Chapter 1

THE TRAVELING SALESMAN PROBLEM: APPLICATIONS, FORMULATIONS AND VARIATIONS

Abraham P. Punnen
Department of Mathematical Sciences
University of New Brunswick-Saint John
New Brunswick, Canada
punnen@unbsj.ca

This chapter is dedicated to the memory of my grandparents
Mr. Korah Abraham and Mrs. Sosamma Abraham

1. Introduction

The traveling salesman problem (TSP) is to find a routing of a salesman who starts from a home location, visits a prescribed set of cities and returns to the original location in such a way that the total distance travelled is minimum and each city is visited exactly once. Although a business tour of a modern day traveling salesman may not seem to be too complex in terms of route planning, the TSP in its generality represents a typical 'hard' combinatorial optimization problem.

TSP was known among mathematicians and statisticians under various names. Karl Menger considered a variation of TSP called the *messenger problem* (Botenproblem) [116, 451, 593]. It may be noted that the messenger problem and TSP are equivalent in the sense that one problem can be reduced to the other using very simple transformations. Bock [116] gives an English translation of the description of messenger problem from [593] as follows:

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

It is believed that the report by Menger [593] is the first published work on TSP. He pointed out that complete enumeration is a possible strategy to compute an optimal solution but a 'nearest neighbor' algorithm does not guarantee an optimal solution.

Mahalanobis [575] presented a talk at the Indian Science Congress in 1939 on the problem of estimating the area under jute (Corchorus Capsularis) farm in Bengal (currently spread over West Bengal in India and Bangladesh). The objective is to keep the cost involved in the land survey operations below a threshold level while maximizing the accuracy of the result. As part of the cost calculations, he was interested in a least cost routing through a finite number of points in the Euclidean plane. Although no attempt was reported in calculating the least possible cost for such a traversal, it is clear from the discussions that Mahalanobis was aware of the difficulties involved in finding such a minimum cost routing. His cost calculations however are based on expected values considering random points in the plane which forms a small part of a more general cost function.

In a 1955 paper, Morton and Land [608] indicated that "the TSP may be familiar to statisticians as *the mean minimum distances problem*" by citing supporting references [358, 458, 583]. In their brief historic account on TSP, Morton and Land wrote:

> "In the United States this problem is known as the Traveling-Salesman problem; the salesman wishes to visit one city in each of the 48 States and Washington, D.C, in such a sequence as to minimize the total road distance traveled. Thinking of the problem independently and on a smaller geographical scale, we used to call it the laundry van problem, where the conditions were a daily service by a one-van laundry. Since the American term was used by Robinson (1949) we propose to adopt it here."

The name "traveling salesman problem" for the optimization problem of our discussion is believed to have originated in the United States. Perhaps the first report using this name was published in 1949 [725]. However, it would be reasonable to say that a systematic study of the TSP as a combinatorial optimization problem began with the work of Dantzig, Fulkerson, and Johnson [239]. The survey papers [474, 590, 592, 591] and the books [548, 711] summarize various developments on the subject.

Let us now give a formal mathematical definition of the TSP. For details of basic terminology and notations in graph theory we refer to Appendix A. Let $G = (V, E)$ be a graph (directed or undirected) and $\mathbb{F}$ be the family of all Hamiltonian cycles (tours) in G. For each edge $e \in E$ a cost (weight) c_e is prescribed. Then the traveling salesman problem is to find a tour (Hamiltonian cycle) in G such that the sum of the costs of edges of the tour is as small as possible.

Without loss of generality, we assume that G is a complete graph (digraph), for otherwise we could replace the missing edges with edges of very large cost. Let the node set $V = \{1, 2, \ldots, n\}$. The matrix $C = (c_{ij})_{n\times n}$, is called the cost matrix (also referred to as the distance matrix or weight matrix), where the $(i, j)^{th}$ entry c_{ij} corresponds to the cost of the edge joining node i to node j in G.

The TSP can also be viewed as a permutation problem. Generally, two variations of the permutation representation of TSP are in use; a *quadratic representation* and a *linear representation.* Let $\mathbb{P}_n$ be the collection of all permutations of the set $\{1, 2, \ldots, n\}$. Then the TSP is to find a $\pi = (\pi(1), \pi(2), \ldots, \pi(n))$ in $\mathbb{P}_n$ such that $c_{\pi(n)\pi(1)} + \sum_{i=1}^{n-1} c_{\pi(i)\pi(i+1)}$ is minimized. In this case $(\pi(1), \pi(2), \ldots, \pi(n))$ gives the order in which cities are visited, starting with city $\pi(1)$. For example, if $n = 5$ and $\pi = (2, 1, 5, 3, 4)$ then the corresponding tour will be $(2, 1, 5, 3, 4, 2)$. Every cyclic shift of π also give the same tour. Thus there are n different permutations which represent the same tour. This representation of TSP is generally used in the context of some sequencing problems and gives rise to a binary quadratic programming formulation of the TSP (see Section 4). Thus, we refer to this the *quadratic permutation representation* of TSP. Chapter 12 of this book uses this quadratic permutation representation of the TSP.

In another permutation representation of TSP, referred to as *linear permutation representation*, only special types of permutations, called cyclic permutations, are considered feasible. Let $\mathbb{C}_n$ be the collection of all cyclic permutations of $\{1, 2, \ldots, n\}$. Then the TSP is to find a $\sigma = (\sigma(1), \sigma(2), \ldots, \sigma(n)) \in \mathbb{C}_n$ such that, $\sum_{i=1}^{n} c_{i\sigma(i)}$ is minimized. Under this representation, $\sigma(i)$ is the successor of city i in the resulting tour, for $i = 1, 2, \ldots, n$. For example if $n = 5$ and $(\sigma(1), \sigma(2), \ldots, \sigma(5))$ $= (2, 4, 5, 3, 1)$, the resulting tour is given by $(1, 2, 4, 3, 5, 1)$. This representation of TSP leads to a binary liner programming formulation of TSP and hence we call it the *linear permutation representation.* Chapter 11 and part of Chapter 15 use the linear permutation representation of the TSP.

Depending on the nature of the cost matrix (equivalently the nature of G), TSP is divided in to two classes. If C is symmetric (i.e. G is undirected) then the TSP is called the symmetric traveling salesman problem (STSP). If C is not necessarily symmetric (equivalently the graph G is directed) then it is called the asymmetric traveling salesman problem (ATSP). Since every undirected graph can be viewed as a directed graph, by duplicating edges one in the forward direction and the other in the backward direction, STSP can be viewed as a special case of ATSP. Interestingly, it is also possible to formulate ATSP as a STSP [471] by doubling the number of nodes. Consider the ATSP on a digraph $D = (N, A)$ with c_{ij} as the cost of arc (i, j). Construct the undirected graph $G^* = (V^*, E^*)$ with node set $V^* = N \cup \{1^*, 2^*, \dots, n^*\}$ where $N = \{1, 2, \dots, n\}$. For each arc (i, j) in D, create an edge (i, j^*) in G^* with cost c_{ij}. Also introduce n dummy edges (i, i^*) of cost $-M$ where M is a very large number. All other edges have cost M. It can be verified that solving the ATSP on D is equivalent to solving the STSP on G^*.

1.1. The Shoelace Problem - A simple TSP

According to Bata shoe museum (Toronto, Canada) evidence of shoe making exists as early as 10,000 BC; the average person walks the equivalent of three-and-a-half times around the earth in a life time; and North Americans spend almost \$18 billion a year on footwear! Having a comfortable pair of shoes doesn't ensure comfortable walking. Proper lacing of the shoe is also important in increasing foot comfort and perhaps increasing the life span of the shoe. Needless to say, proper lacing of shoe is important for a traveling salesman. Our salesman however is interested in lacing his shoes with a lace of minimum length! Optimal shoe-lacing as a mathematical entertainment problem was introduced by Halton [431]. It can be described as follows:

Let $A_0, A_1, A_2, \dots, A_n$ and $B_0, B_1, B_2, \dots, B_n$ be the eyelets of a shoe arranged on two parallel lines. (See Fig: 1.1).

