**Technical Report Documentation Page**

| | | |
|---|---|---|
| 1. Report No.<br>SWUTC/12/476660-00078-1 | 2. Government Accession No. | 3. Recipient's Catalog No. |
| 4. Title and Subtitle<br>USING REAL TIME TRAVELER DEMAND DATA TO OPTIMIZE COMMUTER RAIL FEEDER SYSTEMS | | 5. Report Date<br>August 2012<br>6. Performing Organization Code |
| 7. Author(s)<br>Yao Yu, Randy Machemehl | | 8. Performing Organization Report No.<br>476660-00078-1 |
| 9. Performing Organization Name and Address<br>Center for Transportation Research<br>University of Texas at Austin<br>1616 Guadalupe Street, Suite 4.200<br>Austin, Texas 78701 | | 10. Work Unit No. (TRAIS)<br>11. Contract or Grant No.<br>DTRT07-G-0006 |
| 12. Sponsoring Agency Name and Address<br>Southwest Region University Transportation Center<br>Texas A&M Transportation Institute<br>Texas A&M University System<br>College Station, Texas 77843-3135 | | 13. Type of Report and Period Covered<br>14. Sponsoring Agency Code |

15. Supplementary Notes

Supported by a grant from the US Department of Transportation, University Transportation Centers Program.

16. Abstract

This report focuses on real time optimization of the Commuter Rail Circulator Route Network Design Problem (CRCNDP). The route configuration of the circulator system – where to stop and the route among the stops – is determined on a real-time basis by employing adaptive Tabu Search to timely solve a Mixed Integer Program (MIP) problem with an objective to minimize total cost incurred to both transit users and transit operators. Numerical experiments are executed to find the threshold for the minimum fraction of travelers that would need to report their destinations via smart phone to guarantee the practical value of optimization based on real-time collected demand against a base case defined as the average performance of all possible routes. The adaptive Tabu Search Algorithm is also applied to three real-size networks abstracted from the Martin Luther King (MLK) station of the new MetroRail system in Austin, Texas.

| | | | |
|---|---|---|---|
| 17. Key Words<br>Commuter Rail Circulator Network Design Problem (CRCNDP), Austin Texas, Tabu Search, Mixed Integer Program | | 18. Distribution Statement<br>No restrictions. This document is available to the public through NTIS:<br>National Technical Information Service<br>5285 Port Royal Road<br>Springfield, Virginia 22161 | |
| 19. Security Classif.(of this report)<br>Unclassified | 20. Security Classif.(of this page)<br>Unclassified | 21. No. of Pages<br>111 | 22. Price |

# USING REAL TIME TRAVELER DEMAND DATA TO OPTIMIZE COMMUTER RAIL FEEDER SYSTEMS

by

Yao Yu, B.E., M.S.E.
Graduate Research Assistant
The University of Texas at Austin

and

Randy Machemehl, P.E.
Professor
The University of Texas at Austin

Report SWUTC/12/476660-00078-1
Project 476660-00078

Performed in cooperation with the
Southwest Region University Transportation Center
Center for Transportation Research
1616 Guadalupe Street, Suite 4.202
The University of Texas at Austin
Austin, Texas 78712

August 2012

## ABSTRACT

This report focuses on real time optimization of the Commuter Rail Circulator Route Network Design Problem (CRCNDP). The route configuration of the circulator system – where to stop and the route among the stops – is determined on a real-time basis by employing adaptive Tabu Search to timely solve a Mixed Integer Program (MIP) problem with an objective to minimize total cost incurred to both transit users and transit operators. Numerical experiments are executed to find the threshold for the minimum fraction of travelers that would need to report their destinations via smart phone to guarantee the practical value of optimization based on real-time collected demand against a base case defined as the average performance of all possible routes. The adaptive Tabu Search Algorithm is also applied to three real-size networks abstracted from the Martin Luther King (MLK) station of the new MetroRail system in Austin, Texas.

## EXECUTIVE SUMMARY

Commuter rail systems, operating on unused or under-used railroad rights-of-way, are being introduced into many urban transportation systems. Since locations of available rail rights-of-way were typically chosen long ago to serve the needs of rail freight customers, these locations are not optimal for commuter rail users. The majority of commuter rail users do not live or work within walking distance of potential commuter rail stations, so provision of quick, convenient access to and from stations is a critical part of overall commuter decisions to use commuter rail.

Minimizing access time to rail stations and final destinations is crucial if commuter rail is to be a viable option for commuters. Well-designed feeder routes or circulator systems are regarded as potential solutions to provide train station to ultimate destination access. Transit planning for main line or feeder routes relies upon static demand estimates describing a typical day. Daily and peak-hour demands change in response to the state of the transport system, as influenced by weather, incidents, holiday schedules and many other factors.

Recent marketing successes of "smart phones" might provide an innovative means of obtaining real time data that could be used to identify optimal paths and stop locations for commuter rail circulator systems. Such advanced technology could allow commuter rail users to provide real-time final destination information that would enable real time optimization of feeder routes.

This report focuses on real time optimization of the Commuter Rail Circulator Route Network Design Problem (CRCNDP). The route configuration of the circulator system – where to stop and the route among the stops – is determined on a real-time basis by employing adaptive Tabu Search to timely solve a Mixed Integer Program (MIP) problem with an objective to minimize total cost incurred to both transit users and transit operators. Numerical experiments are executed to find the threshold for the minimum fraction of travelers that would need to report their destinations via smart phone to guarantee the practical value of optimization based on real-time collected demand against a base case defined as the average performance of all possible routes. The adaptive Tabu Search Algorithm is also applied to three real-size networks abstracted from the Martin Luther King (MLK) station of the new MetroRail system in Austin, Texas.

## DISCLAIMER

## ACKNOWLEDGEMENTS

The authors recognize that support for this research was provided by a grant from the U.S. Department of Transportation, University Transportation Centers Program to the Southwest Region University Transportation.

TABLE OF CONTENTS

# LIST OF FIGURES

## LIST OF TABLES

# CHAPTER 1. INTRODUCTION

Population and employment growth are the two primary factors resulting in rush hour travel demand growth. The demographic and economic changes that have taken place over the past 50 years are dramatic. The population has grown from 165 million in 1955 to 295 million in 2005 while economic growth and employment growth have remained healthy and strong to meet the demand of the ever-increasing population. The population growth trend, according to U.S. Census projections, is anticipated to continue through 2055. By that time, more of the population will choose to live in the metropolitan areas due to the economic incentive in these areas and the lasting economic vitality will again promote employment growth.

Traffic congestion due to travel demand exceeding the service capacity of urban transportation systems is a long-standing problem among all issues confronting transportation planners. According to the 2011 version of the annual Texas Transportation Institute Mobility Report (Schrank & Lomax, 2011) urban congestion (based on wasted time and fuel) is estimated to cost about $115 billion in the year 2011 alone. The bulk of transportation research mainly focuses on increasing highway capacity to alleviate congestion. Constraints in construction and maintenance budgets, the lack of available right-of-way and other political and environmental factors often cap the capacity of current highways. With the current demand for urban transportation far exceeding the service supply of existing highway systems, alternative means must be further explored to provide more reliable, accessible and efficient transportation systems.

Commuter rail has been the subject of increasing interest within the United States in recent years, chiefly because it offers the potential for providing attractive, high-quality rapid transit service at a more reasonable cost when compared with other types of urban rail systems, such as light rail or heavy rail. Commuting trips, which account for a large fraction of peak-period transport demand, are usually carried out daily during approximately the same peak hours and usually follow a fixed route with an aversion toward congestions. All the facts that justify commuter rail systems target commuting trips as an appropriate opportunity for an effective and practicable solution to ever-worsening congestion.

## STUDY MOTIVATION

Commuter rail development and expansion programs are in progress at a number of locations in the United States currently. The Federal Transit Administration (FTA), the primary federal funding source for commuter rail projects, allocates investment through its New Starts program. According to FTA officials, FTA will not award full-funding grant agreements to commuter rail projects unless the commuter rail agency resolves relevant track access issues. Table 1 lists the current commuter rail projects in the New Starts Program Annual report for the fiscal year 2012 (FTA, 2011) and indicates whether these new commuter rail systems are going to use existing track or construct new guideway.

**Table 1. Guideway for Commuter Rail Projects.**

| | New Starts Program on commuter rail (Annual Report for Fiscal Year 2012) | Use existing track or construct new track |
|---|---|---|
| 1 | Wilmington to Newark Commuter Rail Improvements | Construction of a third track |
| 2 | Weber County to Salt Lake City Commuter Rail | Operates within an existing railroad corridor parallel to Interstate 15 (I-15), utilizes right-of-way previously acquired by UTA under a rail corridor preservation plan with certain facilities already in place |
| 3 | Northeast Corridor | This section of the Northeast Corridor is currently used only for Amtrak and freight operations |
| 4 | Pawtucket/Central Falls Commuter Rail Station | Facilities already in place |
| 5 | Eagle Commuter Rail | Construction of the east corridor and the Gold line |
| 6 | Central Florida Commuter Rail Transit - Initial Operating Segment | Sharing track with existing freight and Amtrak services |

As indicated in the table, among the six new Commuter Rail projects in fiscal year 2012, only two of them propose to construct new track, and all the others are going to use existing track. Most of the reason is that Right-of-Way (ROW) for commuter rail – obtained either through leasing track from freight-rail operators or using abandoned track is cheap and easily accomplished. The use of existing rail ROW is a major factor in the evaluation of urban transportation options. This trend will continue as ROW continues to be difficult for transit operators to obtain.

However, there are detriments to using existing track, especially those abandoned. Grava (2003) has discussed the issue of commuter rail placement:

> A basic issue related to the use of existing rail alignments is their placement. They were usually established more than 100 years ago to serve a completely different city configuration and respond to the needs of that time. They are not necessarily central to the current corridors of residential and commercial activity.

Much of the rail infrastructure was developed to serve freight traffic to or from industrial centers. These industrial centers are generally associated with low-density land use. Also, freight rail lines usually cause noise and vibration. Citizens generally do not reside near these industrial centers because of the inconvenience and bad environment.

In another words, population densities of the places that the existing rail tracks intend to serve are fairly low. In urban areas with high-density population, additional ROW is not generally available or usually too expensive to obtain to construct rail infrastructure.

Minimizing access time to rail stations and final destinations is critical if commuter rail is to be a viable option for commuters. From the perspective of commuter rail operators, collector and distributor systems are a means to gather potential rail passengers spread out around the railroad to ameliorate the effect of low population density upon the commuter rail system, to maintain a reasonable loading ratio and thus system feasibility. From the perspective of commuter rail users, the circulator systems provide access to commuter rail since the majority of commuter rail users do not live or work within walking distance of existing rail or proposed rail stations. This represents a way to help cultivate and retain ridership in the short term until development near the rail stations can enhance the stability of the system and its ridership.

The two ends of the commute trip have different characteristics. The home end of the trip can usually take advantage of available land in suburban regions when commuter rail is first introduced and access can be provided through the use of park and ride facilities. As these suburban regions mature, parking shortages can arise as continuing development in these regions consumes spare land. At the work end of the trip, it is highly likely that a commuter will not have a personal vehicle at the destination station since he has already driven to the station at the home end, and as discussed previously, it is also unlikely that the commuter will work within walking distance of the destination station, as many station locations are not in the central business district. Additional modes providing access from stations at this end of trip are needed to insure commuter rail feasibility.

This report will assume that park and ride facilities will provide access to the commuter rail line at the home end, and at the work end a circulator bus service is the only available mode. The method presented focuses on the destination end of the commute trip to an urban work location. Instead of solving the Commuter Rail Circulator Network Design Problem (CRCNDP) based upon static demand estimates describing a typical day, real-time daily and peak-hour demands in response to the state of the transport system, as influenced by weather, incidents, holiday schedules and many other factors are collected for route planning. A method that optimally designs the circulator network at the destination end of the commuter rail trip with real-time demand will be developed to better serve passengers at the work trip destination end. Note that having all rail passengers boarding the circulator bus is just one extreme scenario that the transit operator must deal with. Since the case study in this report is a circulator network design for the Martin Luther King (MLK) station of the commuter rail in Austin, TX, where there is no destination within a walk distance to the station, this assumption is fairly reasonable. Regarding cases where other station to destination travel modes are available, i.e. bus, taxi, smart car or walking, this CRCNDP method can still be widely applied since real-time destination information can still be obtained by simply asking passengers how and where they want to go, and the model can still be used to optimize a feeder system for passengers choosing to board the circulator.

## STUDY OBJECTIVES

The goal of this report is to develop a flexible algorithmic solution framework to implement real time computer-aided design of commuter rail circulator networks and provide various good solutions to accommodate real-time travel demand requirements. The proposed work in this report is intended to fulfill the following objectives:

1. To identify knowledge that can reflect current related practice and existing rules of thumb for commuter rail circulator route network design issues;
2. To develop a robust systematic efficient heuristic algorithm that can incorporate the above knowledge, and to test a set of designed algorithmic procedures to search intelligently for an optimal solution;
3. To explicitly account for the multi-objective nature of the commuter rail circulator network design problem. This includes exploring the capability to evaluate performance measures from the points of view of both the operator and transit users for service options and to develop the ability to ascertain the characteristics of tradeoffs between conflicting performance measure variables inherent in the commuter rail circulator network problem.

## EXPECTED CONTRIBUTIONS

Due to the complexity and combinatorial NP-hard nature of the real-time Commuter Rail Circulator Network Design Problem (CRCNDP), traditional exact analytical optimization methodology is impracticable. The proposed work in this report is oriented to developing Meta-heuristic approaches to finding an acceptable and operationally implementable route network that can provide alternative design concepts in a reasonable time domain. The solution methodology differs from existing approaches in many aspects and the expected contributions from this report are summarized as follows:

1. Ability to apply a designed algorithmic procedures to search intelligently for an optimal solution without the loss of applicable service planning guidelines and the transit planners' knowledge and expertise;
2. Ability to produce a route network reflecting the inherent tradeoffs between conflicting performance-measures. This includes explicit consideration of the multi-objective nature of the commuter rail circulator network design problem and the capability to evaluate performance measures and service options from the points of view of both the operator and transit users;
3. Ability to systematically apply heuristic algorithms to produce quality solutions for the CRCNDP and identify the most appropriate one(s) under certain circumstances;
4. Ability to obtain optimal or near-optimal CRCNDP solutions within a strict time limit to realize real-time operation.

**Report Overview**

In this chapter, the significance and motivation of working on the real-time optimization of the Commuter Rail Circulator Network Design Problem (CRCNDP) has been discussed, and study objective and expected contributions are also described.

Chapter 2 presents a review of the literature that is related to the CRCNDP problem. Since CRCNDP is identified as a sub-problem of transit route design problem, a significant amount of literature focusing on general transit network design problems to find out the route configuration and other operating strategies was reviewed. Model objective and decision variables were taken a close look, both analytical model and heuristic models engaged in solving these problems are discussed. Three meta-heuristics methods and their related literature are reviewed in detail as implications for this report.

Chapter 3 provides a detailed explanation of the model formulation for the real-time CRCNDP. A mathematical nonlinear mixed integer programming model is formulated. Constraint sets, decision variables and objective functions are presented.

Chapter 4 presents the solution framework based on adaptive Tabu Search algorithm. The adaptive Tabu Search algorithm is composed of neighborhood definition, initial solution construction and evaluation and updating procedures. And results based on this methodology on two sample cases are also shown in this chapter.

Chapter 5 shows the numerical testing results based on Monte Carlo technique to find the threshold across which more destination data given by additional passengers would make no significant improvement to the optimal solution obtained by the meta-heuristic method. A methodology is also proposed for this purpose.

Chapter 6 presents the results on three real size cases abstracted from the Martin Luther King (MLK) station of the new MetroRail system in Austin, Texas. Solution performance and computational time are all discussed.

Chapter 7 concludes with summaries of the developed algorithm and research results. Suggestions for future research are also provided.

# CHAPTER 2. LITERATURE REVIEW

Generally speaking, technical literature on the bus transit route network design problem involves finding a bus transit route network configuration and other associated operational decisions that achieve a desired objective with a variety of constraints. Depending on the problem characteristic and modeler's perspective, objective and decision variables can be defined in various ways.

## MODEL OBJECTIVES AND DECISION VARIABLES

Kuah and Perl (1988) considered both users' travel cost and transit agency's operation cost while formulating a feeder bus network design problem to design a feeder bus to access rail system. It assumes an M-to-1 demand pattern where transit users would start from different bus stops, transfer at a rail station and then get to a final common rail station that is located in a downtown area. Since the demand starting from each stop is fixed, total access time to the bus stop is constant and cannot be affected by the network configuration, user travel cost in the model formulation includes user waiting cost at both bus stops and rail stations and user riding costs along the bus route and rail line; while the transit agency's operation cost is composed of capital cost and variable cost that is related to vehicle running time. Considering both user travel cost and transit operator cost also turns out to be a widely used way to achieve the optimal balance between demand and supply in the transit network design problem.

Dubois, Bel, and Llibre (1979) believe that to simply minimize the total travel time is appropriate while modifying a transportation network to better serve its existing demand. Instead of including operation cost in the objective function, they actually consider a particular budget as a cost constraint for the existing route modification. Ceder and Israeli (1998) provide a methodology with no preference on the specification of network structure, and account for the operators' benefit. This approach is of interest since it is different from most other works. The objective function in the model includes the empty space hours of vehicles to represent unproductivity from the transit operator side. Lee and Vuchic (2005) provide a detailed discussion of the composition of their conceptual objective function. They mentioned that minimization of user travel cost is a proper objective for public transit agencies, however, for private transit agencies, profit maximization would be more appropriate. The combination of the two objectives, which represents the social benefit maximization or the social cost minimization, tends to be favored by transit planners. The authors believe that if the operators' constraint is satisfied, user travel time minimization is desirable in many public ownership situations. Therefore, user travel time minimization is used as the optimization criterion for simplicity.

Ceder and Wilson (1986) actually provide a description for the bus network design problem and discuss the whole bus planning process as a sequential and systematic decision process which consists of five levels of decisions: network route design, frequency setting, timetable development, followed by bus scheduling and driver scheduling. It is mentioned that early efforts were devoted to bus and driver scheduling since the largest single cost to transit agencies of providing service is the driver wages and benefits, and optimization over these two scheduling problems would seem to be the best way to reduce this cost. Considering the entire planning process, the design of bus routes and defining operating frequencies generally receives the largest amount of attention. The author also mentioned: "If one could consider the full problem domain including alternative bus networks, it is more likely that sub-optimality in the final solution will be introduced by non-systematic rejection (through non-consideration) of feasible networks than through sub-optimality at the stages of bus and driver scheduling, which have already been extensively researched." Later research work is more dedicated to transit route set and frequency definitions. Pattnaik, Mohan and Tom (1998) focus on a problem of urban bus route network design to determine a transit route set and its associated operating frequencies. Route configuration and operation frequency on each route are the two primary sets of decisions that entail the largest amount of work on typical transit design problems. Chakroborty (2003) decomposes the urban transit network design problem into two sub-problems: the Transit Routing Problem (TrRB) and the Transit Scheduling Problem (TrSP). As the TrRB aims to select a set of routes, the TrSP substitutes a frequency optimization procedure for the process of thorough schedule analysis based upon bus arrivals and departures at all stops of the network. Solutions obtained based on this model formulation can actually provide more user-friendly transit information – a timetable for buses on each route. And also this formulation can be modified to incorporate the transfer concept without significant effort.

## ANALYTICAL MODELS

Classical analytical optimization models were used in the early stages of the research on the transit route network design problem. These models focus on developing a continuous convex objective function under assumptions that simplify and idealize the transit network. By solving first-order equations of the objective functions in these models, optimal solutions for stop spacing, headway, frequency, or other route characteristics can be efficiently solved. As noted by Ceder and Wilson (1986), analytic methods are suited to early stage screening in the planning process or conceptual policy decisions where approximate design parameters are adequate but have little practical benefit in solving real world problems. Examples of this traditional operations research analytical optimization model can be seen in the work of Newell (1979), Kuah and Perl (1988) and Chien and Schonfeld (1998).

Newell (1979) discussed issues relating to the optimal design of bus routes. One oft-cited viewpoint from this work is the characteristic that distinguishes the bus transit assignment process from that of the automobile assignment process, which is the higher the demand for trips on a route, the better the level of service that can be provided due to higher frequencies and less wait and transfer time. Newell mainly illustrated the sensitivity of any optimal geometry to the nature of the trip distribution and showed that a square grid of straight-line bus routes does not provide an optimal geometry even under highly idealized conditions where trips are uniformly distributed. And the author stated that potential good geometries should assign routes onto a single main street and past a common junction.

