Diploma Thesis

Public transit network optimization via a novel connectivity parameter

submitted in satisfaction of the requirements for the degree

Master of Science

in Computational Science and Engineering. Presented to TU Wien, Faculty of Civil and Environmental Engineering

---

Diplomarbeit

Optimierung des öffentlichen Nahverkehrsnetzes über einen neuartigen Konnektivitätsparameter

ausgeführt zum Zwecke der Erlangung des akademischen Grads

Master of Science

in CSE. Präsentiert an TU Wien, Fakultät für Bau- und Umweltingenieurwesen

James Harrison Mulhern, BSc

Matr.Nr.: 12112942

Betreuung:	Sr Scientist Dr.techn. **Takeru Shibayama** Assistant Prof. PhD **Bingyu Zhao** Institut für Verkehrswissenschaften Forschungsbereich Verkehrsplanung und Verkehrstechnik Technische Universität Wien Karlsplatz 13/230-1, 1040 Wien, Österreich
Co-Betreuung:	Ao.Univ.Prof. Dipl.-Ing. Dr.techn. **Günther Raidl** Institut für Logic and Computation Forschungsbereich Algorithms and Complexity Technische Universität Wien Favoritenstraße 9, 1040 Wien, Österreich

Wien, im August 2025 _____

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Statement

I hereby declare that I have written this thesis independently, that I have completely specified the utilized sources and resources and that I have definitely marked all parts of the work - including tables, maps and figures - which belong to other works or to the internet, literally or extracted, by referencing the source as borrowed.

I further declare that I have used generative AI tools only as an aid, and that my own intellectual and creative efforts predominate in this work. In the appendix "Overview of Generative AI Tools Used" I have listed all generative AI tools that were used in the creation of this work, and indicated where in the work they were used. If whole passages of text were used without substantial changes, I have indicated the input (prompts) I formulated and the IT application used with its product name and version number/date.

Wien, im August 2025 _____

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Abstract

The public transportation options in suburban North America are typically focused on commuting based routes toward a cities central business district. Routes often pass though suburban towns without connecting to points of interest or population centers. Additionally, North American suburban transportation routes are lacking first mile options with minimal local bus or rail services. This thesis tackles the local transit issue by developing a mathematical framework for the utility of each point in the region, this utility is then aggregated into a connectivity metric. The connectivity metric improves upon standard public transit accessibility metrics by incorporating internal network distances.

The connectivity metric was applied to real world problems by utilizing it as the objective value for the generation and optimization of a new local bus route. The mathematical framework for the utility density can be adapted for any set of spatial data to be used as an input. For this thesis, the population density and points of interests were combined using travel survey data to define the utility density. The utility density value represents general transit attractiveness for all trip types. The connectivity metric was then computed for the region by computing the shortest path between each pair of points and scaling the travel time with the destination utility.

With a well-defined connectivity metric, the town of Winchester, Massachusetts was used as a case study for a route optimization task. The utility density was computed using the mathematical framework and data from public data sources. The route optimization was completed using the generalized variable neighborhood search meta-heuristic. The optimization process returned a route that forms a distinct flower pedal shape centered at the Winchester Center regional rail station and the pedals reach out to different areas of the town. The optimal route was computed thirty times with an average connectivity improvement of 16.0%. The proposed route layout, effectively connects the town based on the optimization parameters and offers useful first mile options due to connections to all existing though town public transit lines.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Abstract

Die öffentlichen Verkehrsmittel in nordamerikanischen Vororten konzentrieren sich typischerweise auf Pendlerstrecken in Richtung der zentralen Geschäftsviertel. Die Strecken führen oft durch Vororte, ohne Anbindung an Sehenswürdigkeiten oder Ballungszentren zu bieten. Zudem mangelt es nordamerikanischen Vorortverkehrsrouten an Optionen für die erste Meile und das Angebot an lokalen Bus- und Bahnverbindungen ist minimal. Diese Arbeit befasst sich mit der Frage des Nahverkehrs, indem sie einen mathematischen Rahmen für die Nutzbarkeit jedes Punktes in der Region entwickelt und diesen Nutzen anschließend zu einer Konnektivitätsmetrik aggregiert. Die Konnektivitätsmetrik verbessert die Standardmetriken für die Erreichbarkeit des öffentlichen Nahverkehrs durch die Berücksichtigung interner Netzwerkdistanzen.

Die Konnektivitätsmetrik wurde auf reale Probleme angewendet, indem sie als Zielwert für die Erstellung und Optimierung einer neuen lokalen Buslinie verwendet wurde. Der mathematische Rahmen für die Nutzdichte kann für beliebige räumliche Daten als Eingabe angepasst werden. Für diese Arbeit wurden Bevölkerungsdichte und Sehenswürdigkeiten anhand von Verkehrserhebungsdaten kombiniert, um die Nutzdichte zu definieren. Der Nutzdichtwert repräsentiert die allgemeine Attraktivität des Nahverkehrs für alle Fahrtarten. Die Konnektivitätsmetrik für die Region wurde berechnet, indem der kürzeste Weg zwischen jedem Punktpaar ermittelt und die Reisezeit mit dem Zielnutzen skaliert wurde.

Mit einer klar definierten Konnektivitätsmetrik diente die Stadt Winchester, Massachusetts, als Fallstudie für eine Routenoptimierung. Die Nutzdichte wurde mithilfe des mathematischen Rahmens und Daten aus öffentlichen Datenquellen berechnet. Die Routenoptimierung erfolgte mithilfe der Metaheuristik „Verallgemeinerte variable Nachbarschaftssuche“. Der Optimierungsprozess ergab eine Route, die eine ausgeprägte Blütenblattform mit Mittelpunkt am Regionalbahnhof Winchester Center bildet und von der aus die Blütenblätter verschiedene Stadtteile erreichen. Die optimale Route wurde dreißig Mal berechnet, wobei sich die Konnektivität um durchschnittlich 16,0% verbesserte. Die vorgeschlagene Routenführung verbindet die Stadt effektiv auf Grundlage der Optimierungsparameter und bietet dank der Anbindung an alle bestehenden öffentlichen Verkehrsmittel im Stadtgebiet nützliche Optionen für die erste Meile.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Contents

1 Introduction **1**

2 Literature Review **5**
2.1 Accessibility Metrics . . . 6
2.1.1 Potential Model . . . 6
2.1.2 ÖV-Güteklassen . . . 7
2.1.3 London Public Transit Accessibility Level . . . 8
2.1.4 Composite Accessibility Index . . . 9
2.2 Route Optimization . . . 10
2.2.1 Optimization Problems . . . 10
2.2.2 Timetable Optimization . . . 11
2.2.3 Stop Optimization . . . 11
2.2.4 Route Optimization Example . . . 12
2.3 Relationship to Current Work . . . 13

3 Mathematical Model **15**
3.1 Connectivity Metric . . . 15
3.2 Utility Computation . . . 16
3.2.1 Utility Components . . . 17
3.2.2 Mapping to Discrete Domain . . . 20
3.3 External Utility . . . 21
3.4 Route Comparison . . . 22

4 Instance Generation **25**
4.1 Network and Utility Modeling . . . 25
4.1.1 Network Model . . . 26
4.1.2 Utility . . . 29
4.2 Case Study Selection . . . 30
4.3 Graph Generation . . . 33
4.3.1 OSM Graphs . . . 33
4.3.2 Transit Graphs . . . 35
4.4 Utility Implementation . . . 38
4.5 Instance Implementation . . . 42
4.5.1 Walking and Road Graphs . . . 42
4.5.2 Transit Graph Generation . . . 42
4.5.3 Additional Data Preparation . . . 43
4.5.4 Optimization Graph . . . 44

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

4.6 Scenarios and Case Study 45

5 Optimization **47**
5.1 Heuristics Background 47
5.2 GVNS Details 48
5.2.1 Construction Heuristics 50
5.2.2 Neighborhood Selection 52
5.3 Implementation Details 53
5.3.1 Instance Definition 54
5.3.2 Instance Data structure 54
5.3.3 Solution Data Structure 56
5.3.4 Local Search Framework and Improvement Heuristics 57
5.3.5 Solution Validity and Shaping 60
5.3.6 GVNS Setup 61
5.3.7 Logging Methods 61
5.4 Objective Computation 62
5.4.1 Standard Algorithm 62
5.4.2 Delta Evaluation Algorithm 64
5.4.3 Computational Performance Improvements 64
5.4.4 GPU Version 67

6 Results **71**
6.1 Scenarios and Case Study 72
6.1.1 Case Study Results 72
6.1.2 Scenario Results 74
6.2 Hyperparameter Tuning 77
6.2.1 Number of Processors 78
6.2.2 Total Iterations 78
6.2.3 Limiting Connections 79
6.2.4 Penalty Terms 80
6.2.5 Waiting Time 84
6.2.6 Number of Stops 84

7 Discussion **89**
7.1 Proposed Optimal Transit Route 89
7.2 Connectivity Metric and Optimization Implementation 90
7.2.1 Connectivity Metric and Utility 90
7.2.2 Instance Implementation 91
7.2.3 Optimization Implementation 91
7.3 Supplementary Results 92
7.3.1 Shape of Route 92
7.3.2 Effectiveness of Penalty Terms 94
7.3.3 Impact of Number of Stops 95

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

7.4 Applicability and Shortcoming of Analysis 95
7.4.1 Applicability to Other Transit Modes 95
7.4.2 Applicability to Other Regions 96
7.4.3 Modification of Utility Values 98
7.5 Future Work 99
7.5.1 Modifications of Existing Routes 99
7.5.2 Change of Solution Limits 100
7.5.3 Future Analysis Tasks 100

8 Conclusion **103**

Bibliography **105**

A Reference Information **109**
A.1 Overview of Generative AI Tools Used 109
A.2 Source Code 109
A.3 OSM Graph Filters 109
A.4 OSM Mapping 110

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

List of Figures

2.1 ÖV-Güteklassen for Winchester, MA. Map generated with QGIS with data from [28]. 8

3.1 Example of computation errors in the path offset 23

4.1 Example intersection with and without unique vertexes 28
4.2 Diagram showing the different network layers. 29
4.3 Zoning and public transit in Winchester, MA. Photos by author. 32
4.4 Simplified OSMnx graphs and non-simplified graphs 34
4.5 The dependency graph used with the partridge python package [31]. . . . 36
4.6 Overview of the external regions. 41
4.7 Two photos of the construction on Winchester Center Station. Photos taken by author, August 2024 43

6.1 Resulting average route for the case study 73
6.2 Manual planned transit route for case study 74
6.3 Resulting average route for Scenario 1 75
6.4 Resulting average route for Scenario 2 76
6.5 Resulting average route for Scenario 3 77
6.6 Objective improvement against iteration value 80
6.7 Change of objective improvement with increasing k value 81
6.8 Summary results for the distance penalty analysis 82
6.9 Summary results for the stop penalty analysis 83
6.10 Waiting time results with repeated vertexes allowed 86
6.11 Waiting time results without repeated vertexes 87
6.12 Results form the parametric study into the number of stops 88

7.1 An example transit route with an internal transfer 93

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

List of Tables

2.1 Tables used to compute the Public Transit Quality Classes (ÖV-Güteklassen), reproduced in English from [33] 7

4.1 Travel habits data from the NHTS 39
4.2 Subset of the mappings from OSM attribute values to trip categories . . . 40
4.3 External region points 41

5.1 Different neighborhoods used as part of the heuristic optimization 53
5.2 All parameters stored in the instance data structure. 55
5.3 All parameters stored in the optimization data structure. 56
5.4 Objective computation benchmarking results. 66

6.1 Computational efficiency of different processor counts 79

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Table of Abbreviations

Abbreviation	Expanded Name
ANOVA	Analysis of Variance
API	Application Programming Interface
CAI	Composite Accessibility Index
CPTAI	Comprehensive Public Transport Accessibility Index
CPU	Central Processing Unit
CSV	Comma Separated Value
EDF	Estimated Doorstep Frequency
EPSG	European Petroleum Survey Group
GIS	Geographic Information System
GPU	Graphics Processing Unit
GTFS	General Transit Feed Specification Format
GVNS	Generalized Variable Neighborhood Search
JSON	JavaScript Object Notation
MassGIS	Massachusetts Office of Geographic and Environmental Information
MB	Mesh Block
MBTA	Massachusetts Bay Transportation Authority
NGAI	Network Geospatial-Based Accessibility Index
NHTS	National Household Travel Survey
OSM	OpenStreetMap
POI	Point of Interest
PSO	Particle Swam Optimization
PTAL	Public Transit Accessibility Level
PTCI	Public Transport Coverage Index
SAP	Service Access Points
SIMD	Single Instruction Multiple Data
SUMO	Simulation of Urban MObility
TfL	Transport for London
TNDFSP	Transit Network Design and Frequency Setting Problem
TSNI	Transit Service Need Index
TSP	Traveling Salesman Problem
UTRP	Urban Transit Routing Problem
VND	Variable Neighborhood Decent

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Chapter 1

Introduction

Public transportation or transit is a network of vehicles that provide a transportation service to the public. Public transit exists in many forms across all regions of the world, and each transit network is tailored to the need of its riders. The transit networks in the United States are often built to service only the demands of commuters to the central business district(CBD) of a metro area. These networks focus on radial routes and typically have a hub-and-spoke network design. This design provides effective service for rides with destinations in the CBD, however trips not to the center of the network are typically longer than optimal. In a hub-and-spoke style network, trips between the tips of the spokes require traveling into the CBD and then out again. An example of this issue is the Greater Boston Metro Area serviced by the Massachusetts Bay Transit Authority(MBTA) In this network there are no circular rail services and fewer than five circular bus routes outside the downtown area. This is a notable difference between the MBTA and European transit networks. The lack of circular lines makes trips not to the CBD ineffective with public transit because of the poor time effectiveness relative to a car trip.

To encourage public transit use for trips not to the CBD, circular lines would need to be added to hub-and-spoke networks. Developing circular routes is difficult, and the best solution is not necessarily clear. There are multiple different paths that could be chosen for the circular route, as well as questions relating to where radially these routes should be placed. A transit planner must evaluate all the potential routes as well as consider existing infrastructure, expected ridership, budget and local knowledge. The scope of this task is large and as a result no manual planning process will produce an ideal set of new routes.

When considering an ideal public transit network, there would be some set of optimal routes to add to the network. These routes would provide the best public transit service based on a defined set of constraints. Finding these routes would be impossible using a manual planning method, however algorithmic techniques could be used to search for the optimal public transit routes. This leads to the fundamental research question of this thesis. Can new public transit routes be planned using optimization algorithms? To answer that question, this thesis works to develop a rigorous framework to define the effectiveness of the public transit system and then implement an optimization task with case study.

Defining the effectiveness of the public transit network is highly subjective and based on what effect the network is trying to achieve. Previous authors have defined many

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

metrics to evaluate the accessibility of different networks [32] [33] [34] [30], however the effectiveness of the network is not directly addressed. The existing measures typically focus on users access time to transit and the frequency of the provided service, but do not consider the shape and layout of the network itself. By omitting the impact of the network layout, there is potential for regions to have good transit accessibility but low transit usefulness. This is exemplified with hub-and-spoke networks where the spokes provide high levels of accessibility but low usefulness for trips not to the hub. To overcome the issues with existing metrics, this thesis models the effectiveness of a transit system by its ability to connect people to useful destinations and the resulting parameter is called connectivity.

When defining the connectivity of a region, the objective was to create a single value that is based on the travel time between every pair of points in the region and the usefulness of each destination. The travel times can be calculated using OpenStreetMap data and existing shortest path algorithms, however the usefulness of destinations needs to be defined with a custom algorithm. The algorithms combined public datasets such as population data, points of interest(POI) and travel habits into a single utility distribution for the region. The framework created for defining the utility distribution provides a high level of flexibility enabling users to tailor the utility value for a particular optimization objective. For this thesis, the goal was to find a transit route that was optimally useful for all trips. To do so, public data sources were used to define a utility distribution that represents the average trip for the average person.

The inspiration for this thesis was to search for new circular routes in a hub and spoke network. These routes would connect the spokes of the network in an optimal way. Unfortunately the resulting optimization problem for circular routes was outside the scope of the thesis and too intensive for the available computing resources. To simplify the analysis, the town of Winchester, Massachusetts was selected as a case study and the circular routes were replaced with a local route. Winchester was selected as a case study because despite the three regional transit routes passing though the town, there is no local transit option. Additionally, review of passenger data for the regional rail stations shows that up to 60% of rail users travel the first mile with a car, this highlights the need for first mile transit options in the town. The goal of the case study optimization was to find a single bus route that would maximize the connectivity of the town as well as offer a compelling first mile option for commuters.

To execute the optimization, an instance was created using public data sources available for the case study area. The instance generation process involved extracting road networks and POI from OpenStreeMap, as well as travel habits from the National Household Travel Survey (NHTS) and population data from Oak Ridge Nation Labs. Existing transit routes and timetables were sourced from the General Transit Feed Specification (GTFS) data for the MBTA. These data sources were combined to define the utility distribution, compute the connectivity metric and generate the network used for the optimization. With the instance defined, a meta-heuristics based optimization could be applied. Generalized Variable Neighborhood Search(GVNS) was selected as the meta-heuristic for this problem and the connectivity metric was used as the objective value.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

By implementing GVNS, a compelling new bus route was identified. The proposed route increased the connectivity of the town by 16%, as well as providing connections to the town center, highly populated areas and existing transit routes. These results show that the connectivity metric defined in this thesis has potential as an optimization tool for route planning tasks. Additionally, the optimization process is effective at finding local first miles routes. In the case study region this could reduce the 60% of transit riders that utilize a car to access the regional rail station. Not only does the route connect to both regional rail stations, the optimized route offers connection to both through town bus routes as well.

Upcoming in Chapter 2 of this thesis, the existing accessibility metrics are discussed in detail, as well as previous work relating to public transit optimization problems. In Chapter 3, the mathematical foundations of the thesis are presented. This covers the connectivity metric, the formal definition of the utility density and a method to analytically compare two solutions. Next in Chapter 4, the process of developing the optimization instance will be discussed. This chapter covers the custom code to create the graphs that define the existing public transit and walking networks as well as how potential bus stops are defined for the proposed route. The optimization process itself is defined in Chapter 5. This chapter focus on the mechanisms of the heuristic optimization as well as algorithmic and progaming improvements made to speed up the computation process. Finally, the results are presented in Chapter 6, as well as discussion and conclusions in Chapter 7 and 8 respectively.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Chapter 2

Literature Review

This thesis is focused on the implementation of a route optimization task with a novel connectivity metric as the objective value. The connectivity metric used in this thesis is built upon the existing public transportation accessibility metrics that are well established in the literature. Many authors have defined effective accessibility metrics, and public transit operators frequently use accessibility metrics. These metrics are used to evaluate the quality of service and visualize transit improvements over time. Despite the diverse use of transit accessibility metrics these metrics are not commonly used as an objective parameter in optimization tasks.

Transit related optimization efforts have little consistency in the topics, methods or objectives used. The optimization topics range from pure mathematical problems, to timetable and stop optimizations. To solve these problems, authors have employed a huge range of different optimization tools. Exact methods, such as Mixed Integer Linear Programming, have been implemented however the majority of the literature employs meta-heuristic algorithms. Common methods include genetic algorithms, ant-colony optimization and partial swarm optimizations. Each optimization algorithm requires an objective value with the most common goal being the minimization of total costs. Cost optimizations are often done as part of a multi-objective optimization because the cost must be balanced with quality of service requirements. A less common objective is to improve the accessibility of the transit system. One author that has utilized transit accessibility as part of the optimization objective is Yoo et al. with their 2023 paper [42].

In this paper the authors executed a route optimization task that modified the transit network to improve the access for the transit disadvantaged. To achieve this the authors modeled the accessibility of bus stops and the transit need for each section of the region. They then used reinforcement learning to modify the existing transit routes in the region. The work done by Yoo et al. is similar to the effort done by this thesis, however this analysis makes numerous improvements.

The optimization task presented in this thesis, builds upon the work done by Yoo et al. by employing a more detailed model for the objective value. The parameter used is referred to as the connectivity of the region. The connectivity value expands upon existing accessibility metrics by incorporating the impact of the entire transit network into the parameter. Additionally, the current work defines a mathematically robust framework for defining the utility of each point in the region. The utility value is combined with the travel times to define the connectivity of the region.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

2.1 Accessibility Metrics

There is no formally defined definition for transit accessibility nor is there a standardized method for computing a transit accessibility value. In general, transit accessibility is used as a measure of users willingness to utilize the transit network. Factors such as walking distance, service frequency, vehicle type, crowdedness and more can have an impact on the accessibility value. One of the simplest accessibility measures is the population potential. This metric evaluates accessibility based on the spatial proximity between sources and destinations. The simplicity of the population potential model limits the effectiveness and as a result municipalities have developed their own systems to monitor the transit accessibility of their network. Two such methods are the Öffentliche Verkehrsmittel Güteklassen used in Austria and the London Public Transit Accessibility Level used by Transport for London. Both of these methods incorporate spatial and service frequency into the ranking system however they do not consider capacity or other factors. In an attempt to correct the shortcomings of existing accessibility metrics, a composite index was developed by Rathod et al. [30]. In their paper the authors undertook a survey of the state of the art accessibility metrics and created a composite method using three recent methods.

2.1.1 Potential Model

One of the oldest accessibility metrics is the population potential model. This metric has its origin in the analysis of industrial facility locations [32]. The goal of the metric is to find the factory location that is most efficiency located with respect to the potential customers. This is done by first identifying a set of destinations D and then for each destination j an attractiveness or "socio-economic mass" M_j is defined. This can be a population value or another factor. For a particular source i the population potential PP_i is computed by summing the attractiveness of each destination divided by the distance to that destination d_{ij}. This results in Equation 2.1

$$PP_i = \sum_{j \in D} \frac{M_j}{d_{i,j}^a}. \tag{2.1}$$

The population potential has been widely used in literature and continues to be adapted and improved. In a 2023 paper by Wang et al. [37], the potential model is improved by considering factors in addition to the destination attractiveness and distance from the source or travel impedance. The authors add terms to Equation 2.1 to model the impact of compounding factors such as, station service capacity, walking distance to the station and the number of bus lines at the station. The authors were able to use this improved metric with a genetic algorithm to improve the average accessibility of Dalian, China.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

2.1.2 ÖV-Güteklassen

The Austrian Spacial Planning Conference (Österreichische Raumordnungskonferenz, ÖROK) defines a metric called Public Transit Quality Classes(Öffentliche Verkehrsmittel Güteklassen or ÖV-Güteklassen) [33]. This metric categorizes the quality of public transit access based on the distance to the nearest transit stop, the type of transit offered and the level of service provided at the stop. The Austrian ÖV-Güteklassen is developed based off of the Swiss system of the same name [17]. Both systems use the same mechanisms to compute the quality classes, however the Austrian system expands the categories and classes to offer ranking for lower frequency transportation modes.

To compute this metric, the public transit stations are first ranked into categories I-VIII based on the average departure interval at the station and the type of transportation service offered. Table 2.1a is used to compute the category ranking for each stop in the network. For example a train station offering high speed rail service with departures every 10 minutes or less would be ranked into Category I, where a bus station with less than one departure every two hours would be put into Category VIII.

Once the category of each station is computed, the quality class can be computed using the station category and the distance from the station. The quality classes range from A to G with A being reserved for regions that are close to station in Category I or II, and class G for areas that are further away from stations of a lower service category. As can be seen in Table 2.1b, the quality class reduces with both increasing distance from the station and worsening service class for the station. To generate maps of the quality classes, isochrones are computed around each station and divided into the different classes based on the distance from the station.

Tab. 2.1: Tables used to compute the Public Transit Quality Classes (ÖV-Güteklassen), reproduced in English from [33]

Average Departure Interval for All Directions	Highest Transportation Category At Station			
	Intercity Trains	Regional Rail/ Subway/ Express Bus/ Local Trains	Trams/ Metro Bus/ 0-Bus	Bus
$x < 5$	I	I	II	III
$5 \leq x < 10$	I	II	III	III
$10 < x < 20$	II	III	IV	IV
$20 \leq x < 40$	III	IV	V	V
$40 \leq x \leq 60$	IV	V	VI	VI
$60 \leq x \leq 120$	V	VI	VII	VII
$120 < x \leq 210$		VII	VIII	VIII
$210 < x$				

(a) Stop Categories Table

Stop Category	Distance To Stop				
	0 - 300 m	301 - 500 m	500 - 750 m	751 - 1000 m	1001 - 1250 m
I	A	A	B	C	D
II	A	B	C	D	E
III	B	C	D	E	F
IV	C	D	E	F	G
V	D	E	F	G	G
VI	E	F	G		
VII	F	G	G		
VII	G	G			

(b) Transit Service Quality Classes

The public transit quality classes are a useful tool to visualize the transit accessibility of a region. The quality class isochrones provide a clear visual of where there are accessibility gaps, however proximity to transit is only part of what makes transit effective. When computing the station category, no consideration is given to the connectivity of the transit services at that station, only the frequency of the departures. This means

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

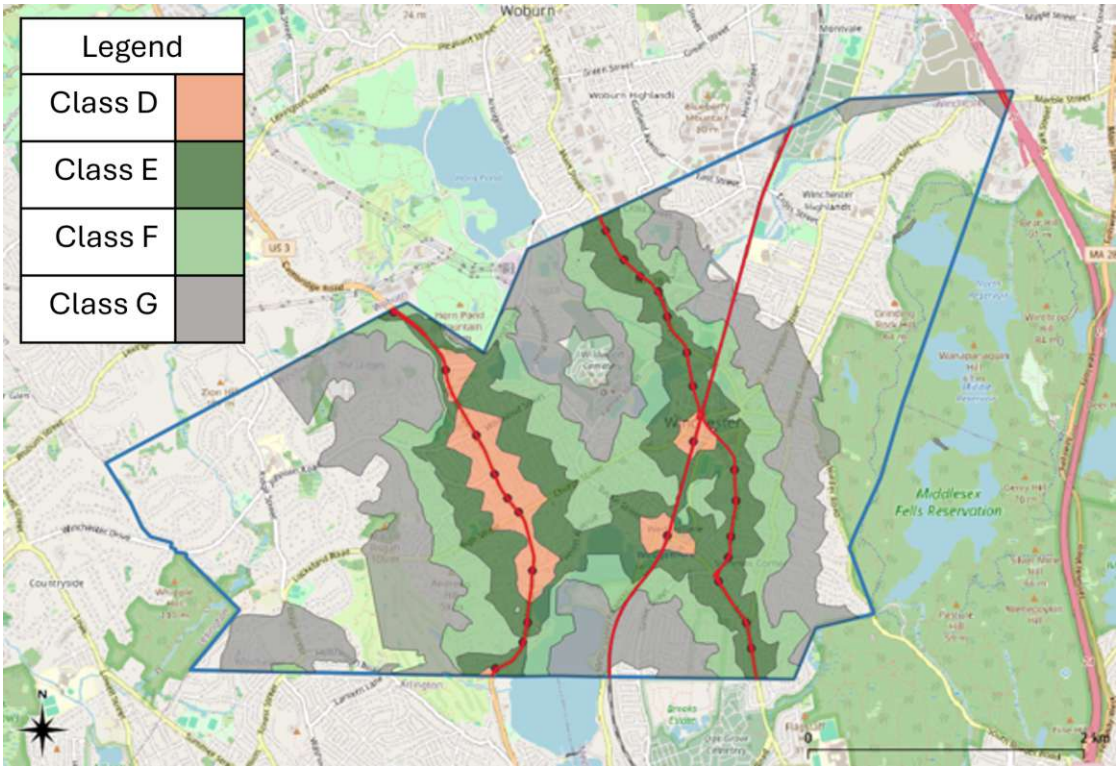


Fig. 2.1: ÖV-Güteklassen for Winchester, MA. Map generated with QGIS with data from [28].

that a region with poor network design could have a high quality transit coverage but poor functional connectivity. Figure 2.1, shows the quality classes for Winchester, MA. In this example there is moderate coverage of the town with classes D though G represented. What is not captured is that the transit routes in Winchester, MA only over service into the Boston metro area and have no interconnections between them. As a result the quality classes are only representative of accessibility when traveling to a specific destination. Computing the ÖV-Güteklassen for Winchester highlights an issue where regions with poor network design can have a high quality transit coverage but poor functional connectivity.

2.1.3 London Public Transit Accessibility Level

Another tool to measure transit accessibility is the London Public Transit Accessibility Level(PTAL) [16]. This method is similar to the ÖV-Güteklassen and has as been used by Transport for London(TfL) since the early 2000's. The major difference between the PTAL and ÖV-Güteklassen is that the ÖV-Güteklassen computes accessibility with relationship to the stations whereas the PTAL computes the accessibility of different points in the region. Additionally, the PTAL utilizes some additional information such

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

as exact walking distances, reliability factors and the specific service offered by each mode.

To compute the PTAL, a set of points called POI is required. This can be a grid of uniformly spaces points or a list of specific places. For each of these points the walking times to all bus stations within 640 meters and rail stations within 980 meters are computed. These stations are called Service Access Points(SAP). For each SAP and each route that services it, the average waiting time is computed by analyzing the service frequency. Using the walking time, the waiting time, and reliability factors, an estimated doorstep frequency(EDF) is computed, the EDF is a value that represents the average waiting time if the route left from the POI location. Next, the accessibility of the route to a particular POI is computed by multiplying the EDF and a defined route weight. The accessibility of the POI is then computed by summing the accessibility of each route that can be accessed from that POI. Lastly the PTAL value is computed by binning the POI accessibility values into different pre-defined bands.

Recently the PTAL method defined by TfL, has been improved on by Su et al. [34] In their 2023 paper, they focused on developing new methods to compute the PTAL value from the accessibility value. The authors expanded upon the standard binning method by testing k-means clustering and statistical grouping methods, such as the equal interval and quantile methods. The results of their analysis showed that a k-means clustering method with six clusters was the best performing method for the case study of Shenzhen, China. Additionally, they found that the clustering method showed variation between the areas with high and low public transit accessibility better than the equal interval, quantile methods, or static range method used by TfL.

2.1.4 Composite Accessibility Index

Another method to compute accessibility is the Composite Accessibility Index developed by Rathod et al. [30]. The goal of this method is to create an accessibility metric that covers the following accessibility categories, system accessibility, system-facilitated accessibility, integral accessibility, and time–space accessibility. To do this the authors combine three existing accessibility metrics, the Public Transport Coverage Index(PTCI) [18], Network Geospatial-Based Accessibility Index(NGAI) [15] and Comprehensive Public Transport Accessibility Index(CPTAI) [40].

The three metrics are combined into the CAI by multiplying each with a weight and summing the products. The weight applied to each supporting metric was carefully considered because the input data can be shared across the multiple metrics. For instance, the service frequency is used in all three metrics, where the number of bus stops in only used by the PTCI and NGAI. To define the weights, the authors created three schemes, one where all inputs are equally considered, one where repeated inputs have a higher impact and one where the all inputs are normalized to the same frequency. ANOVA testing of the schemes showed that there was no significant difference between the weighting methods, and as a result the authors proposed using the normalized weighting scheme.

The ÖV-Güteklassen, PTAL, and CAI metrics are focused on measuring the spatial-accessibility of transit. In general these metrics attempt to quantify the closeness and

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

quality of transit services. One aspect that is missing from this analysis is the connectivity offered by the transit services. In the paper by Rathod et al., the authors used the term system-facilitated accessibility, for accessibility metrics that incorporate system travel times into the analysis. The referenced literature, focuses on using GIS to compute isochrones using the public transit network, and route planning methods that minimize route overlap. While travel time isochrones can effectively show coverage regions from a single point, they do not represent the connectivity of the whole network. The network layout impacts the travel times for the entire region.

2.2 Route Optimization

There are a huge variety of different type of route optimization tasks in the transit planning and computer science fields. The problems range from purely theoretical analysis to real world examples. The most abstract optimization tasks are route planning problems that have been transformed into pure combinatorial computer science problems. One of the more applied problems is timetable optimization. These problems focus on searching for cost optimal schedules based on ridership, staffing and shift requirements. Another type of route optimization is tasks that focus on improving stop locations and making small changes to the route. Lastly there are pure route optimization tasks that involve generation of new routes from scratch. These tasks range from finding last mile routes complimenting existing transit, to tasks that remove entire bus routes and search for entirely new routes.

