To the Graduate Council:

I am submitting herewith a dissertation written by Vanditha Mukund entitled "A DECOMPOSITION-BASED APPROACH TO THE TRAVELING SALESMAN PROBLEM WITH TIME WINDOWS." I have examined the final copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Management Science.

Charles E. Noon, Major Professor

We have read this dissertation
and recommend its acceptance:

Accepted for the Council:

Vice Provost
and Dean of The Graduate School

A DECOMPOSITION-BASED APPROACH TO THE TRAVELING SALESMAN PROBLEM WITH TIME WINDOWS

A Dissertation

Presented for the

Doctor of Philosophy

Degree

The University of Tennessee

Vanditha Mukund

May 1995

Dedicated to

my parents

Smt. Padmavathi

and

Sri. T. V. Chidambariah

Acknowledgements

It is with great pleasure that I thank all the people who have made my dream come true. My heartfelt thanks are due to my advisor Professor Charles E. Noon. His help and support have been invaluable in helping me reach my goal. I sincerely thank Professors Kenneth C. Gilbert, Melissa R. Bowers, Chanaka Edirisinghe and Bruce A. Ralston for all their input and advice.

Sincere thanks are due to Dr. Paul Petersen, Dean of Engineering, Rochester Institute of Technology for allowing me to use the computing facilities at the Rochester Institute of Technology. Thanks are also due to my brother Venkatesh Chidambariah and his wife Anuradha for their help during my many visits to Knoxville.

A project of this magnitude would not have been possible without the complete support of my husband and children. The long hours spent on my dissertation have been at the expense of my family. I am especially grateful to my husband Mukund for his love and support, which have gone a long way in helping me achieve my goal. My daughters Kavitha and Anupama have been very patient and understanding and make it all worthwhile.

Abstract

The Traveling Salesman Problem with Time Windows is a variation of the Traveling Salesman Problem wherein the salesman is required to arrive at each customer node during a specified time interval. This dissertation presents a new solution methodology for the Traveling Salesman Problem with Time Windows. This solution methodology is based on Bender's Decomposition. The master problem can be represented as a Traveling Salesman Problem with additional constraints, while the subproblem is a pure linear program. The master problem provides a route while the subproblem checks for feasibility of the route with respect to time windows. Several different representations of the problem are considered. The nature of cuts generated in each of these representations is studied. A heuristic based on the solution methodology is proposed. The optimal solution methodology is tested on a set of small problems. The heuristic is tested on these small problems as well as on an expanded set of moderately large problems. A variation of the Traveling Salesman Problem with Time Windows that minimizes the total time spent on the tour rather than the cost of the tour is also studied. This problem is referred to as the Traveling Salesman Makespan Problem with Time Windows. An extension of the solution process to solve the Traveling Salesman Makespan Problem with Time Windows is also discussed.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In business environments, we are concerned with the movement of goods or commodities from one or more supply points (warehouses or distribution centers), to one or more demand points (customers or nodes), as efficiently and economically as possible. The movement or service could be performed by a vehicle, or a salesman, or a machine. The desired result is a route (a list of nodes to be visited) and a schedule (times when the nodes can be visited) that minimizes the total cost of the operation.

Effective and flexible planning tools for routing and scheduling activities become necessary in order to achieve these results [1]. Depending on the nature and structure of the problem, various factors come into play when developing an algorithm. The following sections provide a broad classification of routing and scheduling problems as well as an overview of the complexity of the problems.

1.1 Classification

All routing and scheduling problems have essentially the same outcome. For each vehicle, driver, or salesman, a route and a schedule are provided. The route specifies the sequence in which the customers/nodes are to be visited, while the schedule specifies the time at which

the visit can take place or the service can be provided. Routing and scheduling problems can be broadly classified into three categories as specified by Bodin, *et al.* [1]. They are Routing Problems, Scheduling Problems and Routing and Scheduling Problems. More details of classification follow:

Routing Problems: In pure routing problems, the route to be followed is based solely on the location of the customer/node. Temporal considerations are completely ignored. The objective is to minimize the cost of the route. An example of this kind of problem is the determination of a newspaper delivery route. Here, the time of newspaper delivery to the customer is not very important.

Scheduling Problems: Scheduling problems have associated with each entity needing service a specific time when service can begin and end. An entity needing service is said to have a definitive service time if the starting and ending times of the service are specified in advance [1]. The less restrictive form of definitive service times are time windows. (A time window is of the form $[e_i, l_i]$, where e_i is the earliest possible time service can begin at node i and l_i is the latest possible time service can begin at node i. If e_i and l_i coincide, then the result is a definitive time for service at node i).

Routing and Scheduling Problems: This class of problems refers to situations where both spatial and temporal considerations are important. Problems of this type are characterized by precedence relationships between customers as well as time restrictions with regard to service at customers. In such cases, both spatial and temporal considerations become important. An example of an a priori precedence relationship is the case of a Dial-a-Ride Problem [1] where a customer has to be picked up before he is dropped off. Therefore, the pickup activity must precede the delivery activity, thus establishing a precedence relationship between the activities of pickup and delivery of a customer. An example of temporal restrictions is the existence of time windows.

One of the most notable routing problems is the Traveling Salesman Problem (TSP). A salesman has to visit every node of a set exactly once and return to the starting point, while minimizing the total cost of the tour. When a restriction is placed on the time when a node can be visited, the result is the Traveling Salesman Problem with Time Windows (TSPTW). The TSPTW can be used to model situations in various areas, a few of which are

transportation, engineering, medical technology etc. This work involves the development of a new solution methodology based on Bender's Decomposition for the Traveling Salesman Problem with Time Windows. Alternate formulations of the problem are considered and the effect of various aspects of cut generation are studied. A heuristic based on this solution methodology is then developed.

A closely related problem is the Traveling Salesman Makespan Problem with Time Windows (TSMPTW). In this problem, the objective is to minimize the total time spent on the tour by the salesman. Therefore, the time spent waiting for a time window to open is also taken into account. An optimal method of solution for the TSMPTW has been suggested by Baker [2]. An extension of the Bender's Decomposition-based solution methodology for solving the Traveling Salesman Makespan Problem with Time Windows is also discussed.

1.2 Complexity and Solution Methodology

Most routing and scheduling problems can be represented as network problems and, as such, a measure of problem size can be determined by the number of nodes or number of arcs [1]. The computational burden involved in solving any problem is an important consideration. It is reasonable to assume that the computational burden increases with the size of the problem. However, the rate at which computational time increases with respect to the size of the problem is also an important consideration. If computational time increases exponentially for every small change in problem size, the computational burden involved in solving even moderately sized problems becomes large. Therefore, **complexity** is an issue with respect to these problems.

Classification as P or NP Hard: All combinatorial optimization problems can be classified into one of two groups.

1. Those that can be solved by polynomially bounded algorithms i.e., the class **P**.
2. Those for which there are no polynomially bounded algorithms available i.e., the class **NP Hard**.

Algorithms whose computational requirement increases polynomially with an increase in size of the problem, in the worst case, are called **polynomially-bounded** algorithms. The Shortest Path Problem, the Minimal Spanning Tree Problem, the Transportation Problem, the Max Flow Problem, and the Min Cost Flow Problem belong to this category. The other group of problems, i.e., the class NP Hard, contains combinatorial problems for which no polynomially-bounded algorithms have been found. The computational burden for NP Hard problems increases exponentially with the size of the problem, in the worst case, and as such greatly restricts the size of the problem that can be solved. The Capacitated Minimal Spanning Tree Problem, the Traveling Salesman Problem and its variants, and the Vehicle Routing Problem and its variants, belong to this category. Lawler, *et al.* [3] discuss combinatorial optimization problems in detail.

Solution Methodology: Most routing and scheduling problems fall into the NP Hard category. Because solution methods aiming to seek optimal solutions suffer from an exponential growth in computational burden with growth in problem size, approximate methods known as *heuristics* should be considered. A heuristic is a procedure that uses a combination of the mathematical structure of the problem and intuition to reach a good feasible solution. The heuristic solution is not necessarily the optimal solution. Generally, heuristics can solve much larger problems than would have been possible with exact algorithms. Heuristics are useful in solving NP Hard problems of moderate to large dimensions (size).

1.3 Overview of the Dissertation

This dissertation provides a new solution methodology based on Bender's Decomposition for the Traveling Salesman Problem with Time Windows. Initially, an optimal method based on Bender's Decomposition is developed. Alternate problem formulations are examined and the formulation with least number of master problems solved is considered for development of a heuristic. The methodology works well on problems with a wide range of time window widths and time window overlaps. The problem formulation also allows for general internode (arc) costs. A modification in the definition of the objective function and some of the temporal constraints also allows solution of the Traveling Salesman Makespan Problem with Time Windows.

1.3.1 Research Objectives

Most of the work on the Traveling Salesman Problem with Time Windows involves minimizing the total time spent on the tour (essentially the Traveling Salesman Makespan Problem with Time Windows). Hence, one objective of this work has been to develop a formulation that seeks to minimize the total weight of the tour. The internode weights of the problem can refer to cost of travel, distance traveled, etc.; the objective function seeks to minimize the total weight of the Traveling Salesman tour. Another objective has been to develop a solution methodology that works well on problems with a wide range of time window widths and overlaps. The methodology is tested on some real-life problems and some randomly-generated problems.

The Traveling Salesman Problem with Time Windows belongs to the class NP Hard. Therefore, development of a heuristic based on the solution methodology developed here is also studied. The nature of the heuristic is modular. One module solves an Asymmetric Traveling Salesman Problem (ATSP) over the nodes of the network. An Asymmetric TSP is a TSP where the cost of traveling from node i to node j is not the same as the cost of traveling from node j to node i. The modular nature of the heuristic allows for inclusion of better methods of solution for the ATSP as they become available. The heuristic is tested on a problem set consisting of real-life problems as well as randomly-generated problems. The problem sets cover a reasonably wide range of time window widths and overlap.

1.3.2 Organization

The remainder of this dissertation is organized as follows. Chapter 2 provides the relevant background for the work and presents a survey of the literature regarding previous work in the area of routing and scheduling problems. Chapter 2 also presents the Langevin, *et al.* two-commodity flow formulation for the TSPTW [4], a brief description of Baker's branch-and-bound solution procedure for the TSMPTW, and an overview of the basic Bender's Decomposition procedure. This is followed by a brief analysis of the previous work and scope of the research. Chapter 3 is concerned with the development of the proposed solution methodology. It presents several formulations of the TSPTW and the implementation of Bender's Decomposition on each formulation. Chapter 4 presents a comparison

of the different formulations of the TSPTW. In the context of the TSPTW as well as the TSMPTW, the formulation with the least number of master problems solved is then tested on a problem set with gradually widening time windows. The performance of the Bender's Decomposition-based method is compared with the performance of Baker's Method for the Traveling Salesman Makespan Problem with Time Windows. Chapter 5 presents a heuristic based on the solution methodology presented in the previous chapters for the Traveling Salesman Problem with Time Windows. Results of a test of the heuristic's performance are presented. Chapter 6 contains conclusions and defines the scope for further research. The appendices contain details of all the test problems and pseudocodes for some important routines described in this dissertation.

Chapter 2

Background

2.1 Objectives

The main objective of this chapter is to provide a concise survey of the literature pertaining to the Traveling Salesman Problem with Time Windows. It also provides a literature review of routing problems in general and discusses the scope of research.

The background study is initiated by describing the problem under consideration in general terms. The Traveling Salesman Problem is a relaxation of the Traveling Salesman Problem with Time Windows. Therefore, a review of the literature pertaining to the Traveling Salesman Problem is provided. This is followed by a discussion of the literature pertaining to the Traveling Salesman Problem with Time Windows. Langevin *et al.* [4] first addressed the problem of minimizing the total cost of the tour and their paper is discussed in detail. The Traveling Salesman Makespan Problem with Time Windows is closely related to the Traveling Salesman Problem with Time Windows; therefore, literature pertaining to the Traveling Salesman Makespan Problem with Time Windows has also been included. Baker [2] has presented a branch-and-bound scheme for the Traveling Salesman Makespan Problem with Time Windows, and this paper is discussed in detail. The TSPTW and the TSMPTW appear as subproblems in complex vehicle routing and scheduling problems. A brief review of the literature pertaining to some other routing problems is also provided.

Bender's Decomposition provides a basis for the development of the solution scheme described in this dissertation and it is discussed briefly. A summary of the existing solution schemes for the Traveling Salesman Problem with Time Windows and the Traveling Salesman Makespan Problem with Time Windows is provided followed by a discussion of the scope of research.

The organization of this chapter is as follows. Section 2.2 contains some general information about the Traveling Salesman Problem with Time Windows. Section 2.3 contains the literature review pertaining to the Traveling Salesman Problem. The literature review pertaining to the Traveling Salesman Problem with Time Windows is provided in Section 2.4. A description of other routing problems and corresponding literature review is provided in Section 2.5. The Langevin, *et al.* method and Baker's Method are discussed in Section 2.6. The nature of the time windows influences the performance of solution method and this aspect is studied in Section 2.7. A summary of the Bender's Decomposition procedure is provided in Section 2.8. Finally, Section 2.9 contains a summary of existing solution schemes and a discussion of the scope of research.

2.2 General Information

Routing and scheduling problems with time windows have been the focus of increased attention over the last decade. Time windows are a common feature in most real life applications of routing and scheduling problems, and are usually classified as either Hard Time Windows or Soft Time Windows. In the case of Hard Time Windows, if a node is reached early (ie., before the time window has opened), then a wait is incurred. Also, arrival at the node cannot take place after the time window at the node has closed. On the other hand, Soft Time Windows allow the time windows to be violated at a certain cost (penalty). Examples of Hard Time Windows include deliveries at banks, post offices, etc., while an example for Soft Time Windows is the Dial-a-Ride Problem. The Traveling Salesman Problem is a relaxation of the Traveling Salesman Problem with Time Windows. Therefore, the optimal solution to the Traveling Salesman Problem is a valid lower bound to the Traveling Salesman Problem with Time Windows.

2.3 The Traveling Salesman Problem

2.3.1 Definition and Classification

This problem is perhaps the most well known of all discrete optimization problems. The classic form of the Traveling Salesman Problem (TSP), as stated by Parker and Rardin [5] is:

> Given a graph of n nodes (cities) with edges possessing
> real-valued weights (internode distances or costs), determine
> the minimum total weight circuit/cycle in the graph that
> visits each node exactly once, except for the initial and final
> nodes which are the same.

If the internode costs are **symmetric**, i.e., the cost of travel between two nodes is the same irrespective of the direction of travel, then we have a Symmetric Traveling Salesman Problem (STSP). A feasible solution to such a problem consists of a circuit through the nodes of the graph, with two arcs/edges incident to each node. However, if the internode costs depend on the direction of travel between the nodes, then the arc costs are said to be **asymmetric** and the result is an Asymmetric Traveling Salesman Problem (ATSP). A feasible solution to such a problem consists of a circuit through the nodes of the graph, with one arc into and one arc out of every node.

The Traveling Salesman Problem has drawn substantial interest from mathematicians and scientists for well over a century. Several survey papers have been published that provide a summary of the most prevalent methods of solving the Traveling Salesman Problem. The surveys published by Bodin, *et al.* [1], Parker and Rardin [5], Golden, *et al.* [6], Laporte [7], and Melamed, *et al.* [8], [9], provide information about the methods available for solution of the Traveling Salesman Problem.

The Traveling Salesman Problem belongs to the class of problems considered to be **NP Hard** [10], [11]. The Traveling Salesman Problem under both the Euclidean metric and the Rectilinear (Manhattan) metric is **NP Complete** [12], [13]. As such, local search algorithms for the Traveling Salesman Problem, having polynomial complexity per iteration, will generate solutions arbitrarily far from optimal [14]. Solution procedures for the Traveling

Salesman Problem include Exact algorithms and Heuristic procedures. Exact algorithms seek to find the optimal tour over the nodes of the network. Heuristic procedures are non-exact procedures that generate solutions close to the optimal solution. Their advantage is that they are generally very fast and can solve larger problems.

2.3.2 Exact Algorithms

Most of the exact methods used for solving the Traveling Salesman Problem are of the branch-and-bound variety. At each node of the branch-and-bound tree, lower bounds are computed by solving relaxations to the original Traveling Salesman Problem. Different branching rules as well as different ways of computing lower bounds lead to a variety of solution schemes for the Traveling Salesman Problem.

One familiar way of defining the Traveling Salesman Problem is as follows:

$$\min \sum_{(i,j)\in A} c_{i,j} x_{i,j} \tag{2.1}$$

$$\text{s.t.} \sum_{\{j:(i,j)\in A\}} x_{i,j} = 1 \; \forall \; i \in N, \tag{2.2}$$

$$\sum_{\{i:(i,j)\in A\}} x_{i,j} = 1 \; \forall \; j \in N, \tag{2.3}$$

$$x_{i,j} \in \{0,1\} \; \forall \; (i,j) \in A, \tag{2.4}$$

$$X \in T. \tag{2.5}$$

The graph $G(N, A)$ defines the nodes and arcs of the network. X is the matrix of variables $x_{i,j}$ having component values 1 when (i, j) is part of the solution and 0 otherwise. T is the set excluding subtours. If the problem is symmetric, then $c_{i,j} = c_{j,i}$. Obviously, the constraint set (2.5) is the complicating factor in the above formulation. If this constraint set were to be dropped, the result would be an Assignment Problem (AP) which is easy to solve. However, the Assignment Problem allows solutions wherein the nodes of the graph are organized into disconnected subtours. The constraint set (2.5) precludes subtours and

can be expressed in a variety of ways. One way of describing constraint set (2.5) would be:

$$T = \{X : \sum_{i \in S} \sum_{j \in N-S} x_{i,j} \geq 1\}, \forall\, S \subset N, S \neq \phi. \tag{2.6}$$

These constraints are referred to as subtour elimination constraints. Different ways of defining the subtour elimination constraints constitute the difference among the various algorithms.

The Assignment Problem is a relaxation of the Traveling Salesman Problem, and can be easily solved. The class of algorithms called Subtour Elimination Procedures uses the Assignment Problem relaxation within a branch-and-bound framework for solving the Traveling Salesman Problem.

Little, *et al.* [15], have presented a branch-and-bound algorithm for the Traveling Salesman Problem. In this case, the set of all tours (feasible solutions) is broken up into increasingly small subsets by branching. A lower bound on the length of the tours in each subset is calculated. Finally, a subset containing a single tour with length less than or equal to some lower bound for every other tour is found. Ideas frequently used in solving Assignment Problems provide the basis for branching rules and calculation of lower bounds. Most of the branch-and-bound algorithms generally follow a depth-first strategy. However, an optimal algorithm for the Asymmetric Traveling Salesman Problem provided by Carpenato and Toth [16] is a breadth-first branch-and-bound algorithm. Once again the subproblems solved are Assignment Problems. Pekny and Miller [17] have proposed a parallel branch-and-bound algorithm for solving large Asymmetric Traveling Salesman Problems. The algorithm uses an Assignment Problem based lower bounding technique, subtour elimination branching rules, and a subtour patching algorithm as an upper bounding procedure. The algorithm is organized around a data flow framework for parallel branch-and-bound. The problems solved include problem sets with thousands of cities. Subtour Elimination Procedures are more suitable for Asymmetric Traveling Salesman Problems. It is not very suitable for Symmetric Traveling Salesman Problems, since the subtours generated by the Assignment Problem relaxation are generally two-city subtours. This leads to a great degree of branching and slow rate of increase in the size of the subtours.

A second class of exact algorithms for solving the Traveling Salesman Problem is based on Tree Formulation Approaches (Held and Karp [18]). The method can best be explained by

first defining a 1-tree. For any selected node q, a 1-tree consists of a spanning tree on the node set $|N| - \{q\}$ together with two distinct edges at node q. Thus, a 1-tree has a single cycle; this cycle contains node q and node q always has degree two. Clearly, a Traveling Salesman tour is a 1-tree. Minimum weight 1-trees can be computed very efficiently. Let us define

$$F \quad = \quad \{\text{1-trees of } G(N,E)\},$$

where $G(N,E)$ defines the nodes and edges of the network. Then, (2.5) can be rewritten as

$$X \quad \in \quad F. \tag{2.7}$$

If the assignment constraints (2.2) and (2.3) are dropped, the resulting relaxation provides lower bounds for the Traveling Salesman tour.

In this relaxation, a relatively small number of constraints are dropped ($2n$ as compared to $2^n - 2$ in the AP relaxation case; $n = |N|$). When the Traveling Salesman Problem is asymmetric in nature, the notion of 1-trees can be extended to 1-arborescences, i.e., directed 1-trees. However, computation of directed 1-trees is more difficult than computation of undirected 1-trees. Hence, 1-tree relaxations are more often used in the context of the symmetric version of the Traveling Salesman Problem. In the 1-tree methods, within a branch-and-bound framework, a Lagrangean Relaxation subproblem is solved in place of the linear programming relaxation at each node of the branch-and-bound tree [1], [5]. Fischetti and Toth [19] propose the shortest spanning 1-arborescence bound for the Asymmetric Traveling Salesman Problem and have used it within an *additive bounding procedure* that combines five different bounds. Details of the procedure are provided in [7] and [19]. The bounding procedure is embedded within a simple branch-and-bound procedure. Carpenato, Fischetti and Toth [20] have provided new lower bounds, based on the Held and Karp 1-shortest spanning tree, combined with additive bounding procedures for the Symmetric Traveling Salesman Problem. Malik and Fisher [21] have described a dual-ascent algorithm for setting the Lagrangean multipliers in the 1-tree relaxation of the Symmetric Traveling Salesman Problem.

The next class of exact algorithms are the Restricted Lagrangean approaches of Balas and Christofides [5]. Here, the subtour constraints are a set of linear inequalities that both

prohibit subtours as well as enforce integrality. These linear inequalities are relaxed in a Lagrangean fashion. However, the fact that the linear inequalities are very large in number, renders impractical solution schemes by subgradient optimization. Moreover, the exact form of all the inequalities is not even known. The central idea in a Restricted Lagrangean Approach is to limit the search for Lagrange multipliers to cases where the relaxed problem can be easily solved. As long as the lower bound can be improved within such a restriction, the improvement is effected. If a point is reached where no more improvement is possible, then subtour elimination is adopted. The current problem is partitioned into more constrained ones and restricted Lagrangean search attempted again.

In recent years, exact algorithms based on cutting plane methodology [5], [22], [23], [24], [25] have been used to solve Traveling Salesman Problems. For convenience, this method is explained with respect to the symmetric version of the Traveling Salesman Problem on a complete graph $G(N, A)$. The constraints are of the form

$$\sum_{j<i} x_{j,i} + \sum_{j>i} x_{i,j} = 2, i = 1, 2, \ldots, n, \tag{2.8}$$

$$x_{i,j} = 0 \text{ or } 1, i = 1, 2, \ldots, pn;\ j > i, \tag{2.9}$$

$$X \in T, \tag{2.10}$$

where T denotes a set excluding subtours and $n = |N|$. The best possible system of cutting planes would be those that together define the convex hull of solutions to the above set of constraints. Suppose we define the convex hull of solutions to the system (2.8)-(2.10) by H_n. Then, a valid inequality for this system is one satisfied by all $X \in H_n$. A valid inequality is minimal if there is at least one $X \in H_n$ satisfying the inequality as an equality. A valid inequality is proper if there is at least one $X \in H_n$ yielding strict inequality. All facet-defining cuts are minimal and proper. The form of all minimal and proper inequalities is not known for the Traveling Salesman Problem. However, some specific ones are known, one of which is the set of subtour elimination constraints

$$\sum_{i \in S} \sum_{j \in \bar{S}} x_{i,j} \leq |S| - 1.$$

Another class of valid inequalities are derived from the blossom inequalities of Edmonds [5]. Chavatal proposed and Grotschel and Padberg tightened a new class of inequalities that include both the subtour elimination and blossom inequalities. These inequalities are

called **comb** inequalities. Crowder and Padberg [22] have solved large symmetric TSPs with a cutting plane algorithm that generates subtour and comb inequalities as needed. Padberg and Rinaldi [23] describe several procedures for the identification of facet inducing inequalities for the Symmetric Traveling Salesman Polytope. They describe an algorithm for solving the Symmetric Traveling Salesman Problem to optimality [24]. Large scale problems with more than two thousand nodes have been solved to optimality by using this method. Bounds from various versions of the Traveling Salesman Problem are discussed by Christofides [26], [27]. Christofides, Mingozzi and Toth [28] describe a state-space relaxation procedure for computation of bounds for routing problems. This bounding procedure can be used within a branch-and-bound framework for solving the Traveling Salesman Problem. Finally, Feiring [29] has proposed a Bender's Decomposition-based model for the Traveling Salesman Problem. The problem variables have been defined in a manner that facilitates solution of the master problem by dynamic programming. Details regarding performance have not been provided by the author.

2.3.3 Heuristics

Combinatorial problems that belong to the class NP Hard are prime candidates for solution by heuristics. Many heuristic procedures have been developed for solving the Traveling Salesman Problem. Some of the more common heuristic procedures are explained briefly. For details, the reader is referred to [1], [5], [6], [27], [30], [31], [32], [33], [34], [35], [36]. Golden *et al.*, [6] have provided a composite study of many heuristic algorithms encountered in the literature. They classify heuristics into three main categories: tour construction, tour improvement, and composite procedures. Tour construction procedures include the nearest neighbor heuristic, the Clark and Wright savings heuristic, insertion procedures, the convex hull heuristic, and Christofides' heuristic. The insertion procedures for the TSP provide a basis for the insertion heuristic described in Chapter 5 and therefore they are discussed briefly.

The general concept of the insertion procedure is to take a subtour of k nodes at iteration k and attempt to determine which node not in the subtour should join the subtour next (selection step). The next step is to determine where in the subtour the node is to be inserted (insertion step). A few of the insertion procedures have been described.

Nearest Insertion:

- Step 1: Start with a subgraph consisting of node i only.
- Step 2: Find node k such that $c_{i,k}$ is minimal and form subtour $i - k - i$.
- Step 3: This is the selection step. Given a subtour, find node k not in the subtour closest to any node on the subtour.
- Step 4: This is the insertion step. Find the arc (i, j) in the subtour that minimizes $c_{i,k} + c_{k,j} - c_{i,j}$; insert k between i and j.
- Go to Step 3 unless we have a Hamiltonian cycle (a tour through the nodes of the network, visiting each node exactly once and then returning to the starting node).

Cheapest Insertion:

The procedure is the same as nearest insertion except that steps 3 and 4 are replaced by

- Step 3a: Find (i, j) in subtour and k not in the subtour, such that $c_{i,k} + c_{k,j} - c_{i,j}$ is minimal and, then insert k between i and j.

Arbitrary Insertion:

The procedure is the same as that for cheapest insertion except that in Step 3a, arbitrarily select node k not in the subtour to be inserted into the subtour.

The branch exchange heuristics are perhaps the best known heuristics for the Traveling Salesman Problem. They are tour improvement procedures. The $2 - opt$ and the $3 - opt$ procedures were first introduced by Shen Lin [31]. Later, Lin and Kernighan [35] presented the $k - opt$ procedure for $k \geq 3$. The basic principle of operation of the branch exchange heuristics is as follows:

- Step 1: Select an initial tour. Generally, this is a random tour;
- Step 2: Improve the tour using one of the branch exchange heuristics;
- Step 3: Continue Step 2 until no further improvement is possible.

The branch exchange heuristics result in a local optimum ie., the tour is at least as good as its neighboring tours. A $k - change$ of a tour involves replacing k branches of the tour by k other branches to form an improved tour. The tour is considered to be $k - optimal$ if no further improvement is possible through a $k - change$.

The basic composite procedure can be stated as follows:

- Step 1: Use one of the tour construction procedures to generate an initial tour;
- Step 2: Apply a 2-opt procedure to the tour resulting from Step 1;
- Step 3: Apply a 3-opt procedure to the tour resulting from Step 2.

Composite procedures are relatively fast and produce good results.

In related work, recently, Noon *et al.* [37], have proposed a multiplier adjustment method for obtaining a quick lower bound for the LP relaxation of the minimum cost perfect 2-matching problem. The minimum cost perfect 2-matching problem is defined on a weighted undirected graph. It involves finding a minimum weight subset of edges of the graph such that each vertex in the graph is incident to exactly two edges of the minimum weight subset. This is useful in generating lower bounds for the symmetric TSP and the approach is shown to compare favorably with other methods for generating quick lower bounds for the symmetric Traveling Salesman Problem.

2.4 The Traveling Salesman Problem with Time Windows

2.4.1 Definition and Classification

Parker and Rardin's [5] definition of the Traveling Salesman Problem can be extended to define the Traveling Salesman Problem with Time Windows as follows:

> Given a graph of n nodes (cities) with arcs possessing real-valued weights (internode distances or costs) and nodes possessing

real-valued time windows during which they can be visited,
determine the minimum total weight circuit/cycle in the graph
that visits each node during its time window exactly once,
except for the initial and final nodes, which are the same.

The Traveling Salesman Problem with Time Windows belongs to the class NP Hard (since a relaxation of the TSPTW, the Traveling Salesman Problem, belongs to the class NP Hard). Furthermore, Savelsbergh [38] has shown that even finding a feasible tour to the Traveling Salesman Problem with Time Windows is an NP Complete problem.

If the objective is to minimize the completion time, i.e., the total time on the tour, then the problem corresponds to the Traveling Salesman Makespan Problem with Time Windows (TSMPTW).

2.4.2 Previous Work

Research has been quite limited on the Traveling Salesman Problem with Time Windows. Savelsbergh [38] proposed heuristic algorithms based on the k-interchange rule. For the Traveling Salesman Problem with Time Windows, an exact algorithm for minimization of the traveled distance was proposed by Langevin, *et al.* [4] in 1993. These authors propose a two-commodity flow formulation for the TSPTW. The exact procedure adopted is explained in Section 2.6. A few exact algorithms have been developed for the Traveling Salesman Makespan Problem with Time Windows. Baker [2] and Christofides, *et al.* [39], have proposed exact algorithms to minimize the total time on the tour for the TSMPTW. Both these methods are branch-and-bound procedures. Baker's method for the Traveling Salesman Makespan Problem with Time Windows is explained in detail in Section 2.6. Christofides, *et al.* suggest state-space relaxation techniques that can be embedded in a general branch-and-bound scheme for solution of the time-constrained Traveling Salesman Problem (essentially the TSMPTW). In this problem, there are multiple time windows associated with each node and the salesman must visit the node during one of the time windows. The state-space relaxation procedure involves relaxation (reduction in number of states) of the state-space associated with a given dynamic programming (DP) recursion. The relaxation is performed in such a way that the solution to the relaxed recursion provides a lower bound on the value of the true optimum. This state-space relaxation method is analogous to

Lagrangean Relaxation in integer programming. Constraints in integer programming formulations correspond to state variables in dynamic programming recursions; hence, constraint relaxation corresponds to state-space relaxation. The state-space relaxation corresponds to the definition of $g(.)$ as a mapping from the state space S to a space G with smaller cardinality. The recursion is then defined over the new domain. This state-space relaxation of the recursion leads to the development of lower bounds that can be used within a branch-and-bound framework.

The polyhedral aspects of scheduling problems with an application to the time-constrained Traveling Salesman Problem as defined by Baker have been discussed by Joeseph Tama [40]. He defines the term *clique polyhedra* which informally is the convex hull of schedules for the one-machine scheduling problem (corresponding to the TSMPTW). Three classes of facets are derived for the one-machine scheduling problem and this procedure is used in a scheme to solve the Traveling Salesman Makespan Problem with Time Windows. The algorithm can be summarized as follows:

An initial feasible solution is easily found. This is subjected to a *jumping* heuristic, which tries to *jump* nodes to an earlier position in the given sequence of nodes *if* the *jump* will lead to an improved tour. Each time, a check is made to see if:

the upper bound of some time windows can be lowered,
the lower bound of some time windows can be raised,
and if order of precedence between nodes can be determined.

The procedure continues until no more tour improvement is possible. This could lead to a partitioning of the nodes into two sets S and T where every node in S precedes every node in T. Then it is possible to eliminate the nodes in S from the problem. The lower bound for the problem is provided by an initial linear programming relaxation containing the facet inequalities. The linear programming relaxation is obtained by replacing the disjunctive constraints of the Baker formulation of Section 2.6.2 with lower and upper bounds and facet inequalities. The sequence provided by the solution to this linear program is tested for feasibility. If infeasible, the sequence is subjected to a system of exchanges in order to identify a nearby feasible solution. A feasible sequence now is one that is at least as good as the current best tour. The tour is subjected to the jumping heuristic and iteratively improved until no further improvement is possible. If there exists a gap between the lower

bound provided by the linear program and a current best upper bound, then branching is initiated. The algorithm branches in a manner to promote the formation of S, T partitions. This facilitates reduction in problem size. The branching is done on the time window of a particular city so as to facilitate determination of precedence. Thus the chances of time window tightening and determination of precedence among pairs of nodes are increased. A prototype of this algorithm was implemented on a personal computer by the author. The performance was compared to the performance of Baker's method on similar problems. It was found that the number of search nodes (in the branch-and-bound tree) needed by this method were fewer than that needed by Baker's method. However each search node in this method is subjected to more work than a search node in Baker's method. The reason is that this method does both arc elimination and node elimination, whereas Baker's method does only arc elimination. Also, this processing is done at every search node in this method, whereas the processing is only done at the outset in Baker's method.

A survey of the advances made for a class of routing problems with time windows has been presented by Solomon and Desrosiers [41]. Langevin, *et al.*[4] state that problems with constraints that permit reduction of state-space have been solved by dynamic programming-based approaches. Tsitsiklis [42] has studied a variant of the Traveling Salesman Problem with Time Windows. The problem is defined over a graph in which each arc has a given length. There exist a set of jobs, each job i located at some node of the graph with an associated processing time h_i. The execution of the job has to start within a specified time window $[e_i, l_i]$. It is assumed that there is a single server that can move on the arcs of the graph at unit speed and has to execute all the jobs within their respective time windows. The two objectives considered are minimization of time by which all jobs are executed, and minimization of the sum of waiting times. Various beginning times and ending times of time windows have been considered and the complexity of the problems has been settled by either classifying the problems as NP Complete or by providing polynomial (or pseudopolynomial) time algorithms for their solution.

2.5 Other Routing Problems

2.5.1 Characterization

The Traveling Salesman Problem with Time Windows can be characterized as a special case of certain other routing problems with time windows. Examples of such routing problems are the Vehicle Routing Problem with Time Windows (VRPTW), the Pickup and Delivery Problem with Time Windows (PDPTW), the M-Traveling Salesman Problem with Time Windows (MTSPTW). A study of literature pertaining to these problems and other related problems is provided in this section.

2.5.2 Previous Work on VRPTW

The Vehicle Routing Problem with Time Windows has been the focus of much research in the last few years [41]. The Vehicle Routing Problem (VRP) is described by Solomon and Desrosiers [41] as follows:

> The VRP involves the design of a set of minimum cost routes, originating and terminating at a central depot, for a fleet of vehicles which services a set of customers with known demands. Each customer is serviced exactly once and furthermore, all the customers must be assigned to vehicles such that the vehicle capacities are not exceeded.

The VRPTW is a restriction of the VRP, in that the time window constraints are added. Solomon [38], [43] analyzes several of the earlier heuristic algorithms developed for the VRPTW. These heuristics include the savings heuristics, time oriented nearest neighbor heuristics, insertion heuristics, time oriented sweep heuristics, etc. Kolen, *et al.* [44] describe a branch-and-bound scheme for the time-constrained Vehicle Routing Problem (VRP). It is the first exact algorithm available for this problem and was based partially on the Christofides, Mingozzi and Toth method [39], [28], for the Vehicle Routing Problem. They extended the shortest q-path relaxation proposed by Christofides *et al.* [39]. Desrochers, Desrosiers, and Solomon [45] have described a new exact algorithm for the VRPTW. The VRPTW is formulated as a set-partitioning problem. The algorithm is based on a column generation approach for solving the LP relaxation of the set-partitioning for-

mulation of the VRPTW. They state that the method was more efficient for problems with tightly-constrained time windows and a high density (concentration) of time windows. The M-Traveling Salesman Problem with Time Windows is a special case of the VRPTW and Desrosiers, Soumis, and Desrochers [46] have solved a generalization of this problem using a column generation approach. Other work on the Vehicle Routing problem with Time Windows and its variants have been discussed by Van Landeghem [47], Ferland and Fortin [48], Ahn and Shin [49], and Atkinson [50]. Potvin and Rousseau [51] present a parallel route-building algorithm for the vehicle routing and scheduling problem with time windows. More recently, Derigs and Grabenbauer [52], have presented a new heuristic for solving the VRPTW. The heuristic is a two-phase approach. A nearest neighbor approach is adopted in the first phase to construct an initial set of tours. An improvement procedure, based on a node exchange (rather than an arc exchange) concept is used to improve the initial solution. A polynomial-time heuristic for a special version of the VRPTW is presented by Bramel, Li and Simchi-Levi [53]. The model presented by these authors is a simplified version of the general VRPTW since they allow only time windows of a particular type. They also assume a Euclidean distance matrix with travel being possible along both directions of the arc. This heuristic produces a solution that is asymptotically optimal (meaning that the relative error between the heuristic solution and the optimal solution decreases to zero as the number of nodes increases to infinity). Thangiah, Osman, Vinayagamoorthy and Sun [54], compare three heuristics for the VRP with one-sided time windows (ie., time windows where only the deadlines or end of the time window are specified). The three heuristics studied are the sweep approach, an insertion based approach and a third heuristic that combines genetic algorithms with cheapest insertion. They use the results to highlight specific problem characteristics that lead to high quality performance of the three heuristics.

2.5.3 Previous Work on PDPTW

The Pickup and Delivery Problem with Time Windows is a generalization of the VRPTW. It is concerned with the construction of optimal routes to satisfy transportation requests, each requiring both pickup and delivery under precedence capacity and time window constraints. The VRPTW is the particular case of the PDPTW where the destinations are all the common depot [41]. Dumas, Desrosiers, and Soumis [55] have solved the problem

optimally using a column generation scheme with a constrained shortest path problem as the subproblem. Sexton [56] has presented a Bender's Decomposition-based method for the single vehicle Dial-a-Ride Problem. The problem is separated into two components – the routing master problem and the scheduling subproblem. The integer program (master problem) is nonlinear and carries along the precedence constraints and capacity constraints in terms of integer variables. The objective function minimizes the total delivery time deviation and the total excess ride time. An initial feasible route is established with a route being defined as the order of picking up and dropping off customers. Given the route, an optimal schedule is formed with a schedule being defined as the exact times of pickup and dropoff of customers. The optimal schedule is used to change the objective function coefficients of the routing problem leading to an improved route. A new optimal schedule is produced based on this route. The algorithm terminates when no better route can be found in terms of the objective function in the routing step of the procedure. Dynamic programming methods for the Dial-a-Ride Problem have been provided by Psaraftis [57], [58] and Desrosiers, *et al.*, [59] present an optimal solution to the single vehicle Dial-a-Ride Problem with time windows, using a forward dynamic programming approach that significantly reduces the number of states being generated. The solution times are stated to increase linearly with increased problem size. Van Der Bruggen, *et al.*, [60] have developed a local search method for the PDPTW based on a variable-depth search, similar to the Lin-Kernighan algorithm for the Traveling Salesman Problem. This is a two-phase method where in the first phase, a feasible route is constructed, and in the second phase, the route is improved. Solomon and Desrosiers [41] have described other heuristics for the Pickup and Delivery Problem with Time Windows.

2.5.4 Other Related Problems

Some other problems with time windows include the Shortest Path Problem with Time Windows [41], [61], [62], the Time-Dependant Traveling Salesman Problem [63], [64], the Orienteering Problem with Time Windows [65], and the Vehicle Routing problem with Soft Time Windows [66], [67]. Desrosiers, *et al.*, [68] have presented Lagrangean Relaxation methods for solving Minimum Fleet Size Multiple Traveling Salesman Problems with Time Windows in the context of school bus scheduling problems with fairly tight time windows.

It is evident that though there has been a focus in recent years on routing problems with time windows, they have not been as well researched as some other routing problems (a classic example being the Traveling Salesman Problem).

2.6 Details of Some Solution Schemes

2.6.1 Details of the Two-Commodity Flow Formulation

The two-commodity flow formulation for the Traveling salesman problem with Time Windows is defined as follows:

Let $G = (N, A)$ be a directed graph where N is a set of n nodes and A is a set of feasible arcs ie., arcs along which travel is possible. Let $c_{i,j}$ be the cost associated with arc (i, j). Let node 1 represent the depot. Without loss of generality let the interval $[0, T]$ be the time window at the depot node and let $[a_i, b_i], i = 2, \ldots, n$ represent the time window corresponding to node i. If the salesman arrives early at a node, then he has to wait for the time window to open. Also, the due dates cannot be violated, i.e., the node has to be visited before the end of the time window. Let us define $w_i, i = 2, \ldots, n$ as the waiting time at node i and $c_i, i = 2, \ldots, n$ as the cost incurred per unit of waiting time. Let the fixed parameter T be the total amount of resource available at the beginning of any tour. Let node $n+1$ define the arrival depot or the ending node. Let $t_{i,j}$, $i, j = 1, 2, \ldots, n+1$ be the amount of resource needed to travel on arc (i, j). Non-negative flow $Y = (y_{i,j})$ represents the remaining quantity of resource while non-negative flow $Z = (z_{i,j})$ is the accumulated value of resource used at each node. For all arcs (i, j), let $(y_{i,j} + z_{i,j}) \in \{0, T\}$, i.e., let $(y_{i,j} + z_{i,j})/T$ be binary variables.

