Flinders University

School of Computer Science, Engineering and Mathematics

Edge Elimination in 2D Euclidean TSP

Author:
Mohammed Alammar

Supervisor:
Dr. Vladimir Ejov

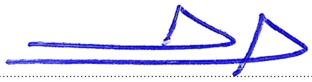
February 13, 2018

Submitted to the School of Computer Science, Engineering, and Mathematics in the Faculty of Science and Engineering in partial fulfilment of the requirements for the degree of Master of Science (Mathematics) at Flinders University – Adelaide Australia.

Declaration

"I certify that this work does not incorporate without acknowledgment any material previously submitted for a degree or diploma in any university; and that to the best of my knowledge and belief it does not contain any material previously published or written by another person except where due reference is made in the text".

MOHAMMED ALAMMAR



Signature

February 13, 2018

Date

Abstract

Although the traveling salesman problem (TSP) has a long research history as a mathematical approach to discover the shortest trip between a set of cities, there is no effective solution for the problem as it is considered an $\mathcal{NP}-hard$ problem of combinatorial optimisation. For example, solving the symmetric traveling salesman problem (STSP) using the fastest computer program, Concorde, for 85,900 vertices (which is the largest instance of TSP which has been solved to provable optimality to date) takes more than 199 CPU days. However, due to the importance of practical applications, methods to find optimal solutions have been developed since the 1950s. Stefan Hougardy and Rasmus T. Schroeder's algorithm for reducing the number of edges in two-dimensional Euclidean instances of TSP is one such method and hastens the STSP process by 11 times in certain test examples. The total runtime for the main part of their algorithm is $O(n^2 \log n)$ for n number of cities. The largest TSP case, 85,900 points, took 2 CPU days to run their algorithm. Concorde needed 16 CPU days to achieve the best outcome for this case. Hougardy and Schroeder's algorithm was presented alongside theoretic graph results showing how they proved that some edges of a TSP instance cannot be part of any optimal TSP trip. This thesis is based on Hougardy and Schroeder's results and shows how unnecessarily edges of a TSP instance can be avoided in any optimal TSP trip. The thesis presents practical results demonstrating how this can be achieved; it also presents a Matlab code for a TSP instance named qatar194. The instance contains 194 nodes and 264 edges after being run with Hougardy and Schroeder's elimination algorithm. This research shows that more edges can be eliminated when $k - opt$ edge exchanges are considered for $k > 3$. Linear programming (LP) is used in combination with subtour elimination constraints (SEC) and comb inequalities. The LP approach is taken by adding extra 28 constraints employing SEC and comb inequalities. The approach allowed the elimination of 58 unnecessarily edges and improved the lower bound to 9,350.6; while the optimal tour length is 9,352. Mixed-integer linear programming (ILP) is then used to find the shortest tour with no unnecessarily edges.

Acknowledgment

Deep gratitude goes first to Dr. Vladimir Ejov, who expertly guided and patiently supported my excitement to complete this research. It is also a great pleasure to acknowledge my extreme sincere gratitude and appreciation to Mr. Serguei Rossomakhine for his encouragement, and his creative and comprehensive advice during the research. I am truly grateful to all the lecturers in the School of Computer Science, Engineering, and Mathematics at Flinders University for their support towards the successful completion of my studies in Australia.

I wish to express my gratitude, appreciation, and warmest affection to my beloved families; for their understanding and endless love, through the duration of my studies. Special gratitude and thanks to my mother for her love, tenderness, devotion and unconditional support.

My appreciation and gladness also go to the government of the Kingdom of Saudi Arabia for allowing me to complete a master's degree at the international level. I would certainly be remiss to not mention and thank the sponsorship authority, Shaqra University, for giving me this opportunity to complete my study aboard. Finally, I sincerely thank Dr.Mohammed Nassar, Head of the Mathematics Department at Shaqra University to whom I am highly indebted and thoroughly grateful for guiding me through to completing my education.

Contents

Introduction

One of the fundamental, classical, and widely studied problems in combinatorial optimisation is the traveling salesman problem (TSP). The general idea is to minimise the total length of the tour of a salesperson who begins from a given city and returns to the same city after visiting each other city exactly once. The salesman must make a closed, completed trip, finishing with the departure city in the shortest distance that is known, also, as a Hamiltonian tour in a graph. The problem was first formulated as a mathematical problem in the 1930s, by Karl Menger, in Vienna and Harvard. In operations research and computer science, since the 1950s, methods have been developed to solve the TSP because of its practical applications. Most of these methods of solving TSPs were motivated by direct applications. For instance, scheduling a school bus route problem was considered by Flood (Flood, 1956). Also, crop survey implementations were studied by Mahalanobis and Jessen (Mahalanobis, 1940; Jessen, 1942). As a result, many exact and approximate algorithms as well as a large number of techniques have been developed to solve this problem.

There are two types of TSP solver, exact and non-exact. The exact solving methods, such as the cutting plane, interior point, branch-and-bound and branch-and-cut are characterised as guaranteeing to find the optimal solution for this problem (Chauhan, Gupta, & Pathak, 2012). Nowadays, the fastest available algorithm to solve large TSP instances optimally is the Concorde TSP solver. The Concorde TSP solver is a computer code proposed by David Applegate, Robert E. Bixby, Vašek Chvátal, and William J. Cook. Although Concorde is the fastest computer program, the total run time for 85,900 vertices (which is the largest instance of TSP which has been solved to provable optimality to date), for instance, needed more than 136 years of total CPU time (Applegate, Bixby, Chvatal, & Cook, 2011). On the other hand, the non-exact solvers, like the Christofides, Clarke-Wright, and Lin-Kernighan (LK) algorithms, usually execute faster but may give a non-optimal solution (Chauhan et al., 2012).

Searching for an optimal tour in a complete undirected graph through a given set of locations is a well known mathematical approach started long ago. Although a TSP is easily understood, solving the problem is known to be computationally difficult. If the cost of traveling from city i to city j equals that from j to i, then the problem is known as a symmetric traveling salesman problem (STSP); otherwise, it is called an asymmetric traveling salesman problem (ATSP). A symmetric problem, like an euclidean TSP, is extremely important in practical applications; it is used as a metric where the vertices are shown as points in the two-dimensional euclidean plane and the length of an edge is the euclidean distance between the two points. However, the majority of the results in this thesis will not be metric, because our results hold for arbitrary symmetric TSP cases.

In this thesis, the major focus is upon theoretical results as presented by Hougardy and Schroeder for eliminating useless edges. Results are supported by analysing two-dimensional euclidean TSP instances. As the edge elimination algorithm given by Hougardy and Schroeder can reduce the total run time, because it is 11 times faster than Concorde alone, applying their algorithm steps was required to be able to eliminate more useless edges faster (Hougardy & Schroeder, 2014). The key purpose, in this thesis, is to eliminate further useless edges that do not belong to any optimal TSP tour. Writing a software code using Matlab to solve two-dimensional euclidean TSPs was the main basis of the research's outcome. Reflecting our practical results was also necessary to show the way we can avoid some extra useless edges of a TSP.

Methodology

The main aim of this paper is to find a way to eliminate useless edges that do not belong to any optimal TSP tour, so we can write a software code using Matlab to solve two-dimensional euclidean TSPs. The paper is organised as follows. The first chapter illustrates the complexity of the TSP and gives a brief look at the history of this problem. This chapter, also, shares about some basic graph definitions and some assumptions and notations that are needed later in this thesis. The initial linear programming relaxation is presented in this stage while the improvement of this relaxation will be reflected afterward. The second chapter represents some methods and theorems for eliminating edges adopted from Stefan Hougardy and Rasmus T. Schroeder (Hougardy & Schroeder, 2014). The third chapter, the results chapter, presents the practical results, demonstrating how we can avoid some extra useless edges of a TSP instance in any optimal TSP trip. To arrive at our aim, there are several steps have been used in the results chapter as follows

- Applying Stefan Hougardy and Rasmus T. Schroeder algorithm steps for a TSP instance. The instance was taken from TSPLIB (a library of sample instances for the TSP collected from different resources) and it's called qatar194. It contains 194 cities and 264 edges after repeating the Stefan Hougardy and Rasmus T. Schroeder algorithm steps.

- Engaging $k - opt$ move edge exchange transformations with $k > 3$, where the $k - opt$ move is a concept that the most powerful heuristic TSP algorithms depend on. This engagement was from employing the subtour elimination constrains (SEC) and comb inequalities for the purpose of eliminating further edges.

- Using these inequalities to solve the problem as a linear programming problem. After adding 28 linear inequalities, as a result, we eliminated 58 further useless edges. The optimal tour must have the number of edges equal to the number of cities, so, we applied the mixed-integer linear programming technique to achieve the optimal tour.

Chapter 1

Mathematical preliminaries

In this chapter, we present some common themes when considering the TSP complexity, brief history, basic graph theory terminologies and linear programming relaxation for the traveling salesman problem.

1.1 The TSP complexity

In the 1960s, Edmonds came up with a substantial theoretical question of whether there is a good algorithm for solving the TSP. The answer for this question remains unknown. Over the past 50 years, the TSP has been set within the subject of complexity theory as a general context. The problem, in this theory, became a decision question where the question can be answered with either yes or no. For example, asking if a trip exists of length less than K instead of asking to find a minimum length trip. Two important complexity classes: $\mathcal{P}$ and $\mathcal{NP}$. If an algorithm exists that guarantees to answer the question correctly in polynomial time, then the problem is known as being in class $\mathcal{P}$. On the other hand, $\mathcal{NP}$ class stands for non-deterministic polynomial time. The problem is in $\mathcal{NP}$ if on any occasion the answer to the decision question is yes, then we are able to check that the proposed solution is indeed a solution in polynomial time. If, for instance, the answer for the TSP is yes, then this can be verified by displaying a trip which has less length than K. Such a problem is called $\mathcal{NP} - complete$ or $\mathcal{NP} - hard$; if a decision problem exists that is solvable in polynomial time, then for every problem in $\mathcal{NP}$ there exists a solution in polynomial time. This existence was proven by Stephen Cook (S. A. Cook, 1971) in 1971, and then, as a result, his initial $\mathcal{NP} - hard$ problem guided Richard Karp (Karp, 1972) to prove the TSP is $\mathcal{NP} - hard$.

1.2 Brief history of TSP

There is a bit of mystery behind the origin of the name "traveling salesman problem". The name first originated in the United States and appeared on a report published in 1949 (Robinson, 1949). In fact, the problem was already known by various names. The Viennese mathematician Karl Menger looked at a variation of TSP called the messenger problem (Botenproblem) (Bock, 1963; Menger, 1932). The messenger problem is described in English by Bock (Bock, 1963) as follows:

> "*We designate as the Messenger Problem (since this problem is encountered by every postal messenger, as well as by many travellers) the task of finding, for a finite number of points whose pairwise distances are known, the shortest path connecting the points. This problem is naturally always solvable by making a finite number of trials below the number of permutations of the given points. The rule, that one should first go from the starting point to the nearest point, then to the point nearest to this etc., does not in general result in the shortest path.* "

While the report by Menger (Menger, 1932) was the first published work on the TSP (Gutin & Punnen, 2006; Held, Hoffman, Johnson, & Wolfe, 1984), the first reference that brought the research community's attention towards the TSP in 1983 is the 1832 German handbook *Der Handlungsreisende-wie er sein soll und was er zu thun hat, um Aufträge zu erhalten und eines glücklichen Erfolgs in seinen Geschäften gewiss zu sein-Von einem alten Commis-Voyageur* by Heiner Müller-Merbach (Müller-Merbach, 1983). The best known algorithm to find an optimal solution for the TSP with the smallest time complexity was developed in 1962 by Held and Karp (Held & Karp, 1962). Their algorithm guarantees to find a solution in time complexity $O(n^2 2^n)$. Since the time complexity is exponential, however, Held and Karp's algorithm does not allow us to solve a large TSP instance within a reasonable time. The substantial work of Dantzig, Fulkerson, and Johnson in 1954 to find the optimal tour for the state capitals of the United States is considered to be the first systematic study of the TSP as a combinatorial optimisation problem (G. Dantzig, Fulkerson, & Johnson, 1954). They proved that a feasible solution is indeed optimal using the LP relaxation. Their technique of solving the 49-city problem was based on an earlier solution found by using a *branch and bound* algorthim for part of the problem containing 42 cities. The most common method to solve large LPs is the simplex algorithm, which was developed by Dantzig in 1947 (Reeb, Leavengood, et al., 1998). Table (1.1) gives the milestones achieved in solving the TSP. The most recent milestone

Table 1.1: Milestones in the solution of the TSP

Year	Names	TSP size
1954	G. Dantzig, R. Fulkerson, S. Johnson	42 cities
1954	G. Dantzig, R. Fulkerson, S. Johnson	49 cities
1971	M. Held, R.M. Karp	57 cities
1971	M. Held, R.M. Karp	64 cities
1975	P.M. Camerini, L. Fratta, F. Maffioli	67 cities
1975	P. Miliotis	80 cities
1977	M. Grötschel	120 cities
1980	H. Crowder, M. W. Padberg	318 cities
1987	M. W. Padberg, G. Rinaldi	532 cities
1987	M. Grötschel, O. Holland	666 cities
1987	M. W. Padberg, G. Rinaldi	1,002 cities
1987	M. W. Padberg, G. Rinaldi	2,392 cities
1994	D. L. Applegate, R.E. Bixby, V. Chvátal, W. J. Cook	7,397 cities
1998	D. L. Applegate, R.E. Bixby, V. Chvátal, W. J. Cook	13,509 cities
2001	D. L. Applegate, R.E. Bixby, V. Chvátal, W. J. Cook	15,112 cities
2004	D. L. Applegate, R.E. Bixby, V. Chvátal, W. J. Cook	24,978 cities
2006	D. L. Applegate, R.E. Bixby, V. Chvátal, W. J. Cook	85,900 cities

was achieved in 2006 by solving a TSP containing 85,900 cities; it is the largest instance of TSP which has been solved to provable optimality to date (W. Cook, 2012). The way of solving such an instance was by the fastest free available TSP solver *Concorde*, written by David Applegate, Robert E. Bixby, Vašek Chvátal, and William J. Cook.

1.3 Basic graph theory terminologies

Definition 1 (Simple graph). *A simple graph, $G = (V, E)$, consists of a finite nonempty set V of objects called vertices (or nodes) and a set E of edges.*

Definition 2 (Adjacent). *Two vertices are adjacent when they share a common edge.*

Definition 3 (Incident). *A vertex x and the edge xy are called incident with each other since they have a common vertex x.*

Definition 4 (Complete graph). *A graph G is said to be complete when every pair of distinct vertices is connected by an edge.*

Definition 5 (Walk). *In a graph theory, a walk W in a graph G is a sequence of vertices $(x_1x_2...x_{k-1}x_k)$ starting from x_1 and ending at x_k if $\{x_i, x_{i+1}\} \in E, \forall\ 1 \leq i \leq k-1$.*

Definition 6 (Path). *A walk in a graph G is a path if every vertex is visited no more than one time.*

Definition 7 (Closed Walk). *A walk of G is classified as a closed if the initial and terminal vertices are the same.*

Definition 8 (Hamiltonian cycle). *A cycle in a graph G that contains all vertices of G is called a Hamiltonian cycle (or tour) of G.*

Definition 9 (Connected and Disconnected Graphs). *A graph G is called connected if a path (or edge) exists between every pair of vertices of G, otherwise G is disconnected.*

Definition 10 (2-Connected Graph). *For every vertex $x \in G(V)$, the graph G is called 2-connected if $G - x$ is connected.*

Definition 11 (Subgraph). *For a graph $G = (V, E)$, $H = (W, F)$ is called a subgragh of G if $W \subseteq V$ and $F \subseteq E$. Furthermore, if $W \subseteq V$ and $F = \{e : e \in E, e \subseteq W$, then $H = (W, F)$ is invited by the subgraph induced by W.*

Definition 12 (Component). *For a graph $G = (V, E)$, H is called component of G if H is not proper subgraph of any connected subgraph of G.*

1.4 Notations and assumptions

Since Stefan Hougardy and Rasmus T. Schroeder's algorithm is the basis of this research, we shall assume that a TSP instance has at least four nodes as they mentioned in their paper. In this research, our aim is to visit each city exactly once in a closed and completed route. Since every city is visited once, there is a finite number of edges in the graph given by $n(n-1)/2$ where n is the number of cities. Any such possible trip is called a solution to the TSP. As the number of possible trips is limited, there must be a trip which has a minimal travel cost. The trip with a minimal travel cost is called an optimal solution for the TSP. There are several notations that must be presented:

- For simplicity we denote the edge $\{p, q\}$ by pq where p and q are two vertices in a TSP instance.

- Let the distance function l be the $2D$ Euclidean function. Also, consider pq and rs to be two edges in a TSP tour. The distance between two points in euclidean space is called euclidean distance. Euclidean distance between two points p and q is denoted as $|pq|$ where:

$$l(pq) - \frac{1}{2} \leq |pq| \leq l(pq) + \frac{1}{2} \tag{1.1}$$

- If T is a solution for a TSP instance, then the total length is defined as $\sum_{e \in E(T)} l(e)$ where $E(T)$ is the edge set of the tour T and l is the symmetric length function when $l : V \times V \rightarrow \mathbb{R}_+$.

- Suppose there are two edges pq and rs in a TSP instance, they are called compatible edges denoted by ($pq \sim rs$) if:

$$max\ (l(pr) + l(qs), l(ps) + l(qr)) \geq l(pq) + l(rs) \tag{1.2}$$

  Otherwise pq and rs are called incompatible. In fact, any two edges sharing at least one vertex are always compatible. Suppose the edges pq and rs are compatible and $t \in rs$, thus, it is an obvious that the edges pq and rt are also compatible. Moreover, any two edges in an optimal TSP tour are compatible. Let us assume the opposite, suppose the edges pq and rs are incompatible in an optimal TSP tour T, by (1.2) we have,

$$l(pr) + l(qs) < l(pq) + l(rs) \quad and \quad l(ps) + l(qr) < l(pq) + l(rs) \tag{1.3}$$

  By 2-opt moves, we can replace edges pq and rs by either pr and qs or ps and qr, so one of these movements must be valid, resulting a shorter tour and that contradicts the assumption that T is an optimal TSP tour.

- In two-dimensional euclidean space, for each point r choose δ_r such that no vertex apart from r lies in the interior of the circle by r with radius δ_r. One can for example use:

$$\delta_r := \frac{1}{2} + max\ \{d \in Z_+ \mid \forall\ s \in V \setminus \{r\}\ l(rs) > d\} \tag{1.4}$$

  the two lengths between an edge pq and a point $r \in V \setminus \{p, q\}$ are given by:

$$l_p := \delta_r + l(pq) - l(qr) - 1 \quad and \quad l_q := \delta_r + l(pq) - l(pr) - 1 \tag{1.5}$$

  Also, for each point $s \in V \setminus \{r\}$, we define $s_r \in rs$ such that $|rs_r| = \delta_r$.