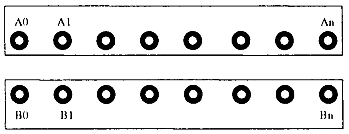

Figure 1.1. Eyelets on a shoe

A feasible lacing is a 'path' from A_0 to B_0 visiting each eyelet exactly once, choosing A_i's and B_j's alternately, and closing the path by moving from B_0 to A_0. It can be represented as an ordered pair (π, σ) of two permutations π and σ of $\{1, 2, \ldots, n\}$. Note that (π, σ) represents the unique lacing

$$A_0 \to B_{\sigma(1)} \to A_{\pi(1)} \cdots \to A_{\pi(n-1)} \to B_{\sigma(n)} \to A_{\pi(n)} \to B_0 \to A_0.$$

The length of a feasible lacing (π, σ), denoted by $L(\pi, \sigma)$, is given by

$$\sum_{i=1}^{n-1} [d(\sigma(i), \pi(i)) + d(\pi(i), \sigma(i+1)) + d(0, \sigma(1)] + d(\sigma(n), \pi(n))$$
$$+ d(\pi(n), 0) + d(0, 0),$$

where $d(i, j)$ is the Euclidean distance between A_i and B_j. Let the distance between two consecutive eyelets on the same side be α and the distance between A_i and B_i be β. Then, it can be verified that

$$L^{\alpha\beta}(\pi, \sigma) = \sqrt{\sigma(1)^2\alpha^2 + \beta^2} + \sqrt{\pi(n)^2\alpha^2 + \beta^2} + \sqrt{(\pi(n) - \sigma(n))^2\alpha^2 + \beta^2}$$
$$+ \sum_{i=1}^{n-1} (\sqrt{(\pi(i) - \sigma(i))^2\alpha^2 + \beta^2} + \sqrt{(\sigma(i+1) - \pi(i))^2\alpha^2 + \beta^2}) + \beta$$

The *shoelace problem* SP(α, β) is to find a feasible lacing $(\bar{\pi}, \bar{\sigma})$ of minimum length $L^{\alpha\beta}(\bar{\pi}, \bar{\sigma})$. Note that there are $(n!)^2$ feasible lacing patterns. Interestingly we will see that one of the most commonly used lacing patterns is indeed optimal. Let us consider four special lacing patterns (π, σ). When $\pi = (2, 4, \ldots, 5, 3, 1)$ and $\sigma = (5, 3, 1, \ldots 2, 4)$, (π, σ) is called a *zig-zag lacing*. When $\pi = (2, 4, \ldots, 3, 1)$ and $\sigma = (1, 2, 4, 6 \ldots 5, 3)$ we have a *straight lacing*. *Quick lacing* corresponds to $\pi = (n, n-1, \ldots, 3, 2, 1)$ and $\sigma = (n, n-1, \ldots, 2, 1)$ and *diagonal lacing* corresponds to $\pi = (1, 2, 3 \ldots n-1, n)$ and $\sigma = (n, n-1, \ldots, 2, 1)$.

It can be verified that the zig-zag lacing is of length $2(\beta + n\sqrt{\alpha^2 + \beta^2})$, straight lacing is of length $(n+1)\beta + 2\sqrt{\beta^2 + \alpha^2} + (n-1)\sqrt{4\alpha^2 + \beta^2}$, quick lacing is of length $(n+1)\beta + n\sqrt{\alpha^2 + \beta^2} + \sqrt{n^2\alpha^2 + \beta^2}$, and diagonal lacing is of length $2(\beta + \sum_{i=1}^{n-1} \sqrt{(n-i)^2\alpha^2 + \beta^2})$. By choosing $n = 10, \alpha = 1$ and $\beta = 2$, the zig-zag, straight, quick, and diagonal lacings have approximate lengths 48.72, 51.92, 54.55, and 103.19, respectively. Thus the zig-zag lacing comes out to be the winner. Indeed, it can be shown that,

Theorem 1 *[431] For $n \geq 3$, zig-zag lacing is the unique optimal solution to the shoelace problem SP(α, β).*

For $n = 2$, all four lacing patterns discussed earlier have the same length. So far we have assumed that $A_0, A_1, A_2, \ldots, A_n$ are arranged

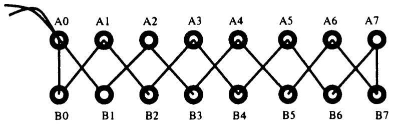

Figure 1.2. Zig-zag Lacing

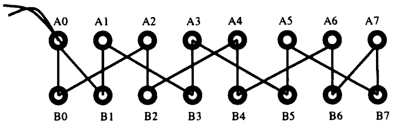

Figure 1.3. Straight Lacing

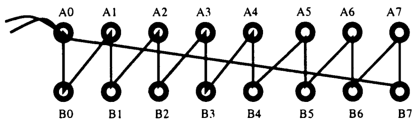

Figure 1.4. quick Lacing

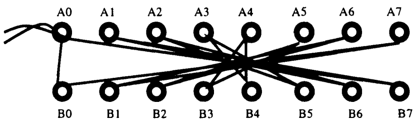

Figure 1.5. diagonal Lacing

on a straight line and $B_0, B_1, B_2, \dots B_n$ are arranged on a parallel line. However, this assumption can be relaxed. If $A_iA_jB_kB_l$ for all $i < j$ and $k > l$ forms a convex quadrilateral, then also the zig-zag lacing can be shown to be optimal [600, 789]. What can we say about the diagonal lacing? Is it the longest? We leave this question for the reader.

Choosing the eyelets $A_0, A_1, \dots A_n$ and $B_0, B_1, \dots B_n$ as nodes of a complete bipartite graph where the edge costs correspond to the Euclidean distances between these points except for the edge (A_0, B_0) the cost of which is $-M$ where M is a very large number, the shoelace problem can be considered as a TSP restricted to a complete bipartite graph which in turn can be viewed as traveling salesman problem on a complete graph by adding edges of large cost.

Although we have a closed form solution for this special case, TSP in general is NP-hard. (For a summary of computational complexity results on TSP, we refer to Appendix B.) This inherent difficulty of the problem and its various practical applications inspired researchers and practitioners to investigate various aspects of the problem. In particular, significant developments have taken place in the recent past in the study of structural properties of the problem, development of efficient exact and approximation algorithms, and identification of efficiently solvable special cases. In the following chapters we study each of these aspects of TSP and its variations. In the rest of this chapter, we will concentrate on applications, variations and mathematical formulations of the problem.

2. Simple Variations of the TSP

Several variations of the TSP that are studied in the literature have been originated from various real life or potential applications. Let us first consider some of these variations that can be reformulated as a TSP using relatively simple transformations.

The MAX TSP: Unlike the TSP, the objective in this case is to find a tour in G where the total cost of edges of the tour is maximum. MAX TSP can be solved as a TSP by replacing each edge cost by its additive inverse. If we require that the edge costs are to be non-negative, a large constant could be added to each of the edge-costs without changing the optimal solutions of the problem. For details on MAX TSP, we refer to Chapter 12.

The bottleneck TSP: In the case of a bottleneck TSP, the objective is to find a tour in G such that the largest cost of edges in the tour is as small as possible. A bottleneck TSP can be formulated as a TSP with exponentially large edge costs. This problem is studied in detail in Chapter 15.

TSP with multiple visits (TSPM): Here we want to find a routing of a traveling salesman, who starts at a given node of G, visits each node *at least* once and comes back to the starting node in such a way that the total distance traveled is minimized. This problem can be transformed into a TSP by replacing the edge costs with the shortest path distances in G. In the absence of negative cycles, shortest path distances between all pairs of nodes of a graph can be computed using efficient algorithms [7]. If G contains a negative cycle, then TSPM is unbounded. Several shortest path algorithms are capable of detecting negative cycles in a graph [7].

Messenger problem: This problem[593] is also known as the wandering salesman problem [474]. Given two specified nodes u and v in G, we want to find a least cost Hamiltonian path in G from u to v. This can be solved as a TSP by choosing a cost of $-M$ for the edge (v, u) where M is a large number. If no nodes are specified, and one wishes to find a least cost Hamiltonian path in G, it can also be achieved by solving a TSP. Modify the graph G by creating a new node and connecting it with all the nodes of G by edges (for directed graphs, introduce two arcs, one in forward and other in backward directions) of cost -M. From an optimal solution to the TSP on this modified graph, an optimal solution to the original problem can be recovered.

