The Traveling Salesperson Problem Under Various Conditions

Bradley W. Mazurek

Supervisor: J. Mark Keil

Abstract

In this paper we examine algorithms and heuristics for the Euclidean traveling salesperson problem. The traveling salesperson problem is to find the shortest tour of n points in a plane such that the tour visits every vertex and returns to it's point of origin. Of particular interest are heuristics and algorithms that provide performance guarantees on solution quality for this $\mathcal{NP}$-complete problem. Finally, we examine various aspects of this problem, identifying characteristics and structures which may prove useful to further research.

1 Introduction

The traveling salesperson problem, or TSP, is among the most famous problems in computer science. Simply stated the problem is as follows: given n points in a plane, find a tour of these n points such that each point is visited once, the tour concludes at the vertex where it began, and the length of this tour is minimized. Although the problem is very simple conceptually, in terms of complexity it is among the most difficult to solve. In fact, the TSP is $\mathcal{NP}$-complete.

Because the TSP is $\mathcal{NP}$-complete and because of the common belief that $\mathcal{P} \neq \mathcal{NP}$, there seems to be little hope of deriving a polynomial time algorithm to solve the TSP in the general case. Therefore, current research on the TSP concentrates on identifying special conditions under which the problem can be solved efficiently or finding heuristics that provide guaranteed bounds on solution quality.

In Section 2 we examine what is known about the TSP. Heuristics providing probabilistically and deterministically good solutions are studied. Finally, we examine some TSP problem variations for which guaranteed approximation algorithms are available. In Section 3 we reveal the path our research took while examining the TSP.

2 Previous Work

For many years the TSP has undergone an enormous amount of research. The focus of this research has been the development of algorithms to produce TSP tours. These algorithms can be divided into two veins: those which provide empirically good solutions, but with no performance guarantees and those algorithms that produce solutions that have proven bounds on solution quality.

In Section 2.1 we examine algorithms which traditionally perform well, despite the fact that little can be said about the quality of the solutions produced. Then, in Section 2.2, we investigate methods for producing tours that are of a proven, guaranteed quality.

2.1 Probabilistically Good Heuristics

Many heuristics for the TSP provide solutions which are quite good. Bentley[4] provided an experimental study of many TSP heuristics. Many of these heuristics are relatively rigid and blind. As a result, examples can be concocted for which a heuristic does arbitrarily poor. This lack of consistency (and proveability) can be troublesome from a theoretical perspective.

2.1.1 Branch and Bound

Branch and bound is basically a brute force search technique. It searches a tree representing possible solutions. The algorithm maintains a value for the best solution found thus far (the bound). If during it's progression down a particular branch of the tree, the weight of the current partial solution becomes larger than the bounding value, the search of that branch of the state space tree is abandoned, as all solutions down that branch are guaranteed to be worse than the current best solution.

Although the algorithm is quite simple and straightforward, branch and bound algorithms may require an exponential amount of time to complete. Empirically, however, a solution obtained from a branch and bound algorithm can be quite good, given a sufficient amount of search time.

A variation of the Branch and Bound algorithm has been examined by Jünger, Reinelt and Thienel[14]. Their implementation is significantly more complicated than traditional branch and bound. The user specifies an error bound, $\epsilon\%$ of optimal, within which a solution is required. Their algorithm apparently performs well, but the time required to obtain the desired solution becomes exponential as $\epsilon \rightarrow 0$.

2.1.2 Simulated Annealing

Simulated annealing is a technique for arriving at a fairly good quality solution for combinatorial optimization problems such as the TSP. Annealing is the metallurgical process of heating an object, then allowing it to cool at a specific rate. As the object cools the object can be manipulated to allow molecules to settle into configurations thereby making the object exhibit new characteristics, such as becoming less brittle.

In a similar sense, simulated annealing *heats* up the initial solution for a problem such as the TSP. This solution is then slowly cooled. As it cools, the algorithm begins making changes to the current solution. When the solution is hot, the algorithm makes fairly broad changes to the existing tour. However, as the temperature of the solution drops, the changes made become less significant. That is, the changes perturb the solution less and less.

