Running Head: TRAVELING SALESMAN

The Traveling Salesman Problem: Deceptivley Easy to State; Notoriously Hard to Solve

David Biron

A Senior Thesis submitted in partial fulfillment
of the requirements for graduation
in the Honors Program
Liberty University
Fall 2006

Acceptance of Senior Honors Thesis

This Senior Honors Thesis is accepted in partial
fulfillment of the requirements for graduation from the
Honors Program of Liberty University.

_____
Monty Kester, Ph.D.
Chairman of Thesis

_____
Timothy Van Voorhis, Ph.D.
Committee Member

_____
Edward Lewis, Ph.D.
Committee Member

_____
James Nutter, D.A.
Honors Program Director

_____
Date

Abstract

The purpose of this thesis is to give an overview of the history of the Traveling Salesman Problem and to show how it has been an integral part of the development of the fields of Integer Programming, and Combinatorial Optimization. The thesis starts in the 1800s and progresses through current attempts on solutions of the problem. The thesis is not meant to describe in detail every attempt made, nor to describe an original solution, but to provide a high level overview of every solution attempt, and to guide the reader on what has been done, and what still can be done.

The Traveling Salesman Problem: Deceptivley Easy to State; Notoriously Hard to Solve

Mathematics has captured some of the greatest intellectual minds. Many times a problem is posed to a mathematician, and it consumes his time until he finds a solution. Every once in a while there is a problem that captures the attention of many mathematicians. The Traveling Salesman Problem (TSP) is a problem whose solution has eluded many mathematicians for years. Currently there is no solution to the TSP that has satisfied mathematicians. The TSP has a very rich history. At the same time the TSP was being eagerly investigated a field called Integer Programming (IP) was also developing. Because the TSP can be formulated as an IP problem the history of these two are intertwined. Breakthroughs in IP were applied to the TSP as a way of showing that the breakthrough was valid. Thus a study of the TSP naturally studies the main aspects of IP.

Before studying the history of the TSP it is important to state the TSP. The TSP is a Combinatorial Optimization problem, simply stated as: "What is the shortest route a traveling salesman can take to visit n cities and return back to his home city, only going through each city once?" From now on the word "tour" will refer to a solution to the TSP or simply one route that the salesman could take. The word "solved" will be used in reference to the optimal or shortest tour. The largest tour to be solved and proven optimal is 24,978 cities in Sweden by David Applegate - AT&T Labs – Research Robert Bixby - ILOG and Rice University, Vašek Chvátal Rutgers University, William Cook - Georgia Tech, and Keld Helsgaun - Roskilde University on April 2001. (http://www.tsp.gatech.edu/) The TSP is obviously easy to state, and may seem simple to

solve. A "brute force and sheer ignorance"[1] approach to solving the TSP would be to find every possible tour and then see which one was the shortest. In order to do this there would be (n-1)! different tours to study and thus for problems with just 20 cities it would be illogical to solve this way even with a computer. To show this a very simplistic algorithm in pseudocode is presented below:

Procedure TSP ()

For i_1 = 1 to n

For i_2 = 1 to n

For i_3 = 1 to n

…

For i_n = 1 to n

If ($i_1, i_2, i_3, ..., i_n$ is a valid tour)

If ($i_1, i_2, i_3, ..., i_n$ is shorter then the current optimal)

TSP = $i_1, i_2, i_3, ..., i_n$

{Returns TSP which is the city progression that is the shortest}

Clearly this algorithm is nowhere near optimality. The algorithm is O(n^n). Obviously worse then the (n-1)! algorithm. Assume that a computer can process information at 1 operation per nanosecond. So for a 5 city tour it would require 3125 operations taking .000003125 seconds. Not so bad. A ten city tour would require 10000000000 operations, requiring 10 seconds, workable. A 20 city tour would require 104857600000000000000000000 operations taking 104857600000000000 seconds or

[1] I must give credit to the Chair of my Thesis, Dr. Monty Kester for this statement. He has used it many times in class to describe an algorithm that uses absolutely no mathematical finesse and tries to solve a problem in the simplest and easiest, yet sometimes longest way possible.

3325012684 years. Naturally we would rather not wait around to see that finish. Thus for a problem as small as 20 cities, the algorithm grows uncontrollably. Like many algorithms a "brute force and sheer ignorance" approach is not even close to optimal. The problem is then to find an algorithm that reduces the amount of computing time. Thus many mathematicians and computer scientists have tried to compile algorithms that solve the TSP in less time.

As we begin our journey into the history of the TSP we struggle to find the exact origin of the problem. Quite possibly the original proposal of the problem goes back to 1856 by Hamilton. In 1857, Hamilton created a game called the "Icosian Game" which was defined as: "the problem of finding a Hamiltonian Circuit along the edges of a dodecahedron, i.e., a path such that every vertex is visited a single time, no edge is visited twice, and the ending point is the same as the starting point" (Weisstein, 2003, p. 1). The undertone of the Icosian Game, now known as a Hamiltonian Circuit is that of the Traveling Salesman Problem. Thus the first real mention of the Traveling Salesman Problem, though nowhere near its current formulation can be traced back to Hamilton.

