Formulating Quadratic Traveling Salesman Problems for Computation

by

Michelle Spencer

B.A. & Sc., Quest University Canada, 2016

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

in the
Department of Mathematics
Faculty of Science

SIMON FRASER UNIVERSITY
Summer 2019

Approval

Name:	**Michelle Spencer**
Degree:	**Master of Science (Operations Research)**
Title:	**Formulating Quadratic Traveling Salesman Problems for Computation**
Examining Committee:	**Chair:** Mary Catherine Kropinski Professor
	Tamon Stephen Senior Supervisor Associate Professor
	Tom Archibald Supervisor Professor
	Ladislav Stacho Internal Examiner Associate Professor
Date Defended:	**August 2, 2019**

Abstract

The Traveling Salesman Problem (TSP) is a fundamental combinatorial optimization problem. Adding costs associated with pairs of edges included in a tour gives the Quadratic Traveling Salesman Problem (QTSP). This increases modeling power by allowing, for example, the inclusion of transfer costs between edges. We consider a general version of this problem, where costs are attached to all pairs of edges.

We give a brief history of computational solvers, especially in relation to the TSP. For the QTSP, we consider modifying the structure of the quadratic cost input and linearizing the quadratic objective function, with detail on how to generate the modifications and linearizations. We study the impact of these structures on computational efficiency for randomly generated instances, using the Gurobi solver. We find that by making the quadratic cost matrix negative semidefinite, we improve solve times, and that solving the problem as a quadratic minimization problem outperforms linearization approaches.

Keywords: Quadratic Traveling Salesman Problem; integer programming solvers; optimization; linearization

Acknowledgements

I would like to thank my senior supervisor, Dr. Tamon Stephen, for his support and feedback in the later stages of this research. I appreciate the time he spent reviewing my work and how I never felt rushed in our meetings. I am also grateful to Dr. Tom Archibald for his encouragement and assistance, in particular for the first chapter of this work, and also for his feedback.

I would like to recognize Dr. Ladislav Stacho, my examiner, for his insightful questions and observations, which improved the thesis. I am also grateful to Dr. Mary Catherine Kropinski for chairing my defence.

I would like to acknowledge Dr. Abraham Punnen for the framework for the research in chapters 2, 3 and 4 of this thesis, and for the support in the initial phases of this research.

I would like to thank the other graduate students for their help and company; especially Aniket Mane, Michael Friesen, Jessica Guo, Shawn Yan, Pooja Pandey and Brad Woods. I am also grateful to the SFU Mathematics Department for their support.

I would like to thank the BC Ministry of Education Education Economics unit for the wonderful co-op term. I learned so much, and am grateful for their flexibility and support.

Finally, I would like to acknowledge my friends and family for their unwavering support. I'd like to thank my cousins for providing moral support as I progressed through the complex stages of this degree. I am particularly indebted to my cousin Mary and my sister Nicole for their words of support and advice through this long process.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Combinatorial Optimization

The Traveling Salesman Problem (TSP) is a central problem in the field of combinatorial optimization. In combinatorial optimization, the objective is to minimize (or maximize) an objective function over a structured discrete set of feasible solutions. Other combinatorial optimization problems include the assignment, knapsack, set covering, and minimum spanning tree problems. Optimization is a key tool in operations research. For more detail on combinatorial optimization, see [KV18], [Sch03], among others.

1.2 Traveling Salesman Problem

The Traveling Salesman Problem (TSP) is a combinatorial optimization problem where the aim is to find the lowest cost route though a given set of nodes that returns to the initial node, forming a *cycle* or *tour* (a path that visits all the points or vertices, without repeating any vertices, before returning to its origin). It is named after the canvassing salesperson, whose goal is to visit a certain set of locations efficiently, before returning to their own home base. The TSP is a fundamental theoretical problem, and as it is NP-hard, it is related to one of the Clay Institute's Millennium Prize Problems [Coo11], [Coo06]. While a central academic problem, the TSP has a variety of uses outside of academia. Companies such as FedEx and UPS face the TSP daily, as they plan effective routes to deliver packages [Coo11].

We begin by formally defining the TSP. Given a graph G $= (V, E)$, where for each edge $ij \in E$, let the cost c_{ij} be known. For any Hamiltonian cycle H of G, let $C(H) = \sum_{ij \in H} c_{ij}$ be the total cost. Then the Traveling Salesman Problem is to find the Hamiltonian cycle – a cycle passing through each vertex exactly once – such that $C(H)$ is minimized. We provide details on different formulations in a later section. For details on the TSP, we refer

to [Coo11], [ABCC07].

Unless otherwise stated, we assume that G is a complete graph, which is to say that all cities are connected to all other cities. Where G is an incomplete graph, we can transform the TSP instance into a problem on a complete graph by adding edges with very high costs in place of the missing edges. No optimal solution would include a high-cost edge unless it was forced to, as the cost would be far higher than any other edge in the graph. Therefore an optimal solution that requires the use of one of those edges would indicate that the original set of cities has no solution. To illustrate, if we had a set of cities in North America, and a set of cities in Europe, and no method to cross the ocean, the artificial edges connecting North America and Europe would have to be selected to complete the tour, but no bridge actually connects the two continents. The inclusion of the high-cost edge indicates that the original problem has no solution.

We formulate the asymmetric n-city TSP as:

$$\text{Minimize: } \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \tag{1.1}$$

$$\text{Subject to: } \sum_{i=1}^{n} x_{ij} = \sum_{j=1}^{n} x_{ij} = 1 \tag{1.2}$$

$$x_{ij} \in \{0,1\} \text{ and } x_{ij},\ \{(i,j) : x_{ij} = 1\} \text{ form a tour} \tag{1.3}$$

The objective function (1.1) is the expression we are seeking to minimize by selecting the least-cost tour through the cities. We guarantee that each city is visited exactly once with the first set of constraints (1.2). The final set of constraints (1.3) ensures that the solution is integral, and that the solution forms a tour, which is closed (returns to starting city). We provide a more detailed description in a later section.

Two versions of the TSP may be considered – symmetric and asymmetric. In the symmetric TSP, the underlying graph G is undirected, while for the asymmetric TSP, it is directed. Referring back to our salesperson example, a city with one-way streets may be considered a case with asymmetric edge costs. The symmetric TSP can be considered a special case of the asymmetric TSP, in the sense that any solution to an asymmetric version of a TSP will be a solution to the symmetric version of the same instance, but the other direction is not always the case [GP07].

There are different classes of the underlying graph, aside from directed and undirected. One common class is the metric TSP, where the distances satisfy the *triangle inequality*. The triangle inequality is the property where $c_{ij} + c_{jk} \geq c_{ik}$ for all $i, j, k \in V$. An

example of a symmetric graph satisfying this property would be a planar map of cities, where you cannot "shortcut" by going through two edges. There are a few ways to satisfy this requirement, such as letting the cost between two cities be the *Euclidean distance* ($c_{ij} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$), which measures the straight-line distance, or the *Manhattan distance* ($c_{ij} = |x_i - x_j| + |y_i - y_j|$), which measures the city-block distance, if laid out on a grid. This can extend to three or more dimensions as well. Problems of each type can be generated, such as by randomly generating points, and calculating the distances between them. We may also use random costs that are not generated by locations for the edges, and which are not guaranteed to possess any particular property.

Tour edges correspond to pairs (i, j) where $x_{ij} = 1$. In the symmetric TSP, $c_{ij} = c_{ji}$ for all (i, j), and we can restrict the objective function to the upper triangular values of C, the matrix of cost coefficients. If we remove the tour restriction, this problem becomes the relatively simple 2-matching problem, which is solvable in polynomial time. However, requiring that the solution forms a tour, or cycle, through all the cities, turns it into a notoriously difficult problem. We want a process, known formally as an *algorithm*, to find the minimum cost tour. Algorithms are common tools in mathematics and computer science. An algorithm is a set of rules or instructions for carrying out a procedure or set of operations. Algorithms are used extensively in mathematics for situations from long division to more complex problems, but they are also used in everyday life, such as cooking from a recipe or changing a tire.

The applications of the TSP model go beyond transportation problems, including in areas such a technology and neuroscience [ABCC07]. Transportation examples include school bus routes, postal deliveries and field inspections [ABCC07]. Transportation problems are perhaps the most obvious applications of the TSP, and for good reason – even small improvements to the large-scale transportation problems faced by the postal service can realize large gains in time and resources spent. The TSP is also used in genome sequencing, computer chip design, circuit board drilling and aiming telescopes [ABCC07]. Therefore, throughout this thesis, we may refer to nodes, vertices or cities, but the nodes may represent something else entirely.

The problem of finding a minimum cost tour or circuit is an old one, though the name "traveling salesman problem" became widespread in the 1950s, when research on this problem gained popularity and traveling salespeople were common, especially in the US [ABCC07]. The general formulation was proposed by George Dantzig, Ray Fulkerson and Selmer Johnson of the RAND Corporation, in a paper which put forward their minimum length tour through 49 cities (one in each 48 states in the continental US, plus Washington, DC) [Coo11][DFJ54]. For more detail on the history and context of the TSP, we refer to

[Coo11].

The challenge with finding the lowest-cost tour is that while it is conceptually possible to enumerate every possible tour through a set of n cities, it is computationally impractical, requiring $(n-1)!$ permutations of cities for the asymmetric TSP. One important goal for the TSP is finding algorithms or heuristics that solve or estimate solutions efficiently. We use the notion of an efficient algorithm from computational complexity, where an "efficient algorithm" is one where the time cost of running the algorithm increases polynomially as the size of the problem increases [Coo11], [Pap03].

We use "big O" notation when categorizing the efficiency of an algorithm, where we write $\mathcal{O}(n)$ and n is the size of the problem [Pap94]. When you compare one example of polynomial time (e.g. $\mathcal{O}(n^2)$) against exponential time (e.g. $\mathcal{O}(2^n)$) or factorial time (e.g. $\mathcal{O}(n!)$), we can easily see that even small increases in n results in very high computational time requirements for the latter two examples. The time cost is an asymptotic upper bound of the number of operations required for a problem based on the size of the input. This notation allows us to quantify the amount of work based on the dominant term of the total number of elementary operations [KV18]. An algorithm requiring n^2+7n operations and an algorithm requiring $3n^2$ operations are both in $\mathcal{O}(n^2)$, as when n increases, the first term will dominate. When we say that an algorithm is "in the order of n^2" or $\mathcal{O}(n^2)$, we mean that the number of steps increases as n^2 does, without worrying about the value of the coefficient on n^2 [Pap94].

This definition was introduced by Edmonds [Edm67] in the 1960s, where he described a "good algorithm" to be one where the "amount of work" required is bounded by a polynomial function of the size of the problem. Therefore it is not enough to have a finite algorithm, but rather one that is practically feasible [Edm65]. The requirement for practicality brings this issue from theoretical to concrete, and is a distinction that appears to be significant in practice. Edmonds notes: "finding a finite algorithm is trivial but finding an algorithm which meets this condition for practical feasibility is not trivial" [Edm65]. He was considering an algorithm for matroid partitions, however the definition has come to apply to other combinatorial problems as well, including the TSP. Enumerating all the possible tour options is a finite algorithm, though it is a great example of one that is not practically feasible, for even modest values of n.

We say that problems that can be solved in polynomial time are in class "P", while those, like the TSP, that can be checked in polynomial time, but perhaps not solved so easily, are in class "NP", which stands for "non-deterministic polynomial time" [Coo11], [Coo06]. When we "check" a solution, we are solving the related *decision problem* associ-

ated with the original optimization problem. A decision problem has a "yes or no" answer, such as "is a number a even?", rather than finding a specific solution, such as "what is a number a divided by two?". In the example of the TSP, if we are given a solution, it is possible to check whether or not it goes through every city and has cost less than some value, but finding that tour and verifying that it is optimal is extremely challenging. The TSP is known to be NP-hard [GP07].

If the two classes, P and NP, were the same, then every problem that we have determined is NP will also be in the class P, meaning that we could solve it in polynomial time provided the right algorithm [Coo06]. Deciding whether P = NP is an open problem in mathematics, and is one of the seven Millennium Prize Problems put forward by the Clay Mathematics Institute in 2000 [Coo06]. If is it proven that P = NP, then the impact will be felt throughout the mathematical world, with consequences not only for solving the TSP and other combinatorial optimization problems, but also for cryptography and bioinformatics [Coo11]. While there have been proposed proofs for the P = NP problem, they have not held up to scrutiny, and generally it is believed that P $\neq$ NP. For now, it seems unlikely that an efficient general algorithm exists for NP-complete problems, and therefore, we settle for finding better algorithms, which is to say finding those that run in less than the $\mathcal{O}(n!)$ it would take use brute force enumeration to find the solution. In fact, Held and Karp have developed an exact algorithm that takes $\mathcal{O}(n^2 2^n)$ operations for solving the TSP, which while better than $\mathcal{O}(n!)$ is still in exponential time, but is not in polynomial time [ABCC07].

There have been a variety of approaches used to tackle the TSP and other integer problems in practice over the years. In general we might divide the approaches into exact methods, which find a globally optimal solution, and heuristic methods, which do not guarantee global optimality. Exact methods for the TSP are very challenging due to the difficulty of the problem. Within inexact methods, we can consider both general heuristics and approximation algorithms.

One exact method is the "branch-and-bound" method, by which the TSP is split into smaller sub-problems [ABCC07]. We have to review every branch of the solution tree if we want to be sure to have found the optimal solution. In the case of the TSP, we may consider all the possible edges leaving a node i. Of $(x_{i1}, x_{i2}, \ldots, x_{in})$, only one edge is permitted to be selected. We can then split the problem based on which of those edge is selected, i.e. one branch is where $x_{i1} = 1$, another is where $x_{i2} = 1$, and so on [ABCC07]. We can then compute the tree of possible solutions, by examining different edges. If it is found that the best possible solution in one subset is worse than the solution from the other subset, or if the subset does not contain a feasible tour, then we can eliminate that branch, and thus

restrict the problem.

Other exact algorithmic approaches include using the minimum spanning tree, as in the Held-Karp method. We also use cutting planes, where constraints are added or lifted (modified to make it stronger), so that the linear programming solution is closer to the integer programming solution [ABCC07]. This method is reviewed in Section 1.2.1. Many of the early researchers in this field were significantly limited by the available computational power. These limitations influenced which techniques were used, and which approaches were attempted [ABCC07], [Bix12].

An independent avenue of research is in heuristic methods, which are not guaranteed to find optimal solutions, but can still provide "good" tours – tours that are not necessarily optimal, but have a reasonable cost. In fact, a heuristic solution may even be the optimal solution, but the methods used do not guarantee this result. This guarantee is what separates approximate algorithms from exact ones, however it is possible to estimate the error on heuristic solutions.

We can categorize heuristics as constructive and non-constructive. In a *constructive* heuristic, we begin with an empty solution – no tour – and create a solution based on some rules. One example of a constructive heuristic is a nearest neighbour algorithm, where, starting at some initial vertex, you add the nearest city not already visited to the tour. For symmetric graphs satisfying the triangle inequality, the nearest neighbour algorithm guarantees a solution that is within $1 + \log_2(n)/2$ of the optimal tour length, however it is possible to generate asymmetric instances where the nearest neighbour algorithm returns tours with very high cost [Coo11]. Consider a TSP that visits a city in every state in the continental US. We may end up crossing the continent more than once to pick up any remaining cities, when a more circular tour would be more efficient (see 1.1). This algorithm is very simple to program, as it is a greedy algorithm, and for the 49-city example, it would be no worse than four times the optimal solution, [Coo11]. Another approximation algorithm was developed by Nicos Christofides in 1976. Christofides' algorithm combines a minimum spanning tree with a minimum matching of odd vertices, both obtainable in polynomial time [Coo11]. The result is an approximation that is no worse than $\frac{3}{2}$ times the optimal tour, the best approximation algorithm to date [Coo11].

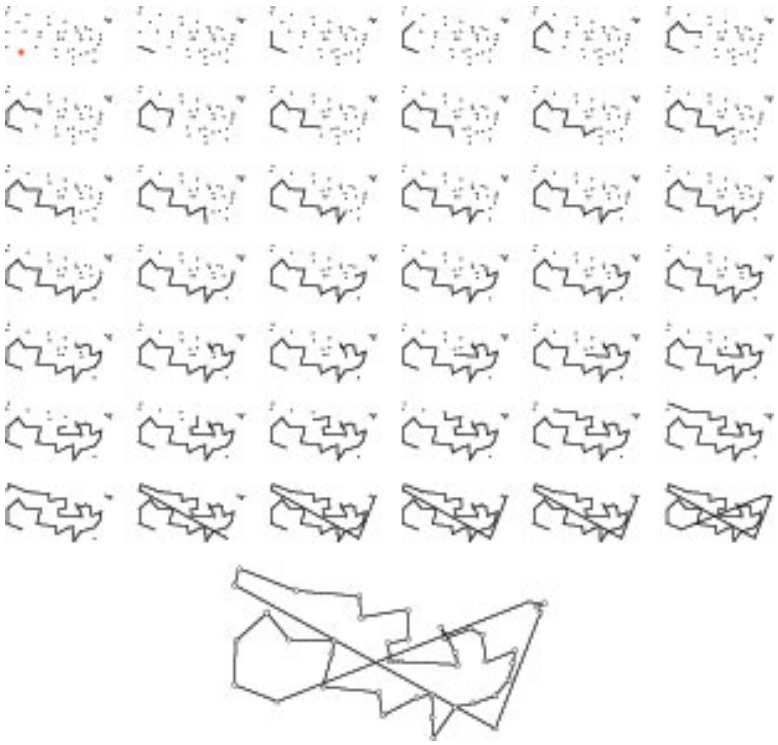

Figure 1.1: Nearest neighbour TSP (from [Coo11])

Non-constructive heuristic methods include local-search algorithms, simulated annealing and tabu search, and genetic algorithms [ABCC07]. In these methods, there is some initial solution presented, and the algorithm finds methods to improve the solution.

One example of a local search algorithm is to exchange pairs of edges for a pair of less costly edges, such as where two paths cross over each other [Coo11]. See Figure 1.2 for an example.

Simulated annealing is a probability-based algorithm, inspired by heating and cooling metals. In simulated annealing, we start with a solution, and move to neighbouring solutions, however we accept some moves that worsen the solution with the hope that they will eventually move us into a better neighbourhood [KGV83]. Initially, we will accept more of these risky moves, but as the temperature cools, we will accept fewer. In local-search and simulated annealing, we may find local minima, which are not guaranteed to be globally optimal. However, as the computational requirements for these types of algorithms are generally lower, we can run the same problem through multiple times with slight variations, such as different starting conditions or temperature functions, and select the best outcome.

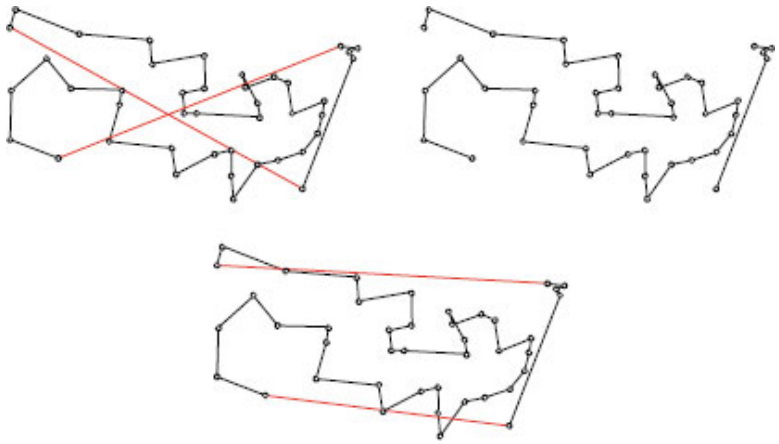

Figure 1.2: Two edge swap TSP (from [Coo11])

One can combine constructive and non-constructive algorithms, such as using the output of a constructive heuristic as the input for a local search algorithm, and they can be used very effectively on larger problems that exact solvers are unable to solve, but ultimately they do not provide the same assurances as exact solvers. One might opt to use a heuristic solver when finding the best possible solution is not required, and it is more important that the solution is found quickly. Finding good heuristics is its own research challenge, where we look at the quality of the solution and the efficiency of the heuristic, and compare using benchmark problems, in a method that parallels the research using exact solvers.

1.2.1 Traveling Salesman as an Integer Programming Problem

A *linear programming* (LP) problem is one where we are seeking to optimize (often minimize) a linear objective function subject to linear constraints.The *feasible set* is the set of values of $\mathbf{x}$ that satisfy the constraints. The feasible set for LP problems forms a convex polyhedron bounded by the constraints, which turns out to be very helpful in finding the solution, as any local minimizer will also be a global minimizer [BT97]. A minimizer is a a solution x that minimizes the objective function. A local minimizer is a solution that minimizes within a neighbourhood, while a global minimizer extends to the entire problem.

We can represent the TSP as an integer programming (IP) problem or a mixed-integer linear programming (MILP) problem, where we represent constraints as linear expressions using variables that are either integers only (IP) or integers and continuous variables (MILP). Linear programming, integer programming and mixed-integer programming have a long history, and there are a variety of techniques that are used to solve these problems. MILP problems are common in industrial optimization, where they are used in scheduling, manufacturing and cellular networks. We generally use the sum form to describe the TSP

rather than the vector form throughout this thesis.

