uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Geetha Patil

AUTEUR DE LA THÈSE / AUTHOR OF THESIS

Master of Computer Science

GRADE / DEGREE

School of Information Technology and Engineering

FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Compatible Tours for the Asymmetric Traveling Salesman Problem

TITRE DE LA THÈSE / TITLE OF THESIS

Sylvia Boyd

DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Lucie Moura

Brett Stevens

Gary W. Slater

LE DOYEN DE LA FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTDOCTORALES /
DEAN OF THE FACULTY OF GRADUATE AND POSTDOCORAL STUDIES

Compatible Tours for the Asymmetric Traveling Salesman Problem

Geetha Patil

Thesis submitted to the
Faculty of Graduate Studies and Postdoctoral Studies
in partial fulfillment of the requirements
for the Master's in Computer Science[1]

School of Information Technology and Engineering
Faculty of Engineering
University of Ottawa

[1]The M.Sc. Program is a joint program with Carleton University, administered by the Ottawa-Carleton Institute of Computer Science

Your file Votre référence
ISBN: 0-494-11378-2
Our file Notre référence
ISBN: 0-494-11378-2

Abstract

The Asymmetric Travelling Salesman Problem (ATSP) is the problem of finding a minimum cost directed Hamiltonian cycle in a complete weighted directed graph with n nodes. A well-known relaxation of this problem is the Asymmetric Subtour Elimination Problem (ASEP). The ASEP provides a good lower bound for the ATSP, and one might wonder if some of the information from the ASEP solution could be used in a useful way to help obtain good solutions for the ATSP. In this thesis, we consider this question through the study of a problem called the Compatible Tour Problem(CTP).Given an optimal ASEP solution x^*, a compatible tour for x^* is a directed Hamiltonian cycle that has exactly one edge coming in and one edge going out of the node set (henceforth called a *tight set*) of every tight cut of value 2 of x^*. The *compatible tour problem* is the problem of finding a minimum cost compatible tour for x^*, where x^* is the optimal solution of the ASEP. Note that it is suspected that CTP is an NP-hard problem.

We first consider the relationship between the CTP and ATSP from a theoretical point of view. Given an edge cost vector c, let ASEP(c), ATSP(c) and CTP(c) represent the corresponding optimal solution values for ASEP, ATSP and CTP, respectively. We know that ASEP(c) $\leq$ ATSP(c) $\leq$ CTP(c). A natural theoretical question is to wonder about the worst–case ratio between CTP(c) and ASEP(c) and between CTP(c) and ATSP(c). In this thesis, we study the worst-case ratios between CTP(c) and ASEP(c) and between CTP(c) and ATSP(c) for a particular node size

n and metric costs, and explain how are able to find their exact values for $4 \leq n \leq 7$ and a lower bound for their values for $8 \leq n \leq 10$.

We also consider the CTP from a more practical point of view. We consider the problem of finding a minimum cost "almost" compatible tour for a given solution x^* for ASEP. The "almost" compatible tour satisfies important tight sets in x^* which are nested with respect to each other. We then develop a heuristic to find this tour tour as a heuristic solution to the ATSP which is based on the well-known cheapest insertion heuristic for the symmetric travelling salesman problem. Finally, we test this heuristic and give the results.

Acknowledgements

It is my pleasure to thank all the people who helped me during both research and writing of this thesis.

My sincere appreciation goes to my supervisor, Professor Dr. Sylvia Boyd, for her enthusiasm, inspiration, and ability to explain things clearly. Throughout my thesis writing period, she provided encouragement, sound advice, good teaching, and lots of good ideas. I would have been lost without her.

I am indebted to my many student colleagues for providing a stimulating and fun environment to learn and grow. I am especially grateful to Karen Meagher and Paul Elliott-Magwood for their guidance while writing this thesis.

I wish to thank my daughter Prerana for being my inspiration. Lastly, but most importantly I thank my husband Raju, my parents Lalitha and Shankrappa Desai, brother Rathan and sisters Joshila and Priya for supporting me throughout my study years. To my beloved parents I dedicate this thesis.

♡ Geetha Patil

Contents

Chapter 1

Introduction

Definition 1.1 *The Travelling Salesman Problem (TSP) is the following minimization problem: Given a set of cities and a vector whose entries are interpreted as the non-negative distance from one city to another, find the minimum distance tour starting and ending in the same city and visiting every city exactly once.*

The TSP is one of the most famous and well-studied NP-complete problem. For any two cities A and B, the distance from A to B and B to A is same in the case of the Symmetric Travelling Salesman Problem (STSP), but it may be different in the Asymmetric Travelling Salesman Problem (ATSP). In the ATSP, it is possible to get different tour lengths if you reverse the order in which cities are visited.

The ATSP can be modelled as the problem of finding a minimum cost directed Hamiltonian cycle in a complete weighted directed graph G=(V, E), where n is the number of nodes, $V = \{1 \dots n\}$ is the set of nodes and $E = \{(i,j) |\ i,j \in V, i \neq j\}$ is the set of directed edges. Each node represents a city and an edge represents a route from one city to another. The cost vector $c \in R^E$ is associated with the edge set E, where c_{ij} represents the cost of going from city i to j. Each ATSP instance with n cities can be specified as a vector of length $n(n-1)$ (whose components, indexed by

the edges of the complete graph, specify the corresponding costs) and a tour is a 0 - 1 incidence vector indexed by the edge set E, i.e., each tour through the n cities can be represented as its incidence vector of length $n(n-1)$ (with each component set to 1 if the corresponding edge is a part of the tour, and set to 0 otherwise); if c denotes the cost vector and if $\mathcal{S}$ denotes the set of the incidence vectors of all the tours, then the ATSP can be stated as

$$\text{minimize } cx \text{ subject to } x \in \mathcal{S}.$$

If the cost vector c satisfies the triangle inequality then it is called a *metric* ATSP, i.e., if $c_{ij} + c_{jk} \geq c_{ik}$ for all distinct $i,\ j,\ k \in V$.

In 1954, Dantzig, Fulkerson and Johnson have shown that the STSP can be formulated as the following integer linear program [10]:

$$\begin{aligned}
&\text{Minimize} && cx && && (1)\\
\text{subject to:}\quad & x(\delta(v)) && = && 2 \text{ for all } v \in V, && (2)\\
& x(\delta(S)) && \geq && 2 \text{ for all } S \subset V \text{ where } 3 \leq |S| \leq \lfloor n/2 \rfloor, && (3)\\
& 0 \leq x_e && \leq 1 && \text{for all } e \in E, && (4)\\
& && x && \text{is integer.} && (5)
\end{aligned}$$

Based on the Integer Linear Programming(ILP) formulation of STSP, the ILP for the ATSP can be formulated as follows:

$$\text{Minimize} \quad cx \tag{6}$$

$$\text{subject to:} \quad x(\delta^{in}(v)) = 1 \text{ for all } v \in V, \tag{7}$$

$$x(\delta^{out}(v)) = 1 \text{ for all } v \in V, \tag{8}$$

$$x(\delta^{out}(S)) \geq 1 \text{ for all } S \subset V \text{ where } 2 \leq |S| \leq \lfloor n/2 \rfloor, \tag{9}$$

$$x_e \geq 0 \quad \text{for all } e \in E, \tag{10}$$

$$x \quad \text{is integer.} \tag{11}$$

For $S \subset V$, $x(\delta^{out}(S))$ represents total value on edges going out of S and $x(\delta^{in}(S))$ represents the sum of the values x_e on edges e directed into S. The constraints (7) and (8) force every node to have in-degree and out-degree 1 and they are called *degree constraints.* The constraints of type (9) force the directed tour to pass through all the nodes and they are called *subtour elimination constraints* or *cut constraints.* The constraint (10) and (11) force x_e to be 0 or 1.

The output from this model is the vector x which will be a 0-1 incidence vector of a tour of graph G

When we solve the ILP for the ATSP with a non-negative metric cost vector, we obtain a minimum cost tour x. However, solving the above model with as few as 40 nodes can be beyond the abilities of even the most sophisticated computers [17]. While such small instances somewhat artificial, most real problems with more than 100 or so nodes are not possible to solve unless they have some specific exploitable structure. Note however, that there are a tremendous number of constraints: for a 20-city problem, that number is roughly 524,288. For a 300-city problem, this would amount to $1.01 * 10^{90}$ constraints [17].

One direction which seems promising is finding good approximation (k-approximate) algorithms for the TSP, i.e. trying to find a heuristic for the TSP which finds a tour which is guaranteed to be of cost at most k times the cost of the optimal tour for

some constant $k \geq 1$. It is known that for both the STSP and ATSP the existence of an approximation algorithm for general costs would imply P=NP [14]. Despite the efforts of many researchers there is no known constant approximation algorithm for the metric ATSP, however for the metric STSP there is a well-known 3/2-approximation algorithm developed by Christofides [7] in 1976. Currently best known polynomial-time algorithm for approximating the metric ATSP was developed by Frieze, Galbiati and Maffioli [12]; their algorithm is a $\log n$ approximation.

One way of finding good lower bounds for the TSP and possibly an approximate solution is by studying the LP-relaxation of the corresponding ILP.
In the case of STSP, the LP-relaxation can be obtained by dropping the integer requirement (5) and it is called the *Subtour Elimination Problem* (henceforth called SEP). The linear program formulation of the SEP is:

$$\begin{aligned}
& \text{Minimize} && cx && && (12)\\
\text{subject to:} \quad & x(\delta(v)) && = && 2 \text{ for all } v \in V, && (13)\\
& x(\delta(S)) && \geq && 2 \text{ for all } S \subset V \text{ where } 2 \leq |S| \leq \lfloor n/2 \rfloor, && (14)\\
& 0 \leq x_e && \leq 1 && \text{for all } e \in E. && (15)
\end{aligned}$$

The associated *subtour polytope*, which is denoted by S^n, is defined by

$$S^n = \{x \in R^E |\ x \text{ satisfies (13), (14) and (15)}\}. \tag{16}$$

In case of an ATSP, the LP-relaxation can be obtained by dropping the integer requirement (11) and it is called the *Asymmetric Subtour Elimination Problem* (henceforth

called ASEP). The linear programming formulation of the ASEP is:

$$\begin{aligned}
& \text{Minimize} && cx && && (17)\\
\text{subject to:} \quad & x(\delta^{in}(v)) && = && 1 \text{ for all } v \in V, && (18)\\
& x(\delta^{out}(v)) && = && 1 \text{ for all } v \in V, && (19)\\
& x(\delta^{out}(S)) && \geq && 1 \text{ for all } S \subset V \text{ where } 2 \leq |S| \leq \lfloor n/2 \rfloor, && (20)\\
& x_e && \geq 0 && \text{for all } e \in E. && (21)
\end{aligned}$$

The associated *asymmetric subtour polytope*, which is denoted by Q_S^n, is defined by

$$Q_S^n = \{x \in R^E | x \text{ satisfies (18), (19), (20) and (21)}\}. \quad (22)$$

For a given optimal solution x^* of the ASEP, we formulate a new problem called the *Compatible Tour Problem* (henceforth called CTP), which is the problem of finding a minimum cost tour compatible with x^*. A tour is called *compatible with* x^* if that tour has exactly one edge coming in and one edge going out of every node set S for which $x^*(\delta^{out}(S)) = 1$. So, a compatible tour is one which is forced to be "tight" in all same places as the solution x^* for ASEP. Although this problem has not been studied for ATSP, it has been studied for STSP [16]. In the case of STSP, it is suspected that it is an NP-hard problem and considered unlikely that an efficient solution method will be found for it. We also suspect CTP is an NP-hard problem in the case of ATSP and trying to develop a heuristic to solve it.

Given an edge cost vector $c \in R^E$, let ASEP(c) and ATSP(c) represent the corresponding optimal solution values for ASEP and ATSP respectively, and let CTP(c) represent the CTP optimal solution value with respect to some extreme point x^* of ASEP which optimises c over ASEP. Thus we have ASEP(c) $\leq$ ATSP(c) $\leq$ CTP(c). For an arbitrary cost function c the values of ASEP(c) and ATSP(c) can be arbitrarily far apart. However for metric cost functions, i.e. cost functions that satisfy the

triangle inequality $c_{ij} + c_{jk} \geq c_{ik}$ for all distinct $i,\ j,\ k \in V$, ASEP(c) and ATSP(c) seem to be very close in practice (see [4]). In [6] and [4], the worst–case ratio

$$\max_{c \in R^E} \left(\frac{ATSP(c)}{ASEP(c)} \right)$$

has been studied for non-negative metric cost functions c and it was shown in [4] and [6] that this ratio can be as large as 2, although for small ATSP problems the ratio is close to one (see [4]).

Since we know that the LP-relaxation ASEP(c) provides a good lower bound for ATSP(c) in practice for metric costs, we wondered if CTP(c) would give a good tour solution for ATSP(c) since it is a tour that is forced to retain some of the structure of the ASEP optimal solution. Thus the main objective of this thesis is to investigate the "goodness" of the CTP tours from both a theoretical and practical point of view.

From a theoretical point of view, we investigate how far apart CTP(c) can be from ASEP(c) and ATSP(c) in the worst–case. To do this, we study the worst–case ratio

$$\max_{c \in R^E} \left(\frac{CTP(c)}{ASEP(c)} \right)$$

as well as the worst–case ratio

$$\max_{c \in R^E} \left(\frac{CTP(c)}{ATSP(c)} \right)$$

for $c \geq 0$ and c metric.

In particular, we find the exact value for the latter ratio for the ATSP problems with n nodes where $4 \leq n \leq 7$ and a lower bound when $n = 8$. In the case of the former ratio, we find the exact worst–case ratio for small ASEP problems on n nodes where $4 \leq n \leq 7$ and a lower bound on the ratio for $8 \leq n \leq 10$.

From a practical point of view, we develop a new heuristic for the ATSP which is based on the CTP in that it attempts to find a solution for CTP(c). This heuristic is based on the cheapest insertion heuristic for the STSP, and finds an "almost"

compatible tour satisfying a subset $\mathcal{TS}$ of tight sets in an optimal solution x^* of ASEP which corresponds to the given ATSP. The set $\mathcal{TS}$ of tight sets is nested inside one another and if our tour $\overline{x}$ satisfies $\overline{x}(\delta^{out}(S)) = 1$ for all $S \in \mathcal{TS}$, then $\overline{x}$ is a "almost" compatible tour for x^*. We test this heuristic on a number of the real-world ATSP test problems posted in the TSP library [17], and report these results.

A summary of the content of this thesis is as follows. In Chapter 2, we introduce the background notation and theory required for the thesis. For the convenience of the reader, we have included in this chapter some of the definitions which have already been stated.

In Chapter 3, we explain about the compatible tours, finding all tight sets in solution x^* of the ASEP and define an ILP for the CTP. In Chapter 4, we describe the method used to find the exact worst–case ratio between the CTP(c) and ASEP(c) for small n, and in Chapter 5, we describe the method used to find the exact worst–case ratio between the ATSP(c) and CTP(c) for small n. In Chapter 6, of this thesis, we describe the heuristic developed by us to solve ATSP that is based on the CTP(c) and the cheapest insertion heuristic for the STSP. In Chapter 7, we present the pseudo-code and implementation details of all the algorithms we developed in the process of developing this heuristic. In Chapter 8, we explain how we tested our heuristic after finding required input data that are optimal solution x^* of ASEP which corresponds to the given ATSP and all tight sets in x^*. Finally, in Chapter 9, we give our conclusions and some ideas for future work.

Chapter 2

Definitions, Notation and Background

A *directed graph* or *digraph* G consists of a set of nodes V and a set of directed edges E, where an *edge* is an ordered pair (u, v) of distinct nodes and it is denoted by G=(V, E). For the directed edge (u, v), the node u is called the tail and node v is called the head.

Definition 2.1 *(Triangle Inequality)*
In a given complete digraph G with edge cost vector c, if $c_{ij}+c_{jk} \geq c_{ik}$ *for all distinct* $i,\ j,\ k \in V$, *we say c satisfies the* triangle inequality.

Definition 2.2 *(Metric ATSP)*
If the cost vector c for an ATSP(TSP) satisfies the triangle inequality then it is called a metric *ATSP(TSP).*

Definition 2.3 *(Polyhedron)*
A polyhedron $P \subseteq R^E$ *is the solution set of a finite system of linear inequalities and equalities. It can be expressed in the form* $P = \{x \in R^E | Ax = b, Dx \leq d\}$.

Definition 2.4 *(Polytope)*

A polytope *is a polyhedron which is bounded.*

Definition 2.5 *(Face)*

A subset F of a polyhedron P is a face *of P if F is either the empty set, P or else the polyhedron obtained by taking the linear system which defines P and replacing some of the inequalities with the corresponding equalities.*

Definition 2.6 *(Cut)*

For any edge set $F \subseteq E$ and $x \in R^E$, let x(F) denote the sum $\sum_{e \in F} x_e$. For any node set W, let $\delta^{out}(W)$ denote $\{(u,v) \in E \mid u \in W, v \in V/W\}$. A cut *in a directed graph is a set of directed edges, $\delta^{out}(W)$, where $\emptyset \subset W \subset V$.*

Definition 2.7 *(Extreme Point)*

An extreme point x^* *is a face of a non-empty polyhedron which consists of a single point in R^E. In this thesis, more specifically an extreme point is the unique feasible solution of an ASEP obtained after setting some of the inequality constraints to equality.*

Throughout this thesis, x will always denote a vector, with components indexed by the edges of the complete directed graph on node-set V, that satisfies

$$0 \leq x_e \leq 1 \qquad \text{for all edges } e \in E \tag{23}$$

$$x(\delta^{in}(v)) = 1 \qquad \text{for all nodes } v \in V \tag{24}$$

$$x(\delta^{out}(v)) = 1 \qquad \text{for all nodes } v \in V \tag{25}$$

The x is called a *half-integer* extreme point if all the nonzero components of the vector x are either $1/2$ or 1.

Definition 2.8 *(Support Graph)*

The support graph *of x, denoted by G_x, is the graph whose node set is V(G), and whose edge set consists of all edges $e \in E(G)$ such that $x_e \neq 0$.*

For all support graphs drawn in figures in this thesis the pictorial representation of each edge will denote the edge value. In Figure 1, we show all different edge styles and corresponding edge values.

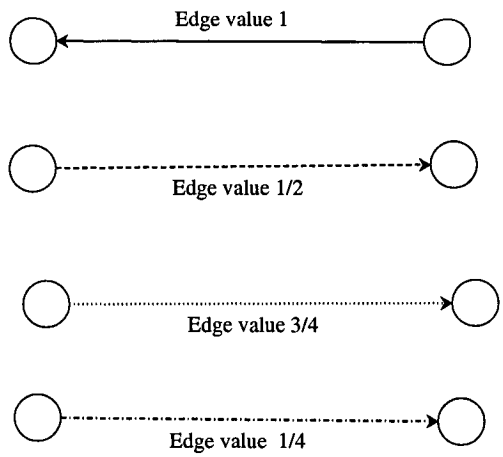

Figure 1: All different edge styles and corresponding edge values

Definition 2.9 *(Edge Value of the Cut)*

The edge value of a cut $\delta^{out}(W)$ *is the value* $x(\delta^{out}(W))$.

Definition 2.10 *(Minimum Cut)*

A minimum cut *is a cut with the minimum edge value.*

Definition 2.11 *(Tight Cut)*

Given a vector x indexed by the edge set E, we say that $\delta^{out}(W)$ is a tight cut *if $x(\delta^{out}(W)) = 1$, i.e the value of that cut is one.*

Definition 2.12 *(Tight Set)*

A tight set *in a directed graph is a node set W for which $x(\delta^{out}(W)) = 1$.*

Some support digraphs of extreme points for the ASEP polytope Q_S^n when $n = 4$ and $n = 5$ are shown in Figure 2, and Figure 3, along with a list of tight sets.