Clustered TSP: In this case, the node set of G is partitioned into clusters $V_1, V_2, \ldots, V_k$. Then the clustered TSP [426, 469] is to find a least cost tour in G subject to the constraint that cities within the same cluster must be visited consecutively. This problem can be reduced to a TSP by adding a large cost M to the cost of each inter-cluster edge.

Generalized TSP(GTSP): As in the previous case, let $V_1, V_2, \ldots, V_k$ be a partition of the node set of G. In a GTSP, we want to find a shortest cycle in G which passes through exactly one node from each cluster V_i, $1 \leq i \leq k$. If $|V_i| = 1$ for all i, GTSP is the same as TSP. Using the reduction described in [629] we now show that GTSP can be reduced to a TSP for arbitrary $|V_i|$'s. Without loss of generality, assume that G is a digraph and the partitions are numbered in such a way that $|V_i| \geq 2$ for $1 \leq i \leq r$ and $|V_i| = 1$ for $r+1 \leq i \leq k$. For any $i \leq r$ let $V_i = \{i_1, i_2, \ldots, i_{n_i}\}$. For each arc $e \in E$, consider a new cost d_e defined as follows:

$$d_{i_j i_{j+1}} = -M, \quad j = 1, \ldots, n_i \text{ with } n_i + 1 \equiv 1, i = 1, \ldots, r$$

This ensures that if an optimal TSP tour in G enters a cluster V_i through node i_j, it visits vertices of V_i in the order $i_j, i_{j+1}, \ldots, i_{n_i}, i_1, \ldots, i_{j-1}$ and leaves the cluster V_i from the node i_{j-1}. We want to translate this outcome equivalent to a GTSP tour entering and leaving the cluster V_i by visiting just node i_j. To replicate this fact, we make the new cost of arcs going out of i_{j-1} corresponds to the original cost of arcs leaving i_j. More precisely,

$$d_{i_{j-1}p} = c_{i_j p}, \quad p \notin V_i, j = 1, 2, \ldots, n_i \text{ with index } 0 \equiv n_i, i = 1, \ldots, r$$

and $d_{uv} = c_{uv}$ for all other edges. From an optimal solution to the TSP on G with the modified costs d_e for $e \in E$, an optimal solution to GTSP can be recovered. GTSP is discussed in detail in Chapter 13.

The m-salesmen TSP: Assume that m salesmen are located at node 1 of G. Starting from node 1, each salesman visits a subset X_i of nodes of G exactly once and returns to node 1. We are interested in finding a partition $X_1, X_2, \ldots, X_m$ of $V - \{1\}$ and a routing of each of the m salesmen such that (i) $|X_i| \geq 1$ for each i, (ii) $\cup_{i=1}^{m} X_i = V - \{1\}$, (iii) $X_i \cap X_j = \emptyset$ for $i \neq j$, and (iv) total distance travelled by all the salesmen is minimized [474]. If G is undirected, the m-salesmen TSP can be formulated as a TSP on a graph $G' = (V', E')$ with $m-1$ additional nodes, where $V' = V \cup \{n+2, n+3, \ldots, n+m\}$, $E' = E \cup \{(i, n+k) : 2 \leq$

$k \leq m, 2 \leq i \leq n\}$. Let c'_{ij} be the cost of edges in G', where $c'_{ij} = c_{ij}$ if $(i,j) \in E$, and $c'_{i,n+k} = c_{i1}$, for $2 \leq k \leq m$, $2 \leq i \leq n$. From an optimal TSP solution on G' an optimal solution to the m-salesmen TSP can be recovered. The case when G is a directed graph can be handled similarly with simple modifications in the above transformation. For variations of the m-salesmen TSP that can be formulated as a TSP we refer to [96, 695].

3. Applications of TSP

Applications of the TSP and its variations go way beyond the route planning problem of a traveling salesman and spans over several areas of knowledge including mathematics, computer science, operations research, genetics, engineering, and electronics. In this section, we discuss some selected applications of TSP.

3.1. Machine Scheduling Problems

Perhaps the most well studied application area of the TSP is machine sequencing and scheduling. A simple scheduling application can be described as follows. There are n jobs $\{1, 2, \ldots, n\}$ to be processed sequentially on a machine. Let c_{ij} be the set up cost required for processing job j immediately after job i. When all the jobs are processed, the machine is reset to its initial state at a cost of c_{j1}, where j is the last job processed. The sequencing problem is to find an order in which the jobs are to be processed such that the total setup cost is minimized. Clearly, finding a permutation π of $\{1, 2, \ldots, n\}$ that minimizes $c_{\pi(n)\pi(1)} + \sum_{i=1}^{n-1} c_{\pi(i)\pi(i+1)}$ solves the problem.

In many practical applications, the jobs are not arbitrary but often can be clustered together so that the setup time, if any, between jobs within a cluster is relatively small compared to setup time between jobs in two different clusters. This is a typical scenario in assembly line. Assembling similar products need minimal setup time in between; but if a different product needs to be assembled, the setup time may be larger as one may require new parts, tools etc. The sequencing problem with such a specialized cost matrix reduces to the clustered TSP. This cluster property can be effectively exploited in developing efficient algorithms. Ozgur and Brown [639] developed a two stage heuristic for such a problem originated from the mechanical push-pull cable control systems.

Let us now consider another scheduling problem - a no wait flow shop problem. There are n jobs each requiring processing on m machines in the order $1, 2, 3, \ldots, m$. No job is allowed to have a waiting time between processing on two consecutive machines. The objective is to find an optimum sequencing of jobs to be processed so that the makespan (total completion time) is minimum. Applications of this type of sequencing problems arise in a variety of situations [272]. The no-wait flow shop problem is strongly NP-hard, but solvable in polynomial time when $m = 2$ in which case it reduces to the well known Gilmore-Gomory problem. (See Chapter 11 for more details on this.)

From an instance of the no-wait flow shop problem, we construct an equivalent TSP instance using the reduction proposed in [272]. Create a complete directed graph G on $n+1$ nodes, where node j corresponds to job j, $1 \leq j \leq n$ and node $n+1$ represents both the start and end of processing. The cost c_{ij} of arc (i,j) in G represents the additional schedule length if job j is the immediate successor of job i in the schedule. Thus a minimum cost tour in G corresponds to a schedule with minimum makespan. To complete the reduction, we must identify the values of c_{ij}. Let p_{jk} be the processing time of job j on processor k, $1 \leq j \leq n$, $1 \leq k \leq m$. Then c_{ij} can be obtained [272] using the equations,

$$\begin{aligned}
c_{n+1,i} &= \sum_{r=1}^{m} p_{ir}, \quad i = 1,2,\ldots,n \\
c_{ij} &= \max_{1\leq k\leq m}\{\sum_{r=1}^{k} p_{ir} + \sum_{r=k}^{m} p_{jr}\} - \sum_{r=1}^{m} p_{ir}, \quad 1 \leq i,j \leq n, i \neq j \\
c_{i,n+1} &= c_{ii} = 0, \quad i = 1,2,\ldots,n.
\end{aligned}$$

The applications we considered so far reduce a given problem to the TSP. Let us now discuss another application of TSP where it is used as a subroutine to develop efficient heuristics for a more complex problem involved in the control of a pick and placement machine. This application is taken form Ahmadi and Mamer [4].

