# Trajectory Based Routing using Frequented Paths

A THESIS
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Pratik Kotwal

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
Master of Science

Advisor: Prof. Shashi Shekhar
June, 2019

# Acknowledgements

I would like to extend my sincere thanks, gratitude and appreciation to all those who have guided and assisted me in the completion of this degree.

Firstly, I would like to thank my advisor, Prof. Shashi Shekhar, for all the guidance and support that he has provided throughout my graduate studies. His timely advice and direction through all stages of the research pushed me to do more.

In addition I really appreciate Prof. Ravi Janardan and Prof. Alireza Khani for being on my committee and providing me with their valuable feedback.

I am also extremely thankful to Yan Li for all his help and support while working on this thesis. I am grateful to all the students of the Spatial Computing Research Group for their insightful comments and suggestions.

Finally I would like to thank my parents and sister for their constant support throughout my education.

# Abstract

A McKinsey Digital report estimates that personal location data could help save consumers about $600 billion by 2020, by providing alternate routes to vehicles that avoid traffic congestion by using next-generation routing algorithms. The variety of spatial big data means that a single "One-size-fits-all" algorithm will not be able to answer all our queries. We would rather need a number of specialized algorithms that can work together to answer these questions.

The Trajectory Based Routing problem aims to find the frequented path with the lowest cost between a given origin and destination. Cost estimation models that make use of trajectory data to estimate costs in a path-centric manner were recently introduced. However in the path selection phase these algorithms extended paths one edge at a time, in a "path + edge" fashion. In contrast in the path selection process, this work extends the paths in a "path + path" manner, speeding up the path selection process.

To do this we make use of a new representation, "Maximal Frequented Path Graph (MFPG)", that combines the spatial graph and trajectory data into one representation. We propose a path selection algorithm , "MFPG Shortest Path Algorithm" that makes use of this representation to find the lowest cost path. We also introduce an admissible heuristic that can be used in the MFPG if an admissible heuristic exists in the spatial graph. The "Informed MFPG Shortest Path Algorithm" makes use of this admissible heuristic to further speed up the algorithm, by guiding the search space towards the destination. We prove both experimentally and theoretically that the proposed method is complete and correct, and computationally faster than the related work.

# Table of Contents

# List of Tables

# List of Figures

# Chapter 1

# Introduction

The amount of spatial data that is generated has exploded over the last decade due to the large availability of location-aware devices, on-board telematics devices installed in many commercial vehicles and the increase in the number of location based services such as Google Maps, Uber, etc. A McKinsey Digital report estimates that personal location data could help save consumers about $600 billion by 2020 [6], by providing routes to vehicles that avoid traffic congestion by using next-generation routing algorithms. This has already been done by companies such as UPS that save around 300 million gallons of fuel every year, reducing $CO_2$ emissions by around 31000 metric tons, by preferring routes that avoid left turns [25]. Given that transportation accounts for 28% of the total energy consumption in the United States [16], providing routes that can minimize energy consumption can lead to drastic savings in fuel. Newer routing algorithms will provide users with custom routes based on their preferences.

## 1.1 Need for a newer routing algorithm

### 1.1.1 Increase in number of location aware devices

Global Positioning System (GPS) became available for civilian use in the 1980's, and ever since there has been a steady increase in the number of location-aware devices around the world. Automotive Navigation Systems were designed making use of GPS devices to provide users with real time navigation.

However the popularity of smartphones with in-built GPS capabilities made location aware devices ubiquitous. The number of smartphones sold each year has been growing ever since 2007, as shown in Figure 1.1, and with this the number of devices capable of providing us with rich trajectory data has been increasing. Mobile applications such as

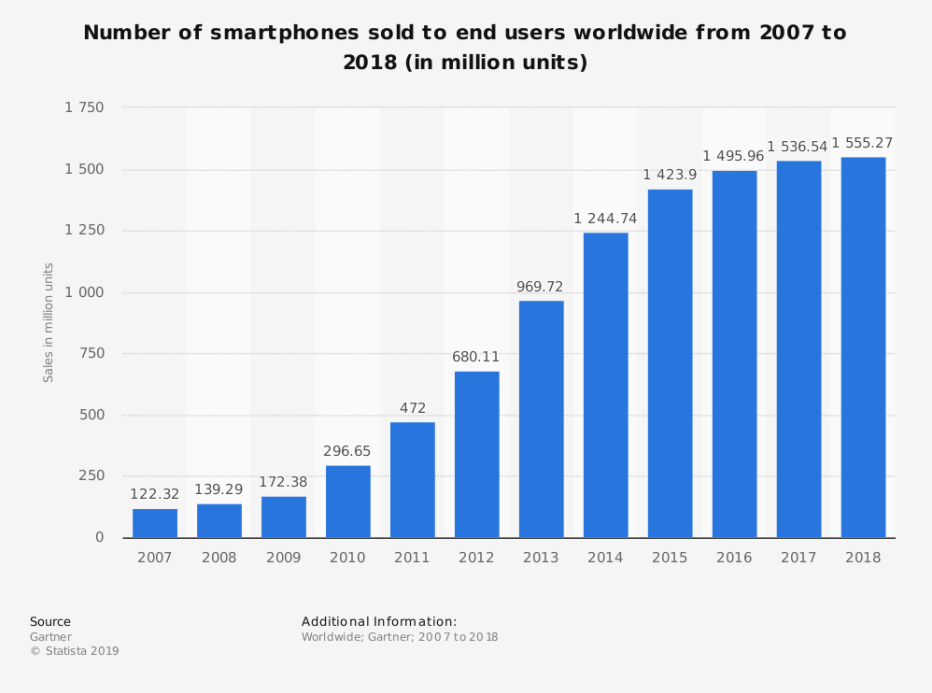



Figure 1.1: Number of smartphones sold worldwide from 2007 to 2018 [8]

Google Maps, Waze, etc. are often used by people on their smartphones to help them navigate. In the process these applications are given access to the trajectory of these users that is stored. This data has a lot of potential to provide these applications with the preferred routes of users as well as the amount of time taken by users on the routes. This data could be used to provide better routes to users saving consumers hundreds of billions of dollars [6].

The increase in on board telematics devices in vehicles provides us with a range of different measures such as fuel consumption, greenhouse gas emissions, etc. that could be used to compute different cost measures that cannot be done by just using the data available in the spatial graph.

### 1.1.2 More information provided by trajectory data

Most traditional routing algorithms only make use of the data available in that spatial graph. However trajectory data can provide us with a lot more information about the preferred routes of travelers, greenhouse gas emissions, and energy consumption that cannot be directly be modelled on to spatial graphs. Since such cost measures differ based on the vehicle itself, they cannot be modelled into the spatial graph and need to be modelled using the trajectory data.

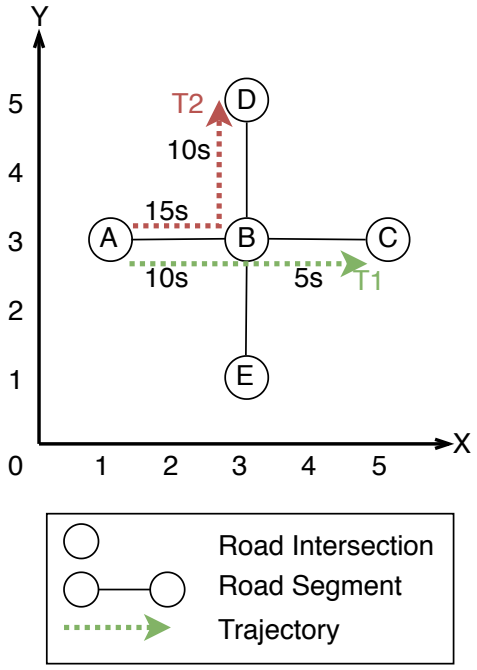



Figure 1.2: An example of a road network demonstrating the need for capturing dependence of turns in cost estimation

With the increasing concern for sustainability, several companies wish to find routes that minimize greenhouse gas emissions or fuel consumption of their vehicles [22, 30]. Transportation accounts for around 28% of the total energy used in the US [16], hence finding better, more energy efficient routes can lead to significant impact on energy consumption.

The growing accessibility of trajectory data warrants newer path selection algorithms that use not only the spatial graph, but also trajectory data, since trajectory data reflects the routing preference of the past travelers in real world, which often contains additional information that is not included in the spatial graph, such as road closure due to temporary factors. Automating map creation and editing using trajectory data has been studied [4, 19], digital maps still lag reality because the frequency of map updates often does not match the changes within cities. This is especially seen in rapidly developing cities such as Doha, Qatar or cities where roads are often occupied for other purposes such as night markets.

### 1.1.3 Capturing the dependence of turns

In domains such as calculating the fuel consumption or greenhouse gas emissions of vehicles, the dependence of turns plays a significant role. We often see that vehicle

idling at traffic signals for left turns lead to higher fuel consumption due to these turns [25]. Traditional routing algorithms that do not capture this dependence, could estimate the cost on a path to be incorrect. In these cases the cost on a path cannot simply be the sum of the edges on the path, as there is some association between the edges on the path that affect the overall cost.

Even in the case of determining the amount of time taken by a vehicle on a given path, the effect of modelling turns can provide more accurate results. For example, at a road intersection, vehicles turning left usually have to wait longer than vehicles turning right or going straight. In such cases computing the time taken by a vehicle on a road segment cannot be calculated just by the historical average on the road segment as such calculations would be skewed based on where the vehicle was going. We need a way to separate these costs based on the entire path chosen.

For example, in Figure 1.2, the circles indicate road intersections and the lines between them indicate road segments. The dashed arrows show the trajectory data available in the network, and the labels on the edges indicate the amount of time taken by the vehicle. Both trajectories $T1$ and $T2$ move along edge $AB$. $T1$ moves straight along at node $B$ to go towards $C$, whereas $T2$ makes a left at $B$ going toward $D$. The time spent on each edge by the trajectories is also shown in the figure. We can see that since $T2$ makes a left turn at intersection B, it spends longer on edge $AB$. Now if we were to compute the amount of time taken by a vehicle along the path $A-B-C$, if we did not capture the dependence of the left turn, the cost would be $(15+10)/2+5 = 17.5s$. But an capturing the dependence of the left turn would give us the cost to be $15s$, which is more accurate. The higher time on edge $AB$ for trajectory $T2$ is due to the increased waiting time for the left turn at $B$. This information cannot be captured when only looking at individual edges and ignoring the dependence between them.

To capture the dependence of turns in a spatial graph, one needs to change their frame of reference. Most algorithms that deal with spatial networks, view the network from an Eulerian frame of reference. This is when the network is viewed from the perspective of a fixed observer as opposed to a Lagrangian frame of reference, which is from the perspective of a traveller in the network.

### 1.1.4 Ensemble of Routing Algorithms

The volume and variety of spatio-temporal data generated, has posed new challenges. Just 10 million location-aware devices produce around 1 terabyte of trajectory data in 1 hour. This data can be leveraged to answer several interesting questions. The challenge

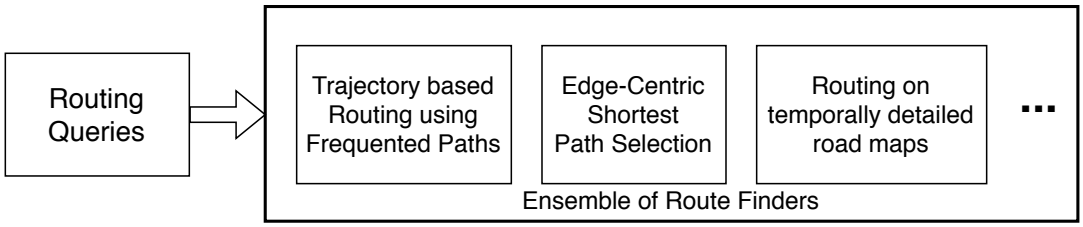



Figure 1.3: An ensemble of routing algorithms to leverage the increase in trajectory data

here is that the variety of data means that there cannot be a "one size fits all" algorithm that can be used to answer all these questions. Instead we would need an ensemble of algorithms to do this. An ensemble of algorithms consists of a variety of algorithms that individually answer a limited type of queries, but together they would be able to answer multiple types of queries by leveraging the trajectory data.