Linear programming problems are often stated in the standard vector form [BT97]:

$$\text{Minimize: } \mathbf{c}^\intercal \mathbf{x} \tag{1.4}$$

$$\text{subject to: } A\mathbf{x} = \mathbf{b} \tag{1.5}$$

$$\mathbf{x} \geq 0 \tag{1.6}$$

where $\mathbf{c} \in \mathbb{R}^n, b \in \mathbb{R}^m, A \in \mathbb{R}^{m \times n}$ and $x \in \mathbb{R}^n$. $\mathbf{x}$ is a vector with components $(x_1, ..., x_n)$, with all elements being nonnegative. The equivalent sum form would be:

$$\text{Minimize: } \sum_{i=1}^{n} c_i x_i \tag{1.7}$$

$$\text{subject to: } \sum_{j=1}^{m} a_{ij} x_j = b_i \; \forall \; i = 1, \ldots, n \tag{1.8}$$

$$x_i \geq 0 \; \forall \; i = 1, \ldots, n \tag{1.9}$$

General LPs can include additional inequality constraints and free variables, however we generally standarize the form to make solving easier. We move to the standard form by introducing slack variables, and making modifications to the problem so that $\mathbf{x}$ is nonnegative. Slack variables transform an inequality constraint into an equality constraint. For example, let our constraint be $3x_1 + 4x_2 \leq 18$. A slack variable will absorb the difference between the left side of the constraint and the right side, and we can replace the inequality with an equal sign. The constraint becomes $2x_1 + 4x_2 + s_1 = 18$ and $s_1 \geq 0$. In practice, this does not change the problem, as any values of x_1, x_2 that satisfy the original constraint also satisfy the modified constraint. We transform free variables (those that can take on any value) into nonnegative variables by introducing additional variables. For example, if x_j is free, then we replace it with $x_j^+ - x_j^-$, where both $x_j^+, x_j^- \geq 0$ [BT97]. By the end of this process, all variables will be nonnegative, and constraints will be of equality type.

For a mixed-integer linear programming problem, some x_i are restricted to integer values, while for integer programming problems, we say that $x \in \mathbb{Z}^n$. When solving LPs, we can stop as soon as we find a local minimizer, however this does not always extend to IPs. Consider a two-dimensional LP. The feasible solutions associated with it will be a convex polygon, e.g. a triangle, which we could represent on a plane. The feasible set for the IP associated with it will be all the integer points located inside the polygon. Once one finds the minimizer of the associated LP, one cannot simply find the nearest integer point and assume that it will be the optimal solution for the IP, making it much more challenging to

solve.

Linear programs are used for minimizing (or maximizing) a continuous expression, subject to constraints,. Industrial applications frequently have integer restrictions on the variables, meaning that they must solve the much more difficult MILP. There are many applications of MILPs beyond the TSP. One early LP is known as the Diet Problem, described by George Stigler in 1945, as a way to determine the best diet for minimal cost for an "active man" (such as one serving in the military) [Sti45]. In this problem, the constraints represent the minimum or maximum values of protein, fat, calories, vitamins and minerals. The variables will be the different available foods, with their nutritional information included in the constraints. The objective function minimizes the total cost of the food selected. In this case we can use a linear programming model, as it is possible to purchase non-integer units of food (e.g. 1.3kg of flour). The Diet Problem predates many of the LP methods discussed in this paper, and Stigler himself solved it by eliminating as many foods as possible and restricting the problem to only the most nutritious, and least-cost items, most of which is done by preprocessing today.

A common example of an IP is called the Nurse Rostering (or Scheduling) Problem, where nurses are scheduled to work at a hospital. The constraints will be minimum staffing requirements, labour laws and union agreements, while the objective function may seek to minimize overtime, or maximize nurses' satisfaction. In this case, we must use an IP or MILP formulation, as a nurse is either assigned to a shift or they are not. These are just two examples of uses of LP/MILP/IP in the field – there are many more including flight scheduling, production management and network flow problems.

Linear programming problems are often solved using the simplex method - an algorithm developed by Dantzig in the 1940s. The simplex algorithm exploits the structure of LP problems, as, if there exists an optimal solution to an LP, it will be found at the intersection of n or more constraints, where n is the number of variables. In effect, the simplex method checks the vertices of polyhedron created by the constraints in a systematic manner, never worsening the solution. While it is possible to generate instances with very high (exponential) iteration counts, in practice it is a very efficient algorithm (linear in the number of constraints) [BT97]. Even now, more than 70 years after it was initially developed, Dantzig's simplex method is critical to solving LP and MILP problems [Bix12].

One way to represent the TSP as an IP problem was described by Dantzig, Fulkerson and Johnson in 1954 [DFJ54] :

$$\text{Minimize: } \sum_{i=1}^{n} \sum_{j=1,\, j\neq i}^{n} c_{ij} x_{ij} \tag{2.25}$$

$$\text{subject to: } \sum_{j=1,\, j\neq i}^{n} x_{ij} = 1 \qquad \forall\, i = 1, \ldots, n \tag{1.10}$$

$$\sum_{i=1\, i\neq j}^{n} x_{ij} = 1 \qquad \forall\, j = 1, \ldots, n \tag{1.11}$$

$$\sum_{i\in S} \sum_{j\in S,\, j\neq i} x_{ij} \leq |S| - 1 \quad \forall\, S \subseteq \{2, \ldots, n\},\, 2 \leq |S| \leq n-1 \tag{1.12}$$

$$x_{ij} \in \{0, 1\} \qquad \forall\, i, j = 1, \ldots, n \tag{1.13}$$

Constraints (1.10) and (1.11) ensure that the tour passes through each city exactly once, while constraints (1.12) eliminates subtours, and are hence called "subtour elimination constraints". There are an exponential number of subtour elimination constraints, making this form difficult to work with explicitly. The above formulation works for the asymmetric TSP. To modify it for the symmetric TSP, we can replace constraints (1.10) and (1.11) with

$$\sum_{j=1,\, j\neq i}^{n} x_{ij} = 2 \qquad \forall\, i = 1, \ldots, n \tag{1.14}$$

and replace the subtour elimination constraints (1.12) with

$$\sum_{i\in S} \sum_{j\in S,\, j>i} x_{ij} \leq |S| - 1 \quad \forall\, S \subseteq \{2, \ldots, n\},\, 2 \leq |S| \leq n-1 \tag{1.15}$$

For both asymmetric and symmetric versions of the TSP, we can relax the integer constraint (1.13) to

$$0 \leq x_{ij} \leq 1 \qquad \forall\, i, j = 1, \ldots, n \tag{1.16}$$

The final constraint, (1.16), relaxes this IP problem into a LP problem (called the LP relaxation), but this can result in tours with fractional edges. With a fractional result, it is not clear how to obtain an integral tour, as we can't half cross an edge – we either traverse an edge ($x_{ij} = 1$) or we do not ($x_{ij} = 0$). When the results are fractional (e.g. $x_{ij} = \frac{1}{2}$), we cannot round to an integer value, as the result may not be a tour. Therefore, while the LP formulation of the TSP does provide some interesting information, it cannot be relied upon to solve the TSP as we intend to, with an integral tour.

There are a variety of other IP formulations for the TSP, including those that use additional variables to reduce the number of subtour elimination constraints, such as the MTZ formulation in 1960 [MTZ60], and the DL and SD formulations [OFF+17]. Other formulations include time-dependent models and commodity flow models. We will detail these formulations, and others, in Section 2.1.

By relaxing the TSP to an LP, we can use the simplex algorithm to solve it, though it may not give us a tour that can be carried out, but rather fractional edges. The simplex method will find solutions at the vertices of the feasible polytope, but many LP formulations have non-integer vertices. In fact, only special cases of IPs will generate integral solutions at vertices. For example, if the constraint matrix is integral and the right-hand side is also integral, (A and b from (1.5)), and A has some specific properties (namely, A is a *totally unimodular matrix*), then the linear system will have an integral minimizer [HK56], [JLN+10]. For more details see [BW05], among others. These special occurrences are interesting, but we are not so lucky with the TSP.

In the TSP formulation described described by Dantzig, Fulkerson and Johnson, they use an exponential number of constraints to eliminate subtours, making the constraint polytope difficult to work with, even for small values of n. Further, the resulting formulation is non-integral, so the simplex algorithm does not find an integer solution. The authors instead left the subtour elimination constraints out initially, solving the much easier underlying LP, and adding the subtour elimination constraints on an as-needed basis [ABCC07], [JLN+10].

A similar method was developed by Gomory in the later 1950s, as he developed the cutting plane method for integer programming using the simplex method [JLN+10], [Gom58]. In Gomory's method, new constraints are generated from the simplex tableau, which cut off non-integer solutions, and shave down the polytope until an optimal integer solution is found [JLN+10]. Since the simplex algorithm finds solutions on boundaries of the solution polytope, the cuts ensure that the solution corners are integral. These additional constraints will be additional hyperplanes that define the feasible polytope. As a simple example, if after solving the simplex tableau, one of the rows leaves the constraint as $3x_1 + x_2 \leq \frac{17}{4}$, then as x_1, x_2 are integer, we can tighten this constraint to $3x_1 + x_2 \leq 4$. Gomory presented a systematic method to use the results of the simplex tableau (the format used to apply the simplex algorithm) to find and apply these cuts as new constraints [Gom58].

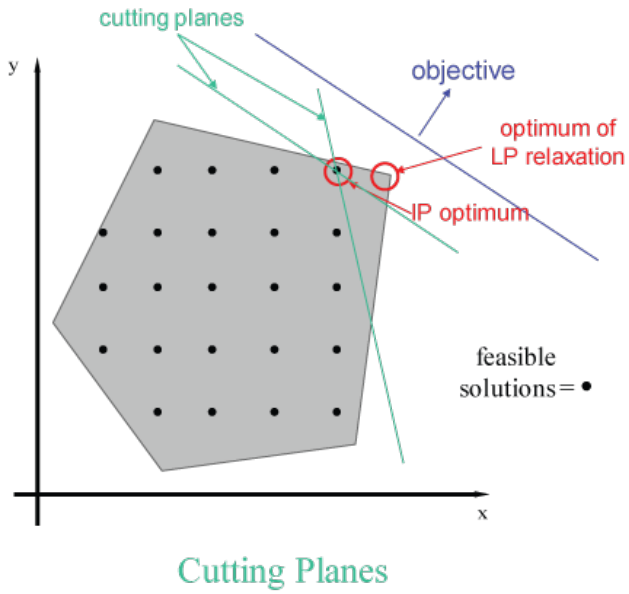

Figure 1.3: Graphic representation of an LP/IP problem with cutting planes (from [Opt])

In Figure 1.3, the simplex algorithm checks the vertices of the polyhedron to find the LP relaxation optimal solution. The points within the polyhedron represent the feasible solutions to the IP problem. In this case, we should select the nearest point to the LP solution as the IP solution, but in higher dimensions it is not so obvious. While it is something we can visualize, in multi-dimensional problems, such a visualization is not possible, and further, we need a way to have a computer "see" the solution. The cutting planes shave off parts of the LP polyhedron so that the IP solution and the LP solution of the new polyhedron are the same.

1.3 Computer Solvers for LP/IP Problems

Regardless of the method used, the TSP and other MILP problems present a computation problem. It is simply too onerous to calculate all the different possible tours, and even relatively efficient methods require significant storage of information. The same is true of other IP problems, though thanks to the simplex and related methods, LP problems are considerably easier to solve. Today it is possible to use out-of-the-box programs such as Gurobi and CPLEX to solve even large instances of LP and IP problems on home computers, but in the early days of TSP history, this was not the case. We will review how the computation methods influenced and aided solving the TSP.

1.3.1 1950s & 1960s

Dantzig and his colleagues at RAND initially used a Card Programmable Calculator to run the simplex algorithm on LP problems, which took eight hours to run through one instance of the common Stigler Diet Problem of 71 variables and 26 constraints [Bix12]. Much of the eight hours was spent feeding the cards into the system. However long that may appear, it was still a huge improvement over the "by-hand" method used previously, which took 120 man-days on a similar instance [Bix12]. The simplex method was then programmed on an IBM, allowing it to tackle larger problems, and was released for non-academic use, specifically in the oil and coal industries [Bix12].

For solving MILP problems, it used an updated algorithm that used a branch-and-bound method, which was used by Land and Doig in their study of discrete optimization, published in 1960 [Bix12], [JLN$^+$10], [LD60]. Land and Doig pioneered the branch-and-bound method, which they used to demonstrate the entire solution tree, with the different feasible solutions in [LD60]. Their method used 37 steps after the initial solve, on a constraint matrix of size 5 by 21 [LD60].

Often these methods were influenced by the type of computation power available, or the data storage methods. In some iterations of the MILP solver implementations, the IBM used only tape storage, which encouraged depth-first searching, rather than breadth-first, meaning that the algorithm searched down the entire branch of the tree to find a solution, then back up to find the best previous bound [Bix12]. In the TSP example, we would examine all tours in which the first branching variable $x_{ij} = 1$, before considering any in which that edge is not selected. The memory required for breadth-first searching is higher, as you may need to keep track of several variables at each level, but may have some advantages to finding solutions that require fewer operations to reach. Land and Doig's team did not have access to the same electronic computation, and so used desk calculators, and a combination of manual programming and printed storage to achieve their result [JLN$^+$10]. This meant that they could put branches of the tree on pause, and investigate other branches, before returning to the first, as the data were stored in printed files, meaning that when different computational resources became available, they had to reconsider how to implement their algorithm in a way a computer could navigate the data [JLN$^+$10].

During this time, Held and Karp developed a dynamic programming method for the TSP that brought the theoretical computation time down to $n^2 2^n$, but did not solve a larger instance of the TSP than had been previously completed [JLN$^+$10]. Dynamic programming uses a recursive method that breaks down a larger problem into multiple smaller problems. Held and Karp's algorithm found optimal paths in subsets of cities, and recur-

sively increased the number of cities in the subgroups [JLN+10]. By the later 1960s, they had also developed a branch-and-bound method that used minimum-cost spanning trees and Lagrangian relaxations to solve instances up to 64 cities [JLN+10]. While this method did still use the branch-and-bound methods, it did not use the cutting plane technique directly from the LP solvers [Coo11].

1.3.2 1970s

By the 1970s technology had advanced enough to permit different methods of tree search, and various new methods were introduced, including heuristics in determining branching variables [Bix12]. Additional computing availability and power also allowed for new methods, such as using the dual simplex method for LPs [Bix12]. Each LP has an associated dual LP, where the problem is transformed such that the variables become associated with constraints, and vice versa, and the objective switches from a minimization problem to a maximization problem. This creates a mirror problem that we can now put through the simplex algorithm. This can be very helpful in some circumstances, since if the dual has an optimal solution, so does the original (or primal problem), and furthermore, they have the same cost [BT97]. These results made IP and LP solvers powerful enough to start to solve major problems, and modifications to coding practices permitted stand-alone LP solvers to be inserted into other procedures, integrating LP subproblems into larger operations [Bix12]. IPs were also approached using polyhedral methods, with a great deal of research advancing the field of polyhedral combinatorics at the time [JLN+10]. In some specific cases IPs, produce integral polyhedra, which means that all of their extreme points are integral, and we can use LP methods to solve them, which is one of the links between these two fields.

During this decade, larger and larger instances of the TSP were also being solved, and in less and less time, including an 80-city instance in less than one minute [Coo11]. This came after a call for a more systematic and organized approach by Gomory in the 1960s [JLN+10]. Methods including using comb-inequalities, which were used in a by-hand and computer combination to solve a 120-city instance, a combination of cutting plane and branching-and-bound known as branch-and-cut, used on a small instance, and cutting planes on a large-scale problem of 318 cities [JLN+10].

1.3.3 1980s & 1990s

In the late 1970s, Khachiyan showed that LPs could be solved in polynomial time using an *ellipsoid* method [Bix12], [Kha80]. Methods to that date relied on the simplex method, which is not guaranteed to have polynomial efficiency, though in practice is typically efficient. Unfortunately, Khachiyan's algorithm was not practical, and as a result was

not pursued for use in LP solvers [Bix12]. This result still served an important purpose in furthering research in polyhedral methods in the 1980s, where it was shown that the LP problem on a polyhedron P is polynomial-time equivalent to the separation problem for P, where for some vector x we can show that either x is in the polyhedron P or we can generate a c such that $c^\intercal y \leq c^\intercal x$ for all $y \in P$ [JLN$^+$10]. This insight has implications beyond LP solvers. Our motivating problem is the TSP, which is solvable in exponential time. If the TSP can be formulated in a way that a polynomial-time separation algorithm can be used to solve it, then it's possible that we will be able to improve on the current solving time bounds [JLN$^+$10].

As desktop computers became available, researchers began developing solving code that would run on PCs, first LP solvers as early as 1983, then MILP solvers later on [Bix12]. Computer speeds continued to improve over this time, and there was a sense that the ongoing development of LP solvers had really slowed, and that improvements would come from the advances in technology rather than from modifying the solving algorithms [Bix12]. In fact, new methods and algorithms would continue to develop, most notably the dual simplex method as a general purpose solver, rather than as a function within other solving methods [Bix12].

Karmarkar built from the ellipsoid method in solving LPs by developing an efficient interior-point method, which in turn created its own branch of study within LP solvers [ABCC07], [Kar84]. In interior-point methods, rather than staying on the boundary of the feasible set, as done in the simplex method, we move through the centre of the polyhedron and work our way to the edge. As with Khachiyan, Karmarkar's algorithm was not immediately practical in computer LP solvers, but led to further developments in the field, including the emergence of log-barrier methods [Bix12]. In barrier methods, we remove constraints from the body of the problem and replace them with a penalty in the objective function, commonly a logarithmic or inverse function. For a simple example, let's say we wish to minimize an objective function ($c^\intercal x$), subject to some constraints, and we require x to be nonnegative, that is $x \geq 0$. We will now associate a penalty with the nonnegativity constraints and move them to the objective function. Now we minimize $c^\intercal x - \mu \sum_{i=1}^{n} \ln x_i$, subject to the original constraints, except for the nonnegativity requirements. It is easy to see that as x approaches zero, the penalty term, even for small values of μ, will become very large. We can modify the value of μ to approach the boundary associated with the constraint. This idea forms the basis for log-barrier algorithms.

By the late 1980s, AT&T developed a commercial LP solver, called the KORBX system, which used Karmarkar's interior-point method, but was ultimately not commercially successful [Bix12]. AT&T was very excited by the product, and tried to protect their investment

by trying to patent Karmarkar's algorithm [Sha12]. While AT&T published accounts that the software was able to solve instances of LPs that were previously unsolvable [CHL+89], other results were not so supportive [Sha12]. Other solvers developed during this time were the OB1, which used a log-barrier method, IBM's OSL code and Bixby's CPLEX LP solver [Bix12]. These solvers took advantage of the dual-simplex algorithm and improvements in managing the linear algebra aspects of solving, especially with regard to large, sparse models [Bix12]. In the simplex method, in each pivot, or move from one corner to another, many systems of equations must be solved, and so by preprocessing in advance of solving the system of equations, the solving step can be completed more efficiently [Bix02]. The result of these advances in solving ability was that by the early 2000s, even large LP problems were considered solvable by some method, whether primal, dual or barrier, and these methods were also used in solving MILP and IP problems [Bix12]. MILP solvers continued to develop over this time frame as well. Initially, improvements in solving time were due to computers becoming more powerful, and the underlying LP solvers improving, rather than any significant change in solving algorithms [Bix12]. In the late 1980s and early 1990s, several MILP solvers were developed for commercial use, including XpressMP and CPLEX, as well as solving codes developed for research purposes, such as Georgia Tech's MINTO code released in 1991 [Bix12].

1.3.4 2000s

In the early 2000s, LPs were again considered a "solved problem" [Bix12]. However, many of the improvements in LP solvers did not reach the MILP solvers until the late 1990s, and so in 1998, CPLEX saw it's biggest version-to-version improvement, when it incorporated the research from the previous few decades [Bix12]. Around this time, more advanced presolving techniques were included in computer solvers [Bix02]. Presolving algorithms modify the original problem to put it into a better form for solving or identify infeasibility. Presolve algorithms remove redundant constraints, improve bounds on variables, and generally try to tighten the model to reduce the amount of work the branch-and-cut algorithms later on [Mah10]. One specific example of a presolve method for MILP is to probe on binary variables [BFG+00]. For example, if when we set a binary variable to 1, the resulting model is infeasible, we can set that variable to 0 and substitute throughout the problem. In this example, we have replaced one variable with a constant, shrinking the size of the problem. Presolving algorithms make the solving algorithms more effective, but the onus is still on the researcher to provide the solver with the best possible input, as the presolvers are limited.

Today, a variety of commercial solvers are available for use. These include an LP solver built into Microsoft Excel, and solvers such as CPLEX and Gurobi that are used through

common programming languages like C and Python, or through languages developed for mathematical modeling, such as AMPL. Commercial math software such as Maple and Matlab also include LP solvers, and there are some open source solvers as well. CPLEX was founded in 1988 by Robert Bixby, a prominent scholar in operations research, who also documented the history of the methods outlined in this section. CPLEX was acquired by ILOG in 1997, which was later acquired by IBM [Rot]. In 2008, Bixby and two of the other members, Zonghao Gu and Edward Rothberg, of the CPLEX research and development leadership team left CPLEX to found Gurobi Optimization, named after combining the first two letters of each of the founders' last names [Rot]. Today's solvers use all the methods mentioned above and others, but advances in technology also allow them to run in parallel, meaning that multiple branches of the tree can be searched simultaneously. In LP problems, the barrier methods have been more effectively parallelized than the simplex methods, meaning that they have some advantages in newer computers [Bix12]. The culmination of all of these advances have meant that larger and larger MILP problems are now solvable, including TSP instances with tens of thousands of nodes [ABCC07]. We are now able to solve even larger instances to within certain tolerances, that is, the result is not guaranteed to be optimal. One such example is the star TSP with 526,280,881 nodes, solved to within 0.796% of an optimal tour [ABCC07]. These very large instances are challenge problems, but successes at the extreme end of the spectrum have applications down the line. Improvements in computation and algorithms that permit the solving of exceptional cases of the TSP also let us solve realistic cases of MILP problems faced by industry today.