A pick-and-place machine includes a work platform, a placement arm, pick and place heads, component delivery carrier (feeder magazines), and tool magazines. Generally these types of machines are used in printed circuit board assembly [4, 83] and available in different architectures such as the placement arm is moving while work platform remains stationary or the placement arm remains stationary while the work platform is moving or a combination of the two. In our model the placement arm moves between two fixed locations - the pick up position and the delivery position. The work platform moves in the x and y directions and feeder-tool carrier moves in the x direction only. The head attached to the placement arm needs to be changed if a different component type is to be placed on the board. Normally tool change operations are more time consuming in comparison to pick and placement operations. The cycle time - i.e. the time required to move the placement arm between the pickup and delivery positions, perform a pick up operation and perform a delivery operation- is assumed to be constant. Let p_i be the position of the i^{th} placement site, c_{ij} be the time required to move the placement board from site p_i to site p_j. When the placement head moves to the pickup position, simultaneously the work platform moves to adjust the next placement site. Let t_{ij} denote the time lost in switching between part i and part j and N_k denote the index set corresponding to placement sites requiring part type k, $k = 1, 2, \dots m$. It is assumed that only one component is placed at a given placement site and the p_i's are numbered consecutively with the part type. Thus $N_1 = \{p_1, p_2, \dots, p_{k_1}\}$, $N_2 = \{p_{k_1+1}, p_{k_1+2}, \dots, p_{k_1+k_2}\}$, ... Define $n = \sum_{i=1}^{m} k_i$. The calculation of c_{ij} and t_{ij} depends on the architecture of the machine. The pickup and placement problem is to find a tour through $p_1, p_2 \dots, p_n$ which minimizes the total time to complete all placements, including the time required to start and return the machine to the resting position.

Although this problem is not exactly a TSP, its close relationship with TSP allows us to develop efficient heuristics by using TSP as a subroutine or by appropriate modifications of the TSP heuristics itself. Ahmadi and Mamer [4] developed a two stage heuristic for this problem using a TSP solver (exact or heuristic) as a subroutine. For details of other research works on printed circuit board assembly that are relevant to the TSP, we refer to [83].

3.2. Cellular Manufacturing

In a cellular manufacturing system families of parts (products) that require similar processing are grouped and processed together in a specialized cell to achieve efficiency and cost reductions. In such systems robots are often used in handling material to improve the efficacy of the cell. Aneja and Kamoun [23] considered the problem of sequencing the robot activities in a two machine robotic cell and showed that the problem can be formulated as a TSP. For some fundamental results on this problem, we refer to [758].

Let $p_1, p_2, \dots, p_n$ be n part-types to be produced. Each part type p_i needs to be processed on two machines M1 and M2 with M1 followed by M2. Let a_i be the processing time of p_i on M1 and b_i be the processing time of p_i on M2. A robot picks up a part from an input storage buffer, loads it on to machine M1, collects from M1 and then loads to M2, collects from M2 and places the product in the output storage buffer. The storage buffers and machines are arranged on a straight line so that the robot travels only along a straight line. The time taken by the robot for movement between input storage buffer and each machine, between each machine and output storage buffer, and between the machines is given by t. Let l be the time taken by the robot in loading or unloading.

Two robotic movement cycles $C1$ and $C2$ with a point of intersection are considered [758]. Let π be a permutation of $\{1, 2, \ldots, n\}$ representing the sequence in which the parts are selected for processing. In cycle $C1$, starting from an initial state, the robot loads a part, say $p_{\pi(i)}$ on M2, waits for it to complete, picks up the processed part from M2 and places it in the output storage buffer, goes back to the input storage buffer, picks up part $p_{\pi(i+1)}$ and moves to M2. The cycle time in this case is given by

$$T^1_{\pi(i)\pi(i+1)} = 6(l + t) + b_{\pi(i)} + a_{\pi(i+1)}.$$

For cycle C2, starting from an initial state, the robot loads $p_{\pi(i)}$ on M2, moves to the input storage buffer, picks up part $p_{\pi(i+1)}$ and loads on to M1, goes to M2 and waits (if necessary) to complete the processing of $p_{\pi(i)}$ on M2, picks up part $p_{\pi(i)}$ from M2 and moves to the output storage buffer to unload the part, moves back to M1 and wait (if necessary) to complete the processing of $p_{\pi(i+1)}$ on M1, picks up $p_{\pi(i+1)}$ and moves to M2. The cycle time in this case can be calculated as

$$T^2_{\pi(i)\pi(i+1)} = 8t + 6l + w_1(i + 1) + w_2(i)$$

where $w_1(i+1)$, the waiting time at M1, is given by $\max\{0, a_{\pi(i+1)} - (4t + 2l + w_2(i))\}$ and $w_2(i)$, the waiting time at M2, is given by $\max\{0, b_{\pi(i)} - (4t + 2l)\}$. Let $X_i = b_i + 2(t + l)$ and $Y_i = a_i + 2(t + l)$. Then T^1_{ij} and T^2_{ij} can be written as

$$T^1_{ij} = 2(t + l) + X_i + Y_j$$

and

$$T^2_{ij} = 2(t + l) + \max\{X_i + Y_i, 6t + 2l\}.$$

Note that each cycle C1 or C2 involves two parts $p_{\pi(k)}$ and $p_{\pi(k+1)}$ for some k, under the schedule π. For the cycles $C1$ and $C2$, the state in which the robot completed loading a part on M2 is common. Thus at such a state the robot needs to make a decision which cycle to be used. Clearly, if $T^1_{\pi(i)\pi(i+1)} < T^2_{\pi(i)\pi(i+1)}$, cycle C1 is selected, otherwise C2 is selected. Now consider the cost matrix $C = (c_{ij})_{n\times n}$ where $c_{ij} = \min\{T^1_{ij}, T^2_{ij}\}$. The optimal TSP tour using cost matrix C provides an optimal part sequencing. This TSP can be solved in $O(n \log n)$ time exploiting the special structure the matrix C (see [23] and Chapter 11).

Instead of the straight line topology for robotic movements, other structures could be considered. For example, M1, M2, the input storage buffer, and the output storage buffer are placed at 4 corners of a rectangle and the robot moves along the perimeter of this rectangle choosing shortest path distances, the resulting part sequencing problem can also be formulated as a TSP.

3.3. Arc Routing Problems

A general arc routing problem, known as mixed windy rural postman problem (MWRPP), can be stated as follows. Let $G = (V, A \cup E)$ be a mixed graph where elements of A are arcs (directed) and elements of E are edges (undirected). Let $A' \subset A$ and $E' \subset E$. The arc and edge costs are assumed to be non-negative. The MWRPP is to find a minimum cost closed walk on G containing all arcs in A' and all edges in E'. Several problems such as mixed Chinese postman problem, windy Chinese postman problem, windy rural postman problem, stacker crane problem, etc. are special cases of MWRPP. Applications of MWRPP also include street sweeping, snow plowing, etc. and the problem is known to be NP-hard. We observe that MWRPP can be solved as an asymmetric TSP. This transformation is taken from Laporte [534].

In G, replace each undirected edge (i,j) by two arcs (i,j) and (j,i) with cost equal to that of (i,j). Let $\bar{G} = (V, \bar{A})$ be the resulting digraph. Define $\bar{A}' = A' \cup \{(i,j),(j,i) \in \bar{A} \ : \ (i,j) \in E'\}$. We now formulate a generalized TSP on the digraph $D = (U, Z)$. For each arc $(i,j) \in \bar{A}'$ there is a node u_{ij} in U. The cost of arc $(u_{ij}, u_{pq}) \in Z$ is defined as the length of the shortest path from node i to node q in $\bar{G}$. Now the MWRPP is equivalent to a GTSP on D where the node set partition $\{U_1, U_2, \ldots U_m\}$ is given by $U_l = \{u_{ij}\}$ if $(i,j) \in A'$, $l = 1, 2, \ldots, |A'|$ and $U_l = \{u_{ij}, u_{ji}\}$ if $(i,j) \in E'$, $l = |A'| + 1, \ldots, |A'| + |E'|$. This GTSP can be reduced to a TSP as discussed earlier. Laporte [534] reports experimental results on several arc routing problems using such transformations.

3.4. Frequency Assignment Problem

In a communication network with a collection of transmitters, the frequency assignment problem is to assign a frequency to each transmitter from a given set of available frequencies satisfying some interference constraints. These constraints can be represented by a graph $G = (V, E)$ where each node i represents a transmitter. A non-negative integer weight c_{ij} is prescribed for each arc (i,j) representing the tolerance. Let $F = \{0, 1, 2, \ldots, R\}$ be a collection of allowable frequencies.

A frequency assignment is to assign number $f(i) \in F$ to node $i \in V$ such that $|f(i) - f(j)| > c_{ij}$ for all $(i, j) \in E$. If such an assignment exists, it is called a feasible assignment. If R is sufficiently large a feasible assignment is always possible. The minimum value of R for which a feasible assignment for G exists is called the span of G and is denoted by $Span(G)$.