Kuah and Perl (1988) specially address the problem of designing an optimal feeder bus network to access a rail line with a differentiation-based method. The analytical model solves for the three basic variables – the route spacing, operating headway and stop spacing. For the first two decision variables, results showed that they are not highly sensitive to changes in the relevant system parameters. With regard to the stop spacing, three cases are considered: uniform spacing; constant spacing along routes; and variable spacing. In the first two cases, the optimal stop spacing is shown to be related to some system parameter, increase with walking speed, value of riding time and average time lost at a bus stop, and decrease with the value of walk time. In the third case, the optimal stop spacing depends on initial conditions, and stop spacing is shown to increase as the distance from the rail line decreases. Besides many of the same restrictive assumptions about the network layout to make the problem solvable with analytical methods, the authors indicate that the assumption of fixed demand for the feeder bus system is applicable when the demand is affected primarily by the level of service and the resulting ridership of the rail service.

Chien and Schonfeld (1998) develop a model to jointly optimize both a rail transit line and a dedicated feeder bus system to serve the rail line. The optimization method solves for rail line length, rail station spacing, bus headway, bus stop spacing, and bus route spacing. With the corridor demand characteristics specified with irregular discrete distributions to realistically represent geographic variations, the total cost of the bus and rail network is minimized with their proposed iterative method – a combination of basic calculus and a successive substitution method. The basic idea of the classical optimization algorithm is to derive, for each decision variable, the gradient vector by setting the first derivative of the objective function with respect to each decision variable equal to zero. However, their method allows changes in all decision variables within one iteration by successive substitution of decision variable values to efficiently solve for all the variables. The numerical results show that the most significant factor in determining the rail line length is the demand. The total cost function is relatively flat near the optimum, and its practical application is that minor changes in the optimal solution would allow transit suppliers flexibility in fitting the route structure to local circumstances without significant deterioration of total cost.

## HEURISTIC MODELS

Chakroborty (2003) has provided a detailed discussion regarding why the urban transit network design problem (UTNDP) cannot be solved with exact algorithms such as Branch-and-Bound, or Branch-and-Cut. First, inclusion of discrete decision variables in the UTNDP increases computational complexity since existing methodologies, for instance, the branch and bound technique, deal with integer variables iteratively by generating artificial constraints. Secondly, the nonlinearity of the UTNDP also makes it hard to solve. Traditional methods would solve this kind of problem through successive linearization, and this again would add a series of additional variables and involve significant computational effort. Thirdly, as also mentioned by Baaj and Mahmassani (1990) defining logical conditions to better describe a realistic transit network in the mathematical program will again introduce more integer variables and further computational complexity. All these listed facts lead to a common problem – a significant computational burden.

Given the limitation of exact algorithms in solving realistic transit network design problems, approximation techniques – heuristics and meta-heuristics are usually preferred in many practical situations. They enable one to solve the real-size network design problem in a reasonable time frame, compared to exact algorithms. However, the problems related to approximation methods are that the solution obtained is not guaranteed to be the global optimal, that it is difficult to state bounds on solution quality, and that the solution quality would be different every time the algorithm runs. Heuristic algorithms are usually guided by modelers' experience without specific rules to follow. The problems are solved by heuristic methods simply according to common sense. Meta-heuristics, as a subset of heuristics, are general algorithm frameworks, and they try to mimic biological, physical or natural phenomena in the real world to intelligently perform local searches. As they work well on combinatorial problems in which an optimal solution is sought over a discrete solution space, they are also employed in transit network design problems.

**Heuristics**

For solving a realistic size transit network design problem, heuristic methods rather than traditional exact optimization methods are usually used. Motivated by the practical problem of scheduling dial-a-ride transportation systems, Stein (1978) presents a heuristic method to solve a many-to-many route design problem. For the single-bus problem, the algorithm first partitions the service area into many smaller sub-regions, finding optimal tours for these regions and then connecting these tours to other regions. And for the multiple-bus problem, the algorithm includes predefined transfer points and allows buses to meet at these points for transfer until all origin-destination pairs can be covered by the resulting route sets.

Dubois, Bel, and Llibre (1979) deal with the problem of modifying a transportation network to better serve its existing demand. They decompose the problem into three sub-problems: choosing an optimal subset of streets (the optimal network problem), selecting bus lines, and then defining optimal service frequencies for the bus lines. The candidate street set and bus line set are both developed with greedy heuristic procedures. Three greedy heuristics are proposed for finding the street links, two of them finding the set by removing links step by step to reduce the total travel time and the other by adding links to minimize the travel time. The bus lines are chosen first by adding lines to connect the whole network, then searching for the main connection nodes and adding lines to decrease the number of these nodes to eliminate indirect trips, and finally joining lines or suppressing unused bus segments. The optimal service frequencies for the bus lines are determined through an evaluation and optimization process. Starting from a given set of bus lines and frequencies, performance of the bus network is evaluated and various characteristics become index components, and based on the index value, new frequencies are sought through a gradient-search routine in the optimization procedure, then the new network will be evaluated and optimized again until the frequencies converge.

Kuah and Perl (1988) define the feeder bus network design problem to provide feeder bus service to access an existing rail system. The problem solution method aims to find the optimal route set and operating frequency for the feeder bus system. A mathematical programming model for an M-to-1 demand pattern is formulated and also generalized to the M-to-M pattern. A heuristic method is presented to first initialize a feasible set of routes and then using a local search algorithm, it seeks to obtain better solutions.

The initialization procedure generalizes the “sequential savings approach’ to consider frequency. For each bus stop, the initial procedure first calculates the direct route costs to all rail stations, and then finds the route link to the rail station with the minimum direct route cost, and both the route link and its associated minimum cost are recorded as the direct route cost for each bus stop. After that, the initial procedure starts to build the network by choosing from all bus stops the one that has the largest direct route cost. And this stop will be used as the starting stop of the first bus route and following stops will be placed between that stop and the train station where the direct route ends. Nodes are sequentially inserted based on the savings due to inclusion in the current emerging route. When predefined conditions occur, either constrains are violated or savings caused by the inclusion of an additional node is below a set value, building of this route will be terminated and a new route construction procedure will be initiated and follow the same insertion process. The initialization will be terminated when all origin-destination demand pairs are served by a feasible route.

After a feasible set of routes was constructed during initialization, an improvement procedure – a local search will be employed to correct the limitations of the initial algorithm. The single route exchange procedure optimizes the order of bus stops in a single route to reduce total route cost. And the displacement will position a bus node to a different route if that action can reduce the total route cost. Although the local search procedure was not guided by any systematic mechanism and was highly problem-specific, it was a successful early effort to provide an heuristic that would provide solutions that were superior to manually designed networks.

## Meta-heuristics

### *Genetic Algorithm*

The Genetic Algorithm, inspired by the process of biological evolution, natural selection and survival of the fittest in living organisms, works on a population of individuals representing solutions to a given problem. Each individual is represented by a string of bits called chromosomes or genes, and each chromosome reflects the solution attributes. A fitness value is assigned to each individual in order to evaluate its optimality analogous to organisms' adaptability to the environment. Individuals with higher fitness values have higher chances to be selected for reproduction. Their offspring would inherit their characteristic.   Hence, favorable characteristic would be able to spread throughout the population over generations and the most promising areas of the search space are explored. Finally, the population should converge to an optimal or near-optimal solution, which should be output as the search result.

Pattnaik, Mohan and Tom (1998) use a Genetic Algorithm to solve the urban transit route network design problem. In their characterization of the problem both the route configuration for a transit system and associated frequencies are determined to achieve the desired objectives. The problem is solved in two phases: First, a set of routes for every terminal node pair were generated and ranked based on performance (associated travel time) as the candidate route set competing for optimal routes. Second, the optimal set was selected using an application of the Genetic Algorithm (GA). With a coding-decoding scheme set up, the route index can be converted to strings of bits, which are manipulated by GA for population initialization and generation production. The GA was adopted by using the fixed string length coding scheme, and a new variable string length coding was also proposed. With the fixed string length coding scheme, the number of routes in a route set is assumed within a range. The number of routes in a feasible solution is fixed during the successive generation. And the evaluation is carried out by varying the size of the route set over a range to find the optimal set. The problem with this method is that the user does not know how many routes would eventually evolve in an optimal solution. The proposed variable string length coding scheme can solve this problem by allowing different numbers of routes to be included in the set to be evaluated in the same generation. It added the insertion and deletion operators to the reproduction process (usually composed of mutation and crossover operator) in the fixed string length scheme so that by producing successive generations the solution route set size and the set of routes can be found simultaneously. Tom and Mohan (2003) applied the variable string length coding scheme in the Genetic Algorithm to a medium-sized network with 75 nodes and 125 links and also proposed a coding scheme to incorporate frequencies into the string representation in GA, in which case, both route configuration and bus operating frequencies can be determined simultaneously.

### *Simulated Annealing*

Simulated Annealing simulates the physical phenomena of the annealing process for solids. The initial temperature and the rate at which the temperature reduces are called the annealing schedule. It is a hill-climbing algorithm with additional ability to escape from local optima in the search space. Using SA to solve problems, a feasible initial solution is constructed first. A neighborhood of this solution is identified based on predefined neighborhood search techniques, and the associated objective function value of this new solution is calculated. If the new solution is better than the current solution in terms of improving the objective function value, the new solution is accepted. If the new solution is not better than the current solution, the new solution is accepted with a certain probability $\exp(-\Delta/T)$, where $\Delta$ is a chosen unfavorable change amount in the objective function value from the old to the new solution and $T$ is the current temperature. The probability decreases exponentially with the badness of the move. The annealing temperature is first set to be high so that the probability of acceptance will also be high, and at the beginning of the search almost all new solutions are accepted. As the temperature gradually decreases, the probability of acceptance of low quality solutions will become very small.  In this algorithm framework, high temperatures allow a better exploration of the search space, while lower temperatures allow a fine-tuning of a good solution. And the whole procedure is less likely to get stuck in a local optimum since bad moves still have a chance of being accepted.

Fan and Machemehl (2006) use a Simulated Annealing algorithm to solve the transit route network design problem. The proposed solution framework is composed of three major steps: An initial candidate route set generation procedure; a network analysis procedure that assigns transit trips to each route and determines service frequencies; and a simulated annealing procedure to perform a search and select an optimal set of routes from the search space. The candidate routes set is generated using Dijkstra's shortest path algorithm and Yen's K-shortest path algorithm to find the shortest path between centroid node pairs. All the candidate routes are subject to the user defined minimum and maximum length constraints. They are indexed and stored in the solution space. At the beginning of SA implementation, the initial route set was randomly selected. With the network analysis procedure, transit trips were assigned to each route, operating frequency was determined in iterative steps and the performance measurement of this solution is also computed and stored. The annealing schedule consists of four components: (1) the initial value of temperature T; (2) a cooling function $\mathrm{T} = \mathrm{T} * \alpha$ $(0 < \alpha < 1)$; (3) the number of iterations performed at each temperature; and (4) the stop criteria to terminate the algorithm. The neighborhood defined in this problem is that for any route $\mathrm{i}$, replacing it with the route right next to it in the solution space produces a new neighborhood solution. Starting from the random initial solution, SA guides the search procedure and updates the current best solution until the number of iterations or termination criteria are satisfied. To measure the solution quality of the SA algorithm, the authors also use genetic algorithm as a benchmark. They test three experimental networks and state that the proposed SA algorithm outperforms the GA algorithm in most cases and that compared to GA, the SA is at least as good as GA as a candidate solution approach for the BTRNDP.

### *Tabu Search*

The Tabu Search technique proposed by Glover (1977) is also widely used for solving combinatorial optimization problems.  Its name is derived from the word 'Taboo' meaning forbidden or restricted.  Tabu Search, like simulated annealing, allows for exploring the search space smartly to escape local optima. The major distinguishing feature of Tabu Search is the use of a short-term memory called a tabu list, in which reverse moves of recently taken moves are controlled.  Moves in the list are considered as prohibited by the Search and cannot be visited again for a predefined number of iterations. The idea is to avoid the problem of cycling since the search many be trapped within a certain neighborhood region, oscillating among solutions that have been previously visited. By prohibiting recently visited moves, the algorithm is guided to explore new regions of the search space in an attempt to escape the trap of local optima.

Extended from the heuristics proposed by Kuah and Perl (1988), Martins and Pato (1998) first expand the initial solution construction procedure by adding a two-phase construction method to the original sequential savings approach. And mostly, they employed Tabu restrictions in the improvement procedure. For both the single route exchange procedure and the internal and external displacement procedures, Tabu restrictions prevent the replacement of a stop in its previous position for a chosen number of iterations. Also, the intensification strategy of the Tabu Search method accentuates the search in a region of good solutions by decreasing the tenure of moves marked tabu, diversification strategies are attempted by increasing the costs of the more frequently moved stops to take more consideration over them when initializing routes. This helps to obtain solutions that are different from the previously visited ones.

Fan and Machemehl (2008) applied the Tabu Search algorithm for the design of public transportation networks. The solution search framework includes: an initial candidate route set generation procedure; a network analysis procedure to assign transit trips and determine service frequencies; and a Tabu Search heuristic method to guide the local search process and select an optimal route set from the huge solution space. All feasible candidate routes connecting terminal node pairs are indexed and stored in the solution space. Starting from an initial feasible route set, the Tabu Search algorithm defines neighborhood solutions of the current solution as those obtained by replacing any route in the current solution by its adjacent routes stored in the solution space and outside the current set. Moves that have been taken are marked as Tabu for a user-defined or randomly generated number of iterations. The authors also include diversification and intensification strategies in the search process. The diversification strategy allows more routes to be replaced by remotely located routes so that the solution space can be traversed and explored more evenly. And to respect the nature of Tabu Search, the intensification strategy allows the diversification strategy to be used only once during a given operation. By conducting sensitivity analysis over three experimental examples, the results show that the Tabu Search with Shakeup and fixed tenure is the best TS algorithm application for solving the bus transit route network design problem.

Lownes and Machemehl (2010) specifically address the CRCNDP problem by formulating a mixed integer programming approach with the objective accounting for transit user travel cost, transit agency operation cost and social cost related to unserved demand. Both exact and heuristic methods are proposed to solve the problem. The exact method utilizes lower bound and additional stopping criterion to reduce computational effort, yet it is still suitable to small to medium-sized networks while the Tabu Search method solves large networks in a reasonable time domain. The real-time CRCNDP in this report is closely related to Lownes' work, but differentiates itself in two aspects: 1. As mentioned in the introduction chapter, this report proposes a scenario in which all passengers alighting the train have no other mode choice except the circulator provided, so that for any un-served destination node, a penalty for resulting long walk trips is added to the objective function rather than categorizing the demand as un-served; 2. Since this effort aims to provide optimal route configuration to each set of arriving train passengers, it requires that the optimal or near-optimal solution be obtained in a limited time domain. That is to say, high computational speed is a primary goal in developing the algorithm that is intended to provide virtually real-time route optimization solutions.

## SUMMARY

This chapter has summarized related literature of transit network design problems based upon their objective function, decision variables and solution framework. A combination of both user travel cost and operators' operation cost is a common practice in defining the objective function. It is also a proper objective in the real-time CRCNDP since other factors, such as capital investment and the impact on demand are both remotely related to and hard to control in the real-time operation. Regarding the decision variables, since a seamless transfer concept is incorporated, that is, all rail passengers use the bus service, only route configurations remain for optimization. Two major sets of methodology for solving the route design problem are identified and the meta-heuristics are chosen for real-time operation due to the inherent NP-hard nature of the transit network design problem, as well as the quick response requirements of the real-time CRCNDP.  The next chapter will provide a detailed explanation of the model formulation for the real-time CRCNDP.

# CHAPTER 3. MODEL FORMULATION

The real-time CRCNDP aims to provide an optimal route for each set of passengers arriving at a commuter rail station on a real time basis. Selection of bus stops and the route connecting them are determined simultaneously to design a current feeder system network. Instead of having a static O-D demand for commuters describing a typical day, a real-time O-D demand would be obtained from passengers on each and every train. In the best case, if all passenger destinations are known, every circulator bus could traverse a uniquely optimized route stopping only at uniquely selected optimally located locations. However, depending on the success of the method chosen to gather passenger destination information, less then 100 percent of the passenger destinations would be known. This reality introduces the question of what fraction of passenger destinations must be known to justify route and stop optimization based upon the partial destination information. Since partial knowledge of the destination set is likely to be the typical case, this question is not trivial. Therefore, it is experimentally addressed in later sections

The total commuter rail demand alighting at the station is assumed to be totally served by the circulator system. This assumption is applicable to those newly built and remotely located rail system stations where neither public transit modes nor walk can be a viable option for accessing the rail station. With a rail system in place, land use policy could allow Transit Oriented development around rail stations over a long time period. In that case, passengers alighting the rail stations might have other travel options (walk, bus, light rail and so on) to get to their destinations. But there is no negative effect on this model since real-time destination information can still be obtained by simply asking passengers how and where they want to go, and the model can still be used to optimize a feeder system for passengers choosing to board the circulator without loss of generality. Since the optimization process may not include every possible destination node in the optimal stop set, long walk penalties are applied to the objective function accordingly. The model will also incorporate the seamless transfer concept, which is a bus for each circulator route should be present at the rail station when a commuter train arrives at the station.

The CRCNDP can be formulated in various ways. Every modeler confronts the decision regarding tradeoffs between modeling precision and computational cost. For a real time optimization algorithm, a model formulation which enables obtaining an optimal or near-optimal solution within a very limited time constraint (in this problem, from the time passengers board the commuter rail train to the time they get to the station served by the circulator) becomes quite necessary. Following is the formulation of the CRCNDP in this report based upon the work of Lownes (2007).

## FORMULATION

The formulation of the CRCNDP uses an idealized, simplified network representation similar to that shown in Figure 1. A rail station shown as the Green node is associated with a number of demand centroids located at the center of each zone. Around each demand centroid, candidate bus stops are represented by the red nodes at the mid-block and intersection locations. There are various methods to select potential stop locations and inclusion of both mid-block and intersection stops makes the network representation more comprehensive, however, the more locations included in the network, the greater complexity of computational effort would arise in each evaluation process. Therefore, decisions have to be made upon stop location selections when dealing with realistic problems. Different from most of the previous work, the route configuration in this report is not based upon zonal demands and it actually incorporates the exact location of bus stops. Then the optimal route configuration obtained based on this network representation would be used to identify a path that connects the actual bus stops rather than "schematically" connecting the demand zones and leaving actual stop location choices to the judgment of the analyst.    This enables the algorithm developed in this report to provide the level of information that transit operators really want in practice. The algorithm will permit the user to specify potential stops at mid-block and intersection locations. However, as the number of potential stops increases the speed with which the algorithm can find the solution decreases. In other words, the user is not limited to a certain number or a characterization of stops that is built into the algorithm. If a user requires quicker response times, this could be obtained most easily by reducing the number of candidate stops.

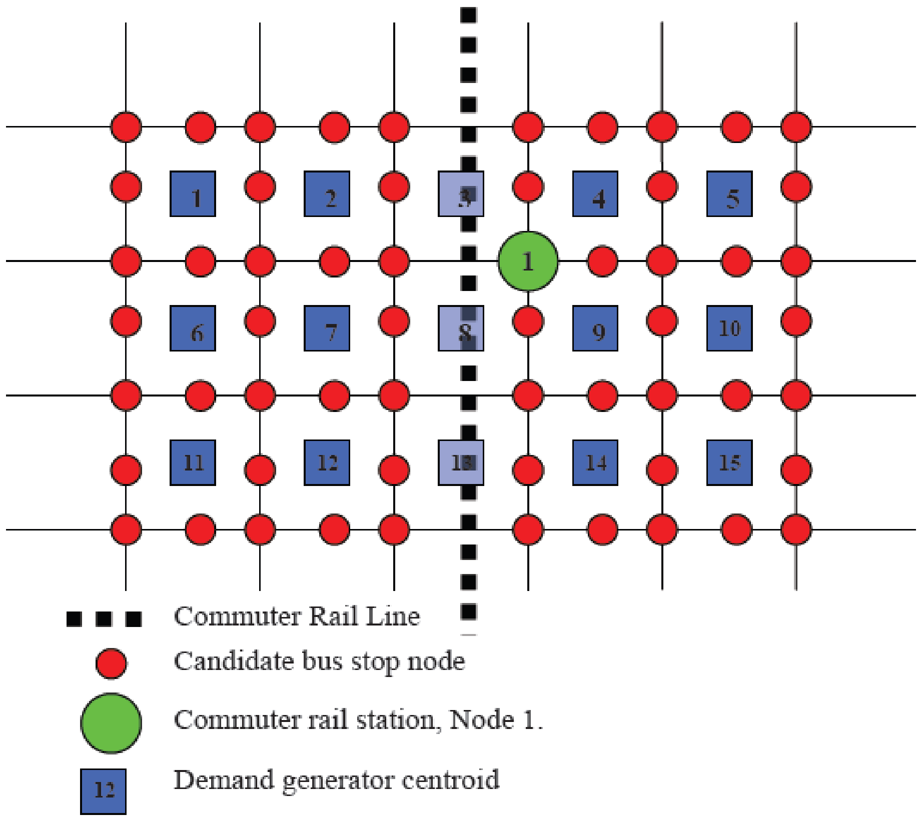

**Figure 1.    Network Representation (Lownes (2007)).**

Before a detailed description of the formulation components is given, to avoid ambiguity, several assumptions are described.