Three sets of constraints are presented along with their meaning and reduction:

Constraints on flow Y at each node:

$$\sum_{j=2}^{n} y_{1,j} = T,$$

$$\sum_{j=2}^{n+1} y_{i,j} = \sum_{j=1}^{n} y_{j,i} - \sum_{j=1}^{n} t_{j,i}(y_{j,i} + z_{j,i})/T,$$

$$i = 2, \ldots, n,$$

$$\sum_{j=2}^{n} y_{j,n+1} - \sum_{j=2}^{n} t_{j,n+1}(y_{j,n+1} + z_{j,n+1})/T \geq 0.$$

The left hand side of each relation gives the remaining quantity of resource at each node, for $i = 1, 2, \ldots, n, n+1$. At the depot node, the total amount of resource available has value T. For node $i, i = 2, \ldots, n$, the resource available has a value given by the sum of flows on arcs entering node i minus the amount of resource needed to travel on the same arcs. The last relation represents the unused (non-negative) amount of resource at the arrival depot node.

Constraints on flow Z at each node:

$$\sum_{j=2}^{n} z_{1,j} = 0,$$

$$\sum_{j=2}^{n+1} z_{i,j} = \sum_{j=1}^{n} z_{j,i} + \sum_{j=1}^{n} t_{j,i}(y_{j,i} + z_{j,i})/T,$$

$$i = 2, \ldots, n,$$

$$\sum_{j=2}^{n} z_{j,n+1} + \sum_{j=2}^{n} t_{j,n+1}(y_{j,n+1} + z_{j,n+1})/T \leq T.$$

The left hand side of each relation gives the accumulated amount of resource at each node, for $i = 1, 2, \ldots, n, n+1$. At the depot node, flow Z starts with a value of zero. At node $i, i = 2, \ldots, n$, the total amount of flow on arcs entering node i is increased by the amount of resource needed to travel on those arcs. The total quantity of resource accumulated at the arrival depot node $n+1$ must not exceed the amount T available at the start.

Complementary flow constraints:

$$\sum_{j=2}^{n} (y_{1,j} + z_{1,j}) = T,$$

$$\sum_{j=2}^{n+1} (y_{i,j} + z_{i,j}) = T, \; i = 2, \ldots, n,$$

$$\sum_{j=2}^{n}(y_{j,n+1} + z_{j,n+1}) \quad = \quad T.$$

At each node, the sum of Y and Z flows is always equal to the total amount of resource at the start.

Given that $z_{1,j} = 0,\ j = 2, \ldots, n$, the equation at node 1 can be removed as it already appears in the complementary flow constraints. The relation at node $n+1$ can also be discarded as it can be obtained using the other two relations at the same node. Finally, for node $i, i = 2, \ldots, n$, adding both sides of equations on flows Y and Z, and using the corresponding complementary flow constraints, results in:

$$\sum_{i=1}^{n}(y_{i,j} + z_{i,j}) \quad = \quad T,\ j = 2, \ldots, n.$$

Finally, arcs $(i, n+1), i = 2, \ldots, n$ entering the arrival depot node are renamed to be arcs $(i, 1), i = 2, \ldots, n$. Also, corresponding flow variables Y and Z, resource, and cost values are redefined. The formulation for the Traveling Salesman Problem with Time Windows is given by

$$\min \sum_{i=1}^{n}\sum_{j=1}^{n} c_{i,j}(y_{i,j} + z_{i,j})/T + \sum_{i=2}^{n} c_i w_i \tag{2.11}$$

$$\text{s.t.} \sum_{i=2}^{n} z_{i,1} + \sum_{i=2}^{n} t_{i,1}(y_{i,1} + z_{i,1})/T \quad \leq \quad T, \tag{2.12}$$

$$\sum_{j=1}^{n}(y_{i,j} + z_{i,j}) \quad = \quad T,\ i \in N \& i \neq 1, \tag{2.13}$$

$$\sum_{i=1}^{n}(y_{i,j} + z_{i,j}) \quad = \quad T,\ j \in N \& j \neq 1, \tag{2.14}$$

$$y_{i,j} + z_{i,j} \quad \in \quad \{0, T\},\ (i,j) \in A, \tag{2.15}$$

$$\sum_{j=1}^{n} z_{i,j} - \sum_{j=1}^{n} z_{j,i} - \sum_{j=1}^{n} t_{j,i}(y_{j,i} + z_{j,i})/T - w_i \quad = \quad 0,$$

$$i \in N \& i \neq 1, \tag{2.16}$$

$$a_i \leq \sum_{j=1}^{n} z_{i,j} \leq b_i, \qquad i \in N \& i \neq 1, \tag{2.17}$$

$$w_i \quad \geq \quad 0,\ i \in N \& i \neq 1, \tag{2.18}$$

$$y_{i,j} \geq 0,\ z_{i,j} \geq 0 \qquad (i,j) \in A. \tag{2.19}$$

The objective function takes into account the waiting costs. If the waiting costs are not important then $c_i = 0, i = 2, \ldots, n$ and variables w_i can be considered as surplus variables. The constraint set (2.16) can be written as inequalities. If the cost on each arc $c_{i,j}$ is identical to the time of travel $t_{i,j}$ and if $c_i = 1, i = 2, \ldots, n$, then the objective function minimizes the total time on the tour. This problem corresponds to the Traveling Salesman Makespan Problem with Time Windows.

The above formulation for the Traveling Salesman Problem with Time Windows is valid only if the fixed parameter T is large enough so as to not eliminate the optimal tour. For an existing solution to the problem the value of T is set $\geq \max_{(i,1)\in A}\{b_i + t_{i,1}\}$. If the integrality constraints (2.15) are relaxed, the TSPTW's linear relaxation formulation contains $\mathcal{O}(n^2)$ variables and $4n - 1$ constraints. For the TSPTW, the lower bound improves with a smaller value of T. For the TSMPTW, the smallest value of T feasible for the linear relaxation, is a lower bound on the optimal total time value.

Some preprocessing in terms of time window reduction and arc elimination is employed in order to tighten the LP relaxation and improve the efficiency of the branch-and-bound solution scheme. These procedures are described below.

The following conditions for time window reduction are applied until no further improvement can be achieved[45]. The convention adopted is that A^+ is the set of feasible arcs in the augmented network (where the depot node is duplicated with time windows of the form $[a_1, b_1] = [0, 0]$ and $[a_{n+1}, b_{n+1}] = [0, T]$).

Earliest arrival time at node k from predecessors, for $k \in \{2, \ldots, n+1\}$ is as follows:

$$a_k = \max\{a_k, \min\{b_k, \min_{(i,k)\in A^+}\{a_i + t_{i,k}\}\}\}.$$

Earliest departure time from node k to successors, for $k \in \{1, \ldots, n\}$ is as follows:

$$a_k = \max\{a_k, \min\{b_k, \min_{(k,j)\in A^+}\{a_j - t_{k,j}\}\}\}.$$

Latest arrival time to node k from predecessors, for $k \in \{2, \ldots, n+1\}$ is as follows:

$$b_k = \min\{b_k, \max\{a_k, \max_{(i,k)\in A^+}\{b_i + t_{i,k}\}\}\}.$$

Latest departure time from node k to successors, for $k \in \{1, \ldots, n\}$ is as follows:

$$b_k = \min\{b_k, \max\{a_k, \max_{(k,j)\in A^+}\{b_j - t_{k,j}\}\}\}.$$

In any problem with time window constraints, infeasible arcs can be identified by using the condition that an arc (i,j) can exist only if $a_i + t_{i,j} \leq b_j$. However, a much stricter condition can be derived for the single-vehicle time window constrained problem (with no service times at the nodes). An arc (i,j) can appear in the solution only if all other nodes can be visited before i or after j. This condition is now defined more formally. Let $S_{k,l}$ be the shortest path from node k to node l. To determine if an arc (i,j) is feasible, one must check if, for all node $l \in N$, it is possible to either visit nodes l before node i, i.e., $a_l + S_{l,i} \leq b_i$, or to visit node l after node j, i.e., $a_j + S_{j,l} \leq b_l$. If both conditions fail for one node, then arc (i,j) is not feasible, i.e., arc (i,j) cannot exist. If the depot is one of the nodes i or j, then only one of the two conditions is checked as it is impossible to visit a node before the starting node or after the arrival node.

During preprocessing, the time window reduction and arc elimination procedures are applied repeatedly until no further improvement is possible. Then, the parameter value T is calculated using the reduced time windows.

The subtour elimination constraints are expressed in terms of the flow variables as follows:

$$\sum_{i,j\in W}(y_{i,j} + z_{i,j}) \leq T(|W| - 1)\ \forall W \subset N, |W| \geq 2.$$

The branch-and-bound scheme is then adopted with lower bounds obtained by solving the LP relaxation of the two-commodity flow formulation and by adding any violated subtour elimination constraints. At a given node of the branch-and-bound tree, an arc $(i,j) \in A$ is chosen for branching. The arc is either fixed at zero or forced to be included in the solution. For symmetric Euclidean problems, this is done by setting $(y_{i,j} + z_{i,j} + y_{j,i} + z_{j,i}) = 0$ or, otherwise $(y_{i,j} + z_{i,j} + y_{j,i} + z_{j,i}) = T$. The arc chosen for branching is the one with fractional value closest to 1/2. The branch-and-bound node chosen for exploration is always the unexplored node having the best lower bound. Throughout the search, at a given branch-and-bound node, subtour elimination constraints previously introduced are not removed from the current LP program unless they become redundant since they define valid inequalities for the two-commodity formulation.

2.6.2 Details of Baker's Method

In this section, some details regarding the time-oriented formulation of the Traveling Salesman Problem, i.e., the Traveling Salesman Makespan Problem with Time Windows (TSMPTW) as defined by Baker [2] are presented. This solution method will be referred to as Baker's Method. The model provided by Baker for this problem is:

$$\min t_{n+1} - t_1 \tag{2.20}$$

$$\text{s.t. } t_i - t_1 \geq d_{1,i} \text{ , } i = 2,3,\ldots,n, \tag{2.21}$$

$$|t_i - t_j| \geq d_{i,j} \text{ , } i = 3,4,\ldots,n;\ 2 \leq j < i, \tag{2.22}$$

$$t_{n+1} - t_i \geq d_{i,1} \text{ , } i = 2,3,\ldots,n, \tag{2.23}$$

$$t_i \geq 0 \text{ , } i = 1,2,\ldots,n+1, \tag{2.24}$$

$$e_i \leq t_i \leq l_i \text{ , } i = 2,3,\ldots,n, \tag{2.25}$$

where,

t_i	=	time that the salesman visits city i,
$\|X\|$	=	absolute value of X,
$d_{i,j}$	=	shortest time of travel from city i to city j,
e_i	=	earliest allowable time of arrival at city i,
l_i	=	latest allowable time of arrival at city i,
t_1	=	time of start from depot,
t_{n+1}	=	time of return to depot.

Baker adopts a branch-and-bound procedure for the solution of this problem. The procedure is initiated by relaxing the absolute value constraints (2.22) and the time window constraints (2.25) of the above model. The resulting relaxed problem (defined as **BP**) is:

$$\min t_{n+1} - t_1 \tag{2.26}$$

$$\text{s.t. } t_i - t_1 \geq d_{1,i} \text{ , } i = 2,3,\ldots,n, \tag{2.27}$$

$$t_{n+1} - t_i \geq d_{i,1} \text{ , } i = 2,3,\ldots,n, \tag{2.28}$$

$$t_i \geq 0 \ , i = 1, 2, \ldots, n+1. \tag{2.29}$$

It can be seen that the dual to this problem is a longest path problem in a directed network with (n+1) nodes. The dual to this problem (defined as **BD**) can be represented as:

$$\max \sum_{j=2}^{n} (d_{1,j} u_{1,j} + d_{j,1} u_{j,n+1}) \tag{2.30}$$

$$\text{s.t.} \sum_{j=2}^{n} -u_{1,j} \leq -1, \tag{2.31}$$

$$u_{1,j} - u_{j,n+1} \leq 0 \ , j = 2, \ldots, n, \tag{2.32}$$

$$\sum_{j=2}^{n} u_{j,n+1} \leq 1, \tag{2.33}$$

$$u_{1,j}, u_{j,n+1} \geq 0 \ , j = 2, \ldots, n, \tag{2.34}$$

where the $u_{i,j}$ are the dual variables of the constraints in **BP**. Therefore, the problem **BP** is actually solved by solving its dual **BD**.

As the cases of the absolute value constraints are added to the model, additional linear constraints are added to the primal problem **BP**, which create additional columns in the dual **BD**, which further translate into additional directed arcs in the longest path network. As arcs are added to the dual network, two basic assumptions must be met. The assumptions are:

- the distances in the original problem and hence the dual network must satisfy the triangle inequality ie., $d_{i,k} + d_{k,j} \geq d_{i,j}$, and
- a maximum cardinality path labelling convention must be adopted in order to resolve ties in the triangle inequality which may allow paths equal in length to the longest path to contain less than n arcs.

In the labelling algorithm used to solve the dual, when a node is permanently labelled, it represents the primal variable t_i for node i. It is easy to enforce the time window constraints in this solution procedure. If the labeling algorithm determines t_i such that $t_i > l_i$, then that particular subproblem will be infeasible. On the other hand, if the algorithm determines a t_i that is less than the corresponding e_i, then t_i is set equal to e_i and the algorithm continues.

Table 2.1: Data for Example

Node	1	2	3	4	5
1	∞	1	2	3	4
2	1	∞	1	4	5
3	2	1	∞	5	6
4	3	4	5	∞	6
5	4	5	6	6	∞

The branch-and-bound procedure is thus used to generate an optimal solution to the Traveling Salesman Makespan Problem with Time Windows.

2.7 Effect of Time Windows on Langevin and Baker Methods

This section provides small examples depicting the Langevin, *et al.* method for the TSPTW and the Baker method for the TSMPTW. The effect of varying widths of time windows on the performance of these methods with respect to the example is also studied.

Consider the following five node problem. The cost of travel from node i to node j, $c_{i,j}$ is assumed to be equal to $d_{i,j}$, the time of travel from node i to node j. For convenience all the quantities are assumed to be expressed in the same units. The service time at the nodes are assumed to be zero. The distance matrix for this example problem is given in Table 2.1. Two different sets of time windows are considered. The first set of time windows is specified in Table 2.2 and the second set of time windows is specified in Table 2.3.

Langevin Method – TSPTW

The Langevin method initiates the branch-and-bound procedure by solving the linear programming relaxation of the two-commodity flow formulation for the TSPTW. The value of the parameter T is set equal to 20 and the distance/cost matrix is preprocessed according to

Table 2.2: Example - Time Window Set I

Node	1	2	3	4	5
Early	0	4	5	8	11
Late	20	7	10	12	16

Table 2.3: Example - Time Window Set II

Node	1	2	3	4	5
Early	0	2	3	6	8
Late	20	9	11	14	16

the technique described previously. The first set of time windows are reasonably tight and restrictive. The solution to the linear programming relaxation is the tour 1,2,3,4,5,6 with a value of 17. Since, this is an integer solution with no subtours, this is indeed the optimal solution. We see that the LP lower bound is equal to the optimal solution. Next, let us consider the second set of time windows, which are wider than the previous set. The linear programming lower bound is found to be 15.6 ($(y_{1,2}+z_{1,2})/T = 13/20$, $(y_{3,2}+z_{3,2})/T = 7/20$, $(y_{3,4}+z_{3,4})/T = 13/20$, $(y_{5,1}+z_{5,1})/T = 13/20$, $(y_{5,4}+z_{5,4})/T = 7/20$, $(y_{2,3}+z_{2,3})/T = 1$, $(y_{4,5}+z_{4,5})/T = 1$) and the solution does not assign integer values to all the integer variables. The arc (3,2) is selected for branching. The branch that sets $(y_{3,2}+z_{3,2}+y_{2,3}+z_{2,3}) = 0$ results in LP infeasibility. Therefore, this node in the branch-and bound tree can be fathomed. The branch that sets $(y_{3,2}+z_{3,2}+y_{2,3}+z_{2,3}) = 20$ results in an LP solution that assigns integer values to all the variables. The solution also does not involve any subtours. Since there exist no more unexplored nodes, this solution is optimal. It can be seen that the LP lower bound is less than the optimal solution. Since the time windows were less restrictive, the LP lower bound was lower than the optimal solution and more nodes had to be explored before the optimal solution was found. Figure 2.1 depicts the steps of the procedure for the second set of time windows.

Baker's Method - TSMPTW

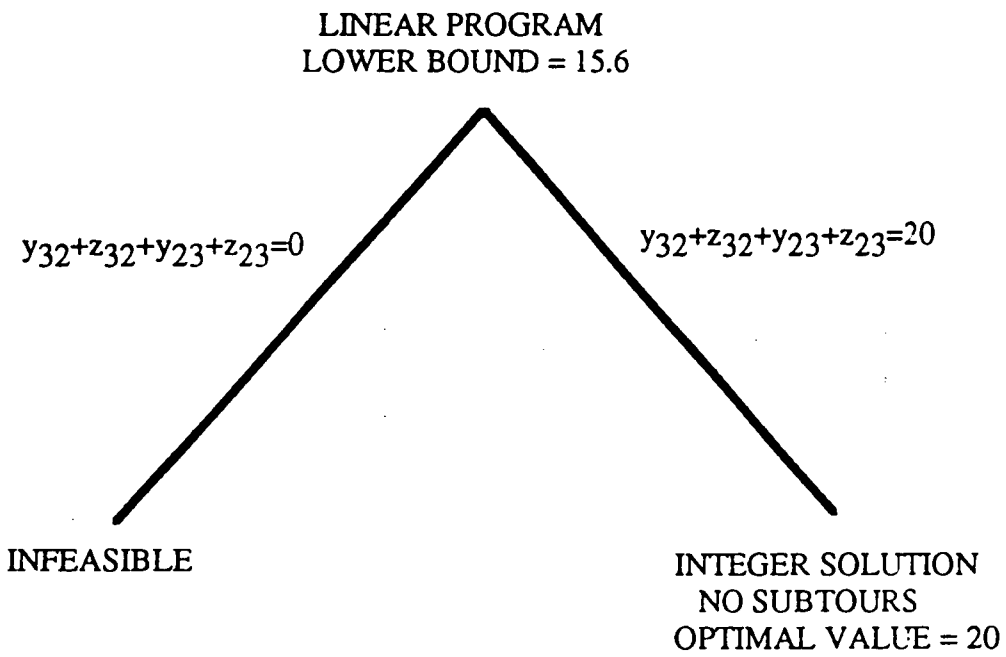

Figure 2.1: Langevin Method on Example with Time Window Set II

The example problem is now used to explain the steps of Baker's method for the Traveling Salesman Makespan Problem with Time Windows. For the first set of time windows, arcs (1,2), (1,3), (1,4), (1,5), (2,6), (3,6), (4,6), (5,6), (2,5), (3,5), (4,5), (2,4), and (3,4) have the directions of travel fixed. Arc (2,3) is the only reversible arc and has a corresponding set of absolute value constraints in the Baker formulation for the TSMPTW. Figure 2.2 displays the steps of the Baker procedure. The lower bound at the root node is 20. The solution to the problem at the root node is not a complete tour. Therefore, branching takes place along the two directions of the reversible arc (2,3). Branching along arc direction (2,3) results in a tour with value 20. This is the incumbent value. Branching along the arc direction (3,2) does not yield a tour. Since there are no more reversible arcs, the incumbent is the optimal solution. We notice that the lower bound at the root node is equal to the optimal solution. Baker's method is now used to solve the problem with the second set of time windows. Figure 2.3 displays this procedure. Arcs (1,2), (1,3), (1,4), (1,5), (2,6), (3,6), (4,6), (5,6), (2,4), (2,5), and (3,5) have directions of travel fixed. Arcs (2,3), (3,4), and (4,5) are reversible arcs. At the root node, a complete tour could not be found. The value of the lower bound at the root node is 13. Initially, branching takes place along the two directions of travel of arc (2,3). Neither of these branches yields a complete tour. The node on the left is explored further, along the two directions of travel of arc (3,4). Once again, neither branch yields a complete tour. The leftmost node is again explored further along the two directions of travel of arc (4,5). The left hand branch yields a complete tour with a total time on tour of 18. This is considered to be the incumbent value. It can be seen that the value of the lower bound at the root node is less than the incumbent value and there still exist many unexplored nodes. Thus, it is observed that when the time windows were more restrictive, few branch-and-bound nodes were explored. However, when the time windows are less restrictive, more branch-and-bound nodes need to be explored.

2.8 Bender's Decomposition - an Overview

A brief overview of Bender's Decomposition is provided [69]. Bender's Decomposition is a technique for solving mixed integer programs (MIP) by solving a sequence of pure linear programs alternating with pure integer programs. It can be a powerful technique if the integer programs are fairly simple. The procedure has been explained in general terms.

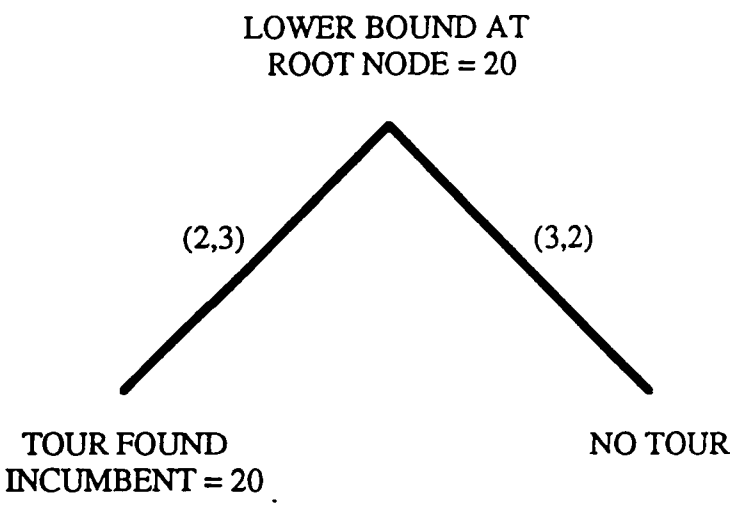

Figure 2.2: Baker's Method on Example with Time Window Set I

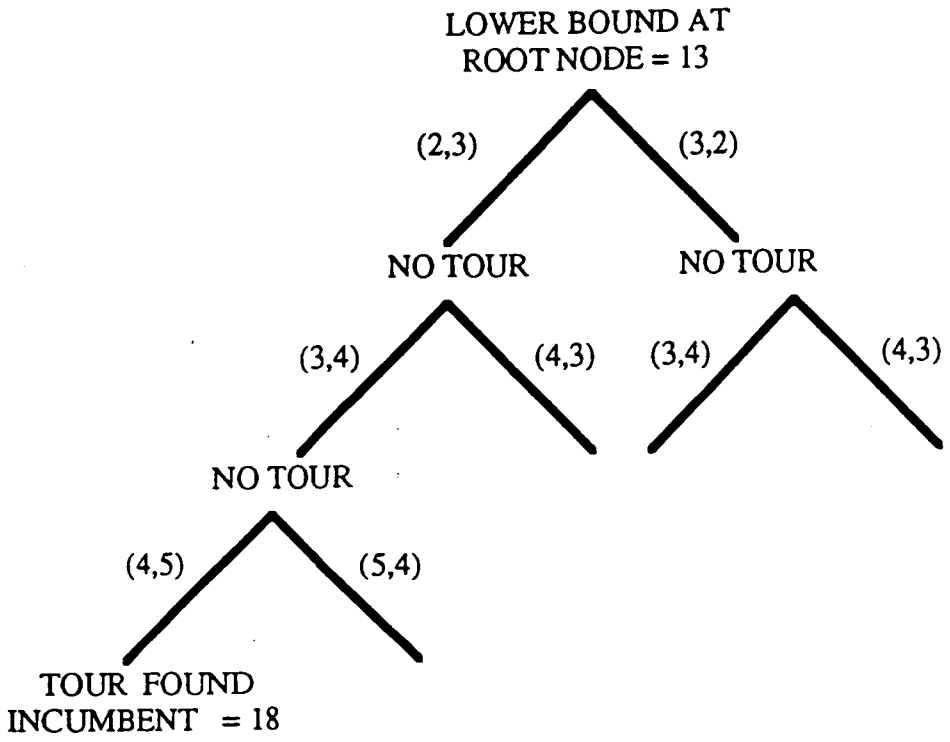

Figure 2.3: Baker's Method on Example with Time Window Set II

Consider the problem (MIP):

$$\min cx + dy \tag{2.35}$$

$$\text{s.t. } Ax + Dy \geq b, \tag{2.36}$$

$$Fx \geq g, \tag{2.37}$$

$$x \geq 0, \text{ integer}, \tag{2.38}$$

$$y \geq 0. \tag{2.39}$$

Let **P** represent the problem defined by (2.35,(2.36), (2.37), (2.38), (2.39). If the solution $\bar{x}$ were given for x, this problem would reduce to:

$$\min dy + c\bar{x} \tag{2.40}$$

$$\text{s.t. } Dy \geq b - A\bar{x}, \tag{2.41}$$

$$y \geq 0. \tag{2.42}$$

Let **PL** represent the problem defined by (2.40), (2.41), (2.42). **PL** is a pure linear program and has the dual:

$$\max u(b - A\bar{x}), \tag{2.43}$$

$$\text{s.t. } uD \leq d, \tag{2.44}$$

$$u \geq 0. \tag{2.45}$$

Let **DL** represent the dual of **PL** defined by the equations (2.43), (2.44) and (2.45). As can be seen, the convex polyhedron:

$$\mathbf{U} = \begin{cases} uD & \leq d \\ u & \geq 0 \end{cases}$$

is independent of x. Different values of $\bar{x}$ simply change the orientation of the objective function. The feasible region remains the same.

Any polyhedron is defined by the set of extreme points u^p, $p = 1, \ldots, P$ and extreme rays v^s, $s = 1, \ldots, S$. The polyhedron is a convex combination of u^p and a non-negative linear

combination of v^s, i.e., for some $w \in \mathbf{U}$, w can be expressed as $w = \alpha u + \beta v$, where $\sum_{i=1}^{P} \alpha_i = 1$, and $\alpha_i \geq 0$, $\forall\, i$ and $\beta \geq 0$. For a given solution $\bar{x}$, the dual **DL** is unbounded if and only if there exists v^s such that $v^s(b - A\bar{x}) > 0$. If the dual **DL** is unbounded, then the primal **PL** is infeasible. The choice of x should be such that it leads to a feasible solution to **PL** i.e., to a feasible $\mathbf{y}$. The choice of x should be restricted to values that ensure that $v^s(b - Ax) \leq 0$, $\forall s = 1, \ldots, S$.

Suppose the choice of $\bar{x}$ does indeed result in a feasible solution for **PL**, then this $\bar{x}$ leads to a finite (bounded) solution for **DL**, and this must occur at one of the extreme points of the polyhedron **U**. The optimal solution for **DL** is the best of these extreme points, therefore **DL** can be restated as

$$\max_{p=1,\ldots,P} u^p(b - A\bar{x}) \tag{2.46}$$

$$\text{s.t. } v^s(b - A\bar{x}) \leq 0\ ,\ s = 1, \ldots, S. \tag{2.47}$$

By the strong duality theorem, this also solves **PL**. Based on the above version of **DL**, the original MIP can now be restated as

$$\min_x(cx + \max_{p=1,\ldots,P} u^p(b - Ax)) \tag{2.48}$$

$$\text{s.t. } v^s(b - Ax) \leq 0\ ,\ s = 1, \ldots, S, \tag{2.49}$$

$$Fx \geq g, \tag{2.50}$$

$$x \geq 0,\ \text{integer},\ \forall\, x. \tag{2.51}$$

This is a pure integer program (in x).

Let,

$$z = cx + \max_{p=1,\ldots,P} u^p(b - Ax).$$

Then, the above formulation can be written as:

$$\min z \tag{2.52}$$

$$\text{s.t. } z \geq cx + u^p(b - Ax),$$

$$p = 1, \ldots, P, \tag{2.53}$$

$$v^s(b - Ax) \leq 0\ ,\ s = 1, \ldots, S, \tag{2.54}$$

$$Fx \geq g, \tag{2.55}$$

$$x \geq 0\ ,\ \text{integer},\ \forall\ x. \tag{2.56}$$

In general, **U** has many extreme points associated with it. Determining all the extreme points would be time consuming as well as lead to a large number of constraints in the integer program. Therefore, the extreme points (and extreme rays) are iteratively generated. The algorithm for this revised version of Benders Decomposition is as follows:

- **Step1:**(Initialization) Set the upper bound $z^u = \infty$. Set the lower bound $z^l = -\infty$. Choose some first solution $\bar{x}$. Go to Step 2.
- **Step 2:**(LP Phase) Solve **DL** with the given $\bar{x}$. The result is an extreme point or an extreme ray. Let $z^u = c\bar{x} + u^p(b - A\bar{x})$ if better than current z^u. Go to Step 3.
- **Step 3:**(IP Phase) Solve the IP with constraints corresponding to all extreme points and rays found so far. Let z^l = new optimal objective value (since this problem is a relaxation of the original problem) and let $\bar{x}$ =new optimal x. Go to Step 4.
- **Step 4:**(Termination Test) If $z^l < z^u$ then go to Step 2. Otherwise, $\bar{x}$ is optimal. Solve **PL** with this $\bar{x}$ to get an optimal y. Stop.
  Alternatively, stop if $(z^u - z^l) < \epsilon$, for a given ϵ.

Benders Decomposition is finite and convergent since the dual convex polyhedron has a finite number of extreme points and extreme rays and each one is generated at most once.

2.9 Scope of Research

This dissertation presents a solution methodology for the Traveling Salesman Problem with Time Windows. A modification of the objective function enables the solution of the Traveling Salesman Makespan Problem with Time Windows. The main characteristics of the new solution scheme are discussed with respect to the main characteristics of some of the solution schemes described in the literature.

The solution procedures of the literature review are summarized below.

- Baker's Method [2] minimizes the total time spent on the tour, i.e., it solves the Traveling Salesman Makespan Problem with Time Windows to optimality. It is not designed to handle general costs along the arcs of the problem. Also, as demonstrated in Section 2.7, tightness of time windows would be an important factor in limiting the size of the branch and bound tree.
- The dynamic programming methods are dependent on the size of the state space. They work well when the problem has additional constraints that can limit the size of the state space.
- Column generation approaches [45] are not applicable to the single vehicle routing problem.
- The two-commodity flow formulation approach due to Langevin, *et al.*, has been tested on problems that respond very well to preprocessing. As a result, a large number of arcs are rendered infeasible. The cardinality of the feasible arc set bears a linear relationship with the number of nodes of the problem. As demonstrated in Section 2.7, the nature of the time windows influences the performance of the algorithm considerably.

The new solution methodology presented in this dissertation exhibits the following characteristics:

- The Bender's Decomposition-based method is expected to work on problems over a range of time window widths and time window overlap.
- The formulation of the problem allows the objective function to represent internode weights in any form e.g., cost of travel, distance traveled etc.
- The method (a heuristic based on the method) is expected to work well on moderately sized problems i.e., problems with upto thirty five nodes.
- The method (heuristic) is modular in nature, with one of the modules being the solution of a pure Traveling Salesman Problem. Improved methods of solving asymmetric TSPs can therefore be incorporated easily into the solution procedure, thus improving the performance of the solution procedure.

- A variation in the objective function allows representation and solution of the Traveling Salesman Makespan Problem with Time Windows.

13optimal

Chapter 3

The Bender's Decomposition-Based Approach

3.1 Objectives

This chapter describes three different ways of representing the Traveling Salesman Problem with Time Windows. The Bender's Decomposition-based approach is discussed with respect to each of these representations. The cuts generated by each of these representations are discussed with respect to the information passed back to the master problem.

The organization of this chapter is as follows. Section 3.2 provides a description of the various terms used in representing the Traveling Salesman Problem with Time Windows. Section 3.3 presents a representation of the TSPTW wherein any tour/sequence is expressed in terms of arc variables. The cuts generated in this case provide information regarding infeasible subsequences represented as a set of contiguous arcs. Section 3.4 provides a representation of the TSPTW wherein any tour/sequence is expressed in terms of positional variables. The cuts in this case restrict the relative positions of nodes in the tour. Section 3.5 provides a

representation of the TSPTW wherein any tour/sequence can be expressed in two ways – in terms of the arcs in the tour and in terms of the positions occupied by the nodes in the tour. The cuts generated in this case provide information regarding infeasible sets of contiguous arcs and restrict relative positions of nodes in a feasible tour.

3.2 Problem Specification

As mentioned in the previous section, three different representations of the Traveling Salesman Problem with Time Windows are discussed. The TSPTW is defined over the graph $G(N, A)$ where N represents a set of n nodes and A represents the set of arcs (i, j) which span the set of nodes. For notational ease, the problem is actually represented as a Hamiltonian Path Problem. Node 1 refers to the depot while node $n + 1$, which is a duplicate of node 1, indicates a return to the depot at the end of the tour. A Hamiltonian Path is a trip through the nodes of the graph, starting at node 1, visiting each of nodes $2, \ldots, n$ exactly once and ending at node $n + 1$. It is assumed that arc (i, j) exists for every node pair i, j ie., the graph $G(N, A)$ is considered to be a complete graph. Infeasible arcs in specific problems are handled by associating a very high cost with the concerned arc. Other quantities pertaining to the problem are defined as follows:

$d_{i,j}$	$\equiv$	the time of travel from node i to node j,
$c_{i,j}$	$\equiv$	the cost of travel from node i to node j,
M	$\equiv$	a large positive integer,
s_i	$\equiv$	the service time at node i,
e_i	$\equiv$	the beginning of the time window at node i (earliest possible time of arrival at node i),
l_i	$\equiv$	the end of the time window at node i (latest possible time of arrival at node i).

All the quantities defined above are assumed to be non-negative integers. It is also assumed that any tour begins at the depot (node 1) at time zero.

3.3 Arc Variable Representation

The arc variable representation of the Traveling Salesman Problem with Time Windows is actually formulated as a Hamiltonian Path Problem. The node representing the depot, i.e., node 1, is duplicated thus increasing the number of nodes to $n+1$. The augmented node set is referred to as N'. The arcs $(i,1)$ are now replaced with the arcs $(i,n+1)$ and this arc set is referred to as A'. A visit to node $n+1$ in a solution to the Hamiltonian Path Problem actually indicates a return to the depot in the Traveling Salesman tour. The formulation is a mixed integer program with the objective being minimization of the cost of the tour.

3.3.1 Problem Formulation

Two types of variables are defined. They are as follows:

$$x_{i,j} = \begin{cases} 1 & \text{if arc (i,j) is selected as part of the tour} \\ 0 & \text{otherwise} \end{cases},$$

y_i = the actual time of arrival at node i.

The Traveling Salesman Problem with Time Windows can then be defined as follows:

$$\min \sum_{j=2}^{n} c_{1,j}x_{1,j} + \sum_{i=2}^{n}\sum_{\substack{j=2 \\ j\neq i}}^{n} c_{i,j}x_{i,j} + \sum_{i=2}^{n} c_{i,n+1}x_{i,n+1} \tag{3.1}$$

$$\text{subject to} \sum_{j=2}^{n} x_{1,j} = 1, \tag{3.2}$$

$$\sum_{\substack{j=2 \\ j\neq i}}^{n+1} x_{i,j} = 1,\ i = 2,\ldots,n, \tag{3.3}$$

$$\sum_{\substack{i=1 \\ i\neq j}}^{n} x_{i,j} = 1,\ j = 2,\ldots,n, \tag{3.4}$$

$$\sum_{i=2}^{n} x_{i,n+1} = 1, \tag{3.5}$$

$$\sum_{i\in S}\sum_{j\in S} x_{i,j} \leq |S|-1,\ \forall S \subset N', |S| \geq 2, \tag{3.6}$$

$$y_j - y_i \geq (s_i + d_{i,j}) + M(x_{i,j} - 1), \quad \forall (i,j) \in A', \tag{3.7}$$

$$y_i \geq e_i, \forall\, i \in N', \tag{3.8}$$

$$-y_i \geq -l_i, \forall\, i \in N', \tag{3.9}$$

$$x_{i,j} \in \{0,1\}, \forall (i,j) \in A', \tag{3.10}$$

$$y_i \geq 0, \forall i \in N'. \tag{3.11}$$

This formulation actually represents the TSPTW as a Hamiltonian Path Problem with time window restrictions. The objective function seeks to minimize the total cost of the tour. Constraints (3.2) – (3.6) and (3.10) define a Hamiltonian Path Problem which is a variation of the pure Traveling Salesman Problem. Constraint (3.2) ensures that the salesman travels to exactly one node from the depot (node 1). Constraint set (3.3) ensures that the salesman departs from every node, except node $(n+1)$ and constraint set (3.4) ensures that the salesman enters every node except the depot (node 1). Constraint set (3.5) ensures that the depot is entered exactly once. Constraint set (3.6) ensures that there are no subtours while constraint set (3.10) defines the binary nature of the $x_{i,j}$ variables.

The temporal nature of the problem is enforced by constraints (3.7), (3.8), (3.9), and (3.11). The variable y_i is a non-negative variable that is used to represent the temporal conditions. Constraint set (3.7) specifies that *if* arc (i,j) is in the Hamiltonian Path, *then* the time of arrival at node j must either be equal to or exceed the sum of the time of arrival at node i, the service time at node i, and the time of travel between node i and node j. Constraint sets (3.8) and (3.9) are time window constraints and specify respectively that the time of arrival at node i must be after the time window at node i has opened and before the time window closes. Constraint set (3.11) specifies the non-negative, continuous variable nature of y_i.

3.3.2 Problem Solution

The Traveling Salesman Problem with Time Windows, defined in Section 3.3.1 can be solved using Bender's Decomposition. To begin, the constraints defining the pure Hamiltonian Path Problem, i.e., the constraints containing $x_{i,j}$ variables only, along with the objective function can be treated as the master problem. Any solution to the master problem is a Traveling Salesman tour since a visit to node $n+1$ indicates a return to the depot (node 1). Therefore, the $x_{i,j}$ variables take on values of 1 or 0 depending on whether the arc (i,j) is in the tour or not. Given a solution $\bar{x}$ to the master problem, the values of $\bar{x}_{i,j}$ can be substituted into constraint set (3.7) resulting in a subproblem having constraints with only y_i variables. This concept can best be explained with the help of an example.