Let G^* be the complete graph obtained from G by adding (if necessary) edges of zero weight. Let $c_{ij}^{'}$ be the weight of edge (i, j) in G^* such that $c_{ij}^{'} = c_{ij} + 1$. Let $C^{'}(H^*)$ be the sum of the weights of edges in a minimum cost Hamiltonian path in G^*. Smith and Hurly [765] showed that $Span(G) \geq C^{'}(H^*)$. Thus TSP can be used to compute a lower bound for the frequency assignment problem. Successful applications of the TSP based bounds and its variations in solving frequency assignment problem are discussed in [17].

3.5. Structuring of Matrices

Let A be a finite set and $f : A \times A \longrightarrow \mathbb{R}$. Consider the matrix $X = (x_{ij})_{m \times n}$ where $x_{ij} \in A$. For each row i of X, let $L_i(X) = \sum_{j=1}^{n} f(x_{ij}, x_{i(j+1)})$ where $x_{i(n+1)} = x_{i1}$. Similarly for each column j, let $C_j(X) = \sum_{i=1}^{m} f(x_{ij}, x_{(i+1)j})$ where $x_{(m+1)j} = x_{1j}$. Define $L(X) = \sum_{i=1}^{m} L_i(X)$ and $C(X) = \sum_{j=1}^{n} C_j(X)$. Let $S(X)$ be the family of matrices obtained by permuting rows of X. For $Y, Z \in S(X)$, Y is preferable to X if $C(Y) < C(Z)$. The set $B = \{Y \in S(X) : C(Y) = \min_{Z \in S(X)} C(Z)\}$ is called matrices with best row structure with respect to X. Structuring of rows of a matrix X corresponds to identifying an element of B. This problem can be solved by solving a TSP on the complete graph $K_m = (V, E)$ where $V = \{1, 2, \ldots, m\}$ with the weight of edge (i, j) is given by $c_{ik} = \sum_{j=1}^{n} f(x_{ij}, x_{kj})$ [516, 555]. In a similar way, structuring of columns of a matrix can be solved as a TSP. Applications structuring of rows/columns include clustering of data [555], phsytogenology, and theory of group decisions [723].

3.6. Additional Applications

We now give some additional references to more applications of TSP. Korostensky et al [513] used the traveling salesman problem to compute a near optimal *multiple sequence alignment* with guaranteed performance bound. In a related work Korostensky et al [514] used TSP based methods in constructing an *evolutionary tree* of optimal 'score'. The traveling salesman model is applicable in a variety of other situations including data analysis in psychology [454], X-Ray crystallography [111], overhauling gas turbine engines [670], warehouse order-picking problems [699], and wall paper cutting [348], to cite a few. Marchand et al [580] formulated an optimal control problem as a TSP. Potential application areas of their model include problems that arise in routing in telecommunication networks, test objective generation, and artificial intelligence [580].

4. Alternative representations of the TSP

We have introduced TSP as the problem of finding a least cost Hamiltonian cycle in a graph (graphical definition). We also discussed quadratic and linear permutation representations of TSP. In this section we study representations of the TSP as a mathematical programming problem.

4.1. Linear programming representation

This is perhaps the most well studied representation of the TSP. Let $\mathbb{F}$ be the family of all tours in G represented as a collection of incidence vectors in $\mathbb{R}^{|E|}$ and $P(\mathbb{F})$ be the convex hull of $\mathbb{F}$. Then the traveling salesman problem can be stated as the following mathematical programming problem:

$$\begin{array}{ll} \text{Minimize} & \sum_{j=1}^{m} c_j x_j \\ \text{Subject to} & X \in P(\mathbb{F}), \end{array}$$

where $X = (x_1, x_2 \ldots, x_m)$, m being the number of edges in G.

A complete linear inequality description of $P(\mathbb{F})$ is not known which somewhat limits application of linear programming techniques to solve TSP. However, it may be noted that linear programming based methods are among the best known approaches for solving TSP which exploits partial linear inequality description of $P(\mathbb{F})$. For the state of the art research in this area we refer to chapters 2 and 3.

4.2. Integer programming formulations

Let us first consider the case of the ATSP. It may be noted that the linear programming relaxation of the integer programming formulations discussed below could be strengthened by adding 'valid' inequalities corresponding to $P(\mathbb{F})$, especially those defining facets or higher dimensional faces of $P(\mathbb{F})$. For a discussion of such inequalities we refer to chapters 2 and 3. Most of the integer programming formulations of the ATSP are based on the assignment problem. A 0-1 programming formulation of the assignment problem can be described as follows:

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

Subject to

$$\sum_{i=1}^{n} x_{ij} = 1, \quad j = 1, \ldots, n,$$

$$\sum_{j=1}^{n} x_{ij} = 1, \quad i = 1, \ldots, n,$$

$$x_{ij} = 0 \text{ or } 1$$

For any solution $X = (x_{ij})_{n \times n}$ of the above assignment problem, consider the set $S = \{(ij) : x_{ij} = 1\}$. Clearly, S represents a tour or a collection of subtours in G. In order to model the ATSP as an integer program, we must include additional restrictions to make sure that S does not contain any subtours. These restrictions are called *subtour elimination constraints.*

4.2.1 Clique Packing Constraints. Introduced by Dantzig, Fulkerson, and Johnson [239], these are the most popular form of subtour elimination constraints. It states that from any clique Q of the subgraph G^* of G induced by the node set $V^* = \{2, 3, \ldots, n\}$, select at most $|Q| - 1$ arcs. There are an exponential number of such constraints which are given by

$$\sum_{i \in Q} \sum_{j \in Q} x_{ij} \leq |Q| - 1 \text{ for all } \emptyset \neq Q \subset \{2, 3, \ldots, n\}$$

The clique packing constraints can be written in an equivalent form as

$$\sum_{i \in Q} \sum_{j \in \bar{Q}} x_{ij} \geq 1 \text{ for all } \emptyset \neq Q \subset \{2, 3, \ldots, n\}$$

where $\bar{Q} = V - Q$.

4.2.2 Circuit Packing Constraints. This class of subtour elimination constraints were introduced by Grotschel and Padberg [401]. (See also [392].) These constraints can be described by replacing cliques Q in the clique packing constraints by circuits in G^*. Again there are an exponential number of circuit packing constraints.

The assignment problem together with clique packing or circuit packing constraints leads to a valid 0-1 programming formulation of the

ATSP. These formulations use only variables that correspond to the arcs and self-loops of the complete graph G and are called *natural formulations* [676]. It may me noted that the self-loop variables x_{ii} could be discarded from the formulation to reduce the problem size. In this case one needs to consider only cliques (circuits) of size two or more in the clique (circuit) packing constraints, further reducing the problem size. We have used the variables x_{ii} in our formulation in order to be consistent with the definition of the classical assignment problem. One could also choose $c_{ii} = M$, where M is a large positive number, to insure they will not appear any optimal solution to the assignment problem. In this case also it is sufficient to consider clique (circuits) of size two or more within the clique (circuit) packing constraints.

By using polynomial (in n) number of additional constraints and variables it is possible to ensure that S is cycle free. These are called *compact representations* of the subtour elimination constraints.

4.2.3 MTZ constraints. Miller, Tucker and Zemlin [595] showed that by using $(n-1)^2$ additional constraints and $n-1$ additional variables, it is possible to ensure that S is free of subtours. These constraints are given by

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

where u_i, $i = 2,3,\ldots,n$ are unrestricted real variables. If S contains any subtour (a cycle with less than n arcs) then S will contain a subtour $\tilde{T}$ that does not contain node 1. If $\tilde{T}$ is a single node, say $\{k\}$, then inequality (1) is violated for $i = j = k$. If $\tilde{T}$ contains more than one node, then adding the above inequalities for arcs in $\tilde{T}$ leads to a contradiction. This establishes that the MTZ constraints force S to be free of subtours.

