Journey Planning Considering Passengers' Preferences Using Genetic Algorithm

A minor thesis submitted in partial fulfilment of the requirements for the degree of
Master of Computer Science

Emil Brøgger Kjer
School of Computer Science and Information Technology
Science, Engineering, and Technology Portfolio,
RMIT University
Melbourne, Victoria, Australia

March 22, 2012

Declaration

This thesis contains work that has not been submitted previously, in whole or in part, for any other academic award and is solely my original research, except where acknowledged.

The base genetic algorithm is implemented using the Pyevolve package version 0.6 by Christian S. Perone, available from `http://pyevolve.sourceforge.net/`.

This work has been carried out since February 2011, under the supervision of Dr. Xiaodong Li and Dr. Margaret Hamilton.

Emil Brøgger Kjer
School of Computer Science and Information Technology
RMIT University
March 22, 2012

Acknowledgements

I am thankful to my supervisors Dr. Margaret Hamilton and Dr. Xiaodong Li whose encouragement, guidance and support enabled me to write this thesis.

I thank the Evolutionary Computation and Machine Learning Group for their critical and constructive comments about this project.

Contents

List of Figures

List of Tables

Abstract

This thesis is concerned with developing a multimodal journey planner that takes passengers' preferences into account. In the proposed model, Genetic Algorithm is adopted to evolve better journey solutions, with the crossover and mutation operators that are specifically designed according to journey planning needs. In addition, Depth First Search algorithm is used and modified to generate feasible initial solutions. The proposed model evolves several alternative journey solutions considering passengers' preferences such as the shortest time and fewest changes. We evaluate our model by using a real world data set from train and tram services in Melbourne, Australia. We demonstrate that the proposed model is capable of producing and recommending several alternative journey solutions to passengers, hence making it realistic for practical use. In comparison with an existing commercial journey planner, a distinct advantage of the proposed model is that it allows several alternative solutions to be produced for a multimodal transportation network.

Chapter 1

Introduction

With the introduction of carbon taxes, decreasing oil resources and urbanisation in countries around the globe, the public transport sector faces rapidly increasing demands. Public transport providers must improve and update their facilities to meet current demands and encourage people to use public transport as an alternative to cars [22]. In order to compete with cars, public transport must allow passengers flexibility, both in their timetabling and in their choice of routes. Therefore, included in these facilities are journey planners.

In Melbourne, commuters have the choice of taking the train, tram or bus into the city. Many of these routes overlap and cross each other enabling multimodal transport. Journey planning for public transport must combine timetables of multimodal transportation and take passengers' preferences into account. The combination of routes is knows an NP-hard problem [25], and passengers therefore need help to plan their journeys [31]. This can mainly be achieved done by using search algorithms.

Traditionally journey planning systems are provided by either the company that provides the transportation facilities or by a limited number of timetabling companies such as Hacon [8, 31, 36]. That journey planners are provided by private entities introduces complications for researchers because their algorithms are protected by patents. Consequently, a researcher may need to invent alternatives to contribute and improve journey planning systems. A new trend for journey planning systems is to enable the passenger to specify individual preferences. Passengers' preferences may be for instance: unimodal transportation; fewest changes between modes; a minimal carbon footprint or shortest walking distance [18, 29]. A recent study has shown current journey planners mostly provide a single journey based on shortest time taken to travel the route. The same study found that there is a need for journey planning systems that provide multiple journey options for different passengers' preferences [36].

This research aims to develop a multimodal journey planner for public transport in a metropolitan city. The model will provide the traveler with multiple journey options based on different passengers' preferences.

1.1 Research Questions

This research aims to address the following key research questions:

1. How can a journey planner that caters for different passengers' preferences be modeled?
2. How can a journey planner provide multiple journey options that are all useful in practice?
3. How is a journey planner affected by the number of intersecting transportation routes?
4. How does a journey planner using an evolutionary approach compare with a commercial journey planner?

1.2 Methodologies

Journey planning is an optimisation problem that has been extensively studied [1, 5, 13, 32]. Generally speaking, the key objective is to find the 'optimal' journey for a passenger intending to travel between two locations using public transportation. Among others, evolutionary computing techniques have been applied to solve the problem [1, 5]. Evolutionary techniques are inspired by the evolution process observed in nature and they have been applied extensively to optimisation problems [10]. Some representative examples of evolutionary computing algorithms are: Ant Colony Optimisation [10], Particle Swarm Optimisation [10] and Genetic Algorithm [6, 24]. We hypothesise that Genetic Algorithm (GA) can be used to solve the optimisation problem. GA has the advantages of being a probabilistic and population based algorithm [6, 24]. The algorithm can provide various solutions for the traveler, not only one optimal solution. The initial candidate solutions are generated with domain specific knowledge, further discussed in chapter 3.

Metropolitan transportation networks combine multiple transportation modes such as bus, train and tram. The choice of data structure to represent these modes is therefore important [16, 28]. This research is applied to existing transportation networks where we take

advantage of the way graphical data structures can be modelled to these kind of problems [1, 2, 5, 25, 32]. This is inspired from the related problems Travelling Sales Man Problem and Vehicle Routing Problem that have several successful applications using the graph data structure [16, 28]. A graph data structure is composed of vertices and the relations of the vertices to each other [4]. The graph data structure enables us to add new transportation mode types independently of the model. To search for journeys in this data structure we use Depth First Search and a modified version of Depth First Search. The modifications of Depth First Search enable us to find appropriate journeys with our model.

In this research we use a real world dataset of the current routes and timetables for trains and trams of the metropolitan city Melbourne in Australia.

The main contribution of this thesis is the development of a journey planner that demonstrates how passengers' preferences can be satisfied utilising GA on a real world dataset.

1.3 Outline

The rest of this thesis is organised as follows: chapter 2 introduces the background of journey planning; outlines what a graph data structure is; explains GA and a defines the minimisation problem. In chapter 3 we discuss our model with an overview introducing each component, followed by an in-depth description of journey planning and related issues. The last section of chapter 3 is a comprehensive description and illustration of our model. In chapter 4 we show experiments and analysis of the results. Chapter 5 concludes the thesis with a summary and future works.

Chapter 2

Background

The aim of this chapter is to provide a basic understanding of the background of journey planning. This chapter will cover: a description of transportation networks in relation to a real world dataset; what a journey is and give examples of how journey planners can take passengers' preferences into account; a description of what a graph data structure is and a definition of the minimisation problem of this journey planner.

2.1 Journey Planning and Public Transport

Journey planning is, as the name indicates, the search for a plan for a journey from an origin to a destination and it is therefore a search problem [1, 5, 13, 32]. *Public transport journey planning* is a special case of journey planning which involves one or more transportation modes that are provided to the public such as bus, train or taxi. To understand a journey planning search we will first define what transportation modes and transportation networks are. In the rest of the thesis we use "journey planning" as synonym for "public transport journey planning".

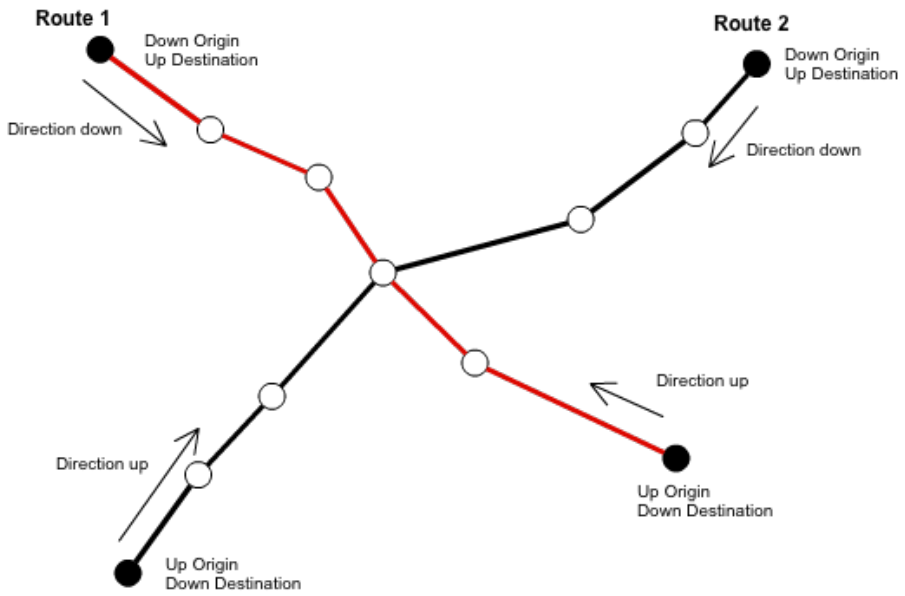


Figure 2.1: Example of a simple transportation network.

2.1.1 Transportation Modes and Transportation Networks

A transportation mode is a specific type of transportation, i.e. train, tram, bus or walking. Transportation modes have different features and traveling speeds. A transportation network has one or more routes which a transportation vehicle travels on. A simplified example of a transportation network is illustrated in Figure 2.1. The network in the figure can be of any type and represents two transportation routes: Route 1 and Route 2. Each route has a beginning and an end location (also known as origin and destination position) and a number of intermediate stop positions here marked with white circles. Routes are split into two sub-routes going between the same two end locations, but in opposite directions, either up or down. This is indicated in the figure with the arrows labeled "Direction down" and "Direction up". A path has two stops, a source and an adjacent stop. Stops are unique and each pair of stops has a path between them. As shown in the figure, a single stop can be associated with multiple paths. The composition of consecutive paths forms the routes. The time it takes to travel from a source stop to an adjacent stop depends on the transportation mode. Routes follow fixed timetables.

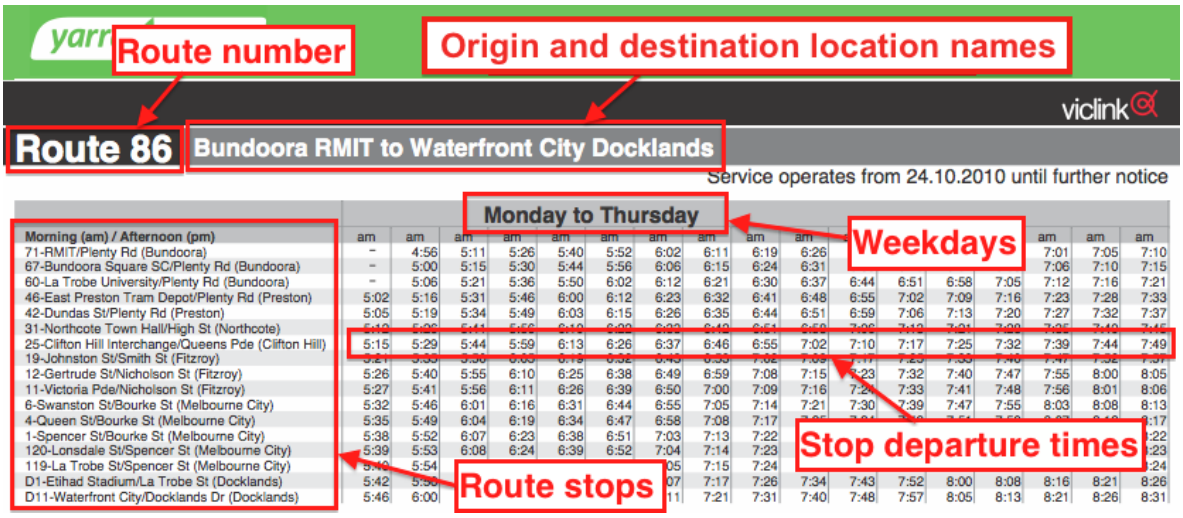


Figure 2.2: Timetable of tram route 86 [18].

In Figure 2.2, timetables for a real world tram route are shown [18]. Each stop has multiple fixed departure times during the week.

Transportation networks of different modes can be combined to represent a single transportation network [1]. For instance in Figure 2.1 the red *Route 1* would be representing one transportation mode where the black *Route 2* could represent another transportation mode. By combining multiple transportation networks the complexity is further increased. This amplifies the argument for the need of a journey planner to guide travelers. Unimodal involves a single transportation mode, and multimodal is defined as using two or more modes[1]. Modes can either refer to different vehicles of the same transportation type or different transportation types like car, bus, train or walking.

[1]New Oxford American Dictionary 3rd edition 2010 by Oxford University Press, Inc.