1.4 Quadratic Traveling Salesman Problem

We will now examine a specific case of a modern variation of the TSP – the Quadratic Traveling Salesman Problem. In the Quadratic Traveling Salesman Problem (QTSP), we associate a cost with each edge (c_{ij}) as well as each pair of edges ($q_{ij,kl}$) [FH13]. Therefore, we are interested in not only which pairs of vertices are connected, but also which pairs of edges occur in the tour [FH13]. This allows us to model problems where there is a cost associated with each edge, but also with transitioning between edges, such as the cost of changing transportation types at that vertex [OFF$^+$17] . The quadratic version of the TSP in general is not well explored in literature, however there are some results for special cases of the QTSP. Since there are few results for this model, and there are applications for the QTSP in real world modeling, we believe that there is motivation to study this formulation in greater detail.

The Quadratic Traveling Salesman Problem was described by Jäger and Molitor in 2008 as a way to model problems in bioinformatics, specifically the Permuted Variable Length

Markov Model [JM08]. This version, named the 2-TSP or "second order TSP", considered edges crossed in succession, and they showed that the asymmetric TSP could be reduced to the asymmetric 2-TSP in polynomial time [JM08]. Their formulation and proposed algorithms rely on the structure of the problem, namely that there are costs associated with triples of vertices in succession [JM08]. We provide a few examples of other ways to model the quadratic objective function for the QTSP below.

Since we are concerned with pairs of edges that contribute to tours, we can represent the QTSP directly in the formulation:

$$\text{Minimize: } \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n} q_{ijk}x_{ij}x_{jk} + \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij}$$

where q_{ijk} represents the quadratic cost associated with edge pairs (i, j) and (j, k), and c_{ij} represents the edge cost [FH13]. The quadratic cost can be interpreted as the cost of the intersection of (i, j, k), or the cost to transfer from (i, j) to (j, k). This assumes that the cost for non-adjacent edges is zero [PW17], which is the primary format of the previous research. We call this form of the QTSP the *quadratic adjacent TSP*. This particular form has applications within transportation, where the quadratic costs may represent transfer costs at a vertex, and robotics, where changing directions is more costly than maintaining the current direction. In the former, we may consider examples in shipping where there are costs associated with traversing each edge, and then additional costs when loading or unloading cargo at a transfer point. The latter is known as the *Angular-Metric Traveling Salesman Problem* [OFF+17], and was first discussed by Aggarwal et al. in the late 1990s [ACK+00].

To formulate the angular-metric TSP, the vertices $(v^1, \ldots, v^n)$ represent points in Euclidean space, and we aim to minimize the sum of angles as the object moves through space [Fis13], [OFF+17], [ACK+00]. Then, every time we change direction (pass through a vertex), we want to calculate the difference between the previous direction (edge (ij)) and the current direction (edge (jk)). This becomes the quadratic cost associated with the angle changes [OFF+17], [Fis13], which are calculated as follows:

$$q^{\angle}_{ijk} := \arccos\left(\left(\frac{v^j - v^i}{||v^j - v^i||}\right)^{\mathsf{T}}\left(\frac{v^k - v^j}{||v^k - v^j||}\right)\right)$$

Like in the non-quadratic TSP, there exist symmetric and asymmetric versions of the QTSP. In the asymmetric QTSP, the cost of the edges depends on the direction of travel, while in the symmetric, they do not. Studies on the QTSP are far more limited than those

on the general TSP. We review some of the recent findings in the next paragraph.

The symmetric version of the quadratic adjacent TSP has been studied by [FH13], [OFF+17] among others. In [FH13], they focus on the structure of the associated polyhedron and lifting subtour elimination constraints to be facet-defining. This results in a closer polyhedron, which can improve the root relaxation and the resulting search tree. In [OFF+17], they test an integral approach for computational results, as well as introducing a mixed-integer linear programming linearization and some theoretical results for the max angle TSP. Linearizing the quadratic problem allows us to use the results of some of the other researchers mentioned in this paper, but can increase the problem size (number of variables and constraints) of the resulting polyhedron is not as close. Rostami and others studied lower-bounding procedures for the asymmetric quadratic adjacent TSP, as well as linearization techniques [RMBG16]. Other researchers have investigated specific instances of the QTSP, such as on Halin graphs [WPS17], or where the cost matrix has a fixed rank [WP17].

While solvers have come a long way in addressing the linear TSP, the QTSP is much more computationally challenging, meaning that even small sizes of the QTSP do not solve optimally within certain time limits. In general, it is not well studied, though certain formulations are important, especially as technology around robotics continues to evolve, and we try to model more complex transportation scenarios.

1.4.1 Linearizing the Quadratic Objective Function

Linearization, or the transformation from a non-linear function (such as a quadratic function) to a linear one, allows us to take advantage of proven solving techniques in Mixed Integer Linear Programming [HM09]. In order to use linear solving techniques, we replace all instances of products of two binary variables with a linear term, using additional variables and linear constraints. These additional variables and constraints increase the size of the problem, however there may be advantages to using linear solving methods over the non-linear ones. We will explore several linearization techniques and their impacts on solve time.

Linearization is a common technique in quadratic binary optimization problems. The drawbacks to linearization are that they can produce weak linear relaxation lower bounds, while simultaneously increasing the problem size through additional constraints and variables [HM09]. Therefore we are hoping to reduce complexity in the objective function, by removing the quadratic elements, while increasing the quantity or complexity of the constraints and variables. We are interested in seeing whether these techniques can be effective

when applied to the QTSP.

Linearization techniques are well established within the field of Operations Research. The first linearization we study, involving replacing the product of two binary variables by an additional binary variable, was proposed by several authors in the 1950s and 1960s [HM09], [Wat67]. This method was described by Watters [Wat67] and applied to a financial optimization program as an example. This method does lead to additional integer variables, which can reduce the benefit of linearization [Wat67], [GW74], [Glo75]. By modifying one of the additional constraints, Glover and Woolsey were able to relax the binary requirement on the linearization variable, establishing what became known as the "standard linearization" [GW74]. This is preferred over the original binary replacement as it keeps the number of integer variables the same as in the unmodified problem [Glo75].

We also study using the McCormick envelopes to surround the quadratic term. This method was proposed by McCormick in the 1970s as a method to transform mixed integer non-linear programming problems into convex non-linear programming problems [McC76]. By using this method, one can more easily find globally optimal solutions to non-linear problems, since the convexity of the transformation ensures that local minima are in fact global minima [McC76]. We achieve this by finding the convex over- and under-estimates for the quadratic portion of the objective function, and rely on the binary restriction of x_{ij}.

Finally, we examine two methods that replace the quadratic term with the difference of two integers, relying on the integrality of the quadratic cost matrix. In the two methods we use integers of base-2 and base-10 in the difference terms. The base-2 expansion was described by [Wat67], and has been used in mixed integer linear programming problems by [OM02] and in the QSCP by [Pan18]. We use the same method for the base-10 expansion. We provide more detail on how to formulate the linearizations in Chapter 4.

Other linearization techniques include a reformulation-linearization method proposed by Sherali and Adams [SA90], various compact methods studied by Glover and others [Glo75], [HM09] as well as other techniques proposed by [HM09], [Pan18], among others. We do not examine these methods in detail.

1.5 Contribution

In this thesis, we review the Traveling Salesman Problem and provide several integer programming formulations. We examine the generalization to a quadratic objective function in its most general form, using an integer programming approach, where we use three formulations (the Dantzig subtour elimination method, the Miller-Tucker-Zemlin (MTZ) method

and a flow-based formulation) in testing the effects on computation time of modifying the quadratic cost matrix and linearizations. In particular, we will use a general form of the QTSP, where there is a quadratic cost associated with any two edges in a tour, rather than applying the quadratic cost only to adjacent edges, as has been studied by other researchers.

1.5.1 Modifying the Quadratic Matrix

We provide seven modifications of the quadratic cost matrix that transforms the instance into an equivalent representation. We study the impact of the following: symmetrization, triangularization, convexification, concavification, node n removal, and symmetrization and triangularization of the node n removed matrix. We also retain the unmodified quadratic matrix as a control. Symmetrization and triangularization are common techniques, demonstrated in [Pan18], [HR70], among others, with applications beyond the QTSP. Convexification, or the transformation of a matrix into a positive semi-definite form, has also been used by [Pan18] and [HR70]. The original method subtracts the minimum eigenvalue of the quadratic cost matrix (Q) from the diagonal entries of Q [HR70], however we choose to add a sufficiently large positive value, and keep that value the same for all instances. We then apply the same method to transform Q into a negative semi-definite form, however we use a sufficiently large negative number. We apply the method described by [PW17] to transform Q into a node n-removed matrix, and then apply the symmetrization and triangularization techniques.

Several of these methods were used in [Pan18] with the aim of comparing their impact on solving instances of the quadratic set covering problem. The methods described have been applied to other problems, apart from the node-n removal, which has been applied to the QTSP, though not for computational experiments. We therefore believe that this presents an opportunity to test these methods on this quadratic problem.

1.5.2 Linearizations

We test the effect of five linearizations on computation time. We study the impact of the following: linearization using additional binary variables, standard linearization, McCormick envelope linearization, base-2 linearization, and base-10 linearization. Several of these methods were used by [Pan18], applied to a related combinatorial optimization problem, the quadratic set covering problem. We do not modify the quadratic matrix, and retain the quadratic form of the instance as the control. These methods have been applied to other quadratic problems, but not to this specific form of the QTSP.

Chapter 2

Integer Programming Formulations

In this chapter we review some of the popular formulations of the TSP and QTSP, and their linear relaxations, and provide some comments on their respective sizes. As there are several ways to describe the constraint set of the TSP, which all use the same objective functions, we describe the constraints which formulate relaxations of the TSP polytope in Section 2.1, and the objective functions in Sections 2.2 and 2.3. In all formulations we assume that there are n cities. We define the following variables:

$$x_{ij} = \begin{cases} 1 \text{ if the edge connecting } (i,j) \text{ is selected in the tour} \\ 0 \text{ otherwise} \end{cases}$$

Therefore, we can represent $x \in \{0,1\}^{n \times n}$.

2.1 Traveling Salesman Problem Constraints

We begin by describing the different ways to formulate the constraint set. Some formulations introduce additional variables to ensure that the solution forms a tour, while others can be used to model commodity flows. Some formulations may be more useful to researchers than others.

2.1.1 Dantzig Subtour Elimination

The general formulation, described by Dantzig, Fulkerson and Johnson in 1954 [DFJ54] while working at the RAND Corporation is as follows:

$$\text{subject to: } \sum_{j=1,\, j\neq i}^{n} x_{ij} = 1 \qquad \forall\, i = 1, \ldots, n \tag{1.10}$$

$$\sum_{i=1\, i\neq j}^{n} x_{ij} = 1 \qquad \forall\, j = 1, \ldots, n \tag{1.11}$$

$$\sum_{i \in S} \sum_{j \in S, j \neq i} x_{ij} \leq |S| - 1 \quad \forall\, S \subseteq \{2, \ldots, n\},\, 2 \leq |S| \leq n-1 \tag{1.12}$$

$$x_{ij} \in \{0, 1\} \qquad \forall\, i, j = 1, \ldots, n \tag{1.13}$$

Constraints (1.10) and (1.11) ensure that the tour passes through each city exactly once, while constraint (1.12) eliminates subtours, and are hence called "subtour elimination constraints". Note that there is an exponential number of subtour elimination constraints. The above formulation works for the asymmetric TSP. To modify it for the symmetric TSP, we can replace constraints (1.10) and (1.11) with

$$\sum_{j=1,\, j \neq i}^{n} x_{ij} = 2 \qquad \forall\, i = 1, \ldots, n \tag{1.14}$$

and replace the subtour elimination constraints with

$$\sum_{i \in S} \sum_{j \in S, j > i} x_{ij} \leq |S| - 1 \quad \forall\, S \subseteq \{2, \ldots, n\},\, 2 \leq |S| \leq n-1 \tag{1.15}$$

For both asymmetric and symmetric versions of the TSP, we can relax the integer constraint (1.13) to

$$0 \leq x_{ij} \leq 1 \qquad \forall\, i, j = 1, \ldots, n \tag{1.16}$$

The general formulation has $2^{n-1} + n - 1$ constraints, due largely to the subtour elimination constraints (1.12), and the resulting polyhedron is not integral, so we typically don't solve it in this precise form [OW06]. Instead, we use a method called *delayed constraint generation* where we solve the initial IP using some constraints (1.10), (1.11), and then add in the subtour elimination constraints only if they are violated. Specifically, we use a callback method within Gurobi, where we check the length of the shortest circuit. If the length of the shortest tour is less than n then Gurobi will add in a subtour elimination constraint that is violated, and reruns the optimization step. This process is repeated until an optimal solution of length n is found. This means that it only uses the constraints it needs to add, which may not be all of them.

2.1.2 Miller-Tucker-Zemlin (MTZ)

The MTZ formulation uses additional continuous variables to prevent subtours. The additional variables, u_i, represent vertex i's position in the tour. Constraints (1.10) and (1.11) are retained, however we replace (1.12) with

$$u_i - u_j + (n-1)x_{ij} \leq n - 2 \quad \forall\, i, j = 2, \ldots, n,\ i \neq j$$

If the tour does contain a subtour, then constraint (2.1) is violated for the subtour not containing vertex 1 [GP07]. Constraints (1.10) and (1.11) require that there must be at least two subtours, thus there must be one not containing vertex 1.

The full asymmetric MTZ formulation is:

$$\text{subject to: } \sum_{j=1,\, j\neq i}^{n} x_{ij} = 1 \qquad \forall\, i = 1, \ldots, n \tag{1.10}$$

$$\sum_{i=1,\, i\neq j}^{n} x_{ij} = 1 \qquad \forall\, j = 1, \ldots, n \tag{1.11}$$

$$u_i - u_j + (n-1)x_{ij} \leq n-2 \quad \forall\, i, j = 2, \ldots, n,\ i \neq j \tag{2.1}$$

$$1 \leq u_i \leq n-1 \quad \forall\, i = 2. \ldots, n \tag{2.2}$$

$$x_{ij} \in \{0, 1\} \qquad \forall\, i, j = 1, \ldots, n \tag{1.13}$$

Like in the general formulation, we can relax to a linear model by replacing (1.13) with:

$$0 \leq x_{ij} \leq 1 \qquad \forall\, i, j = 1, \ldots, n \tag{1.16}$$

Constraints (2.2) restrict u_i, however they are not required and do not affect the LP bound [ÖAL09]. Without the optional constraints, this formulation has n^2 binary variables and $n-1$ continuous variables, and $n^2 - n + 2$ constraints [OW06]. Constraints (2.2) adds $n-1$ constraints.

2.1.3 Desrochers and Laporte (DL)

We can lift the MTZ subtour elimination constraints (2.1) and (2.2) to the stricter forms:

$$u_i - u_j + (n-1)x_{ij} + (n-3)x_{ji} \leq n-2 \quad \forall\, i, j = 2, \ldots, n \tag{2.3}$$

$$1 + (n-3)x_{i1} + \sum_{j=2}^{n} x_{ji} \leq u_i \leq n-1-(n-3)x_{1i} - \sum_{j=2}^{n} x_{ij} \quad \forall\, i = 2, \ldots, n \tag{2.4}$$

as found by Desrochers and Laporte [ÖAL09, pg. 639]. This gives us the following formulation:

$$\text{subject to: } \sum_{j=1,\, j\neq i}^{n} x_{ij} = 1 \qquad \forall\, i = 1, \ldots, n \tag{1.10}$$

$$\sum_{i=1,\, i\neq j}^{n} x_{ij} = 1 \qquad \forall\, j = 1, \ldots, n \tag{1.11}$$

$$u_i - u_j + (n-1)x_{ij} + (n-3)x_{ji} \leq n-2 \quad \forall\, i,j = 2,\ldots,n,\ i \neq j \tag{2.5}$$

$$1 + (n-3)x_{i1} + \sum_{j=2}^{n} x_{ji} \leq u_i \leq n-1-(n-3)x_{1i} - \sum_{j=2}^{n} x_{ij} \ \forall\, i = 2,\ldots,n \tag{2.6}$$

$$x_{ij} \in \{0,1\} \qquad \forall\, i,j = 1,\ldots,n \tag{1.13}$$

To relax to a linear form, we replace (1.13) with:

$$0 \leq x_{ij} \leq 1 \qquad \forall\, i,j = 1,\ldots,n \tag{1.16}$$

This formulation has n^2 binary variables and $n-1$ continuous variables, and n^2+1 constraints, the same number as the full MTZ formulation, however, constraints (2.5) and (2.6) are facet defining [ÖAL09].

2.1.4 Single Commodity Flow (SCF)

The SCF formulation replaces the subtour elimination constraints with flow constraints that allow $n-1$ units of a single commodity to flow from vertex 1 to all other vertices [OW06]. We keep constraints (1.10) and (1.11), and introduce new variables

$$y_{ij} = \text{flow along edge } (i,j)$$

The SCF formulation is:

$$\text{subject to: } \sum_{j=1,\, j\neq i}^{n} x_{ij} = 1 \qquad \forall\, i = 1,\ldots,n \tag{1.10}$$

$$\sum_{i=1,\, i\neq j}^{n} x_{ij} = 1 \qquad \forall\, j = 1,\ldots,n \tag{1.11}$$

$$y_{ij} \leq (n-1)x_{ij} \quad \forall\, i,j = 1,\ldots,n,\ i \neq j \tag{2.7}$$

$$\sum_{j=2}^{n} y_{1j} = n-1 \tag{2.8}$$

$$\sum_{i=1}^{n} y_{ij} - \sum_{k=1}^{n} y_{jk} = 1 \quad \forall\, j = 2,\ldots,n \tag{2.9}$$

$$y_{ij} \geq 0 \quad \forall\, i,j = 1,\ldots,n \tag{2.10}$$

$$x_{ij} \in \{0,1\} \qquad \forall\, i,j = 1,\ldots,n \tag{1.13}$$

To relax to a linear form, we replace (1.13) with:

$$0 \leq x_{ij} \leq 1 \qquad \forall\, i,j = 1,\ldots,n \tag{1.16}$$

Constraint (2.7) allows the commodity to flow along edge i, j only if the edge is selected for the tour, while constraints (2.8) and (2.9) restrict the flow between vertices. These constraints fulfill the subtour elimination requirements [GP07]. We can strengthen (2.7) to:

$$y_{ij} \leq (n-2)x_{ij} \quad \forall\, i, j = 2, \ldots, n$$

since the initial $n-1$ flow from vertex 1 is reduced by one after the first stop of the tour, so all remaining edges must have at most $n-2$ units flowing along them [OW06]. This formulation has $n^2 + 2n$ constraints, n^2 binary variables and n^2 continuous variables.

2.1.5 Two Commodity Flow (TCF)

We can modify the SCF formulation to allow a second commodity type. In this formulation, we allow $n-1$ units of commodity y out of vertex 1, and $n-1$ units of commodity z into vertex 1. This gives us the following formulation [GP07, pg.19]:

$$\text{subject to: } \sum_{j=1,\, j\neq i}^{n} x_{ij} = 1 \qquad \forall\, i = 1, \ldots, n \tag{1.10}$$

$$\sum_{i=1,\, i\neq j}^{n} x_{ij} = 1 \qquad \forall\, j = 1, \ldots, n \tag{1.11}$$

$$\sum_{j=1}^{n} y_{1j} - \sum_{j=1}^{n} y_{j1} = n-1 \tag{2.11}$$

$$\sum_{j=1}^{n} y_{ij} - \sum_{j=1}^{n} y_{ji} = -1 \quad \forall\, i = 2, \ldots, n \tag{2.12}$$

$$\sum_{j=1}^{n} z_{1j} - \sum_{j=1}^{n} z_{j1} = -(n-1) \tag{2.13}$$

$$\sum_{j=1}^{n} z_{ij} - \sum_{j=1}^{n} z_{ji} = 1 \quad \forall\, i = 2, \ldots, n \tag{2.14}$$

$$\sum_{j=1}^{n} y_{ij} + \sum_{j=1}^{n} z_{ji} = n-1 \quad \forall\, i = 2, \ldots, n \tag{2.15}$$

$$y_{ij},\ z_{i,j} \geq 0 \quad \forall\, i, j = 1, \ldots, n \tag{2.16}$$

$$y_{ij} + z_{ij} = (n-1)x_{ij} \quad \forall\, i, j = 1, \ldots, n \tag{2.17}$$

$$x_{ij} \in \{0, 1\} \qquad \forall\, i, j = 1, \ldots, n \tag{1.13}$$

To relax to a linear form, we replace (1.13) with:

$$0 \leq x_{ij} \leq 1 \qquad \forall\, i, j = 1, \ldots, n \tag{1.16}$$

This formulation has $3n^2 + 5n - 1$ constraints, n^2 binary variables, and $2n^2$ continuous variables.