One can think of the TSP solution space as a mountainous landscape. Short tours are like valleys between mountains, the long tours are the mountain tops. Our initial solution starts with high energy and bounces around quite randomly. As the solution cools, the solution begins to settle into the valleys of the solution space. These effects allow the solution to jump into good solutions, and can avoid local minima by jumping out of one valley and into another adjacent valley.

Although simulated annealing does not provide any sort of performance guarantees about the solutions produced, it does very well in practice, generating solutions that are usually within a few percent of optimal.

From an application standpoint, producing solutions that are usually close to optimal is acceptable. However, from a theoretical standpoint, algorithms of this type do not provide a rigorous foundation from which to proceed. With a foundation built solidly in proof-oriented theoretical computer science and mathematics, we would be in a much better position to extend our knowledge, learning more about the fine line that *appears* to separate $\mathcal{P}$ from $\mathcal{NP}$.

In the next section we examine methods used to produce algorithms for the TSP which are of a much more solid nature. These algorithms have characteristics that can be analyzed, providing guarantees about the solutions obtained.

2.2 Guaranteed Approximation Heuristics

Heuristics that provide performance guarantees can be quite enticing. These guarantees provide bounds for not only how "poor" a solution can be, but the algorithms can also be analyzed to determine time requirements. As one would expect, these guarantees come at a price: the solutions are often not as good as those obtained by probabilistically good heuristics.

The *nearest neighbor heuristic*[3] begins with a partial solution consisting of a single point. Iteratively, the algorithm adds the point closest to the end of the path to the partial solution. This now becomes the partial solution. After all points have been included, the path is closed to become a tour. Solutions produced by this algorithm were later shown to be at most $\frac{\lceil \log n \rceil + 1}{2}$ times optimal[22], and can be found in $O(n^2)$ time.

The ability to bound the worst case solution produced by an algorithm is an important characteristic. However, the bound for the nearest neighbor heuristic is not encouraging. Notice that this algorithm did not exploit any geometric information about the set of points. In other words, the nearest neighbor heuristic would work just as well in a graph-theoretic setting. In order to do better, it would be logical to attempt to exploit the geometric nature of the problem. In the following sections we examine two heuristics for the general case of the TSP, while providing impressive bounds on solution quality.

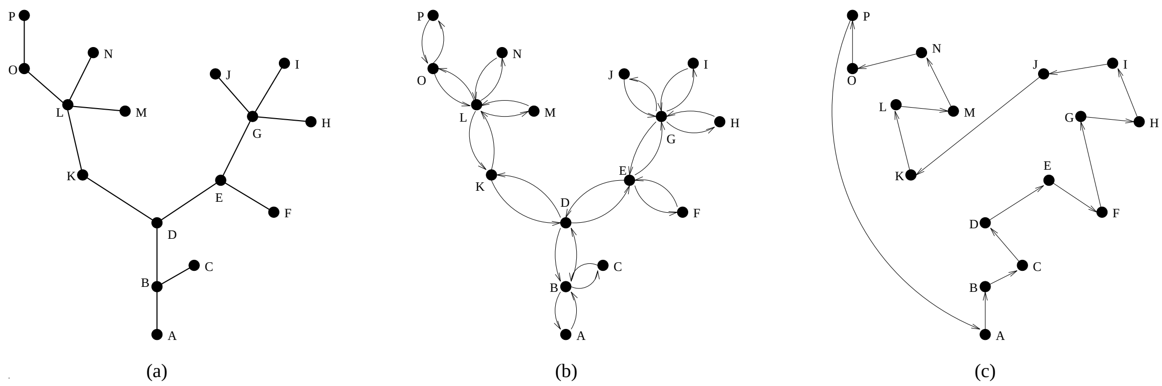

Figure 1: Minimum Spanning Tree and tour generation

2.2.1 Minimum Spanning Tree Heuristics

The *Minimum Spanning Tree* (MST) of a set of points is an important structure used in many algorithms. In the context of the TSP, the MST is often a principal building block when deriving a tour providing performance guarantees.