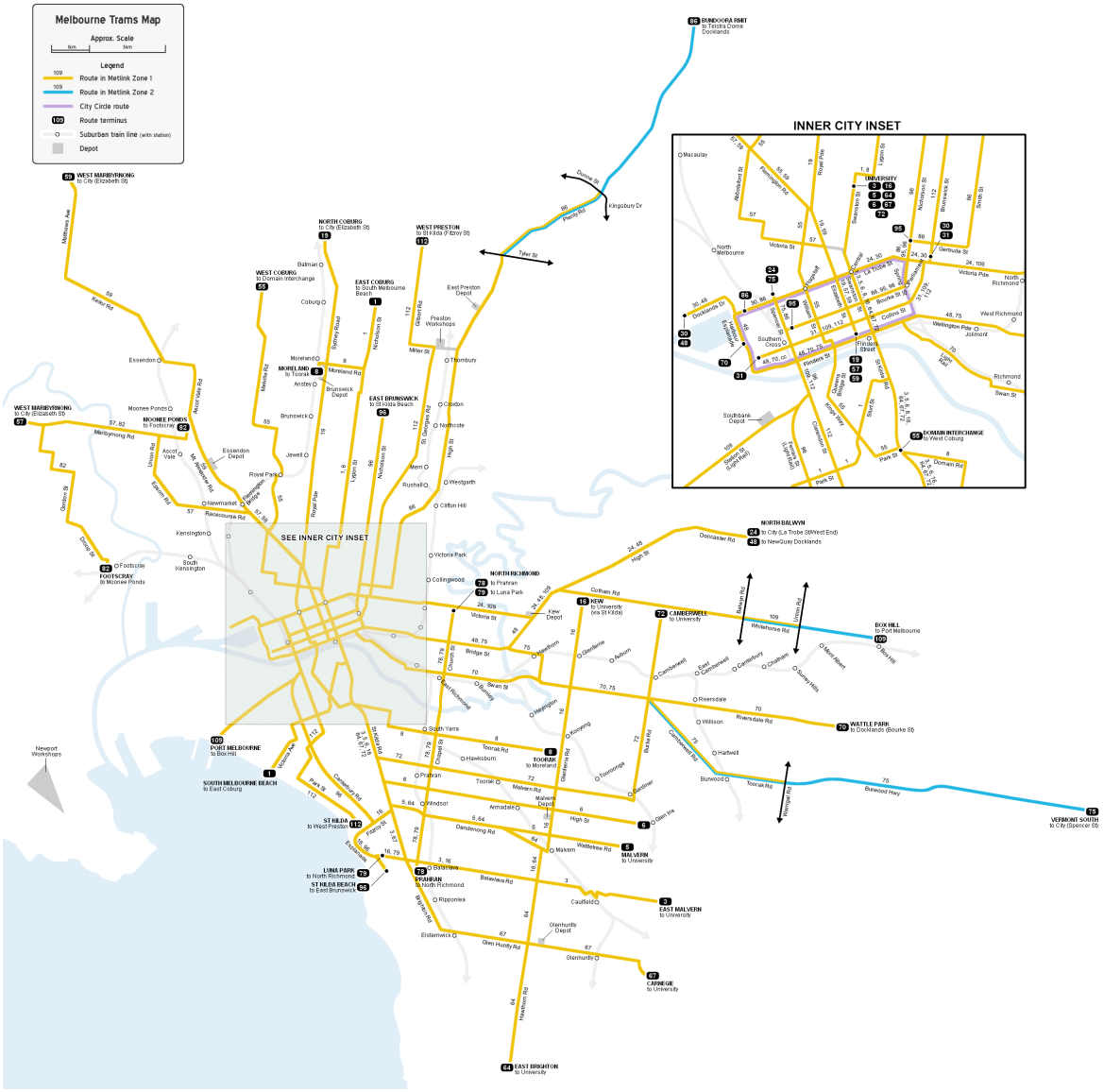


Figure 2.3: Map of the tram network in Melbourne [27].

A real world transportation network is shown in Figure 2.3. The network represents a unimodal transportation mode (tram in Melbourne), where it can be seen that the combination of routes is complex.

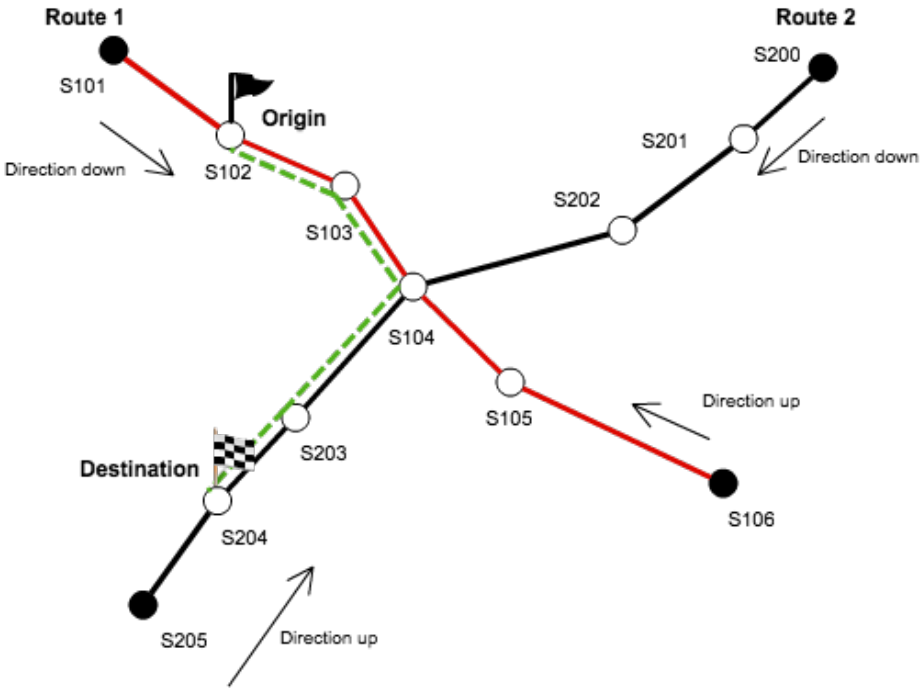


Figure 2.4: Example of a journey from origin S102 to destination S204.

2.1.2 Journey Planning

A journey consists of a combination of partial route sections, the paths. The paths are ordered so that by following these in consecutive order they form a route from an origin to a destination. Journeys have a start time and an arrival time, where the arrival time is dependent on the timetables for each individual path. Journey planning is therefore the combination of various paths according to their timetables.

2.1.3 An Example

The following example clarifies what a journey is and how paths are combined. A transportation network containing two transportation routes *Route* 1 and *Route* 2 is shown in Figure 2.4. Each stop is labeled and it should be noted that the stop $S104$ is shared by both routes. A sample journey, J, is indicated by the green dashed line, with origin at stop $S102$ and destination $S204$. The journey is composed of the stops: $J = [S102, S103, S104, S203, S204]$.

Execution of the journey goes as follows:

1. Start at stop $S102$ and wait until departure on the next transportation vehicle on $Route1$. Note, the adjacent stop is $S103$ so the route direction is *down*.
2. Stay on this transportation mode until stop $S104$, get off the current vehicle and wait until a vehicle departs on $Route2$ in direction *down*.
3. Get off at stop $S204$ and the journey ends.

2.2 Passengers' Preferences

A passenger is an individual who wants to travel from an origin to a destination and is a user of journeys. Passengers have travel preferences which can be subjective and independent for each passenger. We will focus on the preferences that are most relevant to passengers:

- To be able to specify the date and time of departure.
- To have alternative journey options.
- To travel from one location to another location in the shortest time.
- To travel from one location to another with fewest transportation mode changes.

2.3 Commercial Journey Planners and Research

Commercial journey planners for public transport combining multiple transportation modes have existed for many years [13]. The two major companies providing journey planners are Hacos [8] and Google Transit [7], but their journey planner algorithms are not publicly available. The journey planner by Hacos is used in more than 20 countries with cities including: Barcelona, New York, Copenhagen, Oslo and Melbourne [8]. This journey planner enables the traveler to optimise the journey according to a single preference like fewest vehicle changes, least walking or earliest arrival at the destination. The other journey planner is provided by Google Transit which has less features than Hacos' journey planner, but is widely used because it is freely available. The Google Transit journey planner is interesting because it enables researchers to build on top of it with extra features. As an example a research project by University of California named *PATH2Go Trip Planner* has been carried out for

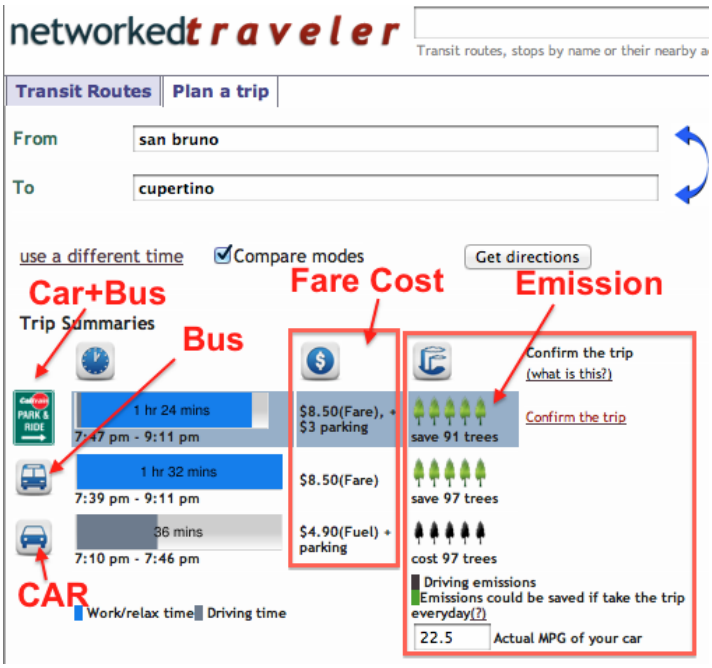
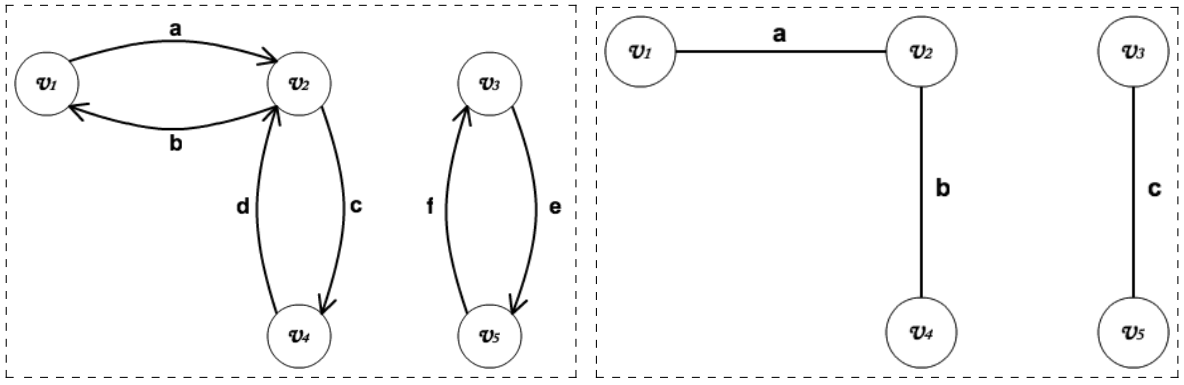


Figure 2.5: Journey Planner including fare cost and carbon emissions [29].

the San Francisco Bay Area [29, 35]. The metropolis has different individual transportation systems all using the Google Transit journey planner, and the PATH2Go interface allows the traveler to search for a journey across all the transportation systems. As shown in Figure 2.5, PATH2Go Trip Planner has a number of additional features: carbon emissions calculated by the number of saved trees in comparison to using a car; prices for traveling with the suggested journeys; the ability to save a selected journey to a mobile device; and receive real time updates of the journey if the transportation system has delays or changes. The next section explain journey planning in terms of graph theory.

2.4 Graphs

A graph, G, consists of a set of vertices, V, and a set of edges, E, connecting the vertices [4]. The edges have directions so that the edge $k(u, v)$ has the direction from vertex u to vertex v, where $u \neq v$. The edge $l(v, u)$ which has the direction from vertex v to vertex u, is the opposite direction to l. We call this kind of graph a *directed graph* or a *di-graph*.



(a) A directed graph.

(b) A undirected graph.

Figure 2.6: Graph examples.

Figure 2.6a is an example of a directed graph. The graph contains the vertices $[v_1, v_2, v_3, v_4, v_5]$ and edges $[a, b, c, d, e, f]$. An alternative graph can represent the same network. Since each edge represents both directions the edge $k(u, v)$ and $l(v, u)$ has $k = l$ for vertices u and v, $u \neq v$; so called *bi-directed* edges. This kind of graph is an un-directed graph and is illustrated for the same network in Figure 2.6b.

A path, $p_{(u,u')}$, from vertex u to u' contains vertices $p_{(u,u')} = [v_0, v_1, v_k]$, such that $u = v_0$ and $u' = v_k$, and $(v_{i-1}, v_i) \in E$ for $i = 1, 2, ..., k$. The path is only valid if each set of vertices has an edge such that u and u' are connected by the edges $(v_0, v_1), (v_1, v_2), ..., (v_{k-1}, v_k)$. In Figure 2.6 the sequence $[v_1, v_2, v_4]$ is a valid path whereas the path $[v_1, v_2, v_3]$ is invalid.

A group of connected vertices make up a component. The graphs in Figure 2.6 are composed of two individual components: $component_1 = [v_1, v_2, v_4]$ and $component_2 = [v_3, v_5]$. They are individual components because they have no edges to or from each set of vertices. Both components contain cycles (loops), where a cycle is formed if the path of vertices $[v_0, v_1, ..., v_k]$ contains at least one repeated vertex. In Figure 2.6 the path $[v_1, v_2, v_1]$ is a cycle, where the path $[v_1, v_2, v_4]$ is not a cycle.

2.5 Search Algorithms

For the data structure defined in section 2.4 many search algorithms are formulated. There are classical algorithms described by Cormen *et. al* [4] such as Bellman-Ford, Breath First Search and Depth First Search. These algorithms are guaranteed to find an optimal path[2]

[2]Smallest possible distance from a source to a target

by exhausting the complete search space. However, by exhausting the search space the time complexity is exponential [4]. To deal with this time complexity variations of the classical algorithms were developed, for example [26]: Dijkstra, A* and Alpha-Beta Pruning. These algorithms strive to find optimal paths without exhausting the complete search space by using a heuristic function [26].

2.5.1 Depth First Search

Depth First Search (DFS) is a classical path finding algorithm [4, 12]. It explores a graph in a depth first manner. It begins from the start vertex and expands the leftmost edges to adjacent vertices until the entire search space has been visited. To avoid visiting the same vertices more than once a bookkeeping of visited vertices is kept. By using this bookkeeping technique the algorithm is guaranteed to explore each vertex only once. Every time a vertex's edges are expanded the edge's target nodes are added to a stack if they are unexplored.