For example Figure 1.3 shows an ensemble of routing algorithms that individually answer specific queries. But the entire system as a whole could leverage these different algorithms to answer a large number of routing queries. In this thesis we will be focusing on one such problem, Trajectory based Routing using Frequented Paths.

## 1.2 Scope

In this thesis we focus on finding the lowest cost path for a given origin-destination pair, making use of trajectory data. We deal only with trajectory data in which edges have non-negative costs e.g., travel time, greenhouse gas emissions, etc. In the case of electric vehicles, if we are considering the cost metric to be the energy consumption, then we cannot consider those cases wherein energy cost is negative due to regenerative braking. Also, since our routing algorithm is completely based upon the historical trajectories, we assume that most of the trajectories in the dataset are correct and represent those paths that would be preferred by majority of the travellers in the network.

## 1.3 Summary of Contributions

In this thesis we introduce a new representation, the Maximal Frequented Path Graph (MFPG) that combines the spatial graph and the trajectory data on it into a single representation. We then propose the MFPG Shortest Path Algorithm which is a path selection algorithm that makes use of this representation. To allow an algorithm like A* to work on this new representation, we define an admissible heuristic in this representation,

given an admissible heuristic in the spatial graph. The Informed MFPG Shortest Path Algorithm makes use of this heuristic to further improve the computational efficiency of the method. The admissible heuristic helps speed up the algorithm, by guiding the search space towards the destination.

We prove that this approach is both correct and complete, and also prove both experimentally and theoretically that this approach is computationally faster than methods used in the related work.

## 1.4 Thesis Organization

The rest of the thesis is organized as follows. Chapter 2 covers the basic concepts of spatial road networks and trajectories, and also formally defines the problem statement. Chapter 3 reviews the related literature in the field and states the limitations of the methods. In Chapter 4 we define the Maximal Frequented Path Graph (MFPG) and explain how the MFPG can be constructed using the spatial graph and trajectory data. Chapter 5 goes over how we can make use of the MFPG to solve the perform Trajectory based Routing using Frequented Paths. We go over two algorithms, the MFPG Shortest Path Algorithm and the Informed MFPG Shortest Path Algorithm. Experimental results are provided in Chapter 6. Chapter 7 concludes the thesis and goes over the future work.

# Chapter 2

# Basic Concepts and Problem Statement

## 2.1 Basic Concepts

**Definition.** *A **spatial graph** $G_S = (N_S, E_S)$ is a graph consisting of a set of spatial nodes $N_S$ and a set of spatial edges $E_S$. Each spatial node $n \in N_S$ in this graph is a point in geographic space. The spatial edge $e = (n_i, n_j) \in E_S$ is an edge that connects spatial nodes $n_i$ and $n_j$, and has a shape defined by a line string.*

A spatial graph differs from an ordinary graph due to the fact that the nodes and edges now represent actual features in geographic space. Because of this, the graph allows for certain operations that are not available in ordinary graphs, such as dynamic segmentation, and allows for left and right turns within the graph.

**Definition.** *A **road network** is a system of interconnected roads composed of road segments and road intersections. The road intersections represent spatial nodes in the spatial graph, while the road segments represent spatial edges.*

A **road segment** is a uniform section of a road with similar characteristics, that connects two road intersections. A **road intersection** is a junction in the road network where two or more road segments meet.

Figure 2.1 shows an example of a spatial graph with 12 spatial nodes(road intersections) and 12 spatial edges(road segments). The circles represent spatial nodes(numbered $n1$ to $n12$) and the lines indicate spatial edges(numbered $e1$ to $e12$) between two spatial nodes.

**Definition.** *A **path** in a spatial graph is a sequence of spatial edges linking an ordered sequence of spatial nodes.*

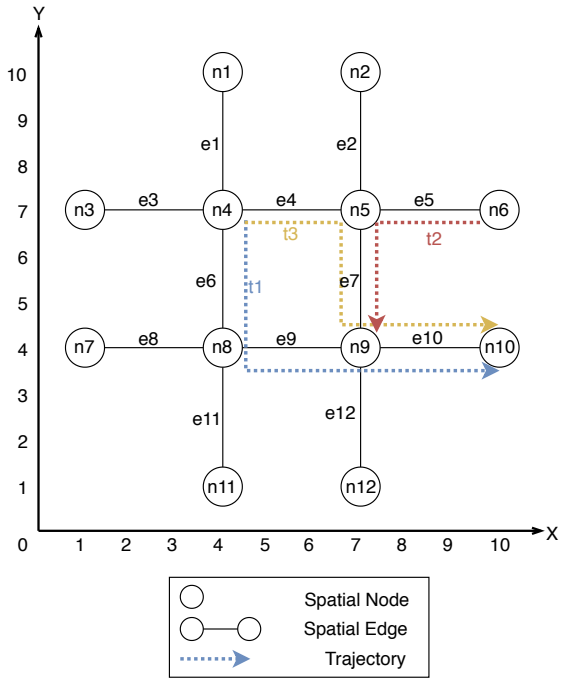



Figure 2.1: An example of a spatial graph with trajectories

**Definition.** *A **subpath** is a subset of a path made up of consecutive spatial nodes on the path.*

**Definition.** *The **union** ($\bigcup$) **of two paths** $P_\alpha$ and $P_\beta$ at a spatial node is composed of all spatial edges of $P_\alpha$ before the spatial node and all spatial edges of $P_\beta$ after the spatial node.*

In Figure 2.1, a path could be represented by the ordered sequence $[e1, e6, e9, e10]$. The sequence $[e1, e6, e9]$ would be a subpath of this path. The union of two paths, $[e1, e6, e9]$ and $[e6, e9, e10]$ would be a new path $[e1, e6, e9, e10]$.

**Definition.** *A **trajectory** is the log of a path travelled by a vehicle in the spatial graph, that consists of the spatial edge and the travel cost along that spatial edge.*

The dotted lines in Figure 2.1 represent trajectories in this spatial graph. There are 3 trajectories in the spatial graph ($t1$, $t2$, and $t3$). Trajectory $t1$ travels along the path $[e6, 39, e10]$.

**Definition.** *A **Frequented Path** ($FP$) is a path along which there are at least a certain number of trajectories($\beta$) in the same direction.*

The $FP$ in the graph, indicate those paths in the graph, that are frequented by travellers. In Figure 2.1, if we assume the number of trajectories along a $FP$ to be 1 ($\beta = 1$), then $[e6]$, $[e9, e10]$, and $[e6, e9, 10]$ would be some of the $FPs$ in the spatial graph. When $\beta$ is 2, [e7] and [e10] are the only $FPs$ in the graph. There are no $FPs$ in the graph when $\beta$ is 3.

**Definition.** *A **Union of Frequented Sub-Paths** ($UFP$) is the union of two or more sub-paths of $FPs$ that have overlapping spatial edges such that it is not the subpath of any other $FP$.*

$UFPs$ represent those paths in the spatial graph, which are not always frequented by travellers in their entirety, but since they are the union of $FPs$, they are still significant paths in the graph. In Figure 2.1 if we still assume the number of trajectories along a $FP$ to be 1, then we can see that $t2$ is along path $[e5, e7]$ and $t3$ is along path $[e4, e7, e10]$. Hence we can see that path $[e5, e7, e10]$ is the union of the sub-paths of the two $FPs$ at $n9$ and is hence a $UFP$. There is however no trajectory along path $[e8, e9, e10]$, so it is neither a $FP$ nor a $UFP$.

For this problem we only focus on finding those paths that are either Frequented Paths($FP$) or Union of Frequented Sub-Paths($UFP$).

The estimation of the cost of a path in a spatial graph can be done in different ways. **Edge-centric approaches** treat spatial edges to be the fundamental units in the spatial graph. They estimate the cost of a path as the sum of the individual spatial edges along that path. In doing so they fail to capture the dependence of edges while estimating the cost on a given path. As seen earlier, the dependence of edges helps us capture the dependence of turns on the overall cost of a path. This plays a significant role when the cost function is a measure of either fuel consumption, greenhouse gas emissions or even time, as these metrics are significantly affected by turns in the spatial graph.

**Path-centric approaches** on the other hand consider a path to be the basic spatial unit of a spatial graph. These methods estimate the cost on a path by decomposing the original path into sub-paths and then combining the paths on them [32, 24]. In doing so, they preserve the dependence of the individual edges and can hence capture the dependence of turns in the cost estimation. Hence these methods are more apt when estimating the cost of greenhouse gas emissions, fuel consumption or any other metrics that are affected by turns. However in order to estimate the cost while preserving the

dependence of edges, these methods often require a lot more data in the network in order to be used.

## 2.2 Problem Definition

The Trajectory based Routing using Frequented Path problem can be formally defined as follows:

**Input:**

1. A spatial graph
2. A set of historic trajectories in the spatial graph
3. A cost estimation model for Frequented Paths and Union of Frequented Sub-Paths
4. The minimum number of trajectories along a $FP(\beta)$
5. Start and end spatial nodes($o$ and $d$)
6. An admissible heuristic in the spatial graph (if using the Informed MFPG Shortest Path Algorithm)

**Output:** Path between o and d with minimum cost (e.g., Energy consumption. time, etc.)

**Objective:** Improve the computational efficiency of the path selection algorithm

**Constraints:**

1. The output path will either be a Frequented Path ($FP$) or a Union of Frequented Sub-Path ($UFP$)
2. The cost on any path is non negative (no regenerative braking in electric vehicles)
3. There are trajectories on all spatial edges in the spatial graph

# Chapter 3

# Literature Review

The Trajectory Based Routing problem is a variant of the Shortest Path Selection problem. The related literature can be divided based upon the spatial unit in the network as shown in Figure 3.1. Some literature considers an edge to be the basic spatial unit of a network, and use this for path cost estimation or path selection (shown as the left branches in the tree). These are edge-centric methods. Other methods use paths as the basic spatial units in the network, and are known as path-centric methods (shown as the right branches in the tree).

## 3.1 Shortest Path Selection

The Trajectory Based Routing problem is a variant of the Shortest Path Selection problem. This problem has been widely studied and several algorithms have been developed to solve it. A lot of these solutions are based on either Dijkstra's algorithm or Bellman-Ford algorithm [13, 5, 18], that work on static-weighted graphs that where the cost on

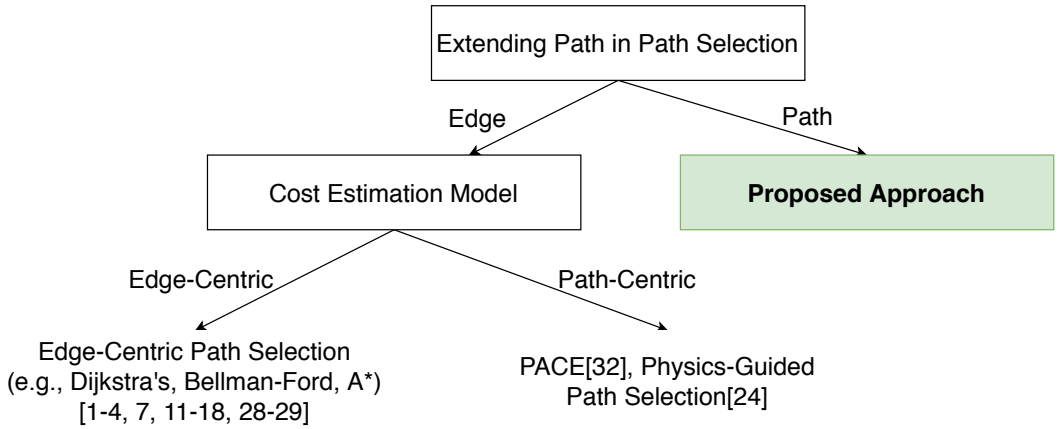



Figure 3.1: Tree of related work

every edge is a constant.