2.2.1 Optimization Problems

Multiple transit planning problems have been abstracted into pure computer science problems. Example problems include the Urban Transit Route Problem (UTRP) [20], and the Transit Network Design and Frequency Setting Problem(TNDFST) [11]. These problems are presented with a standardized set of requirements and optimization goals. The standardization of these problems allows for algorithmic study of the problems and complexity analysis.

The UTRP, an NP-Hard problem, has undergone a considerable amount of study since the initial work by Mandl in 1980 [27]. The problem involves finding a set of routes that satisfy the demand matrix in a particular network. In the initial work by Mandl, the authors proposed using heuristics in a structured route generation and modification framework. Current work on the problem spans meta-heursitcs [13], integer programming [19] and machine learning algorithms [20]. Recent papers have also shown very good results with a partial swarm optimization [23]. When applying this method, the authors were able to reduce computation time by 90% and found a solution that outperformed other methods.

The Transit Network Design and Frequency Setting Problem(TNDFST) is similar to the UTRP because it also involves finding a set of transit routes that satisfy a provided demand matrix. The difference is the incorporation of route frequency setting into the

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

optimization task. Traditionally the routes and frequencies are optimized separately due to the high levels of complexity for each. In the summary paper by Duran et al. [11] the authors identified that the complexity of the TNDFST requires significant simplifications in order to be solved, and they highlight the difference between simplified evaluation instances and real world problems. Despite the complexity, authors have applied many modern heuristic methods to the problem including genetic algorithms [5], swarm optimizations [26] and local-search based optimizations [1].

2.2.2 Timetable Optimization

The timetable is a common focus area for optimization when it comes to public transit. Changes to the timetable are under the domain of the operator and as a result, improvement in this area is easy to implement. Current research into timetable optimization is focused on finding changes that reduce costs while improving outcomes for riders. This creates a multi-objective optimization problems that are solved with heuristic methods and machine learning.

In a paper by Huang et al. [22], a genetic algorithm was used to complete a multi-objective optimization on train timetables. This process aimed to minimize operating costs while maximizing passenger efficiency. The problem was solved by utilizing passenger flow data to model the real time demand and dynamic progaming to simulate train operation. Constraint conditions were also applied to model minimum tracking intervals, maximal number of trains and other operational constraints. The results from this optimization showed that the improved timetable was aligned with the spatial-temportal passenger demand. Additionally, the improved schedule achieved high fitness to the objectives indicating that performance was improved and costs were reduced.

Machine learning algorithms have also been applied to timetable optimization problems. In their 2022 paper, Yan et al. [39] used a Markov decision process and deep reinforcement learning to improve the timetables for a multi-line bus network in Beijing, China. The results of the case study showed that the algorithm was able to outperform heuristic methods and state-of-the-art reinforcement learning methods. Ultimately the authors were able to show as 20.3% reduction of operating passenger costs when compared to the actual timetables.

2.2.3 Stop Optimization

Stop optimization problems are focused on modifying bus stop locations to enhance accessibility, reduce travel time, and optimally allocate resources. Solutions to these problems involve data-driven approaches, spatial analysis and pure optimization algorithms. A recent study in Hohhot City [43] applied a data-driven approach by using k-mean clustering algorithms to analyze and optimize bus stop locations. This study used cellular data in the central businesses district to determine where potential riders were traveling to. This data was grouped into clusters to define potential bus stop locations that were ideally located. The proposed bus stops were then evaluated using POI information. Reviewing the nearby points of interest showed that the new bus stops

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

have higher concentrations of POI than the existing stops. The average number of POI near each bus stop increased from 14.6 to 15.1 after optimization, however the average distance to POI increased from 123 meters to 130 meters.

Additionally, multiple authors have applied traditional heuristic optimization algorithms to bus stop location improvement. In a paper by Li et al. [24], particle swarm optimization(PSO) is used to optimize the bus stop locations on a number of routes in Waterloo, Canada. This optimization process showed that the PSO algorithm can find stop configurations that both reduce the total number of bus stops and the average travel time in the network. In a study by Wang et al. [37], a genetic algorithm was used to find locations for new bus stops in China. To achieve this the authors used a modified potential model to represent the spatial accessibility. The potential model was modified to include effects relating to service capacity, bus frequency, travel impedance and more factors. Using a non-dominated sorting genetic algorithm the authors were able to find 51 new bus stop locations that should be added to the network, after incorporating these stops the accessibility of the network increased by 8.8%.

2.2.4 Route Optimization Example

In a paper by Yoo and Lee [42], a combined accessibility ranking and optimization was performed. In this paper the authors worked to create improvements to the transit network, specifically relating to the transit disadvantaged. They developed the concept of Transit Service Need Index(TSNI), which models the level of transit need in each portion of a region. The TSNI was used in combination with an accessibility score to improve transit routes in Penrith, Australia.

The TSNI was created to model the local need for transit service. For each region of the city, called a Mesh Block(MB) five different parameters are extracted, weighted and linearly combined into the TSNI value. The parameters are the population of the block, the number of disabled individuals, the number of elderly, the average income of the block and the number of personal vehicles in the block. These parameters are weighted by computing the percentile that each MB fall into based on the entire region and ranking it from 0 to 5 based on the percentile value. For all parameters the TSNI value increased with increasing percentile, except the income parameter which is inverted.

In addition to the TSNI value for each MB, the transit accessibility for each MB is computed. This value is a measure of how many destinations can be reached via transit from the bus station in the MB. The accessibility value is determined by counting the number of destinations that can be reached within forty minutes. To compute the score for the entire region, the accessibility score and TSNI value are multiplied for each MB and then the product is summed over all MB's.

With the network transit accessibility defined, reinforcement learning was used to generate new bus routes that would maximize the accessibility. In general reinforcement learning has two modes, exploration or exploitation. In the exploration mode, actions are taken randomly to explore the search space and in the exploitation mode, the best action for the current state is taken. Switching between the modes is based on the epsilon-greedy strategy. For this problem the authors utilized two different phases in

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

the reinforcement learning process. There is a destructive step where existing bus routes are removed from the transit network and a constructive step where actions are taken to add or expand existing routes in the network. In both of these phases, the algorithm uses the epsilon greedy method to either explore or exploit the reward. Details regarding how epsilon is updated can be seen in the source paper [42].

Results for the study area of Perith, Australia showed that the optimization process worked. Four total scenarios were run, cases where one or two routes were modified and cases with and without the TSNI index. All cases showed an increase in the transit accessibility score. In general, the optimization process removed long rural routes with low service frequencies, and added shorter suburban routes with higher service frequencies. Additionally, the authors found that applying the TSNI parameter, improved service to regions with high need.

2.3 Relationship to Current Work

The work presented in this thesis improves on the idea of transit accessibility by defining a new metric called transit connectivity. The connectivity metric quantifies the accessibility and the effectiveness of the transit system. Existing accessibility metrics such as the ÖV-Güteklassen and PTAL only consider the spacial relationship to transit, the frequency of service and the type of service offered. The metrics used as part of the Composite Accessibility Index do consider impacts of the network, however these impacts are only a function of the transit network coverage area and not the travel times inside the network.

To improve upon the existing metrics the connectivity is based on the travel times from the source or origin to any destination based on average travel times utilizing the public transit system. The formula for the connectivity metric(detailed in Section 3.1) is based upon the definition of the potential model shown in Equation 2.1, where the distance $d_{i,j}$ is the total travel time using the transit network. Furthermore, instead of only considering a finite set of destinations, the connectivity of a point considers the travel time to all points in the region. In addition to increasing the scope to all points in the region, the attractiveness of the destination was also modified into a utility value.

In the paper by Yoo et al. [42] the authors made a first step at the idea of destination utility. Firstly they created the TSNI that defined the level of transit need for different parts of the region. Additionally, they attempted to define how useful a transit station is by calculating the number of points of interest that can be reached from that station within forty minutes. In this thesis the utility value is defined as a combination of both of these ideas. The value is fully user customizable and can be based off of population density, points of interest, income, car ownership or any other spatially distributed aspect of the region. Another improvement when compared to the analysis by Yoo et al. is the connectivity metric. Yoo and co authors assess accessibility by looking at the number of points of interest that could be reached within forty minutes of transit travel. While this was effective, it does not capture travel habits well. One major issue is with the limits on travel time to POI. Points of interest that are ten or twenty minutes away are

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

equally weighted with those that are forty minutes away. Additionally, destinations that are more than sixty minutes away are not considered at all. Furthermore, there is no consideration given to the type of POI, destinations of all types are equally weighted. The connectivity metric, overcomes these issues by creating a continuous distribution that represents the utility offered by different POI across the optimization region. This utility value is then directly combined with the travel time to each destination. With this method useful destinations that require longer travel times can have similar or higher utility than less useful nearby destinations

The final major difference between this thesis and the existing literature is the comprehensive and symmetrical computation of the connectivity metric. The connectivity of a region is the sum of the connectivity of every point in the region. This means that every point in the region is considered as a source and a destination point.

The core of the optimization task for this thesis is a route planning optimization using a meta-heuristic. Meta-heuristics are well studied and widely used for optimization tasks, and the literature has many examples of meta-heuristics being used for transit related optimizations [41] [25] [29] [23]. The algorithm applied to the problem in this thesis is a local-search based meta-heuristic called Generalized Variable Neighborhood Search(GVNS). One aspect of the optimization process that is unique is the scope of the possible routes in the search space. Typically, transit route optimizations consider only a small number of existing stops as possible stations in the route. In this thesis, the potential stop locations are indiscriminately distributed across the region. This results in a large search space.

This thesis implements the connectivity parameter and heuristic optimization for the case study region of Winchester, Massachusetts. By doing so, the author shows the usefulness of the connectivity parameter as an evaluation tool for a region. Additionally, the thesis demonstrates the ability of computational search to find new potential bus routes in a region of interest.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Chapter 3

Mathematical Model

This chapter provides the mathematical foundation for the thesis. It covers the development of the connectivity metric, the details of the utility function, the external utility and a method to compare two solutions to one another.

3.1 Connectivity Metric

The creation of the connectivity metric was based upon two major assumptions. Firstly that each point in the region of interest has an inherent utility that can be quantified and secondly that a region with high connectivity will have short travel times to destinations with high utility. The final component to the connectivity metric is not an assumption, but a choice made by the author. This component is that the connectivity of a point should consider trips to all destinations in the region R, not only a pre-defined subset. Using these assumptions a relation can be created for the connectivity of a single point. This relation can be seen in Equation 3.1

$$CM_a \propto \frac{U_b}{D_{a,b}} \qquad \forall b \in R. \tag{3.1}$$

In Equation 3.1 the connectivity metric CM, is computed for a single point $a \in R$. This connectivity value is directly proportional to the utility of a destination b, represented with parameter U_b, and inversely proportional to the distance $D_{a,b}$, which is between the source point a and destination b. Finally, this proportionality holds for all point b in the region R.

Readers familiar with transit planning models will recognize the similarity with the potential and gravity models. The potential model shown in Equation 3.2 [32]

$$P_i = \sum_{j \in D} \frac{M_j}{d_{i,j}^a} \qquad a \in \mathbb{R}, \tag{3.2}$$

is the most similar to the proposed connectivity metric. The major difference between the models is that the potential model is traditionally used on sets of sources and destinations that contain no shared points. For example a set of warehouse locations as sources and customer locations as destinations.

The relation for the connectivity metric shown in Equation 3.1 is also similar to the gravity model shown in Equation 3.3

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

$$T_{ij} = a\frac{P_i P_j}{c_{i,j}^b}. \tag{3.3}$$

In this model, P_i and P_j are the weights of two points i and j. Similarity c_{ij} is the distance between the two points. This model is based on the formula for gravitational potential and is often used to model travel between different regions of a city [12]. This model was not used because complications arise when considering path $i \to j$ and $j \to i$. If both of these connections are utilized then it can result in a double counting of the weights since the weight of the both the source and destination are incorporated into the formula.

The final formula for the connectivity metric was developed by extending the potential model. The method was extended symmetrically to consider the same set of locations for the sources and the destinations. Lastly the connectivity of each location is summed into one global parameter that acts as a metric for the entire region. This results in Definition 3.1.1.

Definition 3.1.1 (Connectivity Metric)**.** The connectivity metric for the bounded region R is given by

$$CM = \iint_{x\in R} \iint_{x'\in R} \frac{u(x')}{d(x,x')}\, dA\, dA, \tag{3.4}$$

where $u : \mathbb{R}^2 \to \mathbb{R}$ is the utility density function and $d : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$ is the distance function between two points.

Depending on the choice of distance function, the connectivity of a region can be unbounded. Since $\lim_{x\to x'} d(x,x') = 0$ the integral can grow without bound if the distance function does not decrease rapidly enough. One method to overcome the issue with unbounded integration is to use a discrete form of the connectivity metric. This discrete form replaces the definite integral with a finite sum over a set of points. The discrete version of the connectivity metric can be seen in Definition 3.1.2.

Definition 3.1.2 (Connectivity Metric - Descrete)**.** Give a set of point $V \subset \mathbb{R}^2$ the connectivity metric for these point is given by

$$CM = \sum_{u\in V} \sum_{v\in V\setminus u} \frac{U_v}{D_{u,v}}, \tag{3.5}$$

where $U_v \in \mathbb{R}$ is the utility of point v and $D_{u,v} \in \mathbb{R}$ is the distance between points u and v.

3.2 Utility Computation

The utility value $u(x)$ is a parameter that should quantitatively represent the attractiveness of a destination point. The challenge when defining this parameter is that the

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

attractiveness of a particular destination is not related to any real world parameters. This means that the magnitude of a particular point's utility will be arbitrary and only the difference in utility between points is important. Since the utility can be defined arbitrarily it is important to have a clear mathematical framework for its generation.

This section will detail the process of the utility computation starting with a continuous form of the utility and then detailing the process to map the continuous utility to a set of finite points.

3.2.1 Utility Components

The utility of a point is based on two factors, firstly the population density at that location and secondly the utility provided by the businesses and other points of interest (POI) in the local area. The population density is provided by the population density function $p : \mathbb{R}^2 \to \mathbb{R}$ that maps a point in the region to the population density. The local utility provided by the POI is modeled with the local utility function $lu(x)$. The specific formulation of $lu(x)$ is nontrivial because the exact POI used in the analysis can vary. Additionally, the POI points can be divided into different types, such as schools, parks or businesses, and depending on the configuration the different POI types may have different contributions to the local utility.

To define the local utility in a rigorous way, the relative weightings of the POI types must be defined. The set of all POI types is designated T with a specific type noted by τ. The weight of each POI type is defined in by the set W, this set is formally defined in Definition 3.2.1

Definition 3.2.1 (POI Weights)**.** Given a set of categories T, the corresponding category weights are defined by

$$W = \left\{ w \in \mathbb{R} \middle| |W| = |T| \wedge \sum_{w \in W} w \ = 1 \right\}, \tag{3.6}$$

where the value of w_τ is equal to the weighting for category T_τ. Additionally, the existence of set W induces set W^* given by

$$W^* = \left\{ W \in \mathcal{P}(\mathbb{R}) \middle| |W| = |T| \wedge \sum_{w \in W} w \ = 1 \right\}. \tag{3.7}$$

In addition to the weights for each POI type, the set of all POI locations must be defined. This set is denoted by Ψ with a specific point indicated by ψ. Additionally, each set of POI can be divided into subsets based on the type of each POI location. These subsets are denoted by Ψ^τ where τ is the specific type.

Definition 3.2.2 (POI Locations)**.** Given a set of POI categories T and the set of all POI locations $\Psi \subset \mathbb{R}^2$, the POI subset for category $\tau \in T$ is given by

$$\Psi^\tau = \{ \psi \in \Psi | \text{typeof}(\psi) = \tau \}, \tag{3.8}$$

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

where the function typeof : $\mathbb{R}^2 \to T$ returns the type of the provided POI location. Additionally, the existence of set Ψ induces the set Ψ^* given by

$$\Psi^* = \left\{ \Psi \in \mathcal{P}(\mathbb{R}^2) \right\}. \tag{3.9}$$

Using the sets defined in Definition 3.2.2 and Definition 3.2.1 a formula for the local utility of a given point can be created. Since the local utility is not related to a physical property the implementer must define a total amount of utility to be distributed across all POI locations. This total utility value is given by $\mathcal{V}$. Using this an initial formula for the local utility is given by Equation 3.10

$$lu : \mathbb{R}^2 \times \Psi^* \times W^* \to \mathbb{R} \qquad x \times \Psi \times W \mapsto lu(x, \Psi, W) := \begin{cases} \frac{w_\tau}{|\Psi^\tau|}\mathcal{V} & x \in \Psi \\ 0 & \text{otherwise} \end{cases}. \tag{3.10}$$

In Equation 3.10 the total utility $\mathcal{V}$ is scaled first by the category weighting w_τ and then by the number of locations in that category $|\Psi^\tau|$. This utility is applied when the input point x is one of the points of interest, however when the point is not a point of interest the local utility function is zero.

The piece wise definition results in a local utility function that is non-continuous, with large spikes at the points inside the POI set. It is hypothesized that this will make the resulting optimization process difficult because in general optimizers perform better when there is a smooth distribution that can be climbed to region of local maxima. To achieve this, a smoothing function can be applied to the utility provided by each POI. This smoothing function spreads the utility out over the local area and transforms the non-continuous function into a continuous one.

For this problem a two-dimensional Gaussian function was chosen as a smoothing function. This function requires three parameters, the center point of the distribution x', the symmetric spread of the function σ and the amplitude of the function A. The standard definition of the Gaussian distribution is given in Equation 3.11

$$G : \mathbb{R}^2 \to \mathbb{R}, \qquad x \mapsto G(x) := A \exp\left(-\frac{(x_1 - x_1')^2 + (x_2 - x_2')^2}{2\sigma} \right). \tag{3.11}$$

The spread of the smoothing function can be set specifically for the implementation, however the amplitude A must be set as to not scale the total utility. To achieve this the amplitude of the smoothing function must be set such that the integral of the smoothing function is equal to the total utility $\mathcal{V}$. This results in the value for A show in Equation 3.12

$$\mathcal{V} = \iint_{\mathbb{R}^2} G(x) dA = 2\pi A \sigma^2 \implies A = \frac{\mathcal{V}}{2\pi\sigma^2}. \tag{3.12}$$

Using the computed value for the amplitude the smoothing function $g(x, x')$ can be defined in Definition 3.2.3.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Definition 3.2.3 (Local Smoothing Function)**.** Given a provided volume $\mathcal{V}$ and spread σ, the local smoothing function is given by

$$g : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}, \qquad x \times x' \mapsto g(x, x') := \frac{\mathcal{V}}{2\pi\sigma^2} \exp\left(-\left(\frac{(x_1 - x_1')^2 + (x_2 - x_2')^2}{2\sigma}\right)\right), \tag{3.13}$$

where x is the input point and x' is the center of the smoothing function.

With the smoothing function defined the local utility function can be updated. This results in Definition 3.2.4.

Definition 3.2.4 (Local Utility Function)**.** Given a set of POI locations Ψ, a set of POI category weights W and a smoothing function $g(x, x')$, the local utility function is given by

$$lu : \mathbb{R}^2 \times \Psi^* \times W^* \to \mathbb{R}, \qquad x \times \Psi \times W \mapsto lu(x, \Psi, W) := \sum_{\tau \in T} \sum_{i \in \Psi^\tau} \frac{w_\tau}{|\Psi^\tau|} g\left(x, \psi_i^\tau\right). \tag{3.14}$$

Since the smoothing function is defined over the entire region of $\mathbb{R}^2$ the local utility of any point is based on the sum of the smoothing functions for every POI location. Naively one would think that the application of the smoothing makes the local utility function not "local" any longer. However, the shape of the Gaussian distribution results in trivially small applied utility at points that are more than 3σ from the center point of the smoothing function.

With the local utility function well-defined the final utility function can be constructed and defined in Definition 3.2.5.

Definition 3.2.5 (Utility Density Function)**.** Given a population density function $p(x)$, a local utility function $lu(x, \Psi, W)$ and constants α and β, the utility density function is given by

$$u : \mathbb{R}^2 \times \Psi^* \times W^* \to \mathbb{R}, \qquad x \times \Psi \times W \mapsto u(a, \Psi, W) := \alpha p(x) + \beta lu(x, \Psi, W). \tag{3.15}$$

The constant α is defined to be the ratio between the importance of the population density to the local utility density. The constant β is a computed value that serves to scale the local utility density such that it has the same impact as the population density when $\alpha = 1$. The computed value for β can be seen in Equation 3.16

$$\alpha = \frac{\iint_R p(x)\, dA}{\beta \iint_R lu(x)\, dA} \underset{\alpha=1}{\Longrightarrow} \beta = \frac{\iint_R p(x)\, dA}{\iint_R lu(x)\, dA} \approxeq \frac{\iint_R p(x)\, dA}{\mathcal{V}}. \tag{3.16}$$

The substitution of $\iint_R lu(x)\, dA$ with $\mathcal{V}$ is possible because of the definition of the smoothing function $g(x, x')$. Proof of this relation is shown in Equation 3.17

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

$$\begin{aligned}
\iint_R lu(x,\Psi,W)\,dA &\approxeq \iint_{\mathbb{R}^2} \sum_{\tau\in T}\sum_{i\in\Psi^\tau} \frac{w_\tau}{|\Psi^\tau|} g\left(x,\psi_i^\tau\right)\,dA \\
&\approxeq \sum_{\tau\in T}\sum_{i\in\Psi^d} \frac{w_\tau}{|\Psi^\tau|} \iint_{\mathbb{R}^2} g\left(x,\psi_i^\tau\right)\,dA \\
&\approxeq \sum_{\tau\in T}\sum_{i\in\Psi^\tau} \frac{w_\tau}{|\Psi^\tau|}\mathcal{V} \\
&\approxeq \mathcal{V}\sum_{\tau\in T} w_\tau \sum_{i\in\Psi^\tau}\frac{1}{|\Psi^\tau|} \\
&\approxeq \mathcal{V}\sum_{\tau\in T} w_\tau \\
&\approxeq \mathcal{V}.
\end{aligned} \tag{3.17}$$

The relation is only an approximation due to issues with the domain of integration. Strictly speaking $\iint_R g\left(x,\psi_i^\tau\right)\,dA \neq \mathcal{V}$ because the integral is only equal to $\mathcal{V}$ when the domain of integration is the entirety of $\mathbb{R}^2$. For practical purposes the error is trivially small when the POI locations in Ψ are more than 3σ from the boundary of R. This is not necessarily true in all cases however in the majority of cases the error is small.

3.2.2 Mapping to Discrete Domain

Instead of using a continuous function to define the local utility density and population density, these functions are discretized to a finite set of grid points spanning the region. This process is necessary for two main reasons. Firstly the population data cannot be described by a continuous function. The data is provided in the form of a population raster. This dataset divides the region into pixels of $90m^2$ and a population count is provided for each pixel. The second reason is that the resulting utility function will be integrated numerically, this integration process would require a numerical discretization in any case.

To discretize the domain, a set of regularly spaced grid points is required. The points should span the entire domain and have a uniform spacing of s_u. This point set X_u can be generated using a lattice and is shown in Equation 3.18

$$X_u = \left\{ x_i \in \mathcal{L}\left(s_u\mathbf{I}^2\right) | s_u \in \mathbb{R}, x_i \in R \right\}. \tag{3.18}$$

The resulting point set is aligned with the horizontal and vertical axes, has uniform spacing of s_u in both directions, and is limited to only points inside the boundary of R.

In addition to the discretized points X_u, there also exists a set of points where the population data is defined. This set of points is labeled as X_p, and is defined by a lattice $\mathcal{L}\left(s_p\mathbf{B}\right)$ where s_p is the spacing of the population data ($s_p = 90$) and $\mathbf{B} \in \mathbb{R}^{2x2}$ is the basis matrix for the population data grid. Using this set of points, the population count

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

information can be encoded into a function $P : x \in X_p \to \mathbb{R}$. This function takes one of points in X_p and returns the population count at that point.

Additionally, we can define a function that take in any arbitrary point and returns the nearest point from the population data lattice. This function is defined in Equation 3.19. This equation returns the point in X_p that is the minimum distance from the input point x

$$\begin{aligned} &NN : \mathbb{R}^2 \times X_p \to \mathbb{R}^2, \\ &\quad x \times X_p \mapsto NN(x, X_p) := \left\{x_i \in X_p | \ \|x, x_i\|_2 = \min\left(\left\{\|x, x_j\|_2 \ \forall x_j \in X_p\right\}\right)\right\}. \end{aligned} \tag{3.19}$$

Using Equation 3.19 a function to compute the population density at any arbitrary point can be derived. The result is shown in Equation 3.20, this equation passes the arbitrary input point to the nearest neighbor function. This returns the nearest point in the population data lattice. With this point the population data function $P(x)$ can be used to get the population count at that location. The return value is then divided by the square of the population data spacing to convert the population count into population density.

$$p : \mathbb{R}^2 \times X_p \to \mathbb{R}, \qquad x \times X_p \mapsto p(x, X_p) := \frac{P\left(NN\left(x, X_p\right)\right)}{s_p^2} \tag{3.20}$$

Since the road network does not span the entire domain, the region of interest is divided into the set V of discrete points. These points are locations in the walking network, such as intersections. In addition to the physical location each point is V is a vertex on the walking graph that define how pedestrians can move around the region. Instead of computing the connectivity metric using continuous functions on the entire region a the discrete version will be used. This version is shown in Equation 3.4. To utilize this function the utility of each point needs to be defined. To compute the utility of each point in V the region is divided into a Voronoi tessellation using the points of V as the center of the tiles. These subregions $R_u \subset R \ \ \forall u \in V$ are the area of R that is nearest to the point u. To compute the utility for the point u the utility density should be integrated over the entire region R_u.

$$U_u = \iint_{R_u} u(x) dA \approxeq \sum_{x \in X_u \cap R_u} u(x) s_u^2 \tag{3.21}$$

3.3 External Utility

In addition to the utility inside the region of interest, the utility of external points should be considered. The external regions are selected areas that can be reached from the region of interest via the exsiting transit connections. Locations such as adjacent town centers, shopping areas and more can be selected as external regions. An important consideration when adding these regions is that the utility of the external regions is scaled correctly to the utility inside the primary region. This ensures that the external

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

regions have the correct relative importance with respect to the POI inside the primary region.

The access to the external regions is achieved via utilizing existing transit routes the pass though the borders of the region. The general case that will be considered is there are a finite number of external regions that are connected back to the main region of interest via existing transit routes. These external regions have their own POI with any subset of the POI types and any number of destinations of each type.

To apply the utility in a balanced way, the usefulness of the external regions must be linked to the utility definition in the main region. To do this the utility share of a single destination of each POI type is computed.

$$\text{share} = \frac{w_\tau \mathcal{V}}{|\Psi^\tau|} \tag{3.22}$$

This share is then applied to each destination of the specific type in each external region. This means that the external regions can have as much or more utility as the region of interest.

Additionally, to simplify the process, the positioning of the POI destinations in the external region is abstracted. Instead of retaining the position of each destination, all the utility of the destination region is applied to a singular point. This simplifies the computation process.

3.4 Route Comparison

In later sections of this thesis a method is required to analyze the similarly between a set of solutions. Due to randomness incorporated into the optimization process, each configuration of the optimization was repeated dozens of times. This set of solutions needed to be analyzed quantitatively to understand what the average route is and the variation around that average route. To do this a method was created to quantify the similarity between any two routes.

Given two solutions S_1 and S_2 a function is needed that maps the difference between these solutions to a real number, $f(S_1, S_2) \mapsto \mathbb{R}$. Due to the nature of the solutions this is nontrivial to compute. The solutions to the optimization problem are represented as a list of points in 2D space. The route is formed by connecting these points in the order provided and then connecting the end point back to the beginning to form a loop.

The challenge with this definition is that when comparing the shape of the route there are multiple unique routes that have the same shape. For instance routes can start at a different vertex on the same looping path. Additionally, the direction is not important a route can have clockwise or counterclockwise direction and still have the same shape. For examples the following routes are equivalent $[(0,0),(1,0),(0,1)] = [(0,0),(0,1),(1,0)] = [(1,0),(0,1),(0,0)]$. Finally, routes can have any number of stops (points) and there is no ideal route that can be used to compare against. These factors make comparing two routes very difficult.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

The general idea of the comparison function $f(S_1, S_2)$ is to map the two solutions to a vector in $\mathbb{R}^n$ and then compute a norm of this vector. The norm maps the vector in $\mathbb{R}^n$ to a value in $\mathbb{R}$. The mapping from the solution into $\mathbb{R}^n$ is fairly straightforward. The solution S_1 is divided into n evenly spaced points, for each of these points the minimum distance to solution S_2 is computed using Equation 3.23. Every line segment in the solution needs to be tested since there is no information on the ordering and position of the segments. The distance stored in the output vector is then the minimum tangent distance to any of the line segments in solution S_2.

$$\text{distance}(x, x', x^0) = \frac{|(x'_2 - x_2)x^0_1 - (x'_1 - x_1)x^0_2 + x'_1x_2 - x'_2x_1|}{||x' - x||} \tag{3.23}$$

Equation 3.23 considers the lines defined by x and x' to be infinite in length. This can create issues when evaluating a point that is past one of the endpoints of a line segment. Since the line segments are extended, the distance can be erroneously small. An example of this can be seen in Figure 3.1. In the figure, the black line is the baseline route and the blue line is the alternate route. The red section is the region of the blue route that has the distance erroneously computed because the points are closer to the dashed line than the vertical black line.

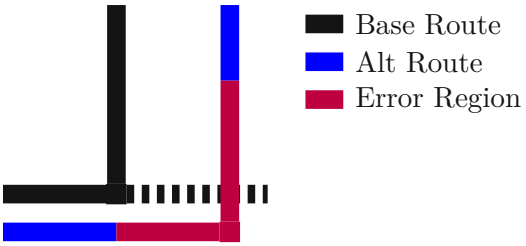


Fig. 3.1: Example of computation errors in the path offset

To overcome this issue, equation 3.23 was only considered valid when the point of interest x^0 was next to the line segment. This condition was checked by computing the offset vector from start of the line segment to the point of interest, $x - x^0$ and then projecting that vector onto the line segment. If the projected distance is greater than zero and less than the length of the line segment then Equation 3.23 is used, otherwise the minimum distance to one of the end points is returned as the distance to that line segment.

With the difference between the two solutions mapped to a single delta vector in $\mathbb{R}^n$ the resulting norm can be computed using the Euclidean Norm. This norm is computed by taking the square root of the sum of squares. This results in the definition of the solution comparison function $f(S_1, S_2)$ shown in Definition 3.4.1

Definition 3.4.1 (Solution Comparison Function)**.** Given two solutions S_1, S_2 which are ordered lists with $S_i = (s^1_i, s^2_i, ..., s^l_i)$ where $s^j_i \in V$, the solution segments can be defined by,

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

$$\text{segments}(S_i) = ss_j = (s_i^j, s_i^{(j+1)\bmod l})|\forall j \in 1...l, \tag{3.24}$$

the length of a solution segment can be defined by

$$\text{length}(ss_i) = ||s_i^{(j+1)\bmod l} - s_i^j||_2, \tag{3.25}$$

and the *path* of a solution is defined as the list of all solution segments from ss_1 to ss_l.

Given a precision constant $n \in \mathbb{R}$, the list of point X is defined by the set of equally spaced points along the path of solution S_1. Each point $x_i \in \mathbb{R}^2$ is spaced such that $||x_{i+1} - x_i||_2 = \frac{\sum length(\text{segments}(S_1))}{n}$.

Next the vector $D \in \mathbb{R}^n$ is defined where

$$d_i = \begin{cases} \min(\text{distance}(ss_j, x_i) \forall ss_j \in \text{segments}(S_2)) & x_i \text{ is adjacent to } ss_j \\ \min(||x_i - ss_j[1]||_2, ||x_i - ss_j[2]||_2) & \text{otherwise} \end{cases}. \tag{3.26}$$

The solution comparison function is then defined as

$$f : \mathcal{P}(S) \times \mathcal{P}(S) \times \mathbb{R} \to \mathbb{R} \qquad f(\mathcal{P}(S), \mathcal{P}(S), \mathbb{R}) \mapsto f(S_1, S_2, n) := ||D||_2. \tag{3.27}$$

With a comparison function defined, this function can be used to compute a metric for the closeness of the entire set of solutions. First the most central solution or baseline solution is found. The baseline solution is found by computing the distance between every pair of solutions, then for each solution, the total distance to all other solutions is computed via a sum. The solution with the minimum sum is treated as the baseline solution for the set.