## ASSUMPTIONS

1. There is no traffic congestion and no incidents along any circulator route and circulator buses operate at constant speed, so that travel time is proportional to the distance between nodes.
2. A non-zero fraction of passengers would provide their destination information and this information will be used as real-time demand.
3. Real-time demand is located at predefined zone activity centroids.
4. Real-time demand is only a reflection of the final destinations of the set of passengers on the train.
5. Each circulator route starts and ends at the commuter rail station (Node 1).
6. The number of passengers boarding each circulator will not exceed its capacity.

The first assumption simplifies the problem by accounting for travel time with its expected value, although stochastic travel time in a transportation network is common and unavoidable due to many factors. This assumption can be relaxed in future work to better describe the real and practical problem and ensure the robustness of the CRCNDP model.

The next three assumptions address the demand used to drive the formulation. Although it is aggressive to assert that all passengers would provide transit operators their final destinations, we start with a 100% sample or complete destination data to test the algorithm and seek in later chapters the potential range of destination sample size that guarantees the value of "real-time" optimization. Depending on feasibility and availability, a variety of destination formats can be assumed applicable in the model formulation. As a starting point, this formulation assumes that real-time demand data will only show the general attractiveness of the destination zone, also called aggregated demand. Conventionally, the layout of the transportation network will affect the demand estimates since it predefines the accessibility of all existing modes in the network. However, in this report, the model is formulated to optimize the circulator route design especially for a certain set of real-time passengers, so that it is reasonable that the long-term interaction between service supply and traffic demand is left without consideration.

The fifth assumption guarantees the seamless transfer concept and it also makes the CRCNDP problem conform to the structure of a Traveling Salesman Problem (TSP) problem. Lastly, since it is a newly-introduced commuter rail, it takes time for ridership of the commuter rail to build up. And during its early stages, it is reasonable to assume that circulators serving the commuter rail at each rail station are not capacity constrained. Even when the rail system becomes mature and a large number of passengers alight at the rail station, the non-constrained capacity concept can still be applicable if multiple circulators are employed along the same route obtained by the algorithm developed in this work.

The formulation components of the CRCNDP problem are sets/indices, parameters and data, decision variables, the objective function and a constraint set.

## SETS/INDICES

In this network, the sets' representation is simple and clear. Locations of the demand centroids are identified as the set G. Since this demand labeling system makes no implications regarding the geographic size of the members of the set, many different levels of aggregation can be used to describe demand and these can be incorporated into this definition. Candidate bus stops surrounding the demand centroids are represented as the set $I$. A demand centroid can be served by any of its surrounding bus stops depending on which are included in the route. And passengers can also alight the feeder bus at one stop even though they are destined for non-adjacent centroids. Among all potential bus stop locations in set $I$, the algorithm will select a subset of locations as $r$ to form a circulator route.

| | |
|---|---|
| $g \in G$ | Circulator demand centroid locations |
| $i, j, k \in I$ | Candidate circulator stop locations |
| $r \subseteq I$ | Subset of demand locations/nodes to visit in circulator route |

Depending upon the capability of the communication application between individual users and the circulator operator, street address or block level demand, or demand for Traffic Analysis Zones (TAZ's) could be estimated. Block level demand aggregation seems to be a reasonable compromise between dis-aggregate street address data and potentially aggregated TAZ data, therefore, block level demand aggregation has been chosen as the basis for the algorithm development. Since all train passengers are assumed to board the circulator, but the optimal stop set likely will not include all possible destination stops, some users will have long walks at the end of their trips. The algorithm will include a "cost" associated with long walks that is directly proportional to the walk lengths and numbers of passengers taking the long walks.

## PARAMETERS AND DATA

The first three parameters are the values of the cost components used in the multi-objective formulation of the CRCNDP. The default values of these parameters rely heavily upon the Transit Cooperative Research Program (TCRP) Report 78 (2002), which gives practical cost parameter values.

| | |
|---|---|
| $C_o$ | Transit bus operation cost (\$/hr) |
| $C_{ivtt}$ | Equivalent cost of traveler's in-vehicle travel time (\$/hr) |
| $C_{ovtt}$ | Equivalent cost of traveler's out-of-vehicle travel time (\$/hr) |

The real-time demand data is considered as given since it is obtained and summarized before the CRCNDP algorithm is executed. Rectilinear distances for node to node pairs are used to determine the route design, which stops to be visited and in what order, and later to calculate the bus operating cost and traveler in-vehicle travel cost. Distance from node to centroid will be used to calculate traveler's out-of-vehicle travel costs.

| | |
|---|---|
| $d_g$ | Real-time demand for service at demand centroid $g$ |
| $\lambda_{ij}$ | Rectilinear distance from node $i$ to node $j$ |
| $\gamma_{ig}$ | Rectilinear distance from node $i$ to demand centroid $g$ |

The commuter rail headway will pose a strict constraint to the CRCNDP problem as the seamless transfer concept is implemented. That is, the circulator will be designed to provide enough passenger spaces to accommodate all alighting passengers at the rail terminal for every train arrival. Typical values for headway (H) range from 15 – 30 minutes. The following two values for bus speed and walking speed are derived from Levinson (1983) for bus operating speeds and the TCRP Report 78 (2002) for walking speed. The default values for these parameters are 10 mph and 2.5 mph, respectively. The bus operating speed is estimated for city routes (as opposed to CBD or suburban routes) and the walking speed is an average value that is typically assumed in the transit planning process.

| | |
|---|---|
| $H$ | Commuter rail headway is the time interval between arriving trains |
| $v_{bus}$ | Circulator bus operating speed |
| $v_{walk}$ | Pedestrian walking speed |

The final three parameters are not given but can be easily calculated. $\rho_{ig}$ is a parameter that signifies which stop is going to serve which demand centroid. For each demand centroid, one stop will be determined to provide the shortest walking path. That is to say, each demand centroid is served by the stop in route $r$ providing minimum walk distance. The sum of the demands for the centroids served by stop $i$ produces the total demand served by stop $i$ ($s_i$). Dwell time is estimated by simply assuming a linear relationship between dwell time and the number of deboarding passengers developed by Levinson (1983). The expression for dwell time in this report is $\delta_i = 4.0 + 1.7s_i$.

| | |
|---|---|
| $\rho_{ig}$ | Binary variable indicating whether demand centroid $g$ is served by stop $i$ in route $r$ |
| $s_i$ | Total demand that is served by node $i$ in route $r$ |
| $\delta_i$ | Dwell time at node $i$ |

## **DECISION VARIABLES**

The decision variables used in this formulation are straightforward. Similar to decision variables in traditional TSPs, the binary variable $x_{ij}$ signifies if a trip from $i$ to $j$ is made during the tour. The variable $w_{ij}$ records the number of passengers that travel from $i$ to $j$ for a tour of the subset r. This variable is important to maintain so that accurate in-vehicle travel costs and bus operation costs can be computed for each segment of the route and flow conservation is maintained at all nodes within the subset r.

| | |
|---|---|
| $x_{ij}$ | Binary variable indicating whether the bus travels from $i$ to $j$ on route r |
| $w_{ij}$ | Link pass variable indicating the number of passengers traveling from $i$ to $j$ on route r |
| $u_i$ | Arbitrary real number at node $i$ used for subtour elimination |

The formulation of the CRCNDP utilizes the subtour elimination strategy developed by Miller, Tucker and Zemlin (1960) and introduces the unrestricted decision variable $u_i$. Although this subtour elimination constraint does not provide as intuitive a method of subtour elimination as other common methods seen today that identify and eliminate disconnected subtours, it solves problems well with the familiar TSP constraints that are discussed below in terms of high speed elimination.

## OBJECTIVE FUNCTION

The objective function based on the model developed by Lownes (2007) contains four cost components:

$$minimize\ z = \sum_i \sum_g C_{ovtt} \frac{\gamma_{ig}}{v_{walk}} \rho_{ig} d_g + \sum_i \sum_j C_{ivtt} \frac{\lambda_{ij}}{v_{bus}} w_{ij} + \sum_i \sum_j C_o \frac{\lambda_{ij}}{v_{bus}} x_{ij} + \sum_i \left( C_o \delta_i + C_{ivtt} \sum_j w_{ij} \delta_i \right) \qquad (1)$$

These four components can be basically categorized into two competing parts: operation cost from the transit operator perspective and travel cost from the transit user side. The first two components are the two major cost portions to transit users – out-of-vehicle travel cost and in-vehicle travel cost. Out-of-vehicle travel cost is computed by identifying the demand served by the walk trips and the lengths of these trips. And in-vehicle travel cost applies the travel cost for trip segment to each passenger onboard the vehicle during that trip segment. The third component identifies the operation cost to transit operators due to operating buses along the route. While passengers alight the circulator bus at each stop along the route, the dwell time will cause both in-vehicle travel cost to transit users that remain on the vehicle and operation cost to transit operator. And these are described by the fourth component.

The model developed by Lownes (2007) accounts for those unserved nodes by a cost component defined as unserved demand cost. The unserved demand cost is applied to the total cost if there are demand centroids left out of the optimized route. For long-term planning based on static travel data, once the route is optimized, it is going to be in place for a long time period. So that it is reasonable to include the unserved demand cost component to partially account for the interaction between the route design and the ridership of the feeder. There is a tradeoff between served demand cost and unserved demand cost when deciding how many stops to be included. With less stops included in the route, transit agencies' operation cost and transit users' travel cost can both be controlled, however, unserved demand cost will increase due to more people being left out of service, and vice versa. The problem defined in this report, is a different case, passengers are assumed to have no other access mode besides the feeder buses and they are providing their destination information on a real-time basis. The route is optimized and selected based on the destinations of passengers on each and every train. The route selected for this set of passengers would not affect the choice of the other set of passengers. In this real-time optimization problem, the penalty for impropriation of route design is actually accounted by inclusion of long-walk trips, which is also part of the out-of-vehicle travel cost.

## Constraint set

The first two constraints, equations (2) and (3) represent the property of a typical TSP problem, which is every node should be visited exactly once. The optimal solution has to satisfy the requirement that only one incoming trip and one outgoing trip is associated with each node in the route. Constraint (4) sets an upper limit of linkpass (the volume of passengers traveling along the link) for each selected trip segment.

$$\sum_{j:j\neq i} x_{ij} = 1 \qquad \forall i \in r \qquad (2)$$

$$\sum_{i:j\neq i} x_{ij} = 1 \qquad \forall i \in r \qquad (3)$$

$$w_{ij} \leq \left(\sum_{g} d_g\right) x_{ij} \qquad \forall i,j \in r \qquad (4)$$

Constraints (5), (6) and (7) further describe the property of linkpass in this particular CRCNDP problem in which (5) is a typical network flow conservation constraint. The number of passengers alighting at node $\boldsymbol{i}$ should be equal to the number of passengers arriving at this node on the bus less the number of passengers remaining on the bus as the vehicle leaves the node. Constraints (6) and (7) set the initial and final trip segment conditions for the route. Since demand served at node 1 (the rail station) is zero, the first segment contains all passengers that will alight at all stops in the route. And there will be no passengers coming back to node 1.

$$s_i = \sum_{k\in r} w_{ki} - \sum_{j\in r} w_{ij} \qquad \forall i \in r \qquad (5)$$

$$\sum_{j} w_{1j} = \left(\sum_{i} s_i\right) - s_1 \qquad (6)$$

$$\sum_{k} w_{k1} = 0 \qquad (7)$$

Constraint (8) incorporates the seamless transfer concept. The circulator bus has to come back to the rail station before the next train arrives.

$$\sum_{i}\sum_{j} \frac{\lambda_{ij}}{v_{bus}} x_{ij} + \sum_{i} \delta_i \leq H \qquad (8)$$

Since this formulation is dealing with a newly introduced commuter rail system, the ridership and operation frequency would be reasonably low so that one circulator bus is sufficient to serve the demand. For a mature commuter rail system (approximately full trains), this CRCNDP formulation constraint should be slightly modified. That is, more passenger spaces could be provided by using larger vehicles or smaller vehicles could be scheduled in tandem.

Constraint (9) presents the Miller, Tucker and Zemlin (1960) subtour elimination constraint. There are different ways to eliminate subtours in the GAMs library and this technique is selected based on its computational efficiency and overall performance.

$$u_i - u_j + |r|x_{ij} \leq |r| - 1 \quad 1 \leq i \neq j \leq n \qquad (9)$$

The final three constraints, just as they imply, restrict $x_{ij}$ to be binary, $w_{ij}$ to be a positive integer and $u_i$ to be unrestricted.

$$x_{ij} \text{ binary variable} \qquad \forall i, j \in I \qquad (10)$$
$$w_{ij} \geq 0 \text{ and integer variable} \qquad \forall i, j \in I \qquad (11)$$
$$u_i \text{ unrestricted } variable \qquad \forall i \in r \qquad (12)$$

## SUMMARY

A mathematical nonlinear mixed integer programming model is formulated in this chapter. A seamless transfer concept is incorporated, both users' and operators' costs are included in the objective function and out of vehicle travel cost are also included in the objective function to penalize long walk trips due to the exclusion of some stop locations on the route. The formulation has both features of a combinatorial optimization problem and the Traveling Sales Problem, so that it is non-deterministic polynomial-time hard. To realize real-time operation based on this formulation, a heuristic algorithm is developed in Chapter 4 to solve this problem in terms of an optimal or nearly optimal solution.

# CHAPTER 4. META-HEURISTIC ALGORITHM

With the inherent complexity and the combinatorial NP-hard nature of the real-time CRCNDP problem, the complete enumeration method which searches over the whole solution space for the global optimal solution by simply enumerating and comparing the objective value for all possible solutions requires tremendous time even for a small size network. Note the objective of this report is development of a real-time practice aimed to identify an optimal route for passengers on the rail train during the time between train boarding and alighting. It is necessary to develop an algorithm which solves the real-time CRCNDP with a good solution in a limited time domain.

The Tabu Search, due to its power and efficiency, has traditionally been used on combinatorial optimization problems and has been widely applied to many integer programming problems, routing and scheduling, traveling salesman and related problems. The real-time CRCNDP coincides with the property of the traveling salesman problem in its routing design. The basic concept of Tabu Search is presented by Glover (1977). The overall approach is to avoid cycling while searching for global optima by forbidding moves which take the solution, in the next iteration, to points in the solution space previously visited, hence the moves were marked as Tabu for a certain number of iterations (the number is usually fixed and called Tabu tenure). The Tabu Search starts with an initial solution and updates the best current solution at each iteration as the optimal solution after searching through the predefined neighborhood space. Again, recent moves are marked in one or more Tabu lists to avoid reversing the steps that have been made during the search process. There is no guarantee that every move leads the search to a better solution; actually, the fact that moves taken by a Tabu Search lead to deterioration of the objective function is part of Tabu's diversification mechanism which enables the algorithm to search beyond local optima.

The static Tabu Search defines the Tabu tenure of each move during the search process as a fixed number regardless of the performance of the incumbent solution reached by these moves. Improvement can be made when slightly modifying the Tabu tenure as adaptive to the performance of the incumbent solution. The rules of adaptive Tabu are intuitive: if the incumbent solution obtained after a certain move is superior to the best current solution ever stored, the Tabu tenure associated with its reverse move is extended to prevent the search away from solution spaces related to this move; similarly, for moves leading to deteriorative solutions, the algorithm marks its reverse move as Tabu and decreases its tenure for later searches to quickly step back from solution space with worse performance. There are a variety of ways to execute Tabu Search, and the performance of these various methods is really a problem-specific issue. Adaptive Tabu Search is applied to a small size network to test its practicality.

It is obvious that there is a tradeoff between improved computational speed and quality of the optimal solution. In this chapter, the quality of the optimal solution obtained by adaptive Tabu Search will also be evaluated.

## ADAPTIVE TABU SEARCH FOR THE REAL-TIME CRCNDP

The solution framework based upon adaptive Tabu Search is illustrated in Figure 2. The algorithm first defines the number of stops to be included in the route and then searches in the neighborhood where alternative solutions contain the same number of stops; while solutions are still feasible, the algorithm increases the number of stops by one and performs the search process until it reaches a region where most solutions are infeasible. The decision on the number of stops to be visited by a route is controlled outside the local search – Tabu Search procedure. Generally, with more stops included in the route, time spent traveling along the route, as well as, time spent dropping off passengers at these stops increases but passenger walking times decrease.  Most of the feasible solutions are typically found in the region where a smaller number of stops is selected to construct the route. For this single route design problem, the strategy takes advantage of this characteristic of the problem while keeping the search procedure well controlled.

Starting from a random selection of a single bus stop location, together with the rail station, two stops are firstly included in the route to construct an initial solution, then local search is performed over predefined neighborhood solutions, and guided by the adaptive Tabu Search algorithm, solutions are evaluated and updated accordingly. As noted previously, while solutions are still feasible, the number of stops is increased by one and the initialization and search procedure are carried out again.

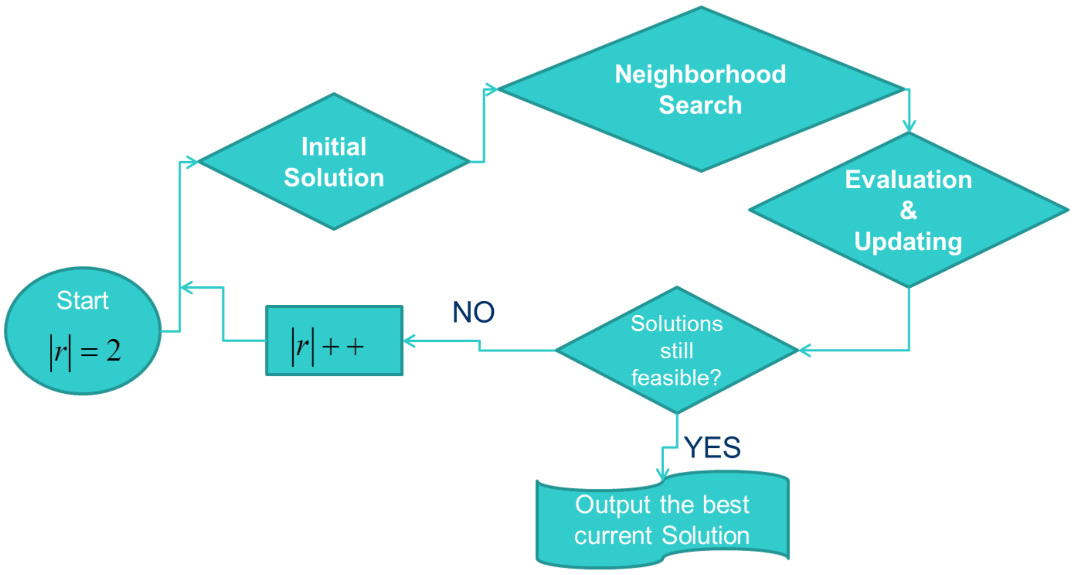

**Figure 2. Adaptive Tabu Search Solution Framework.**

**Initial solution**

The effectiveness of Tabu search, similar to many other Meta-heuristics, is highly dependent on the initialization success – obtaining a good starting solution. Since our model formulation assigns higher unit cost to out-of-vehicle travel time than to in-vehicle travel time, the first intuition for constructing a good starting solution is serving dense demand centroids primarily to shift portions of long walk trips to bus trips and further reduce total travel cost. The original procedures for initial solution construction are as follows:

1. Define the number of nodes $|\mathrm{r}|$ including node 1 to be covered by the route, note that node 1 is for the rail station and is always kept in the route.
2. Randomly select the $|\mathrm{r}| - 1$ nodes (demand centroid or stops) associated with the highest demand and together with the rail station (node 1) form a set $\mathrm{r}$.
3. Call GAMS to solve CRCNDP with the set $\mathrm{r}$, if it is a feasible route to the CRCNDP, keep it as the initial solution; if not, randomly replace members of the set until a feasible set $\mathrm{r}$ is encountered and record it as the initial solution.