In 1832 a German manual was produced for the "successful traveling salesman." The pamphlet was called "The traveling salesman – how he should be and what he has to do, to obtain orders and to be sure of a happy success in his business – by an old traveling salesman" (Aardal (Ed.) et al., 2005, p. 38). Although the pamphlet has little to do with the problem at hand it does have a section in it that states the problem, but gives no real mathematical formulations for it:

> Business brings the traveling salesman now here, then there, and no travel routes can be properly indicated that are suitable for all cases occurring; but sometimes,

> by an appropriate choice and arrangement of the tour, so much time can be gained, that we don't think we may avoid giving some rules also on this. Everybody may use that much of it, as he takes it for useful for his goal; so much of it however we think we may assure, that it will not be well feasible to arrange the tours through Germany with more economy in view of the distances and, which the traveler mainly has to consider, of the trip back and forth. The main point always consists of visiting as may places as possible, without having to touch the same place twice. (Schrijver (Au.) et al., 2005, p. 38)

Thus the TSP sat in relative silence until the 20th century. While the question did exist, it was not discussed much in mathematical circles until the early 1900s.

In 1930 Karl Menger flirted with the problem. Schrijver credits him as the first mathematician to write about the TSP. Schrijver (2005) quotes his paper saying:

> We denote by messenger problem (since in practice this question should be solved by each postman, anyway also by many travelers) the task to find, for finitely many points whose pairwise distances are know, the shortest route connecting the points. Of course, this problem is solvable by finitely many trials. Rules which would push the number of trials below the number of permutations of the given points are not known. The rule that one first should go from the starting point to the closest point, then to the point closest to this, etc., in general does not yield the shortest route. (p. 41)

In his papers Menger shows that the length of simple curves $l(C) = \sup_X \lambda(X)$ where $\lambda(X)$ is the shortest length of a Hamiltonian path. He later showed that

$l(C) = \sup_X \kappa(X)$, where $\kappa(X)$ is the minimum length of a spanning tree on X. Thus Menger in 1930 seems to be the first mathematician to talk about the TSP. (Schrijver Au. Et al., 2005, p41) It is interesting to see that even in the earliest paper on the TSP, Menger realizes that a greedy algorithm, now know as the nearest neighbor algorithm would not produce the shortest route.

Though Menger seems to be the first to write about the TSP Dantzig, Fulkerson, and Johnson (1954) credit Merrill Flood with the one to generate interest of the problem: "Merrill Flood (Columbia University) should be credited with stimulating interest in the traveling-salesman problem in many quarters." (p. 393) Flood (1956) was particularly interested in the TSP during 1937 as he was "struggling with the problem in connection with a school-bus routing study in New Jersey" (p. 61). Though Flood stimulated interest, it is Hassler Whitney who posed the problem: "This problem was posed, in 1934, by Hassler Whitney in a seminar talk at Princeton University." (Flood, 1956, p. 61) Like Flood, Dantzig, Fulkerson and Johnson (1954) also give Whitney credit: "Both Flood and A. W. Tucker (Princeton University) recall that they heard about the problem first in a seminar talk by Hassler Whitney at Princeton in 1934 (although Whitney, recently queried, does not seem to recall the problem" (p. 393). Thus Whitney gets credit for formulating the modern version of the problem and Flood for promoting the problem.

Although there was mathematical interest in the TSP, not until Dantzig, Fulkerson, and Johnson were there any real breakthroughs in the problem: "The origin of this problem is somewhat obscure. It appears to have been discussed informally among mathematicians at mathematics meetings for many years. Surprisingly little in the way of results has appeared in the mathematical literature" (Dantzig, Fulkerson, & Johnson,

1954, p. 393). Flood (1956) claims "there are as yet no acceptable computational methods, and surprisingly few mathematical results relative to the problem" (p. 61). Croes (1958) acknowledged that "The only fairly successful attempt at finding such a method (for symmetrical problems) has been reported by Dantzig et al" (p. 792). Dantzig, Fulkerson, and Johnson solved the problem for a 49 city tour through the capital of every state in America, and Washington, D.C. On http://www.tsp.gatech.edu the claim is made "This is the granddaddy of TSP papers. It reports on the solution of a 49-city TSP via linear-programming methods. Many of the ideas used to solve integer programming problems can be traced back to this paper" (2007, par. 3). Dantzig et al formulate the problem in a linear fashion with x_{ij} as the variable, with i and j representing the city leaving, and the city arriving respectively. This takes the value 0 if this edge is not included in the tour and 1 if the edge is included in the tour. The first set of constraints Dantzig et al. (1954) give are:

$$\sum_{J=1}^{n} x_{ij} = 2\,.\ (x_{ij} \geq 0; I = 1,2,...,n; I \neq J; x_{IJ} \equiv x_{JI}) \qquad (1)$$

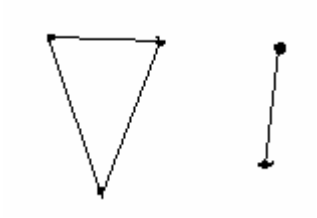
Where the objective would be to find the minimum of:

$$D(x) = \sum_{I>J} d_{IJ} x_{IJ} \qquad (2)$$ (p. 396).

Dantzig et al. (1954) acknowledged that in order to formulate the TSP as a Linear Program (LP) there would be more constraints and finding these constraints would be an extremely difficult task:

> To make a linear programming problem out of this one needs, as we have observed, a way to describe tours by more linear restraints than that given by (1).

> This is extremely difficult to do as illustrated by work of I. Heller and H. Kuhn. They point out that such relations always exist. However, there seems to be no simple way to characterize them and for moderate size n the number of such restraints appears to be astronomical (p. 397).

The difficulty Dantzig et al. ran into is now commonly referred to as subtours. The easiest way to describe a subtour is a picture. 

The points represent the cities and the lines the connectors. Obviously this is not a solution, but this does satisfy (1) and thus more constraints must be added. This is the primary difficulty of the traveling salesman.