Using this baseline an average solution can be computed. This is done by breaking the baseline solution into n equally spaced points. For each point the minimum distance to every other solution is computed using Equation 3.23. This set of distance provides a dataset to compute the mean and standard deviation of the offsets to the baseline. This process is repeated for every point in the baseline solution.

The results can then be plotted to show the mean solution path and the range representing plus or minus two standard deviations.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Chapter 4

Instance Generation

The optimization instance is the representation of the problem to be solved. The instance contains all the information needed to encode the problem. A classic optimization example is the Traveling Salesman Problem (TSP), this problem involves finding the shortest path that visits all target locations. For some applications of the TSP, such as drilling holes in circuit boards, the physical position of the targets are directly related to the distance between the targets. However, this is not always the case. When considering TSP related to traveling between cities, the travel times are only loosely related to the physical distance between the cities. Other aspects like the road type, and layout have an impact and decouple the travel time from the physical distance.

The complexity of these examples can be abstracted away in the instance definition. The physical distance, travel time, or another metric is mapped to a generic weight value. This weight is computed for every pair of points and the resulting matrix acts as the instance for the problem. With this instance definition, the details of the problem are abstracted away, and the optimization process focus only on solving the TSP problem. This abstraction process makes the optimization simple, however care must be taken in the modeling process to ensure that the instance accuracy represents the problem.

In this thesis the instance was designed to achieve the same abstraction as the TSP example. The complexity of the transit system, is obscured in the optimization instance. This process decouples the instance generation process from the optimization itself. As a result, significant care was taken to ensure that the instance generation process accurately models the transit network, road network and interactions between them. To achieve the accuracy required, models were created that represented the various networks as well as the utility distribution throughout the region. With the modeling methods well-defined, the model could be implemented by populating it with data from different sources.

4.1 Network and Utility Modeling

Developing a high quality model for the transportation networks and the utility distribution is essential to ensure high quality results. Not only does the modeling process impact the quality, the usefulness of any results is also dependent on the modeling choices. The two components that must be modeled are the transportation network and the utility distribution. The transportation network consists of three subnetworks and

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

the interconnections between these networks. The subnetworks, the walking network, the public transportation network and the future bus network are modeled differently to individually reflect the requirements of each network. The modeling of the utility distribution is dependent on user choices regarding the outcome of the optimization task. These user defined goals and the mathematical framework for the utility are combined to create the utility distribution.

4.1.1 Network Model

There are three different networks that are modeled as part of this thesis. The networks that are modeled are the walking network, the existing public transit network and the future bus network. Each of these networks have different modeling challenges and considerations that must be addressed. While it would be possible to model every detail of each network, this level of detail would require input data that is difficult or impossible to obtain. To achieve a compromise that would provide an acceptable amount of modeling detail without becoming overly complex three modeling goals were defined.

1. Model the walking network using real world data
2. Accurately model passenger flow into and though the transit network
3. Model the future bus network with a balance of realism and simplicity

These modeling goals were achieved by employing a multitiered modeling approach. Each of the three networks were modeled using the mathematical construct of a graph. Graphs are a structure that contains vertexes and edges, a vertex is an element in the graph and an edge is a link between two vertexes. In a basic graph, the edges are only used to show the relationship between vertexes, however data can be associated with vertexes or edges to add additional information to the graph. If a weight value is applied to each edge, the graph is then called a weighted graph, this is a very common graph type and is used extensively in shortest path problems and more. In addition to applying a weight to each edge, data can be associated with the vertexes, this vertex information was used to model the walking network.

To model the walking network, the real world paths of pedestrians were encoded into a weighted graph. The street network was mapped into a graph by converting every intersection into a graph vertex, and every road or walking path between intersections into an edge between the respective vertexes. The physical location of each intersection was associated with the respective vertex providing a position in 2D space. Additionally, edge weights were determined by computing a travel time along the route between the intersections. This travel time was computed using a constant average walking speed and the real world path distance.

Constructing the walking graph requires detailed data relating to the road networks. This data is available via the open source mapping service called OpenStreetMap(OSM) [28]. The python package OSMnx [3] was used to extract the road network, as well as walking paths and other pieces of pedestrian infrastructure. OSMnx was also used

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

to parse the dataset and convert it into the weighted graph for the walking network model. The modeling process achieves the first modeling goal. The walking network is fully defined based on real world data. While there are many improvements that can be made to this model, the chosen model achieves a high level of accuracy without significant complications. Some improvements that can be made are detailed in Section 7.2.2.

The model for the public transit network is constructed in a very similar way. The same overarching concept is used, a weighted graph is constructed with vertexes representing transit stations, and the edges are the connections between the stations. This naive constriction method works well with the walking network however there are many shortfalls when applied to the public transit network. The problem with the naive implementation is differences in how real transit riders use the network and how a virtual transit rider can traverse the model of the network. When a transit rider arrives at a station on a bus or other public transit vehicle, the rider has only two choices, continue riding on the line or exit the vehicle at the station. In order to transfer to another line, the rider must incur a delay relating to the required transfer time between transit lines.

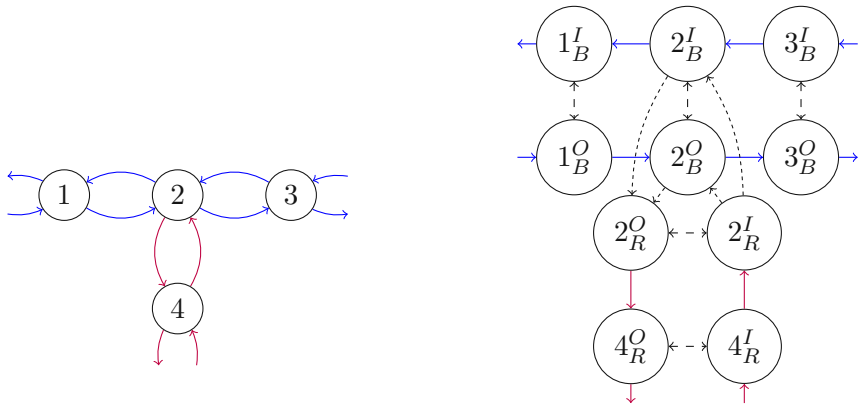
The virtual rider in the naive network does not have to incur any transfer time. The graph does not differentiate between edges that connect to a vertex and as a result a virtual rider is free to choose any edge that is connected to the vertex. As a result, at a station with multiple transit lines, the virtual rider can transfer to any other route without incurring a waiting time penalty. Furthermore, any station that has connections from the same route but in two travel directions, the virtual rider can execute a wait free U-turn in the network. This modeling choice is sufficient when considering walking networks where a pedestrian and chose any path that is connected to an intersection, however it does not accuracy model how transit networks function.

This modeling issue was overcome by increasing the complexity of how a station is modeled in the network. Instead of using a single vertex for each station, multiple vertexes are used to divide the station into multiple co-located stops. These vertexes are uniquely defined by three parameters, the station, the route and the direction. These vertexes are equivalent to modeling each platform of a transit station. The vertexes are then connected to each other with additional edges that have no physical length, but the weighting value represents the required waiting time during a transfer.

An example can be seen in Figure 4.1. This figure shows a T-junction where one transit route continues though the intersection and one route terminates at the intersection station. Figure 4.1a shows the naive implementation of the intersection. Since the same vertex is utilized for both routes, a traversal of this graph can continue seamlessly along the path $4 \to 2 \to 1$ or $4 \to 2 \to 3$ without incurring a waiting time for transferring between the transit routes. In Figure 4.1b all the stations are divided by route and direction. With the layout shown the route $4 \to 2 \to 1$ in the simple graph is now the route $4^I_R \to 2^I_R \to 2^I_B \to 1^I_B$. This adjusted route contains the edge $2^I_R \to 2^I_B$ which is used to represent the transfer time between the red and blue transit routes. Additionally, in the improved layout transferring between any transit line or direction requires utilizing

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

one of the dashed edges. This allows the implementer to set the transfer time between route or between the same route in different directions.



(a) Intersection without transfer edges

(b) Intersection with transfer edges

Fig. 4.1: Example intersection with and without unique vertexes

The last network to model is the future bus network. This is what the optimizer used to select different potential bus stations and form a route between them. To define this network, all the possible bus stops must be identified and added as vertexes. Additionally, the future route could connect between any two bus stops, so the travel time between every pair of possible bus stops must be computed. This results in a fully connected graph, a graph where every vertex connects to every other vertex. To build this graph, the road network information was extracted from OSM data. Using the road network, locations for bus stops can be chosen, and the travel times between the stops can be computed using standard shortest path algorithms. The details of how the bus stop locations are chosen and what defines the routes between them is one aspect of the instance generation process where a lot of complexity could be added. For this thesis a simple approach was taken, however there are many possible refinements detailed in Section 7.2.2.

By linking the different networks together an accurate model of the problem was created. Firstly the walking network and the transit network are joined to create a baseline graph. This graph represents the existing infrastructure. To link these graphs together edges are added between each vertex in the transit graph and the nearest vertex in the walking graph. The length of this edge can be used to model the time that it takes to get into the transit system. This is called station waiting time and is generally half of the route headway on high frequency routes. To connect with the new bus network a similar approach is used. There are two consecutive edges that need to be traversed to reach the bus network. Firstly there is walking time for the distance between the station and the nearest walking vertex. The second edge is the waiting time for the new bus network.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

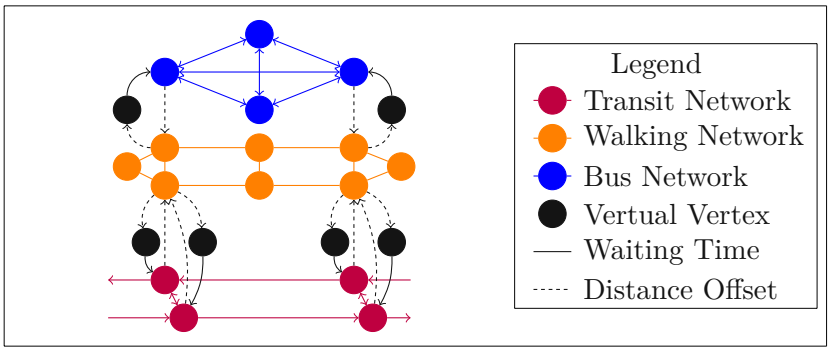


Fig. 4.2: Diagram showing the different network layers.

4.1.2 Utility

The utility of a point in the region is a representation of the usefulness of that location to a potential transit user. The exact utility offered by a location is highly individual and likely unquantifiable, so this value must be modeled. The modeled value is dependent on what assumptions the modeler has on the network users and the intent of the optimization process.

For instance consider two optimization tasks, the first is aimed to searching for a local bus route that could replace dedicated school buses and the second is targeted towards work commuters. For the first task the location of schools should have very high utility whereas office locations should offer no utility. For the second optimization task the opposite is true. Businesses and office locations should have high utility and schools, parks and recreational locations should have low utility.

In order to model the utility in a meaningful way, the intent of the optimization and the transit user goals must be well-defined. For this thesis, the goal of the optimization is to find a bus route that could replace any or all daily trips. The means that the route is not focused on commuters or a specific trip type, but on the general travel habits of the population. In the authors' opinion, a useful transit route is effective for people of all social levels. As a result the potential users of this network is the average population.

With these optimization goals in mind, a utility modeling strategy was developed. The starting point was to find data on different trip types and the frequency of these trip types. This data is available from travel survey data. Using this dataset the different destination types can be identified as well as the weighting between these different types (Sets T and W from Section 3.2.1). With the destination categories defined the set of POI needs to be found, and they need to be assigned to the different categories. To do this the OpenStreetMap data can be queried to find all the locations of a specific type.

By clearly defining the utility computation process in Section 3.2, only the section of the types and POI need to be done by the user. The strong mathematical framework for the remaining computation ensures that there is no ambiguity in the application process.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

4.2 Case Study Selection

The city of Winchester, Massachusetts was selected for analysis by this thesis. The town was chosen because it is the author's hometown, and it has characteristics that are advantageous. What makes Winchester a good optimization candidate is the suburban positioning, existing though town transit, moderate size, good distribution of points of interest and need for additional transit.

Winchester is located in the suburbs of Boston, Massachusetts the capital of Massachusetts and one of the largest cities on eastern coast of the United States. The town is positioned 12 kilometers to the north-west of Boston, has a population of 22,970 and an area of 16.3 square kilometers. The majority of the town is zoned for single family homes with minimum lot sizes varying from 929 to 1858 square meters (10000 to 20000 square feet). The zoning and lot size requirements result in a typical American suburban layout and population density of 1,472.4 residents per square kilometer. In addition to single family zoning, Winchester has other residential building types including, duplexes, apartment complexes and assisted living facilities. This results in a population distribution with moderate variation and locations of high density.

The layout of points of interest is also advantages for the optimization. Winchester has a well-defined town center with four streets that meet at a round-about in the center of the town. The local points of interest include multiple restaurants, shopping opportunities, retail business and office spaces. In addition to the town center, there are significant points of interest along the Main Street. This street extends north from the city center and along it there are approximately 2 kilometers of additional commercial space. This area contains more restaurants and retail as well as automotive businesses and medical office space. The medical space is not limited to the offices on Main Street, there is also a hospital located in the North East corner of the town as well as outpatient and assisted living facilities. Spread across the town there are multiple schools, churches and parks. The diverse set of points of interest and the inclusion of both dense and sparse spacing creates a varied utility distribution across the town. The zoning map adopted in 2009 can be seen in Figure 4.3d.

The town of Winchester also has three existing transit routes that pass though the town. There are two bus lines, the 135 and 350, as well as a regional rail line. The regional rail line travels from the center of Boston, stopping in the town twice and then continuing to the city of Lowell, Massachusetts, around forty kilometers to the north-west of Boston. The bus routes both pass directly though the town and serve as radial connections bring passengers into the Boston Metro area. The 134 bus travels down Main Street and continues towards the south east corner of the town and terminates at the end station of the Orange subway line. The 350 bus only travels along Cambridge Street on the western side of the town. The bus cuts though the town connecting the suburb to Burlington to the end station of the Red subway line. The presence of existing transit is beneficial for the optimization because it allows for any proposed route to connect to the existing transit improving access to areas outside the town. Reference figures of the different bus and regional rail lines can be seen in Figure 4.3.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

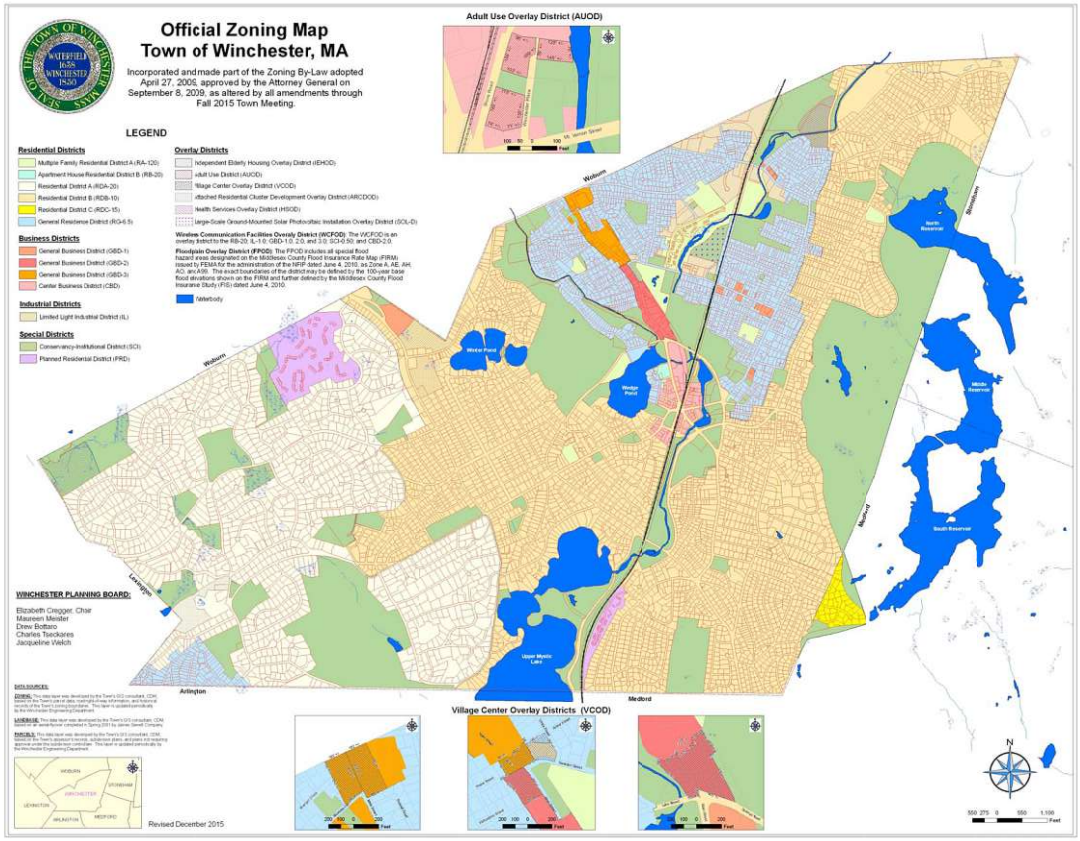
Another aspect of Winchester, MA that makes is a good candidate as a case study is the clear need for transit. There are no existing transit routes that offer a local service to the citizens of the town. Additionally, as mentioned in the Chapter 1 up to 60% of regional rail uses utilize a vehicle for the first mile of the trip [2]. The first mile car use creates an opportunity for the proposed routes to service the needs of commuters into Boston. In addition to connecting to the regional rail, a proposed route could also connect to the existing bus routes allowing for transfers between any combination of the routes. Finally, the author's familiarity with the town allows for the effectiveness and quality of different proposed routes to be qualitatively evaluated.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

(a) Lowell Line
Wedgemare Station

(b) 350 Bus
Cambridge Street opp.
Pond Street

(c) 134 Bus
Main St @ Lake St



(d) Winchester, MA zoning map [9].

Fig. 4.3: Zoning and public transit in Winchester, MA. Photos by author.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

4.3 Graph Generation

The walking graph and the graph of the road network were generated using data from OpenStreetMap and a python package called OSMnx. The generation of the transit graph was done via custom programs that generated graphs from timetable data.

4.3.1 OSM Graphs

The walking and road networks graphs were generated from OpenStreetMap(OSM) data. OSM is a public user contributed database of mapping information. It contains street layouts as well as business, parks, schools and other places. The quality of the dataset is generally quite high, however since it is user generated it is not perfect. The OSM road network data for Winchester, Massachusetts is provided by the Office of Geographic and Environmental Information(MassGIS). As a result confidence in the road network data is very high as it comes from official sources.

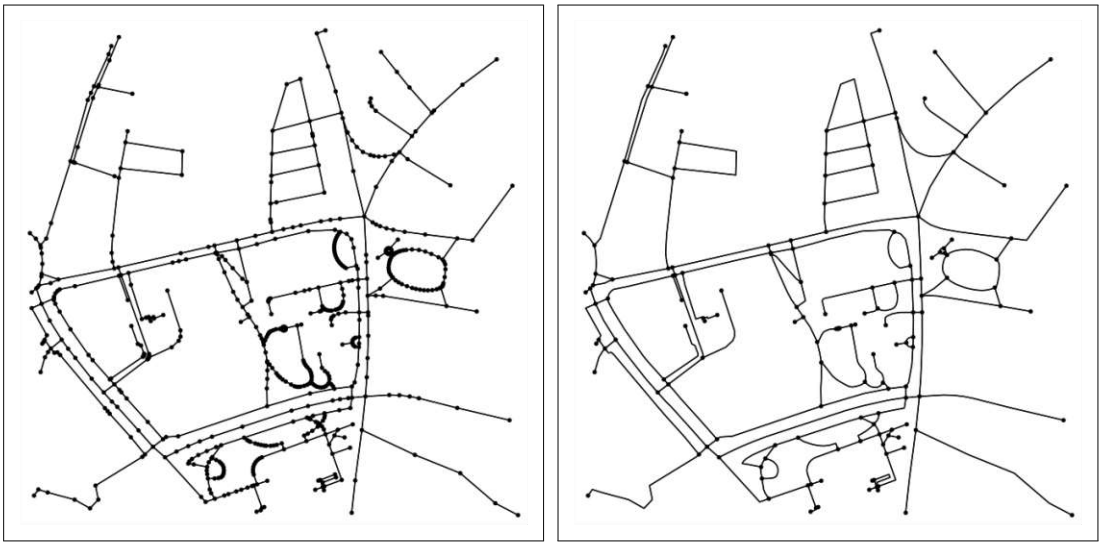
While the OSM data set is extensive, the format of the data is not conducive to modeling as a graph. To do this a conversion between the OSM data and a graph data structure is required. Since this is a common activity there are existing tools that can be utilized. For this project, the python package OSMnx [3] was used to both extract the OSM data using the Overpass API and convert this into a graph.

The OSMnx package functions by the user specifying a specific query. This query allows the user to define a spatial region to search within and incorporate different filters to limit the type of results to return. The spatial region was defined by searching within "Winchester, MA" and the filters were customized from the default settings. OSMnx provides a default "walk" and "drive" filter sets that restricts the scope of returned features to only roadways or walking paths. The default filters were not perfect for the optimization use case because they contained many small roads that added unnecessary complexity to the graph. These small roads were elements like driveways, parking lot aisles and other service roads. Additionally, the walking network contained many small pathways that were eliminated. For instance all the paths inside Wildwood Cemetery are modeled in OSM, it was decided to eliminate these to simplify the resulting graph.

The filters used in the query for the road and walking graph are listed in Appendix A.3. In general, these filters do a few things. Firstly, they filter the road types to only the roads and paths that allow the required mobility type and are available for public use. Additionally, they contain a list of exceptions for path types that are not useful for travel such as "raceway", "abandoned", "construction" and many more types. Finally, the filters eliminate service roads from the returned results. This removes driveways, access roads and some other ways.

In addition to using OSMnx to extract the road data from OSM, the simplification tool provided by OSMnx was used. By default, the graphs generated by OSMnx are composed of many vertexes connected by only straight segments. Roads and paths with curves are modeled by dozens of smaller straight segments that are all connected, this can be seen in Figure 4.4a Additionally, OSM often models divided highways as two separate roads. When these highways intersect it can result in a complicated configuration with

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN



(a) Without simplification **(b)** With simplification

Fig. 4.4: Simplified OSMnx graphs and non-simplified graphs

multiple vertexes for a single intersection. With the simplification tool, all intersections are reduced to a single vertex, divided highways are combined to single edges and the small edges for curves are combined into single larger edges. The differences between a simplified and non-simplified graph can be seen in Figure 4.4.

The final step of the walking and road graph generation process is to add additional refinement to the graphs. The simplified graphs are too simple and do not have the necessarily spatial resolution. The graph vertexes are limited to only the road intersections. As a result the density and spatial uniformity of the vertex positions is poor. To improve the spatial uniformity of the vertexes a refinement method was created that divides any edge over a fixed size into two smaller segments. The method acts iteratively, dividing edges in two until all edges under the specified maximum length. By dividing the edges into smaller segments a higher level of spatial uniformity is created. The uniformity in the vertex locations is useful because the utility distribution will be aggregated to each of the vertexes in the walking graph, and it allows for uniform spacing of the potential bus stops in the optimization graph.

With the OSM graph refined, the data is saved into a CSV format for future use. In addition to saving the vertex and edge relationships that define the graph metadata relating to each vertex and edge is saved. The coordinates of each vertex are stored along with the local position in EPSG:26968, a common reference frame for New England. The edge links are stored by saving the source and destination vertex for each edge. Additionally, the edge length in meters is saved in the data table.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

4.3.2 Transit Graphs

The transit graph is generated from the available public transit scheduling data. This data is provided in the General Transit Feed Specification format (GTFS). GTFS data is a collection of CSV files, each with data relating to different aspects of the transit system. For instance the file `stops.txt` holds information relating to each transit stop in the system, such as the stop name, coordinates, type of access and unique stop id.

The GTFS data format was created to compile all the transit data for a network into a single consumable and maintainable format. With the original intent to be imported into Google Maps, all the physical details of the transit system such as route shape, stop locations and station layout is contained within the specification. Additionally, all the timetable information is available to compute travel time, headway and other transit related parameters. With this information it is possible to construct a graph of the network with real world distances for each link as well as computing the headway of each station.

To generate the transit graph from the dataset two custom Julia packages were created. The first package `GTFSQuerys` is a tool for querying a GTFS dataset and extracting information in a standardized way. The second package `GTFSGraphs`, uses the query tool to create a graph of the transit network. This package is highly flexible and allows for significant user customization. There are existing python packages that can parse GTFS data and create graphs, such as `Partridge` and `Peartree` [31] [4], however these packages are not being actively developed and do not offer the customization required for this analysis.

GTFS Query Tool

The goal of the GTFSQuerys package is to provide a fast and flexible tool to execute generic queries on the GTFS data. The user should be able to specify any set of parameters to query and any set of generic filters to apply. For instance a query of the parameter `stop_name`, with the filter of `route_name = "Red Line"` would return all the station names on the Red Line. Additionally, a second filter of `municipality = "Cambridge"` could be added to only get Red Line stations inside the city of Cambridge.

Efficiently and dynamically parsing GTFS data is a challenging task due to the multiple data tables. Each file is a single data table, however the data required for a useful query usually spans multiple files. To illustrate the challenges with GTFS data a simple example will be used. The goal in this example it to find all the bus arrivals at the station "Cambridge St @ Pond St". Naively one would investigate the `stop_times.txt` table, this table contains all the arrival information, and it is a logical starting point, however the stops are indexed by `stop_id` and not `stop_name` which is provided. To get the `stop_id` the `stops.txt` table must be referenced. While initially this would seem to be sufficient, comparing with the posted schedule on the MBTA website shows that the GTFS data contains many more arrivals. This is because the GTFS data contains multiple route schedules such as weekend, and holiday schedules. To reduce to the correct number of arrivals the `calenders.txt` table must be used to find the currently active

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

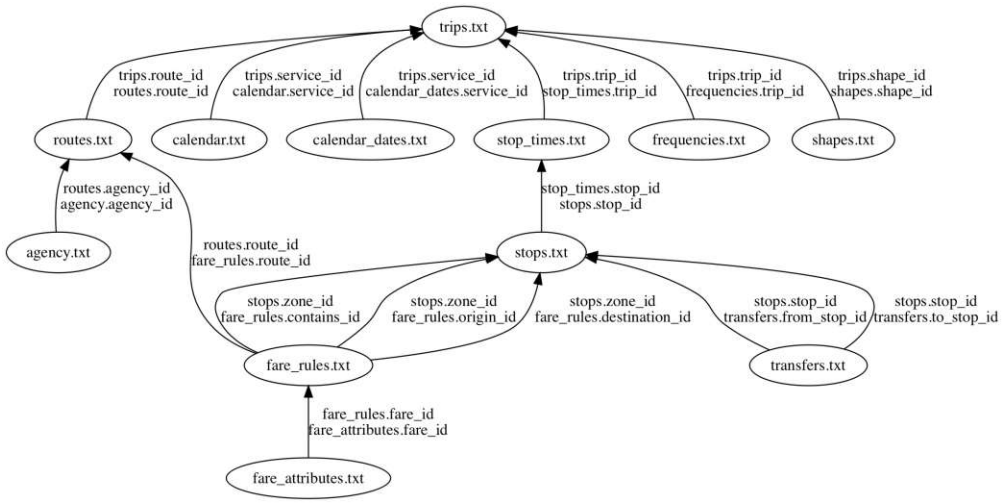


Fig. 4.5: The dependency graph used with the partridge python package [31].

services, however this table does not share any parameters with the `stop_times.txt` table. To link the two the `trips.txt` table must be used. Now there is sufficient information to extract the number of arrivals at the station. This example illustrates that a simple query can span across multiple files, require additional baseline information, and include intermediate files that do not contain any output parameters.

The challenge when parsing the data is identifying the minimal set of files that need to be opened and filtered in order to extract the required parameters and apply the required filters. To do this, a map of the files and the parameters they contain has to be made. In addition to mapping the file names to specific parameters, the shared parameters between the files should be modeled. This information is captured in three data structures. Firstly, there are two dictionaries that map the file names to parameters and each parameter to a file name. The third data structure is a graph. This graph treats every file as a vertex and the shared parameters are edges between each vertex. These data structures are created during the initialization process of the query tool, and they are created dynamically based on what parameters and what files exist in the dataset. The GTFS format is highly flexible with many optional parameters, and capturing this flexibly is an expansion of capabilities when compared to existing packages. Partridge uses the graph shown in Figure 4.5 to model the connections between the files. This is a fixed graph which does not adapt to the provided dataset.

With the groundwork setup, data can be parsed from GTFS. To do this the following steps need to be executed.

1. Identify all requested parameters and parameters required by filters
2. Find the minimal set of files to open to extract all parameters
3. Compute a spanning tree of the required files

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

4. Extract data and combine into the output

The first step to gather all the parameters is very straightforward. The parameters in the query as well as the parameters used in the filters are joined. Additionally, any default filters can be added to the query. The default filters are a set of filters defined in the initiation phase. This can be used to apply a date filter to all queries.

The second step is to compute the minimal set of files to open. Each file contains many parameters and parameters can be shared across many files. To find the minimal set to open, the problem was reduced to a minimal set cover and solved with a linear program. The minimal set cover is a well-known NP-hard problem in the form $Ax \geq b$ where A is a boolean matrix encoding the different sets, x is the solution and b is the points to cover. Each element of the matrix $a_{ij} \in A$ is one if parameter i is in set j [21]. The set covering problem was solved using the `JuMP.jl` package and the `HiGHS.jl` solver.

Even with the required files identified, it is still not guaranteed that they have connections between them. To find the set of files that must be opened, a Steiner tree is computed. A Steiner tree is a tree that connects only a subset of vertexes in a graph. This Steiner tree is then used to compute all the shortest paths from a selected root file to each other file. These shortest paths are converted to a dictionary like structure that encodes the order of files that need to be opened and what parameter should be used to merge them.

The last step is to parse all the files and merge them using an inner join. Starting at the root node the relationship dictionary is used to identify all the child files. The method is then called recursively on the children and the returned value is merged into the output dictionary. This output dictionary now contains all the requested parameters, filter parameters and intermediate parameters. This large dictionary is then selected down to just the requested parameters and any duplicate rows are removed.

Graph Generation

The graph of the transit network is created by the custom package GFTSGraphs.jl. This package is highly flexible and allows the user to adapt the framework to their own use cases and dataset. The core of the framework is three user defined types. The first type L is the unique label for each vertex. This type is typically a tuple of strings that define the set of parameters that uniquely define a route shapes to be added to the graph. The other two types are user defined structs for the vertex data and the edge data. These structs are members of the defined abstract types AbstractVertexData and AbstractEdgeData.

Defining the framework in a generic way allows the programer to store any information they desire in the vertex and edge structs. Each of these structs are used with the MetaGraphsNext package in Julia. This package allows for graph construction using generic types for the label, vertex data and edge data. In this implementation, the label type is L and the vertex and edge data are the assigned types created by the user.

The core of functionality of GTFSGraphs.jl is to add routes shapes to the graph. A route shape is a GTFS term that represents one possible path a transit route would

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

have. For instance, the 350 bus in the MBTA has multiple route shapes. In the outbound direction the primary route is "North Burlington via Mall Road" however there is also the "North Burlington" shape and the "Third Ave" shape. The shapes are alternative routes that offer an express service.

The main function of GTFSGraphs is the method `add!()`, this function uses the predefined label parameters (that uniquely identify a route shape) and any provided filters to find all the route shapes to add to the graph. For each route shape the transit stop information is queried from the GTFS data, additionally the physical path of the route is also extracted. These two datasets are then used to add the route to the graph, adding vertexes for every stop and connecting them with edges. The physical path of the route is used to compute the driving distances between each station.

To include additional information in the graph two other functions are provided. These functions are `apply_edge_funtion!()` and `appy_vertex_function!()`. These functions are generic methods that allow the programmer to extract data relating to a vertex or edge and apply a function. This allows the programmer to compute parameters like station-to-station time, headway and others. In addition to allowing any function the programmer desires, there are a few predefined methods that can be utilized. These methods compute parameters like station-to-station time, headway and dwell times. For these methods to work the user must define the correct parameters in their structs.