Notice that the MST consists of $(n-1)$ edges that form a component connecting all n points. In addition, if an edge is deleted from an optimal TSP tour, we are left with a spanning tree of the points. Therefore, it is clear that the optimal TSP tour is strictly longer than the MST.

Traveling the MST in a depth first fashion and returning to the starting point would allow us to visit every point and forms a tour. This tour would travel every edge of the MST twice. Since the length of the MST is shorter than that of the optimal TSP tour, it becomes clear that a double tour of the MST will be less than twice the length of the optimal TSP tour. This result was obtained by Rosenkrantz, Stearn and Lewis [21]. The computational complexity of this algorithm is the same as finding the MST, $O(n \log n)$.

For example, Figure 1(a) shows a set of points and the MST of those points. Figure 1(b) shows the TSP tour resulting from the depth first traversal of the MST in (a).

Improvements to the resulting tour are quite obvious. In Figure 1(b) we can see the depth first traversal of the points in the MST results in visiting each point twice. To shorten the tour, we can stipulate that during the tour construction, if we are about to visit a point which is already part of the tour, we instead skip ahead and visit the next unvisited point in the depth first MST traversal. For example, in Figure 1(c) we see this results in a significant improvement when applied to the tour produced in (b).

Other algorithms have achieved a similar guaranteed bound of 2 times optimal.[22] Upon close examination however, these algorithms all effectively generate a MST, and their performance guarantees result from the doubling of that MST. Empirically, these algorithms do better than the standard MST algorithm, but the worst-case behavior still remains. Optimizations to the MST do not seem to provide any further guarantees.

2.2.2 Christofides' Algorithm

In order to lower the 2 times optimal guaranteed bound of the MST double tour, we require new insight. Such insight was found in the study of the *Eulerian graph*. An Eulerian graph is a connected graph in which all vertices have even degree. Eulerian graphs contain a tour passing through every edge exactly once. This tour is called an Eulerian tour and can be found in $O(n)$ time given an Eulerian graph.[2]

Finding a minimum length Eulerian graph connecting all cities must have the same length as the minimum traveling salesman tour, and is therefore just as difficult as the TSP. Christofides [9] is responsible for the best approximation currently available for generating the minimum length Eulerian graph of a set of points. This method uses the MST and minimum weight matchings.

Christofides' method is as follows. Given a MST, T, we augment T to form an Eulerian graph. This can be done by simply ensuring that all vertices have even degree. That is, add edges to T until all vertices are of even degree. Obviously, T will contain an even number of these odd-degree vertices. Therefore, a maximum-cardinality minimum-weight matching on odd-degree vertices will allow us to form an Eulerian graph from T with the smallest possible increase in length. The length of this minimum matching will be at most one half the length of the optimal TSP tour. This is combined with the MST, which, as was noted earlier, is strictly shorter than the traveling salesperson tour. Hence, a tour generated using Christofides' algorithm will be at most $\frac{3}{2}$ times optimal. This algorithm has a time complexity of $O(n^{2.5}(\log n)^4)$, which is the cost of the Euclidean minimum weight matching due to Vaidya[28].

As was the case for the tour created using the MST heuristic, the tour produced by Christofides' algorithm contains obvious inefficiences. These can be removed using the shortcutting routine discussed for the MST heuristic.

In Figure 2 we see Christofides' algorithm process the same set of points as in Figure 1. Initially, the MST is found (Figure 2(a)) from which a minimum weight maximum matching of the odd-degree vertices augments the MST forming the Eulerian graph found in Figure 2(b). Figure 2(c) shows the resulting Euler tour. Shortcutting this Euler tour results in the final tour shown in Figure 2(d).

2.3 Problem Modifications

Although the TSP is $\mathcal{NP}$-complete, it remains an active area of research. Current research often focuses not on the TSP itself, but on variations of the problem. Researchers attempt to modify the problem and/or impose conditions on the problem in an attempt to derive polynomial time algorithms.

One very common modification made to the TSP is to modify the goal of the problem. As we will see, the problem is usually of the same general flavour, but the results can often be quite different.