- An edge is classified as a useless edge if it does not belong to any optimal TSP tour.
- For $S \subseteq V$ we denote $\delta(S)$ as the set of edges that have one end in S and one end not in S, that is $\delta(S) = \{e \in E : |e \cap S| = 1\}$.
- For $S \subseteq V$, we denote $E(S)$ as the set of all edges in G having both endpoints in the set S.

1.5 The LP relaxation

One of the most well studied representations of the TSP is the linear programming representation (LP for short). Linear programming is a mathematical technique to maximise or minimise a linear function subject to a set of inequalities. Officially, the LP is a method of using a linear objective function subject to linear equality and linear inequality constraints. The introduction of the simplex method, written by Dantzig in 1947, was the beginning of the LP computation. Although the simplex method has been used since then and has become the major way of solving the LP, the first paper was an unpublished technical report (Agarwala, Applegate, Maglott, Schuler, & Schäffer, 2000; G. B. Dantzig, 1948). Before presenting the standard format for the LP relaxation of the TSP,

- Suppose S expresses the set of incidence vectors of all the tours.
- Assume x_e refers to the variable x corresponding to e.
- To drive a model by way of a linear description, we have to represent the tour as its incidence vector of length $n(n-1)/2$. Therefore, for every trip (or edge e), we present a vector $x \in S$ of decision variables x_e as follows:

$$x_e = \begin{cases} 1 & \text{if edge } e \text{ is selected;} \\ 0 & \text{if not.} \end{cases}$$

For a TSP instance, the length of a tour should be minimised, thus, the linear objective function that we want to optimise can be expressed as:

$$\textit{minimise } c^T x \textit{ subject to } x \in S \tag{1.6}$$

where c are vectors of coefficients, $(.)^T$ is the matrix transpose and x is the vector of variables. As the decision variables above show, for each $x \in S$ and each edge e, we can express that as:

$$0 \leq x_e \leq 1 \quad \textit{for all edges } e \tag{1.7}$$

Also, since every city v should be visited once, it must be entered and then left to find a Hamiltonian cycle. In other words, for any trip $x \in S$, every city v is met by exactly two edges, thus:

$$\sum (x_e : v \ \ \textit{is an end of}\, e) = 2 \ \ \textit{for all cities}\ v \tag{1.8}$$

So far, we can say that the standard format for the LP relaxation of the TSP is given by:

$$\textit{minimise}\ c^T x \ \textit{subject to} \ \ (1.7), (1.8) \tag{1.9}$$

Chapter 2

Elimination Theorems

The second chapter presents some theorems and methods for eliminating edges as presented by Stefan Hougardy and Rasmus T. Schroeder. Briefly, their algorithm to eliminate useless edges is described in three steps. For the first step, they used the Main Edge Elimination Theorem, in which it is necessary to determine two potential points which satisfy a particular condition. Finding the two potential points that satisfy the condition of the Main Edge Elimination Theorem allowed them to decrease the number of edges. After that, the Close Point Elimination Theorem was applied in combination with the Main Edge Elimination Theorem, to avoid further useless edges. Finally, repeating the search of bounded depth was considered to eliminate further edges. This chapter contains two parts. The first part is about the theorems, describing how some edges can be considered as useless edges. The second part shows how these theorems can be satisfied by presenting efficient methods developed for certifying potential points.

2.1 The Main Edge Elimination Theorem

Introducing the concept of potential points is required to formulate this theorem. Let (V, E) be a TSP instance and $pq \in E$. For $r \in V \setminus \{p, q\}$, define:

$$R := \{x \in V \mid rx \in E \wedge pq \sim rx\} \tag{2.1}$$

In addition, let $R_1, R_2 \subset V$ and $R \subset R_1 \cup R_2$ and $R_1 \cap R_2 \neq \phi$. The point r is called a potential point with respect to the edge pq and R_1 and R_2, if for each optimal tour including pq, the two neighbors of r cannot both lie in R_1 or R_2 respectively. The potentiality of r is certified when such a cover is found. In any optimal tour having the edge pq and a potential point r, however, it cannot be connected with $R_1 \cap R_2$. There are effective methods already developed for certifying the potentiality of points presented in the next section.

Theorem 1 (The Main Edge Elimination Theorem)**.** *Let (V, E) be a TSP instance and $pq \in E$. Let r and s be two different potential points with respect to pq with covering R_1 and R_2, and, respectively, S_1 and S_2. Let $r \notin S_1 \cup S_2$ and $s \notin R_1 \cup R_2$. If:*

$$l(pq) - l(rs) + \min_{z \in S_1}\{l(sz) - l(pz)\} + \min_{y \in R_2}\{l(ry) - l(qy)\} \; > \; 0 \qquad (2.2)$$

and

$$l(pq) - l(rs) + \min_{x \in R_1}\{l(rx) - l(px)\} + \min_{w \in S_2}\{l(sw) - l(qw)\} \; > \; 0 \qquad (2.3)$$

then the edge pq is useless.

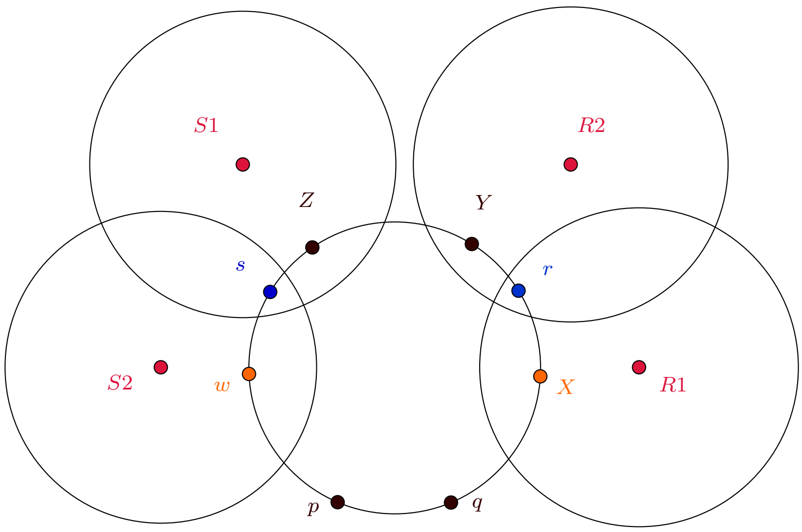


Figure 2.1: Two covered potential points

Proof. Assume that the edge pq contained in an optimal TSP tour T. Let $rx, ry, sz, sw \in T$ be the incident edges of r and s. Consider the vertices x, y, z and w as being classified such that $x \in R_1$, $y \in R_2$, $z \in S_1$ and $w \in S_2$. We assumed that the two points r and s are potential, so $r, s \notin \{p, q\}$ and $rs \notin T$, therefore the four edges rx, ry, sz and sw are distinct. As a result, there are two possible 3-opt moves and one of them must be valid. The first 3-opt move is replacing pq, rx and sw with px, rs and qw, so we have:

$$l(pq) + l(rx) + l(sw) - l(px) - l(rs) - l(qw) \qquad (2.4)$$

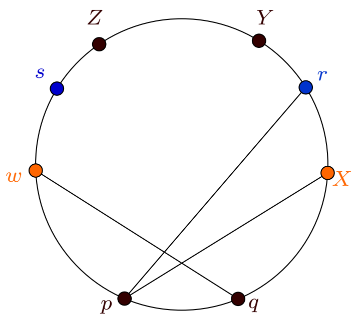
The second 3-opt move is to replace pq, ry and sz with pz, rs and qy, so we get

$$l(pq) + l(ry) + l(sz) - l(pz) - l(rs) - l(qy) \tag{2.5}$$

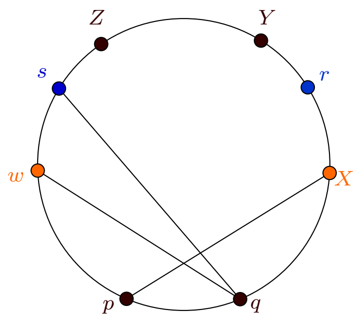
By the two theorem inequalities (2.2 and 2.3), both terms are strictly positive. Since one of these 3-opt moves is valid, a shorter tour for T is found, contradicting the optimality of T.

□

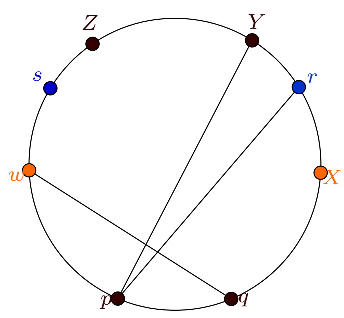
In general, the 3-opt moves can offer multiple valid tours to prove the uselessness of the edge pq. All these possibilities are presented below. However, these tours might not result in a shorter trip than T. To contradict the optimality of a tour we have to find a shorter tour than T from one of these possibilities by using the Main Edge Elimination Theorem inequalities.



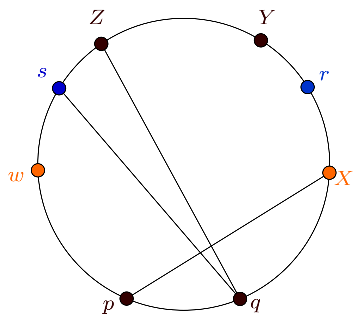
q, w, s, z, y, r, p, x, q



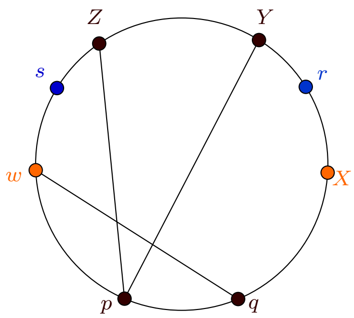
p, x, r, y, z, s, q, w, p



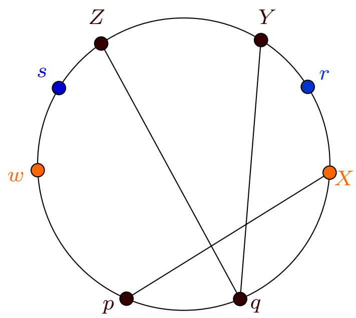
q, w, s, z, y, p, r, x, q



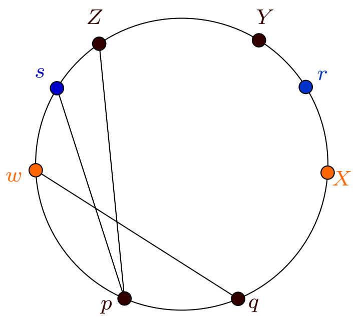
p, x, r, y, z, q, s, w, p



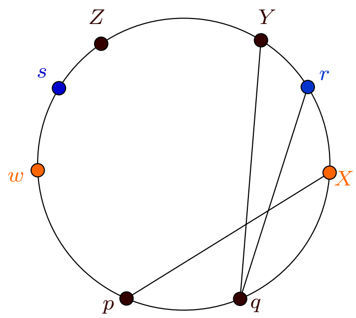
q, w, s, z, p, y, r, x, q



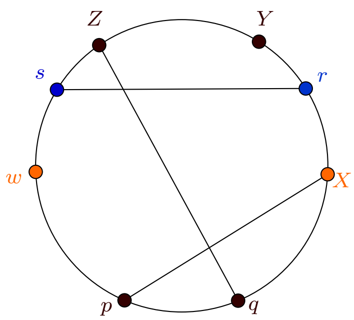
p, x, r, y, q, z, s, w, p



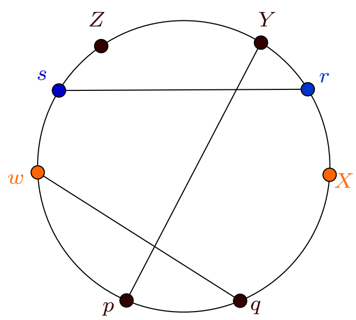
q, w, s, p, z, y, r, x, q



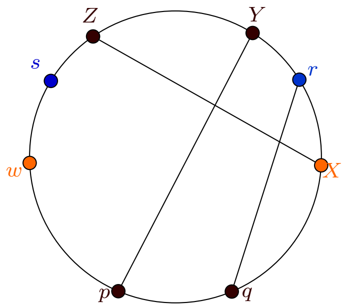
p, x, r, q, y, z, s, w, p



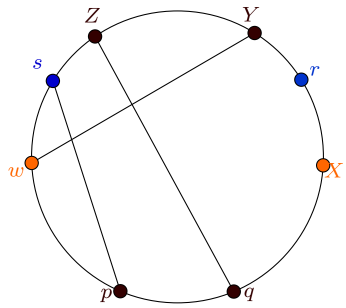
q, z, y, r, s, w, p, x, q



p, y, z, s, r, x, q, w, p



$\mathbf{q, r, y, p, w, s, z, x, q}$



$\mathbf{p, s, z, q, x, r, y, w, p}$

2.2 The Close Point Elimination Theorem

Although the uselessness of an edge in a TSP instance can be proved by The Main Edge Elimination Theorem, there may be many other useless edges present which are not identified by this process, therefore, other methods can be applied. The Close Point Elimination Theorem is unlikely to be able to eliminate further useless edges when it is applied to the complete graph of a TSP instance. However, it will allow us to identify additional useless edges when combined with The Main Edge Elimination Theorem.

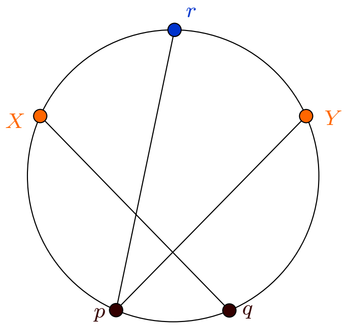
Theorem 2 (Close Point Elimination Theorem)**.** *Let (V, E) be a TSP instance and $pq \in E$. Let $r \in V \setminus \{p, q\}$ define $R := \{x \in V | \, rx \in E \wedge pq \sim rx\}$. If for all $x, y \in R$ with $\{x, y\} \neq \{p, q\}$, we then have*

$$l(xy) + l(pr) + l(qr) \; < \; l(pq) + l(xr) + l(yr) \tag{2.6}$$

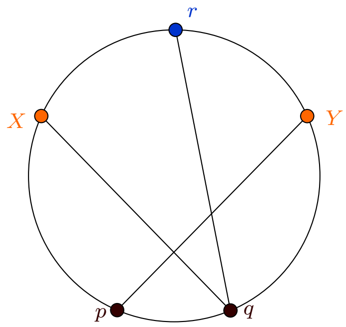
then the edge pq is useless.

Proof. Suppose T is an optimal TSP tour that contains the edge pq. Let xr and ry be the two edges in T that are incident with r. Then $\{x, y\} \neq \{p, q\}$ and x and y must be in R. By assumption, the inequality (2.6) holds, so we can substitute the edges pq, xr and yr with xy, pr and qr and find a tour that is shorter than T. Moreover, if one of x and y equals one of p and q, then it also holds. Thus, the optimality of the tour T contradicted. □

Another two possible 3-opt moves shown below might generate a shorter tour than T.



p, y, q, x, r, p



q, x, p, y, r, q

Using the notation of metric excess, we can obtain a stronger result in a degenerate case such as when $x = p$. This notation allows us to short cut a eulerian subgraph in an instance that does not need to be metric. The metric excess of a vertex z with respect to an edge pq denoted by $m_{pq}(z)$ is defined as:

$$\min_{x,y \in N(z)\setminus\{p,q\}} \quad \max\{l(xz) + l(zp) - l(xp), l(yz) + l(zp) - l(yp),$$
$$l(xz) + l(zq) - l(xq), l(yz) + l(zq) - l(yq)\}.$$

For $k > 2$ we call a set of k edges k-incompatible, if they cannot belong to the same optimal TSP tour.

Theorem 3 (The Strong Close point Elimination Theorem)**.** *Let pq, pr and rx be three edges of a TSP instance (V, E). Let $z \in V \setminus \{p, q, r, x\}$. If*

$$l(xq) + l(rz) + l(zp) - m_{pr}(z) \;<\; l(pq) + l(rx) \tag{2.7}$$

then the edges pq, pr and rx are 3-incompatible.

Proof. Assume the edges pq, pr and rx were contained in an optimal TSP tour T. The Close Point Elimination Theorem introduces how 3-opt moves yield a shorter tour. Thus, we can delete the edges pq and rx and insert the edges qx, pz and rz. This replacement is not a TSP tour as the vertex z has degree four. As $l(xq) + l(rz) + l(zp) - m_{pr}(z) \;<\; l(pq) + l(rx)$ is a short cut, however, it might yield a shorter tour than T. Thus, the optimality of the tour T is contradicted.

□

2.3 Certifying potential points

Major useless edges can be removed efficiently using the the Main Edge Elimination Theorem. Since this theorem needs two strong potential points to be satisfied, the aim of presenting the following lemmas is to certify a strong potential point r with respect to an edge pq in constant time. In addition, another strong potential point s with respect to an edge pq must be proved separately in constant time using the same steps.

Lemma 1. *Let (V, E) be a TSP instance, $pq \in E$, $r \in V \setminus \{p, q\}$ and $s \in V \setminus \{r\}$. The edges pq and rs are incompatible if $|ps_r| < l_p$ and $|qs_r| < l_q$.*

Proof. To show these edges are incompatible, we have to show that both 2-opt moves are driving a shorter length for the edges pq and rs by using (1.1 and 1.5) and, with the triangle inequality, we get:

$$l(ps) \leq |ps| + \frac{1}{2} \qquad \textit{where} \qquad |ps| \leq |ps_r| + |ss_r|$$

$$l(ps) + l(qr) \leq |ps_r| + |ss_r| + \frac{1}{2} + l(qr) < l_p + |ss_r| + \frac{1}{2} + l(qr)$$

We know,

$$l_p := \delta_r + l(pq) - l(qr) - 1$$

Thus,

$$l(ps) + l(qr) < [\delta_r + l(pq) - l(qr) - 1] + |ss_r| + \frac{1}{2} + l(qr)$$

So,

$$l(ps) + l(qr) < l(pq) + [\delta_r + |ss_r| - \frac{1}{2}] \leq l(pq) + l(rs)$$

$l(ps) + l(qr) < l(pq) + l(rs)$ is proven similarly using (1.2), thus, the edges pq and rs are incompatible.

□

Lemma 2. *Let (V, E, ρ) be a TSP instance, where ρ is a distance function on V that satisfies triangular inequality (e.g. Euclidean distance), $pq \in E$, and $r \in V \setminus \{p, q\}$. If*

$$l_p + l_q \geq l(pq) - \frac{1}{2} \tag{2.8}$$

Then, the circle centered by r with radius δ_r intersects both circles centered by p and q with radii l_p and l_q respectively.

Proof. It suffices to show $|ir| - \delta_r \leq l_i \leq |ir| + \delta_r$ for $i \in \{p, q\}$. By (1.5) and (2.8) we get,

$$|pr| - \delta_r \leq l(pr) + \frac{1}{2} - \delta_r = l(pq) - \frac{1}{2} - l_q \leq l_p$$
$$\leq (|pr| + |qr| + \frac{1}{2}) - (|qr| - \frac{1}{2}) + \delta_r - 1 = |pr| + \delta_r$$

Analogously for $i = q$.