During the search a bookkeeping entry of the shortest path to a given vertex from the origin is kept. This bookkeeping entry is updated every time a new vertex is visited. By following the smallest values from a given vertex back to the start the shortest path is found. Another important feature of DFS is its completeness. An algorithm is defined as complete when it guarantees that if a path from the origin to the destination exists the algorithm will find it [4, 26].

Since it is impossible find an optimal journey in polynomial time, the journey planning problem is an NP-hard problem [25]. DFS has the running time $O(V + E)$ because it visits all vertices and edges. The journey planning problem will therefore require time complexity $O(n!)$ [10], where n is the number of vertices. By using a heuristic based search algorithm such as A* [26], the search for an optimal path can be performed with less complexity. As mentioned in the introduction the scope of this thesis is not to find the optimal journey, but find good journeys with alternative options for the passengers. By finding the optimal path, the same journey will be found every time the algorithm is run. Stochastic and population based search algorithms such as Genetic Algorithm (GA) can find alternative journeys and will be discussed in detail in the following section 2.6.

2.6 Genetic Algorithm

As mentioned in the introduction, GA is inspired by the natural selection process observed in nature. GA was developed by John Holland [11] and has been applied to many different optimisation problems. GA is one of many evolutionary algorithms, and is particularly well suited to this application based on the following arguments [1, 10]:

- the string representation of chromosomes is straightforward to model on discrete problems such as journey planning.
- it operates with a population of solutions.
- its probabilistic and non-deterministic structure promotes diverse solutions.
- it is known for its good performance on a wide range of problems.
- it does not require specific knowledge about the environment to operate.
- it is less prone to premature convergence.
- intermediate solutions gradually improve.
- it can provide alternative solutions, including near-optimal solutions.

```
Step 1: initialise population of candidate solutions
Step 2: evaluate fitness of individuals in the population
Step 3: while termination criteria not met do
                    select parents for reproduction
                    perform recombination and mutation of parents
                    evaluate fitness of individuals
        end do
Step 4: return feasible individuals
```

Figure 2.7: Genetic Algorithm - general version.

2.6.1 Detailed Description of Genetic Algorithm

In Figure 2.7 the outline of GA is shown as pseudo-code [1, 11]. Each line of the algorithm is explained in the following paragraph:

Step 1 The initial population is generated. The algorithm operates with a population of solutions which it optimises.

Step 2 The fitness of each individual in the population is evaluated. How the fitness is defined depends on the application of the algorithm.

Step 3 The termination criteria are evaluated. The termination criteria can, for instance, be based on stagnation of improvement in fitness or a fixed number of iterations. Two parents are selected for reproduction following some selection scheme. Two different manipulation mechanisms are in use: mutation by crossover or mutation of an individual. By crossover two parents are combined. The combination usually works by taking a part of each parent and swapping them to produce child individuals. The children might only be kept if they improve on the fitness of their parents. By mutation an individual is manipulated to reproduce a new part of the individual. These two manipulation mechanisms are performed until a certain amount of the population has been modified. The new population replaces the old, and the same process is repeated as in step 1. It is also possible to select elite individuals with the best fitness and keep these for the next generation; so called *elitism*. The fitness of the new population is evaluated and the algorithm repeats.

Step 4 If termination criteria are fulfilled the execution of the algorithm ends.

The time complexity of GA is highly dependent on the operations used in the algorithm [10]. In our case the mechanism of searching for paths will determine the time complexity since the other parts of the algorithm require either linear or constant time complexity.

2.7 Formulation of the Problem

A recent study of journey planning with a real world dataset from a major metropolis has been carried out [1]. The study covered a multimodal journey planner problem which can be defined as the following:

Let i, j and k be vertices in a graph, G, where the graph is formulated as in section 2.4. Each vertex has a label, l. The set of edge labels for each transportation mode between a set of vertices is denoted by L_E. The cost-function is defined as, $\omega : E \rightarrow R$, as the cost in time it requires to travel from vertex i to an adjacent vertex j. A binary value x denotes if path (i, j) is used in the journey. A journey has the vertex pair: an origin, O, and a destination, D. The determination of departure time at a given location, is denoted by τ.

The journey planning problem can not be defined as the minimisation problem:

$$min \sum_{l \in L_E} \sum_{i=0}^{D} \sum_{j=0}^{D} x_{ij}^{l} \omega_{ij}^{l} + \sum_{\substack{l,l' \in L_E \\ l \neq l'}} \sum_{\substack{i=0 \\ i \neq j,k}}^{D} \sum_{\substack{j=0 \\ j \neq i,k}}^{D} \sum_{\substack{k=0 \\ k \neq j,i}}^{D} (\tau_{jk}^{l'} - (\tau_{ik}^{l} + \omega_{ij}^{l})) \tag{2.1}$$

Subject to:

$$\sum_{l \in L_E} \sum_{\substack{j=0 \\ j \neq i}}^{D} x_{ij}^{l} - \sum_{l \in L_E} \sum_{\substack{j=0 \\ j \neq i}}^{D} x_{ji}^{l} \begin{cases} -1 & i = D \\ 0 & i \neq D, 0 \\ 1 & i = 0 \end{cases} \tag{2.2}$$

$$x_{ij}^{l} \in \{0, 1\} \tag{2.3}$$

The cost function, ω, depends on the mode of transportation and can be defined as the average speed of the transportation mode. ω maybe larger for traveling the same distance by foot than by train. However, there might be indirect waiting times related to some transportation modes and not others. The waiting time in (2.1) is defined by the $(\tau_{jk}^{l'} - (\tau_{ik}^{l} + \omega_{ij}^{l}))$. Traveling from location i to k via location j entails a waiting time for departure from location j minus the departure from i plus the weight cost. The problem is by constraint (2.2) and constraint (2.3) to make sure loops do not occur [1].

Summary

This chapter has provided the background theoretical knowledge for journey planning. We have seen that current journey planning can incorporate carbon emissions and journey cost. Graphs are composed of vertices that are connected by edges and search algorithms can find paths in graphs. GA has been described in detail. The last section of this chapter showed how graph representation can be used to represent the journey planning minimisation problem. The next chapter will explain how we model our journey planner to find useful journeys in a public transportation network using real world data.

Chapter 3

Journey Planning

This chapter will give an in-depth description of our model of a journey planner. We will start by giving an overview of the components of our journey planner and how each part of the model interacts. We explain how our data structure is implemented and show how loops in journeys can be eliminated. We have modified the DFS algorithm and these modifications will be described. The last section outlines how we define GA's crossover and mutation operators that are specifically designed for the journey planning model.

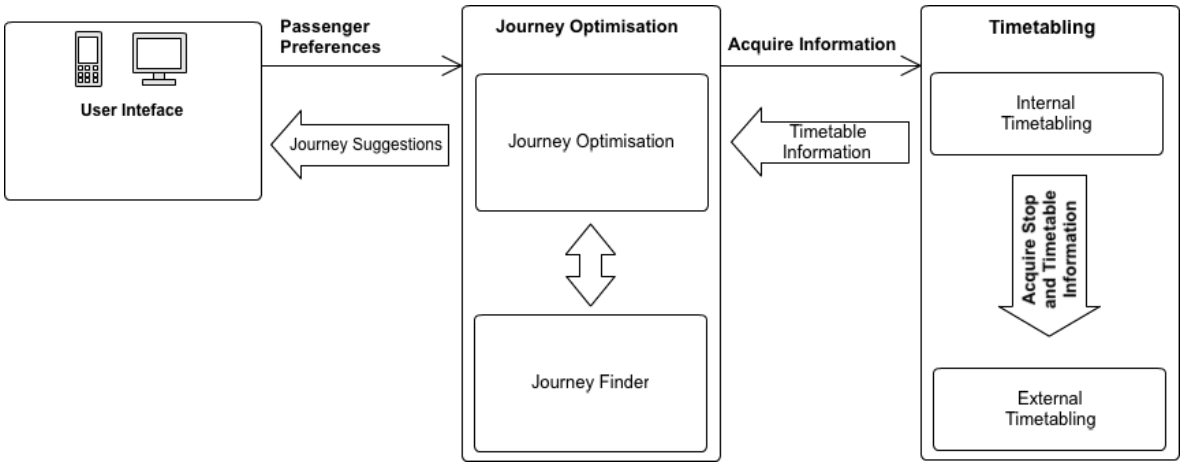


Figure 3.1: Overview of the journey planning model.

3.1 Description of the Model

Our model of a journey planning system is composed of different components that need specific considerations. To give an overview of this system the model is illustrated in Figure 3.1. It is composed of three parts that interact with each other. The *User Interface*-component invokes the *Journey Optimisation*-component with journey settings such as: origin, destination, passenger's preferences and time of day. The *Journey Optimisation*-component then finds journeys with information from the *Timetabling*-component. The *User Interface*-component can, for instance, be represented as an application, a website or a mobile application. Our journey optimisation model is located in the *Journey Optimisation*-component. This component is split into two subcomponents: the *Journey Finder*-component - for finding journeys between two stops - and the *Journey Optimisation*-component - for optimising the journeys received from the *Journey Finder*-component. During the optimisation process the *Journey Optimisation* needs information regarding stops, paths and timetables from the *Timetabling*-component. This component has a core of internal timetable information located in the *Internal Timetabling*-component. An extension to the core is the *External Timetabling*-component through which the core timetable information can be updated and expanded from external timetable systems. The components are separated and grouped so they can be independently changed. It is possible to change the *Journey Optimisation*-component for an alternative method independent of the other components.

3.2 Data Structure

This section describes our data representation of the real world transportation networks on which journey searches can be carried out. This data structure is inspired by graph theory and is a directed graph with weighted edges.

3.2.1 Graph Representation

In the domain of operational research, journey planning can be solved in many different ways. The majority of the research on multimodal transportation networks are represented as graphs [3]. This representation is also used in the mentioned research conducted on data from Beijing and Singapore [9, 20].

As described in section 2.5, it is possible to find a path from one location in a graph to another using search algorithms. This functionality is exactly what we are looking for in this research as we have to find a path in a transportation network between several stops (vertices) via their connecting rails/roads (edges). The transportation networks are therefore represented as graphs, where stops are represented as vertices and the paths between the stops are edges. Edges are defined with a weight function. The weight represents the cost in time it requires to travel from one stop to an adjacent stop. The weight function for a given edge depends on the transportation mode. This enables us to distinguish between two transportation modes going between the same two stops, but with different speeds. For instance, a train might on average travel faster between the same two stops than a bus. In this way our path finding algorithm is independent of the type of transportation modes in the graph. The following illustrations of different transportation modes make the representation more clear. To simplify the illustrations the directions of edges are omitted and each edge represents both directions between two nodes.

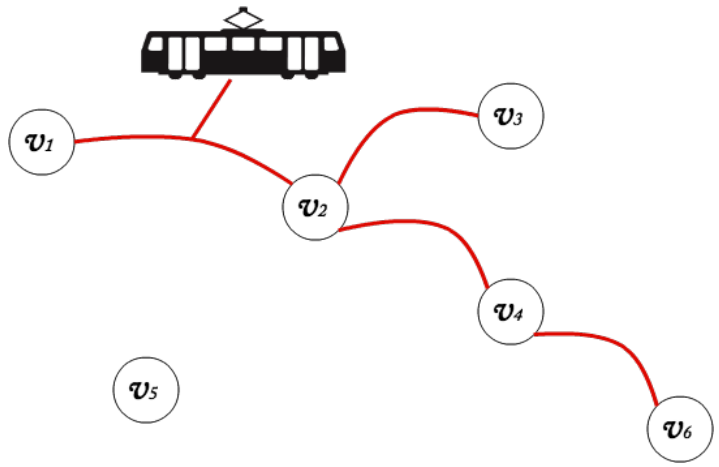


Figure 3.2: This graph, G_{tram}, shows an example of a tram network.

The representation of a transportation network for the tram mode as a graph, G_{tram}, is illustrated in Figure 3.2. On the graph, each stop (vertex) is labeled with a number and the red lines between the vertices represent the rails (edges) between the stops. Figure 3.2 therefore contains the stops $[v_1, v_2, v_3, v_4, v_5, v_6]$, where the stop v_5 is not reachable from the other stops and vice versa, because they have no connecting edges. The tram must follow fixed tracks or paths.

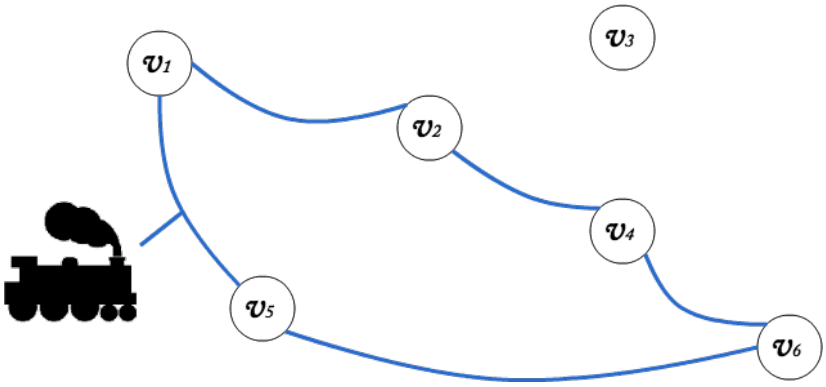


Figure 3.3: This graph, G_{train}, shows an example of a train network.

Similarly it is possible to represent the transportation network of the train mode. As illustrated in Figure 3.3 this network consists of the stops $[v_1, v_2, v_3, v_4, v_5, v_6]$. In this network the stop v_3 is isolated from the other stops.