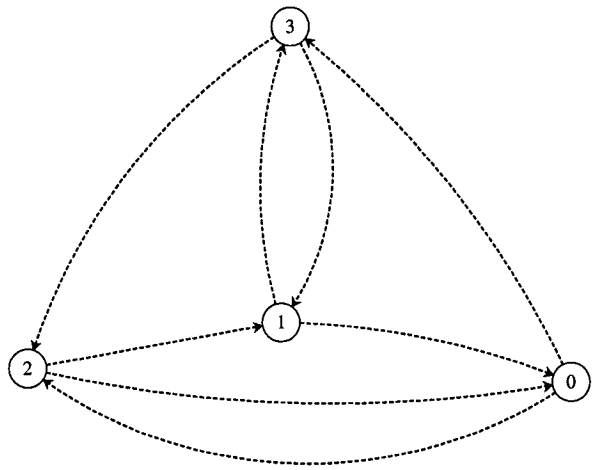

Figure 2: One of the support graphs for Q_S^4 with tight sets {0,2}, {1, 3}

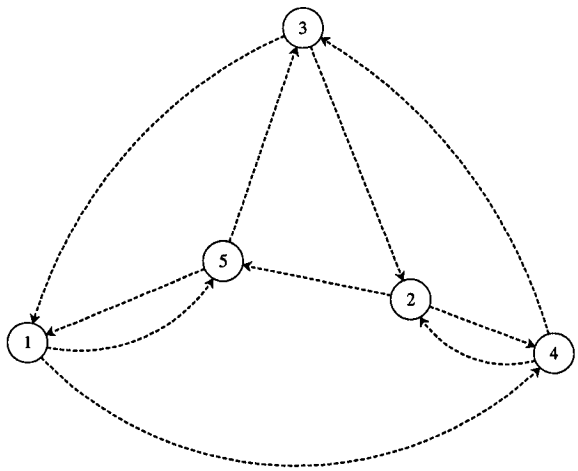

Figure 3: One of support graphs for Q_S^5 with tight sets {1, 5},{2,4}

Definition 2.13 *(Compatible Tour)*
Given an extreme point x^* of the ASEP polytope, a compatible tour *is a tour that,*

1. *Enters each tight set of x^* exactly once.*

2. *Leaves each tight set of x^* exactly once.*

In Figure 4, we can see all possible compatible tours for the extreme point shown in Figure 3. The tight sets found in the digraph in Figure 3 are: {1, 5} and {2, 4}.

Linear programming(LP) is an area linear algebra in which the objective is to minimize or maximize the linear objective function subject to finite number of linear equalities and inequalities.

Following is the general form of a linear programming (LP) problem, which is minimizing the objective function cx over the set of linear inequalities (28) and (29) and set of linear equality is (27).

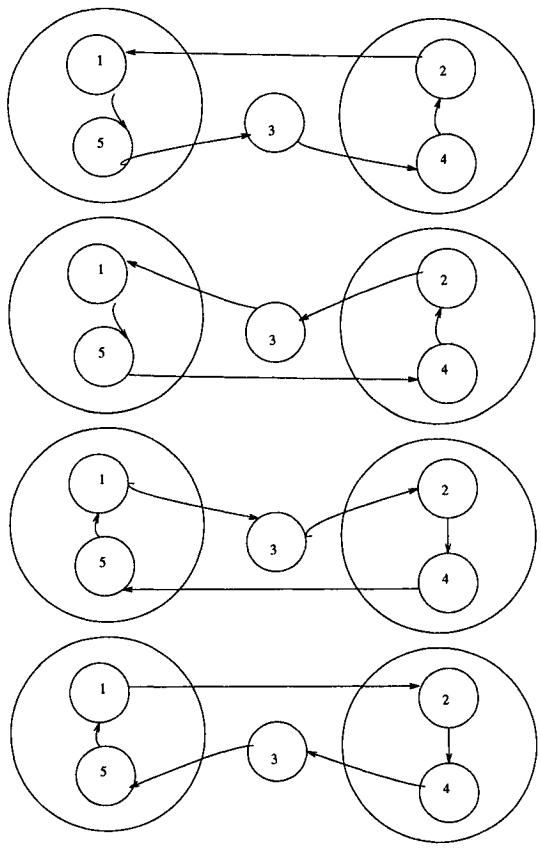

Figure 4: All possible compatible tours for the extreme point shown in Figure 3

$$\text{minimize: } cx \tag{26}$$

$$\text{subject to : } Ax = b, \tag{27}$$

$$Dx \geq d, \tag{28}$$

$$x \geq 0. \tag{29}$$

A vector x is a feasible solution to this LP if it satisfies (27), (28) and (29).The dual LP is a related LP. There are three types of variables in the dual LP of the ASEP.

1. For each out-degree constraint we have a variable w_v for all $v \in V$.

2. For each in-degree constraint we have a variable y_v for all $v \in V$.

3. For each cut constraint we have a variable d_S for all $S \subset V, 2 \leq |S| \leq n-2$.

Let $\mathcal{S} = \{S \in V : 2 \leq |S| \leq n-2\}$, then the dual LP of the ASEP 18–21 is:

$$\text{maximize} \sum_{v \in V} (w_v + y_v) + \sum_{S \in \mathcal{S}} d_S \tag{30}$$

$$\text{subject to: } w_u + y_v + \sum_{\substack{S \in \mathcal{S} \\ uv \in \delta(S)}} d_S \leq c_{uv} \qquad \text{for all } (u,v) \in E, \tag{31}$$

$$d_S \geq 0 \qquad \text{for all } S \in \mathcal{S}. \tag{32}$$

Theorem 2.0.1 *(Duality Theorem) If the primal problem (17) has solution x^* then the dual problem (30) has a solution (y^*, w^*, d^*) and $cx^* = \sum_{v \in V} (w_v + y_v) + \sum_{S \in \mathcal{S}} d_S$.*

Theorem 2.0.2 *(Complementary Slackness Theorem)*
A feasible solution x to the primal problem (17) and a feasible solution (y,w,d) to the dual problem (26) are optimal if and only if,

1. *$x_{uv} > 0$ implies that (y,w,d) satisfies the corresponding dual constraint with equality, i.e. $w_u + y_v + \sum_{\substack{S \in \mathcal{S} \\ uv \in \delta(S)}} d_S = c_{uv}$.*

2. *$x(\delta^{out}(S)) > 1$ in the primal implies that $d_S = 0$ in the dual.*

Chapter 3

Compatible Tours

The focus of our study is to construct a minimum cost tour which is compatible to a minimum cost solution x^* of the ASEP and we call this problem the CTP. Since the LP-relaxation(ASEP) of the ATSP gives a good lower bound on the ILP and the ASEP model has constraints $x(\delta^{in}(v)) = 1$ and $x(\delta^{out}(v)) = 1$ for each node $v \in V$, we have a special interest in all cuts with edge value one (i.e. the tight cuts) in optimal solution x^* of the ASEP and use the corresponding tight sets to help us construct a compatible tour for x^*.

The constraint (20) in ASEP forces the total edge value out of any cut to be at least 1. Thus the problem of finding all the tight cuts in the support graph of an optimal solution x^* of the ASEP is exactly the problem of finding all minimum cuts in this digraph.

The requirements to construct or generate a compatible tour are:

1. An optimal ASEP solution x^* and corresponding cost vector c.

2. All tight sets in x^*.

In later chapters, we explain how we are able to get an optimal ASEP solution x^* and all tight sets in x^* to construct or generate a minimal cost compatible tour. We

minimize the number of tight sets by dropping all tight sets of size 1 (which trivially hold for all tours) and tight sets of size more than $\frac{n}{2}$, as it follows from constraints (19) and (18) in the ASEP model that for any set $W \subset V$, the total edge value going out of W is equal total edge value coming into W, i.e $x(\delta^{out}(W)) = x(\delta^{in}(W))$. Hence it is enough to consider all tight sets with $2 \leq k \leq \lfloor \frac{n}{2} \rfloor$ since when W is a tight set V/W is also a tight set.

3.1 Finding all minimum cuts in a directed graph

Finding minimum cuts in graphs is an interesting problem and has many applications in some areas, including network design. This problem has two variants:
(1) finding a minimum set of edges whose removal disconnects a particular vertex s from a particular vertex t (we call this version the *minimum $s-t$ cut problem*),
(2) finding a minimum set of edges whose removal disconnects the graph (we call this version the *minimum cut problem*). Both versions apply to directed and undirected graphs.

For undirected graphs: Finding a minimum cut, finding all minimum cuts and finding a minimum $s-t$ cut are very well studied problems [2]. For undirected graphs, the Gomory-Hu algorithm runs in $O(n^2 m log(n^2 m))$ time on a graph with n nodes and m edges [15]. Dinitz et al. [11] show a cactus representation of all minimum cuts from which crucial information about the represented cuts can be very efficiently read. The cactus representation is a strong characterisation of all minimum cuts in a undirected graph.
For directed graphs: The minimum $s-t$ cut problem for directed graphs can be solved by using the Maximum flow-Minimum cut theorem which says the maximum

$s-t$ flow in the graph is equal to the minimum $s-t$ cut in the graph. Finding a *minimum cut* is a well studied problem [8], [13] but not much is known regarding finding all minimum cuts a in directed graph. Practically we only find all minimum cuts for small n since there can be an exponential the number of minimum cuts in a digraph and it is much harder with larger n.

3.2 Generating all tight sets for small n

When n is small, say $n \leq 17$, we can generate all tight sets exhaustively, i.e. we generate all possible subsets of V of size 2 to $\lfloor \frac{n}{2} \rfloor$ and then check each subset W to see if $x(\delta^{out}(W))$ is 1.

3.3 ILP formulation of the CTP

Based on the ATSP LP-model, we can formulate an ILP for finding an optimal solution to the CTP as follows. For a given minimum cost solution x^* of the ASEP for cost vector c, let $\mathcal{TS}$ be the set of all tight sets in x^* such that for each set $T \in \mathcal{TS}$ we have $2 \leq |T| \leq \lfloor n/2 \rfloor$.

$$\text{Minimize} \quad cx \tag{33}$$

$$\text{subject to:} \quad x(\delta^{in}(v)) = 1 \text{ for all } v \in V, \tag{34}$$

$$x(\delta^{out}(v)) = 1 \text{ for all } v \in V, \tag{35}$$

$$x(\delta^{out}(S)) \geq 1 \text{ for all } S \subset V \text{ where } 3 \leq |S| \leq \lfloor \frac{n}{2} \rfloor, \tag{36}$$

$$x(\delta^{out}(T)) = 1 \text{ for all } T \in \mathcal{TS}, \tag{37}$$

$$x_e \geq 0 \quad \text{for all } e \in E, \tag{38}$$

$$x_e \text{ is integer} \quad \text{for all } e \in E. \tag{39}$$

In the above model, due to the additional constraints (37) (henceforth called *tight cut constraints*), an optimal solution to (33)–(39) is an optimal compatible tour compatible with x^*.

Given an ATSP with non-negative metric cost vector c, we solve the corresponding ASEP to get a minimum cost solution x^*, we find all tight sets in x^* and construct a tour which is compatible with the x^* satisfying the additional tight cut constraints (37), i.e, the tour which satisfies constraints (34) - (39).

For small n, solving the above ILP would be practical. We expect it to be difficult to solve the ILP of the CTP for big n, because it is believed to be NP-complete. We develop a heuristic for solving the CTP in Chapter 6 in the process of finding a heuristic solution to the ATSP.

Chapter 4

Exact worst–case ratio between CTP and ASEP for small n

For any non-negative metric cost function c, we use CTP(c) to denote an optimal solution value of the CTP and ASEP(c) to denote an optimal solution value of the ASEP. We wish to find the exact worst–case ratio between CTP(c) and ASEP(c) for small n.

When the number of nodes in the complete graph $G = (V, E)$ is fixed in size, for a particular n, the worst–case is the worst–case over all metric costs functions c and it is denoted by β_n, i.e.

$$\beta_n = \max_{\substack{c \geq 0 \\ c \text{ is metric}}} \left(\frac{CTP(c)}{ASEP(c)}\right).$$

To compute β_n, we use a method similar to the one used in [3] for finding the integrality gap for the Symmetric TSP problem. We outline this method from [3] in the next Section 4.1.

4.1 Previous work on finding the integrality gap for fixed n

Boyd and Labonté explain in [3] how they were able to find the exact integrality gap for the Symmetric TSP for small values of n.

For a given metric cost vector on n nodes, let $TOUR$ denote the optimal solution value to the corresponding STSP, and let $SUBT$ denote the optimal value to the SEP and the integrality gap

$$\alpha_n = \max_{\substack{c \geq 0 \\ c \text{ is metric}}} \left(\frac{TOUR}{SUBT}\right).$$

For a particular metric cost vector c, if you divide all edge costs $c \in R^E$ by TOUR, then the new costs also satisfy the triangle inequality, and the new value of TOUR will be 1 and the ratio TOUR/SUBT remains same. Then the integrality gap becomes,

$$\alpha_n = \max_{\substack{c \geq 0 \text{ and } c \text{ is metric} \\ TOUR=1}} \left(\frac{1}{SUBT}\right),$$

or equivalently,

$$1/\alpha_n = \min_{\substack{c \geq 0 \text{ and } c \text{ is metric} \\ TOUR=1}} (SUBT). \tag{40}$$

This leads to the following quadratic model,

$$\text{Minimize } cx \tag{41}$$

$$\text{subject to: } x \quad \text{satisfies constraints (13)} - \text{(15)}, \tag{42}$$

$$c_{ij} + c_{jk} \geq c_{ik} \quad \forall i,\ j,\ k \in V, \tag{43}$$

$$c(T) \geq 1 \quad \forall \text{ tours T}, \tag{44}$$

$$c_e \geq 0 \quad \forall e \in E. \tag{45}$$

The constraint (42) ensures x is an optimal SEP solution. The constraints (43) and (45) ensure the cost vector is non-negative and metric and constraints of type (44) ensure that the minimum tour value is 1 for all tours using cost vector c. If the optimal solution for (41) is OPT, then $\alpha_n = 1/OPT$.

Since they found the above quadratic model impractical, they came up with a new way of modeling the problem as a series of LPs.

Let $X = \{x_{(1)}, x_{(2)}, x_{(3)}, x_{(4)}..., x_{(p)}\}$ be a complete list of the extreme points of the SEP polytope S^n. From the theory of polyhedra, for every cost function $c \in R^E$, there exists at least one extreme point x^* in X such that the c is minimized over S^n at x^*, i.e. $SUBT = cx^*$.

For each extreme point $x_{(i)}$, let

$$C_i = \{c \in R^E : c \text{ is minimized over } S^n \text{ at } x_{(i)}\} \tag{46}$$

and now solve the following problem for each C_i, where $x_{(i)}$ is fixed:

$$(\min cx_{(i)} : c \in C_i \text{ and satisfies constraints}(43), (44) \text{ and } (45)). \tag{47}$$

If OPT_i represents the optimal value of (47) then value of

$$1/\alpha_n = \min_{1 \le i \le p} (OPT_i).$$

Using duality theory, the condition $c \in C_i$ in (47), can also be represented by the set of linear constraints as shown below.

If the dual LP of the SEP has variables $y \in R^V$, $u \in R^E$, and $d \in R^S$ and let $\mathcal{S} = \{S \subset V, 3 \le |S| \le n-3\}$, then the dual LP of the SEP is defined as follows:

$$\text{maximize } 2 \cdot y - 1 \cdot u + 2 \cdot d \tag{48}$$

$$\text{Subject to: } y_i + y_j - u_{ij} + \sum (d_s : S \in \mathcal{S}, ij \in \delta(S)) \le c_{ij} \quad \forall ij \in E, \tag{49}$$

$$u_e \ge 0 \quad \forall e \in E, \tag{50}$$

$$d_S \ge 0 \quad \forall S \in \mathcal{S}. \tag{51}$$

Any feasible solution x^* for the SEP, and its feasible dual LP solution (y, u, d) is said to satisfy the complimentary slackness conditions for the x^* if it satisfies the following:

$$y_i + y_j - u_{ij} + \sum d_s : S \in \mathcal{S}, ij \in \delta(S) = c_{ij} \qquad \forall ij \in E \text{ such that } x^*_{ij} > 0, \quad (52)$$

$$u_e = 0 \qquad \forall e \in E \text{ such that } x^*_e < 1, \quad (53)$$

$$d_S = 0 \quad \forall S \in \mathcal{S} \text{ such that } x^*(\delta(S)) > 2. \quad (54)$$

From the theory of duality and complimentary slackness, $c \in C_i$ if and only if there exists a feasible solution (y, u, d) for the dual LP which satisfies the complimentary slackness conditions (52), (53) and (54) for $x_{(i)}$. Thus for each C_i we can find OPT_i by solving the following LP which has variables c and (y, u, d):

$$OPT_i = (\min cx_{(i)} : c \text{ satisfies constraints } (43) - (45), \quad (55)$$
$$\text{and } (y, u, d) \text{ satisfy} (49) - (54) \text{ for } x_{(i)}).$$

4.2 Finding the exact worst–case ratio

Following the method used to find the exact integrality gap for STSP, here we find the exact worst–case ratio $CTP(c)/ASEP(c)$, i.e

$$\beta_n = \max_{\substack{c\geq 0 \\ c \text{ is metric}}} \Big(\frac{CTP(c)}{ASEP(c)}\Big).$$

If we divide all the edge costs of $c \in R^E$ by CTP(c), then the new costs also satisfy the triangle inequality, the new value of CTP(c) will be 1 and the ratio CTP(c)/ASEP(c) remains the same. Note the new value of CTP(c) using this new cost function will be 1. Thus to solve our problem, it is sufficient to only consider the metric cost functions for which CTP(c) is 1 and we have,

$$\beta_n = \max_{\substack{c\geq 0 \\ c \text{ is metric}}} \Big(\frac{1}{ASEP(c)}\Big),$$

or equivalently,

$$\frac{1}{\beta_n} = \min_{\substack{c\geq 0 \text{ and } c \text{ is metric} \\ \text{CTP(c)} = 1}} (ASEP(c)). \tag{56}$$

This leads to the following quadratic model,

$$\text{Minimize} \quad cx \tag{57}$$

$$\text{subject to: } x \text{ satisfies constraints (18) } -(21), \tag{58}$$

$$c_{ij} + c_{jk} \geq c_{ik} \text{ for all distinct } i,\ j,\ k \in V, \tag{59}$$

$$c(T) \geq 1 \text{ for all compatible tours T of } x, \tag{60}$$

$$c_e \geq 0 \text{ for all distinct } e \in E. \tag{61}$$

Constraints (59) and (61) ensure that the cost vector is non-negative and metric and constraints of type (60) ensure that a minimum cost compatible tour has value 1. If the value of an optimal solution to (57) is k, then $\beta_n = 1/k$.

Since the quadratic model similar to the above was impractical for finding the integrality gap (as outlined in previous Section 4.1), we also started working from a complete list of extreme points of the ASEP polytope Q_S^n.

