6-1-2004

A model to create bus timetables to attain maximum synchronization considering waiting times at transfer stops

Anitha Eranki
University of South Florida

A Model to Create Bus Timetables to Attain Maximum Synchronization Considering Waiting Times at Transfer Stops

by

Anitha Eranki

A thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science in Industrial Engineering
Department of Industrial and Management Systems Engineering
College of Engineering
University of South Florida

Major Professor: Grisselle Centeno, Ph.D.
Tapas Das, Ph.D.
Jian John Lu, Ph.D., P.E

Date of Approval:
March 17, 2004

Keywords: transit network planning, setting timetables, simultaneous arrivals, passenger waiting times, transfer nodes.

DEDICATION

Dedicated to my father, Mr. Satyanarayana Murthy and my mother, Mrs. Lakshmi who have played a major role in what I have accomplished in my life.

ACKNOWLEDGMENTS

I express my sincere gratitude to Dr.Grisselle Centeno, my major professor for her guidance and motivation throughout the course of this research. Her invaluable suggestions helped me a lot to make progress in this research. She has been a great source of inspiration.

I would like to extend my thanks to committee members, Dr. Tapas Das and Dr. Jian John Lu for their support and advice.

Many thanks to my parents, brother, and all my friends for their unconditional love and support. Finally I would like to express my heartful thanks to my husband Avinash, for his love, support and confidence in me.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

A MODEL TO CREATE BUS TIMETABLES TO ATTAIN MAXIMUM SYNCHRONIZATION CONSIDERING WAITING TIMES AT TRANSFER STOPS

Anitha Eranki

ABSTRACT

Due to the steady increased in public transportation demand, there is a need to provide more desirable and user-friendly transit systems. Typically, the public transportation timetables are modeled as an assignment problem, which often has objectives such as reducing the cost of operation, minimizing waiting time between transfer points or improving the quality of performance. This research considers the problem of developing synchronized timetables for bus transit systems with fixed routes when a waiting time limit exist at each transfer stops, for the passengers making connections. The objective of this research is to have maximum number of simultaneous arrivals.

Different to previous studies, a 'simultaneous arrival' has been defined as an arrival of buses of different routes at a transfer point such that the time between these arrivals do not exceed the passenger waiting time range associated with the transfer stop. In other words, at each node, an upper bound and a lower bound are set for the arrivals of two buses and these buses are run within this allowable window.

The heuristic developed has been modeled as a mixed integer linear programming problem and applied to some real life problems to evaluate the outcomes. The total number of synchronizations obtained by the model was compared to the maximum possible simultaneous arrivals at each node. Results show that a larger number of simultaneous arrivals are obtained when the waiting time ranges are relaxed. Finally some important applications of the proposed model compared to the existing models are presented.

CHAPTER 1

INTRODUCTION TO PUBLIC TRANSPORTATION

Public transportation is one of the vital service sectors of the present and the future U.S. economy, and it holds tremendous social significance as an estimated 25% of people rely on mobility. Public transportation (commonly known as transit, mass transit, and mass transportation) is transportation by bus, rail, or other conveyance, either publicly or privately owned, providing general or special service on a regular or continuing basis. Public transportation provides people with mobility and access to employment, community resources, medical care, and recreational opportunities in their communities.

It benefits those who choose to ride, as well as those who have other choices of transportation. For many people, especially low-income and welfare dependent families, public transportation is the only source of mobility. Government and the private sector have been successfully working together to fund, develop and upgrade the U.S. public transportation network. This chapter emphasizes the importance of public transportation and shows the opportunities for improvement as well as need to provide adequate, affordable, and convenient public transportation.

1.1 Need for Improvement

Throughout US, public transportation is undergoing a renaissance. A steady increase in transit investment have dramatically improved and expanded public transportation services providing Americans with increased freedom, choice, opportunity and access. This is suggested by the following facts from U.S. Department of Transportation and Federal Transit Administration.

1. Public transportation customers are diverse: People age 65 or older represent 7% of riders; 18 years and younger, 10%; women, 52%; White, 45%; African-American, 31%; Hispanic, 18%; and Asian and Native Americans, 6%.
2. In 2001, Americans took 9.5 billion trips using public transportation, an increase of 1.9 % from the previous year - the equivalent of more than one million new trips each day.
3. During the same year, ridership grew twice as fast as the U.S. population and outpaced growth in other travel modes.
4. An estimated 14 million Americans ride public transportation each weekday and an additional 25 million use it on a less frequent but regular basis.

In spite of these increase in the statistics many people continue to believe that transportation is undergoing a capacity crisis. Nearly 70% of those surveyed in a recent national poll by Better Roads Magazine say that we have overcrowded roads, airports, and public transit systems struggling to handle a growing population and economy. In addition to service problem, inadequate service reliability is due to large waiting times for

passengers. According to Bureau of Transportation Statistics nearly 47% of people do not use public transportation because of their non-availability to destinations at correct times.

From the above discussion it is evident that although there has been an increase in the performance of the public transportation systems there is a need for improvement in order to accommodate the growing demand and to provide more desirable and user-friendly transit systems.

1.2 Benefits of Public Transportation

The incorporation of public transportation options can help a community expand business opportunities, reduce sprawl, and create a sense of community through transit-oriented development. For these reasons, areas with good public transit systems are economically thriving communities and offer location advantages to businesses and individuals choosing to work or live in them. And in times of emergency, public transportation is critical to safe and efficient evacuation. Public transportation also helps to reduce road congestion, travel times, air pollution, energy, and oil consumption, all of which benefit both riders and non-riders alike. Some of the important benefits by incorporating public transportation as explained by the American Public Transportation Association discussed as follows:

1. Stimulates economic development: According to a transit coalition report (U.S Department of Transportation) every dollar taxpayers invest in public transportation generates six dollars or more in economic returns.
2. Saves money: It is more cost efficient to use public transport, particularly in business and urban areas because of increased congestion and high parking rates. According to American Automobile Authority, the estimated cost of driving a single-occupant vehicle is very huge compared to annual average cost for public transportation for a single adult based upon mileage, time of day, type of vehicle or service.
3. Creates jobs: Increase in public transport creates thousands of jobs in the related areas like engineering, manufacturing, construction, retail, etc. For every $10 million invested in capital projects for public transportation, more than 300 jobs are created and $ 30 million gain can be realized. It also helps in getting more people to work who does not own cars.
4. Decreases traffic congestion: Nearly half of all Americans feel that traffic is a serious problem where they live and do not feel that it will improve over the next three years. According to Texas Transportation Institute, car drivers spent more than 40 hours last year struck in traffic in nearly one-third of the cities. Public transport helps to alleviate a nation's crowded network of roads by providing more options for commuting.
5. Fosters more livable communities and boost real estate values: Public transportation facilities are focal points for economic and social activities. These activities help create strong neighborhood centers that are economically more

stable, productive and safe. This in turn has a positive impact on local property values.

6. Improves air quality and reduce energy consumption: Public transportation helps promote cleaner air by reducing automobile use. Also it can significantly reduce dependency on gasoline, reducing auto fuel consumption by 1.5 billion gallons annually.
7. Ensures safety: Public transportation continues to be one of the safest modes of travel in United States. Transit vehicle operators are highly trained to anticipate and avoid problems. Most transit vehicles are larger, newer and more substantial than autos or vans.

We see that through improved mobility, safety, security, economic opportunity and environmental quality, public transportation benefits every segment of society including individuals, families, businesses, industries and communities and supports important national goals and policies.

1.3 Bus Transportation

Among all the public transportation systems buses are the most popular and most commonly used ones because of their inherent flexibility, adaptability to changing employment and residential patterns, and low capital costs. As shown in Figure 1 buses account for highest ridership (nearly 60%) than any other modes of transit according to Federal highway administration.

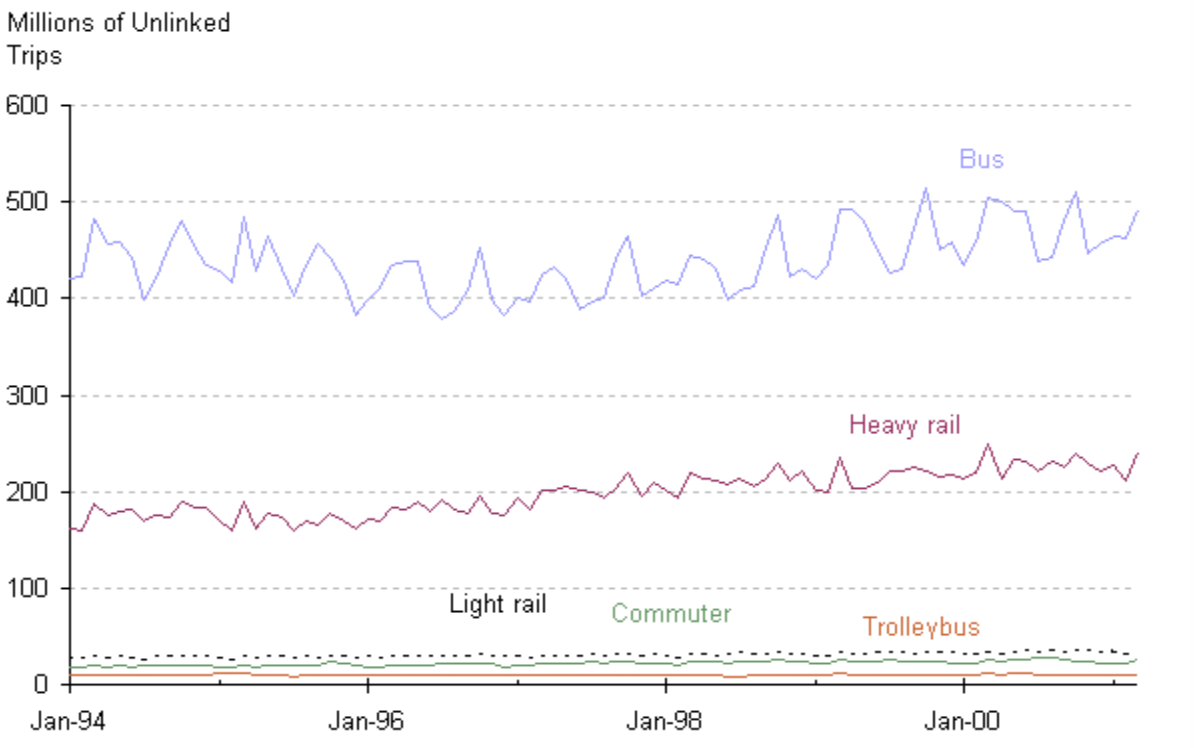

Figure 1: U.S. Transit Ridership by Mode
Source: Federal Highway Administration

According to the General Accounting Office, creating new bus rapid transit lanes costs approximately $9 million per mile compared to $34.8 million per mile for light rail alternatives. Moreover the availability of bus transport is particularly important to people with limited incomes. Users with incomes under $20,000 account for a larger percent of users of bus transit systems as shown in Figure 2 according to Federal Highway Administration.

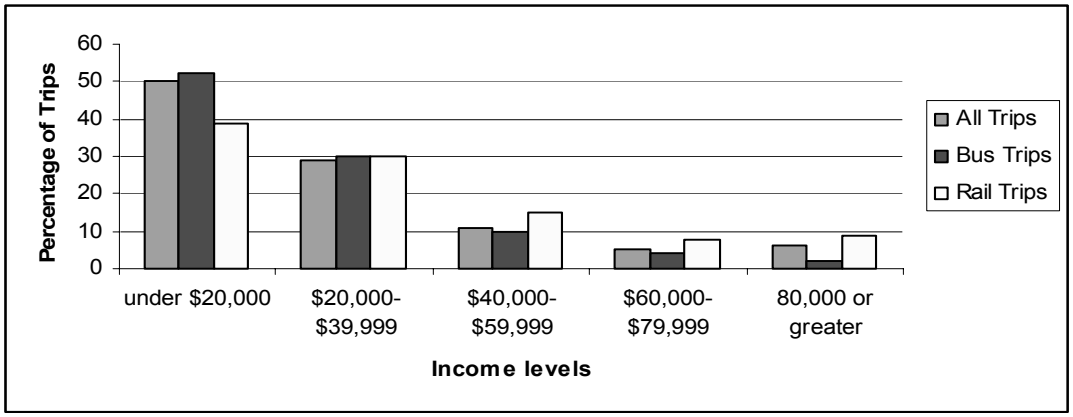

Figure 2: Distribution of Trips by Passenger Income Level
Source: Federal Highway Administration

Therefore, many researchers focused on improving bus transportation systems by reducing the cost of operations, minimizing waiting times, improving the quality of performance etc. This research also focuses on a given bus transport network. This research will improve a given bus transportation network system by creating more attractive timetables. The objective of this research is to have maximum synchronizations at the transfer points. The model developed assigns timetables such that the arrival of two buses at a transfer stop is within the allowed waiting time limits at that transfer point.

This thesis is organized as follows: the Literature review of relevant work related to public transportation is provided in Chapter 2. In Chapter 3 the problem addressed in this research, assumptions, and inputs are described. Chapter 4 discusses the formulation and the methodology used to solve the problem in detail. In Chapter 5, application to some real-life problems is discussed and finally in Chapter 6, conclusions and future work are presented.

