Australasian Transport Research Forum 2013 Proceedings
2 - 4 October 2013, Brisbane, Australia
Publication website: http://www.patrec.org/atrf.aspx

A computational analysis of shortest path algorithms for integrated transit and automobile trip planning

Abstract

While there are many journey planners in use around the world, the vast majority of them focus solely on optimising trips for the transit network, which is based on routes and timetables, or on optimising trips on the road network for drivers of personal vehicles. By integrating these two networks together into a single journey, the Multimodal Multiobjective Trip Planner (MMTP) in development at the Smart Transport Research Centre (STRC), Queensland University of Technology, can provide more flexibility to users on how they travel to their destination.

The primary goal of this paper is to present an analysis of the computational performance of the shortest path algorithms that have been implemented for the MMTP. The algorithms that are used will be outlined, as well as the graph structures that they search. The method for integrating the two networks into a combined trip plan will be discussed. Results will then be presented from tests performed on the Brisbane transit and road networks. This will be followed by an analysis of these results, by comparing the performance of the algorithms for the different categories of journeys that could be experienced by the trip planner in practical use.

1. Introduction

The MMTP that is in development at the STRC has three main goals. They are:

- True multimodal trip planning, combining public transport and private vehicle legs into a single journey via park'n'ride facilities.
- Providing multiple objective choices, such as travel time, distance, cost and other quantifiable objectives.
- The integration of real-time information, such as predicted travel times, incident information, and real-time transit updates, so that a more accurate trip plan can be calculated.

The main focus of this paper is to analyse the performance of the MMTP for multimodal trip planning. This paper is continuing the work from Casey et al. (2012), which presented the data requirements and graph data structures in detail for the MMTP.

This paper has the following format: First, the search algorithms that are used and the graph data structures that they search are presented. The method of integrating the two networks together into a combined trip plan is also discussed. Next, the testing methodology that is used for producing results is outlined. The computational test results are then presented, which is followed by an analysis of these results. Finally, conclusions for the paper are made and future work for the development of the MMTP is discussed.

2. Algorithms and data structures

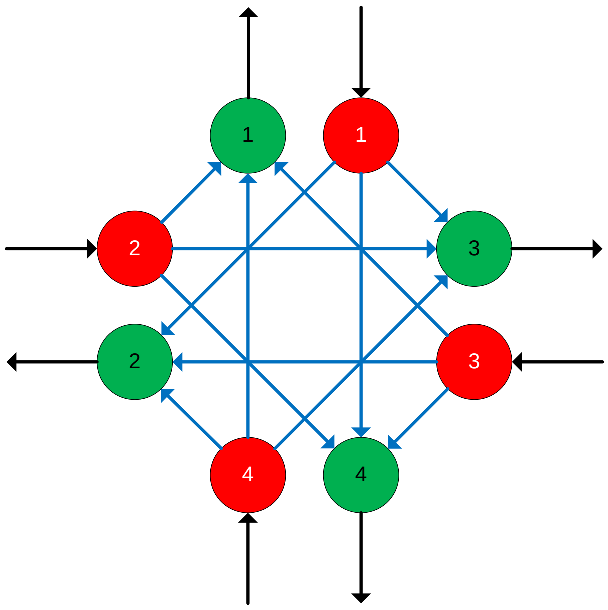
The MMTP makes use of two main data sources: the Road Network and the Public Transport Network. The two graph networks are similar in the fact that they consist of a set of nodes and arcs that connect neighbouring nodes together, however they are different both from a conceptual point of view and for practical implementation purposes. This difference is because the nodes and arcs within the road network represent a physical entity that are geographically separated, while the nodes and arcs within the public transport network represent temporal events and the possible connections allowed between these events.

2.1 Road Network

The graph data structure that is used in the MMTP for the road network is depicted in Figure 1. Each node represents either an entry into (red nodes) or exit from (green nodes) a specific intersection. The directed arcs that connect two nodes together can be of two types:

- An arc that starts from an exit node and ends at an entry node represents a road segment between two intersections (black arcs).
- An arc that starts from an entry node and ends at an exit node represents a valid turning action that can be performed at the intersection, such as turn left, turn right, u-turn or straight-through (blue arcs).

Figure 1: Road network graph data structure representation



2.2 Public Transport Network

The graph data structure that is used in the MMTP for the public transport network is depicted in Figure 2, while Figure 3 shows a more detailed representation of the data structure within a specific stop. We have used the Time-Expanded Approach (TEA) (Schulz, 2005) to represent the public transport network. In the TEA, each node represents a specific event in time at a specific stop. There are three types of events that can occur:

- Arrival event, which represents a transit vehicle arriving at the stop.
- Departure event, which represents a transit vehicle departing from the stop.
- Transfer event, which allows for transfer connections between services.