Consider a graph with five nodes. The constraints of the subproblem for this example (after some re-arrangement) can be described as shown in Table 3.1. Assuming that a solution $\bar{x}$ to the master problem is available, the values of the $x_{i,j}$ variables from this solution can be substituted in the constraints of the subproblem. The result is a set of constraints containing variables of the form y_i only. Suppose the solution to the master problem was the tour 1,2,3,4,5,6 (visit to node 6 indicating a return to the depot). This solution can be expressed as:

$$\bar{x} \equiv (\bar{x}_{1,2}=1, \bar{x}_{2,3}=1, \bar{x}_{3,4}=1, \bar{x}_{4,5}=1, \bar{x}_{5,6}=1, \text{all other } \bar{x}_{i,j}=0)$$

Substitute these values of $\bar{x}_{i,j}$ in the constraints of the subproblem represented by Table 3.1 and Table 3.2. This results in the constraints having the form shown in Table 3.3. As an example, consider CON1 from Table 3.1. It can be expressed as follows:

$$y_2 - y_1 \geq (s_1 + d_{1,2}) - M + M(1) \Longrightarrow y_2 - y_1 \geq (s_1 + d_{1,2})$$

Similarly, CON2 from Table 3.1 can be expressed as follows:

$$y_3 - y_1 \geq (s_1 + d_{1,3}) - M + M(0) \Longrightarrow y_3 - y_1 \geq (s_1 + d_{1,3}) - M$$

The maximum value that the left hand side of the above equation can have is $l_3 - e_1$. By definition, M is much greater that $(s_1 + d_{1,3}) - (l_3 - e_1)$ and hence, this constraint can be considered wholly redundant. Therefore, the subproblem reduces to the form shown in Table 3.3. The concept can be explained in general terms as follows:

Table 3.1: Initial Layout with x and y Variables

	y_1	y_2	y_3	y_4	y_5	y_6	$\geq$	RHS
OBJ	0	0	0	0	0	0		
CON1	-1	1						$(s_1 + d_{1,2}) - M + M\bar{x}_{1,2}$
CON2	-1		1					$(s_1 + d_{1,3}) - M + M\bar{x}_{1,3}$
CON3	-1			1				$(s_1 + d_{1,4}) - M + M\bar{x}_{1,4}$
CON4	-1				1			$(s_1 + d_{1,5}) - M + M\bar{x}_{1,5}$
CON5		-1	1					$(s_2 + d_{2,3}) - M + M\bar{x}_{2,3}$
CON6		-1		1				$(s_2 + d_{2,4}) - M + M\bar{x}_{2,4}$
CON7		-1			1			$(s_2 + d_{2,5}) - M + M\bar{x}_{2,5}$
CON8		-1				1		$(s_2 + d_{2,6}) - M + M\bar{x}_{2,6}$
CON9		1	-1					$(s_3 + d_{3,2}) - M + M\bar{x}_{3,2}$
CON10			-1	1				$(s_3 + d_{3,4}) - M + M\bar{x}_{3,4}$
CON11			-1		1			$(s_3 + d_{3,5}) - M + M\bar{x}_{3,5}$
CON12			-1			1		$(s_3 + d_{3,6}) - M + M\bar{x}_{3,6}$
CON13		1		-1				$(s_4 + d_{4,2}) - M + M\bar{x}_{4,2}$
CON14			1	-1				$(s_4 + d_{4,3}) - M + M\bar{x}_{4,3}$
CON15				-1	1			$(s_4 + d_{4,5}) - M + M\bar{x}_{4,5}$
CON16				-1		1		$(s_4 + d_{4,6}) - M + M\bar{x}_{4,6}$
CON17		1			-1			$(s_5 + d_{5,2}) - M + M\bar{x}_{5,2}$
CON18			1		-1			$(s_5 + d_{5,3}) - M + M\bar{x}_{5,3}$
CON19				1	-1			$(s_5 + d_{5,4}) - M + M\bar{x}_{5,4}$
CON20					-1	1		$(s_5 + d_{5,6}) - M + M\bar{x}_{5,6}$

Table 3.2: Initial Layout with x and y Variables *cont.*

	y_1	y_2	y_3	y_4	y_5	y_6	$\geq$	RHS
CON21	1							e_1
CON22		1						e_2
CON23			1					e_3
CON24				1				e_4
CON25					1			e_5
CON26						1		e_6
CON27	-1							$-l_1$
CON28		-1						$-l_2$
CON29			-1					$-l_3$
CON30				-1				$-l_4$
CON31					-1			$-l_5$
CON32						-1		$-l_6$

Table 3.3: Final Layout with x and y Variables

	y_1	y_2	y_3	y_4	y_5	y_6	$\geq$	RHS
OBJ	0	0	0	0	0	0		
CON1	-1	1						$(s_1 + d_{1,2})$
CON5		-1	1					$(s_2 + d_{2,3})$
CON10			-1	1				$(s_3 + d_{3,4})$
CON15				-1	1			$(s_4 + d_{4,5})$
CON20					-1	1		$(s_5 + d_{5,6})$
CON21	1							e_1
CON22		1						e_2
CON23			1					e_3
CON24				1				e_4
CON25					1			e_5
CON26						1		e_6
CON27	-1							$-l_1$
CON28		-1						$-l_2$
CON29			-1					$-l_3$
CON30				-1				$-l_4$
CON31					-1			$-l_5$
CON32						-1		$-l_6$

Without loss of generality, assume that the sequence $1, 2, 3, \ldots, n, n+1$ is the solution to the master problem and let it be represented by $\bar{x} \equiv (\bar{x}_{1,2} = 1, \bar{x}_{2,3} = 1, \bar{x}_{3,4} = 1, \ldots, \bar{x}_{n,n+1} = 1,$ all other $\bar{x}_{i,j} = 0)$. Then, the subproblem can be represented as:

$$\min 0 \tag{3.12}$$

$$-y_i + y_j \geq (s_i + d_{i,j}) + M(\bar{x}_{i,j} - 1), \ \forall (i,j) \in A^{'}, \tag{3.13}$$

$$y_i \geq e_i, \ \forall i \in N^{'}, \tag{3.14}$$

$$-y_i \geq -l_i, \ \forall i \in N^{'}. \tag{3.15}$$

The constraints represented by equation (3.13) can be separated into two sets and the subproblem can then be represented as:

$$\min 0 \tag{3.16}$$

$$-y_i + y_j \geq (s_i + d_{i,j}), \ \forall (i,j) : \bar{x}_{i,j} = 1, \tag{3.17}$$

$$-y_i + y_j \geq (s_i + d_{i,j}) - M, \ \forall (i,j) : \bar{x}_{i,j} = 0, \tag{3.18}$$

$$y_i \geq e_i, \ \forall i \in N^{'}, \tag{3.19}$$

$$-y_i \geq -l_i, \ \forall i \in N^{'}. \tag{3.20}$$

Consider the constraints represented by equation (3.18). The maximum value of the left hand side of this equation can be represented as $-e_i + l_j$. By definition, M is greater than $(s_i + d_{i,j}) + e_i - l_j$; therefore these constraints are wholly redundant and can be eliminated. Therefore, the subproblem can now be represented as:

$$\min 0 \tag{3.21}$$

$$-y_i + y_{i+1} \geq (s_i + d_{i,i+1}), \ i = 1, 2, \ldots, n, \tag{3.22}$$

$$y_i \geq e_i, \ i = 1, 2, \ldots, n+1, \tag{3.23}$$

$$-y_i \geq -l_i, \ i = 1, 2, \ldots, n+1. \tag{3.24}$$

The objective function has a value of zero. Therefore, to solve the subproblem we only need to determine whether or not the sequence is feasible with respect to the time windows. The

subproblem can be solved by merely checking to see if the route provided by the master problem is feasible with respect to time windows. If the tour is such that every node on the tour is visited within its time window, then a feasible solution to the Traveling Salesman Problem with Time Windows has been found. However, if any node is visited after its time window has closed, then the route provided by the master problem is infeasible with respect to the time windows.

3.3.3 The X-Cuts

If the subproblem is feasible, then the sequence represented by the solution to the master problem satisfies all the time windows. If the route provided by the master problem (again, given as $1, 2, 3, \ldots, n, n+1$) is infeasible with respect to the time windows, then there exists a subsequence $i, i+1, i+2, \ldots, i+m-1, i+m$, where the salesman starts from node i at time e_i, travels through nodes $i+1, \ldots, i+m-1$, and reaches the node $i+m$, such that:

$$e_i + (s_i + d_{i,i+1}) + \ldots + (s_{i+m-1} + d_{i+m-1,i+m}) - l_{i+m} \quad > \quad 0 \qquad (3.25)$$

The subsequence $i, i+1, \ldots, i+m-1, i+m$ is referred to as an *infeasible* subsequence. If the salesman starts at node i at time e_i, travels through the nodes $i+1$, $i+2$ etc. and reaches the node $i+m$ before the close of the time window at node $i+m$, the subsequence $i, i+1, \ldots, i+m-1, i+m$ is referred to as a *feasible* subsequence. The subproblem dual is represented as:

$$\max \sum_{i=1}^{n} \pi_i(s_i + d_{i,i+1}) + \sum_{j=1}^{n+1} \lambda_j e_j - \sum_{j=1}^{n+1} \mu_j l_j$$

$$\begin{array}{lllll}
-\pi_1 & +\lambda_1 & -\mu_1 & = & 0 \\
\pi_1 - \pi_2 & +\lambda_2 & -\mu_2 & = & 0 \\
\pi_2 - \pi_3 & +\lambda_3 & -\mu_3 & = & 0 \\
 & \ddots & & & \\
\pi_{n-1} - \pi_n & +\lambda_n & -\mu_n & = & 0 \\
\pi_n & +\lambda_{n+1} & -\mu_{n+1} & = & 0
\end{array}$$

$$\pi_i, \lambda_j, \mu_j \geq 0$$

$$i = 1, \ldots, n;\ j = 1, \ldots, n+1$$

The above system is homogeneous and its feasible region is a cone emanating from the origin.

Statement: For every subsequence $i, i+1, i+2, \ldots, i+m$, we can construct an extreme ray of the dual of the subproblem (after purging the wholly redundant constraints). This is true whether the subsequence is feasible or not.

Proof: The subproblem dual can be represented as described previously. The columns of the dual corresponding to a subsequence $i, i+1, i+2, \ldots, i+m-1, i+m$ can be represented as depicted by the matrix below.

Row	λ_i	π_i	π_{i+1}	π_{i+2}	π_{i+3}	$\ldots$	π_{i+m-1}	μ_{i+m}
1								
2								
$\vdots$								
i	1	-1						
$i+1$		1	-1					
$i+2$			1	-1				
$i+3$				1	-1			
$\vdots$					$\vdots$			
$i+m-1$							-1	
$i+m$							1	-1
$\vdots$								
$n+1$								

We wish to show that this set of columns corresponds to an extreme ray of the feasible region. This is true if the columns are part of a basis that represents a Basic Feasible Solution (BFS) for a system consisting of the dual augmented with a constraint of the form:

$$\sum_{i=1}^{n} \pi_i + \sum_{i=1}^{n+1} \lambda_i + \sum_{i=1}^{n+1} \mu_i = 1$$

Pick a basis (for the augmented dual) of full row rank that contains the columns corresponding to the subsequence, i.e., the columns corresponding to λ_i, π_i, π_{i+1}, ..., π_{i+m-1}, μ_{i+m}. These columns can be represented as depicted in the matrix below.

Row	λ_i	π_i	π_{i+1}	π_{i+2}	π_{i+3}	...	π_{i+m-1}	μ_{i+m}
1								
2								
$\vdots$								
i	1	-1						
$i+1$		1	-1					
$i+2$			1	-1				
$i+3$				1	-1			
$\vdots$						$\vdots$		
$i+m-1$							-1	
$i+m$							1	-1
$\vdots$								
$n+1$								
$n+2$	1	1	1	1	1		1	1

The remaining columns of the augmented dual can be represented as depicted in the matrix on the next page.

Pick as many columns from this matrix (starting with the column corresponding to π_1) as necessary to form a full basis. The columns of this basis are seen to be linearly independent and a solution that has $\lambda_i = \pi_i = \pi_{i+1} = \ldots = \pi_{i+m-1} = \mu_{i+m} = 1/(m+2)$, all other $\pi_j, \lambda_j, \mu_j = 0$, is feasible. Therefore, the basis with columns corresponding to the subsequence $i, i+1, \ldots, i+m-1, i+m$ is a BFS for the augmented dual. The subsequence therefore corresponds to an extreme ray in the original dual. •

If the primal is feasible then it has an optimal objective value of zero. Therefore, the dual is also feasible with an optimal objective value of zero. If the primal subproblem is infeasible, then the dual can be either infeasible or unbounded. We know that the dual is feasible since there exists a feasible solution obtained by setting all variables to zero, i.e., the extreme point

from which the cone emanates. Therefore, if the primal is infeasible, the dual is unbounded and, hence there exists an extreme ray with diverging objective.

Row	π_1	π_2	...	π_{i-1}	π_{i+m}	...	π_n
1	-1						
2	1	-1					
3		1					
$\vdots$							
$i-1$				-1			
i				1			
$\vdots$							
$i+m$					-1		
$i+m+1$					1		
$\vdots$							
n							-1
$n+1$							1
$n+2$	1	1		1	1		1

Therefore, the primal subproblem can be solved by checking for feasibility of the sequence (with respect to time windows) and if not feasible, then producing an infeasible subsequence $i, i+1, i+2, \ldots, i+m$. The infeasible subsequence in the primal corresponds to an unbounded objective extreme ray in the dual and generates a cut of the form

$$e_i + (s_i + d_{i,i+1}) + M(x_{i,i+1} - 1) + \ldots + (s_{i+m-1} + d_{i+m-1,i+m}) + M(x_{i+m-1,i+m} - 1) - l_{i+m} \leq 0 \qquad (3.26)$$

This equation can be simplified as follows.

$$[e_i - l_{i+m} + (s_i + d_{i,i+1}) + \ldots + (s_{i+m-1} + d_{i+m-1,i+m})] - mM + M(x_{i,i+1} + x_{i+1,i+2}, \ldots, x_{i+m-1,i+m}) \leq 0.$$

The above expression reduces to the form

$$M(x_{i,i+1} + x_{i+1,i+2} + \ldots + x_{i+m-1,i+m}) \leq mM - [e_i + (s_i + d_{i,i+1})$$

$$(s_{i+1} + d_{i+1,i+2}) + \ldots + (s_{i+m-1} + d_{i+m-1,i+m}) \quad - \quad l_{i+m}].$$

This further reduces to the form

$$x_{i,i+1} + x_{i+1,i+2} + \ldots + x_{i+m-1,i+m} \quad \leq \quad m - \epsilon \tag{3.27}$$

where ϵ is a positive quantity defined by the expression

$$\epsilon \quad = \quad \frac{e_i + (s_i + d_{i,i+1}) + \ldots + (s_{i+m-1} + d_{i+m-1,i+m}) - l_{i+m}}{M}.$$

By definition, the denominator is much larger than the numerator in the above expression and hence, ϵ is strictly less than one. The expression (3.27) represents the extreme ray cut. The right hand side can be rounded down to the nearest integer, yielding:

$$x_{i,i+1} + x_{i+1,i+2} + \ldots + x_{i+m-1,i+m} \quad \leq \quad m - 1. \tag{3.28}$$

Expression (3.28) has the same effect on the integer polytope as expression (3.27) and is hence applicative for this approach. This cut is referred to as the **Simple X-Cut**. The Simple X-Cut prevents infeasible subsequences from being a part of any solution to the master problem.

Details regarding the theory of linear programming, which formed the basis for the arguments regarding the dual feasible region and the relationship between the primal and dual, can be found in [70].

3.4 Positional Variable Representation

In this representation of the Traveling Salesman Problem with Time Windows, a new binary variable is introduced. This variable specifies the position in the sequence occupied by each node. The formulation of the Hamiltonian Path master problem is the same as it was in the arc variable case. Additional constraints linking the two binary variables are introduced in this representation. The temporal constraints contain the continuous temporal variables and the binary positional variables.

3.4.1 Problem Formulation

Three types of variables are defined as follows:

$$x_{i,j} = \begin{cases} 1 & \text{if arc (i,j) is selected as part of the tour} \\ 0 & \text{otherwise} \end{cases},$$

$$z_{p,q} = \begin{cases} 1 & \text{if node p is the } q^{th} \text{ node in the sequence} \\ 0 & \text{otherwise} \end{cases},$$

t_q = the time of arrival at q^{th} node in the sequence.

The Traveling Salesman Problem with Time Windows can now be defined as follows:

$$\min \sum_{j=2}^{n} c_{1,j}x_{1,j} + \sum_{i=2}^{n}\sum_{\substack{j=2 \\ j\neq i}}^{n} c_{i,j}x_{i,j} + \sum_{i=2}^{n} c_{i,n+1}x_{i,n+1} \tag{3.29}$$

$$\text{subject to} \sum_{j=2}^{n} x_{1,j} = 1, \tag{3.30}$$

$$\sum_{\substack{j=2 \\ j\neq i}}^{n+1} x_{i,j} = 1,\ i = 2,\ldots,n, \tag{3.31}$$

$$\sum_{\substack{i=1 \\ i\neq j}}^{n} x_{i,j} = 1,\ j = 2,\ldots,n, \tag{3.32}$$

$$\sum_{i=2}^{n} x_{i,n+1} = 1, \tag{3.33}$$

$$\sum_{i\in S}\sum_{j\in S} x_{i,j} \leq |S| - 1,\ \forall S \subset N',|S| \geq 2, \tag{3.34}$$

$$\sum_{q=2}^{n} z_{p,q} = 1,\ p = 2,\ldots,n, \tag{3.35}$$

$$\sum_{p=2}^{n} z_{p,q} = 1,\ q = 2,\ldots,n, \tag{3.36}$$

$$z_{1,1} = 1, \tag{3.37}$$

$$z_{n+1,n+1} = 1, \tag{3.38}$$

$$1 - x_{i,j} - z_{i,q} + z_{j,q+1} \geq 0, \ \forall (i,j) \in A' \text{ and } q = 1, \ldots, n, \tag{3.39}$$

$$1 + x_{i,j} - z_{i,q} - z_{j,q+1} \geq 0, \ \forall (i,j) \in A' \text{ and } q = 1, \ldots, n, \tag{3.40}$$

$$t_2 - t_1 \geq (s_1 + d_{1,j}) + M(z_{1,1} + z_{j,2} - 2), \quad j = 2, \ldots, n, \tag{3.41}$$

$$t_{q+1} - t_q \geq (s_i + d_{i,j}) + M(z_{i,q} + z_{j,q+1} - 2), \quad i, j = 2, \ldots, n, i \neq j; \quad q = 2, \ldots, n-1, \tag{3.42}$$

$$t_{n+1} - t_n \geq (s_i + d_{i,n+1}) + M(z_{i,n} + z_{n+1,n+1} - 2), \quad i = 2, \ldots, n, \tag{3.43}$$

$$t_q \geq e_p + M(z_{p,q} - 1), \quad \forall p, q \in \{1, \ldots, n+1\}, \tag{3.44}$$

$$-t_q \geq -l_p + M(z_{p,q} - 1), \quad \forall p, q \in \{1, \ldots, n+1\}, \tag{3.45}$$

$$x_{i,j} \in \{0,1\}, \ \forall (i,j) \in A', \tag{3.46}$$

$$z_{p,q} \in \{0,1\}, \ \forall p, q = 1, \ldots, n+1, \tag{3.47}$$

$$t_q \qquad \geq \qquad 0,\ q = 1, \ldots, n+1. \tag{3.48}$$

In this representation of the TSPTW, the tour that is submitted to the subproblem is defined in terms of the positions occupied by the nodes. As before, constraints (3.30) – (3.34), and (3.46) specify the conditions for a Hamiltonian Path Problem. Constraint set (3.35) specifies that every node can be assigned to exactly one position, while constraint set (3.36) specifies that to every position in the tour exactly one node can be assigned. Constraints (3.37) and (3.38) specify that node 1 is in the first position (start at the depot) and node $n+1$ is in position $n+1$ (return to depot at the end of the tour) respectively. Constraint set (3.39) specifies that if the salesman travels from node i to node j and node i is in position q in the tour, then node j *must* be in position $q+1$ in the tour. Constraint set (3.40) specifies that if node i is in position q and node j is in position $q+1$, then the salesman *must* travel from node i to node j. Constraint sets (3.39) and (3.40) are referred to as coupling constraints since they link the $x_{i,j}$ variables with the $z_{p,q}$ variables. Thus, the two different binary variables specify the two different ways of representing a tour. Constraints (3.41), (3.42), and (3.43) ensure that the time of arrival at the node in position $q+1$ is greater than or equal to the sum of the time of arrival at the node in position q, the service time at the node in position q, and the time of travel between the nodes in positions q and $q+1$. Constraint sets (3.44) and (3.45) ensure that every node is reached after its time window has opened and before its time window has closed. Finally, constraint set (3.46) defines the binary nature of the $z_{p,q}$ variables while constraint set (3.47) defines the variable t_q as a non-negative real continuous variable. The objective function, as before, seeks to minimize the total cost of the tour.

3.4.2 Problem Solution

In a procedure similar to that adopted for the formulation with x and y variables, the mixed integer program can be solved by using a Bender's Decomposition-based approach. The master problem in this case consists of the objective function and all the constraints that have only integer variables. The solution to the master problem is a Hamiltonian Path. This solution represents a Traveling Salesman tour since a visit to node $n+1$ indicates a return to the depot (node 1). The $z_{p,q}$ variables take on values of 1 or 0. If the variable $z_{p,q}$ has a value 1, then node p is in position q of the tour. Alternately, if variable $z_{p,q}$

has a value 0, then node p is not in position q of the tour. Given a solution to the master problem, the values of $z_{p,q}$ can be substituted in constraints (3.41), (3.42), (3.43), (3.44), and (3.45) resulting in a subproblem having constraints with only t_q variables, i.e., a pure linear program. The subproblem can now be solved to result in either a feasible completion to the TSPTW or in a cut which can be passed on to the master problem. A small example, with five nodes, is used to better explain the concepts of subproblem formulation and cut generation. Assuming availability of a solution to the master problem, the constraints of the subproblem are given in Tables 3.4 and 3.5. Assume the solution to the master problem was the tour 1,2,3,4,5,6 (visit to node 6 indicating a return to the depot). The solution in terms of $z_{p,q}$ variables can be specified as:

$$\bar{z} \equiv (\bar{z}_{1,1} = 1, \bar{z}_{2,2} = 1, \bar{z}_{3,3} = 1, \bar{z}_{4,4} = 1, \bar{z}_{5,5} = 1, \bar{z}_{6,6} = 1 \text{ all other } \bar{z}_{p,q} = 0).$$

Substituting the above values of $\bar{z}_{1,1}$ and $\bar{z}_{2,2}$ in C1 from Table 3.4 yields the inequality which follows:

$$t_2 - t_1 \geq (s_1 + d_{1,2}) + M(1 + 1 - 2) \Longrightarrow t_2 - t_1 \geq (s_1 + d_{1,2}).$$

Substituting the values of $\bar{z}_{1,1}$ and $\bar{z}_{3,2}$ in C2 from Table 3.4 yields the following inequality:

$$t_2 - t_1 \geq (s_1 + d_{1,3}) + M(1 + 0 - 2) \Longrightarrow t_2 - t_1 \geq (s_1 + d_{1,3}) - M.$$

By definition, M is much greater than $(s_1 + d_{1,3}) - (l_3 - e_1)$ where $(l_3 - e_1)$ is the maximum value the left hand side of the above equation can have. Hence this constraint can be considered to be wholly redundant. Therefore, the relevant constraints of the subproblem (corresponding to the tour presented by the master problem) can be represented as indicated by Table 3.6. The concept can be explained in general terms.

Without loss of generality, assume that the sequence $1, 2, 3, \ldots, n+1$ is the solution to the master problem and let it be represented by $\bar{z} \equiv (\bar{z}_{1,1} = 1, \bar{z}_{2,2} = 1, \ldots, \bar{z}_{n,n} = 1, \bar{z}_{n+1,n+1} = 1$, all other $\bar{z}_{p,q} = 0)$. This sequence has node 1 in position 1, node 2 in position 2, ..., node n in position n, node $n+1$ in position $n+1$. The subproblem can be represented as

Table 3.4: Initial Layout with z and t Variables

	t_1	t_2	t_3	t_4	t_5	t_6	$\geq$	RHS
C1	-1	1						$(s_1 + d_{1,2}) + M(\bar{z}_{1,1} + \bar{z}_{2,2} - 2)$
C2	-1	1						$(s_1 + d_{1,3}) + M(\bar{z}_{1,1} + \bar{z}_{3,2} - 2)$
C3	-1	1						$(s_1 + d_{1,4}) + M(\bar{z}_{1,1} + \bar{z}_{4,2} - 2)$
C4	-1	1						$(s_1 + d_{1,5}) + M(\bar{z}_{1,1} + \bar{z}_{5,2} - 2)$
C5		-1	1					$(s_2 + d_{2,3}) + M(\bar{z}_{2,2} + \bar{z}_{3,3} - 2)$
				$\vdots$				
C16		-1	1					$(s_5 + d_{5,4}) + M(\bar{z}_{5,2} + \bar{z}_{4,3} - 2)$
C17			-1	1				$(s_2 + d_{2,3}) + M(\bar{z}_{2,3} + \bar{z}_{3,4} - 2)$
				$\vdots$				
C28			-1	1				$(s_5 + d_{5,4}) + M(\bar{z}_{5,3} + \bar{z}_{4,4} - 2)$
C29				-1	1			$(s_2 + d_{2,3}) + M(\bar{z}_{2,4} + \bar{z}_{3,5} - 2)$
				$\vdots$				
C40				-1	1			$(s_5 + d_{5,4}) + M(\bar{z}_{5,4} + \bar{z}_{4,5} - 2)$
C41					-1	1		$(s_2 + d_{2,6}) + M(\bar{z}_{2,5} + \bar{z}_{6,6} - 2)$
C42					-1	1		$(s_3 + d_{3,6}) + M(\bar{z}_{3,5} + \bar{z}_{6,6} - 2)$
C43					-1	1		$(s_4 + d_{4,6}) + M(\bar{z}_{4,5} + \bar{z}_{6,6} - 2)$
C44					-1	1		$(s_5 + d_{5,6}) + M(\bar{z}_{5,5} + \bar{z}_{6,6} - 2)$
C45	1							$e_1 - M + M\bar{z}_{1,1}$
C46		1						$e_2 - M + M\bar{z}_{2,2}$
				$\vdots$				
C49		1						$e_5 - M + M\bar{z}_{5,2}$
C50			1					$e_2 - M + M\bar{z}_{2,3}$
				$\vdots$				

Table 3.5: Initial Layout with z and t Variables *cont.*

	t_1	t_2	t_3	t_4	t_5	t_6	$\geq$	RHS
C53			1					$e_5 - M + M\bar{z}_{5,3}$
C54				1				$e_2 - M + M\bar{z}_{2,4}$
				⋮				
C57				1				$e_5 - M + M\bar{z}_{5,4}$
C58					1			$e_2 - M + M\bar{z}_{2,5}$
				⋮				
C61					1			$e_5 - M + M\bar{z}_{5,5}$
C62						1		$e_6 - M + M\bar{z}_{6,6}$
C63	-1							$-l_1 - M + M\bar{z}_{1,1}$
C64		-1						$-l_2 - M + M\bar{z}_{2,2}$
				⋮				
C67		-1						$-l_5 - M + M\bar{z}_{5,2}$
C68			-1					$-l_2 - M + M\bar{z}_{2,3}$
				⋮				
C71			-1					$-l_5 - M + M\bar{z}_{5,3}$
C72				-1				$-l_2 - M + M\bar{z}_{2,4}$
				⋮				
C75				-1				$-l_5 - M + M\bar{z}_{5,4}$
C76					-1			$-l_2 - M + M\bar{z}_{2,5}$
				⋮				
C79					-1			$-l_5 - M + M\bar{z}_{5,5}$
C80						-1		$-l_6 - M + M\bar{z}_{6,6}$

Table 3.6: Final Layout with z and t Variables

	t_1	t_2	t_3	t_4	t_5	t_6	$\geq$	RHS
C1	-1	1						$(s_1 + d_{1,2})$
C5		-1	1					$(s_2 + d_{2,3})$
C21			-1	1				$(s_3 + d_{3,4})$
C35				-1	1			$(s_4 + d_{4,5})$
C44					-1	1		$(s_5 + d_{5,6})$
C45	1							e_1
C46		1						e_2
C51			1					e_3
C56				1				e_4
C61					1			e_5
C62						1		e_6
C63	-1							$-l_1$
C64		-1						$-l_2$
C69			-1					$-l_3$
C74				-1				$-l_4$
C79					-1			$-l_5$
C80						-1		$-l_6$

follows:

$$\min 0 \tag{3.49}$$

$$-t_q + t_{q+1} \geq (s_i + d_{i,j}) + M(\bar{z}_{i,q} + \bar{z}_{j,q+1} - 2),$$

$$\forall (i,j) \in A^{'},\ q = 1, \ldots, n, \tag{3.50}$$

$$t_q \geq e_i + M(\bar{z}_{i,q} - 1),\ \ i = 1, \ldots, n+1$$

$$q = 1, \ldots, n+1, \tag{3.51}$$

$$-t_q \geq -l_i + M(\bar{z}_{i,q} - 1),\ \ i = 1, \ldots, n+1$$

$$q = 1, \ldots, n+1. \tag{3.52}$$

Each of the constraint sets in the above formulation can be segregated into two sets and the subproblem can be represented as follows:

$$\min 0 \tag{3.53}$$

$$-t_q + t_{q+1} \geq (s_i + d_{i,j}),\ \ \forall i,j,q : \bar{z}_{i,q} + \bar{z}_{j,q+1} = 2, \tag{3.54}$$

$$-t_q + t_{q+1} \geq (s_i + d_{i,j}) - K(M),\ \ \forall i,j,q : \bar{z}_{i,q} + \bar{z}_{j,q+1} \leq 1, \tag{3.55}$$

$$t_q \geq e_i,\ \ \forall\, i,q : \bar{z}_{i,q} = 1, \tag{3.56}$$

$$t_q \geq e_i - M,\ \ \forall\, i,q : \bar{z}_{i,q} = 0, \tag{3.57}$$

$$-t_q \geq -l_i,\ \ \forall\, i,q : \bar{z}_{i,q} = 1, \tag{3.58}$$

$$-t_q \geq -l_i - M,\ \ \forall\, i,q : \bar{z}_{i,q} = 0. \tag{3.59}$$

Consider the constraints represented by the inequality (3.55), where K represents a constant with value greater than zero. The maximum value of the left hand side of this equation can be represented as $-e_i + l_j$. By definition, M is greater than $(s_i + d_{i,j}) - e_i + l_j$; therefore, these constraints are wholly redundant and can be eliminated. Next, consider the constraints represented by the inequality (3.57). The earliest time the node in position q can be reached is e_i. Rearranging the expression results in $M \geq 0$, and by definition M is greater than zero. Therefore these constraints are also wholly redundant and can be

eliminated. Finally, consider the constraints represented by equation (3.59). The latest time the node in position q can be visited is l_i. Rearranging the expression again yields $M \geq 0$ which, by definition of M, is true. Therefore, these constraints can also be considered wholly redundant and eliminated. Therefore, the subproblem can now be represented as follows:

$$\min 0 \tag{3.60}$$

$$-t_i + t_{i+1} \geq (s_i + d_{i,i+1}), \quad i = 1, 2, \ldots, n, \tag{3.61}$$

$$t_i \geq e_i, \quad i = 1, 2, \ldots, n+1, \tag{3.62}$$

$$-t_i \geq -l_i, \quad i = 1, 2, \ldots, n+1. \tag{3.63}$$

The resultant primal subproblem is identical to the primal subproblem that was generated for the arc variable representation of the Traveling Salesman Problem with Time Windows. Therefore, all the arguments regarding the primal subproblem that were presented for the arc variable representation hold true for the positional variable representation also.

3.4.3 The Z-Cuts

Feasibility of the primal subproblem indicates that the time windows are satisfied by the sequence $1, 2, \ldots, n+1$ provided by the master problem. If this sequence is infeasible with respect to the time windows, then there exists a subsequence $i, i+1, i+2, \ldots, i+m$, where starting at node i at time e_i, and traveling through the nodes $i+1, \ldots, i+m-1$, node $i+m$ is reached such that:

$$e_i + (s_i + d_{i,i+1}) + \ldots + (s_{i+m-1} + d_{i+m-1,i+m}) - l_{i+m} > 0. \tag{3.64}$$

The arguments regarding the correspondence between subsequences and extreme rays that were presented for the arc variable representation of the problem hold true for the positional variable representation as well. Therefore, it follows that the infeasible subsequence corresponds to an extreme ray of the dual and hence a cut can be generated as follows:

$$\begin{aligned} & e_i + M(z_{i,i} - 1) + (s_i + d_{i,i+1}) + M(z_{i,i} + \\ & z_{i+1,i+1} - 2) + \ldots + (s_{i+m-1} + d_{i+m-1,i+m}) + \\ & M(z_{i+m-1,i+m-1} + z_{i+m,i+m} - 2) - l_{i+m} + M(z_{i+m,i+m} - 1) \leq 0. \end{aligned} \tag{3.65}$$

The above expression can be rearranged and expressed as follows:

$$[e_i - l_{i+m} + (s_i + d_{i,i+1}) + \ldots + (s_{i+m-1} + d_{i+m-1,i+m})]$$

$$-2M(m+1) + 2M(z_{i,i} + z_{i+1,i+1} + \ldots + z_{i+m,i+m}) \quad \leq \quad 0.$$

The above expression reduces to:

$$2M(z_{i,i} + z_{i+1,i+1} + \ldots + z_{i+m,i+m}) \quad \leq$$

$$2(m+1)M - [e_i + (s_i + d_{i,i+1})$$

$$+ \ldots + (s_{i+m-1} + d_{i+m-1,i+m}) - l_{i+m}].$$

This expression further reduces to:

$$z_{i,i} + z_{i+1,i+1} + z_{i+2,i+2} + \ldots + z_{i+m,i+m} \quad \leq \quad (m+1) - \epsilon, \tag{3.66}$$

where ϵ is a positive quantity and is defined below.

$$\epsilon = \frac{e_i + (s_i + d_{i,i+1}) + \ldots + (s_{i+m-1} + d_{i+m-1,i+m}) - l_{i+m}}{2M}.$$

By definition, the denominator is much larger than the numerator in the above expression and therefore ϵ is strictly less than one. The expression (3.66) represents the extreme ray cut. The right hand side can be rounded down to the nearest integer yielding:

$$z_{i,i} + z_{i+1,i+1} + z_{i+2,i+2} + \ldots + z_{i+m,i+m} \quad \leq \quad m. \tag{3.67}$$

Expression (3.67) has the same effect on the integer polytope as expression (3.66) and is hence applicative to this approach. This cut is referred to as the **Simple Z-Cut**. The Simple Z-Cut prevents the subsequence which has node i in position i, node $i+1$ in position $i+1$, ..., node $i+m$ in position $i+m$.

An examination of the expressions for the Simple X-Cut and the Simple Z-Cut shows that the Simple X-Cut prevents sequences that contain the infeasible subsequence anywhere in the sequence. The Simple Z-Cut, however, only prevents sequences that contain the infeasible subsequence in the particular positions described by the $z_{p,q}$ variables. Therefore, the Simple X-Cut dominates the Simple Z-Cut. However, there exist conditions where expressing the sequence in terms of positional variables (and therefore generating cuts in terms of positional variables) is advantageous and these conditions will be studied next.

Precedence Cuts: Suppose an infeasible sequence as represented by equation (3.64) is identified, and, it is also true that,

$$e_i + (s_i + d_{i,i+m}) - l_{i+m} \quad > \quad 0. \tag{3.68}$$

Clearly, in problem preprocessing (described in Chapter 2), the arc $(i, i+m)$ would have been identified. However, subsequences such as $i, j_1, j_2, \ldots, j_p, i+m$ for some p would not have been addressed by preprocessing. Then, assuming that

$$d_{i,j} \leq d_{i,k} + d_{k,j}, \ \forall \ i, k, j,$$

any subsequence of the form

$$i, \ldots, \{ \text{ any combination of nodes} \neq i, i+m, 1, n+1 \ \}, \ldots, i+m$$

will cause the tour containing the subsequence to be infeasible. This implies a precedence relationship that specifies that any feasible tour for the TSPTW must have the salesman visit node i after node $i+m$ has been visited. The cuts that ensure this precedence relationship can be developed as shown below.

The tour provided by the master problem would be infeasible if any one of the following situations occurred:

$$e_i + (s_i + d_{i,i+m}) - l_{i+m} \quad > \quad 0, \tag{3.69}$$

$$e_i + (s_i + d_{i,j_1}) + (s_{j_1} + d_{j_1,i+m}) - l_{i+m} \quad > \quad 0,$$

$$j_1 \neq 1, i, i+m, n+1, \tag{3.70}$$

$$e_i + (s_i + d_{i,j_1}) + (s_{j_1} + d_{j_1,j_2}) + (s_{j_2} + d_{j_2,i+m}) - l_{i+m} \quad > \quad 0,$$

$$j_1 \neq j_2 \text{ and } j_1, j_2 \neq 1, i, i+m, n+1, \tag{3.71}$$

$$\vdots$$

$$e_i + (s_i + d_{i,j_1}) + (s_{j_1} + d_{j_1,j_2}) + \ldots +$$

$$(s_{j_{n-3}} + d_{j_{n-3},i+m}) - l_{i+m} \quad > \quad 0,$$

$$j_k \neq j_{k+1}, k = 1, \ldots, n-4,$$

$$j_k \neq 1, i, i+m, n+1; \ k = 1, \ldots, n-3. \tag{3.72}$$

The Z-Cuts needed to prevent the existence of any of the above conditions can now be developed in the manner shown below.

Consider condition (3.69) and assume that node i occupies position 2 and node $i+m$ occupies position 3 in the sequence provided by the master problem. Then, the Simple Z-cut that prevents this infeasible subsequence from occuring can be represented as

$$e_i + (s_i + d_{i,i+m}) - l_{i+m} \leq M(1 - z_{i,2}) + M(2 - z_{i,2} - z_{i+m,3}) + M(1 - z_{i+m,3}).$$

The above equation reduces to

$$e_i + (s_i + d_{i,i+m}) - l_{i+m} \leq 4M - 2M(z_{i,2} + z_{i+m,3}).$$

The above expression can be simplified as

$$M(z_{i,2} + z_{i+m,3}) \leq 2M - [e_i + (s_i + d_{i,i+m}) - l_{i+m}]/2. \tag{3.73}$$

Condition (3.73) is satisfied if:

$$z_{i,2} + z_{i+m,3} \leq 1.$$

Similarly, if the sequence provided by the master problem had node i in position 3 and node $i+m$ in position 4, then the cut preventing this situation would be

$$z_{i,3} + z_{i+m,4} \leq 1,$$

and in general the cut would be of the form

$$z_{i,q} + z_{i+m,q+1} \leq 1,\ q = 2, \ldots, n-1. \tag{3.74}$$

Consider equation (3.70) and assume node i occupies position 2, node j_1 occupies position 3 and node $i+m$ occupies position 4 in the sequence provided by the master problem. Then, adopting the same arguments as before, the cut preventing this condition would be of the form

$$z_{i,2} + z_{j_1,3} + z_{i+m,4} \leq 2,$$

and in general, the cut would be of the form

$$z_{i,q} + z_{j_1,q+1} + z_{i+m,q+2} \leq 2,$$

$$\text{where } q = 2, \ldots, n-2 \text{ and } j_1 \neq 1, i, i+m, n+1. \tag{3.75}$$

The cuts ensuring precedence can then be specified as follows:

$$z_{i,q} + z_{i+m,q+1} \leq 1,$$

$$q = 2, \ldots, n-1, \tag{3.76}$$

$$z_{i,q} + z_{j_1,q+1} + z_{i+m,q+2} \leq 2,$$

$$q = 2, \ldots, n-2,\ j_1 \neq 1, i, i+m, n+1, \tag{3.77}$$

$$\vdots$$

$$z_{i,q} + z_{j_1,q+1} + z_{j_2,q+2} + \ldots + z_{j_{n-3},q+n-3} + z_{i+m,n} \leq n-2,$$

$$j_k \neq j_{k+1},\ k = 1, \ldots, n-4,$$

$$j_k \neq 1, i, i+m, n+1,\ k = 1, \ldots, n-3;\ q = 2. \tag{3.78}$$

The Simple Z-cuts represented by equation set (3.76) - (3.78) can be replaced by the set of cuts defined as follows:

$$z_{i+m,2} - z_{i,3} - z_{i,4} - \ldots - z_{i,n} \leq 0, \tag{3.79}$$

$$z_{i+m,3} - z_{i,4} - z_{i,5} - \ldots - z_{i,n} \leq 0, \tag{3.80}$$

$$\vdots$$

$$z_{i+m,n-1} - z_{i,n} \leq 0, \tag{3.81}$$

$$z_{i+m,n} \leq 0. \tag{3.82}$$

There are $n-1$ cuts in the above set. Equation set (3.76) - (3.78) contains a factorial number of inequalities, and every one of these conditions is ensured by the cuts in the set (3.79) - (3.82). Therefore the equation set (3.79) - (3.82) ensures that node i does not precede node $i+m$ in any feasible tour. These cuts are referred to as **Precedence Cuts**. The Simple X-Cuts needed to achieve the same result are

$$x_{i,i+m} \leq 0, \tag{3.83}$$

$$x_{i,j_1} + x_{j_1,i+m} \leq 1, \tag{3.84}$$

$$\vdots$$

$$x_{i,j_1} + x_{j_1,j_2} + \ldots + x_{j_{n-3},i+m} \quad \leq \quad n-1, \tag{3.85}$$

where $j_1 - j_2 - \ldots - j_k$; $k = 2, \ldots, n-3$ represents all possible subsequences. Equation set (3.83) - (3.85) contains a factorial number of cuts. The number of Simple X-Cuts needed can be expressed as

$$1 + \sum_{g=1}^{n-3} (g! * (\frac{(n-3)!}{g!(n-3-g)!}).$$

Therefore it is advantageous to represent precedence relationships using Z-Cuts. General precedence relationships that exist in real-life problems can very easily be expressed by the Precedence Z-Cuts.

Positional Cuts: Suppose the tour generated by the master problem is infeasible because

$$e_i + (s_i + d_{i,i+1}) + (s_{i+1} + d_{i+1,i+2}) - l_{i+2} \quad > \quad 0. \tag{3.86}$$

In addition, suppose it is true that

$$e_i + (s_i + d_{i,j}) + (s_j + d_{j,k}) - l_k \quad > \quad 0$$

$$\forall\ j \neq 1, i, i+1, i+2, n+1\ ;\ k \neq 1, i, i+1, i+2, n+1 \text{ and } j \neq k. \tag{3.87}$$

Conditions (3.86) and (3.87) indicate that node i must occur in the last two positions, i.e.,positions $n-1$ and n in any feasible tour, and this condition can be specified by the cut:

$$z_{i,1} + z_{i,2} + \ldots + z_{i,n-2} \quad \leq \quad 0. \tag{3.88}$$

This cut is referred to as a **Positional Cut**. The following Simple X-Cuts would be needed to ensure the same results.

$$x_{k,j} + x_{j,n+1} \quad \leq \quad 1, \forall\ j, k \neq i, 1, n+1 \text{ and } k \neq j. \tag{3.89}$$

It can be observed that the number of Simple X-Cuts needed would be of $\mathcal{O}(n^2)$. Situations requiring a certain node to occur in the last two positions of a sequence (excluding the depot) do not occur frequently; however, if such a condition were to exist, then it could be expressed easily using the Positional Z-Cuts.