Let $X = \{x_{(1)}, x_{(2)}, x_{(3)}, x_{(4)}, .., x_{(p)}\}$ be a complete list of extreme points of ASEP polytope Q_S^n. We now know that for every cost function $c \in R^E$, there exists at least one extreme point x^* in X such that c is minimized over Q_S^n at x^*, i.e. $ASEP(c) = cx^*$.

$$\text{Let } C_i = \{c \in R^E : c \text{ is minimized over } Q_S^n \text{ at } x_{(i)}\}. \tag{62}$$

and now we need to solve the following problem for each C_i, where $x_{(i)}$ is fixed:

$$(\min cx_{(i)} : c \in C_i \text{ and satisfies constraints (59), (60) and (61)}). \tag{63}$$

The condition $c \in C_i$ can be represented by a set of linear constraints by using duality theory. There are three types of variables in the dual LP of ASEP.

1. For each out-degree constraint we have a variable: w_v for all $v \in V$.

2. For each in-degree constraint we have a variable: y_v for all $v \in V$.

3. For each cut constraint we have a variable: d_S for all $S \subset V, 2 \leq |S| \leq n-2$.

Let $\mathcal{S} = \{S \in V, 2 \leq |S| \leq n-2\}$, then the dual LP of the ASEP is:

$$\text{maximize } \sum_{v \in V} (w_v + y_v) + \sum_{S \in \mathcal{S}} d_S, \tag{64}$$

$$\text{subject to: } w_u + y_v + \sum_{\substack{S \in \mathcal{S} \\ uv \in \delta(S)}} d_S \leq c_{uv} \qquad \forall (u,v) \in E, \tag{65}$$

$$d_S \geq 0 \qquad \forall S \in \mathcal{S}. \tag{66}$$

For any feasible solution x^* for the ASEP, its feasible dual LP solution (w, y, d) is said to satisfy the complimentary slackness conditions for the x^* if it satisfies the

following:

$$w_u + y_v + \sum_{\substack{S \in \mathcal{S} \\ uv \in \delta(S)}} d_S = c_{uv} \qquad \forall (u,v) \in E \text{ such that } x^*_{uv} > 0, \tag{67}$$

$$d_S = 0 \qquad \forall S \in \mathcal{S} \text{ such that } x^*(\delta^{out}(S)) > 1. \tag{68}$$

If $X = \{x_{(1)}, x_{(2)}, x_{(3)}, x_{(4)} ..., x_{(p)}\}$ is a complete list of the extreme points of the subtour elimination polytope Q^n_S, for each extreme point $x_{(i)}$ from Q^n_S, we find the gap between CTP(c) to ASEP(c) for every $c \in C_i$ and the worst–case is worst among all of them. Thus for particular n, we can find the worst–case ratio β_n by solving the following LP :

$$\min cx_{(i)} \tag{69}$$

$$c \text{ satisfies constraints (59) – (61)}, \tag{70}$$

$$(w, y, d) \text{ satisfy (65) – (68) for } x_{(i)}. \tag{71}$$

If k_i represents the optimal value of (69) then value of

$$1/\beta_n = \min_{1 \le i \le p} (k_i).$$

The CPLEX 8.0 commercial package was used as the LP solver.

4.3 Finding the necessary extreme points of Q^n_S

We need to avoid solving LP (57)–(61) for k_i for every extreme point $x_{(i)} \in X$ as the size of X can be very large even for small values of n. To compute the worst–case ratio β_n, it is sufficient to consider all non-isomorphic extreme points of ASEP polytope for particular n [3].

Boyd and Elliott-Magwood [4] discuss how they were able to find all non-isomorphic extreme points of the ASEP polytope for $4 \le n \le 7$ based on the idea used in [3] to

generate all non-isomorphic extreme points of the SEP polytope. For $8 \leq n \leq 10$, they used other methods to find a large percentage of all non-isomorphic extreme points of ASEP, and in particular all $\frac{1}{2}$-integer extreme points. We make use of this data in finding β_n.

Note that because all the non-isomorphic extreme points are known for $4 \leq n \leq 7$ we find β_n exactly for these values of n. However, for $8 \leq n \leq 10$, we will be finding a lower bound on β_n as we do not have a complete list of the non-isomorphic extreme points.

4.4 Generating all compatible tours for small n

To compute β_n, we need to have all compatible tours for each extreme point $x^* \in Q_S^n$ and here we describe how to generate all of them.

Instead of solving the ILP for the CTP, we exhaustively generate all compatible tours for small n, $4 \leq n \leq 10$.

For each extreme point $x^* \in Q_S^n$, we first find all tight sets. Next, we generate all possible tours satisfying constraints (7)-(11). Finally, we select each tour which is compatible with the extreme point x^* i.e. if the tour additionally satisfies the constraint (37) for all tight sets found before.

4.5 Obtained worst–case ratios

Using the extreme points generated by Boyd and Elliott-Magwood on small n, we were able to compute the β_n.

For $4 \leq n \leq 7$: Exact worst–case ratio has been computed using all non-isomorphic extreme points.

For $8 \leq n \leq 10$: Lower bound on β_n is computed using non-isomorphic extreme

points generated and the results are shown in Tables 1 and 2.

n	Number of all non-isomorphic extreme points	β_n
4	2	5/4
5	5	4/3
6	90	3/2
7	3700	3/2

Table 1: Exact values of β_n

n	Number of non-isomorphic extreme points considered	$\beta_n \geq$
8	1160	8/5
9	10,862	2
10	67550	2

Table 2: Lower bound for β_n

We compare our results with previously computed integrality gap results for small ATSP's. The integrality gap for small metric ATSP's has been computed by Boyd and Elliott-Magwood [4].

$$\alpha_n = \max_{\substack{c \in R^E \\ c \text{ is metric}}} \left(\frac{ATSP(c)}{ASEP(c)} \right)$$

Since we know that $\frac{\text{CTP(c)}}{\text{ASEP(c)}} \geq \frac{\text{ATSP(c)}}{\text{ASEP(c)}}$, we compare the obtained worst–case ratio β_n with the integrality gap α_n for particular n in Table 3.

n	β_n	α_n
4	5/4	6/5
5	4/3	5/4
6	3/2	4/3
7	3/2	4/3
8	8/5	4/3
9	2	11/8
10	2	7/5

Table 3: Comparison table

The support graphs of the extreme points that attained the worst–case ratio for β_n are shown in Figures 5–?? along with the tight sets and obtained cost vectors which gave the maximum gap. Each edge in the support graphs is labeled with the corresponding cost component of the obtained cost vector. All edges with $x_e = \frac{1}{2}$ are shown by dashed edges, edges with $x_e = 1$ are shown by solid edges and edges with $x_e = 0$ are not shown.

There is a unique non-integer extreme point in Q_S^4 that attains the worst-case ratio $\beta_4 = 5/4$. The support graph of this extreme point is shown in Figure 5. For

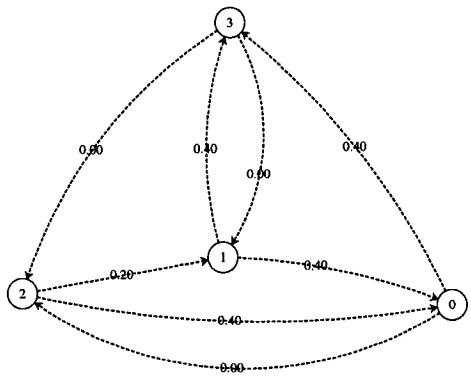

Figure 5: Support graph for unique non-integer extreme point of Q_S^4 and $\beta_4 = 5/4$

	0	1	2	3
0	0.000000	0.200000	0.000000	0.400000
1	0.400000	0.000000	0.400000	0.400000
2	0.400000	0.200000	0.000000	0.600000
3	0.200000	0.000000	0.000000	0.000000

Table 4: Obtained cost vector for β_4

this extreme point, the tight sets are : **{1,3},{0,2}** and an optimal compatible tour is **(0,1,3,2)**.

For n=5, there are 5 non-integer extreme points in Q_S^5. Two of the extreme points in Q_S^5 attain the maximum gap $\beta_5 = 4/3$. The support graph of one of the extreme point for which $\beta_5 = 4/3$ is shown in Figure 6.

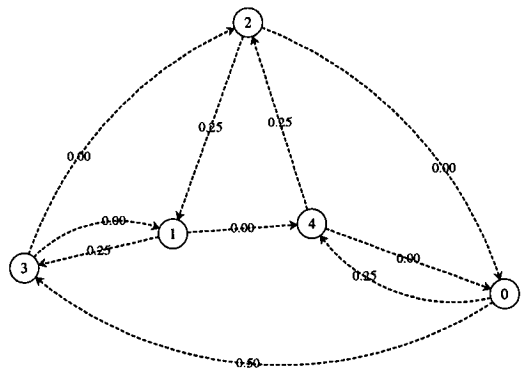

Figure 6: Support graph of the first extreme point in Q_S^5 with $\beta_5 = 4/3$

	0	1	2	3	4
0	0.000000	0.500000	0.500000	0.500000	0.250000
1	0.000000	0.000000	0.250000	0.250000	0.000000
2	0.000000	0.250000	0.000000	0.500000	0.250000
3	0.000000	0.000000	0.000000	0.000000	0.000000
4	0.000000	0.500000	0.250000	0.500000	0.000000

Table 5: Obtained cost vector for β_5

For an extreme point shown in Figure 6, the tight sets are : **{1,3},{0,4}** an optimal compatible tour is **(4,0,2,1,3)**.

Two extreme points attained the maximum gap $\beta_6 = 3/2$ among 90 non-isomorphic extreme points in Q_S^n on $n = 6$. The support graphs of the two of those extreme points for which $\beta_6 = 3/2$ are shown in Figures 7 and 8.

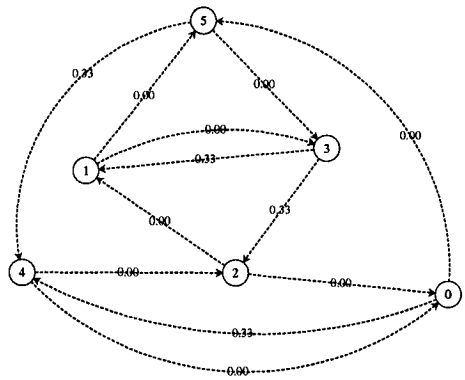

Figure 7: Support graph of the second extreme point in Q_S^6 with $\beta_6 = 3/2$

	0	1	2	3	4	5
0	0.000000	0.000000	0.333333	0.000000	0.333333	0.000000
1	0.000000	0.000000	0.333333	0.000000	0.333333	0.000000
2	0.000000	0.000000	0.000000	0.000000	0.333333	0.000000
3	0.333333	0.333333	0.333333	0.000000	0.666667	0.333333
4	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
5	0.333333	0.333333	0.333333	0.000000	0.333333	0.000000

Table 6: Obtained cost vector for β_6

For an extreme point shown in Figure 7, the tight sets are : **{1,3},{0,4},{1,3,5},{1,2,3}** an optimal compatible tour is **(0,2,1,3,5,4)**.

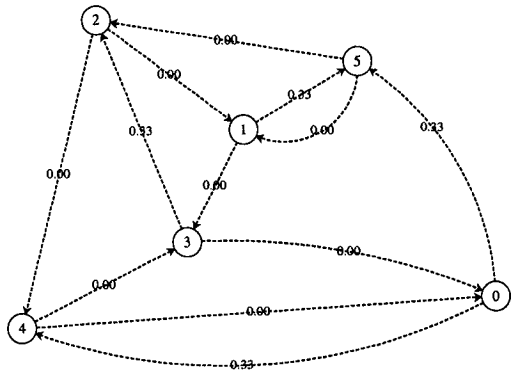

Figure 8: Support graph of the second extreme point in Q_S^6 and $\beta_6 = 3/2$

	0	1	2	3	4	5
0	0.000000	0.333333	0.333333	0.333333	0.333333	0.333333
1	0.000000	0.000000	0.333333	0.000000	0.333333	0.333333
2	0.000000	0.000000	0.000000	0.000000	0.000000	0.333333
3	0.000000	0.333333	0.333333	0.000000	0.333333	0.333333
4	0.000000	0.333333	0.333333	0.000000	0.000000	0.333333
5	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

Table 7: Obtained cost vector for β_6

For an extreme point shown in Figure 8, the tight sets are:**{1,5},{0,4}, {1,2,5}** an optimal compatible tour is **(0,1,5,2,3,4)**.

21 extreme points attained the maximum gap $\beta_7 = 3/2$ among approximately 3700 non-isomorphic extreme points in Q_S^n on $n = 7$. The support graph of one of these 21 extreme points for which $\beta_7 = 3/2$ are shown in Figure 9.

For an extreme point shown in Figure 9, the tight sets are : **{2,5}, {3,4}, {0,1}, {3,4,6}** an optimal compatible tour is **(0,1,2,5,3,4,6)**.

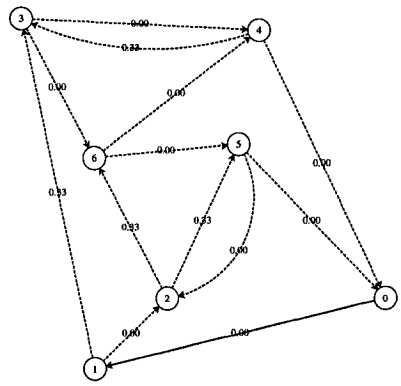

Figure 9: Support graph of a extreme point in Q_S^7 having $\beta_7 = 3/2$

	0	1	2	3	4	5	6
0	0.000000	0.000000	0.000000	0.333333	0.333333	0.333333	0.333333
1	0.000000	0.000000	0.000000	0.333333	0.333333	0.333333	0.333333
2	0.333333	0.333333	0.000000	0.666667	0.333333	0.333333	0.333333
3	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
4	0.000000	0.000000	0.000000	0.333333	0.000000	0.000000	0.333333
5	0.000000	0.000000	0.000000	0.333333	0.333333	0.000000	0.333333
6	0.000000	0.000000	0.000000	0.333333	0.000000	0.000000	0.000000

Table 8: Obtained cost vector for β_7

A unique half-integer extreme point attained the maximum gap $\beta_8 = 8/5$ among all 1160 known extreme points on n=8. The support graph of the that extreme point for which $\beta_8 = 8/5$ is shown in Figure 10.

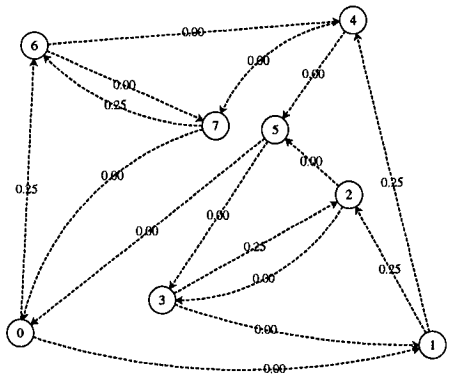

Figure 10: Support graph of the extreme point in Q_S^8 having $\beta_8 = 8/5$

	0	1	2	3	4	5	6	7
0	0.000000	0.000000	0.250000	0.125000	0.250000	0.250000	0.250000	0.250000
1	0.250000	0.000000	0.250000	0.250000	0.250000	0.250000	0.375000	0.250000
2	0.000000	0.000000	0.000000	0.000000	0.125000	0.000000	0.250000	0.125000
3	0.125000	0.000000	0.250000	0.000000	0.250000	0.250000	0.375000	0.250000
4	0.000000	0.000000	0.250000	0.000000	0.000000	0.000000	0.250000	0.000000
5	0.000000	0.000000	0.250000	0.000000	0.250000	0.000000	0.250000	0.125000
6	0.000000	0.000000	0.250000	0.000000	0.000000	0.000000	0.000000	0.000000
7	0.000000	0.000000	0.250000	0.125000	0.250000	0.125000	0.250000	0.000000

Table 9: Obtained cost vector for β_8

For an extreme point shown in Figure 10,
the tight sets are : **{6,7}, {2,3}, {4,6,7}, {2,3,5}, {1,2,3}, {0,6,7}, {1,2,3,5}**
and an optimal compatible tour is **(0,1,2,3,5,4,6,7)**.

We examined 10,862 extreme points on n=9. There are more than 50 extreme points those attain the maximum gap when n=9. There were 67,550 extreme points on n=10 and more than 50 extreme points attained the maximum gap.

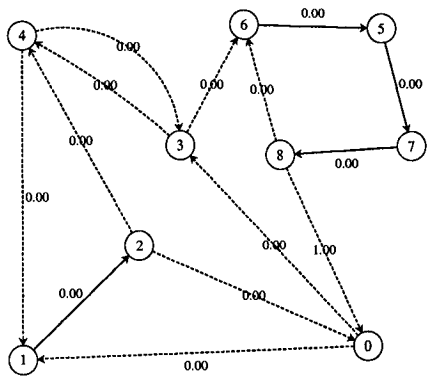

Figure 11: Support graph of the extreme point in Q_S^9 and $\beta_9 = 2$

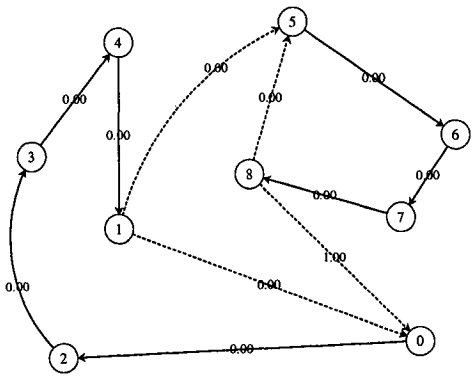

Figure 12: Support graph of the extreme point in Q_S^9 and $\beta_9 = 2$

Chapter 5

Exact worst–case ratio between CTP and ATSP for small n

For any non-negative metric cost function c, we use ATSP(c) to denote an optimal solution value of the ATSP. We wish to find the exact worst–case ratio between the CTP(c) and ATSP(c) for small n. For the complete graph on n nodes, let γ_n denote the worst–case ratio, i.e,

$$\gamma_n = \max_{\substack{c \geq 0 \\ c \text{ is metric}}} \left(\frac{CTP(c)}{ATSP(c)} \right).$$

In order to find the γ_n for particular n, we used similar ideas as those described in Section 4.1 to find the exact worst–case ratio between the CTP(c) and ASEP(c).

5.1 Exact worst–case ratio

If we divide all the edge costs by CTP(c), then the new costs also satisfy the triangle inequality. The ratio CTP(c)/ATSP(c) remains unchanged [3]. Note the new value of the CTP(c) using this new cost function will be 1. Thus to solve our problem, it is

sufficient to only consider the metric cost functions for which CTP(c) is 1 and we have

$$\gamma_n = \max_{\substack{c\geq 0 \text{ and } c \text{ is metric} \\ \text{CTP(c)} = 1}} \left(\frac{1}{ATSP(c)}\right),$$

or equivalently,

$$\frac{1}{\gamma_n} = \min_{\substack{c\geq 0 \text{ and } c \text{ is metric} \\ \text{CTP(c)} = 1}} (ATSP(c)). \tag{72}$$

For particular n, let $\mathcal{T} = \{T^1, T^2, T^3, ..., T^l\}$ be the complete list of tours. Let $X = \{x_{(1)}, x_{(2)}, x_{(3)}, x_{(4)}..., x_{(p)}\}$ be a complete set of the extreme points of the ASEP polytope Q_S^n and for each extreme point $x_{(i)}$ let $\mathcal{T}_{comp}^{x_{(i)}}$ be the complete set of compatible tours for $x_{(i)}$.