While this works well for ideally designed networks (Lownes, 2010), however, difficulties arise when the concept is applied to more realistic case studies. If the stop nodes associated with the highest demand centroids do not generate a stop set comprising a feasible route, the initialization procedure will be highly dependent on a random process to select the stops to be included in the route. When it comes to a solution space where feasible solutions are sparse, a long time is required to finally construct a feasible initial solution since these selections are completely random and each time there is a low probability of finding a feasible solution especially when there is no guidance from any historical memory mechanism.

After experimentation was used to determine why this initialization proposal often fails, another procedure was developed. Instead of seeking an initial solution with potentially lower total cost, this initialization procedure conservatively prefers a solution with a greater likelihood of meeting the feasibility constraints. This procedure selects clustered stops with a priority of producing a feasible route. By having stops close to each other, the vehicle travel time which is a major component of the total time spent on a route, is tightly controlled, and it avoids violation of the time constraints and maintains the solution feasibility. The procedure is shown as follows:

1. Define the number of nodes $|\mathrm{r}|$ including node 1 to be covered by the route, note that node 1 is for the rail station and is always kept in the route.
2. Use the $|\mathrm{r}| - 2$ nodes (demand centroid or stops) from the best current solution and select an additional one from the rest of the stops with the least total distance to all the $|\mathrm{r}| - 2$ nodes.
3. Call GAMS to solve CRCNDP with the set $\mathrm{r}$, if it is a feasible route to the CRCNDP, keep it as the initial solution; if not, randomly replace members of the set until a feasible set $\mathrm{r}$ is found and record it as the initial solution.

The initialization procedure is shown in Figure 3. The differences between the two procedures are as follows: the previous procedure prefers to add a node serving high demand and the selected one prefers to add a node that is close to the current set of stops. Inclusion of a node serving high demand can potentially reduce long walk travel costs; however, if the additional node is itself far from the selected set of stops, the solution created by including this node would violate the time constraint and thus cannot be used as a starting solution. When this occurs, the algorithm switches to the other procedure that selects the stop clustered with the current stops in the set. In this way, feasibility is maintained first, and optimality is left for later neighborhood searches and process updates. The selected procedure for initialization is actually more efficient time-wise and is proven to work well for both small cases and large cases.

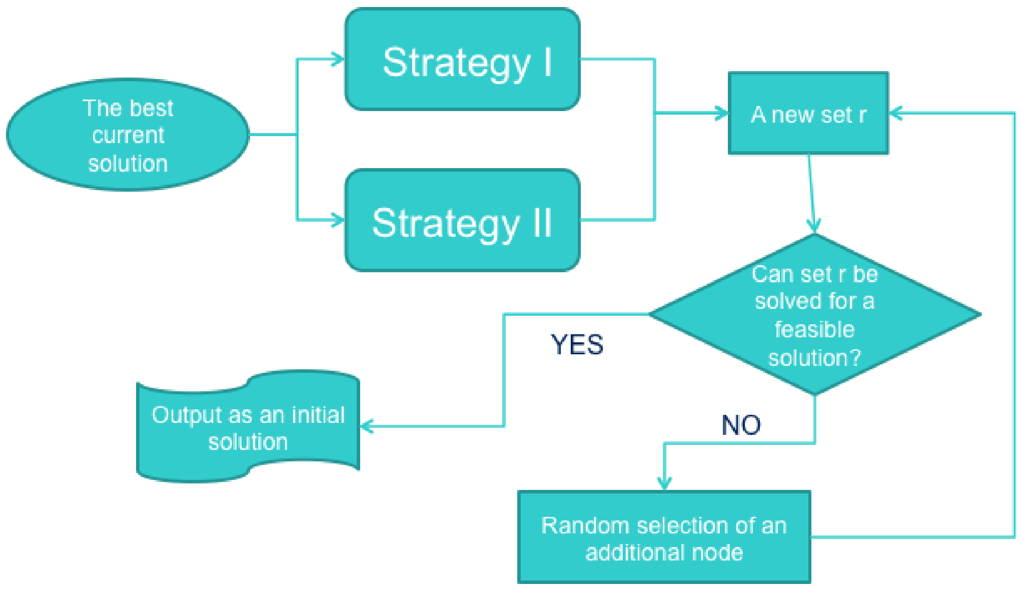
The best current solution
Strategy I
Strategy II
A new set r
Can set r be solved for a feasible solution?
YES
NO
Output as an initial solution
Random selection of an additional node

**Figure 3. Initial Solution Construction.**

## Neighborhood

The neighborhood definition in Tabu Search can also be developed in a variety of ways. For this particular CRCNDP problem, a simple and easy to implement way is to randomly select the leaving node within set r and replace it with a random node outside set r. However, our goal is to make the best use of the limited iterations to which the Tabu search is restricted; the candidate stop set should be smartly selected for CRCNDP evaluation to guarantee the optimality within a reasonable time domain. The employed steps are as follows:

1. Pair up every node in set r with every node outside set r and each pair can be regarded as a possible move for the incumbent solution to get to its neighborhood solution.
2. Select the move based on two guiding strategies to form a new set as a neighborhood solution. Both of the following techniques are being used. For any current solution, the algorithm would first perform technique (a) to construct a neighborhood solution. Technique (b) will be employed either when the neighborhood solution is infeasible or the current solution is being visited again.

a) Select the non-Tabu move with highest value of attributes defined as $p_{ij}$,

$$p_{ij} = \frac{d_j}{\sum_{k\in r, k\neq i} \lambda_{kj}}$$

b) Randomly select from the available moves.

When identifying the node to be switched into the neighborhood, it is preferred that the replacing node is associated with high demand and its inclusion into the route does not incur long travel distance which may induce more travel time cost and operation cost. The move attribute $p_{ij}$, calculated as the ratio of demand served by the replacing node to the total distance between this node to all other nodes excluding the replaced node in the route, describes this preference mathematically.

Random selection is also employed for the following two reasons: Cycling would occur when the same solution was revisited, if all steps are predetermined by known parameters without involving any randomness. The random process does not guide the search directly to the global optima. However, it helps to traverse the solution space and the combination of more move strategies provides opportunities to preserve the aggressiveness of Tabu Search by diversifying the search to new regions.

**Evaluation and updating**

This is the final step taken to ensure that the search traverses infeasible neighborhoods and escapes from local optima to possibly reach the global optimum, and the idea of this procedure is shown in Figure 4. GAMS is called to solve CRCNDP with each neighborhood set $r$ and the related objective value is recorded for the solution evaluation and updating.

1. If a neighborhood solution (incumbent solution) reached by a move performs better than the current best solution, update the current best solution with the incumbent and mark the reverse move as Tabu with a tenure as X+1. (X is a predefined parameter, usually 3);
2. If a move yields an infeasible neighborhood solution, mark the reverse move as Tabu with a tenure as X and make no update to the current best solution;
3. If a move yields an incumbent with no improvement, mark the reverse move as Tabu with a tenure as X-1 and make no update to the current best solution;
4. After a set number of iterations fails to update the current best solution, randomly select new neighborhood solutions until a feasible one is located.

Regarding each move or switching pair of nodes as an attribute of the current solution, we would be able to grade its reverse move according to the performance of the neighborhood solution to which each move has led. The reverse move, Tabu, is marked for a longer duration to keep desirable attributes of solutions available in later search processes. For example, if a move takes the search to a neighborhood solution that is better than the best solution, this attribute (the move or the switch) of the solution would want to be kept longer in later search processes. Conversely, if a move takes the search to worse solutions, the algorithm quickly steps back from the solution with poorer performance, hence it decreases the tenure of its reverse move. By editing the Tabu tenure according the performance of a neighborhood solution, the memory mechanism is maximized to better guide the solution search.

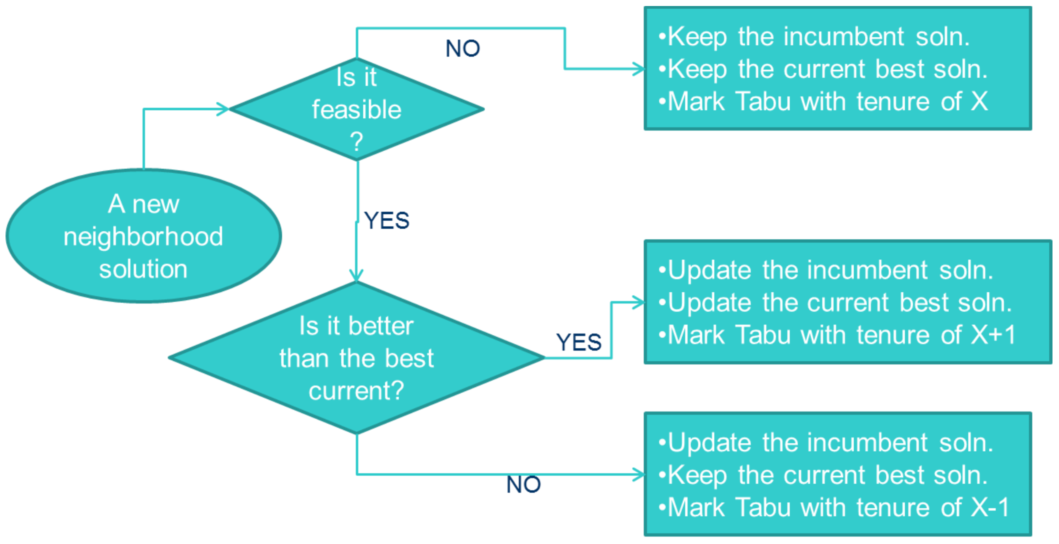
A new neighborhood solution
Is it feasible ?
NO
•Keep the incumbent soln. •Keep the current best soln. •Mark Tabu with tenure of X
YES
Is it better than the best current?
YES
•Update the incumbent soln. •Update the current best soln. •Mark Tabu with tenure of X+1
NO
•Update the incumbent soln. •Keep the current best soln. •Mark Tabu with tenure of X-1

Figure 4. Solution Evaluation and Updating.

## RESULTS SUMMARY

Comparison results on the small size case study with 12 bus stop nodes in the network (Lownes, 2010) and the larger application with 10 centroids and 20 stops are shown in Table 2.

**Table 2. Summary of optimal results based on three methods.**

| EE | Complete Enumeration Method | | Tabu Search (Lownes, 2010) | | Adaptive Tabu Search | |
|---|---|---|---|---|---|---|
| Network Size | Optimal Solution | Computing Time (s) | Optimal Solution | Computing Time (s) | Optimal Solution | Computing Time (s) |
| 12 Centroids | $r^* = \{1,6,8,10,1$ $z^* = 499.89$ | 378 | $r^* = \{1,5,6,7,8$ $z^* = 501.74$ | 232 | $r^* = \{1,5,6,7,8\}$ $z^* = 501.74$ | 39 |
| 10 Centroids/20 Bus Stops | $r^* = \{1,8,10,11,$ $z^* = 351.28$ | 20 hours | $r^* = \{1,8,10,1$ $z^* = 360.25$ | 439 | $r^* = \{1,8,10,11,12,$ $z^* = 354.33$ | 90 |

It can be seen from Table 2 that the optimal solution obtained by the Enumeration Method, Static Tabu Search and Adaptive Tabu Search are fairly comparable in quality. However, regarding the computational time, the Adaptive Tabu Search, compared to both the Complete Enumeration method and the static Tabu Search, applied on the two networks, reduces the computing time very significantly. The time taken by the adaptive Tabu Search algorithm to solve for a near-optimal solution falls within the time frame from passengers alight the rail till they get to the feeder bus.

**Table 3. Comparison of two methods' capability in capturing the true optimal solution.**

| | Probability of hitting Global Optimal | |
|---|---|---|
| Network Size | Static Tabu Search (Lownes, 2010) | Adaptive Tabu Search |
| 12 Centroids | 40% | 80% |
| 10 Centroids/20 Bus Stops | 10% | 60% |

Table 3 shows that with the adaptive Tabu search algorithm, 8 times out of 10 tests, the global optimal solution for the network with 12 stops was captured, and for the 10 centroids/20 stops network, 60% percent of the tests identified the true optimal solution. Using the static Tabu Search, the probabilities of finding the global optimal solution are 40% and 10% for the 12 nodes and 10/20 nodes networks. One might expect that the computational power of the Adaptive Tabu Search could be even more significant on larger networks.

## SUMMARY

In this chapter, the Adaptive Tabu Search based algorithm was developed and applied to two sample case studies. The results show that the Adaptive Tabu Search based solution framework outperforms both the exhaustive search method and the static Tabu Search method (Lownes, 2010). With the inherent strategy in Adaptive Tabu Search to adjust the tenures for each Tabu move according to its performance, the computational time taken to solve the problem has been significantly reduced. Also, the better neighborhood definition and the efficient initial solution construction procedure also helped to reduce the computational time to realize real-time operation.

The next chapter will employ simulation techniques to identify the minimum fraction of passenger destination information that would guarantee the practical value of real-time optimization for the CRCNDP. Since there are many practical obstacles in gathering complete passenger travel destination information especially on a real-time basis, this issue must be addressed.

# CHAPTER 5. NUMERICAL TESTING RESULTS

The meta-heuristic algorithm developed for the CRCNDP in the previous chapter optimizes the circulator route assuming that all passengers alighting the arriving train would provide their destination information. However, this is practically difficult to realize for many reasons, passengers' unwillingness to give out their information, technical communication obstacles between passengers and transit operators and so on. Therefore, another question comes with our algorithm, which is what fraction of passenger destination information would guarantee the practical value of real-time optimization for the CRCNDP. In this chapter, a series of scenarios will be developed to characterize the potential range of destination sampling fraction cases and how they depart from the base case described in the previous chapter. A Monte Carlo simulation technique is employed to determine how many and which passengers would provide their destination information in each scenario. The meta-heuristic algorithm will be applied to each case and the aggregate of all simulated cases will describe the "distribution" of likely differences. This frequency distribution of likely differences, compared to the base case defined as the average performance of all possible routes, can be used to easily describe the potential value of real time optimization.

## NUMERICAL TESTS

To justify the value of CRCNDP optimized based on limited destination information, a few assumptions must be made.

1. All passengers will board the circulator bus regardless of the actual route configuration, whether or not they have provided their destination information.
2. All passengers have and use perfect information on which stop to get off once they board the bus, that is to say, they know the nearest stop in the route to their destination and will alight at this stop.
3. For those calculated optimal routes that violate the seamless transfer constraint due to more dwell time with the load of all passengers in the train, more circulator buses will operate along the same route to guarantee that each bus has reduced dwell time and can come back to the rail station within the train headway time.

With the same 12-node and 10 Centroids/20 Stops network configurations used for algorithm development, it is assumed that the same sets of passengers will arrive at the rail stations. The evaluation procedure defines 7 scenarios to characterize the potential range of destination cases, that is 30%, 40%, 50%, 60%, 70%, 80% and 90% of all passengers alighting the train would have provided their destination information. For each scenario, Monte Carlo techniques are used to decide randomly which passengers would give their information and form a demand vector for the Adaptive Tabu Algorithm to get an optimal route. Then all passengers would board the circulator bus running along this optimal route, and a total cost associated with this optimal route and the set of passengers will be calculated as the evaluation result. For each scenario, the above procedure will be repeated 100 times in this numerical test, and the average total cost and its standard deviation will be recorded for scenario comparison.

## Testing Results

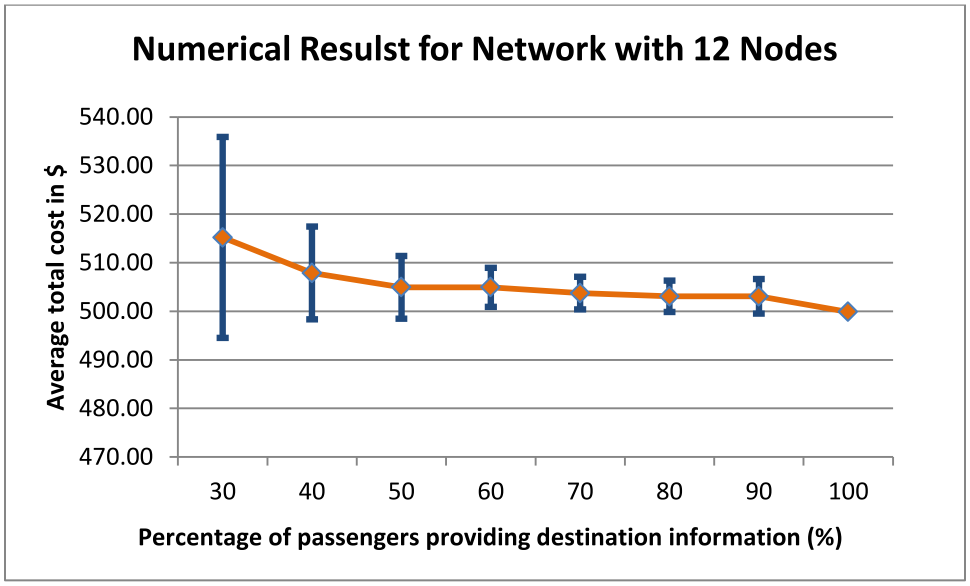

**Figure 5.1. Numerical Results for 12-node Network.**

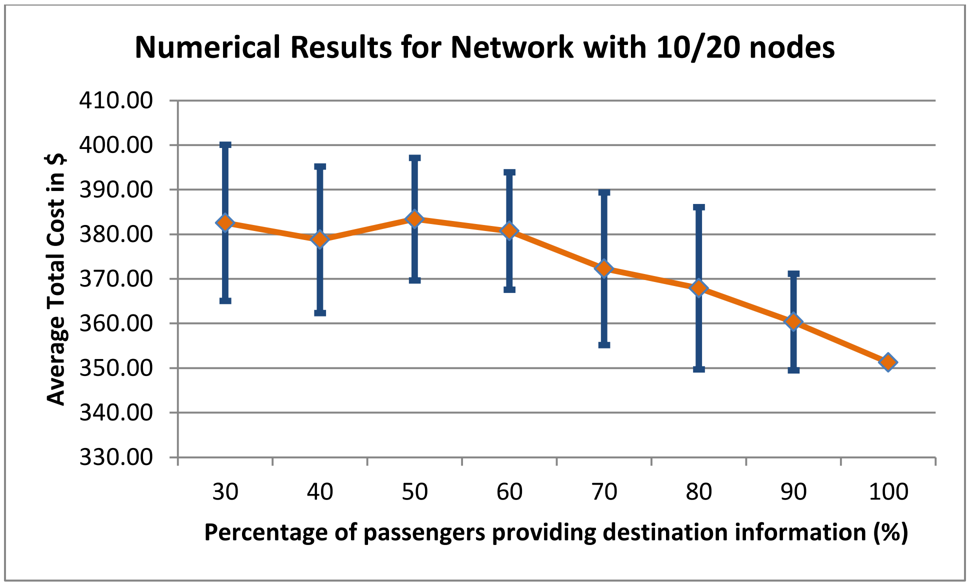

**Figure 5.2. Numerical Results for 10 Centroids/ 20 stops Network.**

The results in Figure 5.1 and Figure 5.2 show that with more passengers providing their destination information, the algorithm for solving real-time CRCNDP produces a better route configuration to serve all passengers boarding the feeder bus. This is intuitive in the sense that accurate information of passenger demands helps the model to perform better. In the 12 node network, the variability associated with the performance of the optimal solution obtained based on different sample sizes becomes larger as the fraction of known destinations decreases. Large variability, in transportation systems, usually indicates unreliable service, and is undesirable. However, the variability based on different samples for the 10 Centroid/20 Stops network doesn't behave exactly the same way. Although the trend of variability can be generalized as decreasing with the increasing percentage of known destinations, the variability in the scenarios where 70% and 80% of passenger destinations are obtained is slightly higher than that in the scenarios where less passenger information is provided. This misbehavior can be introduced by the meta-heuristic algorithms since they aim to find an near-optimal solution and there is no guarantee that every time the same solution is obtained and that a bound can be stated on the objective value. Also, it can be seen after comparing the variability between the two scenarios on the networks where the same amount of travel information can be obtained, the variability in the 10/20 nodes network is always larger than in the 12 node network. With more nodes in the network the solution space is significantly enlarged. The local search path taken by the Tabu Search algorithm is diversified and the optimal solutions obtained can be quite different from each other. And it turns out that the variability in larger network is substantial.