There are four types of arcs that connect pairs of these event nodes together:

- Service link, which connects arrival and departure events. These arcs can represent either the transit vehicle moving from one stop to the next stop, or the vehicle sitting stationary at the stop between its arrival and departure events.
- Transfer link, which connects arrival events to transfer events. These arcs represent the ability for a passenger to disembark from the transit vehicle to either transfer to another service or to complete their public transport journey. Also, the transfer link can connect an arrival event in one stop to a transfer event in another stop, which

represents the passenger disembarking from the transit vehicle and walking to a nearby stop to board another service.

- Boarding link, which connects transfer events to departure events. These arcs represent the ability for a passenger to board a transit vehicle.
- Waiting link, which connects consecutive transfer events within a stop. These arcs represent the waiting time that a passenger spends until their chosen service arrives for boarding.

Transfer and Boarding arcs can be omitted between certain nodes to represent situations where only boarding or disembarking is allowed at a stop for a particular service. However, Service and Waiting arcs are mandatory between consecutive events for a service and within a stop, respectively.

Figure 2: Public transport network graph data structure representation (Schulz, 2005, p. 20)

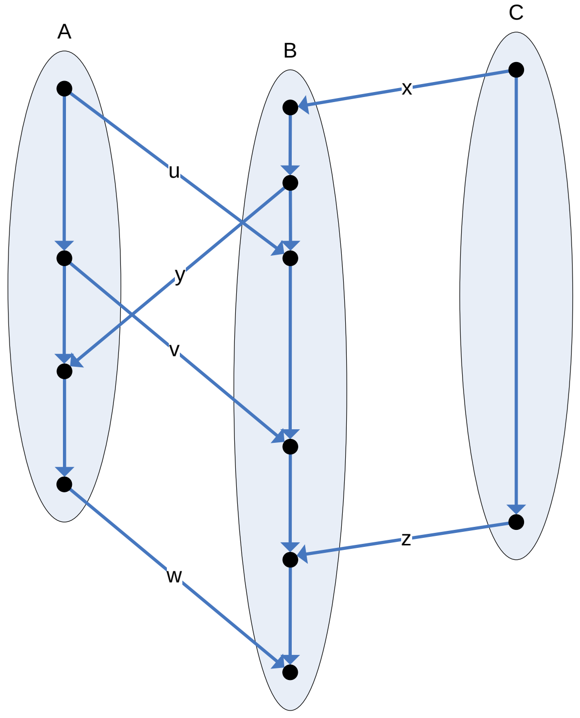
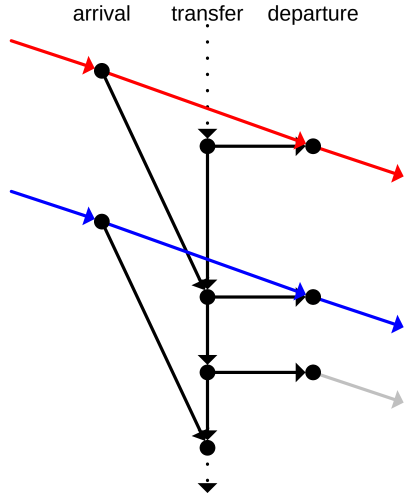


Figure 3: Detailed representation of public transport graph data structure within a stop (Schulz, 2005, p. 39)



2.2 Shortest-Path Algorithms

Two shortest-path algorithms have been implemented for the MMTP, Dijkstra's algorithm and A* heuristic algorithm.

Dijkstra's algorithm is the classical label-setting shortest path algorithm proposed by Edsger Wybe Dijkstra (Dijkstra, 1959). It is actually a one-to-all shortest path algorithm as it finds the minimum cost from the origin node to all other nodes in the graph. The pseudo-code for Dijkstra's algorithm is as follows:

1. Assign an initial value to every node label.
    a. Zero for the origin node;
    b. Infinity for all other nodes.
2. All nodes are marked as ***unvisited***, and construct the ***unvisited set*** from these nodes.
3. Set the origin node as the ***current node***.
4. For the ***current node***, check all of its ***unvisited neighbours*** and calculate the cost of going from the ***current node*** to each neighbour.
    a. If the cost of going from the ***current node*** to an ***unvisited neighbour*** plus the value of the label for the ***current node*** is less than the label for that ***unvisited neighbour***, replace the neighbour's label with this sum.
5. When all ***unvisited neighbours*** have been checked, mark the ***current node*** as ***visited*** and remove it from the ***unvisited set***. A visited node is never checked again, and its cost is minimised.
6. If the ***unvisited set*** is empty, stop the algorithm. All node labels contain minimum costs of travelling from the origin to the node. Otherwise, find the node in the ***unvisited set*** with the smallest label value and set it as the next ***current node***, then return to step 4.