A lot of work has been done to make the computations of these algorithms faster as seen in [1, 3, 11, 29]. Several papers try to add new constraints to the problem [28], such as bi-objective optimization [14], or adding battery constraints to the problem, to work with electric vehicles [2, 15, 21, 33, 9]. Some work has been done in adapting these algorithms to work with multigraphs as well [7]. While most of these methods assume the costs on the edges to be known beforehand, some works do make use of trajectory data as well, to model the travel times on road segments [10, 26]. However these methods only use the trajectory data to get a distribution of the travel time on the road segment [12, 17, 23].

These algorithms however are all edge-centric algorithms. In these algorithms the fundamental unit of a network is an edge and the cost on a given path is calculated by adding the individual weights on the edges that make up the path. In contrast, path-centric based models consider a path to be the basic spatial unit in a graph. These algorithms are far better at capturing the dependence of edges in estimating the cost of an entire path.

## 3.2 Path-centric cost estimation

As mentioned above, a lot of literature has spoken about solving shortest path selection problems using an edge-centric approach, in which an edge is considered to be the fundamental unit of a graph, and calculate the cost on a path to be the sum of the cost on its individual edges. These approaches fail to reflect the dependence of edges calculating the cost on a path. In several applications, the cost metric would be more accurate if the cost estimation model reflected this dependence. Even in the case of determining the amount of time taken by a vehicle on a given path, the effect of modelling turns can provide more accurate results.

Recently some work has been done to model the dependence between edges to provide more accurate cost measures. In [31], they propose an end to end Deep learning model for travel time estimation that can estimate the travel time of an entire path, and hence models all the dependence between the edges. PACE [32] discusses a path-centric approach for cost estimation in a network. In this paper, they decompose paths into coarse overlapping sub-paths that still model the relationships between the edges on the sub-paths, by not splitting the trajectory into small fragments. Building upon this idea of path-centric cost estimation, [24] talks about a physics guided energy efficient path selection algorithm that works to find the most energy efficient route for electric

vehicles. Here too they make use of frequented paths (called trajectory-aware paths) as the basis of their cost estimation model.

However in both [24] and [32], the path selection process still extends a path one edge at a time. When finding the route from an origin to a destination both of these methods extend the current path by only one edge in every iteration. Although this would still give us the correct results, we could speed up this operation and extend paths by multiple edges in each iteration.

# Chapter 4

# Maximal Frequented Path Graph (MFPG)

## 4.1 Frequented Path ($FP$)

Since a spatial graph could have a large number of trajectories, we need a way to efficiently handle all this data. We make use of the Frequented Paths ($FPs$) to remove any outliers in the trajectory data, and only focus on those paths that have been frequently traversed by travellers. A $FP$ would have at least a certain number of trajectories along it ($\beta$). Setting $\beta$ to an appropriate value can help us only focus on these paths that are frequented by travellers, and eliminate some of the noise in the trajectory data. The choice of $\beta$ plays a very important role in determining the number of $FPs$ in the spatial graph. A higher value of $\beta$ will reduce the number of $FPs$ in the spatial graph, and hence reduce the computational time of the path selection algorithm. But since the algorithm only finds paths between the origin and destination that are either Frequented Paths($FPs$) or Union of Frequented Sub-Paths ($UFPs$), a high value of $\beta$ would make some origin-destination pairs unreachable. A lower value of $\beta$ would increase the number of $FPs$ in the spatial graph, and increase computational time. Hence selecting a good value of $\beta$ is essential in the efficient running of the algorithm.

Figure 4.1 shows 2 trajectories in a spatial graph. Trajectory $t1$ moves along the edges $[e1, e2]$ while the other trajectory $t2$ moves along $[e1, e2, e3]$. Figure 4.2 shows the $FPs$ in the spatial graph for $\beta = 1$ and 2. When $\beta = 1$, there are a lot of $FPs$ in the spatial graph, as seen in Figure 4.2a. Increasing the value of $\beta$ from 1 to 2, reduces the number of $FPs$ in the graph. Since 2 trajectories do not travel along edge $e3$, the paths $[e1, e2, e3]$, $[e2, e3]$, and $[e3]$, which were $FPs$ when $\beta$ was 1 are no longer $FPs$ when $\beta$ is 2. Although we see that the number of $FPs$ has reduced, there is now no $FP$

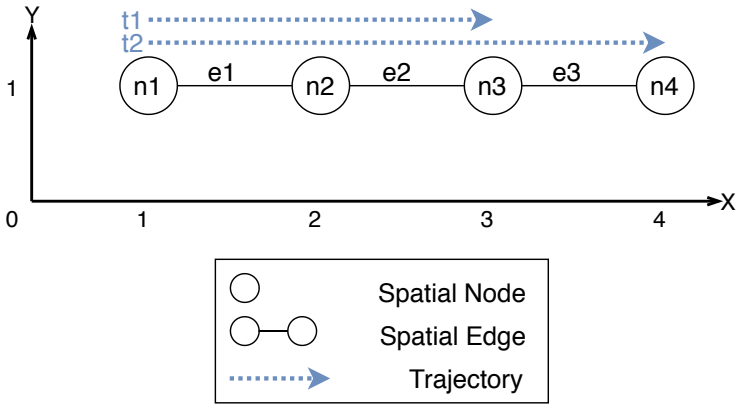
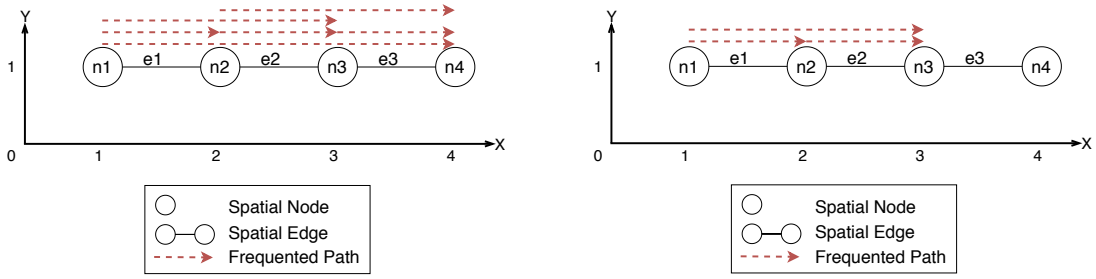



Figure 4.1: Two trajectories in a spatial graph




(a) $\beta = 1$ (b) $\beta = 2$

Figure 4.2: *FPs* in spatial graph for $\beta = 1$ and 2

from the spatial node $n1$ to $n4$. So the algorithm would not find any paths between this origin-destination pair.

## 4.2 Maximal Frequented Path ($MFP$)

Since a $FP$ would have at least $\beta$ trajectories along it, any sub-path of it would also have at least $\beta$ trajectories along it and would also be a $FP$, as seen in Figure 4.2. Hence the number of $FPs$ in the spatial graph would be high. As a means to eliminate the redundancy in the spatial graph, we come up with the concept of a Maximal Frequented Path ($MFP$).

**Definition.** *A **Maximal Frequented Path** ($MFP$) is a Frequented Path that is not the sub-path of any other Frequented Path.*

Any spatial node that was reachable in the spatial graph using $FPs$ would still be reachable by only using $MFPs$. Also, since a $MFP$ is a $FP$, the union of two $MFPs$ at a node might still form a $UFP$. Hence we can model the entire spatial graph correctly

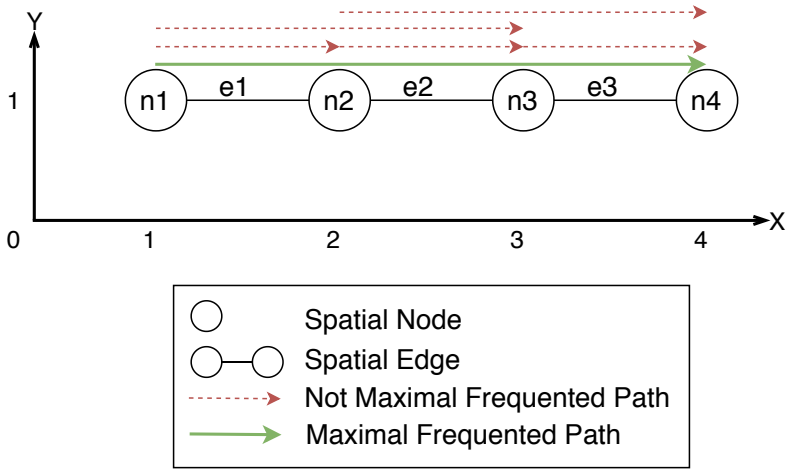



Figure 4.3: Maximal Frequented Paths ($MFPs$) from Figure 4.1 when $\beta = 1$

by just making use of $MFP$s. We would not be losing out on any information and the number of paths to consider is significantly reduced.

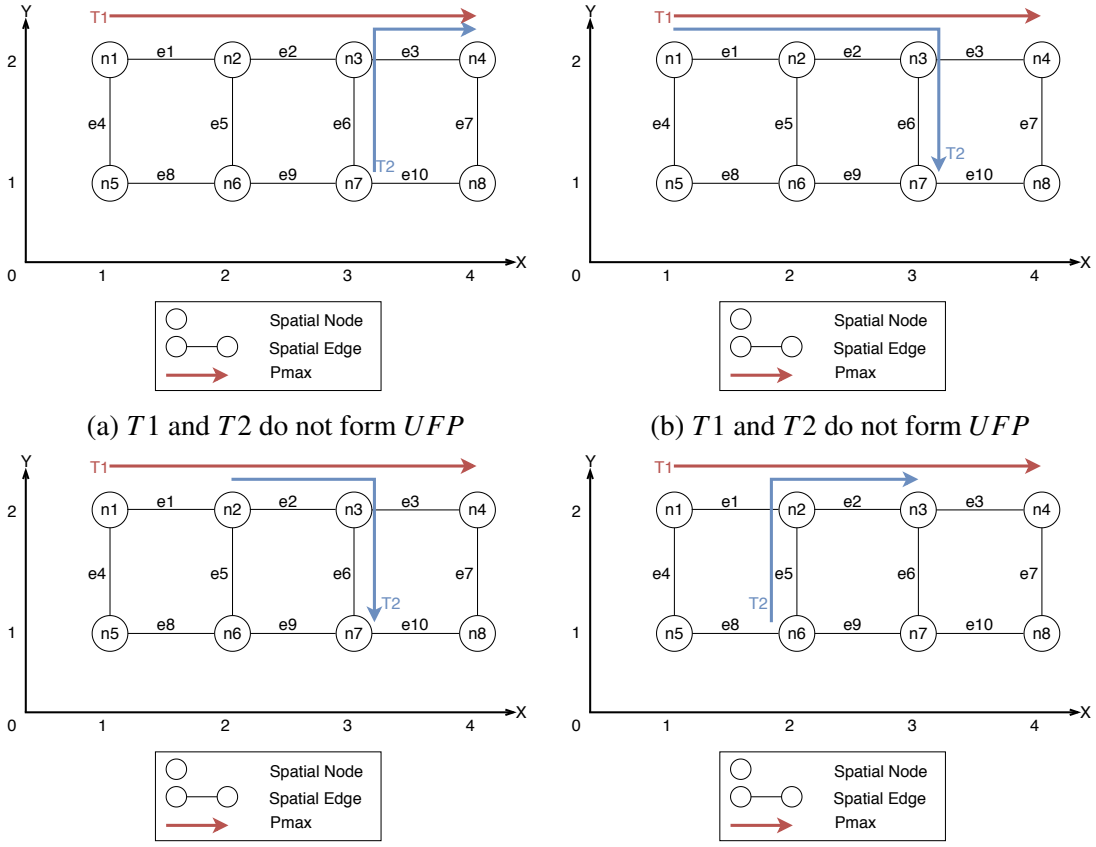
Figure 4.3 shows examples of the $MFP$ from Figure 4.2a. The solid arrows show those $FPs$ that are also $MFP$, as they are not the sub-paths of any other $FPs$. The dashed arrows show the $FPs$ that are not $MFP$s as they are sub-paths of other $FPs$. For example the paths $[e1]$, $[e2]$, $[e3]$, $[e1,e2]$, $[e2,e3]$ are not a $MFP$ because they are all sub-paths of $[e1,e2,e3]$. $[e1,e2,e3]$ is the only $MFP$ in the spatial graph as it a $FP$ that is not the sub-path of any other $FP$.