3.5 Combined Representation

This representation of the Traveling Salesman Problem with Time Windows represents a tour/sequence in two ways – as a sequence of arcs and as a sequences of nodes assigned to specific positions. The problem formulation is similar to that for the Positional Variable Representation and the three kinds of variables used are the same as those used in the Positional Variable Version. The only difference is that constraints (3.41), (3.42), and (3.43) have been replaced by the constraints (3.90), (3.91), and (3.92). These constraints contain variables t, x and z, while constraints (3.41) - (3.43) contain only variables t and z. The constraints (3.90) - (3.92) are:

$$\begin{aligned} t_2 - t_1 \quad &\geq \quad (s_1 + d_{1,j}) \\ &\quad +M(z_{1,1} + z_{j,2} + x_{1,j} - 3), \\ &\quad j = 2, \ldots, n, \end{aligned} \tag{3.90}$$

$$\begin{aligned} t_{q+1} - t_q \quad &\geq \quad (s_i + d_{i,j}) \\ &\quad +M(z_{i,q} + z_{j,q+1} + x_{i,j} - 3), \\ &\quad i, j = 2, \ldots, n, i \neq j; \\ &\quad q = 2, \ldots, n-1, \end{aligned} \tag{3.91}$$

$$\begin{aligned} t_{n+1} - t_n \quad &\geq \quad (s_i + d_{i,n+1}) \\ &\quad +M(z_{i,n} + z_{n+1,n+1} + x_{i,n+1} - 3), \\ &\quad i = 2, \ldots, n. \end{aligned} \tag{3.92}$$

3.5.1 Problem Solution

Once again, a Bender's Decomposition-based approach is adopted. The master problem has the same form as described in Section 3.4.2 and the solution to the master problem is a Traveling Salesman tour. In this case, the tour is represented in two ways – as a sequence

of arcs that form the tour ($\bar{x}_{i,j}$ variables) and as a sequence of positions occupied by the nodes in the tour ($\bar{z}_{p,q}$ variables). The values of $\bar{x}_{i,j}$ and $\bar{z}_{p,q}$ are substituted in the temporal constraints to result in a subproblem having t_q variables only. The subproblem is now solved. The result is either a feasible completion to the TSPTW or a set of cuts that can be passed back to the master problem. A small example (with five nodes) is again used to explain the concepts of subproblem formulation. The constraints of the subproblem are shown in Tables 3.7 and 3.8. Assume the solution to the master problem to be the tour 1,2,3,4,5,6 (visit to node 6 indicating a return to the depot), the solution in terms of $x_{i,j}$ variables can be specified as

$$\bar{x} \equiv (\bar{x}_{1,2} = 1, \bar{x}_{2,3} = 1, \bar{x}_{3,4} = 1, \bar{x}_{4,5} = 1, \bar{x}_{5,6} = 1 \text{ all other } \bar{x}_{i,j} = 0),$$

and the solution in terms of $z_{p,q}$ variables can be specified as

$$\bar{z} \equiv (\bar{z}_{1,1} = 1, \bar{z}_{2,2} = 1, \bar{z}_{3,3} = 1, \bar{z}_{4,4} = 1, \bar{z}_{5,5} = 1, \bar{z}_{6,6} = 1 \text{ all other } \bar{z}_{p,q} = 0).$$

Substituting these values of $\bar{x}_{1,2}$, $\bar{z}_{1,1}$ and $\bar{z}_{2,2}$ in C1 from Table 3.7 results in

$$t_2 - t_1 \geq (s_1 + d_{1,2}) + M(1 + 1 + 1 - 3) \Longrightarrow t_2 - t_1 \geq (s_1 + d_{1,2}).$$

Substituting the values of $\bar{x}_{1,3}$, $\bar{z}_{1,1}$ and $\bar{z}_{3,2}$ in C2 from Table 3.7 results in

$$t_2 - t_1 \geq (s_1 + d_{1,3}) + M(1 + 0 + 0 - 3) \Longrightarrow t_2 - t_1 \geq (s_1 + d_{1,3}) - 2M.$$

As explained in Section 3.3.2 and Section 3.4.2, the constraint C2 can be considered redundant. The relevant constraints of the subproblem (corresponding to the tour provided by the master problem) can be represented as indicated by Table 3.9. The subproblem merely checks for feasibility of the tour (provided by the master problem) with respect to the time windows. If this tour is feasible with respect to the time windows then a feasible solution to the TSPTW has been found. Otherwise, cuts are generated that are passed back to the master problem.

3.5.2 Cuts for Combined Representation

This formulation of the TSPTW allows both X-Cuts and Z-Cuts to be passed back to the master problem. The generation of these cuts is explained briefly in this section. More details can be found in Section 3.3.3 for the X-Cuts and Section 3.4.3 for the Z-Cuts.

Table 3.7: Initial Layout with x, z and t Variables

	t_1	t_2	t_3	t_4	t_5	t_6	$\geq$	RHS
C1	-1	1						$(s_1 + d_{1,2}) + M(z_{1,1} + z_{2,2} + x_{1,2} - 3)$
C2	-1	1						$(s_1 + d_{1,3}) + M(z_{1,1} + z_{3,2} + x_{1,3} - 3)$
C3	-1	1						$(s_1 + d_{1,4}) + M(z_{1,1} + z_{4,2} + x_{1,4} - 3)$
C4	-1	1						$(s_1 + d_{1,5}) + M(z_{1,1} + z_{5,2} + x_{1,5} - 3)$
C5		-1	1					$(s_2 + d_{2,3}) + M(z_{2,2} + z_{3,3} + x_{2,3} - 3)$
				$\vdots$				
C16		-1	1					$(s_5 + d_{5,4}) + M(z_{5,2} + z_{4,3} + x_{5,4} - 3)$
C17			-1	1				$(s_2 + d_{2,3}) + M(z_{2,3} + z_{3,4} + x_{2,3} - 3)$
				$\vdots$				
C28			-1	1				$(s_5 + d_{5,4}) + M(z_{5,3} + z_{4,4} + x_{5,4} - 3)$
C29				-1	1			$(s_2 + d_{2,3}) + M(z_{2,4} + z_{3,5} + x_{2,3} - 3)$
				$\vdots$				
C40				-1	1			$(s_5 + d_{5,4}) + M(z_{5,4} + z_{4,5} + x_{5,4} - 3)$
C41					-1	1		$(s_2 + d_{2,6}) + M(z_{2,5} + z_{6,6} + x_{2,6} - 3)$
C42					-1	1		$(s_3 + d_{3,6}) + M(z_{3,5} + z_{6,6} + x_{3,6} - 3)$
C43					-1	1		$(s_4 + d_{4,6}) + M(z_{4,5} + z_{6,6} + x_{4,6} - 3)$
C44					-1	1		$(s_5 + d_{5,6}) + M(z_{5,5} + z_{6,6} + x_{5,6} - 3)$
C45	1							$e_1 - M + Mz_{1,1}$
C46		1						$e_2 - M + Mz_{2,2}$
				$\vdots$				
C49		1						$e_5 - M + Mz_{5,2}$
C50			1					$e_2 - M + Mz_{2,3}$
				$\vdots$				

Table 3.8: Initial Layout with x, z and t Variables *cont.*

	t_1	t_2	t_3	t_4	t_5	t_6	$\geq$	RHS
C53			1					$e_5 - M + Mz_{5,3}$
C54				1				$e_2 - M + Mz_{2,4}$
				⋮				
C57				1				$e_5 - M + Mz_{5,4}$
C58					1			$e_2 - M + Mz_{2,5}$
				⋮				
C61					1			$e_5 - M + Mz_{5,5}$
C62						1		$e_6 - M + Mz_{6,6}$
C63	-1							$-l_1 - M + Mz_{1,1}$
C64		-1						$-l_2 - M + Mz_{2,2}$
				⋮				
C67		-1						$-l_5 - M + Mz_{5,2}$
C68			-1					$l_2 - M + Mz_{2,3}$
				⋮				
C71			-1					$-l_5 - M + Mz_{5,3}$
C72				-1				$-l_2 - M + Mz_{2,4}$
				⋮				
C75				-1				$-l_5 - M + Mz_{5,4}$
C76					-1			$-l_2 - M + Mz_{2,5}$
				⋮				
C79					-1			$-l_5 - M + Mz_{5,5}$
C80						-1		$-l_6 - M + Mz_{6,6}$

Table 3.9: Final Layout with x, z and t Variables

	t_1	t_2	t_3	t_4	t_5	t_6	$\geq$	RHS
C1	-1	1						$(s_1 + d_{1,2})$
C5		-1	1					$(s_2 + d_{2,3})$
C21			-1	1				$(s_3 + d_{3,4})$
C35				-1	1			$(s_4 + d_{4,5})$
C44					-1	1		$(s_5 + d_{5,6})$
C45	1							e_1
C46		1						e_2
C51			1					e_3
C56				1				e_4
C61					1			e_5
C62						1		e_6
C63	-1							$-l_1$
C64		-1						$-l_2$
C69			-1					$-l_3$
C74				-1				$-l_4$
C79					-1			$-l_5$
C80						-1		$-l_6$

Without loss of generality, assume a subsequence to be represented as $i, i+1, \ldots, i+m-1, i+m$. Consider the infeasible subsequence:

$$e_i + (s_i + d_{i,i+1}) + \ldots + (s_{i+m-1} + d_{i+m-1,i+m}) - l_{i+m} \quad > \quad 0. \tag{3.93}$$

As explained in Section 3.3.3 and Section 3.4.3, any subsequence corresponds to an extreme ray and an infeasible subsequence results in an unbounded objective. The condition for feasibility can be specified as:

$$\begin{aligned} & e_i + M(z_{i,i} - 1) + (s_i + d_{i,i+1}) + M(z_{i,i} + z_{i+1,i+1} + x_{i,i+1} - 3) + \\ & + \ldots + (s_{i+m-1} + d_{i+m-1,i+m}) + M(z_{i+m-1,i+m-1} + z_{i+m,i+m} + \\ & \qquad x_{i+m-1,i+m} - 3) - l_{i+m} + M(z_{i+m,i+m} - 1) \quad \leq \quad 0. \end{aligned} \tag{3.94}$$

To generate the Simple X-Cut, substitute the values of $\bar{z}$ variables from the tour. Therefore,

$$\begin{aligned} & e_i + (1-1)M + (s_i + d_{i,i+1}) + M(1 + 1 + x_{i,i+1} - 3) + \ldots \\ & +(s_{i+m-1} + d_{i+m-1,i+m}) + M(1 + 1 + x_{i+m-1,i+m} - 3) + \\ & \qquad -l_{i+m} + M(1-1) \quad \leq \quad 0. \end{aligned} \tag{3.95}$$

The above equation reduces to the form

$$\begin{aligned} & e_i + (s_i + d_{i,i+1}) + \ldots + (s_{i+m-1} + d_{i+m-1,i+m} - l_{i+m} \\ & \qquad +M(x_{i,i+1} + \ldots + x_{i+m-1,i+m}) - mM \quad \leq \quad 0. \end{aligned} \tag{3.96}$$

The above expression further reduces to the form

$$(x_{i,i+1} + \ldots, x_{i+m-1,i+m}) \quad \leq \quad m - \epsilon, \tag{3.97}$$

where ϵ is a positive quantity (with value strictly less than one) defined as

$$\epsilon \quad = \quad \frac{e_i + (s_i + d_{i,i+1}) + \ldots + (s_{i+m-1} + d_{i+m-1,i+m}) - l_{i+m}}{M}. \tag{3.98}$$

As before, the right hand side can be rounded down to the nearest integer, yielding

$$x_{i,i+1} + \ldots + x_{i+m-1,i+m} \quad \leq \quad m - 1. \tag{3.99}$$

To generate the basic Z-Cut, (that will lead to all other forms of Z-Cut) substitute the $\bar{x}$ values from the tour. Therefore,

$$e_i + M(z_{i,i} - 1) + (s_i + d_{i,i+1}) + M(z_{i,i} + z_{i+1,i+1} + 1 - 3) + \ldots +$$

$$M(z_{i+m-1,i+m-1} + z_{i+m,i+m} + 1 - 3) - l_{i+m} + M(z_{i+m,i+m} - 1) \quad \leq \quad 0. \quad (3.100)$$

The above equation reduces to

$$e_i + (s_i + d_{i,i+1}) + \ldots + (s_{i+m-1} + d_{i+m-1,i+m}) - l_{i+m} +$$

$$2M(z_{i,i} + z_{i+1,i+1} + \ldots + z_{i+m,i+m}) - 2M(m+1) \quad \leq \quad 0. \qquad (3.101)$$

This equation further reduces to

$$z_{i,i} + z_{i+1,i+1} + \ldots + z_{i+m,i+m} \quad \leq \quad (m+1) - \epsilon, \qquad (3.102)$$

where ϵ is a positive quantity that is defined as

$$\epsilon \quad = \quad \frac{e_i + (s_i + d_{i,i+1}) + \ldots + (s_{i+m-1} + d_{i+m-1,i+m}) - l_{i+m}}{2M}.$$

Rounding down the right hand side of the above equation to the nearest integer, the expression for the simple Z-Cut can be stated as

$$z_{i,i} + z_{i+1,i+1} + \ldots, z_{i+m,i+m} \quad \leq \quad m. \qquad (3.103)$$

As explained in Section 3.4.3, this Simple Z-Cut can be used to develop the other forms of the Z-Cut (namely the Precedence Cuts and the Positional Cuts).

For every tour (provided by the master problem) that is infeasible with respect to time windows, both X-Cuts and Z-Cuts are developed and passed back to the master problem

3.5.3 The Optimal Solution

The three methods of representing the Traveling Salesman Problem with Time Windows described previously are all mixed integer programs and are effectively solved by Bender's Decomposition as demonstrated in the previous sections. The feasible region of the subproblem dual has been shown to be a cone emanating from the origin. Therefore, the

feasible region of the subproblem dual contains a single extreme point which is the origin. Unbounded feasible solutions for the dual correspond to infeasible solutions for the primal subproblem. The single bounded feasible solution for the dual (extreme point) corresponds to a feasible solution to the primal subproblem. Therefore, the first feasible solution to the TSPTW is optimal.

3.6 The Bender's Decomposition-Based Method on the TSMPTW

A close relative of the Traveling Salesman Problem with Time Windows is the Traveling Salesman Makespan Problem with Time Windows. In this problem, the objective is to minimize the total time spent on the tour i.e., the completion time of the tour. The formulations presented in Section 3.2, Section 3.3 and Section 3.4 can be modified to represent the Traveling Salesman Makespan Problem with Time Windows.

The objective function in all three versions of the Traveling Salesman Problem with Time Windows is replaced with:

$$\min \sum_{j=2}^{n} c_{1,j}x_{1,j} \quad + \sum_{i=2}^{n} \sum_{\substack{j=2 \\ j \neq i}}^{n} c_{i,j}x_{i,j} \quad + \sum_{i=2}^{n} c_{i,n+1}x_{i,n+1} + \sum_{j=2}^{n+1} w_j,$$

where w_j represents the waiting time at node j, i.e., the time spent in waiting for the time window at node j to open. The temporal constraint linking the time of arrival at node i and the time of arrival at node $i+1$ is modified in the manner described below.

For the arc variable representation of the TSPTW, constraint (3.7) is replaced by the constraint

$$y_j - y_i - w_j \geq (s_i + d_{i,j}) + M(x_{i,j} - 1),$$

$$\forall\, (i,j) \in A^{'},\ i \neq j. \tag{3.104}$$

For the positional variable representation of the TSPTW, constraints (3.41), (3.42), and (3.43) are replaced by the constraints

$$t_2 - t_1 - w_2 \geq (s_1 + d_{1,2}) + M(z_{1,1} + z_{j,2} - 2),$$

$$j = 2, \ldots, n, \tag{3.105}$$

$$t_{q+1} - t_q - w_{q+1} \geq (s_i + d_{i,j}) + M(z_{i,q} + z_{j,q+1} - 2),$$

$$i, j = 2, \ldots, n,\ i \neq j,$$

$$q = 2, \ldots, n-1, \tag{3.106}$$

$$t_{n+1} - t_n - w_{n+1} \geq (s_i + d_{i,n+1}) + M(z_{i,n} + z_{n+1,n+1} - 2),$$

$$i = 2, \ldots, n. \tag{3.107}$$

For the combined representation of the TSPTW, constraints (3.90), (3.91), and (3.92) are replaced by the constraints

$$t_2 - t_1 - w_2 \geq (s_1 + d_{1,2}) + M(z_{1,1} + z_{j,2} + x_{1,j} - 3),$$

$$j = 2, \ldots, n, \tag{3.108}$$

$$t_{q+1} - t_q - w_{q+1} \geq (s_i + d_{i,j}) + M(z_{i,q} + z_{j,q+1} + x_{i,j} - 3),$$

$$i, j = 2, \ldots, n,\ i \neq j,$$

$$q = 2, \ldots, n-1, \tag{3.109}$$

$$t_{n+1} - t_n - w_{n+1} \geq (s_i + d_{i,n+1}) + M(z_{i,n} + z_{n+1,n+1} + x_{i,n+1} - 3),$$

$$i = 2, \ldots, n. \tag{3.110}$$

For convenience, all further discussions will be with respect to the arc variable representation. However, as observed with respect to the TSPTW, all the arguments are valid for the positional variable representation and the combined representation as well.

3.6.1 Correspondence Between Subsequences and Extreme Points

The resultant primal subproblem (given a sequence $1, 2, \ldots, n, n+1$ as solution to the master problem) can be represented as

$$\min \sum_{j=2}^{n+1} w_j \tag{3.111}$$

$$-y_i + y_{i+1} - w_{i+1} \geq (s_i + d_{i,i+1}),\ i = 1, 2, \ldots, n, \tag{3.112}$$

$$y_i \geq e_i,\ i = 1, 2, \ldots, n+1, \tag{3.113}$$

$$-y_i \geq -l_i,\ i = 1, 2, \ldots, n+1, \tag{3.114}$$

$$w_j \geq 0,\ j = 2, \ldots, n+1. \tag{3.115}$$

Observe that

$$y_{n+1} = \sum_{j=2}^{n} c_{1,j}x_{1,j} + \sum_{i=2}^{n}\sum_{\substack{j=2 \\ j\neq i}}^{n} c_{i,j}x_{i,j} + \sum_{i=2}^{n} c_{i,n+1}x_{i,n+1} + \sum_{j=2}^{n+1} w_j.$$

Therefore,

$$\sum_{j=2}^{n+1} w_j = y_{n+1} - (\sum_{j=2}^{n} c_{1,j}x_{1,j} + \sum_{i=2}^{n}\sum_{\substack{j=2 \\ j\neq i}}^{n} c_{i,j}x_{i,j} + \sum_{i=2}^{n} c_{i,n+1}x_{i,n+1}).$$

There is no need to specify l_{n+1} since y_{n+1} is being minimized. Also,

$$\sum_{j=2}^{n} c_{1,j}x_{1,j} + \sum_{i=2}^{n}\sum_{\substack{j=2 \\ j\neq i}}^{n} c_{i,j}x_{i,j} + \sum_{i=2}^{n} c_{i,n+1}x_{i,n+1}$$

is a constant and can be ignored. Therefore, the primal subproblem can be represented as

$$\min y_{n+1} \tag{3.116}$$

$$-y_i + y_{i+1} - w_{i+1} \geq (s_i + d_{i,i+1})\ i = 1, 2, \ldots, n, \tag{3.117}$$

$$y_i \geq e_i\ i = 1, 2, \ldots, n+1, \tag{3.118}$$

$$-y_i \geq -l_i\ i = 1, 2, \ldots, n, \tag{3.119}$$

$$w_j \geq 0,\ j = 2, \ldots, n+1. \tag{3.120}$$

This formulation affects the analysis of the feasible region and the Bender's implementation presented previously. The subproblem no longer has a zero objective. Therefore, the right hand side of the subproblem dual changes and there exists a 1 in the row corresponding to y_{n+1}. The subproblem dual can be represented as

$$\max \sum_{i=1}^{n} \pi_i(s_i + d_{i,i+1}) + \sum_{i=1}^{n+1} \lambda_i e_i - \sum_{i=1}^{n} \mu_i l_i$$

$$
\begin{array}{rrrcl}
-\pi_1 & +\lambda_1 & -\mu_1 & = & 0 \\
\pi_1 - \pi_2 & +\lambda_2 & -\mu_2 & = & 0 \\
\pi_2 - \pi_3 & +\lambda_3 & -\mu_3 & = & 0 \\
 & \ddots & & & \\
\pi_{n-1} - \pi_n & +\lambda_n & -\mu_n & = & 0 \\
\pi_n & +\lambda_{n+1} & & = & 1 \\
-\pi_1 & & & \leq & 0 \\
-\pi_2 & & & \leq & 0 \\
 & \ddots & & & \\
-\pi_{n-1} & & & \leq & 0 \\
-\pi_n & & & \leq & 0 \\
 & & \pi_i, \lambda_j, \mu_i & \geq & 0
\end{array}
$$

$$i = 1, \ldots, n, \;\; j = 1, \ldots, n+1$$

The last set of constraints are redundant since these conditions are ensured by the nonnegativity constraints. Therefore, the dual reduces to

$$\max \sum_{i=1}^{n} \pi_i (s_i + d_{i,i+1}) + \sum_{i=1}^{n+1} \lambda_i e_i - \sum_{i=1}^{n} \mu_i l_i$$

$$
\begin{array}{rrrcl}
-\pi_1 & +\lambda_1 & -\mu_1 & = & 0 \\
\pi_1 - \pi_2 & +\lambda_2 & -\mu_2 & = & 0 \\
\pi_2 - \pi_3 & +\lambda_3 & -\mu_3 & = & 0 \\
 & \ddots & & & \\
\pi_{n-1} - \pi_n & +\lambda_n & -\mu_n & = & 0
\end{array}
$$

$$\pi_n \qquad + \lambda_{n+1} \qquad = 1$$

$$\pi_i, \lambda_j, \mu_i \geq 0$$

$$i = 1, \ldots, n, \quad j = 1, \ldots, n+1$$

Any subsequence $i, i+1, \ldots, i+m-1, i+m$ where $i+m \neq n+1$ corresponds to an extreme ray as shown with respect to the TSPTW.

Statement: For any feasible sequence, there exists an extreme point that contains, in the basis, columns corresponding to a minimal cardinality subsequence $i, i+1, \ldots, n, n+1$. The subsequence $i, i+1, \ldots, n, n+1$ has no inserted waiting time.

Proof: Consider the columns corresponding to the subsequence $i, i+1, \ldots, n, n+1$. Add enough columns (from the set corresponding to π_1, π_2, ..., π_{i-1}) to create a basis of full row rank (i.e., of rank $n+1$). This basis can be represented as

Row	π_1	π_2	...	π_{i-1}	λ_i	π_i	π_{i+1}	...	π_{n-1}	π_n
1	-1									
2	1	-1								
$\vdots$										
$i-1$			...	-1						
i			...	1	1	-1				
$i+1$			...			1	-1			
$\vdots$										
$n-1$								...	-1	
n								...	1	-1
$n+1$								...		1

The columns of the above basis are linearly independent. The basis corresponds to a feasible solution ($\lambda_i = \pi_i = \pi_{i+1} = \ldots = \pi_n = 1$, all other $\pi_j, \lambda_j, \mu_j = 0$) and, therefore, the above set of columns represents a BFS of the subproblem dual. Therefore, a feasible sequence $1, 2, \ldots, n, n+1$ corresponds to an extreme point whose basis contains columns corresponding to the subsequence $i, i+1, \ldots, n, n+1$. •

Therefore, solving the primal subproblem results in one of two situations:

- the sequence obtained is feasible,
- or the sequence obtained is infeasible.

3.6.2 Cuts Corresponding to a Feasible Subsequence

The value of the dual objective function is given by the expression

$$\lambda_i e_i + \pi_i(s_i + d_{i,i+1}) + \ldots + \pi_n(s_n + d_{n,n+1}).$$

It has been shown, previously, that a feasible solution to the dual has

$$\lambda_i = \pi_i = \pi_{i+1} = \ldots = \pi_n = 1.$$

Therefore,

$$\text{dual objective value} = e_i + (s_i + d_{i,i+1}) + \ldots + (s_n + d_{n,n+1}).$$

The corresponding subsequence in the master problem would be

$$e_i + (s_i + d_{i,i+1}) + M(x_{i,i+1} - 1) + \ldots + (s_n + d_{n,n+1}) + M(x_{n,n+1} - 1).$$

The Simple X-Cut prohibiting this subsequence from ever occuring as part of a sequence can be represented as

$$e_i + (s_i + d_{i,i+1}) + M(x_{i,i+1} - 1) + \ldots + (s_n + d_{n,n+1}) + M(x_{n,n+1} - 1) \quad \leq \quad 0.$$

Expanding this expression results in

$$M(x_{i,i+1} + \ldots + x_{n,n+1}) - M(n + 1 - i) + [e_i + (s_i + d_{i,i+1}) + \ldots + (s_n + d_{n,n+1})] \quad \leq \quad 0.$$

The expression now reduces to

$$M(x_{i,i+1} + \ldots + x_{n,n+1}) \quad \leq \quad M(n + 1 - i) - [e_i + (s_i + d_{i,i+1}) + \ldots + (s_n + d_{n,n+1})],$$

and can be further simplified as

$$x_{i,i+1} + \ldots + x_{n,n+1} \quad \leq \quad n + 1 - i - \epsilon,$$

where ϵ is defined as

$$\epsilon = \frac{e_i + (s_i + d_{i,i+1}) + \ldots + (s_n + d_{n,n+1})}{M}.$$

Every component of the numerator in the expression for ϵ is a positive quantity. Therefore, the numerator is a positive quantity. The denominator, by definition, is a large positive number. Therefore, the value of ϵ is strictly greater than zero and strictly less than 1. Therefore, the expression for the Simple X-Cut can be represented as

$$x_{i,i+1} + x_{i+1,i+2} + \ldots + x_{n,n+1} \leq n - i. \tag{3.121}$$

Equation (3.121) represents the Simple X-Cut that prevents the subsequence from being part of any feasible solution to the master problem.

In the implementation of Bender's Decomposition on the TSMPTW, the first feasible solution is *not necessarily* the optimal solution. The first feasible solution simply corresponds to one of the extreme points of the feasible region. However, this feasible solution provides a valid upper bound to the Traveling Salesman Makespan Problem with Time Windows. The objective function value corresponding to an optimal solution to the master problem represents a valid lower bound to the TSMPTW. The procedure continues by solving the master problem and upgrading the lower bound if the solution to the master problem is greater than the current lower bound. Similarly, the upper bound is replaced with a better solution to the TSMPTW, whenever it becomes available. The procedure continues until the lower bound becomes greater than or equal to the upper bound. Then, the current upper bound solution will be the optimal solution to the Traveling Salesman Makespan Problem with Time Windows.

Chapter 4

An Enumerative Implementation of the Bender's Decomposition-Based Solution Methodology

4.1 Objectives

The main objective of this chapter is to show that, over a limited set of test problems, the Bender's Decomposition-based solution methodology can perform well over a range of time window widths. A secondary objective is to demonstrate the difficulties encountered by Baker's method in problems with widening time windows while solving the Traveling Salesman Makespan Problem with Time Windows.

As a first step, it is necessary to examine the performance of the Bender's Decomposition-based solution methodology using the three methods for cut generation as described in Chapter 3. Each of the problem formulation versions in Chapter 3 corresponds to a Bender's

Decomposition approach using a different set of cuts returned to the master problem. The performance of each of these three solution procedures was evaluated over a common set of test problems. The results of the evaluation were then compared to see how the three versions perform relative to one another.

The effect of gradually widening time windows on the performance of the Bender's Decomposition-based solution methodology and Baker's method was studied. The best performing method for cut generation was selected for studying the performance of the Bender's Decomposition-based solution methodology when solving the Traveling Salesman Problem with Time Windows as well as the Traveling Salesman Makespan Problem with Time Windows. Baker's method [2] is a branch-and-bound approach which is known to generate the optimal solution to the Traveling Salesman Makespan Problem with Time Windows. In this approach, the total time spent on the tour is minimized by the objective function. The same set of test problems was used to perform both studies. The problem sets were preprocessed using preprocessing techniques adopted by Langevin, *et al.*, as described in Chapter 2.

The organization of the chapter is as follows: Section 4.2 contains details about all the different problem sets used for the studies performed in this chapter. The effect of preprocessing on these problems is also discussed. Section 4.3 provides a comparative study of the performance of the Bender's Decomposition-based solution methodology for the three versions of cut generation. This study was performed over Problem Set I. The best performing version was then used to solve the Traveling Salesman Problem with Time Windows over all the problem sets. Section 4.4 provides a study of the performance of Baker's Method while solving a Traveling Salesman Makespan Problem with Time Windows version of the problems. The behavior of Baker's method over problems with widening time windows was studied. Section 4.5 provides a study of the performance of the Bender's Decomposition-based solution methodology while solving a TSMPTW version of the problems. The final section of this chapter provides observations and conclusions.

4.2 Description of the Problem Sets

This section contains a description of the various problem sets used in the studies described in this chapter. Details regarding all the problems considered in this chapter are provided in the appendices. A brief description of the problem sets is provided.

Problem Set I

The comparative study was performed on Problem Set I. The solution methodology, using the three different methods of cut generation, was applied to the set of twelve test problems. The various versions of problem SK141, i.e., SK141-1, SK141-2, and SK141-3 and problem SK142-1 are real-life problems with realistic data. The other eight problems have randomly-generated node coordinates. The procedure used to generate the time windows is explained with reference to Problem Set II. Details regarding node coordinates and time windows are provided in Appendix B.

Problem Set II

This problem set contains twenty ten-node problems. The node coordinates have been generated using a random number generator and are the same for all the problems. Details are provided in Appendix C. The problems differ in the nature of their time windows. An attempt has been made to gradually increase the width and amount of time window overlap. The steps of the procedure are as follows:

- **Step 1**: Generate a random sequence. One example is the sequence $1-2-3-\ldots-n$.
- **Step 2**: Find the total time required to pass through the sequence. Assuming the service times at every node to be 0, this is the time of travel through the sequence. Let it be T. Also, find a_i, the time of arrival at each node i.
- **Step 3**: The time window for each of the nodes is generated as follows.

$$e_i = \alpha * a_i - \alpha * T * \beta$$

$$l_i = \alpha * a_i + \alpha * T * \beta$$

where α varies between 0.7 and 1.0 and β varies between 0.025 and 0.25. The width of the time windows vary with the values of α and β selected. It can be observed that the time windows gradually widen over the problem set.

Problem Set III

This problem set contains ten problems, of which, seven have a node size of twenty while the other three have a node size of fifteen. These problems have been generated according to the procedure described by Langevin, *et al.*; the steps of the procedure have been described in Chapter 2. The internode costs/distances are Euclidean and Symmetric and were randomly generated from a uniform distribution according to the following equation:

$$c_{i,j} = [(x_i - x_j)^2 + (y_i - y_j)^2]^{1/2}$$

$$\text{with } (x_i, y_i) \sim U[0, 100]^2,\ i < j$$

All distance/cost matrices were rounded to the nearest integer. The travel time matrix was considered to be identical to the cost matrix. A second nearest-neighbor tour was found (at each step of the tour construction process, the node that was the second closest to the last node in the partial tour was selected to join the tour). This tour was then used as a starting point to generate the time windows. Let us define the second nearest neighbor tour as $(\pi_1, \pi_2, \ldots, \pi_n, \pi_1)$ with $\pi_1 = 1$ (ie., the depot). The visiting time at each node was determined as follows:

$$v_{\pi_1} = 0$$

$$v_{\pi_k} = v_{\pi_{k-1}} + t_{\pi_{k-1},\pi_k},\ k = 2, \ldots, n$$

The time windows were then generated around the visiting times at each node as follows:

$$e_{\pi_1} = 0$$

$$e_{\pi_k} = v_{\pi_k} - U[0, width],\ k = 2, \ldots, n$$

$$l_{\pi_k} = v_{\pi_k} + U[0, width],\ k = 2, \ldots, n$$

$$l_{\pi_1} = \max_{k=2,\ldots,n} l_k + t_{k,1}$$

Table 4.1: Effects of TW Reduction and Arc Elimination - Problem Set I

Problem Name	No. of nodes	Average TW width before preprocess	Average TW width after preprocess	Feasible arcs before preprocess	Feasible arcs after preprocess
R10-1	10	182.00	165.00	62	31
R10-2	10	274.00	244.00	62	32
R10-3	10	135.00	118.00	61	31
R10-4	10	219.00	205.00	71	56
R12-1	12	379.00	340.00	100	72
R12-2	12	361.00	324.00	95	64
R12-3	12	287.00	271.00	107	96
R12-4	12	195.00	185.00	98	69
SK141-1	14	319.00	306.00	180	177
SK141-2	14	290.00	277.00	163	141
SK141-3	14	305.00	292.00	167	150
SK142-1	14	215.00	210.00	176	168

where $U[0, width]$ refers to the range of the uniform distribution used for generation of the time windows. The parameter *width* refers to the width of the time window. Parameter values of 20, 30 and 40 were used for generation of the time windows. Data relating to this problem set is provided in Appendix D.

Effects of Preprocessing

The preprocessing techniques described in Chapter 2 i.e., the Langevin, *al.* method of processing was applied to all the three problem sets described herein. The effect of preprocessing on the problems is presented in Table 4.1 for Problem Set I, Table 4.2 for Problem Set II and 4.3 for Problem Set III. It was observed that the problems that were generated according to the Langevin, *et al.*, method of problem generation responded very well to time window reduction and arc elimination. The result was that the number of feasible arcs was quite small. Such dramatic reduction in the number of feasible arcs was not exhibited by

Table 4.2: Effects of TW Reduction and Arc Elimination - Problem Set II

Problem Name	No. of nodes	Average TW width before preprocess	Average TW width after preprocess	Feasible arcs before preprocess	Feasible arcs after preprocess
BAK1	10	66.00	44.00	56	16
BAK2	10	80.00	58.00	57	18
BAK3	10	133.00	109.00	58	21
BAK4	10	135.00	113.00	59	23
BAK5	10	143.00	120.00	60	25
BAK6	10	154.00	133.00	61	29
BAK7	10	165.00	144.00	62	31
BAK8	10	175.00	152.00	63	34
BAK9	10	180.00	158.00	64	37
BAK10	10	186.00	163.00	65	39
BAK11	10	191.00	167.00	66	43
BAK12	10	194.00	170.00	67	45
BAK13	10	196.00	172.00	68	48
BAK14	10	207.00	181.00	69	50
BAK15	10	209.00	183.00	70	51
BAK16	10	217.00	194.00	71	56
BAK17	10	227.00	204.00	72	61
BAK18	10	232.00	210.00	73	62
BAK19	10	237.00	215.00	74	65
BAK20	10	256.00	236.00	75	66

Table 4.3: Effects of TW Reduction and Arc Elimination - Problem Set III

Problem Name	No. of nodes	Value of TW width parameter	Average TW width after preprocess	Feasible arcs before preprocess	Feasible arcs after preprocess
C20-1-20	20	20	6.00	209	20
C20-2-20	20	20	7.00	212	25
C20-3-20	20	20	10.00	212	31
C20-1-30	20	30	10.00	209	20
C20-2-30	20	30	6.00	216	30
C20-3-30	20	30	17.00	214	34
C20-1-40	20	40	20.00	216	35
C15-1-20	15	20	8.00	120	18
C15-1-30	15	30	17.00	123	28
C15-1-40	15	40	23.00	124	29

the other two problem sets.

4.3 Comparative Study of the Three Problem Versions

Details of the comparative study are presented in this section. An algorithm describing the steps of the solution procedure is presented. The procedure adopted for identifying minimum cardinality infeasible subsequences is presented. The comparative study is performed on Problem Set I and an analysis of the results of the comparative study is presented in this section.

4.3.1 The Comparison Procedure

The algorithm for the comparison study is presented. The input information includes n (the number of nodes) and the cost matrix. Each individual cell of the cost matrix is

referred to as $c_{i,j}$ and corresponds to the cost of traveling from node i to node j. In this implementation of the Bender's Decomposition-based solution methodology, the depth-first search enumerates sequences with cost less than or equal to a predetermined upper bound. Thus, the number of sequences enumerated is reduced. A further reduction in the number of sequences enumerated was achieved by solving an Assignment Problem over the cost matrix and then performing the depth-first search over the reduced cost matrix. The upper bound is now replaced by the difference of the former upper bound and the Assignment Problem solution value. This new upper bound is called the reduced cost upper bound and is defined as

Reduced cost upper bound = (TSPTW upper bound) - (AP lower bound)

This is done to further reduce the number of sequences enumerated. The steps of the algorithm follow:

- **Step 1:** Solve an Assignment Problem over the cost matrix (using the Hungarian Method). Find the reduced cost matrix corresponding to the Assignment Problem solution by subtracting the dual values u_i and v_j from each individual cell $c_{i,j}$. Replace the original cost matrix with the reduced cost matrix. Let $c_{i,j}$ now refer to the entry in the cell corresponding to the i^{th} row and j^{th} column of the reduced cost matrix. Go to Step 2.

- **Step 2:** Using a depth-first search routine, find all possible sequences with cost of sequence being less than or equal to the reduced cost upper bound. A sequence corresponds to a combination of nodes with node 1 (depot) occuring first and all nodes occuring exactly once. Go to Step 3.

- **Step 3:** Arrange the sequences in a list that is non-descending with respect to cost of the sequence. Call the list SEQ and let row i represent the i^{th} sequence in the list. Refer to this sequence as sequence S_i. Set $l = 1$ and go to Step 4.

- **Step 4:** Treat this sequence S_l as the current solution to the master problem. This solution is used to assign appropriate values (0 or 1) to the integer variables associated with the problem. Go to Step 5.

- **Step 5:** Substitute the values of the integer variables (from the solution to the master problem) into the constraints of the subproblem. The result is a subproblem that is a

pure linear program. Go to Step 6.

- **Step 6:** Check the problem for feasibility as follows. S_l is the sequence that represents the solution to the master problem. Beginning at the depot (node 1) travel through the sequence, summing time of travel between nodes, waiting time at a node and service time at a node, until one of the two conditions listed below occurs.

  Condition A: The time of arrival at a node is after its time window has closed. Therefore, the sequence is infeasible with respect to the time windows. Go to Step 7.

  Situation B: Travel through the entire sequence S_l is accomplished without violating any time window (i.e., every node is reached before its time window has closed). Therefore, the given sequence is feasible with respect to the time windows. Go to Step 9.

- **Step 7:** Generate cuts using the appropriate method of cut generation. A detailed description of the cut generation procedure is provided at the end of this section. Add the cuts to the lists XCUTLIST and ZCUTLIST respectively, where XCUTLIST contains X-Cuts and ZCUTLIST contains Z-Cuts. The cuts in XCUTLIST and ZCUTLIST actually represent additional constraints that can be applied to the master problem. Go to Step 8.

- **Step 8:** If $l = |SEQ|$ go to Step 9; otherwise, set $l = l + 1$. Check if S_l is feasible to the master problem with additional constraints (described by XCUTLIST and ZCUTLIST). If S_l is not feasible to the master problem, go to Step 8. Otherwise, go to Step 4.

- **Step 9:** Stop. The optimal solution has been found.

The procedures adopted for generating the various cuts can be best described by the pseudocodes provided in Appendix A. A summary of the procedures is provided in this section. Detailed description of the procedures can be obtained by referring to Appendix A.

Procedure to Generate X-Cuts

The routine successively selects every node in the sequence. Starting at the node selected at the beginning of its time window, travel through succeeding nodes of the sequence, summing

the service time at a node and the time of travel between nodes. If a wait occurs at a node, then include the waiting time. Continue until some node is reached after its time window closed. Record the beginning node and the node at which the sequence became infeasible. Continue until all the nodes in the sequence have been processed. Then, for every node where the sequence becomes infeasible, find the minimal subsequence preceding it that has no inserted waiting time at any of the intermediate nodes. This is the subsequence that is used to generate the X-Cut. Therefore, for every master problem solution that is not feasible with respect to time windows, there exist one or more X-Cuts. This procedure generates all possible X-Cuts. Without loss of generality, if the minimal infeasible subsequence without inserted waits has been identified as $i, i+1, \ldots, i+m-1, i+m$, then the X-Cut is represented as

$$x_{i,i+1} + x_{i+1,i+2} + \ldots + x_{i+m-1,i+m} \leq m-1$$

All the X-Cuts thus generated are added to the list containing the X-Cuts, namely XCUT-LIST.