The original Dijkstra's algorithm did an unordered search when finding the next candidate node, and has been shown to have a complexity of $O(n^2)$, where n is the number of nodes. Various improvements have been made on the performance of the algorithm by storing candidate nodes in a heap or priority queue structure, and retrieving the node at the front of the queue as the next ***current node***, which speeds up the node selection and has a complexity of $O(m \log(n))$ (Johnson, 1972), where m is the number of arcs. This is the approach that has been used in the MMTP. Other improvements include the S-bucket algorithm (Dial, 1969) and an algorithm that uses a Fibonacci heap. This last algorithm has a complexity of $O(m+l \log(n))$ (Ahuja et al., 1990).

The A* heuristic algorithm (Hart et al., 1968) improves upon Dijkstra's algorithm for one-to-one shortest path problems, as it takes into consideration where the destination node is located. Rather than sort candidate nodes in the order of their cost from the origin node, A* uses an estimate of the cost to travel from the candidate node to the destination node, plus the calculated cost from the origin node to the candidate node, and orders the nodes based on this sum. It has been proven that if the estimate always underestimates the actual cost of travelling from the candidate node to the destination, the heuristic is admissible and will always calculate the optimal shortest path.

The main challenge to using the A* heuristic algorithm is the calculation of the estimate from the node to the destination. When the cost being considered is the shortest distance, the straight-line distance between the node and the destination is used and is relatively simple to calculate. However, for other objective costs, this estimation function is not as simple. For instance, the minimisation of travel time requires the knowledge of the maximum speed used between the node and the destination, and the estimate of the travel time between the node and destination is the straight-line distance divided by the maximum speed. The quality of the estimate has an effect on the computational performance of the A* algorithm. Ideally, the estimate should be as close as possible to the actual shortest path cost. The further the estimate is from the actual cost, the greater in number of nodes will be visited by the search algorithm, and therefore increases the time need to complete the search.

Another group of shortest path algorithms are the Bidirectional methods, such as Bidirectional Dijkstra (Dantzig, 1960, Nicholson, 1966) and Bidirectional A* (Pohl, 1969). These methods adapt the unidirectional algorithms to search from both the origin and destination at the same time, which reduces the overall search space compared to the original algorithms.

Finally, more recent research over the last two decades, since the late 1990's has seen the development of more sophisticated and efficient algorithms that require different amounts of pre-processing. Pre-processing has become more attractive due to the exponential increase in processing speed and memory capacity, which has made these approaches more viable. Some of these approaches include ALT (A* Landmarks with Triangle inequality) (Goldberg and Harrelson, 2005), Contraction Hierarchy approaches (Geisberger et al., 2008) and Hierarchical Encoded Path Views (HEPV) (Jing et al., 1998). Fu et al. (2006) provides a comprehensive overview of many of these methods that are available.

The main reason that the Dijkstra and A* algorithms have been implemented for the MMTP, and the more complex and efficient algorithms have not, is due to the simplicity of their implementation, as the focus of this research has been on the development of an approach to integrate the road and transit networks together to calculate multimodal shortest paths. Section 2.2 will present the approach that has been developed in detail, but in essence, this approach allows for the current shortest path algorithms to be replaced by more efficient algorithms once they have been implemented in the future.

2.2 Network integration for multimodal search

In the process of designing the MMTP, two methods were proposed for integrating the road and public transport networks together when performing a multimodal search. These approaches are the Multi-Level Network approach and the Multi-Step Algorithm approach.

The Multi-Level Network approach constructs a single graph data structure, where each different mode of transport is on a separate level, and connections are made between nodes in different levels where transitions between the modes occur, such as at Park'n'ride locations between automobiles and public transport. A dynamic version of this approach is used by Bousquet et al. (2009) to compute a one-way multimodal shortest path, and they also propose a strategy for solving the two-way approach.