CHAPTER 2

LITERATURE REVIEW

In this chapter a literature review is provided on different areas related to public transportation including 1) Transit network planning, 2) Setting timetables, and 3) Synchronization process. In section 2.1 various phases involved in network planning process are discussed and work related to each phase is presented. In section 2.2, various techniques to create user-friendly timetables are discussed, and finally, in section 2.3, the importance of synchronizing timetables is discussed. This study will provide a better understanding of the ways to model a public transit system.

2.1 Transit Network Planning

The public transportation operation planning process includes four basic components performed usually in sequence as mentioned in Ceder (2002).

1. Network route design
2. Setting timetables
3. Vehicle scheduling
4. Crew scheduling

The framework of transit planning process, inputs and output of each component appears in Figure 3 (Ceder, 2002). Usually each of these phases is treated as a separate problem in itself and the output of one component is the input to the next component.

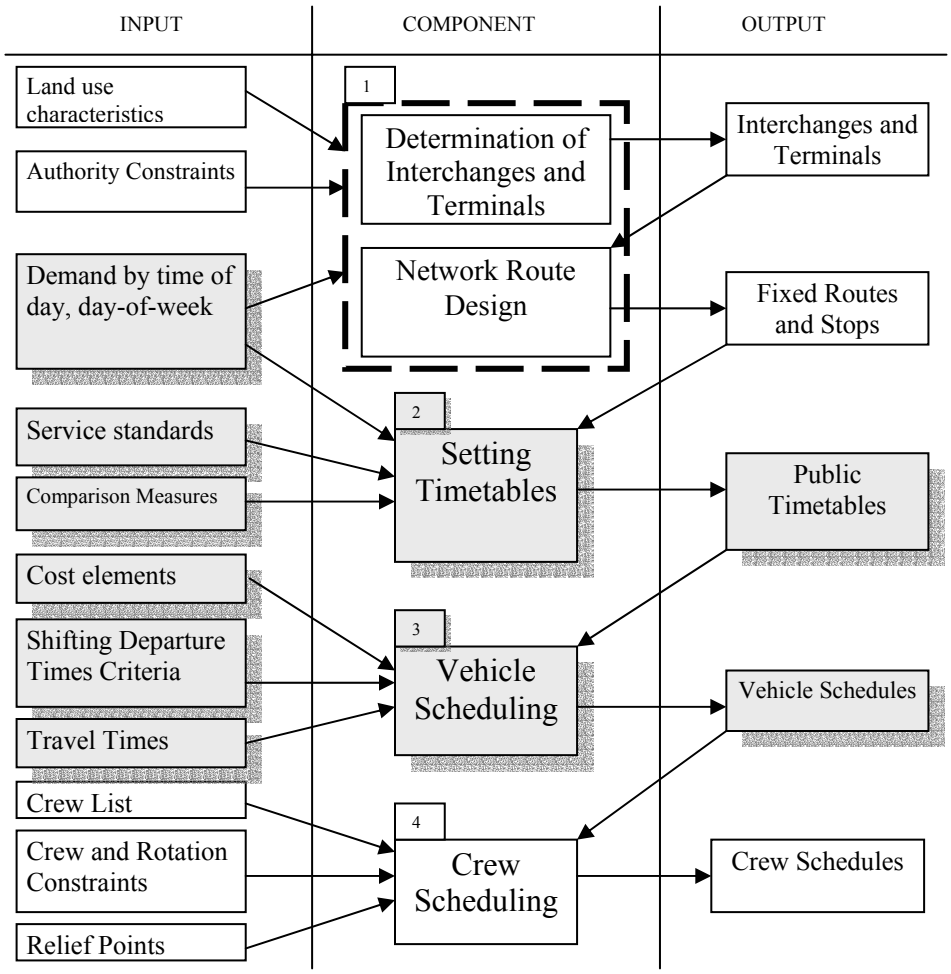

Figure 3: Framework of Public Transportation Operational Planning Process

The basic function of network route design includes: 1) identification of the location of the origins and destinations; 2) establishing relationship between those origins

and destinations with transit stops and routes; 3) identification of transit service schedules, vehicle locations, and transfer points; and 4) identification of optimal paths.

The general network design problem is to determine adding new links or expanding the old ones in the existing network. The network design problem of urban transportation consists of minimizing some objective function, subject to the equilibrium constraint (e.g., minimization of overall time or cost measures).

Baaj and Mahmassani (1995) described a hybrid route generation heuristic algorithm for network route design. The route generation algorithm (RGA) discussed determines a set of routes that correspond to different trade-offs between user and operative costs. It starts by determining initial set of skeletons and expands them to form transit routes. This process of expansion continues until a prespecified minimum percentage of total demand can be satisfied. Once a set of routes is generated, the routes are analyzed taking into account the assignment of demand to the transit network.

Gao, Sun and Shan (2003) proposed a continuous equilibrium network design model where the attention was mainly on setting optimal transit line frequencies. A bi-level programming technique with an upper-level problem and a lower-level problem has been used for this transit network design problem. In the upper-level problem the objective function is to minimize the total deterrence of the transit system and cost caused by frequency setting. The lower-level model is a transit equilibrium assignment model that is used to describe the path alternative activities to transit users. A heuristic solution

based on sensitivity analysis is designed to solve this model to obtain optimal frequencies that optimize the systems performance.

The timetable component is designed to meet the demand associated with the public transportation network. This demand varies according to the time of day, from one season to another and even from one year to another. This demand reflects the business, industrial, cultural, educational, social and recreational transportation needs of the community. It is the purpose of this component to set appropriate timetables for each transit route to meet the variation in the public demand. The various works on obtaining timetables are further discussed in section 2.2. The timetables created at this stage serve as inputs to all the other components of the planning process.

The main function of vehicle scheduling is to schedule vehicles to trips according to given timetables. The scheduler's task is to list all daily chains of trips for a vehicle, ensuring the fulfillment of the timetable requirements and the operator requirements (refueling, maintenance, etc.). The major objective is to minimize the number of vehicles required.

Park and Song (1997) discussed the vehicle scheduling problems with time-varying speeds in which the travel speed between two locations depends on the passing areas and time of day. They developed three heuristics by extending and modifying conventional vehicle scheduling problems.

Haghani and Banihashemi (2002) proposed a heuristics approach for solving a large-scale vehicle-scheduling problem with route time constraints. In this work a new formulation for multi depot vehicle scheduling (MDVS) and multi depot vehicle scheduling problem with route time constraints (MDVSRTC) have been proposed. To solve a medium size MDVSRTC problem it provides an exact and heuristic solution procedure that cannot solve a real time problem. Hence for a real-world application, they have proposed a solution procedure that reduces the size of a large-scale problem by decreasing the number of trips and by decreasing the number of variables. It is shown that the solution obtained from proposed strategy has decreased the number of vehicles required and also the operating costs.

Crew scheduling is concerned with the scheduling of manpower resources to meet the operational requirements of the transit system. It is a critical activity that operates round-the-clock where workers are scheduled to work on multiple shifts. Crew scheduling is often modeled as an optimization problem for constructing cost-optimal schedules that do not violate given scheduling constraints.

Wren and Wren (1995) proposed a genetic algorithm for solving a public transport driver-scheduling problem. The genetic algorithm proposed in this work uses a new crossover operator. It is shown that the algorithm would produce more efficient results than the presented existing method in terms of quality of result and time taken to obtain the schedule. Beasley and Cao (1996) presented an algorithm for crew scheduling

problem based upon the lagrangean relaxation of a linear integer-programming problem, together with a sub-gradient optimization and tree search procedure.

It is necessary that all the four components be planned simultaneously to exploit the system's capability to the greatest extent and to maximize the system's productivity and efficiency. The complexity involved in the public transport operational planning process challenges many researches to develop an automated computerized procedure to solve the problem efficiently at each of the steps involved.

The discussion of different steps involved in public transportation planning process provides an insight into the various functions involved at each step. This research mainly focuses on the second step, setting timetables, and ways to improve the performance of the system by constructing efficient schedules.

2.2 Setting Timetables

This section focuses on the different works related to timetables creation. Ceder (1986) describes alternative methods for creating bus timetables based on passenger load data. It attempts to fulfill six major objectives: (1) to evaluate alternative timetables in terms of required resources, (2) to improve the correspondence of bus departure times with passenger demand, (3) to provide alternative timetables for the schedule's use in specific scheduling situations, (4) to permit direct bus frequency changes for possible exceptions (known to the scheduler) which do not rely on passenger demand data, (5) to

allow construction of timetables with headway smoothing techniques and (6) to integrate different headway setting and different timetable construction methods. Four different frequency construction methods are described here which in turn are used to obtain bus headways. The procedure develops set of bus departure times for evenly spaced headways and unevenly spaced headways.

The problem faced by many schedulers, is how to set departure times in the transition segments between adjacent time periods. Using the common average headway rule may result in over-crowding. Ceder (1986) developed smoothing techniques whose objective is to set in the transition time an average desired occupancy rather than an average headway. In case of unevenly spaced headways procedures for balancing passenger load are described where the objective is to set the departure time of each bus in a given time period so that its maximum load will approach the desired occupancy associated with that period.

Palma and Lindsey (2001) analyzed optimal timetables for a given number of vehicles on a single transit line. In this method it is assumed that the preferred travel times and unit schedule delay costs vary from person to person. Here two models are considered: line model and the circle model. In the line model preferred travel times of the individuals are distributed over a segment of the day and rescheduling trips between days is impossible. But the circle model the preferred travel times are distributed round the clock and rescheduling trips between days is possible. The analysis for both the models proceeds in two steps. The first step is to determine for an arbitrary timetable of

vehicles, which individuals will travel on which vehicles. The second step is to determine the timetable that minimizes total schedule delay costs given the behavior of individuals identified in step one. This model can be best applied to create timetables when the objective of the transit planners is to reduce the riders delay costs.

Yan and Chen (2002) developed a solution algorithm to produce timetables and bus routes/schedules for inter-city bus carriers. Urban bus and inter-city bus operations differ in terms of their scheduling practices and demand arrival patterns, mainly due to the fixed time schedule set for the latter, and the rough service frequency set for the former.

In this research they developed a model for inter-city carriers with given passenger trip demand, bus fleet size and related cost data. The model demanded the optimal management of both bus and passenger movements in the network through the systematic manipulation of direct bus trips, multi-stop bus trips, and passenger transfer operations, utilize data as the projected passenger trip demand, the available fleet size, the bus operating speed, the station turn-around time, the passenger trip ticket fare, and the related cost data. Mathematically, the model is formulated as a mixed integer multiple commodity network flow problem. This method of constructing timetables can be better applied when the objective is to maximize the system profits.

In a follow-up study Ceder (2002), three different procedures are proposed and analyzed for better matching the passenger demand with a given timetable while

attempting to minimize the number of departures. Procedure 1 produces departure times with evenly spaced headways while considering smooth transition between adjacent hours. Procedure 2 determines departure times such that, in average sense, vehicles will carry on even loads (equal to desired occupancy) at hourly max load points. Procedure 3 derives the departure times such that, in average sense, the on-board passenger load will not exceed desired occupancy, and will be equal to desired occupancy at each individual vehicle max load point. All these three procedures are applicable to situations when the transit planners want to have balanced loads on all the buses to prevent over crowding.

The previous studies were done in order to understand the different techniques for constructing timetables. It is evident that the past research on bus scheduling has commonly been on cost minimization or profit/service maximization under a given/variable demand. Apart from profit maximization and cost minimization the scheduler also faces the task of minimizing waiting times. This research focuses of this aspect of timetable scheduling i.e., creating synchronized timetables for a given transit network.

2.3 Synchronization

According to Ceder et al. (2001), the importance of transfers in public transport service is motivated by several operational reasons as shown in Figure 4. In a large public transport network, all the origins and destinations are connected not connected by a single route and have a number of transfer points. In this case passengers who want to

travel between different routes have to change routes at transfer points. If there are a large numbers of such transfer points a "perfect" timetable can be achieved only if the waiting time between the transfer points is minimized.

In large networks it is extremely difficult to completely eliminate these transfers. Transfers improve the transit network service characteristics by: increasing the possible number of travel paths, improving transit network operational flexibility and efficiency, making the system cost effective, and improving transportation infrastructure which guarantee high service quality and efficient resource utilization.

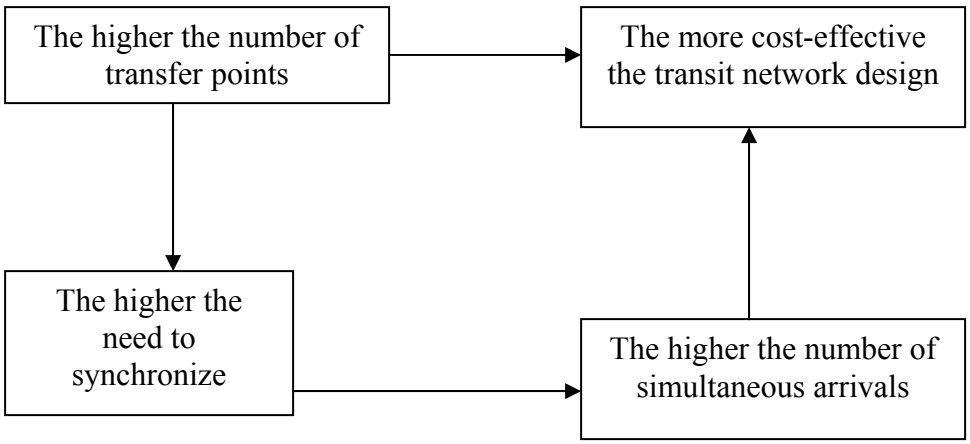

Figure 4: Importance of Synchronization

Unfortunately, transfers involve certain inconveniences connected with discomfort of boarding a new vehicle (necessity of passenger orientation and walking between vehicles on feeder and receiving lines), negative perception of waiting for arrival of a vehicle and existence of some delay during a trip. The elimination of these inconveniences by schedule synchronization to provide an attractive service level with

easy access and transfer possibilities is continuously a challenging problem in timetable construction.