Dantzig et al. (1954) provided four different devices in their paper to simplify the TSP. The first was to use undirected tours. This has now been given the name the symmetric traveling salesman. The symmetric TSP is much easier that then the asymmetric TSP and it is not unrealistic to view the TSP as though it is symmetric. Their second device is to not use all linear constraints, but to add in addition to (1) just enough constraints so that no sub-tours are given. Thirdly is a device that speeds up the iterative process, and fourthly finding a tour that is nearly optimal and then listing all possible tours that have not yet been eliminated by the constraints provided. Dantzig, Fulkerson, Johnson in a later paper describe the process in the paper as this:

> The linear-programming approach suggestion in reference 1 (this is their original paper) is to start with a tour and a small number of linear equality and inequality constraints that are satisfied by all tours, then use the simplex method to move to

> a new basic solution. If the new solution is not a tour, impose an additional constraint on the problem that cuts out this solution but no tours, and again, in the new convex set thus defined, move to an adjacent solution. As a suitable stage in the process, it is usually advantageous to use the estimation procedure described in reference 1 in conjunction with a combinatorial analysis of undominated tours. (Dantzig, Fulkerson, & Johnson, 1959, p. 58-59)

Thus Dantzig et al. started the beginning of many methods that would be applied to many integer programming problems. Dantzig et. al. did not have the advantages of modern day computers. Thus much of their work including their solution to the 49 city problem was done by hand. The two biggest contributions that are still seen in modern solutions are to relax the "subtour" elimination constraints and add them as necessary and to formulate the problem as a LP.

Flood in 1955 reported on the TSP. Among his talk he discussed some heuristics. Heuristics are algorithms that do not attempt to give an optimal solution, but try and find a near optimal solution instead focusing on keeping computation time low. Flood proposed what is now considered the Nearest Neighbor solution. While Menger may have mentioned it in his paper Flood (1956) showed how effective it really was:

> It may be of some interest to compare the length of optimal tour found among the 49 cities by Dantzig et al with that which would be followed by a salesman living in Washington, D.C., who always went next to the closest city not already visited. This turns out to be 904 units against 699 units, or an increase of nearly 30 percent. It seems likely that a considerably better route than that produced by the operator's rule could usually be found quite easily, and it also seems likely that

> rather simple methods could be found to yield a tour much nearer optimal than 30 percent. However, even a few percent gain would be well worth-while in some cases, so the problem does seem to have practical importance as well as mathematical interest. (p. 65)

Thus Flood realized that the Nearest Neighbor method is not a good estimate of the TSP but it created a decent first solution.

In 1962 a contest brought the TSP national recognition through a contest given by Proctor and Gamble. A flyer of the contest is pictured below.

The traveling salesman problem recently achieved national prominence when a soap company used it as the basis of a promotional contest. Prizes up to $10,000

> were offered for identifying the most correct links in a particular 33-city problem. Quite a few people found the best tour… A number of people, perhaps a little over-educated, wrote the company that the problem was impossible – an interesting misinterpretation of the state of the art. (Little et al., 1963, p. 973)

In the 1960's there were two main breakthroughs, the "branch and bound" method and "dynamic programming." The term branch and bound was coined by Little, Murty, Sweeny, and Karel. They discussed two papers that proposed similar algorithms. One by Rossman, Twery, and Stone where they proposed an idea called combinatorial programming solve a 13-city problem that was solved in 8 man-days. Little et al claim to have solved it using their method in 3 and a half hours by hand. Likewise Eastman in a doctoral thesis gave a similar idea, but with significant variances. Eastman used it to solve a 10 city problem, but gave no computational time. (Little et al., 1963, p. 974) Thus it appears as if Little et al. can be accurately accredited with the formulation of the branch and bound method (b&b). Little et al. (1963) summarize their method:

> The basic method will be to break up the set of all tours into smaller and smaller subsets and to calculate for each of them a lower bound on the cost (length) of the best tour therein. The bounds guide the partitioning of the subsets and eventually identify an optimal tour – when a subset is found that contains a single tour whose cost is less than or equal to the lower bounds for all other subsets, that tour is optimal. (p. 974)

The b&b can be describe by divide and conquer. In each iteration of the b&b a lower bound is calculated by finding a tour that includes subtours. Then from that lower bound

two branches come off of it. One branch forbids the tour to contain the edge i,j. The other branch forces the tour to contain the edge i,j. Each of those is then solved and new lower bounds found. Then one branch is chosen, and the process continues until the lower bound equals a tour that is allowable. Then the solution is deemed optimal. Below is a diagram taken from Little et al.'s paper showing the branching and bounding. (1,4) represents a tour that is forced to have (1,4) in it, and ($\overline{1,4}$) represents a tour that is forced to not have the edge (1,4) in it.