On the other hand, Khani et al. (2012) propose an intermodal path algorithm, which performs a sequential application of transit and automobile shortest paths, which results in the optimal intermodal path and indicates which park-and-ride location is optimal for transferring between private and public modes. This is similar to the Multi-Step Algorithm approach that was implemented for this research, although the algorithm steps differ in the order they are performed. The Multi-step algorithm approach keeps the two graph data structures separate, and generates an overall journey by performing the following algorithm:

1. Find the park'n'ride locations within a specified distance of the origin (based on the cost of driving from the origin to destination by automobile only)
    a. Calculate the shortest path to drive by automobile from the origin to each of the park'n'ride locations.
2. For each of the park'n'ride locations, find the stops within a specified walking or cycling distance of these locations.

    a. Calculate the shortest path to walk from the park'n'ride location to each of its neighbouring stops.
3. Find the stops within the specified walking or cycling distance of the destination.
    a. Calculate the shortest path to walk from each of the neighbouring stops to the destination
4. Using the stops surrounding the park'n'ride locations as origin stops, and the stops surrounding the destination as destination stops, calculate the shortest paths via public transport from the set of origin stops to the set of destination stops.

The fourth step, where the public transport legs are calculated, makes use of a modified version of the Dijkstra's algorithm that takes as input a set of origin stops along with their cost from the actual origin, and a set of destination stops along with the cost to the actual destination. Conceptually, it is similar to Dijkstra's algorithm, however each of the origin nodes are set as visited nodes with their label values set to the costs input into the algorithm. If a destination stop is reached during the search, the node is saved in a list, and the algorithm terminates when either a specific number of nodes have been saved to the list, or there are no more nodes in the ***unvisited set***. This approach improves on a brute-force approach, where shortest path searches are performed separately for each pair of origin and destination stops, as it ensures that each node in the network is visited only once overall in the Multi-Step Algorithm approach

There are also additional restrictions that are checked when checking neighbours of the current node in the public transport search. These include a maximum period of time that has passed between the start of the journey and the time of the current event node, or the total number of transfers has exceeded a specified maximum. This reduces the overall search space covered by the algorithm, therefore improving computation time.

3. Testing methodology

A testing methodology plan has been developed in order to produce results that can be analysed and from which conclusions can be made on the computational performance of the MMTP algorithms. The methodology plan is detailed in this section.

First, a list of suburbs in the South East Queensland (SEQ) region is chosen. Figure 4 shows the geographical distribution of these suburbs. They are in the following four categories:

- **Major Destinations (Green)**
  - Central Business District (CBD)
  - Airport
  - Fortitude Valley
- **Suburbs with major shopping centres (Blue)**
  - Chermside
  - Indooroopilly
  - Carindale
  - Upper Mount Gravatt
  - Strathpine
- **Suburbs at the extremities of the SEQ region (Yellow)**
  - Ipswich
  - Southport
  - Coolangatta

  - Maroochydore
  - Caboolture
- **Metropolitan suburbs (Red)**
  - New Farm
  - St Lucia
  - Redcliffe
  - Wynnum
  - Samford Village
  - Albany Creek
  - Forest Lake
  - Graceville