As before we can partition the cost function c into sets. For $i = 1, 2, 3, .., p$ let

$$C_i = \{c \in R^E : c \text{ is minimized over } X \text{ at } x_{(i)}, i = 1, 2, 3...p\}. \tag{73}$$

Now we need to solve the following problem for each tour T^j in $\mathcal{T}$, j=1, 2, 3...l and each $x_{(i)}, i = 1, 2, 3..p$ where $x_{(i)}$ is fixed, T^j is fixed, and c is a variable vector.

$$\text{Minimize} \quad cT^j \tag{74}$$

subject to:

$$c(T_{comp}) \geq 1 \text{ for all compatible tours } T_{comp} \in \mathcal{T}_{comp}^{x_{(i)}}, \tag{75}$$

$$c_{ij} + c_{jk} \geq c_{ik} \text{ for all distinct } i,\ j,\ k \in V, \tag{76}$$

$$c_e \geq 0 \text{ for all distinct } e \in E, \tag{77}$$

$$c \in C_i. \tag{78}$$

In the above model constraint (75) ensured the minimum compatible tour length is 1 for all tours compatible with $x_{(i)}$ and constraints (76)–(77) ensure c is metric and non-negative.

In the above model (74)–(78) everything is linear except the condition (78). This can also be represented by a set of linear constraints by duality theory as explained in Chapter (4).

Now for each extreme point $x_{(i)}, i = 1, 2, 3...p$, solve the following model for each $T^j \in \mathcal{T}$.

$$\min cT^j \tag{79}$$

$$c \text{ satisfies constraints } (75) - (77), \tag{80}$$

$$(w, y, d) \text{ satisfy } (64) - (68) \text{ for } x_{(i)}. \tag{81}$$

If k_i represents the optimal value of (79), and p is number of extreme points in Q_S^n then the value of γ_n is,

$$\gamma_n = \max_{1 \leq i \leq p*(n-1)!} \left(\frac{1}{k_i}\right)$$

.

To summarise how we find γ_n for particular n, if $X = \{x_{(1)}, x_{(2)}, x_{(3)}, x_{(4)}..., x_{(p)}\}$ is a complete list of the extreme points of Q_S^n, then for each extreme point $x_{(i)}, i = 1, 2, 3..p$:

1. Find the complete list of tours $\mathcal{T} = \{T^1, T^2, T^3, ..., T^l\}$.

2. Find the complete list of compatible tours $\mathcal{T}_{comp}^{x_{(i)}}$ for $x_{(i)}$.

3. For each tour $T^j, j = 1, 2, 3..l$, solve LP (79) to find a cost vector c from C_i and an optimal solution k_i.

4. Find the minimum k_i value among all computed in (3) to find the γ_n by calculating the $1/k_i$.

5.2 Obtained worst–case ratios

For $4 \leq n \leq 7$: The exact worst–case ratio has been computed using all non-isomorphic extreme points. For $n = 8$: a lower bound on γ_n is computed using half-integer extreme points generated and the results are shown in Tables 10 and 11. The support digraphs of some of the extreme points, which attain maximum gap are shown in Figures 13 - 17.

n	Number of all non-isomorphic extreme points	γ_n	Number of LPs solved
4	2	5/4	6
5	5	4/3	120
6	90	3/2	10800
7	3700	3/2	2664000

Table 10: Exact values of γ_n

n	Number of all non-isomorphic half integer extreme points	$\gamma_n \geq$	Number of LPs solved
8	41	3/2	206640

Table 11: Lower bound on γ_n

Since we know that $\frac{\text{CTP(c)}}{\text{ASEP(c)}} \geq \frac{\text{CTP(c)}}{\text{ATSP(c)}}$, we compare the obtained worst–case ratio β_n with the integrality gap α_n for particular n in Table 12.

n	β_n	γ_n
4	5/4	5/4
5	4/3	4/3
6	3/2	3/2
7	3/2	3/2
8	8/5	3/2

Table 12: Comparison Table

5.3 Generating all possible tours for small n

We generate all tours for particular n to compute β_n and γ_n. Since the ATSP can be mathematically represented as a complete digraph G on n nodes, there will be $(n-1)!$ number of tours in G.

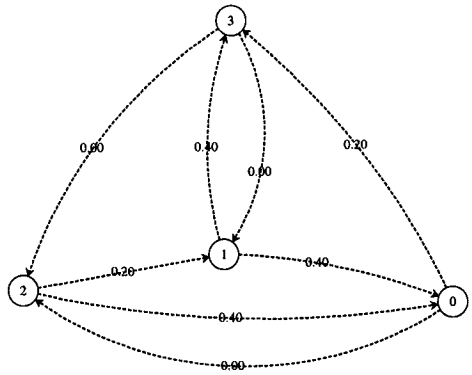

Figure 13: Support graph of the unique non-integer extreme point of Q_S^4 and $\gamma_4 = 5/4$

	0	1	2	3
0	0.000000	0.200000	0.000000	0.200000
1	0.400000	0.000000	0.400000	0.400000
2	0.400000	0.200000	0.000000	0.600000
3	0.400000	0.000000	0.000000	0.000000

Table 13: Obtained cost vector for γ_4

For an extreme point shown in Figure 13, the tight sets are :**{1,3}, {0,2}**, the best tour:**(2,1,0,3)** and the best compatible tour:**(3,0,2,1)**.

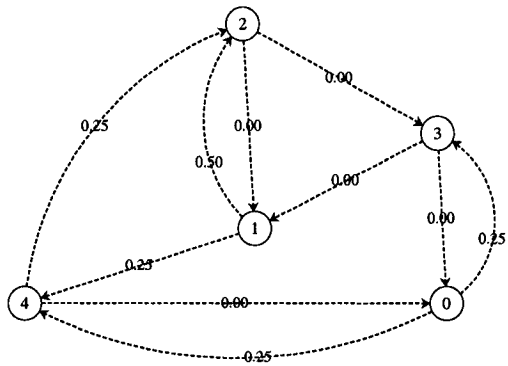

Figure 14: Support graph of the extreme point in Q_S^5 with $\gamma_5 = 4/3$

	0	1	2	3	4
0	0.000000	0.250000	0.500000	0.250000	0.250000
1	0.250000	0.000000	0.500000	0.500000	0.250000
2	0.000000	0.000000	0.000000	0.000000	0.250000
3	0.000000	0.000000	0.500000	0.000000	0.250000
4	0.000000	0.250000	0.250000	0.250000	0.000000

Table 14: Obtained cost vector for γ_5

For an extreme point shown in Figure 14, the tight sets are :**{1,2}, {0,4}, {0,3}**, an optimal compatible tour is **(0,4,1,2,3)**.

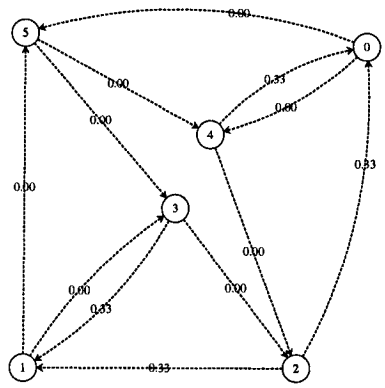

Figure 15: Support graph of the extreme point in Q_S^6 with $\gamma_6 = 3/2$

	0	1	2	3	4	5
0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
1	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
2	0.333333	0.333333	0.000000	0.333333	0.333333	0.333333
3	0.333333	0.333333	0.000000	0.000000	0.333333	0.333333
4	0.333333	0.333333	0.000000	0.333333	0.000000	0.333333
5	0.333333	0.333333	0.000000	0.000000	0.000000	0.000000

Table 15: Obtained cost vector for γ_6

For an extreme point shown in Figure 15, the tight sets are :**{1,3}, {0,4}, {1,3,5}, {1,2,3}**, the best tour:**(4,2,3,1,0,5)** and the best Compatible Tour:**(5,0,4,2,1,3)**.

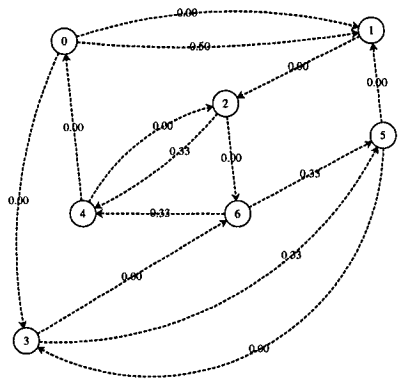

Figure 16: Support graph of the extreme point in Q_S^7 with $\gamma_7 = 3/2$

	0	1	2	3	4	5	6
0	0.000000	0.000000	0.000000	0.000000	0.333333	0.333333	0.000000
1	0.000000	0.000000	0.000000	0.000000	0.333333	0.333333	0.000000
2	0.333333	0.333333	0.000000	0.333333	0.333333	0.333333	0.000000
3	0.333333	0.333333	0.333333	0.000000	0.333333	0.333333	0.000000
4	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
5	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
6	0.333333	0.333333	0.333333	0.333333	0.333333	0.333333	0.000000

Table 16: Obtained cost vector for γ_7

For an extreme point shown in Figure 16,

The tight sets are :**{0,1}, {2,4}, {3,5}, {3,5,6}, {2,4,6}**, the best tour:**(5,3,4,2,1,0,6)** and the best Compatible Tour:**(0,1,2,4,6,3,5)**.

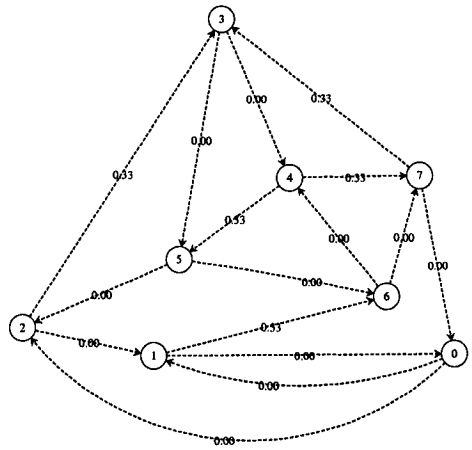

Figure 17: Support graph of the extreme point in Q_S^8 and $\gamma_8 = 3/2$

	0	1	2	3	4	5	6	7
0	0.000000	0.000000	0.000000	0.330000	0.330000	0.330000	0.330000	0.330000
1	0.000000	0.000000	0.000000	0.330000	0.330000	0.330000	0.330000	0.330000
2	0.000000	0.000000	0.000000	0.330000	0.330000	0.330000	0.330000	0.330000
3	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.330000	0.330000
4	0.330000	0.330000	0.330000	0.660000	0.000000	0.330000	0.330000	0.330000
5	0.000000	0.000000	0.000000	0.330000	0.330000	0.000000	0.000000	0.330000
6	0.000000	0.000000	0.000000	0.330000	0.000000	0.330000	0.000000	0.000000
7	0.000000	0.000000	0.000000	0.330000	0.330000	0.330000	0.330000	0.000000

Table 17: Obtained cost vector for γ_8

For an extreme point shown in Figure 17, the tight sets are :**{0,1}**, **{0,1,2}**, the best tour:**(0,1,2,3,4,5,6,7)** and the best Compatible Tour:**(0,1,2,3,5,4,6,7)**.

Chapter 6

Construction of an "almost" compatible tour

6.1 Outline of our heuristic

In this chapter, we describe our heuristic for the ATSP which constructs a tour for the given ATSP with non-negative metric cost vector c. This heuristic uses the ideas from the cheapest insertion heuristic method for the STSP (see [9]), and finds an "almost" compatible tour T for an optimal solution x^* of the ASEP which corresponds to the given ATSP. Instead of satisfying all tight sets of x^*, T will be forced to satisf y an important subset of tight sets $\mathcal{TS}$ where the sets in $\mathcal{TS}$ are nested inside over one another. This nested structure of $\mathcal{TS}$ is used to guide the cheapest insertion heuristic in its insertion choices in such a way that the resulting tour T is tight for all sets in $\mathcal{TS}$.

Given a STSP on n nodes with a non-negative cost vector, the cheapest insertion method finds a tour by starting with a subtour, and extending this subtour by inserting the remaining nodes, one after the other, until all nodes have been inserted. At each step the node selected to be inserted into the subtour is the one with the

cheapest insertion cost at that time.

We have adapted the cheapest heuristic method to construct a compatible tour to satisfy the following additional constraints:

1. Subtour: The initial subtour consists of two selected nodes that do not belong to any of the tight sets.

2. Tight set: For each tight set, there should be only one edge entering into and leaving from it (i.e. the cut constraint is tight).

Given an ATSP on n nodes with non-negative cost metric c, the process of constructing a compatible tour is:

1. Find an optimal solution x^* of the ASEP which corresponds to the given ATSP.

2. Find a nested set $\mathcal{TS}$ of tight sets in x^*.

3. Construct an initial subtour consists of two nodes.

4. Every time, select a node with cheapest insertion cost to add to the subtour until all nodes are inserted.

5. At any point in time, the growing subtour should not violate any tight cut constraint for sets $S \in \mathcal{TS}$.

6.2 An optimal ASEP solution

For a given ATSP with non-negative metric cost vector c, we used the software called Concorde to get an optimal solution x^* of the corresponding ASEP. Concorde is a very powerful computer code designed specifically to work with the STSP. It was developed by David Applegate, Robert Bixby, Vasek Chvátal and William Cook. It can be freely downloaded for the academic purposes from http://www.keck.caam.rice.

edu/concorde.html. We will below explain how to adapt the problem to use this software for the ATSP.

Given a STSP $G = (V, E)$ and a vector $c \in R^E$ of edge costs, Concorde first formulates the Fractional 2-Factor problem which consists of finding $x \in R^E$ which minimizes cx subject to x satisfying the degree constraints (14) and the lower and upper bound constraints (15) defined in Chapter 1. The Fractional 2-Factor problem amounts to solving:

$$\text{minimize } cx \tag{82}$$

$$\text{Subject to : } x(\delta(v)) = 2 \quad \forall v \in V, \tag{83}$$

$$0 \leq x_e \leq 1 \quad \forall e \in E. \tag{84}$$

Each time while solving the above model, Concorde checks for the violated cuts in x^*. If found, Concorde resolves the same model with the violated cuts constraints added until there are no more violated cuts and an optimal solution x^* of the SEP is found.

Since Concorde works only on STSP, we do the following steps to get an optimal solution x^* of the ASEP which corresponds to the given metric ATSP. From now on in this thesis, we use y^* as an optimal solution of the SEP and x^* as an optimal solution of the ASEP.

1. We convert the ATSP instance to the STSP instance.
2. The Concorde solves the STSP and gives an optimal solution y^* of the SEP.
3. We convert the y^* of the SEP to an optimal solution x^* of the ASEP.

6.2.1 Converting the ATSP instance to the STSP instance.

We convert the ATSP instance to the STSP instance by adding extra nodes to the ATSP [9]. Each node in the ATSP instance will get replaced by 3 nodes in the STSP.

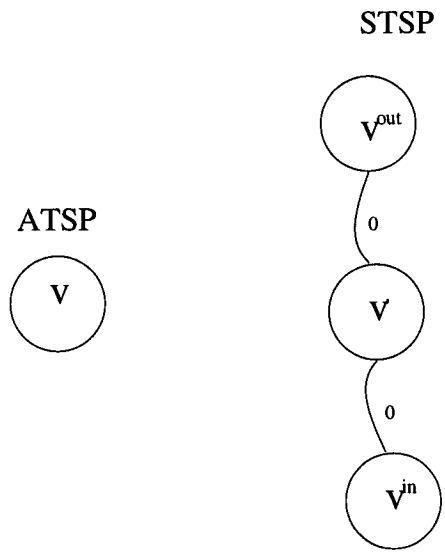

Figure 18: For each node v in ATSP the corresponding nodes in the STSP

Figure 18 shows the naming concept we used while converting. For each node v_i in the ATSP the corresponding nodes in the STSP are: $v_i^{in}, v_i^{'}$ and v_i^{out}. The steps we follow to convert each ATSP instance to STSP are:

1. The directed edge having v_i as tail and u_i as head in ATSP will be an edge between v_i^{out} and u_i^{in} in the STSP.

2. The directed edge having v_i as head and u_i as tail in ATSP will be edge between v_i^{in} and u_i^{out} in the STSP.

3. Edges of type $(v_i^{'}, v_i^{in})$ will have the cost 0 and edges of type $(v_i^{'}, v_i^{out})$ will also have cost as 0 in the STSP.

4. Edges between $v_i^{'}$ and all other nodes in STSP will have the cost infinity.

5. Edges of type (v_i^{in}, v_i^{out}) will also have cost as infinity.

6.2.2 Converting the solution y^* of the SEP to solution x^* of the ASEP

We convert y^* of SEP to the x^* of ASEP by replacing three nodes of type v_i', v_i^{in} and v_i^{out} (added while converting the ATSP instance to the STSP) in y^* by v_i in x^*. The SEP has following as one of the constraint,

$$x(\delta(v)) = 2 \qquad \forall v \in V.$$

Because of the 0 cost edges between every v_i^{in} and v_i' and between v_i' and v_i^{out} and large costs for all other edges from v', Concorde always makes $x_{v_i'v_i^{out}} = 1$ and $x_{v_i'v_i^{in}} = 1$ in y^*.

We follow the steps mentioned below for converting y^* to x^*:

1. For nodes of type v_i', v_i^{in} and v_i^{out} in y^*, we have v_i in x^*.
2. We ignore all edges of type (v_i^{in}, v_i') and (v_i', v_i^{out}) in y^*.
3. For edges of type (v_i^{out}, v_j^{in}) for all distinct i and j in y^* we have a directed edge (v_i, v_j) in x^*.
4. The value of y_e^* in y^* remains the same for the corresponding directed edge e in x^*.

6.3 Finding a nested set of tight sets in x^*

We find a smaller number of tight sets in x^* which imply the others as described in the following two subsections and later a nested set $\mathcal{TS}$ of these tight sets.

6.3.1 Finding the main set of tight sets in x^*

The final LP used by Concorde to find y^* contains a set of cut constraints sufficient to define y^*. The subset of these that are tight for y^* provide a set of tight sets such that any other tight set for y^* is implied by these plus the other constraints. We take advantage of this to obtain the important set of tight sets as follows:

1. As Concorde finds violated cuts constraints added to the LP, we write these to a file.

2. Convert y^* to x^* and cuts of y^* found in 1 to cuts of x^* and write them to a new file.

3. Remove all the cuts from new file if they are not tight for x^*.

6.3.2 Find all tight sets of size 2 in x^*

As Concorde does not consider the cut constraints $x(\delta(S))) \geq 2$ for $|S| = 2$ (it uses the equivalent form $x_e \leq 1$ instead), we also add to our file all cuts $x(\delta(\{u, v\})) \geq 2$ for all sets $\{u, v\}$ such that $x^*_{uv} = 1$ in x^* and name the final list as the list of tight sets.

6.3.3 Finding a nested set of tight sets

From the list of tight sets obtained in Section 6.3.1 and 6.3.2, we find a nested set $\mathcal{TS}$ of these tight sets. If two tight sets P and S cross, i.e. $P \cap S \neq \emptyset$, $P \backslash S \neq \emptyset$ and $S \backslash P \neq \emptyset$, then remove one of P or S from our list. We scan the list to remove a tight set if it crosses with some other tight set from same the list. At the end we have a nested set $\mathcal{TS}$ of tight sets for x^*.