## 4.3 Maximal Frequented Path Graph (MFPG)

We need to model the relationship between several $MFP$ in the spatial graph and how they join to form $UFP$s. To do this we make use of a representation called a Maximal Frequented Path Graph.

**Definition.** *A **Maximal Frequented Path Graph (MFPG)** is a directed graph whose MFP-Nodes are Maximal Frequented Paths (MFPs). An MFP-Edge exists between two MFP MFP-Nodes ($P_a$ and $P_b$) if $P_a \bigcup P_b$ form a UFP at a spatial node in the spatial graph.*

The MFP-Nodes in the Maximal Frequented Path Graph represent the $MFP$ that are paths in the spatial graph, and have associated travel costs that are estimated using the trajectory data. Since the MFPG represents both the path and the travel cost on the path estimated using the trajectory data available for that path, the MFPG is a combination




(a) $T1$ and $T2$ do not form $UFP$

(b) $T1$ and $T2$ do not form $UFP$

(c) $T1$ and $T2$ form $UFP$ at $n3$

(d) $T1$ and $T2$ form $UFP$ at $n3$

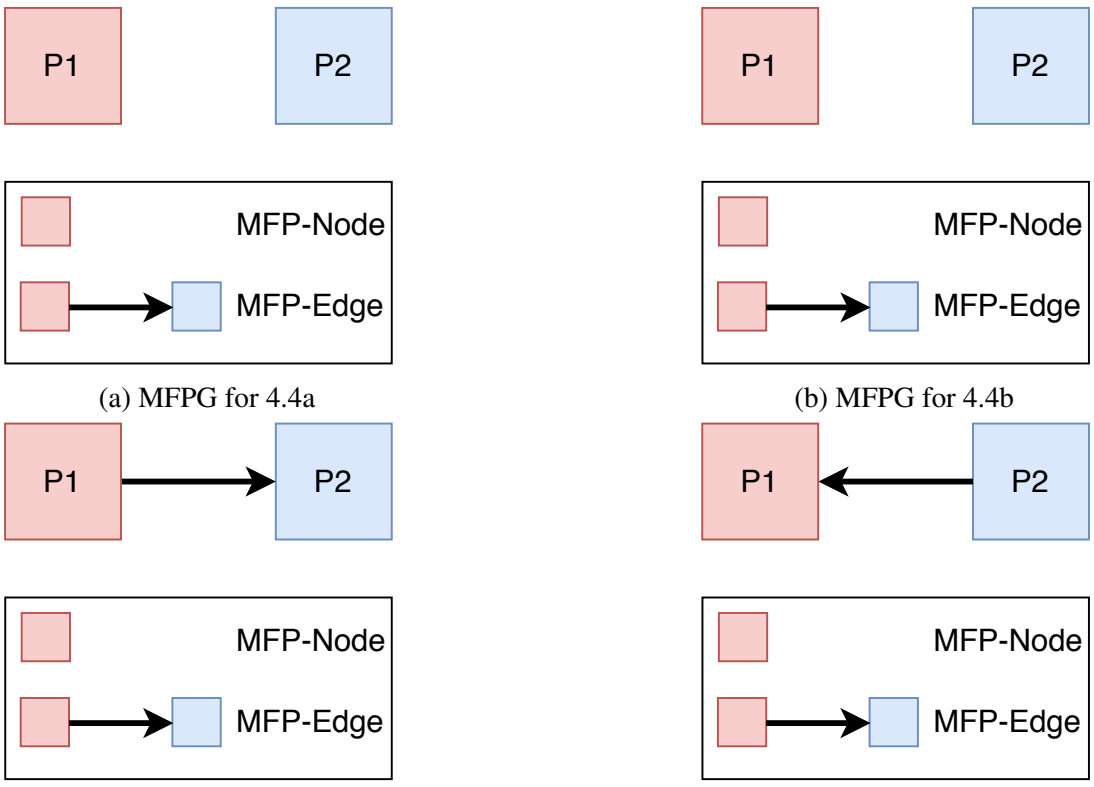
Figure 4.4: Example showing when $MFPs$ form $UFPs$ at nodes

of both the spatial graph and the trajectory data on it. It combines the trajectory data and spatial graph into one representation.

To differentiate the nodes and edges in the spatial graph and the MFPG, we refer to the nodes and edges in the spatial graph as base nodes and base edges, and the nodes and edges in the MFPG as MFP-Nodes and MFP-Edges.

### 4.3.1 Constructing the Maximal Frequented Path Graph

Given a spatial graph and trajectory data on the spatial graph, we can convert it into a MFPG, using the transformation based on the definition of the MFPG. First, every $MFP$ in the spatial graph will become a MFP-Node. Now the MFP-Edges is added between two MFP-Nodes if the two $MFPs$ form a $UFP$ at a base node. Figure 4.4 shows a few examples of when $MFPs$ form a $UFP$ in the spatial graph. Figure 4.5 shows the corresponding MFPGs for these examples. In Figure 4.4a, $T1$ and $T2$ do not form a $UFP$ even though they overlap along the base edge $e3$. This is because along




(a) MFPG for 4.4a

(b) MFPG for 4.4b

(c) MFPG for 4.4c

(d) MFPG for 4.4d

Figure 4.5: MFPGs for the Examples in Figure 4.4

this base edge, a traveller can either be on $T1$ or $T2$, but there is no way to switch from either $T1$ to $T2$ or $T2$ to $T1$. Hence Figure 4.5a shows no MFP-Edge between the two MFP-Nodes. Similarly in Figure 4.4b, $T1$ and $T2$ do not form a $UFP$. At $n3$ if one travels toward $n4$ along $e3$, it would be as though they were always on $T1$. A similar case occurs if they go along $e6$ to $n7$. Hence Figure 4.5b shows no MFP-Edge between the two MFP-Nodes. In Figure 4.4c, if one is travelling along $T1$ from $n1$, then at $n3$, if they follow the path along $T2$, and go towards $n7$, then the path $[e1, e2, e6]$, becomes a $UFP$. Since one can switch from $T1$ to $T2$ at $n3$, in Figure 4.5c, an MFP-Edge exists between the two MFP-Nodes, directed from $P1$ toward $P2$. In the final case in Figure 4.4d, if one is travelling along $T1$ from $n1$, then that traveller would always be on $T1$. If however a traveller is travelling along $T2$, from $n6$, then at $n3$, if the traveller continues travelling along $e3$, they are now on $T1$. So we see that $T1$ and $T2$ form a $FP$ at $n3$. Since we can travel from $T2$ to $T1$, in Figure 4.5d an MFP-Edge exists between $P1$ and $P2$ directed from $P2$ to $P1$.

Figure 4.6 shows an example of a few more trajectories in a spatial graph. As before the circles represent spatial nodes and the solid lines indicate spatial edges between two

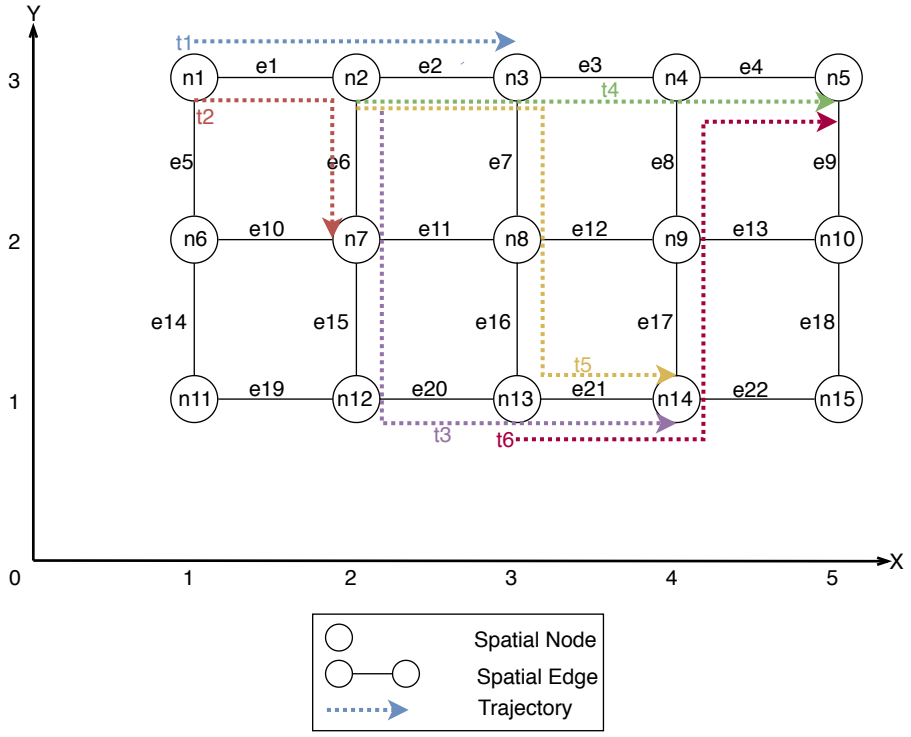



Figure 4.6: Example of trajectories on a spatial graph

| ID | Trajectory Records | | | | |
|---|---|---|---|---|---|
| t1 | Road Segment | e1 | e2 | | |
| | Cost | 2 | 9 | | |
| t2 | Road Segment | e1 | e6 | | |
| | Cost | 1 | 1 | | |
| t3 | Road Segment | e6 | e15 | e20 | e21 |
| | Cost | 3 | 2 | 4 | 3 |
| t4 | Road Segment | e2 | e3 | e4 | |
| | Cost | 7 | 9 | 2 | |
| t5 | Road Segment | e2 | e7 | e16 | e21 |
| | Cost | 9 | 2 | 2 | 7 |
| t6 | Road Segment | e21 | e17 | e8 | e4 |
| | Cost | 3 | 2 | 2 | 2 |

Table 4.1: Example of trajectory data

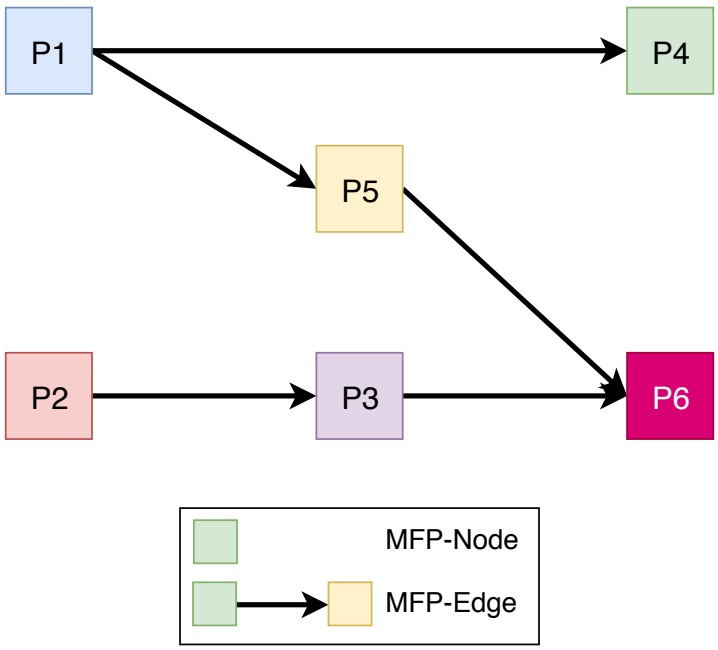



Figure 4.7: MFPG for the example in Figure 4.6

spatial nodes. The dotted arrows represent trajectories in this graph. Table 4.1 provides more details about the trajectory data. Each row in the table indicates the record for one trajectory. It contains information about the spatial edge taken and the cost on that spatial edge.