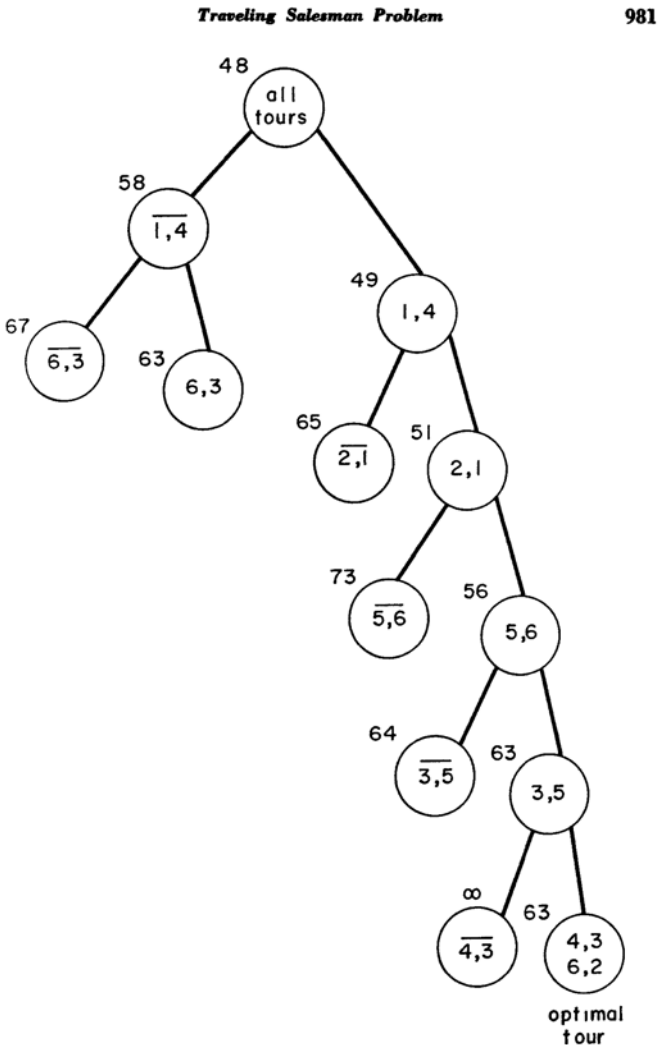


Figure 6. Final tree

Little et al. (1963) formulation of the b&b is very significant. Two things really attributed to its continued popularity. The first is that it successfully reduced problems to make them easier to solve. It was very systematic and could be applied to a variety of problems. It was more an approach to a problem than just merely a solution to one. The second is that it was very easily coded into a computer. Its algorithmic nature allowed it to be turned easily into code. Thus as the computer became poplar, this algorithm was able to be coded into the computer, taking advantage of the computer's computational speed. The b&b method still is a very popular method for solving I.P. problems, and many textbooks on IP include it as an effective way to solve problems. Thus one major breakthrough in the TSP was applied directly to IP.

Dynamic Programming (DP) was also formulated in the 1950s by Richard Bellman. How Bellman (1984) came up with name is quite interesting:

> An interesting question is, 'Where did the name, dynamic programming, come from?' the 1950s were not good years for mathematical research. We had a very interesting gentleman in Washington named Wilson. He was Secretary of Defense, and he actually had a pathological fear and hatred of the word, research. I'm not using the term lightly; I'm using it precisely. His face would suffuse, he would turn red, and he would get violent if people used the term, research, in his presence. You can imagine how he felt, then, about the term, mathematical. The RAND Corporation was employed by the Air Force, and the Air Force had Wilson as its boss, essentially. Hence, I felt I had to do something to shield Wilson and the Air Force from the fact that I was really doing mathematics inside the RAND Corporations. What title, what name could I choose? In the first place

> I was interested in planning, in decision making, in thinking. But planning, is not a good word for various reasons. I decided therefore to use the word, 'programming.' I wanted to get across the idea that this way dynamic, this was multistage, this was time-varying – I thought, let's kill two birds with one stone. Let's take a word that has an absolutely precise meaning, namely dynamic, in the classical physical sense. It also has a very interesting property as an adjective, and that is it's impossible to use the word, dynamic, in a pejorative sense. Try thinking of some combination that will give it a pejorative meaning. It's impossible. Thus, I thought dynamic programming was a good name. It was something not even a Congressman could object to. So I used it as an umbrella for my activities. (p. 159)

Thus DP was born in the RAND Corporation by Richard Bellman. As the TSP had gained much popularity by now, many new developments in integer programming and Combinatorial Optimization were then immediately applied to the TSP. Thus is the case with Bellman. Bellman (1961) wrote a short paper that described the TSP in a dynamic programming sense: "The purpose of this note is to show that this problem can easily be formulated in dynamic programming terms, and resolved computationally for up to 17 cities" (p. 61). Thus Bellman (1961) formulated in DP terms and discussed some advantages to using DP: "One advantage of the dynamic programming approach is that one can readily incorporate all types of realistic constraints involving the order in which cities can be visited" (p. 63). Bellman (1962) realized that there were problems with using DP to solve the TSP: "The only problem to be faced in using the foregoing algorithm to obtain a solution to the traveling salesman problem for an arbitrarily large

number of cities is the storage problem" (p. 62). Bellman (1962) goes to show that computers of the day could solve a 11 city TSP, a 17 city TSP would require the best computer of the day, and that "problems involving 21 cities are for a few years at least beyond our reach" (p. 62). Thus DP was used to solve the TSP.