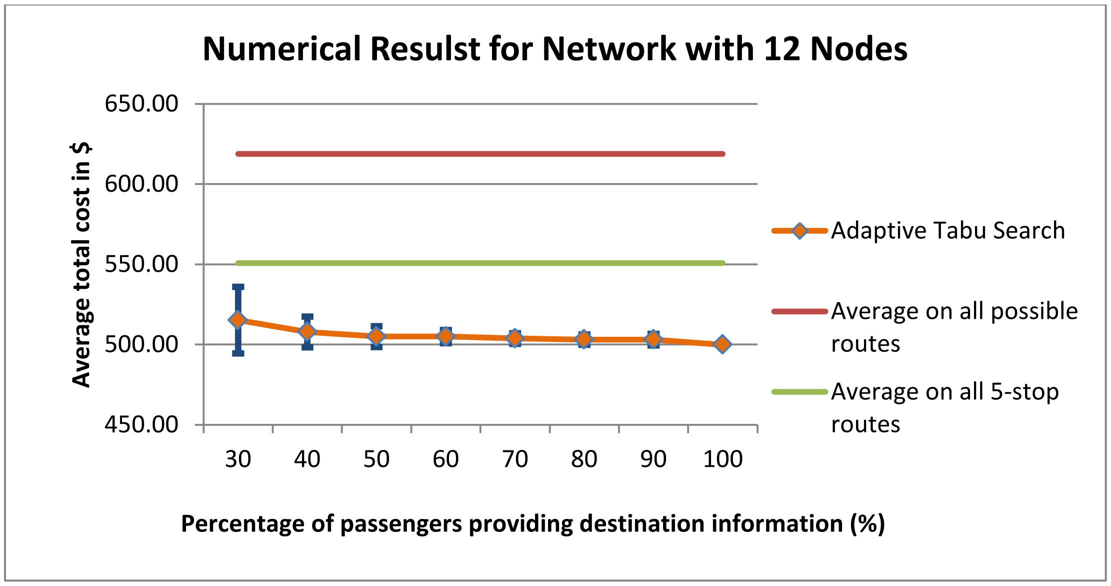

**Figure 6.1. Numerical Results for 12-node Network.**

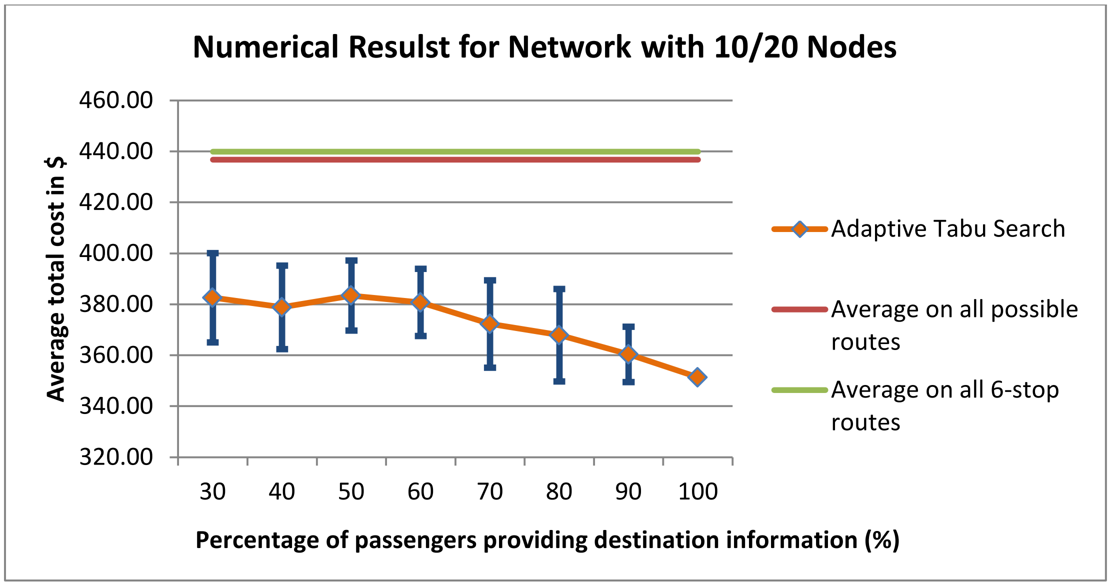

**Figure 6.2. Numerical Results for 10 Centroids/ 20 stops Network.**

In Figure 6.1 and Figure 6.2, the red lines show the average performance of all possible routes in the two sample networks. The average performance of all possible routes was used as a base case to test the practical value of the real-time route optimization. It can be seen that the optimized route obtained by the Adaptive Tabu Search algorithm reduced the total cost significantly compared to the base case. In both cases, even if only 30 percent of passengers provide their travel information, the optimized route obtained based on their input can still perform better than the average performance of all possible routes.

As the performance of the optimal solution obtained based on the Adaptive Tabu Search improves with the amount of available travel destination information, the following is a bold assumption made by the author: there is a threshold across which more destination data given by additional passengers would guarantee the practical value of searching for the optimal route obtained by the meta-heuristic method against either staying with an currently operating route or the average performance of all possible routes.

## Proposed Methodology to find the minimum destination information

To find the threshold across which more destination data given by additional passengers would guarantee the practical value of finding the optimal solution on a real-time basis, the following procedure is proposed:

1. Set up the heuristic or meta-heuristic algorithm for solving the CRCNDP problem;
2. Assume a set of arriving passengers and each of them with fixed destination information. Since this procedure is mainly designed for testing purposes, arriving passengers and their destination information can be simply assumed, or based on observed average demand data and land use information, or it could be gathered from a set of arriving passengers on a real-time basis.

3. For this specific set of passengers, employ a Monte Carlo sampling technique to randomly select samples of passengers who provide their destination information. In this step, two parameters must be predefined: the percentage of passengers to provide travel information, and the number of random samples in each scenario where the fraction of "know destination" is fixed.
4. Run the heuristic or meta-heuristic algorithm for each single sample selected in step 3 and obtain the optimal solutions. Then load all the passengers to the optimal routes achieved in each sample and calculated the objective function value. It is obvious that not all the optimal routes would still be optimal for the set of all arriving passengers, the penalty for non-optimal design is accounted for by additional cost caused by long walk trips, or could be taken into consideration by other ways if the model was formulated differently.
5. Compute the average objective function value and its standard deviation for each scenario representing a chosen fraction of known travel destinations.
6. Define a base case, which can be either the average performance of all possible routes for the network or the currently operating route. It is easier to evaluate the current route by loading passengers on the route and calculating the total cost. If the average performance of all possible routes is selected as the base case, when the size of the network gets large, it might be too time-consuming or computational complex to exhaust all possible solutions to finally produce a base case result for comparison. In this case, sampling techniques over the entire solution space can be employed to reduce computational efforts.
7. Compare the average cost of each scenario to the base case, the minimum fraction of travel information to guarantee the real-time optimization to have equivalent performance as the current route or the average case will be identified.

Employing the Tabu Search algorithm to solve for an optimal solution is a problem-specific matter. There are no general guidelines to define for which problem and under what conditions the optimality is guaranteed. Numerical tests before its application in practice, especially for real time operation tend to be necessary. With the aim to provide a route configuration that is superior compared to those currently operating, these numerical tests combined with simulation techniques can to some degree indicate the level of cooperation needed from transit users – that is their willingness and capability in providing real-time travel information.

## SUMMARY

In this chapter numerical tests have been performed on the two sample cases to see the effect of available destination information on the performance of the optimal route obtained by the proposed solution algorithm. A series of scenarios have been developed to characterize the potential range of destination sampling fraction cases and how they depart from the base case described in the previous chapter. Monte Carlo simulation techniques are used for sample selection in each scenario. Performance location and variability behaves as expected. With larger travel data sampling fractions, the solution performs better and behaves more like the true optimal. This chapter has also proposed a methodology to find the threshold across which more destination data given by additional passengers would make no significant improvement to the optimal solution.

# CHAPTER 6. CASE STUDY

In previous chapters, the development and performance of the CRCNDP and the adaptive Tabu Search solution method was illustrated, and examples of different network structures have been tested. During the developmental stages it was found that the adaptive Tabu search method performed well, providing good (and in some cases, optimal) solutions in a very short amount of time compared to the enumerative method. This performance generates confidence in the adaptive Tabu search's ability to provide good solutions on a real-time basis. However, these examples are limited since they are constructed using a small network for experimental purposes. To guarantee its feasibility for real size problems, the adaptive Tabu Search solution method will be implemented to three real size cases abstracted from the Martin Luther King (MLK) station of the new MetroRail system in Austin, Texas.

## NETWORK DESCRIPTION

Figure 7 provides an aerial overview of the MLK Station area. The black circle centered on the MLK station represents a 2-mile radius coverage area used to limit the demand zones served by the station. The station, UT campus and the CBD are all identified in the map.

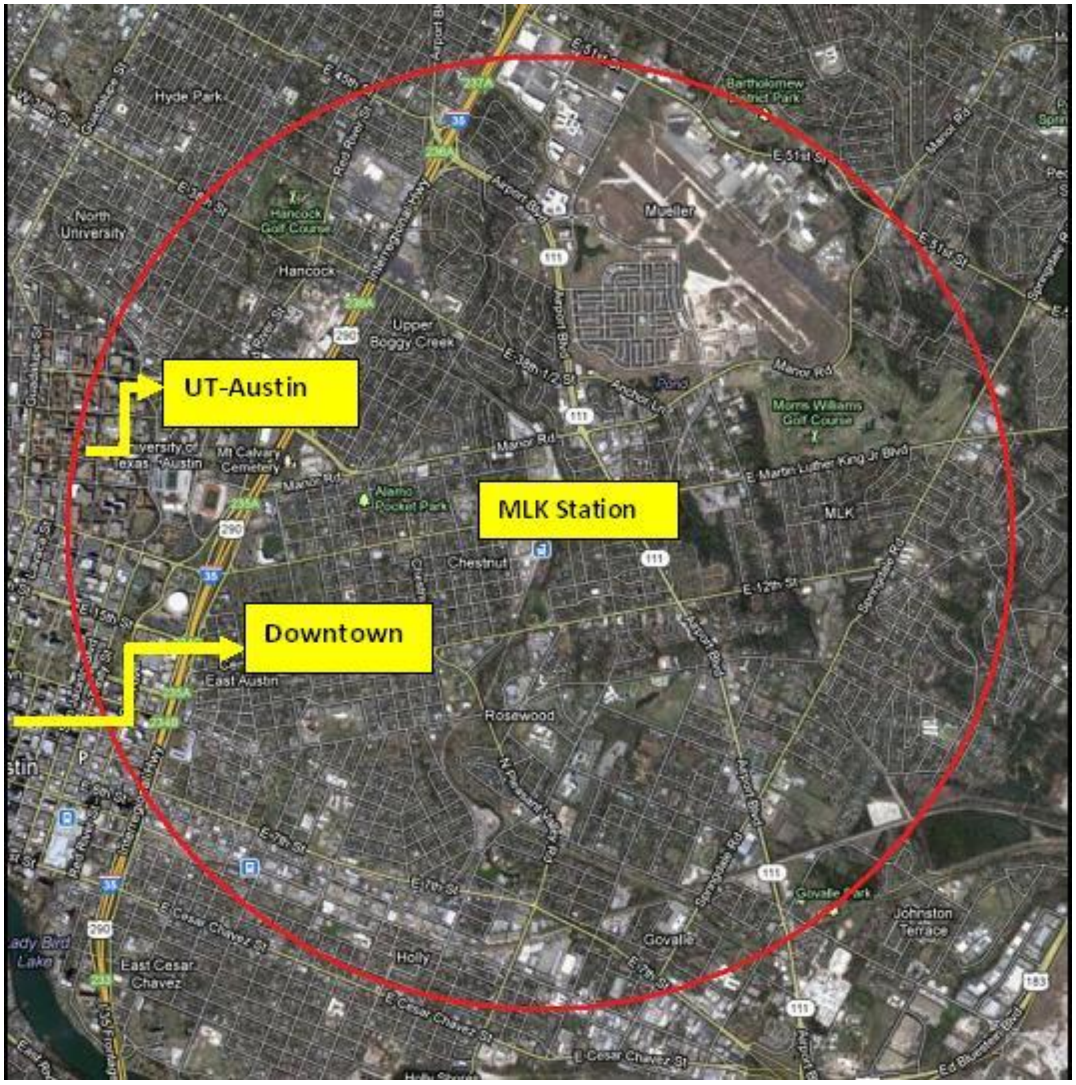

**Figure 7. Overview of MLK Station Coverage Region.**

In Figure 8 the red square identifies the MLK station. The green circle centered on the MLK station represents a 2-mile boundary about the station used to limit the number of demand zones that are considered in the analysis. The demand centroids are located as shown as green squares and represent those passengers that are theoretically considered to have access to commuter rail at the destination end of the commuter rail trip. In this case study application, there are 46 demand centroids within the 2-mile boundary and 84 candidate stop locations (pink dots) from which the route to serve MLK Station will be constructed. The demand of each of the 46 centroids is given in Table 4.

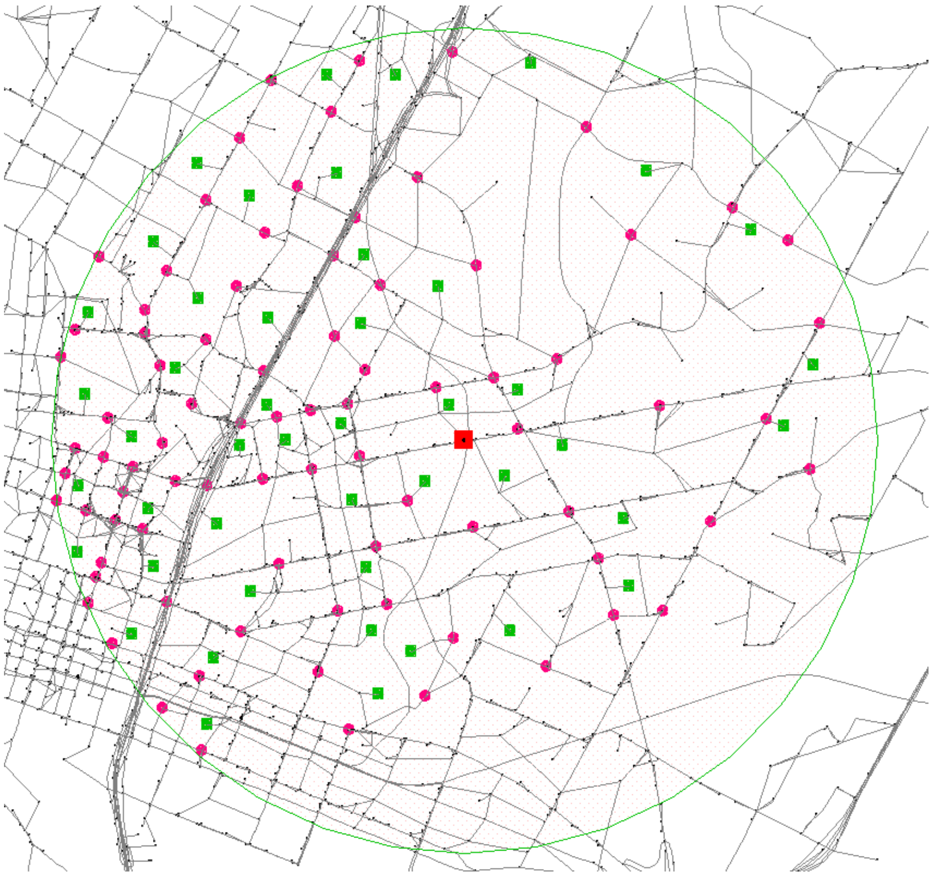

**Figure 8. Demand Centroid (46) and Candidate Stop Locations (84) within MLK Station Coverage Region.**

**Table 4. Case Study Zonal Demand.**

| Centroid | Demand | Centroid | Demand | Centroid | Demand | Centroid | Demand |
|---|---|---|---|---|---|---|---|
| 1 | 6 | 13 | 5 | 25 | 1 | 37 | 4 |
| 2 | 6 | 14 | 2 | 26 | 2 | 38 | 2 |
| 3 | 5 | 15 | 6 | 27 | 4 | 39 | 2 |
| 4 | 0 | 16 | 3 | 28 | 3 | 40 | 4 |
| 5 | 6 | 17 | 2 | 29 | 3 | 41 | 1 |
| 6 | 2 | 18 | 1 | 30 | 6 | 42 | 1 |
| 7 | 10 | 19 | 5 | 31 | 4 | 43 | 1 |
| 8 | 10 | 20 | 3 | 32 | 0 | 44 | 10 |
| 9 | 6 | 21 | 4 | 33 | 4 | 45 | 0 |
| 10 | 10 | 22 | 2 | 34 | 0 | 46 | 1 |
| 11 | 1 | 23 | 3 | 35 | 3 | | |
| 12 | 6 | 24 | 6 | 36 | 1 | | |

Test Results showed that the adaptive Tabu Search Algorithm solves this case study on average in 105 seconds. This is much less than the on-train travel time for essentially all passengers. Even if destination information cannot be collected before passengers alight the train, in which case the algorithm could still be run while passengers walk from the train station to the feeder bus. This time is estimated as 3 to 5 minutes and it is still adequate for the algorithm to run to a near-optimal or optimal solution. Results for 10 test runs are shown in Table 5.

**Table 5. MLK Station Case Study Results.**

| Run # | Objective Value | Optimal Solution | Calculation Time (S) |
|---|---|---|---|
| 01 | 1890.47 | 1, 16, 38, 41, 48 | 103 |
| 02 | 1912.9 | 1, 16, 41, 48, 64 | 107 |
| 03 | 1864.56 | 1, 16, 36, 41, 48 | 100 |
| 04 | 1962.12 | 1, 38, 45, 47, 78 | 144 |
| 05 | 1942.02 | 1, 8, 15, 38, 80 | 102 |
| 06 | 1897.83 | 1, 16, 38, 47, 75 | 103 |
| 07 | 1905.52 | 1, 16, 46, 75, 84 | 95 |
| 08 | 1895.84 | 1, 15, 16, 38, 75 | 99 |
| 09 | 2062.43 | 1, 8, 15, 23, 38 | 101 |
| 10 | 1903.77 | 1, 15, 16, 38, 41 | 100 |

As shown in Table 6.2, the computational times for each execution of the adaptive Tabu Search algorithm lie in the range from 1.5 minutes to 2 minutes on a 2 dualcore, hyperthreading 3.73 GHz Xeon processor with 2GB of memory. In the case that the optimization must be performed while transit users are walking to the feeder bus from the rail station, this algorithm should work fine to get a fairly good solution. In the case that destination information can be collected and well-prepared way before transit users alight the train, this algorithm can be executed more than one time to better search for a good solution. In short, the adaptive Tabu search should be applicable to practical problems in a real feeder system serving a commuter rail station.

In figure 9, the best route for this case study is depicted in the map. Since in the assumed demand data, centroids associated with the UT campus were intentionally assigned larger demands than other centroids, the optimal route, not surprisingly, provides service primarily to the University area.

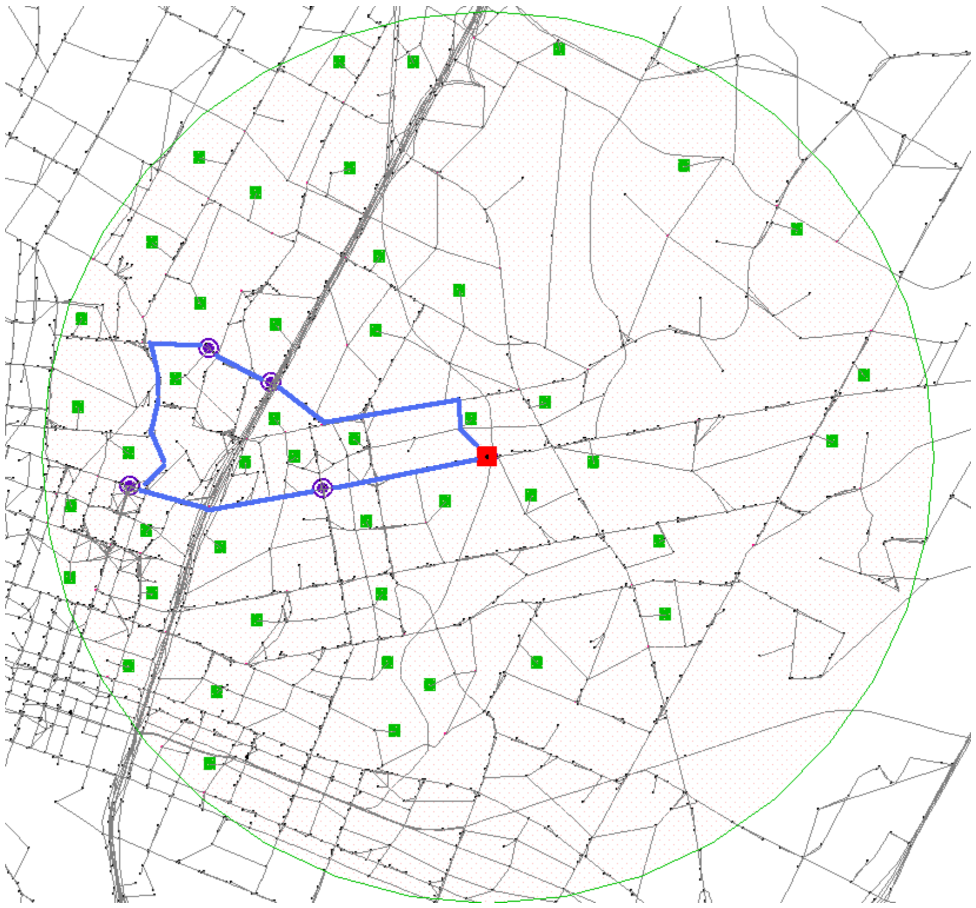

**Figure 9. Route solution for the 46/84 MLK network: Best Route.**

Although the network structure and size are fairly representative for a feeder system serving a commuter rail station, additional tests are performed to find the computational time required for problems of different sizes. A smaller network with 30 centroids and 50 candidate bus stops and a larger network with 62 centroids and 118 candidate bus stops are both constructed for experiments. The network configuration and the optimal route obtained based on the Adaptive Tabu Search are shown in Figure 10, 11, 12 and 13. The demand data is included in Table 6 and Table 7.