Bookbinder and Désilets (1992) proposed transfer optimization in a transit network to minimize the overall inconvenience to passengers. Bus trips are scheduled to depart from their terminal so as to minimize some objective function measuring that inconvenience. A mean disutility function is defined here which is used to evaluate the inconvenience under random bus travel times of a transfer connection. This disutility function $g(w)$ is some function of waiting time, which gives the desirability of a waiting time w, as perceived by the user. To obtain a heuristic solution, an iterative improvement procedure is used. This procedure starts with an initial solution and looks for improvements by changing the departure times for each route from a set of possible starting times, until no further improvement can be obtained.

This method is from the perspective of the user's that reduces the inconvenience caused to the user of the transit system. Ceder et al. (2001) proposed a method for synchronization from the perspective of the scheduler creating a useful synchronization tool for schedulers. They attempted to maximize the total number of synchronizations of bus arrivals at bus stops by formulating a model for the maximum synchronization of arrivals as a mixed-integer problem (MIP). Also, a heuristic algorithm that would solve this problem in a reasonable time was proposed. In each step of the algorithm, a node is selected based on some criteria, provided that at this node not all the departure times have

yet been determined. Once the departure time is resolved, all its corresponding arrival times are also set. The algorithm has been constructed to handle one time period *T*.

A heuristic model based on node-oriented approach is developed. Each iteration attempts to optimize transfers taking place at a selected node. Quak (2003) in his work "Bus line planning", have discussed in detail the first two phases (i.e., designing routes and setting timetables) of bus transit planning. He adjusted the objective function of Ceder et al. and constructed a different heuristic to set the departure times. This is based on Line-Oriented Departures Setting Method (LODSM), i.e., to synchronize departures from the point of view of routes instead of the nodes.

Ceder et al. (2001) defined simultaneous arrivals as the arrival of two buses at the transfer node at the same time. This seems to be applicable to only peak period. For non-peak period transit operation, the frequency of buses in generally low and the system is characterized with a certain waiting time. Our research creates timetables for off-period operations by incorporating the waiting times at each transfer point.

In this chapter a review of work related to various phases of bus transit planning is provided. It is evident from the review that there are various techniques to create timetables, each employing different objective suitable to a specific situation. In transit planning the method employed to create user-friendly timetables by the planners depends on the need of the public and the characteristics of that particular transit network.

CHAPTER 3

PROBLEM DESCRIPTION

In this chapter, the problem addressed in this research is presented. In addition, various inputs and assumptions required for constructing timetable of a given bus transit network are discussed.

3.1 Description

As seen in all of previous studies models to create timetables for public transportation networks, different objectives suitable to specific situation are considered. Ceder et al. (2001) have created timetables for a specific time period, which maximizes the total simultaneous arrivals at the transfer nodes. They defined simultaneous arrival as the arrival of two buses at a node at the same time.

A waiting time of zero is considered to be optimal. But in practice the passengers do not appreciate a transfer waiting time of zero. Due to the uncertainties involved in the public transit system, passengers generally prefer a certain minimum transfer waiting time rather than zero. Creating timetables with this minimum possible waiting time makes the transit system more robust.

Moreover, during non-peak period, transit management is more likely to operate buses with a decreased frequency than would be required purely on capacity grounds during off-peak hour and to reduce operating cost. Because of this decreased frequency the transit system is characterized by waiting time for the passengers at the transfer points. This waiting time is the time the passengers wait at the transfer points and depends on different operating characteristics like passenger arrival distribution at various points in the network, headways, frequency, time of operation, congestion, etc. It also depends on various physical characteristics associated with each transfer point like its location and surroundings.

Apart from this, it seems more realistic to model the network with some minimum waiting time at transfer point for smooth transfer of passengers from one route to another. But the waiting time should not be too long which will make the system unreliable. The model developed in this research can be applied to transit networks with certain minimum waiting time involved in making transfers.

In our research, it is assumed that at each transfer node there is a minimum and maximum waiting time limit for passengers. The challenge of this research is to set the departure times of the routes as good as possible inside the respective periods so that their arrival times at a transfer node are close to each other, in order to have minimum possible waiting time for the passengers. In this model 'simultaneous arrivals' are defined in a different way. It is defined as the arrival of two buses at transfer points such that the time

gap between the arrivals does not exceed the required waiting time. The objective of this research is to have maximum number of simultaneous arrivals.

3.1.1 Definitions

1. Route

A route contains ordered sequence of bus stops. This sequence consists of at least two elements. The starting stop represents the origin of the trip and the final stop is the destination.

2. Node

A node is a location where the buses stop to either to pick up or to deliver one or more passengers.

3. Headway

Headway is defined as the time between successive departures on each route.

4. Frequency

Frequency is defined as the possible number of departures on each route in a specified time interval.

In this research timetables are created for a given bus transit network. The network is represented by a set of bus-routes (arcs) and bus-stops (nodes) as shown in Figure 5. According to this figure certain routes meet at bus-stops and passengers might transfers between routes at these stops.

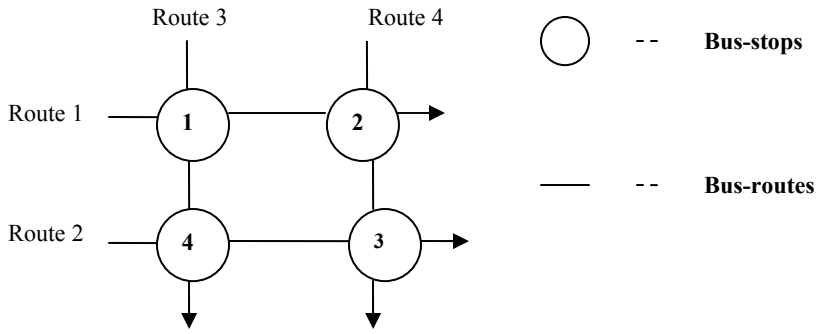

Figure 5: Bus Transit Network

Each route is characterized by a certain minimum frequency (number of buses per hour) and headway (time between successive departures). Each stop is associated with the different routes passing through it and waiting time for the passengers making a transfer between these routes. Appropriate departure times have been assigned so that simultaneous arrivals at the transfer nodes are maximized. The following extensions to the work done by Ceder et al. (2001) are incorporated in this research:

1. The simultaneous arrival of buses at transfer points is defined as the arrival of two buses at the transfer point in such a way that the time gap between the arrivals does not exceed a specified allowable waiting time instead of defining it as arrival of buses at the same time.
2. In the transit network, each transfer point has a pre-determined limit for waiting time.
3. A model that maximizes the number of simultaneous arrivals is developed.

4. An algorithm incorporating the waiting times is developed to solve this problem of assigning timetables.

3.2 Inputs Required

The following are the inputs required for the problem described in this research.

1. Details of existing bus network like origin-destination of various routes and the nodes present on these routes. The given network is presented by a directed graph, $G\{A, N\}$, where A is a set of arcs representing the traveling path of the bus routes and N is a set of transfer nodes in the network.
2. Period during which the departure times of buses can be set. This is called planning horizon represented by [*0, T*].
3. Number of bus routes in the network, *M*.
4. Maximum and minimum headway (time between each successive departures) for each route *i*, ($1 \le i \le M$) represented by $[H\min_i, H\max_i]$.
5. Minimum required frequency, f_i (number of buses to be scheduled for a particular time period) on each route.
6. Traveling time, t_{ik} from starting point of route *i* ($1 \le i \le M$) to node k ($1 \le k \le N$).
7. Permissible waiting time limits for passengers making transfers at each node *k*, ($1 \le k \le N$) $[WT\min_k, WT\max_k]$

3.3 Assumptions

The following are the assumptions made for the problem presented in this research.

1. The planning horizon in which the departure times are set is a discrete interval and also large enough so that all the departures can be set.
2. For each route i, $H\max_i \geq H\min_i$.
3. Traveling times and the waiting times are considered deterministic.
4. The waiting time limits for the passengers at each node are assumed to be predetermined by the transit planners.
5. The first departure of each route i must take place in the interval $[0, H\max_i]$.
6. The travel on all routes is in one particular direction.
7. The buses leave the terminal (nodes) as soon as they arrive.

3.4 Research Objectives

The following are the objectives of this research

1. To create a synchronized timetable, this ensures that the time between different buses arrive at transfer points, does not exceed the allowed waiting time.
2. To model this problem as a Mixed Integer-programming (MIP) problem.
3. To present an appropriate heuristic algorithm to solve this problem.
4. Present some numerical examples to explain the algorithm.
5. To apply the proposed model to some real life problems.

3.5 Problem Size

With the inclusion of waiting times, the size of the problem increases and also the number of iterations required to solve the problem. For each possible waiting time from the specified interval there will be a set of departure times. To have an idea of the size of the problem we analyze the number of iterations required to solve. For a problem with four nodes and four routes and with each route having a frequency of three buses per hour, we have 12 iterations to perform to set 12 departures times. Considering the waiting times this number increases, if there are n number of possible waiting times then the number of iterations performed are 12^n to set 12 departure time by choosing the one with less waiting time.

Example problems to test the results are taken from Ceder et al. (2001), which do not consider waiting times. For these examples, we randomly considered certain limit for waiting times at transfer nodes making sure that the largest possible waiting time does not exceed the maximum headway of routes passing through the nodes. The algorithm is developed considering these waiting times.

The model developed in our research will represent more realistically the waiting time parameter that currently exists in the system. It will be greatly useful to transit planners whose aim is to provide an effective bus transit system with minimum transfer time.

CHAPTER 4

METHODOLOGY

This chapter discusses the methodology that is implemented in this research to create synchronized timetables. This includes formulating the problem as a Mixed Integer programming problem and an algorithm to solve this problem.

4.1 Notations

The following are the notation used in defining the problem and are used in the entire research.

N – Total number of nodes k present in the network.

M – Total number of bus routes present in the network.

T – Planning horizon during which the departure times are constructed.

$H\min_i$ – Minimum required headway for route i.

$H\max_i$ – Maximum required headway for route i.

t_{ik} – Travel time from the starting point (origin) on route i to node k.

$WT\min_k$ – Minimum allowed waiting time at node k.

$WT\max_k$ – Maximum allowed waiting time at node k.

X_{ip} – Departure time of p^{th} bus on route *i*.

T_{ip}^{k} – Arrival time of p^{th} bus on route on route *i* at node *k*.

B_i – Set of nodes contained on route *i*.

$B_{i,j}$ – Set of common nodes contained on route *i* and route *j*.

f_i – Frequency (number of buses departing in a given time period) on each route.

4.2 Formulating the Model

The decision variable Y_{ijkpq} is defined as,

Y_{ijkpq} = 1, if the arrivals of p^{th} bus on route *i* and q^{th} bus on route *j* at node *k* are separated by a time that is within the required waiting time limit.

Y_{ijkpq} = 0, otherwise.

The arrival time of buses at a node is calculated by adding the departure time and the time taken to travel to that node, i.e., $T_{ip}^{k} = X_{ip} + t_{ik}$.

The objective function of the model presented is to maximize the number of simultaneous arrivals.

$$\max \sum_{i=1}^{M-1} \sum_{j=i+1}^{M} \sum_{k \in B_{i,j}} \sum_{p=1}^{f_i} \sum_{q=1}^{f_j} \{Y_{ijkpq}\}$$

The constraints are given by the following equations

$$X_{i1} \le H\max_i;\ 1 \le i \le M \tag{1}$$

$$X_{if_i} \le T;\ 1 \le i \le M \tag{2}$$

$$H\min_i \le X_{i(p+1)} - X_{ip} \le H\max_i;\ 1 \le i \le M; 1 \le p \le f_i - 1 \tag{3}$$

$$Y_{ijkpq} = 1\ if\ WT\min_k \le \left|(X_{ip} + t_{ik}) - (X_{jq} + t_{jk})\right| \le WT\max_k,\ k \in B_{i,j} \tag{4}$$

Y_{ijkpq} =0;

$$if\ \left|(X_{ip} + t_{ik}) - (X_{jq} + t_{jk})\right| < WT\min_k\ \ or\ \left|(X_{ip} + t_{ik}) - (X_{jq} + t_{jk})\right| > WT\max_k, \tag{5}$$

Constraint (1) ensures that the first departure time of each route will not be beyond maximum headway from the start of time horizon and constraint (2) ensures that the last departure is within the planning horizon. Constraint (3) indicates the headway limits. Constraint (4) shows that the decision variable takes a value of 1 if the arrivals at the node are within the waiting limits and constraint (5) ensures that Y_{ijkpq} takes the value 0 otherwise.

4.3 Complexity

The numbers of integer variables present in the problem generally indicates the complexity of a Mixed Integer Programming problem. In our problem, the difference between the arrivals of two buses at a transfer node can take any value in the permissible

waiting time. Therefore there is an integer variable for every combination of two buses of different routes that meet at a node for every possible waiting time that is in the permissible limits. According to Ceder et al. (2001), the number of integer variables without waiting times is $O(M^2F^2)$ where F is the maximum frequency of all routes. For our model, where waiting time is incorporated into the model, the problem becomes more complex and the number of integer variables increases as $O(M^2F^2W)$ at each node where W is the number of possible waiting times ($W = WT\max_k - WT\min_k$) at that node. With this complexity it is difficult to solve a real size problem. Hence a heuristic has been developed.