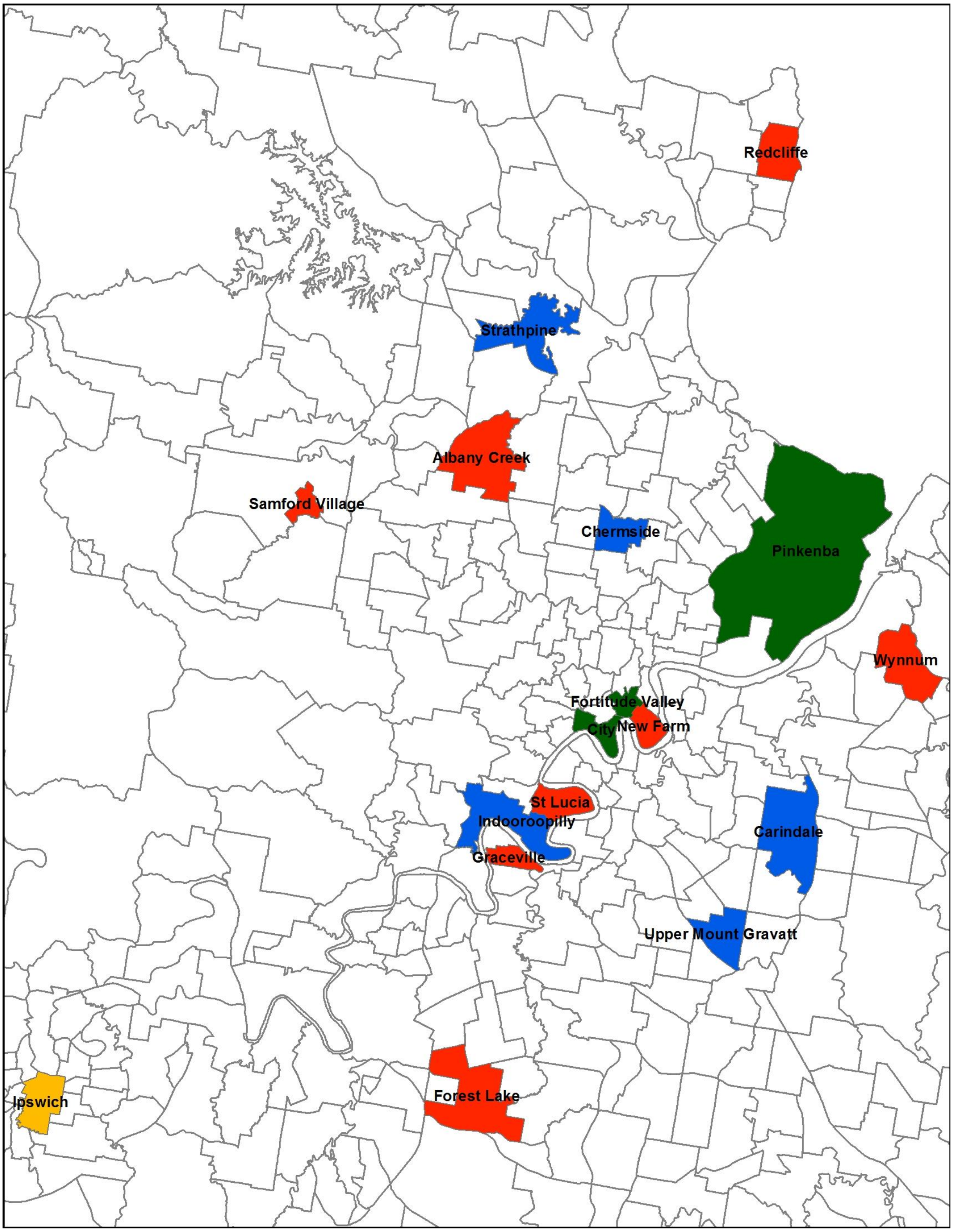
The major destinations and shopping centre suburbs have been chosen because the performance of searches to and from these locations are important, since it is likely that many requests will involve one of these suburbs. The suburbs at the extremities of the SEQ region have been chosen to test the performance of long-distance journeys, which are the worst-case scenarios for the MMTP. The other suburbs have been chosen for a number of reasons. New Farm, St Lucia and Graceville are relatively close to the CBD; however a direct straight-line trip is not possible due to the curves of the Brisbane River, so they should produce more interesting journey plans. The other metropolitan suburbs have been chosen to produce results in a number of different cardinal directions.

Tests are run using each of these suburbs as an origin, excluding the CBD, and the CBD as the destination. The tests with each suburb pair is run using automobile only, public transport only, and multimodal mode choices. For the public transport only and multimodal choices, tests are performed for 4 time periods: morning peak (6am – 9am), afternoon peak (4pm – 7pm), inter-peak period (9am – 4pm) and evening/early morning (7pm – 6am). 50 tests are run for each combination of (suburb, CBD) origin/destination, transport mode and time period, with a random origin and destination coordinate chosen from within the respective suburbs.

The following parameter values are set for the tests:

- Objective type: Travel Time
- Query type: Depart After
- Maximum Number of Transfers: 3
- Maximum Walking Distance: 1500 metres
- Maximum Transfer Distance: 400 metres
- Maximum number of itineraries to be returned: 1

Figure 4: Geographical Distribution of Test Suburbs



4. Test Results

This section presents a collection of the test results that have been obtained by following the testing methodology detailed in Section 3. These tests have been performed on a desktop computer with an Intel Core I5 650 processor operating at 3.2 GHz, 16 GB of RAM and Windows 7 64-bit as the operating system. For readability and space purposes, a selection of the suburbs will be presented in the graphs in Figure 5, Figure 6, and Figure 7.