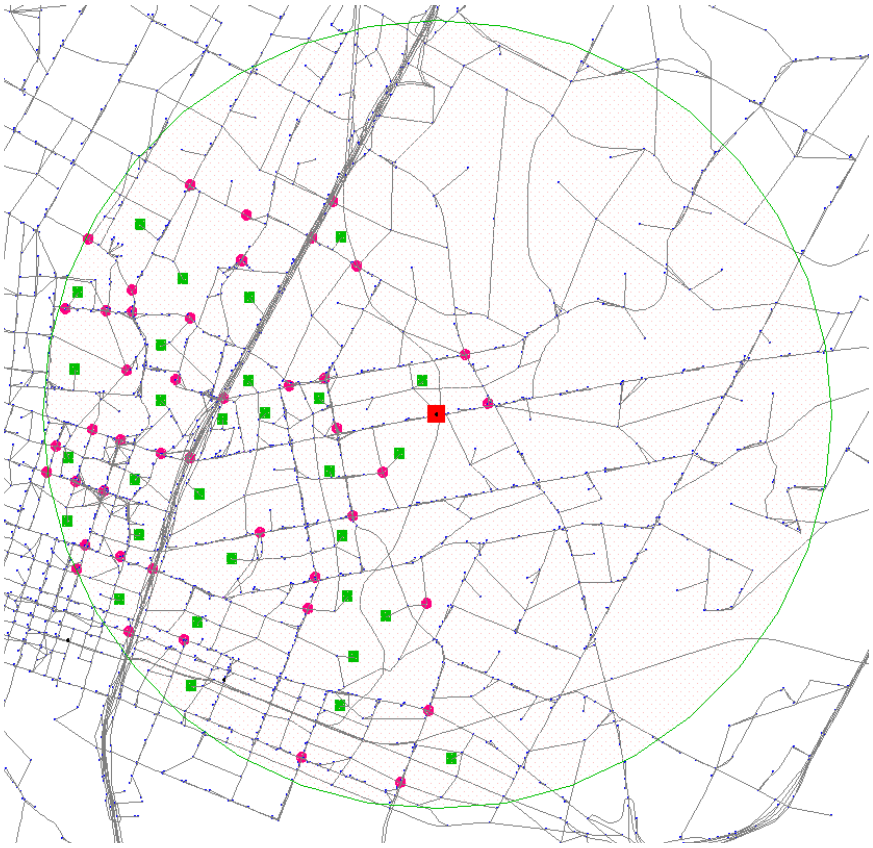

**Figure 10: Demand Centroid (30) and Candidate Stop Locations (50) within MLK Station Coverage Region**

**Table 6. Case Study Zonal Demand in the 30/50 network.**

| Centroid | Demand | Centroid | Demand | Centroid | Demand |
|---|---|---|---|---|---|
| **1** | 3 | **11** | 10 | **21** | 6 |
| **2** | 6 | **12** | 0 | **22** | 6 |
| **3** | 1 | **13** | 5 | **23** | 6 |
| **4** | 5 | **14** | 4 | **24** | 0 |
| **5** | 0 | **15** | 2 | **25** | 5 |
| **6** | 1 | **16** | 4 | **26** | 4 |
| **7** | 10 | **17** | 4 | **27** | 1 |
| **8** | 5 | **18** | 1 | **28** | 6 |
| **9** | 10 | **19** | 0 | **29** | 2 |
| **10** | 4 | **20** | 0 | **30** | 3 |

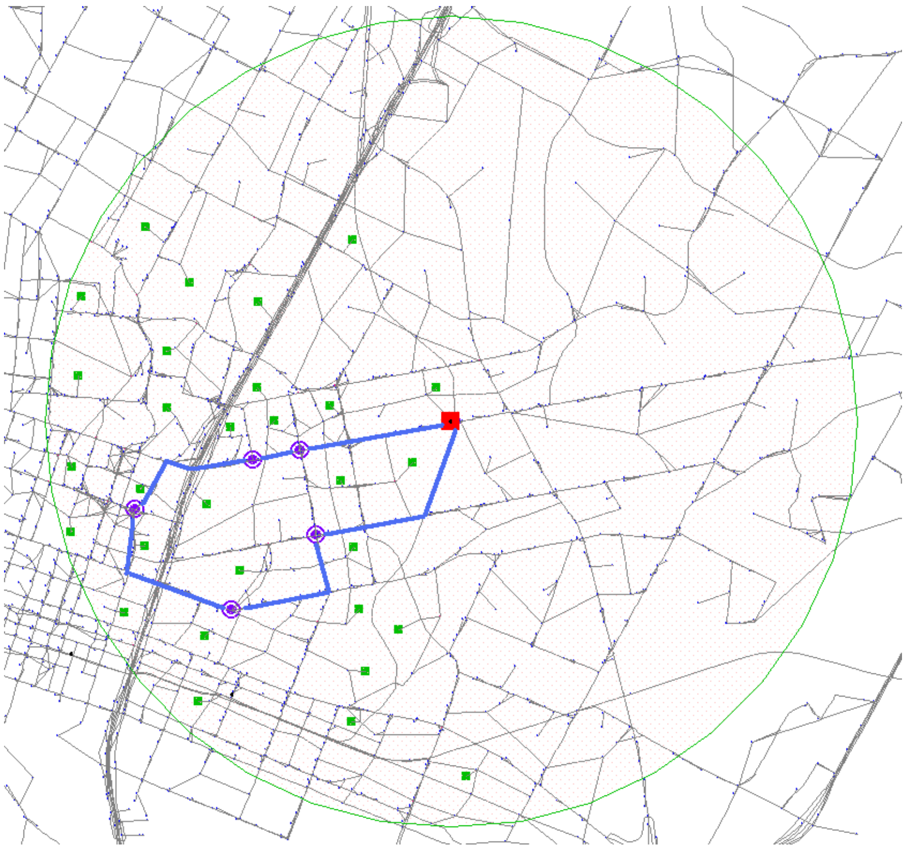

**Figure 11. Route solution for the 30/50 MLK network: Best Route.**

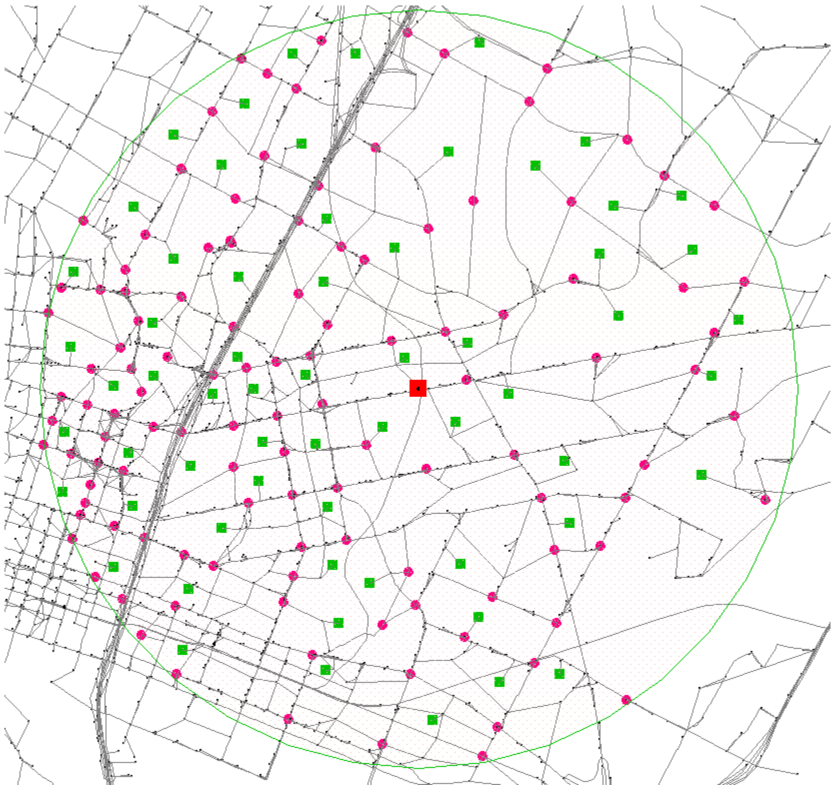

**Figure 12. Demand Centroid (62) and Candidate Stop Locations (118) within MLK Station Coverage Region.**

Table 7: Case Study Zonal Demand in the 62/118 network

| Centroid | Demand | Centroid | Demand | Centroid | Demand | Centroid | Demand |
|---|---|---|---|---|---|---|---|
| **1** | 2 | 17 | 3 | **33** | 3 | **49** | 1 |
| **2** | 3 | 18 | 2 | **34** | 4 | **50** | 10 |
| **3** | 4 | 19 | 2 | **35** | 3 | **51** | 0 |
| **4** | 4 | 20 | 3 | **36** | 2 | **52** | 2 |
| **5** | 4 | 21 | 2 | **37** | 3 | **53** | 0 |
| **6** | 3 | 22 | 3 | **38** | 4 | **54** | 3 |
| **7** | 10 | 23 | 2 | **39** | 0 | **55** | 0 |
| **8** | 2 | 24 | 0 | **40** | 4 | **56** | 2 |
| **9** | 10 | 25 | 3 | **41** | 1 | **57** | 3 |
| **10** | 3 | 26 | 4 | **42** | 0 | **58** | 2 |
| **11** | 10 | 27 | 2 | **43** | 4 | **59** | 0 |
| **12** | 2 | 28 | 4 | **44** | 0 | **60** | 3 |
| **13** | 4 | 29 | 2 | **45** | 2 | **61** | 2 |
| **14** | 4 | 30 | 3 | **46** | 0 | **62** | 2 |
| **15** | 4 | 31 | 1 | **47** | 0 | | |
| **16** | 0 | 32 | 1 | **48** | 3 | | |

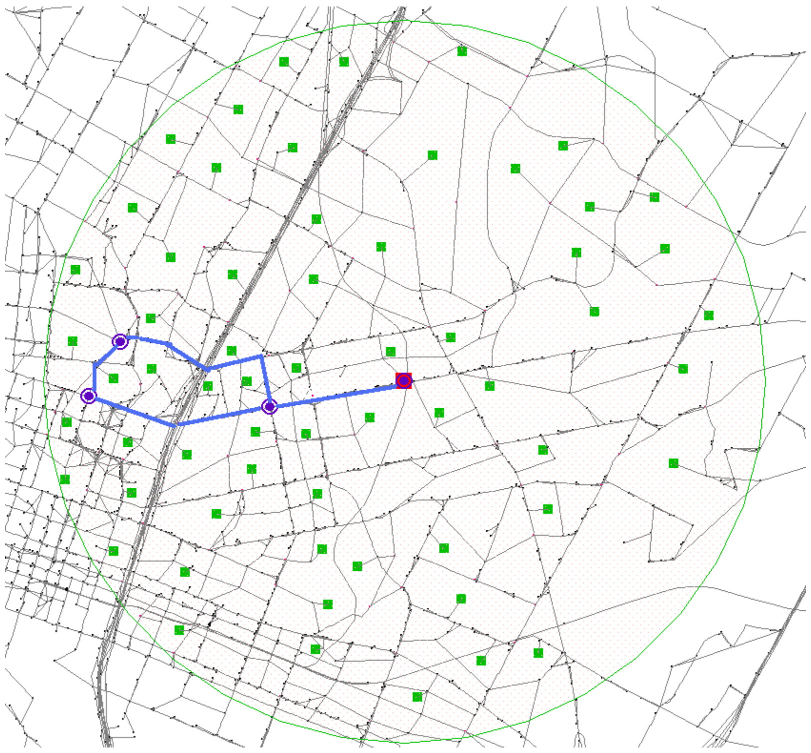
Figure 13. Route solution for the 62/118 MLK network: Best Route.

The demand dataset, for these two networks includes random demand assignments except for the centroids associated with the UT campus where larger demands have been assumed. Again, the optimal route provides service to the University area with priority because of the larger specified demands. As shown when centroids are densely located as in the 62/118 MLK network, the probability of a given stop serving more than one destination is higher, so fewer stops can serve a larger area, while if demand centroids are far apart, as in the 30/50 MLK network, the feeder buses must stop at more locations to provide service to an equivalent number of destinations.

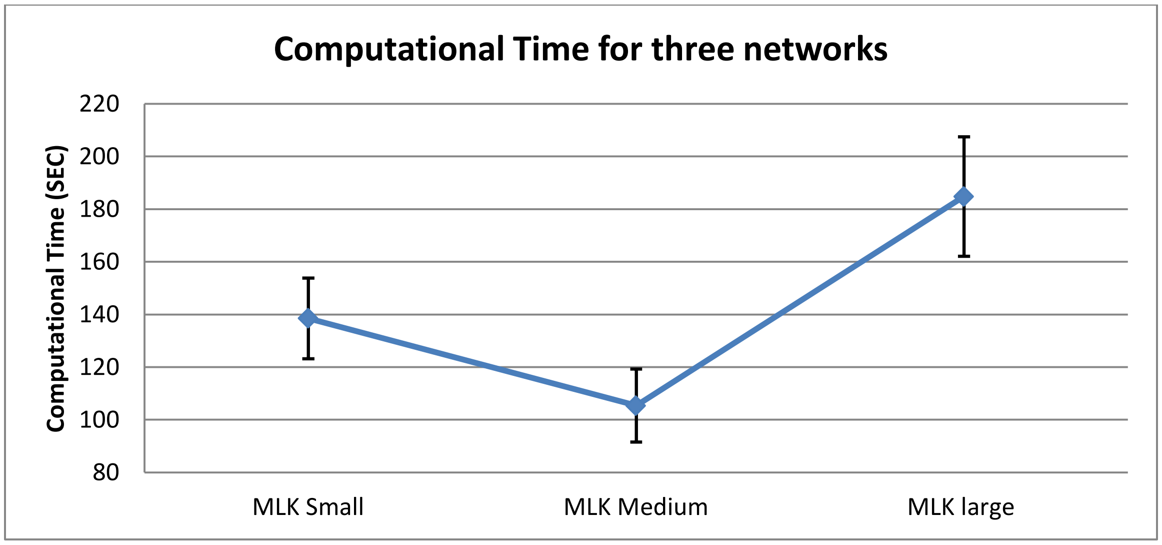

**Figure 14. Computational Time for three networks of different sizes.**

The computation complexity of the Tabu Search based heuristic developed for the CRCNDP are mostly dependent on the predefined number of iterations. Hence, without looking closely at the problem and the specific algorithm developed for it, the complexity of computational efforts taken by the algorithm would not be revealed. However, to some degree meta-heuristic methods guarantee that with the scale of the problem increasing, the computational efforts do not necessarily expand dramatically and it provides practically useful values where solution time duration is a tight constraint for the problem solving.

In the CRCNDP problem, it can be informally said that the adaptive Tabu Search method adequately takes the place of the exhaustive search process to make decisions on which potential stops to be included in the route, and it significantly reduces computational effort. While complexity analysis is not valid on heuristic methods, approximate estimation of computational time taken by the algorithm is still worthwhile.

The first decision to be made within the adaptive Tabu search algorithm is the number of stops used to construct the route. Tests on the network with 46 centroids and 62 stops within the solution space show the algorithm searched for solutions that included from 2 stops to 5 stops, as

4 major steps (2, 3, 4 and 5). The algorithm searches for solutions with 2 stops, and then constructs a starting solution with 3 stops by smartly adding one stop to the best current solution with 2 stops, and so on. In each major step, local searches of a user-defined number of iterations are performed to update the best current solution. After local search over the solution space with 5 stops was completed, the algorithm selects the best solution it has ever encountered as the optimal solution. The computational time taken by the algorithm can be roughly calculated as follows:

$$\text{Time} = t * C * (\text{iterations}) * S$$

$t$ is the time taken for a single route optimization with selected stops. Each route optimization has constraints as $O(|r||r|)$ and variables as $O(|I||I|)$, again $r$ is the set of selected stops and $I$ is the set of all candidate stops. $C$ is a factor larger than or equal to 1 and is used to account for additional evaluations performed in each major step due to the existence of infeasible or interior solutions. $S$ is the number of major steps defined in the algorithm.

In Figure 14, the blue dots show the average computational time for each of the three tested networks with 10 random runs and the error bars represent the standard deviation of computational time of these random samples. The average computational time for the three tested networks formed a polyline. The smaller network took longer time to solve due to the fact that the algorithm has to go through 6 major steps to finally search over solution space that contains 7 stops to get a good solution while the medium size network was solved in only 4 major steps. As to solve the medium size network and the large size network took the same number of major steps, the time taken for each evaluation in every major step becomes the dominant factor. Although the number of steps is the same as the other networks the network with more centroids and stop nodes requires significantly more computational effort in each single evaluation. This explains the second part of the polyline.

## SUMMARY

In this chapter, the adaptive Tabu Search solution method has been implemented to three real size cases abstracted from the Martin Luther King (MLK) station of the new MetroRail system in Austin, Texas to test its feasibility for real-time operation. The results have shown that the solution algorithm works for the small, medium and large size network, although the medium size network is representative for real world problems. A detailed explanation for the computational time difference for the three cases has also been provided after analysis over the algorithm procedures. The developed solution framework can be a candidate option for real-time operation in CRCNDP.

# CHAPTER 7. SUMMARY, CONCLUSIONS AND RECOMMENDATIONS

Commuter rail has been widely recognized as an effective and practicable solution to ever-worsening urban congestion caused mainly by commuter trips since it offers the potential for attractive, high-quality rapid transit service at a reasonable cost. New commuter rail systems are typically built along old freight rail tracks. Although use of existing right-of-way reduces time and cost of construction, the paths do not provide for optimal station locations. Therefore minimizing traveler access time from rail stations to final destinations is critical for commuter rail to be a reasonable option for commuters.

This report aims to develop a robust optimization tool using meta-heuristic approaches to design circulator routes on a real-time basis with real-time demand data. The CRCNDP involves minimization of generalized costs subject to a variety of constraints. The decision one seeks to make is the determination of a circulator configuration including the stops to be visited and the route among them. An adaptive Tabu Search method is employed to solve the problem in an efficient manner to realize real-time operation.

The sections in this chapter are organized as follows. In section 7.1, the principal features of the solution approaches designed for the CRCNDP are reviewed and a summary of conclusions for the numerical results derived from computational tests is discussed. Section 7.2 presents a brief discussion of the limitations of the current approaches and possible directions for future research are also given.

## SUMMARY AND CONCLUSIONS

As mentioned, the CRCNDP problem addressed in this report involves finding a feeder bus route configuration that achieves a desired objective with a variety of given constraints. Related literature describing previous solution approaches to the Transit Route Network Design problem has been reviewed. As mentioned by several researchers including Baaj (1990), several main sources of complexity often preclude finding a unique optimal solution for the Transit Route Network Design problem. Some of these are also applicable to CRCNDP and they are discussed as follows: (1) great difficulty in defining the decision variables and expressing the objective function; (2) combinatorial complexity arises from the discrete nature of the route design problem, making the CRCNDP NP-hard; (3) many important tradeoffs among conflicting objectives need to be addressed, making the CRCNDP an inherently multi-objective decision making problem. Additionally, for our solution approaches to be applied to real-time operation, computational efficiency is another challenge to this report.