2.3.1 The Minimum Latency Problem

Imagine a repairman with a list of clients whom he wishes to visit. In order to keep the majority of customers happy, the repairman wishes to visit them in an order that will minimize the average waiting time experienced by the customer. This problem is an instance of the time dependent traveling salesperson problem known as the *minimum latency problem*(MLT)[7], but has also been referred to

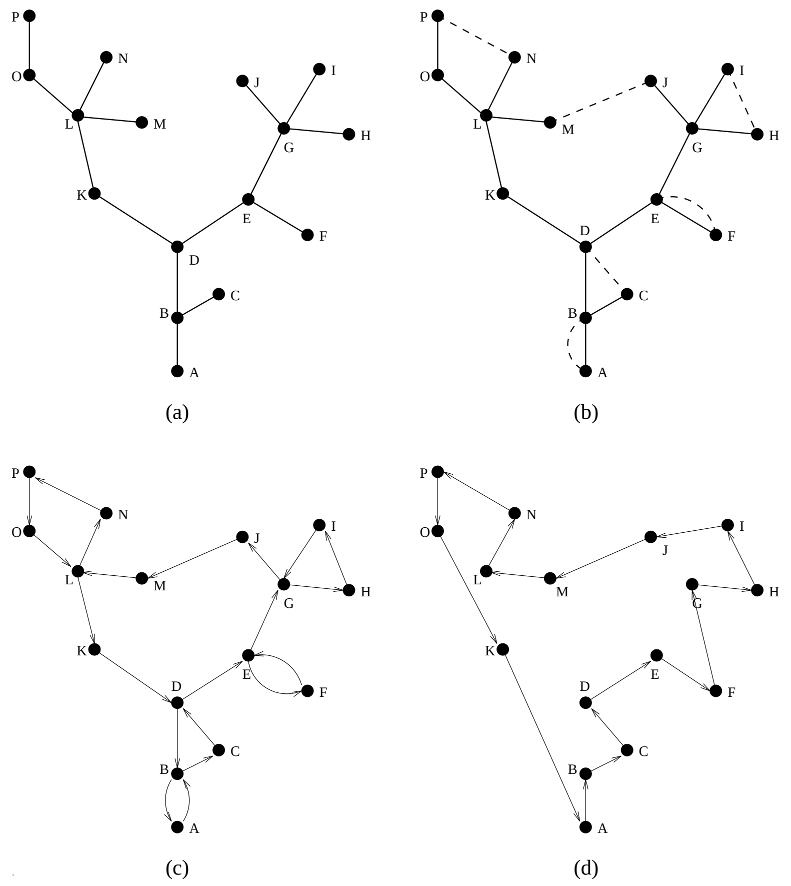

Figure 2: Progress of Christofides' Algorithm

in the literature as the *cumulative traveling salesman problem*[6], the *deliveryman problem*[17] and the *traveling repairman problem*[1, 27]. It has been shown that the MLT is at least as hard as the TSP[7].

Despite this Blum et al present numerous results regarding the MLT. They present an exact solution applicable if the point sets are vertices of a tree where all edges have length 1. Dynamic programming techniques are used when the points are on the real line, or are vertices of a diameter 3 tree, obtaining an optimal solution to the MLT in $O(n^2)$ time.

In the MLT's general case, Blum et al derive a 72-approximation, and cite personal communication with M. Goemans and J. Kleinberg who have apparently derived an approximation of just under 29 times optimal[7].

2.3.2 Many Salespeople Problem

Also known as the k-TSP, this generalization of the TSP is formulated as follows: Given a set of n points (cities), a starting point (say, city 1) and k salespeople to visit the n cities, form k tours such that each of the tours contains city 1, each of the remaining $n-1$ cities is visited by a tour, and the length of the maximum length subtour of this collection is minimized.

An algorithm by Frederickson, Hecht and Kim[13] obtains an approximation to the k-TSP by partitioning a normal TSP tour into k pieces. Using Christofides' algorithm for the initial tour for their partitioning algorithm yields solutions that are guaranteed to be within $(\frac{5}{2} - \frac{1}{k})$ times optimal.