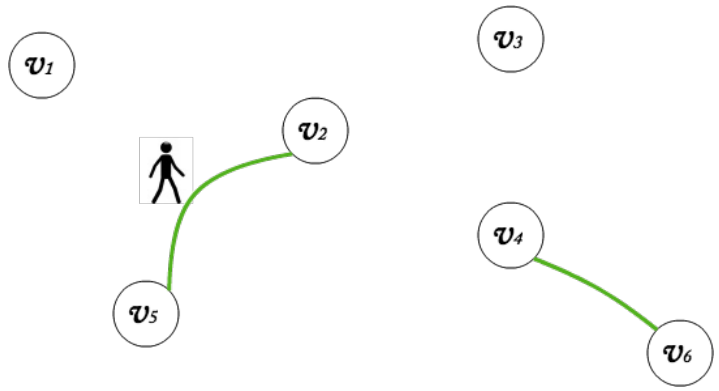


Figure 3.4: This graph, G_{walk}, shows an example of how walking between stops is represented.

It is assumed that walking can be represented as following fixed paths between selected locations, in this case stops (vertices). In Figure 3.4 the representation of a network with vertices $[v_1, v_2, v_3, v_4, v_5, v_6]$ is illustrated. It can be seen that it is possible to walk between the vertices v_2 and v_5 and vertices v_4 and v_6. Note that it is a directed graph so it is possible to walk from either vertex v_2 to vertex v_5 or walk from vertex v_5 to vertex v_2.

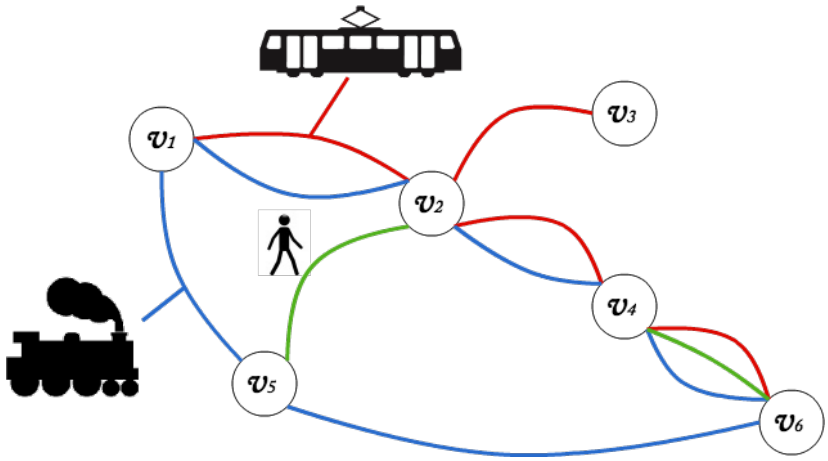


Figure 3.5: This is a graphical representation of the graph, $G_{transport}$, which is composed of the graphs for tram, train and walking.

We have now defined the three network types: tram, train and walking, as illustrated in Figure 3.2, 3.3 and 3.4. Each of these networks represent different transportation modes.

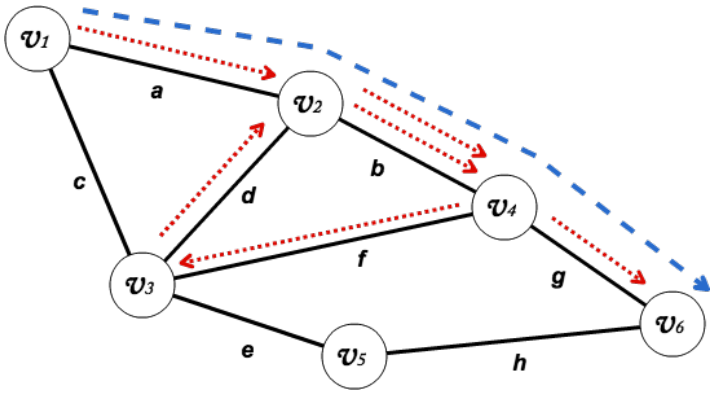


Figure 3.6: Journeys: ***RED:*** $[v_1, v_2, v_4, v_3, v_2, v_4, v_6]$ ***BLUE:***$[v_1, v_2, v_4, v_6]$

As the data structure is the same it is possible to merge all graphs to a single graph as is illustrated in Figure 3.5. Multiple transportation modes share the same vertices and it is therefore possible to travel between the same set of vertices with different modes. It is, for instance, possible to travel from vertex v_6 to vertex v_4 by using either train, tram or walking. Which type of transportation mode used is determined by the edge. Some vertices are only reachable by a single transportation mode, e.g. vertex v_3. As an example of going from vertex v_1 to vertex v_6 the following sequence of steps could be executed: Start in v_1 and go by Train to v_5, walk to v_2 and go by Tram to v_4 and v_6 where the sequence ends.

Studies have been done using matrices for storing information of weights and transportation routes [32]. We similarly hold this information in a database, where the reference to the information is coupled to the edge between adjacent vertices. Information derived from edges is: route number and direction, timetable, weight, start and target vertex, transportation type, and transportation network.

3.3 Loop Handling

When composing and manipulating the journeys we risk having loops in the journeys [9, 20]. That is, the journey covers locations that have been visited in the same journey, which make the solution invalid. The graph in Figure 3.6 illustrates an arbitrary network with vertices and labeled bi-directed edges. In the figure two journeys are marked: The *red*-dotted journey which has a loop and is composted of the vertices $[v_1, v_2, v_4, v_3, v_2, v_4, v_6]$ and the *blue*-striped journey which does not have loops and is composed of the vertices $[v_1, v_2, v_4, v_6]$.

To remove the loops in a journey a simple loop-removal algorithm is used:
The list of vertices is iterated from the beginning, and each vertex is remembered. If a vertex is seen twice the intermediate vertices are removed. The search for loops is continued until the whole journey has been scanned because a journey can contain multiple loops. For example the loop in the red journey in Figure 3.6 is eliminated by removing the vertices between index 2 and 4, such that the resulting journey is $[v_1, v_2, v_4, v_6]$. The new journey is exactly the same as the blue journey, which has no loops.

3.4 Journey Planning with Genetic Algorithm

We have earlier identified that most existing journey planners do not provide multiple journey options to passengers. To satisfy these needs the evolutionary computing technique Genetic Algorithm (GA) is utilised. Evolutionary computing techniques are inspired by processes in nature which are modelled to solve optimisation problems. Examples of processes are: the Ant Colony Optimisation algorithm which is inspired by how ants find paths from food back to their nest [10]; the Particle Swarm Optimisation Algorithm is inspired by how birds collaborate to find food [14]; GA takes advantage of how genes evolve by combining or mutating them to form better species [10].

We use GA because it has the advantages of being a probabilistic and population based algoritm [6, 24]. Since it is population based it can be used to produce alternative journey options depending on the passengers' preferences. Since the goal is to provide alternative journey options, these options are not guaranteed to be optimal as can be computed by using heuristics such as with A* [5, 15, 26]. However, the journeys should be sufficiently optimised to be used. The passengers' preferences are shortest journeys based on time and the fewest changes between transportation modes. GA manipulates journeys during its execution. It is important to verify that each journey is sound and valid when a journey is manipulated, for instance, after crossover or mutation [9]. We will in the following section describe how DFS is modified to be used in the initialisation and mutation operator of GA, then we explain the representation of chromosomes, the notion of *flagstop* and finally describe each specific operator as it is used in our model of GA.

Depth First Search Modifications

As discussed earlier, there are different approaches to search for paths in a graph, and some are better than others. DFS is an interesting algorithm because it does not get stuck in cycles (local minima). DFS records visited vertices and we can modify the algorithm to search in a randomised manner.

In this thesis we will be using DFS in its original form as described in section 2.5 and in a modified form described below. The algorithm is modified with the purpose of being used in GA.

To compute journeys with different preferences we modify DFS to add the expanded edges to its stack in random order. This entails the algorithm will explore the graph in a randomised manner. The algorithm is still complete, but the modification means the algorithm is not guaranteed to be optimal. This is exactly what we want to get diversity in the journeys. The algorithm will preserve its theoretical time complexity, but we hope that practical time complexity is reduced. It is beyond the scope of this thesis to prove that completeness and time complexity are maintained by this modification.

In this research we modify the DFS algorithm to terminate as soon as the goal is reached, and not when all vertices are discovered. The aim of this modification is to get journeys that might follow different paths from origin to destination each time the algorithm is run. Note, this argument is only valid if the above described exploration of the graph is also performed.

The above described approach to search for paths from an origin to a destination might suffer from being too naive during its search. Since no heuristic is used, there is a potential risk for the algorithm to search in a direction not leading to the goal and therefore waste time compared to using a heuristic function that guides the algorithm [26].

Flag Stops

Each route has two directions: for a given direction each stop has an opposite equivalent stop. Each of these pairs of opposite stops are identified with a flag stop number. The search algorithm is therefore searching this pair of opposite stops.

```
input: provide passenger preferences(e.g. origin, destination, time)
pop_cur <- initialise population of journeys
fittest_time <- select the most fit individual by time in population
fittest_routes <- select the individual with lowest changes and fit by time in the ↩
    population

while true do:
   while pop_cur not empty do:
      select parents for mutation and remove from pop_cur
         vertex_intersect <- find intersecting vertices of parents
         vertex_crossover <- select random vertex among vertex_intersect
         perform a crossover at vertex_crossover
         add parents to pop_new

      if random P_mutate > 0.8:
         chrome_mutate <- remove random individual from pop_cur
         vertex_mutatepoint <- select random vertex
         if P_rand < 0.5:
            chrome_new <- replace from start of chrome to vertex_mutatepoint
         else:
            chrome_new <- replace from vertex_mutatepoint to end of chrome
         eliminate loops in chrome_new
         add chrome_new to pop_new
   end do

   evaluate fitness of pop_new:
      fit_cur <- select individual with best fitness based on time in pop_new
      if fit_cur fitter than fittest_time:
         fittest_time <- fit_cur
      fit_cur <- select individual with best fitness based on routes and time in ↩
          pop_new
      if fit_cur fitter than fittest_routes:
         fittest_routes <- fit_cur
      if max generation or population converged:
         break
end do

return fittest_time and fittest_routes
```

Figure 3.7: Genetic Algorithm for journey planning.

Pseudo-code of the Model

Our journey planning model is defined as in Figure 3.7. The model is based on GA as defined in Figure 2.7.

Chromosome Representation

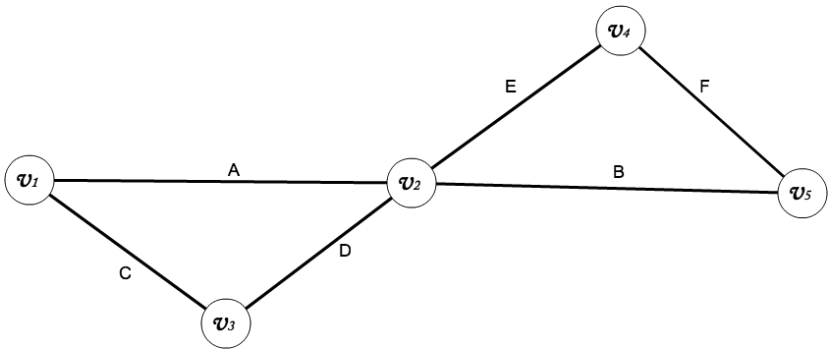
The representation of the chromosomes are journeys, which are represented by a list of vertices (representing stops) going from an origin to a destination [1, 20]. An example of the representation of a journey from vertex v_1 to vertex v_6 is in Figure 3.5, which illustrates the chromosome: [*Origin*, v_1, $Train$, v_5, $walk$, v_2, $Tram$, v_4, $Tram$, v_6, *Destination*]. This sequence represents the following execution: Start in the origin v_1 and go by Train to v_5, walk to v_2 and go by Tram to v_4 and v_6 where the sequence terminates at the destination.

Initialisation

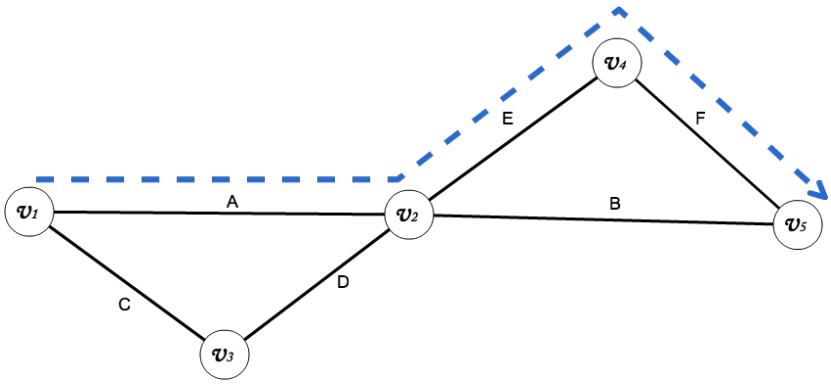
As described, the GA operates with a population of solutions which it optimises. In the initialisation process a number of individual journeys are generated so that each journey represents a chromosome. To generate the journeys we use the modified version of DFS as described earlier in this chapter. With this initialisation mechanism we hope the population will contain diverse individuals.