4.4 Heuristic Approach

This section will present the algorithm developed to solve our problem of setting departure times. The basic outline of the algorithm is based on the algorithm developed by Ceder et al. (2001). The incorporated change in the definition of simultaneous arrival is applied in the different procedures used to set the departure times. The flow chart of the algorithm is shown in Figure 6. The algorithm is based on the selection of nodes. There are three possible states for a node. A node can be 'new', 'possible' or 'not possible'.

1. A node is 'new' if none of the departure times of routes passing through the node are set.
2. A node is defined as 'possible' if:

- There is at least one route passing through it and not all the departure times for that route are set.
- There is a possibility to create more synchronized arrivals at the node.

3. A node is 'not possible' if all the departures of the routes passing through it are set and no more simultaneous arrivals are possible.

The flow chart of the algorithm is presented in Figure 6. Details about input values, node selection process and procedures are discussed in the following sub-sections. The following are the steps in the algorithm:

1. Take the input values and initialize all the nodes as 'new';
2. Identify the node '*SELECTED NODE*' by following the Node Selection Procedure in section 4.4.2;
3. If *'SELECTED NODE'* is new perform *PROCEDURE 1*, otherwise perform *PROCEDURE 2*;
4. Are there any 'new' or 'possible' nodes? If yes, go to Step 2. Otherwise continue;
5. Are there any routes with unassigned departures? If yes, perform *PROCEDURE 3*, otherwise stop.
6. Are there any possible nodes? If yes, go to Step 2. Otherwise stop.

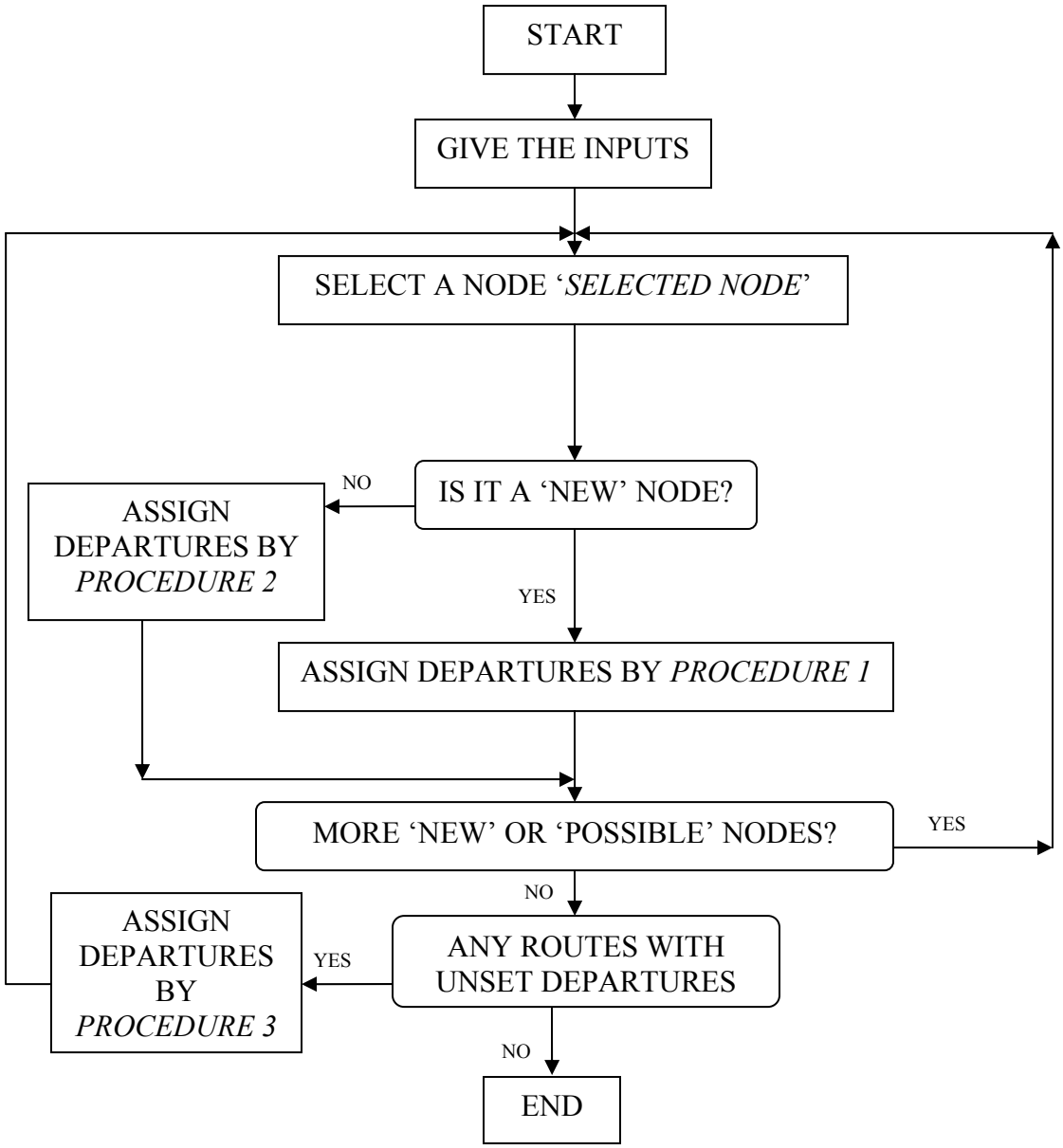

Figure 6: Flow Chart of the Algorithm

4.4.1 Model Inputs

The following are the assumptions made on the input data for each route *i*:

1. $H\max_i \geq H\min_i$
2. $T \geq (f_i - 1) \cdot H\min_i$

3. The maximum possible limit on the planning horizon is the maximum value given by ($f_i \cdot H \max_i$) among all routes.
4. The minimum possible waiting time at each node is some value greater than zero.
5. The maximum waiting time at each node is a value that does not exceeds the maximum headway of routes passing through it.

4.4.2 Node Selection Procedure

In each iterative step of the algorithm, a node is selected from among all the 'new' and 'possible' nodes. There are three steps for selecting a node. They are:

1. Among the 'new' and 'possible' nodes find the node that has maximum number of already set arrival times. If no arrival times are set or if ties exist go to Step 2. If only one 'new' or 'possible' node is identified, label this node as '*SELECTED NODE*' (k^*) and exit.
2. Identify the node with the maximum number of routes passing through it. If ties exist go to Step 3. If only one node is identified, label this node as '*SELECTED NODE*' (k^*) and exit.
3. Calculate the maximum travel time from the origin of each route to these nodes. Select the node with the minimum value and label it as '*SELECTED NODE*' (k^*). If a tie exists break it arbitrarily.

4.4.3 *PROCEDURE 1*

This procedure assigns departure times for routes meeting at the '*SELECTED NODE*' if it is 'new'. Suppose that two routes meet at the node, then this procedure assigns the departure time of the route that takes maximum time to arrive to that node. It assigns a departure time of 0 (i.e., the starting time of the planning horizon) to the 1st bus on this route that takes maximum travel time. For the other route it assigns the departure time such that the arrival times of these two routes at the selected node is within the specified waiting time limits.

The procedure first checks if it is possible to have the minimum allowed waiting time and if it is possible, departure times are assigned accordingly. If it is not satisfied it increases the time by one (discrete time) and verifies for the next possible waiting time from the limit and so on till the maximum limit is reached. The subsequent departures for these routes are fixed after a time d from the last departure. The procedure finds the minimum possible d that is given by:

$$d = \min_{i=1,2...M} [\max_{i=1,2...M}(H\min_i), \min_{i=1,2...M}(H\max_i)] \ \forall i, \text{ passing through the selected node.}$$

This procedure performs the following steps:

Step 1: At the *SELECTED NODE* (k^*),

- For all routes i, passing through it calculate minimum possible d satisfying,

$$d = \min_{i=1,2...M} [\max_{i=1,2...M}(H\min_i), \min_{i=1,2...M}(H\max_i)] \ \forall i \text{ passing through } k^*.$$

- Set *maxtime* = maximum travel time to reach k^* and identify the route associated with *maxtime* to reach k^* and label it as i^*.

Step 2: For the route i^*, set the first departure (p = 1) as $X_{i^*1} = 0$

Step 3: For the other routes i, passing through this node:

- If ($maxtime$ - $WT\min_{k^*}$ - t_{ik^*} >0)

  Set X_{i1} = ($maxtime$ - $WT\min_{k*}$ - t_{ik^*}) and go to Step 5.

- Otherwise, set $w = WT\min_{k^*}$

Step 4: If ($maxtime + w$) $\geq t_{ik^*}$ and if ($maxtime + w - t_{ik^*}$) $\leq H\max_i$

- Set X_{i1} = ($maxtime + w - t_{ik^*}$) and go to Step 5.

Else, set $w = w + 1$. If $w \leq WT\max_{k^*}$ repeat step 4, else exit.

Step 5: For these routes i and i^*, if the procedure is able to find the value of d in Step 1, then the subsequent departures i.e., $\forall p$ = 2,...., min(f_i, f_{i^*}) are assigned after an interval of d from the previous departure. That is:

- Set $X_{ip} = X_{i(p-1)} + d$
- Set $X_{i^*p} = X_{i^*(p-1)} + d$

Else go to Step 6.

Step 6: For these routes i and i^* compute the arrival times of all the set departures (p = 1,2,...., min(f_i, f_{i^*})) to each 'possible' and 'new' nodes on the route as:

- $T_{ip}^k = X_{ip} + t_{ik}$ $\forall k$ = 1,…., N present on i.
- $T_{i^*p}^k = X_{i^*p} + t_{i^*k}$ $\forall k$ = 1,…., N present on i^*.

Step 7: Label all the other nodes on these routes, as 'possible' and label k^* as 'not possible' and exit.

4.4.4 *PROCEDURE 2*

This method sets the departure times when the selected node is 'possible'. For a selected 'possible' node there will be some routes whose starting times are already set using *PROCEDURE 1*. Hence the departure times of the routes that are not set are assigned to have a simultaneous arrival with the set arrivals at the node. If no more assignments are possible the node is marked as 'not possible'. The following steps are performed by this procedure:

Step 1: At the *SELECTED NODE*, k^* that is 'possible',

- Set i^* as the route passing through k^* and all the departure times are already set using *PROCEDURE 1*. Set arrival times of i^* from the origin to k^* as:

  $$T_{i^*p}^{k^*} = X_{i^*p} + t_{ik^*} \quad \forall p = 1....f_{i^*}$$

Step 2: For the other routes *i*, passing through k^*, set *p* as the minimum un-assigned frequency where $p \in (1....f_i)$.

For each $T_{i^*p}^{k^*}$ that is, $T_{i^*1}^{k^*}, T_{i^*2}^{k^*}, \ldots\ldots, T_{i^*f_{i^*}}^{k^*}$ set $w = WT \min_{k^*}$

Step 3: For route *i*, if $0 \le (T_{i^*p}^{k^*} - w - t_{ik^*}) \le H \max_i + X_{i(p-1)}$

For $p = 1$, set $X_{ip} = (T^{k^*}_{i^*p} - w - t_{ik^*})$ and go to Step 4

For $p > 1$, if $(T^{k^*}_{i^*p} - w - t_{ik^*}) - X_{i(p-1)} \geq H \min_i$, set $X_{ip} = (T^{k^*}_{i^*p} - w - t_{ik^*})$ and go to Step 4.

Else if $0 \leq (T^{k^*}_{i^*p} + w - t_{ik^*}) \leq H \max_i + X_{i(p-1)}$

For $p = 1$, set $X_{ip} = (T^{k^*}_{i^*p} + w - t_{ik^*})$ and go to Step 4

For $p > 1$, if $(T^{k^*}_{i^*p} + w - t_{ik^*}) - X_{i(p-1)} \geq H \min_i$, set $X_{ip} = (T^{k^*}_{i^*p} + w - t_{ik^*})$ and go to Step 4.

Other wise set $w = w + 1$ and, if $w \leq WT \max_{k^*}$ repeat Step 3. Otherwise set $T^{k^*}_{i^*p} = T^{k^*}_{i^*p+1}$ and set $w = WT \min_{k^*}$ and go to Step 3. Else exit.

Step 4: For route i if $p < f_i$ – go to Step 2, Otherwise label k^* as 'not possible'.

Step 5: For routes i passing through k^* and for the departure times that are set in Step 3, compute the arrival time to each 'possible' and 'new' nodes present on the route as, $T^k_{ip} = X_{ip} + t_{ik} \quad \forall k = 1, \ldots..,$ N on route i.

Step 6: Label all the other 'new' nodes on these routes, as 'possible'.

4.4.5 *PROCEDURE 3*

This procedure checks if there are any un-assigned departures that are not created by the first two procedures.

1. If there is only one unassigned departure on route i, and set p as the minimum unassigned frequency where $p \in (1....f_i)$,
    - Set this departure time using the minimum headway from the last departure as $X_{ip} = X_{i(p-1)} + H\min_i$ and exit.
2. If there are more than one unassigned departures on different routes,
    - Identify the route i, passing through the maximum number of nodes, break ties arbitrarily.
    - Assigns its next departure by using minimum headway from the last departure. i.e., $X_{ip} = X_{i(p-1)} + H\min_i$
    - All the nodes through which the identified route passes are labeled as 'possible' again.