Procedure to Generate Z-Cuts

The procedure to generate Z-Cuts is described. The minimal infeasible subsequence with no inserted waits is detected as explained for the X-Cuts. Then, a check is made to see if the sum of the early time at the first node of the subsequence, the service time at the node, and the time of travel between the first and last nodes of the subsequence is greater than the late time of the last node. If it is, then a set of Precedence cuts, as explained in Chapter 3, is generated. If the subsequence contains only three nodes, then the possibility of generating Positional cuts is explored. Details of this procedure are also explained in Chapter 3. Thus, all possible Precedence cuts and Positional cuts are generated.

4.3.2 Results of Comparison Study

The results of the comparison study are now presented. Column three in Table 4.4 refers to the total number of possible sequences of the nodes of the problem. Column four indicates the reduced cost upper bound. Column five represents the number of sequences that were actually enumerated, i.e., the number of sequences that had costs not exceeding the previ-

Table 4.4: Comparison of Three Versions of Decomposition-Based Method

Problem Name	No. of nodes	Total number of tours	Assign-ment Problem Lower Bound	Reduced cost Upper Bound	No. of tours with cost $\leq$ UB	Optimal tour index	No. of master problems solved		
							Ver1	Ver2	Ver3
R10-1	10	362880	561	36	12	1	1	1	1
R10-2	10	362880	804	64	3	2	2	2	2
R10-3	10	362880	441	149	31	4	3	3	3
R10-4	10	362880	317	136	120	2	2	2	2
R12-1	12	3991680	587	142	88	2	2	2	2
R12-2	12	3991680	587	142	47	2	2	2	2
R12-3	12	3991680	358	123	1545	181	5	5	5
R12-4	12	3991680	368	118	159	37	6	5	5
SK141-1	14	6.23×10^9	459	40	39276	35861	794	1189	711
SK141-2	14	6.23×10^9	460	40	6956	5776	311	430	275
SK141-3	14	6.23×10^9	459	40	21879	18932	555	741	448
SK142-1	14	6.23×10^9	432	26	474	88	27	14	13

ously determined upper bound. Column six indicates how far down the list of sequences the optimal solution was found. Columns seven, eight, and nine indicate the number of master problems solved before finding the optimal solution for the three versions of the Bender's Decomposition-based solution methodology. Columns two and three in Table 4.5 indicate the number of X-Cuts generated in the first and third versions respectively. Column four indicates the number of Simple Z-Cuts generated in the second version. Columns five and six indicate the number of Precedence Z-Cuts generated in the second and third versions respectively. Solution time (in seconds) for the three versions are given in columns seven, eight, and nine.

An observation of the results presented in Tables 4.4 and 4.5 indicates that the third version solved the least number of master problems before finding the optimal solution. In other words, this version performed at least as well as or better than the other two versions. This version of the Bender's Decomposition-based solution methodology consists of a subproblem that has x, z and t variables in it, and generates both X-Cuts and Z-Cuts to be passed back to the master problem. Also, the ability of this problem formulation to represent a priori precedence constraints efficiently and generate X-cuts to eliminate infeasible subsequences efficiently make it applicative to the solution of the Traveling Salesman Problem with Time Windows. Therefore, this version is selected for use in all future studies. The details of the best performing version of the Bender's Decomposition-based solution methodology are summarized in Table 4.6. All the studies described in this section were performed on a VAX/VMS mainframe computer (VAX 7000-640). The Bender's Decomposition based-solution methodology was then used to solve the Traveling Salesman Problem with Time Windows over Problem Sets II and III. The results are presented in Tables 4.7 and 4.8 for Problem Set II and Table 4.9 for Problem Set III.

4.4 Baker's Method

4.4.1 Performance of Baker's Method

Baker's method [2] is known to generate the optimal solution for the Traveling Salesman Makespan Problem with Time Windows. As described in chapter 2, Baker's formulation of

Table 4.5: Comparison of Three Versions *cont.*

Problem Name	No. of X-Cuts		No. of simple Z-Cuts	No. of precedence Z-Cuts (sets)		Solution time in seconds		
	Ver1	Ver3	Ver2	Ver2	Ver3	Ver1	Ver2	Ver3
R10-1	0	0	–	–	–	0.13	0.34	0.38
R10-2	1	1	–	9(1)	9(1)	0.15	0.32	0.47
R10-3	2	2	2	–	–	0.17	0.47	0.48
R10-4	3	3	–	27(3)	27(3)	0.12	0.40	0.48
R12-1	2	2	–	22(2)	22(2)	0.15	0.38	0.46
R12-2	1	1	–	11(1)	11(1)	0.14	0.36	0.42
R12-3	4	4	–	44(4)	44(4)	0.36	0.66	0.58
R12-4	8	6	–	66(6)	66(6)	0.22	0.45	0.49
SK141-1	1464	1215	2805	26(2)	26(2)	94.13	591.61	110.33
SK141-2	494	432	713	117(9)	104(8)	11.74	43.02	18.58
SK141-3	968	785	1386	78(6)	78(6)	42.56	179.56	77.53
SK142-1	54	16	14	39(3)	26(2)	0.51	0.64	0.91

Table 4.6: Decomposition-Based Method (TSPTW) - Problem Set I

Problem Name	Assign-ment Problem Lower Bound	Reduced Cost Upper Bound (UB)	No. of tours with cost $\leq$UB	Time for Depth First Search	Opt. tour index	No. of master prob-lems solved	Time for opt-imal search	Total solu-tion time	Opt. cost of tour	Time with wait on tour
R10-1	561	36	12	0.10	1	1	0.38	0.48	574	622
R10-2	804	64	3	0.06	2	2	0.47	0.53	868	871
R10-3	441	149	31	0.12	4	3	0.48	0.60	521	521
R10-4	317	136	120	0.38	2	2	0.48	0.86	395	526
R12-1	587	142	88	0.82	2	2	0.46	1.28	661	966
R12-2	587	142	47	0.60	2	2	0.42	1.02	673	976
R12-3	358	123	1545	7.88	181	5	0.58	8.46	445	573
R12-4	368	118	159	1.55	37	5	0.49	2.08	445	697
SK141-1	459	40	39276	784.29	35861	711	110.33	894.62	499	526
SK141-2	460	39	6956	131.10	5776	275	18.58	149.68	499	526
SK141-3	459	40	21879	358.70	18932	448	77.53	436.23	499	526
SK142-1	432	26	474	17.80	88	13	0.91	18.71	453	537

Table 4.7: Decomposition-Based Method (TSPTW) - Problem Set II

Problem Name	Assignment Problem Lower Bound	Reduced cost Upper Bound (UB)	No. of tours with cost $\leq$UB	Time for Depth First Search	Opt. tour index	No. of master problems solved	Time for optimal search	Total solution time	Opt. cost of tour	Time with wait on tour
BAK1	596	0	1	0.08	1	1	0.48	0.56	596	626
BAK2	596	4	2	0.09	1	1	0.45	0.54	596	619
BAK3	583	17	2	0.11	1	1	0.49	0.60	596	596
BAK4	583	17	3	0.07	1	1	0.46	0.53	596	596
BAK5	561	39	3	0.09	1	1	0.46	0.55	596	596
BAK6	561	13	1	0.10	1	1	0.45	0.55	574	606
BAK7	561	13	1	0.09	1	1	0.40	0.49	574	600
BAK8	543	31	3	0.08	3	2	0.45	0.53	574	595
BAK9	543	31	4	0.09	4	2	0.44	0.53	574	592
BAK10	543	31	6	0.10	6	2	0.46	0.56	574	589

Table 4.8: Decomposition-Based Method (TSPTW) - Problem Set II *cont.*

Problem Name	Assignment Problem Lower Bound	Reduced cost Upper Bound (UB)	No. of tours with cost $\leq$UB	Time for Depth First Search	Opt. tour index	No. of master problems solved	Time for optimal search	Total solution time	Opt. cost of tour	Time with wait on tour
BAK11	427	147	38	0.21	11	2	0.51	0.72	518	546
BAK12	427	147	51	0.22	9	2	0.45	0.67	511	538
BAK13	427	147	133	0.37	29	4	0.48	0.85	511	536
BAK14	427	47	6	0.10	1	1	0.41	0.51	465	558
BAK15	367	107	47	0.24	30	6	0.53	0.77	465	557
BAK16	367	75	24	0.24	17	4	0.49	0.73	439	552
BAK17	340	134	307	0.68	70	7	0.52	1.20	439	547
BAK18	340	99	71	0.43	70	8	0.58	1.01	439	544
BAK19	338	101	119	0.50	118	9	0.69	1.19	439	541
BAK20	338	101	120	0.51	23	4	0.55	1.06	409	500

Table 4.9: Decomposition-Based Method (TSPTW) - Problem Set III

Problem Name	Assign-ment Problem Lower Bound	Reduced cost Upper Bound (UB)	No. of tours with cost $\leq$UB	Time for Depth First Search	Opt. tour index	No. of master prob-lems solved	Time for opt-imal search	Total solu-tion time	Opt. cost of tour	Time with wait on tour
C20-1-20	820	0	1	0.07	1	1	0.45	0.52	820	820
C20-2-20	809	5	2	0.07	2	2	0.44	0.51	814	814
C20-3-20	698	0	1	0.08	1	1	0.46	0.54	698	725
C20-1-30	820	0	1	0.10	1	1	0.46	0.56	820	820
C20-2-30	809	0	2	0.13	1	1	0.46	0.59	809	812
C20-3-30	698	14	4	0.24	1	1	0.46	0.70	698	724
C20-1-40	790	30	10	0.42	3	2	0.47	0.89	803	818
C15-1-20	702	0	1	0.09	1	1	0.41	0.50	702	702
C15-1-30	666	13	5	0.11	1	1	0.49	0.60	666	706
C15-1-40	666	13	5	0.10	1	1	0.44	0.54	666	703

the Traveling Salesman Makespan Problem with Time Windows lends itself to a branch-and-bound procedure with the procedure being initiated by relaxing the absolute value constraints and the time window constraints. The dual to this relaxed problem is the longest path problem and at every stage of the branch-and-bound procedure, one such longest path problem is solved. The absolute value constraints correspond to reversible arcs, i.e., arcs along which both directions of travel are permitted. Branching takes place along the two directions of the reversible arc. At each node of the branch-and-bound tree, a longest path problem is solved using the routine L-PATH. Details about this routine are provided by the pseudocode in Appendix A. A summary of the procedure is provided in this section.

To begin, all arc costs are multiplied by -1. The result is a shortest path problem with negative arc costs. A shortest path problem is solved over the resultant cost matrix using the primal algorithm described in [71]. This algorithm does not require the arc costs to be non-negative. In its simplest form, this algorithm maintains a spanning tree in the network, which is an out-tree rooted at the origin (node 1). Each step in the algorithm changes the out-tree by just one arc. The algorithm terminates after a finite number of steps following either

1. the detection of a negative length circuit in the network, or
2. the detection of an out-tree; in this out-tree, the node price vector satisfies the condition $\pi_j - \pi_i \leq c_{i,j}, \forall (i,j) \in A$, where A represents the arcs of the network, (i.e., the arcs in A-LIST).

The algorithm is described below.

Initialization: For all $j \in N$, $j \neq 1$, where N represents the nodes of the network, if there is no arc from 1 to j, introduce an artificial arc $(1,j)$ and make cost of the arc $c_{1,j} = \infty$. Let T^0 be the initial spanning out-tree rooted at node 1, consisting of arcs $(1,j)$, $j \in N$, $j \neq 1$. Label node 1 with $(\phi, 0)$ and node j with $(1, c_{1,j})$. In this algorithm, the label on node i is always of the form $(P(i), d_i)$ where:

- $P(i)$ = predecessor index if node i,
- d_i = length of chain from 1 to i in the present spanning out-tree.

General Step: For $i \in N$, suppose the label on i is $(P(i), \pi_i^r)$ at the end of the previous step (r is the step number). Look for an arc $(i,j) \in A$ satisfying:

$$\pi_j \quad > \quad \pi_i + c_{i,j}. \tag{4.1}$$

If there is no arc (i,j) satisfying condition (4.1), ie., if $\pi_j \leq \pi_i + c_{i,j}$ for all $(i,j) \in A$, the present out-tree is a shortest chain tree rooted at node 1 in the original network. If the out-tree contains some artificial arcs, eliminate them. What is left is an out-tree which is not spanning. However, it is the shortest chain tree in the nodes it spans and there exist no chains from node 1 to nodes outside the out-tree. Terminate the algorithm.

If an arc $(i,j) \in A$ satisfying condition (4.1) is found, two cases can occur.

- **Case 1**: Node j occurs on the current predecessor path of node i.
  Checking to see if this condition occurs is referred to as the *ancestor checking operation.* If node j is indeed an ancestor of node i, then a negative length circuit has been detected. Therefore, the algorithm terminates.

- **Case 2**: Node j is not on the predecessor path of node i.
  Let **D** denote the set of descendents of node j in the present out-tree. The present in-tree arc $(P(j), j)$ is replaced by the arc (i,j). This is achieved by changing the label on node j from the present $(P(i), \pi_j^r)$ to $(i, \pi_i^r + c_{i,j})$. Then, for each node $t \in D$, change its distance from π_t^r to $\pi_t^r + \pi_i^r + c_{i,j} - \pi_j^r$; do not change its predecessor index. This is the operation termed *correcting the distance index* on each descendent of node j. Continue.

In Baker's branch-and-bound scheme, the arc costs $c_{i,j}$ are represented as follows:

$c_{i,j}$ = −service time at i − travel time between nodes i and j.

When a negative length circuit is detected in the network, the current vertex in the branch-and-bound scheme is fathomed. Further, if the algorithm determines π_i for node i such that $-\pi_i$ is greater than the close of the time window, l_i, then the current problem in the enumeration is deemed infeasible. If $-\pi_i$ is less than the beginning of the time window e_i, then π_i is set equal to $-e_i$ and the procedure continues. The arc costs in the network must satisfy the triangle inequality and a maximum cardinality path labeling convention must be adopted to resolve ties. This ensures that the longest path solution to the problem contains n

arcs, where n refers to the number of nodes in the network. Table 4.10 presents information regarding the number of reversible arcs, the time required to solve the problem, the number of branch-and-bound nodes enumerated, and the lower bound at the root node for each one of the test problems of Problem Set II. As can be observed, the number of reversible arcs gradually increases with an increase in the width of the time windows. The solution time, as well as the number of branch-and-bound nodes enumerated, also increase with an increase in the number of reversible arcs, i.e., with an increase in width of time windows. Figure 4.1 demonstrates this behavior of Baker's method. It is observed that the solution time as well as the number of branch-and-bound nodes enumerated is quite low when the number of reversible arcs is small, i.e., when the time windows are not very wide. The narrowness of the time windows effectively removes certain directions of travel (along arcs) from consideration. The direction of travel is fixed along certain arcs thus reducing the number of arcs along which both directions of travel are allowed, i.e., reversible arcs. The value of the lower bound at the root node is quite close to the optimal solution. However, once the time windows begin to widen and overlap to a greater degree, the number of reversible arcs increases and the difference between the value of the lower bound at the root node and the optimal solution also increases. The time needed for solution of the problem increases rapidly. The number of branch-and-bound nodes enumerated also increases rapidly. This indicates that although Baker's method performs well in the case of problems with narrow time windows, the time needed for solution increases rapidly with an increase in the width of the time windows.

Baker's method was next applied to Problem Set I. Every problem solution procedure was initiated with a valid upper bound solution as the incumbent solution. For the first eight problems of this problem set, the upper bound solution is found by using an insertion heuristic. The details of the insertion heuristic are provided in Chapter 5. For the last four problems, a previously known feasible solution was used as the incumbent solution. The results are available in table 4.11. It is observed that Baker's method was unable to find the optimal solution for the problems marked with an asterisk (*). The value presented here is the incumbent value.

The last four problems of Problem Set I have wide time windows with a significant amount of overlap. Therefore, the number of reversible arcs is large and a significantly large number

Table 4.10: Performance of Baker's Method (TSMPTW) - Problem Set II

Problem Name	Number of rev. arcs	B&B nodes enumerated	LB at root node	Opt. time with wait	Cost of tour	Solution time (secs)
BAK1	2	5	602	612	600	0.04
BAK2	3	12	595	605	600	0.02
BAK3	4	13	588	596	596	0.04
BAK4	5	29	588	596	596	0.06
BAK5	6	58	588	596	596	0.09
BAK6	7	105	543	596	596	0.23
BAK7	8	204	537	596	596	0.37
BAK8	9	285	532	595	578	0.54
BAK9	10	716	507	592	583	1.59
BAK10	11	1028	504	589	583	2.37
BAK11	12	1481	502	546	531	3.84
BAK12	13	2828	500	538	511	7.79
BAK13	14	4910	499	536	524	14.46
BAK14	15	6153	493	531	524	18.39
BAK15	16	8173	492	530	524	28.83
BAK16	17	10761	488	520	519	34.72
BAK17	18	20759	482	515	506	79.07
BAK18	19	31148	479	512	506	119.51
BAK19	20	43991	477	509	506	174.56
BAK20	21	63776	466	500	464	257.52

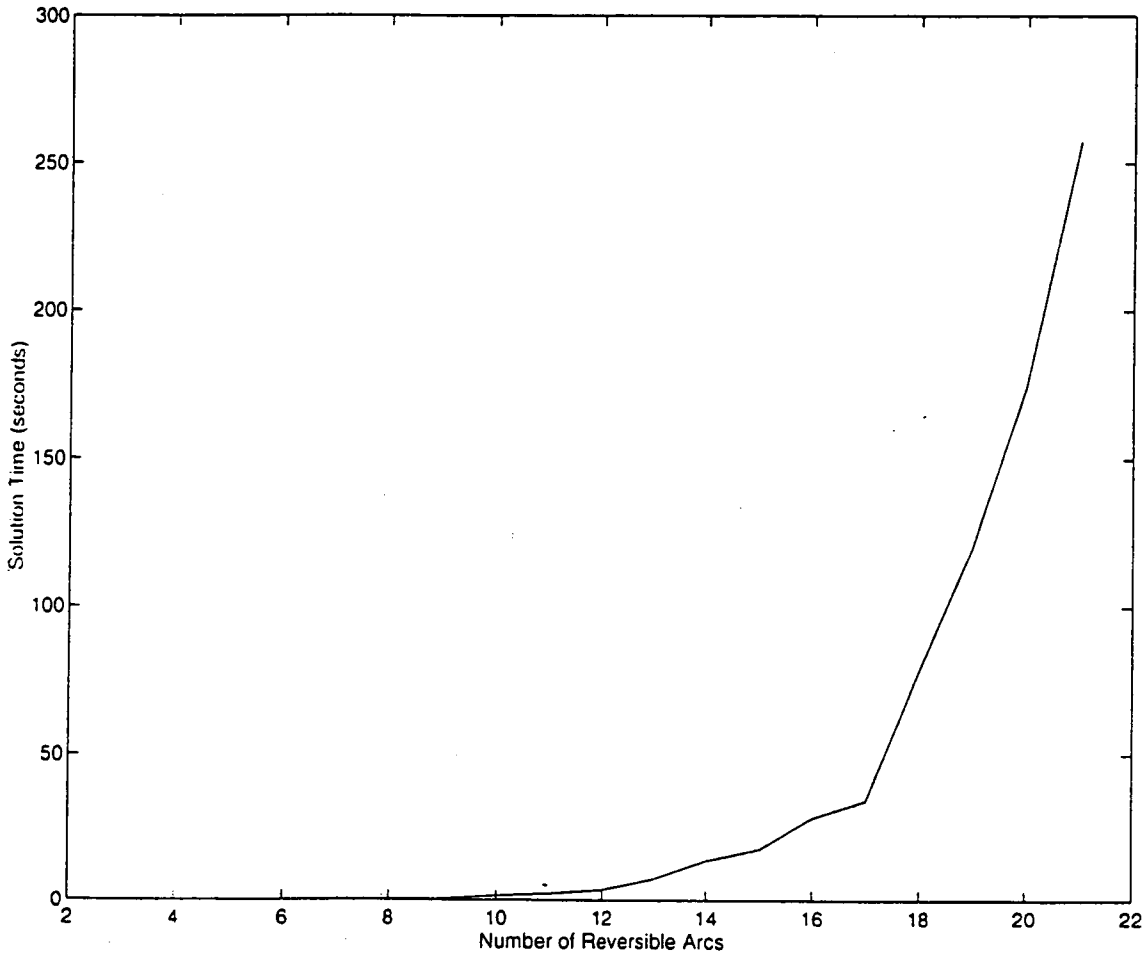

Figure 4.1: Solution Time *vs* Number of Reversible Arcs – Baker's Method

Table 4.11: Performance of Baker's Method (TSMPTW) - Problem Set I

Problem Name	No. of. reversible arcs	Incumbent solution value	B & B nodes enumerated	LB at root node	Opt. time with wait	Cost of tour	Solution time (secs)
R10-1	8	641	102	568	600	600	0.36
R10-2	8	871	47	833	871	868	0.22
R10-3	7	590	122	487	521	521	0.47
R10-4	17	595	4483	497	504	444	25.43
R12-1	23	966	224329	914	928	928	1762.64
R12-2	18	976	21895	924	936	936	165.99
R12-3	30	573	54277	569	569	561	540.52
R12-4	21	697	38767	585	592	583	284.51
SK141-1	76	518	> limit	479	518*	503*	15979.28
SK141-2	59	518	> limit	479	518*	503*	13693.34
SK141-3	63	518	> limit	479	518*	503*	8480.83
SK142-1	72	510	> limit	457	510*	484*	2415.67

of branch-and-bound nodes must be enumerated before the optimal solution can be found. For the last four problems of this set, Baker's method was unable to find the optimal solution within the preset limit of 400,000 branch-and-bound nodes.

Finally, Baker's method was tested on Problem Set III. This set contains problems that were generated according to the procedure adopted by Langevin, *et al.* These problems respond very well to the preprocessing techniques suggested by Langevin *et al.* The results of this study are presented in Table 4.12. All the studies described in this section were performed on a VAX/VMS minicomputer (VAXstation 4000-90).

It was observed that Baker's Method performed well on Problem Set III. The nature of the problems in this set, as well as the preprocessing techniques adopted, are such that a large number of arcs are rendered infeasible. The value of the lower bound at the root node is either equal to or very close to the optimal solution value. Therefore, very few branch-and-bound nodes were enumerated before reaching the optimal solution. Exact details about the number of arcs rendered infeasible are provided in section 4.2 of this chapter.

4.5 Bender's Decomposition-Based Method on the TSMPTW

The best performing version of the Bender's Decomposition-based solution methodology was selected to solve the Traveling Salesman Makespan Problem with Time Windows over the three problem sets described earlier. The results of this study are presented in Table 4.13 and Table 4.14 for Problem Set 1, Table 4.15, Table 4.16, Table 4.17 and, Table 4.18 for Problem Set II, and Table 4.19 and Table 4.20 for Problem Set III. All the studies described in this section were performed on a VAX/VMS minicomputer (VAXstation 4000-90). The time needed to solve the problems does not increase rapidly with increase in time window width and overlap. If there were no preprocessing of any kind, then the decomposition-based method would favor problems with wide time windows, since a lower cost sequence would be optimal. In our enumerative approach, a lower cost sequence would be found earlier in the list of sequences. A problem with tight time windows would have a higher cost sequence as the optimal solution and hence, in our enumerative approach, be found later in the list of sequences. However, preprocessing helps to prune the list such that

Table 4.12: Performance of Baker's Method (TSMPTW) - Problem Set III

Problem Name	No. of reversible arcs	B & B nodes enumerated	LB at root node	Opt. time with wait	Cost of tour	Solution time (secs)
C20-1-20	0	1	820	820	820	0.03
C20-2-20	3	5	813	814	814	0.09
C20-3-20	3	7	725	725	710	0.12
C20-1-30	0	1	820	820	820	0.06
C20-2-30	6	40	812	812	809	0.25
C20-3-30	5	11	724	724	710	0.13
C20-1-40	7	64	818	818	803	0.44
C15-1-20	1	1	700	702	702	0.04
C15-1-30	4	13	698	698	679	0.07
C15-1-40	5	22	697	697	679	0.12

Table 4.13: Decomposition-Based Method (TSMPTW) - Problem Set I - 1

Problem Name	Reduced cost Upper Bound (UB)	No. of tours with cost $\leq$ UB	Time for Depth First Search	First feasible tour index	Optimal tour index	Time for optimal search	Total solution time (secs)
R10-1	80	42	0.11	1	15	0.52	0.63
R10-2	71	7	0.08	2	2	0.41	0.49
R10-3	149	31	0.12	4	4	0.39	0.51
R10-4	278	2050	2.18	2	79	2.50	4.68
R12-1	379	4964	13.72	2	3194	20.18	33.90
R12-2	389	1706	4.80	2	1221	10.57	15.37
R12-3	215	40818	240.08	181	23321	195.39	435.47
R12-4	329	12006	36.28	37	716	25.30	61.58
SK141-1	59	–	–	–	–	–	–
SK141-2	58	–	–	–	–	–	–
SK141-3	59	–	–	–	–	–	–
SK142-1	78	–	–	–	–	–	–

Table 4.14: Decomposition-Based Method (TSMPTW) - Problem Set I - 2

Problem Name	No. of master problems solved	No. of X-Cuts	No. of precedence Z-Cuts	Optimal time on tour with wait	Cost of tour
R10-1	11	10	0	600	600
R10-2	5	4	9(1)	871	868
R10-3	3	2	0	521	521
R10-4	115	117	90(10)	504	444
R12-1	663	664	132(12)	928	928
R12-2	531	530	88(8)	936	936
R12-3	2952	2955	123(13)	569	553
R12-4	708	710	176(16)	592	519
SK141-1	–	–	–	–	–
SK141-2	–	–	–	–	–
SK141-3	–	–	–	–	–
SK142-1	–	–	–	–	–

Table 4.15: Decomposition-Based Method (TSMPTW) - Problem Set II - 1-1

Problem Name	Reduced cost Upper Bound (UB)	No. of tours with cost $\leq$ UB	Time for Depth First Search	First feasible tour index	Optimal tour index	Time for optimal search	Total solution time (secs)
BAK1	100	4	0.06	1	2	0.36	0.42
BAK2	100	6	0.09	1	2	0.49	0.58
BAK3	17	2	0.09	1	1	0.45	0.54
BAK4	17	3	0.08	1	1	0.44	0.52
BAK5	39	3	0.12	1	1	0.47	0.59
BAK6	45	15	0.18	1	6	0.48	0.66
BAK7	39	15	0.17	1	10	0.53	0.70
BAK8	52	17	0.19	3	3	0.55	0.74
BAK9	49	18	0.17	4	4	0.59	0.76
BAK10	49	27	0.16	6	6	0.61	0.77

Table 4.16: Decomposition-Based Method (TSMPTW) - Problem Set II - 1-2

Problem Name	No. of master problems solved	No. of X-Cuts	No. of precedence Z-Cuts	Optimal time on tour with wait	Cost of tour
BAK1	3	3	0	612	600
BAK2	3	3	0	605	600
BAK3	1	0	0	596	596
BAK4	1	0	0	596	596
BAK5	1	0	0	596	596
BAK6	6	5	0	596	596
BAK7	7	6	0	596	596
BAK8	9	9	18(2)	595	574
BAK9	9	8	18(2)	592	574
BAK10	11	10	18(2)	589	574

Table 4.17: Decomposition-Based Method (TSMPTW) - Problem Set II - 2-1

Problem Name	Reduced cost Upper Bound (UB)	No. of tours with cost $\leq$ UB	Time for Depth First Search	First feasible tour index	Optimal tour index	Time for optimal search	Total solution time (secs)
BAK11	159	63	0.30	11	11	0.52	0.82
BAK12	158	81	0.40	9	9	0.61	1.01
BAK13	157	172	0.53	29	29	0.65	1.18
BAK14	131	155	0.42	1	44	1.04	1.46
BAK15	190	291	0.71	30	123	1.21	1.92
BAK16	188	614	1.20	17	267	1.48	2.68
BAK17	207	1304	2.24	70	653	2.18	4.42
BAK18	204	1534	2.58	70	719	2.24	4.82
BAK19	203	2123	3.55	118	1031	2.55	6.10
BAK20	192	2004	3.54	23	23	4.00	7.54

Table 4.18: Decomposition-Based Method (TSMPTW) - Problem Set II - 2-2

Problem Name	No. of master problems solved	No. of X-Cuts	No. of precedence Z-Cuts	Optimal time on tour with wait	Cost of tour
BAK11	6	6	18(2)	546	518
BAK12	9	9	18(2)	538	511
BAK13	11	11	9(1)	536	511
BAK14	46	46	27(3)	531	511
BAK15	58	57	54(6)	530	511
BAK16	65	66	36(4)	520	506
BAK17	91	92	45(5)	515	506
BAK18	85	86	63(7)	512	506
BAK19	98	100	63(7)	509	506
BAK20	206	206	72(8)	500	409

Table 4.19: Decomposition-Based Method (TSMPTW) - Problem Set III - 1

Problem Name	Reduced cost Upper Bound (UB)	No. of tours with cost $\leq$ UB	Time for Depth First Search	First feasible tour index	Optimal tour index	Time for optimal search	Total solution time (secs)
C20-1-20	0	1	0.16	1	1	0.45	0.61
C20-2-20	5	2	0.13	2	2	0.48	0.61
C20-3-20	27	6	0.30	1	1	0.53	0.83
C20-1-30	0	1	0.15	1	1	0.46	0.61
C20-2-30	10	3	0.12	1	1	0.49	0.61
C20-3-30	34	11	0.47	1	1	0.58	1.05
C20-1-40	30	10	0.50	3	3	0.52	1.02
C15-1-20	0	1	0.09	1	1	0.47	0.56
C15-1-30	36	11	0.18	1	4	0.56	0.74
C15-1-40	35	13	0.19	1	4	0.60	0.79

Table 4.20: Decomposition-Based Method (TSMPTW) - Problem Set III - 2

Problem Name	No. of master problems solved	No. of X-Cuts	No. of precedence Z-Cuts	Optimal time on tour with wait	Cost of tour
C20-1-20	1	0	0	820	820
C20-2-20	2	1	0	814	814
C20-3-20	5	5	0	725	698
C20-1-30	1	0	0	820	820
C20-2-30	3	2	0	812	809
C20-3-30	6	5	0	724	698
C20-1-40	6	5	0	818	803
C15-1-20	1	0	0	702	702
C15-1-30	9	9	0	698	679
C15-1-40	11	11	28(2)	697	679

many lower cost sequences that are infeasible would be effectively removed from the list of sequences. Therefore, the decomposition-based approach works well for problems with tight time windows also. The solution time depends on the number of master problems solved before reaching the optimal solution.

Our enumerative implementation of the decomposition-based method requires that the upper bound on sequence cost be set such that the optimal solution is not eliminated from consideration. For the TSMPTW version of the decomposition-based method, the first feasible solution is not the optimal solution. In order to ensure that the optimal solution is not eliminated from the list of sequences, the reduced cost upper bound is selected as follows:

Reduced cost upper bound = (TSMPTW upper bound) - (AP lower bound)

The reduced cost upper bound in this case is greater than or equal to the reduced cost upper bound for the TSPTW. In the case of problems SK141-1, SK141-2, SK141-3 and SK142-1, the reduced cost upper bound was so high that all the sequences with reduced cost less than or equal to this upper bound could not be enumerated. Therefore, these problems could not be solved by our enumerative implementation of the decomposition-based method.

4.6 Conclusions

The major conclusions that can be drawn based on the results presented in this chapter are listed below:

1. The third version of problem formulation, i.e. the Combined Representation solves the least number of master problems and hence is considered to be the best version of solution by the Bender's Decomposition-based solution methodology.

2. Baker's Method exhibits a rapid increase in solution time with an increase in width of time windows. This further indicates that while Baker's Method works well on problems with narrow time windows, it is not suitable for solution of problems with wide time windows with reasonably large overlap. Baker's method could not solve problems SK141-1, SK141-2, SK141-3, and SK142-1.

3. The Bender's Decomposition-based solution methodology works well on problems with narrow time windows as well as problems with wide and overlapping time windows for the TSPTW. With respect to the TSMPTW, the number of sequences with reduced cost less than or equal to the predefined reduced cost upper bound were too many to be enumerated for problems SK14-1, SK141-2, SK141-3, and SK152-1. Therefore, we do not have any results for these problems. Of course, the enumerative method is a viable proposition only if there exists a tight upper bound that greatly limits the number of sequences being enumerated.

4. Problems with tight time windows respond well to preprocessing, thereby reducing the number of feasible arcs in the network. Therefore, the problem can be solved easily by any of the methods discussed. This has been demonstrated by the ease with which all the methods perform when solving the problems of Problem Set III.

Chapter 5

Development of Heuristic

5.1 Objectives

The main objective of this chapter is to present a heuristic for the Traveling Salesman Problem with Time Windows. This heuristic is based on the Bender's Decomposition-based solution methodology described in Chapter 3. The method of cut generation employed is the one described in the combined variable representation of the TSPTW.

The heuristic for the Traveling Salesman Problem with Time Windows is tested on a set of test problems with a range of time window widths. The heuristic is first described in general terms. This is followed by specific details of the heuristic. The performance of the heuristic is compared with the performance of an insertion heuristic. Details of the insertion heuristic are also provided in this chapter. Finally, the effect of the Langevin, *et al.*, type of preprocessing on this problem set is studied.

The heuristic is described in general terms in Section 5.2. Section 5.3 presents specific details of the heuristic. Section 5.4 provides details of the insertion heuristic used. Section 5.5 presents the results of a computational study using the proposed heuristic as well as the insertion heuristic. The effects of time window reduction and arc elimination on the test problem set are also studied. Section 5.5 presents some observations.

5.2 General Description of the Heuristic

The heuristic is based on the Bender's Decomposition-based approach to solving the Traveling Salesman Problem with Time Windows. The combined variable representation of the TSPTW (that has x, z and t variables in the subproblem) is used in the development of the heuristic. The master problem in this representation is a Hamiltonian Path problem with additional constraints. A feasible solution to the master problem represents a Traveling Salesman tour since a visit to node $n+1$ indicates a return to the depot. The structure of the master problem allows us to represent a feasible Hamiltonian Path (Traveling Salesman tour) in two ways:

1. as a sequence of nodes to be visited (using arc variables), and
2. as a list of the positions in the sequence occupied by each node (using positional variables).

The subproblem merely checks to see if the solution to the master problem is feasible with respect to the time windows. If the solution to the master problem is not feasible with respect to the time windows, then cuts that prevent certain infeasible subsequences from occurring are generated and passed back to the master problem. As described in Chapter 3, the different sequence representations result in the generation different kinds of cuts (returned to the master problem). The new master problem is again a Traveling Salesman Problem with additional constraints and is solved using suitable methods of solution. The process continues until a sequence that is feasible with respect to the time windows is found. This is the solution to the Traveling Salesman Problem with Time Windows.

The details of the procedure are now explained. First, some relevant terms are defined.

TOURLIST – TOURLIST is a list of tours (that are feasible solutions to the master problem). T_i refers to the i^{th} tour (sequence) in the list. The cost of the sequence is also stored in this list.

XCUTLIST – XCUTLIST is the list of X-Cuts, i.e., cuts corresponding to the arc variable representation of the sequence.

X_i refers to the i^{th} cut in this list.

ZCUTLIST – ZCUTLIST is the list of Z-Cuts, i.e., cuts corresponding to the positional variable representation of the sequence. Z_i refers to the i^{th} cut in this list.

COSTMAT – COSTMAT represents the cost matrix. Each individual entry $c_{i,j}$ refers to the cost of traveling from node i to node j.

Let us consider a problem of n nodes where node 1 is the depot. The heuristic is initiated by solving a Traveling Salesman Problem over the cost matrix COSTMAT. The result is a tour that can be represented as a Hamiltonian Path by denoting a return to the depot as a visit to node $n+1$. This sequence is treated as the first entry in the list TOURLIST. The subproblem now checks to see if this tour is feasible with respect to the time windows. If the sequence is indeed feasible with respect to the time windows then the TSPTW has been solved. Otherwise, various cuts are generated and passed back to the master problem. These cuts are stored in the lists XCUTLIST (for X-Cuts) and ZCUTLIST (for Z-Cuts).

The sequence with minimum cost is selected from the list TOURLIST. Let this sequence be referred to as T_{min}. This sequence represents a Hamiltonian Path. In order for it to be a feasible solution to the master problem, the sequence should satisfy all the constraints represented by the X-Cuts and Z-Cuts. Therefore, a check is performed to see if any of the X-Cuts (in XCUTLIST) or Z-Cuts (in ZCUTLIST) are violated by the sequence T_{min}. The following example is used to better explain the procedure:

Without loss of generality, define the sequence T_{min} as $1, \ldots, i, i+1, \ldots, i+m, \ldots, n+1$. Also, consider the X-Cut X_m which is of the form:

$$x_{i,i+1} + x_{i+1,i+2} + \ldots + x_{i+m-1,i+m} \leq m-1$$

It follows from T_{min} that $x_{i,i+1} + x_{i+1,i+2} + \ldots + x_{i+m-1,i+m} = m$ and thus the cut X_m is violated. Therefore, T_{min} is not a feasible solution to the master problem. In order to generate a feasible solution to the master problem, it is essential to ensure that $x_{i,i+1} + x_{i+1,i+2} + \ldots + x_{i+m-1,i+m} \leq m-1$, i.e., at least one of $x_{i,i+1}$, $x_{i+1,i+2}$, $\ldots$, $x_{i+m-1,i+m}$ must be equal to zero. The heuristic proceeds by successively setting each of the costs

$c_{i,i+1}$, $c_{i+1,i+2}$, ..., $c_{i+m-1,i+m}$ equal to a large value and solving a Traveling Salesman Problem over the resultant cost matrix. Suppose, the cost $c_{i,i+1}$ is set equal to infinity. Then, solution of the Traveling Salesman Problem over the resultant cost matrix will result in a Hamiltonian sequence that does not contain the arc $(i, i+1)$. Therefore, the constraint $x_{i,i+1} + x_{i+1,i+2} + \ldots + x_{i+m-1,i+m}$ is satisfied. This process is carried out for every arc of the subsequence prevented by the X-cut X_m. Figure 5.1 helps explain this concept better. Let $1, 2, 3, 4, 5, 6$ be the current minimum cost sequence for a five node problem (node 6 indicates a return to the depot). Suppose this sequence violates the X-Cut $x_{2,3} + x_{3,4} \leq 1$. Therefore, we successively set arc costs $c_{2,3} = \infty$ and $c_{2,4} = \infty$ and solve a TSP over the resultant cost matrix and add them to the list TOURLIST. Let the two new sequences added be $1, 2, 4, 3, 5, 6$ and $1, 3, 2, 4, 5, 6$ respectively. Suppose, the next minimum cost sequence selected is $1, 2, 4, 3, 5, 6$ and let this sequence violate the X-Cut $x_{2,4} + x_{4,3} + x_{3,5} \leq 2$. Therefore, three new sequences are generated. Note that arc $(2, 3)$ still has a cost of infinity in each of these three new subproblems. The three new sequences are added to TOURLIST and the procedure continues until a sequence that does not violate any of the X-Cuts is found. A record of the arcs with infinite cost in the cost matrix has to be kept for every sequence in TOURLIST. For example, a record must be kept stating that the cost matrix that generated sequence $1, 4, 3, 2, 5, 6$ had $c_{2,3} = \infty$ and $c_{2,4} = \infty$.

Next, consider the Z-Cut Z_l. The Z-Cuts essentially represent precedence relationships. Without loss of generality, assume that Z_l specifies that $i + m$ must precede i in any feasible solution to the master problem, and further assume that T_{min} represents the sequence $1, \ldots, i, i+1, \ldots, i+m, \ldots, n+1$. Therefore, this sequence violates cut Z_l. The infeasible subsequence in this case can be represented as $i, i+1, \ldots, i+m$. As with the X-Cut, the heuristic proceeds by successively assigning costs $c_{i,i+1} = \infty$, $c_{i+1,i+2} = \infty$, ..., $c_{i+m-1,i+m} = \infty$ and solving a Traveling Salesman Problem over the resultant cost matrix yielding a Hamiltonian sequence that does not contain the infeasible subsequence $i, i+1, \ldots, i+m$. The precedence Z-Cuts are essentially converted into X-Cuts that correspond to the infeasible subsequence and the procedure described previously is adopted to generate new sequences.