4.2.4 Network flow constraints. Network flow based formulations can also be used to obtain compact representations of subtour elimination constraints. We first consider a general form of subtour elimination constraints and observe that most of the network flow based subtour elimination constraints studied in literature fall within this general framework. Consider the inequality system:

$$AY \leq b \tag{2}$$

$$Y \geq 0 \tag{3}$$

$$BX + DY \leq g, \tag{4}$$

where $X = (x_{11}, x_{12}, \ldots, x_{1n}, x_{21}, \ldots, x_{nn})$ is a solution to the assignment problem, $Y = (y_1, y_2, \ldots, y_p) \in \mathbb{R}^p$, A, B, D are matrices, and b

and g are column vectors. Each y_k is associated with an arc (i,j) of G. It is possible that y_k and y_r are associated with the same arc (i,j) for $k \neq r$, but no y_k is associated with two arcs. The inequalities within the system (2) and (3) are called *flow constraints* and the inequalities within the system (4) are called *coupling constraints*. For any feasible solution Y^* of the flow constraints, consider the subgraph G_{Y^*} induced by the arcs of G associated with positive components of Y^*. Then G_{Y^*} is called the *flow graph* of Y^*.

Theorem 2 *The assignment problem together with the flow and coupling constraints gives a valid mixed 0-1 programming formulation of the TSP if the following conditions are satisfied.*

(i) *If the assignment solution $X^0 = (x^0_{ij})$ represents a Hamiltonian cycle, then there exists $Y^0 \geq 0$ such that (X^0, Y^0) satisfies (2) and (4).*

(ii) *For any feasible solution (X^*, Y^*), the flow graph G_{Y^*} is connected.*

(iii) *For any feasible solution (X^*, Y^*), if $x^*_{ij} = 0$ then all y^*_k's associated with arc (i,j) are zeros.*

Proof. The assignment constraints insure that $S = \{(i,j) : x_{ij} = 1\}$ defines either a tour or a collection of subtours. In view of *(iii)*, if *(ii)* is satisfied, then S is free of subtours. The result follows from *(i)*. ■

Let us now consider a single commodity flow formulation suggested by Gavish and Graves [350] which satisfies the conditions of Theorem 2. Let y_{ij} be the 'flow' of commodity y along arc (i,j) in G. Consider the following constraints:

Flow constraints:

$$\sum_{j=1}^{n} y_{ij} - \sum_{j=1}^{n} y_{ji} = 1, \quad , i = 2, 3, \ldots, n \tag{5}$$

$$\sum_{j=1}^{n} y_{1j} = n - 1 \tag{6}$$

$$y_{ij} \geq 0 \tag{7}$$

Coupling constraints:

$$(n-1)x_{ij} - y_{ij} \geq 0 \text{ for } i, j = 1, 2, \ldots, n \tag{8}$$

It is easy to verify that these flow constraints and coupling constraints satisfy the conditions of Theorem 2 and hence are valid subtour elimination constraints. The coupling constraints (8) could be strengthened as

$$(n-1)x_{1j} - y_{1j} \geq 0 \text{ for } j = 2, \ldots, n \tag{9}$$

$$(n-2)x_{ij} - y_{ij} \geq 0 \text{ for } i, j = 2, 3, \ldots, n \tag{10}$$

Another slightly strengthened variation of the coupling constraints can be described as follows:

$$y_{j1} = 0, \text{ for } j = 2, 3, \ldots, n \tag{11}$$

$$y_{1j} = (n-1)x_{1j} \text{ for } j = 2, 3, \ldots, n \tag{12}$$

$$x_{ij} \leq y_{ij} \leq (n-2)x_{ij} \text{ for } i, j = 2, 3, \ldots, n \tag{13}$$

This variation is equivalent to the single commodity flow formulation proposed by Gouveia and Voß [392].

A two-commodity flow formulation of the subtour elimination constraints was proposed by Finke, Claus, and Gunn [291]. Let y and z be two commodities and y_{ij} be the flow of commodity y along arc (i,j) and z_{ij} be the flow of commodity z along arc (i,j). The resulting flow constraints are given by:

$$\sum_{j=1}^{n} y_{1j} - \sum_{j=1}^{n} y_{j1} = n-1 \tag{14}$$

$$\sum_{j=1}^{n} y_{ij} - \sum_{j=1}^{n} y_{ji} = -1, \quad , i = 2, 3, \ldots, n \tag{15}$$

$$\sum_{j=1}^{n} z_{1j} - \sum_{j=1}^{n} z_{j1} = -(n-1) \tag{16}$$

$$\sum_{j=1}^{n} z_{ij} - \sum_{j=1}^{n} z_{ji} = 1, \quad , i = 2, 3, \ldots, n \tag{17}$$

$$\sum_{j=1}^{n} y_{ij} + \sum_{j=1}^{n} z_{ij} = n-1, \quad , i = 1, 2, \ldots, n \tag{18}$$

$$y_{ij} \geq 0 \text{ for } i, j = 1, 2, \ldots, n \tag{19}$$

$$z_{ij} \geq 0 \text{ for } i, j = 1, 2, \ldots, n \tag{20}$$

with coupling constraints:

$$y_{ij} + z_{ij} = (n-1)x_{ij} \text{ for all } (ij). \tag{21}$$

It is not difficult to verify that these constraints satisfy the conditions of Theorem 2 and hence are valid subtour elimination constraints.

Using multicommodity flows, Wong [827] presented another compact representation of the subtour elimination constraints. He used $2(n-1)$ commodities y^k and z^k, $k = 2, \ldots, n$. Let y_{ij}^k be the flow of commodity y^k along arc (i,j) and z_{ij}^k be the flow of commodity z^k along arc (i,j), $i,j = 1, 2, \ldots n$ and $k = 2, 3, \ldots, n$. The flow constraints of Wong can be stated as follows.

$$\sum_{j=1}^{n}(y_{ij}^k - y_{ji}^k) = \begin{cases} 1 & \text{if } i = 1,\ k = 2,3,\ldots,n, \\ -1 & \text{if } i = k,\ k = 2,3,\ldots,n, \\ 0 & \text{if } i \neq 1 \text{ and } k,\ k = 2,3,\ldots,n \end{cases} \tag{22}$$

$$\sum_{j=1}^{n}(z_{ij}^k - z_{ji}^k) = \begin{cases} 1 & \text{if } i = 1,\ k = 2,3,\ldots,n, \\ -1 & \text{if } i = k,\ k = 2,3,\ldots,n, \\ 0 & \text{if } i \neq 1 \text{ and } k,\ k = 2,3,\ldots,n \end{cases} \tag{23}$$

with coupling constraints:

$$0 \le y_{ij}^k \le x_{ij} \quad \text{for all } i,j,k \tag{24}$$

$$0 \le z_{ij}^k \le x_{ij} \quad \text{for all } i,j,k. \tag{25}$$

Note that the source node for commodity y^k is 1 for all k and for commodity z^k, it is node k. Similarly, the sink node of commodity y^k is node k and for commodity z^k it is node 1 for all k. Constraints (22) and (23) insures that one unit of commodity y^k travels from node 1 to node k while one unit of commodity z^k travels from node k to node 1. Thus the corresponding flow graph is strongly connected, establishing condition (i) of Theorem 2. Conditions (ii) and (iii) of Theorem 2 are easy to verify. Thus constraints (22) to (25) represent valid subtour elimination constraints.

Eliminating the z_{ij} variables from the above formulation, Claus [202] presented a multicommodity flow formulation of the subtour elimination constraints as described below:

Flow constraints:

$$\begin{aligned}
\sum_{i=1}^{n} y_{1i}^{k} &= 1 \text{ for } k = 2,3,\ldots,n && (26)\\
\sum_{i=1}^{n} y_{i1}^{k} &= 0 \text{ for } k = 2,3,\ldots,n && (27)\\
\sum_{i=1}^{n} y_{ik}^{k} &= 1 \text{ for } k = 2,3,\ldots,n && (28)\\
\sum_{i=1}^{n} y_{ki}^{k} &= 0 \text{ for } k = 2,3,\ldots,n && (29)\\
\sum_{i=1}^{n} y_{ij}^{k} - \sum_{i=1}^{n} y_{ji}^{k} &= 0 \text{ for } j,k = 2,3,\ldots,n; j \neq k && (30)\\
y_{ij}^{k} &\geq 0 && (31)
\end{aligned}$$

Coupling constraints:

$$y_{ij}^{k} \leq x_{ij} \text{ for } i,j = 1,2,\ldots,n; k = 2,3,\ldots,n \qquad (32)$$

Other related multicommodity flow formulations of the TSP which are variations of Wong [827] are proposed by Langevin [531] and Loulou [567]. The linear programming (LP) relaxations of all these multicommodity formulations yield the same objective function value as that of the LP relaxation of Dantzig et al [239]. However, LP relaxations of the single and two commodity formulations yield weaker bounds. Other well studied formulations of TSP include time-stage dependent formulations [637]. For a comparative study of LP relaxations of various (mixed) integer programming formulations of ATSP we refer to [392, 532, 637, 643].