2.3.3 The Stacker Crane Problem

The stacker crane problem is yet another generalization of the TSP. For this problem, the tour must traverse a given set of edges and traverse these edges in a specified direction. An instance of the problem consists of a set of points and a set of arcs, A. The problem can be thought of as the route of a delivery truck picking up packages and delivering them one at a time.

Frederickson, Hecht and Kim[13] provide two heuristics for this problem. Each provide respectable performance guarantees. If both heuristics are applied independently and the best resulting tour taken, an interesting result can be shown: a stronger performance guarantee than either individual heuristics will result.

The first heuristic, called the *large arcs algorithm* has a time complexity of $O(|A|^3)$. The tour generated for a particular instance, I, will have a length, $LA(I)$, satisfying:

$$LA(I) \leq 3 \cdot OPT(I) - 2 \cdot Length(A)$$

where $Length(A)$ is the sum of the arc lengths in A. The second heuristic, called the *small arcs algorithm* also has a time complexity of $O(|A|^3)$ and produces a tour, $SA(I)$, satisfying:

$$SA(I) \leq \frac{3}{2} \cdot OPT(I) + \frac{1}{2} \cdot Length(A)$$

However, by performing both algorithms and choosing the best tour we can obtain a performance guarantee of $\frac{9}{5}$ times optimal.

2.4 Point Set Restrictions

To cope with the $\mathcal{NP}$-complete nature of the TSP, researchers often impose regularities or conditions upon the placement of the points. This rigid structure often allows polynomial time algorithms to be developed.

2.4.1 N-line Traveling Salesperson Problem

Rote[23] has developed a polynomial time algorithm when the n points lie on N "almost parallel" line segments, where N is a small constant. The method involves dynamic programming, and results in an $O(n^N)$ algorithm, hence the requirement that N be a small constant. The algorithm works for parallel line segments, but has been extended to include line segments adhereing to the following conditions:

- Segments may not be perpendicular to the x-axis. (However, the x-axis line may be chosen arbitrarily.)
- For every pair of non-parallel segments, when the intersection point of their carrying lines is projected onto the x-axis, this point lies outside (*i.e.* never on or between) the projections of the line segments onto the x-axis.

For example, in Figure 3[23] we see three line segments which fulfill these conditions. Given n points on these three line segments, we could obtain the optimal TSP tour in $O(n^N) = O(n^3)$ time using dynamic programming.

3 Results

During our examination of research on the TSP we examined many avenues in an attempt to produce results of our own. Some of these ideas are interesting, and warrant further examination.

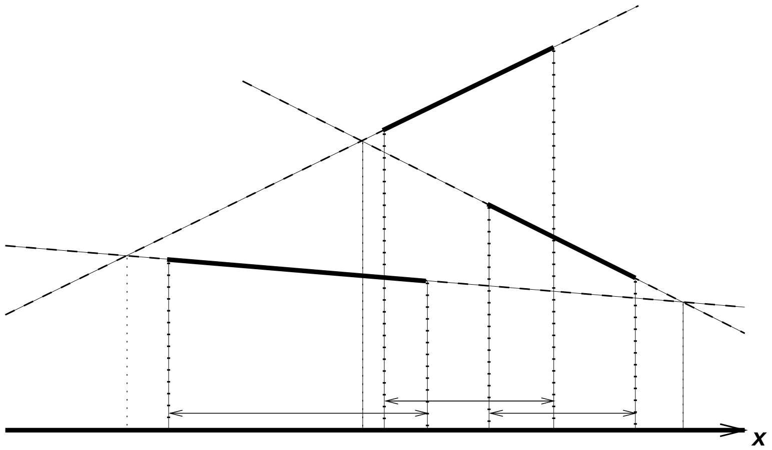

Figure 3: Three allowable line segments

3.1 Geometric Investigation

Upon examining the double MST tour, the method used to produce shortcuts yields improvements to the tour, but the question arises as to how much they can improve it. In Figure 4(a) we see that a particular shortcut may result in no savings, even though we obtain an optimal TSP tour. Figure 4(b) shows another MST shortcut resulting in an optimal tour. In (a) very little is saved over the double MST tour, whereas in (b) the result is cutting the tour length almost in half. Thus, even when the shortcut results in an optimal tour, it is difficult to conclude how much shortcutting will save.