Lastly there is a method that connects stations that are close to each other. This connection function is what can facilitate transfers. Without the connections each unique shape is separate in the transit graph. This is by design because it limits a traversal of the graph to only routes that would be possible as a transit rider. To simulate transfers between routes, the links are added. The weight of these links can be based on the average transfer time or the distance between the two nodes.

4.4 Utility Implementation

The utility was implemented using the methods described in Section 4.1.2 and with the math model detailed in Section 3.2. This section will detail the specifics of what data sources were used in the modeling process.

Firstly the population data was taken from a dataset provided by Oak Ridge National Laboratory and their LandScan project [38]. There are multiple datasets provided on their website, however the data for this analysis is the LandScan USA - Night dataset. This data is provided in the form of a raster with 90 meter resolution. There are both night and day assets offered. The night assets were chosen because they more closely represents where people live, whereas the day assets showed high population concentrations around workplaces. This raster was imported into QGIS for manipulation. The gray channel in the raster was trimmed to the region boundary and then converted into a point set. The resulting set of grid points were saved as a CSV file.

The trip data was taken from travel survey data. The National Household Travel Survey(NHTS) is a USA nationwide survey of travel habits [14]. The NHTS has an online tool that allows for extraction of survey parameters of interest. Using this tool

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

the parameter WHYTRP1S was extracted with Census Division as a modifying parameter. The WHYTRP1S is a parameter that represents the trip purpose summary. This parameter has nine categories that the trip could fall into. These categories are shown in Table 4.1.

Tab. 4.1: Travel habits data from the NHTS

Destination Type	**WTTRDFIN [B]**	**Count**	**Type**	**Weight:** W
Home	4.57	680	1	N/A
Work	1.66	202	2	0.2597
School/Daycare/Religious	0.70	93	2	0.1085
Medical/Dental services	0.21	37	2	0.0334
Shopping/Errands	1.78	304	2	0.2777
Social/Recreational	1.17	213	2	0.1822
Transport someone	0.63	84	0	N/A
Meals	0.89	139	2	0.1385
Something else	0.14	25	0	N/A

The majority of the trip purposes are straightforward and have a clear mapping to different points of interest. However, "Transport Someone" and "Something Else" do not have a straightforward way to map to a POI. For this reason these responses were removed from the analysis. Additionally, the trip to home were separated into a different category because these trips are to locations defined by population data. To compute the constant α and the weights in set W, the following procedure was undertaken. Firstly the sum of all trip types that were not removed or "home" is computed(Type 2 in Table 4.1). Next to compute α the ratio between the trips to home and all other trips is computed, with resulting value 0.713. Next the weights for each destination type are computed. These are computed by taking the percentage of each destination type when compared to the total value. These weight values are shown in Table 4.1.

The value used for this analysis is the WTTRDFIN value returned from the query. This value is computed by the NHTS and is a corrected value. The raw survey responses are corrected to account for response rates, demographics, income levels and other considerations. Lastly the data used was limited to only results from New England. This is computed by using the augmenting parameter CENUS_D, or the census district. By apply this limitation we are using data that is more accurate for the target region.

While the travel survey data is very useful, there is no direct connection between the trip types and the POI that can be extracted from the OpenStreetMap database. To link the data sources, a custom mapping between the OSM attribute values and the trip types was created. A list of attributes was extracted from the OSM documentation, and these attributes were then mapped by hand to different trip types based on reasonable assumptions. The full table of attribute mappings can be seen in Appendix A.4, however a subset can also be seen in Table 4.2.

With the mapping to the different categories defined, the POI were extracted using QGIS and automatically generated queries to OSM using the overpass API. Using the

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Tab. 4.2: Subset of the mappings from OSM attribute values to trip categories

Key	Value	Label	Category
shop	*	Shopping	Shopping_Errands
tourism	*	Recreational	Social_Recreational
office	*	Work	Work
leasure	*	Recreational	Social_Recreational
amenity	school	School	School_Daycare_Religious
amenity	restaurant	Meals	Meals
amenity	hospital	Medical	Medical_DentalServices

"*" indicates all possible values

function builder in QGIS, a pipeline was created to automatically query the POI from OSM, process this data into a set of points, transform them to the correct coordinate frame and save them into a CSV file. One challenge with the POI point pipeline is that some results are returned as a polygon, and some are returned as a single point. Parks for instance are returned as a polygon showing the shape of the park, whereas most businesses are represented as a single point. To deal with the polygons, the centroid was computed and used as the POI point. This results in some inaccuracies because the POI point does not necessarily align with the entrance to the building or park. Additionally, when querying the data for schools, the whole school ground is returned as a polygon, as well as a polygon for the school building and a point for the school itself. This results in three points for the single school element. This inaccuracy was accepted because filtering out these issues was deemed unnecessary, and the utility of the schools will be shared equally among these points anyway. In general, there are multiple issues with the POI data however, correcting these issues was not feasible. For correctness, spot checks were undertaken to make sure that the returned results were what was expected.

Finally, the external utility was computed. Many methods were considered for this; however the chosen one was to define a handful of key external points that are connected to the region by public transit. These points would then act as the centers of small external regions to extract the POI data for. The POI for the external region would be mapped back to those points and the points would be connected to the baseline graph by the existing transit network. For these external regions, it was decided that if it was connected by a bus, the region would be a circle of radius 600m and for locations connected by the train the radius would be 1200m. These buffer regions represent a reasonable area that can be reached from the bus or train service. For simplicity not all stops on the bus line were added however all stations on the commuter rail were considered as external points. These points and buffer regions can be seen in Figure 4.6, the regions are overlaid on a map of the Boston suburbs. The regional map was generated in QGIS using the OSM default maps. With the regions set, the automatic POI extraction tool was run on all the external regions and the POI data was saved to different CSV files for future use. A list of all points used as external regions can be seen in Table 4.3.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

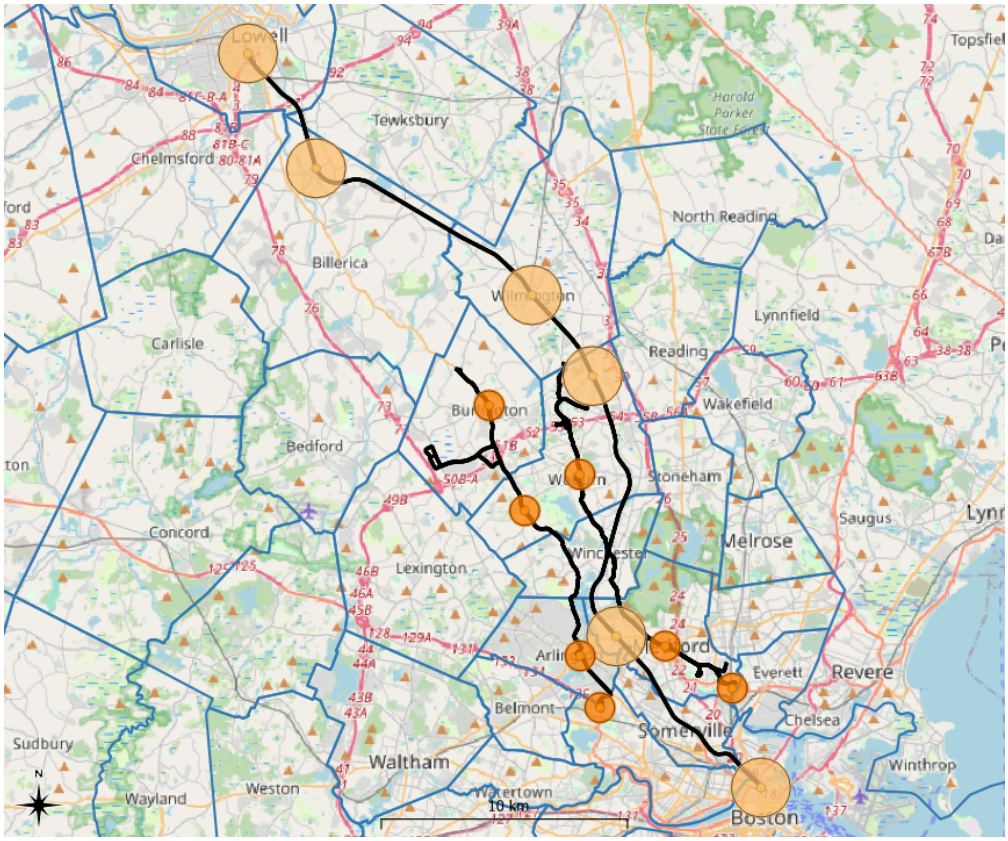


Fig. 4.6: Overview of the external regions.

Tab. 4.3: External region points

Lowell Line	**Bus 350**	**Bus 134**
North Station	Alewife Station	Wellington Station
West Medford	Arlington Center	Medford Center
Anderson/Woburn	Four Corners	Woburn Center
Wilmington	Burlington Center	
North Billerica		
Lowell		

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

4.5 Instance Implementation

To generate the final instances, a tool was created to read a configuration file and generate the instance data based on the provided inputs. Before the instances are created the utility is computed using a similar configuration file method. The configuration file executes the following actions in the order provided. This method allows for multiple configuration files to be created that clearly define what to add to each instance.

1. Walking and Road Graph Generation
2. Transit Graph Generation
3. Modify Transit Graph
4. Add Vertex Weights
5. Trim Transit
6. Merge Graphs
7. Add Bus Stops
8. Generation Optimization Graph
9. Link Graphs and Save

4.5.1 Walking and Road Graphs

Both the road graph and the walking graph are generated using the same function with different parameters. The region to extract the data from was defined to be "Winchester, MA" and the built-in simplification function was used. The query used for the walking network and road network are shown in Section A.3. Additionally, the refinement was set to 150m for the walking graph and 250m for the driving graph. Finally, the data was extracted in September 2024. OSM is constantly updated and as a result differences may exist between utilized graphs and current data.

4.5.2 Transit Graph Generation

To generate the transit graph, the GTFS data was taken from the MBTA. This dataset was taken from the website `transitfeeds.com` [10] and is data from December 12 of 2023. Additionally, a single date was chosen as the representative day when constructing the graph. This date was set to March 21, 2024, and the times were limited between 06:00 and 20:00 hours.

To ensure that no erroneous transfers were possible, three parameters were identified as being required to uniquely mark a station. These parameters are the "stop_id", "route_id" and "direction_id" and these parameters are set as the label tuple for the GTFSGraphs tool. Additionally, the vertex and edge structs were created to store data

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Fig. 4.7: Two photos of the construction on Winchester Center Station. Photos taken by author, August 2024

like the location and coordinates of each stop as well as the length of edges and bus travel times.

To generate the transit graph, the filter "municipality = Winchester" was used to add all the transit routes that pass though the town of Winchester at any point. These transit routes consist of the 350 and 134 bus routes as well as the regional rail Lowell line. These routes offer regular service with an average headway on the order of thirty minutes.

Lastly the metadata was computed and added to the structs. The computed values were station to station time, headway, dwell time, and running time. These parameters were computed based on the schedules in the GTFS data.

The task of modifying transit is required because of ongoing construction in the town of Winchester. The commuter rail station "Winchester Center" was under construction at the time the GTFS data was extracted(See Figure 4.7). This means that the station is not included in the graph by default. It was decided that this station should be added and to do this a custom function was written to modify the transit graph and add additional vertexes for the missing station. The added vertexes, one for each direction, are placed at the correct location based on the GTFS data. This is possible because the station information is still available in the GTFS dataset. The distance to the adjoining station was computed using Google Maps and the service frequency was set to be the same as the neighboring station "Wedgemere".

4.5.3 Additional Data Preparation

There are a number of small actions that are performed on the datasets to prepare them for the instance generation. Firstly the vertex weights are computed using the utility data provided. The vertex weights are computed by finding all the points from the utility dataset that are nearest to each vertex in the walking graph. This is done by finding the

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

walking vertex that is nearest to each point in the utility dataset. The Julia package NearestNeighbors.jl was utilized for this.

Next the transit graph is trimmed to the region of interest. The transit graph contains the full transit routes and are not limited to only the town of Winchester. These routes need to be trimmed to only the stations that are inside the town. Additionally, these route extensions need to be used to connect the external regions back to the main region of interest. To do this only the paths out of the town are kept. To simplify the graph the intermediate stops are removed and only the stations with defined external utility are kept. The entirety of the external utility for each region is then applied to the correct station.

Finally, the walking and transit graphs are merged into one baseline graph. To merge the graphs together, the two data structures are combined. This brings both graphs into one data structure however there are no connections between the walking and transit graphs. To connect them, a link is added between each station in the transit graph and the nearest vertex in the walking graph. The weight of this edge is set to the walking time for the distance between the points. Due to the refinement added to the walking graph the maximum distance can be 150m. However, the majority of the stations are much closer than this because the physical station locations are often close to intersections.

4.5.4 Optimization Graph

The last two steps are related to generating the optimization graph. The graph generation first requires that the possible bus stop be defined. There are many possible methods to define the bus stops. Two methods were considered for this project firstly the simplest method was to treat every vertex in the road graph as a potential bus stop. This method is not ideal because of the way paths are formed in graphs. Similar to the issue detailed with the transit graph. When forming routes between the bus stops using the intersections allows for the bus route to make unrealistic turns.

To partially counteract this, the second method was developed. This method involved splitting every edge in half and placing a new vertex at the midpoint of this edge. These midpoint vertexes are then used as the potential bus stops. This provides a more realistic version of a bus stop were turning around in the middle of street is not possible. It is still possible for U-turns at every intersection, however that is a limitation of the road graph.

With the midpoint of every edge marked as a potential bus stop, the optimization graph is then constructed. Each bus stop is treated as a vertex and the minimum travel times between each pair of vertexes is computed. This is computed by finding the shortest paths on the road network graph. The shortest path is found, and the travel time is computed using a trapezoidal velocity profile model. This model assumes that the bus accelerates up to a maximum speed and then decelerates at a constant rate.

By manipulating the kinematic equations, direct formulas for the travel time can be computed. These formulas are shown in Equation 4.1. For this project the acceleration rate set to be $a_{acc} = 1.2[m/s^2]$, the maximum bus speed is $V_{max} = 15.65[m/s]$, and the deceleration rate was set to $a_{dec} = 1.6[m/s^2]$ [35].

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

$$t = \begin{cases} \frac{d}{V_{max}} - \frac{V_{max}}{2a_{acc}} - \frac{V_{max}}{2a_{dec}}, & d \geq \frac{V_{max}^2}{2}\left(\frac{1}{a_{acc}} - \frac{1}{a_{dec}}\right) \\ \sqrt{2d\left(\frac{1}{a_{acc}} - \frac{1}{a_{dec}}\right)}, & \text{otherwise} \end{cases} \tag{4.1}$$

This results in an optimization graph with 1,603 vertexes and 2,569,609 edges between the vertexes.

4.6 Scenarios and Case Study

The implementation details presented in this section are used for the definition of the Case Study instance. Optimization on this instance is the primary goal for this thesis. It contains the utility from all the POI categories, population datasets, and current transit routes.

In addition to the Case Study instance, three additional instances were defined. These instances are referred to as Scenarios 1 through 3. The goal of these Scenarios was to create simplified instances that could be used to evaluate if the optimization process is working as expected.

The simplest Scenario 1, contains only the eight public schools in the town. It does not contain any population data, existing transit routes, or external regions. All the utility for the region is divided among the eight schools and spread over a large local area. The second Scenario is the same however it uses all the POI from the Schools/Religious/-Daycare category. The last Scenario focus on the Shopping/Errands POI category. This category has many points in it and results a less well-defined utility distribution for the instance.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Chapter 5

Optimization

There is a wealth of different optimization tools and methodologies that could be applied to the optimization problem presented in this thesis. The task of the optimization problem is to search for a new bus route that will maximize the connectivity of the network. The connectivity value is representative of the effectiveness of the transit system and is used as the objective value of the optimization process. To construct the route a combinatorial optimization must be performed, and the basis for the optimization is the bus stops defined in the optimization graph. A subset of these stops will be selected to form a hypothetical future bus route. The complexity of selecting a subset of stops and of computing the objective function makes applying some optimization techniques challenging. Approximation algorithms and mixed integer programming would be difficult to apply to this problem due to the non-linearity of the objective function. The method selected for this problem was a heuristic optimization, this method does not solve for the exact solution, but heuristics generally produce good solutions in reasonable time.

As mentioned in the literature review, heuristics involve generating an initial solution and then modifying that solution incrementally. Each small modification improves makes an improvement and brings the incumbent solution closer to a local or global optimum. To utilize a heuristic optimization, many smaller component parts must be defined and then implemented. These small components are the methods to generate the initial solution, functions that modify the incumbent solution and the overall strategy to apply these functions and methods. In this thesis each of these functions and methods were defined and then implemented in the Julia programming language.

5.1 Heuristics Background

Three major types of heuristics were used in the optimization process, construction heuristics, improvement heuristics and meta-heuristics. Construction heuristics are algorithms that construct a solution based on the information provided in the instance, and they usually build the solution up one component at a time. The next component of the solution can be selected in a greedy way, meaning choosing the solution component that maximizes the increase in objective is selected at each step. Additionally, construction methods are usually randomized in some way as to avoid a deterministic construction algorithm. Multiple design strategies exist relating to construction heuristics, some are designed be simple and fast while other employ more sophisticated algorithms. The

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

simple methods often produce initial solutions that are valid but far from the optimal solution, while a more sophisticated algorithm can produce results that are closer to optimal. The second type of heuristic is an improvement heuristic.

Improvement heuristics are a type of algorithm that takes an existing solution and improves it. These methods are executed after a construction method and are run iteratively. The improvement heuristics are applied iteratively until a *local optimum* is found. At a local optima there is no iterative improvement that can be made to improve the objective value, and the process of applying an improvement method until an optimum is found is called *local search*. To fully define an improvement function and local search process additional details need to be defined. Firstly, a methodology for modifying the solution needs to be created. This methodology is a clearly defined process for making a change to the solution and the process defines a *neighborhood structure*. The neighborhood structure is the set of all possible solutions that are one iteration away from the current solution. For example, consider a solution encoded as a vector of ones and zeros. A simple neighborhood for this solution is call a one-flip neighborhood, this neighborhood is composed of all the solutions where exactly one bit is changed from a zero to a one or vice versa. With initial solution $[0, 0, 0]$ the one-flip neighborhood is $\mathcal{N}_{1-flip}([0, 0, 0]) = \{[1, 0, 0], [0, 1, 0], [0, 0, 1]\}$.

When searching a neighborhood, there are multiple methods for determining when to stop and select the next solution. The methods for stopping are called *step functions*. The most basic step function is called first improvement. Here neighbors are checked until the first neighbor with a higher objective is found, this solution is then selected directly. An extension of first improvement neighborhood is the best improvement neighborhood. With this method all the neighbors are evaluated and the neighbor with the best objective value overall is chosen for the next iteration. Finally, the last step function is the random step function. Here a random neighbor is selected regardless of the change to the objective function.

The final type of heuristic is a meta-heuristic. Meta-heuristics are high level algorithmic frameworks for solving optimization problems. These frameworks prescribe different methods and orderings for applying neighborhoods, step functions and local searches. By having a clearly defined framework for the optimization process, researchers can often make stronger statements about the performance of a meta-heuristic. There are many meta-heuristics to choose from, all with different levels of complexity. For this analysis it was decided to use generalized variable neighborhood search or GVNS.

5.2 GVNS Details

Generalized variable neighborhood search is a meta heuristic that combines a simpler meta-heuristic called variable neighborhood decent (VND) with randomized shaking moves. VND is a meta heuristic that searches for a set of different neighborhoods in an iterative fashion. The idea is that the local optimum of one neighborhood structure is not necessarily the local optimum of another neighborhood. Be searching through multiple neighborhoods one can find a local optimum with respect to all neighborhoods

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

in the search space. The power of VND is that with the right neighborhoods the global optimum can be found, however to achieve a global optimum the designed neighborhoods must span the entire search space. Unfortunately in reality this often results in very large neighborhoods that require significant running time.

Algorithm 1 Variable Neighborhood Decent

Input: $x \leftarrow$ initial solution, $\mathcal{N} \leftarrow \{\mathcal{N}_1, ..., \mathcal{N}_{max}\}$
Output: $x \leftarrow$ local optimum

1: $I \leftarrow 1$ ▷ Index of neighborhood structure
2: **repeat**
3: $x' \leftarrow \{x' \in \mathcal{N}_I(x) \,|\, \text{obj}(x') \geq \text{obj}(x''), \forall x'' \in \mathcal{N}_I(x)\}$ ▷ Local search on $\mathcal{N}_I$
4: **if** $\text{obj}(x') > \text{obj}(x)$ **then**
5: $x \leftarrow x'$ ▷ Store x' if improvement
6: $I \leftarrow 1$
7: **else**
8: $I \leftarrow I + 1$ ▷ Change neighborhood if no improvement
9: **end if**
10: **until** $I > I_{max}$

In Algorithm 1 the pseudocode for VNS is detailed. This function requires an initial solution x, and a set of neighborhood structures $\mathcal{N}$ as an input, it returns a local optimum stored in x. The algorithm starts by selecting the first neighborhood structure, and then enters a loop. On line 3, the local search over the neighborhood is executed. This local search is using the best improvement step function because x' is compared against all the neighbors in $\mathcal{N}_I$. With the local optimum of neighborhood $\mathcal{N}_I$ found the objective is compared with the incumbent solution. If the objective has improved, then x is overwritten and the neighborhood index is reset to one. If an improvement is not found then the neighborhood index is increased. This process repeats until a solution is found that is a local optimum in all neighborhoods.

GVNS extends the VND algorithm by adding a shaking move after the VND process has completed, and a local optimum has been found. A shaking move modifies the solution, similarly to an improvement heuristic, however it changes the solution randomly. The random change usually lowers the objective value and moves the solution away from the local optimum. Once the shaking move has been competed the VND process is restarted to find the next local optimum.

The primary idea behind GNVS is that the locally optimal solutions will have a significant amount of similarly between each other. By making a small random change and restarting the VND process, the GVNS meta-heuristics can theoretically find a second local optimum in fewer iteration then would be required if re-starting from an initial solution provided by a construction heuristic. The exact details of how the GVNS algorithm works can be seen in Algorithm 2

Algorithm 2 requires the same inputs as VND, however an additional set of shaking moves N is provided. In the pseudocode for GVNS one can see on line 1 that an initial

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Algorithm 2 Generalized Variable Neighborhood Search

Input: $x \leftarrow$ initial solution, $\mathcal{N} \leftarrow \{\mathcal{N}_1, ..., \mathcal{N}_{max}\}$, $N \leftarrow \{N_1, ..., N_{max}\}$
Output: x

```
1:  x ← VND(x, N)
2:  repeat
3:      k ← 1
4:      repeat
5:          x' ← x' ∈_R N_k(x)          ▷ Modify x with random shaking move
6:          x' ← VND(x', N)
7:          if obj(x') > obj(x) then
8:              x ← x'
9:              k ← 1
10:         else
11:             k ← k + 1
12:         end if
13:     until k > k_max
14: until Stopping Criteria Satisfied
```

local optimum is found using VND. Next on line 5 a candidate solution x' is created by executing a randomized shaking move. This candidate solution is then improved using VND and compared against the local optimum. If a new optimum has been found then that solution is stored and the shaking moves restarted, if the candidate solution is not a local optimum then the next shaking move is selected by incrementing the index. This inner loop is repeated for all the shaking moves provided. The outer loop repeats until a generic stopping criterion is satisfied. The stopping criteria can be based on the number of iterations, total running time, change in objective value or any other factor.

GVNS is a basic meta-heuristic, however it often provides good results on many problems. It was selected for this project because of the ease of implementation. There are other meta-heuristics that would likely to perform well if not better on this problem. Some of these methods include Large Neighborhood Search and Adaptive Large Neighborhood Search. These meta-heuristics were not implemented due to time and scope constraints.

5.2.1 Construction Heuristics

A construction heuristic is a function the creates an initial solution. These methods are used to create the first valid solution that is the basis for the optimization process. These methods are only limited by the imagination of the user and therefore can have a high level of variety. Construction heuristics can either be deterministic or randomized. Deterministic construction heuristics will always produce the same result based on the same input instance. The algorithms can be complicated approximation algorithm or extremely simple greedy algorithms. Greedy algorithm construct a solution one compo-

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

nent at a time, always selecting the component that will maximize the objective value by its addition to the solution.

While greedy heuristics can be very effective, randomized construction methods are preferred because they allow for variation in the initial solution. By injecting randomness into initial solution, repeated runs of the optimization process will explore more of the total search space. One canonical randomized construction heuristic is to build a totally random solution. Purely random solutions generally have very poor objective values however for most problems they are simple to implement.

The simplicity of implementation was the primary motivation for the generation of a purely random construction heuristic for this thesis. The purely random construction method starts with the user specifying a maximum number of stops in the solution. Based on the maximum number of nodes a random set of vertexes are selected from the optimization graph. This candidate solution is then checked to ensure validity relating to total route length and number of stops. If the solution is invalid then the maximum size is reduced by one and a new set of vertexes are selected. This process repeats until a valid solution is found, and this solution can be used as the initial solution.

The second construction heuristic that was implemented was a method named "Connect Peaks". The idea behind this method is that a good transit route would connect a set of nodes with high utility. Consider a network where all the utility is located at only five vertexes. In this case an optimal route would likely contain all five of these points. The idea behind the "Connect Peaks" construction is that it should be possible to identify these points and use them to construct a basic route. However, simply choosing the X nodes with the highest utility value would produce a deterministic construction method not a randomized one. To avoid creating a deterministic method random vertexes with locally maximal utility were chosen. This was done by first selecting a random point in the region. The random point is then connected to the nearest vertex in the walking graph. The connected vertex is unlikely to be a local utility maximum. The local utility maximum is then found by traversing the walking graph in direction of increasing utility. When the vertex with higher utility than all of its neighbors is found, that vertex is added to the initial solution. The process of finding a random peak is repeated until the initial solution has reached appropriate size.

One shortcoming of this construction method is that there is no consideration to the ordering of the vertexes in the solution. Even with a fixed set of vertexes, different orderings will have different objective values because ordering of the vertexes implies the stop order for the route. There is an ordering of the vertexes that would produce the maximum objective value, however finding this ordering is difficult. There are $\mathcal{O}(t!)$ permutations of the vertexes to consider, checking the objective value of all the permutations would be unfeasible for all but the smallest initial solutions. Another solution method would be to treat the stops as vertexes in a traveling salesmen problem. The TSP is a well-studied problem and a solution could be found using existing algorithms, however the route with the minimal distance may not have the best objective value. The objective value is based on the impact of the route on the entire region and not only one the connection between the stops. While it is expected that there is a correlation

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

between decreasing route distance and increasing objective value, the exact impact is unknown.

Ultimately it was decided to not attempt to solve the optimal ordering of the nodes. The ordering of the nodes in the solution is kept the same as the generation order. This is acceptable because the optimization process includes a method to change the ordering of the nodes.

5.2.2 Neighborhood Selection

When developing the neighborhood structures for this problem, three types of modifications to the solution were considered. The exact details of the solution representation will be detailed in Section 5.3, however the solution needs to define a transit route in some way. For this project the proposed solutions model the transit route by encoding the stop locations in a list of vertexes. This list of vertexes are the bus stations in the optimization graph defined in Section 4.5.4. When considering this set of stop locations, there are three natural types of modifications, changing the order of the stops, changing the stop locations, and adding a stop to the route.

These different types of modifications formed the basis of four different neighborhood structures that were used in this optimization process. The first neighborhood is the shuffle neighborhood. This neighborhood considers shuffling the order of the stops in the solution. A move in this neighborhood considers exchanging the position of any two stops in the solution. With a solution length of t this results in a neighborhood size of $\mathcal{O}(t^2)$.

The second neighborhood structure is the 1-swap neighborhood. This method is focused on exchanging stops in the solution for other stops. The number of potential moves in this neighborhood is large, every stop in the current solution can be exchanged with any of the other potential stops in the problem. If there are n total potential stop locations, then the 1-swap neighborhood has a size $\mathcal{O}(nt)$.

The final two neighborhood structures are related to expanding the solution size. The first expansion neighborhood is the 1-insert neighborhood. This neighborhood considers inserting any of the potential stops into the existing solution at any location. This will expand the size of the solution by one. The second expansion method is the bulge neighborhood. This neighborhood not only adds a stop to the solution, but it also adds a duplicate of the pre-insertion stop. For example, an initial solution composed of three stops $[1, 2, 3]$ would be transformed to $[1, 4, 1, 2, 3]$. In this example the solution "bulges" out to stop 4 and then returns to stop 1. The neighborhood size for both of these methods is $\mathcal{O}(nt)$ because there are t insertion location and n potential stops that can be added. A summary of all neighborhood structures can be seen in Table 5.1.

The majority of the methods defined in Table 5.1 have size $\mathcal{O}(tn)$, since $t << n$ this would make the neighborhoods linearly sized with n. A linearly sized neighborhood is small, however for this problem the objective computation is quite slow. This results in long running times when using the best improvement step function because all $\mathcal{O}(tn)$ candidate solutions must be checked. Running time issues also appear when using the first improvement step function. As the optimization process approaches the local opti-

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Tab. 5.1: Different neighborhoods used as part of the heuristic optimization

Name	Idea	Size	Example
Shuffle	Change order of solution	$\mathcal{O}(t^2)$	$[1,2,3] \Rightarrow [2,1,3]$
1-Swap	Exchange stops in solution	$\mathcal{O}(tn)$	$[1,2,3] \Rightarrow [4,2,3]$
1-Insert	Add a new stop to solution	$\mathcal{O}(tn)$	$[1,2,3] \Rightarrow [1,4,2,3]$
Bulge	Add a new stop to solution	$\mathcal{O}(tn)$	$[1,2,3] \Rightarrow [1,4,1,2,3]$

mum the likelihood of finding an improvement deceases and a larger percentage of the neighborhood must be searched. To circumvent the long running times, the neighborhoods were limited to only the k the closest stops. Instead of considering swapping or inserting all other stops, only the k nearest would be candidates and included in the neighborhood. This reduces the neighborhood size to $\mathcal{O}(tk)$ and makes the search space linear with respect to the solution size t. The reduction in neighborhood size provided a significant speedup in the running time per iteration and the details of how this was implemented is in Section 5.3.2.

5.3 Implementation Details

This section details the programming implementation of the optimization process, related functions and heuristics. The optimization problem was constructed in the Julia programming language using a generic optimization framework called `MHLib.jl`. This framework contains algorithms for multiple meta-heuristics, and a robust scheduling system. The GVNS meta-heuristic is supported by the framework and the scheduling system allows for easy manipulation of global stopping criteria, such as maximum iterations or total running time. Additionally, the framework has built-in logging and analysis tools. To apply the MHLib framework to a specific problem, the user must specify different data structures to contain optimization and instance data as well as the different methods for construction and improvement heuristics. To fully implement the optimizer the user must specify the following,

- Instance input format
- Instance data structure
- Solution data structure
- Methods to perform the following
  - Construction heuristics
  - Neighborhood structures
  - Shaking Moves

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

5.3.1 Instance Definition

The optimization process is agnostic to many of the details of the instance generation process defined in Section 4.5. This means that the scope of information required for the optimization is quite limited. Considerations like the utility definition are important because it effects the meaning of the solution, however the actual optimizer itself only considers the numerical values applied to each vertex and edge. The data saved for the instance was limited to two datasets, the baseline graph and the optimization graph.

The data of the instance is saved in an uncompressed zip file. Inside this file there are four comma separated value files with tables of information. Each graph in the instance is saved as a table with vertex information and a table with edge information. This format was chosen because it allowed for utilization of existing parsing tools, resulting in a single file for the instance and is human-readable with a text editor or Microsoft Excel.

The baseline graph is the graph that contains the walking network merged with the existing transit network (See Section 4.3). This graph also contains a vertex weight based on the utility of each vertex in the walking graph and external regions. The vertex index, weight, and position information is all stored in the vertex table for the baseline graph. The edge table for the baseline graph contains the source and destination index as well as the edge weight. As mentioned in Section 4.5.1 the edge weights for the baseline graph is the travel time in seconds along the respective edge. This could be via transit or via walking.