4.3. Symmetric TSP

Since the symmetric traveling salesman problem is a special case of ATSP (as discussed in Section 1), the above formulations are valid for STSP also. However, exploiting symmetry, we could replace the assignment constraints by 2-matching constraints. Let $\delta(v)$ be the set of edges incident on node v of G. Then a generic integer programming formulation of STSP is given by

$$\text{Minimize} \sum_{e \in E} c_e x_e$$

Subject to

$$\sum_{e \in \delta(v)} x_e = 2, \quad v \in V$$

$$x_e = 0 \text{ or } 1, \; e \in E$$

$$\{e \in E : x_e = 1\} \text{ contains no subtours.}$$

As discussed earlier, additional constraints can be used to make sure that $\{e \in E : x_e = 1\}$ is free of subtours. For other integer programming formulations of the STSP we refer to [39, 41, 169, 42, 40]

4.4. Binary quadratic programming formulation

We have seen that by adding new constraints to the assignment problem (2-matching problem), various mixed integer linear programming formulation for the ATSP (STSP) can be obtained. While the assignment constraints represent all permutations of $\{1, 2, \ldots, n\}$, the additional constraints (subtour elimination constraints) make sure that only cyclic permutations are selected. We have seen in Section 1 that under the quadratic permutation representation of TSP, any pemutation is feasible and hence the assignment constraints are good enough in this case. However the objective function now is no longer linear. The quadratic permutation representation of TSP yields the following binary quadratic programming representation of TSP:

$$\text{Minimize} \sum_{i,j,k=1}^{n} d_{ij} x_{ik} x_{j(k+1)}$$

subject to the assignment constraints, where $x_{ij} = 1$ is interpreted as 'the j^{th} city visited is i' and the index $n + 1 \equiv 1$. This problem is represented sometimes using a cyclic permutation matrix. Let T be a cyclic permutation matrix of order n, i.e. its $(i, j)^{th}$ element t_{ij} is given by $t_{i,i+1} = 1$, for $i = 1, \ldots n - 1$, $t_{n1} = 1$, and all other entries zero. Now define $d_{ijkl} = c_{ij} t_{kl}$. Then TSP is equivalent to the 0-1 quadratic programming problem [545]

$$\text{Minimize} \sum_{i,j,k,l=1}^{n} d_{ijkl} x_{ik} x_{jl}$$

subject to the assignment constraints.

4.5. Three Matroid Intersection

Let E be a finite set and F be a family of subsets of E. $M = (E, F)$ is a *matroid* if (1) $Y \in F$, $X \subset Y$ implies $X \in F$. (2) if $|X| < |Y|$ and $X, Y \in F$ implies there exists an $e \in Y - X$ such that $X \cup \{e\} \in F$. Let $S_1, S_2, \ldots, S_p$ be a partition of E and F be the collection of all subsets of E with at most one element from each S_i, $i = 1, 2, \ldots, p$. Then $M = (E, F)$ is called a *partition matroid.* If E is the arc set of a directed graph G and S_i contains all arcs coming into node i of G generates a partition matroid. Similarly if S_i is the collection of all arcs going out of node i in G we have another partition matroid. If F contains the collection of all 1-forests (a forest with one additional arc) in G, then $M = (E, F)$ is a matroid called the *1-graphic* matroid. Any collection of n edges of G that is common to the two partition matroids defined above and the 1-graphic matroid defines a Hamiltonian cycle in G. Thus the ATSP can be formulated as a minimum cost three matroid intersection problem. Since the STSP is a special case of the ATSP, it is also a three matroid intersection problem.

5. Matrix Transformations

Let us now discuss some useful transformations of distance matrices associated with a TSP that leaves an optimal solution invariant. Let $C = (c_{ij})$ and $D = (d_{ij})$ be two $n \times n$ matrices associated with a TSP on G. For any Hamiltonian cycle H of G, for $\alpha > 0$ and $\beta \in \mathbb{R}$, if

$$\sum_{ij \in H} d_{ij} = \alpha(\sum_{ij \in H} c_{ij}) + \beta$$

then we say that C and D are *equivalent* with respect to TSP. Let $\phi : \mathbb{R} \longrightarrow \mathbb{R}$ such that $\phi(c_{ij}) = d_{ij}$. If C and D are equivalent with respect to TSP, then ϕ is called an *equivalent matrix transformation*(EMT). Thus under an equivalent matrix transformation, the optimal solution of the TSP remains unchanged. For any $\gamma > 0$ and $\delta \in \mathbb{R}$, $d_{ij} = \gamma c_{ij} + \delta$ is an EMT. Similarly for $a_i, b_i \in \mathbb{R}$ and $\gamma > 0$, $d_{ij} = \gamma c_{ij} + a_i + b_j$ is an EMT which is perhaps the most popular EMT studied for TSP. However, under this EMT, the matrix D need not be symmetric even if C is symmetric. By choosing $a_i = b_i$ the EMT leaves symmetry of a matrix invariant. Another important structural property of a matrix useful in the study of TSP is the *parameterized triangle inequality*(τ-triangle inequality) [22, 117]. We say that the matrix C satisfies the τ-triangle inequality [22, 117] if

$$c_{ij} \leq \tau(c_{ik} + c_{kj}), \text{ for all } i \neq j \neq k$$

If $\tau = 1$, the we say that the matrix C satisfy the *triangle inequality.* If $\tau = 1/2$ all elements of C are equal. Thus for $1/2 \leq \tau < 1$, τ-triangle inequality is a restriction on the triangle inequality while for $\tau > 1$ it is a relaxation on the triangle inequality. Under the EMT $d_{ij} = c_{ij} + a_i + b_j$ the matrix D need not satisfy τ-triangle inequality even if C satisfies the τ-triangle inequality for moderate values of τ. However, for any $1/2 < \tau$ it is possible to choose values of a_i and b_i such that D satisfies the τ-triangle inequality even if C does not satisfy the τ-triangle inequality. To achieve this choose a_i's and b_i's such that

$$(\tau - 1)(a_i + b_j) + \tau(a_k + b_k) \geq c_{ij} - \tau(c_{ik} + c_{kj}) \ \forall \ i \neq j \neq k \qquad (33)$$

For $\tau > 1/2$ the above inequality is always feasible since $a_i = b_i = M$ is a solution for large enough M. If one prefers smaller values of a_i's and b_i's, a minimization linear program with objective function $\sum(a_i + b_i)$ subject to (33) as constraints, can be solved. Thus using an EMT, any cost matrix can be reduced to an equivalent cost matrix satisfying τ-triangle inequality for any $\tau > 1/2$.

A cost matrix C for a TSP is said to be in *standard reduced form* if $c_{nj} = c_{jn} = 0$, $j = 1, 2, \ldots n-1$ and $c_{12} = 0$

Theorem 3 *[489] For any cost matrix C, there exists an EMT which produces a standard reduced matrix.*

Proof. For $i = 1, 2, \ldots n-1$ define $a_i = c_{in} + x/2$ and $b_i = c_{ni} + x/2$ where $x = c_{12} - c_{1n} - c_{n2}$. Let $a_n = b_n = \frac{-x}{2}$. Now the matrix D obtained by the EMT $d_{ij} = c_{ij} - a_i - b_j$ will be a standard reduced matrix. If C is symmetric, it can be verified that D is also symmetric. ∎

Using cost matrix of a TSP in standard reduced form allows simplification of some proofs [489]. (See also Chapter 11). As a consequence of the above theorem, we could reduce the cost of all arcs (incoming and outgoing) incident on a given node i, to zero as well as the cost of another arc not incident on i leaving the optimal solution invariant.