Mutation

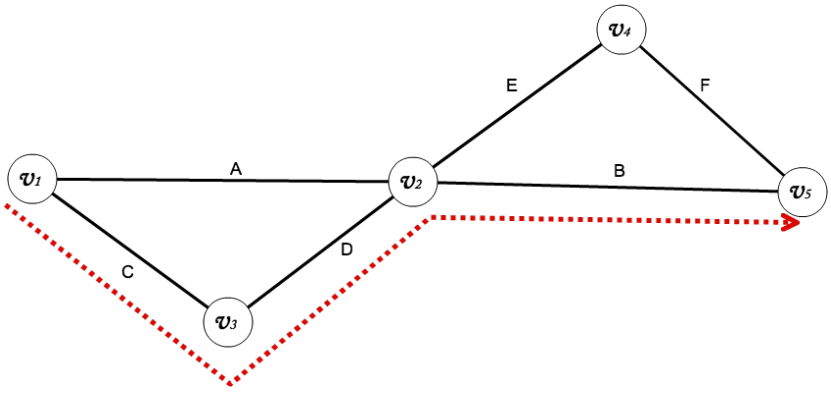
In the mentioned research with real-world datasets from Singapore [20] an interesting approach of replacing a sub-path between two randomised points was used. In their research they generated the sub-path by searching from the two points until they met at the same point. With inspiration from this research, the mutation operator used in this thesis is based on a randomised approach. First we select a random position in the journey. At this location the journey is recalculated by one of two approaches: first from the origin of the journey to the position, second from the position to the destination of the journey. The recalculated journey section is replaced in the current journey. The choice of approach is based on a uniform distribution with 50% probability. We use a probability to recompute the journey from the start or to the end to encourage diversity of the journeys. It is important after the mutation of journeys that we verify its validity again. To recalculate the selected part of the journey we use the same modified version of DFS as in the initialisation process.



(a) Simplified transportation network.

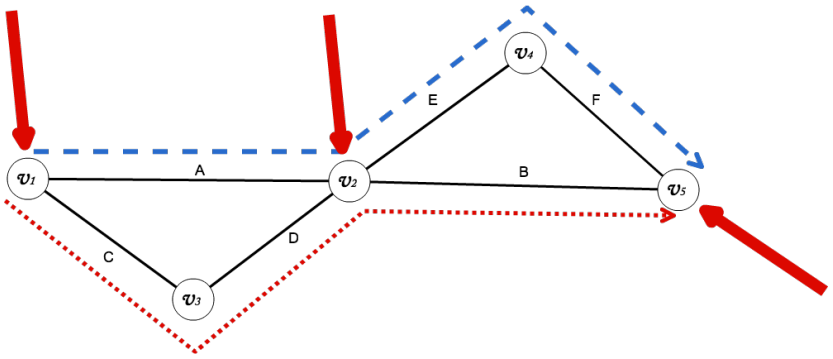


(b) Journey BLUE from origin: v_1 to destination: v_5. **BLUE** $[v_1, v_2, v_4, v_5]$.

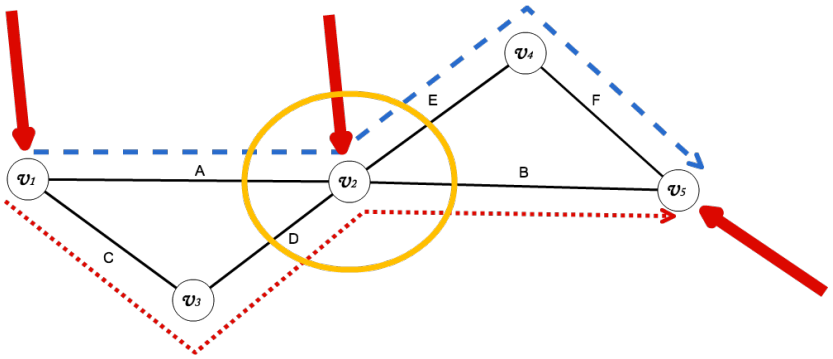


(c) Journey RED from origin: v_1 to destination: v_5. **RED** $[v_1, v_3, v_2, v_5]$.

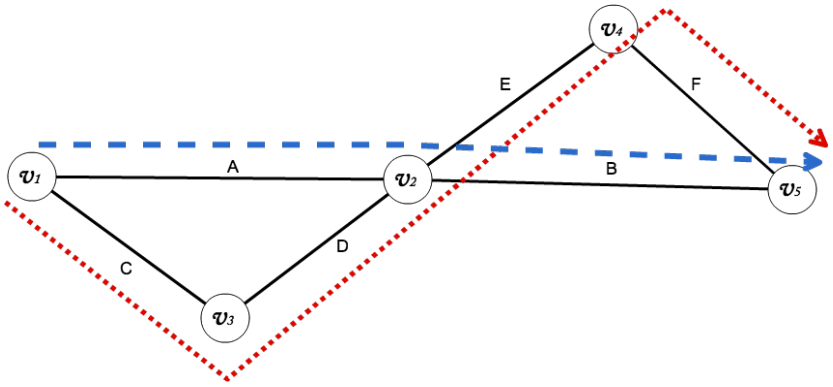
Figure 3.8: Graph and journeys illustrated for crossover example.



(a) Both journeys shown in the graph and intersecting vertices are marked with red arrows.



(b) Crossover point at node v_2



(c) New journeys after cross-over **RED** $[v_1, v_3, v_2, v_4, v_5]$ **BLUE** $[v_1, v_2, v_5]$.

Figure 3.9: Crossover example performed on journeys.

Crossover

Recent studies of journey planning using GA [5, 9, 20] have inspired us to use the next crossover technique. To perform a crossover between two parent chromosomes all intersecting stops between the parents are found, excluding the origin and destination stops. A random

stop among these stops is selected and a crossover is performed. This results in two new children.

An example is provided in Figure 3.8 to illustrate this. The graph in Figure 3.8a represents a simple transportation network with the stops $[v_1, v_2, v_3, v_4, v_5]$. A number of different routes can be composed using this graph, and we will examine two different routes with origins at stop v_1 and destinations at stop v_5. The first journey called "BLUE" has length 4, is composed of the stops $[v_1, v_2, v_4, v_5]$ and is illustrated in Figure 3.8b. The second journey called "RED" has length 4, is composed of the stops $[v_1, v_3, v_2, v_5]$ and is illustrated in Figure 3.8c. In Figure 3.9a both journeys are shown and intersecting vertices are marked with arrows at vertices v_1, v_2 and v_5. The journeys share only one intersecting point, namely v_2 as marked in Figure 3.9b. We perform a crossover at this point, where BLUE will be composed of the start of BLUE to the point and from the point to the end of RED, and vice versa. The resulting journeys have changed in composition and we can conclude that the journey BLUE is optimised to length of 3 vertices and the RED is less optimal with the new length of 5 vertices. The resulting journeys are BLUE $[v_1, v_2, v_5]$ and RED $[v_1, v_3, v_2, v_4, v_5]$ and are shown in Figure 3.9c.

Selection

The selection is based on the amount of time taken for a journey. The less time it takes, the fitter the journey is. The individuals with the different passengers' preferences are recorded during the generations. In this way we can present the best solutions for certain criteria for the passenger.

Termination

There are multiple criteria we can use to determine when to stop the GA runs. Termination can be based on a predefined number of generations, or when the fitness of the population is not improving based on convergence. The convergence is measured by the average fitness of the whole population and if the new population has the same average fitness as the preceding population it is said to be converged.

3.4.1 Evaluation Function

As detailed in section 2.7, we treat the journey planning problem as a minimisation problem. This defines the cost function for evaluation in GA: the so called fitness function. Time is our

main objective. The smaller the total travel time, the better the journey. The cost function is evaluated both in *line 3*, *line 27* and *line 30* in the algorithm shown in Figure 3.7. In the evaluation step we also recorded the journey for the given generation that has the lowest number of transportation modes and lowest travel time. This journey might have a larger travel time than the minimum journey, but require fewer transportation mode changes.

Summary

In this chapter we have seen how multiple transportation modes can be represented in the same graph. We have shown how loops can be eliminated from journeys. Each operator of the GA was defined, and we have described how the DFS algorithm is modified to traverse a graph from the origin in an randomised manner, and used for initialisation and mutation operations. In the next chapter we present experiments and an analysis of the results.

Chapter 4

Experimental Results and Analysis

This chapter aims to demonstrate how GA can be utilised to find journeys in a transportation network using data from Melbourne, Australia. During the experiments we examine how GA converges and gives alternative solutions. We will investigate if the density of intersecting routes has an impact on the computed journeys. To give an indication of the time complexity related to the study, an empirical investigation of the execution time for depth first search and GA is carried out. Finally we present results from a commercial journey planner [19] compared with results from our journey planner.

4.1 Experiment Design

Initially we assumed the timetables for each transportation route would be relatively easy to handle. During the process of capturing the timetable data and converting them to our data structure different challenges became apparent. Each day of the week has an individual timetable and public holidays and sports events introduce exceptions to these. During the day a particular route can have its starting point at different locations and skip stops along the route depending on the time of day. We therefore assume the timetables from Monday to Thursday are the same. The timetables for Friday, Saturday and Sunday are separate timetables. Each day has the time range 6am to 12am. Routes that deviate from the majority of routes in the dataset are left out. Special days like New Years Day and other public holidays are left out. We assume the travelers are located at the origin stop and can only follow paths defined by the edges in the data structure, i.e. trains and trams follow tracks but, walking also follows fixed routes between the stops.

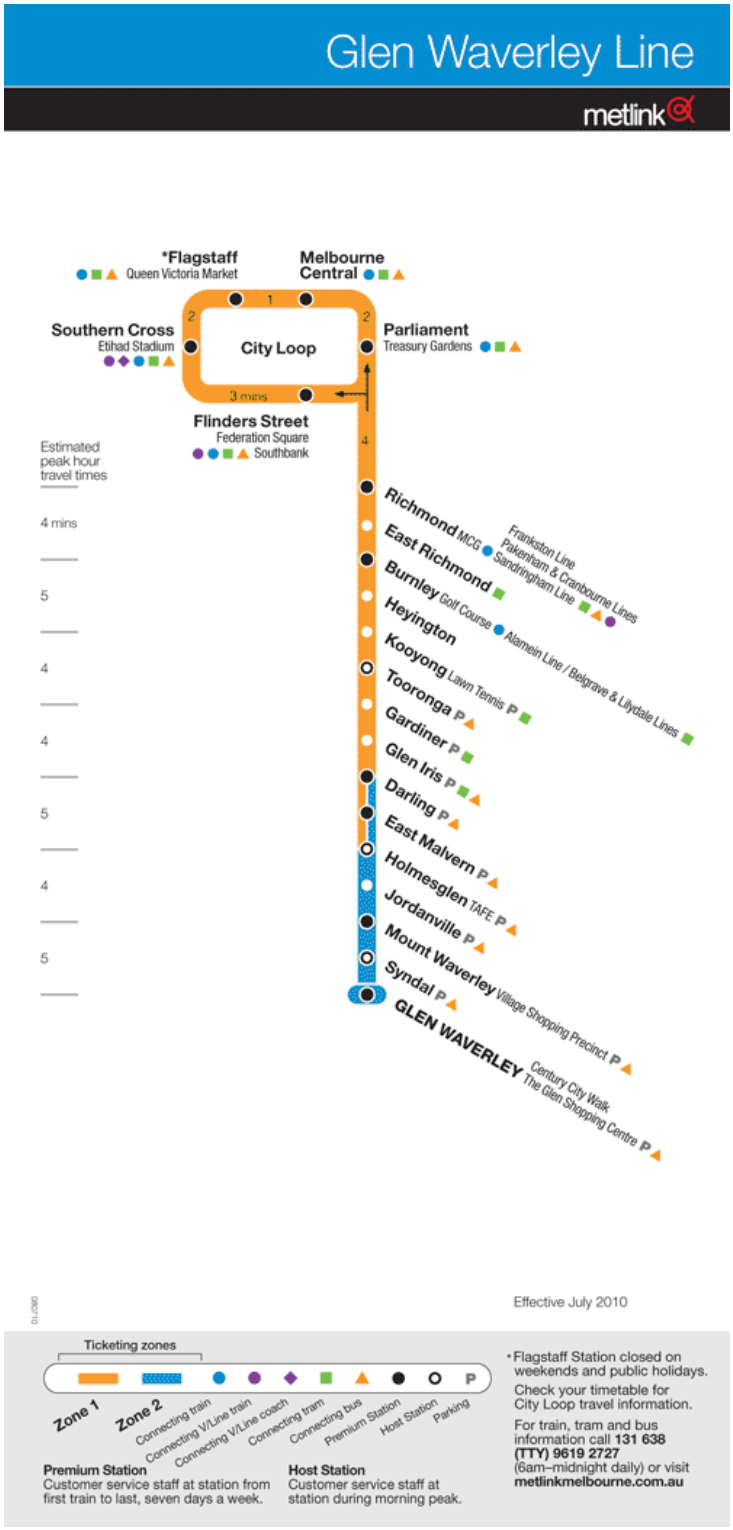


Figure 4.1: Train line: Glen Waverley Line [23].

We focus on the train and tram networks with data sourced from Yarra Trams' webservice [33] and Metlink's online timetables [17]. The tram network has about 1773 tram stops over 26 routes [34]. A map of the tram network is shown in Figure 2.3. The train network consists of a number of train lines. We have included one to show the multimodal feature of our journey planner [30]. We use the tram network in all experiments and add the train route when investigating multimodality and comparing our journey planner with a commercial journey planner. All experiments are conducted on the following dataset: tramlines[1]: 1, 3, 5, 6, 8, 16, 19, 24, 30, 31, 48, 57, 59, 64, 67, 70, 71, 72, 75, 78, 79, 82, 86, 96, 109 and 112. The train route is the *Glen Waverley Line*; see route description in Figure 4.1.

The settings for GA are (each term is described in section 3.4):

- Population with fixed size of 5 individuals.
- Mutation rate of 2%.
- Crossover rate of 80%.
- Selection by Roulette Wheel method[2].
- Termination either by 5 generation or convergence.
- Fitness by time in seconds.