The sequence T_{min} is checked against The X-Cuts and Z-Cuts until it is found to violate a cut. New sequences (corresponding to the violated cut) are generated and added to the list TOURLIST. If no cut is violated, then a feasible solution to the master problem has been

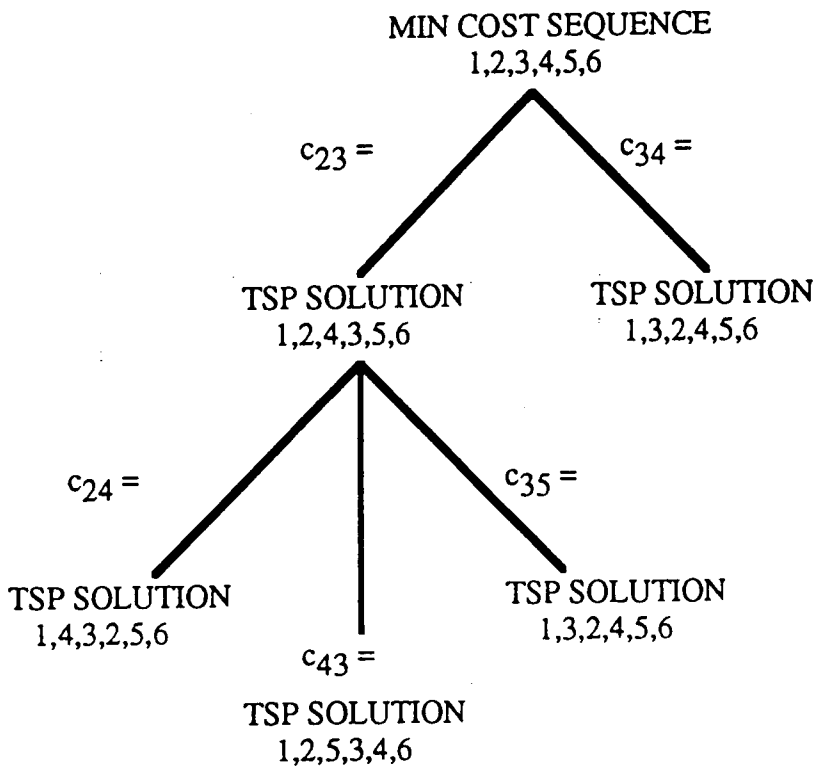

Figure 5.1: Example for sequence generation

found. Otherwise, the current T_{min} is removed from TOURLIST. A new minimum cost sequence T_{min} is selected and the procedure continues as before until a feasible solution to the master problem is found.

The feasible solution to the master problem is checked for feasibility with respect to the time windows. If the sequence is indeed feasible with respect to the time windows, then the Traveling Salesman Problem with Time Windows has been solved. Otherwise, the X-Cuts and Z-Cuts are generated as explained in chapters 3 and 4, added to the lists XCUTLIST and ZCUTLIST respectively, and the procedure continues. A variation of Lin-Kernighan's $3-opt$ method suitable for asymmetric TSPs was used to solve the master problem every time.

It is advantageous to account for the time spent waiting for a time window to open in the cost of traveling from node i to node j. This compensation can be achieved as described below.

Consider two nodes, i and j. Let $w_{i,j}$ represent the average waiting time at node j (if the salesman traveled from node i to node j). An initial measure of $w_{i,j}$, would be:

$$w_{i,j} = \frac{(e_j - a_1) + (e_j - a_2)}{2}.$$

The quantities a_1 and a_2 can be defined as:

$$a_1 = \max(e_i, d_{1,i}) + s_i + d_{i,j},$$

$$a_2 = l_i + s_i + d_{i,j}.$$

The cost of waiting can then be accounted for by redefining the cost of traveling from node i to node j (nodes i and $j \neq 1$) as:

$$c_{i,j}^{new} = c_{i,j}^{old} + \lambda(w_{i,j}),\ \lambda \geq 0.$$

For the first leg of the journey, the new cost can be defined as:

$$c_{1,j}^{new} = c_{1,j}^{old} + \lambda(e_j - d_{1,j}),\ \lambda \geq 0.$$

The quantity $c_{i,j}^{new}$ refers to the cost of traveling from node i to node j and waiting at node j for the time window to open.

5.3 Specific Details of Heuristic

Specific details of the heuristic based on the Bender's Decomposition-based solution methodology for the Traveling Salesman Problem with Time Windows can now be stated.

- **Step 0:** This step is the preprocessing step. The cost matrix is preprocessed according to the procedure proposed by Langevin, *et al.*, [4]. The details of this preprocessing procedure have been explained in Chapter 2. Also, initialization of the list containing feasible tours/sequences (TOURLIST) and the lists containing X-Cuts (XCUTLIST) and Z-Cuts (ZCUTLIST) is performed. Go to Step 1.

- **Step 1:** Compensate for waiting at nodes for the time windows to open by recomputing the cost matrix as described in the previous section. Go to Step 2.

- **Step 2:** The master problem is solved using a TSP solver. The TSP solver used is a variation of the implementation of Lin-Kernighan's 3-opt procedure (by William Stewart) suitable for asymmetric TSPs. The result is a Traveling Salesman tour. Go to Step 3.

- **Step 3:** Add the above determined tour, along with the cost of the tour, to the list of feasible tours, i.e., TOURLIST. Also, record the arcs that have infinite cost in a corresponding list called COSTLIST. Go to Step 4.

- **Step 4:** Pick the minimum cost sequence. Call it T_{min}. Convert the TSP tour into a Hamiltonian Path by defining a return to the depot as a visit to node $n+1$. Go to Step 5.

- **Step 5:** Check T_{min} with respect to the X-Cuts in list XCUTLIST and the Z-Cuts in ZCUTLIST. If the tour satisfies every cut (constraint), go to Step 8. If the tour fails to satisfy any cut, record the cut and go to Step 6.

- **Step 6:** Compute the cost matrix corresponding to this tour based on the arcs that have infinite cost and were recorded in COSTLIST. If the tour is the first one to be generated, no such arcs exist. Identify the infeasible subsequence corresponding to the cut recorded. Select the arcs of the infeasible subsequence one by one and assign a high cost. Solve a TSP problem over the resulting cost matrix. Add the

tour generated to the list TOURLIST, and record the arcs that have high cost in the list COSTLIST. Continue until every arc in the subsequence corresponding to the cut under consideration has been processed. Figure 5.1 explains this procedure. Go to Step 7.

- **Step 7:** Remove the current T_{min} from TOURLIST. Pick the minimum cost tour from TOURLIST. Call it T_{min}. Make a note of the corresponding arcs from COSTLIST. Go to Step 5.
- **Step 8:** At this stage, a feasible solution to the master problem is available. Substitute the x and z values of this solution to the master problem in the constraints of the subproblem. The result is a pure linear program in the variable t. Go to Step 9.
- **Step 9:** The subproblem is now algorithmically checked for feasibility, i.e., a check is performed to see if the tour generated by the master problem is feasible with respect to the time windows. If the tour is indeed feasible to the subproblem, then go to Step 11. Otherwise, go to Step 10.
- **Step 10:** The given sequence is not feasible with respect to the time windows. Therefore, the X-Cuts and the Z-Cuts must be generated as explained in chapters 3 and 4 and added to the lists XCUTLIST and ZCUTLIST. Go to Step 5.
- **Step 11:** Stop. The TSPTW has been solved.

A flowchart describing the steps of the heuristic can be found in figure 5.2.
Statement: If the TSP problem (in the heuristic) is solved using an optimal TSP solver, the solution generated by the heuristic for the TSPTW is the optimal solution.

Facts:

1. Every TSP problem at the new tour generation step (Step 6) is solved using an optimal TSP solver.
2. Let us refer to arcs with cost $= \infty$ as *blocked arcs*. The children nodes (in the TSP generating tree) have more blocked arcs as compared to their parent nodes.
3. Therefore, the cost of the optimal TSP tour at a child node is greater than or equal to the cost of the optimal TSP tour at its parent node.

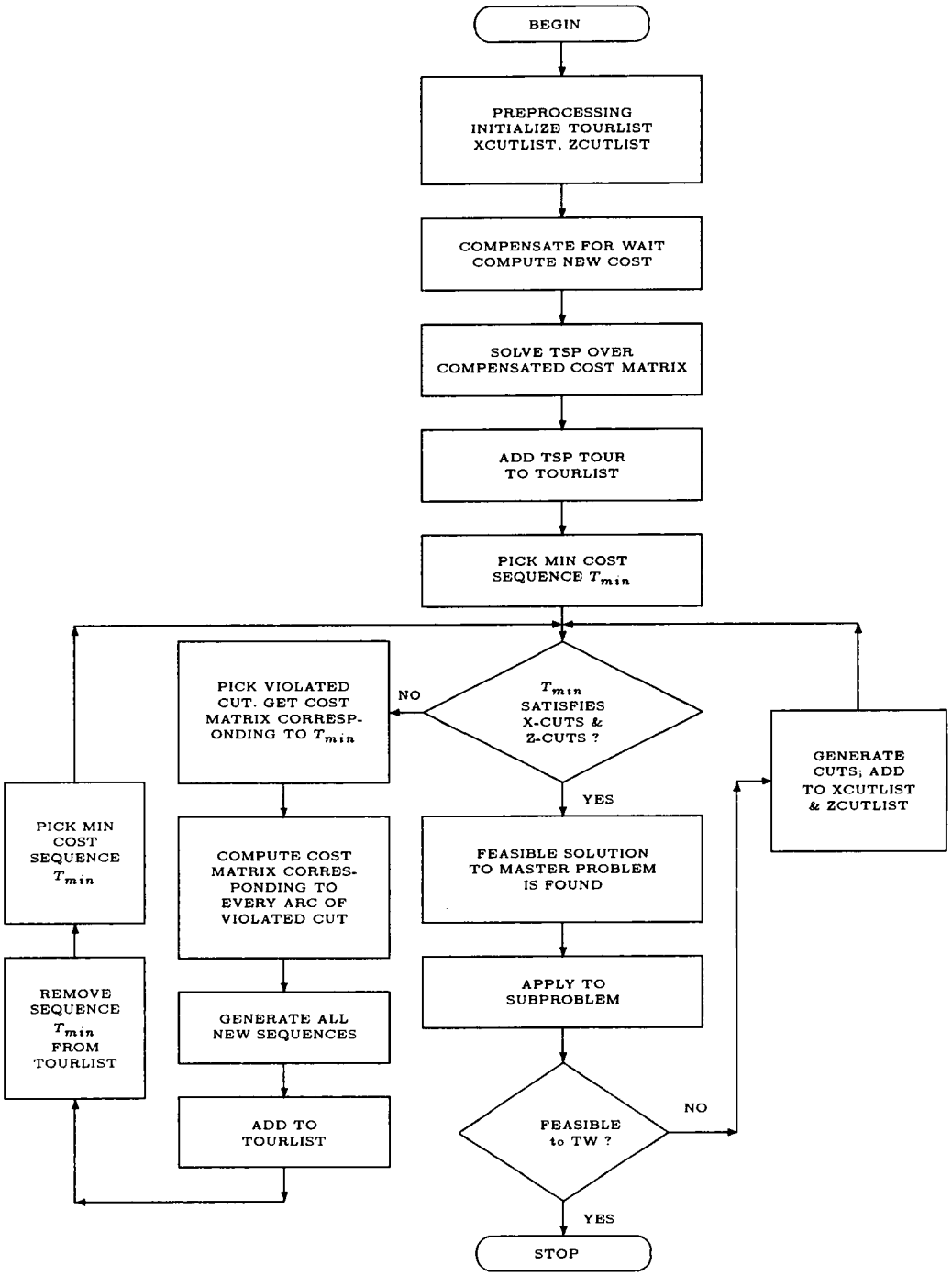

Figure 5.2: Flowchart

4. The minimum cost sequence from the list TOURLIST is selected for consideration as solution to the master problem. Call it T_{min}.

5. The T_{min} selected in step $n+i$ (for any $i > 0$) has a cost that is greater than or equal to the cost of T_{min} selected in step n.

The optimal solution of the TSPTW is a TSP tour that is feasible for the time windows. The heuristic basically enumerates all TSP tours in non-descending order of cost. The union of a parent node's children nodes consists of all TSP tours of the parent node except those tours that include the infeasible subsequence. Since, the cost of the optimal TSP tour at a child node is greater than or equal to the cost of the optimal tour at its parent node, there cannot exist a TSP tour with lower cost than the current TSP tour among the children nodes. Therefore if the current TSP tour is feasible for the time windows, then it is the optimal TSPTW tour.

5.4 Insertion Heuristic

The Insertion Heuristic described in this section identifies the node that is the cheapest to insert anywhere on a partial tour. It is similar to the Insertion Heuristic for Vehicle Routing Problems with Time Windows (as described in [43]) with respect to satisfying the time window constraints. This heuristic first tries to find the best insertion position on a partial tour for every unrouted node. An unrouted node is a node that is not included in the partial tour. It then selects the node that is the cheapest to insert. The heuristic consists of two main steps – the first step tries to find a good insertion position on the partial tour for all unrouted nodes and the second step selects the node to be inserted. Because the procedure is a tour-construction procedure, it is essential to first examine the necessary and sufficient conditions for time feasibility when inserting a node u between the nodes i_{p-1} and i_p, $1 \leq p \leq n+1$ on a partially constructed feasible route $(i_1, i_2, \ldots, i_{n+1})$, where i_1 and i_{n+1} refer to the depot. Let us denote by b_{i_r}, the time to begin service at node i_r for $1 \leq r \leq n+1$. It is also assumed that the salesman leaves the depot at time 0, which is the beginning of the time window for the depot. Let $b_{i_p}^{new}$ denote the new time at which service at node i_p begins (given the insertion of node u) and let w_{i_r} be the waiting time at node

i_r for $p \leq r \leq n+1$. Assuming the triangle inequality holds for both travel distances and times, this insertion defines a *push forward* in the schedule at i_p:

$$PF_{i_p} = b_{i_p}^{new} - b_{i_p} \geq 0.$$

Furthermore,

$$PF_{i_{r+1}} = max\left\{0, PF_{i_r} - w_{i_{r+1}}\right\},\ p \leq r \leq n.$$

The necessary and sufficient conditions for time feasibility when inserting a node, e.g. node u, between i_{p-1} and i_p, $1 \leq p \leq n+1$, on a partially constructed feasible route $(i_1, i_2, \ldots, i_{n+1})$, have been specified in [43] as:

$$b_u \leq l_u \text{ and } b_{i_r} + PF_{i_r} \leq l_{i_r},\ p \leq r \leq n+1$$

where l_{i_r} represents the close of the time window at node i_r and nodes i_1 and i_{n+1} refer to the depot.

The necessary and sufficient conditions ensure that insertion of node u between nodes i_{p-1} and i_p maintains time feasibility. The next step in the procedure is to select the node that is to be inserted. The route was first initialized as (i_1, i_{n+1}). Then, in every iteration, all unrouted nodes u were considered for insertion on the partially constructed tour while satisfying time feasibility. The best position for inserting node u is determined by computing

$$c_1(i(u), u, j(u)) = min[c_1(i_{p-1}, u, i_p)],\ p = 2, \ldots, n+1.$$

The quantity $c_1(i, u, j)$ is defined as:

$$c_1(i,u,j) = \alpha_1 c_{11}(i,u,j) + \alpha_2 c_{12}(i,u,j),$$

$$\alpha_1 + \alpha_2 = 1,\ \alpha_1 \geq 0,\ \alpha_2 \geq 0,$$

$$c_{11}(i,u,j) = d_{i,u} + d_{u,j} - \mu d_{i,j},\ \mu \geq 0,$$

$$c_{12}(i,u,j) = b_{j_u} - b_j,$$

where b_{j_u} is the new time for service to begin at node j, given that u is on the route. The node u^* to be inserted into the partial route is then selected by computing:

$$c_1(i(u^*), u^*, j(u^*)) \quad = \quad min[c_1(i(u), u, j(u)],\ u \text{ unrouted and feasible.}$$

In the present implementation of this insertion heuristic, μ is set equal to 1, while the values of α_1 and α_2 are varied to generate the best possible results.

5.5 Performance

The heuristic described in the previous section was used to solve Problem Sets I, II, and III as described in Chapter 4. For these problems, the value of the parameter λ that accounts for the cost of waiting for a time window to open, was considered to be zero. These results are presented in Tables 5.1, 5.2, 5.3, 5.4, 5.5, and 5.6. Of the forty-two problems in the three problem sets, the heuristic found the optimal solution for thirty-six problems. Therefore, the heuristic found the optimal solution approximately eighty-five percent of the time. With respect to Problem Set I, the heuristic found the optimal solution for problems R10-1, R10-3, R10-4, R12-2, R12-3, SK141-1, SK141-2, and SK142-1. For problem R10-2, the optimal solution had a value 868, while the value of the heuristic solution was 869.The optimal solution for problem R12-1 had a value 661, while the heuristic solution had a value 673. For problem R12-4, the optimal solution value was 445, while the heuristic solution value was 460. In the case of problem SK141-3, the optimal solution value was 499, while the heuristic solution value was 503. The heuristic solution time was considerably less than the optimal solution time for all problems except SK141-3. In the case of SK141-3, the heuristic solution time was considerably greater than the optimal solution time. With regard to Problem Set II, the heuristic found the optimal solution for eighteen of the twenty problems. For problem BAK11, the optimal solution value was 518, whereas the heuristic solution value was 574. For problem BAK17, the optimal solution value was 439 and the heuristic solution value was 449. The heuristic solution time was less than the optimal solution time for all problems in this set. In the case of Problem Set III, the heuristic found the optimal solution for all the problems. Again, the heuristic solution times were found to be considerably less than the optimal solution times. In order to test the performance of the heuristic over a wider set of problems, a new problem set consisting of thirty problems ranging in node size from fifteen to thirty-five, was solved using this heuristic. This set of problems is referred to as Problem Set IV. Details about this problem set can be found in Appendix E. The node coordinates for these problems were generated using a random number generator and the time windows were generated according to the procedure explained in Section 4.2 (with reference to Problem Set II).

Table 5.1: Heuristic on Problem Set I - 1

Problem Name	Time on tour with wait	Cost of tour	No. of master prob-lems solved	No. of X-Cuts	No. of prece-dence Z-Cuts (sets)	No. of TSPs sol-ved	Maximum size of sequence list - TOURLIST	Optimum cost of tour
R10-1	622	574	1	0	0	2	0	574
R10-2	924	869	1	0	0	2	0	868
R10-3	521	521	2	1	0	8	2	521
R10-4	526	395	4	3	27(3)	14	2	395
R12-1	966	673	2	2	22(2)	6	2	661
R12-2	976	673	1	0	0	2	0	673
R12-3	573	445	3	3	0	20	6	445
R12-4	697	460	5	6	66(6)	26	2	445
SK141-1	539	499	311	578	26(2)	28268	2193	499
SK141-2	526	499	151	253	91(7)	8248	607	499
SK141-3	518	503	351	702	65(5)	71638	4817	499
SK142-1	537	453	9	10	26(2)	82	7	453

Table 5.2: Heuristic on Problem Set I - 2

Problem Name	TSP solution time	Feasibility check & cut generation time	Time to pick min cost sequence	Total solution time
R10-1	N	N	N	0.19
R10-2	0.01	N	N	0.24
R10-3	0.01	N	N	0.28
R10-4	0.01	N	N	0.18
R12-1	0.04	N	N	0.27
R12-2	0.02	N	N	0.20
R12-3	0.04	0.01	N	0.36
R12-4	N	N	N	0.27
SK141-1	46.37	0.30	2.14	459.93
SK141-2	14.15	0.10	0.36	64.10
SK141-3	117.85	0.41	4.89	2030.53
SK142-1	0.15	0.02	N	0.63

Table 5.3: Heuristic on Problem Set II – 1

Problem Name	Time on tour with wait	Cost of tour	No. of master problems solved	No. of X-Cuts	No. of precedence Z-Cuts	No. of TSPs solved	Maximum size of sequence list TOURLIST	Optimal cost of tour
BAK1	626	596	1	0	0	2	0	596
BAK2	619	596	1	0	0	2	0	596
BAK3	596	596	1	0	0	2	0	596
BAK4	596	596	1	0	0	2	0	596
BAK5	596	596	1	0	0	2	0	596
BAK6	606	574	1	0	0	2	0	574
BAK7	600	574	1	0	0	2	0	574
BAK8	595	574	2	1	9	6	2	574
BAK9	592	574	2	1	9	6	2	574
BAK10	589	574	2	1	9	6	2	574
BAK11	586	574	5	4	27	26	5	518
BAK12	538	511	1	0	0	2	0	511
BAK13	536	511	4	3	0	16	3	511
BAK14	558	465	1	0	0	2	0	465
BAK15	557	465	6	5	27	24	1	465
BAK16	552	439	3	3	27	10	1	439
BAK17	567	449	8	8	36	54	3	439
BAK18	544	439	8	8	36	56	4	439
BAK19	541	439	8	8	27	90	7	439
BAK20	500	409	3	2	18	30	5	409

Table 5.4: Heuristic on Problem Set II – 2

Problem Name	TSP solution time	Feasibility check & cut generation time	Time to pick min cost sequence	Total solution time
BAK1	0.01	N	N	0.18
BAK2	N	N	N	0.20
BAK3	N	0.02	N	0.22
BAK4	0.01	N	N	0.19
BAK5	0.01	N	N	0.21
BAK6	0.01	0.01	N	0.22
BAK7	0.01	N	N	0.19
BAK8	N	N	N	0.22
BAK9	N	N	N	0.21
BAK10	0.01	0.01	N	0.21
BAK11	0.03	0.01	N	0.33
BAK12	N	N	N	0.16
BAK13	0.01	N	0.01	0.26
BAK14	0.02	0.01	N	0.22
BAK15	0.01	N	N	0.29
BAK16	0.01	N	N	0.26
BAK17	0.03	N	N	0.39
BAK18	0.05	N	0.03	0.41
BAK19	0.10	N	N	0.43
BAK20	0.04	N	N	0.31

Table 5.5: Heuristic on Problem Set III-1

Problem Name	Time on tour with wait	Cost of tour	No. of master prob-lems solved	No. of X-Cuts	No. of prece-dence Z-Cuts	No. of TSPs sol-ved	Maximum size of list of sequences TOURLIST	Optimal cost of tour
C20-1-20	820	820	1	0	0	2	0	820
C20-2-20	814	814	2	1	0	12	1	814
C20-3-20	725	698	1	0	0	2	0	698
C20-1-30	820	820	1	0	0	2	0	820
C20-2-30	812	809	1	0	0	2	0	809
C20-3-30	724	698	1	0	0	2	0	698
C20-1-40	818	803	2	1	0	6	2	803
C15-1-20	702	702	1	0	0	2	0	702
C15-1-30	706	666	1	0	0	2	0	666
C15-1-40	703	666	1	0	0	2	0	666

Table 5.6: Heuristic on Problem Set III – 2

Problem Name	TSP solution	Feasibility check & cut generation time	Time to pick min cost sequence	Total solution time
C20-1-20	0.03	N	N	0.25
C20-2-20	0.06	0.01	0.01	0.39
C20-3-20	0.03	0.01	N	0.24
C20-1-30	0.02	0.01	N	0.23
C20-2-30	0.03	N	N	0.23
C20-3-30	0.03	0.01	N	0.23
C20-1-40	0.04	0.01	N	0.33
C15-1-20	N	N	N	0.19
C15-1-30	0.03	N	N	0.22
C15-1-40	0.01	N	N	0.21

The problems in Problem Set IV were solved for four different values of the parameter λ namely, 0, 0.25, 0.5 and 1. The results are presented in Tables 5.7, 5.8, 5.9 and 5.10 for $\lambda = 0$, 5.11, 5.12, 5.13 and 5.14 for $\lambda = 0.25$, 5.15, 5.16, 5.17 and 5.18 for $\lambda = 0.5$, and 5.19, 5.20, 5.21 and 5.22 for $\lambda = 1$. The quantity N in a row indicates that the concerned solution time was negligible, i.e., less than 0.01. The quantity G in a row indicates that the total solution time was greater than a preset time limit of 3600 seconds. Since the problem could not be solved within the time limit, none of the concerned quantities could be measured. It can be observed that the major contributor to the total solution time was the time for solving the TSPs. All the computational tests were performed on a VAX/VMS mainframe computer (VAX 7000-640). It is observed that compensating for waiting for a time window to open (by setting $\lambda > 0$) helps in finding a solution within the prespecified time limit.

In order to get an estimate of the quality of the heuristic solution, Problem Set IV was also solved using an insertion heuristic. Five pairs of values for α_1 and α_2 were considered. They were, $\alpha_1 = 1, \alpha_2 = 0$, $\alpha_1 = 0.75, \alpha_2 = 0.25$, $\alpha_1 = 0.5, \alpha_2 = 0.5$, $\alpha_1 = 0.25, \alpha_2 = 0.75$, and $\alpha_1 = 0, \alpha_2 = 1$. These results are presented in Table 5.23 for the first set of values, Table 5.24 for the second set of values, Table 5.25 for the third set of values, Table 5.26 for the fourth set of values, and Table 5.27 for the fifth set of values.

It is found that the most number of problems were solved (using the insertion heuristic) with $\alpha_1 = 0, \alpha_2 = 1$. The best set of results obtained by using the heuristic to solve the problems was then compared with the best set of results obtained by using the Insertion Heuristic to solve the same set of test problems. This comparison is presented in Table 5.28 and Table 5.29. We see that the Decomposition Based Heuristic generally found a better solution than the Insertion Heuristic for the Traveling Salesman Problem with Time Windows. For two of the problems, the Insertion Heuristic could not find a solution. However, as expected, the Insertion Heuristic is much faster.

Finally, the effect of the Langevin, *et al.*, [4] method of preprocessing on Problem Set IV is presented in Table 5.30 and Table 5.31.

Table 5.7: Heuristic on Problem Set IV – 1; $\lambda = 0$

Problem Name	Time on tour with wait	Cost of tour	No. of master problems solved	No. of X-Cuts	No. of precedence Z-Cuts	No. of TSPs solved	Maximum size of list of sequences TOURLIST
R15-1	875	801	1	0	0	2	0
R15-2	835	646	4	4	56	14	2
R15-3	1181	815	4	4	56	104	14
R15-4	1088	807	28	32	112	2004	144
R15-5	1024	910	5	4	0	54	7
R15-6	967	858	1	0	0	2	0
R20-1	1026	742	3	3	38	14	2
R20-2	1019	634	9	12	228	814	103
R20-3	1541	850	5	9	133	3232	241
R20-4	1199	930	3	2	0	22	5
R20-5	1174	887	12	17	228	5912	576
SK20	–	–	–	–	–	–	–
R25-1	1201	751	4	11	264	120	12
R25-2	1145	662	2	4	96	16	4
R25-3	1798	1082	2	2	48	1676	190

Table 5.8: Heuristic on Problem Set IV - 2; $\lambda = 0$

Problem Name	TSP solution time	Feasibility check & cut generation time	Time to pick min cost sequence	Total solution time
R15-1	0.01	N	N	0.20
R15-2	0.06	0.02	N	0.41
R15-3	0.31	0.01	N	0.91
R15-4	7.71	0.03	0.07	17.21
R15-5	0.13	N	N	0.50
R15-6	0.03	N	N	0.27
R20-1	0.16	0.01	N	0.49
R20-2	7.15	0.02	0.02	13.38
R20-3	30.98	0.02	0.12	57.33
R20-4	0.21	N	N	0.58
R20-5	72.53	0.02	0.17	137.52
SK20	–	–	–	G
R25-1	3.00	N	0.01	4.91
R25-2	0.44	N	N	0.91
R25-3	52.41	0.01	0.06	75.85

Table 5.9: Heuristic on Problem Set IV -1; $\lambda = 0$ *cont.*

Problem Name	Time on tour with wait	Cost of tour	No. of master prob-lems solved	No. of X-Cuts	No. of prece-dence Z-Cuts (sets)	No. of TSPs sol-ved	Maximum size of list of sequences TOURLIST
R25-4	1703	961	8	11	240(10)	38490	4600
R25-5	1514	1232	2	1	24	6	2
R25-6	1514	949	11	19	312(13)	9908	1217
R30-1	1412	922	8	16	348(12)	2616	342
R30-2	1468	1281	2	1	29	10	3
R30-3	2292	2023	6	6	145	194	33
R30-4	2158	1268	5	14	406(14)	6462	973
R30-5	1983	1697	5	5	29(1)	168	33
R30-6	–	–	–	–	–	–	–
R35-1	1640	1231	3	2	0	18	3
R35-2	1616	1022	5	17	408(12)	9250	1254
R35-3	2401	2050	19	23	272(8)	394	39
R35-4	1910	1319	4	7	204(6)	2190	261
R35-5	2137	1590	3	3	34(1)	96	11
R35-6	1905	1563	2	4	102(3)	502	56

Table 5.10: Heuristic on Problem Set IV -2; $\lambda = 0$ *cont.*

Problem Name	TSP solution time	Feasibility check & cut generation time	Time to pick min cost sequence	Total solution time
R25-4	1097.70	0.02	2.39	2349.70
R25-5	0.18	N	N	0.50
R25-6	285.58	0.02	0.49	477.63
R30-1	177.47	0.03	0.14	246.43
R30-2	0.35	N	N	0.83
R30-3	5.58	N	N	11.21
R30-4	460.59	0.02	0.27	646.52
R30-5	5.42	N	N	9.86
R30-6	-	-	-	G
R35-1	1.61	N	N	2.50
R35-2	1217.97	0.03	0.35	1621.74
R35-3	33.60	0.04	0.05	49.74
R35-4	351.99	N	0.07	438.05
R35-5	9.03	N	0.09	12.86
R35-6	49.33	N	0.03	69.96

Table 5.11: Heuristic on Problem Set IV – 1; $\lambda = 0.25$

Problem Name	Time on tour with wait	Cost of tour	No. of master problems solved	No. of X-Cuts	No. of precedence Z-Cuts (sets)	No. of TSPs solved	Maximum size of list of sequences TOURLIST
R15-1	875	801	1	0	0	2	0
R15-2	835	635	3	4	56(4)	62	6
R15-3	1105	812	4	7	98(7)	144	27
R15-4	1062	806	27	32	112(8)	956	51
R15-5	1024	910	5	4	0	54	7
R15-6	967	858	1	0	0	2	0
R20-1	1026	742	3	3	38(2)	14	2
R20-2	1019	634	5	8	133(7)	708	87
R20-3	1532	884	7	10	152(8)	8498	483
R20-4	1199	930	1	0	0	2	0
R20-5	1244	907	10	17	228(12)	2936	280
SK20	535	523	19	21	0	258	48
R25-1	1175	767	4	10	240	316	40
R25-2	1145	686	4	7	168	4054	455
R25-3	1801	1085	3	3	72	148	12

Table 5.12: Heuristic on Problem Set IV – 2; $\lambda = 0.25$

Problem Name	TSP solution time	Feasibility check & cut generation time	Time to pick min cost sequence	Total solution time
R15-1	0.01	N	N	0.20
R15-2	0.18	0.01	N	0.65
R15-3	0.57	N	N	1.37
R15-4	3.61	0.04	0.09	7.79
R15-5	0.09	0.03	0.01	0.52
R15-6	0.02	0.01	N	0.22
R20-1	0.13	N	N	0.46
R20-2	7.44	0.03	0.05	12.97
R20-3	86.30	0.01	0.37	165.92
R20-4	0.02	N	N	0.22
R20-5	25.56	0.01	0.20	58.08
SK20	1.03	N	N	3.00
R25-1	7.15	N	0.03	11.89
R25-1	128.85	N	0.10	201.93
R25-3	4.38	0.01	N	6.83

Table 5.13: Heuristic on Problem Set IV – 1: $\lambda = 0.25$ *cont.*

Problem Name	Time on tour with wait	Cost of tour	No. of master prob-lems solved	No. of X-Cuts	No. of prece-dence Z-Cuts (sets)	No. of TSPs sol-ved	Maximum size of list of sequences TOURLIST
R25-4	1703	969	6	8	168	16320	1741
R25-5	1514	1232	2	1	24	6	2
R25-6	1495	985	13	26	384	7972	1008
R30-1	1412	922	6	17	406(14)	3862	480
R30-2	1468	1281	2	1	29(1)	10	3
R30-3	2292	2023	6	7	145(5)	198	33
R30-4	2158	1265	6	17	493(17)	9796	1422
R30-5	1983	1697	5	5	29(1)	168	33
R30-6	1996	1458	6	16	377(13)	25838	2945
R35-1	1640	1231	3	2	0	18	3
R35-2	1617	1020	6	19	408(12)	6844	832
R35-3	2401	2050	19	24	272(8)	398	39
R35-4	1871	1318	4	7	204(6)	1460	187
R35-5	2137	1590	2	1	34(1)	32	3
R35-6	1905	1563	2	4	102(3)	426	43

Table 5.14: Heuristic on Problem Set IV – 2; $\lambda = 0.25$ *cont.*

Problem Name	TSP solution time	Feasibility check & cut generation time	Time to pick min cost sequence	Total solution time
R25-4	421.82	0.02	0.37	703.97
R25-5	0.15	0.01	N	0.49
R25-6	202.31	N	0.45	368.95
R30-1	264.81	N	0.11	369.78
R30-2	0.40	N	N	0.85
R30-3	5.74	N	0.02	11.21
R30-4	584.16	0.04	0.61	903.13
R30-5	5.41	N	N	9.90
R30-6	1367.18	0.02	1.15	2428.70
R35-1	1.79	N	N	2.64
R35-2	923.51	0.02	0.40	1228.49
R35-3	33.45	0.02	0.04	50.15
R35-4	227.24	N	0.11	283.65
R35-5	3.19	N	N	4.60
R35-6	94.08	0.02	N	61.36

Table 5.15: Heuristic on Problem Set IV – 1; $\lambda = 0.5$

Problem Name	Time on tour with wait	Cost of tour	No. of master problems solved	No. of X-Cuts	No. of precedence Z-Cuts (sets)	No. of TSPs solved	Maximum size of list of sequences TOURLIST
R15-1	875	801	1	0	0	2	0
R15-2	835	661	2	3	42(3)	42	5
R15-3	1180	814	4	7	98(7)	132	20
R15-4	1173	851	25	28	98(7)	788	53
R15-5	1024	910	5	4	0	54	7
R15-6	1013	840	2	1	14(1)	8	2
R20-1	1026	742	3	3	38(2)	14	2
R20-2	1019	669	2	1	19(1)	24	7
R20-3	1541	867	11	12	171(9)	2424	246
R20-4	1199	930	1	0	0	2	0
R20-5	1123	952	8	10	133(7)	8708	659
SK20	539	527	25	32	0	508	72
R25-1	1201	756	4	8	192(8)	114	13
R25-2	1145	686	6	9	216(9)	1442	242
R25-3	1801	1085	3	3	72(3)	148	12

Table 5.16: Heuristic on Problem Set IV - 2; $\lambda = 0.5$

Problem Name	TSP solution time	Feasibility check & cut generation time	Time to pick min cost sequence	Total solution time
R15-1	0.02	0.01	N	0.20
R15-2	0.13	0.01	N	0.53
R15-3	0.58	N	N	1.29
R15-4	3.05	0.02	0.01	6.50
R15-5	0.11	0.01	N	0.51
R15-6	0.04	N	N	0.29
R20-1	0.14	N	N	0.50
R20-2	0.26	0.01	N	0.70
R20-3	21.72	0.01	0.09	42.26
R20-4	0.04	N	N	0.26
R20-5	72.51	0.01	0.35	179.70
SK20	1.99	0.02	0.03	5.38
R25-1	2.52	N	0.01	4.21
R25-2	40.17	0.03	0.06	63.52
R25-3	4.43	0.01	N	6.87

Table 5.17: Heuristic on Problem Set IV – 1; $\lambda = 0.5$ *cont.*

Problem Name	Time on tour with wait	Cost of tour	No. of master prob-lems solved	No. of X-Cuts	No. of prece-dence Z-Cuts (sets)	No. of TSPs sol-ved	Maximum size of list of sequences TOURLIST
R25-4	–	–	–	–	–	–	–
R25-5	1514	1241	2	1	24(1)	6	2
R25-6	1495	976	7	17	288(12)	4230	515
R30-1	1460	946	7	12	232(8)	1416	162
R30-2	1468	1281	2	1	29(1)	10	3
R30-3	2292	2023	6	7	145(5)	198	33
R30-4	2158	1265	5	14	406(14)	6800	903
R30-5	1983	1697	5	5	29(1)	168	33
R30-6	1996	1435	6	14	319(11)	592	76
R35-1	1640	1231	3	2	0	18	3
R35-2	1630	1031	6	18	442(13)	11350	1370
R35-3	2401	2050	19	24	272(8)	398	39
R35-4	1871	1318	4	7	204(6)	1460	187
R35-5	2137	1590	2	1	34(1)	32	3
R35-6	1903	1561	3	3	68(2)	254	20

Table 5.18: Heuristic on Problem Set IV – 2; $\lambda = 0.5$ *cont.*

Problem Name	TSP solution time	Feasibility check & cut generation time	Time to pick min cost sequence	Total solution time
R25-4	–	–	–	G
R25-5	0.17	N	N	0.48
R25-6	111.87	N	0.12	186.31
R30-1	98.65	N	0.01	133.10
R30-2	0.40	0.02	N	0.84
R30-3	5.57	N	N	11.38
R30-4	424.21	0.02	0.25	628.76
R30-5	5.54	N	N	9.94
R30-6	38.16	N	0.02	54.76
R35-1	1.69	N	N	2.61
R35-2	1638.60	0.08	0.57	2199.28
R35-3	33.39	0.04	0.02	49.61
R35-4	229.01	0.02	0.07	285.61
R35-5	3.25	N	N	4.66
R35-6	23.89	0.01	0.03	34.44

Table 5.19: Heuristic on Problem Set IV – 1; $\lambda = 1$

Problem Name	Time on tour with wait	Cost of tour	No. of master problems solved	No. of X-Cuts	No. of precedence Z-Cuts (sets)	No. of TSPs solved	Maximum size of list of sequences TOURLIST
R15-1	875	823	1	0	0	2	0
R15-2	835	661	2	3	42(3)	50	5
R15-3	1112	819	2	3	42(3)	10	2
R15-4	1173	851	14	15	70(5)	490	31
R15-5	1024	910	5	4	0	54	7
R15-6	1013	840	2	1	14(1)	8	2
R20-1	1026	742	3	3	38(2)	14	2
R20-2	1000	708	7	10	114(6)	1016	134
R20-3	1385	936	8	8	114(6)	11936	1013
R20-4	1199	930	1	0	0	2	0
R20-5	1123	973	5	5	95(5)	944	74
SK20	539	527	281	299	0	9190	700
R25-1	1201	777	4	8	192(8)	146	16
R25-2	1150	772	5	8	192(8)	6684	915
R25-3	1801	1093	3	3	72(3)	148	12

Table 5.20: Heuristic on Problem Set IV - 2; $\lambda = 1$

Problem Name	TSP solution time	Feasibility check & cut generation time	Time to pick min cost sequence	Total solution time
R15-1	0.01	0.01	N	0.23
R15-2	0.13	N	N	0.65
R15-3	0.04	N	N	0.31
R15-4	1.95	N	0.01	4.06
R15-5	0.13	0.01	N	0.56
R15-6	0.06	N	N	0.34
R20-1	0.11	0.01	N	0.47
R20-2	9.87	0.02	0.02	17.76
R20-3	133.28	0.01	0.34	265.17
R20-4	0.05	N	N	0.26
R20-5	8.48	0.01	0.05	16.54
SK20	47.57	0.29	0.21	130.14
R25-1	3.17	N	N	5.28
R25-2	173.03	0.01	0.29	321.61
R25-3	4.35	0.01	0.03	6.81

Table 5.21: Heuristic on Problem Set IV - 1: $\lambda = 1$ *cont.*

Problem Name	Time on tour with wait	Cost of tour	No. of master prob-lems solved	No. of X-Cuts	No. of prece-dence Z-Cuts (sets)	No. of TSPs sol-ved	Maximum size of list of sequences TOURLIST
R25-4	1706	1015	3	3	72(3)	10	2
R25-5	1514	1241	2	1	24(1)	6	2
R25-6	1495	1028	9	15	288(12)	7906	928
R30-1	1460	946	7	12	232(8)	1338	150
R30-2	1468	1281	2	1	29(1)	10	3
R30-3	2292	2023	6	6	145(5)	194	33
R30-4	2158	1265	5	14	406(14)	6776	891
R30-5	1983	1697	5	5	29(1)	168	33
R30-6	1996	1435	2	3	58(2)	10	3
R35-1	1640	1253	3	2	0	18	3
R35-2	1617	1026	8	17	442(13)	12942	1748
R35-3	2401	2050	19	23	272(8)	394	39
R35-4	1871	1318	4	7	204(6)	1460	187
R35-5	2137	1590	2	1	34(1)	32	3
R35-6	1905	1572	2	4	102(3)	426	43

Table 5.22: Heuristic on Problem Set IV – 2: $\lambda = 1$ *cont.*

Problem Name	TSP solution time	Feasibility check & cut generation time	Time to pick min cost sequence	Total solution
R25-4	0.29	N	N	0.71
R25-5	0.15	N	N	0.43
R25-6	209.86	0.02	0.26	350.75
R30-1	92.03	0.05	0.07	124.67
R30-2	0.38	0.02	N	0.84
R30-3	5.84	0.02	N	11.51
R30-4	421.15	0.01	0.31	624.41
R30-5	5.67	N	N	9.92
R30-6	0.46	N	N	1.01
R35-1	1.72	N	N	2.64
R35-2	1722.34	0.01	0.62	2372.81
R35-3	33.17	0.05	0.02	49.62
R35-4	229.16	0.02	0.06	285.52
R35-5	3.10	N	N	4.44
R35-6	40.02	N	0.01	57.66

Table 5.23: Insertion Heuristic on Problem Set IV ($\alpha_1 = 1, \alpha_2 = 0$)