2.1.6 Multi-Commodity Flow (MCF)

We can extend the flow formulation to include n commodities going to n vertices. We alter the flow variables to

$$y_{ij}^k = \text{flow of commodity } k \text{ along edge } i, j$$

We replace the flow constraints from above to give us the following formulation [OW06]:

$$\text{subject to: } \sum_{j=1,\, j\neq i}^{n} x_{ij} = 1 \qquad \forall\, i = 1, \ldots, n \tag{1.10}$$

$$\sum_{i=1,\, i\neq j}^{n} x_{ij} = 1 \qquad \forall\, j = 1, \ldots, n \tag{1.11}$$

$$y_{ij}^k \leq x_{ij} \quad \forall\, i, j = 1, \ldots, n,\ k = 2, \ldots, n \tag{2.18}$$

$$\sum_{i=1}^{n} y_{1i}^k = 1 \quad \forall\, k = 2, \ldots, n \tag{2.19}$$

$$\sum_{i=1}^{n} y_{i1}^k = 0 \quad \forall\, k = 2, \ldots, n \tag{2.20}$$

$$\sum_{i=1}^{n} y_{ik}^k = 1 \quad \forall\, k = 2, \ldots, n \tag{2.21}$$

$$\sum_{j=1}^{n} y_{kj}^k = 0 \quad \forall\, k = 2, \ldots, n \tag{2.22}$$

$$\sum_{i=1}^{n} y_{ij}^k - \sum_{i=1}^{n} y_{ji}^k = 0 \quad \forall\, j, k = 2, \ldots, n, j \neq k \tag{2.23}$$

$$y_{ij}^k \geq 0 \qquad \forall\, i, j, k = 1, \ldots, n \tag{2.24}$$

$$x_{ij} \in \{0, 1\} \qquad \forall\, i, j = 1, \ldots, n \tag{1.13}$$

To relax to a linear form, we replace (1.13) with:

$$0 \leq x_{ij} \leq 1 \qquad \forall\, i, j = 1, \ldots, n \tag{1.16}$$

Constraints (2.18) allow the flow of any commodity only along edges that are selected for the tour. Constraints (2.19) and (2.20) allow one unit of each commodity out of vertex 1, and prevent any flow into vertex 1. Constraints (2.21) and (2.22) allow one unit of commodity k into vertex k, and prevents any flow of commodity k out of vertex k. Constraints (2.23) require all other commodities to balance [OW06].

This formulation has $n^3 + 3n - 2$ constraints, n^2 binary variables and n^3 continuous variables.

2.1.7 Formulation Strength Comparison

Öncan et al. summarized the relationships between the different integer formulations of the asymmetric TSP in [ÖAL09]. They showed that the MTZ formulation is weaker than the SCF formulation and that the SCF is weaker than the Dantzig Subtour formulation [ÖAL09]. As the DL formulation uses lifted MTZ constraints, we see that the DL formulation is stronger. The projected polyhedron of the MCF formulation is a proper subset of the projected polyhedron of the SCF, making the MCF a stronger formulation, while the SCF and TCF formulations are equivalent [ÖAL09].

2.2 Linear Objective Function

The objective of the TSP is to minimize the total cost of the tour. Therefore, we sum the cost over all edges, and the constraint set (whichever is used) will guarantee that the edges selected form a tour. We define the following:

$$c_{ij} = \text{cost (or distance) of traversing edge } (i, j)$$

We can represent $C \in \mathbb{M}^n$ with diagonal entries zero, where $\mathbb{M}^n$ is a square matrix of size n.

The objective function for the linear TSP can be formulated as follows:

$$\text{Minimize: } \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} c_{ij} x_{ij} \tag{2.25}$$

$$\tag{2.26}$$

2.3 Quadratic Objective Function

We will focus on a more general formulation of the asymmetric quadratic TSP of the objective function, where there is an additional cost associated with any two edges in the tour, even if the two are not adjacent. Rather than using the four-value index $[i, j, k, l]$, we will index by edge-pair. The edge-pair set, E, has size $n(n-1)$, to represent all combinations $(i, j), i \neq j$. As the cost $c_{ii} = 0$ and we do not permit loops from a vertex to itself, we can represent the cost matrix C as a vector in $\mathbb{R}^{n(n-1)}$. Note that x_{ii} does not exist, and so $q_{ii,jj}$ does not exist for any i, j. Therefore we can represent $Q \in \mathbb{M}^{n(n-1)}$. We aim to minimize the total cost of the tour. This will include the linear cost, which is the cost associated with

each edge, and the quadratic cost, which is the cost associated with each pair of edges. We therefore have two sums – one for the linear cost $\mathbf{c}$ and one for the quadratic cost Q. We formulate the objective function as follows:

$$\text{Minimize: } \sum_{ij \in E} \sum_{kl \in E} q_{ij,kl} x_{ij} x_{kl} + \sum_{ij \in E} c_{ij} x_{ij} \tag{2.27}$$

We note that the QTSP is NP-hard as it can be considered a generalization of the TSP, which is NP-hard. We can modify the previous formulations to take into account a quadratic objective function and edge-pair index.

Chapter 3

Modifying the Q Matrix

We consider modifying the cost matrix Q such that the cost of the tour remains unchanged for all tours, but the structure of the matrix may allow us to solve the QTSP in less time. We investigate the computational effects of the following equivalent representations: symmetrization, triangularization, negative semidefiniteness, positive semidefiniteness, node n removal, and symmetrization and triangularization of the node n removed matrix.

Upper case letters indicate matrices, while lower case letters indicate entries within a matrix. We use lowercase **bold** to indicate vectors. We use superscripts (Q^m) to indicate the various modifications we plan to explore, where m is the shorthand for the modification. For entries within a matrix, we use $q^m_{ij,kl}$ where ij and kl represent the position within the matrix, and m is the modification.

Let $(Q, \mathbf{c})$ be the initial quadratic and linear costs of a problem instance. We describe Q in section 2.3. Recall that $Q \in \mathbb{M}^{n(n-1)}$, while $\mathbf{c} \in \mathbb{R}^{n(n-1)}$. Then let $f(Q, \mathbf{c}, \mathbf{x})$ be the QTSP associated with a solution $\mathbf{x} \in \{0,1\}^{n(n-1)}$. Then

$$f(Q, C, \mathbf{x}) := \sum_{ij \in E} \sum_{kl \in E} q_{ij,kl} x_{ij} x_{kl} + \sum_{ij \in E} c_{ij} x_{ij}$$

We will modify the cost matrices such that $f(Q, \mathbf{c}, \mathbf{x}) = f(Q^m, \mathbf{c}^m, \mathbf{x}) + \delta^m$, where δ^m is a constant term based on the size of n. We can then say that $(Q^m, \mathbf{c}^m)$ is *equivalent* to $(Q, \mathbf{c})$ up to a constant δ^m. The goal of these modifications is to put $(Q, \mathbf{c})$ into a more favourable form for solving.

3.1 Symmetric Q

Consider the matrix $Q^S = \frac{1}{2}(Q + Q^\intercal)$. Entrywise, $q^S_{ij,kl} = \frac{1}{2}(q_{ij,kl} + q_{kl,ij})$, for all ij, kl in the edge set E. Given that $x_{ij}x_{kl} = x_{kl}x_{ij}$, the quadratic cost associated with this technique is unchanged, and no modifications are required for $\mathbf{c}, \delta$. Therefore we claim that

$f(Q, \mathbf{c}, \mathbf{x}) = f(Q^S, \mathbf{c}, \mathbf{x})$, and $(Q^S, \mathbf{c})$ is equivalent to $(Q, \mathbf{c})$.

3.2 Upper Triangular Q

We define an upper triangular matrix Q^U where $Q^U = (Q+Q^T)$ for the upper triangular values, and $Q^U = 0$ for the lower triangular values. That is,

$$q^U_{ij,kl} = \begin{cases} q_{ij,kl} + q_{kl,ij} & \text{if } kl > ij \\ q_{ij,kl} & \text{if } ij = kl \\ 0 & \text{otherwise} \end{cases}$$

Given that $x_{ij}x_{kl} = x_{kl}x_{ij}$, $q_{ij,kl}x_{ij}x_{kl} + q_{kl,ij}x_{kl}x_{ij} = x_{ij}x_{kl}(q_{ij,kl} + q_{kl,ij})$, and $f(Q, \mathbf{c}, x) = f(Q^U, \mathbf{c}, \mathbf{x})$. Therefore, $(Q^U, \mathbf{c})$ is equivalent to $(Q, \mathbf{c})$.

3.3 Positive Semi-Definite Q

We replace Q with a positive semi-definite matrix by letting $Q^P = Q + MI$, where M is a sufficiently large real number, and I is an $n(n-1)$ identity matrix. We modify only the diagonal values of Q^P, by letting

$$q^P_{ij,kl} = \begin{cases} q_{ij,kl} + M & \text{if } ij = kl \\ q_{ij,kl} & \text{otherwise} \end{cases}$$

Choosing a sufficiently large M value ensures that all eigenvalues of Q^P are nonnegative, and therefore the matrix is positive semi-definite. We can choose a value of M that guarantees that the resulting Q^P is diagonally dominant, as the diagonal entries will be greater than the absolute sum of the remaining elements of each row, and the diagonal entries are positive [Bri]. Gershgorin's Circle Theorem implies that Q^P has nonnegative eigenvalues, and thus it is positive semi-definite [Wei], [Ser10]. We provide the full statement of the theorem from [HJ85]:

> **Gershgorin's Circle Theorem.** Let A be a complex square $n \times n$ matrix. We define
>
> $$R_i = \sum_{j=1, j\neq i}^{n} |a_{ij}|$$
>
> Then each eigenvalue, λ, will be in at least one of the disks defined by (a_{ii}, R_i), where a_{ii} is the centre of the disk and R_i is the radius, such that $|\lambda - a_{ii}| \leq R_i$.

We want to choose a value for M that ensures that $a_{ii} + M$ is nonnegative, so that the extreme parts of the disk are nonnegative. Thus it is sufficient to take $M = \max\{R_i + |a_{ii}|\}$,

but we may also take it to be larger. We provide more detail on choosing an M in the results section.

We now show that $f(Q, \mathbf{c}, \mathbf{x}) = f(Q^P, \mathbf{c}, \mathbf{x}) + \delta$ and determine the value of δ:

$$\begin{aligned} f(Q^P, \mathbf{c}, \mathbf{x}) &= f(Q + MI, \mathbf{c}, \mathbf{x}) \\ &= \sum_{ij \in E} \sum_{kl \in E} q_{ij,kl} x_{ij} x_{kl} + \sum_{ij \in E} M x_{ij} x_{ij} + \sum_{ij \in E} c_{ij} x_{ij} \end{aligned}$$

Since $x_{ij} \in \{0, 1\}, x_{ij} x_{ij} = x_{ij}$

$$= \sum_{ij \in E} \sum_{kl \in E} q_{ij,kl} x_{ij} x_{kl} + M \sum_{ij \in E} x_{ij} + \sum_{ij \in E} c_{ij} x_{ij}$$

Since $\mathbf{x}$ forms a tour, $\sum_{ij \in E} x_{ij} = n$, therefore

$$= \sum_{ij \in E} \sum_{kl \in E} q_{ij,kl} x_{ij} x_{kl} + Mn + \sum_{ij \in E} c_{ij} x_{ij}$$

$$f(Q^P, \mathbf{c}, \mathbf{x}) = f(Q, \mathbf{c}, \mathbf{x}) + Mn$$

Therefore, $\delta = -Mn$ and $f(Q, \mathbf{c}, \mathbf{x}) = f(Q^P, \mathbf{c}, \mathbf{x}) - Mn$, with Mn being a constant. Then $(Q^P, \mathbf{c})$ $-Mn$ is equivalent to $(Q, \mathbf{c})$.

3.4 Negative Semi-Definite Q

We replace Q with a negative semi-definite matrix by letting $Q^N = Q - MI$, where M is a sufficiently large number, and I is an $n(n-1)$ identity matrix. We modify only the diagonal values of Q^N, by letting

$$q^N_{ij,kl} = \begin{cases} q_{ij,kl} - M & \text{if } ij = kl \\ q_{ij,kl} & \text{otherwise} \end{cases}$$

We use the same method as above to guarantee that the matrix is diagonally dominant, however the diagonal entries are now negative, ensuring that the matrix is negative semi-definite.

We use the process detailed above to show that $f(Q, \mathbf{c}, \mathbf{x}) = f(Q^N, \mathbf{c}, \mathbf{x}) + \delta$ and determine the value of δ:

$$f(Q^N, \mathbf{c}, \mathbf{x}) = f(Q, \mathbf{c}, \mathbf{x}) - Mn$$

We then observe that $f(Q, \mathbf{c}, \mathbf{x}) = f(Q^N, \mathbf{c}, \mathbf{x}) + Mn$, so $(Q^P, \mathbf{c})$ $+Mn$ is equivalent to $(Q, \mathbf{c})$.

3.5 Node n Removal

In this modification, we replace costs associated with the final node with zeros. We then capture those costs in the other parts of the Q matrix and $\mathbf{c}$ vector, such that given a tour $\mathbf{x}$, $f(Q, \mathbf{c}, \mathbf{x}) = f(Q^R, \mathbf{c}^R, \mathbf{x})$. Specifically, we use the method described in [PW17]:

$$q^R_{ij,kl} = \begin{cases} q_{ij,kl} - q_{ij,kn} - q_{ij,nl} - q_{in,kl} + q_{in,kn} \\ \quad + q_{in,nl} - q_{nj,kl} + q_{nj,kn} + q_{nj,nl} & \text{if } i,j,k,l \neq n \text{ and either } i \neq k \text{ or } j \neq l \\ 0 & \text{otherwise} \end{cases}$$

We define

$$c^R_{ij} = c_{ij} + q^R_{ij,ij} + \sum_{k=1}^{n-1} (q_{ij,kn} + q_{ij,nk} + q_{kn,ij} - q_{kn,in} - q_{kn,nj} + q_{nk,ij} - q_{nk,in} - q_{nk,nj})$$

for $i \neq j$, and where

$$q^R_{ij,ij} = \begin{cases} q_{ij,ij} - q_{ij,in} - q_{ij,nj} - q_{in,ij} + q_{in,in} + q_{in,nj} - q_{nj,ij} + q_{nj,in} + q_{nj,nj} & \text{if } i,j \neq n \\ 0 & \text{otherwise} \end{cases}$$

We demonstrate the use of this method with an instance of three cities. We first calculate Q^R. All entries associated with an edge entering or leaving the nth node (in this case city 2) will be zero. We also let the diagonal values of Q^R be zero, and add $q^R_{ij,ij}$ to c_{ij}.

$$Q^R = \begin{array}{c} \\ 01 \\ 02 \\ 10 \\ 12 \\ 20 \\ 21 \end{array} \begin{array}{c} \begin{array}{cccccc} 01 & 02 & 10 & 12 & 20 & 21 \end{array} \\ \begin{bmatrix} 0 & 0 & q^R_{01,10} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ q^R_{10,01} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array}$$

where

$$q^R_{01,10} = q_{01,10} - q_{01,12} - q_{01,20} - q_{02,10} + q_{02,12} + q_{02,20} - q_{21,10} + q_{21,12} + q_{21,20}$$
$$q^R_{10,01} = q_{10,01} - q_{10,02} - q_{10,21} - q_{12,01} + q_{12,02} + q_{12,21} - q_{20,01} + q_{20,02} + q_{20,21}$$

We want to show that the cost associated with a tour is the same for both the modified and unmodified versions of Q. The cost from Q^R will be zero, as the pairs associated with this tour have cost 0 in the matrix, and all those costs are loaded on the linear portion of the cost structure. We now calculate $\mathbf{c}^R$:

$$\mathbf{c}^R = \begin{bmatrix} c_{01} \\ c_{02} \\ c_{10} \\ c_{12} \\ c_{20} \\ c_{21} \end{bmatrix} + \begin{bmatrix} c^r_{01} \\ c^r_{02} \\ c^r_{10} \\ c^r_{12} \\ c^r_{20} \\ c^r_{21} \end{bmatrix}$$

where $c^r_{ij} = c^R_{ij} - c_{ij}$, which is to say, the portion of $\mathbf{c}^R$ associated with the modification. We now find the values of $\mathbf{c}^r$:

$$\begin{aligned} c^r_{01} &= q^R_{01,01} + q_{01,02} + q_{01,20} + q_{02,01} - q_{02,02} - q_{02,21} + q_{20,01} - q_{20,02} - q_{20,21} \\ &\quad + q_{01,12} + q_{01,21} + q_{12,01} - q_{12,02} - q_{12,21} + q_{21,01} - q_{21,02} - q_{21,21} \end{aligned}$$

where $q^R_{01,01} = q_{01,01} - q_{01,02} - q_{01,21} - q_{02,01} + q_{02,02} + q_{02,21} - q_{21,01} + q_{21,02} + q_{21,21}$

$$\begin{aligned} c^r_{02} &= q^R_{02,02} + q_{02,02} + q_{02,20} + q_{02,02} - q_{02,02} + q_{20,02} - q_{20,02} \\ &\quad + q_{02,12} + q_{02,21} + q_{12,02} - q_{12,02} + q_{21,02} - q_{21,02} \end{aligned}$$

where $q^R_{02,02} = 0$

$$\begin{aligned} c^r_{12} &= q^R_{12,12} + q_{12,02} + q_{12,20} + q_{02,12} - q_{02,12} + q_{20,12} - q_{20,12} \\ &\quad + q_{12,12} + q_{12,21} + q_{12,12} - q_{12,12} + q_{21,12} - q_{21,12} \end{aligned}$$

where $q^R_{12,12} = 0$

$$\begin{aligned} c^r_{10} &= q^R_{10,10} + q_{10,02} + q_{10,20} + q_{02,10} - q_{02,12} - q_{02,20} + q_{20,10} - q_{20,12} - q_{20,20} \\ &\quad + q_{10,12} + q_{10,21} + q_{12,10} - q_{12,12} - q_{12,20} + q_{21,10} - q_{21,12} - q_{21,20} \end{aligned}$$

where $q^R_{10,10} = q_{10,10} - q_{01,02} - q_{10,20} - q_{12,10} + q_{12,12} + q_{12,20} - q_{20,10} + q_{20,12} + q_{20,20}$

$$\begin{aligned} c^r_{20} &= q^R_{20,20} + q_{20,02} + q_{20,20} + q_{02,20} - q_{02,20} + q_{20,20} - q_{20,20} \\ &\quad + q_{20,12} + q_{20,21} + q_{12,20} - q_{12,20} + q_{21,20} - q_{21,20} \end{aligned}$$

where $q^R_{20,20} = 0$

$$c^r_{21} = q_{21,02} + q_{21,20} + q_{02,21} - q_{02,21} + q_{20,21} - q_{20,21}$$

$$+ q_{21,12} + q_{21,21} + q_{12,21} - q_{12,21} + q_{21,21} - q_{21,21}$$

where $q^R_{21,21} = 0$

Which simplifies to:

$$\begin{aligned}
c^r_{01} &= q_{01,20} + q_{02,01} + q_{20,01} - q_{20,02} - q_{20,21} + q_{01,12} - q_{12,02} - q_{12,21} + q_{01,01} \\
c^r_{02} &= q_{02,02} + q_{02,20} + q_{02,12} + q_{02,21} \\
c^r_{12} &= q_{12,02} + q_{12,20} + q_{12,12} + q_{12,21} \\
c^r_{10} &= q_{02,10} - q_{02,12} - q_{02,20} + q_{10,12} + q_{10,21} + q_{21,10} - q_{21,12} - q_{21,20} + q_{10,10} \\
c^r_{20} &= q_{20,02} + q_{20,20} + q_{20,12} + q_{20,21} \\
c^r_{21} &= q_{21,02} + q_{21,20} + q_{21,12} + q_{21,21}
\end{aligned}$$

The TSP of size $n = 3$ has only two tours in the asymmetric case.

$$T_1 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{bmatrix} \qquad T_2 = \begin{bmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

We complete the example using T_1. The cost associated with the unmodified Q for T_1 will be:

$$f(Q, \mathbf{c}, T_1) =$$
$$c_{01} + c_{12} + c_{20} + q_{01,12} + q_{12,01} + q_{20,01} + q_{01,20} + q_{12,20} + q_{20,12} + q_{01,01} + q_{12,12} + q_{20,20}$$

Recall that the quadratic cost associated with Q^R is zero, as edges 01 and 10 are not used together. Then the total cost for the quadratic reduced tour will be $f(Q^R, \mathbf{c}^R, T_1) = c^R_{01} + c^R_{12} + c^R_{20} = c^r_{01} + c^r_{12} + c^r_{20} + c_{01} + c_{12} + c_{20}$. After summing and cancelling, we can see that $f(Q, \mathbf{c}, T_1) = f(Q^R, \mathbf{c}^R, T_1)$. In this instance, the Q^R matrix is left with two entries, and the remainder of the costs are loaded on the $\mathbf{c}^R$ vector, but in general the format is that the rows and columns indexed by n are 0s, as are the diagonal elements.

Note that node n removal can be done in conjunction with either making the matrix symmetric or upper triangular.

3.5.1 Symmetric Node n Removal

We make the above modification symmetric by following the procedure outlined for Q^S. Therefore, $Q^{RS} = \frac{1}{2}(Q^R + (Q^R)^\intercal)$. Then, $f(Q, \mathbf{c}, \mathbf{x}) = f(Q^{RS}, \mathbf{c}^R, \mathbf{x})$.

3.5.2 Upper Triangular Node n Removal

We modify Q^R to make it into an upper triangular matrix, following the same procedure as we did in in Q^U. Therefore,

$$q^{RU}_{ij,kl} = \begin{cases} q^R_{ij,kl} + q^R_{kl,ij} & \text{if } kl > ij \\ q^R_{ij,kl} & \text{if } ij = kl \\ 0 & \text{otherwise} \end{cases}$$

Then, $f(Q, \mathbf{c}, \mathbf{x}) = f(Q^{RU}, \mathbf{c}^R, \mathbf{x})$

3.6 Computational Experiments

We tested the effect of the modifications and initial quadratic cost matrices on three quadratic TSP models: the original Dantzig formulation, the MTZ formulation, and the single commodity flow formulation. All models were written in Python and solved using Gurobi 8.0.1, on a workstation with the following configuration: 16GB RAM, 64-bit Windows 10 OS, Intel i7-4770 3.40 GHz processor.

The models were coded using GurobiPy, and can be located at the following url:
https://github.com/mrosespencer/QTSP

We include the python models, cost matrices for Q and C, and the code used to generate the random instances. The experiment files run through the different modifications for each instance, and outputs data on the time, objective value, Gurobi code and resulting tour. More details are provided in the github repository.