This will allow the algorithm can to set additional simultaneous arrivals at 'possible' nodes.

4.5 Numerical Example

The algorithm explained in the previous section is coded in C language to test some example problems on bus networks. These problems are from Ceder et al. (2001) and at each node waiting times are introduced. The network with two routes and two nodes along with the travel time on each link is shown in Figure 7. The inputs for this network are given in Table 1 and Table 2. The planning horizon is [0, 60] minutes.

Step 1: Initialize Node 1 and Node 2 as 'new'.

Step 2: Identify the node '*SELECTED NODE*' k^*

1. No arrival times are set.

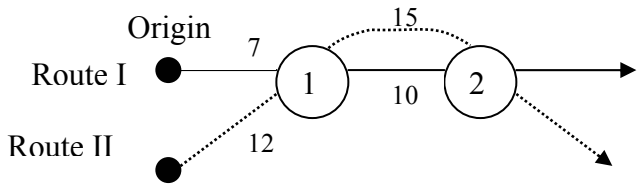

Figure 7: Numerical Example – 1

Table 1: Inputs for the Routes in Example – 1

Routes i	Minimum Headway, $H\min_i$	Maximum Headway, $H\max_i$	Frequency, f_i
I	5	15	4
II	8	20	3

Table 2: Inputs for the Nodes in Example - 1

Nodes, k	Number of routes	Minimum waiting time, $WT\min_k$	Maximum waiting time, $WT\max_k$
1	2	4	9
2	2	10	13

2. The number of routes crossing Node 1 = 2, and Node 2 = 2
3. Maximum travel time for node 2: maximum (17, 27) = 27

   and for node 1: maximum (7, 12) = 12. Minimum value is 12.

4. k^* = Node 1.

Step 3: As node 1 is 'new' perform Procedure 1.

1. The d value obtained is, d = min [8, 15] = 8.

   $maxtime$ = 12 and the route i^* is route II.

2. For route II, set the first departure (X_{II1}) at zero minutes.

3. For route I, ($maxtime$ - $WT\min_1$ - t_{I1}) = 12 - 4 - 7 = 1 which is greater than zero, hence set the first departure time of route I (X_{I1}) = $maxtime$ - $WT\min_1$ - t_{I1} = 1.

5. For routes I and II the number of subsequent departures set by this procedure are p = 2,….,min(f_I, f_{II}) = min (4, 3) = 3. These are set after a time d from the previous departures, i.e.,

   For route i, (i = I):

   $X_{I2} = X_{I1} + d = 1+8 = 9$

   $X_{I3} = X_{I2} + d = 9+8 = 17$

   Similarly for route i^*, (i^* = II):

   $X_{II2} = X_{II1} + d = 0+8 = 8$

   $X_{II3} = X_{II2} + d = 8+8 = 16$

6. The arrival times of routes I buses to node 1 are:

   $T^1_{I1} = 1+7 = 8$; $T^1_{I2} = 9+7 = 16$; $T^1_{I3} = 17+7 = 24$,

   and to node 2 they arrive at,

   $T^2_{I1} = 1+17 = 18$; $T^2_{I2} = 9+17 = 26$; $T^2_{I3} = 17+17 = 34$.

Similarly, the arrival times of route II buses at node 1 are 12, 20, 28 and 27, 35 and 43 minutes are the arrival times at node 2.

7. Label node 2 as ‘possible’ and node 1 as ‘not possible’. The results of this procedure are summarized in Table 3.

Table 3: Procedure 1 Results for Example 1

Departures	Route I	Route II	Arrival time at Node 1		Arrival time at Node 2	
			Route 1	Route 2	Route 1	Route 2
1	1	0	8	12	18	27
2	9	8	16	20	26	35
3	17	16	24	28	34	43
4	Not yet assigned	-	Not yet assigned	Not yet assigned	-	-

Step 4: Since node 2 is ‘possible’ – go to step 2.

1. The ‘*SELECTED NODE*’, k^* = Node 2.

Step 3: Since is ‘possible’, Procedure 2 is applied.

1. Route i^* is II and $T_{II1}^2, T_{II2}^2, T_{II3}^2$ are 27, 35 and 43 minutes respectively.
2. Route I has an un-assigned frequency which is $p = 4$. For each $T_{II\ p}^2 = 27$, 35 and 43 set $w = 10$.
3. For $T_{II\ p}^2 = 27$ and $w = 10$, 11 the procedure fails to set departure time. When w is 12 all the conditions are satisfied and the departure time is set as,

   $X_{I4} = T_{II1}^2 + w - t_{I2} = 27 + 12 - 17 = 22$ minutes.
4. No more departures of route I are un-assigned. Hence stop the procedure and label node 2 as ‘not possible’.

5. The arrival times of route I buses at node 2 are 18, 26, 34, 39 minutes.

Step 4: There are no more ‘new’ or ‘possible’ nodes.

Step 5: No more unset departures on any routes, stop.

The final results are shown in Table 4 and the seven simultaneous arrivals obtained are seven shown in Table 5. For example, $Y_{I,II,2,1,1} = 1$ shows that the first departure on route I and first departure of route II arrive simultaneously at node 2.

Table 4: Final Timetables for Example 1

Departures	Route I	Route II
1	1	0
2	9	8
3	17	16
4	22	-

Table 5: Table Showing Simultaneous Arrivals for Example - 1

Simultaneous Arrivals	
At Node 1	At Node 2
$Y_{I,II,1,1,1} = 1$	$Y_{I,II,2,4,1} = 1$
$Y_{I,II,1,2,1} = 1$	
$Y_{I,II,1,2,2} = 1$	
$Y_{I,II,1,3,2} = 1$	
$Y_{I,II,1,3,3} = 1$	
$Y_{I,II,1,4,1} = 1$	

The next example is a bus network with four routes and four nodes. The network and the travel times are shown in Figure 8. The inputs are given in Table 6 and Table 8. The planning horizon is [0, 45] minutes.

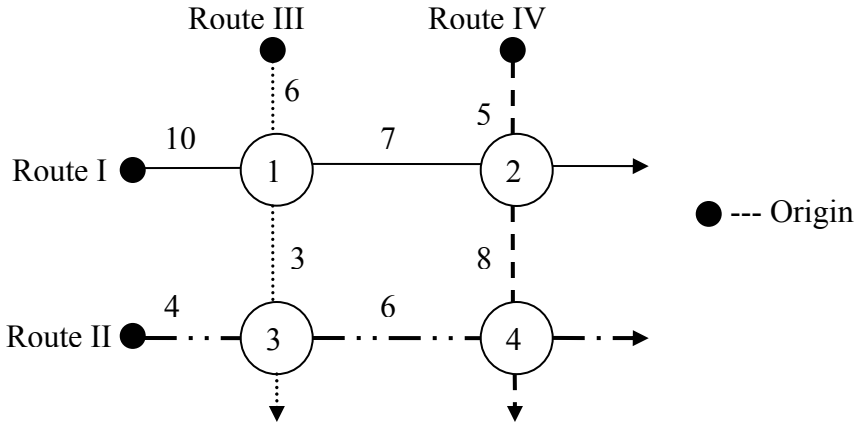

Figure 8: Numerical Example – 2

Table 6: Inputs for the Routes in Numerical Example – 2

Routes I	Minimum Headway, $H\min_i$	Maximum Headway, $H\max_i$	Frequency, f_i
I	8	15	2
II	10	15	3
III	10	15	3
IV	14	20	2

Table 7: Inputs for Nodes in Example - 2

Nodes, k	Number of routes	Minimum Waiting Time, $WT\min_k$	Maximum Waiting Time, $WT\max_k$
1	2	10	14
2	2	12	15
3	2	9	12
4	2	13	15

Step 1: Initializes all the nodes as 'new'.

Step 2: Identify the node '*SELECTED NODE'* k^*

1. Number of set departure times (= 0) for all nodes.
2. Number of crossing routes (= 2) for all nodes.
3. Maximum travel times are 10, 17, 9, and 13 respectively. The minimum value is 9, so k^* = Node 3.

Step 3: Performs *PROCEDURE 1* at node 3 on routes II and III.

1. The value of d is 10. The route i^* is III and *maxtime* = 9 minutes.
2. The first departure on route III (X_{III1}) is set at zero minutes.
3. Set $w = 9$.
4. For $w = 9$ the conditions are satisfied and the first departure is assigned as,

   $X_{II1} = (maxtime + w - t_{II3}) = 9 + 9 - 4 = 14$ minutes.
5. The number of departures created for route II and III are min (3, 3) =3. They are set as,

   For route II:

   $X_{II2} = X_{II1} + d = 14 + 10 = 24$

   $X_{II3} = X_{II2} + d = 24 + 10 = 34$

For route III:

$X_{III2} = X_{III1} + d = 0 + 10 = 10$

$X_{III3} = X_{III2} + d = 10 + 10 = 20$

6. The arrival time of route II buses at Node 3: T^3_{II1}, T^3_{II2}, T^3_{II3} are 18, 28, 38 and at Node 4: T^4_{II1}, T^4_{II2}, T^4_{II3} are 24, 34, 44 minutes. For route III the arrival times at Node 1: T^1_{III1}, T^1_{III2}, T^1_{III3} are 6, 16, 26 and at Node 3: T^3_{III1}, T^3_{III2}, T^3_{III3} are 9, 19, 29 minutes.
7. Label Node 1 and Node 2 as 'possible' and Node 3 as 'not possible'.

Step 4: There are 'possible' and 'new' nodes - go to step 2.

Step 2: Identify the node 'S*ELECTED NODE*' k^*.

1. Three arrival times have been set at nodes 1 and three at node. Hence the number of set arrivals at node 1 and node 4 are same = 3,
2. Number of routes passing through both these nodes is same = 2,
3. The maximum travel time of node 1 and node 4 are 10 and 13 respectively, hence k^* = Node 1.

Step 3: As node 1 is 'possible' *PROCEDURE 2* is performed.

1. Route i^* is route III and T^1_{III1}, T^1_{III2}, T^1_{III3} are 6, 16 and 26 respectively.
2. For Route I no departures have been assigned, so set p = 1. For $T^1_{III\,1}$ = 6 set w = 10.
3. The first departure time on route I is set as,

   $X_{I1} = T^1_{III1} + w - t_{I1} = 6 + 10 - 10 = 6$ minutes.
4. Second departure is still unassigned, go to Step 2.

2. $p = 2$ and for $T^1_{III\ 2}$, set $w = 10$.

3. The procedure sets the second departure as,

   $X_{I2} = T^1_{III2} + w - t_{I1} = 16 + 10 - 10 = 16$ minutes.

4. Since p = f_I, label Node 1 as 'not possible'.

5. The arrival times of route I buses at Node 2 are 23 and 33 minutes.

6. Label Node 2 as 'possible'.

Step 4: More 'possible' nodes, go to step – 2.

Step 2: Identify the node '*SELECTED NODE'* k^*.

1. The number of already set arrivals at node 2 is two (from route I buses). The number of already set arrivals at node 4 is three (from route II buses). Hence k^* = Node 4.

Step 3: *PROCEDURE 2* creates only first departure, $X_{IV\ 1}$ for route IV. For route IV only one departure time is set using this procedure.

Step 4: No more 'possible' node, continue.

Step 5: Route IV has an unassigned departure. Perform *PROCEDURE 3*. Since only one unassigned departure, assigns it as,

$X_{IV\ 2} = X_{IV\ 2} + H\min_{IV} = 8 + 14 = 22$ minutes.

Step 6: No more 'possible' nodes. Stop.

The final results are shown in Table 8. The number of simultaneous arrivals obtained is eight and are shown in Table 9. It was observed that the number of simultaneous arrivals obtained mainly depends on the waiting time limits considered at each node.

Table 8: Final Timetables for Example- 2

Departures	Route I	Route II	Route III	Route IV
1	6	14	0	8
2	16	24	10	22
3		34	20	

Table 9: Table Showing the Simultaneous Arrivals for Example – 2

At Node 1	At Node 2	At Node 3
$Y_{I,III,1,1,1} = 1$	$Y_{II,III,2,1,1} = 1$	$Y_{II,IV,3,2,1} = 1$
$Y_{I,III,1,2,2} = 1$	$Y_{II,III,2,2,2} = 1$	
$Y_{I,III,1,1,3} = 1$	$Y_{II,III,2,1,3} = 1$	
	$Y_{II,III,2,3,3} = 1$	

This chapter describes the algorithm used in our research to create timetables. A detailed description of each procedure and the steps involved in our heuristic approach are presented. To illustrate the algorithm certain numerical examples are provided.

CHAPTER 5

APPLICATION AND DISCUSSION

In this Chapter, the heuristic developed is applied to two real-life problems that incorporate realistic data on number of routes, nodes and waiting times. The model is tested for different waiting time ranges. The total number of synchronization for each node is given and compared with the maximum value of possible synchronizations.

5.1 Problem 1

The proposed model is applied to some real-life problems to evaluate the results. The following real-life problem shown in Figure 9 that was introduced by Ceder et al (2001) was considered. Some of the travel times have been modified to represent the waiting time component. This network has three nodes and six routes. The headway limits, $H\min_i$ and $H\max_i$ were set to be 14 and 20 minutes respectively for each route *i*. The planning horizon is [0, 240] minutes and frequency of each route f_i = 12. The travel times on each link are shown in the Figure 9. For this problem the waiting time limits, $WT\min_k$ and $WT\max_k$ are 5 and 10 minutes respectively at each node.