The second graph in the instance is the optimization graph. The vertexes of this graph are all the potential stop locations in the region. The edges of this graph are the travel time between any pair of stops. This graph must be stored as a complete graph due to the trapezoidal travel times defined in Section 4.5.4. The travel time between two stops is not a linear combination of the edge lengths between these stops and therefore all edges must be stored. The vertex table for the optimization graph contains the vertex index and the position information. The edge table contains the source, destination, and weight information similar to the baseline graph. In addition to these parameters, the vertex table also contains an `offset` value and a `link_index`. The link index is the index of the nearest vertex in the baseline graph. This creates a virtual edge between the optimization graph and the baseline graph. The offset value is the travel time along this edge in seconds. These two graphs spread over four tables define the entirety of the instance.

5.3.2 Instance Data structure

The data related to the instance is stored in an immutable struct data structure. This data structure is a container for all the information related to the instance. The graph data is stored directly in the instance struct however the instance struct also contains a number of different pre-computed values. The details of the instance data structure can be seen in Table 5.2

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Tab. 5.2: All parameters stored in the instance data structure.

Type	Data Type	Size	Use
Value	Integer	1	Vertexes in the walking graph - n_w
Value	Integer	1	Vertexes in optimization graph - n_{opt}
Matrix	Float	$n_w \times n_w$	Walking graph path matrix
Matrix	Float	$n_w \times n_w$	Copy of walking path matrix
Matrix	Float	$n_{opt} \times n_{opt}$	Optimization graph path matrix
Vector	Float	n_w	Utility of each vertex
Vector	Float	n_{opt}	Offset vector
Vector	Integer	n_{opt}	Link Vector
Value	Integer	1	k Value
3D Array	Float32	$n_{opt} \times n_{opt} \times k$	Move Matrix
Struct	N/A	N/A	GPU Datasets

The graph sizes and k value are simple constants that are stored to avoid recomputing the size of the matrices. Additionally, the utility or vertex weight of each vertex in the walking graph is stored as a vector, as well as the link node between the optimization graph and the walking graph, and the associated offset over that link.

The first large pre-computed data structures are matrices with the travel times between any pair of vertexes in the baseline graph and the optimization graph. During the instance loading process the edge weights are loaded into a matrix and the Floyd-Warshall algorithm is applied to compute the shortest paths. This algorithm was chosen because of the simple data structure requirements and straightforward implementation. One matrix is stored for the optimization graph while two copies of the shortest path matrix are stored for the walking graph. One copy acts as an immutable reference and the other is pre-allocated memory for in place operations during the objective computation.

The most complex data structure stored in the instance is the move array. The three-dimensional array acts as a lookup table for the possible insertion moves. When applying the neighborhood size limitation specified at the end of Section 5.2.2 the lookup array is essential to avoid re-computation of the nearby stops. The array is created by iterating over every pair of points in the optimization graph. For each pair of points a vector of potential insertion vertexes is created by enumerating a list the size of the optimization graph and then removing the source and destination vertexes. With the intermediate vertexes defined, the shortest path matrix for the optimization graph is used to compute the total distance from source to intermediate stop to the destination vertex. This is repeated for every identified intermediate vertex, and the results are compiled into a vector. The resulting vector is then sorted and the k nearest stops are saved in the move matrix. This data structure allows for the n^{th} closest insertion stop between any two vertexes to be referenced on $\mathcal{O}(1)$ time, apposed to the $\mathcal{O}(n \log(n))$ time to find and sort all possible intermediate vertexes. The cost of the fast lookup time is an additional $\mathcal{O}(kn^2)$ space requirements. The increased memory requirements is not a concern for this project because the total memory footprint is not a limiting factor.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

The final data structure in the instance struct is the container for the GPU data. This container contains references to data on the GPU. This data is necessary for the execution of the custom kernel for GPU based objective computation. The details of this kernel are explained in Section 5.4.4. The data structures in the instance are large and as a result transferring this data to the GPU takes a significant amount of time. To avoid the data transfers and memory allocations of the GPU, duplicates of the main arrays in the instance are initialized on the GPU. The data sets initialized on the GPU are the two copies of the shortest path matrix for the baseline graph, the vector of vertex weights and the vector of offset values.

5.3.3 Solution Data Structure

The second data structure for the optimization process is a mutable struct for the solution information. This struct contains the instance as well as any solution related information needed by the improvement heuristics. In addition to the instance and solution data, this struct must contain the objective value and an objective validity boolean for the functionality of the optimization framework. The details of the solution data structure can be seen in Table 5.3.

Tab. 5.3: All parameters stored in the optimization data structure.

Type	Data Type	Size	Use
Struct	N/A	N/A	Instance data struct
Value	Float	1	Objective value
Value	Bool	1	Objective validity boolean
Value	Float	1	Score - Objective without penalty terms
Value	Integer	1	Size of solution - t
Vector	Integer	t	Solution vector
Vector	Float	t	Distance vector
Vector	Integer	t	Link vector
Value	Integer	1	k Value
Value	Bool	1	GPU active boolean

The configuration of the solution is up to the programmer, for this problem it was decided to store the solution as a vector of the stops in the route. Each stop in the route is represented by its unique vertex index in the optimization graph. This results in the solution represented by a vector of integers. In addition to the solution vector, the distances to the next stop and the link index are stored for each vertex in the solution. These vectors are convenience data structures to eliminate lookups in the large vectors and matrices.

The remaining variables in the solution data structure are a constant with the solution size, a boolean indicating if the GPU should be used and a floating point value called the solution score. The solution score is the same as objective value in most cases. The cases where it is different are outlined in Section 5.3.5

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

In addition to the struct defining the solution a number of functions need to be defined to customize the functionality of the MHLib framework. The major functions that need to be defined for the solution are the copy functions. These functions define specific copy functionality for the solution structs. When defining these functions specific attention was paid to what datasets are duplicated into new memory addresses, verse when simply the reference to the object is copied. The copy function is used significantly during the GVNS and local search processes, because of this eliminating unnecessary memory allocations is critical. The main step to achieve this goal was to ensure that the instance information is not duplicated when the solution is copied. Because the instance is quite large containing many matrices of size $\mathcal{O}(n^2)$ and one of size $\mathcal{O}(kn^2)$ avoiding copying this is critical. The remaining parameters in the solution struct are duplicated either because they are static types, or the copied version of the solution needs to be able to mutate the values without impacting the original version.

Finally, a few other functions were defined for the solution type. These functions include a validity check `is_valid` that will be detailed in Section 5.3.5, a comparison function `is_equal`, and an initialization function `initalize!`. The comparison function compares the solution vectors instead of comparing the objective values, this is a simpler comparison then recomputing the objective. The standard check function recomputes the objective and does a bitwise numerical comparison to the previous value. The function had to be overwritten because the objective value is computed as a floating point number and is therefor sensitive to floating point errors. Floating point errors are a non-issue when computing on the CPU because the computation order is deterministic. However, when the objective is computed on the GPU there is not a fixed computation order and floating point errors occur. To account for the error the bitwise equality operation was replaced with the Julia function `isapprox`. This function returns true if the compared values are within a specified absolute or relative tolerance. The last required function is `calc_objective`, however this function will be detailed in Section 5.4.

5.3.4 Local Search Framework and Improvement Heuristics

Since MLLib provides only the meta-heuristic framework for GVNS, the user must implement their own code for different improvement functions. These functions must follow a pre-describe syntax in order to function inside the framework, however the functionality of the method is up to the user. When creating the improvement heuristics for this analysis the goal was to have multiple neighborhood structures that could be utilized with any of the three step-functions. To avoid making a unique function for each combination of neighborhood and step function, a generic local search method was created that allowed different neighborhoods to be searched by applying a unique move type.

Local Search

Pseudocode for the local search function is shown in Algorithm 3. This function requires a solution to modify as well as a set of possible moves and a step function. The moves are different custom types that define how to change the solution based on a particular

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

neighborhood structure. Since there are four neighborhoods, four more concrete types were defined. All these concrete types are children of the abstract type `Move`. This allows for the local search function to be defined for the `Move` type, and the user can pass in a move definition for any of the neighborhoods.

The local search function starts by creating two copies of the provided solution. The solution s' will act as the temporary solution to hold the current changes, and s_{org} is an unmodified copy of the original solution. The current best solution is always be stored in the s variable. The second step in the local search is to start a for loop over all the potential moves provided. This for loop is executed over the moves in a random order. Next the `apply_move` function is called. There are different versions of the function defined for the different move types. Julia's multiple dispatch system will automatically utilize the correct version of the function based on the provided variable types.

The `apply_move` function is what modifies the solution. The type of the provided move variable determines what neighborhood is being explored. The data in the move variable defines what, shuffles, swaps or instructions should be executed on the solution. In addition to modifying the solution vector, the supporting variables are also updated in the `apply_move` function. This function returns a true if the move is applied successfully and false if the move cannot be applied. The if statement on line 5 of Algorithm 3 checks that the move was applied successfully and continues to the next move if not. On line 9 of the algorithm the validity of the solution is checked, this process will be detailed in Section 5.3.5.

Next on line 13 of the algorithm the step function is checked. For the random step function the local search should exit after a single move. Finally, the objective of the modified solution s' is compared to the incumbent solution s. In the pseudocode this is done with a simple comparison. In reality due to floating point errors in the objective values, the modified solution needs to be larger than the incumbent and not approximately the same, meaning the solutions differ by more than the square root of the machine precision. If the modified solution is an improvement, then the incumbent is overwritten, and the improvement flag is set. Another check of the step function is made to allow for the local search to exit on the first improvement. If best improvement is being used for the step function, then the loop continues until all the moves have been checked.

Using the generic local search function as the base, multiple different methods can be created easily. The methods for the improvement heuristics all have to have the same input parameters. The functions need the following inputs, a solution data structure to modify, a generic integer parameter, and a results object. The integer input can be used for anything the programmer wants. In this case it was used to select the step function for the local search. This allows each of the improvement heuristic methods to have the same structure. Firstly the input parameter is used to define the step function that is passed into the local search. The next step is to generate the vector of possible moves. For each improvement function the required vector of moves is generated. Finally, the local search function is called using the provided solution, vector of moves and step function.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Algorithm 3 Local Search

Input: $s \leftarrow$ Current Solution, $M \leftarrow \{\text{Moves in } \mathcal{N}\}$, stepFunc $\in \{First, Best, Random\}$
Output: foundImprove $\in \{0, 1\}$

```
1:  s' ← s
2:  s_org ← s
3:  foundImprove ← 0
4:  for m ∈_R M do
5:      if !apply_move(s', m) then                ▷ Apply move m to working solution
6:          s' ← s_org
7:          continue
8:      end if
9:      if !is_valid(s') then                     ▷ Check if solution is valid
10:         s' ← s_org
11:         continue
12:     end if
13:     if stepFunc = Random then                 ▷ Return is using random step function
14:         s ← s'
15:         foundImprove ← 1
16:         return foundImprove
17:     end if
18:     if obj(s') > obj(s) then                  ▷ Check if objective has improved
19:         s ← s'
20:         foundImprove ← 1
21:         if stepFunc = First then              ▷ Return if using first improvement step function
22:             return foundImprove
23:         end if
24:     end if
25:     s' ← s_org
26: end for
27: return foundImprove
```

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

The last functions created are the shaking moves. The shaking move has two steps, firstly a percentage of the existing solution is deleted. The next step is to execute a number of random swap moves. The removal step is straightforward however care was taken to ensure that the entire solution is not removed. Additionally, a check is made to ensure that there are no repeated stops because this can cause errors. To execute the random swap moves the 1-swap improvement heuristic is used, with a random step function. This is repeated the number of times specified by input parameter.

5.3.5 Solution Validity and Shaping

Due to the nature of the objective function, the unconstrained optimal route would likely consist of all possible bus stops. Providing more connections would almost always bring an improvement to the objective value and as a result all stops would be utilized. User set limits were placed on the total number of stops as well as the combined route length to provide a mechanism for the user to limit the scope of the proposed transit route.

The limits on the solution are implemented in a very simple way. A function `is_valid` was created, this function is executed after a modification is made to the solution and checks the validity requirements. If a proposed solution is found to be invalid, then it is immediately rejected from consideration. A better method would be to limit the set of move to only moves that are valid, however this requires significant additional processing when generating the set of moves.

The number of stops is limited by a check on the total size of the solution vector. The limits on the total time required by the route is more complicated. Firstly, the user must set the dwell time of each stop. The dwell time is the duration that the transit unit is stationary at each bus stop. For this analysis the dwell time was set to 20 seconds because it is a reasonable value from literature [36]. The total route time is then computed by summing the station-to-station time for each segment of the route and then adding this to the product of the solution size and the dwell time.

$$t_{total} = \sum_{s_i \in S} t_{s2s}(s_i, s_{(i+1) \bmod |S|}) + t_{dwell}|S| \tag{5.1}$$

In addition to adding constraints on the validity of the solution, functionality was added to create a penalty on the objective value for increasing solution length and size. The objective function was modified to add penalty terms that are scaled off of the number of stops and the total route length. The idea behind these penalty terms is that they will shape the optimal route into a form that is similar to what a transit planner would design. The scope of the shaping constraints and penalties that could be added is massive and this project was not able to explore them all. Application of route shaping was limited to the simple linear penalties on the total route length and number of stops. More details of the penalty terms can be found in Section 6.2.4

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

5.3.6 GVNS Setup

The last step in utilizing the MHLib is setting up the GVNS algorithm and putting together the custom defined pieces. To set up the GVSN algorithm the instance and solution objects are created based on the specified instance. Additionally, numerous parameters need to be set by the user. These parameters include the total number of iterations, total running time, limits on the number of stops and route length, the k value that limits the neighborhood size and the multipliers for the penalty terms.

With the parameters defined, and the solution object created, the GVNS algorithm can be defined. To create the algorithm, the construction heuristic, improvement heuristics and shaking moves need to be selected. Multiple configurations for these heuristics were tested as part of the analysis. However, the most commonly used configuration was to generate the initial solution using the Connect Peak constructor with the number of stops set to one quarter of the total max solution size. Next the set of improvement methods was defined to be the shuffle move, one swap and one insert methods. This execution order was chosen because the shuffle move would allow for the best stop ordering to be found first for each candidate solution. Next the swap move was chosen before the insert because this limits the solution size. Lastly the one insert method was used to find the best stop to add to the solution.

5.3.7 Logging Methods

The final modification that was made to the MHLib system was an improvement to the logging system. While the framework provided extensive capability for logging each iteration and the resulting objective value, there is little flexibility in the logging system. The current logging system provides enough capability to see how the objective value is changing over the course of an optimization run however it does not provide any insight into the form of the solution. In order to see how the route shapes changed during the optimization, a more detailed logging system was required.

This detailed logging system was made in a generic way by modifying the MHLib codebase to allow for custom user created loggers. To do this a new logging module was created that implements a basic logger type and a `get_logger` function. The basic logger allows the user to set the `'-ofile'` input parameter to define a file to save the output messages to. In addition, the user has the option to override the `get_logger` function to return a custom logger of their design. This allows for the iteration data to be saved into different file formats, and to store custom information.

A custom defined logger was programmed and used to save the iteration data to a `.json` file. The logger saves the iteration information including the current solution value to a new row in a json vector. The logger also allows for the solution details to be saved at every iteration without impacting the standard logging output on the terminal. These JSON files are then parsed using existing Julia packages for post-processing and visualization.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

5.4 Objective Computation

The algorithm and function used for the objective computation is one of the most important elements to the performance of the optimization. With neighborhoods of size $\mathcal{O}(tn)$ and $t \simeq 20, n \simeq 1500$ the objective can be computed on the order of thirty thousand times per iteration. This means that small improvements to the objective computation will have a large impact on the running time of the optimization as a whole. To compute the objective effectively a lot of attention was given to the structure of the objective computation algorithm. A common performance improvement is to use delta evaluation to compute the objective, this was considered in depth but deemed to be infeasible, see Section 7.2.3. Finally, benchmarks were used to evaluate different performance optimizations to the objective function.

5.4.1 Standard Algorithm

The general formula for the objective value is detailed in Section 3.1 and Equation 3.4. This equation requires the utility value for each point in the region and the travel time between every pair of vertexes. The utility data is provided in the instance, however the travel times must be re-computed for every candidate solution because the distance matrix changes depending on the chosen bus route.

Computing the minimum travel time matrix is a well-known problem in graph theory and is called the all pairs shortest paths problem. A naive solution to this problem is to directly compute the shortest path between each pair of points directly using a standard shortest path algorithm. This naive algorithm has running time $\mathcal{O}(V^3 \log(V))$, where V is the number of vertexes in the graph. This running time is far from optimal because established all pairs shortest paths algorithms such as Floyd-Warshall and Dijkstra have running times $\mathcal{O}(V^3)$ and $\mathcal{O}(V^2 \log(V))$ respectively [7] [8].

For the objective value computation, a running time of $\mathcal{O}(tV^2)$, where t is the solution size, can be achieved by exploiting the nature of the problem. In the basic case where there is an empty solution, the objective value is computed based on the travel times in the baseline graph only. When a candidate solution is applied, the transit route acts like a set of shortcuts between some vertexes in the baseline graph. When recomputing the shortest paths with these shortcuts, only paths that utilize these shortcut vertexes need to be updated. Since there are no changes to the baseline graph, we know that any shorter path must be via one of the nodes in the transit route. Knowing this, when starting the baseline graph shortest path matrix D, all the improvements for a given solution S can be computed with Equation 5.2

$$D_{i,j} = \min(D_{i,j}, D_{i,k} + D_{k,j}), \forall k \in S, \forall i, j \in V \tag{5.2}$$

From Equation 5.2 one can clearly compute the improved running time of $\mathcal{O}(tV^2)$ where $t = |S|$.

While the idea of the objective computation algorithm is simple the implementation is not. For the minimum computation in Equation 5.2 to be valid the path distance for $D_{i,j}$

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

and $D_{k,j}$ must also be the minimum distance. This means that the minimum distance must be updated in a specific order for the computation to be valid. Pseudocode for the objective computation is shown in Algorithm 4. This algorithm requires four inputs, D is the baseline shortest path matrix, T is the shortest path matrix for the transit network, L is the set of indexes that link the transit nodes with the baseline nodes and O is the link distances. Lines 1-5 of Algorithm 4, write the transit matrix into the baseline distance matrix. The for loops on lines 6-12 then update the distances from each transit stop to any other vertex. These loops are executed first to ensure that the travel time from any of the transit stop is already updated to the minimum distance. On lines 13-19 the remaining nodes are updated, here all the nodes that are not in set L have the shortest path distances updated. From the pseudocode in Algorithm 4 one can clearly see the $\mathcal{O}(tn^2)$ running time, due to the triple for loops, two of size n and one of size t.

Algorithm 4 Update Shortest Paths

Input: $D \in \mathbb{R}^{(n,n)}, T \in \mathbb{R}^{(t,t)}, L \in \{\mathbb{Z}^t | l \leq n \quad \forall l \in L\}, O \in \mathbb{R}^m$
Output: $D|D_{s,d}$is the shortest path utalizing T

1: **for** $s \in \{1...t\}$ **do**
2: **for** $d \in \{1...t\}$ **do**
3: $D_{L_s,L_d} \leftarrow T_{s,d} + O_s + O_d + waitingTime$
4: **end for**
5: **end for**
6: **for** $s \in L$ **do**
7: **for** $d \in \{1...n\}$ **do**
8: **for** $k \in L$ **do**
9: $D_{s,d} = min(D_{s,d}, D_{s,k} + D_{k,d})$
10: **end for**
11: **end for**
12: **end for**
13: **for** $s \in \{\{1...n\} \setminus L\}$ **do**
14: **for** $d \in \{1...n\}$ **do**
15: **for** $k \in L$ **do**
16: $D_{s,d} = min(D_{s,d}, D_{s,k} + D_{k,d})$
17: **end for**
18: **end for**
19: **end for**

To fully compute the objective value, Algorithm 4 must be executed in addition to computing the shortest path between all the transit stops in the network and the final objective value must be computed. Both of these computations add to the total running time but not the overall computational complexity. When computing the travel time matrix for the transit nodes T, care must be taken because the solution can contain duplicate transit stops. These duplicates must be removed and not incorporated into

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

the distance matrix T. The repeated stops are removed by adding the additional edges to the adjacency matrix, but not an additional vertex. With the adjacency matrix populated, the shortest paths can be computed using the Floyd-Warshall Algorithm. The computation of the shortest path matrix for only the transit network has a running time of $\mathcal{O}(t^3)$. The final objective computation is completed using Equation 3.4, the equation requires two nested for loops both of size n and as a result the running time is $\mathcal{O}(n^2)$. The computation of the two shortest path matrices, and the final objective value results in an overall running time of $\mathcal{O}(tn^2+n^2+t^3)$, the running time is dominated by the tn^2 term for the full shortest path update. The actual running time of the computation can be improved via performance optimizations, these performance changes are detailed in Section 5.4.3.

5.4.2 Delta Evaluation Algorithm

Delta evaluation is a very powerful tool that can drastically decreases the computational requirements of updating the objective value. Instead of recomputing the objective value from scratch on each iteration, only the change in objective value is computed. The change in objective is then used to increment the objective by the required amount. Again consider a solution that is encoded by a vector of ones and zeros, with an objective value that is the sum of all ones in the solution. Computing the objective value from scratch requires $\mathcal{O}(n)$ checks to consider every bit in the vector. However, if only one bit is changed when exploring a candidate solution in a 1-flip neighborhood, the objective value can simply be incremented or decremented according to the change. This can be executed in $\mathcal{O}(1)$, which is a huge improvement.

Delta evaluation was considered for this problem however it was not able to be applied. The challenge in applying delta evaluation is the scope of the updates required. When a solution is updated, the change can have wide sweeping effects on the number of the shortest paths that utilize the new bus route. Significant effort was put into developing algorithms to detect the set of the shortest paths that are effected by a change to the solution however no meaningful progress was made. Fundamentally, a solution that has a high objective value will result in a large percentage of the shortest paths utilizing the transit network and since the future bus route is formed as a single route, any change to the solution will result in at least half of the shortest paths requiring an updated distance value.

A second complication to the use of delta evaluation is the non-linear relationship between the distances and the objective value. Because the objective value is inversely proportional to the length of the shortest path between any pair of points, changes to any segment in that path require the contribution of the full path to be recomputed. Details of this issue are explained in 7.2.1.

5.4.3 Computational Performance Improvements

Since the objective can be computed as many as thirty thousand times per iteration, improving the running time can provide a huge speedup in the overall computational

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

time. An initial speedup was achieved via the algorithm design in Section 5.4.1, however there are many programmatic speedups that were implemented to achieve a higher level of computational performance.

The total running time for the objective computation is $\mathcal{O}(tn^2+n^2+t^3)$, this running time is dominated by the shortest paths update and the objective computation steps having running times of $\mathcal{O}(tn^2)$ and $\mathcal{O}(n^2)$. Additionally, the both of these operations have very low computational complexity with the shortest path update requiring four memory references for two operations and the objective computation requiring two memory access for two operations. The computational complexity of the overall computation can be improved because the memory data used in both calculations is the same. Both the updates to the shortest path matrix and the objective computation utilize the same distance values. By combining these two steps into one function, the computational complexity can be increased and the overall running time improved. This combined algorithm is shown in Algorithm 5 and 6.

Multiple performance optimizations were incorporated into Algorithm 5 and 6, the primary change being the combination of the shortest path update and objective computation. The other performance improvements were changes to the loop ordering, updating matrix indexing, branch elimination and variable pre-allocation. Due to the increased complexity of the algorithm the function is broken into two different algorithm blocks. The high level function is shown in Algorithm 5, this pseudocode is responsible for copying the transit matrix into the shortest path matrix and for the path updates to the transit nodes L and the remaining nodes. The sub-function 6, updates the shortest paths for a single vertex and returns the connectivity of that vertex.

The change to the loop ordering enabled the majority of the improvements to the algorithm. In the basic algorithms the for loops to compute the shortest paths are iterated in the order $s \to d \to k$, looping over the source vertex, the destination vertex and then the intermediate vertex. In the improved algorithm the loop order is $s \to k \to d$, this allows for two major improvements. Firstly, the distance from the source to the intermediate vertex can be cashed in the middle loop, this is the offset value initialized on line 7 of Algorithm 6. This eliminates a memory access from the innermost for loop. Additionally, by looping over the destination index last, vectorized instructions can be utilized. Vectorized instructions, also known as single instruction multiple data or SIMD, execute the same operation on multiple data values at the same time, with modern processors utilizing up to 512 bits of data at the same time.

The improvement from SIMD instructions is compounded by the change to the indexing of the distance matrices. In the standard algorithm the matrix the row defines the source index and the column defines the destination matrix. In the improved algorithm this is reversed. The reversal is because Julia uses a column major matrix format. This means that each column of data in the matrix is stored continuously in memory. By changing the matrix to store the destination in the columns, the data being accessed is adjacent in memory, this means that the cache lines are fully utilized, and the data can be sent directly into the SIMD instructions.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

By combining the path update and the objective computation, $\mathcal{O}(n^2)$ memory accesses are eliminated, providing a large speedup because $n \simeq 2500$. This change can be seen in Algorithm 6, lines 5 to 13 updates the shortest path matrix except for the last transit vertex $L_{|L|}$, when updating the shortest paths using the last transit vertex the objective is also computed on line 21. These updated distance computations also utilize the final improvement to the algorithm. Instead of checking the difference in the path lengths using a call to a minimum function or an if statement, a brachless statement is used. This branchless comparison seen on line 11 and 19 of Algorithm 6, is functionally equivalent to an if statement however the exact same instructions are executed on every iteration of the for loop. The change produces a pipeline improvement, where the CPU can predict what instructions are upcoming and prepare the correct data.

Algorithm 5 Compute Objective

Input: $D \in \mathbb{R}^{(n,n)}, T \in \mathbb{R}^{(t,t)}, L \in \{\mathbb{Z}^t | l \leq n \quad \forall l \in L\}, O \in \mathbb{R}^m, M \in \mathbb{R}^n$
Output: $obj \leftarrow ObjectiveValue$

```
1:  for s ∈ {1...t} do
2:      for d ∈ {1...t} do
3:          D_{L_s,L_d} ← T_{s,d} + O_s + O_d + waitingTime
4:      end for
5:  end for
6:  obj ← 0
7:  for s ∈ L do
8:      obj ← obj + UpdateColumn(s, D, L, M)
9:  end for
10: for s ∈ {{1...n} \ L} do
11:     obj ← obj + UpdateColumn(s, D, L, M)
12: end for
```

To evaluate the performance of the different algorithms, a simple benchmark was devised. Three version of the objective computation function were implemented and tested. The first version follows the structure of Algorithm 5 with no performance improvements. The second algorithm tested was the optimized using the changes mentioned in this section. The last function that was benchmarked is the GPU based implementation. This method is fully detailed in Section 5.4.4.

Tab. 5.4: Objective computation benchmarking results.

Algorithm	Seconds Per Iteration
CPU - Naive	0.2629
CPU - Optimized	0.01788
GPU - Optimized	0.001248

The results in Table 5.4, show the benefits of the programming improvements defined in this section. The change from the basic algorithm to the optimized algorithm is

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Algorithm 6 UpdateColumn

Input: $s \in \mathbb{Z}, D \in \mathbb{R}^{(n,n)}, L \in \{\mathbb{Z}^t | l \leq n \quad \forall l \in L\}, M \in \mathbb{R}^n$

1: $obj_{col} \leftarrow 0$
2: $offset \leftarrow 0$
3: $a \leftarrow 0$
4: $b \leftarrow 0$
5: **for** $i \in \{1...(|L|-1)\}$ **do**
6: $\quad k \leftarrow L_i$
7: $\quad offset \leftarrow D_{k,s}$
8: $\quad$ **for** $d \in \{1...n\}$ **do**
9: $\quad\quad a \leftarrow D_{d,s}$
10: $\quad\quad b \leftarrow D_{d,k} + offset$
11: $\quad\quad D_{d,s} \leftarrow (a \leq b)a + (b > a)b$
12: $\quad$ **end for**
13: **end for**
14: $k \leftarrow L_{|L|}$
15: $offset \leftarrow D_{k,s}$
16: **for** $d \in \{1...n\}$ **do**
17: $\quad a \leftarrow D_{d,s}$
18: $\quad b \leftarrow D_{d,k} + offset$
19: $\quad D_{d,s} \leftarrow (a \leq b)a + (b > a)b$
20: $\quad$ **if** $s \neq d$ **then**
21: $\quad\quad obj_{col} \leftarrow \frac{M_d}{D_{d,s}}$
22: $\quad$ **end if**
23: **end for**

a 93.1% reduction in the running time per iteration, or a 14.7 times speedup. This improvement is due to the utilization of the correct matrix indexing and the resulting benefits from cache line optimization and SIMD instructions. The last improvement was to adapt the code to run on the GPU. When the GPU is utilized a speedup of 14.3 times or 93.0% was achieved. This speedup is due to the processing efficiency of the GPU as apposed to algorithmic changes.

5.4.4 GPU Version

Even with the objective computation process optimized as stated in Section 5.4.3, the overall optimization running time was not sufficient for the scope of analysis that was to be performed. To achieve a larger speedup in the objective computation processes the code was ported for computation on the GPU. This conversion required additional variables to be added to the Solution and Instance data structures as mentioned in Section 5.3.2 and 5.3.3. Additionally, the objective computation algorithm needed to be updated slightly to account for parallel code execution. The main function to compute the objective via the GPU can be seen in Algorithm 7 this function calls a number of

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

GPU kernels. One GPU kernel cannot be used because different steps in the computation process must be executed in serial order and using different kernels is the best method to ensure all threads have finished work.

Algorithm 7 Objective Computation GPU

Input: D, D_{org}, T, L, M
Output: $res \leftarrow ObjectiveValue$
1: $resetDistances(D, D_{org})$
2: $writeTransit(D, T)$
3: $updateTransitNodes(D, L)$
4: $res \leftarrow 0$
5: $updateAndCompute(D, M, L, res)$
6: **return** res

Each line in Algorithm 7 except line 4 is a call to a separate GPU kernel. The *resetDistances* kernel resets the working matrix D with the baseline distance matrix D_{org}, the kernel *writeTransit* writes the transit matrix T into the distance matrix D, the kernel *updateTransitNodes* updates the shortest paths for the transit nodes only, and the kernel *updateAndCompute* updates the remaining nodes and computes the objective value *res*. What one can see from this function is that only the objective value variable is created on the GPU and copied back to the CPU. What is not obvious is that the vector L and matrix T are also copied to the GPU as part of the function call. This results in $\mathcal{O}(t^2)$ variables copied to the GPU which is a very reasonable amount of data and much less than the size of the distance matrix, $t^2 << n^2$. The remaining arrays are stored on the GPU from the initialization process.

The GPU kernels *resetdistances* and *writeTransit* are very straightforward. These are simple kernels where each thread writes to a single value in the array. The thread ID is used to compute the index in the array and the correct value is copied into that location.

The kernels *updateTransitNodes* and *updateAndCompute* shown in Algorithm 8have the same structure. They both mimic the structure of the serialized version. The major difference is that the for loops over the vertex pairs are eliminated, and a thread is utilized for each source and destination pair. Each thread operates a for loop that checks each of the transit vertexes to see if there is a shorter route that uses the transit network. This minimal value is then written back into the distance matrix. An additional difference is the computation of the objective value. This is only done in the *updateAndCompute* kernel. Each thread computes the contribution of its respective distance value to the overall objective. To compute the final objective value, warp shuffles are used to aggregate the objective value for each warp. After the values are summed across each warp, atomic operations are used to sum the objective into a single variable. Specific details of the algorithm can be seen in Algorithm 8.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Algorithm 8 UpdateAndCompute-GPU

Input: $res \leftarrow 0, D \in \mathbb{R}^{(n,n)}, L \in \{\mathbb{Z}^t | l \leq n \quad \forall l \in L\}, M \in \mathbb{R}^n$
Output: $res \leftarrow ObjectiveValue$ ▷ Overwritten in place

1: $i \leftarrow (Block_x^{idx} - 1)Block_x^{dim} + Thread_x^{idx}$
2: $totalThreads \leftarrow Grid_x^{dim} Block_x^{dim}$
3: $res^{local} \leftarrow 0$
4: $a,\ b,\ s,\ d \leftarrow 0$
5:
6: **while** $i <= n^2$ **do** ▷ Computes the shortest path and objective for one pair (d, s)
7: $d \leftarrow \left\lfloor \frac{i-1}{n} \right\rfloor + 1$
8: $s \leftarrow i + n \left\lfloor \frac{i-1}{n} \right\rfloor$
9: $a \leftarrow D_{d,s}$
10: **for** $z \in \{1...t\}$ **do**
11: $k \leftarrow L_x$
12: $b \leftarrow D_{k,s} + D_{d,k}$
13: $a \leftarrow (a <= b)a + (b > a)b$
14: **end for**
15: $D_{d,s} \leftarrow a$
16: $res^{local} \leftarrow res^{local} + (s \neq d)\frac{M_d}{a}$
17: $i \leftarrow i + totalThreads$
18: **end while**
19:
20: $n \leftarrow \frac{WarpSize}{2}$ ▷ Compute objective for one warp
21: **for** $n > 0$ **do**
22: $res^{local} \leftarrow res^{local} + shuffleDownSync(FULLMASK, res^{local}, n, WarpSize)$
23: $n \leftarrow \frac{n}{2}$
24: **end for**
25:
26: **if** $LaneID() = 1$ **then** ▷ Atomic add between warps
27: $AtomicAdd(res, res^{local})$
28: **end if**

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Chapter 6

Results

This chapter presents the computational results of this thesis starting with the final routes generated for the case study and a set of scenarios. In order to have confidence in these results, various hyperparameters were tuned using parametric studies. The hyperparameters tuning process involved testing dozens of different parameters configurations to understand each parameter's impact on the resulting solution. The specific set of hyperparmeters used for each analysis is displayed in a table below each figure.