□

Lemma 3. *Let (V, E) be a TSP instance, $pq \in E$. For $r \in V \setminus \{p, q\}$ let $R \subset R_p \cup R_q$ with $R = \{x \in V \mid rx \in E \wedge pq \sim rx\}$. If for $i \in \{p, q\}$:*

$$l(pq) + l(rx) + l(ry) > l(pr) + l(rq) + l(xy) \qquad \forall\ x, y \in R_i, \tag{2.9}$$

then R_p and R_q certify the potentiality of r.

Proof. Assume T is an optimal tour that contains pq. And, $rx, ry \in T$ with $x, y \in R_i$ for $i \in \{p, q\}$. Then replacing the edges pq, rx and ry by the edges pr, rq and xy is a valid 3-opt move. According to the (2.9) inequality, this 3-opt move generates a shorter trip than T; thus, the optimality of the tour T is contradicted.

□

Lemma 3 gives a method to check the potentiality of a vertex r in non-constant time $O(n^2)$. Now, our aim is to show that the vertex r is potential in constant time. To do that, we assume the edge pq is a part of an optimal tour T. Also, let the covering be R_p and R_q as described previously. Let us denote the angles of the cones R_p and R_q as α_p and α_q respectively. The following lemmas certify the potentiality of r in constant time.

Lemma 4. *Let T be an optimal TSP tour that contains pq, $r \in V \setminus \{p, q\}$ and the angle γ between the two edges of T incident with r in T satisfies*

$$\gamma \ > \ \max\{\alpha_p, \alpha_q\}. \tag{2.10}$$

Then, the neighbors of r in T cannot lie in both R_p and R_q.

Proof. W.l.o.g. we assume both neighbors of r in T lie in R_p. This directly implies $\gamma \leq \alpha_p$, contradicting the inequality (2.10).

□

Lemma 5. *Assume (V, E) be a TSP instance and T an optimal tour. Let $pq \in T$, and $r \in V \setminus \{p, q\}$. Assume the inequality (2.8) holds. If we define the angle γ_r as*

$$\gamma_r := \cos^{-1}\left(1 - \frac{(l_p + l_q - l(pq) + \frac{1}{2})^2}{2\delta_r^2}\right) \tag{2.11}$$

Then, the angle γ between the two edges of T incident with vertex r satisfies

$$\gamma \;\geq\; \gamma_r \tag{2.12}$$

Proof. let $rx,\ ry\ \in T$ be the two incident edges of r. Let $\mu := |x_r y_r|$. The cosine rule drives the equation

$$\mu^2 = 2\delta_r^2 - 2\delta_r^2 \cos\gamma \tag{2.13}$$

As T is an optimal tour, therefore, there is no valid 3-opt move which generates a shorter trip, so:

$$l(pq) + l(rx) + l(ry) \leq l(pr) + l(qr) + l(xy)$$

$$l_p + l_q + |x_r x| + |y_r y| - l(pq) + 1 \leq l(xy)$$

$$l_p + l_q - l(pq) + \frac{1}{2} \leq \mu$$

$$\left(l_p + l_q - l(pq) + \frac{1}{2}\right)^2 \leq \mu^2 = 2\delta_r^2 - 2\delta_r^2 \cos\gamma$$

$$\cos\gamma \leq 1 - \frac{(l_p + l_q - l(pq) + \frac{1}{2})^2}{2\delta_r^2}$$

$$\gamma \geq \cos^{-1}\left(1 - \frac{(l_p + l_q - l(pq) + \frac{1}{2})^2}{2\delta_r^2}\right)$$

Thus,

$$\gamma \;\geq\; \gamma_r$$

□

Therefore, from Lemma 4 and Lemma 5, we get the following result.

Lemma 6. *Let pq be an edge contained in an optimal TSP tour T, and $r \in V \setminus \{p, q\}$. Assume that the inequality (2.8) holds. If*

$$\gamma_r > \max\{\alpha_p, \alpha_q\} \tag{2.14}$$

then the sets R_p and R_q certify the potentiality of r.

Both angles α_p and α_q of the cones R_p and R_q respectively can be computed in constant time as:

$$\alpha_p = 2 \cdot \cos^{-1}\left(\frac{l_q^2 - \delta_r^2 - |rq|^2}{2\delta_r|rq|}\right) \quad \textit{and,} \tag{2.15}$$

$$\alpha_q = 2 \cdot \cos^{-1}\left(\frac{l_p^2 - \delta_r^2 - |rp|^2}{2\delta_r|rp|}\right). \tag{2.16}$$

The results so far present a method to prove in constant time that a given vertex, r, is potential. However, not every potential point can be detected using this approach (Hougardy & Schroeder, 2014). If we suppose pq is an edge and $r \in V\setminus \{p, q\}$, then the vertex r is called *strongly potential* with respect to pq, if the conditions (2.8) and (2.14) hold. Therefore, we can check whether a point r is a strongly potential point in constant time $O(n)$ (Supposing that the value of δ_r is known, which can be calculated in the preprocessing step for all vertices). The Main Edge Elimination Theorem's inequalities still need $O(n)$ time to be verified. Showing how this can be done in constant time by calculating suitable lower bounds for the inequalities (2.2) and (2.3) is our next aim.

Lemma 7. *Let (V, E) be a TSP instance and r be a strongly potential point with respect to pq. Consider R_p and R_q be the covering certifying r. Then*

$$\min_{x \in R_p}\{l(rx) - l(px)\} \geq \delta_r - 1 - \max\{|px_r| : x \in R_p\} \quad \textit{and} \tag{2.17}$$

$$\min_{y \in R_q}\{l(ry) - l(qy)\} \geq \delta_r - 1 - \max\{|qy_r| : y \in R_q\} \tag{2.18}$$

Proof. Let $x \in R_p$. Then

$$\begin{aligned} l(rx) - l(px) &\geq |rx| - |px| - 1 \geq \delta_r + |x_r x| - (|px_r| + |x_r x|) - 1 \\ &\geq \delta_r - 1 - \max\{|px_r| : x \in R_p\} \end{aligned}$$

Similarly one can prove this for the set R_q.

□

Let the point r be the center of the circle C_r with radius δ_r. Define the two arcs

$$B_p := \{x \in C_r \mid |qx| \geq l_q\} \quad \text{and} \quad B_q := \{y \in C_r \mid |py| \geq l_p\}$$

Also, let C_r contain two points $\tilde{p}$ and $\tilde{q}$ with the greatest distances to p and q respectively. Since B_p and B_q are connected, the maxima in the inequalities

(2.17) and (2.18) can only be attained at $\tilde{p}$ and $\tilde{q}$ respectively, or at the endpoints of B_p and B_q respectively. Considering the case that:

$$|p\tilde{q}| \leq l_p \quad and \quad |q\tilde{p}| \leq l_q \tag{2.19}$$

This implies

$$\max\{|px_r| : x \in R_p\} \leq \max\{t \mid t \in B_p\} \quad and \tag{2.20}$$

$$\max\{|qy_r| : y \in R_q\} \leq \max\{t \mid t \in B_q\} \tag{2.21}$$

Using some lemmas presented in the appendix of Hougardy and Schroeder's paper, the right hand sides of (2.20) and (2.21) can be calculated in constant time.

Chapter 3

Solving the TSP

After choosing a TSP instance from TSPLIB (a library of sample instances for the TSP collected from different resources) which is called qatar194, repeating Stefan Hougardy and Rasmus T. Schroeder's algorithm steps was required. The instance contains 194 cities, so, the original number of edges is 18,721. However, applying their algorithm steps reduced the number of edges to become 264 edges only, which means 18,457 useless edges were eliminated. Here, the number of useless edges will be increased by adding some linear inequalities as constraints. This chapter is organised as follows. The first section shows how we achieve the aim in detail and presents every added inequality to the list of constraints. The second section summarises our work and talks about the chosen platform (Matlab) and gives some definitions of the functions we used.

3.1 Solving the LP relaxation

As we have mentioned in chapter one (section 1.5), the standard LP relaxation for the TSP is formulated as

$$\textit{minimise } c^T x \textit{ subject to}$$

$$\sum (x_e : v \;\; \textit{is an end of } e) = 2 \;\; \textit{for all cities } v$$

$$0 \leq x_e \leq 1 \quad \textit{for all edges } e$$

That means a feasible solution exists for every solution to the TSP. Although solving the above LP relaxation does not lead to a Hamiltonian tour, it provides an optimal solution x^* to the TSP relaxation. The optimal solution x^* is characterised by, or called, the *lower bound* for the TSP, which is the shortest length of a tour that can be used as a measurement of the quality of any suggested trip. Thus, there is no trip shorter than $c^T x^*$. In this instance, the optimal solution using the initial LP is shown in the following figure.

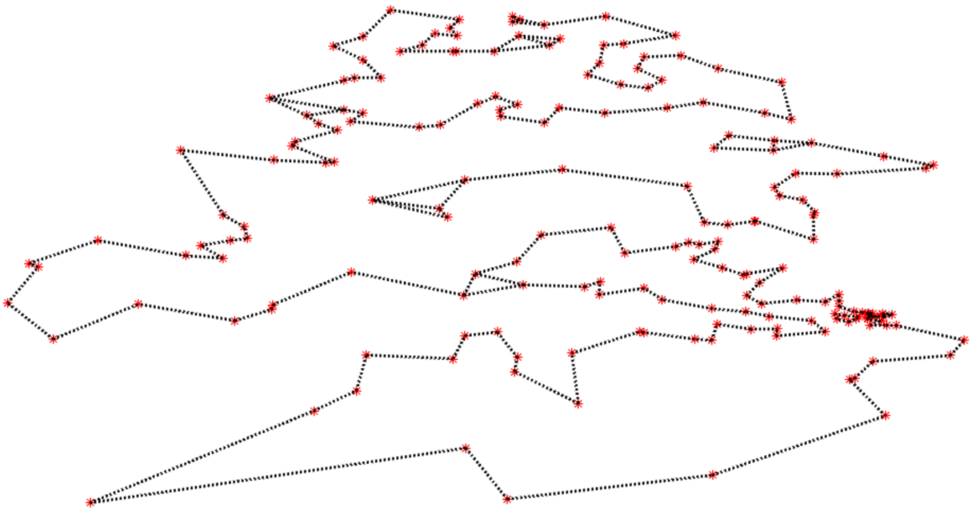


Figure 3.1: Solution of the initial LP relaxation

From figure (3.1), the solution x^* is disconnected; there are three components. These components (or subsets) contain 162, 26 and six vertices (the subtour with six vertices can be seen in figure (3.2)). Some of these vertices have an odd number of edges, which means there are some edges having fractional values. There are two possible values of an edge e in this figure, either it carries $x_e^* = 1$ or $x_e^* = 1/2$. Throughout this thesis, only those edges e with corresponding variable $x_e^* = 0$ are displayed in figures. Here, the number of non-zero edges is 206 with the total length, or *lower bound*, equal to (9,267) where the optimal tour is known to have length (9,352). In this case, the solution of the initial LP does not correspond to a Hamiltonian cycle. Therefore, we have to add suitable linear inequalities to the list of constraints to eliminate these subtours and improve the LP relaxation.

3.2 Improving the LP relaxation

From the previous section, the solution x^* produced a disconnected graph with three subtours. Now, adding extra constraints to the list of constraints is required to eliminate these subtours. To avoid the occurrence of subtours for a graph G with n vertices on node set S with $3 \leq |S| \leq n-1$. For $S \subseteq V$ we define the following

$$E(S) = \{(u, v) \in E \ : \ u \in S \ , \ v \in S\} \quad and,$$
$$\delta(S) = \{(u, v) \in E : \ either\ u\ or\ v \in S\}$$

To break the graph into disjoint parts, we have to remove all edges in $\delta(S)$ for all nonempty proper components S of vertices. Now, we are able to express the subtour elimination constraint (SEC) for eliminating these subtours as follows

$$\sum_{e \in E(S)} x_e \ \leq \ |S| - 1 \ \ for\ all\ \ S \subset V,\ 2 \leq |S| \leq |V| - 1. \qquad (3.1)$$

If the set $|S| = 2$, then these constraints decrease to $x_e \leq 1$ where e is the unique edge in $E(S)$. To eliminate these subtours and raise the LP value, we have to add three constraints (c_1, c_2 *and* c_3) to the list of constraints. For instance, to write the SEC for the smallest subtour that contains six cities, we have to know what the numbers (or names) of these cities are first, then we will be able to write the constraint. The subtour contains

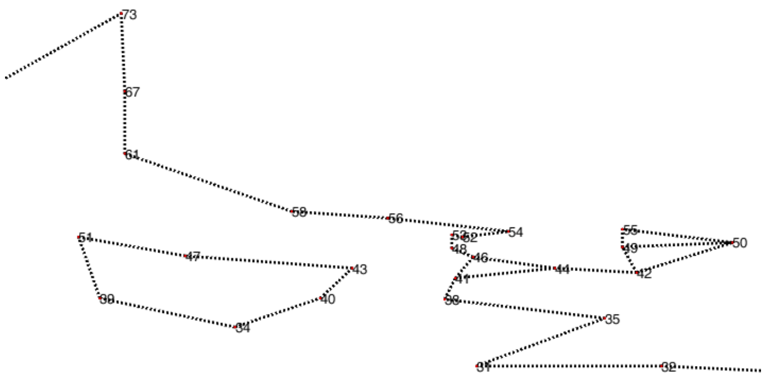


Figure 3.2: A part of the graph G shows S_1

$S_1 = \{34\ 39\ 40\ 51\ 43\ 47\}$. The constraint that the LP needs to eliminate this subtour using the formula (3.1) is given by

$$c_1 : \ x3439 + x3440 + x3947 + x3951 + x4043 + x4347 + x4751 \leq 5$$

where $x3439$ is the edge starting from the vertex number (34) and finishing at the vertex number (39). The following two constraints are required for eliminating the other two subtours.

$$
\begin{aligned}
c_2: \; & x0104 + x0106 + x0203 + x0204 + x0305 + x0509 + x0608 + x0711 + \\
& x0717 + x0816 + x0910 + x1012 + x1114 + x1215 + x1316 + x1323 + \\
& x1425 + x1519 + x1726 + x1821 + x1833 + x1930 + x2063 + x2065 + \\
& x2124 + x2227 + x2229 + x2325 + x2426 + x2737 + x2829 + x2833 + \\
& x3032 + x3132 + x3135 + x3538 + x3542 + x3544 + x3659 + x3663 + \\
& x3745 + x3841 + x3844 + x4144 + x4146 + x4244 + x4249 + x4250 + \\
& x4446 + x4448 + x4449 + x4557 + x4648 + x4654 + x4852 + x4853 + \\
& x4854 + x4856 + x4950 + x4955 + x5055 + x5253 + x5254 + x5356 + \\
& x5455 + x5456 + x5658 + x5760 + x5861 + x5962 + x6069 + x6167 + \\
& x6282 + x6468 + x6470 + x6585 + x6668 + x6673 + x6773 + x6974 + \\
& x7077 + x7176 + x7180 + x7182 + x7274 + x7275 + x7278 + x7576 + \\
& x7578 + x7680 + x7687 + x7784 + x7891 + x7981 + x7983 + x8087 + \\
& x8184 + x8388 + x8586 + x8698 + x87102 + x8892 + x8893 + x8990 + \\
& x8994 + x9098 + x9193 + x91103 + x9295 + x9297 + x9396 + x9499 + \\
& x9596 + x9597 + x99101 + x101104 + x102103 + x104111 + x111130 + \\
& x125126 + x125127 + x126132 + x126138 + x127130 + x127132 + \\
& x130132 + x130156 + x132134 + x134137 + x137140 + x138139 + \\
& x138142 + x139146 + x139154 + x140142 + x140145 + x141144 + \\
& x141147 + x141152 + x142146 + x143148 + x143160 + x144150 + \\
& x145149 + x145156 + x146149 + x147151 + x147152 + x148155 + \\
& x148160 + x149156 + x150153 + x150154 + x151155 + x152153 + \\
& x152159 + x153157 + x154157 + x155158 + x155162 + x156161 + \\
& x158159 + x158162 + x159165 + x160166 + x161163 + x161169 + \\
& x162167 + x163164 + x164169 + x164172 + x165168 + x166171 + \\
& x167168 + x167170 + x168178 + x169176 + x170171 + x170180 + \\
& x171185 + x172174 + x172179 + x173174 + x173175 + x174179 + \\
& x175177 + x175184 + x176182 + x177181 + x177184 + x178180 + \\
& x178181 + x179186 + x180185 + x181184 + x182194 + x183186 + \\
& x183187 + x184189 + x185193 + x186194 + x187190 + x188189 + \\
& x188191 + x188193 + x189191 + x189192 + x190192 + x190194 + \\
& x191192 \leq 161
\end{aligned}
$$

$$\begin{aligned} c_3: \;& x100108 + x100110 + x105106 + x105107 + x106118 + x107108 + \\ & x109113 + x109114 + x110112 + x112115 + x113114 + x113119 + \\ & x114119 + x115116 + x116117 + x117121 + x118122 + x118131 + \\ & x119122 + x120121 + x120123 + x123124 + x124128 + x128133 + \\ & x129131 + x129133 + x129135 + x131136 + x133135 + x135136 \leq 25 \end{aligned}$$

Here, c_2 and c_3 are constraints for the subtour with 162 and 26 vertices respectively, where

$$S_3 = \{100\ 108\ 110\ 107\ 112\ 105\ 115\ 106\ 116\ 118\ 117\ 122\ 121\ 119\ 120\ 113\ 114\ 123\ 109\ 124\ 128\ 133\ 129\ 135\ 131\ 136\}$$

and,

$$S_2 = \{V - (S_1 \cup S_3)\}$$

Adding c_1, c_2 and c_3 constraints to the LP, as a result, improved the lower bound which became (9281). However, the addition of these constraints leaded to another two subsets, as the following figure shows:

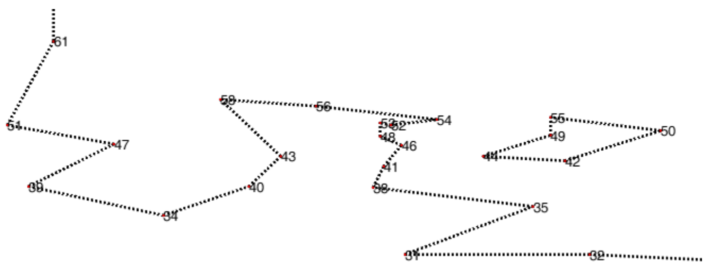


Figure 3.3: A part of the graph G shows S_4

Figure (3.3) shows the new subtour appearing in G. Thus, we repeat the same procedure with these two new subtours by adding two SECs to the list of constraints. The first, or small, subtour has five vertices $S_4 = \{42\ 44\ 50\ 49\ 55\}$. Thus, we add

$$c_4: \; x4244 + x4249 + x4250 + x4449 + x4950 + x4955 + x5055 \leq 4$$

And another constraint should be added which contains the rest of graph G's vertices, such that $S_5 = \{V - S_4\}$, as follows:

$$c_5: x0104 + x0106 + x0203 + x0204 + x0305 + x0509 + x0608 + x0711 + x0717+$$
$$x0816 + x0910 + x1012 + x1114 + x1215 + x1316 + x1323 + x1425 + x1519+$$
$$x1726 + x1821 + x1833 + x1930 + x2063 + x2065 + x2124 + x2227 + x2229+$$
$$x2325 + x2426 + x2734 + x2737 + x2829 + x2833 + x3032 + x3132 + x3135+$$
$$x3439 + x3440 + x3538 + x3659 + x3663 + x3739 + x3745 + x3840 + x3841+$$
$$x3947 + x3951 + x4043 + x4143 + x4146 + x4347 + x4356 + x4358 + x4557+$$
$$x4648 + x4654 + x4751 + x4852 + x4853 + x4854 + x4856 + x5161 + x5253+$$
$$x5254 + x5356 + x5456 + x5658 + x5760 + x5861 + x5962 + x6069 + x6167+$$
$$x6282 + x6468 + x6470 + x6585 + x6668 + x6673 + x6773 + x6974 + x7077+$$
$$x7176 + x7180 + x7182 + x7274 + x7275 + x7278 + x7576 + x7578 + x7680+$$
$$x7687 + x7784 + x7891 + x7981 + x7983 + x8087 + x8184 + x8388 + x8586+$$
$$x8698 + x87102 + x8892 + x8893 + x8990 + x8994 + x9098 + x9193 + x91103+$$
$$x9295 + x9297 + x9396 + x9499 + x9596 + x9597 + x97106 + x99101+$$
$$x102103 + x102109 + x103106 + x104111 + x105106 + x105107 + x106118+$$
$$x107108 + x109113 + x109114 + x110112 + x111114 + x111130 + x112115+$$
$$x113114 + x113119 + x114119 + x114125 + x114126 + x115116 + x116117+$$
$$x117121 + x118122 + x118131 + x119122 + x119126 + x120121 + x120123+$$
$$x123124 + x124128 + x125126 + x125127 + x126132 + x126138 + x127130+$$
$$x127132 + x128133 + x129131 + x129133 + x129135 + x130132 + x130156+$$
$$x131136 + x132134 + x133135 + x134137 + x135136 + x135143 + x136143+$$
$$x136155 + x137140 + x138139 + x138142 + x139146 + x139154 + x140142+$$
$$x140145 + x141144 + x141147 + x141152 + x142146 + x143148 + x143160+$$
$$x144150 + x145149 + x145156 + x146149 + x147151 + x147152 + x148155+$$
$$x148160 + x149156 + x150153 + x150154 + x151155 + x152153 + x152159+$$
$$x153157 + x154157 + x155158 + x155162 + x156161 + x158159 + x158162+$$
$$x159165 + x160166 + x161163 + x161169 + x162167 + x163164 + x164169+$$
$$x164172 + x165168 + x166171 + x167168 + x167170 + x168178 + x169176+$$
$$x170171 + x170180 + x171185 + x172174 + x172179 + x173174 + x173175+$$
$$x174179 + x175177 + x175184 + x176182 + x177181 + x177184 + x178180+$$
$$x178181 + x179186 + x180185 + x181184 + x182194 + x183186 + x183187+$$
$$x184189 + x185193 + x186194 + x187190 + x188189 + x188191 + x188193+$$
$$x100108 + x100110 + x101104 + x189191 + x189192 + x190192 + x190194+$$
$$x191192 \leq 188$$

The result after adding c_4 and c_5 together increased the LP value to (9282). Yet, the LP solution x^* is disconnected, as the next figure shows:

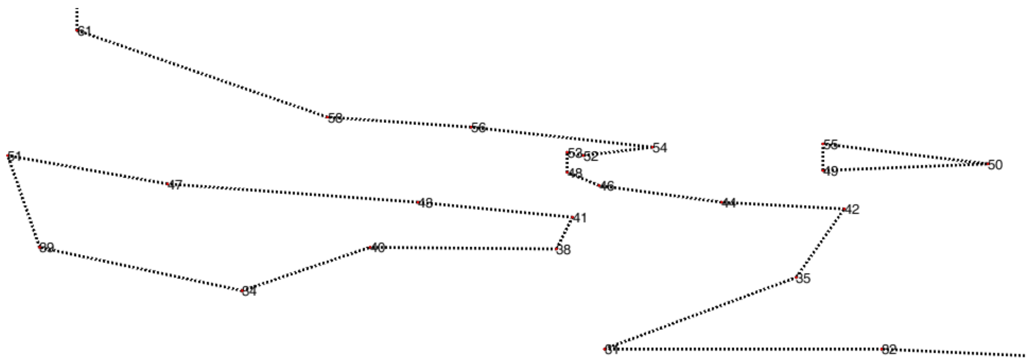


Figure 3.4: A part of the graph G shows S_6 and S_7

From figure (3.4), there are three subtours. Similarly, the next three iterations are completed by adding three more subtour elimination constraints to the list of constraints, one with each of the following,

$$S_6 = \{49\ 50\ 55\}$$

$$S_7 = \{34\ 39\ 40\ 51\ 38\ 47\ 41\ 43\}$$

$$S_8 = \{V - (S_6 \cup S_7)\}$$

. So, we have

$$c_6 : \ x4950 + x4955 + x5055 \leq 2$$

$$c_7 : \ x3439 + x3440 + x3840 + x3841 + x3947 + x3951 + x4043 + x4143 + x4347 + x4751 \leq 7$$

$$
\begin{aligned}
c_8:\ & x0104 + x0106 + x0203 + x0204 + x0305 + x0509 + x0608 + x0711 + \\
& x0717 + x0816 + x0910 + x1012 + x1114 + x1215 + x1316 + x1323 + \\
& x1425 + x1519 + x1726 + x1821 + x1833 + x1930 + x2063 + x2065 + \\
& x2124 + x2227 + x2229 + x2325 + x2426 + x2737 + x2829 + x2833 + \\
& x3032 + x3132 + x3135 + x3542 + x3544 + x3659 + x3663 + x3745 + \\
& x4244 + x4446 + x4448 + x4557 + x4648 + x4654 + x4852 + x4853 + \\
& x4854 + x4856 + x5253 + x5254 + x5356 + x5456 + x5658 + x5760 + \\
& x5861 + x5962 + x6069 + x6167 + x6282 + x6468 + x6470 + x6585 + \\
& x6668 + x6673 + x6773 + x6974 + x7077 + x7176 + x7180 + x7182 + \\
& x7274 + x7275 + x7278 + x7576 + x7578 + x7680 + x7687 + x7784 + \\
& x7891 + x7981 + x7983 + x8087 + x8184 + x8388 + x8586 + x8698 + \\
& x87102 + x8892 + x8893 + x8990 + x8994 + x9098 + x9193 + x91103 + \\
& x9295 + x9297 + x9396 + x9499 + x9596 + x9597 + x97106 + x99101 + \\
& x100108 + x100110 + x101104 + x102103 + x102109 + x103106 + \\
& x104111 + x105106 + x105107 + x106118 + x107108 + x109113 + x109114 + \\
& x110112 + x111114 + x111130 + x112115 + x113114 + x113119 + x114119 + \\
& x114125 + x114126 + x115116 + x116117 + x117121 + x118122 + x118131 + \\
& x119122 + x119126 + x120121 + x120123 + x123124 + x124128 + x125126 + \\
& x125127 + x126132 + x126138 + x127130 + x127132 + x128133 + x129131 + \\
& x129133 + x129135 + x130132 + x130156 + x131136 + x132134 + x133135 + \\
& x134137 + x135136 + x135143 + x136143 + x136155 + x137140 + x138139 + \\
& x138142 + x139146 + x139154 + x140142 + x140145 + x141144 + x141147 + \\
& x141152 + x142146 + x143148 + x143160 + x144150 + x145149 + x145156 + \\
& x146149 + x147151 + x147152 + x148155 + x148160 + x149156 + x150153 + \\
& x150154 + x151155 + x152153 + x152159 + x153157 + x154157 + x155158 + \\
& x155162 + x156161 + x158159 + x158162 + x159165 + x160166 + x161163 + \\
& x161169 + x162167 + x163164 + x164169 + x164172 + x165168 + x166171 + \\
& x167168 + x167170 + x168178 + x169176 + x170171 + x170180 + x171185 + \\
& x172174 + x172179 + x173174 + x173175 + x174179 + x175177 + x175184 + \\
& x176182 + x177181 + x177184 + x178180 + x178181 + x179186 + x180185 + \\
& x181184 + x182194 + x183186 + x183187 + x184189 + x185193 + x186194 + \\
& x187190 + x188189 + x188191 + x188193 + x189191 + x189192 + x190192 + \\
& x190194 + x191192 \leq 182
\end{aligned}
$$

So far, we have added eight constraints to the LP. The last three iterations raised the lower bound to (9284). It also, however, generated another two subtours, one with $S_9 = \{42\ 49\ 50\ 55\}$, the other with $S_{10} = \{V - S_9\}$. The following figure shows S_9:

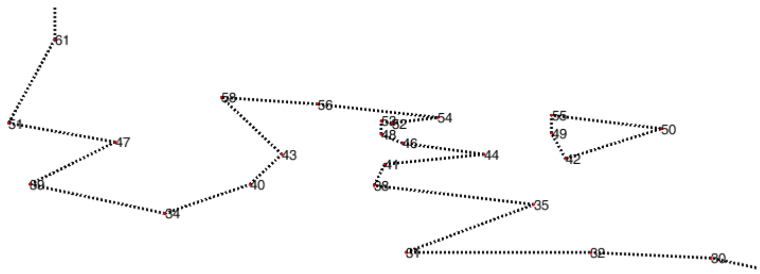


Figure 3.5: A part of the graph G shows S_9

Similarly, we add two more subtour elimination constraints c_9 and c_{10}, thus,

$$c_9: \ x4249 + x4250 + x4950 + x4955 + x5055 \leq 3$$

and,

$$\begin{aligned} c_{10}: \ & x0104 + x0106 + x0203 + x0204 + x0305 + x0509 + x0608 + x0711 + \\ & x0717 + x0816 + x0910 + x1012 + x1114 + x1215 + x1316 + x1323 + \\ & x1425 + x1519 + x1726 + x1821 + x1833 + x1930 + x2063 + x2065 + \\ & x2124 + x2227 + x2229 + x2325 + x2426 + x2734 + x2737 + x2829 + \\ & x2833 + x3032 + x3132 + x3135 + x3439 + x3440 + x3538 + x3544 + \\ & x3659 + x3663 + x3739 + x3745 + x3840 + x3841 + x3844 + x3947 + \\ & x3951 + x4043 + x4143 + x4144 + x4146 + x4347 + x4356 + x4358 + \\ & x4446 + x4448 + x4557 + x4648 + x4654 + x4751 + x4852 + x4853 + \\ & x4854 + x4856 + x5161 + x5253 + x5254 + x5356 + x5456 + x5658 + \\ & x5760 + x5861 + x5962 + x6069 + x6167 + x6282 + x6468 + x6470 + \\ & x6585 + x6668 + x6673 + x6773 + x6974 + x7077 + x7176 + x7180 + \end{aligned}$$

$$x7182 + x7274 + x7275 + x7278 + x7576 + x7578 + x7680 + x7687 + x7784+$$
$$x7891 + x7981 + x7983 + x8087 + x8184 + x8388 + x8586 + x8698 + x87102+$$
$$x8892 + x8893 + x8990 + x8994 + x9098 + x9193 + x91103 + x9295 + x9297+$$
$$x9396 + x9499 + x9596 + x9597 + x97106 + x99101 + x100108 + x100110+$$
$$x101104 + x102103 + x102109 + x103106 + x104111 + x105106 + x105107+$$
$$x106118 + x107108 + x109113 + x109114 + x110112 + x111114 + x111130+$$
$$x112115 + x113114 + x113119 + x114119 + x114125 + x114126 + x115116+$$
$$x116117 + x117121 + x118122 + x118131 + x119122 + x119126 + x120121+$$
$$x120123 + x123124 + x124128 + x125126 + x125127 + x126132 + x126138+$$
$$x127130 + x127132 + x128133 + x129131 + x129133 + x129135 + x130132+$$
$$x130156 + x131136 + x132134 + x133135 + x134137 + x135136 + x135143+$$
$$x136143 + x136155 + x137140 + x138139 + x138142 + x139146 + x139154+$$
$$x140142 + x140145 + x141144 + x141147 + x141152 + x142146 + x143148+$$
$$x143160 + x144150 + x145149 + x145156 + x146149 + x147151 + x147152+$$
$$x148155 + x148160 + x149156 + x150153 + x150154 + x151155 + x152153+$$
$$x152159 + x153157 + x154157 + x155158 + x155162 + x156161 + x158159+$$
$$x158162 + x159165 + x160166 + x161163 + x161169 + x162167 + x163164+$$
$$x164169 + x164172 + x165168 + x166171 + x167168 + x167170 + x168178+$$
$$x169176 + x170171 + x170180 + x171185 + x172174 + x172179 + x173174+$$
$$x173175 + x174179 + x175177 + x175184 + x176182 + x177181 + x177184+$$
$$x178180 + x178181 + x179186 + x180185 + x181184 + x182194 + x183186+$$
$$x183187 + x184189 + x185193 + x186194 + x187190 + x188189 + x188191+$$
$$x188193 + x189191 + x189192 + x190192 + x190194 + x191192 \leq 189$$

Solving the LP with the previous ten constraints raised the lower bound to (9284.5) and led to the following graph:

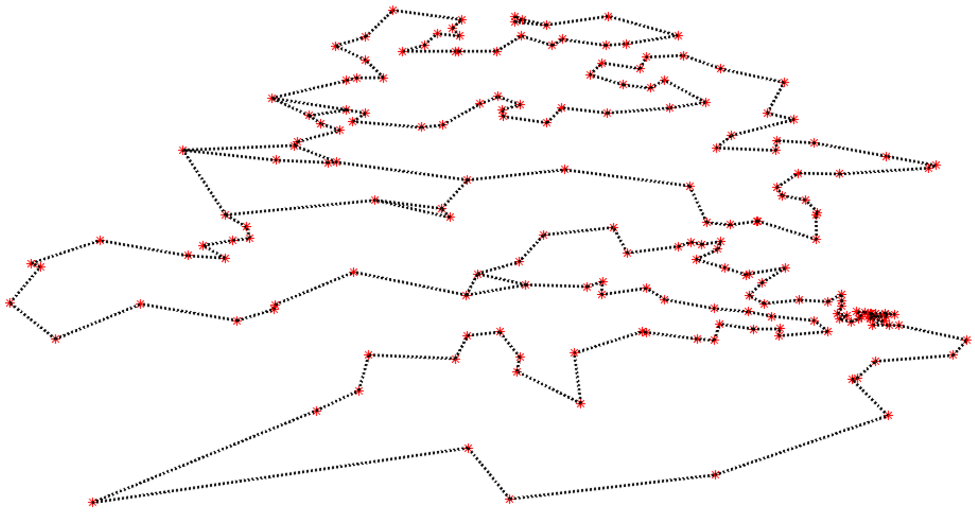


Figure 3.6: The LP solution with 10 SEC

The graph G in Figure (3.6) presents an updated version of the LP solution x^*. The solution becomes connected now; however, it is not 2-connected. That means, with the removal of vertex 188, the graph becomes two connected subsets, as the next figure illustrates:

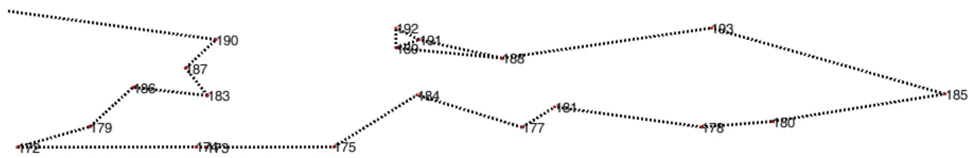


Figure 3.7: S_{11} before remove $x188193$

Figure (3.7) shows how removing of the edge $x188193$ splits the graph into two connected components: one with cities $S_{11} = \{188\ 189\ 191\ 192\}$ and the other with cities $S_{12} = \{V - S_{11}\}$. Since dividing the graph into connected components with city sets $S_1, S_2, ..., S_k$ ($k \geq 2$), it is possible by removing a single city, 188 in this case, then

$$\sum (x_e^* \ : e \textit{ has one end in } S_i \textit{ and one end not in } S_i) \leq 1 \tag{3.2}$$

for at least one set S_i. Back to the same method, we add two more subtour elimination constraints c_{11} and c_{12} for S_{11} and S_{12} respectively. Thus:

$$c_{11}: \ x188189 + x188191 + x189191 + x189192 + x191192 \leq 3$$

$$
\begin{aligned}
c_{12}:\ & x0104+x0106+x0203+x0204+x0305+x0509+x0608+x0711+\\
& x0717+x0816+x0910+x1012+x1114+x1215+x1316+x1323+\\
& x1425+x1519+x1726+x1821+x1833+x1930+x2063+x2065+\\
& x2124+x2227+x2229+x2325+x2426+x2734+x2737+x2829+\\
& x2833+x3032+x3132+x3135+x3439+x3440+x3538+x3542+\\
& x3544+x3659+x3663+x3739+x3745+x3840+x3841+x3844+\\
& x3947+x3951+x4043+x4143+x4144+x4146+x4244+x4249+\\
& x4250+x4347+x4356+x4358+x4446+x4448+x4449+x4557+\\
& x4648+x4654+x4751+x4852+x4853+x4854+x4856+x4950+\\
& x4955+x5055+x5161+x5253+x5254+x5356+x5455+x5456+\\
& x5658+x5760+x5861+x5962+x6069+x6167+x6282+x6468+\\
& x6470+x6585+x6668+x6673+x6773+x6974+x7077+x7176+\\
& x7180+x7182+x7274+x7275+x7278+x7576+x7578+x7680+\\
& x7687+x7784+x7891+x7981+x7983+x8087+x8184+x8388+\\
& x8586+x8698+x87102+x8892+x8893+x8990+x8994+x9098+\\
& x9193+x91103+x9295+x9297+x9396+x9499+x9596+x9597+\\
& x97106+x99101+x100108+x100110+x101104+x102103+x102109+\\
& x103106+x104111+x105106+x105107+x106118+x107108+x109113+\\
& x109114+x110112+x111114+x111130+x112115+x113114+x113119+\\
& x114119+x114125+x114126+x115116+x116117+x117121+x118122+\\
& x118131+x119122+x119126+x120121+x120123+x123124+x124128+\\
& x125126+x125127+x126132+x126138+x127130+x127132+x128133+\\
& x129131+x129133+x129135+x130132+x130156+x131136+x132134+\\
& x133135+x134137+x135136+x135143+x136143+x136155+x137140+\\
& x138139+x138142+x139146+x139154+x140142+x140145+x141144+\\
& x141147+x141152+x142146+x143148+x143160+x144150+x145149+\\
& x145156+x146149+x147151+x147152+x148155+x148160+x149156+\\
& x150153+x150154+x151155+x152153+x152159+x153157+x154157+\\
& x155158+x155162+x156161+x158159+x158162+x159165+x160166+\\
& x161163+x161169+x162167+x163164+x164169+x164172+x165168+\\
& x166171+x167168+x167170+x168178+x169176+x170171+x170180+
\end{aligned}
$$

