I have to be careful about, though. When removing an edge, I have to make sure that the removal will cause neither of the two vertices that the edge connects together to go below degree 2 or force me to close a short cycle. Also, if there are multiple longest edges which connect to a highest degree vertex, which one should be removed? In such a case, I look at the vertices that the longest edges are connected to at their other ends, and remove the edge which at its other end has the highest possible degrees (the highest degree here does not have be the highest in the graph at the time). I should point out that there could be an instance where the by-the-rules edge removal determination (which among the multiple longest edges connected to a highest degree vertex should be removed) would be impossible, and then the edge choice must be arbitrary. An extreme example of such an instance can be a complete graph that has all the edges of the same length.

So far, HEaD turned out to be able to provide optimal tours for some TSP instances, on average exceeding the H-Klb by about 5%.

Why Do Algorithms Fail?

The difficulty with the TSP instances is that the shortest possible edge may not always be included in an optimal tour while sometimes the longest possible one has to be a part of an optimum. That is why, for instance, both Greedy and HEaD algorithms fail, providing results which exceed optimums. The Greedy Algorithm may force the use of an unwanted short edge, while HEaD may eliminate a long edge that should be used in an optimal tour. So far, no one has discovered a way around the issue and all of the current algorithms produce results only fairly close to optimal according to H-Klb, even though there are TSP examples where certain algorithms do produce optimal tours.

There is also an undefined step for Greedy algorithm: Let two edges have the same length, which would be the shortest length available at the moment. If the use of both those edges would result in closing a smaller cycle, then only one of them could be used. However, there is no rule which can determine which one of the edges to choose and include in the tour.

Combining Algorithms

At this point the best method for attempting to solve TSP instances is using multiple algorithms and techniques for a single instance. For example, a tour can be produced by the Greedy Algorithm, and then a series of 2-opt moves can be used to improve the tour.

Solved cases

There is also a computer program called Concorde based on multiple algorithms. Over the years, with some improvements along the way, it has shown to be a wonderful tool capable of fairly quickly finding near optimal solutions.

The largest instance that the Concorde was working on is the World TSP Tour of 1,904,711 places around the globe. "The current best lower bound on the length of a tour for the World TSP is 7,512,218,268" (Georgia Institute of Technology: http://www.tsp.gatech.edu/) with the best found tour length of 7,515,796,609. "The bound shows that Keld Helsgaun's tour," of 7,515,877,991, (best until May 4, 2010) "has length at most 0.0477% greater than the length of an optimal tour" (Georgia Institute of Technology: http://www.tsp.gatech.edu/) which is extremely close!

The TSP has been and still is unsolvable but there is a hope for a perfect solution!

References

Applegate, David L., Robert E. Bixby, Vašek Chvátal, and William J. Cook. *The Traveling Salesman Problem: A Computational Study*. Princeton: Princeton UP, 2006. Print.

Chartrand, Gary, and Ping Zhang. *Introduction to Graph Theory*. New York: McGraw-Hill Higher Education, 2005. Print.

Chartrand, Gary. *Introductory Graph Theory*. Mineola: Dover Publications, 1985. Print.

Chen, Der-San, Robert G. Batson, and Yu Dang. *Applied Integer Programming*. Hoboken: John Wiley & Sons, 2010. Print.

Eppstein, David. "ICS 161: Design and Analysis of Algorithms." Lecture. *University of California, Irvine*. 2 May 2000. Web. 14 Mar. 2010. <http://www.ics.uci.edu/~eppstein/161/960312.html>.

Garey, Michael R., and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. San Francisco: W.H. Freeman and, 1979. Print.

Gutin, Gregory, and Abraham P. Punnen, eds. *The Traveling Salesman Problem and Its Variations*. New York: Springer, 2007. Print.

Johnson, D. S., L. A. McGeoch, and E. E. Rothberg. *Asymptotic Experimental Analysis for the Held-Karp Traveling Salesman Bound*. 1996.

Jones, Antonia J., and Christine L. Valenzuela. *Estimating the Held-Karp Lower Bound for the Geometric TSP*.

Jünger, Michael, and Denis Naddef, eds. *Computational Combinatorial Optimization: Optimal or Provably Near-Optimal Solutions*. Berlin: Springer, 2001. Print.

Lawler, E. L., J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys, eds. *The TravelingSalesman Problem: A Guided Tour of Combinatorial Optimization*. New York: John Wiley & Sons, 1985. Print.

The Traveling Salesman Problem. Georgia Institute of Technology, May 2010. Web. 26 Sept. 2010. <http://www.tsp.gatech.edu/>.

Thulasiraman, K., and M. N. S. Swamy. *Graphs: Theory and Algorithms*. New York: John Wiley & Sons, 1992. Print.