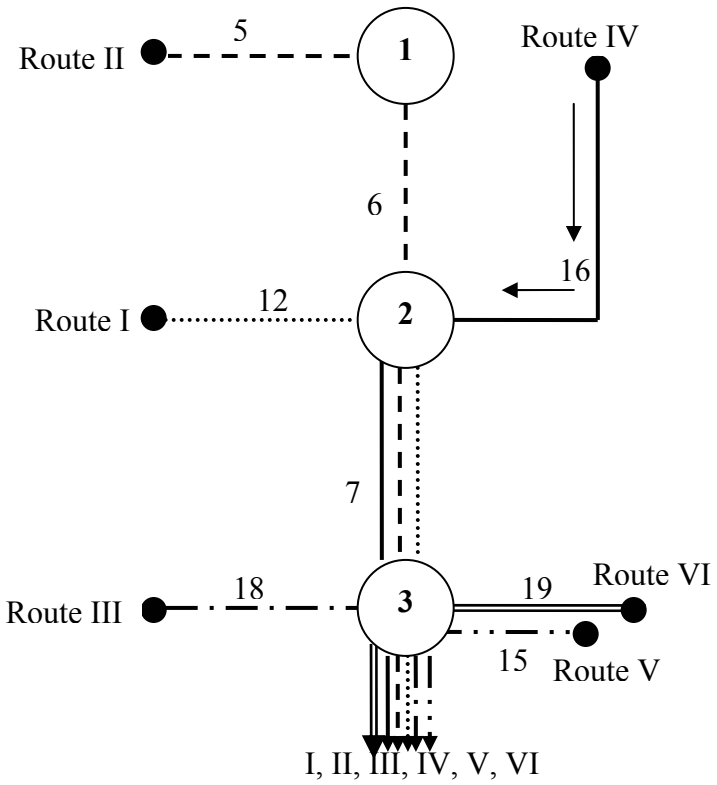

Figure 9: Real Life Problem – 1

The results obtained by our method are the departure times in minutes of 12 buses from the starting time of the planning horizon as shown in Table 10. The number of simultaneous arrivals obtained is: 58 - at Node 2 and at 187 - at Node 3 resulting in 245 total simultaneous arrivals for the system as show in Table 11.

The proposed algorithm first creates timetables of the route that takes maximum travel time to reach the selected node. It then creates timetables for each of the remaining routes comparing them with the already assigned route. Hence to find the maximum

Table 10: Timetables for Network Shown in Figure 9

Frequency	Route I	Route II	Route III	Route IV	Route V	Route VI
Bus 1	9	0	0	0	3	9
Bus 2	23	14	14	14	17	23
Bus 3	37	28	28	28	31	37
Bus 4	51	42	42	42	45	51
Bus 5	65	56	56	56	59	65
Bus 6	79	70	70	70	73	79
Bus 7	93	84	84	84	87	93
Bus 8	107	98	98	98	101	107
Bus 9	121	112	112	112	115	121
Bus 10	135	126	126	126	129	135
Bus 11	149	140	140	140	143	149
Bus 12	163	154	154	154	157	163

possible simultaneous arrivals at a node a pair wise comparison of all the route frequencies that meets at the node is made. In our model, according to the assumptions made, the maximum waiting time limit at a node can be the maximum headway of routes passing through that node. Hence, when we make pair wise comparisons, it is possible that one bus from any given route meets at most two buses on the other route. Considering this assumption, the maximum number of simultaneous arrivals for this problem is: at Node 1 - 0, at Node 2 - 72, and at Node 3 - 360. The comparison of the values obtained and the maximum possible values is shown in Figure 10.

Table 11: Simultaneous Arrivals for Different Waiting Times Limits for Problem 1

Waiting time limits at Node 1	Waiting time limits at Node 2	Waiting time limits at Node 3	Simultaneous Arrivals at Nodes			Total number of simultaneous arrivals in the system
			1	2	3	
[2, 4]	[2, 4]	[2, 4]	0	24	60	84
[2, 5]	[2, 5]	[2, 5]	0	24	60	84
[2, 6]	[2, 6]	[2, 6]	0	24	60	84
[2, 7]	[2, 7]	[2, 7]	0	24	60	84
[3, 5]	[3, 5]	[3, 5]	0	24	60	84
[3, 6]	[3, 6]	[3, 6]	0	24	60	84
[3, 8]	[3, 8]	[3, 8]	0	24	60	84
[4, 10]	[4, 10]	[4, 10]	0	46	115	161
[4, 12]	[4, 12]	[4, 12]	0	46	115	161
[5, 8]	[5, 8]	[5, 8]	0	24	60	84
[5, 10]	[5, 10]	[5, 10]	0	58	187	245

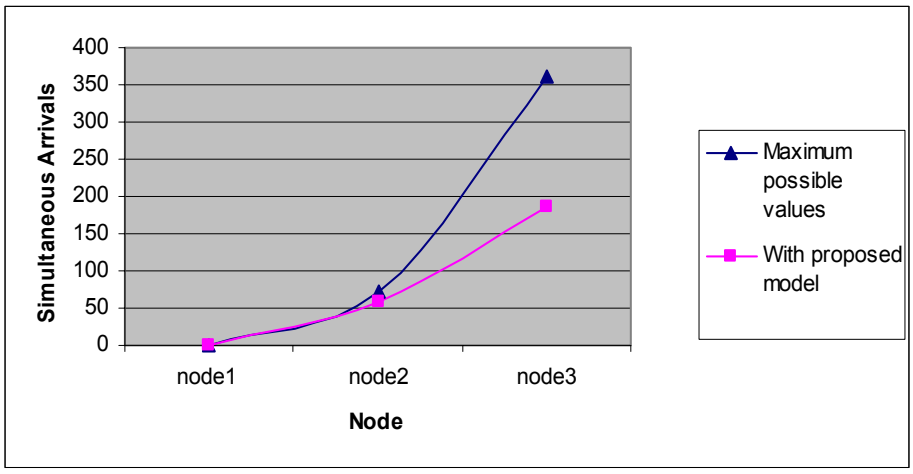

Figure 10: Comparison of the Number of Simultaneous Arrivals with Maximum Possible Values at Each Node

From this figure, it can be observed that the maximum number of synchronizations obtained using the proposed model for problem -1 is 80% at node 2 and 50% at node 3 when waiting time limits are [5, 10] – which represents the maximum waiting range. When waiting time limits are changed, different values are obtained, as shown in Table 11 and these values are plotted in Figure 11.

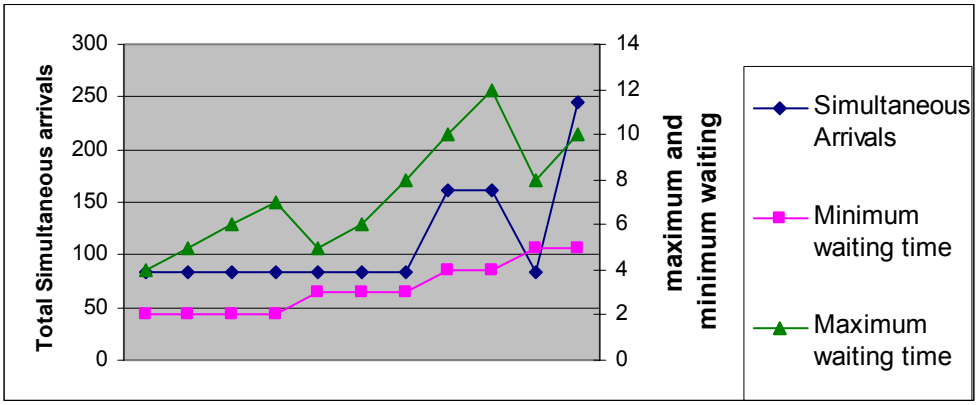

Figure 11: Simultaneous Arrivals for Different Waiting Time Limits for Problem -1

From this figure it can be observed that:

1. The number of simultaneous arrivals obtained depends on the waiting time limits.
2. Synchronizations are directly proportional to the waiting time limit range. That is, when the waiting time range is small (2-3 minutes) the number of synchronizations is smaller than when range provided is large (more than 4 minutes). This is because when the range is larger, there is a possibility that an arrival of a route can overlap with more than a single arrival on the other route at a particular node.

3. It can be observed that for smaller ranges of waiting time limits, the number of simultaneous arrivals is 24 - at node 2 and 60 - at node 3. These numbers correspond to the number of synchronizations when every arrival of a given route at a node meets at least one arrival from the route takes maximum travel time to reach that node. As the waiting time limit increases it is possible that more than one arrival from one route meets arrival on another route.

5.2 Problem 2

Problem 2 has nine nodes and five routes as shown in Figure 12. The minimum and maximum headways for each route are 15 and 30 minutes respectively and the frequency on each route is 4. The planning horizon considered is [0, 120] minutes. The travel time on each link is 5 minutes. The minimum and maximum waiting times at each node are 5 and 10 minutes respectively.

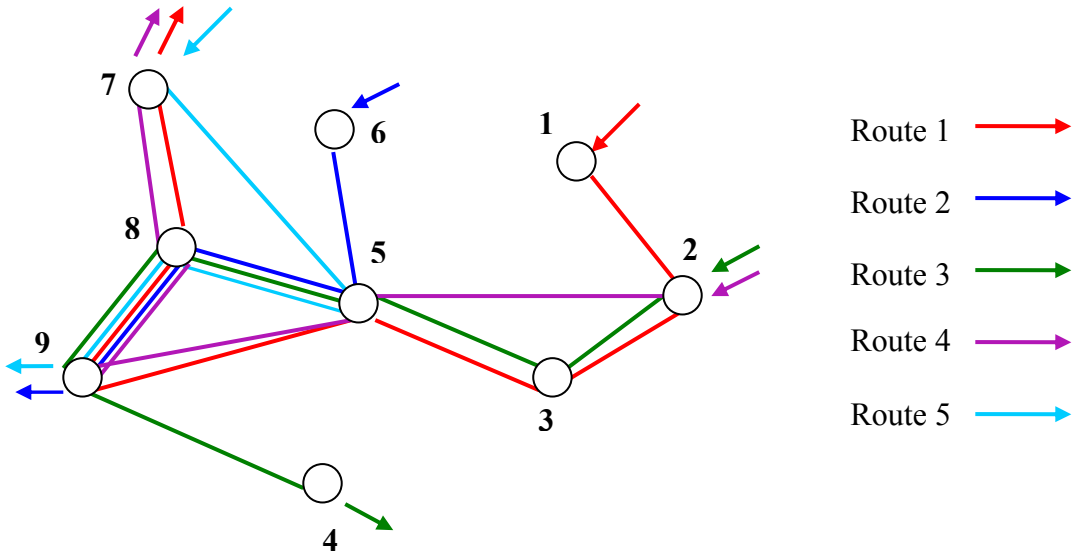

Figure 12: Real Life Problem-2

The departure times obtained using our procedure is given in Table 12. The total numbers of simultaneous arrivals obtained are 143, and the values obtained at each node separately and maximum possible values are shown in Table 13. The total number of simultaneous arrivals obtained by the model for each case is shown in this table.

Table 12: Timetables for Network Shown in Figure 12

Frequency, f_i	Route 1	Route 2	Route 3	Route 4	Route 5
1	0	5	0	5	5
3	15	20	15	20	20
3	30	35	30	35	35
4	45	50	45	50	50

Table 13: Simultaneous Arrivals by Changing Waiting Time Limits for Problem 2

$[WT\min_k, WT\max_k]$ at each Node	Simultaneous arrivals at Nodes									Total number of simultaneous arrivals in the system
	1	2	3	4	5	6	7	8	9	
[2, 4]	0	8	4	0	16	0	4	4	16	52
[2, 5]	0	12	4	0	16	0	7	16	28	83
[2, 6]	0	12	4	0	16	0	7	16	28	83
[2, 10]	0	15	4	0	16	0	14	46	37	132
[3, 5]	0	8	4	0	16	0	7	16	16	67
[3, 6]	0	8	4	0	16	0	7	16	16	67
[3, 8]	0	8	4	0	16	0	12	37	16	93
[4, 6]	0	8	4	0	16	0	9	25	16	78
[4, 10]	0	11	4	0	16	0	14	46	25	116
[5, 8]	0	8	4	0	16	0	9	24	24	85
[5, 10]	0	14	7	0	28	0	17	49	28	143
Maximum possible Simultaneous arrivals	0	24	8	0	80	0	24	80	80	

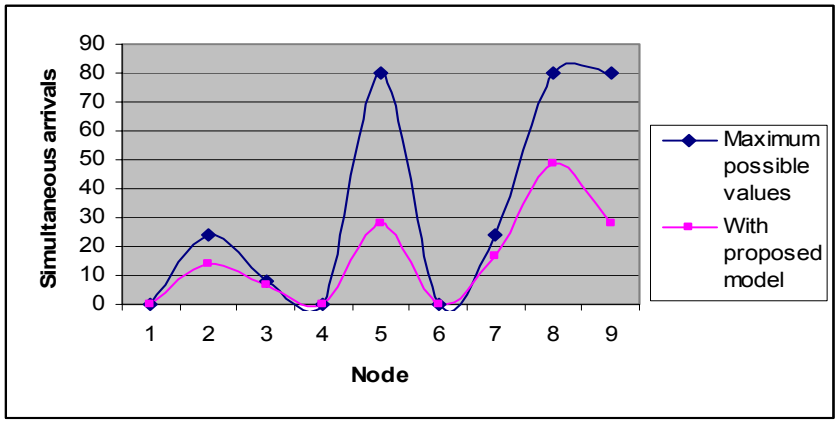

Figure 13: For Problem-2 Comparing the Number of Simultaneous Arrivals with Maximum Possible Values at Each Node

From the above figure it can be seen that the proposed model obtained 48% of maximum possible simultaneous arrivals in the entire network when waiting times limits are [5, 10] at all nodes. In this problem, as well as in problem-1, waiting time range increase will have a direct impact on the total number of synchronizations. Table 13 values are displayed in Figure 14.