All the results presented in this chapter were computed on a server provided by the Institute for Transport at the Technical University of Vienna. The specifications for the server are detailed below. Additionally, all the code was written in Julia 1.10.5, and more than 5018.37 CPU hours were used for the computation of intermediate and final results.

- CPU: 2x Intel Xeon 5218R @ 2.10 GHz
- Memory: 128GB DDR4
- GPU Details
  - Model: GeForce RTX 3060
  - Memory: 12288 MiB
  - GPU Driver: 535.183.01
  - CUDA Version: 12.2

Since meta-heuristics utilize randomness as part of the optimization process, any single solution may not be representative of the optimization process as a whole. To ensure that an outlier is not reported, the common practice is to repeat the analysis a number of times to gather a set of solutions. For this thesis, each analysis was repeated thirty times to create a statistical sample. The set of solutions are then considered as a whole. Comparing multiple routes in 2D space is a challenging problem, and to address it the comparison method described in Section 3.4 was used. The route comparison was applied to the set of solutions and used to compute the average route as well as the variation of the solution set with respect to that average. All the figures in this chapter display the average route as a solid red line and the dashed red lines represent two standard deviations in the route shape. Finally, the utility applied to the regions for each analysis is also plotted with the routes. The blue represents low utility while orange and red colors are higher utility.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

6.1 Scenarios and Case Study

The case study for this analysis is the town of Winchester, Massachusetts. The town is located in the suburbs of Boston and covers just over 16 square kilometers. The instance for the case study follows the details laid out in Section 4.5, and it applies all the data sources available. The full set of POI categories, population data, travel habits and external transit connections are used to create the instance. In addition to the case study instance, three scenarios were developed to evaluate the performance of the optimization process.

The scenarios were defined to create simplified instances where the shape of the optimal solution is clearer. Data sources were stripped out of the case study instance to simplify the utility distribution. The Scenario 1 is focused only on the public schools in the town. Only the eight public schools are considered as POI, and no other data sources are used. The scenario does not include any existing transit, population data, or other POI categories. The goal for this instance was to define a set of points that the optimal route should definitely pass though.

The second scenario expands upon Scenario 1 by including the entirety of the School-s/Religious/Daycare type. This POI category includes the public and private schools as well as the multitude of churches in the town. This limits the utility to a larger set of points than used in Scenario 1. The set of points is large enough and spread out enough that the optimization will have to select a subset to connect with the new bus route.

The final scenario, Scenario 3, uses the Shopping/Errands trip type. This POI dataset has numerous points spread across the center of the town and along Main Street to the north of the town center. The utility is no longer discrete points, the smoothing function create a smooth distribution across the areas with a lot of shopping and other stores. The goal for this scenario was to evaluate how the optimizer performs when the utility distribution is more ambiguous.

6.1.1 Case Study Results

Figure 6.1 shows the resulting average route for the case study of Winchester, Massachusetts. The route forms a flower pedal shape with three distinct loops or pedals originating from the center of the town. These pedals extend to regions of the town with higher utility and connects them back to the town center. In general the thirty iterations produced very similar routes. There is some deviation in the route shapes however all solutions formed the three pedal shape with the pedals following roughly the same layout.

The transit route show in Figure 6.1, contains three loops or pedals, one to the north, one directly west, and one to the south and west. All three pedals converge at a potential stop location next to the Winchester Center regional rail station. This center point is expected because a connection to the regional rail provides a huge amount of utility from the access to the external regions. Additionally, it is logical that Winchester Center is chosen verse the Wedgemere Station because of the local utility in the town center.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

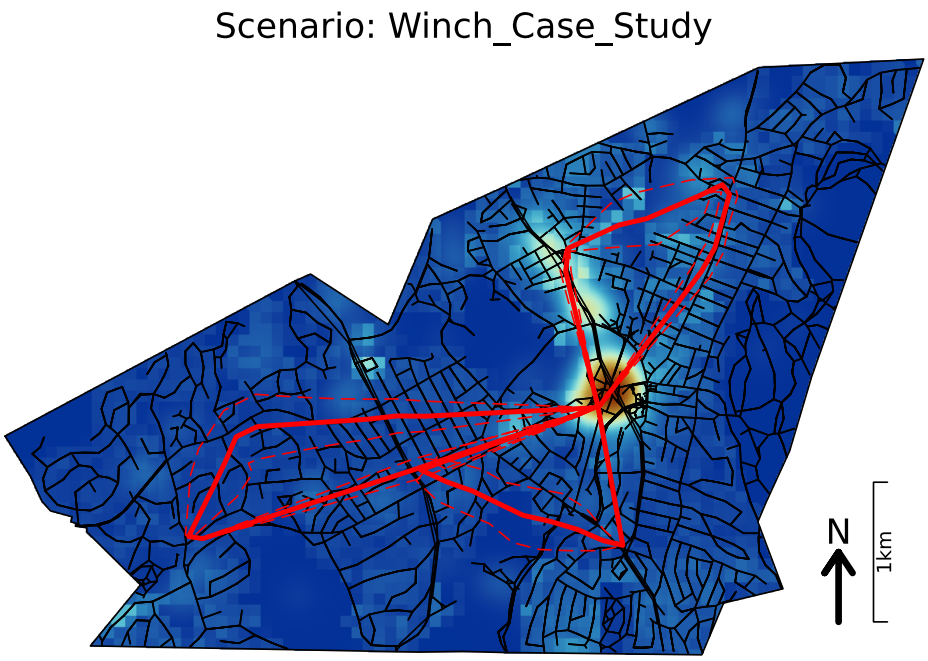


Fig. 6.1: Average route for the Case Study.

Optimization Parameters

Max Iterations: 2500	Waiting Time: 300	Connections: 1000
Max Time: 28800	Dwell Time: 20	Stop Penalty: 0.0
Max Stops: 10	Max Route Time: 3000	Distance Penalty: 0.0

The north loop travels along Main Street, this is a major thoroughfare and has a high concentration of shopping and work-related POI locations. The loop then turns at the intersection of Main Street and Swanton Street. This turn is located near the end of the main business section of Main Street and is adjacent to a large apartment complex. Additionally, there is a bus station at this intersection connecting with the 134 MBTA bus route. From this point onward the exact route is less clear, but there is an obvious stop at the intersection of Cross Street and Washington Street. This stop is located near many POI such as Winchester Hospital, Leonard Field, Muraco School and St. Peter's Anglican Church. The loop then returns to Winchester Center.

The Western and Southern loops are less well-defined. Both utilize Church Street, likely due to the high concentration of POI and population. The western loop appears to cross the region known as "The Flats", a section of gridded streets with moderate population density. It then continues westward toward Lexington, MA, looping back near Vincent Owen School. In the western region of the town there are multiple schools and a large area of population. Additionally, this loop connects to the 350 MBTA bus since it crosses Cambridge Street. The southern loop also travels along Church Street and then heads south-west. This path connects to Wedgemere station, and to the 134

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

MBTA bus at intersection of Bacon Street and Main Street. The loop appears to continue along the southern portion of Main Street back to Winchester Center station.

In general, the proposed route found in the optimization process aligns closely with a reasonable manually planned route. The route shown in Figure 6.2 was created as part of a university exercise for the course Methods and Models for Transit Planning(Winter 2023). This route, shown in orange follows the same general layout of the north and western loops of the proposed optimal route. This route was planned using local knowledge of the town and transit system. While the manually created route was not designed by a transit planner, the similarity shows that the optimizer does produce results that make sense and generally align with expectations.

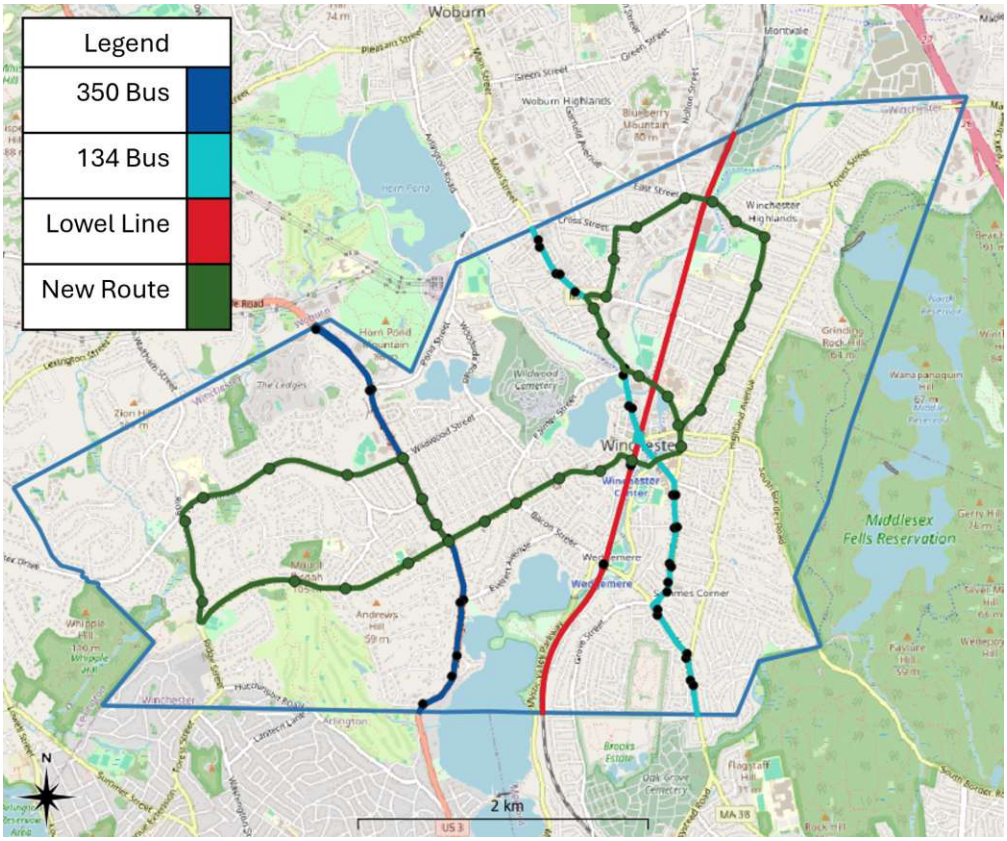


Fig. 6.2: Manually planned transit route for Winchester, MA.

6.1.2 Scenario Results

The resulting transit routes for the scenarios show the effectiveness of the optimization as well as some shortcomings. The routes for Scenario 1 and 2 are aligned with expectations, however the route for Scenario 3 exhibits strange characteristics.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Scenario 1

The route produced for Scenario 1 behaves exactly as expected. The eight public schools are all connected by the route and the stations are located near the center of each utility concentration. There is some variation in the route shape in the north portion of the route however the majority shows very little variation. The resulting variation in route shapes is likely the result of using ten stops instead of eight. With ten total stations there is flexibility with the locations for two of them. This flexibility could lead to the uncertainty in the route shape.

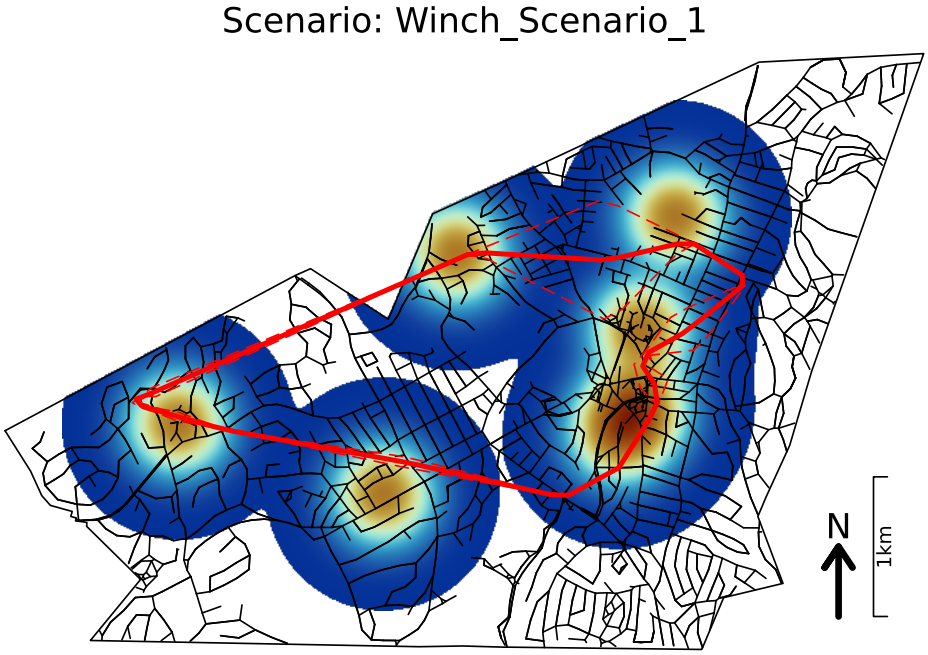


Fig. 6.3: Average route for Scenario 1.

Optimization Parameters

Max Iterations: 2500	Waiting Time: 300	Connections: 1000
Max Time: 28800	Dwell Time: 20	Stop Penalty: 0.0
Max Stops: 10	Max Route Time: 3000	Distance Penalty: 0.0

Scenario 2

Scenario 2 contains all the POI in the School/Daycare/Religious trip category. The majority of the religious locations are positioned in the center of the town or along Church Street. While the optimal route cannot be predicted, it is expected that any optimal route should visit the town center and likely travel along Church Street. It is not expected that all locations are visited, however there are many possible routes that can easily reach the majority of the POI locations.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Reviewing the proposed route from Figure 6.4, one can see that there is a high concentration of stations in the town center. This is as expected. Additionally, the western portion of the route passes near the religious POI on Church Street and continues to Vincent Owen school in the far western part of the town. While the average route does not pass though the center of each POI, it does loosely follow the structure of the locations. The shows that the optimizer is generally working as expected.

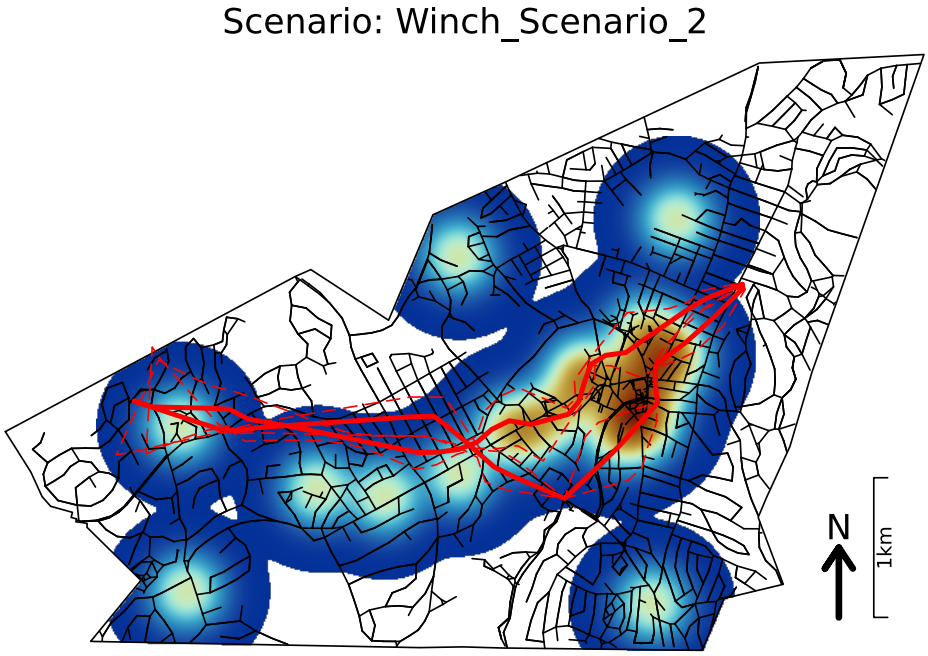


Fig. 6.4: Average route for Scenario 2.

Optimization Parameters

Max Iterations: 2500	Waiting Time: 300	Connections: 1000
Max Time: 28800	Dwell Time: 20	Stop Penalty: 0.0
Max Stops: 10	Max Route Time: 3000	Distance Penalty: 0.0

Scenario 3

Scenario 3 is more complex and contains all the POI from the Shopping/Errands trip category. The POI from this category are highly concentrated to the town center and Main Street which extends northward from the town center. It is hypothesized that an optimal route would be limited to the town center and the shopping location along Main Street, since these are the only locations of high utility.

The route in Figure 6.5 shows that the optimizer found a solution that is unexpected. While the proposed route does have a loop that extends from the town center to the north along Main street, it also has a large loop to the west that appears to provide little value to increasing the objective. The loop to the north makes sense and is aligned

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

with the expectations for Scenario 3, however the western loop appears to be illogical or in error. Additionally, the lack of any variation in the thirty solutions shows that there is a strong driving force to this solution. It is not clear what could cause this.

A detailed review of the entire instance, optimization and visualization pipeline, was completed to ensure that no mistakes or data errors were made. This review returned no errors, and it appears that the proposed route and utility heatmap are visualized accuracy. Additionally, the datasets used in the optimization process were confirmed to be the same datasets used in the visualization process. The only conclusion is that the optimizer is behaving in unexpected ways. This could be due to the high number of stops and generally highly concentrated utility. As a result the station in area of no-utility could have some unintuitive impact on the objective value.

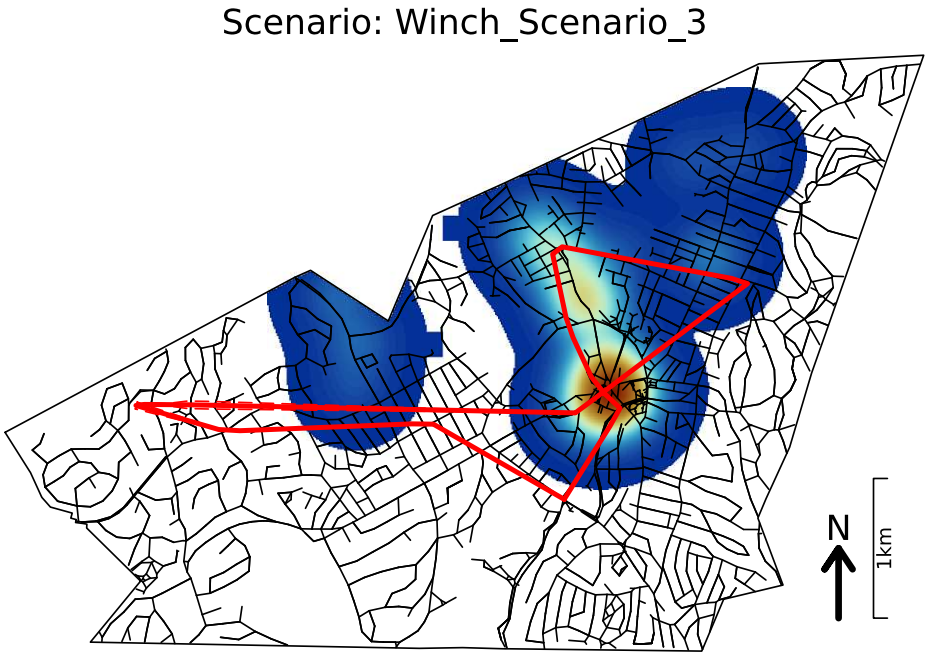


Fig. 6.5: Average route for Scenario 3.

Optimization Parameters

Max Iterations: 2500	Waiting Time: 300	Connections: 1000
Max Time: 28800	Dwell Time: 20	Stop Penalty: 0.0
Max Stops: 10	Max Route Time: 3000	Distance Penalty: 0.0

6.2 Hyperparameter Tuning

Hyperparameters are parameters that control aspects of the meta-heuristic. These parameters are used to adjust internal mechanisms of the algorithm. Often these the tuning of hyperparameters is done to maximize computation speed, however depending on the

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

parameter the tuning can also have an impact on the quality of the solution. To ensure the results of this thesis are of good quality, the relevant hyperparameters were evaluated and tuned to ensure high solution quality with reasonable running times.

6.2.1 Number of Processors

Since each optimization configuration must be repeated thirty times, a huge running time speedup can be achieved by executing multiple optimization instances at the same time. The optimization algorithm is programmed to execute with one CPU core, this means that multiple instances of the optimization can be launched at the same time on different cores. With forty cores available on the compute server, all thirty instances could be run at the same time, however all optimization process share the same GPU for the objective computation. To maximize the utilization of the computing resources a test was executed to determine the optimal number of optimization instances to launch at any time.

To determine the optimal number of instances, a parametric test was devised. In this test, increasingly more instances were run in parallel until a maximum of 16 instances were launched at the same time. For each processor count, the total running time and total number of completed iterations was determined. These values allowed for the number of seconds per iteration to be computed.

The resulting data is presented in Table 6.1. The results in this table are computed by summing the total running time across all instances. This value is divided by the total iterations across all instances and the total number of processors, see Equation 6.1

$$TimePerIteration = \frac{1}{NumberOfProcessors} \frac{\sum_{i \in I} time(i)}{\sum_{i \in I} iterations(i)} \quad I = \{\text{All Test Instances}\}. \tag{6.1}$$

Table 6.1 shows the results of the analysis in seconds per iteration. The values for 12 and 16 processors, shown in red, must be rejected. When running the analysis for these results, multiple instances crashed and terminated for an unknown reason. This invalidates the results for these sizes. Based on the results in Table 6.1, it was determined to run eight optimization instances in parallel.

6.2.2 Total Iterations

One important aspect with a heuristic optimization is to consider how many iteration the program should run for. In general, the objective continues to improve with an increasing number of iterations however, eventually the amount of improvement reduces. This reduction in improvement can either be from achieving solutions near the global maximum or the optimization process is stuck in a local maximum that the randomized moves cannot escape.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Tab. 6.1: Computational efficiency of different number of processors

Optimization Parameters		
Max Iterations: 3000	Waiting Time: 300	Connections: 1000
Max Time: 1800	Dwell Time: 20	Stop Penalty: 0
Max Stops: 10	Max Route Time: 2000	Distance Penalty: 0.0

Processors	Time Per Iteration
1	0.6685
2	0.7324
3	0.6802
4	0.7050
8	0.6453
12	0.5789
16	0.3459

The goal of this experiment was to determine a reasonable lower bound for the number of iterations. This lower bound was determined by running the optimization for an excessive number of iterations to ensure that the solutions have reached a strong result and then computing an acceptable lower bound from these results. To compute this lower bound, thirty optimization instances were run for a maximum of 30,000 iterations or 25,000 seconds. This running time was selected based on experience with running the optimization and the very upper end of possible running times. The results of these optimization runs can be seen in Figure 6.6.

The minimum iteration threshold was calculated by finding the point where the worst solutions achieved a large percentage of the best obtained objective. To find this point the 5th percentile objective value was computed for each iteration. The threshold was set at the iteration where the 5th percentile objective achieved 95 percent of the maximum objective value. This point is highlighted in Figure 6.6 and occurs at 920 iterations with an objective improvement of 21.95%. This iteration threshold was only computed for one configuration of optimization parameters, as parameters are adjusted this value would change, when using a k value of 1000, achieving at least 1000 iterations is a reasonable target value.

6.2.3 Limiting Connections

As described in Section 5.2.2, the parameter k was introduced to limit the size of the local search neighborhoods. The parameter k is used to limit the size of the neighborhood to only the k the nearest connections from each bus stop. The value for k can be anything between 1 and the size of the instance. With a small value the neighborhoods will be very small and allow for many iterations, however the quality of the nodes is limited, while a larger value of k results in larger neighborhoods and a slower search, however the vertex section will be of higher quality.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

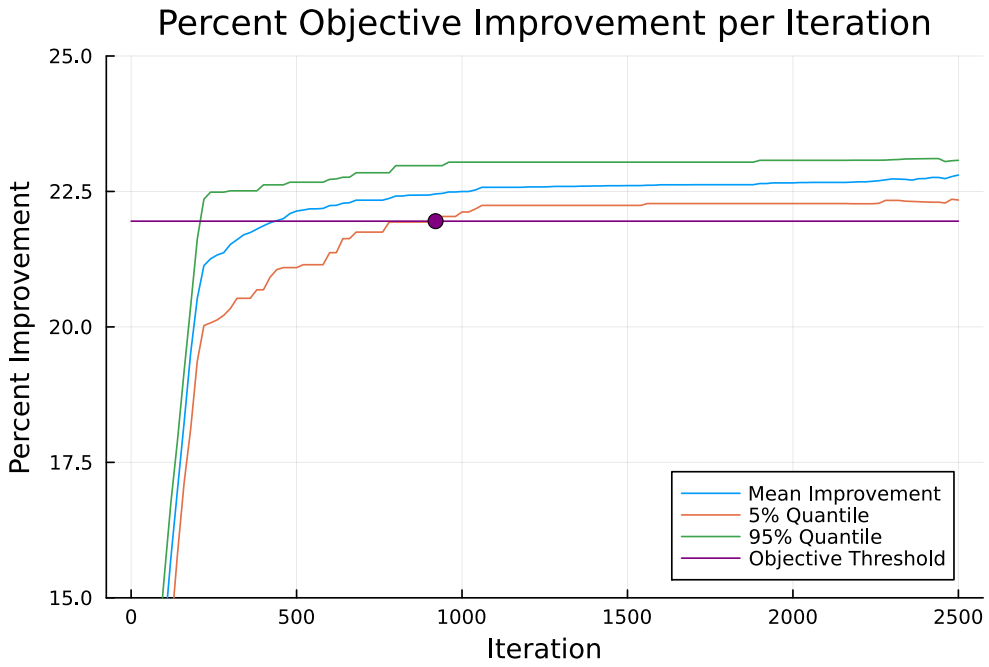


Fig. 6.6: The objective improvement against iteration value.

Optimization Parameters

Max Iterations: 30000	Waiting Time: 300	Connections: 1000
Max Time: 25000	Dwell Time: 20	Stop Penalty: 400
Max Stops: 30	Max Route Time: 3000	Distance Penalty: 2.0

To determine the best value for k a range of values between 50 and 1500 were tested. The configurations with different k values were run for a fixed amount of time on the Scenario 1 instance. The results can be seen in Figure 6.7. From this figure one can clearly see that as k increases the objective value also increases. One can also clearly see that the number of completed iterations reduces with increasing k. It was hypothesized that the lower values of k would allow the optimization to "walk" toward the best solution over multiple iterations. While it is not clear if this is possible, it is clear from these results that the increased number of iterations is not sufficient to achieve the same objective values as the larger k values. Finally, reviewing the chart the value of $k \approx 1000$ is a clear choice, it is the smallest value that achieved the best objective results.

6.2.4 Penalty Terms

During the development process, the initial solutions showed traits that were confusing and illogical. The resulting solutions were zigzagging across the region with no apparent consistency or logic to the route. To improve the overall shape and layout of the routes, a number of shaping penalty terms were added to the objective values. These terms were

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

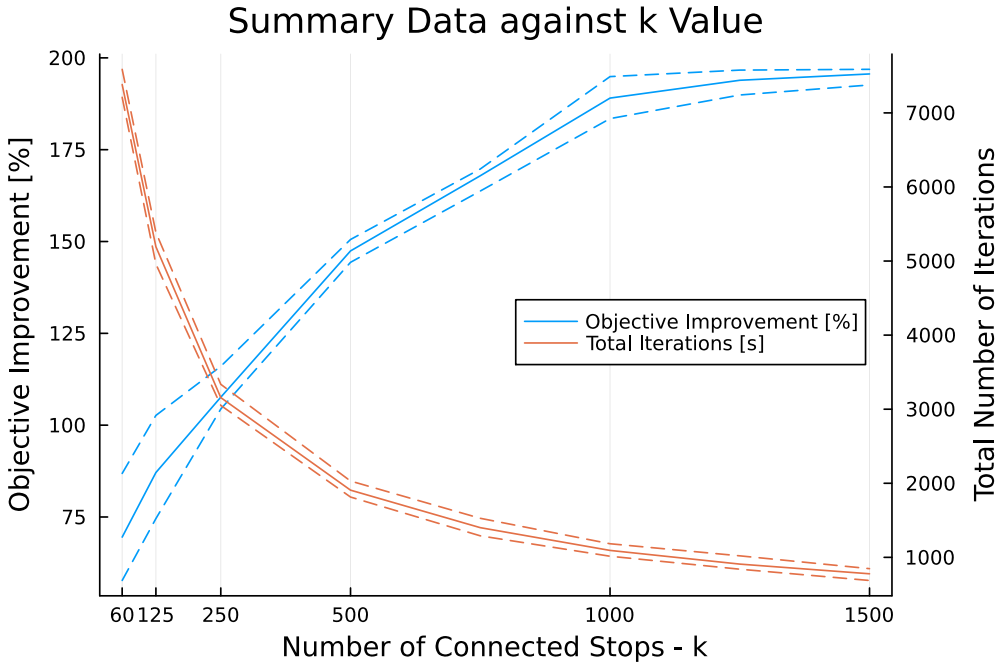


Fig. 6.7: Change of objective improvement with increasing k value.

Optimization Parameters

Max Iterations: ∞	Waiting Time: 300	Connections: NA
Max Time: 1200	Dwell Time: 20	Stop Penalty: 0
Max Stops: 10	Max Route Time: 3000	Distance Penalty: 0

used to penalize route characteristic that are not desired. In this case terms were added that reduce the objective as route length and number of stations increase in value.

By applying these penalty terms, the optimizer would only add a station or extend the route if the resulting solution provided a larger increase in objective then the cost via the penalty. The strength of the penalty terms must be adjustable in order to allow for the impact to be tuned. This results in a new formula for the objective value shown in Equation 6.2

$$Obj_{new} = Obj_{old} - C_1 * \text{length}(S) - C_2 * \text{size}(S). \tag{6.2}$$

In Equation 6.2 the function length(S) return the total length of the route by computing the sum of the all edges in solution S. The function size(S) returns the number of stops in the solution. The constants C_1 and C_2 are hyperparameters that determine the weighting of the penalty terms, when $C_1 = C_2 = 0$ they have no effect.