$x171185 + x172174 + x172179 + x173174 + x173175 + x174179 + x175177+$
$x175184 + x176182 + x177181 + x177184 + x178180 + x178181 + x179186+$
$x180185 + x181184 + x182194 + x183186 + x183187 + x185193 + x186194+$
$x187190 + x190194 \leq 189$

Adding two more subtour elimination constraints improved the lower bound to (9301) and the optimal solution x^* is drawn in the following figure:

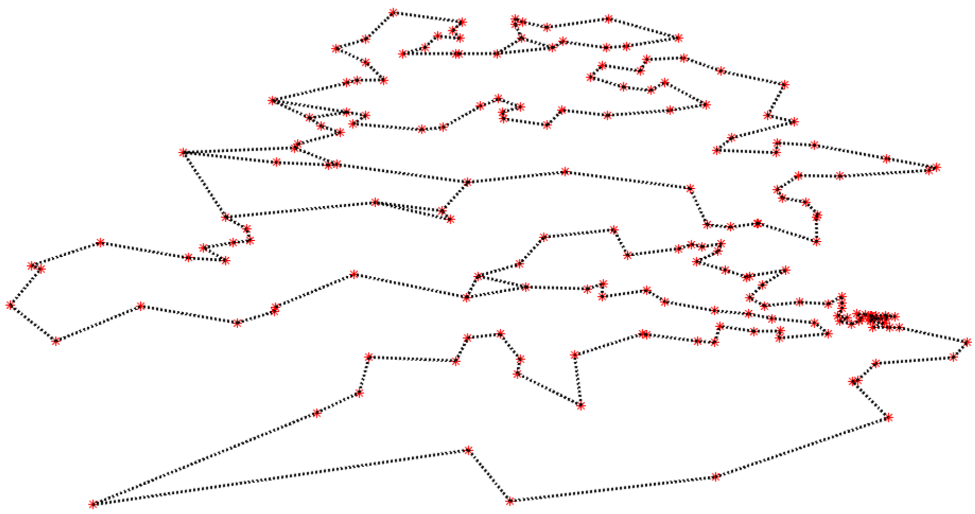


Figure 3.8: The LP solution with 12 SEC

Again, the graph is not 2-connected as we could divide it into two connected sets by removing the city 175 as the next figure clarifies:

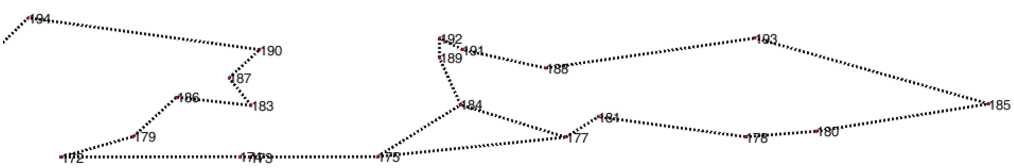


Figure 3.9: S_{13} before remove $x173175$

Splitting the graph G by removing the edge $x173175$ generates two subsets, one with vertices $S_{13} = \{175\ 177\ 184\ 181\ 189\ 178\ 192\ 180\ 191\ 185\ 188\ 193\}$ and the other with vertices $S_{14} = \{V - S_{13}\}$. The next two constraints are given below:

$$c_{13}: \; x175177 + x175184 + x177181 + x177184 + x178180 + x178181 + x180185 + x181184 + x184189 + x185193 + x188189 + x188191 + x188193 + x189191 + x189192 + x191192 \leq 11$$

$$c_{14}: \; x0104 + x0106 + x0203 + x0204 + x0305 + x0509 + x0608 + x0711 + x0717 + x0816 + x0910 + x1012 + x1114 + x1215 + x1316 + x1323 + x1425 + x1519 + x1726 + x1821 + x1833 + x1930 + x2063 + x2065 + x2124 + x2227 + x2229 + x2325 + x2426 + x2734 + x2737 + x2829 + x2833 + x3032 + x3132 + x3135 + x3439 + x3440 + x3538 + x3542 + x3544 + x3659 + x3663 + x3739 + x3745 + x3840 + x3841 + x3844 + x3947 + x3951 + x4043 + x4143 + x4144 + x4146 + x4244 + x4249 + x4250 + x4347 + x4356 + x4358 + x4446 + x4448 + x4449 + x4557 + x4648 + x4654 + x4751 + x4852 + x4853 + x4854 + x4856 + x4950 + x4955 + x5055 + x5161 + x5253 + x5254 + x5356 + x5455 + x5456 + x5658 + x5760 + x5861 + x5962 + x6069 + x6167 + x6282 + x6468 + x6470 + x6585 + x6668 + x6673 + x6773 + x6974 + x7077 + x7176 + x7180 + x7182 + x7274 + x7275 + x7278 + x7576 + x7578 + x7680 + x7687 + x7784 + x7891 + x7981 + x7983 + x8087 + x8184 + x8388 + x8586 + x8698 + x87102 + x8892 + x8893 + x8990 + x8994 + x9098 + x9193 + x91103 + x9295 + x9297 + x9396 + x9499 + x9596 + x9597 + x97106 + x99101 + x100108 + x100110 + x101104 + x102103 + x102109 + x103106 + x104111 + x105106 + x105107 + x106118 + x107108 + x109113 + x109114 + x110112 + x111114 + x111130 + x112115 + x113114 + x113119 + x114119 + x114125 + x114126 + x115116 + x116117 + x117121 + x118122 + x118131 + x119122 + x119126 + x120121 + x120123 + x123124 + x124128 + x125126 + x125127 + x126132 + x126138 + x127130 + x127132 + x128133 + x129131 + x129133 + x129135 + x130132 + x130156 + x131136 + x132134 + x133135 + x134137 + x135136 + x135143 + x136143 + x136155 + x137140 + x138139 + x138142 + x139146 + x139154 + x140142 + x140145 + x141144 + x141147 + x141152 + x142146 + x143148 + x143160 + x144150 + x145149 + x145156 + x146149 + x147151 + x147152 + x148155 + x148160 + x149156 + x150153 + x150154 + x151155 + x152153 + x152159 + x153157 + x154157 +$$

$x155158 + x155162 + x156161 + x158159 + x158162 + x159165 + x160166 +$
$x161163 + x161169 + x162167 + x163164 + x164169 + x164172 + x165168 +$
$x166171 + x167168 + x167170 + x169176 + x170171 + x172174 + x172179 +$
$x173174 + x174179 + x176182 + x179186 + x182194 + x183186 + x183187 +$
$x186194 + x187190 + x190194 \leq 181$

While the LP value grows to (9,307.75), the result of adding the last two linear inequalities appears in the next two figures:

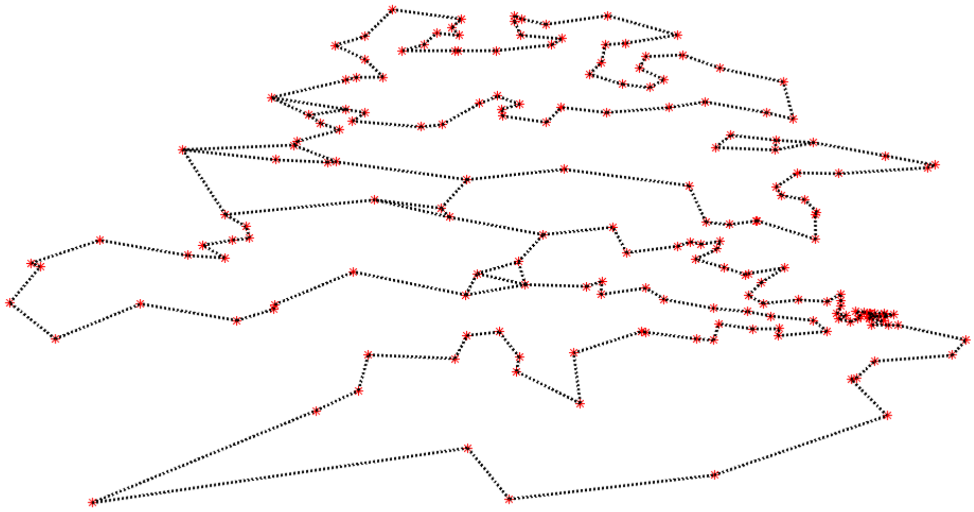


Figure 3.10: The LP solution with 14 SEC

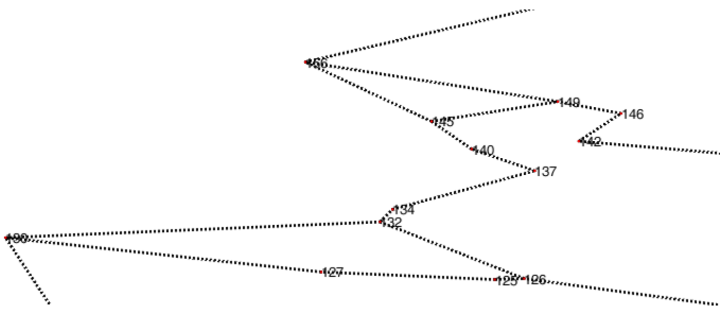


Figure 3.11: S_{15} before remove $x140145$

Now, the LP solution x^* again becomes not 2-connected, as figure (3.11) confirms. Similarly, taking off the city 145 produces two subsets, one with 56 cities $S_{15} = \{145\ 149\ 146\ 142\ 138\ ...\ 163\ 161\ 156\}$ and the other one with 138 cities $S_{16} = \{V - S_{15}\}$. Thus, we then add the following:

$$c_{15}: \; x138139 + x138142 + x139146 + x139154 + x141144 + x141147 + x141152 + x142146 + x143148 + x143160 + x144150 + x145149 + x145156 + x146149 + x147151 + x147152 + x148155 + x148160 + x149156 + x150153 + x150154 + x151155 + x152153 + x152159 + x153157 + x154157 + x155158 + x155162 + x156161 + x158159 + x158162 + x159165 + x160166 + x161163 + x161169 + x162167 + x163164 + x164169 + x164172 + x165168 + x166171 + x167168 + x167170 + x168178 + x169176 + x170171 + x170180 + x171185 + x172174 + x172179 + x173174 + x173175 + x174179 + x175177 + x175184 + x176182 + x177181 + x177184 + x178180 + x178181 + x179186 + x180185 + x181184 + x182194 + x183186 + x183187 + x184189 + x185193 + x186194 + x187190 + x188189 + x188191 + x188193 + x189191 + x189192 + x190192 + x190194 + x191192 \leq 55$$

$$c_{16}: \; x0104 + x0106 + x0203 + x0204 + x0305 + x0509 + x0608 + x0711 + x0717 + x0816 + x0910 + x1012 + x1114 + x1215 + x1316 + x1323 + x1425 + x1519 + x1726 + x1821 + x1833 + x1930 + x2063 + x2065 + x2124 + x2227 + x2229 + x2325 + x2426 + x2734 + x2737 + x2829 + x2833 + x3032 + x3132 + x3135 + x3439 + x3440 + x3538 + x3542 + x3544 + x3659 + x3663 + x3739 + x3745 + x3840 + x3841 + x3844 + x3947 + x3951 + x4043 + x4143 + x4144 + x4146 + x4244 + x4249 + x4250 + x4347 + x4356 + x4358 + x4446 + x4448 + x4449 + x4557 + x4648 + x4654 + x4751 + x4852 + x4853 + x4854 + x4856 + x4950 + x4955 + x5055 + x5161 + x5253 + x5254 + x5356 + x5455 + x5456 + x5658 + x5760 + x5861 + x5962 + x6069 + x6167 + x6282 + x6468 + x6470 + x6585 + x6668 + x6673 + x6773 + x6974 + x7077 + x7176 + x7180 + x7182 + x7274 + x7275 + x7278 + x7576 + x7578 + x7680 + x7687 + x7784 + x7891 + x7981 + x7983 + x8087 + x8184 + x8388 + x8586 + x8698 + x87102 + x8892 + x8893 + x8990 + x8994 + x9098 + x9193 + x91103 + x9295 + x9297 + x9396 + x9499 + x9596 + x9597 + x97106 + x99101 + x100108 + x100110 + x101104 + x102103 + x102109 +$$

$$x103106 + x104111 + x105106 + x105107 + x106118 + x107108+$$
$$x109113 + x109114 + x110112 + x111114 + x111130 + x112115+$$
$$x113114 + x113119 + x114119 + x114125 + x114126 + x115116+$$
$$x116117 + x117121 + x118122 + x118131 + x119122 + x119126+$$
$$x120121 + x120123 + x123124 + x124128 + x125126 + x125127+$$
$$x126132 + x127130 + x127132 + x128133 + x129131 + x129133+$$
$$x129135 + x130132 + x131136 + x132134 + x133135 + x134137+$$
$$x135136 + x137140 \leq 137$$

Consequently, the lower bound increased to become (9308) and the new LP solution with 16 SEC is given by

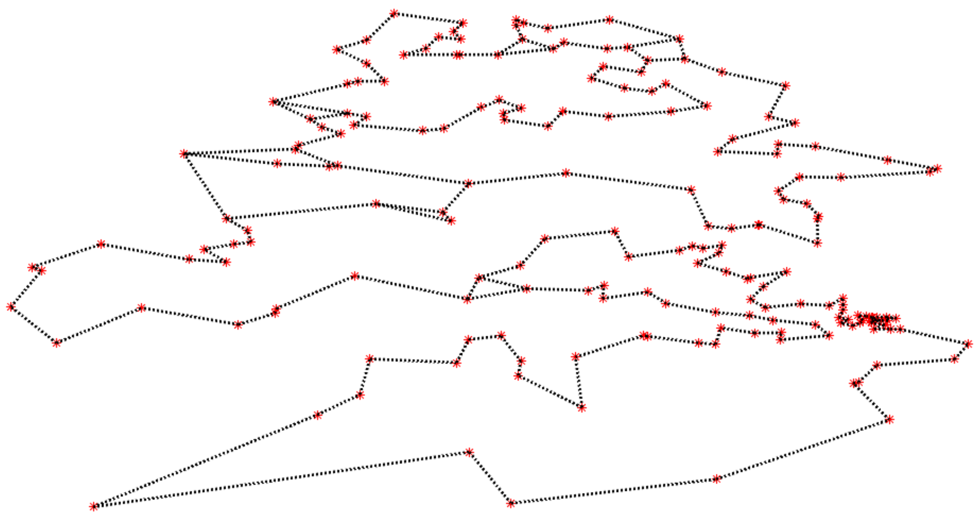


Figure 3.12: The LP solution with 16 SEC

As can be seen in the mid-left figure (3.12), the solution is still not 2-connected, therefore we continue adding subtour elimination constraints after the removal of city 111 (the edge $x104111$). There are two connected components, one contains 91 cities $S_{17} = \{111\ 130\ ...\ 114\}$, and the second one has 103 cities $S_{18} = \{V - S_{17}\}$. The next two constraints are listed below:

$c_{17}:$ $x100108 + x100110 + x105106 + x105107 + x106118 + x107108+$
$x109113 + x109114 + x110112 + x111114 + x111130 + x112115+$
$x113114 + x113119 + x114119 + x114125 + x114126 + x115116+$
$x116117 + x117121 + x118122 + x118131 + x119122 + x119126+$
$x120121 + x120123 + x123124 + x124128 + x125126 + x125127+$
$x126132 + x126138 + x127130 + x127132 + x128133 + x129131+$
$x129133 + x129135 + x130132 + x130156 + x131136 + x132134+$
$x133135 + x134137 + x135136 + x135143 + x136143 + x136155+$
$x137140 + x138139 + x138142 + x139146 + x139154 + x140142+$
$x140145 + x141144 + x141147 + x141152 + x142146 + x143148+$
$x143160 + x144150 + x145149 + x145156 + x146149 + x147151+$
$x147152 + x148155 + x148160 + x149156 + x150153 + x150154+$
$x151155 + x152153 + x152159 + x153157 + x154157 + x155158+$
$x155162 + x156161 + x158159 + x158162 + x159165 + x160166+$
$x161163 + x161169 + x162167 + x163164 + x164169 + x164172+$
$x165168 + x166171 + x167168 + x167170 + x168178 + x169176+$
$x170171 + x170180 + x171185 + x172174 + x172179 + x173174+$
$x173175 + x174179 + x175177 + x175184 + x176182 + x177181+$
$x177184 + x178180 + x178181 + x179186 + x180185 + x181184+$
$x182194 + x183186 + x183187 + x184189 + x185193 + x186194+$
$x187190 + x188189 + x188191 + x188193 + x189191 + x189192+$
$x190192 + x190194 + x191192 \leq 90$

$c_{18}:$ $x0104 + x0106 + x0203 + x0204 + x0305 + x0509 + x0608+$
$x0711 + x0717 + x0816 + x0910 + x1012 + x1114 + x1215+$
$x1316 + x1323 + x1425 + x1519 + x1726 + x1821 + x1833+$
$x1930 + x2063 + x2065 + x2124 + x2227 + x2229 + x2325+$
$x2426 + x2734 + x2737 + x2829 + x2833 + x3032 + x3132+$
$x3135 + x3439 + x3440 + x3538 + x3542 + x3544 + x3659+$
$x3663 + x3739 + x3745 + x3840 + x3841 + x3844 + x3947+$
$x3951 + x4043 + x4143 + x4144 + x4146 + x4244 + x4249+$
$x4250 + x4347 + x4356 + x4358 + x4446 + x4448 + x4449+$
$x4557 + x4648 + x4654 + x4751 + x4852 + x4853 + x4854+$

$$
\begin{aligned}
&x4856 + x4950 + x4955 + x5055 + x5161 + x5253 + x5254+\\
&x5356 + x5455 + x5456 + x5658 + x5760 + x5861 + x5962+\\
&x6069 + x6167 + x6282 + x6468 + x6470 + x6585 + x6668+\\
&x6673 + x6773 + x6974 + x7077 + x7176 + x7180 + x7182+\\
&x7274 + x7275 + x7278 + x7576 + x7578 + x7680 + x7687+\\
&x7784 + x7891 + x7981 + x7983 + x8087 + x8184 + x8388+\\
&x8586 + x8698 + x87102 + x8892 + x8893 + x8990 + x8994+\\
&x9098 + x9193 + x91103 + x9295 + x9297 + x9396 + x9499+\\
&x9596 + x9597 + x99101 + x101104 + x102103 \leq 102
\end{aligned}
$$

With adding the last two subtour elimination constraints, we obtain a relaxation of value (9,311.5) and the optimal solution $x^*$is presented in the following figure:

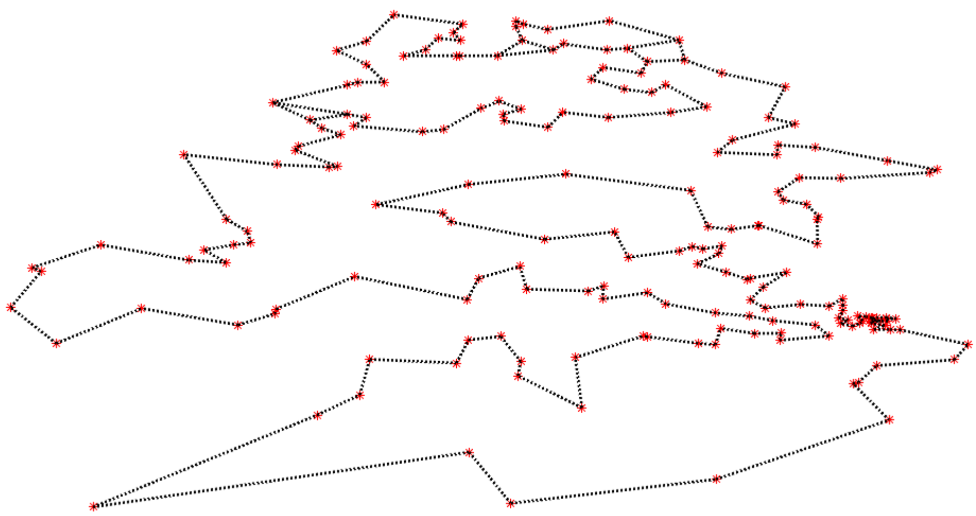


Figure 3.13: The LP solution with 18 SEC

Apparently, the solution is now becoming 2-connected and removing another city will not cause any division. Since the simplex algorithm is our approach to finding the optimal solution x^* and the solution is not a tour yet, there must be a linear inequality satisfied by all x in S and violated by x^* (Applegate et al., 2011). The inequality is called a *cutting plane*, or simply a *cut*.