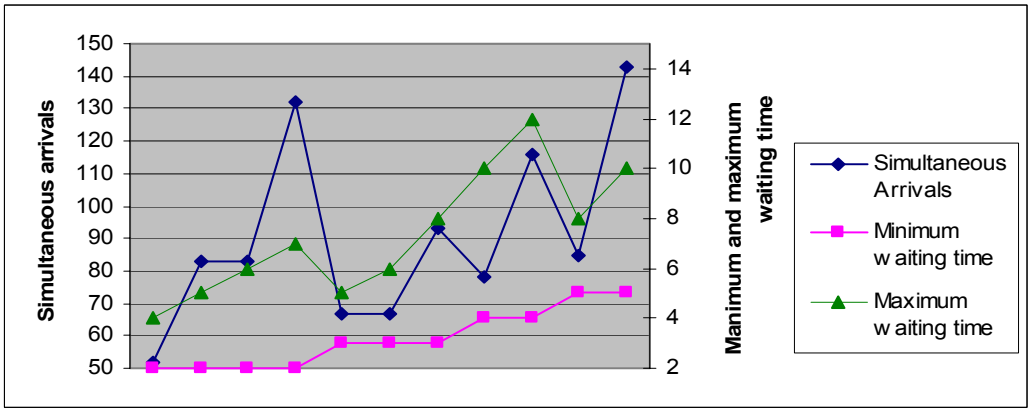

Figure 14: Simultaneous Arrivals for Different Waiting Time Limits for Problem 2

From these two problems we can conclude that there is always a trade off between the waiting time and transfer optimization and it is up to transit planners to decide upon how much waiting time should be allowed in order to increase synchronizations. In reality the waiting time limits depends on factors such as the structure of the network, headways, travel times, frequency etc. Hence it can be said that the performance of the model will be impacted by these network components.

CHAPTER 6

CONCLUSIONS AND FUTURE SCOPE

In this research a model to construct timetables for a bus transit network, incorporating waiting time limits at transfer nodes has been developed. From the practical point of view it is beneficial to have some waiting time between bus arrivals that provide passengers some buffer time to walk form one terminal to another or in case of traffic delays. This attribute also facilitates transit management by providing a more robust transit system.

The model assumes that the minimum waiting time at each node is a value greater than zero and a maximum waiting time value that does not exceeds the maximum headway of routes passing through that node. The challenge of the research was to set the timetables of the routes considering headways, in order to have minimum possible waiting time for the passengers and maximum simultaneous arrivals.

The mathematical model was formulated as a Mixed Integer Programming problem. For large networks, it is difficult to solve the problem using software for MIP problems, for that reason, a heuristic algorithm was developed. The heuristic is based on a node-oriented approach, which creates timetables of route passing through some selected node. The heuristic was implemented in C language and various scenarios were

tested for model verification. It was observed that the performance of the algorithm in terms of the number of simultaneous arrivals obtained is highly dependent on the network design itself.

The proposed model reduces waiting time by transfer coordination rather than decreasing headways (which is other option to achieve synchronization). It is assumed that passengers making connections prefer some waiting time greater than zero but refuse too large waiting times. Hence our model creates timetables considering this factor and obtains maximum possible number of simultaneous arrivals under these conditions. The synchronized arrivals set at each node, makes the transit system be 'on-time' standards and robust.

The following are some of the extensions that can be incorporated in our research:

1. In our model we assumed that the travel times are deterministic. But the randomness of bus travel times can cause inconveniences in transfers. Hence incorporating this randomness can be considered.
2. We assumed that there is a no-hold policy at nodes, i.e., the buses depart the node as soon as they arrive. The control variables in the present model are the waiting time limits at the nodes and the departure times are determined by these variables. Instead transfer optimization can be obtained by incorporating some holding policies for buses at nodes.

REFERENCES

Baaj, H., and Mahmassani, H., 1995. Hybrid route generation heuristics algorithm for the design of transit networks. *Transportation Research Part C*, 3 (1), 31-50.

Beasley, J.E., and Cao, B., 1996. A tree search algorithm for the crew scheduling problem. *European Journal of Operational Research*, 94, 517-526.

Bookbinder, J., and Désilets, A., 1992. Transfer optimization in a transit network. Transportation Science, 26 (2), 106-118.

Ceder A., 1986. Methods for creating bus timetables. *Transportation Research Part A*, 21A (1), 59-83.

Ceder, A., Golany, B., and Tal, O., 2001. Creating bus timetables with maximum synchronization. *Transportation Research Part A*, 35, 913-928.

Ceder, A., 2002. Public Transport Timetabling and Vehicle Scheduling. *Advanced Methods for Transit Operations and Service Planning*. Honk- Kong.

Gao, Z., Sun, H., and Shan, L., 2003. A continuous equilibrium network design model and algorithm for transit systems. *Transportation Research Part B,* Article in Press.

Haghani, A., and Banihashemi, M., 2002. Heuristic approaches for solving large-scale bus transit vehicle scheduling problem with route time constraints. *Transportation Research Part A*, 36, 309-333.

Palma, A., and Lindsey, R., 2001. Optimal timetables for public transportation. *Transportation Research Part B*, 35, 789-813.

Park, Y., and Song, S., 1997. Vehicle scheduling problems with time-varying speed. *Computers and Industrial Engineering*, 33 (3-4), 853-856.

Quak, C.B., 2003. Bus line planning, http://www.isa.ewi.tudelft.nl/~roos/buslineplanning.pdf

Wren, A., and Wren, D., 1995. A genetic algorithm for public transport driver scheduling. *Computers and Operations Research*, 22 (1), 101-110.

Yan, S., and Chen, H., 2002. A scheduling model and a solution algorithm for inter-city bus carriers. *Transportation Research Part A*, 36, 805-825.

APPENDICES

Appendix A: C Program

```
# include "stdio.h"
# define MAX 200
main ( )
{

int  i, j, k, wt, m, p , routes, loop_1, loop_2, time[MAX][MAX]={0},
        travel_time[MAX][MAX]={0},select_node[MAX], temp, TRUE=1,nr[MAX];
int  hmin[MAX], hmax[MAX], f[MAX], wt_min[MAX], wt_max[MAX],
        r_nodes[MAX],node_list[MAX][MAX],crit1_satisfied=0;
int  n_routes[MAX], route_tin_max[MAX]={0}, tin_max[MAX]={0}, FALSE =1,
        max_n_routes = 0, no = 0, number = 0,number_route= 0, max_wt = 0;
int  dep_time[MAX][MAX]={0},arr_time[MAX][MAX]={0},
        fixed_time[MAX][MAX], min_freq,freq,d_max,d_min, d, num_list[MAX]={0};
int  node_check[MAX]={0},procedure=0,assnd_node=0,assnd_route,
        check_route [MAX] = {0}, crit1_node [MAX] = {0};
int  max_arr[MAX][MAX]={0}, max_arr_node[MAX]={0},max_arr_crit1=0,count=0,
        min_tin_max=0, min_arr_node [MAX], plan_hor;
int  count_nodes[MAX]={0},stop_freq[MAX]={0},route_sel,max_count_nodes=0,
        choose = 0, assnd_node_freq =1, s_freq [MAX] = {0}, sync [MAX]={0},
         freq1, freq2, u, v;
        printf ("\n Enter the number of routes you have in the network\n");
        scanf ("%d",&routes);
        printf ("\nEnter total number of nodes in the network\n");
        scanf ("%d",&m);
        for (i=1; i<=routes; i++)
        {
           printf ("\nEnter the number of nodes on route %d\n", i);
           scanf ("%d", &r_nodes[i]);
        }
        for (i=1; i<=routes; i++)
        {
           for (j=1 ;j<=m; j++)
           {
                printf ("\nEnter 1if node %d is on route %d, Else enter 0\n", j, i);
                scanf ("%d", &node_list[i][j]);
                printf ("\nEnter the travel time on origin of route %d to node %d\n", i, j);
```

Appendix A (Continued)

```
        scanf ("%d", &travel_time[i][j]);
        }
    }
    for (i=1; i<=routes; i++)
    {
            printf ("\nEnter the Min Headway for the route %d\n", i);
            scanf ("%d", &hmin[i]);
            printf ("\nEnter the Max Headway for the route %d\n", i);
            scanf ("%d", &hmax[i]);
            printf ("\nEnter the frequency for the route %d\n", i);
            scanf ("%d", &f[i]);
    }
    for (i=1; i<=m; i++)
    {
            printf ("\nEnter the number of routes passing through node %d\n", i);
            scanf ("%d", &n_routes[i]);
            printf ("\n Enter the minimum allowed waiting time at node %d\n",i);
            scanf ("%d", &wt_min[i]);
            printf ("\n Enter the maximum allowed waiting time at node %d\n",i);
            scanf ("%d", &wt_max[i]);
    }
    for (i=1; i<=m; i++)
    {
            max_wt = wt_max[i];
         break;
    }
    for (i=1; i<=m; i++)
    {
     if(wt_max[i]> max_wt)
      {
              max_wt = wt_max[i];
      }
}
    printf ("\n Enter the planning horizon(in minutes) during which u want to
    schedule departure\n");
    scanf ("%d",&plan_hor);
    for ( i=1; i<=routes; i++)
    {
     if ((plan_hor  < ((f[i]-1)*hmin[i])) | | (plan_hor > max_wt + (f[i]*hmax[i])))
     {
      printf ("\n problem is infeasible due to planning horizon");
```

Appendix A (Continued)

```
exit (1);
        }

  }
          for ( i=1; i<=routes; i++)
          {
             if (hmin[i]>hmax[i])
             {
                 printf ("\n Problem is Infeasible as hmin[%d] is greater than hmax[%d]",i,i);
                   exit (1);
          }
       }
      for ( i=1; i<=routes; i++)
          {
                 stop_freq[i] = 0;
          }
          for ( j=1; j <=m; j++)
          {
            for (i=1; i<=routes; i++)
            {
                if(node_list[i][j] == 1)
                {
                    tin_max[j] = travel_time[i][j];
                    route_tin_max[j]=i;
                        break;
                }
             }
            }
          for ( j=1; j<=m; j++)
          {
            for (i=1; i<=routes; i++)
            {
               if (node_list[i][j] = = 1)
                {
                    if (travel_time[i][j] >= tin_max[j])
                    {
                       tin_max[j] = travel_time[i][j];
                       route_tin_max[j] = i;
                }
                }
             }
          }
          for ( i=1; i<=routes; i++)
```

Appendix A (Continued)

```
        {
          for (j=1; j<=m; j++)
          {
             printf ("\n Arrival time = %d", travel_time[i][j]);
                 printf ("\n Max Travel time = %d", tin_max[j]);
             printf ("\n Route with Max Travel time = %d", route_tin_max[j]);
          }
        }
         /* NODE SELECTION PROCEDURE */
while (assnd_node == 0)
  {
       no = 0;
       number = 0;
       number_route = 0;
   /*  STEP 1   */
     for ( j=1; j<=m; j++)
       {
         if (node_check[j] = = 0)
         {
        for (i=1; i <= routes; i++)
           {
                 if (node_list[i][j] = = 1)
                 {
                   if (check_route[i] == 1)
                   {
                         max_arr[i][j] = f[i];
                         crit1_node[j] = 1;
                    }
              }
            }
       }
          max_arr_node[j] = 0;
    }
       for (j=1; j<=m; j++)
       {
        if ((node_check[j] = = 0) && (crit1_node[j] = = 1))
        {
         for (i=1; i<=routes; i++)
         {
           if (node_list[i][j] = = 1)
           {
                if (max_arr[i][j] > max_arr_node[j])
```

Appendix A (Continued)

```
                    {
                       max_arr_node[j] = max_arr[i][j];
                  }
                 }
                }
               }
          }
for( j=1; j <= m; j++)
          {
            if ((node_check[j] = = 0)&&(crit1_node[j] = = 1))
            {
                  max_arr_crit1 = max_arr_node[j];
                  break;
            }
       }
        for (j=j+1; j<=m; j++)
          {
            if ((node_check[j] = = 0) && (crit1_node[j] = = 1))
            {
               if (max_arr_node[j] >= max_arr_crit1)
                  {
                    max_arr_crit1 = max_arr_node[j];
                  }
            }
       }
       count = 1;
          for (j=1; j<=m; j++)
          {
            if ((node_check[j] = = 0) && (crit1_node[j] = = 1))
            {
               if (max_arr_node[j] = = max_arr_crit1)
                  {
                    if (count = = 1)
                    {
                     no = j;
                     crit1_satisfied = 1;
                  }
                    else
                    {
                     no = 0;
                     crit1_satisfied = 0;
                     }
```