Problem Name	Cost of tour	Time on tour with wait	Solution time	Problem Name	Cost of tour	Time on tour with wait	Solution time
R15-1	866	972	0.02	R25-4	1099	1779	0.07
R15-2	789	1010	0.02	R25-5	1383	1551	0.07
R15-3	929	1184	0.02	R25-6	1256	1542	0.06
R15-4	931	1169	0.02	R30-1	1077	1508	0.12
R15-5	–	–	–	R30-2	1411	1540	0.11
R15-6	840	1013	0.02	R30-3	2309	2419	0.11
R20-1	–	–	–	R30-4	1569	2174	0.12
R20-2	765	1055	0.03	R30-5	–	–	–
R20-3	903	1640	0.03	R30-6	1532	1899	0.11
R20-4	–	–	–	R35-1	–	–	–
R20-5	–	–	–	R35-2	1253	1720	0.18
SK20	–	–	–	R35-3	–	–	–
R25-1	889	1175	0.05	R35-4	–	–	–
R25-2	924	1296	0.07	R35-5	–	–	–
R25-3	1240	1803	0.05	R35-6	–	–	–

Table 5.24: Insertion Heuristic on Problem Set IV ($\alpha_1 = 0.75, \alpha_2 = 0.25$)

Problem Name	Cost of tour	Time on tour with wait	Solution time	Problem Name	Cost of tour	Time on tour with wait	Solution time
R15-1	833	892	0.02	R25-4	1236	1689	0.07
R15-2	765	884	0.02	R25-5	-	-	-
R15-3	1074	1275	0.02	R25-6	1271	1546	0.07
R15-4	1074	1252	0.01	R30-1	1186	1447	0.11
R15-5	-	-	-	R30-2	1439	1498	0.11
R15-6	1025	1070	0.02	R30-3	2266	2348	0.12
R20-1	941	1154	0.03	R30-4	1680	2230	0.12
R20-2	905	1101	0.03	R30-5	-	-	-
R20-3	1173	1616	0.04	R30-6	-	-	-
R20-4	-	-	-	R35-1	-	-	-
R20-5	-	-	-	R35-2	1263	1619	0.19
SK20	-	-	-	R35-3	-	-	-
R25-1	1002	1249	0.07	R35-4	-	-	-
R25-2	1004	1294	0.06	R35-5	-	-	-
R25-3	1372	1781	0.07	R35-6	-	-	-

Table 5.25: Insertion Heuristic on Problem Set IV ($\alpha_1 = 0.5, \alpha_2 = 0.5$)

Problem Name	Cost of tour	Time on tour with wait	Solution time	Problem Name	Cost of tour	Time on tour with wait	Solution time
R15-1	837	892	0.02	R25-4	1256	1689	0.06
R15-2	730	884	0.02	R25-5	–	–	–
R15-3	1018	1105	0.02	R25-6	1114	1514	0.07
R15-4	1018	1082	0.02	R30-1	1150	1579	0.12
R15-5	–	–	–	R30-2	–	–	–
R15-6	–	–	–	R30-3	2201	2348	0.12
R20-1	945	1154	0.03	R30-4	1715	2230	0.12
R20-2	905	1101	0.04	R30-5	2064	2107	0.12
R20-3	1192	1616	0.04	R30-6	–	–	–
R20-4	–	–	–	R35-1	–	–	–
R20-5	1070	1209	0.04	R35-2	1281	1612	0.21
SK20	–	–	–	R35-3	2280	2364	0.21
R25-1	903	1175	0.06	R35-4	–	–	–
R25-2	903	1119	0.05	R35-5	–	–	–
R25-3	1409	1869	0.05	R35-6	–	–	–

Table 5.26: Insertion Heuristic on Problem Set IV ($\alpha_1 = 0.25, \alpha_2 = 0.75$)

Problem Name	Cost of Tour	Time on tour with wait	Solution time	Problem Name	Cost of tour	Time on tour with wait	Solution time
R15-1	857	912	0.01	R25-4	1458	1902	0.07
R15-2	687	832	0.02	R25-5	–	–	–
R15-3	1000	1105	0.01	R25-6	1275	1534	0.06
R15-4	1018	1082	0.01	R30-1	1083	1579	0.12
R15-5	1044	1070	0.02	R30-2	–	–	–
R15-6	–	–	–	R30-3	2184	2338	0.11
R20-1	893	1064	0.04	R30-4	1697	2230	0.12
R20-2	812	998	0.03	R30-5	2030	2107	0.11
R20-3	1078	1430	0.04	R30-6	–	–	–
R20-4	1115	1216	0.03	R35-1	1450	1678	0.20
R20-5	1131	1140	0.02	R35-2	1214	1611	0.20
SK20	–	–	–	R35-3	2290	2380	0.20
R25-1	910	1175	0.07	R35-4	1696	1957	0.20
R25-2	883	1112	0.07	R35-5	1866	2137	0.22
R25-3	1364	1741	0.07	R35-6	–	–	–

Table 5.27: Insertion Heuristic on Problem Set IV ($\alpha_1 = 0, \alpha_2 = 1$)

Problem Name	Cost of tour	Time on tour with wait	Solution time	Problem Name	Cost of tour	Time on tour with wait	Solution time
R15-1	857	912	0.02	R25-4	1695	1866	0.05
R15-2	687	832	0.02	R25-5	–	–	–
R15-3	1064	1180	0.02	R25-6	1571	1511	0.07
R15-4	1064	1157	0.02	R30-1	1425	1550	0.11
R15-5	1044	1070	0.02	R30-2	1504	1504	0.11
R15-6	890	967	0.02	R30-3	2303	2323	0.12
R20-1	1043	1072	0.03	R30-4	1826	2169	0.13
R20-2	812	998	0.04	R30-5	2030	2107	0.12
R20-3	1352	1514	0.03	R30-6	1825	1897	0.12
R20-4	1115	1216	0.03	R35-1	1619	1673	0.20
R20-5	1131	1140	0.03	R35-2	1612	1735	0.20
SK20	–	–	–	R35-3	2429	2443	0.20
R25-1	910	1175	0.07	R35-4	1838	1936	0.20
R25-2	883	1112	0.07	R35-5	1957	2134	0.20
R25-3	1360	1741	0.07	R35-6	–	–	–

Table 5.28: Comparison of Proposed Heuristic and Insertion Heuristic

Problem Name	Decomposition Based Heuristic			Insertion Heuristic		
	Best cost	Corresponding time on tour	Solution time	Best cost	Corresponding time on tour	Solution time
R15-1	801	875	0.20	833	892	0.02
R15-2	635	835	0.65	687	832	0.02
R15-3	812	1105	1.37	929	1184	0.02
R15-4	806	1162	7.79	931	1169	0.02
R15-5	910	1024	0.50	1044	1070	0.02
R15-6	840	1013	0.29	840	1013	0.02
R20-1	742	1026	0.49	893	1064	0.04
R20-2	634	1019	13.38	765	1055	0.03
R20-3	850	1541	57.33	903	1640	0.03
R20-4	930	1199	0.22	1115	1216	0.03
R20-5	887	1174	137.52	1070	1209	0.04
SK20	523	535	2.61	–	–	–
R25-1	751	1201	4.91	889	1175	0.05
R25-2	662	1145	0.91	883	1112	0.07
R25-3	1082	1798	75.85	1240	1803	0.05

Table 5.29: Comparison of Proposed Heuristic and Insertion Heuristic *cont.*

Problem Name	Decomposition Based Heuristic			Insertion Heuristic		
	Best cost	Corresponding time on tour	Solution time	Best cost	Corresponding time on tour	Solution time
R25-4	961	1703	2349.70	1099	1779	0.07
R25-5	1232	1514	0.49	1383	1551	0.07
R25-6	949	1514	477.63	1114	1514	0.07
R30-1	922	1412	246.43	1077	1508	0.12
R30-2	1281	1468	0.83	1411	1540	0.11
R30-3	2023	2292	11.21	2184	2348	0.11
R30-4	1265	2158	624.41	1569	2174	0.12
R30-5	1697	1983	9.86	2030	2107	0.11
R30-6	1435	1996	1.01	1532	1899	0.11
R35-1	1231	1640	2.50	1450	1678	0.20
R35-2	1020	1617	1228.49	1214	1611	0.20
R35-3	2050	2401	49.61	2280	2364	0.21
R35-4	1318	1871	283.65	1696	1957	0.20
R35-5	1590	2137	4.44	1866	2137	0.22
R35-6	1561	1903	34.44	–	–	–

Table 5.30: Effects of TW Reduction and Arc Elimination - Problem Set IV

Problem Name	Average TW width before preprocessing	Average TW width after preprocessing	Number of feasible arcs before preprocessing	Number of feasible arcs after preprocessing
R15-1	162.00	152.00	136	58
R15-2	241.00	226.00	152	106
R15-3	400.00	372.00	159	123
R15-4	443.00	418.00	164	137
R15-5	183.00	167.00	132	53
R15-6	288.00	262.00	150	100
R20-1	212.00	202.00	249	134
R20-2	313.00	304.00	278	213
R20-3	550.00	529.00	290	244
R20-4	245.00	230.00	240	106
R20-5	387.00	362.00	279	205
SK20	401.00	393.00	380	380
R25-1	246.00	237.00	395	227
R25-2	365.00	357.00	438	342
R25-3	363.00	348.00	398	230

Table 5.31: Effects of TW Reduction and Arc Elimination - Problem Set IV *cont.*

Problem Name	Average TW width before preprocessing	Average TW width after preprocessing	Number of feasible arcs before preprocessing	Number of feasible arcs after preprocessing
R25-4	538.00	521.00	437	348
R25-5	302.00	287.00	386	208
R25-6	418.00	397.00	429	312
R30-1	232.00	225.00	558	297
R30-2	146.00	142.00	502	132
R30-3	226.00	219.00	507	142
R30-4	402.00	389.00	571	330
R30-5	193.00	185.00	500	132
R30-6	343.00	330.00	556	290
R35-1	164.00	160.00	697	231
R35-2	229.00	223.00	740	361
R35-3	236.00	231.00	694	198
R35-4	339.00	327.00	760	402
R35-5	252.00	243.00	699	228
R35-6	300.00	289.00	734	342

Preprocessing reduced the number of feasible arcs for each of the problems in this set. However, not much reduction in time window width was achieved.

5.6 Conclusions

The details of the heuristic based on the Decomposition-Based Solution Methodology are presented in this chapter. The performance of the heuristic was tested by solving problems for which the optimal solution is known (Problem Sets I, II, and III). The results demonstrate that the heuristic found the optimal solution in eighty five percent of the problems. Define the relative deviation between the optimal solution and heuristic solution as

$$\frac{\text{(heuristic solution)} - \text{(optimal solution)}}{\text{(optimal solution)}}$$

Then the relative deviation between the optimal solution and the heuristic solution for problems R10-1, R12-1, R12-4, SK141-3, BAK11, and BAK17 is 0.0015, 0.018, 0.337, 0.008, 0.108, and 0.0228 respectively. Therefore, the largest value of relative deviation is approximately ten percent.

The heuristic was then tested on a larger set of problems with node size varying from fifteen to thirty five. It was seen that compensating for waiting at a node (for the time window to open) helped to find a solution (within the preset time limit). However, larger values of λ (0.5 and 1) lead to higher cost solutions. An insertion heuristic was used to solve this new set of problems in order to estimate the quality of the decomposition-based heuristic solution. It was found that the decomposition-based heuristic solution was better than the insertion heuristic solution for all the problems tested. As expected, the insertion heuristic was much faster (on problems that it could solve).

Problem Set IV has a wide range of time window widths. The time are also characterized by considerable overlap. The Langevin preprocessing, when applied to this problem set, did not produce much reduction in the width of the time windows. Define the following two quantities.

Relative reduction in Time Window width =

$$\frac{\text{(original TW width)} - \text{(reduced TW width)}}{\text{(original TW width)}}$$

Percentage of arcs that are feasible =

$$\frac{\text{(number of feasible arcs)}}{\text{(original number of arcs)}}$$

The original number of feasible arcs is $n(n-1)$, n being the number of nodes. With respect to Problem Set IV, the average relative reduction in time window width is 0.0655 and the average percentage of arcs that are feasible is 59.4 percent. Thus, it is observed that more than fifty percent of the arcs remain feasible, on the average, in Problem Set IV. A comparison of Problem Sets I, II, and III (from Chapter 4) shows that the average relative reduction in time window widths for the three problem sets is 0.07, 0.14 and 0.57 respectively. The average percentage of arcs that are feasible (after preprocessing) in the three problem sets is sixty-two percent, forty-five percent and nine percent respectively. It is seen that Problem Set III achieves the maximum amount of reduction in time window width. Also, the cardinality of the feasible src set (after preprocessing) is very small.

Chapter 6

Conclusions and Future Research

The Traveling Salesman Problem with Time Windows has been studied and a new solution methodology based on Bender's Decomposition has been investigated in the preceding chapters. Since the problem belongs to the class NP Hard, optimal methods in general encounter difficulties with regard to storage and computation time. Hence, a heuristic based on the optimal solution methodology has been proposed. The optimal solution methodology and the heuristic have been tested on randomly-generated problems with Euclidean cost matrices as well as real-life problems with Euclidean cost matrices. Also, a modification of the objective function is considered for solution of the Traveling Salesman Makespan Problem with Time Windows. Some observations and conclusions are presented in Section 6.1. Ideas for future research are presented in Section 6.2.

6.1 Observations and Conclusions

The observations and conclusions presented refer to the solution methodology in general. The Bender's Decomposition-based solution methodology has been shown to work well for problems with a reasonably wide range of time window widths and time window overlaps.

Many solution procedures work well when the time windows are tight and restrictive in nature, but do not work as well when the time windows are wide and tend to overlap. This is because tight time windows render many arcs infeasible and hence the number of feasible arcs are reduced. However, when the time windows are not very tight and there exists considerable overlap between the time windows, fewer arcs are rendered infeasible. Therefore, the number of feasible arcs increases and this tends to increase the number of solutions enumerated in most branch-and-bound schemes. The Bender's Decomposition-based solution procedure sequentially selects Traveling Salesman tours (beginning with the optimal Traveling Salesman tour) until a tour feasible with respect to time windows is found. This procedure enumertes TSP tours in non-descending order of cost until a tour that is feasible for the time windows is found. Therefore, the first feasible solution is the optimal one. Reduction of the number of viable arcs does not have as great an impact on this method as it has on branch-and-bound schemes. The Bender's Decomposition-based solution methodology has been shown to optimally solve problems where the time windows are very wide and as a result the cardinality of the feasible arc set is almost equal to the cardinality of the original arc set.

As observed in chapters 3 and 4, the tour can be represented in terms of either arc variables or positional variables. Each representation has its own advantages. Arc variable representation of the tour enables the cuts to be represented in terms of arc variables. These cuts ensure that the corresponding infeasible subsequence does not occur anywhere in a feasible solution to the master problem. Positional variable representation of the tour enables cuts to be represented in terms of positional variables resulting in the establishment of certain precedence relationships. Both kinds of cuts prevent infeasible sequences from being selected as solutions to the master problem.

The formulation described here can be extended to handle other special constraints such as a priori precedence between the nodes. This precedence can be easily expressed in terms of positional variable cuts. A priori knowledge of infeasible subsequences (in terms of contiguous arcs) can also be easily incorporated in terms of arc variable cuts. The modular nature of the solution procedure is advantageous in that it can easily accommodate improved solution procedures for the asymmetric Traveling Salesman Problem into the heuristic solution procedure. This can lead to an overall improvement in the performance of the heuristic.

subsequence.

6.2 Future Research

The master problem in the formulation presented in the preceding chapters is an asymmetric Traveling Salesman Problem with additional constraints. Better methods of solving the asymmetric Traveling Salesman Problem could lead to better solutions to the Traveling Salesman Problem with Time Windows. This aspect remains to be studied. Solution of the master problem by means of Lagrangean Relaxation of the additional constraints or Lagrangean Decomposition of the master problem can also be studied. Another area open to investigation is the inclusion of additional constraints such as capacity constraints, multiple time window constraints, and pickup and dropoff constraints. Yet another area of research is establishing the correspondence between extreme rays of the subproblem dual and subsequences. It has been shown that a subsequence $i, i+1, \ldots, i+m$ corresponds to an extreme ray of the subproblem dual. It would be advantageous to show that every extreme ray of the subproblem dual corresponds to a subsequence of the form $i, i+1, \ldots, i+m$. Then, identification of an extreme ray with diverging objective in the dual would necessarily lead to identification of an infeasible

The Traveling Salesman Problem with Time Windows has various applications. These include, but are not limited to, transportation and engineering. Examples of applications related to transportation include routing of vehicles in each route of a Vehicle Routing Problem or Multiple Traveling Salesman Problem and certain cases of the Single Vehicle Pickup and Delivery Problem. A study of the literature summarized in Chapter 2 provides greater details. Applications of the Traveling Salesman Problem with Time Windows in the area of engineering have been less studied. The applicability of the TSPTW for modeling problems in the various fields of engineering have yet to be studied. For example, in electrical engineering, one specific application is that of routing signals (electrical waveforms that carry information) in high performance Very Large Scale Integration (VLSI). Global signals, all across the VLSI chip have to be routed so as to arrive at predetermined times to various functional blocks. The difficulty encountered in this routing is the loss of the signal in the interconnects. On an average, about seventy percent of the chip area is taken up by

interconnects. One way of tackling this problem would be by minimizing the routing length and have the signals arrive at the functional blocks during specified time periods. The effectiveness of the Traveling Salesman Problem with Time Windows for this application needs to be studied. Even though this is one specific example application in engineering, the general nature of the Traveling Salesman Problem with Time Windows should lend itself to many other applications as well.

Bibliography

Bibliography

[1] L. Bodin, B. Golden, A. Assad, and M. Ball. Routing and Scheduling of Vehicles and Crew: The State of the Art. *Computers and Operations Research*, 10(2):63–211, 1983.

[2] Edward K. Baker. An Exact Algorithm for the Time-Constrained Traveling Salesman Problem. *Operations Research*, 31(5):938–945, 1983.

[3] E. L. Lawlwr, J. K. Lenstra, A. H. G. Rinnooy Kan, and D. B. Shmoys. *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization.* John Wiley and Sons, New York, 1985.

[4] A. Langevin, M. Desrochers, J. Desrosiers, S. Gelinas, and F. Soumis. A Two-Commodity Flow Formulation for the Traveling Salesman and Makespan Problems with Time Windows. *Networks*, 23:631–640, 1993.

[5] R. G. Parker and R. L. Rardin. The Traveling Salesman Problem: An Update of Research. *Naval research Logistics Quarterly*, 30:69–96, 1983.

[6] Bruce Golden, L. Bodin, T. Doyle, and W. Stewart, Jr. Approximate Traveling Salesman Algorithms. *Operations Research*, 28(3):694–711, 1980.

[7] G. Laporte. The Traveling Salesman Problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59:231–247, 1992.

[8] I. I. Melamed, S. I. Sergeev, and I. Kh. Sigal. Traveling Salesman Problem. II. Exact Methods. *Automation and Remote Control*, 50:1303–1324, 1990.

[9] I. I. Melamed, S. I. Sergeev, and I. Kh. Sigal. Traveling Salesman Problem. Approximate Algorithms. *Autamation and Remote Control*, 50:1459–1479, 1990.

[10] R. M. Karp. On the Computational Complexity of Combinatorial Problems. *Networks*, 5:45–68, 1975.

[11] Richard M. Karp. Reducibility among Combinatorial Problems. *Complexity of Computer Computations(Edited by R. Miller and J. Thatcher)*, pages 85–104, 1972.

[12] R. L. Graham M. R. Garey and D. S. Johnson. Some NP-Complete Geometric Problems. *Proceedings of the 8th SIGACT symposium on the Theory of Computing*, pages 10–22, 1976.

[13] Christos H. Papadimitriou. *Theoretical Computer Science*, 4:237–244, 1977.

[14] Christos M. Papadimitriou and Kenneth Steiglitz. Some Complexity Results for the Traveling Salesman Problem. *Combinatorial Optimization:algorithms and complexity*, pages 1–9, 1982.

[15] John D. C. Little, Katta G. Murty, Dura W. Sweeney, and Caroline Karel. An Algorithm for the Traveling Salesman Problem. *Operations Research*, 11:972–989, 1963.

[16] Giorgio Carpenato and Paolo Toth. Some New Branching and Bounding Criteria for the Asymmetric Traveling Salesman Problem. *Management Science*, 26(7):736–743, 1980.

[17] J. F. Pekny and D. L. Miller. A Parallel Branch-and-Bound Algorithm for Solving Large Asymmetric Traveling Salesman Problems.

[18] Michael Held and Richard M. Karp. The Traveling Salesman Problem and Minimum Spanning Trees. *Operations Research*, 18:1138–1162, 1970.

[19] M. Fischetti and P. Toth. An Additive Bounding Procedure for the Symmetric Traveling Salesman Problem. *Mathematical Programming*, 53:173–197, 1992.

[20] G. Carpenato, M. Fischetti, and P. Toth. New Lower Bounds for the Symmetric Traveling Salesman Problem. *Mathematical Programming*, 45:233–254, 1989.

[21] K. Malik and M. L. Fisher. A Dual Ascent Algorithm for the 1-Tree Relaxation of the Symmetric Traveling Salesman Problem. *Perations Research Letters*, 9:1–7, 1990.

[22] Harlan Crowder and Manfred W. Padberg. Solving Large-Scale Symmetric Traveling Salesman Problems to Optimality. *Management Science*, 26(5):495–509, 1980.

[23] M. Padberg and G. Rinaldi. Facet Identification for the Symmetric Traveling Salesman Polytope. *Mathematical Programming*, 47:219–257, 1990.

[24] M. Padberg and G. Rinaldi. A Branch-and-Cut Algorithm for the Resolution of Large-Scale Symmetric Traveling Salesman Problems. *SIAM Review*, 33:60–100, 1991.

[25] M. Grotschel and O. Holland. Solution of Large-Scale Symmetric Traveling Salesman Problems. *Mathematical Programming*, 51:141–202, 1991.

[26] Nicos Christofides. The Traveling Salesman Problem. *Combinatorial Optimization (Edited by N. Christofides, R. Mingozzi, P. Toth and C. Sandi)*, pages 131–149, 1979.

[27] Nicos Christofides. Bounds for the Traveling Salesman Problem. *Operations Research*, 20:1044–1056, 1972.

[28] N. Christofides, A. Mingozzi, and P. Toth. State-Space Relaxation Procedures for the Computation of Bounds to Routing Problems. *Networks*, 11:145–164, 1980.

[29] B. R. Feiring. An Efficient Procedure for the n-City Traveling Salesman Problem. *Mathematical and Computer Modelling*, 13:95–98, 1990.

[30] Nicos Christofides. Worst-Case Analysis of a New Heuristic for the Traveling Salesman Problem. *Report No. 388, Graduate School of Industrial Administration, Carnegie Mellon University*, Feb 1976.

[31] Shen Lin. Computer Solutions of the Traveling Salesman Problem. *The Bell System Technical Journal*, pages 2245–2269, Dec 1965.

[32] M. H. J. Webb. Some Methods of Producing Approximate Solutions to Traveling Salesman Problems with Hundreds or Thousands of Cities. *Operational Research Quarterly*, 22(1):49–66, 1971.

[33] Nicos christofides and Samuel Eilon. Algorithms for Large-Scale Traveling Salesman Problems. *Operational Research Quarterly*, 23(4):511–518, 1972.

[34] Robert A. Russell. An Effective Heuristic for the M-Tour Traveling Salesman Problem with Some Side Constraints. *Operations Research*, 25(3):517–524, 1977.

[35] S. Lin and B. W. Kernighan. An Effective Heuristic Algorithm for the Traveling Salesman Problem. *Operations Research*, 21:496–516, 1973.

[36] John P. Norback and Robert F. Love. Geometric Approaches to Solving the Traveling Salesman Problem. *Management Science*, 23(11):1208–1223, 1977.

[37] Charles E. Noon, Guey-Mei You, and Thomas J. Chan. A Fast Lower Bound for the Minimum Cost Perfect 2-Matching Linear Program. *American journal of mathematical and management sciences*, 13:357–370, 1993.

[38] M. M. Solomon. On the Worst-Case Performance of Some Heuristics for the Vehicle Routing and Scheduling Problem with Time Window Constraints. *Networks*, 16:161–174, 1986.

[39] N. Christofides, A. Mingozzi, and P. Toth. Exact Algorithms for the Vehicle Routing Problem, Based on Spanning Tree and Shortest Path Relaxations. *Mathematical Programming*, 20:255–282, 1981.

[40] Joseph M. Tama. Polyhedral Aspects of Scheduling Problems with an Application to the Time Constrained Traveling Salesman Problem. *Ph.D. Dissertatation.*

[41] Marius M. Solomon and Jacques Desrosiers. Time Window Constrained Routing and Scheduling Problems. *Transportation Science*, 22(1):1–13, 1988.

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

[43] Marius M. Solomon. Algorithms for Vehicle Routing and Scheduling Problems with Time Window Constraints. *Operations Research*, 35(2):254–265, 1987.

[44] A. W. J. Kolen, A. H. G. Rinnooy Kan, and H. W. J. M. Trienekens. Vehicle Routing with Time Windows. *Operations Research*, 35(2):266–273, 1987.

[45] Martin Desrochers, Jacques Desrosiers, and Marius Solomon. A New Optimizing Algorithm for the Vehicle Routing Problem with Time Windows. *Operations Research*, 40(2):342–354, 1992.

[46] Jacques Desrosiers, Francois Soumis, and Martin Desrochers. Routing with Time Windows by Column Generation. *Networks*, 14:545–565, 1984.

[47] H. R. G. Van Landeghem. A Bi-Criteria for the Vehicle Routing Problem with Time Windows. *European Journal of Operational Research*, 36:217–226, 1988.

[48] J. A. Ferland and L. Fortin. Vehicles Scheduling with Sliding Time Windows. *European Journal of Operational Research*, 38:213–225, 1989.

[49] B. Ahn and J. Shin. Vehicle-Routing with Time Windows and Time-Varying Congestion. *Journal of the Operational Research Society*, 42:393–400, 1991.

[50] J. B. Atkinson. A Greedy Look-Ahead Heuristic for Combinatorial Optimization: An Application to Vehicle Scheduling with Time Windows. *Journal of the Operational Research Society*, 45:673–684, 1994.

[51] J. Potvin and J. Rousseau. A Parallel Route Building Algorithm for the Vehicle Routing and Scheduling Problem with Time Windows. *European Journal of Operational Research*, 66:331–340, 1993.

[52] Ulrich Dergis and Gregor Grabenbauer. Intime - A New Heuristic Approach to the Vehicle Routing Problem with Time Windows, with a Bakery Fleet Case. *American journal of mathematical and management sciences*, 13:249–266, 1993.

[53] Julien Bramel, Chung-Lun Li, and David Simchi-Levi. Probabilistic Analysis of a Vehicle Routing Problem with Time Windows. *American journal of mathematical and management sciences*, 13:267–322, 1993.

[54] Sam R. Thangiah, Ibrahim H. Osman, Rajini Vinayagamoorthy, and Tong Sun. Algorithms for the Vehicle Routing Problem with Time Deadlines. *American journal of mathematical and management sciences*, 13:323–355, 1993.

[55] Y. Dumas, J. Desrosiers, and F. Soumis. The Pickup and Delivery problem with Time Windows. *European Journal of Operational Research*, 54:7–22, 1991.

[56] Thomas R. Sexton. The Single Vehicle Many to Many Routing and Scheduling Problem. *Ph.D. Dissertation, Dept. of Appl. Math. and Stat., SUNY, Stony Brook*, 1979.

[57] Harilaos N. Psaraftis. An Exact Algorithm for the Single Vehicle Many-to-Many Dial-a-Ride Problem with Time Windows. *Transportation Science*, 17(3):351–357, 1983.

[58] Harilaos N. Psaraftis. A Dynamic Programming Solution to the Single Many-to-Many Immediate Request Dial-a-Ride Problem. *Transportation Science*, 14(2):130–154, 1980.

[59] Jacques Desrosiers, Yvan Dumas, and Francois Soumis. A Dynamic Programming Solution of the Large-Scale Single-Vehicle Dial-a-Ride Problem with Time Windows. *American Journal of Mathematical and Management Sciences*, 6(3 and 4):301–325, 1986.

[60] L. J. J. Van Der Bruggen, J. K. Lenstra, and P. C. Schuur. Variable-Depth Search for the Single-Vehicle Pickup and Delivery Problem with Time Windows. *Transportation Science*, 27:298–311, 1993.

[61] Martin Desrochers and Francois Soumis. A Generalized Permanant Labelling Algorithm for the Shortest Path Problem with Time Windows. *INFOR*, 26(3):191–212, 1988.

[62] M. Desrochers and F. Soumis. A Reoptimization Algorithm for the Shortest Path Problem with Time Windows. *European Journal of Operational Research*, 35:242–254, 1988.

[63] K. R. Fox, B. Gavish, and S. C. Graves. An n-Constraint Formulation of the (Time-Dependent) Traveling Salesman Problem. *Operations Research*, 28:1018–1021, 1980.

[64] A. Lucena. Time-Dependent Traveling Salesman Problem-The Deliveryman Case. *Networks*, 20:753–763, 1990.

[65] M. G. Kantor and M. B. Rosenwein. The Orienteering Problem with Time Windows. *Journal of the Operational Research Society*, 43:629–635, 1992.

[66] N. Balakrishnan. Simple Heuristics for the Vehicle Routing Problem with Soft Time Windows. *Journal of the Operational Research Society*, 44:279–287, 1993.

[67] Y. A. Koskosidis, W. B. Powell, and M.M. Solomon. An Optimization-Based Heuristic for Vehicle Routing and Scheduling with Soft Time Windows. *Transportation Science*, 26:69–85, 1992.

[68] Jacques Desrosiers, Michel Sauve, and Francois Soumis. Lagrangian Relaxation Methods for Solving the Minimum Fleet Size Multiple Traveling Salesman Problem with Time Windows. *Management Science*, 34(8):1005–1022, 1988.

[69] Charles E. Noon. Class Notes for Integer Programming. 1987.

[70] Katta G. Murty. *Linear Programming.* John Wiley and Sons, Inc, 1936.

[71] Charles E. Noon and Katta G. Murty. Class Notes for Networks. *Based on book by K.G. Murty*, 1989.

Appendices

Appendix A

Pseudocodes

This section of the appendices provides pseudocodes for the procedures adopted for generating X-Cuts and Z-Cuts. A pseudocode that describes Baker's method for the Traveling Salesman Makespan Problem with Time Windows is also provided. All comments are written in regular font while the program statements are written in italics.

A pseudocode for generating the X-Cuts is described below. Let n be the number of nodes with node 1 being the depot. Arrays EARLY and LATE define the time windows for the nodes. Array TIME contains the service time at the nodes while array TRVLTIME refers to the time of travel between the nodes and TOUR describes the current sequence, i.e., the current solution to the master problem. Array XCUTLIST refers to the list containing the X-Cuts while xcut_no refers to the current X-Cut number.

Routine Generate-X-Cuts(EARLY,LATE,TIME,TRVLTIME,
TOUR,XCUTLIST,xcut_no)

Check for infeasibility in the sequence presented.

do i = 1,n

beg = i

sum = EARLY(TOUR(i))

diff = LATE(TOUR(i))-sum

j = i

do while ((diff $\geq$ *0) AND (j* $\leq$ *n-1))*

j = j + 1

sum = sum+TIME(TOUR(j-1))+

TRVLTIME(TOUR(j-1),TOUR(j))

if (sum < EARLY(TOUR(j))) then

sum = EARLY(TOUR(j))

end if

diff = LATE(TOUR(j)) -sum

end do

if (diff < 0) then

ctr = ctr + 1

TEMP(ctr,1) = beg

TEMP(ctr,2) = j

end if

end do

fin = TEMP(1,2)

i = 1

do while (TEMP(i,1) $\neq$ *0)*

j = i

If infeasibility is detected, then form the X-Cuts.

do while (TEMP(j,2) = fin)

beg = TEMP(j,1)

j = j + 1

end do

Call Routine FORM-CUTX

fin = TEMP(j,2)

i = j

end do

Routine FORM-CUTX(TOUR,beg,fin,XCUTLIST,xcut_no)

Add to XCUTLIST cut of the form

$x_{TOUR(beg),TOUR(beg+1)} + x_{TOUR(beg+1),TOUR(beg+2)}+$

$\ldots + x_{TOUR(fin-1),TOUR(fin)} \leq (fin - beg) - 1$

increment xcut_no

return

The pseudocode that describes the procedure for generating Z-Cuts is described below. The array ZCUTLIST refers to the list of Z-Cuts while zcut_no refers to the current Z-Cut number.

Routine Generate-Z-Cuts(EARLY,LATE,TIME,TRVLTIME,
TOUR,ZCUTLIST,zcut_no)

The sequence is checked for feasibility. The procedure to detect infeasible subsequences in the tour is the same as in the case of the X-Cuts. The routine that generates the actual Z-Cuts is provided below.

Routine FORM-CUTZ(TOUR,beg,fin,ZCUTLIST,zcut_no,EARLY,
LATE,TRVLTIME,TIME)

nd1 = TOUR(beg)
nd2 = TOUR(fin)
if ((fin-beg) = 2) then
Generate Positional Cuts
do for all combinations of i,j $\neq$ *nd1*
sum = EARLY(nd1)+TIME(nd1)+TRVLTIME(nd1,i)+
TIME(i)+TRVLTIME(i,j)
if (sum > LATE(j)) then
ctr = ctr + 1
if (ctr = no. of combinations of i,j $\neq$ *nd1) then*
create cut of form
$z_{nd1,1} + z_{nd1,2} + \ldots + z_{nd1,n-2} \leq 0$
increment zcut_no
end if
sum = EARLY(nd1)+TIME(nd1)+TRVLTIME(nd1,nd2)
if (sum > LATE(nd2)) then
Generate Precedence Cuts
add to ZCUTLIST cuts of the form

$z_{nd2,2} - z_{nd1,3} - z_{nd1,4} - \ldots - z_{nd1,n} \leq 0$

$z_{nd2,3} - z_{nd1,4} - z_{nd1,5} - \ldots - z_{nd1,n} \leq 0$

$\vdots$

$z_{nd2,n-1} - z_{nd1,n} \leq 0$

$z_{nd2,n} \leq 0$

Addition of each cut is followed by increment of the zcut_no

else

Generate Simple Cuts

Add to ZCUTLIST cut of the form

$z_{TOUR(beg),beg} + z_{TOUR(beg+1),beg+1} + \ldots +$

$z_{TOUR(fin),fin} \leq fin - beg$

increment zcut_no

end if

return

The pseudocode for Baker's method of solving the Traveling Salesman Makespan Problem with Time Windows is provided below

Program for Baker's Method

Create a list of fixed arcs A-LIST

Create a list of reversible arcs REV

Let REV-NO indicate the number of reversible arcs

Let N be the number of nodes in the graph

Read in incumbent tour TOUR and value INC

Set number of live vertices V = 1

As a first step, a longest path problem is solved over the fixed arcs present at the root vertex ie., the arcs in A-LIST

Call L-PATH(TOUR,K,VAL,A-LIST)

Let K be the number of nodes on the resultant longest path and let VAL be the cost of the resultant longest path

Set DEPTH(ROOT VERTEX) = 1

If (K = N+1)

if (VAL <= INC) then

Change incumbent tour and incumbent value

end if

end if

do m = 2,REV-NO+1

Let the nodes of the arc corresponding to REV(m-1) be

p and q. Then, two branches can be developed

for every live vertex with DEPTH(VERTEX) = m-1.

One branch adds the arc (p, q) to A-LIST

corresponding to VERTEX, while the other branch

adds the arc (q, p) to A-LIST corresponding to VERTEX.

Add arc (p, q) to A-LIST corresponding

to VERTEX with DEPTH(VERTEX) = m-1

Call L-PATH(TOUR,K,VAL,A-LIST)

if (resultant path has K < N+1) then

if (VAL < INC) then

Set V = V + 1; Set DEPTH(VERTEX) = m

Record all arcs present at this node.

end if

else

if (VAL < INC) then

record new incumbent tour TOUR and value

INC and fathom

end if

end if

Add arc (q,p) to A-LIST corresponding

to VERTEX with DEPTH(VERTEX) = m-1

Call L-PATH(TOUR,K,VAL,A-LIST)

if (resultant path has K < N+1) then

if VAL < INC) then

Set V = V + 1; Set DEPTH(VERTEX) = m

Record all arcs present at this node.

end if

```
    else
        if (VAL < INC) then
            record new incumbent tour TOUR and value
                INC and fathom
        end if
    end if
end do
```

Appendix B

Problem Set I

In the first eight problems, the time of travel between nodes i and j is the Euclidean value, i.e.,

$$d_{i,j} = [(x_i - x_j)^2 + (y_i - y_j)^2]^{1/2},$$

where x_i refers to the X-Coordinate and y_i to the Y-Coordinate of node i. The values of $c_{i,j}$ are defined for each individual problem. Also, e_i, l_i and s_i refer to the beginning of the time window, end of the time window and service time respectively for node i. If service time has not been specified then, it is assumed to be 0. In the case of the last four problems, $d_{i,j}$ is defined by the relationship

$$d_{i,j} = 0.18181818 * [(x_i - x_j)^2 + (y_i - y_j)^2]^{1/2}.$$

The cost of travel $c_{i,j}$ is considered to be equal to the time of travel $d_{i,j}$.

Table B.1: Problem R10-1

Node	1	2	3	4	5	6	7	8	9	10
X-Coord	2	72	45	66	8	87	6	32	73	47
Y-Coord	3	45	79	56	98	83	23	25	98	6
Early	0	0	33	64	136	216	317	343	427	523
Late	759	174	217	248	320	400	501	527	611	707

Table B.2: Problem R10-2

Node	1	2	3	4	5	6	7	8	9	10
X-Coord	2	72	45	66	8	87	6	32	73	47
Y-Coord	6	90	158	112	196	165	46	50	195	11
Early	0	0	43	94	196	281	425	451	602	788
Late	1117	248	321	372	474	559	703	729	880	1066

Table B.3: Problem R10-3

Node	1	2	3	4	5	6	7	8	9	10
X-Coord	3	45	79	56	98	83	23	25	98	6
Y-Coord	5	17	13	46	88	79	40	18	23	64
Early	0	0	9	49	108	125	197	219	292	393
Late	599	113	147	187	246	263	335	357	430	531

Table B.4: Problem R10-4

Node	1	2	3	4	5	6	7	8	9	10
X-Coord	2	72	45	66	8	87	6	32	73	47
Y-coord	2	9	62	11	53	35	65	78	85	26
Early	0	0	17	72	144	225	311	340	382	446
Late	730	182	241	296	368	449	535	564	606	670

Table B.5: Problem R12-1

Node	1	2	3	4	5	6
X-Coord	2	72	45	66	8	87
Y-Coord	6	90	158	112	196	165
Early	0	0	0	29	131	216
Late	1329	313	386	437	539	624
Node	7	8	9	10	11	12
X-Coord	6	32	73	47	21	5
Y-Coord	46	50	195	11	45	93
Early	360	386	537	723	766	817
Late	768	794	945	1131	1174	1225

Table B.6: Problem R12-2

Node	1	2	3	4	5	6
X-Coord	2	72	45	66	8	87
Y-Coord	6	90	158	112	196	165
Early	0	0	0	49	151	236
Late	1307	293	366	417	519	604
Node	7	8	9	10	11	12
X-Coord	6	32	73	47	21	5
Y-Coord	46	50	195	11	45	93
Early	380	406	557	743	786	837
Late	748	774	925	1111	1154	1205

Table B.7: Problem R12-3

Node	1	2	3	4	5	6
X-Coord	2	72	45	66	8	87
Y-Coord	3	45	79	56	98	83
Early	0	0	0	8	80	160
Late	874	230	273	304	376	456
Node	7	8	9	10	11	12
X-Coord	6	32	73	47	21	5
Y-Coord	23	25	98	6	23	46
Early	261	287	371	467	498	526
Late	557	583	667	763	794	822

Table B.8: Problem R12-4

Node	1	2	3	4	5	6
X-Coord	2	72	45	66	8	87
Y-Coord	2	9	62	11	53	35
Early	0	0	30	85	157	238
Late	786	169	228	283	355	436
Node	7	8	9	10	11	12
X-Coord	6	32	73	47	21	5
Y-Coord	65	78	85	26	72	26
Early	324	353	395	459	512	561
Late	552	551	593	657	710	759

Table B.9: Node Coordinates - Problems SK141-1, SK141-2 and SK141-3

Node	1	2	3	4	5	6	7
X-Coord	250	269	355	378	376	467	421
Y-Coord	200	262	236	203	160	67	237
Node	8	9	10	11	12	13	14
X-Coord	458	447	484	391	383	363	295
Y-Coord	218	189	179	196	181	187	235

Table B.10: Time Windows and Service Time - Problem SK141-1

Node	1	2	3	4	5	6	7
Early	0	0	0	0	0	0	0
Late	540	502	510	487	512	487	472
Time	0	26	10	30	4	7	36
Node	8	9	10	11	12	13	14
Early	220	257	272	318	295	420	397
Late	486	479	481	400	499	504	512
Time	16	25	16	114	17	15	18

Table B.11: Time Windows and Service Time - Problem SK141-2

Node	1	2	3	4	5	6	7
Early	0	0	0	0	0	0	0
Late	540	502	440	435	439	457	430
Time	0	26	10	30	4	7	36
Node	8	9	10	11	12	13	14
Early	220	257	272	318	295	420	397
Late	452	440	444	400	499	504	512
Time	16	25	16	114	17	15	18

Table B.12: Time Windows and Service Time - Problem SK141-3

Node	1	2	3	4	5	6	7
Early	0	0	0	0	0	0	0
Late	540	502	510	487	512	457	430
Time	0	26	10	30	4	7	36
Node	8	9	10	11	12	13	14
Early	220	257	272	318	295	420	397
Late	452	440	444	400	499	504	512
Time	16	25	16	114	17	15	18

Table B.13: Problem SK142-1

Node	1	2	3	4	5	6	7
X-Coord	250	228	313	278	273	240	267
Y-Coord	200	296	382	374	349	326	316
Early	0	0	0	287	281	247	320
Late	540	504	346	475	505	503	513
Time	0	18	159	33	8	14	6
Node	8	9	10	11	12	13	14
X-Coord	324	293	352	399	304	286	215
Y-Coord	295	269	271	122	202	207	204
Early	315	396	430	315	291	336	442
Late	510	513	513	499	519	528	530
Time	8	12	4	10	11	5	4

Appendix C

Problem Set II

The problems in this set all have the same number of nodes and node coordinates. However, the width and overlap of the time windows gradually increase over the problem set. The node coordinates are first specified. The details regarding time windows are given for each individual problem in the set. The service times at the nodes are all assumed to be 0. In all these problems the time of travel between the nodes i and j is defined as $d_{i,j} = [(x_i - x_j)^2 + (y_i - y_j)^2]^{1/2}$ and the cost of travel is defined as $c_{i,j} = d_{i,j}$.