We assume the minimum number of trajectories on a $FP(\beta)$ to be 1, so that all of the trajectories are also $FPs$. We also assume that the minimum number of edges that should overlap for two $FPs$ to form a $UFP$ is 1. We observe that none of the trajectories are sub-paths of each other, so for this spatial graph all of the trajectories are also $MFPs$. We now convert this spatial graph into a Maximal Frequented Path Graph(MFPG) as shown in Figure 4.7. Each $MFP$ in the base spatial graph will be a MFP-Node in the MFPG. So we have 6 MFP-Nodes in the MFPG, $P1$ to $P6$, that corresponds to the 6 $MFP$, $t1$ to $t6$ in the base graph. To model the MFP-Edges, we need to find the $MFP$ whose union form a $UFP$ at a base node. For example we observe that $t1$ and $t4$ form a $UFP$ at $n3$. So an MFP-Edge exists between $P1$ and $P4$. Similarly MFP-Edges exist between $P1$ and $P5$ because they form a $UFP$ at $n3$, $P2$ and $P3$ at $n7$, $P3$ and $P6$ at $n14$ and $P5$ and $P6$ at $n14$.

We observe that $P1$ and $P2$ are not connected by an MFP-Edge in the MFPG even though they seem to have an overlap of 1 base edge($e1$). This is because, although they do have an overlap at $e1$, there is no way to transition from $P1$ to $P2$. If from $e1$ one travels to $e2$, then at $e1$ they are on $P1$, and if from $e1$ one travels along $e6$, then at $e1$

they are on $P2$. But one cannot transition between the 2. A similar observation can be made at MFP-Nodes $P4$ and $P6$, where the $MFP$s end at the same base node. Since one cannot transition between $P4$ to $P6$, these MFP-Nodes do not have an MFP-Edge between them in the MFPG.

### 4.3.2 Internal Representation of the MFPG

The MFP-Nodes in the MFPG, are actually $MFPs$ from the base graph. So internally each MFP-Node contains paths from the base graph. Figure 4.8 shows the internal representation of a single MFP-Node. In Figure 4.8a there are 4 MFP-Nodes in the MFPG, $P1$ to $P4$. All of these MFP-Nodes represent $MFPs$ in the base graph. The MFP-Edges show the $MFPs$ that form $UFPs$ in the base graph. For example the MFP-Edge between $P1$ and $P2$ indicates that $P1$ and $P2$ form a $UFP$ at some base node. Figure 4.8b shows the internal representation of MFP-Node $P2$ from Figure 4.8a. Internally, it is made up of a path from the spatial base graph, consisting of the spatial nodes $n1$ to $n7$. At some of these spatial nodes, the $MFP$ forms a $UFP$ with another $MFP$. $P2$ forms a $UFP$ with $P3$ at the base node $n2$, and it forms a $UFP$ with $P4$ at the base node $n6$. Since these are the only two base nodes in the path at which the $MFP$ forms a $UFP$ with any other $MFP$, these base nodes are called Transfer Spatial nodes, and are shaded in Figure 4.8b.

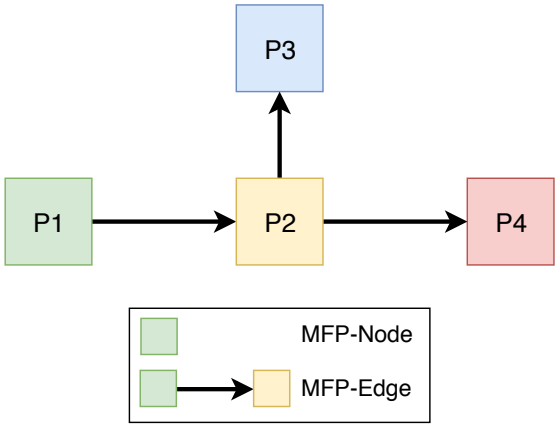
**Definition.** *A **Transfer Spatial Node** is a spatial node from the base graph at which a $MFP$ forms a $UFP$ with another $MFP$.*

Since these are the base nodes at which two $MFPs$ from a $UFP$, the transfer spatial nodes represent the outgoing MFP-Edges of MFP-Nodes. The MFP-Nodes that do not form $UFPs$ with any other MFP-Nodes do not have any transfer nodes.

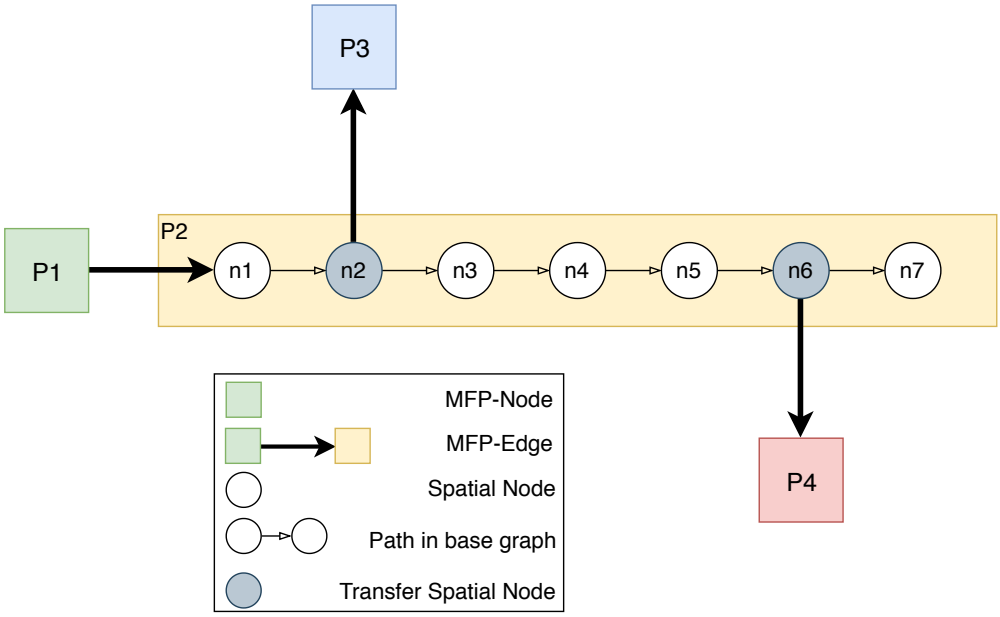
Figure 4.9 shows the internal representation of the MFPG constructed in Figure 4.7. Here too we see that every MFP-Node contains the entire $MFP$ from the base graph. Some of the nodes in the MFP-Nodes are shaded to indicate these are the transfer spatial nodes. For example in the MFP-Node $P1$, $n3$ is a transfer node as $P1$ forms a $UFP$ with both $P4$ and $P5$ at the base node $n3$. Since $P4$ and $P6$ do not form $UFPs$ with any other MFP-Nodes, they do not contain any transfer nodes.

### 4.3.3 Path in a Maximal Frequented Path Graph

A path in a MFPG is an ordered sequence of MFP-Nodes that are connected by MFP-Edges in the MFPG. Since each MFP-Node in the MFPG represents a $MFP$, the union




(a) Example of a MFPG with 4 MFP-Nodes




(b) Internal representation of the MFPG in 4.8a

Figure 4.8: Internal representation and transfer spatial nodes in a MFPG

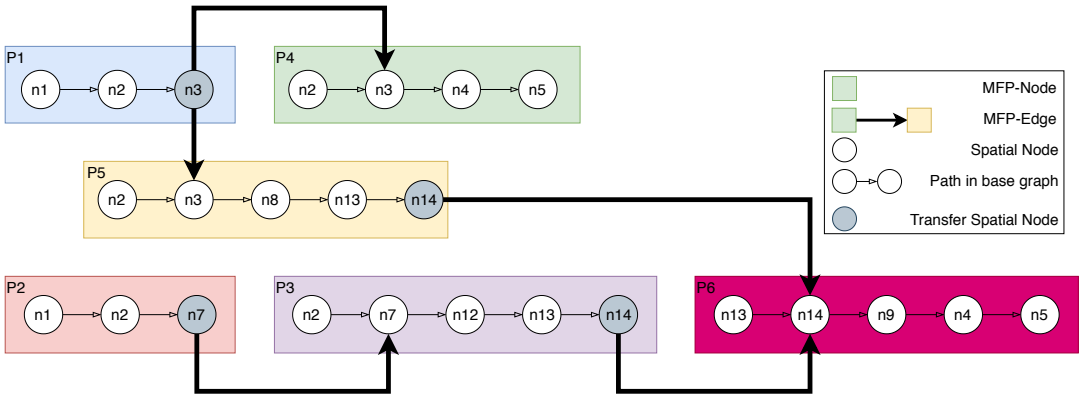



Figure 4.9: Internal representation and transfer spatial nodes of the MFPG in Figure 4.7

of $MFP$s will give us a path in the spatial graph as well. A path in the spatial graph is represented by a path in the MFPG, if the path in the spatial graph is a subset of the path formed by the union of the $MFP$s in the MFPG, and the origin and the destination in the spatial graph lies in the first and last $MFP$ of the path in the MFPG respectively.

Since a $MFP$ contains several spatial nodes in it, several paths in the spatial graph could be represented by a path in the MFPG. The cost of a path in the MFPG is the cost of the $UFP$ made up of all but the last $MFP$ in the path and excluding the spatial edges before the origin. Because of this, the cost of the path in the MFPG is always less than or equal to the cost of the path in that spatial graph. For example, the cost of the path $[P1, P5, P6]$ in Figure 4.7 , having the origin at $n1$ is represented by the cost of the $UFP$ $[e1, e2, e7, e16, e21]$.

The cost of the entire path from the origin to the destination is calculated by adding the cost of the last $MFP$ till the destination base node to the cost of the path in the MFPG.

To estimate the cost on a given path we use a simplified version of the cost estimation models used in [32, 24]. The model estimates travel cost on a $FP$ by the average cost of the trajectories along it, and estimates that on the overlapping spatial edges of two $FPs$ by the mean of their cost on the spatial edge. For example, the cost on each spatial edge of the path $[e1, e2]$ would be [2, 9], while the cost on the path $[e1, e2, e3, e4]$ would be [2, 8, 9, 2]. So the cost alone $[P1, P4]$ in the MFPG, would be [2, 8].

### 4.3.4 Dynamic Multigraph

Since $FPs$ and hence $MFP$s can merge and split at multiple different base nodes, in the MFPG the MFP-Nodes could have multiple MFP-Edges between them. Hence the MFPG is a multigraph. There could also be cycles in the MFPG.

However a MFP-Edge in the MFPG is determined by the previous MFP-Nodes and MFP-Edges in the path. For example in Figure 4.10, we have 2 $MFP$s in the spatial graph, $P1$ and $P2$. In the MFPG for the spatial graph, an MFP-Edge exists between $P1$ and $P2$ at $n3$. MFP-Edges exist between $P2$ and $P1$ at $n3$ and $n5$. If the MFPG was a common multigraph then we could move between MFP-Edges $P1$ and $P2$ multiple times because of the existence of the cycle. However, the MFPG is representing the paths on a spatial graph. If we move from $P2$ to $P1$ at either $n3$ or $n5$, then we are not be able to go back to $P2$, as we have already crossed all the possible base nodes that allow this transfer in the path. Hence there is no way to get back to $P2$. This avoids the existence of cycles in the MFPG. Hence a path on the MFPG must keep track of both

the MFP-Nodes (the $MFP$) on the path, as well as the MFP-Edges (the transfer spatial nodes at which the 2 $MFP$s form a $UFP$).