6. More Variations of the TSP

We now discuss additional variations of TSP studied in the literature. Our discussion on these variations are confined to their definitions only. For more details on these problems, see the corresponding references cited. Further, the references we cite need not be the paper in which the problem was originally introduced and the reference list is not exhaustive. Through out this section G is a complete graph (digraph) with c_e as the cost of edge e in G.

The time dependent TSP: For each arc (i,j) of G, n different costs c^t_{ij}, $t = 1, 2, \ldots, n$ are given. The cost c^t_{ij} corresponds to the 'cost' of going from city i to city j in time period t. The objective is to find a tour $(\pi(1), \pi(2), \ldots \pi(n), \pi(1))$, where $\pi(1) = 1$ corresponds to the home location which is in time period zero, in G such that $\sum_{i=1}^{n} c^i_{\pi(i)\pi(i+1)}$ is minimized. The index $n+1$ is equivalent to 1. For all (i,j), if $c^1_{ij} = c^2_{ij} = \cdots = c^n_{ij}$ then the time dependent TSP reduces to the traveling salesman problem. For details on this problem, we refer to [319, 392, 669].

Period TSP: This generalization of TSP considers a k-day planning period, for given k. We want to find k cycles in G such that each node of G is visited a prescribed number of times during the k-day period and the total travel cost for the entire k-day period is minimized [177].

The delivery man problem: This problem is also known as the *minimum latency problem* and the *traveling repairman problem.* Let H be a tour in G and v_1 be a starting node. For each vertex v_i of G, define the *latency* of v_i with respect to H, denoted by $L_i(H)$, is the total distance in H from v_1 to v_i. The delivery man problem is to find a tour H^* in G such that $\sum_{i=1}^{n} L_i(H^*)$ is as small as possible. The DMP is strongly NP-complete. For details on this problem, we refer to [1, 385, 599, 293, 572]. It can be verified that this problem is a special case of the time dependent TSP.

Black and White TSP: This problem is a generalization of the TSP. Here, the node set of G is partitioned in to two sets, B and W. The elements of B are called *black nodes* and the elements of W are called *white nodes.* A tour in G is said to be feasible if the following two conditions are satisfied. (i) The number of white nodes between any two consecutive black nodes should not exceed a positive integer I and the distance between any two consecutive black nodes should not exceed a positive real number R. The black and white TSP is to find a minimum cost feasible tour in G. Applications of this problem include design of ring networks in the telecommunication industry [133]. A variation of the black and white TSP with applications in the air-line industry, known as TSP with replenishment arcs, has been discussed in [577].

Angle TSP: Let $v_1, v_2, \ldots, v_n$ be n points of the Euclidean plane. Consider a two edge path from v_i to v_k passing through v_j. The 'cost' of passing through v_j is defined as the angle between the vectors $\overrightarrow{v_iv_j}$ and $\overrightarrow{v_jv_k}$. For a tour H through $v_1, v_2, \ldots, v_n$, let $v_i(H)$ be the cost of passing through v_i. Then the angle TSP is to find a tour H that minimizes $\sum_{i=1}^{n} v_i(H)$. The angle TSP is NP-hard.

Film-copy Deliverer Problem: For each node i of G, a non-negative integer p_i is prescribed. We want to find a routing of a film-copy deliverer, starting at node 1, visit each node exactly p_i times and come back to node 1 in such a way that the total distance travelled is minimized [182]. This problem generalizes TSP and some of its variations. A generalization of the Film-Copy Deliverer problem has been studied in [549]. In this case, a minimum cost cycle in G is sought that passes through each node i of G at least p_i times and at most q_i times.

The selective TSP: [355] This problem is also known as the orienteering problem (see Chapter 13). For each node i of G, a weight w_i is given. Then the selective traveling salesman problem is to find a cycle Y in G such that the sum of the weights of nodes in Y is maximized while keeping $\sum_{ij \in Y} c_{ij} \leq L$, where L is a given number. This problem is closely related to the prize collecting traveling salesman problem studied in Chapter 14.

Resource constrained TSP: For each edge e of G a requirement r_{ij} is also given in addition to its cost. Then the resource constrained traveling salesman problem is to find a minimum cost tour H in G such that $\sum_{ij \in H} c_{ij} \leq K$ where K is a given constant [663].

Serdyukov TSP: Let $X_1, X_2 \ldots, X_n$ be a collection of subsets of the node set of G such that $|X_i| \leq k$ for a given constant k. Starting from a city in X_1, we want to find a minimum cost tour in G such that the i^{th} city visited must be from X_i [752]. For this problem we assume that G need not be complete. If d is the maximum vertex degree in G, then we call the resulting problem a (k, d)-problem. Serdyukov [752] showed that the (k, d)-problem is NP-hard for all $k \geq 2$, $d \geq 3$.

Ordered Cluster TSP: [25] This problem is a simplified version of the clustered TSP and is defined as follows. The node set of G is partitioned into k clusters $X_1, X_2, \ldots, X_k$ where X_1 is singleton representing the home location. The salesman starting and ending at the home location must visit all nodes in cluster X_2, followed by all nodes in cluster X_3, and so on such that the sum of the edge costs in the resulting tour is minimized.

Precedence Constrained TSP: Let B be a subset of $V \times V$ defined by $B = \{(i,j) : i \in V, j \in V, i,j \neq 1, \text{ and } i \text{ must be visited before } j\}$ represents the precedence constraints. Then the precedence constrained TSP is find a least cost TSP tour starting at node 1, visits each node of G exactly once and returns back to node 1 in such a way that the precedence constraints B are satisfied [71]. A special case of this problem is the dial-a-ride problem [674].

k-Peripatetic Salesman Problem: [240] A k-peripatetic salesman tour in G is the collection of k edge disjoint Hamiltonian cycles in G. The objective is to find a least cost k-peripatetic salesman tour in G. This problem has applications in network design.

Covering Salesman Problem: [233] In this case, we want to find a minimum cost cycle L in G such that the distance between L and any node i of G not on L, denoted by $d(L,i)$, is within a prescribed limit, where $d(L,i) = \min\{c_{ij} : j \text{ is a node of } L\}$.

TSP with time windows: [44, 45] For each arc (i,j) of G, let c_{ij} be the cost and t_{ij} be the traversal time. For each node i of G a triplet $\{a_i, b_i, s_i\}$ is given where $a_i \leq b_i$ and $0 \leq s_i \leq b_i - a_i$. The quantity s_i represents the service time at node i and the job at node i must be completed in the time interval $[a_i, b_i]$. If the salesman arrives at node i before a_i, he will have to wait until a_i. Then TSP with time windows is to find a least cost tour in G satisfying the restriction discussed above.

Moving Target TSP: A set $X = \{x_1, x_2, \ldots, x_n\}$ of n objects placed at points $\{p_1, p_2, \ldots, p_n\}$. Each object x_i is moving from $p_i \in \mathbb{R}^n$ at velocity v_i. A pursuer starting at origin moving with speed v wants to intercept all points $x_1, x_2, \ldots x_n$ in the fastest possible time [447]. This problem is related to the time dependent TSP.

Remote TSP: For a given integer k, we are interested in finding a subset S of V with $|S| = k$ such that the minimum cost of a tour in the subgraph of G induced by S is maximized [430].

There are several other variations that one could discuss depending on various application scenarios. These variations include traveling salesman location problem, traveling salesman problem with backhauls [353], k-best TSP [800], minmax TSP [767], multi-criteria TSP [271], stochastic TSP (different variations), k-MST [113, 601], minimum cost biconnected spanning subgraph problem [604], graph searching problem [519],

k-delivery TSP [175], quadratic assignment problem [545] etc., and combinations of one or more of the variations discussed here. Vehicle routing problems form another very important class of generalizations of TSP which are not discussed here.

Acknowledgement: I am thankful to G.Gutin and K.P.K. Nair for their comments on an earlier version of this chapter. Special thanks are due to S.N. Kabadi and H. Marchand for several discussions. This work was partially supported by the NSERC grant OPG0170381.