We address the following question: which modification to the quadratic cost matrix produces the best time results?

3.6.1 Gurobi Solver

In the introduction, we reviewed the history of computational solvers and some of the varied methods they use. We choose to use Gurobi for our experimentation, using an academic license. We do not know precisely how Gurobi is solving the models we submit, however, for quadratic linear programming problems, Gurobi uses simplex and barrier algo-

rithms, and for mixed integer problems, it uses a variety of algorithms, including heuristics, cutting planes and solution improvement. When we review the results, we see that Gurobi adds cutting planes and uses a search tree to compare solutions.

3.6.2 Problem Generation

We choose to use randomly generated instances due to the complexity of the QTSP. There are TSP libraries, such as [tsp], from which problem instances can be retrieved, however these are currently limited to linear instances of the TSP. Due to the quadratic terms in the model, the QTSP is more computationally difficult than linear forms of the TSP [OFF+17]. The smallest TSP in the TSPLIB is size 17, however we found that QTSP codes did not solve random problems of size 15. We therefore generate test instances ourselves, with a variety of properties, detailed below. These cost matrices, as well as the code used to generate them, are available in the github repository associated with this research.

We generate several initial Q matrices for our experimentation, as described in [PP18]. See Table 3.1 for details on quadratic cost generation. We let all values of $\mathbf{c}$ be uniformly randomly set to either 0 or 1 for all formulations. For all random numbers, we use the randint function within Python. We generated ten of size 8, five of size 10, one of size 12, and one of size 15. We let the solve time limit be 3 hours for problems of all sizes. We report the average solve time in seconds for the ten size 8 problems, and the solve time in seconds for each of the size 10 and 12 problems. Size 15 problems do not solve within the time limit and are therefore excluded.

Table 3.1: Quadratic Cost Generation

	Property of Q	Method of generation
1	Non-negative elements	Uniformly distributed random integers in range [5, 10]
2	Random and balanced	Uniformly distributed random integers in range [-5, 5]
3	Random and positively skewed	Uniformly distributed random integers in range [-5, 10]
4	Random and negatively skewed	Uniformly distributed random integers in range [-10, 5]
5	Positive semi-definite	A random matrix $\mathbf{B}$ is generated with elements uniformly distributed in the range [-5, 5], and Q is set to $\mathbf{BB}^T$
6	Non-negative and positive semi-definite	A random matrix $\mathbf{B}$ is generated with elements uniformly distributed in the range [5, 10], and Q is set to $\mathbf{BB}^T$
7	Rank 1	We let $\mathbf{a}$ be a column vector of uniformly random integers in the range [-10, 10], and $\mathbf{b}$ be a row vector of uniformly random integers in the range [-5, 5]. We set Q to $\mathbf{ab}$
8	Rank 2	We let $\mathbf{a}_1, \mathbf{a}_2$ be column vectors of uniformly random integers in the range [-10, 10], and $\mathbf{b}_1, \mathbf{b}_2$ be row vectors of uniformly random integers in the range [-5, 5]. We set Q to $\mathbf{a}_1\mathbf{b}_1 + \mathbf{a}_2\mathbf{b}_2$

For each characteristic (numbered 1 through 8 in Table 3.1) we generated 100 size 5 problems, 10 size 8 problems, 5 size 10 problems, 5 size 12 problems, and one each of the larger problems (sizes 15, 20, 25 and 30). We make the same number of C matrices for each size. All cost data are located in the Github repository in a folder labelled "Cost".

3.7 QTSP Modifications Results

We now present the results of the computational experiments. We report the time in seconds to solve to optimality for each of the modifications. In instances where the time limit of three hours is reached, we replace the time value with "-". We turn off Gurobi's Presolve and PreQLinearize parameters. We use $M = 10000$ for the PSD and NSD modifications. We detail how we decided to use this value for M in Section 3.7.4. The modification with the best solve time is **bolded** for each Q. We use the following short hand:

S	Sym	Symmetric
U	UT	Upper Triangular
P	PSD	Positive Semi-Definite
N	NSD	Negative Semi-Definite
R	QR	Quadratic Reduced (node n removal)
s	Sym QR	Symmetric Quadratic Reduced
u	UT QR	Upper Triangular Quadratic Reduced

3.7.1 Quadratic Dantzig Subtour Elimination Formulation

We use the constraints described in Section 2.1.1 and the objective function described in Section 2.3. We use Gurobi's $addVar$ to create the binary and continuous variables, and use $QuadExpr$ for the quadratic objective function. We add the constraints (1.10) and (1.11) as described in section 2.1.1, and set a time limit of three hours. We then use a Gurobi callback function to add in the subtour elimination constraints (1.12) one at a time as they are violated. To achieve this, we find the shortest length subtour within the tour. If the length of the shortest subtour is less than $n-1$, we add in the subtour constraints that are violated. This approach is modified from [ECI].

Table 3.2: Q Dantzig Nonnegative Q Time Values

Size	Original	Sym	UT	PSD	NSD	QR	Sym QR	UT QR
8	0.735	0.726	0.714	0.873	0.245	**0.233**	**0.233**	0.236
10	47.0	46.7	45.8	50.2	**3.84**	11.7	11.7	11.7
10	52.7	52.3	52.7	49.1	**3.42**	8.51	8.41	8.45
10	50.1	50.0	49.9	52.1	**3.89**	15.7	15.7	15.6
10	47.0	47.8	47.5	53.9	**5.09**	10.6	10.5	10.6
10	50.8	50.7	50.8	50.4	**4.03**	14.3	14.5	14.3
12	-	-	-	-	**312**	6200	6270	7470

Table 3.3: Q Dantzig Balanced Q Time Values

Size	Original	Sym	UT	PSD	NSD	QR	Sym QR	UT QR
8	0.731	0.730	0.744	0.700	**0.237**	0.283	0.280	0.271
10	50.2	49.4	49.2	53.3	**7.06**	14.7	14.5	14.6
10	52.7	51.3	52.7	49.3	**6.91**	18.8	18.8	18.7
10	48.9	49.2	49.2	54.1	**7.92**	20.0	19.7	20.1
10	54.8	54.9	55.2	54.6	**7.77**	15.0	15.1	15.2
10	50.7	50.8	50.4	51.5	**7.48**	14.8	14.7	14.6
12	-	-	-	-	**1070**	-	-	-

Table 3.4: Q Dantzig Positively Skewed Q Time Values

Size	Original	Sym	UT	PSD	NSD	QR	Sym QR	UT QR
8	0.813	0.769	0.780	0.775	**0.230**	0.267	0.265	0.266
10	60.1	59.5	59.3	56.4	**12.4**	17.8	17.8	17.9
10	60.9	60.9	61.0	58.5	**11.8**	18.1	18.1	18.1
10	61.9	61.5	61.9	56.2	**9.75**	13.8	13.7	13.6
10	57.2	56.2	56.7	58.7	**8.42**	17.6	17.4	17.5
10	57.6	58.3	58.2	61.6	**9.37**	13.6	13.7	13.9
12	-	-	-	-	**1660**	9570	9760	10 500

Table 3.5: Q Dantzig Negatively Skewed Q Time Values

Size	Original	Sym	UT	PSD	NSD	QR	Sym QR	UT QR
8	0.834	0.808	0.801	0.850	0.724	**0.328**	0.332	0.329
10	59.2	58.3	57.9	57.7	47.8	18.2	18.2	**17.9**
10	58.3	59.3	59.1	60.5	48.7	**20.6**	20.9	20.9
10	62.3	61.4	60.3	62.7	44.1	26.2	26.2	**26.2**
10	58.7	58.2	58.3	60.1	47.8	34.2	**33.8**	34.6
10	61.9	61.4	60.8	60.7	48.9	20.9	**20.7**	21.0
12	-	-	-	-	-	-	-	-

Table 3.6: Q Dantzig Positive Semi-Definite Q Time Values

Size	Original	Sym	UT	PSD	NSD	QR	Sym QR	UT QR
8	0.779	0.743	0.760	0.701	**0.282**	0.288	0.290	0.296
10	56.0	55.4	54.2	52.6	**10.9**	19.4	19.7	19.4
10	51.6	52.0	52.4	55.6	**10.3**	18.0	18.1	18.1
10	51.4	50.6	50.6	49.6	**9.64**	16.3	16.4	16.4
10	55.7	55.7	55.6	54.7	**10.3**	14.7	14.6	14.6
10	46.0	45.5	45.8	45.3	**9.28**	15.7	15.7	15.9
12	-	-	-	-	**6580**	-	-	-

Table 3.7: Q Dantzig Nonnegative and Positive Semi-Definite Q Time Values

Size	Original	Sym	UT	PSD	NSD	QR	Sym QR	UT QR
8	0.790	0.774	0.785	0.834	0.369	0.270	**0.259**	0.260
10	50.7	49.0	49.8	52.3	**12.4**	15.4	15.5	15.3
10	52.2	52.6	53.0	51.2	**13.4**	13.8	13.9	13.6
10	50.4	50.4	50.3	53.8	**12.3**	14.2	14.0	14.1
10	49.6	48.6	48.7	51.0	12.9	11.5	11.2	**11.2**
10	52.8	51.1	51.2	51.0	13.7	9.93	**9.76**	9.78
12	-	-	-	-	**2640**	9130	9490	9890

Table 3.8: Q Dantzig Rank One Q Time Values

Size	Original	Sym	UT	PSD	NSD	QR	Sym QR	UT QR
8	0.277	0.250	0.263	0.444	0.236	0.200	0.203	**0.197**
10	13.1	12.6	12.7	24.5	12.0	10.3	10.4	**10.2**
10	8.16	8.18	8.24	22.2	**5.08**	7.24	7.29	7.31
10	**5.41**	5.48	5.47	19.0	6.71	8.17	8.21	8.04
10	9.24	9.22	**9.20**	19.8	9.54	15.5	15.4	15.8
10	20.3	20.0	20.6	19.5	4.34	**4.33**	4.86	4.46
12	891	904	911	-	**869**	1040	1040	1070

Table 3.9: Q Dantzig Rank Two Q Time Values

Size	Original	Sym	UT	PSD	NSD	QR	Sym QR	UT QR
8	0.510	0.493	0.499	0.495	**0.242**	0.261	0.247	0.246
10	14.3	13.8	13.9	28.9	**6.02**	8.49	8.37	8.23
10	42.0	41.6	41.5	40.4	15.0	13.0	**12.9**	13.0
10	21.6	21.6	21.6	22.6	4.32	**4.01**	4.14	4.06
10	13.5	13.5	13.4	36.7	**8.51**	11.3	11.2	11.4
10	24.2	24.0	24.1	25.8	**6.18**	10.2	10.4	10.2
12	-	-	-	-	**416**	3700	3450	3800

3.7.2 Quadratic MTZ Formulation

We use the constraints described in Section 2.1.2 and the objective function described in Section 2.3. We use Gurobi's $addVar$ to create the binary and continuous variables, and use $QuadExpr$ for the quadratic objective function.

Table 3.10: QMTZ Nonnegative Q Time Values

Size	Original	Sym	UT	PSD	NSD	QR	Sym QR	UT QR
8	0.678	0.686	0.706	0.670	0.300	0.305	**0.286**	0.289
10	39.0	38.8	38.3	38.0	**4.71**	12.6	13.3	13.0
10	40.5	39.3	40.6	37.4	**3.52**	9.55	9.98	9.94
10	42.6	44.4	43.4	38.4	**3.95**	16.7	16.8	16.6
10	37.7	38.9	37.7	39.1	**6.05**	10.3	10.1	10.1
10	33.8	34.1	34.2	35.2	**4.24**	12.0	12.1	12.0
12	10 300	-	-	-	**280**	2820	3080	2930

Table 3.11: QMTZ Balanced Q Time Values

Size	Original	Sym	UT	PSD	NSD	QR	Sym QR	UT QR
8	0.588	0.603	0.582	0.533	**0.272**	0.322	0.317	0.327
10	32.9	33.0	32.9	36.8	**6.13**	12.3	12.3	12.5
10	36.3	36.1	36.6	35.5	**6.27**	18.1	18.6	18.2
10	36.3	36.5	36.5	37.3	**7.62**	16.8	16.7	16.6
10	31.8	32.1	32.2	35.4	**7.06**	14.3	14.5	14.4
10	36.2	36.1	36.2	37.0	**6.31**	13.1	13.1	13.2
12	7440	7190	7400	7320	**522**	2290	2410	2390

Table 3.12: QMTZ Positively Skewed Q Time Values

Size	Original	Sym	UT	PSD	NSD	QR	Sym QR	UT QR
8	0.567	0.550	0.545	0.606	0.288	**0.277**	0.278	0.286
10	34.1	35.0	34.1	36.7	**11.2**	16.2	16.1	16.5
10	34.1	34.6	34.1	35.8	**10.9**	15.9	15.9	15.9
10	34.8	33.9	34.3	35.5	**8.09**	12.2	11.8	11.9
10	34.9	35.0	34.9	36.4	**7.79**	14.6	14.5	14.4
10	36.3	35.8	37.0	35.6	**7.42**	12.1	12.2	12.0
12	-	7670	8240	8480	**697**	1530	1520	1610

Table 3.13: QMTZ Negatively Skewed Q Time Values

Size	Original	Sym	UT	PSD	NSD	QR	Sym QR	UT QR
8	0.659	0.649	0.633	0.619	0.584	**0.349**	0.364	0.361
10	38.1	37.5	37.5	37.6	37.7	17.6	17.7	**17.6**
10	37.5	37.6	37.4	38.5	36.5	17.7	**17.6**	17.7
10	35.6	35.2	35.9	35.9	36.1	22.0	**22.0**	22.0
10	34.4	34.8	34.5	37.3	37.2	29.0	29.1	**28.9**
10	36.9	36.8	37.0	37.8	35.0	18.5	18.3	**18.2**
12	6680	8850	-	-	10 100	4190	4170	**4040**

Table 3.14: QMTZ Positive Semi-Definite Q Time Values

Size	Original	Sym	UT	PSD	NSD	QR	Sym QR	UT QR
8	0.580	0.592	0.571	0.557	**0.306**	0.322	0.320	0.317
10	33.7	33.6	33.7	37.7	**10.2**	18.3	18.5	17.9
10	36.7	36.9	36.7	37.6	**8.86**	14.9	14.2	14.1
10	35.8	35.0	35.1	35.5	**9.73**	14.0	13.5	13.7
10	35.1	35.4	35.1	36.0	**8.41**	12.6	12.9	12.7
10	35.1	35.2	35.0	35.4	**9.87**	15.1	15.1	15.3
12	7230	7500	7490	7330	**1540**	2970	2950	2910

Table 3.15: QMTZ Nonnegative and Positive Semi-Definite Q Time Values

Size	Original	Sym	UT	PSD	NSD	QR	Sym QR	UT QR
8	0.680	0.689	0.689	0.650	0.395	**0.281**	0.292	0.300
10	37.5	37.4	37.5	36.3	**13.5**	14.9	14.8	14.9
10	40.0	39.9	40.0	34.1	13.5	13.6	13.2	**13.1**
10	31.8	32.1	31.9	36.4	**9.80**	13.0	13.0	13.2
10	36.6	36.7	36.5	34.0	**11.6**	11.9	11.7	11.7
10	38.2	37.9	38.1	35.9	11.9	9.11	9.21	**9.04**
12	-	-	-	-	**845**	2390	2360	2490

Table 3.16: QMTZ Rank One Q Time Values

Size	Original	Sym	UT	PSD	NSD	QR	Sym QR	UT QR
8	0.272	0.280	0.278	0.455	0.270	0.238	0.233	**0.230**
10	12.3	12.2	12.2	21.7	11.9	10.1	10.1	**10.1**
10	8.26	7.67	7.74	19.0	**4.95**	7.67	7.37	7.45
10	5.44	**5.29**	5.30	16.3	6.08	7.94	7.94	7.93
10	9.33	9.36	**9.32**	18.8	9.36	14.8	14.9	15.7
10	18.8	18.2	18.3	18.2	4.14	4.10	**4.09**	4.14
12	**480**	481	484	3130	553	630	649	629

Table 3.17: QMTZ Rank Two Q Time Values

Size	Original	Sym	UT	PSD	NSD	QR	Sym QR	UT QR
8	0.505	0.481	0.493	0.488	0.290	**0.253**	0.272	0.263
10	11.7	13.5	12.3	22.2	**4.47**	6.52	6.54	6.39
10	30.4	29.8	29.8	31.4	12.0	11.7	**11.4**	11.5
10	17.9	18.0	17.8	17.6	**3.47**	3.92	3.80	3.92
10	12.1	11.9	11.8	25.3	**6.61**	10.0	10.1	10.0
10	18.9	19.3	19.8	21.8	**4.58**	8.89	8.60	8.69
12	5580	5340	5410	5370	**304**	1350	1350	1360

3.7.3 Quadratic Single Commodity Flow Formulation

We use the constraints described in Section 2.1.4 and the objective function described in Section 2.3. We use Gurobi's $addVar$ to create the binary and continuous variables, and use $QuadExpr$ for the quadratic objective function.

Table 3.18: QSCF Nonnegative Q Time Values

Size	Original	Sym	UT	PSD	NSD	QR	Sym QR	UT QR
8	1.14	1.10	1.13	1.11	**0.377**	0.398	0.411	0.400
10	45.6	46.5	45.8	41.5	**6.14**	16.8	17.2	16.8
10	48.6	48.5	48.8	41.9	**5.03**	12.2	12.2	12.1
10	49.7	49.5	49.9	43.6	**5.45**	22.1	22.3	22.4
10	42.3	41.8	42.3	41.6	**8.13**	14.6	14.7	14.5
10	43.2	42.5	42.2	42.0	**5.66**	18.3	18.1	18.1
12	-	-	-	-	**363**	2750	2770	2660

Table 3.19: QSCF Balanced Q Time Values

Size	Original	Sym	UT	PSD	NSD	QR	Sym QR	UT QR
8	0.974	0.956	0.972	0.965	**0.380**	0.472	0.467	0.468
10	44.2	44.2	44.2	47.8	**8.38**	18.8	18.9	19.1
10	42.3	43.5	42.3	47.7	**7.24**	22.8	22.4	22.3
10	46.7	46.5	46.6	42.8	**10.0**	22.9	23.0	23.0
10	47.1	47.1	47.2	47.9	**10.7**	18.5	18.7	18.7
10	42.0	42.2	42.2	41.6	**8.89**	16.6	16.7	16.8
12	-	-	-	-	**629**	3350	3290	3290

Table 3.20: QSCF Positively Skewed Q Time Values

Size	Original	Sym	UT	PSD	NSD	QR	Sym QR	UT QR
8	1	1.02	1.00	1	**0.358**	0.439	0.437	0.425
10	44.4	44.2	44.5	49.5	**15.7**	20.8	20.9	20.7
10	42.0	41.4	42.5	49.6	**13.1**	21.6	21.7	21.5
10	44.0	44.1	43.8	48.3	**12.0**	15.3	15.3	15.3
10	47.7	47.4	47.4	42.9	**10.9**	20.9	20.5	20.3
10	45.3	45.5	45.4	48.8	**10.7**	18.0	17.7	18.0
12	-	-	-	-	**922**	5210	4780	5270

Table 3.21: QSCF Negatively Skewed Q Time Values

Size	Original	Sym	UT	PSD	NSD	QR	Sym QR	UT QR
8	1.01	1.01	1.01	1.02	0.960	0.543	**0.541**	**0.541**
10	47.0	47.4	46.9	43.5	47.2	24.4	24.2	**24.2**
10	41.5	41.5	41.2	42.8	45.0	24.2	24.3	**24.0**
10	42.8	43.8	43.1	42.6	47.3	**29.5**	30.2	30.0
10	42.6	42.9	42.9	41.8	47.4	**35.9**	36.8	36.6
10	45.4	45.2	45.1	42.9	41.5	23.4	**23.4**	23.4
12	-	-	-	-	-	**6710**	7310	7200

Table 3.22: QSCF Positive Semi-Definite Q Time Values

Size	Original	Sym	UT	PSD	NSD	QR	Sym QR	UT QR
8	0.983	0.962	0.969	0.946	**0.416**	0.466	0.466	0.469
10	43.5	43.7	44.3	47.9	**13.5**	24.3	24.2	25.0
10	46.3	46.7	46.6	48.8	**11.6**	20.7	20.5	20.4
10	43.7	43.6	43.7	47.9	**11.3**	19.5	19.2	19.6
10	45.3	45.2	45.4	42.3	**11.8**	16.7	16.6	16.5
10	41.3	41.1	41.3	40.7	**14.6**	22.5	22.4	22.3
12	-	-	-	-	**1500**	3730	3710	3670

Table 3.23: QSCF Nonnegative and Positive Semi-Definite Q Time Values

Size	Original	Sym	UT	PSD	NSD	QR	Sym QR	UT QR
8	1.10	1.08	1.07	1.11	0.576	0.422	0.422	**0.417**
10	45.0	44.0	44.3	42.0	**17.8**	20.3	20.1	20.6
10	48.8	48.3	48.8	43.7	**17.8**	20.0	20.1	20.2
10	40.9	40.7	40.8	42.3	**14.6**	18.1	18.6	18.7
10	48.7	47.6	48.4	42.2	**15.8**	16.3	16.5	16.3
10	47.4	47.4	47.2	42.6	17.0	**14.0**	14.1	14.2
12	-	-	-	-	**978**	2560	2300	2280