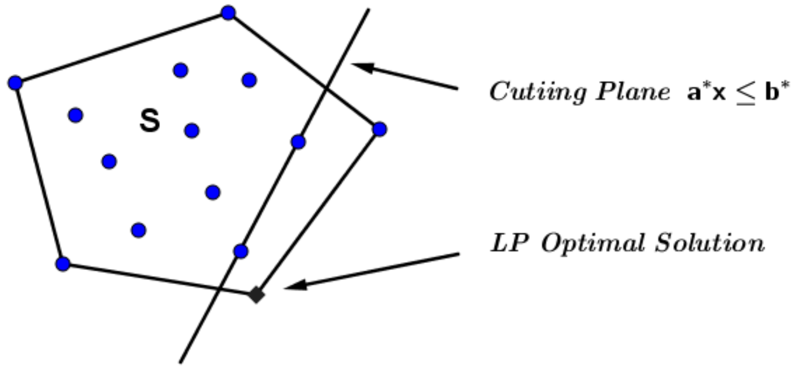


Figure 3.14: Cutting plane to narrow down the feasible set

The cut takes the form $a^*x \leq b^*$, where a^* is a new row of the matrix A, and b^* is a new element of the vector b. Adding a cut pushes up the lower bound and narrows down our feasible set S for the LP relaxation. We repeat this procedure until get a relaxation to the problem in the form:

$$\textit{minimise } c^T x \textit{ subject to } a^* x \leq b^* \tag{3.3}$$

with the solution obtained by the *cutting plane* method. The next step of our approach is to add cuts, or linear inequalities, to raise the lower bound. A common way to find cuts is by using *Comb inequality* (Grötschel & Padberg, 1979). Any family of subsets of V that consists of a single "handle" and a set of "teeth" and that is characterised by

- the handle meeting each tooth but not containing any teeth,
- the teeth being pairwise disjoint,
- and the number of teeth k being an odd integer number and $k \geq 3$

is called a *comb*. For every comb with handle H and teeth $T_1, T_2, ..., T_k$, the inequality

$$x(E(H)) + \sum_{i=1}^{k} x(E(T_i)) \leq |H| + \sum_{i=1}^{k} |T_i| - [\frac{3k+1}{2}] \tag{3.4}$$

is called a comb inequality (Letchford & Lodi, 2002). The optimal solution x^* drawn in figure (3.13) gives four combs, as shown in the next figure:

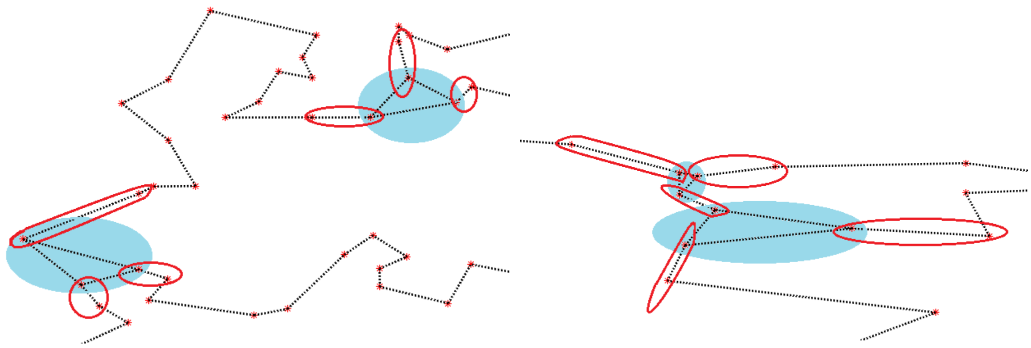


Figure 3.15: Parts of x^* where combs are found

Figure (3.15) shows the four combs, where the blue circles represent handles. Each of these handles are intersected by three teeth, determined by red ovals. We have to define these combs first and then write a linear inequality of each one using the inequality (3.4). The first comb contains the following:

$$H_1 = \{41\ 44\ 46\}, \quad T_1 = \{38\ 41\},\ T_2 = \{42\ 44\},\ T_3 = \{46\ 48\}$$

Using the comb inequality (3.4) where $|H|$ is the number of the handle's vertices $|H_1| = 3$, $|T_1| = |T_2| = |T_3| = 2$ and k is the number of teeth ($k = 3$). Thus, we can express the constraint as

$$c_{19}:\ x3841 + x4144 + x4146 + x4244 + x4446 + x4648 \leq 4$$

The second comb,

$$H_2 = \{48\ 52\ 53\}, \quad T_1 = \{46\ 48\},\ T_2 = \{52\ 54\},\ T_3 = \{53\ 56\}$$

Thus,

$$c_{20}:\ x4648 + x4852 + x4853 + x5253 + x5254 + x5356 \leq 4$$

The third comb

$$H_3 = \{145\ 149\ 156\}, \quad T_1 = \{140\ 145\},\ T_2 = \{146\ 149\},\ T_3 = \{156\ 161\}$$

So,

$$c_{21}:\ x140145 + x145149 + x145156 + x146149 + x149156 + x156161 \leq 4$$

Then, the last comb

$$H_4 = \{175\ 177\ 184\}, \quad T_1 = \{173\ 175\},\ T_2 = \{177\ 181\},\ T_3 = \{184\ 189\}$$

The comb inequality is written as

$$c_{22}: \; x173175 + x175177 + x175184 + x177181 + x177184 + x184189 \leq 4$$

The result of adding the last four linear inequalities to the list of constraints is that the lower bound increased to (9328.167) and the optimal solution x^* became disconnected, as the next two figures show:

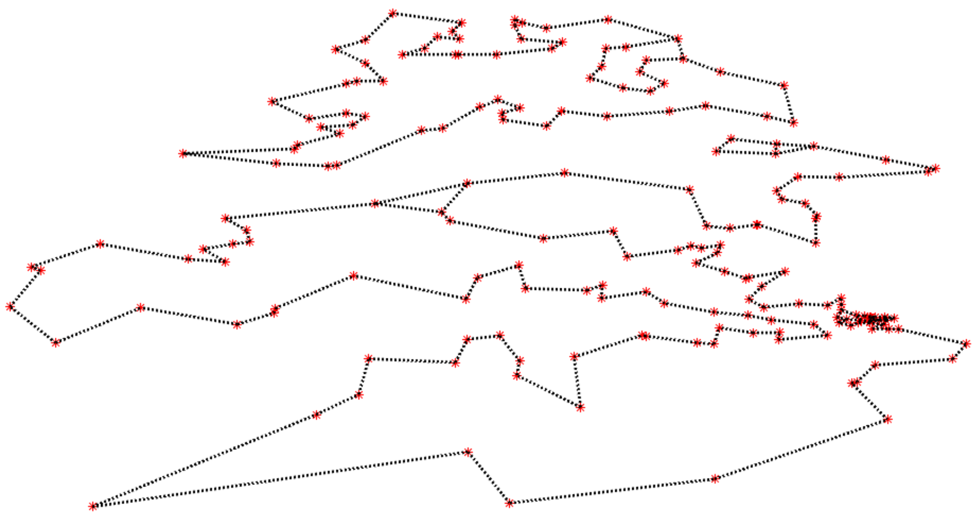


Figure 3.16: The LP solution with 18 SEC and 4 combs inequality

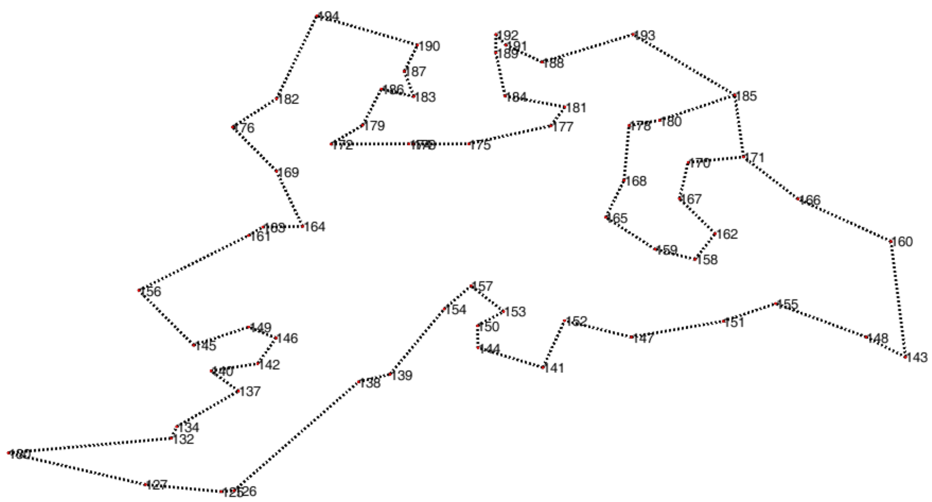


Figure 3.17: The upper part of Figure (3.16)

The next two iterations are added to eliminate these subtours using SEC (3.1). One of these subtours, figure (3.17), has 64 cities $S_{23} = \{130\,127...\,132\}$, and other one has 130 cities $S_{24} = \{V - S_{23}\}$. Thus, we add the following constraints:

$$
\begin{aligned}
c_{23}:\ & x125126 + x125127 + x126132 + x126138 + x127130 + x127132+\\
& x130132 + x130156 + x132134 + x134137 + x137140 + x138139+\\
& x138142 + x139146 + x139154 + x140142 + x140145 + x141144+\\
& x141147 + x141152 + x142146 + x143148 + x143160 + x144150+\\
& x145149 + x145156 + x146149 + x147151 + x147152 + x148155+\\
& x148160 + x149156 + x150153 + x150154 + x151155 + x152153+\\
& x152159 + x153157 + x154157 + x155158 + x155162 + x156161+\\
& x158159 + x158162 + x159165 + x160166 + x161163 + x161169+\\
& x162167 + x163164 + x164169 + x164172 + x165168 + x166171+\\
& x167168 + x167170 + x168178 + x169176 + x170171 + x170180+\\
& x171185 + x172174 + x172179 + x173174 + x173175 + x174179+\\
& x175177 + x175184 + x176182 + x177181 + x177184 + x178180+\\
& x178181 + x179186 + x180185 + x181184 + x182194 + x183186+\\
& x183187 + x184189 + x185193 + x186194 + x187190 + x188189+\\
& x188191 + x188193 + x189191 + x189192 + x190192 + x190194+\\
& x191192 \leq 63
\end{aligned}
$$

And,

$$
\begin{aligned}
c_{24}:\ & x0104 + x0106 + x0203 + x0204 + x0305 + x0509 + x0608 + x0711+\\
& x0717 + x0816 + x0910 + x1012 + x1114 + x1215 + x1316 + x1323+\\
& x1425 + x1519 + x1726 + x1821 + x1833 + x1930 + x2063 + x2065+\\
& x2124 + x2227 + x2229 + x2325 + x2426 + x2734 + x2737 + x2829+\\
& x2833 + x3032 + x3132 + x3135 + x3439 + x3440 + x3538 + x3542+\\
& x3544 + x3659 + x3663 + x3739 + x3745 + x3840 + x3841 + x3844+\\
& x3947 + x3951 + x4043 + x4143 + x4144 + x4146 + x4244 + x4249+\\
& x4250 + x4347 + x4356 + x4358 + x4446 + x4448 + x4449 + x4557+\\
& x4648 + x4654 + x4751 + x4852 + x4853 + x4854 + x4856 + x4950+
\end{aligned}
$$

$$x4955 + x5055 + x5161 + x5253 + x5254 + x5356 + x5455 + x5456+$$
$$x5658 + x5760 + x5861 + x5962 + x6069 + x6167 + x6282 + x6468+$$
$$x6470 + x6585 + x6668 + x6673 + x6773 + x6974 + x7077 + x7176+$$
$$x7180 + x7182 + x7274 + x7275 + x7278 + x7576 + x7578 + x7680+$$
$$x7687 + x7784 + x7891 + x7981 + x7983 + x8087 + x8184 + x8388+$$
$$x8586 + x8698 + x87102 + x8892 + x8893 + x8990 + x8994 + x9098+$$
$$x9193 + x91103 + x9295 + x9297 + x9396 + x9499 + x9596 + x9597+$$
$$x97106 + x99101 + x100108 + x100110 + x101104 + x102103+$$
$$x102109 + x103106 + x104111 + x105106 + x105107 + x106118+$$
$$x107108 + x109113 + x109114 + x110112 + x111114 + x112115+$$
$$x113114 + x113119 + x114119 + x115116 + x116117 + x117121+$$
$$x118122 + x118131 + x119122 + x120121 + x120123 + x123124+$$
$$x124128 + x128133 + x129131 + x129133 + x129135 + x131136+$$
$$x133135 + x135136 \leq 129$$

With adding two more subtour elimination constraints, the LP value rose to (9,336.667) and the LP solution became disconnected again. The LP solution now looks as follows:

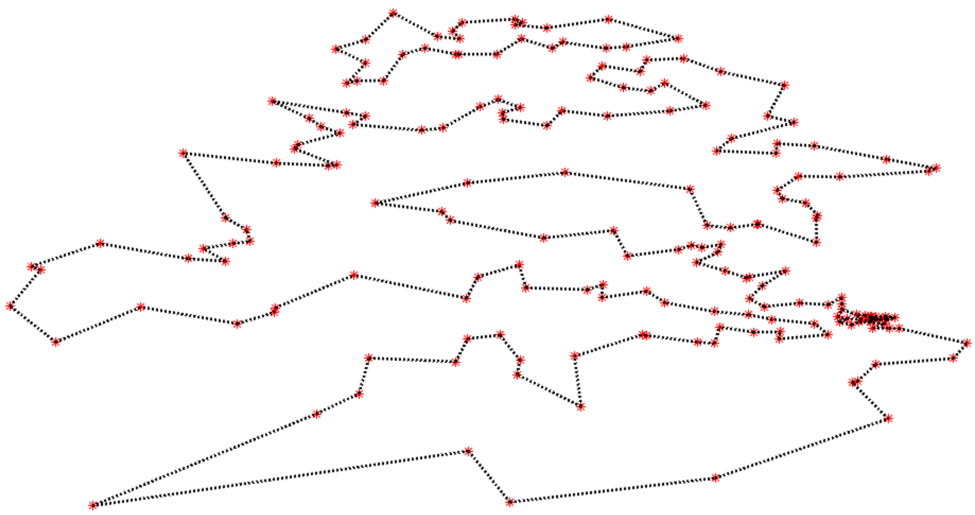


Figure 3.18: The LP solution with 20 SEC and 4 combs inequality

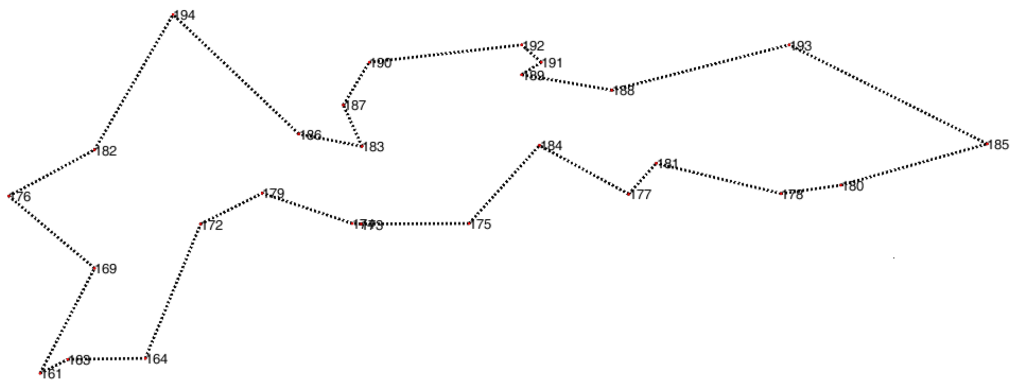


Figure 3.19: The top part of Figure (3.18)

We repeat the same argument of adding more subtour elimination constraints, one with 27 cities $S_{25} = \{185\ 193\ ...\ 178\ 180\}$, and the other one with 167 cities $S_{26} = \{V - S_{25}\}$. Thus, we write the SEC for them as follows:

$$\begin{aligned} c_{25}:\ & x161163 + x161169 + x163164 + x164169 + x164172 + x169176+ \\ & x172174 + x172179 + x173174 + x173175 + x174179 + x175177+ \\ & x175184 + x176182 + x177181 + x177184 + x178180 + x178181+ \\ & x179186 + x180185 + x181184 + x182194 + x183186 + x183187+ \\ & x184189 + x185193 + x186194 + x187190 + x188189 + x188191+ \\ & x188193 + x189191 + x189192 + x190192 + x190194 + x191192 \leq 26 \end{aligned}$$