To encourage a low number of transportation mode changes in each journey, a penalty for changing mode is set at 60 seconds added to ω. For each experiment the origin and destination stops are different. All experiments have the start time 2011-09-05 08:00:00 GMT+10. All experiments are carried out by repeating the executions a minimum of 20 times.

All tests are executed on a Apple Macbook Pro 2.7Ghz Intel Core i7 with 4 GB 1333MhZ DDR3 model 2011. Operation system: OS X Version 10.7.2.

[1]Data is downloaded from Yarra Trams dataservice on date 06.09.2011.
[2]Common randomised method for GA [10].

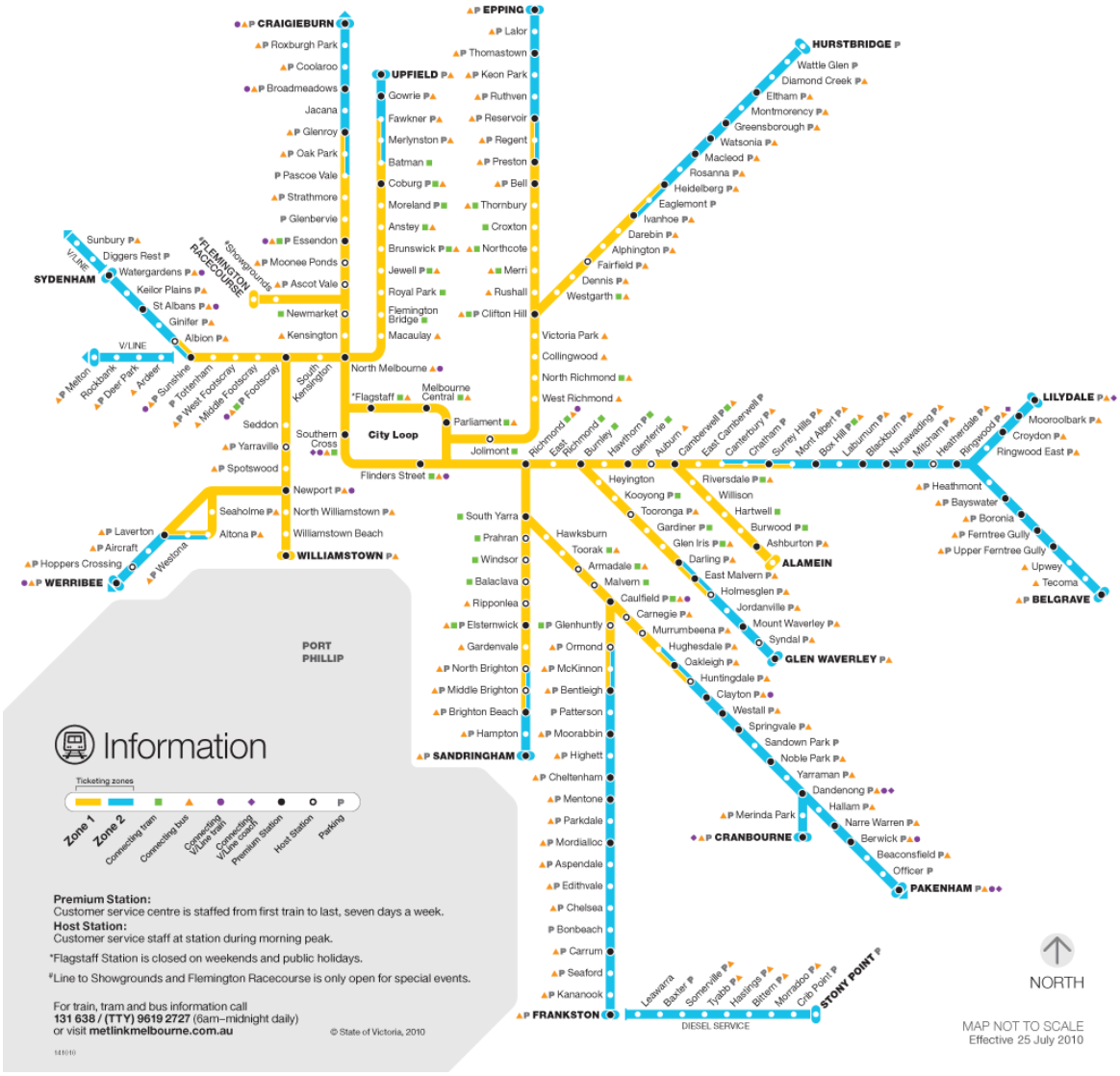


Figure 4.2: Map of the train network in Melbourne [23].

4.2 Real World Dataset

The data used in the experiments can be described as follows: each tram line starts at an outer location of the metropolitan area and terminates near the city center. The trains follows a similar pattern with a loop in the city center. They start from an outer location, go to the city center, around the loop and back out again. During morning rush hours the trains follow one direction of the loop, and during the evening peak the trains travel in the opposite direction

of the loop; see the map of the train network in Figure 4.2. The density of intersecting routes increases towards the center of the city.

We will investigate the behaviour of our model on the tram network. The results are shown in the tables: 4.1 to 4.6. The tables show the minimum and maximum fitness values in each generation, and an average fitness value of the whole population. The average is computed by the fitness sum of all individuals in a given generation divided by the number of individuals. The individuals of best fit and best number of routes are highlighted in bold. The values in parentheses represent the number of modes used in a journey. This investigation indicates the impact of multiple intersecting routes on the behaviour of the journey planner using real world data.

Figure 4.3: Inner city inset of the tram network map in Melbourne [27].

Assuming it is more desirable to travel with fewer mode changes, our model should favour the least numer of routes in its journeys. We have introduced a penalty for changing routes, and

Fitness \ Generation	0	1	2	3	4	5
min	18960 (1)	18960 (1)	18960 (1)	18960 (1)	18960 (1)	**18960 (1)**
max	18960 (1)	18960 (1)	18960 (1)	18960 (1)	18960 (1)	18960 (1)
average	18960	18960	18960	18960	18960	18960

Table 4.1: Selected mode from multiple options (from stop id 1387 to stop id 1353).

therefore expect the model to use a single route in the case where multiple routes are available. Table 4.1 shows the results of a run using two stops with three different transportation routes between them. A single transportation mode was used for each journey (indicated in parenthesis).

The results of the experiments consistently show the value of fitness is constant from generation 0 to generation 5. This means the journey with the best fitness is in the initial population. To determine the precise reason for the convergence of 5 generations requires further analysis of the dataset, our algorithm and the relation between nodes and their timetables. However, the following will briefly elaborate on possible reasons. Former research on real-world datasets [20] concluded that the number of iterations is related to the complexity of the map and a small population size of 2 produced useful results. As earlier mentioned, in this research a penalty for changing route during a journey is introduced. Factors that have an impact on faster convergence than expected are: the penalty for changing route; node complexity by lower number of intersection routes; few scheduled departures from each stop. The listed factors impact on the convergence is the following. By examining the map in Figure 4.3 we calculate the approximate complexity at each node to be $O(n^3)$, this entails the complexity at each node is significantly smaller than an exponential complexity of $O(n!)$. Furthermore, the penalty entails that fewer changes between routes in the journeys are encouraged and therefore decreases the search space. Finally, if the number of scheduled departures for the routes at each node is small, this will entail a further decrease of the search space.

The density of intersecting routes is higher in some areas of the transportation network than other areas. A higher density entails a larger number of alternative journeys between two stops. To get an indication of the density in the dataset we use stop sets from the inner city; see the 'Inner City Inset' map in Figure 2.3. The behaviour of the model in both high and low density areas is shown in Table 4.2, 4.4 and 4.3. We investigate sets of stops with either a short or longer distance between them.

Table 4.2 shows a trip between two stops in an area where the transportation network is dense. In general only two modes are used in each journey, except for generation 4. The

Value \ Generation	0	1	2	3	4	5
min	540 (2)	540 (2)	540 (2)	540 (2)	540 (4)	**540 (2)**
max	55620 (6)	540 (2)	540 (2)	540 (2)	540 (4)	540 (2)
average	6672	540	540	540	540	540

Table 4.2: A short trip within a dense area of the transport network (from stop id 1719 to stop id 3508).

initial generation 0 has a worst case journey using 6 different modes. It can be concluded that the algorithm has converged to a minimum in generation 0 and could therefore have been stopped before generation 5. The quick convergence of the whole population might be due to the low number of alternative solutions in this area of the transportation network.

Value \ Generation	0	1	2	3	4	5
min	2520 (4)	2520 (4)	2520 (4)	2520 (5)	2520 (4)	**2520 (4)**
max	46020 (7)	46020 (7)	30840 (7)	30840 (6)	30840 (7)	30840 (7)
average	24708	22548	19512	19512	19512	13848

Table 4.3: A long trip within a dense area of the transport network (from stop id 3251 to stop id 3509).

Table 4.3 shows a journey where the two stops within a transport dense area have a larger distance between each other than in Table 4.2. The number of modes needed for the journeys are larger than we saw for a shorter journey. As it is experienced in other research [20], this may be due to the increased number of combinations available as the distance between the stops increases. Examining the average fitness value shows the population is taking longer time to converge, and requires more generations to converge, than if the transportation network is dense with many intersections.

Value \ Generation	0	1	2	3	4	5
min	13620 (2)	13620 (2)	13620 (2)	13620 (2)	13620 (2)	**13620 (2)**
max	43860 (2)	43860 (2)	13620 (2)	13620 (2)	13620 (2)	13620 (2)
average	19668	19668	13620	13620	13620	13620

Table 4.4: Locations in a less dense network (from stop id 2926 to stop id 1767).

In Table 4.4 are two stops farther apart from each other than in Table 4.2, but in a less dense network. The journey planner generates options with the same numer of modes, but

different fitness. By comparing this with the average fitness values in Table 4.3 we see the average fitness value of the population in Table 4.4 appears to be converged in generation 2. This observation indicates that the number of generations and size of population might not be optimal in the scenario shown in Table 4.3.

Value \ Generation	0	1	2	3	4	5
min	2460 (6)	2460 (6)	2460 (8)	2460 (9)	**2460 (6)**	48240 (6)
max	46051 (8)	79140 (8)	79140 (9)	79140 (12)	79140 (9)	79140 (8)
average	38376	51444	51444	36108	51444	45264

Table 4.5: Short journey using many modes (from stop id 3006 to stop id 3621).

The journey planner finds solutions with a converged minimal finess, but the suggested journeys have a high number of modes. These journeys might not be useful in practice because it is inconvenient for the traveler to change modes too many times. An example is shown in Table 4.5, where the most fit journeys require a minimum of six different modes. In this dataset walking between stops is along designated paths. An investigation into whether unrestricted walking between stops would lower the number of modes is left for future work.

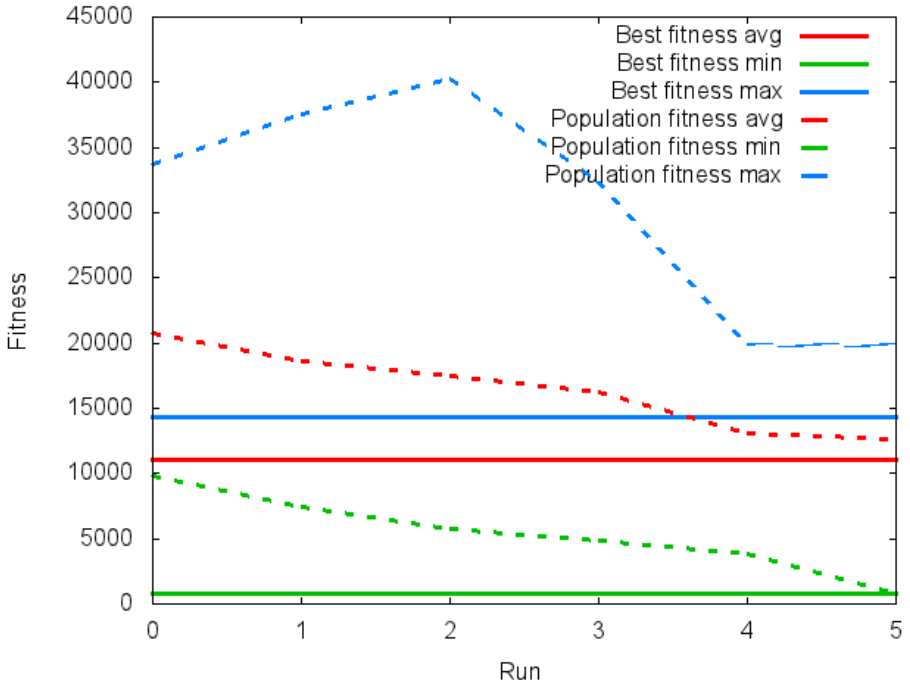


Figure 4.4: Convergence: fitness versus generation.

4.3 Convergence

Our model is based on randomised methods and it is therefore not expected to find the same solutions in each run. We expect the whole population over generations to converge to a minimum fitness value. To investigate this behaviour two stops with a long distance between each other in a dense area of the network are selected. The fitness is measured over 30 repeated runs and plotted in Figure 4.4. To investigate the fitness of the whole population the average population fitness is included. We can see the minimum fitness value does not change between each generation. As discussed earlier, this means the solution with best fitness is found in generation 0 and convergence therefore does not occur. If we analyse the average fitness of the whole population, it converges to the minimum fitness value which is expected.