In fact, examples can be concocted to show arbitrarily good or bad savings when shortcutting the MST tree. In Figure 4(a) and (b) we saw optimal tours being produced from shortcuts. In Figure 4(c)[16] we see a MST which, even after shortcuts, has a length approaching twice the optimal tour length.

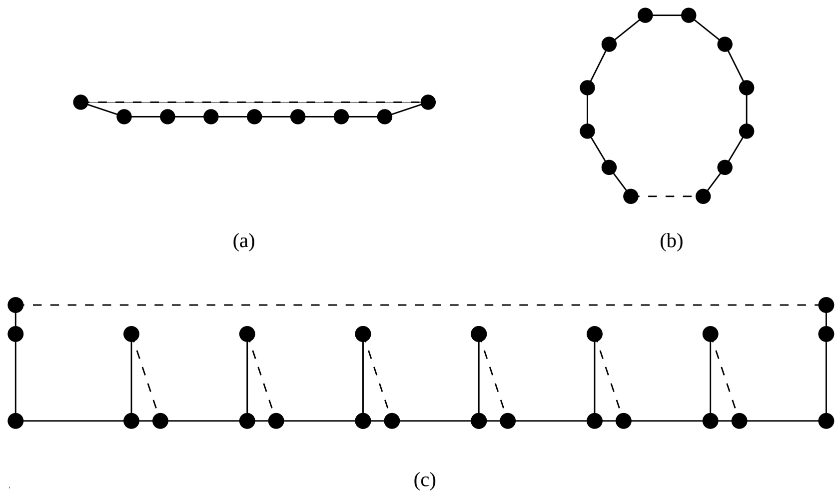

Figure 4: Quality of tour shortcuts

To this end, the properties of the MST were examined closely. During the examination of Christofides' algorithm, we became aware that finding the best set of shortcuts while converting the MST to an Eulerian graph is $\mathcal{NP}$-hard[19], so we can expect to do no better in our tour shortcutting.

Because the MST has become such a useful structure when examining the TSP, we thought an examination of related structures might prove useful. The *Delaunay Triangulation* (DT) is an important structure in computational geometry. The dual of the *Voronoi Diagram*, the DT contains much of the proximity information available for a given set of points. As such, the MST is a subset of the edges in the DT. Related to the Delaunay triangulation and the MST are *Gabriel graphs* (GG) and *relative neighborhood graphs* (RNG). The relationship between these structures for a set of points is as follows:

$$\text{MST} \subseteq \text{RNG} \subseteq \text{GG} \subseteq \text{DT}$$

These supersets of the MST were examined to varying degrees, but the results of Kantabutra[15] and Dillencourt[10, 11] were not encouraging. In [15] Kantabutra showed by way of a counterexample that an optimal TSP tour is not always a subgraph of the Delaunay triangulation. In fact, his counterexample is not even Hamiltonian. The counterexample, however, was a degenerate case of the Delaunay triangulation. (Note: The Delaunay triangulation is not a true triangulation. Degenerate cases exist where k-gons are part of the "triangulation", for $k \geq 4$.) Dillencourt[11] further strengthened this by providing an example of a non-degenerate DT that was also non-Hamiltonian.

In addition, Dillencourt[10] proved, also by means of a counterexample, that even when the DT is Hamiltonian, the optimal TSP tour is not necessarily a subgraph of the DT. His counterexample did provide some solace however. The optimal TSP tour for his example was of length 35.41. The shortest tour through the DT had a length of 35.48. This suggests that the DT may be an excellent candidate for deriving guaranteed approximations to the TSP and warrants further examination.

3.2 Modifying Christofides' Algorithm

Christofides' algorithm provides the best quality guarantees for TSP solutions. However, the cost of the algorithm ($O(n^{2.5}(\log n)^4)$ time) makes it somewhat prohibitive. The high cost of the algorithm is due to the maximum-cardinality minimum-weight matching performed on the odd-degree vertices. So to obtain a faster TSP approximation, we simply need to perform matchings faster.