Given a nested set of tight sets, we can partition these sets into families of sets that contained one inside the other i.e. the sets that have non-empty intersection and

are contained one inside the other; we call such a family a *properly nested set.*

Definition 6.1 *(A properly nested set) A nested set N of tight sets is* properly nested *if for all $I, R \in N$ sets we have $I \subset R$* or *$R \subset I$.*

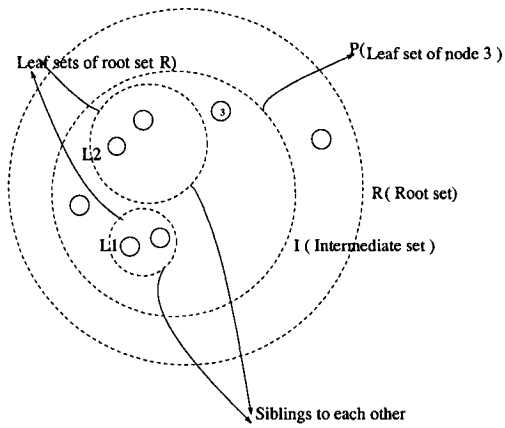

Figure 19: Properly nested tight sets: the *root set* , the *leaf set* and the *intermediate sets*

Some specific terminologies for the heuristic when R and I are properly nested tight sets with $I \subset R$ is the following (See Figure 19):

1. The tight set R is called a *parent* of the set I.
2. The tight set I is called a *child* of the set R.
3. If a tight set has more than one child, then each child is a sibling of the other child (L1 and L2).
4. The tight set which does not have a *parent* (the outer most tight set) is called the *root set* and the tight sets which do not have a *child* (inner most tight sets) are called *leaf sets* and all others are called *intermediate sets.*
5. The smallest tight set containing a node m is called the *leaf set of m* and the largest set containing node m is called the *root set of m.*

6. If the tight set does not have a parent nor child it is called an *orphan set* (plain tight set).

7. All the tight sets $P \subset R$ are called *descendants* of R.

8. For all tight sets S such that $S \subset R$, R is called an *ancestor* of S.

6.4 Insertion cost of a node

In what follows, we are defining the insertion cost of a node into a subtour and later we explain how we compute it.

Definition 6.2 *(Insertion cost of a node with respect to a directed edge in a subtour) The* insertion cost of a node *with respect to a directed edge in a subtour is the cost of inserting that node between end points of that directed edge.*

Let *new node* denote the most recently inserted node to the current subtour we are growing. To add a node between end points of a directed edge, the existing edge has to be replaced by two new edges. The direction of the two new edges will be same as the original edge. The tail of the original edge is the tail of the one of the new edge with its head being the new node. The head of the original edge is the head of the other new edge with its tail being the new node (see Figure 22).

Computing the insertion cost of a node

For a non-inserted node m, we compute the insertion cost over each edge in the subtour. The insertion cost of a node m with respect to a particular edge (u, v) from the subtour is

$$c(u, m) + c(m, v) - c(u, v).$$

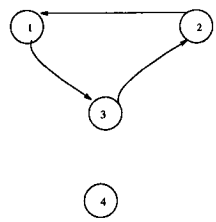

Figure 20: Part of existing problem

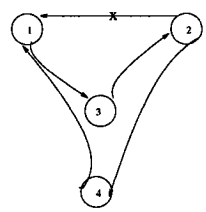

Figure 21: Insertion cost of node 4 with respect to the directed edge (2, 1)

Now the cheapest insertion cost of m is the smallest cost among the costs computed with respect to all edges in the subtour and an edge to be removed is the one which gave the smallest cost. Similarly, we find cheapest insertion cost for all non-inserted nodes and select the node which is cheapest among them.

Example 6.1 Figure 20 shows part of an ATSP instance along with the existing subtour $\{(1,3),(3,2),(2,1)\}$ and the non-inserted node 4.

Figure (21) shows the insertion of node 4 to the existing subtour by replacing the edge $(2,1)$ by two new edges $(2,4)$ and $(4,1)$. The cost of adding node 4 between 2 and 1 is :

$$c(2,4) + c(4,1) - c(2,1)$$

and the cost of new subtour can be found by the cost of old subtour:
Cost of new subtour = cost of old subtour + c(2, 4) + c(4, 1) - c(2, 1). Figure (22) shows the insertion of node 4 to the existing subtour by replacing the directed edge $(1,3)$ by two new edges $(1,4)$ and $(4,3)$. Figure (23) shows the insertion of node 4 to the existing subtour by replacing the edge $(3,2)$ by two new edges $(3,4)$ and $(4,2)$.

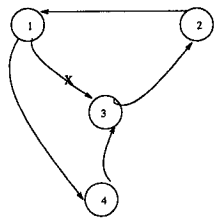

Figure 22: Insertion cost of node 4 with respect to the directed edge (1, 3)

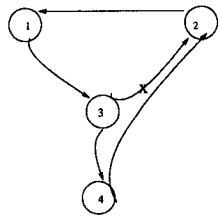

Figure 23: Insertion cost of node 4 with respect to the directed edge (3, 2)

The final minimum cost for node 4 is the minimum of all of these above.

6.5 Overview of the algorithm

The initial subtour of the compatible tour consist of two randomly selected nodes. We grow the subtour by inserting one node at a time. Every time, a cheapest insertion cost node m is selected and inserted to the subtour. To maintain the tight cut constraints in the subtour, we ensure that, immediately after adding new node m, we include all nodes from the root set for m in the subtour if m belongs to any of the properly nested tight sets; this is explained in more detail in Sections 6.5.1–6.5.5.

After every insertion of a new node m, before selecting a next node to be inserted, we find the following at each iteration i:

1. Node set S_i = {set of all non-inserted nodes to be considered at iteration i}.

2. Edge set T_i = {set of all edges in the subtour to be considered at iteration i}.

The insertion of any node from S_i replacing one of the edges from T_i should not violate any tight cut constraints. The detail description about these sets will be explained later more specifically.

The node m is one of the initially selected two nodes. The two nodes those do not belong to the properly nested tight sets are selected as initial nodes. After insertion of m, it could be one of the following five cases,

Case 1: The new node m does not belongs to any of the tight sets.

Case 2: The new node m belongs to an orphan set P.

Case 3: The new node m belongs to a leaf set.

Case 4: The new node m belongs to a root set.

Case 4: The new node m belongs to an intermediate set.

After updating the sets S_i and T_i, we follow the steps mentioned below to select a node to insert to the subtour:

1. Compute the insertion cost for each node in S_i with respect to each edge in T_i.
2. Find the cheapest insertion cost for every node in S_i with respect to the edges in T_i.
3. Find the node m from S_i which has the cheapest insertion cost among every node in S_i.

6.5.1 Growing the subtour outside of all properly nested tight sets

While growing the subtour, if the new node m does not belong to any of the properly nested tight sets, then all non-inserted nodes from the node set V are considered (see

Figure 24). In this case

$$S_i = \{\text{all non-inserted nodes from set } V\},$$

$T_i = \{$edges in the subtour that do not have both ends lying inside any of the tight sets$\}$.

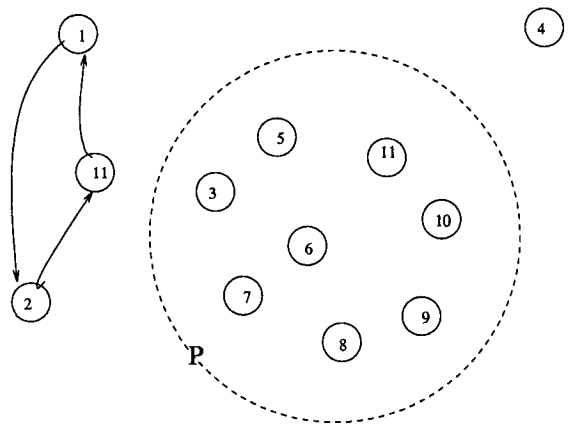

Figure 24: A new node m which is labeled 11 in this example does not belongs to any of the tight sets

6.5.2 Growing the subtour in a orphan set

If new node m belongs to an orphan set P, then all non-inserted nodes from the set P are considered and we insert all of them before inserting any other (see Figure 25). Until all nodes from P are inserted,

$$S_i = \{\text{all non-inserted nodes from set } P\},$$

$$T_i = \{\text{edges in the subtour with one or both ends in } P\}.$$

In Figure 25, the new node 3 is getting inserted. There is a directed edge entering the tight set P and one edge leaving it, which has to be maintained always. Now we

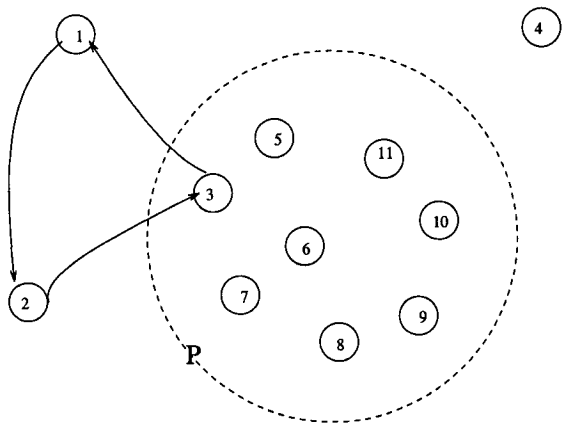

Figure 25: A new node m which is labeled 3 in this example is in an orphan set P.

want to insert all nodes from the tight set P. To select a next node to be inserted, S_i is $\{5, 6, 7, 8, 9, 10, 11\}$ and T_i is $\{(2, 3), (3, 1)\}$ If one of the edges incident with 3 gets replaced, then there is still exactly one edge entering the tight set P and one leaving it.

Compute the insertion cost

After computing insertion cost for each node in S_i with respect to each edge in T_i, the node m from S_i with minimum cost over all of them will be selected and the respective edge denoted by *replacing-edge* is the one which has to be replaced. If node 5 has a cheapest cost with respect to edge $(3, 1)$, then m is 5 and replacing-edge is $(3, 1)$.

Insert the selected node to subtour

Insert the selected node 5 to the existing subtour by replacing the edge $(3, 1)$ by two new edges $(3, 5)$ and $(5, 1)$. The length of the subtour will be increased by one and the tight cut constraint is satisfied by the subtour.

To insert the rest of the nodes from P, $S_{i+1} = \{S_i\} - \{m\}$ and $T_{i+1} = \{T_i\} \cup$

{newly added edges } – { replacing-edge}.

6.5.3 Growing the subtour in a leaf set

While growing the subtour, if a new node m belongs to a leaf set, then we consider all tight sets containing node m. Let L be the leaf set of m (the smallest tight set containing m) and R be the root set of m (the largest tight set containing m). All nodes from R are inserted before inserting any nodes from outside of it (see Figure 26).

We start inserting nodes from the set P until all nodes from L are inserted. In this case

$$S_i = \{\text{all non-inserted nodes from set } P\ \},$$

$$T_i = \{\text{edges in the subtour with one or both ends in } P\ \}$$

After inserting all nodes from P, we start inserting from its parent I. At every move

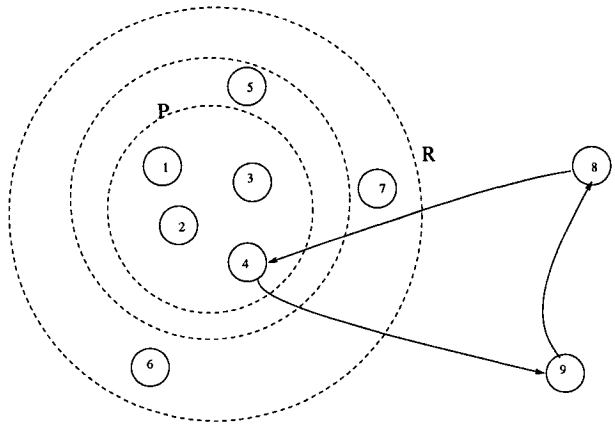

Figure 26: A new node 4 is from the leaf set P

from child set to parent set,

$$S_i = \{\text{all non-inserted nodes from the parent set }\},$$

$T_i = \{$edges in the subtour with one or both ends in I/edges having both ends in descendant of $I\}$.

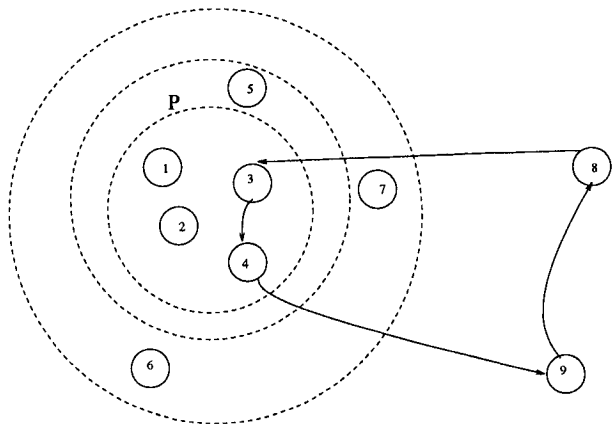

Figure 27: Next new node 3 is also from same leaf set P

Compute the insertion cost

We compute insertion cost for each node from set S_i with respect to each edge from set T_i. The node from S_i with minimum cost over all of them will be selected. After inserting all nodes from P, we go to insert from its parent. Other siblings of P are considered while inserting nodes from its parent.

If Figure 26 is a snapshot of the process of building a compatible tour, the new node 4 belongs to the set P, then to select the next node to be inserted,
S_i is {1, 2, 3} and T_i is {(8,4),(4,9)}. If node 3 has cheapest insertion cost with respect to edge $(8, 4)$ then m is 3 and the replacing-edge is $(8, 4)$.

Insert the selected node to subtour

Insert the selected node 3 to the existing subtour by replacing the edge $(8, 4)$ by two new edges $(8, 3)$ and $(3, 4)$. The number of edges in the subtour will be increased by one and the tight cut constraint is satisfied by the subtour.

Until all nodes from L are inserted,

$$S_{i+1} = \{S_i\} - \{m\}$$

and $T_{i+1} = \{T_i\} \cup \{\text{newly added edges }\} - \{\text{ replacing-edge}\}$ (see Figure 27).
Until all nodes from the parent of P are inserted,

$$S_{i+|L|} = \{\text{all non-inserted nodes from the parent}\},$$

$$T_{i+|L|} = \{\text{edges in the subtour with one or both ends in the parent}\}/$$

$$\{\text{ edges having both ends in } L\}.$$

We continue this way to insert nodes from all ancestors of L, until all nodes from R have been included in the subtour.

6.5.4 Growing the subtour in a root set

If the new node m is from a root set R, then all of the nodes from the root set R will be inserted into the existing subtour before inserting any nodes from outside R (see Figure 28). In this case

$$S_i = \{\text{all non-inserted nodes from set } R\},$$

$$T_i = \{\text{edges in the subtour with one or both ends in the set } R\}.$$

Start inserting nodes from the root set R one at time until a new node m belongs to any of the descendants of R and apply Case 3.

Compute the insertion cost

We compute the insertion cost for each node from set S_i with respect to each edge from set T_i. The node from S_i with minimum cost over all of them will be selected.

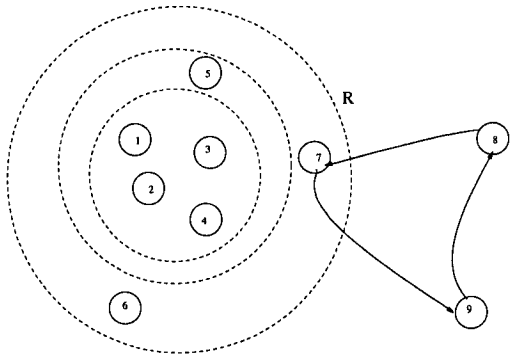

Figure 28: A new node is from the root set R

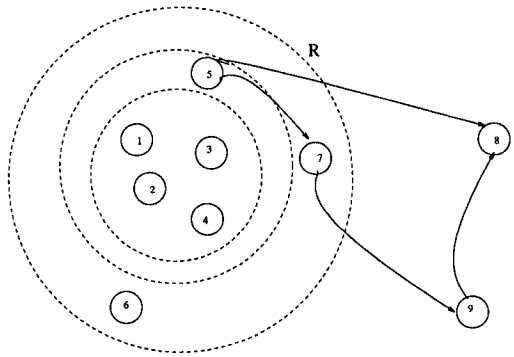

Figure 29: Next new node is one of node from the same root set R.

The node set S_i in Figure 28 is {1, 2, 3, 4, 5, 6}, the edge set T_i is {(8,5),(5,9)} and in Figure 29 the node set is : {1, 2, 3, 4, 6} and the edge set is {(8,5),(5,7),(7,9)}.

6.5.5 Growing the subtour in an intermediate set

If new node m is from an intermediate set I, then all the nodes from the root set of I, called R will be inserted into the existing subtour before inserting any nodes from outside R. To insert all nodes from I, descendants of I and root set of I, we apply the case 3 which is the case of inserting all nodes from the leaf set (see Figure 29).

6.6 Detailed example

In this section, we present an example of constructing a compatible tour for the ATSP using the heuristic just described. All different cases we consider while growing the subtour in a nested set of tight sets are explained in detail. The 17–city ATSP instance along with its metric cost matrix is considered. $G = (V, E)$ is a complete directed graph with $n = 17, V = \{1, 2, 3, ..., 17\}$, $E = \{(i, j) : \forall i, j \in V\}$ and a cost vector c is represented by an $n * n$ matrix, whose entries represent the cost of visiting city i from city j, which is given in Table 18

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
1	0	3	5	48	48	8	8	5	5	3	3	0	3	5	8	8	5
2	3	0	3	48	48	8	8	5	5	0	0	3	0	3	8	8	5
3	5	3	0	72	72	48	48	24	24	3	3	5	3	0	48	4	24
4	48	48	74	0	0	6	6	12	12	48	48	48	48	74	6	6	12
5	48	48	74	0	0	6	6	12	12	48	48	48	48	74	6	6	12
6	8	8	50	6	6	0	0	8	8	8	8	8	8	50	0	0	8
7	8	8	50	6	6	0	0	8	8	8	8	8	8	50	0	0	8
8	5	5	26	12	12	8	8	0	0	5	5	5	5	26	8	8	0
9	5	5	26	12	12	8	8	0	0	5	5	5	5	26	8	8	0
10	3	0	3	48	48	8	8	5	5	0	0	3	0	3	8	8	5
11	3	0	3	48	48	8	8	5	5	0	0	3	0	3	8	8	5
12	0	3	5	48	48	8	8	5	5	3	3	0	3	5	8	8	5
13	3	0	3	48	48	8	8	5	5	0	0	3	0	3	8	8	5
14	5	3	0	72	72	48	48	24	24	3	3	5	3	0	48	48	24
15	8	8	50	6	6	0	0	8	8	8	8	8	8	50	0	0	8
16	8	8	50	6	6	0	0	8	8	8	8	8	8	50	0	0	8
17	5	5	26	12	12	8	8	0	0	5	5	5	5	26	8	8	0

Table 18: Cost Matrix for the 17 city ATSP instance.

The support graph of the optimal solution x^* of the ASEP found for the corresponding ATSP is shown in Figure 30.

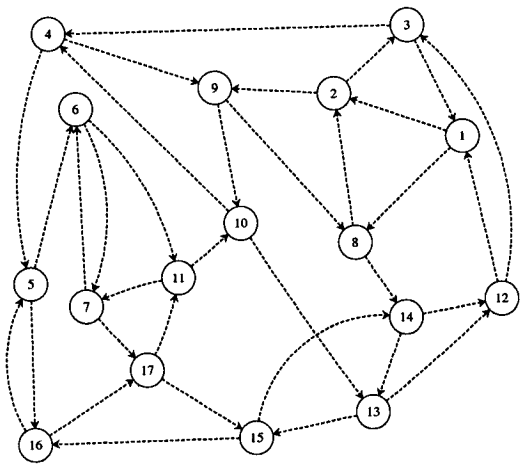

Figure 30: An optimal solution x^* of the corresponding ASEP.

All tight sets we have to consider for x^* are tabulated in Table 19.

6 7
6 7 11
6 7 11 17
5 6 7 11 16 17
5 6 7 11 15 16 17

Table 19: All tight sets in x^*.