Figure 5: Road Only Search Computation Time

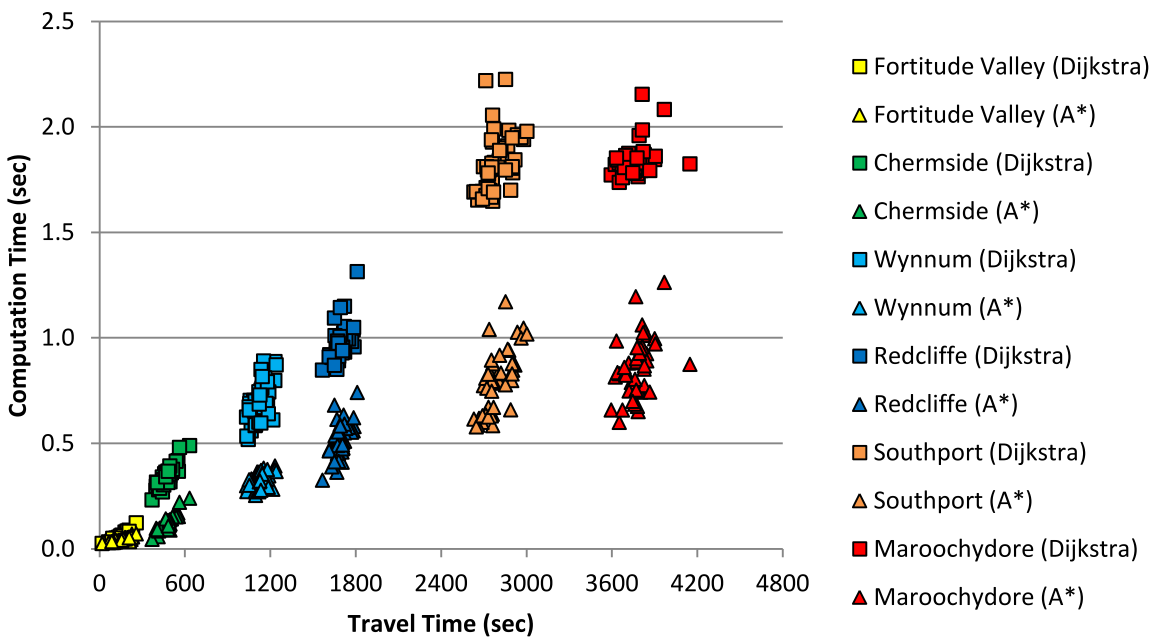


Figure 6: Public Transport Only Search Computation Time

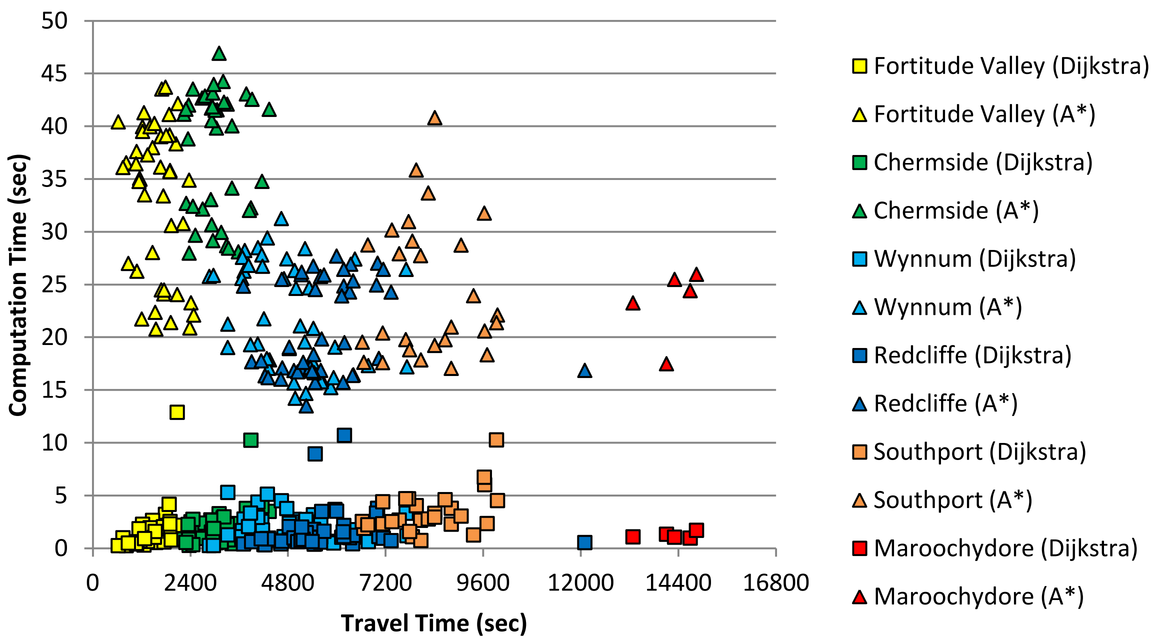
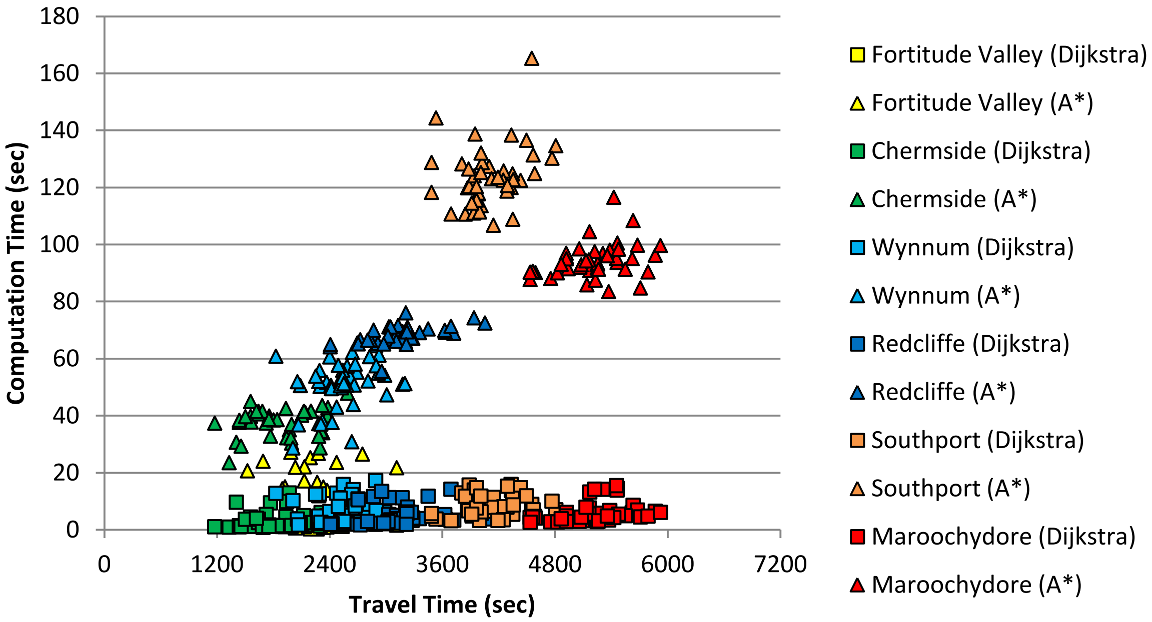


Figure 7: Multimodal Search Computation Time



5. Analysis of Results

Figure 5 presents the computational times for the road-only searches. A relationship can be seen between the travel time and the computational time required to complete the search. This is because the computational time is directly related to the number of nodes that are searched, and the longer a journey is, the more nodes will be visited. A* clearly performs better than Dijkstra's algorithm in all cases, and the difference in this performance increases as the travel time is increased. It is interesting to see the similarities in computation time of the Southport and Maroochydore searches, while the travel times are greater for Maroochydore compared to Southport. This is most likely due to the density of nodes encountered in the searches, as the road network is more dense travelling north from Southport to the CBD, compared to travelling south from Maroochydore to the CBD.

Figure 6 presents the computational times for the public transport only searches. A relationship between travel time and computation time cannot be clearly seen in this data, and this is because the nodes in the public transport graph represent events that occur at transit stops and are not directly related to distance or travel-time. As opposed to the road-only searches, using Dijkstra for the shortest path searches between the origin/destination and their surrounding stops performs markedly better than using A*. This is because a single Dijkstra 1-to-many search was utilised, as opposed to having to use separate 1-to-1 A* searches for each stop, and thus the nodes in the network are only visited once in the overall search.

Figure 7 presents the computational times for the multimodal searches. These results provide for similar conclusions that were made for the public transport only searches. The use of single Dijkstra 1-to-many searches for the road and walking legs performs considerably better than the use of many 1-to-1 A* for each individual stop/park'n'ride combination. The main difference in Figure 7 when compared to Figure 6 is that there is some relationship between computation time and travel time. This is due to the driving legs of the multimodal journey, and is most visible in the A* searches. While it is not that visible in Figure 7, the computation time of the Dijkstra searches does increase slightly with respect to the travel time.

The practical use of these algorithms within a trip planner application must be considered as well. Most users will be expecting a result within a few seconds of their request, therefore the average and worst case of computation time must be analysed. For road-only searches, the MMTP performs adequately, producing results within a second for almost all searches, including the worst case scenarios of travelling from Maroochydore to the CBD.