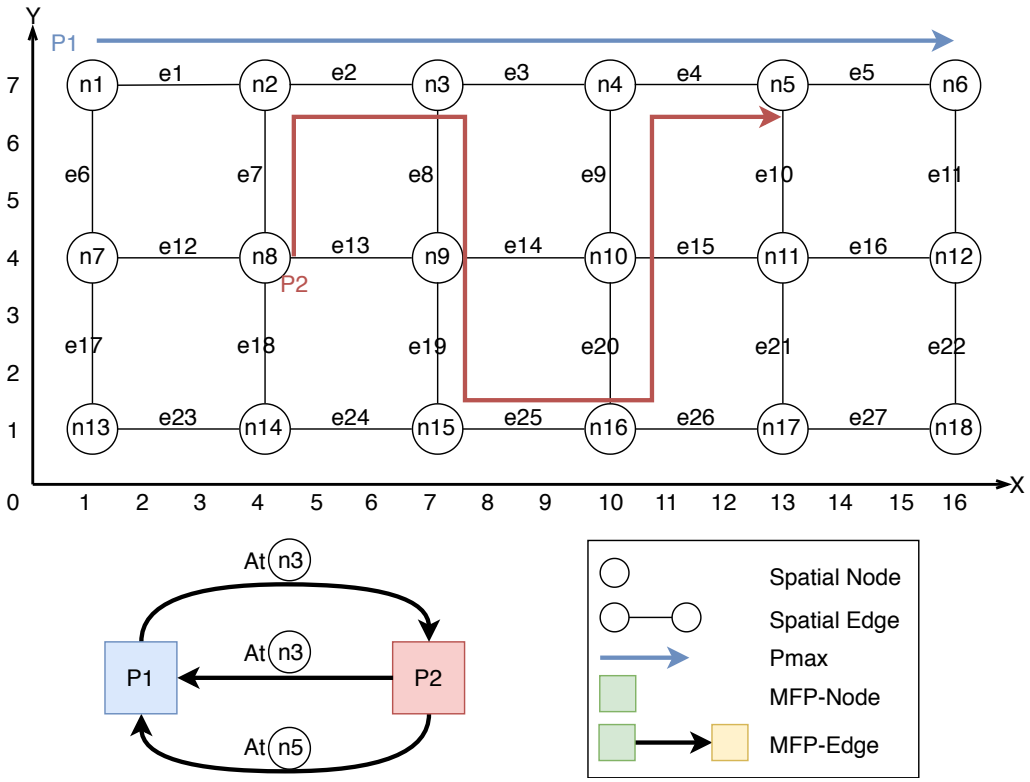



Figure 4.10: Example of Dynamic nature of MFPG

# Chapter 5

# Frequented Path Lowest Cost Path Selection

## 5.1 Path selection algorithm framework

Most path selection algorithms follow a general framework as shown in Algorithm 1 to find a path between an origin and destination. Given a spatial graph $G_S$, and an origin and destination ($o$ and $d$), the algorithm makes use of the provided cost estimation model to find the path from $o$ to $d$ that satisfies a certain criteria. The criterion could be minimizing the overall cost of the path, or in some cases a more complex cost estimation model could be used for bicriterion optimization[14], etc.

Line 1 of the algorithm initializes a list of candidate paths that are usually those spatial edges that the origin is on. For each iteration of the loop (2-8) these paths are extended, until some condition is met. In most cases, such as those used in Dijkstra's [13] or Bellman-Ford [5] or any other algorithm based on these[18], the stop condition is that of finding a path from $o$ to $d$ with the minimum cost. These algorithms, extend the paths one spatial edge at a time in each iteration on Line 4. These methods follow the "path + edge" pattern where one spatial edge is added to the currently selected path to get the new path. After the new path is computed, Line 5 estimates the cost of this new path and it is added to the list of candidate paths.

In edge-centric models, the cost estimation is done in a edge-centric fashion, wherein the cost of the path is computed as the sum of the cost of the individual spatial edges that make up the path. However in a path-centric approach, the costs are estimated using more sophisticated ways, that decompose the paths into sub-paths rather than spatial edges to compute a more accurate cost, as they can capture the dependence of spatial edges in the overall cost estimation of a given path. Such cost estimation methods have

been used in [32, 24]. These methods however still extend the paths in a "path + edge" approach, as edge-centric models.

**Algorithm 1** General Algorithm Framework

**Require:**
$G$ : A spatial Network
$o$ and $d$ : Two nodes
$model$ : A cost estimation model
**Ensure:**
The path between $o$ and $d$ satisfying the criteria
1: candidate partial-paths $CP \leftarrow$ initialization
2: **while** stop criteria are not met **do**
3: $p \leftarrow$ the most promising partial-path in $CP$
4: **for all** extensions $p'$ of $p$ **do**
5: compute the cost of $p'$
6: add $p'$ to $CP$
7: **end for**
8: **end while**

A detailed execution trace of finding the path from $n1$ to $n5$ in Figure 4.6, using the method in [24] is shown in Table 5.1. $CP$ shows the list of candidate paths available at each step, where as Cost shows the costs of these paths. The last column $p$, indicates the Candidate path with the lowest cost that is selected for expansion in the next step. On Line 1 edge $e1$ is the only available candidate path that the origin is on. For Line 2, the stop condition is that there is no candidate path that has a cost lower than the lowest cost path from the origin to the destination. In every iteration the candidate path with the lowest cost selected and is extended by adding one edge to it. Only those edges that form paths that are $FPs$ or $UFPs$ are selected in each iteration. For example while expanding edge $e1$, we observe that $[e1, e2]$ and $[e1, e6]$ are both $FPs$, so these edges are selected. Since there are no trajectories along $[e1, e5]$, it is not a $FP$ or a $UFP$ so it is not selected. These new paths are added back to the set of candidate paths, while the old path is removed. When a path reaches the destination, its cost is computed and if it is lower than the current lowest cost path, the lowest cost path is updated. This process continues until all paths in the $CP$ have a cost greater than the lowest cost path. Line 11 of Table 5.1 shows a path $[e1, e6, e15, e20, e21, e17, e8, e4]$ that reaches the destination with a cost of 18. This path is then removed from the list of candidate paths and stored as the lowest cost path. On the last line we see that the cost of the other two candidate paths is greater than that of the lowest cost path. So the Algorithm terminates returning the path with cost 18.

| Step | *CP* | Cost | *p* |
|---|---|---|---|
| 1 | [e1] | 1.5 | [e1] |
| 2 | [e1, e6], [e1, e2] | 2,11 | [e1, e6] |
| 3 | [e1, e6, e15], [e1, e2] | 5,11 | [e1, e6, e15] |
| 4 | [e1, e6, e15, e20], [e1, e2] | 9,11 | [e1, e6, e15, e20] |
| 5 | [e1, e6, e15, e20, e21], [e1, e2] | 12,11 | [e1, e2] |
| 6 | [e1, e6, e15, e20, e21], [e1, e2, e3], [e1, e2, e7] | 12,19,13 | [e1, e6, e15, e20, e21] |
| 7 | [e1, e6, e15, e20, e21, e17], [e1, e2, e3], [e1, e2, e7] | 14,19,13 | [e1, e2, e7] |
| 8 | [e1, e6, e15, e20, e21, e17], [e1, e2, e3], [e1, e2, e7, e16] | 14,19,15 | [e1, e6, e15, e20, e21, e17] |
| 9 | [e1, e6, e15, e20, e21, e17, e8], [e1, e2, e3], [e1, e2, e7, e16] | 16,19,15 | [e1, e2, e7, e16] |
| 10 | [e1, e6, e15, e20, e21, e17, e8], [e1, e2, e3], [e1, e2, e7, e16, e21] | 16,19,22 | [e1, e6, e15, e20, e21, e17, e8] |
| 11 | [e1, e6, e15, e20, e21, e17, e8, e4], [e1, e2, e3], [e1, e2, e7, e16, e21] | 18,19,22 | [e1, e6, e15, e20, e21, e17, e8, e4] |
| 12 | [e1, e2, e3], [e1, e2, e7, e16, e21] | 19,22 | |

Table 5.1: Execution trace of Physics Guided Path Selection Algorithm

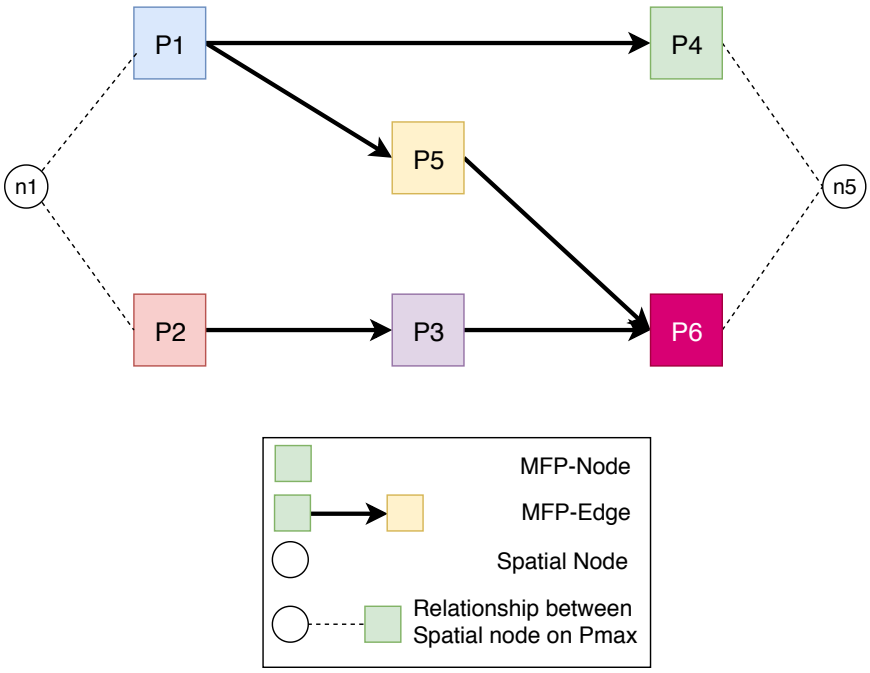



Figure 5.1: MFPG augmented with origin and destination for path selection

## 5.2 MFPG Shortest Path Algorithm

In the MFPG Shortest Path Algorithm, we wish to extend the paths in a "path + path" fashion. To enable this the candidate paths would now be $MFPs$ from the MFPG instead of base edges as in the previous method. The algorithm will follow the general framework as shown in Algorithm 1, but will differ in how the paths are extended. On Line 1, the candidate paths are initialized with all the $MFPs$ that the origin lies on. The stop criteria in Line 2, is that there is no Candidate path that has not been explored that has a cost lower than the cost of the lowest cost path found between the origin and destination. In Line 4 the $MFPs$ are added to the currently selected path, if the union of the two form a $UFP$. This is equivalent to traversing an MFP-Edge in the MFPG.

When the path is extended to the destination it is removed from the candidate path set and its cost is estimated. If the cost is less than the current lowest cost path from the origin to the destination, the path and the costs are updated to reflect the new lowest cost paths.

If we want to find the lowest cost path from $n1$ to $n5$ as above, now we would first convert the input spatial graph and trajectory data into its equivalent MFPG. We would also need to augment the graph with some spatial data about the origin and destination to indicate the $MFPs$ on which the origin and destination lie. This is shown by the dashed lines in Figure 5.1. The input spatial graph with 15 nodes and 22 edges is

| Step | *CP* | Cost | *p* | Result Path (Cost) |
|---|---|---|---|---|
| 1 | [P1], [P2] | 0,0 | [P1] | |
| 2 | [P1, P4], [P1, P5], [P2] | 10,11,0 | [P2] | [P1, P4] (21) |
| 3 | [P1, P5], [P2, P3] | 11,3 | [P2, P3] | [P1, P4] (21) |
| 4 | [P1, P5], [P2, P3, P6] | 11,12 | [P1, P5] | [P2, P3, P6] (18) |
| 5 | [P1, P5, P6] | 20 | | [P2, P3, P6] (18) |

Table 5.2: Execution trace of the MFPG Shortest Path Algorithm

converted into a new graph that has 6 nodes and 5 edges. The problem is now converted to finding the shortest path in the MFPG between the set of $MFPs$ $P1, P2$ and $P4, P6$.