Table 3.24: QSCF Rank One Q Time Values

Size	Original	Sym	UT	PSD	NSD	QR	Sym QR	UT QR
8	0.382	0.377	0.366	0.691	0.339	**0.316**	0.336	0.321
10	15.6	15.5	15.4	27.2	14.6	13.1	13.2	**13.0**
10	10.2	9.88	9.90	24.0	**6.34**	9.84	10.8	9.89
10	6.99	7.07	**7.04**	22.9	8.04	10.7	10.6	10.5
10	**11.2**	11.2	11.2	24.7	13.1	19.5	19.6	19.5
10	25.2	24.9	25.4	21.9	**4.99**	5.98	5.99	5.80
12	607	603	**596**	4310	720	830	816	817

Table 3.25: QSCF Rank Two Q Time Values

Size	Original	Sym	UT	PSD	NSD	QR	Sym QR	UT QR
8	0.706	0.712	0.712	0.748	0.383	0.376	0.376	**0.374**
10	17.4	17.3	17.3	27.3	**5.86**	10.3	10.2	10.2
10	36.2	36.5	36.4	37.0	15.6	15.0	**15.0**	15.2
10	22.9	22.9	22.8	21.7	**4.38**	5.52	5.44	5.66
10	17.5	17.0	17.0	31.9	**8.44**	14.5	14.5	14.4
10	26.6	26.4	26.4	24.7	**7.34**	10.9	10.6	10.7
12	-	-	-	-	**404**	1920	1960	1850

3.7.4 Choice of M for Enforcing Semidefiniteness

We tested several values of M for modifying Q to be positive semi-definite and negative semi-definite on one problem. We tested the following values of M: 1000, 10,000, $10n(n-1)$ and $\max(R_{ij}) + \max(|q_{ij,ij}|)$ over $ij + 10$, where $R_{ij} = \sum_{kl,kl \neq ij} |q_{ij,kl}|$. The first two values are large positive numbers, while the latter two choices are set to scale with the

size of the input. In the final case we test a number just above a bound that enforces semidefiniteness. We chose one formulation to test these values on: the Dantzig subtour elimination formulation, balanced values of Q. We provide the time data below for five problems of size 10. We exclude smaller instances, as there were many instances with similar times. Instances of size 15 did not solve to optimality, and so we exclude them from this analysis. The best time result is bolded for each instance.

$$
\begin{aligned}
M_1: &\quad 1000 \\
M_2: &\quad 10000 \\
M_3: &\quad 10n(n-1) \\
M_4: &\quad \max\left(\sum_{kl,kl\neq ij} |q_{ij,kl}|\right) + \max\left(|q_{ij,ij}|\right) + 10
\end{aligned}
$$

Table 3.26: Q Dantzig Plus/Minus M Time Values

$Q + M_iI$					$Q - M_iI$				
Size	M_1	M_2	M_3	M_4	Size	M_1	M_2	M_3	M_4
10	**53.386**	55.023	70.362	75.007	10	6.889	**6.611**	9.194	8.002
10	**62.822**	68.184	62.974	64.864	10	8.695	**7.611**	8.751	8.82
10	70.547	73.223	**70.522**	78.239	10	10.319	**8.573**	10.541	11.005
10	72.479	69.634	58.803	**56.657**	10	10.293	9.287	10.458	**8.296**
10	**53.264**	55.99	56.682	55.158	10	7.24	6.972	**6.925**	7.251

This experiment suggests that size of M does not have a large impact on the time results, or solve larger instances of the QTSP, though it does suggest that making the diagonal more negative ($Q - MI$) does present some advantage. We therefore chose $M = 10000$ for the experiments in Sections 3.7.1, 3.7.2, and 3.7.3, which is sufficiently large to make up to size 15 problems positive/negative semidefinite for all instances.

3.7.5 Analysis

We review the results based on size, noting that some problems of size 12 did not solve in the time allotted.

Size 8

The size 8 problems solved quickly for all instances, with occasional tied best times however the negative semidefinite narrowly performed best, followed by the symmetric quadratic reduced, then the general quadratic reduced modifications.

Table 3.27: Size 8 Time Results Summary

Formulation	Original	Sym	UT	PSD	NSD	QR	Sym QR	UT QR
Dantzig	0	0	0	0	4	2	2	1
MTZ	0	0	0	0	2	4	1	1
SCF	0	0	0	0	4	1	1	3
Total	0	0	0	0	10	7	4	5

Size 10

The negative semidefinite modification was the best modification for all formulations and inputs of size 10, with the best time in 85 of 120 experiments. This size did not produce any tied results. Upper triangular quadratic reduced, symmetric quadratic reduced and quadratic reduced also produced best times, but on average were slightly slower than the negative semidefinite modification. The unmodified, symmetric and uppertriangular modifications did worse overall, and positive semidefinite did worst, though the time average was only slightly worse than the original, symmetric and UT modifications. The results are summarized below.

Table 3.28: Size 10 Time Results Summary

Formulation	Original	Sym	UT	PSD	NSD	QR	Sym QR	UT QR
Dantzig	1	0	1	0	27	3	4	4
MTZ	0	1	1	0	28	0	4	6
SCF	1	0	1	0	30	3	2	3
Total	2	1	3	0	85	6	10	13

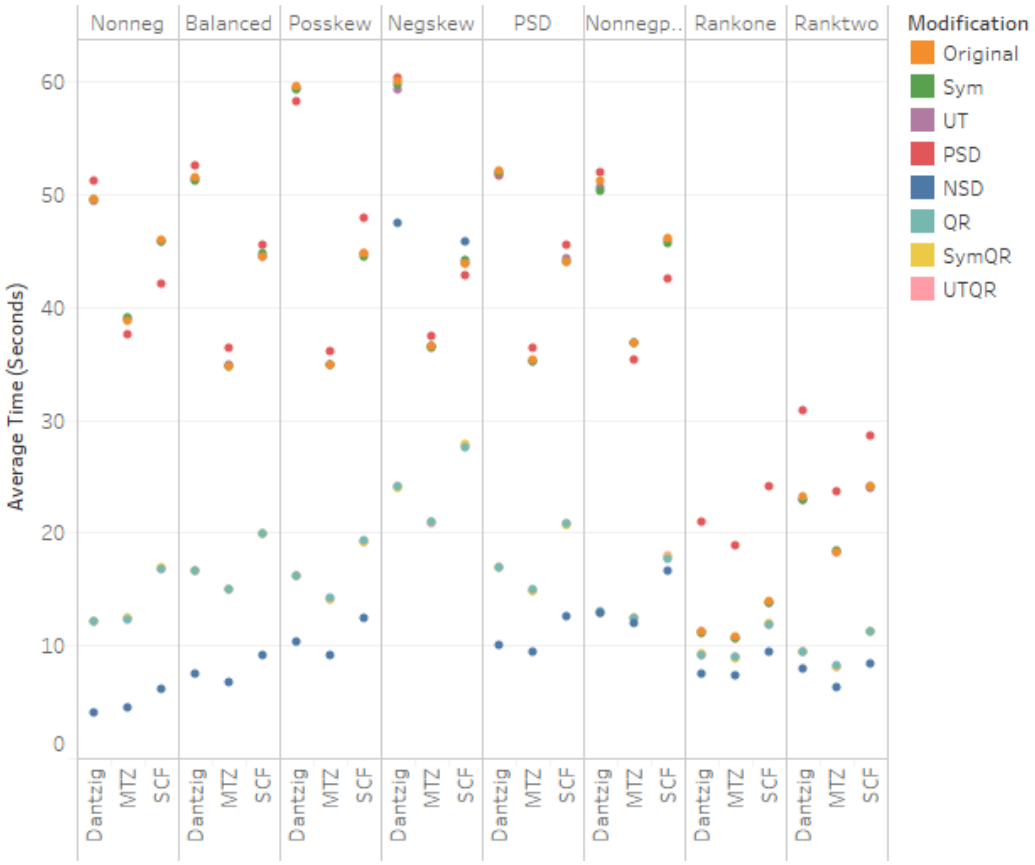

Figure 3.1: Visual summary of the average solve times over all trials for the size 10 problem.

Visually, we can see that the negative semidefinite modification provides the best times for nearly all characteristics of Q, with the exception of the negatively skewed Q. In that instance, the quadratic reduced form performs best, however we see that the NSD modification performs much worse. In the rank one and rank two instances, the worse modifications somewhat catch up to the better performers.

Size 12

The negative semidefinite modification was the best modification for trials of size 12, and in some instances, permitted us to solve larger sizes of problems than others inside the three hour time limit. In the Dantzig Subtour Elimination formulation, no modification solved the negatively skewed input within the three hour time limit. In the same formulation, for two inputs only the negative semidefinite modification solved within three hours.

Table 3.29: Size 12 Time Results Summary

Formulation	Original	Sym	UT	PSD	NSD	QR	Sym QR	UT QR
Dantzig	0	0	0	0	7	0	0	0
MTZ	1	0	0	0	6	0	0	1
SCF	0	0	1	0	6	1	0	0
Total	1	0	1	0	19	1	0	1

We also summarize which problems solved in three hours. We see that the MTZ formulation solved more problems within the three hours than the Dantzig subtour elimination and SCF formulations. The negative semidefinite, and the quadratic reduced variations solved more instances within three hours than the unmodified, symmetric, upper triangular and positive semidefinite modifications.

Table 3.30: Size 12 Solving Summary

Formulation	Original	Sym	UT	PSD	NSD	QR	Sym QR	UT QR
Dantzig	1	1	1	0	7	5	5	5
MTZ	7	6	5	5	8	8	8	8
SCF	1	1	1	1	7	7	8	8
Total	9	8	7	6	22	20	21	21

Conclusion

The experiments indicate that negative semidefinite modification was the most effective among the modifications tested at improving the solve times for the QTSP for the formulations studied. The quadratic reduced method, and its variations also performed well, while leaving the matrix unmodified, and the upper triangular and symmetric modifications did not perform as well. Making the matrix positive semidefinite performed worse overall. We speculate that making the problem convex did not help in finding the points on the feasible polytope, if the global minimizer was not feasible. The largest problem size that solved in three hours was 12, and that was not consistent among the modifications, indicating that modifying the input matrices may permit us to solve larger instances of the QTSP. At the smaller sizes, the result is less apparent, while at the larger sizes, the negative semidefinite modification stands out as the fastest method, which indicates that it's unlikely that running the experiments for longer than three hours would change these results.

Chapter 4

Linearization of the Quadratic Objective Function

We will now detail some methods to linearize the quadratic objective function. These methods add additional variables and constraints, and may greatly increase the problem size. However, we will now be optimizing a linear objective function with linear constraints, resulting in a mixed integer linear program, rather than the nonlinear problem we were trying to solve in the previous chapter. As such, despite the increase in number of constraints and variables, it is possible we will be able to solve larger problems. In the computational experiments for this section, we will compare the time results for the different linearized forms, as well as retaining the original quadratic form for comparison. In effect, we wish to see if the benefits of linearization outweigh the negatives associated with the larger problem size.

We provide formulations for five linearizations. Each linearization can be applied to the TSP constraints (Sections 2.1.1 – 2.1.6) and we replace the quadratic objective function (Section 2.3) with linearized objective functions. We begin by describing the linearization techniques, then we describe the experiments.

4.1 Linearizations

In this section we detail five methods to reformulate the QTSP as a MILP with a linear objective. Each method produces an extended formulation, where additional variables are added to the model. However, in each formulation, projecting back to the variables associated with the original formulation (x_{ij}), gives the original feasible set. Thus, these formulations create an equivalent model using additional variables and constraints that still produces feasible tours with the same objective cost.

4.1.1 MILP Reformulation Using Additional Binary Variables

The TSP restricts x_{ij} to $\{0, 1\}$, which allows us to substitute the product $x_{ij}x_{kl} = y_{ij,kl}$, a new binary variable [HM09].

$$\text{Minimize: } \sum_{ij \in E} \sum_{kl \in E} q_{ij,kl} y_{ij,kl} + \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{4.1}$$

We force $y_{ij,kl} = 1$ if both $x_{ij} = 1$ and $x_{kl} = 1$ and 0 otherwise by including the following constraints [HM09]:

$$x_{ij} + x_{kl} - y_{ij,kl} \leq 1 \qquad \forall\, ij \in E, kl \in E \tag{4.2}$$

$$-x_{ij} - x_{kl} + 2\, y_{ij,kl} \leq 0 \qquad \forall\, ij \in E, kl \in E \tag{4.3}$$

$$y_{ij,kl} \in \{0, 1\} \tag{4.4}$$

This adds $n^2(n-1)^2$ binary variables, and $3n^2(n-1)^2$ constraints to the original problem formulation.

4.1.2 Standard Linearization

We can modify the above formulation by strengthening constraint (4.3) by replacing it with two separate constraints

$$y_{ij,kl} \leq x_{ij} \qquad \forall\, ij \in E, kl \in E \tag{4.5}$$

$$y_{ij,kl} \leq x_{kl} \qquad \forall\, ij \in E, kl \in E \tag{4.6}$$

which allows us to relax the binary restriction on $y_{ij,kl}$ by replacing (4.4) with $y_{ij,kl} \geq 0$ [HM09]. We are left with a stronger form of the previous formulation, which is now purely a linear program. This is known as the *standard linearization*

$$\text{Minimize: } \sum_{ij \in E} \sum_{kl \in E} q_{ij,kl} y_{ij,kl} + \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} \tag{4.7}$$

$$\text{subject to: } x_{ij} + x_{kl} - y_{ij,kl} \leq 1 \qquad \forall\, ij \in E, kl \in E \tag{4.2}$$

$$y_{ij,kl} \leq x_{ij} \qquad \forall\, ij \in E, kl \in E \tag{4.5}$$

$$y_{ij,kl} \leq x_{kl} \qquad \forall\, ij \in E, kl \in E \tag{4.6}$$

$$y_{ij,kl} \geq 0 \tag{4.8}$$

This method adds $n^2(n-1)^2$ continuous variables, and $3n^2(n-1)^2$ constraints to the original formulation.

4.1.3 The McCormick Envelopes

In this linearization, we relax several quadratic terms to a single variable, and bound that variable by its convex and concave envelopes. For details on McCormick Envelopes we refer to [Pan18], [McC76].

We begin by defining a bilinear term that we will later replace with a single variable. We split the products into two parts. We let $\sum_{kl \in E} q_{ij,kl} x_{kl} = v_{ij}$, where v_{ij} is a real variable. This is the first part of the product and we note that we can define bounds, $\underline{v}_{ij} \leq v_{ij} \leq \overline{v}_{ij}$ based on the values of $q_{ij,kl}$. We let $\overline{v}_{ij} = \sum_{kl \in E} \max\{q_{ij,kl}, 0\}$ and $\underline{v}_{ij} = \sum_{kl \in E} \min\{q_{ij,kl}, 0\}$. The second half of the product will then be x_{ij}, with bounds $0 \leq x_{ij} \leq 1$. Thus we can find the McCormick envelopes of the product $x_{ij}v_{ij}$, as we know that the product will also be bounded by the terms' upper and lower bounds.

We then substitute $w_{ij} = x_{ij}v_{ij}$ and find the bounds on w_{ij} by calculating the McCormick envelopes by manipulating the bounds on x_{ij} and v_{ij}:

$$
\begin{aligned}
x_{ij} - 0 &\geq 0 \\
1 - x_{ij} &\geq 0 \\
v_{ij} - \underline{v}_{ij} &\geq 0 \\
\overline{v}_{ij} - v_{ij} &\geq 0
\end{aligned}
$$

Since each difference above is nonnegative, their products will also be nonnegative. We now find the products of multiplying the bounds together to find the bounds on w_{ij}

$$
\begin{aligned}
(x_{ij} - 0)(v_{ij} - \underline{v}_{ij}) &\geq 0 \\
w_{ij} - x_{ij}\underline{v}_{ij} &\geq 0
\end{aligned}
$$

$$
\begin{aligned}
(x_{ij} - 0)(\overline{v}_{ij} - v_{ij}) &\geq 0 \\
x_{ij}\overline{v}_{ij} - w_{ij} &\geq 0
\end{aligned}
$$

$$
\begin{aligned}
(1 - x_{ij})(v_{ij} - \underline{v}_{ij}) &\geq 0 \\
v_{ij} - \underline{v}_{ij} - w_{ij} + x_{ij}\underline{v}_{ij} &\geq 0
\end{aligned}
$$

$$(1 - x_{ij})(\overline{v}_{ij} - v_{ij}) \geq 0$$
$$\overline{v}_{ij} - v_{ij} - x_{ij}\overline{v}_{ij} + w_{ij} \geq 0$$

We can now simplify the above inequalities into:

$$w_{ij} \leq x_{ij}\overline{v}_{ij} \tag{4.9}$$
$$w_{ij} \geq x_{ij}\underline{v}_{ij} \tag{4.10}$$
$$w_{ij} \leq v_{ij} - \underline{v}_{ij}(1 - x_{ij}) \tag{4.11}$$
$$w_{ij} \geq v_{ij} - \overline{v}_{ij}(1 - x_{ij}), \tag{4.12}$$

Recall that $w_{ij} = x_{ij}v_{ij}$, and $v_{ij} = \sum_{kl \in E} q_{ij,kl}x_{kl}$. We can now replace the quadratic term with w_{ij} and include the bounds on w_{ij}.

The linearization is:

$$\text{Minimize: } \sum_{i=1}^{n}\sum_{j=1}^{n}[c_{ij}x_{ij} + w_{ij}] \tag{4.13}$$
$$\text{subject to: } w_{ij} \leq x_{ij}\overline{v}_{ij} \qquad \forall\, ij \in E \tag{4.9}$$
$$w_{ij} \geq x_{ij}\underline{v}_{ij} \qquad \forall\, ij \in E \tag{4.10}$$
$$\sum_{k=1}^{n}\sum_{l=1}^{n} q_{ij,kl}x_{kl} - w_{ij} + x_{ij}\underline{v}_{ij} \geq \underline{v}_{ij} \qquad \forall\, ij \in E \tag{4.11}$$
$$\sum_{k=1}^{n}\sum_{l=1}^{n} q_{ij,kl}x_{kl} - w_{ij} + x_{ij}\overline{v}_{ij} \leq \overline{v}_{ij} \qquad \forall\, ij \in E \tag{4.12}$$

This linearization has $n(n-1)$ additional continuous variables, and $4(n(n-1))$ constraints.

4.1.4 Base-2 Linearization

In this linearization, we take advantage of the integer properties of Q and replace the quadratic term with the differences of integers of base 2. As we are using base 2, the variables associated with each power of 2 will be binary [Wat67]. We follow the procedure outlined in [Pan18].

We begin by letting $\sum_{kl \in E} q_{ij,kl}x_{kl} = v_{ij}$, as we did for the McCormick envelopes. As all entries in Q are integers, v_{ij} is also an integer, bounded by $\underline{v}_{ij} \leq v_{ij} \leq \overline{v}_{ij}$, where $\overline{v}_{ij} = \sum_{kl \in E} \max\{q_{ij,kl}, 0\}$ and $\underline{v}_{ij} = \sum_{kl \in E} \min\{q_{ij,kl}, 0\}$. We can now rewrite the objective

function and associated constraint as:

$$\text{Minimize: } \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} + \sum_{i=1}^{n}\sum_{j=1}^{n} x_{ij}v_{ij}$$
$$\text{subject to: } \sum_{kl\in E} q_{ij,kl}x_{kl} = v_{ij} \quad \forall\, ij \in E$$

We can now represent v_{ij} as the difference of two positive integers $v_{ij} = t^1_{ij} - t^2_{ij}$, where $0 \le t^1_{ij} \le \overline{v}_{ij}$ and $0 \le t^2_{ij} \le |\underline{v}_{ij}|$. We now expand t^1_{ij}, t^2_{ij} in base 2, so we find the upper bound for each expansion as:

$$p^1_{ij} = \begin{cases} \lfloor \log_2(\overline{v}_{ij}) \rfloor + 1 & \text{if } \overline{v}_{ij} > 0, \\ 0 & \text{otherwise} \end{cases}$$
$$p^2_{ij} = \begin{cases} \lfloor \log_2(|\underline{v}_{ij}|) \rfloor + 1 & \text{if } \underline{v}_{ij} < 0, \\ 0 & \text{otherwise} \end{cases}$$

Therefore, $v_{ij} = t^1_{ij} - t^2_{ij}$ can be represented as:

$$v_{ij} = \sum_{g=1}^{p^1_{ij}} 2^{g-1} t^1_{ij,g} - \sum_{g=1}^{p^2_{ij}} 2^{g-1} t^2_{ij,g}$$

where t^1_{ij}, t^2_{ij} are binary variables.