The 17 city ATSP instance along with the nested set of tight sets is shown in Figure 31. We always compute insertion costs for two newly added edges and store them for later use as well. For the reader's convenience, in Figures (31)–(47) we have,

1. The tight sets considered are *circled.*
2. The newly added edges in iteration will be *darkened.*
3. The insertion cost tables will have node set S_i and edge set T_i considered at that time.

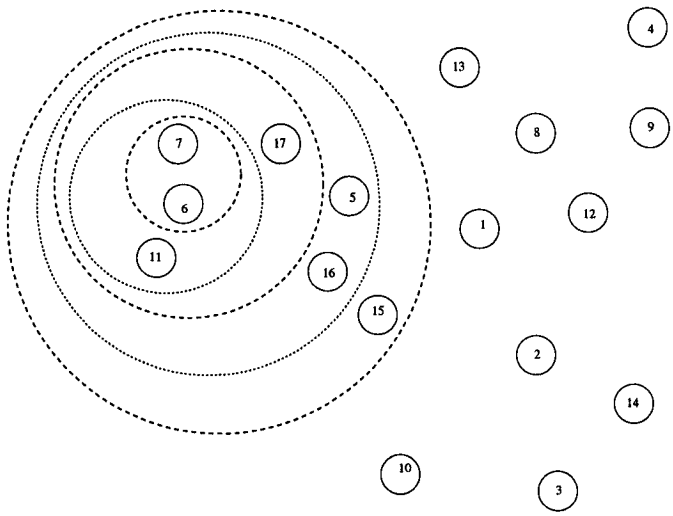

Figure 31: ATSP instance on 17 nodes with tight sets circled

The initial subtour has two randomly selected nodes 1 and 2 and edge set $\{(1,2),(2,1)\}$. The initial nodes 1 and 2 do not belong to the nested set of tight sets. The next node to be inserted can be any non-inserted node from node set V replacing either $(1,2)$ or $(2,1)$.

So node set $S_1 = \{3,4,5,6,7,8,9,10,11,12,13,14,15,16,17\}$, and

edge set $T_1 = \{(1,2),(2,1)\}$.

Since we only compute the insertion cost for newly added edges, the insertion cost for each node from set S_1 is computed over each edge in T_1. Node 10 has the cheapest

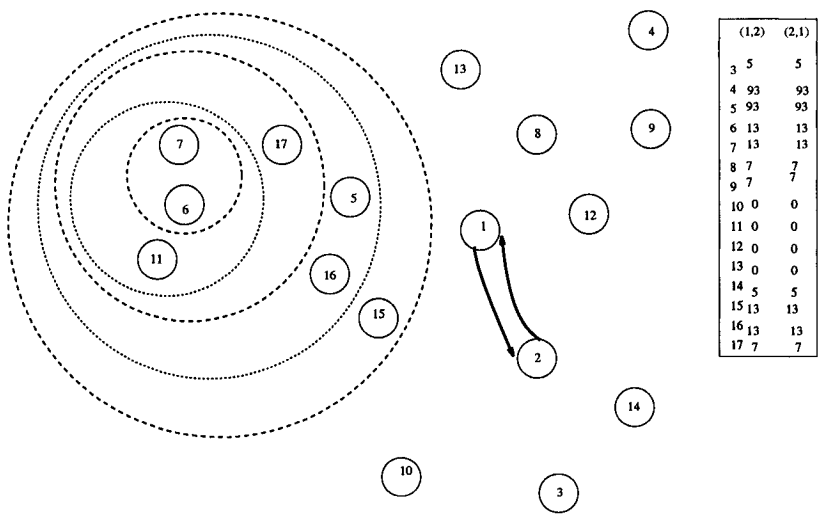

Figure 32: Initial subtour on nodes 1 and 2

insertion cost among all non-inserted nodes if it replaces either the edge $(1,2)$ or edge $(2,1)$ and Figure 33 reflects the inclusion of node 10 to the existing subtour by replacing the edge (1, 2) with two new edges $(1,10)$ and $(10,2)$ (shown with dark lines).

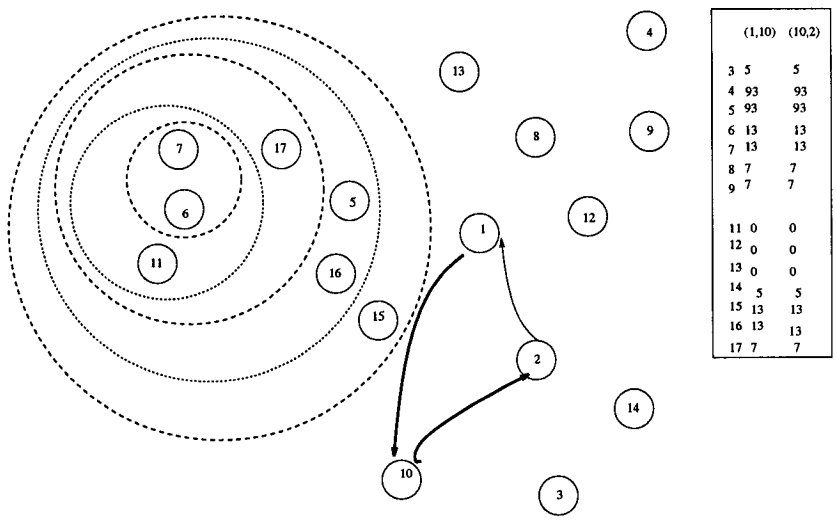

Figure 33: Node 10 is getting inserted to the subtour

The node 10 do not belongs to the nested set of tight sets, so the next node to be inserted to the subtour is any non-inserted node from set V.

So node set $S_2 = \{3, 4, 5, 6, 7, 8, 9, 11, 12, 13, 14, 15, 16, 17\}$, and

edge set $T_2 = \{(2, 1), (1, 10), (10, 2)\}$.

The insertion cost is computed for every node from S_2 with respect to edge $(1, 10)$ and $(10, 2)$ because we have already computed the insertion cost for the edge $(2, 1)$. Newly computed insertion costs are shown in Figure 33 and insertion cost of each node from set S_2 with respect to edge (2, 1) are shown in Figure 32. Node 11 has the cheapest insertion cost with respect to the edge $(1, 10)$ among all nodes from S_2.

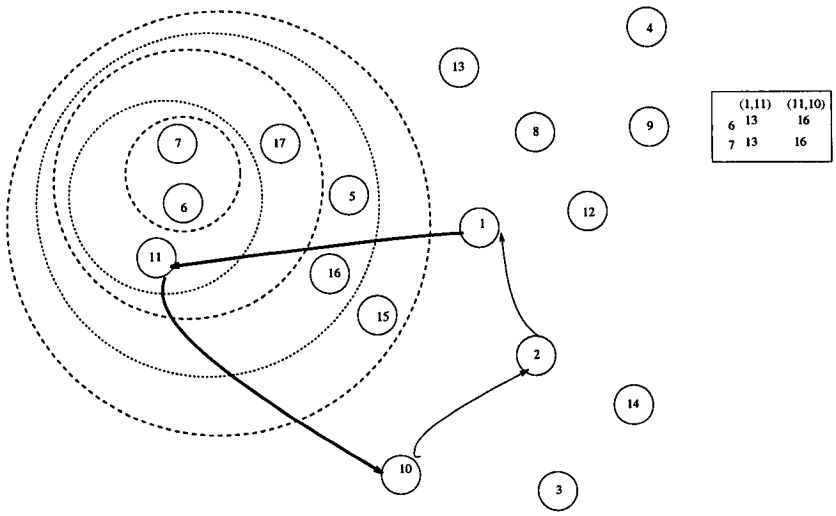

Figure 34: New node 11 is from the tight set

We include node 11 in the subtour by replacing the edge $(1, 10)$. Node 11 belongs to the family of properly nested set of tight sets:

$$\{5, 6, 7, 11, 15, 16, 17\},$$
$$\{5, 6, 7, 11, 16, 17\},$$
$$\{6, 7, 11, 17\},$$
$$\{6, 7, 11\}.$$

This is the case when new node 11 is from an intermediate set. The root set containing node 11 is $\{5, 6, 7, 11, 15, 16, 17\}$ and the leaf set containing 11 is $\{6, 7, 11\}$. Now we want to insert all nodes from the root set of 11, i.e. all nodes from set $\{5, 6, 7, 11, 15, 16, 17\}$. We start inserting nodes from the leaf set of 11, i.e all non-inserted nodes from the set $\{6, 7, 11\}$. Now

$$S_3 = \{6, 7\}, \text{ and } T_3 = \{(1, 11), (11, 10)\}.$$

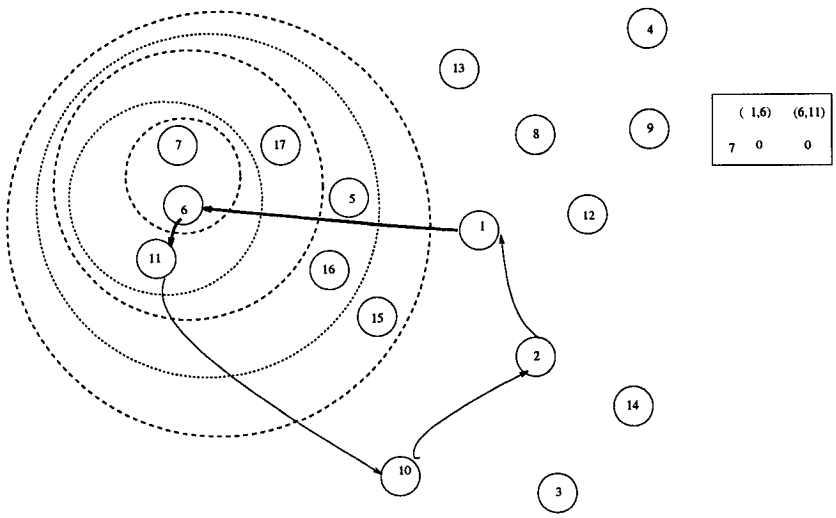

Figure 35: Node 6 from the same tight set is getting inserted

Now we compute insertion costs for nodes 6 and 7 with respect to edges $(1, 11)$ and $(11, 10)$ from T_3. The newly computed insertion costs are tabulated in Figure 34.

Node 6 has cheapest insertion cost with respect to the edge $(1, 11)$ among all nodes from S_3 and inclusion of node 6 to the subtour is shown in Figure 35.

To select the next node to be inserted, we have

$$S_4 = \{7\}, \text{ and } T_4 = \{(1,6), (6,11)\}.$$

The last node 7 from a leaf set $\{6, 7, 11\}$ gets inserted and its insertion costs are computed and tabulated in Figure 35. After inserting all nodes from the leaf set,

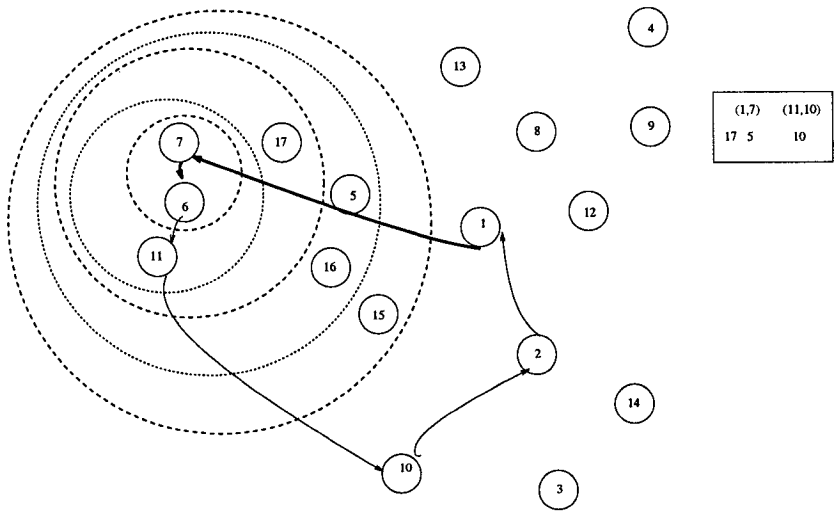

Figure 36: Node 7 is added to the growing subtour

we start inserting from its parent, but all the nodes from its parent set {6, 7, 11} are already inserted to the subtour. Now we move to insert nodes from one of the ancestor set {6, 7, 11, 17} and the only non-inserted node is 17.

$$\text{So } S_5 = \{17\}, \text{ and } T_5 = \{(1,7), (11,10)\}.$$

The insertion cost of node 17 over edges (1, 7) and (11, 10) are tabulated in Figure 36 and the inclusion of node 17 is shown in Figure 37 after replacing the directed edge (1, 7).

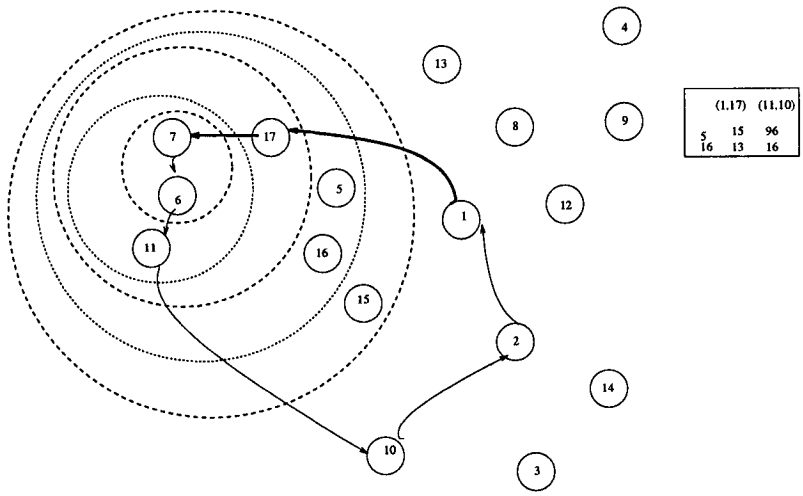

Figure 37: Node 17 is added to the growing subtour

To select the next node to be inserted, we have

$$S_6 = \{5, 16\}, \text{ and } T_6 = \{(1,17), (11,10)\}.$$

The insertion costs are tabulated in Figure 37 and the inclusion of node 16 is shown in Figure 38 after replacing the edge (1, 17) with two new edges $(1, 16)$ and $(16, 17)$.

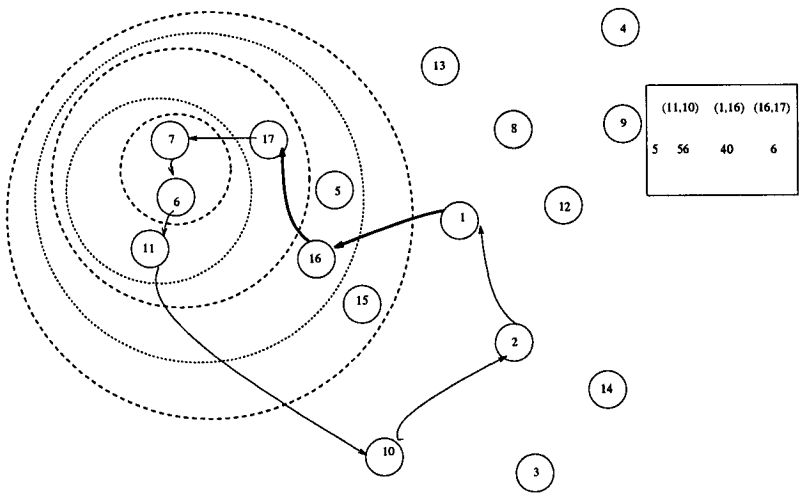

Figure 38: Node 16 is added to the growing subtour

To select the next node to be inserted, we have

$$S_7 = \{5\}, \text{ and } T_7 = \{(1,16), (11,10), (16,17)\}.$$

The insertion costs are tabulated in Figure 38 and node 5 has cheapest insertion cost with respect to the edge $(16, 17)$. The inclusion of node 5 is shown in Figure 39 after replacing edge (16, 17). Now we have

$$S_8 = \{15\}\ T_8 = \{(1,16), (11,10)\}$$

The insertion costs for node 15 are tabulated in Figure 39 and inclusion of node 15 is shown in Figure 40 after replacing edge (1, 16).

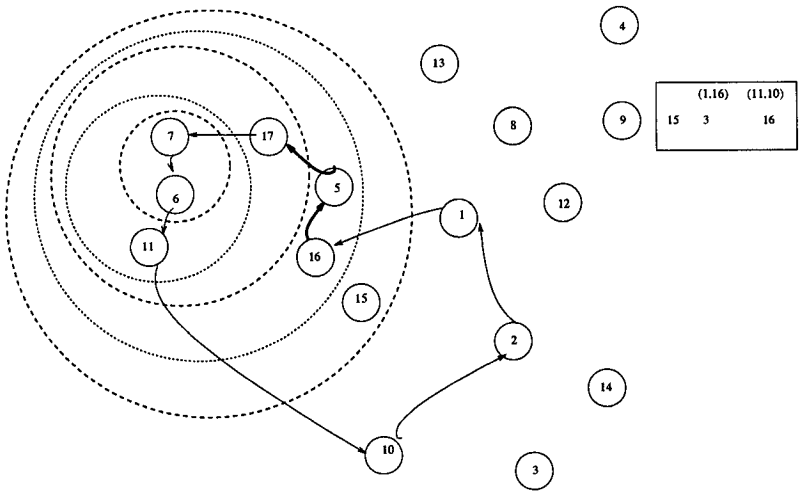

Figure 39: Node 5 is added to the growing subtour

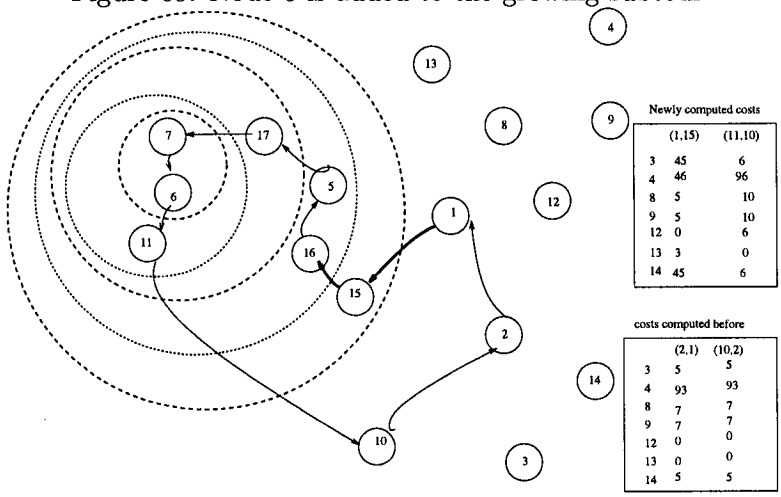

Figure 40: Node 15 is added to the growing subtour

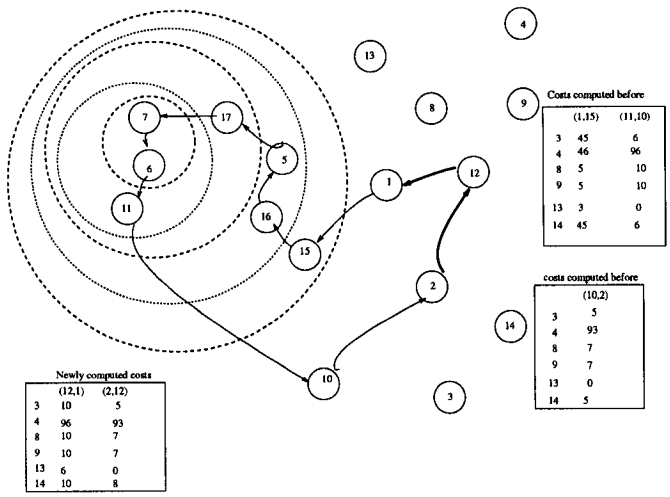

Figure 41: Node 12 is added to the growing subtour

Now there are no non-inserted nodes in the root set of node 11. So the next node to be insert is any non-inserted node from the set V, but the edges for which the insertion cost has to be computed are:(1, 15) and (11, 10) and the same is tabulated in Figure 40. S_9={3, 4, 8, 9, 12, 13, 14 }, and

T_9={(1,15), (11,10), (2,1)}.