A third solution applied to the TSP in the 1960s was Gomory Cuts: "Miller, Tucker, and Zemlin, whose experiments using an all-integer program of Gomory did not produce results in cases with ten cities although some success was achieved in cases of simply four cities" (Bellman, 1962, p. 61). Miller, Tucker, and Zemlin seem to be the first to formulate the TSP in an "Integer Programming" sense. An Integer Program (IP) is essentially a LP where the variables are required to be integers. Miller et al. actually formulated in what is now known as a Binary IP (BIP). They formulated it as this:

Minimize the linear form:

$$\sum_{0 \le i \ne j \le n} \sum d_{ij} x_{ij}$$

Over the set determined by the relations

$$\sum_{\substack{i=0 \\ i \ne j}}^{n} x_{ij} = 1 \qquad (j = 1, \ldots, n)$$

$$\sum_{\substack{j=0 \\ j \ne i}}^{n} x_{ij} = 1 \qquad (i = 1, \ldots, n)$$

$$u_i - u_j + p x_{ij} \le p - 1 \quad (1 \le i \ne j \le n)$$

(Miller, Tucker, & Zemlin, 1960, p. 327)

After formulating it as an IP Miller et al. used Gomory Cuts (GC) to solve the TSP. Out of GC grew a methodology called the Cutting Planes method in which new constraints are imposed by making valid inequalities so that the problem becomes easier

to solve, without ever removing the optimal solution. Cutting Planes and GC's are not very effective. Miller et al. found that even for a 10 city problem it was encountering problems. Thus they conclude "It seems hopeful that more efficient integer programming procedures now under development will yield a satisfactory algorithmic solution to the traveling salesman problem, when applied to this model. In any case, the model serves to illustrate how problems of this sort may be succinctly formulated in integer programming terms" (Miller, Tucker, & Zemlin, 1960, p. 329). It is interesting to read this with the knowledge we now know, because one excellent algorithm combines the b&b and cutting planes to solve the TSP. Thus GC on their own are not very efficient but will be found later that when combined with other IP techniques strengthen those techniques significantly.

In the 1960s heuristics started to appear for the TSP. A heuristic is an estimate of the solution that gets close to the optimal solution. Karg and Thompson in 1964 came up with a heuristic that starts with two random cities, and inserts the next city in away to make the tour as small as possible. Thus it gives a solution to the TSP that can be found in linear time but yet not guaranteed to be optimal:

> In this paper we shall discuss a method, suitable for electronic computers, that has proved capable of quickly obtaining solutions for problems having about 60 cities or less in symmetric and some nonsymmetric problems. Although the code does not guarantee finding the optimum tour, it can be used over and over several times and in various ways to get a probabilistic idea of how good the best answer found is relative to the set of observed answers. (Karg & Thompson, 1964, p. 226)

Two other heuristic worth noting would be S. Lin with a 3 –opt algorithm and S. Reiter and G. Sherman with a Local Search heuristic: (http://www.tsp.gatech.edu/history/biblio/1960.html)

> In the 1970s Saman Hong wrote a Doctoral Thesis in which this is said.
>
> Hong's thesis was written under the supervision of M. Bellmore. His work is the most significant (computational) contribution to the linear programming approach to the TSP since the original paper of Dantzig, Fulkerson, and Johnson [1954]. The algorithm presented here goes a long way towards automating Dantzig, Fulkerson, and Johnson's method. Hong uses a dual LP algorithm for solving the linear-programming relaxations; the Ford-Fulkerson max-flow algorithm for finding violated subtour inequalities; a heuristic for finding violated blossom inequalities; and a branch-and-bound scheme that includes the addition of subtour inequalities at the nodes of the branch-and-bound tree (such algorithms are now known as "branch-and-cut" (Padberg and Rinaldi [1991])). In short, Hong had most of the ingredients of the current generation of linear-programming based algorithms for the TSP. His computational tests were carried out on random Euclidean instances having up to 20 cities. On the 60 instances that he tests, 59 were solved without branching and the remaining instance required a single branch. Larger instances were not tested due to difficulties with his LP solver. (http://www.tsp.gatech.edu/history/biblio/1970.html)

Thus as we get to the 1970s we see the main ingredients for solutions of the TSP.

Another interesting solution in the 1970s was by Held and Karp. They related the TSP to minimum spanning trees. The defined what they called a 1 – tree and showed that

the TSP is in fact a 1 – tree. "A 1 – tree consist of a tree on the vertex set {2,3,…,n}, together with two distinct edges at vertex 1. Thus, a 1 – tree has a single cycle, this cycle contains vertex 1, and vertex 1 always has degree two … A 1 – tree can be found by constructing a minimum spanning tree on the vertex set {2, … , n} and then adjoining two edges of lowest weight at vertex 1" (Held & Karp, 1970, p. 1139). Held and Carp (1970) made three observations that was the basis of their algorithm. "We observe that (i) a tour is precisely a 1 – tree in which each vertex has degree 2, (ii) a minimum 1 – tree is easy to compute, and (iii) the transformation on 'intercity distances' $c_{ij} \longrightarrow c_{ij} + \pi_i + \pi_j$ leaves the traveling-salesman problem invariant but changes the minimum 1- tree" (p. 1138). Held & Carp used a method in their paper in which Wolsey calls the "Lagrangian Dual." Wolsey (1998) says about their paper "The successful solution of what were at the time very large TSPs made the approach popular" (p. 180). Wolsey shows that the TSP can be relaxed into a 1-tree by a Lagrangian Relaxation. He then goes on in his text to show how to use the process of using the Lagrangian dual to solve the TSP. He uses the example of Held & Carp to show that it is an effective way to solve the TSP and thus is a good procedure. Thus Held and Karp used Minimum Spanning Trees and an early formulation of the Lagrangian dual to solve the TSP easily.