4.4 Practical Application

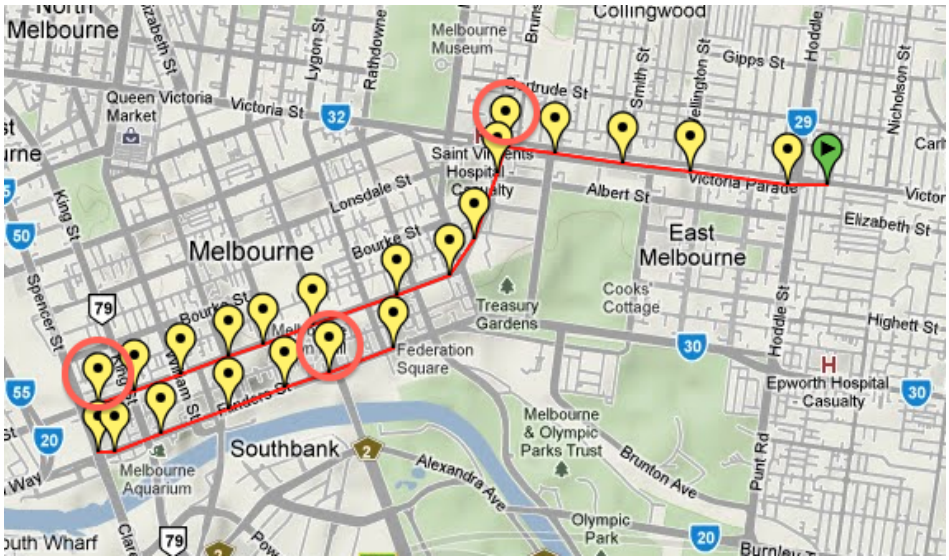


Figure 4.5: Journey from stop id 1719 to stop id 3606.

Value \ Generation	0	1	2	3	4	5
min	2340 (3)	**2340 (3)**	2340 (4)	2340 (3)	2340 (4)	**2340 (4)**
max	31740 (6)	16620 (6)	16620 (7)	16620 (6)	2340 (5)	2340 (4)
average	14076	8052	8052	5196	2340	2340

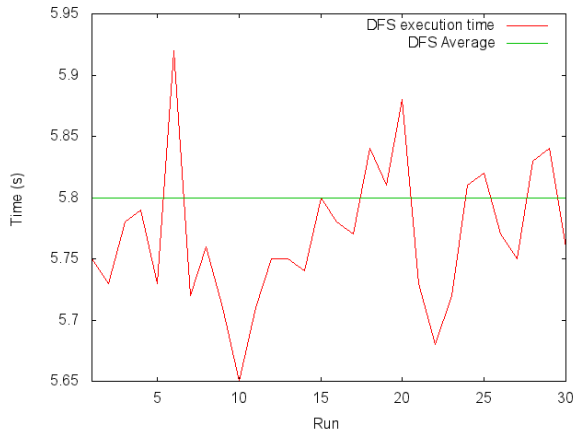
Table 4.6: Practical application of the journey planner (from stop id 1719 to stop id 3606).

To illustrate how our model would work in practice we show a journey on a map; see Figure 4.5. This example was selected as a realistic situation in which a traveler would need a journey planner. The two stops are within the inner city inset and are more than 400 m apart (five minutes walk) [21]. The journey has its origin at stop id 1719 and destination at stop id 3606. The results are shown in Table 4.6. The outline of the network in the inner city in Figure 4.3 shows the density of modes in this area. The results show that multiple journeys with the same fitness are available with different mode combinations. Since many journeys with the same travel time are available the model picks the one it discovers first. The option selected with best travel time is illustrated in Figure 4.5; it combines four different modes. Locations where the traveler changes mode are marked with a red circle. The alternative journey (not shown on the map) uses only three different modes.

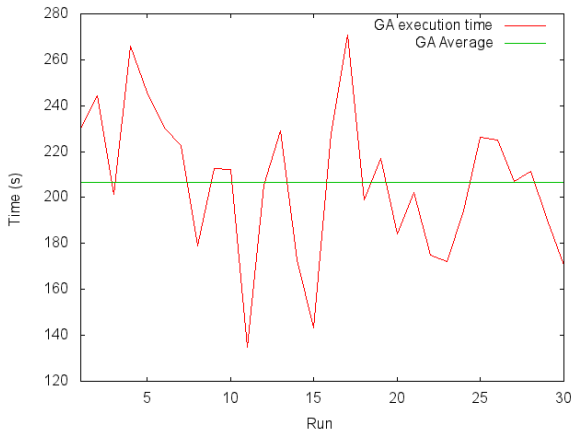
The resulting journey contains the following stops: 1719, 1718, 1716, 1715, 1713, 3512, 3511, 3510, 3509, 3508, 3506, 3505, 3504, 3503, 3502, 3501, 3255, 3601, 3602, 3603, 3604, 3605 and 3606. The journey time is 2340 seconds which seems relatively reasonable taking into account the numer of changes between modes. The description of the journey is:

1. Take tram number 24 from stop id 1719.
2. Change at stop id 3512 to tram number 31.
3. Get off at stop id 3501 and change to tram number 75.
4. Get off at stop id 3605 and change to tram number 70.
5. Get off at stop id 3606.

It is interesting that the journey planner chooses to change from tram number 75 to number 70 because both routes are following the same rails in that area of the transportation network. Going by tram number 75 appears much slower than changing to tram number 70 even though there is a penalty of 60 seconds for changing vehicle. As mentioned, between the same two locations an alternative journey with the same fitness combines the following tram routes: 24, 30 and 70. It should be up to the traveler to choose which journey option to travel by. A scenario where an alternative journey might be chosen is if the traveler had knowledge of delays or congestion on a particular transportation route. In this example, if the traveler knew tram line 30 was often delayed, it might be more beneficial to go by tram 31 and 75 instead.



(a) Exhaustive journey search using DFS.



(b) Journey search using GA.

Figure 4.6: Execution time versus run.

4.5 Execution Time of Search

The tram network is composed of long routes. Each time two routes intersect the complexity of composing journeys increases. The complexity of a search is therefore denoted by the number of intersections between the routes. The execution time is expected to be high if the number of intersections is high. To quantify this, we investigate the time it takes to perform an exhaustive search on the real world dataset. As discussed, the original method of DFS is an exhaustive search method and therefore used. To compare we also measure the execution time of our model. We expect the investigations to give an indication of the performance issues that are related to journey planning.

Figure 4.6a shows a repeated run from one location to another using DFS. The average execution time of 30 runs is 5.8 seconds.

GA is a population based approach and our model therefore searches for journeys multiple times. Since our operation for generating initial solutions is based on a modified version of DFS the execution time of the model is expected to be higher than DFS. The worst case execution time can be computed as: the execution time of DFS multiplied by the number of generations used multiplied by the number of individuals in the population. With our settings of GA we expect the execution time to be:

$$5.8 seconds \cdot 5 generations \cdot 5 individuals = 145 seconds \tag{4.1}$$

As indicated in the former research on real-world datasets [20], the actual execution time for computing journeys is an issue related to the use of GA. In Figure 4.6b the plot of 30 runs of GA is shown. The average time is 207 seconds. As this is higher than the expected time, it indicates the operations in GA demand more time of the optimisation process.

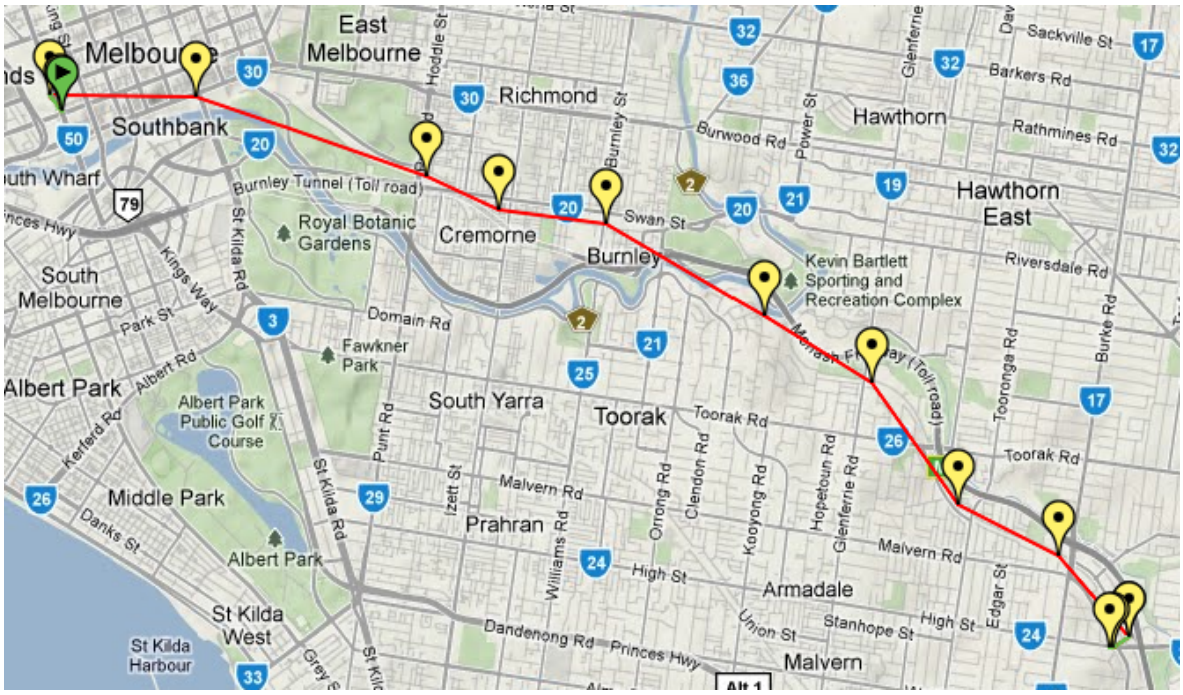


Figure 4.7: Multimodal journey computed by our model. Red line = train, green line = walk and tram.

4.6 Multimodality

As described in section 3.2, a benefit of using graph data structure is the ease of combining multiple transportation modes without affecting the model.

The following experiments will investigate multimodality between train, tram and walking. The tested journey is shown to be more efficient by train than by tram, but the solutions solely by tram are also explored. In this experiment the journey's origin is at a tram stop near a train line and the destination is a tram stop near a train line in another part of the network. For train travel walking from the tram stop to the train stop is required.

In Figure 4.7 the optimal journey based on time is shown. The different coloured lines in the journey (green lines at each end of journey), show that different transportation modes

Generation \ Journey	$Journey_1$	$Journey_2$
0	[75, walk, glenwaverleyline, walk, 6, 3, 1, 8]	[75, 96, 16, 3, 1, 6]
1	[75, walk, glenwaverleyline, walk, 6, 3, 1]	[75, 96, 16, 3, 1, 6]
2	[75, walk, glenwaverleyline, walk, 6, 3, 1]	[75, walk, glenwaverleyline, walk, 6, 3, 1]
3	[75, 96, 16, 3, 64, 5, 1, 8, 6]	[75, walk, glenwaverleyline, walk, 6, 3, 1]
4	[75, 96, 16, 3, 64, 5, 1, 6]	[75, 96, 16, 3, 64, 5, 1, 6]
5	[75, walk, glenwaverleyline, walk, 6]	[75, walk, glenwaverleyline, walk, 6]
Generation \ Journey	$Journey_3$	$Journey_4$
0	[75, 96, 16, 3, 1, 6]	[75, walk, glenwaverleyline, walk, 6]
1	[75, walk, glenwaverleyline, walk, 6, 3, 1]	[75, walk, glenwaverleyline, walk, 6, 3, 1]
2	[75, walk, glenwaverleyline, walk, 6, 3, 1]	[75, walk, glenwaverleyline, walk, 6, 3, 1]
3	[75, 96, 16, 3, 64, 5, 1, 8, 6]	[75, 96, 16, 3, 64, 5, 1, 8, 6]
4	[75, 96, 16, 3, 64, 5, 1, 6]	**[75, walk, glenwaverleyline, walk, 6]**
5	[75, walk, glenwaverleyline, walk, 6]	[75, walk, glenwaverleyline, walk, 6]
Generation \ Journey	$Journey_5$	
0	[75, walk, glenwaverleyline, walk, 6, 3, 1]	
1	[75, 96, 16,3, 1, 6]	
2	[75, walk, glenwaverleyline, walk, 6, 3, 1]	
3	[75, 96, 16, 3, 64, 5, 1, 8, 6]	
4	[75, 96, 16, 3, 64, 5, 1, 6]	
5	[75, walk, glenwaverleyline, walk, 6]	

Table 4.7: Multimodal Journey Search.

are used, i.e. the journey is multimodal. To illustrate the behaviour of the model during optimisation Table 4.7 shows which modes the journeys are composed of in each generation. The numbered modes are tram routes, and the 'glenwaverleyline' mode is by train. By examining the table, it can be seen that some journeys only use the tram mode, which indicates the model explores solutions that are not optimal. This feature provides more alternatives and caters for more passengers' preferences. The most optimal journey by time is $journey_4$ in generation 4. This journey is composed of:

- origin at tram stop id 3254
- take tram route 75 to stop id 3354
- walk to train stop id 8817 and board 'glenwaverleyline' train
- get off at train stop id 9908
- walk to tram stop id 1175 and take tram route 6
- tram is used to reach the destination at stop id 8004

The results in Table 4.7 are a good illustration of how the model in each generation tries different combinations of modes.