We can now return to the objective function, and replace v_{ij} with the difference found above.

$$\text{Minimize: } \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} + \sum_{ij\in E} x_{ij} \sum_{g=1}^{p^1_{ij}} 2^{g-1} t^1_{ij,g} - \sum_{ij\in E} x_{ij} \sum_{g=1}^{p^2_{ij}} 2^{g-1} t^2_{ij,g}$$

The products $x_{ij}t^1_{ij,g}$ and $x_{ij}t^2_{ij,g}$ are the products of two binary variables, so we can linearize them using the standard linearization. We add new variables $y^1_{ij,g} = x_{ij}t^1_{ij,g}$ and $y^2_{ij,g} = x_{ij}t^2_{ij,g}$. We now replace those terms in the objective function, and add the standard linearization constraints for each product to get the full linearization:

$$\text{Minimize: } \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} + \sum_{ij\in E} \sum_{g=1}^{p^1_{ij}} 2^{g-1} y^1_{ij,g} - \sum_{ij\in E} \sum_{g=1}^{p^2_{ij}} 2^{g-1} y^2_{ij,g} \tag{4.14}$$

$$\sum_{kl \in E} q_{ij,kl} x_{kl} = \sum_{g=1}^{p_{ij}^1} 2^{g-1} t_{ij,g}^1 - \sum_{g=1}^{p_{ij}^2} 2^{g-1} t_{ij,g}^2 \quad \forall\, ij \in E \tag{4.15}$$

$$x_{ij} + t_{ij,g}^1 - y_{ij,g}^1 \leq 1 \quad \forall\, ij \in E, g = 1, \ldots, p_{ij}^1 \tag{4.16}$$

$$y_{ij,g}^1 \leq x_{ij} \quad \forall\, ij \in E, g = 1, \ldots, p_{ij}^1 \tag{4.17}$$

$$y_{ij,g}^1 \leq t_{ij,g}^1 \quad \forall\, ij \in E, g = 1, \ldots, p_{ij}^1 \tag{4.18}$$

$$x_{ij} + t_{ij,g}^2 - y_{ij,g}^2 \leq 1 \quad \forall\, ij \in E, g = 1, \ldots, p_{ij}^2 \tag{4.19}$$

$$y_{ij,g}^2 \leq x_{ij} \quad \forall\, ij \in E, g = 1, \ldots, p_{ij}^2 \tag{4.20}$$

$$y_{ij,g}^2 \leq t_{ij,g}^2 \quad \forall\, ij \in E, g = 1, \ldots, p_{ij}^2 \tag{4.21}$$

$$y_{ij,g}^1, y_{ij,g}^2, t_{ij,g}^1, t_{ij,g}^2 \in \{0, 1\} \quad \forall\, ij, g \tag{4.22}$$

where (4.16), (4.17), (4.18) are the standard linearization constraints associated with $y_{ij,g}^1 = x_{ij} t_{ij,g}^1$, and (4.19), (4.20), (4.21) are the standard linearization constraints associated with $y_{ij,g}^2 = x_{ij} t_{ij,g}^2$.

This linearization adds $2 \sum_{ij \in E} (p_{ij}^1 + p_{ij}^2)$ binary variables and $5 \sum_{ij \in E} (p_{ij}^1 + p_{ij}^2) + n^2$ additional constraints.

4.1.5 Base-10 Linearization

In this linearization, we replace the base 2 from the previous linearization with a base 10. We follow the procedure outlined in [Pan18]. We begin by following the same process as shown in the base 2 formulation, however, we use base 10.

We let $\sum_{kl \in E} q_{ij,kl} x_{kl} = v_{ij}$, and $v_{ij} = t_{ij}^1 - t_{ij}^2$. The objective function becomes:

$$\text{Minimize: } \sum_{i=1}^{n} \sum_{j=1}^{n} c_{ij} x_{ij} + \sum_{i=1}^{n} \sum_{j=1}^{n} x_{ij} v_{ij}$$

$$\text{subject to: } \sum_{kl \in E} q_{ij,kl} x_{kl} = v_{ij} \quad \forall\, ij \in E$$

We find the bounds for the expansion:

$$p_{ij}^1 = \begin{cases} \lfloor \log_{10}(\overline{v}_{ij}) \rfloor + 1 & \text{if } \overline{v}_{ij} > 0, \\ 0 & \text{otherwise} \end{cases}$$

$$p_{ij}^2 = \begin{cases} \lfloor \log_{10}(|\underline{v}_{ij}|) \rfloor + 1 & \text{if } \underline{v}_{ij} < 0, \\ 0 & \text{otherwise} \end{cases}$$

Therefore, $v_{ij} = t^1_{ij} - t^2_{ij}$ can be represented as:

$$v_{ij} = \sum_{g=1}^{p^1_{ij}} 10^{g-1} t^1_{ij,g} - \sum_{g=1}^{p^2_{ij}} 10^{g-1} t^2_{ij,g}$$

where t^1_{ij}, t^2_{ij} are integer variables in $\{0, 1, \ldots, 9\}$.

We now adjust the objective function:

$$\text{Minimize: } \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} + \sum_{ij\in E} x_{ij} \sum_{g=1}^{p^1_{ij}} 10^{g-1} t^1_{ij,g} - \sum_{ij\in E} x_{ij} \sum_{g=1}^{p^2_{ij}} 10^{g-1} t^2_{ij,g}$$

We will now linearize the products $x_{ij}t^1_{ij,g}$ and $x_{ij}t^2_{ij,g}$, and since $t^1_{ij,g}, t^2_{ij,g}$ are integer variables, we will use the McCormick envelopes to linearize. We let $r^1_{ij,g} = x_{ij}t^1_{ij,g}$ and $r^2_{ij,g} = x_{ij}t^2_{ij,g}$.

We now calculate the McCormick envelopes. We provide detailed steps for $t^1_{ij,g}$, however they are identical for $t^2_{ij,g}$. This method is also detailed above. Recall that $0 \le x_{ij} \le 1$ and $0 \le t^1_{ij,g} \le 9$. Therefore:

$$\begin{aligned}
x_{ij} - 0 &\ge 0 \\
1 - x_{ij} &\ge 0 \\
t^1_{ij,g} - 0 &\ge 0 \\
9 - t^1_{ij,g} &\ge 0
\end{aligned}$$

We can now find the bounds on $r^1_{ij,g}$ by finding the products of the bounds on x_{ij} and $t^1_{ij,g}$:

$$\begin{aligned}
(x_{ij} - 0)(t^1_{ij,g} - 0^1) &\ge 0 \\
r^1_{ij,g} &\ge 0
\end{aligned}$$

$$\begin{aligned}
(x_{ij} - 0)(9 - t^1_{ij,g}) &\ge 0 \\
9x_{ij} - r^1_{ij,g} &\ge 0
\end{aligned}$$

$$\begin{aligned}
(1 - x_{ij})(t^1_{ij,g} - 0^1) &\ge 0 \\
t^1_{ij,g} - r^1_{ij,g} &\ge 0
\end{aligned}$$

$$(1 - x_{ij})(9 - t^1_{ij,g}) \geq 0$$
$$9 - t^1_{ij,g} - 9x_{ij} + r^1_{ij,g} \geq 0$$

We can now simplify the above inequalities into:

$$r^1_{ij,g} \geq 0$$
$$r^1_{ij,g} \leq 9x_{ij}$$
$$r^1_{ij,g} \leq t^1_{ij,g}$$
$$t^1_{ij,g} + 9x_{ij} - r^1_{ij,g} \leq 9$$

We are now able to complete the linearization.

$$\text{Minimize: } \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} + \sum_{ij\in E}\sum_{g=1}^{p^2_{ij}} 10^{g-1} r^1_{ij,g} - \sum_{ij\in E}\sum_{g=1}^{p^2_{ij}} 10^{g-1} r^2_{ij,g} \tag{4.23}$$

$$\sum_{kl\in E} q_{ij,kl}x_{kl} = \sum_{g=1}^{p^1_{ij}} 10^{g-1} t^1_{ij,g} - \sum_{g=1}^{p^1_{ij}} 10^{g-1} t^2_{ij,g} \quad \forall\, ij \in E \tag{4.24}$$

$$9x_{ij} + t^1_{ij,g} - r^1_{ij,g} \leq 9 \quad \forall\, ij \in E, g = 1, \ldots, p^1_{ij} \tag{4.25}$$

$$r^1_{ij,g} \leq 9x_{ij} \quad \forall\, ij \in E, g = 1, \ldots, p^1_{ij} \tag{4.26}$$

$$r^1_{ij,g} \leq t^1_{ij,g} \quad \forall\, ij \in E, g = 1, \ldots, p^1_{ij} \tag{4.27}$$

$$r^2_{ij,g} \leq 9x_{ij} \quad \forall\, ij \in E, g = 1, \ldots, p^2_{ij} \tag{4.28}$$

$$r^2_{ij,g} \leq t^2_{ij,g} \quad \forall\, ij \in E, g = 1, \ldots, p^2_{ij} \tag{4.29}$$

$$9x_{ij} + t^2_{ij,g} - r^2_{ij,g} \leq 9 \quad \forall\, ij \in E, g = 1, \ldots, p^2_{ij} \tag{4.30}$$

$$r^1_{ij,g}, r^2_{ij,g}, t^1_{ij,g}, t^2_{ij,g} \in \{0, 1, \ldots, 9\} \quad \forall\, ij, g \tag{4.31}$$

This linearization adds $2\sum_{ij\in E}(p^1_{ij} + p^2_{ij})$ integer variables and $5\sum_{ij\in E}(p^1_{ij} + p^2_{ij}) + n^2$ additional constraints.

4.2 Formulations

We will apply the linearizations to three TSP formulations: the Dantzig Subtour Elimination, MTZ and SCF formulations. For each formulation, we add the variables and constraints associated with each linearization, and replace the quadratic objective function with the objective achieved by linearizing the quadratic term.

Table 4.1: Summary of Linearized Formulations

	Dantzig	MTZ	SCF
Binary	Constraints: 2.1.1, 4.1.1 Objective: 4.1	Constraints: 2.1.2, 4.1.1 Objective: 4.1	Constraints: 2.1.4, 4.1.1 Objective: 4.1
Standard	Constraints: 2.1.1, 4.1.2 Objective: 4.7	Constraints: 2.1.2, 4.1.2 Objective: 4.7	Constraints: 2.1.4, 4.1.2 Objective: 4.7
McCormick	Constraints: 2.1.1, 4.1.3 Objective: 4.13	Constraints: 2.1.2, 4.1.3 Objective: 4.13	Constraints: 2.1.4, 4.1.3 Objective: 4.13
Base 2	Constraints: 2.1.1, 4.1.4 Objective: 4.14	Constraints: 2.1.2, 4.1.4 Objective: 4.14	Constraints: 2.1.4, 4.1.4 Objective: 4.14
Base 10	Constraints: 2.1.1, 4.1.5 Objective: 4.23	Constraints: 2.1.2, 4.1.5 Objective: 4.23	Constraints: 2.1.4, 4.1.5 Objective: 4.23

4.3 Computational Experiments

In these sets of experiments, we address the following questions:

1. Which linearization technique is most efficient for solving QTSP? Is linearization using one of these techniques more efficient than leaving the QTSP in its quadratic form?

2. Do the different formulations of QTSP have different best linearization techniques?

To asses the impact of the linearization techniques, we repeat the experiments from the previous section, however we use the unmodified quadratic matrix as the input. We use the same computer setup as described in Section 3.6. We turn off presolve for all linearizations. We include the quadratic form as a comparison.

The models were coded using GurobiPy, and can be located at the following url:
https://github.com/mrosespencer/QTSP

For each formulation, we provide a main experiment file which draws on the quadratic model from the previous section, as well as the linearized forms, found in a subfolder of the github repository. All cost files are the same as the previous section and can be found on the site linked above.

4.4 Linearized QTSP Formulations Results

We report time in seconds for each of the five linearizations, as well as the quadratic form. Where time exceeds the three hour time limit, we indicate with "-". The best overall time for each size is indicated with **bold** font, while the best linearization (if different) is

indicated with *italicized* font. We report average time for 100 trials for size 5, and time for 5 trials of size 10 and one of size 12.

4.4.1 Linearized Dantzig Formulation Results

Table 4.2: Linearized Dantzig Nonnegative Q Time Values

Size	Quadratic	Binary	Classic	McCormick	Base 2	Base 10
8	**0.714**	22.4	11.0	*3.32*	21.9	5.14
10	**48.0**	1770	1550	*167*	702	741
10	**58.2**	1940	1350	*176*	648	221
10	**56.2**	1920	1730	*163*	644	395
10	**52.8**	1920	1530	275	694	*234*
10	**57.7**	1580	1600	*191*	658	237
12	-	-	-	-	-	-

Table 4.3: Linearized Dantzig Balanced Q Time Values

Size	Quadratic	Binary	Classic	McCormick	Base 2	Base 10
8	**0.702**	13.6	12.3	*3.58*	15.6	11.0
10	**46.8**	505	1040	*250*	2090	2250
10	**52.1**	519	954	*224*	2460	643
10	**49.6**	513	1130	*280*	2470	2080
10	**58.7**	559	1130	*240*	2230	660
10	**55.0**	650	938	*232*	2470	1310
12	-	-	-	-	-	-

Table 4.4: Linearized Dantzig Positively Skewed Q Time Values

Size	Quadratic	Binary	Classic	McCormick	Base 2	Base 10
8	**0.755**	8.42	8.90	*2.86*	14.4	9.28
10	**56.8**	306	574	*129*	4130	1300
10	**63.0**	300	624	*149*	1090	1920
10	**63.7**	258	723	*131*	1220	2510
10	**55.3**	243	673	*104*	1440	1150
10	**55.9**	298	560	*153*	3890	1020
12	-	-	-	-	-	-

Table 4.5: Linearized Dantzig Negatively Skewed Q Time Values

Size	Quadratic	Binary	Classic	McCormick	Base 2	Base 10
8	**0.815**	25.8	18.0	*3.83*	23.1	15.3
10	**57.9**	1410	1380	*404*	2290	1670
10	**65.5**	1560	2190	*270*	2090	780
10	**62.9**	3100	1680	*293*	2290	1180
10	**67.9**	1660	1490	*375*	2530	1800
10	**70.2**	1490	1860	*319*	2910	1130
12	-	-	-	-	-	-

Table 4.6: Linearized Dantzig Positive Semidefinite Q Time Values

Size	Quadratic	Binary	Classic	McCormick	Base 2	Base 10
8	**0.734**	17.3	9.38	*1.84*	15.7	10.2
10	**53.7**	752	705	*175*	3000	567
10	**59.8**	697	553	*158*	4290	1870
10	**51.2**	681	564	*137*	6560	605
10	**58.8**	667	550	*171*	1850	2250
10	**51.7**	747	656	*171*	1480	2550
12	-	-	-	-	-	-

Table 4.7: Linearized Dantzig Nonnegative and Positive Semidefinite Q Time Values

Size	Quadratic	Binary	Classic	McCormick	Base 2	Base 10
8	**0.766**	27.2	9.92	3.62	10.9	*3.28*
10	**52.4**	1910	1790	*199*	1960	453
10	**60.9**	1650	1620	*510*	1090	2020
10	**60.6**	1970	1290	*479*	923	1770
10	**56.3**	1550	1460	*173*	1820	1770
10	**59.9**	1610	1430	*528*	739	1950
12	-	-	-	-	-	-

Table 4.8: Linearized Dantzig Rank One Q Time Values

Size	Quadratic	Binary	Classic	McCormick	Base 2	Base 10
8	**0.230**	8.93	6.42	*1.17*	13.1	8.52
10	**12.0**	127	138	*36.9*	1060	355
10	**8.58**	163	146	*19.3*	802	425
10	**6.36**	105	114	*23.1*	1260	423
10	**9.49**	124	220	*26.0*	826	274
10	**20.0**	124	115	*27.7*	1310	646
12	**1060**	2790	4680	*2110*	-	-

Table 4.9: Linearized Dantzig Rank Two Q Time Values

Size	Quadratic	Binary	Classic	McCormick	Base 2	Base 10
8	**0.449**	9.37	8.71	*2.58*	18.8	10.0
10	**12.0**	148	265	*80.5*	2690	621
10	**40.8**	372	565	*122*	1550	674
10	**22.0**	106	349	*35.7*	1930	490
10	**13.1**	174	578	*114*	2120	708
10	**23.5**	247	172	*57.4*	1860	472
12	-	***6860***	-	-	-	-

4.4.2 Linearized MTZ Formulation Results

Table 4.10: Linearized MTZ Nonnegative Q Time Values

Size	Quadratic	Binary	Classic	McCormick	Base 2	Base 10
8	**0.645**	14.2	7.79	*1.59*	6.68	2.98
10	**38.1**	2060	1530	*329*	999	378
10	**40.6**	1770	1350	*271*	911	388
10	**42.6**	2370	1550	*257*	585	519
10	**37.5**	1590	1560	*320*	878	466
10	**36.5**	1590	1430	*274*	915	427
12	-	-	-	-	-	-

Table 4.11: Linearized MTZ Balanced Q Time Values

Size	Quadratic	Binary	Classic	McCormick	Base 2	Base 10
8	**0.557**	13.0	11.4	*3.15*	11.9	6.05
10	**33.1**	496	880	*303*	3750	522
10	**36.8**	458	715	*423*	5770	888
10	**38.9**	571	787	*334*	4090	495
10	**32.6**	585	790	*383*	10 300	510
10	**36.9**	553	771	*378*	5160	490
12	-	-	-	-	-	-

Table 4.12: Linearized MTZ Positively Skewed Q Time Values

Size	Quadratic	Binary	Classic	McCormick	Base 2	Base 10
8	**0.526**	8.29	8.01	*1.88*	10.4	4.90
10	**34.9**	287	509	*237*	1280	1160
10	**35.0**	333	530	*210*	1260	1140
10	**34.8**	257	451	*104*	1640	978
10	**35.7**	357	493	*229*	805	1180
10	**43.2**	288	468	*223*	1600	1110
12	-	-	-	-	-	-

Table 4.13: Linearized MTZ Negatively Skewed Q Time Values

Size	Quadratic	Binary	Classic	McCormick	Base 2	Base 10
8	**0.623**	24.2	16.7	*3.27*	15.1	14.2
10	**37.9**	1430	1400	*576*	3450	1780
10	**38.3**	3250	1490	*492*	2540	1350
10	**36.2**	1520	1390	*406*	3520	1650
10	**35.7**	3530	1450	*461*	4460	1700
10	**37.6**	3460	1380	*492*	8190	1580
12	-	-	-	-	-	-

Table 4.14: Linearized MTZ Positive Semidefinite Q Time Values

Size	Quadratic	Binary	Classic	McCormick	Base 2	Base 10
8	**0.547**	13.2	8.92	*1.51*	14.8	8.58
10	**34.7**	832	536	*148*	2720	417
10	**38.6**	784	584	*135*	1410	513
10	**35.6**	605	568	*115*	2430	511
10	**36.2**	732	617	*132*	3600	478
10	**35.9**	584	519	*92.5*	2120	478
12	-	-	-	-	-	-

Table 4.15: Linearized MTZ Nonnegative and Positive Semidefinite Q Time Values

Size	Quadratic	Binary	Classic	McCormick	Base 2	Base 10
8	**0.651**	13.0	7.90	*1.73*	6.18	2.73
10	**38.4**	2530	1620	344	2090	*288*
10	**41.0**	1960	1510	*349*	1490	477
10	**33.0**	2000	1500	*351*	1240	395
10	**37.9**	2120	1650	*338*	1720	466
10	**39.2**	1540	1500	355	1040	*300*
12	-	-	-	-	-	-

Table 4.16: Linearized MTZ Rank One Q Time Values

Size	Quadratic	Binary	Classic	McCormick	Base 2	Base 10
8	**0.249**	8.56	6.11	*1.03*	11.7	3.84
10	**12.5**	135	141	*49.7*	2070	575
10	**7.88**	123	94.1	*19.4*	771	496
10	**5.72**	78.4	92.0	*20.3*	2600	558
10	**9.68**	92.5	114	*37.3*	813	182
10	**19.0**	116	101	*37.5*	1960	659
12	**582**	3650	4770	*2440*	-	-

Table 4.17: Linearized MTZ Rank Two Q Time Values

Size	Quadratic	Binary	Classic	McCormick	Base 2	Base 10
8	**0.455**	9.34	8.79	*1.65*	14.8	6.42
10	**11.3**	175	192	*88.5*	5140	1390
10	**31.2**	224	450	*140*	4610	489
10	**18.5**	125	137	*32.2*	2460	435
10	**12.7**	167	409	*104*	1550	473
10	**19.5**	97.6	161	*35.8*	1630	451
12	**8160**	*9390*	-	-	-	-

4.4.3 Linearized SCF Formulation Results

Table 4.18: Linearized SCF Nonnegative Q Time Values

Size	Quadratic	Binary	Classic	McCormick	Base 2	Base 10
8	**1.11**	14.0	9.47	*2.46*	6.32	2.64
10	**46.2**	2120	1670	421	665	*396*
10	**48.8**	2010	1560	*379*	983	527
10	**50.2**	2110	1470	*427*	545	491
10	**42.7**	2060	1530	*410*	732	457
10	**42.5**	2130	1430	386	1140	*328*
12	-	-	-	-	-	-

Table 4.19: Linearized SCF Balanced Q Time Values

Size	Quadratic	Binary	Classic	McCormick	Base 2	Base 10
8	**0.937**	13.5	12.7	*2.54*	12.3	6.60
10	**44.2**	558	939	*271*	1240	457
10	**42.9**	641	764	*265*	4520	510
10	**46.7**	647	1030	*162*	4900	491
10	**48.6**	785	1040	*327*	4160	396
10	**42.5**	574	853	*317*	3710	408
12	-	-	-	-	-	-

Table 4.20: Linearized SCF Positively Skewed Q Time Values

Size	Quadratic	Binary	Classic	McCormick	Base 2	Base 10
8	**0.993**	8.82	9.14	*2.11*	11.1	4.52
10	**44.8**	360	554	*105*	1360	1160
10	**42.1**	396	621	*254*	957	1170
10	**44.5**	274	659	*186*	617	1070
10	**48.0**	335	566	*173*	810	1100
10	**46.0**	307	492	*118*	1660	1060
12	-	-	-	-	-	-

Table 4.21: Linearized SCF Negatively Skewed Q Time Values

Size	Quadratic	Binary	Classic	McCormick	Base 2	Base 10
8	**0.989**	25.1	18.8	*3.93*	13.7	14.4
10	**47.0**	2900	1450	*370*	2220	1380
10	**42.3**	3120	1550	*469*	3770	1240
10	**43.7**	3710	1480	*441*	3220	1410
10	**43.5**	3920	1850	*400*	2720	1520
10	**45.5**	3770	1820	*430*	4990	1350
12	-	-	-	-	-	-

Table 4.22: Linearized SCF Positive Semidefinite Q Time Values

Size	Quadratic	Binary	Classic	McCormick	Base 2	Base 10
8	**0.947**	14.1	10.1	*1.80*	14.2	9.36
10	**46.6**	898	549	*123*	1510	417
10	**48.7**	792	626	*128*	1300	455
10	**45.2**	829	440	*107*	2520	450
10	**46.4**	886	643	*134*	1750	537
10	**42.7**	655	538	*124*	874	485
12	-	-	-	-	-	-

Table 4.23: Linearized SCF Nonnegative and Positive Semidefinite Q Time Values

Size	Quadratic	Binary	Classic	McCormick	Base 2	Base 10
8	**1.08**	13.2	9.35	*1.75*	6.09	2.72
10	**46.9**	1950	1680	482	1320	*401*
10	**50.5**	2650	1630	514	980	*282*
10	**45.3**	2020	1260	*233*	1150	422
10	**50.1**	1940	1450	410	1770	*284*
10	**50.1**	1600	1620	497	1590	*323*
12	-	-	-	-	-	-

Table 4.24: Linearized SCF Rank One Q Time Values

Size	Quadratic	Binary	Classic	McCormick	Base 2	Base 10
8	**0.346**	8.76	6.64	*1.17*	11.5	3.70
10	**16.0**	147	144	*41.4*	1100	578
10	**10.3**	137	106	*21.4*	1110	479
10	**7.48**	103	109	*24.1*	3240	587
10	**11.6**	106	111	*27.4*	844	410
10	**26.2**	110	157	*26.6*	1300	831
12	**702**	4730	6840	*2150*	-	-

Table 4.25: Linearized SCF Rank Two Q Time Values

Size	Quadratic	Binary	Classic	McCormick	Base 2	Base 10
8	**0.679**	9.59	9.67	*1.92*	15.9	6.35
10	**18.0**	148	207	*79.6*	4220	1370
10	**38.0**	230	497	*113*	2730	446
10	**24.2**	95.7	150	*58.9*	2440	849
10	**17.2**	182	533	*87.3*	1060	374
10	**28.4**	102	200	*54.0*	3570	445
12	-	***9490***	-	-	-	-

4.4.4 Analysis

The McCormick linearization was the fastest linearization, though the quadratic (non-linearized) formulation performed best overall. Problems of size 15 did not solve optimally, so are not included in this analysis. This would indicate that of the five linearizations

studied, none provide an advantage over using Gurobi's quadratic solver. We present the linearization with the best solve time in the tables below.