Everything before the 1980s laid the groundwork for the algorithms that are available today. With computers becoming more powerful and algorithms being tweaked to become more efficient higher and higher cases of the TSP are able to be solved and proved optimal. In the 1980s a new field was being formed. This was metaheuristics. A metaheuristic is designed to guide a regular heuristic and help it overcome local optimum. "A metaheuristic is a general solution method that provides both a general

structure and strategy guidelines for developing a specific heuristic method to fit a particular kind of problem. Metaheuristics have become one of the most important techniques in the toolkit of OR practitioners" (Hillier & Lieberman, 2005, p. 617). In the 1980s this field was not existence, though Fred Glover was credited with the first metaheuristic, the Tabu Search. He published a two part series on the Tabu search in the ORSA Journal on computing. As stated by him "Tabu search is a strategy for solving combinatorial optimization problems whose applications range from graph theory and matroid settings to general pure and mixed integer programming problems. It is an adaptive procedure with the ability to make use of many other methods, such as linear programming algorithms and specialized heuristics, which it directs to overcome the limitations of local optimality" (Glover, 1989, p. 190). The Tabu Search for the TSP runs off of the Nearest Neighbor (NN) heuristic. The Nearest Neighbor heuristic also referred to as a 2-opt algorithm, looks to swap two cities in order to make the solution smaller. The NN often hits a local optimum before getting to the true optimal solution. The idea of the Tabu search is to keep the NN algorithm running when it hits a local optimum. The basic concept is that every step along the way the Tabu search keeps a list of already made moves and does not allow those moves to be made again. At each step of the way it will either make the best available switch or the switch that makes the least increase if no decreases are available. To keep from cycling back to already found optimums it keeps track of switches it has already made and adds them to a list called a "tabu list" in which moves on this list are "tabu" or cannot be made. The process is run until there are no more switches to be made, or a certain amount of iterations has happened. Thus the

Tabu Search was applied to the TSP to show its effectiveness in finding a solution. It proved to have incredible results.

Kirkpatrick, Gelatt, & Vecchi (1983) wrote an article comparing "statistical mechanics (the behavior of systems with many degrees of freedom in thermal equilibrium at a finite temperature) and multivariate or combinatorial optimization (finding the minimum of a given function depending on many parameters)" (p. 671). It is interesting to note Kirkpatrick et al.'s (1983) comment, "Of classic optimization problems, the traveling salesman problem has received the most intensive study. To test the power of simulated annealing, we used the algorithm on traveling salesman problems with as many as several thousand cities." (p. 671) Simulated Annealing (SA) is similar to the Tabu Search method. It guides the descent algorithm and helps it to overcome local minimum. SA mimics a natural process: "The objective of physical annealing is to produce low-energy states of a solid in a heat bath. Annealing has two steps: (1) The temperature of the heat bath is raised to just below the boiling point of the material when the particles are disorganized and the energy of the system is high. (2) The temperature is carefully lowered until the particles of the liquid arrange themselves into a more orderly state of minimum energy. It is a natural minimization process" (Albright, 2007, p. 38). In the SA algorithm, it generates a neighbor. The algorithm accepts the neighbor under two conditions either the neighbor produces a solution that is smaller then the current best, or with a certain probability. Albright (2007) defines the probability as:

$$P\{\text{accept } j\} = \begin{cases} 1 & \text{if } f(j) \le f(i) \\ \exp\left(\frac{f(i)-f(j)}{c_k}\right) & \text{if } f(j) > f(i) \end{cases}$$ (p. 39)

Thus is the basic of the SA algorithm. Later on in their paper, Kirkpatrick et al. apply it to a very complicated problem on the physical design of computers. So to gain

credibility for the algorithm Kirkpatrick et al. apply it to the TSP and show SA can get close to optimal if not optimal on TSP instances that are proved optimal. Thus SA gained a lot of credibility. The SA algorithm is popular today also because of its ease to apply it to other problems: "Implementing the appropriate Metropolis algorithm to simulate annealing of a combinatorial optimization is straightforward and easily extended to new problems" (Kirkpatrick, Gelatt, & Vecchi, 1983, p. 679).

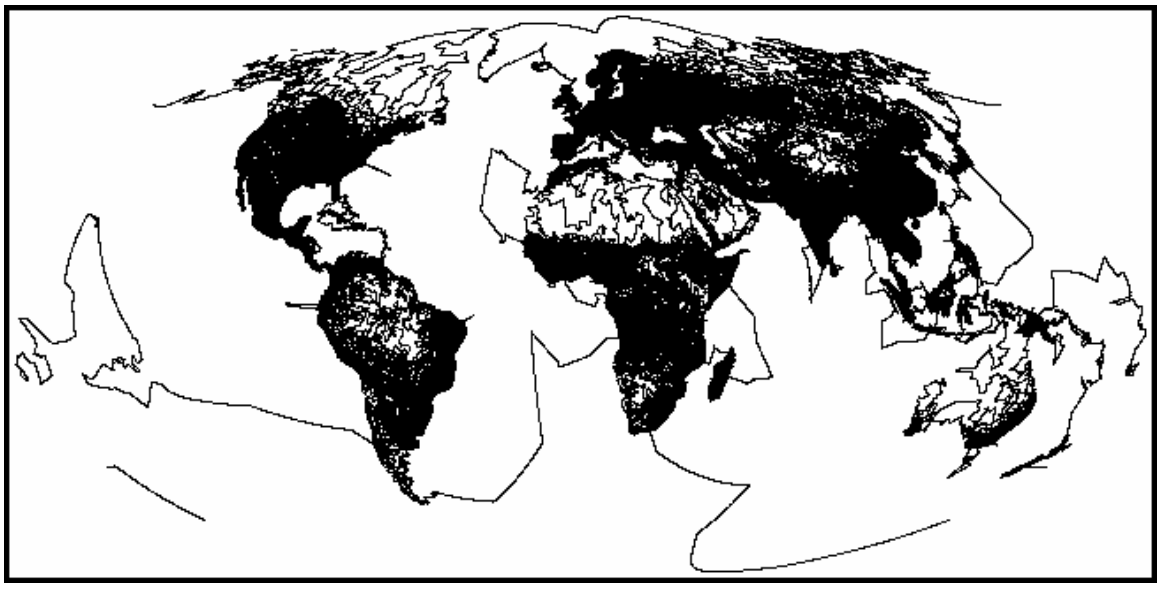
In 1995 D. Applegate, R. Bixby, V. Chvátal, and W. Cook published a paper on the TSP. Applegate et al. have been the authority on the TSP since then. They have developed software base off of their paper called Concorde. Concorde is open source software available for free on http://www.tsp.gatech.edu/concorde/index.html. On this website they describe the purpose of the Concorde:

> Concorde is a computer code for the symmetric traveling salesman problem (TSP) and some related network optimization problems. The code is written in the ANSI C programming language and it is available for academic research use; for other uses, contact William Cook for licensing options.
>
> Concorde's TSP solver has been used to obtain the optimal solutions to 107 of the 110 TSPLIB instances; the largest having 15,112 cities.
>
> The Concorde callable library includes over 700 functions permitting users to create specialized codes for TSP-like problems. All Concorde functions are thread-safe for programming in shared-memory parallel environments; the main TSP solver includes code for running over networks of UNIX workstations.

> Concorde now supports the QSopt linear programming solver. Executable versions of concorde with qsopt for Linux and Solaris are now available (http://www.tsp.gatech.edu/concorde/index.html, 2007)

Thus with Concorde it is feasible to solve very large instances of the TSP. Thus Concorde, or a variation of it has been used to solve the largest TSPs solved.

There still are TSPs that are unsolved. One of those is the world problem. The world problem is a TSP formulated for the world in which in contains 1,904,711 cities. It can be found on http://www.tsp.gatech.edu/world/index.html. According to the website, Keld Helsguan currently has the best solution, with the tour length of 7,516,024,785 units. He used a variation of a heuristic that he had created. The best lower bound for the TSP is 7,512,082,035 units created from Concorde code using the CPLEX solver. (http://www.tsp.gatech.edu/world/index.html, 2007) Below is a picture of Helsguan's tour around the world.



"Go to the ant, O sluggard; consider her ways, and be wise." (Proverbs 6:6, ESV) In 1999 just that was done and a new metaheuristic emerged. It was called Ant Colony

Optimization (ACO) and was originally formulated by Dorigo, Di Caro and Gambardella. (http://www.aco-metaheuristic.org/, 2006) Due to the current popularity of the TSP ACO was used to find solutions to the TSP to gain credibility:

> The traveling salesman problem is an extensively studied problem in the literature and for a long time has attracted a considerable amount of research effort. The TSP also plays an important role in ACO research: the first ACO algorithm, called Ant System, as well as many of the ACO algorithms proposed subsequently, was first tested on the TSP.
>
> There are several reasons for the choice of the TSP as the problem to explain the working of ACO algorithms: it is an important NP-hard optimization problem that arises in several applications; it is a problem to which ACO algorithms are easily applied; it is easily understandable, so that the algorithm behavior is not obscured by too many technicalities; and it is a standard test bed for new algorithmic ideas— a good performance on the TSP is often taken as a proof of their usefulness. Additionally, the history of ACO shows that very often the most efficient ACO algorithms for the TSP were also found to be among the most efficient ones for a wide variety of other problems. (Dorigo & Stèutzle, 2004, p. 65)

ACO started by studying ants. They realized that ants have a natural way of optimizing different procedures. One way was that of the shortest path to food. Through careful studying of ants a computerized version of these ants were formed and used to find the shortest path between points. ACO is fairly straightforward to apply to many different instances. Thus with successful application to the TSP, ease of application to other

problems, and a vast amount of unknowns to be explored ACO is growing in interest among mathematical circles. Like many Metaheuristics it is not know exactly why it works so well, it is just know that it works. Many mathematicians are trying study and explain the reasons why it is so effective. Thus the latest research on the TSP is the ACO algorithm and how to refine it.

The TSP has hit a pseudo-stopping point. While there are still unsolved instances of the TSP and the TSP has no polynomial time bound algorithm to solve every instance, Concorde has taken a lot of the interest away from creating that software. Also dissuading mathematicians from attempting is the fact that the TSP is NP-hard and many mathematicians believe that it is impossible to find such an algorithm. While the TSP still has a small interest in solving new instances, such as the world problem, a large amount of the current research is done by mathematicians who are not seeking an answer to the TSP, but validity in a new approach or a new heuristic that they plan on applying to another area. With so many instances of the TSP solved, and with the past and current popularity of the problem, the TSP is a fast and easy way for a mathematician to gain validity of a procedure. New algorithms find and prove a solution optimal faster, or heuristics gain close of not optimal solutions using the already proved optimal solutions as their base for their claim. Thus one looking to advance the TSP should focus on making current algorithms faster, or inventing a new algorithm to shake up the mathematical world.