To determine the best set point for the penalty values, a range of different candidate values were tested. The test was done using Scenario 1 and the range of values was devised such that it swept from zero to a value large enough that the penalty term is equal to the total objective value. Only one parameter was tested at a time for simplicity.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

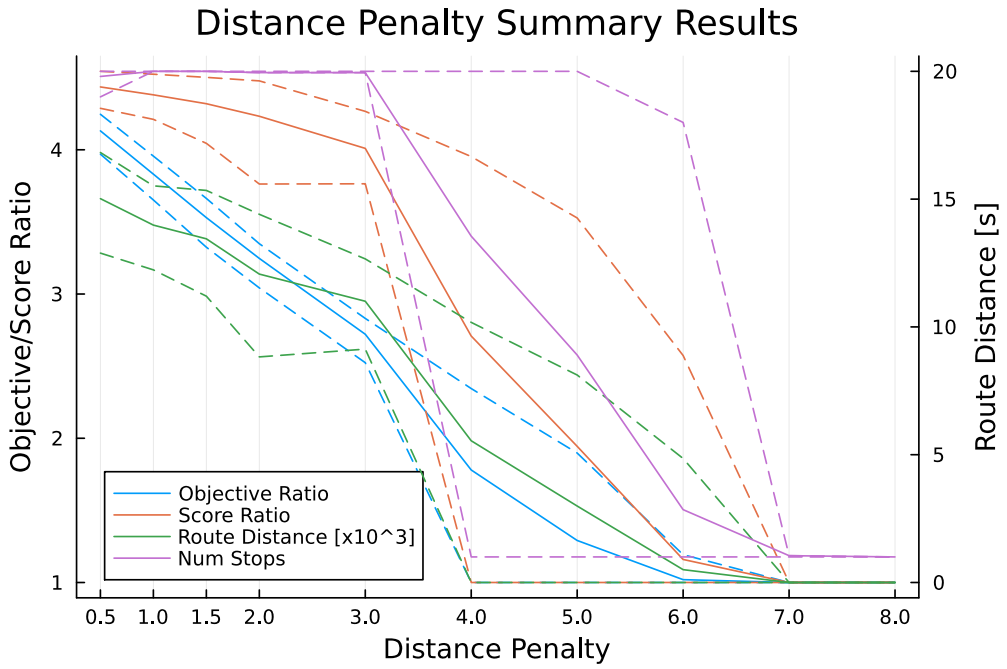


Fig. 6.8: Summary results for the distance penalty analysis.

Optimization Parameters

Max Iterations: 2000	Waiting Time: 300	Connections: 1000
Max Time: 1200	Dwell Time: 20	Stop Penalty: 0.0
Max Stops: 20	Max Route Time: 2000	Distance Penalty: Various

The results of an experiment to determine the value of C_1 is shown in Figure 6.8. The figure shows the mean values in solid lines with the dashed lines indicating the fifth and ninety-fifth percentiles for each value. Additionally, the objective ratio is a normalized value representing the ratio of the current objective to the objective with an empty solution. In this experiment the penalty coefficient was varied between one half and eight. Considering the green line in Figure 6.8, it is clear that as the penalty coefficient increase from 0.5 to 4, the route distance is reduced from around $15 * 10^3$ to around $6 * 10^3$. This shows that the penalty is working as expected. However, as the penalty coefficient increases above 4 the penalty starts to become too strong, and the solution quality breaks down. At this point, some solutions start to have only one stop, this is clear from the drastic increase in the deviation in the number of stops(purple). Additionally, the orange line shows the pre-penalty objective value(denoted score), is reduced massively also indicating poor quality.

The same evaluation process was applied to the coefficient for the penalty on the number of stops in the solution. The value of C_2 from Equation 6.2, was varied from a value of 50 to a value of 800. The results of this experiment are shown in Figure 6.9.

The penalty term on the number of stops is highly ineffective at controlling the total number of stops in the solution. Analysis of Figure 6.9 shows that as the stop penalty

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

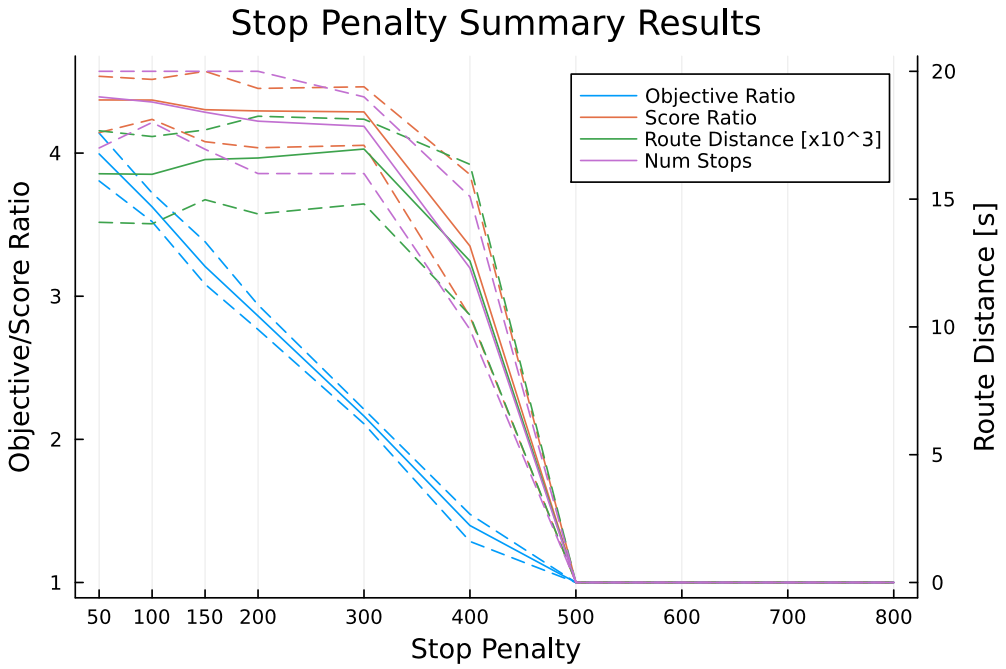


Fig. 6.9: Summary results for the stop penalty analysis.

Optimization Parameters

Max Iterations: 2000	Waiting Time: 300	Connections: 1000
Max Time: 1200	Dwell Time: 20	Stop Penalty: Various
Max Stops: 20	Max Route Time: 2000	Distance Penalty: 0.0

coefficient is increased from 50 to 300, there is little effect on the total number of stops in the solution, however there is a strong effect on the overall objective ratio. Once the coefficient reaches 500 all solutions are broken, and return empty sets. The solutions breakdown because any initial solution created by the construction heuristics cannot provide enough change to the objective to overpower the penalty term.

Overall, the penalty terms were not effective at controlling the total route distance nor the number of stops in the solution. Additional adjustments to the implementation of this penalty could have resulted in meaningful methods of solution shaping, however it was deemed to be a low value path of investigation. The penalty terms were abandoned for two primary reasons, they would require per instance tuning, and they ultimately were not needed. Each instance has a different baseline objective that is computed for an empty solution. To implement the penalty coefficient the values must be tuned to this baseline objective value and the resulting improvements that can be obtained from any solution. This is a difficult process and limits the effectiveness of the penalty terms. In addition to very difficult tuning the penalty terms were abandoned because the initial need was a visualization error not an issue with the optimization process.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Due to the poor performance and lack of need for the route distance and number of stop penalties, the coefficient values were set to zero for the case study analysis and the majority of the hyperparameter tuning.

6.2.5 Waiting Time

The bus frequency or headway has a major impact on a particular's route effectiveness with respect to improving the connectivity of the region. To understand the impact, a parametric study was used to vary the waiting time across a range of reasonable values. The values chosen were $W = \{0, 150, 300, 450, 900\}$, representing the average passenger standing time in seconds. These values are directly related to the headway of the bus service. When riders' arrival times are uniformly random, the average headway can be assumed to be twice the waiting time. This results in headway value of $\{0, 5, 10, 15, 30\}$ minutes. These are all very common headway values, with zero included to understand the route shapes at the limit.

Reviewing the resulting route shapes one can see little difference in the coverage area. All the routes form a flower petal shape, with multiple loops that connect back to the center of the region. The route with waiting time of zero shown in Figure 6.10a appears to be marginally more compact than the other routes.

The average objective values and total route length are shown in Figure 6.10f. This figure shows that there is a clear decline in total objective with increasing waiting time, as would be expected. With the largest waiting times, the objective ratio is approaching 1.05, this indicates a very small improvement over the connectivity with no transit route. Additionally, as the waiting time increases, there is a clear increase in the total route length. This indicates that longer routes are more optimal with higher waiting times.

The results shown in Figure 6.10 were generated with stop repetition allowed. Stop repetitions allows for virtual transfers inside the route. These virtual transfer artificially inflate the objective value and could impact the resulting route shape. To understand the effect of the virtual transfers, this analysis was repeated with duplication of vertexes disabled. These results are shown in Figure 6.11. Directly comparing to the results with virtual transfers one can see effectively no difference. The summary result in Figure 6.11f and 6.10f show that nearly identical route length and objective ratio values were achieved. Furthermore, qualitative analysis of the route shapes shows that the routes have very similar configurations at each waiting time. Due to the minor impact of waiting time on the route shape, a value of 300 was chosen for the analysis. This is representative of a route with ten minutes of headway, a common frequency on a local route.

6.2.6 Number of Stops

In this experiment the number of stops was varied to see the impact that the total number of stops had on the route shape. The set of total number of stops considered was $N = [6, 8, 10, 12, 18, 24, 30]$ This range was chosen manually starting at a small value and increasing in regular intervals. The maximum value was set based on user experience

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

with the optimization process and to reduce overall run time since the neighborhood size increase with the solution size.

The results shown in Figure 6.12 show that increasing the maximum number of stops has a direct impact on the suggested route shape. All the routes form a flower pedal configuration where the route forms multiple pedals or loops that all pass though a central location. The number of pedals is directly related to the maximum number of stops. The results for six stops only create two loops as can be seen in Figure 6.12a, while six pedals are formed when 24 or 30 stops are used. This can be seen in Figure 6.12f and 6.12g. Additionally, these loops are spread out covering the entire region to capture as much utility as possible.

In Figure 6.12h one can see the change in solution parameters as the number of stops increases. What is immediately clear is that the objective ratio improves with increasing number of stops. Additionally, the total route length also increases with increasing number of stops. The norm to the average route remains constant with increasing number of stops however with fewer stops there is more variation in the norm among any given set of solutions. Finally, the running time to find the optimal solution also increases with an increasing number of stops.

For the case study analysis the limit on the number of stops was set at 10. This value was chosen because the route shape with a lower number of stops are more realistic. Ideally a route would have a number of stops along each leg however the inclusion of additional only adds to the number of pedals in the route. Because of this, the number of stops was limited to ten. Ten is a sufficient number of allow for detailed exploration of the region with a reasonable number of route pedals. From a practical standpoint, allowing numerous stations significantly increases the optimization time because the neighborhood size is a function of the solution size.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

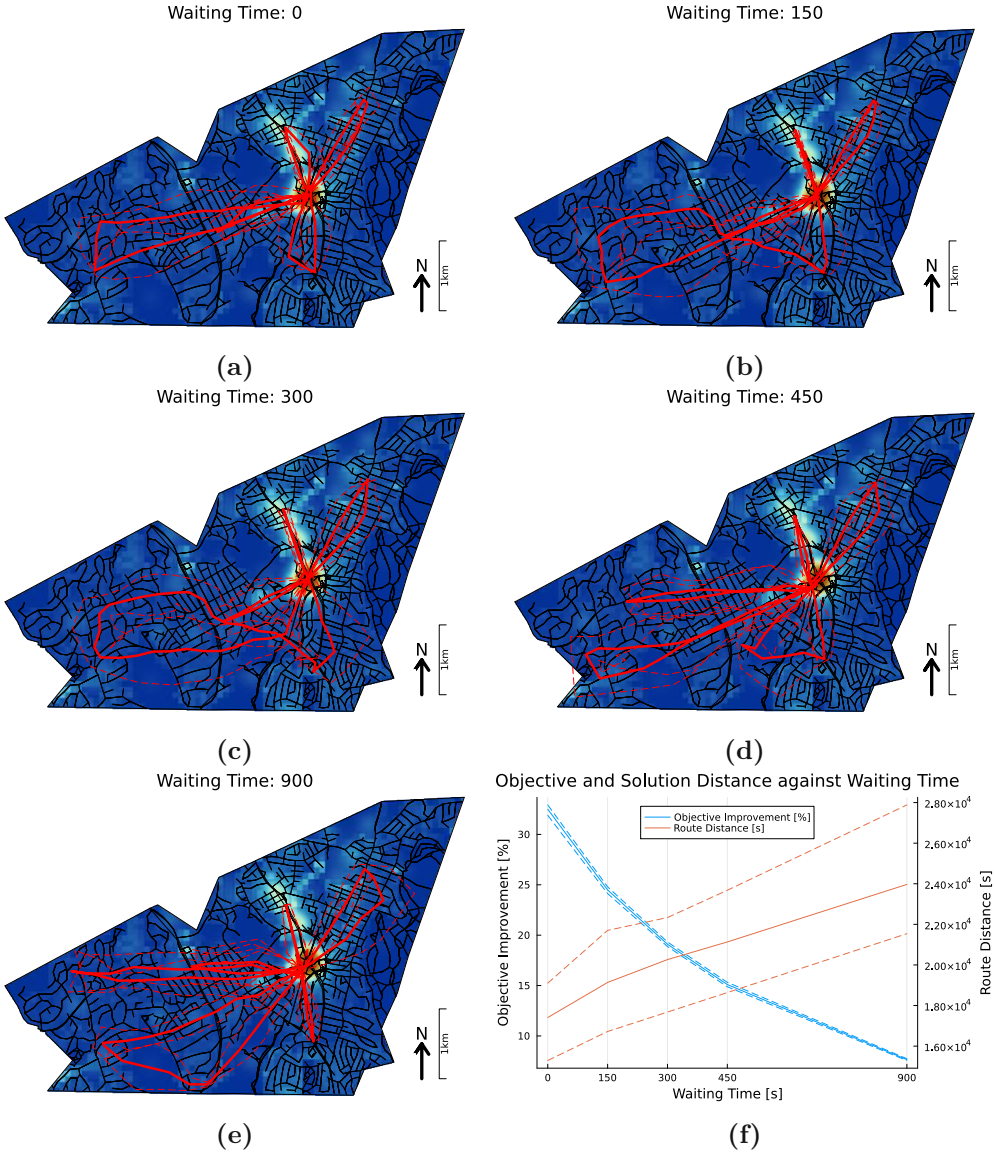


Fig. 6.10: Waiting time results with repeated vertexes allowed.

Optimization Parameters

Max Iterations: 1200	Waiting Time: Various	Connections: 1000
Max Time: ∞	Dwell Time: 20	Stop Penalty: 0.0
Max Stops: 16	Max Route Time: 3000	Distance Penalty: 0.0

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

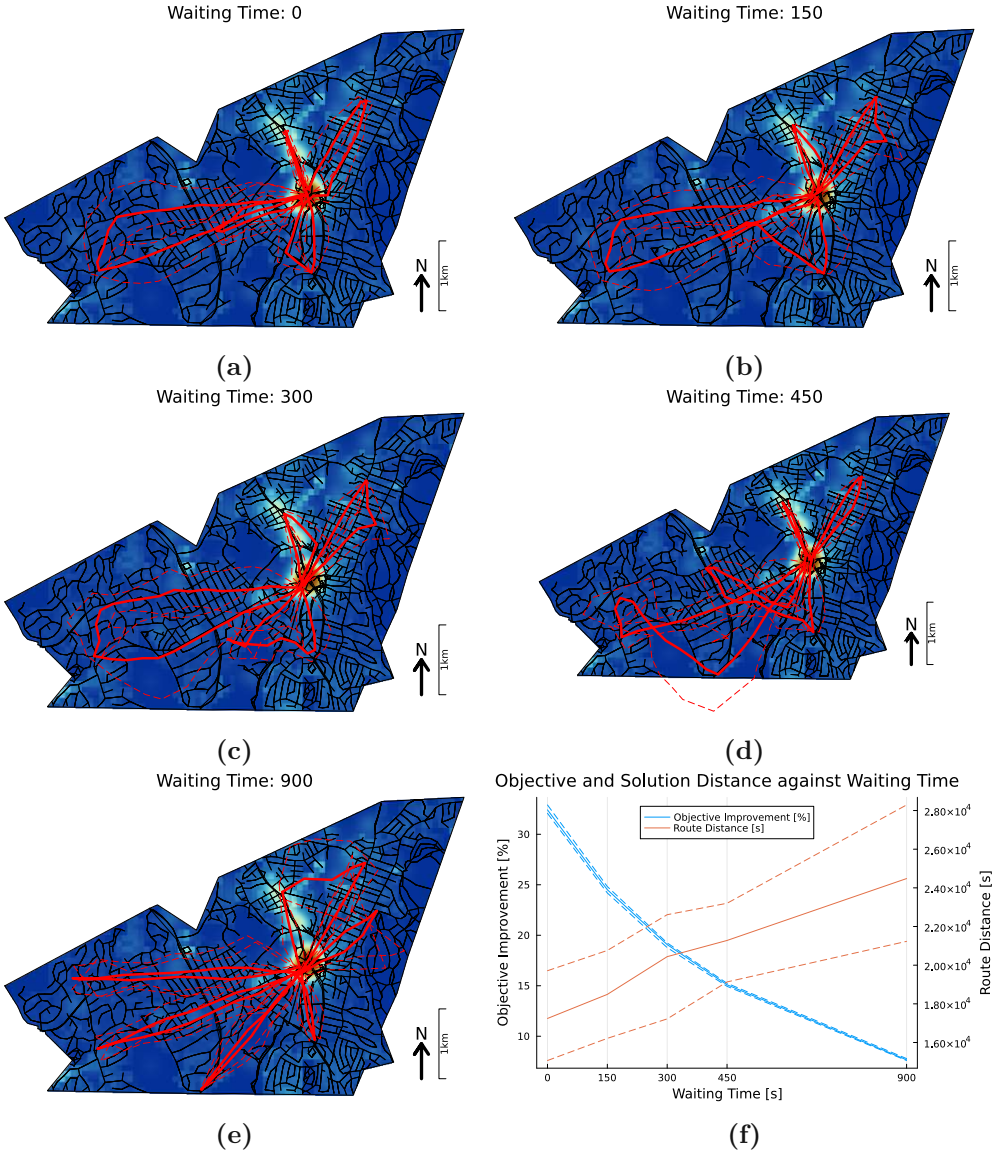
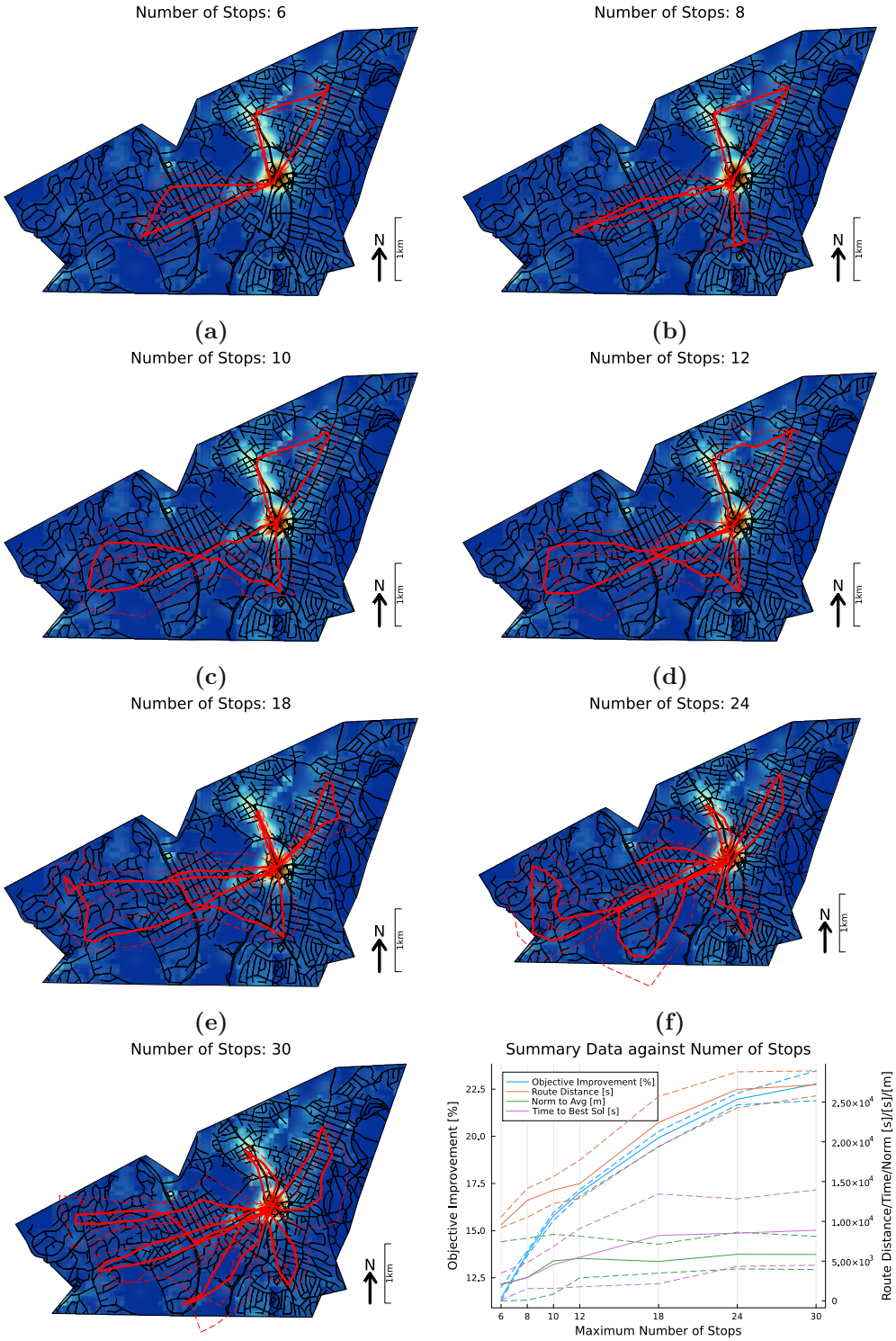


Fig. 6.11: Waiting time results without repeated vertexes.

Optimization Parameters

Max Iterations: 1200	Waiting Time: Various	Connections: 1000
Max Time: ∞	Dwell Time: 20	Stop Penalty: 0.0
Max Stops: 16	Max Route Time: 3000	Distance Penalty: 0.0

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.



(a) (b) (c) (d) (e) (f) (g) (h) Summary Data

Fig. 6.12: Results form the parametric study into the number of stops.

Optimization Parameters

Max Iterations: 1200	Waiting Time: 300	Connections: 1000
Max Time: 14400	Dwell Time: 20	Stop Penalty: 0.0
Max Stops: Various	Max Route Time: 3000	Distance Penalty: 0.0

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Chapter 7

Discussion

The implementation and results of this thesis achieved the overall objective. The connectivity value as defined is a useful metric with respect to route planning optimization and the utility value provides an adaptable mechanism to model the socio-economic values of different areas. The combination of the connectivity metric and the utility was effective as an optimization parameter. The resulting transit routes meet the design requirements and align with manual design efforts.

7.1 Proposed Optimal Transit Route

The optimized route for the case study is realistic based on the instance definition and the optimization goals for this thesis. The route is similar to a manually planned route and the set of all proposed routes show a high level of similarity. The high level of similarity indicates that the optimization process is converging to a single route shape. Additionally, the proposed route shape is a very reasonable route for the case study area.

The proposed route achieves connections to the majority of the town, the high density areas and the existing transit system. Firstly, the three flower pedals of the route all connect back to the center of the town. This is the area of the town with the highest utility and is a natural convergence point. Additionally, the location of the town center station is the Winchester Center regional rail station. This rail station provides direct access to the center of Boston and this transit link is already utilized by many commuters. By implementing the proposed transit route, a first mile solution would be provided and a reduction in personal vehicle first mile trips could be enacted. In addition to connecting to Winchester Center station, the route connects with both through town bus lines and the second regional rail station.

In addition to the transit connection, the proposed route provides connections to the business direct along Main street, as well as points of high population density. The major apartment complexes in the town are in proximity to the route as well as the majority of schools.

Overall the proposed case study route would likely be effective at providing a high level of connectivity to the town of Winchester, Massachusetts. This achieves the design goals set out at the beginning of this thesis and shows that new public transit routes can be designed using optimization algorithms.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

7.2 Connectivity Metric and Optimization Implementation

Overall the implementation of the thesis was successful. The connectivity metric, utility, instance generation, and optimization process were all able to achieve the goals of the thesis. Despite the success, each element of the implementation has room for improvement or additional accuracy.

7.2.1 Connectivity Metric and Utility

The design of the connectivity metric and utility value were able to effectively achieve the modeling goals for this thesis. The connectivity metric captured the connectedness of all the points in the region, and it was able to be meaningfully increased with the addition of new transit routes. This is enabled by the consideration of network effects on the connectivity value. Incorporation of the network effects sets the concept of connectivity apart from the existing accessibility metrics.

The utility value is a powerful modeling tool that was able to incorporate data from different sources and meaningfully combine them into a single distribution. The mathematical foundation of the utility value creates a high potential for the value, since any arbitrary data sources could be incorporated into it. Transit planners can use other socio-economic data, such as income levels, age, car access and more, to tailor the utility value to any particular modeling objective.

One aspect of the connectivity metric that was not effective was the use of the potential model as a base equation. This made the connectivity a non-linear function with respect to the distance between the source and destination points. The non-linear relationship is not an issue when calculating the connectivity however it becomes a large issue when re-computing the connectivity when using delta evaluation. Equation 7.1 requires that whenever a distance is modified the entire fraction must be recomputed

$$\frac{1}{a+b} \neq \frac{1}{a} + \frac{1}{b}. \tag{7.1}$$

By reconfiguring the objective into a form where $CM_{ab} \propto U_b d_{ab}$ the objective is linearized with respect to the distance between vertexes. This linearization allows for utilization of the transitive property of addition, and it opens up many possibilities for improvements to the objective computation process. This is highlighted in the following Equation

$$CM_{ad} = U_b d_{ad} = U_b(d_{ab} + d_{bc} + d_{cd}) \mapsto \Delta CM_{ad} = U_b \Delta d_{bc}. \tag{7.2}$$

Consider the shortest path $a \to b \to c \to d$ and segment bc increases by Δd_{bc}. When using the linearized form, the change in the connectivity metric ΔCM_{ad} can be computed directly from the change in the length of d_{bc} and the utility of the destination. The contribution of the entire path does not need to be recalculated. Additionally, if the full set of routes utilizing segment bc is known then the change for the whole set can be as easily computed.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

7.2.2 Instance Implementation

The implementation of the instance is the most critical part of the optimization process. The quality of the instance directly impacts the usefulness of the generated solution. In this thesis the instance was implemented with high quality for the modeling goals however the modeling goals are not closely aligned with transit planning objectives.

When considering a new route, transit planners consider more than how effectively the route connects a particular region. Planners must consider the cost of the route, expected ridership, the feasibility of stop locations, impact on local communities and more. None of these aspects were modeled in this thesis, which limits the usefulness of any proposed route. Modifications to the utility value, the optimization graph and objective value could model these effects. For instance, changes to how the bus stops locations are generated would allow planners to only generate potential stops in feasible locations. The generation process could be modified to avoid features like cul-de-sacs, steep grades and dead end streets.

In addition to the usefulness of the implementation, the accuracy of the instance could also be improved. The walking graph and road graphs in the analysis were created in a canonical way using public data sources and existing graphing tools. The simple method resulted in the center of roads being used as the walking path. The model could be enhanced by adding sidewalks and pedestrian crossing. Another enhancement that could be made is to incorporate the local environment into the edge weights of the walking network. Instead of basing the edge weights on travel time, effects such as path quality, environment, safety and elevation gain can be incorporated. Applying these improvements would enable the virtual pedestrians to follow paths that are more realistic when compared to real pedestrians.

The accuracy of the road network model could also be improved. All streets were modeled in the same way with the same maximum speed, and no consideration is given to the complexity of intersections. This model of the road network is far from reality. The acceptable bus speed on different roads is highly variable when comparing major roads with neighborhood streets. Additionally, bus access to each road is not feasible nor guaranteed. Intersections are also modeled simplistically in this thesis. A vehicle on the model road network can make a turn in any direction at any intersection with no time penalty. All the mentioned issues can be eliminated with additional modeling. The software SUMO can generate road networks that have localized speed limits, intersections with realistic signaling and more. Utilizing this software would improve the computed bus travel times and therefor the resulting proposed route.

7.2.3 Optimization Implementation

The implementation of the optimization process was highly effective and overall preformed well. The GVNS algorithm is a simple meta-heuristic however it produced good results in reasonable time. Authors in the literature have applied more complex results, with generic algorithm being particularly common. The effectiveness of GVNS shown

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

by this thesis indicates that local search based algorithms should not be overlooked for transit related optimization tasks.

One area of the optimization process that was not effective was the limited size of the local search neighborhoods. The k implementation was functional, but it did not produce useful results for anything but the largest values for k. The original concept for limiting the size of the neighborhoods was that with sufficient smoothness to the utility distribution, any particular stop could slowly move to the optimal location over the course of multiple iterations. These moves would be small and therefor the next vertex would be close to the current vertex. When experimenting with the optimization, it was found that the solutions would not "walk" to the optimum, instead they would get stuck in local optima with much lower objective values. To get good solutions the k had to be set to over 1000 and as a result it provided little reduction in the neighborhood size. Additionally, since the majority of vertexes were considered, the smoothing of the utility probably could have been removed or at least reduced.

Delta evaluation is a very common when using meta-heuristcs, it involves only computing how much the objective has changed rather than recomputing the entire objective value. On large instances delta evaluation can provide a huge speedup because the change in objective can be computed in near constant time. Despite significant effort, delta evaluation was not able to be implemented for this analysis. The first major issue is related to the function used to compute the objective. As mentioned in Section 7.2.1 the objective is non-linear with respect to the distance between the stops. As a result, a change to a single segment of a shortest path results in the contribution of the whole path needing to be updated. This is compounded by the loop nature of the solution. The looping route causes more than half of all paths though the route to change when a single edge in the route changes length. This means that the number of shortest paths that need to be updated is linear with the total number of paths that utilize the route. As a result it is not currently known if delta evaluation can be implemented for this objective function.

7.3 Supplementary Results

The case study is not the only meaningful results from this thesis. The analysis of the hyperparameters showed interesting results with respect to the shape of the routes, penalty terms, and the number of stops.

7.3.1 Shape of Route

What is clear from all the results is that the routes generally form a characteristic "flower pedal" shape. The routes have a clearly defined center near the center of the region and then form multiple pedals that extend out from the center, sweep though part of the region and then return to the center.

This shape is very clear when reviewing the results from the number of stops testing. Here you can see that with a low number of stops there are a small number of pedals,

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

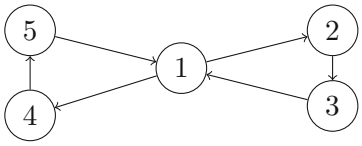


Fig. 7.1: An example transit route with an internal transfer

and as the number of stops increases the number of pedals also increases. Additionally, the waiting time results show that all the waiting time values formed the flower pedal shape.

It was hypothesized that the cause of the flower pedal shape was the internal transfers inside each route. When a route repeats a stop, the shortest paths in the transit network can now execute transfers at the stop without paying any transfer penalty. This is highlighted in Figure 7.1. In this figure the route follows the path $[1, 2, 3, 1, 4, 5]$, with the final connection back to vertex 1 happening automatically to close the loop. Now consider a path in the route that travels from station 3 to station 2. A rider on the bus would have to follow the whole route passing though the stops $[3, 1, 4, 5, 1, 2]$, however due to the implementation details, internal transfers are allowed. This enables the shortest path between vertexes 3 and 2 to be simply $[3, 1, 2]$. By exploiting these internal transfers the optimizer can create routes that provide additional connectivity.

The internal transfers are an artifact of computing the shortest paths in the transit network. Routes with repeated stops present a challenge when computing the shortest paths, because the duplicate vertexes break the mapping into the overall distance matrix. Because the shortest paths are computed twice, once in the transit matrix and once as part of the objective computation, the internal transfer would be computed in the objective computation process even if the correct distances were kept during the transit matrix generation. This could be avoided by creating new vertexes for each repetition of a vertex, however this would change the size of the distance matrix and require dynamically sizing this large matrix for each solution vector. While this would be possible it was deemed to have to significant of a performance impact.

To evaluate if the repeated vertexes have a significant impact on the route shape, the waiting time analysis was repeated with repeated stops disabled. This was implemented by rejecting any solutions that repeat any stops. It is clear from the results in Figure 6.11, that the flower pedal shape is retained even with no repeated stops. This indicates that the flower pedal shape is not solely due to internal transfers. Considering the utility distribution for the region, the flower pedal shape could be driven from the very high concentration of utility in the town center. Additionally, in the town center there is a connection to the regional rail. This regional rail connection offers access to a very large amount of external utility, since this rail line extends to the city of Boston and other major population centers.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

7.3.2 Effectiveness of Penalty Terms

The penalty term results presented in Section 6.2.4 showed that it was possible to impact the shape of the solution by applying controls on the objective value. In general, it would be possible to develop a penalty term strategy that can shape the solution into a form that is desired by the operator.