For practicality, the expensive matching portion of the algorithm is often replaced with faster approximations. This degrades the quality of the solution, but tours would be attainable much more quickly. For example, Supowit and Reingold[25] provide an algorithm for matching with a $0.707\sqrt{n}$ bound on worst case performance. Utilizing this matching algorithm would degrade Christofides' algorithm to producing solutions that are $1.414\sqrt{n}$ times optimal[26], yet would require only $O(n \log n)$ time.

Another alternative is to examine the location of the odd-degree vertices in the MST. If something can be realized about the locations of these points, there will probably be an algorithm that can use this fact to produce a maximum-cardinality minimum-weight matching faster than the general case algorithm.

One of the most promising algorithm/point set combinations of this type are the algorithms of Marcotte and Suri[18]. There the authors present an $O(n \log n)$ time algorithm for the matching problem when the points lie on a convex hull. The authors also present an $O(n \log^2 n)$ time algorithm for points on the boundary of a simple, non-convex polygon. There is one catch for the latter algorithm: Only points visible to each other can be matched. This concept of point visibility suggests methods employed in art gallery problems may be useful here.

Perhaps many of these ideas can be combined. Art gallery problems often utilize a triangulation which is used when *unrolling* polygons. The Delaunay triangulation may prove useful in such cases. However, the most difficult part of this problem will be producing a polygon that allows the matching algorithm of Marcotte and Suri to perform well. Finding a polygon that will lead to the minimum weight matching of the odd-degree vertices is probably $\mathcal{NP}$-hard. However, algorithms similar to Stewart's convex hull insertion procedure[24] for the TSP may prove useful in this case. Stewart's algorithm uses the convex hull as an initial tour. Points are inserted iteratively such that the point chosen for insertion at each iteration increases the length of the tour by the least amount.

Our research took us in many directions, often simultaneously. But because TSP results extend in so many directions, it was often difficult to decide where to proceed.

4 Conclusion

Although the traveling salesperson problem is a very old and deeply investigated problem in combinatorial optimization, research on this $\mathcal{NP}$-complete problem has not slowed. Recent research has tended to avoid the general case of the TSP, and instead focus on imposing conditions upon the problem or even modifying the problem goal in order to develop polynomial time algorithms or guaranteed approximations.

Approximation algorithms such as Christofides' heuristic piece together minimum spanning tree and matching algorithms. Therefore, to improve existing TSP approximation algorithms, one simply needs to improve existing algorithms for these components. The most promising of these are matching algorithms. The general matching algorithm of Vaidya[28] is computationally expensive and is therefore a likely candidate for improvements. Finding cases where matchings can be performed faster is a promising area for performing further research.

Despite the findings of Dillencourt[10, 11], another area that warrants further examination is a study of the geometric nature of the minimum spanning tree, the relative neighborhood graph, Gabriel graph, Gabriel triangulation and the Delaunay triangulation. Any information these studies reveals would likely impact the traveling salesperson problems.

References

[1] Afrati, F., Cosmadakis, S., Papadimitriou, C., Papageorgiou, G. and Papakostantinou, N. The Complexity of the Traveling Repairman Problem. *Informatique Theorique et Applications (Theoretical Informatics and Applications)*, 20 (1) (1986), pp. 79–87.

[2] Aho, A. H., Hopcroft, J. E. and Ullman, J.D. *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA, 1974.

[3] Bellmore, M. and Nemhauser, L. The Traveling Salesman Problem: A Survey. *Operations Research*, 16 (1968), pp. 538–558.

[4] Bentley, J. L. Experiments on Traveling Salesman Heuristics. *Proc.* 1^{st} *Ann. ACM–SIAM Symp. on Disc. Alg.*, 1990, pp. 91–99.

[5] Berman, O. and Simchi-Levi, D. Minisum location of a travelling salesman on simple networks. *European Journal of Operational Research*, 36 (1988), pp. 241–250.

[6] Bianco, L., Mingozzi, A. and Ricciardelli, S. The Traveling Salesman Problem with Cumulative Costs. *Networks*, 23 (1993), pp. 81–91.