Table 4.26: Size 8 Best Time Linearized Results Summary

Formulation	Binary	Classic	McCormick	Base 2	Base 10
Dantzig	0	0	7	0	1
MTZ	0	0	8	0	0
SCF	0	0	8	0	0
Total	0	0	23	0	1

Table 4.27: Size 10 Best Time Linearized Results Summary

Formulation	Binary	Classic	McCormick	Base 2	Base 10
Dantzig	0	0	40	0	0
MTZ	0	0	38	0	2
SCF	0	0	34	0	6
Total	0	0	112	0	8

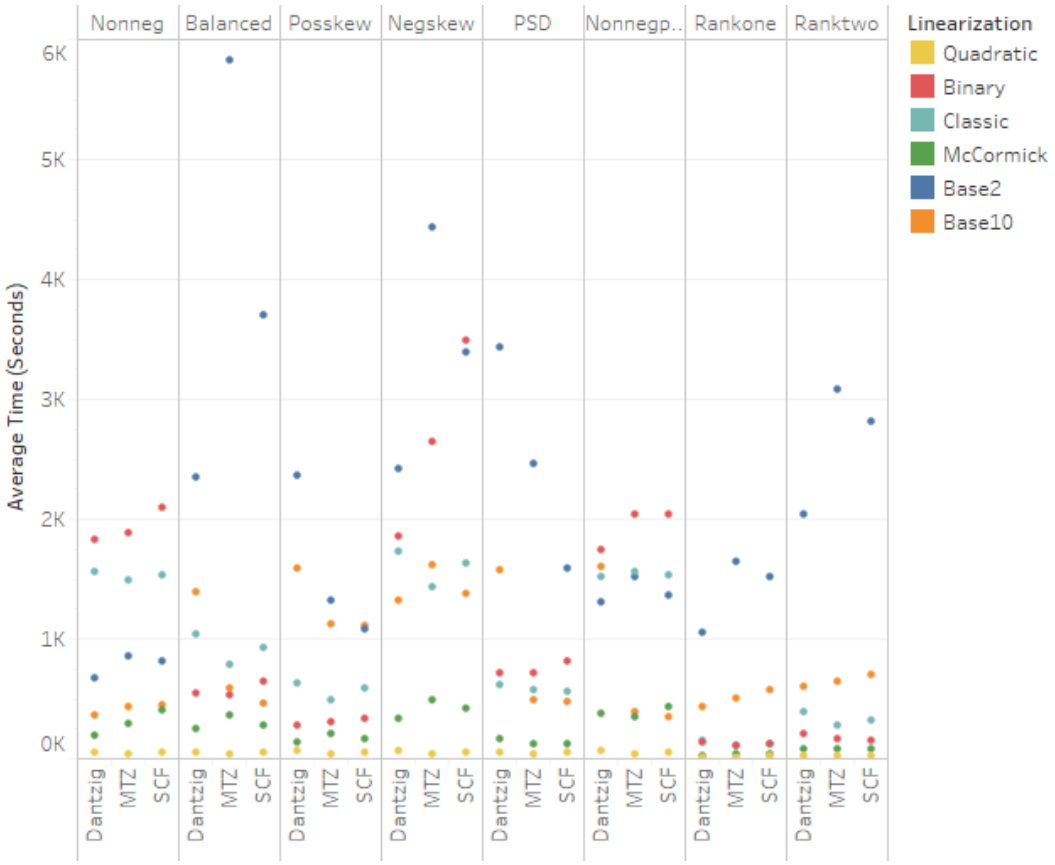

Figure 4.1: Visual summary of the average solve times over all trials for the size 10 problem.

Very few instances of size 12 solve within three hours. We summarize the number of instances solved in the table below. There are a few instances where the binary replacement method does solve, while the quadratic formulation does not, these occur in the rank two classes, which are somewhat more sparse. In general it appears that these linearization methods are not a good way to solve larger instances of this quadratic TSP.

Table 4.28: Size 12 Instances Solved Linearized Results Summary

Formulation	Quadratic	Binary	Classic	McCormick	Base 2	Base 10
Dantzig	1	2	1	1	0	0
MTZ	2	2	1	1	0	0
SCF	1	2	1	1	0	0
Total	4	6	3	3	0	0

All linearizations studied required adding variables and constraints to the original formulation. These additional variables and constraints increase the overall size of the problem, but change the objective function to a linear version. We theorize that these additional vari-

ables and constraints do not improve the tightness of the model, and in fact relax it. We tested this theory by reviewing the linear relaxation of the MTZ TSP formulation. The relationship between the MTZ, SCF and Dantzig Subtour Elimination models are known (see Section 2.1.7), so we focus on the relationship between the linear relaxation forms of the linearized problems.

In the linear relaxation of the models, we replace the integer variables with continuous variables with the same bounds. We present the linear relaxation objective values for the size 12 MTZ model.

Table 4.29: Linear Relaxation Size 12 MTZ Objective Values

Characteristic	IP Objective	Binary	Classic	McCormick	Base 2	Base 10
Balanced	−203	−2385	−2154	−2452	−3060	−11 990
Negskew	−661	−5879	−5460	−5872	−7000	−11 990
Nonneg	976	0	0	0	0	0
Nonnegpsd	1957	0	0	0	0	0
Posskew	70	−1715	−1509	−1833	−3057	−11 990
PSD	−401	−3217	−2864	−3213	−5434	−11 990
Rank1	−2842	−16 040	−11 100	−18 250	−24 560	−120 000
Rank2	−3007	−21 660	−16 390	−24 500	−38 550	−120 000

Table 4.30: Additional variables and constraints size summary

Formulation	Additional Variables	Additional Constraints
Binary	$n^2(n-1)^2$ binary	$3n^2(n-1)^2$
Classic	$n^2(n-1)^2$ continuous	$3n^2(n-1)^2$
McCormick	$n(n-1)$ continuous	$4n(n-1)$
Base 2	$2\sum_{ij\in E}(p_{ij}^1 + p_{ij}^2)$ binary	$5\sum_{ij\in E}(p_{ij}^1 + p_{ij}^2) + n^2$
Base 10	$2\sum_{ij\in E}(p_{ij}^1 + p_{ij}^2)$ integer	$5\sum_{ij\in E}(p_{ij}^1 + p_{ij}^2) + n^2$

These models solved very quickly, and the LP optimal solutions suggest that the linearizations provide poor LP relaxations to the quadratic model. The relative speed of the models appears to depend on the overall size of the linearization and how well the additional constraints describe the original problem. The McCormick envelopes are the fastest linearization, in terms of solve time, and is quite small in terms of additional variables and constraints, but the classic linearization is the tightest.

In conclusion, the quadratic (unmodified) model was faster than the linearized models tested. This speaks to the strength of the nonlinear solver within Gurobi, though we also see that the binary replacement method shows some indications that it could be better than

the McCormick method at the larger sizes. It is possible that larger instances would see a different pattern than the smaller instances, however, at this time, the computational time requirements are too high to complete larger size experiments.

Chapter 5

Conclusion

In this thesis, we consider formulating the Quadratic Traveling Salesman Problem for computational experiments. In particular, we are interested in reformulating the quadratic cost matrix and quadratic objective function. We begin with some research on the history of computer solvers for integer programming problems, especially in relation to the Traveling Salesman Problem. We then review IP formulations for the TSP, as well as provide a variation on the objective function to make the problem quadratic. We then run two sets of experiments on the QTSP, where we modify the quadratic cost matrix, or modify the quadratic objective function, to observe the impact that these modifications have on solve time or size of problems solved to optimality.

We found that modifying the the quadratic matrix to make it negative semidefinite by subtracting a large value from the diagonal values was the most successful reformulating strategy, and we are able to solve more problems using this method. Other methods, such as replacing the quadratic costs associated with the nth node with zeros, and its variations, were also effective at reducing solve times and solving larger problem sizes. At this point we don't have a good explanation for why the NSD formulation works so much better than the other methods, but in practice this formulation is faster and produces legitimate tours. Open problems include determining why the NSD modification was worse for negatively skewed random instances compared to other classes of random instances. The NSD formulation acts as expected and makes the eigenvalues large and negative for this class, however it does not have any advantage over the quadratic reduced method for this particular class.

We examined five relaxations of the quadratic objective function to a linear function. These methods did not provide an advantage over leaving the model in its quadratic form and using the quadratic solver. We suspect that the reasons for this are that the linearizations provided poor relaxations and greatly expanded the size of the problem. Of the methods tested, the McCormick envelopes performed best. Open problems in this area includes developing new methods of linearization for the QTSP that provide better relaxations.

Bibliography

[ABCC07] David L. Applegate, Robert Bixby, William Cook, and Vašek Chvátal, *The Traveling Salesman Problem A Computational Study*, Princeton Series in Applied Mathematics, Princeton University Press, Princeton, February 2007.

[ACK+00] A. Aggarwal, D. Coppersmith, S. Khanna, R. Motwani, and B. Schieber, *The Angular-Metric Traveling Salesman Problem*, SIAM Journal on Computing **29** (2000), no. 3, 697–711.

[BFG+00] Robert E. Bixby, Mary Fenelon, Zonghao Gu, Ed Rothberg, and Roland Wunderling, *Mip: Theory and practice – closing the gap*, System Modelling and Optimization (M. J. D. Powell and S. Scholtes, eds.), IFIP – The International Federation for Information Processing, Springer US, 2000, pp. 19–49.

[Bix02] Robert E. Bixby, *Solving Real-World Linear Programs: A Decade and More of Progress*, Operations Research **50** (2002), no. 1, 3–15.

[Bix12] Robert Bixby, *A Brief History of Linear and Mixed-Integer Programming Computation*, Documenta Mathematica (2012), 107–121.

[Bri] Keith Briggs, *Diagonally Dominant Matrix*, http://mathworld.wolfram.com/DiagonallyDominantMatrix.html.

[BT97] Dimitris Bertsimas and John N. Tsitskilis, *Introduction to linear optimization*, Athena Scientific series in optimization and neural computation ; 6, Athena Scientific, Belmont, Massachusetts, 1997.

[BW05] Dimitris Bertsimas and Robert Weismantel, *Optimization over integers*, Belmont, Mass: Dynamic Ideas, 2005.

[CHL+89] Y. Cheng, D. J. Houck, J. Liu, M. S. Meketon, L. Slutsman, R. J. Vanderbei, and P. Wang, *The AT &T KORBX®system*, AT & T Technical Journal **68** (1989), no. 3, 7–19.

[Coo06] Stephen Cook, *The P versus NP Problem*, The Millennium Prize Problems (J. Carlson, A. Jaffe, and A. Wiles, eds.), Clay Mathematics Institute, Cambridge, MA, 2006.

[Coo11] William J. Cook, *In Pursuit of the Traveling Salesman: Mathematics at the Limit of Computation*, Princeton University Press, December 2011.

[DFJ54] G. Dantzig, R. Fulkerson, and S. Johnson, *Solution of a Large-Scale Traveling-Salesman Problem*, Journal of the Operations Research Society of America **2** (1954), 393–410.

[ECI] Emilien Dupont, Chris Maes, and Gurobi Optimization Inc, *The Travelling Salesman Problem with Integer Programming and Gurobi*, http://examples.gurobi.com/traveling-salesman-problem/.

[Edm65] Jack Edmonds, *Minimum Partition of a Matroid Into Independent Subsets*, Journal of Research of the National Bureau of Standards **69B** (1965), no. 1-2.

[Edm67] ———, *Optimum Branchings*, Journal of Research of the National Bureau of Standards **71B** (1967), no. 4.

[FH13] Anja Fischer and Christoph Helmberg, *The symmetric quadratic traveling salesman problem*, Mathematical Programming **142** (2013), 205–254.

[Fis13] Anja Fischer, *A Polyhedral Study of Quadratic Traveling Salesman Problems*, PhD, Chemnitz University of Technology, Chemnitz, Germany, April 2013.

[Glo75] Fred Glover, *Improved Linear Integer Programming Formulations of Nonlinear Integer Problems*, Management Science **22** (1975), no. 4, 455–460.

[Gom58] Ralph E. Gomory, *Outline of an algorithm for integer solutions to linear programs*, Bulletin of the American Mathematical Society **64** (1958), no. 5, 275–279.

[GP07] Gregory Gutin and Abraham P Punnen (eds.), *The Traveling Salesman Problem and its Variations*, Combinatorial Optimization, vol. 12, Springer US, Boston, 2007.

[GW74] Fred Glover and Eugene Woolsey, *Converting the 0-1 Polynomial Programming Problem to a 0-1 Linear Program*, Operations Research **22** (1974), no. 1, 180.

[HJ85] Roger A. Horn and Charles R. Johnson, *Matrix Analysis*, Cambridge University Press, December 1985.

[HK56] A. J. Hoffman and J. B. Kruskal, *Integral Boundary Points of Convex Polyhedra*, Linear inequalities and related systems (Harold W Kuhn and Albert W Tucker, eds.), Annals of mathematics studies; no. 38, Princeton University Press, Princeton, 1956.

[HM09] Pierre Hansen and Christophe Meyer, *Improved compact linearizations for the unconstrained quadratic 0-1 minimization problem*, Discrete Applied Mathematics **157** (2009), 1267–1290.

[HR70] Peter L. Hammer and Abraham A. Rubin, *Some remarks on quadratic programming with 0-1 variables*, RAIRO - Operations Research - Recherche Opérationnelle **4** (1970), no. V3, 67–79.

[JLN+10] Michael Jünger, Thomas Liebling, Denis Naddef, George Nemhauser, William Pulleyblank, Gerhard Reinelt, Giovanni Rinaldi, and Laurence Wolsey (eds.), *50 Years of Integer Programming 1958–2008: From the Early Years to the State-of-the-Art*, Springer Berlin Heidelberg, 2010.

[JM08] Gerold Jäger and Paul Molitor, *Algorithms and Experimental Study for the Traveling Salesman Problem of Second Order*, Combinatorial Optimization and Applications (Boting Yang, Ding-Zhu Du, and Cao An Wang, eds.), Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2008, pp. 211–224.

[Kar84] N. Karmarkar, *A new polynomial-time algorithm for linear programming*, ACM, December 1984, pp. 302–311.

[KGV83] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, *Optimization by Simulated Annealing*, Science **220** (1983), no. 4598, 671–680.

[Kha80] L. G. Khachiyan, *Polynomial algorithms in linear programming*, Zh. Vychisl. Mat. Mat. Fiz **20** (1980), no. 1, 51–68 (Russian).

[KV18] Bernhard Korte and Jens Vygen, *Combinatorial Optimization: Theory and Algorithms*, Algorithms and Combinatorics, vol. 21, Springer Berlin Heidelberg, Berlin, Heidelberg, 2018 (eng).

[LD60] A. H. Land and A. G. Doig, *An Automatic Method of Solving Discrete Programming Problems*, Econometrica **28** (1960), no. 3, 497–520.

[Mah10] Ashutosh Mahajan, *Presolving Mixed Integer Linear Programs*, Tech. report, July 2010.

[McC76] Garth P. McCormick, *Computability of global solutions to factorable nonconvex programs: Part I convex underestimating problems*, Mathematical Programming **10** (1976), no. 1, 147–175.

[MTZ60] C. E. Miller, A. W. Tucker, and R. A. Zemlin, *Integer Programming Formulation of Traveling Salesman Problems*, Journal of the ACM (JACM) **7** (1960), no. 4, 326–329.

[ÖAL09] Temel Öncan, I. Kuban Altinel, and Gilbert Laporte, *A comparative analysis of several asymmetric traveling salesman problem formulations*, Computers & Operations Research **36** (2009), 637–654.

[OFF+17] Aichholzer Oswin, Anja Fischer, Frank Fischer, J. Fabian Meier, Ulrich Pferschy, Alexander Pilz, and Rostislav Stanek, *Minimization and maximization versions of the quadratic travelling salesman problem*, Optimization **66** (2017), 521–546.

[OM02] Jonathan H. Owen and Sanjay Mehrotra, *On the Value of Binary Expansions for General Mixed-Integer Linear Programs*, Operations Research **50** (2002), no. 5, 810–819.

[Opt] Gurobi Optimization, *Mixed-Integer Programming (MIP) Basics | Gurobi*, http://www.gurobi.com/resources/getting-started/mip-basics.

[OW06] A. J. Orman and H. Paul Williams, *Advances in Computational Management Science, Optimisation, Econometric and Financial Analysis*, A survey of different integer programming formulations of the travelling salesman problem (Erricos John Kontoghiorghes and Cristian Gatu, eds.), Springer, Berlin, Heidelberg, 2006, pp. 91–104.

[Pan18] Pooja Pandey, *Topics in Binary Optimization Problems*, PhD, Simon Fraser University, Surrey, BC, July 2018.

[Pap94] Christos H. Papadimitriou, *Computational Complexity*, Addison-Wesley, 1994.

[Pap03] ———, *Computational complexity*, Encyclopedia of Computer Science (Anthony Ralston, Edwin D. Reilly, and David Hemmendinger, eds.), John Wiley and Sons Ltd., Chichester, UK, 4th ed., January 2003.

[PP18] Abraham P. Punnen and Pooja Pandey, *Representations of quadratic combinatorial optimization problems: A case study using the quadratic set covering problem*, arXiv:1802.00897 [cs, math] (2018), arXiv: 1802.00897.

[PW17] Abraham P. Punnen and Brad D. Woods, *A Characterization of Linearizable instances of the Quadratic Traveling Salesman Problem*, arXiv:1708.07217 [cs] (2017), arXiv: 1708.07217.

[RMBG16] Borzou Rostami, Federico Malucelli, Pietro Belotti, and Stefano Gualandi, *Lower bounding procedure for the asymmetric quadratic traveling salesman problem*, European Journal of Operational Research **253** (2016), no. 3, 584–592.

[Rot] Edward Rothberg, *Gurobi Optimization*, https://www.informs.org/Impact/O.R.-Analytics-Success-Stories/Industry-Profiles/Gurobi-Optimization.

[SA90] H. Sherali and W. Adams, *A Hierarchy of Relaxations between the Continuous and Convex Hull Representations for Zero-One Programming Problems*, SIAM Journal on Discrete Mathematics **3** (1990), no. 3, 411–430.

[Sch03] A. Schrijver, *Combinatorial optimization: polyhedra and efficiency*, Algorithms and combinatorics, no. 24, Springer, Berlin ; New York, 2003 (eng).

[Ser10] Denis Serre, *Matrices with Real or Complex Entries*, Matrices: Theory and Applications (S Axler and K.A. Ribet, eds.), Graduate Texts in Mathematics, Springer New York, New York, NY, 2 ed., 2010, pp. 83–108.

[Sha12] David Shanno, *Who Invented the Interior-Point Method?*, Documenta Mathematica (2012), 65–73.

[Sti45] George J. Stigler, *The Cost of Subsistence*, Journal of Farm Economics **27** (1945), no. 2, 303–314.

[tsp] *TSPLIB*, https://wwwproxy.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95/.

[Wat67] Lawrence J. Watters, *Reduction of Integer Polynomial Programming Problems to Zero-One Linear Programming Problems*, Operations Research **15** (1967), no. 6, 1171–1174.

[Wei] Eric W. Weisstein, *Gershgorin Circle Theorem*, http://mathworld.wolfram.com/GershgorinCircleTheorem.html.

[WP17] Brad Woods and Abraham Punnen, *A class of exponential neighbourhoods for the quadratic travelling salesman problem*, arXiv:1705.05393 [cs] (2017), arXiv:1705.05393.

[WPS17] Brad Woods, Abraham Punnen, and Tamon Stephen, *A linear time algorithm for the 3-neighbour Travelling Salesman Problem on a Halin graph and extensions*, Discrete Optimization **26** (2017), 163–182.