For public transport only searches, using Dijkstra as the shortest-path algorithm produces adequate results, with the average search completing within 5 seconds, with a few outliers that take around 10 seconds. These outliers are actually due to memory-allocation issues caused by using C#, which takes care of recovering memory itself via a garbage collector. Therefore, implementing the MMTP in C++, where the memory allocation is coded explicitly, may remove this issue and these outliers. A* is clearly not viable in practice, with the minimum computation time required at 15 seconds, and varies considerably depending on the specifics of the request.

For multimodal searches, the average time required for a search when using Dijkstra is below 10 seconds, with the worst cases taking under 20 seconds to complete. This is probably on the borderline of what would be acceptable by a user of the application, however this computation time can be improved upon by modifying various input parameters to the search, such as maximum number of transfers, maximum transfer distance between nearby stops, and the maximum time allowed for the journey to take. Again, the performance of A* would not be viable in practice.

6. Conclusion

This paper has presented the design of the Multimodal Multi-objective Trip Planner, detailing the data structures and algorithms used in its implementation. It then outlined a methodology for testing the computational performance of the MMTP, and presented results obtained from tests based on the SEQ region. An analysis of these results has been discussed, comparing the performance of Dijkstra and A* for road only, public transport only and multimodal searches, as well as discussing the practical expectations of performance from the user's perspective and whether these algorithms satisfy these expectations.

The future work that will be performed on the MMTP is to extend it to allow for the integration of real-time information, such as predicted travel times. This will require the design and implementation of more sophisticated time-dependent shortest path algorithms, as well as modifying the data structures to incorporate this time-dependent data.

References

AHUJA, R. K., MEHLHORN, K., ORLIN, J. & TARJAN, R. E. 1990. Faster algorithms for the shortest path problem. *Journal of the Association for Computing Machinery,* 37**,** 213-223.

BOUSQUET, A., CONSTANS, S. & EL FAOUZI, N.-E. On the adaptation of a label-setting shortest path algorithm for one-way and two-way routing in multimodal urban transport networks. International Network Optimization Conference, 2009.

CASEY, B., BHASKAR, A. & CHUNG, E. Data requirements and graph data structures for a multi-modal, multi-objective trip planner. 25th ARRB Conference Proceedings, 2012.

DANTZIG, G. B. 1960. On the shortest route through a network. *Management Science,* 6**,** 187-190.

DIAL, R. B. 1969. Algorithm 360: shortest-path forest with topological ordering [H]. *Communications of the ACM,* 12**,** 632-633.

DIJKSTRA, E. W. 1959. A note on two problems in connexion with graphs. *Numerische Mathematik,* 1**,** 269-271.

FU, L., SUN, D. & RILETT, L. R. 2006. Heuristic shortest path algorithms for transportation applications: State of the art. *Computers and Operations Research,* 33**,** 3324-3343.

GEISBERGER, R., SANDERS, P., SCHULTES, D. & DELLING, D. 2008. Contraction hierarchies: Faster and simpler hierarchical routing in road networks. *Experimental Algorithms* Springer.

GOLDBERG, A. V. & HARRELSON, C. Computing the shortest path: A search meets graph theory. Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms, 2005. Society for Industrial and Applied Mathematics, 156-165.

HART, P. E., NILSSON, N. J. & RAPHAEL, B. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics,* 4**,** 100-107.

JING, N., HUANG, Y. W. & RUNDENSTEINER, E. A. 1998. Hierarchical encoded path views for path query processing: an optimal model and its performance evaluation. *IEEE Transactions on Knowledge and Data Engineering,* 10**,** 409-432.

JOHNSON, E. L. On shortest paths and sorting. Proceedings of the ACM annual conference - Volume 1 1972 Boston, Massachusetts, United States. 569965: ACM, 510-517.

KHANI, A., LEE, S., HICKMAN, M., NOH, H. & NASSIR, N. Intermodal Path Algorithm for Time-Dependent Auto Network and Scheduled Transit Service. Transportation Research Record: Journal of the Transportation Research Board, 2012 Washington, D.C.: Transportation Research Board of the National Academies, 40-46.

NICHOLSON, T. A. J. 1966. Finding the Shortest Route between Two Points in a Network. *The Computer Journal,* 9**,** 275-280.

POHL, I. 1969. *Bi-directional and heuristic search in path problems.* Department of Computer Science, Stanford University.

SCHULZ, F. 2005. *Timetable information and shortest paths* PhD, Universit¨at Karlsruhe.