Previous approaches that were used to solve the Transit Route Network Design problem can be generally categorized into two major groups: (1) analytical optimization models for idealized situations; (2) meta-heuristic approaches for more practical problems. Few of them really focused on the circulator route design, not to mention an algorithm developed for real-time operation. Building on several previous approaches, mainly the Tabu Search method developed by Lownes (2008), the solution methodology proposed in this report includes the following major features: (1) Ability to account for the inherent tradeoffs between conflicting performance-measures; (2) Systematic heuristic methods for circulator route generation and improvement; (3) systematic use of context-specific knowledge to guide the search technique; (4) Ability to provide a route configuration that includes the exact bus stop location rather than corresponding to just demand centers; (5) Computational efficiency to obtain an optimal or near-optimal CRCNDP solutions to apply real-time control.

The proposed approaches – Adaptive Tabu Search consists of three main components: an initial candidate route generation procedure that generates a feasible route as a starting point; a neighborhood definition and search procedure that searches locally for a solution with better performance; a memory mechanism that guides the search procedure to avoid cycling and to search beyond local optima for a possibly global optimum.

Numerical tests have been performed to find the minimum fraction of passengers' destination information that guarantee the practical value of the real-time optimization control. And results have shown as expected that with more passengers providing travel information, the algorithm produces a better route configuration to serve all passengers boarding the feeder bus and that the variability associated with the performance of the optimal solution based on sample size becomes larger as the fraction of known destinations decreases. However, a bold assumption is made in this report, there is a threshold across which that more destination data given by additional passengers would guarantee the practical value of real-time optimization. The trend can be seen from the figures in Chapter 5 and is intuitively supported by the inherent nature of meta-heuristic method aimed at obtaining near-optimal or good enough solutions in an efficient manner rather than the global optimum. The procedures are proposed to find the threshold for the minimum fraction of travelers that would need to report their destinations via smart phone to guarantee the practical value of optimization based on real-time collected demand.

The adaptive Tabu Search algorithm was finally applied to three case study networks that surround the MLK station on the Austin MetroRail commuter rail line. The three networks are marked as small, medium and large networks. The number of demand centroids in the three networks is 30, 46 and 62 respectively, and the number of candidate bus stop locations is 50, 84 and 118. Although, from the practical point of view, the size of the medium network is large enough for a real world circulator system, the algorithm developed in this report produced a good solution for all these three problems in a limited time domain.

## FUTURE RESEARCH DIRECTIONS

One extension of this work is the accommodation of multiple routes in the formulation and solution methods. The current methods assume that multiple vehicles follow the same route to provide unique service. This situation assumption facilitates the communication with regard to transit service between operators and users. There are no ambiguities regarding which vehicle should the passenger board since all buses are going to travel along the same route. Incorporation of multiple routes would enable the system to reduce the amount of long walk trips and the associated costs. The inclusion of multiple routes will certainly increase the complexity and of the current solution method. A more sophisticated initial solution construction method and a more complicated neighborhood search procedure might have to be developed to further guarantee the possibility of real-time operation.

For this algorithm to be implemented into practice, additional application should also be developed. A smart phone app to bridge the communication between transit operators and users would be necessary. This app should enable the users to provide destination information in various formats such as TAZ centroids, block level or a detailed street number address.  And it should also facilitate the transit operator data collection and preparation.

The numerical results were tested using a computer equipped with a 2 dualcore, hyperthreading 3.73 GHz Xeon processor and 2GB memory. From the transit operation perspective, this type of machine or an even faster machine is achievable. So that an operationally and economically efficient optimal circulator network can be obtained in a real-time basis, making the commuter rail a more viable option for commuters. Benefits can be gained through less traffic congestion, reduced air pollution and lower energy consumption.

# APPENDICES

## APPENDIX A: CODE ILLUSTRATING IMPLEMENTATION OF THE ADAPTIVE TABU SEARCH ALGORITHM

```
#include <iostream>
#include <cstdlib>
#include <fstream>
#include <iomanip>
#include <ctime>
#include <string>
#include <sstream>

using namespace std;

//Function to obtain the index number of the maximum demand node (used in
generating initial solution)

int maxIndex(double a[], int size, int check[]);

//Function to obtain the index number of the minimum demand node (used in
generating initial solution)

int minIndex(double a[], int size, int check[]);

//The qsort() function... to sort the set r[][]

int compare (const void * a, const void * b)
{
  return ( *(int*)a - *(int*)b );
}

int factorial (int num)
{
 if (num==1)
  return 1;
 return factorial(num-1)*num; // recursive call
}

int i,j,t,z,o,v,check,check2;

int carry[84][84] = {0};

int main()
{
      srand(time(0));  // Initialize random number generator.

    //The time function will determine the number of seconds elapsed between
the start and finish of the program

      time_t t1, t2;
      t1 = time (NULL);

      /*the variables and paramters defination*/

      const int CENTROIDS = 46;
    const int NUMSTOPS = 84;

    double demandCEN[CENTROIDS] = {0};
    double demand[NUMSTOPS] = {0};
      double gamma[NUMSTOPS][CENTROIDS] = {0};
      int lambda[NUMSTOPS][NUMSTOPS] = {0};

      /*demand and distance data from text files*/

      ifstream fin("demand.txt");

      for (i=0; i < CENTROIDS; i++) {
            fin >> demandCEN[i];
      }

      ifstream fin1("gamma.txt");
```

```
for (i=0; i < NUMSTOPS; i++) {
      for (j = 0; j < CENTROIDS; j++){
            fin1 >> gamma[i][j];
      }
}

for ( i = 0; i < NUMSTOPS; i++){
      for (j = 0; j < CENTROIDS; j++) {
            if (gamma[i][j]<= 1980) {
                  demand[i] = demand[i] + demandCEN[j];
            }
      }
}

cout<<"demand /";

      for (i = 0; i < NUMSTOPS; i++) {
            cout<<demand[i];

            if (i < (NUMSTOPS - 1)) {
            cout<<", ";
            }
      }

      cout<<"/;"<<endl<<endl;

ifstream fin2("lambda.txt");

for (i=0; i < NUMSTOPS; i++) {
      for (j = 0; j < NUMSTOPS; j++){
            fin2 >> lambda[i][j];
      }
}
```

```
	int n[NUMSTOPS] =
{1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29
,30,31,32,33,34,35,36,37,38,39,40,41,42,43,44,45,46,47,48,49,50,51,52,53,
54,55,56,57,58,59,60,61,62,63,64,65,66,67,68,69,70,71,72,73,74,75,76,77,78,79,
80,81,82,83,84};

	double zoptimal[6] = {1000000000};
	int roptimal[NUMSTOPS][6] = {0};


	for (int OUTCOUNT = 2; OUTCOUNT < 8; OUTCOUNT++) {

		int SETSIZE = OUTCOUNT;
		int ID = (SETSIZE + 1);
		int cont,tran1,leave,redo;

		int ** r;
	  //memory allocated for elements of rows.
	  r = new int *[SETSIZE] ;
	  //memory allocated for  elements of each column.
	  for( i = 0 ; i < SETSIZE ; i++ )
	  r[i] = new int[ID];

		int * b;
		b = new int [SETSIZE];
		int * enter;
		enter = new int [ID];

		double * zstar;
		zstar = new double [ID];

		for (i = 0; i < ID; i++) {
			zstar[i] = 0;
		}
```

```
          //Need to include the station (marked as 1)in every set r

          for (j = 0; j < ID; j++) {
                  r[0][j] = 1;
                  }



          double BEST = 1000000000;



          int ** bestset;
        //memory allocated for elements of rows.
        bestset = new int *[SETSIZE] ;
        //memory allocated for  elements of each column.
          for( i = 0 ; i < SETSIZE ; i++ )
        bestset[i] = new int[SETSIZE];

            int flag1[NUMSTOPS] = {0};

            double ratio[NUMSTOPS][NUMSTOPS] = {0};
            double ratio1[NUMSTOPS] = {0};



          int flag[NUMSTOPS*NUMSTOPS] = {0};

            //to generate an initial set r, start with the highest demand
centroids plus the station.

            if (OUTCOUNT < 3) {
                  for (i = 1; i < SETSIZE; i++) {
                        r[i][0] = n[maxIndex(demand, NUMSTOPS, flag1)];
                        int tran = maxIndex(demand, NUMSTOPS, flag1);
                        flag[tran] = 1;
                  }
            }
            cout<<"bestset OUTCOUNT -1 /";

            for (v = 0; v < SETSIZE-1; v++) {
                  cout<<carry[v][OUTCOUNT-1];

                  if (v < (SETSIZE - 2)) {
                  cout<<", ";
                  }
            }

            cout<<"/;"<<endl<<endl;

            if (OUTCOUNT > 2) {

                  for (i = 1; i < SETSIZE - 1; i++) {
                        r[i][0] = carry[i][OUTCOUNT - 1];
                  }

                  do{
                        tran1 = rand()%(NUMSTOPS - 1) + 2;
                      cont =0;

                        for (i = 0; i < SETSIZE - 1; i++){
                              if (r[i][0] == tran1) {
                                    cont++;
                              }
                        }

                  }while (cont > 0);

                  r[SETSIZE - 1][0] = tran1;
            }



   //to sort set r
```

```
for (j = 0; j < ID; j++) {

      for (i = 0; i < SETSIZE; i++) {
            b[i] = r[i][j];
      }

      qsort(b, SETSIZE, sizeof(int), compare);

      for (i = 0; i < SETSIZE; i++) {
            r[i][j] = b[i];
      }
}

//Output the set r to the gams input file, combinations.inc

ofstream file("combinations.inc", ios::out | ios::trunc);

file<<"set r(i) /";

for (i = 0; i < SETSIZE; i++) {
      file<<r[i][0];

      if (i < (SETSIZE - 1)) {
      file<<", ";
      }

}

file<<"/;"<<endl;

system("/usr/local/gams/23.5.2/gams TabuGams lo=2");

ifstream opt("optimal.txt");

opt>>zstar[0];
```

```
    cout<<"set r(i) /";

        for (i = 0; i < SETSIZE; i++) {
            cout<<r[i][0];

            if (i < (SETSIZE - 1)) {
                cout<<", ";
            }
        }

    cout<<"/;"<<endl;

    cout<<zstar[0]<<endl;

    while (zstar[0] < 1) {

        cout<<"set r(i) /";

        for (i = 0; i < SETSIZE; i++) {
            cout<<r[i][0];

            if (i < (SETSIZE - 1)) {
            cout<<", ";
            }

        }

        cout<<"/;"<<endl;

        cout<<"Infeasible!"<<endl;

        leave = rand()%(SETSIZE - 1) + 1;

    // a small do-loop to check that the entering node is not already in the
set r
```

```
            do{
                  redo = 0;
                  enter[0] = rand()%(NUMSTOPS - 1) + 2;

                  for (i=0; i < SETSIZE; i++){
                        if (r[i][0] == enter[0]){
                              redo++;}
                  }
            }while(redo > 0);

            cout<<"Leaving Stop index: "<<leave<<" Entering Stop:
"<<enter[0]<<endl;

            r[leave][0] = n[enter[0]-1];

            cout<<"set r(i) /";

            for (i = 0; i < SETSIZE; i++) {
                  cout<<r[i][0];

                  if (i < (SETSIZE - 1)) {
                  cout<<", ";
                  }
            }

            cout<<"/;"<<endl;

            for (j = 0; j < ID; j++) {

                  for (i = 0; i < SETSIZE; i++) {
                        b[i] = r[i][j];
                  }

                  qsort(b, SETSIZE, sizeof(int), compare);
```

```
        for (i = 0; i < SETSIZE; i++) {
                r[i][j] = b[i];
        }
}

cout<<"set r(i) /";

for (i = 0; i < SETSIZE; i++) {
        cout<<r[i][0];

        if (i < (SETSIZE - 1)) {
                cout<<", ";
        }
}

cout<<"/;"<<endl;

ofstream file("combinations.inc", ios::out | ios::trunc);

file<<"set r(i) /";

for (i = 0; i < SETSIZE; i++) {
        file<<r[i][0];

        if (i < (SETSIZE - 1)) {
        file<<", ";
        }

}
file<<"/;"<<endl;

system("/usr/local/gams/23.5.2/gams TabuGams lo=2");

ifstream opt("optimal.txt");

opt>>zstar[0];
```

```
            cout<<zstar[0]<<endl;

            BEST = zstar[0];
            zoptimal[OUTCOUNT-2] = BEST;

            //zoptimal[OUTCOUNT - 2] = BEST;

            for (int q = 0; q < SETSIZE; q++) {
                  bestset[q][OUTCOUNT-1] = r[q][0];
                  carry[q][OUTCOUNT] = bestset[q][OUTCOUNT-1];
                  roptimal[q][OUTCOUNT-2] = bestset[q][OUTCOUNT-1];
            }

      }


/*An initial feasible solution in r[][0] has been constructed.  This will
serve as the starting point for the tabu search algorithm.  From this point
onward the focus of the algorithm is to smartly search over the neighborhood
and update the best current solution.*/

/*Declare and initialize the tabu parameters.*/


      cout<<"--------Begin Tabu--------"<<endl;

      int zcount = 0;

      int iter = 0;

      if (OUTCOUNT < 3) {

            while (iter < 10) {

      iter++;
```

```
//Set the current set for all neighborhoods to the best from the previous
iteration

            for (i = 0; i < SETSIZE; i++){

            for (j = 1; j < ID; j++) {
                  r[i][j] = r[i][0];
            }
      }



            int added[NUMSTOPS] = {0};
            int removed[NUMSTOPS] = {0};
            int infeasible[NUMSTOPS] = {0};

            int temp4;

//Select a leaving node

check2 = 0;

      do {

            check = 0;

            leave = rand()%(SETSIZE - 1) + 2;

            int temp2 = r[leave - 1][0];

            if (added[temp2-1] < 1) {

                  removed[temp2-1] = SETSIZE;

                  check++;
            }
            else if (added[temp2 - 1] >= 1) {
                  check2++;
            }
            if(check2 > (SETSIZE -1)) {
                  leave = rand()%(SETSIZE - 1) + 2;
                  check++;
                  check2 = 0;
            }
      }while (check < 1);

      cout<<"Original set r(i) /";

      for (i = 0; i < SETSIZE; i++) {
            cout<<r[i][0];

            if (i < (SETSIZE - 1)) {
            cout<<", ";
            }

      }

      cout<<"/;"<<endl;


            int flag3[NUMSTOPS] = {0};


//Select the four neighborhood solutions

            for (j = 1; j < ID; j++) {

      //a small do-loop to check that the entering node is not already in the
set r

            int totdist = 0;
```

```
for (i = 1; i < NUMSTOPS; i++) {
      for (int v = 1; v < SETSIZE; v++) {
            totdist = totdist + lambda[i][r[v][0]-1];
      }

      ratio1[i] = demand[i]/totdist;
      totdist = 0;
}

cout<<"ratio1 [";
for (i=0; i<NUMSTOPS; i++) {
      cout<<" "<<ratio1[i];
}

cout<<"]"<<endl;


do{

      redo = 0;

      int remainder = iter % 2;

      if (remainder > 0) {

            enter[j] = n[maxIndex(ratio1, NUMSTOPS, flag3)];

            int temp7 = maxIndex(ratio1, NUMSTOPS, flag3);

            flag3[temp7] = 1;

            temp4 = enter[j];

      }
```

```
                    else if (remainder == 0) {
                          enter[j] = rand()%(NUMSTOPS - 1) + 2;
                          temp4 = enter[j];
                    }

                    for (t = 1; t < ID; t++) {

                          for (i=0; i < SETSIZE; i++){
                                if (r[i][t] == temp4 || added[temp4-1] > 0 ||
removed[temp4-1] > 0 || infeasible[temp4-1] > 0) {
                                redo++;
                                }
                          }
                    }
              }while(redo > 0);

              cout<<"Leaving Stop Index: "<<leave<<" Entering Stop:
"<<enter[j]<<endl;

              r[leave-1][j] = enter[j];

//to ensure that neighborhoods are distince from each other

              cout<<"Neighborhood "<<j<<" set r(i) /";

              for (i = 0; i < SETSIZE; i++) {
                    cout<<r[i][j];

                    if (i < (SETSIZE - 1)) {
                    cout<<", ";
                    }
              }

              cout<<"/;"<<endl;

        }
```

```
for (j = 1; j < ID; j++) {

      for (i = 0; i < SETSIZE; i++) {
            b[i] = r[i][j];
      }

      qsort(b, SETSIZE, sizeof(int), compare);

      for (i = 0; i < SETSIZE; i++) {
                  r[i][j] = b[i];
      }

      cout<<"Neighborhood "<<j<<" Sorted set r(i) /";

      for (i = 0; i < SETSIZE; i++) {
            cout<<r[i][j];

            if (i < (SETSIZE - 1)) {
                  cout<<", ";
            }
      }

      cout<<"/;"<<endl;

}



for (j = 1; j < ID; j++) {

      ofstream file("combinations.inc", ios::out | ios::trunc);

      file<<"set r(i) /";

      for (i = 0; i < SETSIZE; i++) {
            file<<r[i][j];
```

```
                if (i < (SETSIZE - 1)) {
                file<<", ";
                }

        }

        file<<"/;"<<endl;

        system("/usr/local/gams/23.5.2/gams TabuGams lo=2");

        ifstream opt("optimal.txt");

        opt>>zstar[j];

        cout<<"Neighborhood "<<j<<" Iteration "<<iter<<" Optimal Solution: 
"<<zstar[j]<<endl;

        if (zstar[j] < 1) {

                infeasible[enter[j] - 1] = 2;

                zcount++;

                cout<<zcount<<endl;
        }

        if (zstar[j] > 0 && zstar[j] < BEST) {

                BEST = zstar[j];
                zoptimal[OUTCOUNT-2] = BEST;

                for (int q = 0; q < SETSIZE; q++) {

                        bestset[q][OUTCOUNT-1] = r[q][j];
                        carry[q][OUTCOUNT] = bestset[q][OUTCOUNT-1];
```

```
            roptimal[q][OUTCOUNT-2] = bestset[q][OUTCOUNT-1];
      }
}

if ((zstar[j] > 0) && (zstar[j] < zstar[0])) {

            zstar[0] = zstar[j];

            added[enter[j] - 1] = SETSIZE;

            zcount = 0;

            cout<<zcount<<endl;

            for (int p = 1; p < SETSIZE; p++) {
        r[p][0] = r[p][j];

            }

            cout<<"do I get here?"<<endl;
}

else if ((zstar[j] > 0) && (zstar[j] > zstar[0])) {

            zcount++;
            cout<<zcount<<endl;
            cout<<"do I get here?"<<endl;

      if (zcount > (SETSIZE+SETSIZE) ) {

            int temp5 = rand()%(ID-1) + 1;

            for (int o = 1; o < SETSIZE; o++) {
          r[o][0] = r[o][temp5];

            }
```

```
                    zcount = 0;
                }
            }

        }

            cout<<"Added[]: [";

          for (t = 0; t < NUMSTOPS; t++ ) {
                cout<<" "<<added[t];
            }

            cout<<"]"<<endl;

            cout<<"Removed[]: [";

          for (t = 0; t < NUMSTOPS; t++ ) {
                cout<<" "<<removed[t];
            }

            cout<<"]"<<endl;

            cout<<"Infeasible[]: [";

          for (t = 0; t < NUMSTOPS; t++ ) {
                cout<<" "<<infeasible[t];
            }

            cout<<"]"<<endl<<endl;


            for (t = 0; t < NUMSTOPS; t++ ) {

                if ( added[t] > 0 ){
                    added[t] = added[t] - 1;
                }

                if ( removed[t] > 0 ){
                      removed[t] = removed[t] - 1;
                }

                if ( infeasible[t] > 0 ){
                      infeasible[t] = infeasible[t] - 1;
                }
          }

          }
    }

    if (OUTCOUNT > 2) {

    while (iter < 10) {

          iter++;

          //Set the current set for all neighborhoods to the best from the
previous iteration

          for (i = 0; i < SETSIZE; i++){
                for (j = 1; j < ID; j++) {
                      r[i][j] = r[i][0];
                }
          }

          cout<<"Original set r(i) /";

        for (i = 0; i < SETSIZE; i++) {
                cout<<r[i][0];

                if (i < (SETSIZE - 1)) {
                      cout<<", ";
                }
            }

            cout<<"/;"<<endl;

            //Select an exchanging pair, that is a leaving node and an
entering node

          int totdist = 0;

            int * leaverow;
            leaverow = new int [ID];

            int * entercol;
            entercol = new int [ID];

          int temp, temp1, temp2, temp3;

          int tabu[NUMSTOPS][NUMSTOPS] = {0};

            for (i=0; i<SETSIZE; i++) {

                  for (j=1; j<NUMSTOPS; j++){

                        if (j == r[i][0] - 1){

                              for (o = 1; o < NUMSTOPS; o++) {

                                    for  (v = 1; v < SETSIZE; v++) {
                                          if ( j != r[v][0] - 1) {
                                                totdist = totdist +
lambda[o][r[v][0]-1];

                                          }
                                    }

                                    ratio[j][o] = demand[o]/totdist;
```

```
                                        totdist = 0;
                                    }
                              }
                        }
                  }

            cout<<"ratio {{";

            for ( i = 0; i < NUMSTOPS; i++) {

                  for (j = 0; j < NUMSTOPS; j++) {

                        cout<<ratio[i][j];

                      if (j < (NUMSTOPS - 1)){
                              cout<<",";
                        }
                  }

                  if (i < (NUMSTOPS - 1)) {

                        cout<<"},"<<endl<<endl;
                        cout<<"{";
                  }
            }

            cout<<"}};"<<endl;

            double tempratio[NUMSTOPS*NUMSTOPS] = {0};

            for (i = 0; i < NUMSTOPS; i++) {

                  for (o = 0; o < NUMSTOPS; o++ ) {

                        tempratio[i*NUMSTOPS + o] = ratio[i][o];
                  }
```

```
 }

 cout<< "tempratio/";

 for (int o = 0; o < NUMSTOPS*NUMSTOPS; o++) {
       cout<< tempratio[o];
       if ( o < (NUMSTOPS*NUMSTOPS -1)) {
            cout<<",";
      }
}
cout<<"/;"<<endl;


        int flag[NUMSTOPS*NUMSTOPS] = {0};

      //Select the neighborhood solutions

      for (j = 1; j < ID; j++) {

            do {
                  redo = 0 ;
                int remainder = iter%2;

                if (remainder > 0) {

                  temp = maxIndex(tempratio,NUMSTOPS*NUMSTOPS,flag);
                  double max = tempratio[temp];
                  cout<<"temp:"<<temp<<endl;
                  cout<<"max:"<< max<<endl;
                  flag[temp] = 1;

                  entercol[j] = (temp+1)%NUMSTOPS;
                  leaverow[j] = ((temp+1)-entercol[j])/NUMSTOPS + 1;
                  temp1 = leaverow[j];
                  temp2 = entercol[j];
        }

        if (remainder == 0) {

                temp3 = rand()%(SETSIZE - 1) + 2;
                leaverow[j] = r[temp3 - 1][0];
                entercol[j] = rand()%(NUMSTOPS - 1) + 2;
                temp1 = leaverow[j];
                temp2 = entercol[j];

        }


                for (i = 0; i < NUMSTOPS; i++){

                        for (v = 0; v < NUMSTOPS; v++){

                                if (tabu[temp1-1][temp2-1] > 1) {

                                        redo++;

                                }

                        }

                }

                for ( i = 0; i < SETSIZE; i++) {

                        if (r[i][0] == temp2) {
                                redo++;
                        }
                }
}while(redo > 0 );
```

```
            cout<<"leaving stop:"<< leaverow[j]<< "  Entering
Stop:"<<entercol[j]<<endl;

            for ( i = 0; i < SETSIZE; i++) {
                  if ( r[i][j] == leaverow[j] ) {
                        r[i][j] = entercol[j];
                  }
            }


            //to ensure that neighborhoods are distince from each other

            cout<<"Neighborhood "<<j<<" set r(i) /";

            for (i = 0; i < SETSIZE; i++) {
                  cout<<r[i][j];

                  if (i < (SETSIZE - 1)) {
                  cout<<", ";
                  }
            }

            cout<<"/;"<<endl;

      }

      for (j = 1; j < ID; j++) {

            for (i = 0; i < SETSIZE; i++) {
                  b[i] = r[i][j];
            }

            qsort(b, SETSIZE, sizeof(int), compare);

            for (i = 0; i < SETSIZE; i++) {
                        r[i][j] = b[i];
```

```
        }

        cout<<"Neighborhood "<<j<<" Sorted set r(i) /";

        for (i = 0; i < SETSIZE; i++) {
              cout<<r[i][j];

              if (i < (SETSIZE - 1)) {
                    cout<<", ";
              }
        }

        cout<<"/;"<<endl;

  }

  for ( i = 0; i < NUMSTOPS; i++) {

        for (j = 0; j< NUMSTOPS; j++) {

              if (tabu[i][j] > 0) {
                    tabu[i][j] = tabu[i][j] - 1;
              }
        }
  }




  for (j = 1; j < ID; j++) {

        ofstream file("combinations.inc", ios::out | ios::trunc);

        file<<"set r(i) /";

        for (i = 0; i < SETSIZE; i++) {
```

```
			file<<r[i][j];

			if (i < (SETSIZE - 1)) {
			file<<", ";
			}

		}

		file<<"/;"<<endl;

		system("/usr/local/gams/23.5.2/gams TabuGams lo=2");

		ifstream opt("optimal.txt");

		opt>>zstar[j];

		cout<<"Neighborhood "<<j<<" Iteration "<<iter<<" Optimal Solution: 
"<<zstar[j]<<endl;

		if (zstar[j] < 1) {

			tabu[leaverow[j]-1][entercol[j] - 1] = 2;

			zcount++;

			cout<<zcount<<endl;
		}

		if (zstar[j] > 0 && zstar[j] < BEST) {

			BEST = zstar[j];
			zoptimal[OUTCOUNT-2] = BEST;

			for (int q = 0; q < SETSIZE; q++) {

				bestset[q][OUTCOUNT-1] = r[q][j];
```

```
                carry[q][OUTCOUNT] = bestset[q][OUTCOUNT-1];
                roptimal[q][OUTCOUNT-2] = bestset[q][OUTCOUNT-1];
        }
}

if ((zstar[j] > 0) && (zstar[j] < zstar[0])) {

                zstar[0] = zstar[j];

                tabu[entercol[j]-1][leaverow[j] - 1] = SETSIZE + 1;

                zcount = 0;

                cout<<zcount<<endl;

                for (int p = 1; p < SETSIZE; p++) {
           r[p][0] = r[p][j];

                }

                cout<<"do I get here?"<<endl;
}

else if ((zstar[j] > 0) && (zstar[j] > zstar[0])) {

                zcount++;
                cout<<zcount<<endl;
                cout<<"do I get here?"<<endl;

        if (zcount > (SETSIZE+SETSIZE) ) {

                int temp5 = rand()%(ID-1) + 1;

                for (int o = 1; o < SETSIZE; o++) {
             r[o][0] = r[o][temp5];
```

```
                        }
                        tabu[entercol[temp5]-1][leaverow[temp5] - 1] = SETSIZE
- 1;

                        zcount = 0;
                    }
                }

            }
                cout<<"tabu {{";

              for ( i = 0; i < NUMSTOPS; i++) {

                    for (j = 0; j < NUMSTOPS; j++) {

                        cout<<tabu[i][j];

                      if (j < (NUMSTOPS - 1)){
                            cout<<",";
                        }
                    }

                  if (i < (NUMSTOPS - 1)) {
                        cout<<"},"<<endl<<endl;
                      cout<<"{";
                    }
                }

              cout<<"}};"<<endl;

            }
            }

            cout<<"Best Solution: "<<BEST<<endl;
            cout<<"Best Solution Stop Set: ";

            cout<<"r(i) /";
            for (int v = 0; v < SETSIZE; v++) {
                  cout<<bestset[v][OUTCOUNT-1];

                  if (v < (SETSIZE - 1)) {
                  cout<<", ";
                  }
            }

            cout<<"/;"<<endl<<endl;

            cout<<OUTCOUNT<<"YUYAO"<<endl;

            cout<<zoptimal[OUTCOUNT-2]<<endl;


}

    cout<<"YUYAO";
    cout<<"zoptimal(6) /";
    for (v = 0; v < 6; v++) {
            cout<<zoptimal[v];

            if (v < 5) {
                  cout<<",";
            }
      }
      cout<<"/;"<<endl<<endl;


      int mark = 0;
      int mark1[6] = {0};

    mark = minIndex(zoptimal, 6, mark1);

      cout<<mark<<endl<<endl;
```

```
cout<<"Final Best Solution:"<<zoptimal[mark]<<endl;
cout<<"Final Best Solution Stop Set:";

      cout<<" /";

      for (v = 0; v < NUMSTOPS; v++) {
            cout<<roptimal[v][mark];

            if (v < (NUMSTOPS - 1)) {
            cout<<", ";
            }
      }

cout<<"/;"<<endl<<endl;

int mark2=0;

for(i = 0; i < NUMSTOPS; i++) {
      if(roptimal[i][mark] > 0) {mark2++;}
}

int * routput;
routput = new int [mark2];

for(i = 0; i < mark2; i++) {
      routput[i] = roptimal[i][mark];
}

ofstream file("combinations.inc", ios::out | ios::trunc);

file<<"set r(i) /";

for (i = 0; i < mark2; i++) {
      file<<routput[i];

      if (i < (mark2 - 1)) {
            file<<", ";
            }

      }

      file<<"/;"<<endl;

      system("/usr/local/gams/23.5.2/gams xfix lo=2");
      system("/usr/local/gams/23.5.2/gams evaluation lo=2");


      t2 = time (NULL);
      double diff = t2 - t1;
      cout<<"Computation Time: "<<diff<<endl;

      return(0);

}



//=============================================== maxIndex
// From algorithms/arrayfuncs.cpp
// Returns the index of the maximum value in an array.
int maxIndex(double a[], int size, int check[]) {

    int maxIndex = 0;
    for (z=1; z<size; z++) {
        if (a[z] > a[maxIndex] && check[z] < 1) {
            maxIndex = z;
        }
      }
    return maxIndex;
}//end maxIndex

//=============================================== minIndex
```

```
// From algorithms/arrayfuncs.cpp
// Returns the index of the minimum value in an array.
int minIndex(double a[], int size, int check[]) {

    int minIndex = 0;
    for (z=1; z<size; z++) {
        if (a[z] < a[minIndex] && check[z] < 1) {
            minIndex = z;
        }
      }
    return minIndex;
}//end minIndex
```