[7] Blum, A., Chalasani, P., Coppersmith, D., Pulleyblank, B., Raghavan and P., Sudan, M. The Minimum Latency Problem. *Proc.* 26^{th} *Ann. ACM Symp. on Theory of Comput.*, 1994, pp. 163–171.

[8] Bonomi, E. and Lutton, J.-L. The N-City Travelling Salesman Problem: Statistical Mechanics and the Metropolis Algorithm. *SIAM Review*, 26 (4), October 1994, pp. 551–568.

[9] Christofides, N. *Worst-Case Analysis of a New Heuristic for the Traveling Salesman Problem*, Report 388, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, PA, 1976.

[10] Dillencourt, M. B. Traveling Salesman Cycles Are Not Always Subgraphs of Delaunay Triangulations or of Minimum Weight Triangulations. *Information Processing Letters*, 24 (1987), pp. 339–342.

[11] Dillencourt, M. B. A Non-Hamiltonian, Nondegenerate Delaunay Triangulation. *Information Processing Letters*, 25 (1987), pp. 149–151.

[12] Du, D.-Z. and Hwang, F. *Computing in Euclidean Geometry*, World Scientific, Singapore, 1992.

[13] Frederickson, G. N., Hecht, M. S. and Kim, C. E. Approximation algorithms for some routing problems. *SIAM J. Comput*, 7 (1978), pp. 178–193.

[14] Jünger, M., Reinelt, G. and Thienel, S. *Provably good solutions for the traveling salesman problem*, Report 92.114, Institut für Informatik, Universität Zu Köln. Köln, 1992.

[15] Kantabutra, V. Traveling Salesman Cycles Are Not Always Subgraphs of Voronoi Duals. *Information Processing Letters*, 16 (1983), pp. 11–12.

[16] Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H. and Shmoys, D. B. *The Traveling Salesman Problem*, John Wiley & Sons, Chichester, 1985.

[17] Lucena, A. Time-Dependent Traveling Salesman Problem—The Deliveryman Case. *Networks*, 20 (1990), pp. 753–763.

[18] Marcotte, O. and Suri, S. Fast Matching Algorithms For Points On A Polygon. *SIAM J. Comput.*, 20 (3) (June 1991), pp. 405–422.

[19] Papadimitriou, D. H. and Vazirani, U. V. On two geometric problems related to the travelling salesman problem. *J. Algorithms*, 5 (1984), pp. 231–246.

[20] Preparata, F. P. and Shamos, M. I. *Computation Geometry: An Introduction*, Springer-Verlag, New York, 1985.

[21] Rosenkrantz, D. J., Stearns, R. E. and Lewis, P. M. Approximate algorithms for the traveling salesperson problem. *Fifteenth Annual IEEE Symoposium on Switching and Automata Theory*, May 1974, pp. 33–42.

[22] Rosenkrantz, D. J., Stearns, R. E. and Lewis, P. M. II. An analysis of several heuristics for the traveling salesman problem. *SIAM J. Comput.*, 6 (1977), pp. 563–581.

[23] Rote, G. The N-line Traveling Salesman Problem. *Networks*, 22 (1992), pp. 91–108.

[24] Stewart, W. R. Jr. A computationally efficient heuristic for the traveling salesman problem. *Proc.* 13^{th} *Annual Meeting of S. E. TIMS*, 1977, pp. 75–85.

[25] Supowit, K. J. and Reingold, E. M. Divide and Conquer Heuristics for Minimum Weighted Euclidean Matching. *SIAM J. Comput.*, 12 (1) (Feb. 1983), pp. 118–143.

[26] Supowit, K. J., Reingold, E. M. and Plaisted, D. A. The Traveling Salesman Problem and Minimum Matching in the Unit Square. *SIAM J. Comput.*, 12 (1) (Feb. 1983), pp. 144–156.

[27] Tsitsiklis, J. N. Special Cases of Traveling Salesman and Repairman Problems with Time Windows. *Networks*, 22 (1992), pp. 263–282.

[28] Vaidya, P. M. Geometry Helps in Matching. *Proc. of the* 20^{th} *Ann. ACM Symp. on Theory of Computing*, (1988), pp. 422–425.