4.7 Comparison with a Live Journey Planner

From: 19-North Richmond Railway Station/Victoria St (Richmond)
To: 7-101 Collins St (Melbourne City)
Fares: Zone(s) 1 Metcard

Departing: 8:00 am, Wed 5 September 2012
Arriving: 8:13 am, Wed 5 September 2012
Total Time: 0 hr 13 min

Travel by		Time	Details	Map	Information
	DEP:	Wed, 8:00 am	From Stop 19-North Richmond Railway Station/Victoria St (Richmond)	STOP MAP	
			Take the Route **24 tram towards City(Docklands)** or take the Route **109 tram towards Port Melbourne**		Time 1 min Frequency: 4 min Zone(s): 1 Operator: Yarra Trams Leg Timetable
	ARR:	Wed, 8:01 am	Get off at stop 13-Lansdowne St/Victoria Pde (East Melbourne)		
	DEP:	Wed, 8:05 am	From Stop 13-Lansdowne St/Victoria Pde (East Melbourne)	STOP MAP	
			Take the Route **109 tram towards Port Melbourne**		Time 8 min Frequency: 4 min Zone(s): 1 Operator: Yarra Trams Leg Timetable
	ARR:	Wed, 8:13 am	Get off at stop 7-101 Collins St (Melbourne City)	STOP MAP	

MODIFY SEARCH AGAIN RETURN JOURNEY ONWARD JOURNEY

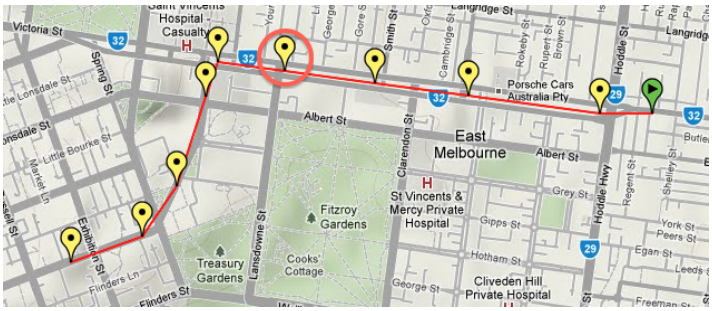
Legend: Metro Train, Metro Tram, Metro/Regional Bus, V/Line Train, V/Line Coach, Walk, Car, Cycle

Figure 4.8: Description of journey with live journey planner.

The organisation Metlink is responsible for promoting public transport in Melbourne. Their live journey planner [19] uses the same dataset as we use. Metlink uses an algorithm by Hacos [8], as discussed in section 2.3. By comparing our model with Metlink's live journey planner we are comparing with a journey planner that is used in many places around the world. The test journey originates at 'North Richmond Railway Station/Victoria St (Richmond)', tram stop id 1719, and has the destination '101 Collins St (Melbourne City)', tram stop id 3508. In line with Metlink's settings: only tram routes are selected; and the amount of walking is limited to the smallest value. Both journey planners are set to the same weekday and same time of day (Wednesday 8:00am). Metlink's journey planner is limited to only search for future journeys, but our data is for the first week of September 2011. This might result in different journey options, however we expect them to be similar.

Metlink's journey planner provides the traveler with different journey options, with the most interesting one shown in Figure 4.8. This journey is similar to the one our journey planner finds, which is a combination of tram routes 24 and 31. The journey suggested by our journey

(a) Journey by commercial journey planner, Metlink.



(b) Journey by our journey planner.

Figure 4.9: Maps comparing the journey by a commercial journey planner and our journey planner.

planner is via the stops: 1719, 1718, 1716, 1715, 1713, 3512, 3511, 3510, 3509, and 3508. The fitness in time is 1140 seconds, which is greater than the time of Metlink's suggested journey - 780 seconds (13 minutes). This difference can be explained by the possible differences in datasets, and the penalty of 60 seconds for mode change in our model. The journeys provided by the two journey planners are marked in Figure 4.9. It can be seen that both journeys are identical in which stops they contain.

Summary

Our journey planner is able to find journeys using a real world dataset, and provide different journey options based on minimum time and minimum number of changes between transportation modes. The density of the network has an impact on the number and variety of modes for a journey. Our results show the model always finds a journey with minimum fitness based on time in the initial population. The journey planner favours a small number of different routes in journeys. Two important results of the experiments are that our journey planner is multimodal and can find journeys that are similar to real world journey planners.

Chapter 5

Conclusions

This thesis is an empirical investigation of: how a journey planner can be implemented using evolutionary techniques and its behaviour using a real world dataset. Our introduction covered: the background of journey planning; the use of search algorithms on a graph data structure; a description of evolutionary computing with regard to the Genetic Algorithm (GA); and how GA can be modeled to solve journey planning problems and optimise journeys for passenger's preferences. The graph data structure enabled us to represent different transportation modes in the same data structure. The model was therefore independent for different transportation modes. Our model was able to find solutions that were multimodal. The model chose the most feasible journeys based on time and transportation modes by combining different modes such as train, tram and walking. We described the model in detail in chapter 3, giving pseudocode in Figure 3.7.

The research questions have been addressed as in the following:

RQ1 *How can a journey planner that caters for different passengers' preferences be modeled?*
The hypothesis that the features of GA can be utilised to compute journeys with different passengers' preferences was confirmed. The main features of GA which wave been adapted for our model are: the discrete string representation of genomes modeled by lists of stops in the journeys; the fact that it is population based; and how mutations are performed. Our experiments showed that GA enabled us to generate solutions with different features and allow passengers to pick one that satisfied their preference.

RQ2 *How can a journey planner provide multiple journey options that are all useful in practice?*

To test our model we used a real world dataset taken from Yarra Trams' webservice [33] and Metlink's online timetables [17]. We demonstrated the usefulness of the journeys in practice by illustrating them on maps and with descriptions. We showed that an exhaustive search can solve the problem in reasonable computation time, and by using GA, alternative journey options are produced.

RQ3 *How is a journey planner affected by the number of intersecting transportation routes?*

The density of the network has an impact on the number and variety of modes for a journey. The greater the density of the transportation network, the more modes are likely to be used in a journey as discussed in section 4.2.

RQ4 *How does a journey planner using an evolutionary approach compare with a commercial journey planner?*

We compared the results of our journey planner with a commercial journey planner, provided by Metlink, with similar results. This indicates that journeys generated by our model appear comparable by having the same stops in their journeys. The computed travel time by our journey planner was consistent with the travel time suggested by the commercial journey planner.

5.1 Future Work

While a number of research questions have been addressed in this thesis, further research can be conducted on this topic.

Questions that could be answered include: Will it improve the quality of the solutions to permit walking from any location and not along fixed routes? How would it improve the model by adding alternative transportation modes as cars or bike-riding? Would alternative approaches for generating initial solutions improve the performance of the GA? How does the model behave if the dataset contains a larger number of intersecting routes? Also, different functions than DFS for generating initial solutions might be beneficial to explore.

The resulting journeys of our research did not distinguish it self significantly from the other mentioned research based on real-world dataset [9, 20] and an empirical investigation on the same dataset for comparison it is therefore necessary to conclude if resulting journeys of our model is different from theirs.

It might be interesting to investigate the opportunities for incorporating real time updates in the data structure. For this to be achieved, an additional weight function on the edges might be useful. This weight function could correspond to current delays in the transportation system.

This thesis was not an investigation of multiobjective optimisation, even though our model can be expanded to handle multiple objectives [24]. It might be interesting to investigate how multiobjective optimisation with regard to passengers' preferences would affect diversity in journey options.

There are many features wich can be implemented and modelled using our journey planner. This is only the beginning. We plan to incorporate CO_2 emissions calculations and many more modes to provide passengers with more personalised options to select from when planning their journeys.

Bibliography

[1] Rahim A. Abbaspour and Farhad Samadzadegan. An evolutionary solution for multimodal shortest path problem in metropolises. *Computer Science and Information Systems*, 7(4):789–811, 2010.

[2] Georgia Aifadopoulou, Athanasios Ziliaskopoulos, and Evangelia Chrisohoou. Multiobjective optimum path algorithm for passenger pretrip planning in multimodal transportation networks. *Transportation Research Record: Journal of the Transportation Research Board*, 2032 / 2007:26–34, 2007.

[3] Maurizio Bielli, Azedine Boulmakoul, and Hicham Mouncif. Object modeling and path computation for multimodal travel systems. *European Journal of Operational Research*, 175(3):1705 – 1730, 2006.

[4] Thomas Cormen, Charles Leiserson, Ronald Rivest, and Clifford Stein. *Introduction to Algorithms*. McGraw-Hill and MIT Press, Second Edition, 2001.

[5] Liviu Adrian Cotfas and Andreea Diosteanu. Public transport route finding using a hybrid genetic algorithm. *Informatica Economic*, 15(1):62–68, 2011.

[6] Mitsuo Gen and Runwei Cheng. *Genetic Algorithms and Engineering Optimization (Engineering Design and Automation)*. Wiley-Interscience, December 2000.

[7] Google. Google transit. `http://transit.google.com`, 2 October 2011.

[8] Hacon. Hafas journey planner. `http://www.hacon.de/company/press/downloads/hafas_broschuere_download_e.pdf`, 2 October 2011.

[9] Feng Lu Haicong Yu. A multi-modal route planning approach with an improved genetic algorithm. In *Joint International Conference on Theory, Data Handling and Modelling in GeoSpatial Information Sciencen*, volume 38, pages 343–348, May 2010.

[10] Randy I. Haupt and Sue Ellen Haupt. *Practical Genetic Algorithms*. John Wiley and Sons New York, 1998.

[11] John H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. The MIT Press, April 1992.

[12] John Hopcroft and Robert Tarjan. Efficient planarity testing. *J. ACM*, 21:549–568, October 1974.

[13] Mark E. T. Horn. Procedures for planning multi-leg journeys with fixed-route and demand-responsive passenger transport services. *Transportation Research: Part C*, 12(1):33, 2004.

[14] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on*, volume 4, pages 1942–1948, August 2002.

[15] S. Meena Kumari and Dr N. Geethanjali. A survey on shortest path routing algorithms for public transport travel. *Global Journal of Computer Science and Technology*, 9(5), 2010.

[16] Penousal Machado, Jorge Tavares, Francisco B. Pereira, and Ernesto Costa. Vehicle Routing Problem: Doing It The Evolutionary Way. In W. B. Langdon, Cantú E. Paz, K. Mathias, R. Roy, D. Davis, R. Poli, K. Balakrishnan, V. Honavar, G. Rudolph, J. Wegener, L. Bull, M. A. Potter, A. C. Schultz, J. F. Miller, E. Burke, and N. Jonoska, editors, *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, New York, July 2002. Morgan Kaufmann Publishers.

[17] Metlink. Train data. `http://www.metlinkmelbourne.com.au/timetables`, May 2011.

[18] Metlink-Melbourne. Metlink. `http://metlinkmelbourne.com.au/`, May 2011.

[19] Metlink-Melbourne. Metlink journey planner. `http://jp.metlinkmelbourne.com.au/`, May 2011.

[20] S.C. Nanayakkara, D. Srinivasan, Lai Wei Lup, X. German, E. Taylor, and S.H. Ong. Genetic algorithm based route planner for large urban street networks. In *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pages 4469 –4474, September 2007.

[21] Department of Planning New South Wales Government and Infrastructure. Planning guidelines for walking and cycling, December 2004.

[22] Department of Infrastructure. Meeting our transport challenges, connecting Victorian communities, May 2006.

[23] State of Victoria. Metropolitan train network map, 2010.

[24] M.S. Osman, M.A. Abo-Sinna, and A.A. Mousa. An effective genetic algorithm approach to multiobjective routing problems (morps). *Applied Mathematics and Computation*, 163(2):769 – 781, 2005.

[25] Petrica C. Pop, Camelia-Mihaela Pintea, and Corina Pop Sitar. An ant-based heuristic for the railway traveling salesman problem. In *Proceedings of the 2007 EvoWorkshops 2007 on EvoCoMnet, EvoFIN, EvoIASP,EvoINTERACTION, EvoMUSART, EvoSTOC and EvoTransLog: Applications of Evolutionary Computing*, pages 702–711, Berlin, Heidelberg, 2007. Springer-Verlag.

[26] Stuart J. Russel and Peter Norvig. *Artificial Intelligence - a modern approach*. Prentice Hall series in Artificial Intelligence. Prentice Hall, Upper Saddle River, New Jersey, second edition, 2003.

[27] Johnno Shadbolt. Map of melbourne tram networks, 2008.

[28] Steven Skiena and Miguel Revilla. Graph algorithms. In *Programming Challenges*, Texts in Computer Science, pages 217–244. Springer New York, 2003.

[29] Berkeley University of California. Path2go. `http://www.networkedtraveler.org/`, May 2011.

[30] Vicsig. Train stops. `http://www.vicsig.net/suburban/fleet`, May 2011.

[31] Vladimir Vishnevsky and Roman Zhelezov. Algorithm to find the shortest path on a high-dimensional graph and its application for timetable information systems. *Transport and Telecommunication*, 10(4), 2009.

[32] Hongmei Wang, Ming Hu, Wei Xiao, and Hongmei Wang. A new public transportation data model and shortest-path algorithms. In *Proceedings of the 2nd international Asia conference on Informatics in control, automation and robotics - Volume 1*, CAR'10, pages 456–463, Piscataway, NJ, USA, 2010. IEEE Press.

[33] Yarra-Trams. Tram data. `http://ws.tramtracker.com.au/pidsservice/pids.asmx`, May 2011.

[34] Yarra-Trams. Tram stops. `http://www.yarratrams.com.au/desktopdefault.aspx/tabid-47//74_read-117/`, May 2011.

[35] Liping Zhang, Jingquan Li, Kun Zhou, Somak Datta Gupta, Meng Li, Wei-Bin Zhang, Mark A. Miller, and James A. Misener. Design and implementation of a traveler information tool with integrated real-time transit information and multi-modal trip planning. In *TRB Annual Meeting 2011*, 2011. California PATH, University of California, Berkeley, `http://www.networkedtraveler.org/nttrb.pdf`.

[36] Konstantinos G. Zografos, Konstantinos N. Androutsopoulos, and John D. Nelson. Identifying travelers' information needs and services for an integrated international real time journey planning system. In *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, pages 998 –1004, September 2010.