Figures 42–47 shows the insertion of the rest of the nodes from the set S_9.

The compatible tour built is (1, 9, 8, 4, 15, 16, 5, 17, 7, 6, 11, 10, 14, 3, 2, 13, 12) and comparison of its cost with best tour found so far is tabulated in Table 20.

Cost of the compatible tour obtained by our heuristic	Cost of the tour obtained by Concorde	Cost of the best compatible tour
66	39	48

Table 20: Comparison table.

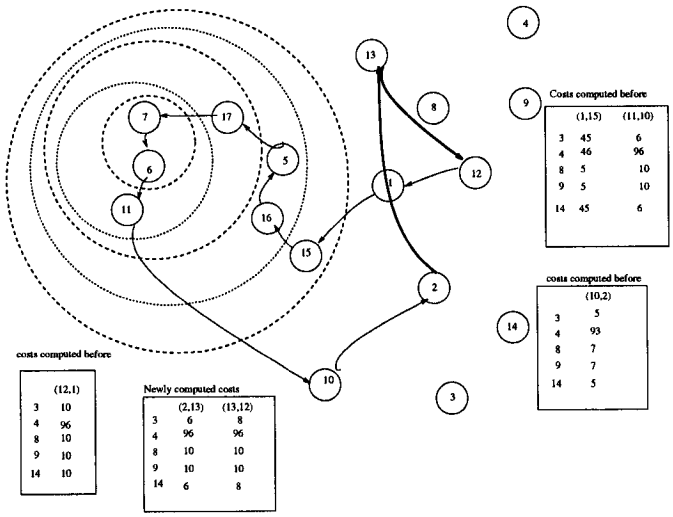

Figure 42: Node 13 is added to the growing subtour

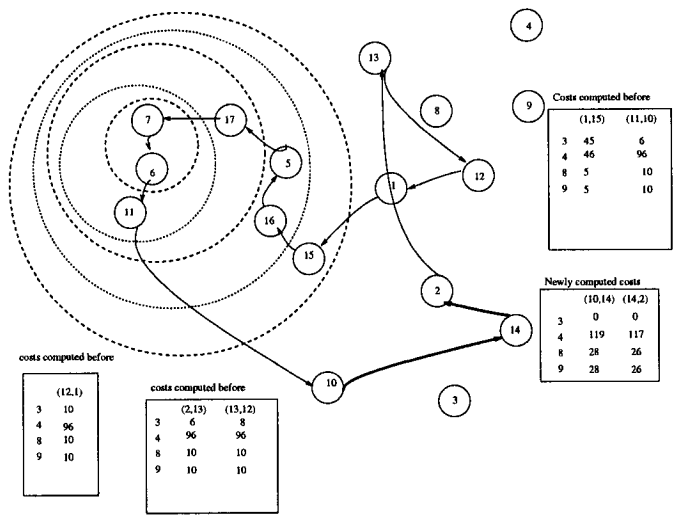

Figure 43: Node 14 is added to the growing subtour

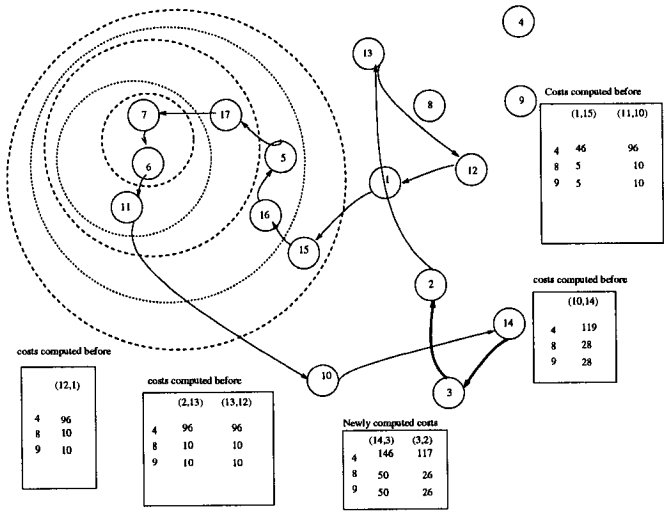

Figure 44: Node 3 is added to the growing subtour

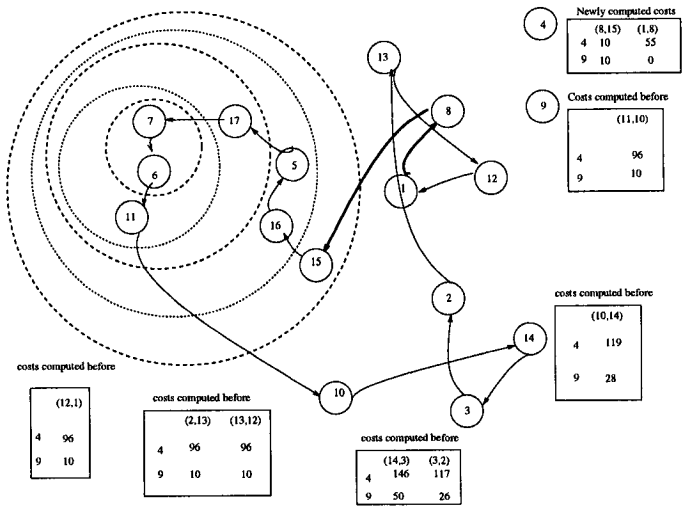

Figure 45: Node 8 is added to the growing subtour

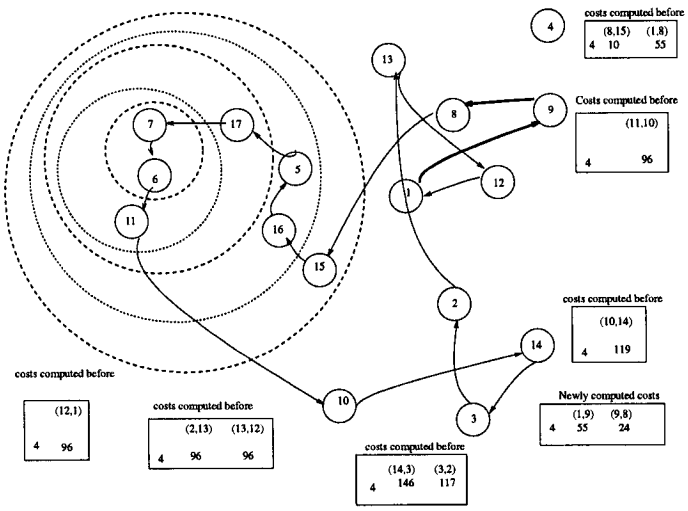

Figure 46: Node 9 is added to the growing subtour

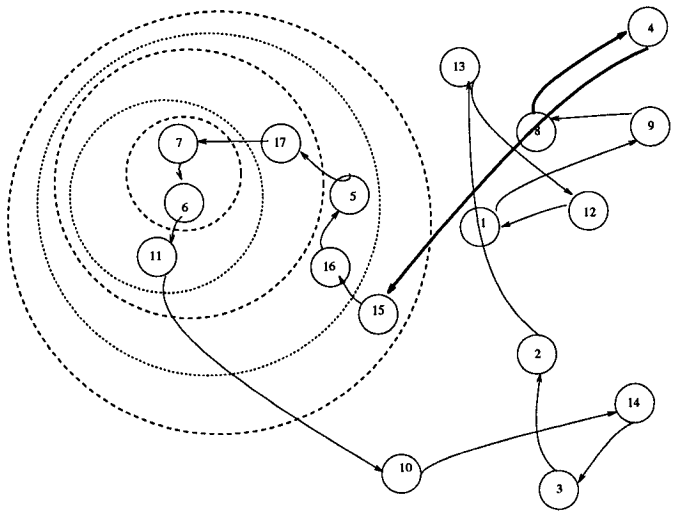

Figure 47: Node 4 is added to the growing subtour

Chapter 7

Computer implementation

In this chapter, we give detailed description of all the algorithms we developed in the process of computing the worst–case ratio between CTP(c) and ASEP(c), between CTP(c) and ATSP(c) and to construct a compatible tour. First, we present and explain the pseudo-code for finding all the tight sets in an extreme point x^* of the ASEP polytope that are used to compute β_n and γ_n in Chapters 4 and 5 when n is small. Second, we present and explain the generation of all compatible tours for x^* that are used to compute β_n and γ_n in Chapters 4 and 5 when n is small. Third, we present the heuristic to construct an "almost" compatible tour and discuss its computer implementation in detail.

7.1 Generating all tight sets for extreme point x^* of the ASEP polytope

We find all tight sets in x^* in Q_S^n when n is small and x^* is given. Exhaustively, we generate all subsets of V of size 2 to $\lfloor n/2 \rfloor$ and check each subset S for its out-going edge value as 1, i.e. $\delta^{out}(S)$ is 1. There can be an exponential number of subsets for V

and an exponential number of tight sets in x^*. The pseudo-code for this is described in Algorithm 1.

Pseudo-code for finding all the tight sets

Tight-sets(x^*, V, n, $\mathcal{TS}$)

Input: An optimal solution x^* of ASEP
Number of nodes n in ASEP.
Given node set V in ASEP
Output: Set of all tight sets, denoted by $\mathcal{TS}$.

```
1: For V, find all subsets of size 2 to ⌊n/2⌋.
2: for each subset p do
3:    compute total out-going edge value of p
4:    if total out-going edge value is equal to 1 then
5:       store p as a tight set in TS.
6:    end if
7: end for
```

Algorithm 1: Algorithm to find all tight sets

Computer implementation of the for finding all tight sets

This algorithm is implemented using the C language on a Unix platform. We find all subsets of V and the data structure used to store them is a linked list. Each node has fields like, node-value and pointer to next node. The following Figure 48 shows the structure of the node.

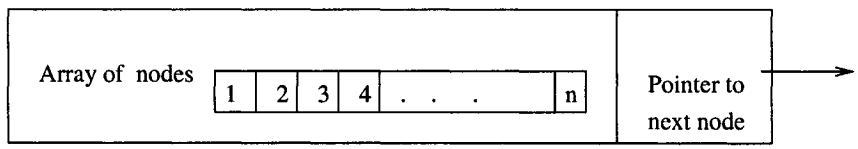

Figure 48: Node structure used in algorithm to find all tight sets

The number of nodes in a linked list will be the number of possible subsets of V of size 2 to $\lfloor n/2 \rfloor$. The node-value is an array indexed by node set V. That is, the node-value[i] $\in \{0,1\}^V$ and node-value represents the set of nodes for which node-value[i]=1.

When $n = 4$, if the node-value[1]=1, node-value[2]=0, node-value[3] = 0 and node-value[4]= 1 then the subset is {1,4}.

The edge value is 0 if both of i and j has node-value as 1, otherwise it is the value of going from i to j. For the set {1, 4}, the directed edges to consider are c(1,2), c(1,3), c(4,2) and c(4,3). We compute the total edge value and if it is 1, then it is a tight set.

7.2 Generating all compatible tours for x^* when n is small

We exhaustively generate all compatible tours for small n, $4 \leq n \leq 17$. For each $x^* \in Q_S^n$, first, we find all tight sets in x^* as explained in Section 7.1. Next we generate all possible tours satisfying (7)-(11). Finally, we select each tour which is compatible with x^* i.e. if tour additionally satisfies the constraint (37) for set of tight sets TS which need not to be nested. The pseudo-code for this is described in Algorithm 2.

Pseudo-code for generating all compatible tours when n is small

For n nodes, the number of possible tours is $(n-1)!$ and the set of possible tours is denoted by $\mathcal{T}$. Each tour $T \in \mathcal{T}$ is checked to see its compatibility with each tight set $p \in \mathcal{TS}$. If tour T is not compatible with any one of the tight sets p, then we immediately decide that T is not a compatible tour and go to see the next tour in $\mathcal{T}$

Compatible-Tours($\mathcal{TS}$, V, n, $\mathcal{T}_{comp}$)

Input: Set of all tight set TS in the ASEP solution x^*
Number of nodes n in ASEP.
Given node set V in ASEP
Output: List of all compatible tours $\mathcal{T}_{comp}$ for given x^*

```
1: for each tour T do
2:   for each tight set p ∈ TS do
3:     if tour T is not compatible with tight set p then
4:       go to step 2
5:     end if
6:   end for
7:   store tour T as compatible tour in set T_comp.
8: end for
```

Algorithm 2: Algorithm to generate all compatible tours

until all tours are considered.

7.3 Construction of an "almost" compatible tour

For a given ATSP on n nodes with a non-negative metric cost vector c, we construct an "almost" compatible tour compatible with a nested subset $\mathcal{TS}$ of the tight sets for an optimal solution x^* of the ASEP which corresponds to the given ATSP. After finding an optimal solution x^* of the ASEP and a nested set of important tight set $\mathcal{TS}$ in x^*, we build an initial subtour with two selected nodes a and b.

We add a node at time to the current subtour by cheapest insertion method until all nodes are inserted. After every insertion, the growing subtour should be tight with all tight sets considered so far. Before selecting a node to insert to the subtour, we compute set S_i, which is a subset of non-inserted nodes to consider from node set V at iteration i and an edge set T_i which is a subset of the edges to consider to replace from the subtour at iteration i. The insertion of any node from S_i replacing

an edge from T_i will not violate the tight constraint in the subtour. Insertion cost is computed as specified in Section 6.4 for each non-inserted node in S_i with respect to each edge in T_i.

The heuristic was implemented in the ANSI C compiler on a Linux platform. The size of the code is more than 2000 lines and the main data structures used in the implementation are tree and linked list. The pseudo-code for this is described in Algorithm 4. The testing was done on a SunW UltraSPARC-II Server running Unix with a singla 400 Mhz processor. The complexity of this algorithm is based on the following four main operations we do to find an "almost" compatible tour:

1. **Building a forest where each component represents a properly nested tight set.**

   We store each properly nested tight set as a connected component in a forest. Each node belongs to only one properly nested tight set. The memory used to store each properly nested tight set is made free soon after inserting all nodes from it.

2. **Select a node set S_i and edge set T_i before selecting any node to insert.**

   If a new node does not belong to any of the tight sets then the S_i contains all non-inserted nodes from the set V. To select a node set S_i for the first time when a new node m is selected from a properly nested tight set, the reference array indexed by node set is used where array[m] refers to the root of node m. There can be at most $n/2$ nodes in a tight set. We keep finding the set S_i with nodes from that component until all nodes from that connected component are inserted. Every time we want to select a S_i while inserting all nodes from the properly nested tight set to which node m belongs, we use a reference array to see if new node is from any of the descendant's set or ancestor set of the current node set S_i. So the total time complexity for this operation is $O(n^2)$.

The time complexity to select an edge set T_i for the first time when a new node m is from a properly nested tight set is $O(1)$ because that time T_i is going to contain only newly added edges to the subtour. Every time we want to select a edge set T_i while inserting all nodes from the properly nested tight set to which node m belongs, we consider edges from the previous T_i that are incident with nodes from current S_i , edges in the subtour incident with the nodes of currently handing properly nested tight set and among two newly added edges. We separately store the edges those are from the subtour incident with the nodes of the currently handing component and there can be at most $n/2 - 1$ edges and the maximum size of the set S_i is also $n/2$. So the total time complexity for this operation is $O(n^2)$.

3. **Compute insertion cost of all non-inserted nodes with respect to two most recently added edges.**

   The tedious work of computing insertion cost is drastically reduced by storing the previously computed results for all edges in the subtour. For each edge, we maintain the list of all non-inserted nodes along with their insertion cost with respect to that edge until all nodes are inserted.

   It takes $O(n)$ time for this operation because we compute insertion cost for all non-inserted nodes with respect to two edges included most recently to the subtour.

4. **For each edge, scan the insertion cost list before selecting a node to insert.**

   For each edge, we scan the list of non-inserted nodes along with insertion cost before selecting a cheapest insertion cost node. The most recently inserted node has to be removed from all the insertion cost lists immediately after its insertion. The number of non-inserted nodes decreses every time by 1 and number of edges

on the subtour increases by 1. So over the entire heuristic, It takes $\Sigma_{i=1}^{n-2} i(n-i)$ steps in total to compute S_i and T_i until all non-inserted nodes are inserted. So this step is $O(n^3)$.

5. Overall the complexity of the heuristic based on the operations explained above is $O(n^3)$.

7.3.1 Representing each tight set as a connected component

Each properly nested tight set is represented as a connected component (tree), at the end we have forest of connected components. The pseudo-code how we build the forest and an example are shown below. One example of connected component built for a properly nested set of tight sets is shown if Figure 7.3.1.

build-forest($\mathcal{TS}$)

Input: Nested set $\mathcal{TS}$ of tight sets in x^* of ASEP solution.
output: Forest of properly nested tight sets.

1: **if** element a of a tight set I is already in connected component of a growing forest **then**
2: R= {the root set of a.}
3: P= {all nodes from I } $\cap$ { all nodes from R.}
4: **if** $P == I$ **then**
5: all node from I will have new parent with value -1, which will be child of old parent of nodes from R.
6: **end if**
7: root of R and all elements from P will have a new root with value -1.
8: **else**
9: create a connected component of elements of tight set I, making children of the new root with value -1.
10: **end if**

Algorithm 3: Algorithm to represent all properly nested tight sets

1,2
3,4
8,9
15,16
19,20
17,18
11,12
13,14
1,2,3,4,5,6
1,2,3,4,5,6,7,8,9,11,12,13,14

Table 21: Example of properly nested tight set.

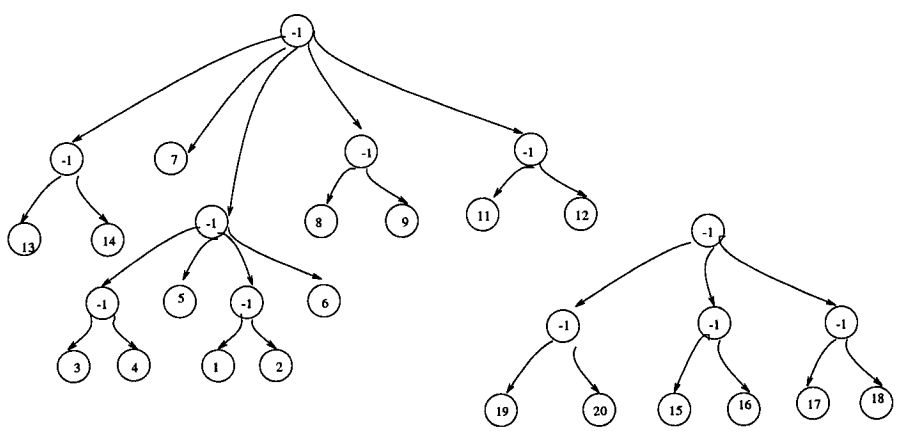

Figure 49: Forest

Compatible-Tour$(G, c, n, \mathcal{TS})$

Input: A graph G=(V, E).
Non-negative metric cost vector c.
$\mathcal{TS}$ is the nested set of tight sets.
i is an iteration number
Output: A compatible tour is built.