# REFERENCES

Adamski, A. (1992). Probabilistic Models of Passengers Service Processes at Bus Stops. Transportation Research Part B, , 253-259.

Andersson, P., & Scalia-Tomba, G. (1981). A Mathematical Model of an Urban Bust Route. Transportation Research Part B: Methodological , 249-266.

Baaj, M., & Mahmassani, H. (1990). TRUST: A Lisp Program for the Analysis of Transit Route Configuration. Transportation Research Record , 125-135.

Barnes, J., Laguna, M., & Glover, F. (1995). An Overview of Tabu Search Approaches to Production Scheduling Problems. Intelligent Scheduling Systems , 101-127.

Battiti, R. (1994). Simulated Annealing and Tabu Search in the long run: A Comparison on QAP Tasks. Computers & Mathematics with Applications , 1-8.

Ceder, A. (1986). Methods for Creating Bus Timetables. Transportation Research Part A , 59-83.

Ceder, A. (2001). Operational Objective Functions in Designing Public Transport Routes. Journal of Advanc Transportation , 125-144.

Ceder, A. (1991). Transit Scheduling. Journal of Advanced Transportation , 137-160.

Ceder, A., & Israeli, Y. (1998). User and Operator Perspectives in Transit Network Design. Transportation Research Record: Journal of the Transportation Research Board , 3-7.

Ceder, A., & Wilson, N. (1986). Bus Network Design. Transportation Research Part B , 20B, 331-344.

Chakroborty, P. (2003). Genetic Algorithms for Optimal Urban Transit Network Design. Computer-Aided Civil and Infrastructure Engineering , 184-200.

Chang, S. a. (1991b). Optimization Models for Comparing Conventional and Subscription Bus Feeder Services. Transportation Science , 281-298.

Chang, S., & Schonfeld, P. (1991). Multiple Period Optimization of Bus Transit Systems. Transportation Research Part B , 25B, 453-478.

Chien, S., & Schonfeld, P. (1998). Joint Optimization of a Rail Transit Line and its Feeder Bus System. Journal of Advanced Transportation , 32, 253-284.

Chien, S., & Schonfeld, P. (1997). Optimization of Urban Grid Transit System in Heterogeneous Urban Environmental. Journal of Transportation Engineeing , 28-35.

Chien, S., & Yang, Z. (2000). Optimal Feeder Bus Routes with Irregular Street Networks. Journal of Advanced Transportation , 213-248.

Chien, S., Dimitrijevic, B., & Spasovic, L. (2001). Bus Route Planning in Urban Grid Commuter Networks. Transportation Research Board 80th Annual Meeting. Washington, D.C.

Corberan, A., Fernandez, E., Lagunay, M., & Marti, R. (2000). Heuristic Solutions to the Problem of Routing School Buses with Multiple Objectives. Journal of the Operational Research Society , 427-435.

Crainic, T., Malucelli, F., Nonato, M., & Guertin, F. (2005). Meta-Heuristics for a Class of Demand-Responsive Transit Systems. . INFORMS Journal on Computing , 10-24.

Dial, R. (1967). Transit Pathfinder Algorithm. Highway Research Record , 67-85.

Ding, Y., & Chien, S. (2001). Improving Transit Service Quality and Headway Regularity with Real-time Control. Transportation Research Board. Washington, D.C.

Dubois, D., Bel, G., & Llibre, M. (1979). A Set of Methods in Transportation Synthesis and Analysis. The Journal of the Operational Research Society , 30, 797-808.

Dufourd, H., Gendreau, M., & Laporte, G. (1996). Locating a Transit Line Using Tabu Search. Location Science , 1-19.

ECONorthwest and Parsons Brinckerhoff Quade & Douglas, I. (2002). Estimating the Benefits and Costs of Public Transit Projects: A Guidebook for Practitioners. Washington, D.C.: Transportation Research Board, National Research Council.

Fan, W. (2004). Optimal Transit Route Network Design Problem: Algorithms, Implementations, and Numerical Results. Doctoral Report, Department of Civil Engineering,The University of Texas at Austin, Austin, Texas .

Fan, W., & Machemehl, R. B. (2008). Tabu Search Strategies for the Public Transportation Network Optimizations with Variable Transit Demand. Computer-Aided Civil and Infrastructure Engineering , 502-520.

Fan, W., & Machemehl, R. B. (2006). Using a Simulated Annealing Algorithm to Solve the Transit Route Network Design Problem . Journal of Transportation Engineeing , 122-132.

FTA. (2011). Annual Report on Funding Recommendations Fiscal Year 2012. http://www.fta.dot.gov/documents/Annual_Report_main_text_FINAL_2_11_11%281%29.pdf.

Glover, F. (1986). Future Paths for Integer Programming and Links to Artificial Intelligence. Computers and Operations Research , 533-549.

Glover, F. (1977). Heuristics for Integer Programming Using Surrogate Constraints. Decision Sciences , 8, 156-166.

Glover, F. (1997). Tabu Search and Adaptive Memory Programming - Advances, Applications and Challenges. INTERFACES IN COMPUTER SCIENCE AND OPERATIONS RESEARCH , 1-75.

Glover, F. (1989). Tabu search-part I. ORSA Journal on Computing , 190-206.

Glover, F. (1990). Tabu search-part II. ORSA Journal on Computing , 4-32.

Glover, F., & Laguna, M. (1997). Tabu Search. Boston: Kluwer Academic Publishers.

Goczyla, K., & Cielatkowski, J. (1995). Optimal Routing in a Transportation Network. European Journal of Operational Research , 214-222.

Grava, S. (2003). Urban Transportation Systems: Choices for Communities. McGraw-Hill.

Han, A., & Wilson, N. (1993). Efficient Search Algorithms for Route Information Services of Direct and Connecting Transit Trips. Transportation Research Record , 1-5.

Henk, R., & Hubbard, S. (1996). Developing an Index of Transit Service Availability. Transportation Research Record , 12-19.

Hennan, B., & Ardekani, S. (1984). Characterizing Traffic Conditions in Urban Areas. Transportation Science , 101-140.

Hickman, M., & N.H.M., W. (1995). Passenger Travel Time and Path Choice Implications of Real-time Transit Information. Transportation Research Part C: Emerging Technologies , 211-226.

Hsu, J., & Surti, V. (1975). Framework of Route-Selection in Bus Network Design. Transportation Research Record , 44-57.

Jerby, S., & Cedar, A. (2006). Optimal Routing Design for Shuttle Bus Service. Transportation Research Record: Journal of the Transportation Research Board , 14-22.

Kuah, G., & Perl, J. (1987). A Methodology for Feeder-Bus Network Design. Transportation Research Record: Journal of the Transportation Research Board , 40-51.

Kuah, G., & Perl, J. (1988). Optimization of Feeder Bus Routes and Bus-stop Spacing. Journal of Transportation Engineering , 114.

Laguna, M., Barnes, J., & Glover, F. (1993). Intelligent Scheduling with Tabu Search: An application to Jobs with Linear Delay Penalties and Sequence-Dependent Setup Costs and Times. Journal of Applied Intelligence , 159-172.

Laguna, M., Barnes, W., & Glover, F. (1991). Tabu Search Methods for a Single Machine Scheduling Problem. Journal of Intelligent Manufacturing , 63-74.

Lampkin, W., & P.D., S. (1967). The Design of Routes, Service Frequencies and Schedules for a Municipal Bus Undertaking: A Case Study. Operation Research Ouarterly , 375-397.

Leblanc, L. (1988). Transit System Network Design. Transportation Research Part (22), 383-390.

Lee, Y.-J., & Vuchic, V. (2005). Transit Network Design with Variable Demand. Journal of Transportation Engineering, ASCE , 131.

Li, L., & Fu, Z. (2002). The School Bus Routing Problem: A Case Study. Journal of the Operational Research Society , 552-558.

List, G. (1990). Toward Optimal Sketch-Level Transit Service Plans. Transportation Research Part B , 325-344.

Lownes, N. (2007). The Commuter Rail Circulator Network Design Problem:Formulation, Solution Methods, and Applications. Doctoral Report, Department of Civil Engineering,The University of Texas at Austin, Austin, Texas .

Lownes, N., & Machemehl, R. (2010). Exact and heuristic methods for public transit circulator design. Transportation Research Part B , 309-318.

Luenberger, D. (1984). Linear and Nonlinear Programming, Second Edition. Addison-Wesley Publishers.

Martins, C. L., & Pato, M. V. (1998). Search Strategies for the Feeder Bus Network Design Problem. European Journal of Operational Research , 425-440.

Martins, C., & Pato, M. (1998). Search Strategies for the Feeder Bus Network Design Problem. European Journal of Operational Research , 425-440.

Moellering, H., Gauthier, H., & Osleeb, J. (1977). An interactive Graphic Transit Planning System Based on Individuals. Urban Systems , 2, 93-103.

Newell, G. (1979). Some Issues Relating to the Optimal Design of Bus Routes. Transportation Science , 13, 20-35.

Oldfield, R., & Bly, P. (1988). An Analytic Investigation of Optimal Bus Size. Transportation Research Part B , 319-337.

Osuna, E., & Newell, G. (1972). Control Strategies for an Idealized Public Transportation System. Transportation Science , 57-72.

Pattnaik, S., Mohan, S., & Tom, V. (1998). Urban Bus Transit Route Network Design Using Genetic Algorithm. Journal of Transportation Engineering , 124, 368-375.

Raza, S. A., & AlTurki, U. (2007). A Comparative Study of Heuristic Algorithms to Solve Maintenance Schedulling Problem. Journal of Quality in Maintenance Engineering , 398-410.

Repoussis, P., & Tarantilis, C. (2010). Solving the Fleet Size and Mix Vehicle Routing Problem with Time Windows via Adaptive Memory Programming. Transportation Research Part C , 695-712.

Schrank, D., & Lomax, T. (2011). The 2011 Urban Mobility Report. College Station, Texas: Texas Transportation Institute, Texas A&M University.

Spasovic, L., & Schonfeld, P. (1993). Method for Optimizing Transit Service Coverage. Transportation Research Record , 28-39.

Stein, D. M. (1978). An Asymptotic, Probabilistic Analysis of a Routing Problem. Mathematics of Operations Research , 3, 89-101.

Tom, V. M., & Mohan, S. (2003). Transit Route Network Design Using Frequency Coded Genetic Algorithm. Journal of Transportation Engineering , 186-195.

Van Nes, R., & Bovy, P. (2000). Importance of Objectives in Urban Transit Network Design. Transportation Research Record , 25-34.

Van Nes, R., Hamerslag, R., & Immer, B. (1988). The Design of Public Transport Networks. Transportation Research Record , 74-83.

Victor, D., & Santhakumar, S. (1986). Simulation study of bus transit. Journal of Transportation Engineering , 199-211.

Willoughby, K. (2002). A Mathematical Programming Analysis of Public Transit Systems. Omega , 137-142.

Wilson, N., & Gonzalez, S. (1982). Methods for Service Design. Transportation Research Record , 74-83.