And,

$$\begin{aligned} c_{26}:\ & x0104 + x0106 + x0203 + x0204 + x0305 + x0509 + x0608 + x0711+ \\ & x0717 + x0816 + x0910 + x1012 + x1114 + x1215 + x1316 + x1323+ \\ & x1425 + x1519 + x1726 + x1821 + x1833 + x1930 + x2063 + x2065+ \\ & x2124 + x2227 + x2229 + x2325 + x2426 + x2734 + x2737 + x2829+ \\ & x2833 + x3032 + x3132 + x3135 + x3439 + x3440 + x3538 + x3542+ \\ & x3544 + x3659 + x3663 + x3739 + x3745 + x3840 + x3841 + x3844+ \\ & x3947 + x3951 + x4043 + x4143 + x4144 + x4146 + x4244 + x4249+ \end{aligned}$$

$$x4250 + x4347 + x4356 + x4358 + x4446 + x4448 + x4449 + x4557+$$
$$x4648 + x4654 + x4751 + x4852 + x4853 + x4854 + x4856 + x4950+$$
$$x4955 + x5055 + x5161 + x5253 + x5254 + x5356 + x5455 + x5456+$$
$$x5658 + x5760 + x5861 + x5962 + x6069 + x6167 + x6282 + x6468+$$
$$x6470 + x6585 + x6668 + x6673 + x6773 + x6974 + x7077 + x7176+$$
$$x7180 + x7182 + x7274 + x7275 + x7278 + x7576 + x7578 + x7680+$$
$$x7687 + x7784 + x7891 + x7981 + x7983 + x8087 + x8184 + x8388+$$
$$x8698 + x87102 + x8892 + x8893 + x8990 + x8994 + x9098 + x9193+$$
$$x8586 + x91103 + x9295 + x9297 + x9396 + x9499 + x9596 + x9597+$$
$$x97106 + x99101 + x100108 + x100110 + x101104 + x102103 + x102109+$$
$$x103106 + x104111 + x105106 + x105107 + x106118 + x107108+$$
$$x109113 + x109114 + x110112 + x111114 + x111130 + x112115+$$
$$x113114 + x113119 + x114119 + x114125 + x114126 + x115116+$$
$$x116117 + x117121 + x118122 + x118131 + x119122 + x119126+$$
$$x120121 + x120123 + x123124 + x124128 + x125126 + x125127+$$
$$x126132 + x126138 + x127130 + x127132 + x128133 + x129131+$$
$$x129133 + x129135 + x130132 + x130156 + x131136 + x132134+$$
$$x133135 + x134137 + x135136 + x135143 + x136143 + x136155+$$
$$x137140 + x138139 + x138142 + x139146 + x139154 + x140142+$$
$$x140145 + x141144 + x141147 + x141152 + x142146 + x143148+$$
$$x143160 + x144150 + x145149 + x145156 + x146149 + x147151+$$
$$x147152 + x148155 + x148160 + x149156 + x150153 + x150154+$$
$$x151155 + x152153 + x152159 + x153157 + x154157 + x155158+$$
$$x155162 + x158159 + x158162 + x159165 + x160166 + x162167+$$
$$x165168 + x166171 + x167168 + x167170 + x170171 \leq 166$$

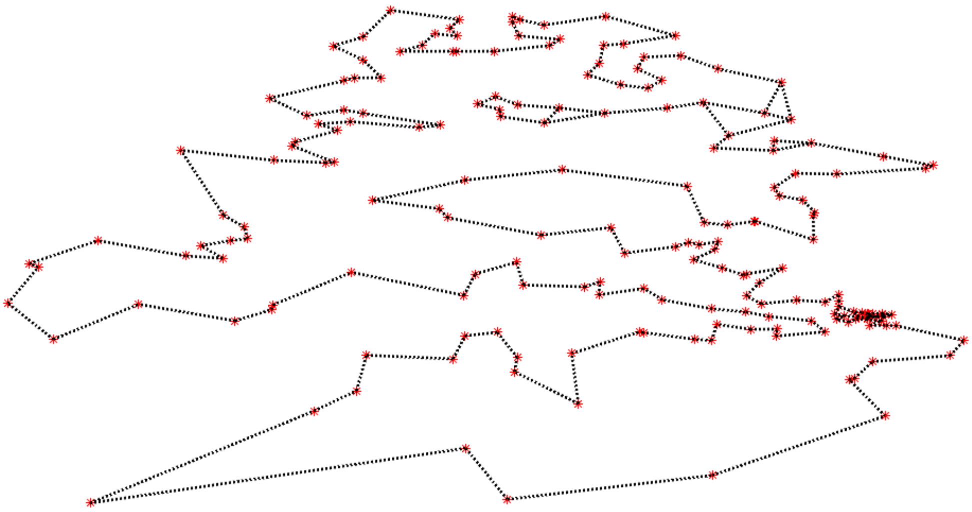


Figure 3.20: The LP solution with 22 SEC and 4 combs inequality

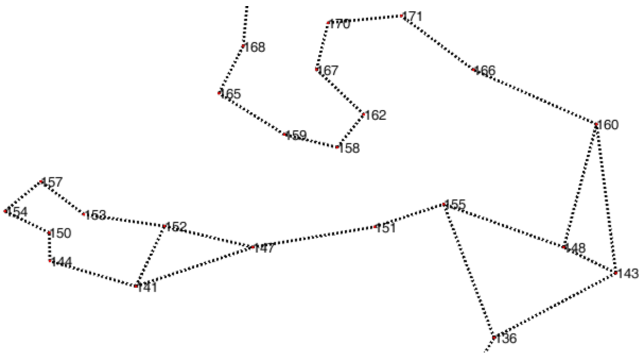


Figure 3.21: S_{23} before remove $x147151$

The result of adding the last two linear inequalities appears in the LP solution figure (3.20). The lower bound increased to (9,338.167) and the graph became connected but not 2-connected. As can be seen from Figure (3.21), the removal of city 147 (the edge $x147151$) split the graph into two connected components; one consisting of 8 cities $S_{27} = \{141\ 144\ 147\ 152\ 150\ 153\ 154\ 157\}$ and the other one containing 186 cities $S_{28} = \{V - S_{27}\}$. We add the following inequalities:

$$c_{27}: \; x141144 + x141147 + x141152 + x144150 + x147152 + x150153 + x150154 + x152153 + x153157 + x154157 \leq 7$$

$$
\begin{aligned}
c_{28}:\ & x0104 + x0106 + x0203 + x0204 + x0305 + x0509 + x0608 + x0711+\\
& x0717 + x0816 + x0910 + x1012 + x1114 + x1215 + x1316 + x1323+\\
& x1425 + x1519 + x1726 + x1821 + x1833 + x1930 + x2063 + x2065+\\
& x2124 + x2227 + x2229 + x2325 + x2426 + x2734 + x2737 + x2829+\\
& x2833 + x3032 + x3132 + x3135 + x3439 + x3440 + x3538 + x3542+\\
& x3544 + x3659 + x3663 + x3739 + x3745 + x3840 + x3841 + x3844+\\
& x3947 + x3951 + x4043 + x4143 + x4144 + x4146 + x4244 + x4249+\\
& x4250 + x4347 + x4356 + x4358 + x4446 + x4448 + x4449 + x4557+\\
& x4648 + x4654 + x4751 + x4852 + x4853 + x4854 + x4856 + x4950+\\
& x4955 + x5055 + x5161 + x5253 + x5254 + x5356 + x5455 + x5456+\\
& x5658 + x5760 + x5861 + x5962 + x6069 + x6167 + x6282 + x6468+\\
& x6470 + x6585 + x6668 + x6673 + x6773 + x6974 + x7077 + x7176+\\
& x7180 + x7182 + x7274 + x7275 + x7278 + x7576 + x7578 + x7680+\\
& x7687 + x7784 + x7891 + x7981 + x7983 + x8087 + x8184 + x8388+\\
& x8586 + x8698 + x87102 + x8892 + x8893 + x8990 + x8994 + x9098+\\
& x9193 + x91103 + x9295 + x9297 + x9396 + x9499 + x9596 + x9597+\\
& x97106 + x99101 + x100108 + x100110 + x101104 + x102103 + x102109+\\
& x103106 + x104111 + x105106 + x105107 + x106118 + x107108 + x109113+\\
& x109114 + x110112 + x111114 + x111130 + x112115 + x113114 + x113119+\\
& x114119 + x114125 + x114126 + x115116 + x116117 + x117121 + x118122+\\
& x118131 + x119122 + x119126 + x120121 + x120123 + x123124 + x124128+\\
& x125126 + x125127 + x126132 + x126138 + x127130 + x127132 + x128133+\\
& x129131 + x129133 + x129135 + x130132 + x130156 + x131136 + x132134+\\
& x133135 + x134137 + x135136 + x135143 + x136143 + x136155 + x137140+\\
& x138139 + x138142 + x139146 + x140142 + x140145 + x142146 + x143148+\\
& x143160 + x145149 + x145156 + x146149 + x148155 + x148160 + x149156+\\
& x151155 + x155158 + x155162 + x156161 + x158159 + x158162 + x159165+\\
& x160166 + x161163 + x161169 + x162167 + x163164 + x164169 + x164172+\\
& x165168 + x166171 + x167168 + x167170 + x168178 + x169176 + x170171+\\
& x170180 + x171185 + x172174 + x172179 + x173174 + x173175 + x174179+\\
& x175177 + x175184 + x176182 + x177181 + x177184 + x178180 + x178181+
\end{aligned}
$$

$x179186 + x180185 + x181184 + x182194 + x183186 + x183187 + x184189 + x185193 + x186194 + x187190 + x188189 + x188191 + x188193 + x189191 + x189192 + x190192 + x190194 + x191192 \leq 185$

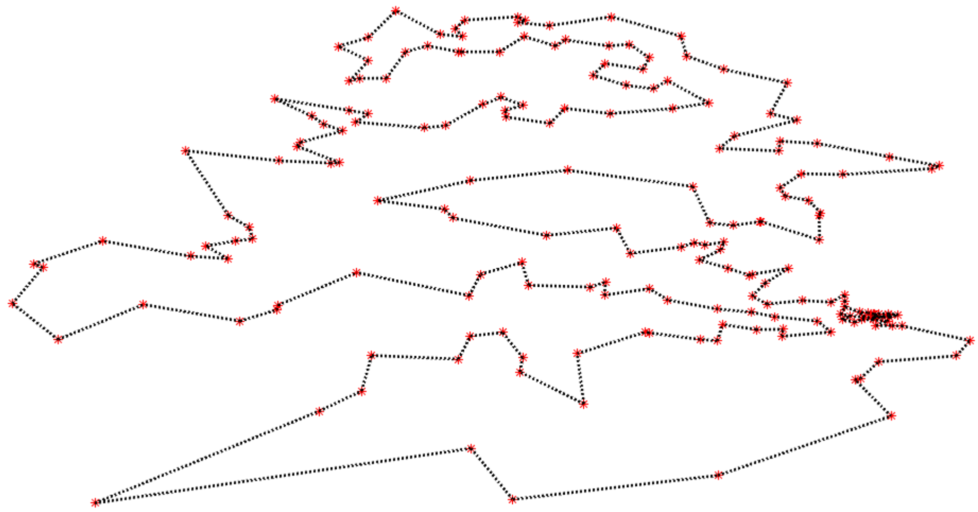


Figure 3.22: The LP solution with 24 SEC and 4 combs inequality

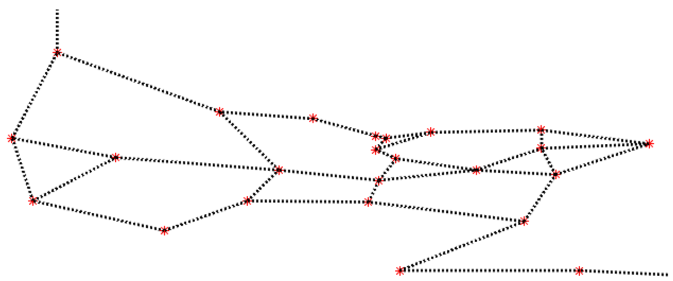


Figure 3.23: A part of Figure (3.22) where the extra uneliminated edges exist

The optimal solution x^* after we added the last two constraints is shown in figure (3.22). The lower bound arrived (9,350.66) and there are no more edges in the graph except for those that appear in figure (3.23). The edges in figure (3.23) either have the value 0.33 or 0.66; thus we need, instead of comb inequalities, a more advanced inequality type such as domino parity inequalities. Yet, after running Hougardy and Schroeder's algorithm, subtour elimination constraints and comb inequalities were only our tools, in this thesis, to eliminate further useless edges. With 24 SEC and four comb inequalities, the number of edges was reduced to 206, which means 58 useless edges were eliminated. One way to achieve the best outcome, or the shortest tour, is by applying this mixed-integer linear programming (ILP).

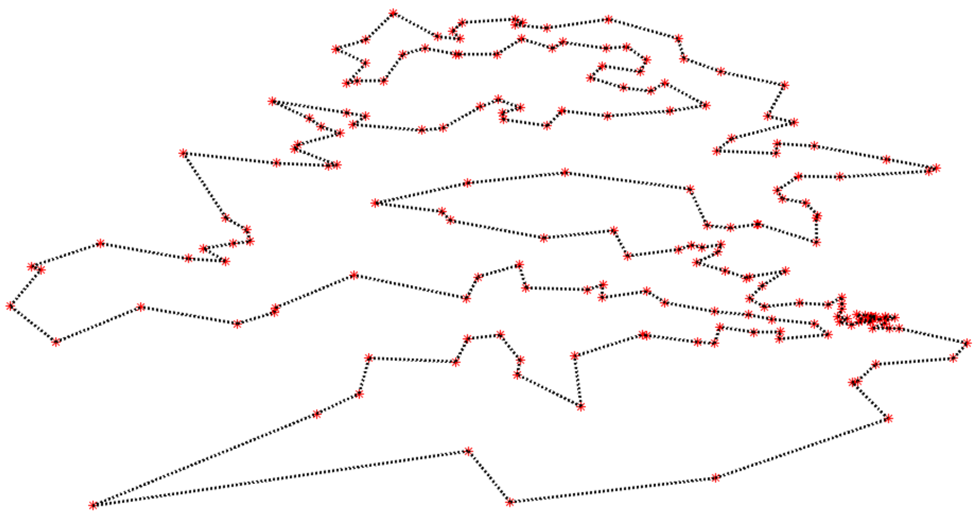


Figure 3.24: The optimal tour for qatar194

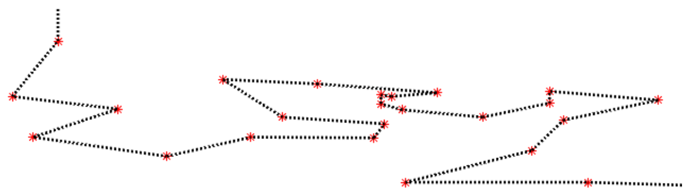


Figure 3.25: A part of Figure (3.24) where were the extra useless edges exist

The total length, the lower bound, of the optimal tour of qatar194 is (9352), which means the number of edges is equal to the number of cities. The way to hit that is by applying ILP representation using our list of constraints (24 SEC and 4 comb inequalities). Figure (3.24) and Figure (3.25) present the optimal tour.

3.3 The Algorithm

In this section we shall summarise our work and define some functions that are used in our Matlab code.

Table 3.1: Additional constraints summary

	Inequality Type	Number of Constraints	Total Constraints	Lower Bound
1	SEC	3	3	9,281
2	SEC	2	5	9,282
3	SEC	3	8	9,284
4	SEC	2	10	9,284.50
5	SEC	2	12	9,301
6	SEC	2	14	9,307.75
7	SEC	2	16	9,308
8	SEC	2	18	9,311.50
9	Comb	4	22	9,328.16
10	SEC	2	24	9,336.66
11	SEC	2	26	9,338.16
12	SEC	2	28	9,350.66

Solving the LP relaxation provided us a lower bound which was equal to (9,267). Then, while improving the LP relaxation section, we were improving the lower bound by adding more constraints to the list of constraints using two essential linear inequalities. The first linear inequality type is called subtour elimination constraints (SEC) and the other one is named comb inequalities. Table (3.1) shows that the more constraints we add, the closer the lower bound is to the optimal tour (9,352). The table illustrates how we improved the LP value by answering when, what and how many linear inequalities were added. After 28 constraints, we eliminated extra 58 edges. Then we achieved the optimal tour with length (9,352) using mixed-integer linear programming (ILP).

All calculations and visualisations to solve and analyse this TSP have been done using Matlab. Our primary function is known as *linprog*, which is a function defined by Matlab to solve linear programming problems. The job of this function is to solve the LP again each time we add more constraints. We decided to solve each LP problem by the default algorthim known as the *dual simplex* method in Matlab. In the end of the work, to solve the problem of mixed-integer linear programming, we applied *intlinprog*, which is also defined by Matlab. The author created a function called *plot-cities-with-numbers* which shows both the numbers of cities and their locations (see Appendix Listing 3.2). It was usually used to certify the numbers of cities when the graph was not 2-connected. For visualisation, we used a function called *updateSalesmanPlot-modified* which is an updated version of *updateSalesmanPlot*. The *updateSalesmanPlot* is a defined function by Matlab, however, our modified function was able to count drawn edges (see Appendix Listing 3.3). This function draws every edge that is not equal to zero. To define components' number, sizes and elements, we used a function called *networkComponents* created by Daniel Larremore in 2014 (Larremore, 2014), (see Appendix Listing 3.4). Daniel Larremore's function was an extremely useful function that allowed us to add our linear inequalities easily.

The Matlab code (see Appendix Listing 3.1) started by reading two Excel files; one for cities' locations and the second contained Hougardy and Schroeder' edges after applying their algorithm for eliminating useless edges. Then we went through a number of steps to be able to solve the TSP. We began with measuring distances between each pair of cities and writing the objective function with initial constraints (1.9). Then, we discovered the number of components and added subtour elimination constraints until the graph was connected. Note, the code could not figure out whether the graph was 2-connected or not, so we had to check each time. After the graph became 2-connected, the code was able to find combs under validation conditions and then add them to the list of constraints. The Matlab code was also asking Matlab to present all added constraints in a single file called "tsp.lp", so we could include them in the solution easily.

Conclusion

The traveling salesman problem (TSP for short) is one of the most widely studied NP-hard problems in combinatorial optimisation. In 1954, Dantzig, Fulkerson, and Johnson's method to solve the 49-cities problem was the first appearance of the most common way to solve the TSP. Their approach was to use a branch and bound algorithm in combination with the cutting planes method. One of the strongest implementations of this method is the fastest available algorithm to solve TSPs, Concorde. It was written by David Applegate, Robert E. Bixby, Vašek Chvátal, and William J. Cook in 2006. Although Concorde is the fastest TSP solver, the total run time for 85,900 vertices (the largest instance of TSP which has been solved to provable optimality to date), for instance, required more than 136 years of total CPU time (Applegate et al., 2011). In 2014, Stefan Hougardy and Rasmus T. Schroeder discovered a method to eliminate useless edges that do not belong to any two-dimensional euclidean TSPs. Their approach in combination with Concorde is able to solve large TSP instances more than 11 times faster than Concorde alone (Hougardy & Schroeder, 2014).

Based on Hougardy and Schroeder's results, this thesis aimed to extend the number of eliminated useless edges for a TSP instance called qatar194. The instance contains 194 cities and originally 18,721 edges. Yet, after applying Hougardy and Schroeder's algorithm steps, only 264 edges remained in the instance. Our approach was by engaging $k - opt$ move edge exchange transformations with $k > 3$. This engagement was from employing the subtour elimination constraints (SEC) and comb inequalities for the purpose of eliminating further edges. After adding 28 constraints (24 SEC and 4 combs), 58 further useless edges were eliminated using linear programming technique. Then, to eliminate all other useless edges and arrive at the optimal tour of qatar194, we applied the mixed-integer linear programming implementation. The main basis of the research's outcome was to write a Matlab code to solve two-dimensional euclidean TSPs (see appendix listing (3.1)), and to present our practical results (in chapter 3) to demonstrate how we can avoid some extra useless edges of qatar194.