1: Build initial subtour with any two inputed nodes a and b.
2: **if** first node a belongs to any of the tight set **then**
3: /*Insert all nodes from the root set R containing node a before - -adding any other node outside root */.
4: **while** Not all nodes from root set R containing i are inserted **do**
5: Find node set S_i and edge set T_i.
6: Compute insertion cost for all nodes in S_i with respect to each edge in T_i.
7: Select a node $m \in S_i$ with cheapest insertion cost.
8: insert node m to the subtour.
9: i=i+1
10: **end while**
11: **end if**
12: $m = b$
13: **while** Length of the subtour is less than n **do**
14: **if** m belongs to any of the tight set **then**
15: /*Insert all nodes from the root set R containing node m before - -adding any other node outside root */
16: **while** All nodes from root set R containing m are not inserted **do**
17: Find node set S_i and edge set T_i
18: Compute insertion cost for all nodes in S_i with respect to each edge in T_i.
19: Select a node $m \in S_i$ with cheapest insertion cost.
20: insert node m to the subtour.
21: i=i+1
22: **end while**
23: **else**
24: Find node set S_i and edge set T_i
25: Compute insertion cost for all nodes in S_i with respect to each edge in T_i.
26: Select a node $m \in S_i$ with cheapest insertion cost.
27: insert node m to the subtour.
28: i=i+1
29: **end if**
30: **end while**

Algorithm 4: Algorithm to construct a compatible tour

Chapter 8

Computational Results

In this chapter, we test the heuristic which is developed in this thesis. In particular in Section 8.1 of this chapter, we compare our results with the best tours found so far for each problem we considered. The tests performed were executed on of all metric ATSP instances obtained by TSPLIB. The TSPLIB is a collection of several different kind of TSP's from real world problems. Note that the 19 problems we looked at ranged from 17 to 443 nodes.

Sections 6.2– 6.3 explain how each required input data used to test our heuristic is collected.

The flow chart in Figure 51 shows how we were able to construct a compatible tour for all metric ATSP instances obtained by TSPLIB. In this flow chart, softwares and other input data available to us are put in squares and programs written by us are put in circles.

In Figure 51, the converter *'C1'* is a program written to convert all ATSP instances to STSP instances. The converter *'C2'* is a program written to convert an optimal SEP solution to an optimal ASEP solution. The converter *'C3'* is a program written to convert all SEP cuts found as violated cuts while solving SEP by Concorde to corresponding ASEP cuts. Among cuts in x^* of ASEP, the program *"check for tight*

cuts" keeps only tight cuts of x^*. Among all tight cuts in x^* of ASEP, the program *"find nested set of tight sets"* keeps only non-crossing tight cuts.

For a given ATSP on n nodes and non-negative metric cost vector c, an optimal solution x^* is obtained by Concorde software available on the web for academic purpose (mentioned previously in Section 6.2). Concorde is a powerful code for solving STSP instances. We utilized this software to obtain the solution x^* for ATSP by converting all of our ATSP instances to the STSP instances; we explained how we converted them in Section 6.2.1. Concorde solves only STSP instances and gives either tour and cost of that tour or an optimal solution y^* of the SEP for the corresponding STSP.

While solving the converted STSP instances, we asked Concorde to give a y^* of the SEP instead of a tour for STSP. Concorde formulates the SEP as a fractional 2-Factor problem [1] and solves it by repeatedly adding violated subtour elimination cuts. We collected all violated cuts Concorde was finding while solving the SEP. If any of the cut has size more than $\frac{n}{2}$, we considered its complement set of it.

The solution y^* of SEP is converted back to a solution x^* of the ASEP (see Section 6.2.2). All collected violated cuts are also converted to ASEP cuts. The cost of an "almost" compatible tours found are tabulated along with the best tour found so far for each ATSP instance from the TSPLIB website.

8.1 Testing the usefulness of the heuristic developed in Chapter 6

The cost of the "almost" compatible tours found for all real time problems from the TSPLIB website are compared with the cost of the best tours found so far for the same problems. In Table 22, the first column is the problem name, second part

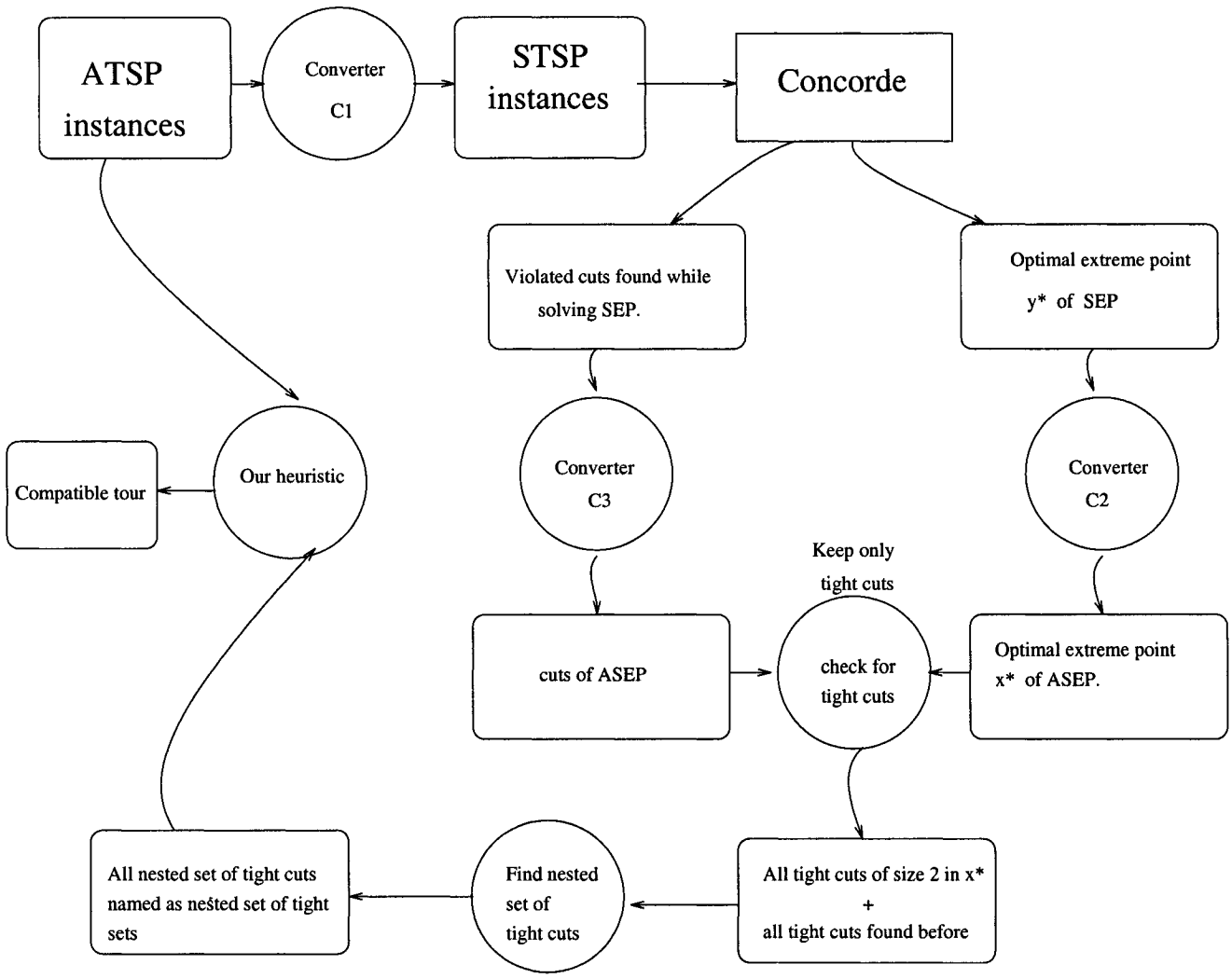

Figure 50: The process how we built a compatible tour for all real time problems from the TSPLIB website

of which represents the number of nodes in that particular problem. The TSBLIB website has the cost of best tours found so far for all ATSP instances we worked on. We also solved all ATSP instances using Concorde to get a tour for each problem and the cost of the tour obtained by Concorde were the same as those reported as the best tours found so far on the TSPLIB website. The computed percentage gap for all problems are tabulated in Table 23 and the average percentage gag is 81.08274. The percentage gap for each problem is,

$$\text{Percentage gap} = \frac{\text{Cost of compatible tour - cost of best tour}}{\text{cost of best tour}} * 100$$

ATSP instance	Compatible tour	Best tour found so far	No of tight sets considered
ft53	9967	6905	51
ft70	47436	38673	72
ftv33	1654	1286	41
ftv35	2478	1473	33
ftv38	3287	1530	45
ftv44	2431	1613	42
ftv47	2877	1776	51
ftv55	2723	1608	53
ftv64	2987	1839	70
ftv70	3673	1950	83
ftv90	2987	1579	101
ftv100	2389	1788	120
ftv110	3988	1958	111
ftv120	4512	2166	119
ftv130	4976	2307	120
ftv140	5673	2420	151
ftv150	5433	2611	135
ftv160	5021	2683	156
ftv170	5398	2755	179
kro124	50341	36230	130
p43	7437	5620	47
rbg323	3646	1326	330
rbg358	3221	1163	370
rbg403	3267	2465	419
rbg443	5987	2720	442
ry48p	20876	14422	53

Table 22: Comparison Table.

ATSP instance	Percentage gap
ft53	44.34467777
ft70	22.65921961
ftv33	28.61586314
ftv35	68.22810591
ftv38	114.8366013
ftv44	50.71295722
ftv47	61.99324324
ftv55	69.34079602
ftv64	62.4252311
ftv70	88.35897436
ftv90	89.17036099
ftv100	33.61297539
ftv110	103.6772217
ftv120	108.3102493
ftv130	115.6913741
ftv140	134.4214876
ftv150	108.0811949
ftv160	87.14125978
ftv170	95.93466425
kro124	38.94838532
p43	32.33096085
rbg323	174.9622926
rbg358	176.9561479
rbg403	32.53549696
rbg443	120.1102941
ry48p	44.75107475

Table 23: Percentage gap for all ATSP problems solved.

8.1.1 Testing the usefulness of the tight sets of size 2

Since the percentage gap obtained for all the "almost" compatible tours is quite high, instead of using a nested set of tight sets we decided to investigate the importance of the sets {u, v} of size 2 with $x_{uv} = 1$ in an optimal ASEP solution. To find this out, we again used Concorde more than once for each problem to get a final ASEP solution where we have only edges of type $x_{uv} = 1$ enforced.

For each ATSP instance, we first find an optimal ASEP solution x^* using Concorde. Next we find all the connected components from the support graph of x^* that consist of directed paths with having all edges (u, v) having $x^*_{uv} = 1$ Now we create a new problem which consists of nodes u not in any of the directed 1-path components in x^* and a pseudo–node created by shrinking each directed path of 1^s found. Now the problem is resized with all nodes of type u and pseudo–nodes. The cost to the pseudo–node from all other nodes is the cost to the tail node of the corresponding directed path of that pseudo–node. The cost from the pseudo–node to all other nodes is the cost from the head node of the corresponding directed path of that pseudo–node. We keep track of the node labels which are different in the new problem from the original one. The new problem is also converted to an STSP and solved by Concorde. We again do same as before for the new ASEP solution until all edges (u, v) have $x_{uv} = 1$. Finally when considering all edges (u, v) with $x_{uv} = 1$, we get a tour because of constraint (19).

We could test this for only 3 problems from TSPLIB website, because it was very difficult to keep track of node labels in resized problems when Concorde takes more than two iterations to find the final ASEP solution. The cost of the tours obtained are tabulated in Table 24.

The cost of the tour obtained for the 17–city problem is the same as the cost of the best tour found so far. It shows that the edges (u, v) with $x_{uv} = 1$ have some strong meaning in the ASEP solution and these tight sets seem to be more important

ATSP instance	Compatible tour
br17	39
ftv33	1518
ftv35	2351

Table 24: Comparison Table.

for the optimal ATSP solution.

The tour obtained for the two problems satisfies all the tight sets (u, v) with $x_{uv} = 1$ found in the first iteration of the ASEP solution given by Concorde, but later may be due to resizing the problem with the new cost vector, we get a different tour than the one closest to the best.

The solution of the final ASEP given by Concorde for the problem on 17 nodes is shown in Table 25 and considering all edges (u, v) with $x_{uv} = 1$, we obtain a tour (12, 1, 14, 3, 13, 10, 2, 11, 6, 15, 16, 7, 4, 5, 8, 17, 9) which gives the total cost of 39.

Node u	Node v	value of x_{uv}
12	1	1.000000
9	12	1.000000
17	9	1.000000
8	17	1.000000
4	5	1.000000
7	4	1.000000
16	7	1.000000
6	15	1.000000
15	16	1.000000
2	11	1.000000
10	2	1.000000
13	10	1.000000
3	13	1.000000
14	3	1.000000
5	8	1.000000
11	6	1.000000
1	14	1.000000

Table 25: An optimal ASEP solution for $n = 17$ with node labelling in original problem

Chapter 9

Conclusion and Future work

In this thesis, we have developed a heuristic for the Asymmetric travelling salesman problem. The key idea used in this thesis is to find a tour which will be "tight" in most of the places where x^* of the corresponding ASEP solution is tight. For this, we defined a new problem called Compatible Tour Problem (CPT) which is the problem of finding a minimum cost compatible tour for each given ATSP.

Since it is suspected that CTP is an NP-hard problem, in this thesis we developed a heuristic to find a good solution to the CTP with the aim of finding good solution to the ATSP. We tested our heuristic by finding "almost" compatible tours for all real time ATSP problems from the TSPLIB website. Other than the cost vector, the required inputs for finding a compatible tour are an optimal solution x^* of corrosponding ASEP and important tight sets in x^*. We successfully collected all the required data using software called Concorde.

We also determined the theoretical gap between an optimal compatible tour and an optimal ATSP tour and the gap between an optimal compatible tour and an optimal ASEP solution for small n. The obtained gaps were compared with the integrality gaps of ATSP for certain small n.

In Chapter 8 we discussed the computational results comparing the cost of the

"almost" compatible tours with the cost of the best tours known and calculated the percentage gap of each problem. We suspected that the cost of an optimal compatible

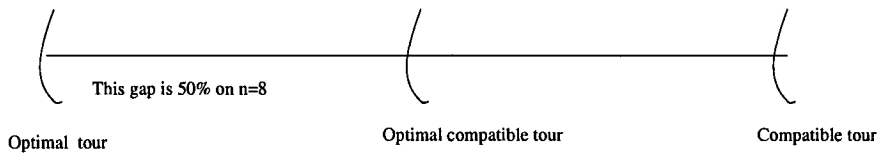

Figure 51: ATSP(c) and CTP(c) are far apart.

tour may be far from the cost of our "almost" compatible tour. But because the computed gap between CTP(c) and ATSP(c) is 50% for $n = 8$, we suspect that the cost of an optimal ATSP tour is far from the cost of optimal compatible tour. So trying to find a best compatible tour may still produce a big gap.

We have the following ideas for possible future work on this problem:

1. As discussed in Section 8.1.1, the edges e with $x_e = 1$ seem to be important for the optimal ATSP solution. It would be good to test this idea further. One possible method would be to shrink all the directed 1-paths from the ASEP solution x^* (as discussed in Section 8.1.1) and then simply solve the corresponding ATSP using Concorde. If this done carefully, the tour found by Concorde, along with the 1-paths that were originally shrunk, will give a solution for for the original ATSP. The solution found this way could then be compared to the test ATSP solution known for the best problems used.

2. It is possible that we get different "almost" compatible tours if we start with a different pair of starting nodes. It would be useful to which starting pair of nodes works better. Some possible types of starting pairs of nodes which could be tested are as follows:

   - Both the nodes from outside of all the tight sets.

- One from outside of all tight sets and the other one from any of the leaf sets in a properly nested tight set.
- One from outside of all tight sets and the other one from any root set in a properly nested tight set.
- Both of them from two different tight sets.

3. Although it is suspected that the CTP is an NP-hard problem for both the STSP and the ATSP, this has not been proven for either problem. It would be useful if such a proof could be found. Note that knowing that the CTP is NP-hard for the ATSP may imply that it is also NP-hard for the STSP due to the fact that we can transform an ATSP instance into an STSP instance as discussed in Chapter 6.

4. It would be useful to investigate the structure of the extreme points of the ASEP that attained the maximum gap for $\frac{CTP(c)}{ASEP(c)}$ and $\frac{CTP(c)}{ATSP(c)}$ to see if there are any patterns that are common in these extreme points. In this way it might be possible to establish some information regarding the values of β_n and γ_n for larger values of n and in general.

Bibliography

[1] D. Applegate, R. Bixby, V. Chvátal and W. Cook. Concorde-A code for solving Traveling Salesman Problems.15/12/99 Release.
Avilable at http://www.tsp.gatech.edu//concorde.html, website.

[2] D. Applegate, R. Bixby, V. Chvátal and W. Cook. Finding cuts in the TSP. DIMACS Technical Report, 95–05 1995

[3] S. Boyd and G. Labonté. Finding the exact integrality gap for small travelling salesman problems. *Integer Programming and Combinatorial Optimization 2002*, Springer-Verlag, Berlin Heidelberg, 83–92 (2002).

[4] S. Boyd and P. Elliott-Magwood. Finding the Exact Integrality Gap for Small ATSP's. Talk presented at the Combinatorial Optimization conference (CO2004), Lancaster, England, (2004).

[5] S. Boyd and P. Elliott-Magwood.
Private communications.

[6] M. Charikar, M. Goemans, and H. Karloff. On the Integrality Ratio for Asymmetric TSP. *Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, to appear, (2004).

[7] N. Christofides. worst–case analysis of a new heuristic for the traveling salesman problem. Report 388, Graduate School of Industrial Administration, Carnegie Mellon University, Pittsburgh (1976).

[8] C.S. Chekuri, A. V. Goldberg, D.R. Karger, M.S.Levine , and C.Stein. Experimental Study of Minimum Cut Algorithms.

[9] W.J. Cook, W.H. Cunningham, W.R. Pulleyblank and A. Schrijver. *Combinatorial Optimization.* John Wiley & Sons, Chichester, 37–85 (1998).

[10] G. Dantzig, D. Fulkerson and S. Johnson. Solution of large-scale travelling salesman problems. *Operations Research* 2, 393-410 (1954).

[11] E.A. Dinitz, A.V. Karzanow, M.V. Lomonosov. On the structure of a family of minimal weighted cuts in a graph. A.A. Fridman (Ed), studies in Discrete Optimization, Nauka, Moscow,pp-290–306 (1976).

[12] A. Frieze, G. Galbiati, and F. Maffioli. On the Worst-case Performance of Some Algorithms for the Asymmetric Traveling Salesman Problem. *Networks*, 12: 23 – 39, (1982)

[13] J. Hao and J.B. Orlin. A faster algorithm for finding the minimum cut in a directed graph. Journal of Algorithms,17, 424–464, (1994).

[14] D.S. Johnson and C.H. Papadimitriou. Computational Complexity. In The Traveling Salesman Problem. John Wiley & Sons, Chichester, 37-85 (1985).

[15] M. Jünger, G. Reinelt and G. Rinaldi, *Handbook on operations research and management sciences: Networks* (M. Ball, T. Magnanti, C.L. Monma and G. Nemhauser, eds.), North-Holland, 225–330 (1995).

[16] A. Lodi, M. Fortini and A.N. Letchford. On Traveling Salesman Compatible Tours. Talk presented at the Combinatorial Optimization conference (CO2004), Lancaster, England, (2004).

[17] M.A. Trick. A Tutorial on Integer Programming.
Available at http://mat.gsia.cmu.edu/orclass/integer/node10.html website.