The execution trace of the MFPG Shortest Path Algorithm is shown in Table 5.2. In Line 1 of the algorithm, the candidate paths is initialized to $P1$ and $P2$, the two $MFP$ on which the origin lies. Since both these have the same cost we can expand either one. $P1$ is expanded by adding a $MFP$ to it giving us $[P1, P4]$ and $[P1, P5]$. $[P1, P4]$ already reaches the origin, so the cost along it is estimated to be 21 and it is removed from the set of candidate paths. At every step the process continues extending the most promising paths one $MFP$ at a time. In step 4 we see another path $[P2, P3, P6]$ reaches the destination, so its cost is estimated and found to be 18. Since this cost is lower than the current minimum of 21, the result cost and path is updated to reflect this. Finally at step 5, the stop condition is met wherein there are no paths in the set of candidate paths with a cost less than 18. So the Algorithm terminates and returns the path with a cost of 18.

When compared to the execution trace from Table 5.1, we can see that the number of iterations taken by the MFPG Shortest Path Algorithm is significantly lesser than that of the baseline method. The MFPG Shortest Path Algorithm takes 5 iterations to find the path compared to the 12 iterations taken by the baseline method.

### 5.2.1 Analysis of proposed approach

#### Completeness

The completeness of the algorithm discussed above is that provided there exists a path between the origin and destination that is either a $FP$ or a $UFP$, the algorithm will find a path between the origin and destination.

If we were to explore all $FPs$ and $UFPs$ then the algorithm would be complete.

However the proposed algorithm we explore all sub-paths of $MFPs$ and their unions, instead of exploring $FPs$ and $UFPs$. But since the definition of a $MFP$ is a $FP$ that is not the sub-path of any other $FP$, we come up with the following lemma.

**Lemma 5.1.** *Any $FP$ is a sub-path of at least one $MFP$.*

*Proof.* The lemma can be proved by contradiction. Assume there is a $FP$ that is not the sub-path of any other $MFP$. If this is the case then it is a $MFP$ itself. Since it would be a sub-path of itself, this is a contradiction. If it is the sub-path of at least one $FP$ that is not a $MFP$ then, due to the transitive nature of the sub-path relation, there would be an infinite number of $FPs$ that are not $MFPs$. But this is contradicted due to the limited size of the road network and trajectories on it. □

**Lemma 5.2.** *Any $UFP$ formed by a union of a set of $FPs$ is a sub-path of a $UFP$ formed by a union of a set of $MFPs$.*

*Proof.* From the lemma above, for any $UFP$ formed by the union of a set of $FPs$, we can find a set of $MFPs$ each of which has a sub-path in the former set of $FPs$. The origin and destination of a $UFP$ id the origin of the first $FP$ and the destination of the last $FP$ that lie in the first and last $MFP$. □

Hence, by exploring $UFPs$ formed by $MFPs$ in the algorithm, we explore all $FPs$ and $UFPs$ in the spatial graph.

### Correctness

The correctness of the proposed algorithm means that, provided there exists a path between the origin and destination that is either a $FP$ or a $UFP$, the path returned by the algorithm from the origin to the destination would be a $FP$ or $UFP$ with the minimum estimated cost.

The correctness of the algorithm can be proved by contradiction. Assume there is a path $P^*$ with a cost lower than the cost of the path returned by the algorithm, $P$. If $P^*$, was found by the algorithm when the algorithm terminates, then the cost of $P$ would have been lower, since the algorithm returns the path with the lowest cost. If however $P^*$, is not found. Then since the algorithm explores all $FPs$ and $UFPs$, and $P^*$ must be a $FP$ or a $UFP$, then there must exist a sub-path $\bar{P}^*$ of $P^*$ that is a candidate path when the algorithm terminates. Since the costs on all paths is assumed to be positive, the cost of $\bar{P}^*$ must be less than $P^*$ and $P$. But this is contradicted by the stop criteria of the algorithm that there is no candidate path with a cost less than the cost of the path found.

**Complexity**

The complexity of the path selection algorithm depends linearly on the number of nodes ($|V|$), the number of edges ($|E|$), and the number of input trajectories ($T$) in the graph. The worst case time complexity of the general path selection is given by $O(|E||V|T)$.

The number of spatial nodes and spatial edges are $|V_{SG}|$ and $|E_{SG}|$, and the number of MFP-Nodes and MFP-Edges are $|V_{MFP}|$ and $|E_{MFP}|$. By choosing an appropriate value of $\beta$, the size of the MFPG could be smaller than that of the spatial graph. In such situations we have $|E_{MFP}| < |E_{SG}|$ and $|V_{MFP}| < |V_{SG}|$. In such cases we would see that $O(|E_{MFP}||V_{MFP}|T) < O(|E_{SG}||V_{SG}|T)$. Hence by reducing the size of the graph over which the path selection algorithm is run, the MFP algorithm speeds up the path selection process.

## 5.3 Informed MFPG Shortest Path Algorithm

The MFPG Shortest Path Algorithm described above, follows an uninformed search strategy to find the path from the origin to the destination. The uninformed search strategy does not make any use of the information we have about the destination to help in the search process. We can however speed up the search by making use of the information we have about the destination. We design an admissible heuristic that can guide the search space towards the destination and still guarantee to find the correct path.

### 5.3.1 Designing the admissible heuristic

To guarantee correctness of the algorithm, we need an admissible heuristic that can be used in the MFPG. The main challenge in doing so is that a MFP-Node is composed of several base nodes, as it is a $MFP$. So any heuristic in that is admissible in the MFPG must also be admissible in the spatial graph. Hence we take an admissible heuristic in the spatial graph as an input and use that to compute an admissible heuristic in the MFPG.

As previously mentioned, the $MFP$ in the MFPG, is made up of several base nodes. If an admissible heuristic can be easily calculated at the base nodes, then we can set for the MFP-Node, we could design its heuristic to be the minimum of all the base nodes that it contains. If the heuristic is admissible in the spatial graph, then using the minimum of these values would give us a heuristic that is admissible in the MFPG. However, as we noted earlier, a $MFP$ does not form a $UFP$ with other $MFPs$ at all

the base nodes in the spatial graph. We only need to consider the transfer spatial nodes at which two $MFPs$ form a $UFP$, since it is only at these outgoing MFP-Edges that one MFP-Node is connected to another. We can then modify our definition of the admissible heuristic in the MFPG to only consider the transfer spatial nodes. In most cases, since the number of transfer spatial nodes is much smaller compared to the total number of base nodes in the path, computing the admissible heuristic in the MFPG is not expensive.

Let a MFP-Node $P$ be made up of several base nodes in a set $N$. Let $h(n)$ be the admissible heuristic function of any base node ($n$). Hence the admissible heuristic for a MFP-Node $P$,

$$H(P) = \min \{\ h(n) \mid n \in N \text{ and is a transfer spatial node in } P\ \}$$
$$H(G) = 0 \mid G \text{ is a MFP-Node that is a Goal State}$$

If a MFP-Node $P$ does not contain any transfer spatial nodes and is not a goal state itself, then the heuristic value for that node would be $\infty$. This is because there would be no path from $P$ to the goal state.

For example, in Figure 4.8a, we have 4 $MFP$ in the MFPG, $P1$ to $P4$. Figure 4.8b shows the internal representation of the MFP-Node $P2$ which is made up of the base nodes $n1$ to $n7$. $n2$ and $n7$ are the transfer spatial nodes in $P2$ that connect to $P3$ and $P4$. If $h(n2)$ and $h(n7)$ are the admissible heuristics of the transfer spatial nodes $n$ and $n7$, then

$$H(P2) = \min\{\ h(n2), h(n7)\ \}$$

### 5.3.2 Informed MFPG Shortest Path Algorithm

Once we have an admissible heuristic for the MFPG, we can use an informed search method such as a modification of the A* algorithm to efficiently find the minimum cost path between the origin and destination.

In the A* algorithm, the cost function depends on both, the cost to reach a node, as well as the admissible heuristic, that determines the expected cost from the node to the destination. For our algorithm the cost function is $F(P) = G(P) + H(P)$. Here $G(P)$ would be the cost to reach the given MFP-Node $P$ from the origin. In our case this would be the value we get from the cost estimation model. $H(P)$ represents the heuristic at MFP-Node $P$, that tries to estimate the cost from $P$ to the destination, using the function from the previous section. Using $F(P)$ now would guide the search towards the destination.

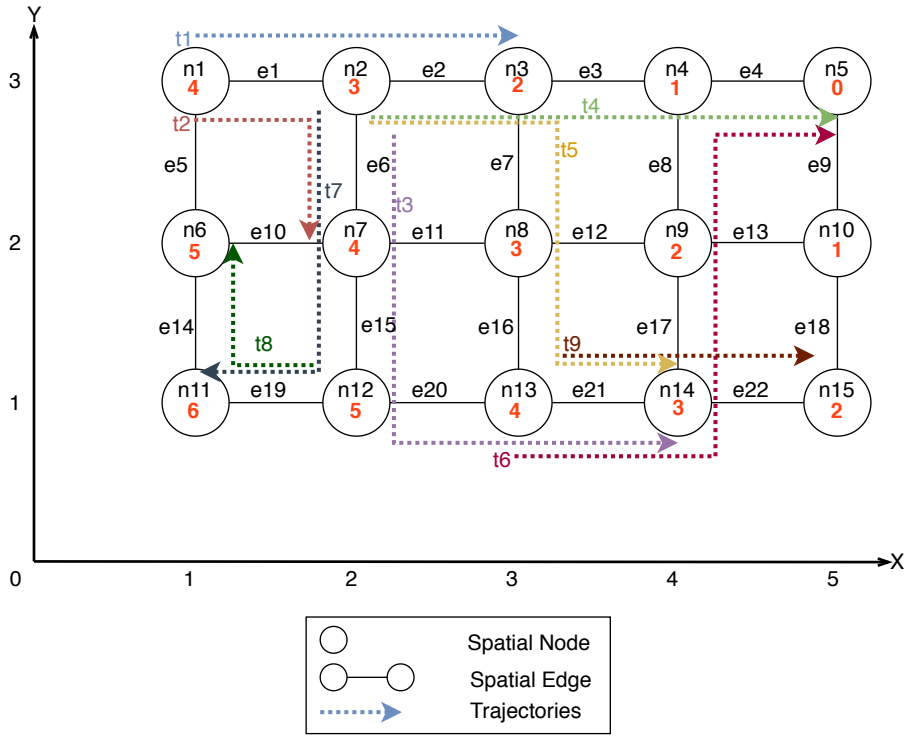



Figure 5.2: Additional trajectories added to spatial graph to demonstrate use of the admissible heuristic

This algorithm also follows the general algorithm framework provided in Algorithm 1. Here instead of using the path cost as used before, we would use the cost function $F(P)$ as described above. Also, a path is said to be found when the destination node is expanded, rather than when the destination node is first reached. Provided that the heuristic is admissible, the algorithm will find the correct solution.

To show how the admissible heuristic helps guide the search space towards the destination we modify the spatial graph from Figure 4.6 to include 3 more trajectories. Figure 5.2 shows the trajectories $t7$, $t8$, and $t9$, added in the network. Table 5.3 provides more details about the spatial edges traversed by the trajectories and the cost on each edge. Again if we consider $\beta$ to be 1, all of these trajectories would be $FPs$ and since neither of them is the sub-path of any other $FP$, all of them are $MFPs$.

Figure 5.3 shows the MFPG created for the spatial graph and trajectories in Figure 5.2. We can see that since at $n7$, $t2$ and $t7$ form a $UFP$, there exists an MFPG-Edge between them. Similarly MFPG-Edges exist between $t7$ and $t8$, $t3$ and $t9$, and $t5$ and $t9$.