Appendix

Listing 3.1: Matlab code to solve the TSP

```
close all; clear; clc;

% Cities location
cities = 194;
CitiesLocation = xlsread('qatar194.xlsx','Sheet1','
   A1:B194' );
x = CitiesLocation (:,1);
y = CitiesLocation (:,2);

% To plot these cities with numbers
plot_cities_with_numbers(x,y,cities);
fq = fopen('tsp.lp','w');
fprintf(fq,'Minimize\n');
fprintf(fq,'obj: ');

% Trips given by Stefan Hougardy
HougardyTrips = xlsread('HougardyTrips_qatar194.xlsx
   ','Sheet1','A1:B264' );
HT1 = HougardyTrips(:,1);
HT2 = HougardyTrips(:,2);

% Measure all the trip distances integerly
dist = round( hypot(y(HT1) - y(HT2), x(HT1) - x(HT2)
   ));
lendist = length(dist);

% To write the object function in the tsp.lp file
distance = 0;
cnt = 0;
```

```
for i = 1:length(HougardyTrips)
    cnt = cnt + 1;
    distance = dist(i);
    fprintf(fq,'%d x%02d%02d',distance,HT1(i),HT2(i)
       );
    if (mod(cnt,10) == 0)
        fprintf(fq,'\n');
    end
    if i < length(HougardyTrips)
        fprintf(fq,' + ');
    end
end

% Number of trips
AA = spones(1:length(HougardyTrips));
BB = cities;

% The salesman should enter and leave each city
   exactly once
G = spalloc(cities,length(HougardyTrips),cities*(
   cities-1));
Aeq = [AA;G];
for i = 1:cities
    whichHT = (HougardyTrips == i);
    whichHT = sparse(sum(whichHT,2));
    Aeq(i+1,:) = whichHT';
end
Beq = [BB; 2*ones(cities,1)];

% To write above constraints in the tsp.lp file
fprintf(fq,'\nSubject To\n');
for ii = 1 : cities
    cnt = 0;
    for jj = 1 : cities
        if(ii == jj)
            continue;
        end
        cnt = cnt + 1;
        if(ii < jj)
            fprintf(fq,'x%02d%02d',ii,jj);
        else
```

```
                fprintf(fq,'x%02d%02d',jj,ii);
            end
            if (mod(cnt,11) == 0)
                fprintf(fq,'\n');
            end
            if jj < cities
                if  ii == cities && jj == cities - 1
                    break;
                end
                fprintf(fq,' + ');
            end
        end
        fprintf(fq,' = 2 \n');
end

lb = zeros(lendist,1);
ub = ones(lendist,1);

% Solve the initial LP
A = []; B =[];
opts = optimoptions('linprog','Algorithm','dual-simplex');
[x_tsp]= linprog(dist,A,B,Aeq,Beq,lb,ub,[],opts);

lh = zeros(cities,1);
[lh1, l1] = updateSalesmanPlot_modified(lh,x_tsp,HougardyTrips,x,y);
fprintf('# Number of edges (initial LP relaxation): %d\n',l1);

% The total length of the tour is [dist'*x_tsp], Optimal tour = 9352
opt_sol = dist'*x_tsp;
counter = 1;

% While loop here will end up with the optimal tour
while opt_sol ~= 9352

    % How many subtours
    G_x1 = HT1;
    G_y1 = HT2;
```

```
    for i = 1:length(x_tsp)
        if x_tsp(i) == 0
            G_x1(i) = 0;
            G_y1(i) = 0;
        end
    end
    DG = sparse(nonzeros(G_x1)',nonzeros(G_y1)',true,cities,cities);
    [nComponents,sizes,member_of_subsets] = networkComponents(DG);

    if counter == 1;
        B = [[];zeros(nComponents ,1)];
        A = zeros([],length(HougardyTrips));
    end

    % ADD subtour elimination constraints
    while nComponents > 1
        for i = 1:nComponents
            for jj = 1:length(HougardyTrips)
                H = union (HT1(jj),HT2(jj));
                if ismember (H, member_of_subsets{i}) == [1 1];
                    A(counter,jj) = 1;
                    if HT1(jj) < HT2(jj)
                        fprintf(fq,'x%02d%02d',HT1(jj),HT2(jj));
                        fprintf(fq,' + ');
                    else
                        fprintf(fq,'x%02d%02d',HT2(jj),HT1(jj));
                        fprintf(fq,' + ');
                    end
                end
            end
            B(counter) =  sizes(i) - 1;
            fprintf(fq,' <= %d \n', B(counter));
            counter = counter + 1;
        end

        % Try to optimise again
```

```
        opts = optimoptions('linprog','Algorithm','dual-simplex');
        [x_tsp]= linprog(dist,A,B,Aeq,Beq,lb,ub,[],opts);

        % Measure the total length (lower bound)
        opt_sol = dist'*x_tsp;
        fprintf('# Total length: %d\n',opt_sol);

        % Visualise result
        [lh2, l2]= updateSalesmanPlot_modified(lh,x_tsp,HougardyTrips,x,y);



        % How many subtours this time
        G_x2 = HT1;
        G_y2 = HT2;
        for iv = 1:length(x_tsp)
            if x_tsp(iv) == 0
                G_x2(iv) = 0;
                G_y2(iv) = 0;
            end
        end
        DG2 = sparse(nonzeros(G_x2)',nonzeros(G_y2)',true,cities,cities);
        [nComponents,sizes,member_of_subsets] = networkComponents(DG2);
    end

    % Find handle H for the Comb inequalities
    handle_x = zeros(size(HT1));
    handle_y = zeros(size(HT2));
    handle_find = zeros(size(x_tsp));
    for i = 1:length(x_tsp)
        if x_tsp(i) == 0.5
            handle_x(i) = HT1(i);
            handle_y(i) = HT2(i);
            handle_find(i)= 1;
        end
    end
```

```
    H_edges = [handle_x , handle_y];
    H_edges_without_zeros = H_edges(any(H_edges,2),:);
    H_nodes = nonzeros(H_edges)';

    % Draw these handles
    updateSalesmanPlot_modified(lh,handle_find,H_edges,x,y);
    H_G_X = nonzeros(handle_x)';
    H_G_Y = nonzeros(handle_y)';
    CV = sparse(H_G_X,H_G_Y,true,cities,cities);
    [NComponent_handle,Sizes_handle,Members_handles]=networkComponents(CV);

    % Find real handles and their teeth only
    real_handle = {};
    teeth_find = zeros(size(x_tsp));
    for i = 1 : length(x_tsp)
        T = union (G_x2(i), G_y2(i));
        cnt = 1;
        for j = 1:NComponent_handle
            if Sizes_handle(j) > 2 && mod(Sizes_handle(j),2)==1;
                real_handle (cnt) = Members_handles(j);
                cnt=cnt+1;
            end
        end
        for ii = 1:length(real_handle)
            if  ismember (T , real_handle {ii}) == [1 0];
                teeth_find(i) = 1;
            elseif ismember (T , real_handle {ii}) == [0 1];
                teeth_find(i) = 1;
            end
        end
    end

    teeth_x = zeros(size(HT1));
    teeth_y = zeros(size(HT2));
```

```
    for i = 1:length(x_tsp)
        if teeth_find(i) == 1
            teeth_x(i) = HT1(i);
            teeth_y(i) = HT2(i);
        end
    end
    teeth_x_y = [teeth_x teeth_y];
    The_teeths = teeth_x_y(any(teeth_x_y,2),:);
    [lh4,l3] = updateSalesmanPlot_modified(lh,
       teeth_find,teeth_x_y,x,y);
    fprintf('# The Teeths: %d\n',l3);

    % Define real combs
    combs = {};
    for ii = 1:length(real_handle)
        combs{ii} = real_handle{ii};
        for i = 1: l3
            if ismember(The_teeths(i,:) ,
               real_handle{ii}) == [1 0];
                combs {ii} = unique([The_teeths(i,:)
                   , combs{ii}]);
            elseif ismember(The_teeths(i,:) ,
               real_handle{ii}) == [0 1];
                combs {ii} = unique([The_teeths(i,:)
                   , combs{ii}]);
            end
        end
    end

    % What the teeth of real combs
    comb_teeth = {};
    for i = 1: length (combs)
        cnt = 1;
        for ii = 1: l3
            if ismember(The_teeths(ii,:) , combs{i})
                == [1 1];
                if cnt == 1
                    comb_teeth{i} = The_teeths(ii,:)
                       ;
                    cnt = cnt+1;
                else
```

```
                    comb_teeth{i}=unique([The_teeths
                       (ii,:),comb_teeth{i}]);
                    cnt = cnt+1;
                end
            end
        end
    end

    % Only valid combs will be considered
    valid_comb = {};
    S = {};
    valid_handle = {};
    cnt_for_valid_comb = 1;
    for i = 1:length(comb_teeth)

        if mod(length(comb_teeth{i}),2)==0 && ...
                length(comb_teeth{i}) >= 6 && ...
                mod(length(real_handle{i}),2) == 1 ;

            valid_comb {cnt_for_valid_comb} = combs{
               i};
            S {cnt_for_valid_comb} = length(
               comb_teeth{i} - 1)/2;
            valid_handle {cnt_for_valid_comb} =
               real_handle{i};
            cnt_for_valid_comb = cnt_for_valid_comb
               + 1 ;
        end
    end

    % Print number of valid combs
    fprintf('# Number of valid Comb: %d\n',length(
       valid_comb));

    if cnt_for_valid_comb ~= 1;
        for i = 1: length(valid_handle)
            cnt = 0;
            for ii = 1: length (HougardyTrips)
                D = union (HT1(ii),HT2(ii));

```

```
        if ismember (D , valid_handle{i}) == [1 1];
            A(counter,ii) = 1;
            if HT1(ii) < HT2(ii)
                fprintf(fq,'x%02d%02d',HT1(ii),HT2(ii));
                fprintf(fq,' + ');
            else
                fprintf(fq,'x%02d%02d',HT2(ii),HT1(ii));
                fprintf(fq,' + ');
            end
        end

        for iii = 1: length(The_teeths)
            F = intersect (The_teeths(iii,:),valid_comb{i});
            if ismember (D,F) == [1 1];
                A(counter,ii) = 1; cnt = cnt + 1;

                if HT1(ii) < HT2(ii)
                    fprintf(fq,'x%02d%02d',HT1(ii),HT2(ii));
                    fprintf(fq,' + ');
                else
                    fprintf(fq,'x%02d%02d',HT2(ii),HT1(ii));
                    fprintf(fq,' + ');
                end
            end
        end
    end
    B(counter) = length(valid_handle{i}) + 2*cnt - (3*cnt + 1)/2;
    fprintf(fq,' <= %d \n', B(counter));
    counter = counter+1;

    % ReOptimise
    opts = optimoptions('linprog','Algorithm','dual-simplex');
```

```
            [x_tsp]= linprog(dist,A,B,Aeq,Beq,lb,ub,[],opts);

            % ReMeasure the total length (lower bound)
            opt_sol = dist'*x_tsp;
            fprintf('# Total length with LP: %d\n',opt_sol);

            % Visualise result
            [lh4, l4] = updateSalesmanPlot_modified(lh,x_tsp,HougardyTrips,x,y);
            fprintf('# Number of edges %d\n',l4);
        end
    end

    if cnt_for_valid_comb == 1 && nComponents == 1

        % ReOptimise with MILP
        intcon = 1:lendist;
        [x_tsp]=intlinprog(dist,intcon,A,B,Aeq,Beq,lb,ub);

        for i = 1:length(x_tsp)
            if x_tsp(i) < 0.1
                x_tsp(i) = 0;
            end
        end

        % ReMeasure the total length (lower bound)
        opt_sol = dist'*x_tsp;
        fprintf('# Total length with ILP: %d\n',opt_sol);

        % Visualise result
        [lh5, l5]= updateSalesmanPlot_modified(lh,x_tsp,HougardyTrips,x,y);
        fprintf('# Number of edges %d\n',l5);
    end
end
```

```
% Lower & upper bounds for tsp.lp file
all_possible_trips = nchoosek(1:cities,2);
fprintf(fq,'\nBounds\n') ;
for i = 1:length(all_possible_trips)
    fprintf(fq,'0 <= x%02d%02d <= 1\n',
       all_possible_trips(i,1), ...
        all_possible_trips(i,2));
end
fprintf(fq,'End\n') ;
```

Listing 3.2: To show cities location and number in the graph

```
function plot_cities_with_numbers(x,y,cities)

hold on
for ii=1:cities
    plot (x(ii,1),y(ii,1),'r*');
    ln = findobj('type','line');
    set(ln,'marker','.','markers',5,'markerfa','w')
    text(x(ii,1),y(ii,1),num2str((ii)));
end
hold off
drawnow;
```

Listing 3.3: An updated version of updateSalesmanPlot

```
function [lh, l] = updateSalesmanPlot_modified(lh, xopt,idxs,stopsLat,stopsLon)
% Plotting function for tsp_intlinprog example

%   Copyright 2014 The MathWorks, Inc.

 if ( lh ~= zeros(size(lh)) ) % First time through lh is all zeros
     set(lh,'Visible','off');  % Remove previous lines from plot
 end

segments = find(xopt); % Indices to trips in solution
l = length(segments);
% Loop through the trips then draw them
Lat = zeros(3*length(segments),1);
Lon = zeros(3*length(segments),1);
for ii = 1:length(segments)
    start = idxs(segments(ii),1);
    stop = idxs(segments(ii),2);

    % Separate data points with NaN's to plot separate line segments
    Lat(3*ii-2:3*ii) = [stopsLat(start); stopsLat(stop); NaN];
    Lon(3*ii-2:3*ii) = [stopsLon(start); stopsLon(stop); NaN];
end

lh = plot(Lat,Lon,'k:','LineWidth',2);
set(lh,'Visible','on');
drawnow; % Add new lines to plot
```

Listing 3.4: Find network components, sizes, and lists of member nodes

```
function [nComponents,sizes,members] = networkComponents(A)
N = size(A,1); % Number of nodes
A(1:N+1:end) = 0; % Remove diagonals
A=A+A'; % make symmetric, just in case it isn't

isDiscovered = zeros(N,1);
members = {};
% To check every node
for n=1:N
    if ~isDiscovered(n)
        members{end+1} = n; % started a new group so add it to members
        isDiscovered(n) = 1; % account for discovering n
        ptr = 1;
        while (ptr <= length(members{end})) % find neighbors
            nbrs = find(A(:,members{end}(ptr)));
            newNbrs = nbrs(isDiscovered(nbrs)==0);
            isDiscovered(newNbrs) = 1;
            members{end}(end+1:end+length(newNbrs)) = newNbrs;
            ptr = ptr+1;
        end
    end
end
nComponents = length(members); % number of components
for n=1:nComponents
    sizes(n) = length(members{n}); % compute sizes of components
end
[sizes,idx] = sort(sizes,'descend');
members = members(idx);
end
```

Bibliography

Agarwala, R., Applegate, D. L., Maglott, D., Schuler, G. D., & Schäffer, A. A. (2000). A fast and scalable radiation hybrid map construction and integration strategy. *Genome Research*, *10*(3), 350–364.

Alevras, D., & Padberg, M. W. (2001). *Linear optimization and extensions: problems and solutions*. Springer Science & Business Media.

Applegate, D. L., Bixby, R. E., Chvatal, V., & Cook, W. J. (2011). *The traveling salesman problem: a computational study*. Princeton university press.

Bock, F. (1963). Mathematical programming solution of traveling salesman examples. *Recent Advances in Mathematical Programming,(RL GRAVES, AND P. WOLFE, eds.)*, 339–341.

Chauhan, C., Gupta, R., & Pathak, K. (2012). Survey of methods of solving tsp along with its implementation using dynamic programming approach. *International Journal of Computer Applications*, *52*(4).

Cook, S. A. (1971). The complexity of theorem-proving procedures. In *Proceedings of the third annual acm symposium on theory of computing* (pp. 151–158).

Cook, W. (2012). *In pursuit of the traveling salesman: mathematics at the limits of computation*. Princeton University Press.

Dantzig, G., Fulkerson, R., & Johnson, S. (1954). Solution of a large-scale traveling-salesman problem. *Journal of the operations research society of America*, *2*(4), 393–410.

Dantzig, G. B. (1948). Programming in a linear structure. *Washington, DC*.

Fleischer, L., Letchford, A., & Lodi, A. (n.d.). Separating simple domino parity inequalities.

Flood, M. M. (1956). The traveling-salesman problem. *Operations Research*, *4*(1), 61–75.

Grötschel, M., & Padberg, M. W. (1979). On the symmetric travelling salesman problem i: inequalities. *Mathematical Programming*, *16*(1), 265–280.

Gutin, G., & Punnen, A. P. (2006). *The traveling salesman problem and its variations* (Vol. 12). Springer Science & Business Media.

Held, M., Hoffman, A. J., Johnson, E. L., & Wolfe, P. (1984). Aspects of the traveling salesman problem. *IBM journal of Research and Development*, *28*(4), 476–486.

Held, M., & Karp, R. M. (1962). A dynamic programming approach to sequencing problems. *Journal of the Society for Industrial and Applied Mathematics*, *10*(1), 196–210.

Hougardy, S., & Schroeder, R. T. (2014). Edge elimination in tsp instances. In *International workshop on graph-theoretic concepts in computer science* (pp. 275–286).

Jessen, R. J. (1942). Statistical investigation of a sample survey for obtaining farm facts. *Research Bulletin (Iowa Agriculture and Home Economics Experiment Station)*, *26*(304), 1.

Karp, R. M. (1972). Reducibility among combinatorial problems. In *Complexity of computer computations* (pp. 85–103). Springer.

Larremore, D. (2014). *Find Network Components.* `https://au.mathworks.com/matlabcentral/fileexchange/42040-find-network-components?focused=3818140&tab=function/`. ([Online; accessed 19-May-2017])

Letchford, A. N., & Lodi, A. (2002). Polynomial-time separation of simple comb inequalities. *Lecture notes in computer science*, 93–108.

Mahalanobis, P. C. (1940). A sample survey of the acreage under jute in bengal. *Sankhyā: The Indian Journal of Statistics*, 511–530.

Menger, K. (1932). Das botenproblem. *Ergebnisse eines mathematischen kolloquiums*, *2*, 11–12.

Müller-Merbach, H. (1983). Zweimal travelling salesman. *DGOR-Bulletin*, *25*, 12–13.

Reeb, J. E., Leavengood, S. A., et al. (1998). *Using the simplex method to solve linear programming maximization problems* (Tech. Rep.). Corvallis, Or.: Extension Service, Oregon State University.

Robinson, J. (1949). *On the hamiltonian game (a traveling salesman problem)* (Tech. Rep.). RAND PROJECT AIR FORCE ARLINGTON VA.

Vos, T. (2016). Basic principles of the traveling salesman problem and radiation hybrid mapping.