Appendix A (Continued)

```
            count++;
            }
        }
 }
 if(crit1_satisfied = = 1)
{
        node_check[no] = 1;
}
if(crit1_satisfied = = 0)
{
     /*  STEP 2   */
            for ( j=1; j<=m; j++)
    {
      if(node_check[j] = = 0)
      {
            printf ("\n Crit2, node_check[%d] = %d", j, node_check[j]);
            max_n_routes = n_routes[j];
            number_route = j;
          break;
      }
 }
    for (j=j+1; j<=m; j++)
    {
      if (node_check[j] == 0)
      {
       if (max_n_routes <= n_routes[j])
       {
            max_n_routes = n_routes[j];
            number_route = j;
       }
       }
    }
    printf ("\n Max_n_routes = %d", max_n_routes);
    for (j=1; j<=m; j++)
    {
     if (node_check[j]==0)
     {
      if (n_routes[j]==max_n_routes)
      {
      number = number+1;
      num_list[j] = 1;
  }
```

Appendix A (Continued)

```
        }
      }
         if ( number = =1)
         {
          no = number_route;
          node_check[no] = 1;
         }
     for ( j=1; j<=m; j++)
         {
         printf ("\n Max Arrival Time for Node %d is %d", j, tin_max[j]);
         }
/*  STEP 3   */
         for (j=1; j<=m; j++)
         {
           if ((node_check[j] == 0) && (num_list[j]==1))
           {
             min_tin_max = tin_max[j];
               break;
           }
      }
         for(j=1; j<=m; j++)
         {
          if((node_check[j] = = 0) && (number!=1))
              {
                  if (num_list[j]==1)
               {
                if((tin_max[j] <= min_tin_max) && (n_routes[j] = = max_n_routes))
                    {
                      no = j;
                 }
               }
              }
     }
    node_check[no] = 1;
    }
    printf ("\n The Node selected is %d", no);
  for (j=1; j<=m; j++)
  {
        num_list[j]=0;
  }
/* END OF NODE SELECTION */
        procedure = 0;
```

Appendix A (Continued)

```
/* CHECK IF ANY PREVIOUS ROUTE ON SELECTED NODE HAS BEEN
ASSIGNED*/
for (i=1; i<=routes; i++)
        {
          if (node_list[i][no] == 1)
           {
                 if (check_route[i] == 1)
                        procedure = 1;
           }
      }
/*PROCEDURE 1*/
        if (procedure = = 0)
        {
              for (i=1; i<=routes; i++)
              {
               if (node_list[i][no] = = 1)
               {
                     d_min = hmin[i];
                     d_max = hmax[i];
                      break;
               }
            }
              for (i=1; i<=routes; i++)
              {
                 if(node_list[i][no] == 1)
                 {
                        if(d_min <= hmin[i])
                          d_min = hmin[i];
                        if(d_max >= hmax[i])
                          d_max = hmax[i];
                        if(d_max < d_min)
                  d = 0;
                        else
                          d = d_min;
                   }
        }
                  printf("\n Value of d = %d", d);
        for ( i=1; i<=routes; i++)
        {
                    if (node_list[i][no] == 1)
                     {
                        min_freq = f[i];
```

Appendix A (Continued)

```
                    break;
                  }
        }
        for (i=1; i<=routes; i++)
                {
                if (node_list[i][no] = = 1)
                {
                   if (min_freq >= f[i])
                   {
                         min_freq = f[i];
                   }
                }
                 }
                printf ("\n Minimum freq = %d",min_freq);
           for (i=1; i<=routes; i++)
                {
                 if (node_list[i][no] = = 1)
                 {
                if(i = = route_tin_max[no] )
                {
                     arr_time[i][1] = tin_max[no];
                     dep_time[i][1] = (arr_time[i][1] - travel_time[i][no]);
                     stop_freq[i] = 1;
                     check_route[i] = 1;
                   }
                    if(i!= route_tin_max[no])
                    {
if (((tin_max[no] - wt_min[no]) >= travel_time[i][no])&&(tin_max[no]-wt_min[no]-
                                        travel_time[i][no])>=0)
                     {
                      arr_time[i][1] = (tin_max[no] - wt_min[no]);
                      dep_time[i][1] = (arr_time[i][1] - travel_time[i][no]);
                      stop_freq[i] = 1;
                      check_route[i] = 1;
                     }
                        else
                        {
                          for(wt=wt_min[no];wt<=wt_max[no];wt++)
                            {
        if((tin_max[no]+wt>=travel_time[i][no])&&((tin_max[no]+wt-
        travel_time[i][no])<=hmax[i]))
                                    {
```

Appendix A (Continued)

```
                                    arr_time[i][1] = (tin_max[no]+wt);
                                    dep_time[i][1] = (arr_time[i][1] - travel_time[i][no]);
                                    stop_freq[i] = 1;
                                    check_route[i] = 1;
                                    break;
                                }
                            }
                        }
                    }
            for( freq=2; freq<=min_freq; freq++)
                    {
                        if(d = = 0)
                        {
                          stop_freq[i] = freq - 1;
                          break;
                          break;
                        }
                          dep_time[i][freq] = d + dep_time[i][freq-1];
                          arr_time[i][freq] = dep_time[i][freq]+travel_time[i][no];
                          stop_freq[i] = freq;
                    }
                }
               }
                for (i=1; i<=routes; ++)
                {
                    if (node_list[i][no]= =1)
                    {
                     for (freq=1; freq<=stop_freq[i]; freq++)
                     {
 printf("\n the departure time of %d bus on route %d is= %d\n",freq,i,dep_time[i][freq]);
printf("\n the arrivals time of %d bus on route %d at node %d
          is=%d\n",freq,i,no,arr_time[i][freq]);
                     }
                }
               }
 } /*END OF PROCEDURE 1 IF LOOP */

 else /* PROCEDURE 2 */
          {
              assnd_route = 0;
 printf("\n Starting Middle, check_Route[%d] = %d, assnd_node = %d",1,check_route[1],
          assnd_node);
```

Appendix A (Continued)

```
    for (i=1; i<=routes; i++)
     {
       if (node_list[i][no] = = 1)
       {
              if ((check_route[i] = = 1) && (stop_freq[i]= =f[i]))
              {
                    assnd_route = i;
                    for (freq=1; freq<=stop_freq[i]; freq++)
                    {
                        fixed_time[no][freq] = dep_time[i][freq] + travel_time[i][no];
                     }
                    break;
            }
     }
    printf ("\n Exiting Flag Post 1, assnd Route = %d", assnd_route);
      }
  printf ("\n The assigned route = %d", assnd_route);
  for (freq=1;freq<=stop_freq[assnd_route];freq++)
  {
   printf ("\n the fixed_time[no][freq]=%d\n", fixed_time[no][freq]);
  }
  for (i=1; i<=routes; i++)
  {
   if (node_list[i][no] = =1)
   {
 if (i! = assnd_route)
    {
    for(freq=1;freq<=stop_freq[i];freq++)
    {
     arr_time[i][freq]= dep_time[i][freq]+travel_time[i][no];
     }
 }
}
}
  loop_1=0;
  loop_2=0;
  if(assnd_route != 0)
  {
    for( i=1; i<=routes; i++)
     {
      if (node_list[i][no] = = 1)
       {
```

Appendix A (Continued)

```
                if ((i!= assnd_route) && (stop_freq[i]< f[i]))
                {
                  for (freq=1; freq<=stop_freq[assnd_route]; freq++)
                  {
                        for (wt = wt_min[no]; wt <= wt_max[no]; wt++)
                        {
                              k = stop_freq[i]+1;
  if ((arr_time[i][k-1]<=(fixed_time[no][freq]- wt))&&((fixed_time[no][freq]- wt-
        travel_time[i][no])>0))
              {
                  arr_time[i][k] = (fixed_time[no][freq] - wt);
                  dep_time[i][k] = (arr_time[i][k] - travel_time[i][no]);
                  stop_freq[i] = k;
                  assnd_node_freq = assnd_node_freq + 1;
                  loop_1= 1;
  if ((k>1)&&((dep_time[i][k] - dep_time[i][k-1]) < hmin[i]))
            {
                  arr_time[i][k] = 0;
                  dep_time[i][k] = 0;
                  stop_freq[i] = k-1;
                  assnd_node_freq = assnd_node_freq - 1;
                  k=k-1;
                  loop_1 = 0;
                   if(k >= f[i])
                  {
                  break;
                  break;
                  }
            }
        break;
        }
if  (((fixed_time[no][freq] + wt) >= arr_time[i][k-1])&&((fixed_time[no][freq] + wt-
        travel_time[i][no])<= (hmax[i]+dep_time[i][k-1])))
            {
             arr_time[i][k] = (fixed_time[no][freq] + wt);
             dep_time[i][k] = (arr_time[i][k] - travel_time[i][no]);
             stop_freq[i] = k;
             assnd_node_freq = assnd_node_freq + 1;
             loop_2 = 1;
             printf("\n Reached loop2 for wt = %d and freq = %d", wt, freq);
              if ( (dep_time[i][k] - dep_time[i][k-1]) < hmin[i])
                    {
```

Appendix A (Continued)

```
                    arr_time[i][k] = 0;
                    dep_time[i][k] = 0;
                    stop_freq[i] = k-1;
                   assnd_node_freq = assnd_node_freq - 1;
                   k=k-1;
                   loop_2 = 0;
                    if(k >= f[i])
                    {
                          break;
                          break;
                   }
              }
          }
     if ((loop_1==0)&&(loop_2 == 0))
      {
        k = stop_freq[i];
      }
          printf("\n Exiting Flag Post 2");
    }
     if (k>= f[i])
           {
              break;
              break;
              }
       }
    }
   }
 check_route[i] = 1;
  }
}
   procedure3  = 0;
           for ( j=1; j<=m; j++)
           {
             for ( i=1; i<=routes; i++)
             {
              if (node_list[i][j] == 1)
                  {
               if (node_check[j] == 0)
               {
                     procedure 3 = 0;
                      break;
               }
```

Appendix A (Continued)

```
            else if((node_check[j] == 1) && (stop_freq[i] < f[i]))
           {
             procedure3 = 1;
           }
          }
         }
      }
       }   /*END OF ELSE LOOP FOR PROCEDURE 2    */
/*PROCEDURE 3  */
       if (choose = = 1)
       {
        for  ( i=1; i<=routes; i++)
          {
            count_nodes[i] = 0;
                  for ( j=1; j<=m; j++)
                  {
                           if((node_list[i][j] = = 1) && (dep_time[i][f[i]]= =0))
                           {
                                  count_nodes[i] = count_nodes[i] + 1;
                           }
                  }
           }
         for ( i=routes; i>=1; i--)
                {
                  if(dep_time[i][f[i]] = = 0)
                  {
                     max_count_nodes = count_nodes[i];
                     route_sel = i;
                     break;
                  }
               }
                  for( i=routes; i>=1; i--)
                  {
                    if((dep_time[i][f[i]] = = 0) && (count_nodes[i]>=max_count_nodes))
                    {
                           route_sel = i;
                    }
                  }
           for (freq = (stop_freq[route_sel]+1); freq<=f[route_sel]; freq++)
            {
            dep_time[route_sel][freq] = dep_time[route_sel][freq - 1] + hmin[route_sel];
               stop_freq[route_sel] = freq;
```

Appendix A (Continued)

```
                break;
              }
              for ( i=1; i<=routes; i++)
              {
                    for (j=1; j<=m; j++)
                    {
                          if ((node_list[i][j] = = 1) && (stop_freq[i] < f[i]))
                          {
                                node_check[j] = 0;
                          }
                          else if(node_list[i][j] == 1)
                          {
                                node_check[j] = 1;
                          }
                     }
               }
     }  /* END OF PROCEDURE 3   */
 for ( j =1; j < =m; j++)
   {
    for (i=1; i<=routes; i++)
              {
                   if (node_list[i][j] == 1)
                   {
            if ((check_route[i] != 0) && (dep_time[i][f[i]] != 0))
                    {
                          assnd_node = 1;
                    }
                    else
                    {
                          assnd_node = 0;
                          break;
                          break;
                     }
                  }
              }
          }
         crit1_satisfied = 0;
} /* END OF WHILE LOOP */
/* Loops for printing the Final Results */
for (i=1; i<=routes; i++)
{
  for ( freq=1; freq<=f[i]; freq++)
```

Appendix A (Continued)

```
   {
printf ("\n Departure Times for Bus %d on route %d = %d", freq, i, dep_time[i][freq]);
 printf("\n");
   }
 }
 /* Finding the number of simultaneous arrivals*/
  for( j=1; j<=m; j++)
   {
     for (i=1; i<=routes-1; i++)
     {
        for (p=i+1; p <= routes; p++)
           {
            if ((node_list[i][j]==1)&&(node_list[p][j]==1))
            {
             for (freq1=1; freq1 <= f[i]; freq1++)
             {
              for (freq2=1; freq2 <= f[p]; freq2++)
              {
                    u = dep_time[i][freq1]+travel_time[i][j];
                    v = dep_time[p][freq2]+travel_time[p][j];
                    if ((abs(u-v) >=  wt_min[j]) && (abs(u-v) <=  wt_max[j]))
                    {
                     sync[j] = sync[j] + 1;
                    }
              }
          }
         }
        }
       }
      }
    for (j=1; j<=m; j++)
     {
      printf("\n the number of synchronizations at node %d is = %d\n", j, sync[j]);
       }
}/* END OF MAIN*/
```