Practically the simplistic method attempted in Section 6.2.4 was generally not effective. Firstly, the route length penalty only had a small impact of the route length. There was a slight decrease in route length as the penalty was increased, however the resulting change are within the variation of the route lengths. As the penalty terms become larger, it overpowers the objective and the optimization process fails to produce any routes. Secondly the number of stops penalty showed similar results. With low values the penalty term did not have a strong effect, with only a few routes having fewer stops. When the penalty was large enough there was a strong impact on the total number of stops, however there was not a smooth transition. Additional tuning would be required to find an effective value for the number of stops penalty. This tuning however, would be specifically tailored to a particular instance.

In general the issue with the penalty terms is that since they are a constant scale on the route length or number of stops, they have to be specifically tuned based on the actual objective values that can be achieved on the instance. This requires the instance to be run without penalty terms and then the terms can be correctly scaled and applied. This is a suboptimal process.

Another consideration is the intent of the penalty terms. When they were initially considered it was to attempt to limit the "spikiness" of the routes that were being produced. The spikiness turned out to be a now fixed visualization error and not an error in the optimization process. The wrong graph was used as part of the visualization piping resulting in the route vertexes being plotted in the wrong physical locations.

An additional goal of the penalty terms was to try to force the routes to be more similar to a bus route that would be designed by a human transit operator. In general, the implemented controls were not effective at achieving this. The main differences between the discovered routes and a human designed route is that a human designed route would have more stops along the existing path and would likely not form the pedal shapes. Additionally, self-crossings are typically avoided by transit planners. The implemented penalty terms are not effective at implementing these controls. There are other methods that could be applied to attempt to achieve these goals.

For instance, a penalty could be added based on the number of crosses in the route, evaluations of the area covered by the route or another metric could be used. This would allow the operator to force the route into particular shapes. In addition, a better penalty system would involve adding multiple phases to the optimization. For instance, when there is a fixed number of maximum stops, the optimizer always uses the maximum number allowed. In general, it prefers to use these stops to create additional flower pedals that reach out from the town center. What could be done is to implement a multiphase approach, the first phases allow only a small number of stops forcing the route to span the important areas of the town. In the second phase, a large penalty

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

could be applied to the route length and the total number of stops could be increased. In theory this would allow the optimizer to find intermediate stops along the route that are in optimal locations.

7.3.3 Impact of Number of Stops

The results from the number of stops analysis show that routes always use the maximum number of stops. This makes sense from an optimization perspective because there were no penalties implemented based on the number of stops. Additionally, the route length increased as the number of stops increased. This also makes sense as the optimal route adds length to span more of the region. Finally, the objective increases with the number of stops. This also makes sense due to the increased number of transit connections in the network.

Looking at the route shapes there are some interesting qualitative results. Firstly, one can see that all the routes form the flower pedal shape. The number of pedals clearly increases with the number of stops. Adding additional stops allows for the optimizer to add another pedal and span more of the region. With a small number of stops the pedals are limited in size and focused only on areas with very high utility. This would be expected. As more stops are added, and the routes start to increase in size. Eventually when there are sufficient stops, additional pedals are added. These additional pedals go to regions where there is lower utility.

From a route planning perspective, the optimizer is clearly focused on connecting the regions of high utility. Only when there is sufficient resources does the route travel to areas where there is lower utility(i.e. residential regions). There are two different type of pedals in the route there are small pedals that span the business districts and larger ones that cover the residential neighborhoods. If these routes were to be planned by a human, the longer pedals would travel though the business areas and then continue onto the residential region. This likely has a lower objective value than the proposed solutions, however the route shape is more realistic. By adding penalty terms or modifications to the optimization process, it may be possible to force this type of route however there was not sufficient time to explore these options.

7.4 Applicability and Shortcoming of Analysis

The results presented in Section 6 and the discussion previously in this section show that the optimization process was effective at finding a new transit route in the case study city of Winchester, Massachusetts. This highlights the potential for computational based transit route planning however the applicability of different case studies and optimization use cases should be considered.

7.4.1 Applicability to Other Transit Modes

The optimization process presented in this thesis could easily be applied to other modes of traditional transit. In addition to bus routes, modes such as trams, subways, ferries

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

or regional rail could be considered. To optimize for another mode modifications would need to be made to the creation of the optimization graph however the rest of the instance generation process can remain the same. When considering demand responsive transit, the applicability of the optimization is more limited.

When applying the optimization process to another mode of fixed transit, the only changes that must be made are to the optimization graph. For this thesis the optimization graph was made with two assumptions, firstly that a bus stop can be placed on any street and secondly that each bus stop can connect to any other via the shortest path. Both of these assumptions are invalid when considering other modes. Tram, subway and rail stations cannot be placed anywhere, and it is not necessarily possible to connect any two stations via the shortest path. To generate an optimization graph for one of the alternative modes, the station locations would need to be selected by transit planners and the routes between the stations would need to be created based on feasibility. The resulting network would be representative of every path in the region the mode could travel along and every location a stop could exist. This network can then be modeled and encoded into the optimization graph. The rest of the instance and optimization would perform the same.

The optimization tool presented in this thesis has limited applicability to demand responsive transit(DRT). DRT is a transit mode that allows riders to request transport at any time and a vehicle will be dispatched to service the request. The mode typically does not follow fixed routes or schedules and as a result the route optimization process is not directly useful for DRT.

The utility distribution and supporting definition could be applied to DRT. If the DRT operator wanted to implement a fixed set of pickup points for DRT riders, the utility value could be effective for selecting these locations. By defining the utility for the region, the pickup locations could be found by solving a continuous k-centering problem. In this problem, k pickup locations are optimally placed in the distribution such that the maximum distance to any pickup location is minimized.

In addition to the pickup location the route optimization may be able to assist in the generation of route skeletons for DRT. A DRT route skeleton is a base route that the DRT vehicles follow when executing a pickup. The skeleton route provides a baseline that is minimally modified to capture the pickup. While the optimizer could find potential routes to use as skeletons, it is not clear how effective the proposed routes would be. When defining a DRT skeleton, considerations should be made for ease of modification, expected users, expected destinations and more factors. Currently, it is not clear how these factors can be incorporated into the optimization process therefor the usefulness for route skeleton generation is limited.

7.4.2 Applicability to Other Regions

The instance generation and optimization process has no limitations that would stop its application to any region. While the optimization can be applied to any region, the resulting route may not be useful. To ensure the proposed routes are useful, the region of interest must meet a few criteria. Firstly, region of interest must have a clear need

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

for a transit route. Secondly, the utility distribution for the region must support the identified need and have sufficient variation across the region of interest. Finally, the utility distribution must not be susceptible to quantization issues.

For the resulting transit route to be useful, there must be an identified shortcoming or gap in the current transit system. This criterion is natural however it does limit the applicability of the optimization process. Regions that are very small, or that already have significant existing transit are not good optimization candidates. Small regions such as rural European villages are not good candidates because the village can be small enough that the proposed route is unnecessary. Additionally, routes in the CBD or areas with a high level of existing transit have little margin for improvement. If the connectivity is already very high then any potential improvement from a new route is small. Additionally, routes inside the CBD of a city face challenges due to the uniformity of the utility distribution.

In addition to addressing a need, a proposed route is only useful if the route is unique. The uniqueness of the optimum route is dependent on many factors however a major factor is the variation in the utility distribution. When the utility distribution is highly uniform then there can be many routes that achieve the same connectivity improvement. Consider a region with a perfect grid of streets and uniform utility. In this case, whatever optimal route is found, the rotated and mirrored versions of the route provide the same connectivity. While this example is clearly contrived the issue remains on real instances. If there is not sufficient variation in the utility distribution, then the uniqueness of the routes is impacted and repeated iteration of the optimization can result in significantly different proposed solutions.

The required variation in the utility presents a challenge with applying the optimization process to rural regions and very dense regions. Dense urban areas such as the CBD and urban cores, are challenging from a utility perspective because the utility is uniformly high across the area. Additionally, when considering the applicability to North American rural communities the same issue can arise. The town are often characterized by single family home on large lots. This creates a population distribution that is uniform across the entire region with few to no points of interest. This uniform distribution minimizes the effectiveness of any proposed route because a similar amount of additional connectivity could be generated by a similar route in a different part of the town. In addition to the uniformity, the utility distribution in North American rural towns is effected by quantization issues.

Quantization issues in the utility distribution are issues stemming from the impact of single elements on the overall utility distribution. The utility of a point should be representative of the usefulness of the local area, however when the total number of POI is limited, each POI can have a large effect on the utility. Ideally the utility distribution is composed in a way that the elimination of any POI would have a negligible effect on the overall distribution. By meeting the criteria, the proposed routes are connecting areas of the region with higher and lower utility and not specific POI to other POI. A great example of this is the population distributions for rural towns in the United States. These towns often have single family homes on large lots. When paired with

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

the 90-meter population data, the population counts for different points in the dataset can be attributed to single homes. The variation in this population distribution is from the quantization of individual homes instead of overall trends in the population. These quantization effects change the meaning of the resulting routes. Instead of optimizing based on overall trends, the optimal route is being impacted by specific POI or even homes in the population data. The optimizer can still be applied to these regions however care must be taken to ensure there is sufficient smoothness to the utility distribution, so the effect of individual POI is minimized.

7.4.3 Modification of Utility Values

In general the optimization presented in this thesis can be applied to any region or any task. To ensure the outcome is useful, the utility distribution should be tailored in a way that the resulting route is meaningful and useful.

For instance in Section 7.4.2, a cities CBD was identified as a suboptimal region for analysis. This is true in a general because of the uniformity of the utility distribution, however a more specific task can still be solved. Consider a CBD that contains multiple end stations for existing regional transit. The optimizer in this thesis could be used to find a local route that effectively provided connection between these end stations and the POI in the CBD. Additionally, the utility value can be tailored to focus on workplaces in the CBD for weekday commuting routes. Weekend routes could be found by increasing the weight of recreational POI in the CBD. This would generate last mile routes that can be utilized by suburban residents on recreational trips.

The utility distribution would also need to be adapted to apply the optimizer to regional routes. Two good candidates for optimization are European rural areas and suburban regional routes. Both of these cases involving connecting the centers of different villages or towns over a wide area. However, to do so effectively, the utility value must be adapted to represent the need of regional trips. When developing the utility value for this thesis, the POI categories were weighted based on the average trip data; this system would not be effective for a regional route. Each town will offer basic services such as grocery stores and recreational areas. No transit riders will use regional transit route to utilize a service that is available locally. To account for this the utility value would need to be adapted to only recognize specialized services offered by each town. How this could be implemented is currently unknown.

Modifying the method used to generate the utility value is a powerful tool that allows the optimization process to be applied to a wide range of different regions and optimization scenarios. The definition of the utility should be shaped to represent the goals of the needed transit route. When a clear optimization goal is defined and the utility distribution supports this goal, the optimization process can identify routes that maximize the connectivity for most regions. Additionally, the optimization graph can be modified to find routes for any transit mode using fixed routes and stations.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

7.5 Future Work

The continuation of this research should primarily focus on improving the usefulness of the resulting solutions. The usefulness can be increased by the incorporation of additional data sources. These implementation improvements have been detailed previously in Section 7.2. The majority of the implementation improvements focus on utilizing higher resolution data to improve the modeling. Changes can be made to the models of the walking and driving networks to compute more accurate walking and driving times. These changes are feasible and can be implemented with sufficient programming time.

In addition to modeling improvements there are many improvements that can be made to the optimization process. The most impactful change would be to implement a delta evaluation strategy. Delta evaluation would enable larger instances to be solved due to a reduction in the objective value computation time. In order to employ delta evaluation, significant work must be completed with respect to algorithm development. Additionally, it may require changes to the formula for the objective value due to mathematical difficulties.

The final improvements that can be made to this analysis revolve around changes to the scope of the analysis. By expanding the scope of the analysis interesting results can be computed. A major aspect to consider is to allow the optimization process to modify existing transit routes or generate two or more routes as part of the analysis. These improvements would require significant structural changes to the layout of the optimization process and instance. Another structural change would be to modify the way constraints are applied to the optimization problem. By transitioning from parameter based constraints to cost based constraints a route could be tailored to budgetary limits for a particular problem.

The analysis done in this thesis was for a general route that would service all trips. While this is an ideal case, it does not attempt to solve a direct transit need. By changing the makeup of the utility value, the solution can be tailored towards a particular need such as first mile commuting, or school bus routes.

7.5.1 Modifications of Existing Routes

A natural progression of the optimization process would be to allow for the existing transit routes to be included in the optimization process. An ideal optimization tool would allow for the addition of two or more routes as well as extensions or modifications to the existing routes. There is already a need to change existing routes because the existing transit can be modified slightly to vastly improve the connectivity. The 134 bus route passes though the center of Winchester, MA however it does not stop in the town center. Additionally, there is no stop near the regional rail station Winchester Center. There are two nearby stops on the 134 bus, one on each side of the town center however both stops are around 500 meters from the commuter rail station.

To allow for the modification of existing transit, an overhaul the to route modeling process is required. Currently, the route is modeled as a simple loop though all the stops in the solution. To optimize over multiple routes, the proposed solution would

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

need to be modeled as a graph so that shortest paths can be computed between routes that intersect at arbitrary locations. Additionally, the solution representation would need to be updated to signify the different routes and how they interact. With these changes, existing transit can be modeled by generating an initial solution that matches the existing transit for the town. This starting point would model the as-built transit that can then be modified. Finally, the connections to the external regions would also have to be modeled. It is not immediately clear how this can be done in a flexible way.

Changes would also need to be made to the optimization algorithm. The connonical idea of combining all routes into a single large solution would result in unreasonably larger neighborhood due to the number of vertexes in the solution. Additionally, if one assumes that the existing transit routes are already close to a local optimum, large portions of the neighborhood would provide no improvement and would have to be considered anyway. In this case it may be beneficial to utilize an Alternating Optimization method, where only one route is considered for optimization at any time. This would reduce the neighborhood size and potentially allow for the optimization process to focus on the routes where significant improvement is possible.

7.5.2 Change of Solution Limits

In this project hard limits were placed on the number of stops and the total route travel time. While this was effective, the usefulness of these limits is small. A more useful limit process would be to define a total budget for the route. With a maximum cost for the route, the headway could be set as an optimization parameter and connected to running costs by computing the required number of drivers and fuel/maintenance costs for the route.

Additionally, fixed costs for the route could be modeled by estimating costs for each new bus station and the number of buses required to provide the desired level of service. By relating these fixed costs and the operating costs, the proposed routes can be limited to only those that fit within some pre-defined budget.

Lastly, it may be possible to model the revenue of the proposed route. This could be done by estimating the number of riders based on the service area combined with the population data. All of these changes required a level of transit planning knowledge that was not possible for this analysis.

7.5.3 Future Analysis Tasks

The framework developed in for this thesis offers a high degree of flexibility that can be used to support many types of analysis. Two other scenarios were considered for analysis, neither require changes to the framework, only manipulation of the instance dataset. Firstly a school bus replacement network could be considered. This network is an interesting approach because the town already offers a school bus service for all grade levels. The town budget for 2026 shows that the expected transportation costs is on the order of one million dollars for the entire network [6]. Reallocation of this funding would allow for the net cost of the bus network to be reduced by a large factor. To model

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

such a network, the instance can be adjusted to prioritize the schools as POI and the population data can be adjusted to only represent the areas where school aged children live.

The second scenario focuses on providing better connections to the existing transit in the town. The town has multiple exiting transit routes that pass though it and continue to major transport hubs or business centers. These transit routes are utilized by the working population of the town, however up to 60% of the riders travel the first mile via car [2]. A route could be created to minimize these trips and allow for more commuter riders by connecting the population to the transit stations. To model this the instance could be modified to only included the external utility and the population data. This would focus the route on connecting the population to the existing transit stops.

Another analysis not relating to the current case study is research into the relationship between waiting time, connectivity improvement and region size. From the results in Section 6.2.5, one can see that the route length increases, and the objective improvement decreases as waiting time increases. This would indicate that longer waiting times are ineffective on small service regions regardless of route shape. This optimization framework could be used to study this effect a potentially determine recommendations for maximum waiting times based on service region size.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

Chapter 8

Conclusion

The results presented by this thesis show that it is possible to generate new transit routes using computerized search methods. The instance generation and optimization process lined out in this thesis were applied to the case study region of Winchester, Massachusetts. The route developed for the instance, increased the connectivity of the region by an average of 16%. Additionally, the suggested route is similar to a manually designed route, connecting the same regions and provideding access to the existing transit in the town. The access to existing transit would allow the proposed route to offer a first mile service that could reduce the up to 60% of regional rail riders that utilize a car as part of the first mile. This connection to local transit was incorporated into the route without explicitly programming a reward for these types of connections. The application of the utility value and the connectivity metric enabled the optimizer to find these connections naturally.

The connectivity valued developed for this thesis, was able to model the effectiveness of the transit system in the town. By utilizing travel times as a primary parameter of the metric allowed for the effects of the network to be correctly modeled. This separates the connectivity parameter from existing transit accessibility metrics, which typically focus on local access to stations and service quality. The effectiveness of the connectivity metric was also due to the utility valued developed to model the socio-economic value of each point in the town.

The utility value used in this thesis, models the value of any point in the region by computing a value based on population data, points of interest data and travel habits. The value was constructed using open source data, and it generated a successful model with high traffic areas such as the town center, apartment complexes and business district having comparatively high utility.

The consistency of the routes generated for the case study and test scenarios show that the optimization algorithm is effective at finding the optimal solution. The GVNS optimization algorithm was able to find solutions despite the relative simplicity of the meta-heuristics and the large instance size, with over 1600 vertexes in the optimization graph. The algorithm performance improved due to both the design of the search neighborhoods and the performance optimization work done to improve computational efficiency.

The performance improvements were necessary due to the major drawback of the connectivity parameter. When the connectivity and therefor the objective value is computed, the shortest paths between all pairs of points in the instance need to be

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

recomputed. The incorporation of the shortest paths enable the effectiveness of the connectivity parameter however it also makes the computation process challenging. A combination of algorithmic, programming and GPU utilization allow for the computation time of the objective function to be reduced to reasonable levels.

The routes generated for the cases study, satisfy the design goals, however they fall short of criteria set by transit planners. Numerous simplification were utilized to reduce the complexity of the instance generation and optimization process. The complexity of the modeling and optimization process can be increased by adding factors such as potential ridership, overall route costs, complex path finding and more. With the addition of these complexities, the routes produced by the optimization process would be closer to meeting the needs of transit planners.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Bibliography

[1] Sh Afandizadeh, Hassan Khaksar, and Navid Kalantari. Bus fleet optimization using genetic algorithm a case study of Mashhad. *International Journal of Civil Engineering*, 11(1):43–52, 2013. Publisher: International Journal of Civil Engineering.

[2] Massachusetts Bay Transportation Authority. OPMI Presents the latest MBTA System-Wide Passenger Survey Results, April 2024.

[3] Geoff Boeing. Modeling and Analyzing Urban Networks and Amenities with OSMnx, 2025. _eprint: 2505.00736.

[4] Kuan Butts. peartree, 2021. Publication Title: GitHub repository.

[5] Shushan Chai and Qinghuai Liang. An Improved NSGA-II Algorithm for Transit Network Design and Frequency Setting Problem. *Journal of advanced transportation*, 2020(1):2895320, 2020. Publisher: Wiley Online Library.

[6] Winchester School Committee. FY26 Public Hearing Slidedeck, January 2025.

[7] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. 24.3 - Dijkstra's algorithm. In *Introduction to algorithms*, pages 595–601. MIT press, 2 edition, 1990.

[8] Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. 26.2 - The Floyd–Warshall Algorithm. In *Introduction to algorithms*, pages 558–565. MIT press, 1 edition, 1990.

[9] Elizabeth Cregger, Maureen Meister, Drew Bottaro, Charles Tseckares, and Jacqueline Welch. Offical Zoning Map Town of Winchester, MA, December 2015.

[10] Open Mobility Data. MBTA GTFS December 13 2023, December 2023.

[11] Javier Durán-Micco and Pieter Vansteenwegen. A survey on the transit network design and frequency setting problem. *Public Transport*, 14(1):155–190, March 2022.

[12] Sven Erlander and Neil F Stewart. *The gravity model in transportation analysis: theory and extensions*, volume 3. Vsp, 1990.

[13] Lang Fan and Christine L. Mumford. A metaheuristic approach to the urban transit routing problem. *Journal of Heuristics*, 16(3):353–372, June 2010.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

[14] Fedral Highway Administration. 2022 NextGen National Household Travel Survey Core Data, 2022. Available online: http://nhts.ornl.gov.

[15] Pedro José Farias Fernandes. Accessibility Index Elaboration by Network Geospatial Analysis. *Mercator*, 16, 2002.

[16] Transport for London. Assessing transport connectivity in London. Technical report, Transport for London, April 2015.

[17] Bundesamt für Raumentwicklung ARE. ÖV-Güteklassen – Berechnungsmethodik ARE. Technical report, Bendesamt fuer Raumentwicklung ARE, January 2022.

[18] Mintesnot Gebeyehu and Shin-ei Takano. Demand responsive route design: GIS application to link downtowns with expansion areas. *Journal of Public Transportation*, 11(1):43–62, 2008. Publisher: Elsevier.

[19] Valérie Guihaire and Jin-Kao Hao. Transit network design and scheduling: A global review. *Transportation Research Part A: Policy and Practice*, 42(10):1251–1273, 2008. Publisher: Elsevier.

[20] Andrew Holliday, Ahmed El-Geneidy, and Gregory Dudek. Learning Heuristics for Transit Network Design and Improvement with Deep Reinforcement Learning. *arXiv preprint arXiv:2404.05894*, 2024.

[21] Holger H. Hoos and Thomas Stützle. 10 - Other Combinatorial Problems. In Holger H. Hoos and Thomas Stützle, editors, *Stochastic Local Search*, The Morgan Kaufmann Series in Artificial Intelligence, pages 467–525. Morgan Kaufmann, San Francisco, 2005.

[22] Guangjun Huang, Tiancheng Ming, Wutong Wang, Lixia Yan, and Saru Kumari. Urban Train Timetable Optimization Based on Multi-Objective Optimization. In *Intelligent Transportation and Smart Cities*, pages 3–11. IOS Press, 2024.

[23] Vasileios Kourepinis, Christina Iliopoulou, Ioannis Tassopoulos, and Chrysanthi Aroniadi. An Improved Particle Swarm Optimization Algorithm for the Urban Transit Routing Problem. *Electronics*, 12:3358, August 2023.

[24] Changyu Li, Ran Ge, Xiangcheng Wu, and Alaa Khamis. Optimal Placement of Bus Stops using Particle Swarm Optimization. In *2023 IEEE International Conference on Smart Mobility (SM)*, pages 15–20, 2023.

[25] Guangchun Li, Lei Nie, Feng Gao, and Zhenhuan He. Optimization of bus stop layout considering multiple factors including passenger flow direction. *PloS one*, 19(11):e0313040, 2024. Publisher: Public Library of Science San Francisco, CA USA.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

[26] Yi Liu, Xuesong Feng, Lukai Zhang, Weixing Hua, and Kemeng Li. A pareto artificial fish swarm algorithm for solving a multi-objective electric transit network design problem. *Transportmetrica A: Transport Science*, 16(3):1648–1670, 2020. Publisher: Taylor & Francis.

[27] Christoph E Mandl. Evaluation and optimization of urban public transportation networks. *European Journal of Operational Research*, 5(6):396–404, 1980. Publisher: Elsevier.

[28] OpenStreetMap contributors. Planet dump retrieved from https://planet.osm.org, 2017. Published: https://www.openstreetmap.org.

[29] Ingo Pastl and Danilo Araújo. Optimization of Public Transport Networks by Considering Alternative Positions for Network Stations. *IEEE Latin America Transactions*, 22(6):468–474, 2024.

[30] Rohit Rathod, Gaurang Joshi, and Shriniwas Arkatkar. Composite Accessibility Index: A Novel and Holistic Measure for Evaluating Transit Accessibility. *Transportation Research Record*, 2679(2):1359–1378, 2025. _eprint: https://doi.org/10.1177/03611981241270156.

[31] Remix. partridge, 2023. Publication Title: GitHub repository.

[32] D. C. Rich. Population Potential, Potential Transportation Cost and Industrial Location. *Area*, 10(3):222–226, 1978. Publisher: Royal Geographical Society (with the Institute of British Geographers), Wiley.

[33] Johannes Rossbacher and Markus Seidl. Die Österreichweiten ÖV-Güteklassen Rahmen, Struktur & Beispiele. Technical report, Österreichische Raumordnungskonferenz, November 2022.

[34] Haitao Su, Menghan Li, Xiaofeng Zhong, Kai Zhang, and Jingkai Wang. Estimating Public Transportation Accessibility in Metropolitan Areas: A Case Study and Comparative Analysis. *Sustainability*, 15(17), 2023.

[35] Vukan R Vuchic. Capacity, Speed, Accelerated and Special Operations. In *Urban transit: operations, planning, and economics*, page 108. John Wiley & Sons, 2017.

[36] Vukan R Vuchic. *Urban transit: operations, planning, and economics*. John Wiley & Sons, 2017.

[37] Sini Wang, Zhongyi Zuo, and Yan Liu. Study on Location of Bus Stop in Subway Service Area Based on Residents' Travel Accessibility. *Sustainability*, 15(5), 2023.

[38] Eric Weber, Jessica Moehl, Spencer Weston, Amy Rose, Christa Brelsford, and Taylor Hauser. LandScan USA 2021, 2022. Section: July 09, 2022.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

[39] Haoyang Yan, Zhiyong Cui, Xinqiang Chen, and Xiaolei Ma. Distributed multiagent deep reinforcement learning for multiline dynamic bus timetable optimization. *IEEE Transactions on Industrial Informatics*, 19(1):469–479, 2022. Publisher: IEEE.

[40] Ruqin Yang, Yaolin Liu, Yanfang Liu, Hui Liu, and Wenxia Gan. Comprehensive public transport service accessibility index—A new approach based on degree centrality and gravity model. *Sustainability*, 11(20):5634, 2019. Publisher: MDPI.

[41] Sunhyung Yoo, Hyun Kim, and Jinwoo Lee. Combining Reinforcement Learning With Genetic Algorithm for Many-To-Many Route Optimization of Autonomous Vehicles. *IEEE Access*, 12:26931–26942, 2024.

[42] Sunhyung Yoo and Jinwoo Lee. Revising bus routes to improve access for the transport disadvantaged: A reinforcement learning approach. *Journal of Public Transportation*, 25, March 2023.

[43] Jingqiao Yu, Yuan Zhu, Chaowen Wu, and Guoqing Song. Optimization of Bus Stop Location based on Spatial Analysis and K-Means Clustering Method–A Case Study in Hohhot City. *Research Square*, 2024.

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

Appendix A

Reference Information

A.1 Overview of Generative AI Tools Used

No generative AI tools were used in the creation of this thesis. During the literature review the AI tool SciSpace was used to search for and summarize existing scientific literature. Any statements in this thesis relating to existing literature were confirmed by referencing the source material.

A.2 Source Code

The source code for this thesis is available on GitHub.

GTFSQuerys	https://github.com/jamesmulhern/GTFSQuerys.jl
GTFSGraphs	https://github.com/jamesmulhern/GTFSGraphs.jl
Instance Generation	https://github.com/jamesmulhern/InstanceGeneration
Optimization	https://github.com/jamesmulhern/Transit_Route_Optimization

A.3 OSM Graph Filters

Waling Graph Filters

```
["highway"]["area"!~"yes"]["access"!~"private"]
["highway"!~"abandoned|bus_guideway|construction|cycleway|motor|no|
planned|platform|proposed|raceway|razed"]
["foot"!~"no"]["service"!~"private"]
["highway"!~"service"]
```

Road Graph Filters

```
["highway"]["area"!~"yes"]["access"!~"private"]
["highway"!~"abandoned|bridleway|bus_guideway|construction|
corridor|cycleway|elevator|escalator|footway|no|path|
pedestrian|planned|platform|proposed|raceway|razed|service|
steps|track"]
```

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN

```
["motor_vehicle"!~"no"]["motorcar"!~"no"]
["service"!~"alley|driveway|emergency_access|parking|
parking_aisle|private"]
```

A.4 OSM Mapping

Key	Value	Label	Category
shop	*	Shopping	Shopping_Errands
tourism	*	Recreational	Social_Recreational
office	*	Work	Work
leasure	*	Recreational	Social_Recreational
natural	beach	Recreational	Social_Recreational
landuse	cementary	Recreational	Social_Recreational
amenity	bar	Meals	Meals
amenity	biergarten	Meals	Meals
amenity	cafe	Meals	Meals
amenity	fast_food	Meals	Meals
amenity	food_court	Meals	Meals
amenity	ice_cream	Meals	Meals
amenity	pub	Meals	Meals
amenity	restaurant	Meals	Meals
amenity	college	School	School_Daycare_Religious
amenity	dancing_school	Recreational	Social_Recreational
amenity	driving_school	Recreational	Social_Recreational
amenity	first_aid_school	Medical	Medical_DentalServices
amenity	kindergarten	Daycare	School_Daycare_Religious
amenity	language_school	Recreational	Social_Recreational
amenity	library	Recreational	Social_Recreational
amenity	surf_school	Recreational	Social_Recreational
amenity	toy_library	Recreational	Social_Recreational
amenity	research_institute	Work	Work
amenity	music_school	Recreational	Social_Recreational
amenity	school	School	School_Daycare_Religious
amenity	university	School	School_Daycare_Religious
amenity	atm	Errands	Shopping_Errands
amenity	payment_terminal	Errands	Shopping_Errands
amenity	bank	Errands	Shopping_Errands
amenity	bureau_de_change	Errands	Shopping_Errands
amenity	money_transfer	Errands	Shopping_Errands
amenity	payment_centre	Errands	Shopping_Errands
amenity	clinic	Medical	Medical_DentalServices
amenity	dentist	Medical	Medical_DentalServices

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.

amenity	doctors	Medical	Medical_DentalServices
amenity	hospital	Medical	Medical_DentalServices
amenity	nursing_home	Medical	Medical_DentalServices
amenity	pharmacy	Medical	Medical_DentalServices
amenity	social_facility	Social	Social_Recreational
amenity	veterinary	Errands	Shopping_Errands
amenity	arts_centre	Recreational	Social_Recreational
amenity	brothel	Recreational	Social_Recreational
amenity	casino	Recreational	Social_Recreational
amenity	cinema	Recreational	Social_Recreational
amenity	community_centre	Recreational	Social_Recreational
amenity	conference_centre	Recreational	Social_Recreational
amenity	events_venue	Recreational	Social_Recreational
amenity	exhibition_centre	Recreational	Social_Recreational
amenity	gambling	Recreational	Social_Recreational
amenity	love_hotel	Recreational	Social_Recreational
amenity	music_venue	Recreational	Social_Recreational
amenity	nightclub	Recreational	Social_Recreational
amenity	planetarium	Recreational	Social_Recreational
amenity	public_bookcase	Recreational	Social_Recreational
amenity	social_centre	Recreational	Social_Recreational
amenity	stage	Recreational	Social_Recreational
amenity	stripclub	Recreational	Social_Recreational
amenity	studio	Recreational	Social_Recreational
amenity	swingerclub	Recreational	Social_Recreational
amenity	theatre	Recreational	Social_Recreational
amenity	courthouse	Errands	Shopping_Errands
amenity	fire_station	N/A	N/A
amenity	police	N/A	N/A
amenity	post_office	Errands	Shopping_Errands
amenity	townhall	Errands	Shopping_Errands
amenity	funeral_hall	Errands	Shopping_Errands
amenity	grave_yard	Recreational	Social_Recreational
amenity	internet_cafe	Social	Social_Recreational
amenity	marketplace	Shopping	Shopping_Errands
amenity	monastery	Religious	School_Daycare_Religious
amenity	place_of_worship	Religious	School_Daycare_Religious
amenity	public_bath	Recreational	Social_Recreational
amenity	public_building	Errands	Shopping_Errands

Die approbierte gedruckte Originalversion dieser Diplomarbeit ist an der TU Wien Bibliothek verfügbar
The approved original version of this thesis is available in print at TU Wien Bibliothek.
TU Bibliothek Your knowledge hub WIEN