The Traveling Salesman Problem has a rich history in the past 50 years. It is well documented and well researched among mathematicians. Though the true history traces back to the 18th century much of the work on it did not appear until the 1930's. Credit

and thanks must be given to Hassler Whitney for posing the question and Merrill Flood for generating a lot of interest in it. Dantzig, Fulkerson, and Johnson were the first to have any real breakthroughs in the solution. Proctor and Gamble increased public interest in the solution with the addition of an award to those who studied it and this had great influence on those who studied it. Saman Hong is to be credited for coming up with the basis of what all of the modern day solutions have, and can be credited for algorithms such as the Branch and Cut algorithm. D. Applegate, R. Bixby, V. Chvátal, and W. Cook have kept current interest in the TSP and have developed Concorde in which many instances of the TSP can be easily solved from a home computer. They also did the public a service by making it open source, free and easily accessible to anybody with internet access. Interest in the TSP has recently shifted from trying to pioneer new solutions, to making old algorithms better and more efficient and has become a testing ground for many IP, combinatorial optimization, and heuristic algorithms. The field of Metaheuristics can be easily seen from the view of the TSP as every efficient Metaheuristic has been applied to the TSP. Three main Metaheuristics are the Tabu Search created by Fred Glover, Simulated Annealing by Kirkpatrick, Gelatt, & Vecchi, and Ant Colony Optimization by Dorigo, Di Caro and Gambardella. It is interesting to note that these metaheuristics while designed for a specific problem all used the TSP to prove that it was an effective algorithm and used the TSP to show an easy way to implement the algorithm. These algorithms have been applied to a vast amount of different applications because they are designed to be flexible and easy to apply to different situations. The TSP now remains as an unsolved problem, still keeping a few

mathematicians interest, but now serves the mathematical world in a different light, that light being an easy testing ground to new theorems and improved heuristics

References

Albright, B. (2007). An introduction to simulated annealing. *The College Mathematics Journal*, *38*, 37-42.

Balaprakash, P., & Montes De Oca, M. A. (Eds.). (2006, November 06). *Ant Colony Optimization*. Retrieved March 24, 2007, from http://www.aco-metaheuristic.org/

Bellman, R. (1962). Dynamic programming treatment of the traveling salesman problem. *Journal of the ACM (JACM), 9*, 61-63. Retrieved March 9, 2007, from http://doi.acm.org/10.1145/321105.321111

Bellman, R. (1984). *Eye of the hurricane: An autobiography*. Singapore: World Scientific.

Crowes, G. A. (1958). A method for solving traveling-salesman problems. *Operations Research*, *6*, 791.

Dantzig, G. B., Fulkerson, D. R., & Johnson, S. M. (1959). On a linear-programming, combinatorial approach to the traveling-salesman problem. *Operations Research*, *7*, 58. Retrieved March 8, 2007, from http://search.ebscohost.com/login.aspx?direct=true&db=buh&AN=7685485&site=ehost

Dantzig, G., Fulkerson, R., & Johnson, S. (1954). Solution of a large traveling-salesman problem. *Journal of the Operations Research Society of America*, *2*, 393.

Dorigo, M., & Stèutzle, T. (2004). *Ant colony optimization*. Cambridge, MA: MIT Press.

Eiselt, H. A., & Sandblom, C. (2000). *Integer programming and network models*. New York: Springer.

Flood, M. M. (1956). The traveling-salesman problem. *Operations Research, 4*, 61. Retrieved March 8, 2007, from http://search.ebscohost.com/login.aspx?direct=true&db=buh&AN=7678846&site=ehost

Foulds, L. R. (1984). *Combinatorial optimization for undergraduates*. New York: Springer-Verlag.

Glover, F. (1989). Tabu Search - Part I. *ORSA Journal on Computing, 1*, 190-206. Retrieved December 15, 2006, from EBSCOHost database.

Held, M., & Karp, R. M. (1970). Traveling–salesman problem and minimum spanning trees. *Operations Research, 18*, 1138. Retrieved from http://search.ebscohost.com/login.aspx?direct=true&db=buh&AN=8990999&site=ehost

Hillier, F. S., & Lieberman, G. J. (2005). *Introduction to operations research* (8th ed.). Boston: McGraw-Hill Higher Education.

Karg, R. L., & Thompson, G. L. (1964). A heuristic approach to solving traveling salesman problems. *Management Science, 10*, 225. Retrieved from http://search.ebscohost.com/login.aspx?direct=true&db=buh&AN=7438034&site=ehost

Kirkpatrick, S., Gelatt, C. D., & Vecchi, M. P. (1983). Optimization by simulated annealing. *Science, 220*, 671-680. Retrieved March 9, 2007, from http://links.jstor.org/sici?sici=0036-8075%2819830513%293%3A220%3A4598%3C671%3AOBSA%3E2.0.CO%3B2-8

Little, J. D., Murty, K. G., Sweeney, D. W., & Karel, C. (1963). An algorithm for the traveling salesman problem. *Operations Research, 11*, 972. Retrieved from http://search.ebscohost.com/login.aspx?direct=true&db=buh&AN=7684068&site=ehost

Miller, C. E., Tucker, A. W., & Zemlin, R. A. (1960). Integer programming formulation of traveling salesman problems. *Journal of the ACM (JACM), 7*, 326-329. Retrieved March 9, 2007, from http://doi.acm.org/10.1145/321043.321046

Schrijver, A. (2005). On the history of combinatorial optimization. In R. Weismantel, G. L. Nemhauser, & K. Aardal (Eds.), *Handbook of discrete optimization* (pp. 1-68). Amsterdam: Elsevier.

Traveling salesman problem. (2007, January). Retrieved November 11, 2006, from http://www.tsp.gatech.edu/index.html

Weisstein, E. W. (2003, December 24). *"Icosian Game" from MathWorld--A Wolfram Web Resource*. Retrieved March 4, 2007, from http://mathworld.wolfram.com/IcosianGame.html

Wolsey, L. A. (1998). *Integer programming*. New York: Wiley.