Table C.1: Node Coordinates for Problems BAK1 - BAK20

Node	1	2	3	4	5	6	7	8	9	10
X-Coord	2	72	45	66	8	87	6	32	73	47
Y-Coord	3	45	79	56	98	83	23	25	98	6

Table C.2: Time Windows for Problem BAK1

Node	1	2	3	4	5	6	7	8	9	10
Early	0	41	79	107	172	244	355	358	434	520
Late	695	107	146	174	238	310	401	425	500	587

Table C.3: Time Windows for Problem BAK2

Node	1	2	3	4	5	6	7	8	9	10
Early	0	34	72	100	165	237	328	351	427	513
Late	703	114	153	181	245	317	408	432	507	594

Table C.4: Time Windows for Problem BAK3

Node	1	2	3	4	5	6	7	8	9	10
Early	0	7	46	74	139	211	302	325	401	487
Late	731	140	179	207	272	344	435	458	534	620

Table C.5: Time Windows for Problem BAK4

Node	1	2	3	4	5	6	7	8	9	10
Early	0	6	45	73	137	209	300	324	399	486
Late	732	142	180	208	273	345	436	459	535	621

Table C.6: Time Windows for Problem BAK5

Node	1	2	3	4	5	6	7	8	9	10
Early	0	2	41	68	133	205	296	320	395	482
Late	737	146	182	212	277	349	440	463	539	625

Table C.7: Time Windows for Problem BAK6

Node	1	2	3	4	5	6	7	8	9	10
Early	0	0	35	63	128	200	291	314	390	476
Late	743	151	190	218	283	355	446	469	545	631

Table C.8: Time Windows for Problem BAK7

Node	1	2	3	4	5	6	7	8	9	10
Early	0	0	29	57	122	194	285	308	384	470
Late	749	157	196	223	288	360	451	475	550	637

Table C.9: Time Windows for Problem BAK8

Node	1	2	3	4	5	6	7	8	9	10
Early	0	0	24	52	117	189	280	303	379	465
Late	755	162	201	229	294	366	457	480	556	642

Table C.10: Time Windows for Problem BAK9

Node	1	2	3	4	5	6	7	8	9	10
Early	0	0	21	49	114	186	277	300	376	462
Late	758	165	204	232	297	369	459	483	558	645

Table C.11: Time Windows for Problem BAK10

Node	1	2	3	4	5	6	7	8	9	10
Early	0	0	18	46	111	183	274	297	373	459
Late	760	168	207	234	299	371	462	486	561	648

Table C.12: Time Windows for Problem BAK11

Node	1	2	3	4	5	6	7	8	9	10
Early	0	0	16	44	108	180	271	295	370	457
Late	763	171	209	237	302	374	465	488	564	650

Table C.13: Time Windows for Problem BAK12

Node	1	2	3	4	5	6	7	8	9	10
Early	0	0	14	42	107	179	270	293	369	455
Late	765	172	211	239	304	376	467	490	566	652

Table C.14: Time Windows for Problem BAK13

Node	1	2	3	4	5	6	7	8	9	10
Early	0	0	13	41	106	178	268	292	367	454
Late	766	173	212	240	305	377	468	491	567	653

Table C.15: Time Windows for Problem BAK14

Node	1	2	3	4	5	6	7	8	9	10
Early	0	0	7	35	100	172	263	286	362	448
Late	772	179	218	246	310	382	473	497	572	659

Table C.16: Time Windows for Problem BAK15

Node	1	2	3	4	5	6	7	8	9	10
Early	0	0	6	34	99	171	262	285	361	447
Late	774	180	219	247	312	384	475	498	574	660

Table C.17: Time Windows for Problem BAK16

Node	1	2	3	4	5	6	7	8	9	10
Early	0	0	2	30	94	166	257	281	356	443
Late	778	185	223	251	316	388	479	502	578	664

Table C.18: Time Windows for Problem BAK17

Node	1	2	3	4	5	6	7	8	9	10
Early	0	0	0	24	89	161	252	275	351	437
Late	784	190	229	257	321	393	484	508	583	670

Table C.19: Time Windows for Problem BAK18

Node	1	2	3	4	5	6	7	8	9	10
Early	0	0	0	21	86	158	249	272	348	434
Late	787	193	232	259	324	396	487	511	586	673

Table C.20: Time Windows for Problem BAK19

Node	1	2	3	4	5	6	7	8	9	10
Early	0	0	0	19	83	155	246	270	345	432
Late	790	196	234	262	327	399	490	513	589	675

Table C.21: Time Windows for Problem BAK20

Node	1	2	3	4	5	6	7	8	9	10
Early	0	0	0	8	72	144	235	259	334	421
Late	802	207	245	273	338	410	501	524	600	686

Appendix D

Problem Set III

This problem set has been generated using the method for Symmetric Euclidean matrices described by Langevin, *et al.* The time windows for the problem have also been derived based on the procedure described by Langevin *et al.*. Details of the procedure are provided in Chapter 2. This set contains ten problems. The cardinality of the node set in the first seven problems is twenty while the cardinality of the node set in the last three problems is fifteen.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1																			
2	43																		
3	16	43																	
4	40	52	55																
5	75	34	76	69															
6	70	66	83	31	67														
7	72	30	67	79	26	86													
8	96	53	93	98	31	98	25												
9	80	37	77	84	22	87	10	16											
10	93	50	92	88	19	84	30	17	21										
11	101	60	103	88	26	77	47	34	38	18									
12	63	34	70	46	25	42	46	56	45	43	42								
13	28	17	31	38	48	58	47	70	54	66	73	38							
14	58	17	54	68	29	78	14	39	23	40	54	41	34						
15	79	36	78	75	7	74	21	24	16	14	26	32	51	28					
16	53	54	66	14	64	17	79	94	81	82	80	39	44	69	70				
17	35	40	20	70	73	95	58	83	68	86	99	73	37	46	73	79			
18	74	41	79	55	22	46	47	52	44	37	33	10	48	45	29	47	81		
19	47	7	48	49	29	60	30	51	36	46	55	26	19	19	32	50	48	34	
20	98	60	102	82	27	70	51	42	43	25	9	37	72	56	30	73	100	27	54

Table C.1: Travel Time Matrix for Problems C20-1-20, C20-1-30 and C20-1-40

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1																			
2	77																		
3	102	33																	
4	28	64	95																
5	103	53	77	77															
6	20	57	84	21	85														
7	29	67	98	4	79	24													
8	87	44	73	61	16	69	63												
9	84	8	26	72	57	64	75	49											
10	55	30	62	36	51	35	39	35	38										
11	50	33	52	49	79	33	52	65	37	32									
12	58	19	45	50	64	39	53	51	25	22	15								
13	40	53	86	14	63	27	16	47	61	24	45	42							
14	59	64	69	72	114	51	76	100	66	67	35	49	74						
15	42	82	114	22	83	42	18	68	90	52	70	70	28	83					
16	36	68	99	4	78	25	1	63	75	39	53	54	16	77	17				
17	38	60	92	9	68	27	10	53	68	31	50	48	7	77	22	10			
18	45	84	116	24	83	45	21	69	92	54	73	72	30	96	3	20	24		
19	69	29	36	67	82	52	71	71	30	44	19	22	62	36	89	72	68	91	
20	96	20	16	84	61	76	57	57	13	50	48	37	74	72	102	88	80	104	36

Table C.2: Travel Time Matrix for Problems C20-2-20 and C20-2-30

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
1																			
2	76																		
3	119	72																	
4	54	48	65																
5	88	12	70	57															
6	66	15	67	33	25														
7	25	51	99	37	63	41													
8	69	83	73	37	92	69	64												
9	74	11	64	39	18	8	49	74											
10	68	25	57	25	33	12	45	59	15										
11	99	31	46	55	25	34	75	84	27	32									
12	12	67	107	42	79	56	17	60	64	57	88								
13	80	103	87	56	111	88	80	20	93	79	103	72							
14	58	35	62	13	44	20	37	49	26	12	44	47	69						
15	100	25	80	72	15	40	75	107	33	48	34	91	126	59					
16	20	56	103	39	68	46	5	64	54	49	79	12	79	40	80				
17	79	11	83	57	15	25	54	94	22	36	39	71	113	45	21	59			
18	59	26	66	22	36	11	36	58	18	9	40	48	78	9	51	40	35		
19	79	73	53	32	79	59	67	20	62	48	67	68	37	41	94	68	84	50	
20	119	80	13	65	79	73	101	66	71	62	57	107	78	65	90	104	91	70	47

Table C.3: Travel Time Matrix for Problems C20-3-20 and C20-3-30

Table D.4: Travel Time Matrix for Problems C15-1-20, C15-1-30 and C15-1-40

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1	∞													
2	76													
3	119	72												
4	54	48	65											
5	88	12	70	57										
6	66	15	67	33	25									
7	25	51	99	37	63	41								
8	69	83	73	37	92	69	64							
9	74	11	64	39	18	8	49	74						
10	68	25	57	25	33	12	45	59	15					
11	99	31	46	55	25	34	75	84	27	32				
12	12	67	107	42	79	56	17	60	64	57	88			
13	80	103	87	56	111	88	80	20	93	79	103	72		
14	58	35	62	13	44	20	37	49	26	12	44	47	69	
15	100	25	80	72	15	40	75	107	33	48	34	91	126	59

Table D.5: Time Windows for Problem C20-1-20

Node	1	2	3	4	5	6	7	8	9	10
Early	0	70	688	470	115	508	183	152	337	119
Late	838	96	714	510	119	534	213	154	367	153
Node	11	12	13	14	15	16	17	18	19	20
Early	582	235	28	57	171	749	405	307	33	277
Late	614	253	28	75	183	785	435	309	61	285

Table D.6: Time Windows for Problem C20-2-20

Node	1	2	3	4	5	6	7	8	9	10
Early	0	170	283	27	446	120	41	716	252	155
Late	825	210	291	29	450	130	63	738	270	165
Node	11	12	13	14	15	16	17	18	19	20
Early	335	208	81	615	50	23	26	514	226	370
Late	343	210	115	639	90	41	58	548	236	404

Table D.7: Time Windows for Problem C20-3-20

Node	1	2	3	4	5	6	7	8	9	10
Early	0	152	295	59	533	107	691	633	214	129
Late	740	172	305	89	561	143	715	645	240	145
Node	11	12	13	14	15	16	17	18	19	20
Early	249	30	386	87	190	20	155	92	350	452
Late	259	34	394	123	198	20	191	100	356	484

Table D.8: Time Windows for Problem C20-1-30

Node	1	2	3	4	5	6	7	8	9	10
Early	0	63	681	461	115	501	176	151	330	110
Late	848	103	721	519	119	541	220	155	374	162
Node	11	12	13	14	15	16	17	18	19	20
Early	573	230	28	53	167	739	398	306	25	275
Late	623	258	28	79	187	795	442	310	69	287

Table D.9: Time Windows for Problem C20-2-30

Node	1	2	3	4	5	6	7	8	9	10
Early	0	161	281	27	444	118	35	719	247	153
Late	831	219	293	29	452	132	69	744	275	167
Node	11	12	13	14	15	16	17	18	19	20
Early	332	207	73	609	41	19	18	506	224	362
Late	346	211	123	645	99	45	66	556	238	412

Table D.10: Time Windows for Problem C20-3-30

Node	1	2	3	4	5	6	7	8	9	10
Early	0	148	293	51	526	98	684	631	208	125
Late	747	176	307	97	568	152	722	647	246	149
Node	11	12	13	14	15	16	17	18	19	20
Early	246	29	385	78	188	20	147	91	349	444
Late	262	35	395	132	200	20	199	101	357	492

Table D.11: Time Windows for Problem C20-1-40

Node	1	2	3	4	5	6	7	8	9	10
Early	0	57	674	451	114	495	169	151	322	101
Late	857	109	728	529	120	547	227	155	382	171
Node	11	12	13	14	15	16	17	18	19	20
Early	565	225	27	48	164	730	391	306	18	273
Late	631	263	29	84	190	804	449	310	76	289

Table D.12: Time Windows for Problem C15-1-20

Node	1	2	3	4	5	6	7	8
Early	0	118	295	60	506	143	25	359
Late	705	154	303	64	516	159	25	385
Node	9	10	11	12	13	14	15	
Early	107	72	175	427	619	95	201	
Late	143	102	195	437	625	103	237	

Table D.13: Time Windows for Problem C15-1-30

Node	1	2	3	4	5	6	7	8
Early	0	109	293	59	504	139	25	353
Late	706	163	305	65	518	163	25	391
Node	9	10	11	12	13	14	15	
Early	98	64	171	424	618	94	193	
Late	152	110	199	440	626	104	245	

Table D.14: Time Windows for Problem C15-1-40

Node	1	2	3	4	5	6	7	8
Early	0	100	292	58	501	136	25	346
Late	707	172	306	66	521	166	25	398
Node	9	10	11	12	13	14	15	
Early	89	56	166	421	617	92	184	
Late	161	118	204	443	627	106	254	

Appendix E

Problem Set IV

This problem set contains problems of node size ranging from fifteen nodes to thirty five nodes. As before, the time of travel between nodes i and j is defined as $d_{i,j} = [(x_i - x_j)^2 + (y_i - y_j)^2]^{1/2}$ and $c_{i,j} = d_{i,j}$. The service time at the nodes for all problems except SK20 is zero. The remaining details are as given below.

Table E.1: Problem R15-1

Node	1	2	3	4	5	6	7	8
X-Coord	2	72	45	66	8	87	6	32
Y-Coord	3	45	79	56	98	83	23	25
Early	0	1	44	75	147	227	328	354
Late	1032	163	206	237	309	389	490	516
Node	9	10	11	12	13	14	15	
X-Coord	73	47	21	5	74	73	98	
Y-Coord	98	6	23	46	21	22	83	
Early	438	534	565	593	666	667	733	
Late	600	696	727	755	828	829	895	

Table E.2: Problem R15-2

Node	1	2	3	4	5	6	7	8
X-Coord	2	72	45	66	8	87	6	32
Y-Coord	3	45	79	56	98	83	23	25
Early	0	0	3	34	106	186	287	313
Late	1079	204	247	278	350	430	531	557
Node	9	10	11	12	13	14	15	
X-Coord	73	47	21	5	74	73	98	
Y-Coord	98	6	23	46	21	22	83	
Early	397	493	524	552	625	626	692	
Late	641	737	768	796	869	870	936	

Table E.3: Problem R15-3

Node	1	2	3	4	5	6	7	8
X-Coord	2	9	62	11	53	35	65	78
Y-Coord	6	90	158	112	196	165	46	50
Early	0	0	0	33	127	163	286	300
Late	1546	290	376	445	539	575	698	712
Node	9	10	11	12	13	14	15	
X-Coord	85	26	72	26	47	48	41	
Y-Coord	195	11	45	93	42	45	166	
Early	445	638	695	761	816	819	940	
Late	857	1050	1107	1173	1228	1231	1352	

Table E.4: Problem R15-4

Node	1	2	3	4	5	6	7	8
X-Coord	2	9	62	11	53	35	65	78
Y-Coord	6	90	158	112	196	165	46	50
Early	0	0	0	10	104	140	263	277
Late	1573	313	399	468	562	598	721	735
Node	9	10	11	12	13	14	15	
X-Coord	85	26	72	26	47	48	41	
Y-Coord	195	11	45	93	42	45	166	
Early	422	615	672	738	793	796	917	
Late	880	1073	1130	1196	1251	1254	1375	

Table E.5: Problem R15-5

Node	1	2	3	4	5	6	7	8
X-Coord	3	45	79	56	98	83	23	25
Y-Coord	10	35	26	91	176	158	80	36
Early	0	0	0	60	155	178	276	320
Late	1191	143	178	247	342	365	463	507
Node	9	10	11	12	13	14	15	
X-Coord	98	6	23	46	21	22	83	
Y-Coord	45	129	49	175	136	142	135	
Early	394	519	601	729	775	781	842	
Late	581	706	788	916	962	968	1029	

Table E.6: Problem R15-6

Node	1	2	3	4	5	6	7	8
X-Coord	3	45	79	56	98	83	23	25
Y-Coord	10	35	26	91	176	158	80	36
Early	0	0	0	3	98	121	219	263
Late	1256	199	234	303	398	421	519	563
Node	9	10	11	12	13	14	15	
X-Coord	98	6	23	46	21	22	83	
Y-Coord	45	129	49	175	136	142	135	
Early	337	462	544	672	718	724	785	
Late	637	762	844	972	1018	1024	1085	

Table E.7: Problem R20-1

Node	1	2	3	4	5	6	7	8	9	10
X-Coord	2	72	45	66	8	87	6	32	73	47
Y-Coord	3	45	79	56	98	83	23	25	98	6
Early	0	0	18	49	121	201	302	328	412	508
Late	1237	189	232	263	335	415	516	542	626	722
Node	11	12	13	14	15	16	17	18	19	20
X-Coord	21	5	74	73	98	65	82	67	92	49
Y-Coord	23	46	21	22	83	12	85	61	57	36
Early	539	567	640	641	707	785	860	888	913	961
Late	753	781	854	855	921	999	1074	1102	1127	1175

Table E.8: Problem R20-2

Node	1	2	3	4	5	6	7	8	9	10
X-Coord	2	72	45	66	8	87	6	32	73	47
Y-Coord	3	45	79	56	98	83	23	25	98	6
Early	0	0	0	0	68	148	249	275	359	455
Late	1293	242	285	316	388	468	569	595	679	775
Node	11	12	13	14	15	16	17	18	19	20
X-Coord	21	5	74	73	98	65	82	67	92	49
Y-Coord	23	46	21	22	83	12	85	61	57	36
Early	486	514	587	588	654	732	807	835	860	908
Late	806	834	907	908	974	1052	1127	1155	1180	1228

Table E.9: Problem R20-3

Node	1	2	3	4	5	6	7	8	9	10
X-Coord	2	9	62	11	53	35	65	78	85	26
Y-Coord	6	90	158	112	196	165	46	50	195	11
Early	0	0	0	0	48	84	207	221	366	559
Late	1956	369	455	524	618	654	777	791	936	1129
Node	11	12	13	14	15	16	17	18	19	20
X-Coord	72	26	47	48	41	39	83	14	24	42
Y-Coord	45	93	42	45	166	25	170	123	113	72
Early	616	682	737	740	861	1002	1154	1237	1251	1296
Late	1186	1252	1307	1310	1431	1572	1724	1807	1821	1866

Table E.10: Problem R20-4

Node	1	2	3	4	5	6	7	8	9	10
X-Coord	3	45	79	56	98	83	23	25	98	6
Y-Coord	10	35	26	91	176	158	80	36	45	129
Early	0	0	0	27	122	145	243	287	361	486
Late	1956	369	455	524	618	654	777	791	936	1129
Node	11	12	13	14	15	16	17	18	19	20
X-Coord	23	46	21	22	83	12	85	61	57	36
Y-Coord	49	175	136	142	135	119	176	113	42	45
Early	568	696	742	748	809	882	975	1042	1113	1134
Late	1186	1252	1307	1310	1431	1572	1724	1807	1821	1866

Table E.11: Problem R20-5

Node	1	2	3	4	5	6	7	8	9	10
X-Coord	3	45	79	56	98	83	23	25	98	6
Y-Coord	10	35	26	91	176	158	80	36	45	129
Early	0	0	0	0	46	49	167	211	285	410
Late	1517	251	286	355	450	473	571	615	689	814
Node	11	12	13	14	15	16	17	18	19	20
X-Coord	23	46	21	22	83	12	85	61	57	36
Y-Coord	49	175	136	142	135	119	176	113	42	45
Early	492	620	666	672	733	806	899	966	1037	1058
Late	896	1024	1070	1076	1137	1210	1303	1370	1441	1462

Table E.12: Problem SK20

Node	1	2	3	4	5	6	7	8	9	10
X-Coord	250	225	212	205	193	206	258	285	288	294
Y-Coord	200	229	244	254	236	173	147	135	113	86
Early	0	0	0	0	0	0	0	0	0	0
Late	540	526	523	509	522	526	508	511	510	512
Service	0	7	6	18	6	5	22	16	13	6
Node	11	12	13	14	15	16	17	18	19	20
X-Coord	275	7	5	10	22	65	50	249	333	288
Y-Coord	34	157	174	260	255	248	249	250	212	191
Early	150	0	0	120	288	30	350	380	276	310
Late	498	489	479	471	486	489	498	485	498	495
Service	11	6	16	24	11	16	5	46	27	38

Table E.13: Problem R25-1

Node	1	2	3	4	5	6	7	8	9
X-Coord	2	72	45	66	8	87	6	32	73
Y-Coord	3	45	79	56	98	83	23	25	98
Early	0	0	1	32	104	184	285	311	395
Late	1415	206	249	280	352	432	533	559	643
Node	10	11	12	13	14	15	16	17	18
X-Coord	47	21	5	74	73	98	65	82	67
Y-Coord	6	23	46	21	22	83	12	85	61
Early	491	522	550	623	624	690	768	843	871
Late	739	770	798	871	872	938	1016	1091	1119
Node	19	20	21	22	23	24	25		
X-Coord	92	49	94	31	52	33	28		
Y-Coord	57	36	48	67	60	58	37		
Early	896	944	991	1057	1079	1098	1120		
Late	1144	1192	1239	1305	1327	1346	1368		

Table E.14: Problem R25-2

Node	1	2	3	4	5	6	7	8	9
X-Coord	2	72	45	66	8	87	6	32	73
Y-Coord	3	45	79	56	98	83	23	25	98
Early	0	0	0	0	41	121	222	248	332
Late	1480	269	312	343	415	495	596	622	706
Node	10	11	12	13	14	15	16	17	18
X-Coord	47	21	5	74	73	98	65	82	67
Y-Coord	6	23	46	21	22	83	12	85	61
Early	428	459	487	560	561	627	705	780	808
Late	802	833	861	934	935	1001	1079	1154	1182
Node	19	20	21	22	23	24	25		
X-Coord	92	49	94	31	52	33	28		
Y-Coord	57	36	48	67	60	58	37		
Early	833	881	928	994	1016	1035	1057		
Late	1207	1255	1302	1368	1390	1409	1431		

Table E.15: Problem R25-3

Node	1	2	3	4	5	6	7	8	9
X-Coord	2	9	62	11	53	35	65	78	85
Y-Coord	6	90	158	112	196	165	46	50	195
Early	0	0	0	55	149	185	308	322	467
Late	2106	268	354	423	517	553	676	690	835
Node	10	11	12	13	14	15	16	17	18
X-Coord	26	72	26	47	48	41	39	83	14
Y-Coord	11	45	93	42	45	166	25	170	123
Early	660	717	783	838	841	962	1103	1255	1338
Late	1028	1085	1151	1206	1209	1330	1471	1623	1706
Node	19	20	21	22	23	24	25		
X-Coord	24	42	71	99	6	45	32		
Y-Coord	113	72	97	134	120	115	74		
Early	1352	1397	1435	1481	1575	1614	1657		
Late	1720	1765	1803	1849	1943	1982	2025		

Table E.16: Problem R25-4

Node	1	2	3	4	5	6	7	8	9
X-Coord	2	9	62	11	53	35	65	78	85
Y-Coord	6	90	158	112	196	165	46	50	195
Early	0	0	0	0	57	93	216	230	375
Late	2202	360	446	515	609	645	768	782	927
Node	10	11	12	13	14	15	16	17	18
X-Coord	26	72	26	47	48	41	39	83	14
Y-Coord	11	45	93	42	45	166	25	170	123
Early	568	625	691	746	749	870	1011	1163	1246
Late	1120	1177	1243	1298	1301	1422	1563	1715	1798
Node	19	20	21	22	23	24	25		
X-Coord	24	42	71	99	6	45	32		
Y-Coord	113	72	97	134	120	115	74		
Early	1260	1305	1343	1389	1483	1522	1565		
Late	1812	1857	1895	1941	2035	2074	2117		

Table E.17: Problem R25-5

Node	1	2	3	4	5	6	7	8	9
X-Coord	3	45	79	56	98	83	23	25	98
Y-Coord	10	35	26	91	176	158	80	36	45
Early	0	0	0	0	93	116	214	258	332
Late	1807	204	239	308	403	426	524	568	642
Node	10	11	12	13	14	15	16	17	18
X-Coord	6	23	46	21	22	83	12	85	61
Y-Coord	129	49	175	136	142	135	119	176	113
Early	457	539	667	713	719	780	853	946	1013
Late	767	849	977	1023	1029	1090	1163	1256	1323
Node	19	20	21	22	23	24	25		
X-Coord	57	36	48	67	60	58	37		
Y-Coord	42	45	5	6	137	164	93		
Early	1084	1105	1147	1166	1297	1324	1398		
Late	1394	1415	1457	1476	1607	1634	1708		

Table E.18: Problem R25-6

Node	1	2	3	4	5	6	7	8	9
X-Coord	3	45	79	56	98	83	23	25	98
Y-Coord	10	35	26	91	176	158	80	36	45
Early	0	0	0	0	31	54	152	196	270
Late	1873	266	301	370	465	488	586	630	704
Node	10	11	12	13	14	15	16	17	18
X-Coord	6	23	46	21	22	83	12	85	61
Y-Coord	129	49	175	136	142	135	119	176	113
Early	395	477	605	651	657	718	791	884	951
Late	829	911	1039	1085	1091	1152	1225	1318	1385
Node	19	20	21	22	23	24	25		
X-Coord	57	36	48	67	60	58	37		
Y-Coord	42	45	5	6	137	164	93		
Early	1022	1043	1085	1104	1235	1262	1336		
Late	1456	1477	1519	1538	1669	1696	1770		

Table E.19: Problem R30-1

Node	1	2	3	4	5	6	7	8	9	10
X-Coord	2	72	45	66	8	87	6	32	73	47
Y-Coord	3	45	79	56	98	83	23	25	98	6
Early	0	0	8	39	111	191	292	318	402	498
Late	1639	199	242	273	345	425	526	552	636	732
Node	11	12	13	14	15	16	17	18	19	20
X-Coord	21	5	74	73	98	65	82	67	92	49
Y-Coord	23	46	21	22	83	12	85	61	57	36
Early	529	557	630	631	697	775	850	878	903	951
Late	763	791	864	865	931	1009	1084	1112	1137	1185
Node	21	22	23	24	25	26	27	28	29	30
X-Coord	94	31	52	33	28	63	60	73	78	52
Y-Coord	48	67	60	58	37	23	30	97	15	5
Early	998	1064	1086	1105	1127	1165	1173	1241	1323	1351
Late	1232	1298	1320	1339	1361	1399	1407	1475	1557	1585

Table E.20: Problem R30-2

Node	1	2	3	4	5	6	7	8	9	10
X-Coord	2	72	45	66	8	87	6	32	73	47
Y-Coord	3	45	79	56	98	83	23	25	98	6
Early	0	9	52	83	155	235	336	362	446	542
Late	1593	155	198	229	301	381	482	508	592	688
Node	11	12	13	14	15	16	17	18	19	20
X-Coord	21	5	74	73	98	65	82	67	92	49
Y-Coord	23	46	21	22	83	12	85	61	57	36
Early	573	601	674	675	741	819	894	922	947	995
Late	719	747	820	821	887	965	1040	1068	1093	1141
Node	21	22	23	24	25	26	27	28	29	30
X-Coord	94	31	52	33	28	63	60	73	78	52
Y-Coord	48	67	60	58	37	23	30	97	15	5
Early	1042	1108	1130	1149	1171	1209	1217	1285	1367	1395
Late	1188	1254	1276	1295	1317	1355	1363	1431	1531	1541

Table E.21: Problem R30-3

Node	1	2	3	4	5	6	7	8	9	10
X-Coord	2	9	62	11	53	35	65	78	85	26
Y-Coord	6	90	158	112	196	165	46	50	195	11
Early	0	0	57	126	220	256	379	393	538	731
Late	2464	198	284	353	447	483	606	620	765	958
Node	11	12	13	14	15	16	17	18	19	20
X-Coord	72	26	47	48	41	39	83	14	24	42
Y-Coord	45	93	42	45	166	25	170	123	113	72
Early	788	854	909	912	1033	1174	1326	1409	1423	1468
Late	1015	1081	1136	1139	1260	1401	1553	1636	1650	1695
Node	21	22	23	24	25	26	27	28	29	30
X-Coord	71	99	6	45	32	93	95	35	46	79
Y-Coord	97	134	120	115	74	46	60	193	29	10
Early	1506	1552	1646	1685	1728	1795	1809	1955	2119	2157
Late	1733	1779	1873	1912	1955	2022	2036	2182	2346	2384

Table E.22: Problem R30-4

Node	1	2	3	4	5	6	7	8	9	10
X-Coord	2	9	62	11	53	35	65	78	85	26
Y-Coord	6	90	158	112	196	165	46	50	195	11
Early	0	0	0	35	129	165	288	302	447	640
Late	2558	288	374	443	537	573	696	710	855	1040
Node	11	12	13	14	15	16	17	18	19	20
X-Coord	72	26	47	48	41	39	83	14	24	42
Y-Coord	45	93	42	45	166	25	170	123	113	72
Early	697	763	818	821	942	1083	1235	1318	1332	1377
Late	1105	1171	1226	1229	1350	1491	1643	1726	1740	1785
Node	21	22	23	24	25	26	27	28	29	30
X-Coord	71	99	6	45	32	93	95	35	46	79
Y-Coord	97	134	120	115	74	46	60	193	29	10
Early	1415	1461	1555	1594	1637	1704	1718	1864	2028	2066
Late	1823	1869	1963	2002	2045	2112	2126	2272	2436	2474

Table E.23: Problem R30-5

Node	1	2	3	4	5	6	7	8	9	10
X-Coord	3	45	79	56	98	83	23	25	98	6
Y-Coord	10	35	26	91	176	158	80	36	45	129
Early	0	0	0	55	150	173	271	315	389	514
Late	2161	147	182	251	346	369	467	511	585	710
Node	11	12	13	14	15	16	17	18	19	20
X-Coord	23	46	21	22	83	12	85	61	57	36
Y-Coord	49	175	136	142	135	119	176	113	42	45
Early	596	724	770	776	837	910	1003	1070	1141	1162
Late	792	920	966	972	1033	1106	1199	1266	1337	1358
Node	21	22	23	24	25	26	27	28	29	30
X-Coord	48	67	60	58	37	23	30	97	15	5
Y-Coord	5	6	137	164	93	166	199	39	103	115
Early	1204	1223	1354	1381	1455	1529	1563	1736	1840	1856
Late	1400	1419	1550	1577	1651	1725	1759	1932	2036	2052

Table E.24: Problem R30-6

Node	1	2	3	4	5	6	7	8	9	10
X-Coord	3	45	79	56	98	83	23	25	98	6
Y-Coord	10	35	26	91	176	158	80	36	45	129
Early	0	0	0	0	72	95	193	237	311	436
Late	2244	225	260	329	424	447	545	589	663	788
Node	11	12	13	14	15	16	17	18	19	20
X-Coord	23	46	21	22	83	12	85	61	57	36
Y-Coord	49	175	136	142	135	119	176	113	42	45
Early	518	646	692	698	759	832	925	992	1063	1084
Late	8709	998	1044	1050	1111	1184	1277	1344	1415	1436
Node	21	22	23	24	25	26	27	28	29	30
X-Coord	48	67	60	58	37	23	30	97	15	5
Y-Coord	5	6	137	164	93	166	199	39	103	115
Early	1126	1145	1276	1303	1377	1451	1485	1658	1762	1778
Late	1478	1497	1628	1655	1729	1803	1837	2010	2114	2130

Table E.25: Problem R35-1

Node	1	2	3	4	5	6	7	8	9
X-Coord	2	72	45	66	8	87	6	32	73
Y-Coord	3	45	79	56	98	83	23	25	98
Early	0	0	43	74	146	226	327	353	437
Late	1786	164	207	238	310	390	491	517	601
Node	10	11	12	13	14	15	16	17	18
X-Coord	47	21	5	74	73	98	65	82	67
Y-Coord	6	23	46	21	22	83	12	85	61
Early	533	564	592	665	666	732	810	885	913
Late	697	728	756	829	830	896	974	1049	1077
Node	19	20	21	22	23	24	25	26	27
X-Coord	92	49	94	31	52	33	28	63	60
Y-Coord	57	36	48	67	60	58	37	23	30
Early	938	986	1033	1099	1121	1140	1162	1200	1208
Late	1102	1150	1197	1263	1285	1304	1326	1364	1372
Node	28	29	30	31	32	33	34	35	
X-Coord	73	78	52	15	21	23	100	61	
Y-Coord	97	15	5	21	21	31	20	15	
Early	1276	1358	1386	1426	1432	1442	1520	1559	
Late	1440	1522	1550	1590	1596	1606	1684	1723	

Table E.26: Problem R35-2

Node	1	2	3	4	5	6	7	8	9
X-Coord	2	72	45	66	8	87	6	32	73
Y-Coord	3	45	79	56	98	83	23	25	98
Early	0	0	10	41	113	193	294	320	404
Late	1820	197	240	271	343	423	524	550	634
Node	10	11	12	13	14	15	16	17	18
X-Coord	47	21	5	74	73	98	65	82	67
Y-Coord	6	23	46	21	22	83	12	85	61
Early	500	531	559	632	633	699	777	852	880
Late	730	761	789	862	863	929	1007	1082	1110
Node	19	20	21	22	23	24	25	26	27
X-Coord	92	49	94	31	52	33	28	63	60
Y-Coord	57	36	48	67	60	58	37	23	30
Early	905	953	1000	1066	1088	1107	1129	1167	1175
Late	1135	1183	1230	1296	1318	1337	1359	1397	1405
Node	28	29	30	31	32	33	34	35	
X-Coord	73	78	52	15	21	23	100	61	
Y-Coord	97	15	5	21	21	31	20	15	
Early	1243	1325	1353	1393	1399	1409	1487	1526	
Late	1473	1555	1583	1623	1629	1639	1717	1756	

Table E.27: Problem R35-3

Node	1	2	3	4	5	6	7	8	9
X-Coord	2	9	62	11	53	35	65	78	85
Y-Coord	6	90	158	112	196	165	46	50	195
Early	0	0	51	120	214	250	375	387	532
Late	2597	203	289	358	452	488	611	625	770
Node	10	11	12	13	14	15	16	17	18
X-Coord	26	72	26	47	48	41	39	83	14
Y-Coord	11	45	93	42	45	166	25	170	123
Early	725	782	848	903	906	1027	1168	1320	1403
Late	963	1020	1086	1141	1144	1265	1406	1558	1641
Node	19	20	21	22	23	24	25	26	27
X-Coord	24	42	71	99	6	45	32	93	95
Y-Coord	113	72	97	134	120	115	74	46	60
Early	1417	1462	1500	1546	1640	1679	1722	1789	1803
Late	1655	1700	1738	1784	1878	1917	1960	2027	2041
Node	28	29	30	31	32	33	34	35	
X-Coord	35	46	79	68	71	77	60	88	
Y-Coord	193	29	10	42	43	62	40	30	
Early	1949	2113	2151	2185	2188	2208	2236	2266	
Late	2187	2351	2389	2423	2426	2446	2474	2504	

Table E.28: Problem R35-4

Node	1	2	3	4	5	6	7	8	9
X-Coord	2	9	62	11	53	35	65	78	85
Y-Coord	6	90	158	112	196	165	46	50	195
Early	0	0	0	19	95	123	222	233	349
Late	2652	239	308	363	438	467	565	577	693
Node	10	11	12	13	14	15	16	17	18
X-Coord	26	72	26	47	48	41	39	83	14
Y-Coord	11	45	93	42	45	166	25	170	123
Early	503	549	602	646	648	745	858	979	1046
Late	847	893	945	989	992	1089	1201	1323	1389
Node	19	20	21	22	23	24	25	26	27
X-Coord	24	42	71	99	6	45	32	93	95
Y-Coord	113	72	97	134	120	115	74	46	60
Early	1057	1093	1123	1160	1235	1267	1301	1355	1366
Late	1401	1437	1467	1504	1579	1610	1645	1698	1709
Node	28	29	30	31	32	33	34	35	
X-Coord	35	46	79	68	71	77	60	88	
Y-Coord	193	29	10	42	43	62	40	30	
Early	1483	1614	1644	1671	1674	1690	1712	1736	
Late	1826	1957	1988	2015	2017	2033	2056	2080	

Table E.29: Problem R35-5

Node	1	2	3	4	5	6	7	8	9
X-Coord	3	45	79	56	98	83	23	25	98
Y-Coord	10	35	26	91	176	158	80	36	45
Early	0	0	0	25	120	143	241	285	359
Late	2394	177	212	281	376	399	497	541	615
Node	10	11	12	13	14	15	16	17	18
X-Coord	6	23	46	21	22	83	12	85	61
Y-Coord	129	49	175	136	142	135	119	176	113
Early	484	566	694	740	746	807	880	973	1040
Late	740	822	950	996	1002	1063	1136	1229	1296
Node	19	20	21	22	23	24	25	26	27
X-Coord	57	36	48	67	60	58	37	23	30
Y-Coord	42	45	5	6	137	164	93	166	199
Early	1111	1132	1174	1193	1324	1351	1425	1499	1533
Late	1367	1388	1430	1449	1580	1607	1681	1755	1789
Node	28	29	30	31	32	33	34	35	
X-Coord	97	15	5	21	21	31	20	15	
Y-Coord	39	103	115	53	44	78	81	138	
Early	1706	1810	1826	1890	1899	1934	1945	2002	
Late	1962	2066	2082	2146	2155	2190	2201	2258	

Table E.30: Problem R35-6

Node	1	2	3	4	5	6	7	8	9
X-Coord	3	45	79	56	98	83	23	25	98
Y-Coord	10	35	26	91	176	158	80	36	45
Early	0	0	0	0	70	91	179	218	285
Late	2421	197	229	291	377	397	485	525	592
Node	10	11	12	13	14	15	16	17	18
X-Coord	6	23	46	21	22	83	12	85	61
Y-Coord	129	49	175	136	142	135	119	176	113
Early	397	471	586	628	633	688	754	838	898
Late	704	778	893	935	940	995	1061	1144	1205
Node	19	20	21	22	23	24	25	26	27
X-Coord	57	36	48	67	60	58	37	23	30
Y-Coord	42	45	5	6	137	164	93	166	199
Early	962	981	1018	1036	1153	1178	1244	1311	1342
Late	1268	1287	1325	1342	1460	1484	1551	1618	1648
Node	28	29	30	31	32	33	34	35	
X-Coord	97	15	5	21	21	31	20	15	
Y-Coord	39	103	115	53	44	78	81	138	
Early	1497	1591	1605	1663	1671	1702	1712	1764	
Late	1804	1898	1912	1970	1978	2009	2019	2070	

Vita

Vanditha Mukund nee Chidambariah was born in Bangalore, India on October 30, 1959. She obtained a Bachelor's Degree in Electrical Engineering from Bangalore University in 1982 and graduated with distinction. She has a Master's Degree from the Indian Institute of Technology, Madras in Aeronautical Production Engineering, obtained in 1986. She worked with Hindustan Aeronautics Ltd., India as a Management Trainee and Industrial Engineer from 1982 - 1986 and as a Graduate Teaching Assistant at the University of Tennessee from 1986 - 1990.

Vanditha is a member of the Institute for Operations Research and the Management Sciences.