If we want to find a path between $n1$ and $n5$ as before, we would need to identify the MFP-Nodes that the origin and destination lie on. Here too the origin lies on $P1$

| ID | Trajectory Records | | | | |
|---|---|---|---|---|---|
| t1 | Road Segment | e1 | e2 | | |
| | Cost | 2 | 9 | | |
| t2 | Road Segment | e1 | e6 | | |
| | Cost | 1 | 1 | | |
| t3 | Road Segment | e6 | e15 | e20 | e21 |
| | Cost | 3 | 2 | 4 | 3 |
| t4 | Road Segment | e2 | e3 | e4 | |
| | Cost | 7 | 9 | 2 | |
| t5 | Road Segment | e2 | e7 | e16 | e21 |
| | Cost | 9 | 2 | 2 | 7 |
| t6 | Road Segment | e21 | e17 | e8 | e4 |
| | Cost | 3 | 2 | 2 | 2 |
| t7 | Road Segment | e6 | e15 | e9 | |
| | Cost | 1 | 1 | 1 | |
| t8 | Road Segment | e19 | e14 | | |
| | Cost | 1 | 1 | | |
| t9 | Road Segment | e21 | e22 | | |
| | Cost | 1 | 1 | | |

Table 5.3: Additional trajectories added to spatial graph

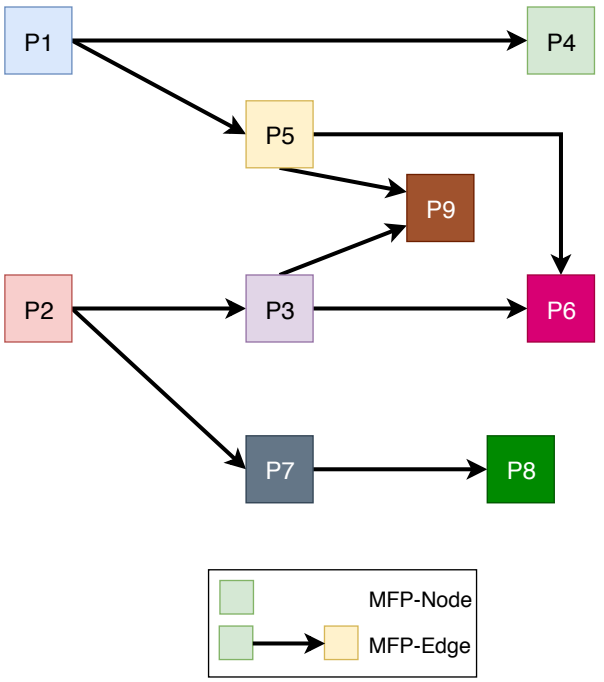



Figure 5.3: MFPG for the example in Figure 5.2

and $P2$, and the destination lies on $P4$ and $P6$. In this spatial graph, we can see that the cost on every edge is at least 1. So in this case for simplicity we can make use of the Manhattan Distance to be an admissible heuristic in the spatial graph. The Manhattan Distance between 2 points $P(x_i, y_i)$ and $Q(x_j, y_j)$ is $L_1(P,Q) = |x_i - x_j| + |y_i - y_j|$. In this case since the destination is $n5$, $h(n) = L_1(n, n5)$ In Figure 5.2, the numbers inside the spatial nodes indicate the heuristic at each of the spatial nodes. We can see that at the destination, $n5$ the heuristic is 0, and since the cost of traversing any base edge is at least 1, $h(n)$ is an underestimation at every base node. We calculate the heuristic at all the MFP-Nodes as shown in Table 5.4. The underlined MFP-Nodes $P4$, and $P6$ indicate they are the destination MFP-Nodes.

We now compare how the MFPG Shortest Path Algorithm and the Informed MFPG Shortest Path Algorithm will run in this scenario to demonstrate how the admissible heuristic helps the algorithm expand fewer nodes.

### MFPG Shortest Path Algorithm Execution Trace

Table 5.5 shows the execution trace of the MFPG Shortest Path Algorithm with the new trajectory data. In step 1, the candidate paths is initialized to $P1$ and $P2$, the two $MFP$ on which the origin lies. Since both these have the same cost we can expand either one. $P1$ is expanded by adding a $MFP$ to it giving us $[P1, P4]$ and $[P1, P5]$. $[P1, P4]$

| MFP-Node | Transfer Spatial Nodes | H(P) |
|---|---|---|
| P1 | n3 | 2 |
| P2 | n7 | 7 |
| P3 | n14 | 3 |
| P4 | - | 0 |
| P5 | n14 | 3 |
| P6 | - | 0 |
| P7 | n11 | 6 |
| P8 | - | $\infty$ |
| P9 | - | $\infty$ |

Table 5.4: Admissible Heuristic values calculated at MFP-Nodes

already reaches the origin, so the cost along it is estimated to be 21 and it is removed from the set of candidate paths. At every step the process continues extending the most promising paths one $MFP$ at a time. In step 2, $P2$ is expanded and $[P2, P3]$ and $[P2, P7]$ are added to the candidate paths. Since $[P2, P7]$ has the lowest cost, it is expanded. Again in step 5, $[P2, P7, P8]$ has the lowest cost, so it is expanded, however since it does not form a $UFP$ with any other $MFP$ it does not add any new nodes to the set of candidate paths. A similar situation occurs in step 6, when $[P2, P3, P9]$ is expanded. A lot of MFP-Nodes are expanded in this method, some of which are moving away from the destination.

**Informed MFPG Shortest Path Algorithm Execution Trace**

We now look at the execution trace of the Informed MFPG Shortest Path Algorithm as shown in Table 5.6. Here, as described above the cost function is $F(P) = G(P) + H(P)$, where $G(P)$ is the value returned from the cost estimation model, and $H(P)$ is the value of the admissible heuristic.

In step 1, the candidate paths is initialized to $P1$ and $P2$, the two $MFP$ on which the origin lies. Since both these have the same cost we can expand either one. $P1$ is expanded by adding a $MFP$ to it giving us $[P1, P4]$ and $[P1, P5]$. $[P1, P4]$ already reaches the origin, so the cost along it is estimated to be 21 and it is removed from the set of candidate paths. At every step the process continues extending the most promising paths

| Step | *CP* | Cost | *p* | Result Path (Cost) |
|---|---|---|---|---|
| 1 | [P1],[P2] | 0,0 | [P1] | |
| 2 | [P1, P4], [P1, P5],[P2] | 10,11,0 | [P2] | [P1, P4] (21) |
| 3 | [P1, P5], [P2, P3], [P2, P7] | 11,3,2 | [P2, P7] | [P1, P4] (21) |
| 4 | [P1, P5], [P2, P3], [P2, P7, P8] | 11,3,4 | [P2, P3] | [P1, P4] (21) |
| 5 | [P1, P5], [P2, P3, P6], [P2, P3, P9], [P2, P7, P8] | 11,12,11,4 | [P2, P7, P8] | [P2, P3, P6] (18) |
| 6 | [P1, P5], [P2, P3, P9] | 11,11 | [P2, P3, P9] | [P2, P3, P6] (18) |
| 7 | [P1, P5] | 11 | [P1, P5] | [P2, P3, P6] (18) |
| 8 | [P1, P5, P6], [P1, P5, P9] | 20,19 | | [P2, P3, P6] (18) |

Table 5.5: Execution trace of the MFPG Shortest Path Algorithm with new trajectory data

one $MFP$ at a time. In step 2, $P2$ is expanded and $[P2, P3]$ and $[P2, P7]$ are added to the candidate paths. Now in the MFPG Algorithm $[P2, P7]$ would have been expanded here since it has the lowest cost. However using the admissible heuristic we see that $[P2, P7]$ does not take us closer to the destination. The high value oh $H(P)$ prevents this node from being expanded here. $[P2, P3]$ is expanded instead. Similarly the MFPG algorithm expanded nodes $[P2, P7, P8]$ and $[P2, P3, P9]$ that will not be expanded by this method, because they do not lead to the origin. Hence we can see that the use of the admissible heuristic minimizes the number of iterations it takes for the algorithm, to find the path between $n1$ and $n5$ from 8 iterations to 6 iterations. In larger networks where the origin and destination are further apart this results in a far fewer nodes being expanded by the Informed MFPG Algorithm.

### 5.3.3 Analysis of proposed approach

#### Completeness and Correctness

We now prove the correctness and completeness of our heuristic method approach. We know that A* is complete and correct if the heuristic used is admissible. So we need

| Step | *CP* | G(P) | H(P) | F(P) | *p* | Result Path(Cost) |
|---|---|---|---|---|---|---|
| 1 | [P1], [P2] | 0,0 | 2,4 | 2,4 | [P1] | |
| 2 | [P1,P4], [P1,P5], [P2] | 10, 11, 0 | 0, 3, 4 | 10, 14, 4 | [P2] | [P1,P4](21) |
| 3 | [P1,P5], [P2,P3], [P2,P7] | 11, 3, 2 | 3, 3, 6 | 14, 6, 8 | [P2,P3] | [P1,P4](21) |
| 4 | [P1,P5], [P2,P3,P6], [P2,P3,P9], [P2,P7] | 11, 12, 11, 2 | 3, 0, ∞, 6 | 14, 12, ∞, 8 | [P2,P7] | [P2,P3,P6](18) |
| 5 | [P1,P5], [P2,P3,P9], [P2,P7,P8] | 11, 11, 4 | 3, ∞, ∞ | 14, ∞, ∞ | [P1,P5] | [P2,P3,P6](18) |
| 6 | [P1,P5,P6], [P1,P5,P9], [P2,P3,P9], [P2,P7,P8] | 20, 19, 11, 14 | 0, ∞, ∞, ∞ | 20, ∞, ∞, ∞ | | [P2,P3,P6](18) |

Table 5.6: Execution trace of the Informed MFPG Shortest Path Algorithm

to prove that the heuristic function used here, $H(P)$ is admissible. We assume here that the heuristic used in the spatial graph, $h(n)$ is admissible.

**Lemma 5.3.** *For any spatial node n in the spatial graph, if $h(n)$ is an admissible heuristic, then $H(P)$ is an admissible heuristic in the MFP-Node P that contains n.*

*Proof.* We have,

$$H(P) = \min \{\ h(n) \mid n \in N \text{ and is a transfer spatial node in } P\ \}$$
$$H(G) = 0 \mid G \text{ is a MFP-Node that is a Goal State}$$

At the goal MFP-Node, $G$ the H(G) is 0. For every other MFP-Node $P$, we have $H(P)$ = min $\{\ h(n) \mid n \in N$ and is a transfer spatial node in $P\ \}$. Let the cost from $P$ to $G$ be $c$, and let $n^*$ be the transfer spatial node in $P$ with the minimum $h(n)$. So $H(P) = h(n^*)$. If $c \geq H(P)$, then the heuristic is admissible. If $c < H(P)$ then we have 2 cases. If the minimum cost path from $P$ to $G$ passes through $n^*$, then $h(n^*)$ would have to be greater than $c$, which would be a contradiction, as $h(n^*)$ is admissible. If however the minimum cost path from $P$ to $G$ does not pass through $n^*$. Let this path pass through another node $n'$. Then since $h(n')$ is admissible, $h(n') < c$. So $h(n') < h(n^*)$. Since the path passes through $n'$, it is a transfer spatial node with the minimum heuristic. But this is again a contradiction as $n^*$ was the transfer spatial node in $P$ with the minimum heuristic. □

### Complexity

The complexity of the Informed MFPG Shortest Path Algorithm is the same as that of the MFPG Shortest Path Algorithm discussed earlier, and is $O(|E_{MFP}||V_{MFP}|T)$.

# Chapter 6

# Experiments and Evaluation

In this Chapter we discuss the experimental evaluation conducted to validate the previously discussed algorithms. The experiments were conducted on both synthetically generated data, as well as real trajectory data from 3 UPS delivery trucks in Fort Worth, TX.

## 6.1 Experiment Setting

### 6.1.1 Experiment Goals

We want to compare how the algorithms discussed in the previous Chapter compare to the baseline approach used in [24]. We also want to study how the algorithms behave on changing the input parameters such as the minimum number of trajectories on a $FP(\beta)$, the number of input trajectories $(T)$ , and the number of road segments on the lowest cost path. We wish to answer the following questions as a result of our experiments:

**Comparative analysis:**

- How do the different methods compare to the baseline approach?

**Sensitivity analysis:**

- How are the proposed methods affected by the number of input trajectories $(T)$ ?
- How are the proposed methods affected by the minimum number of trajectories along a $FP(\beta)$ ?
- How are the proposed methods affected by the length of the result path ?

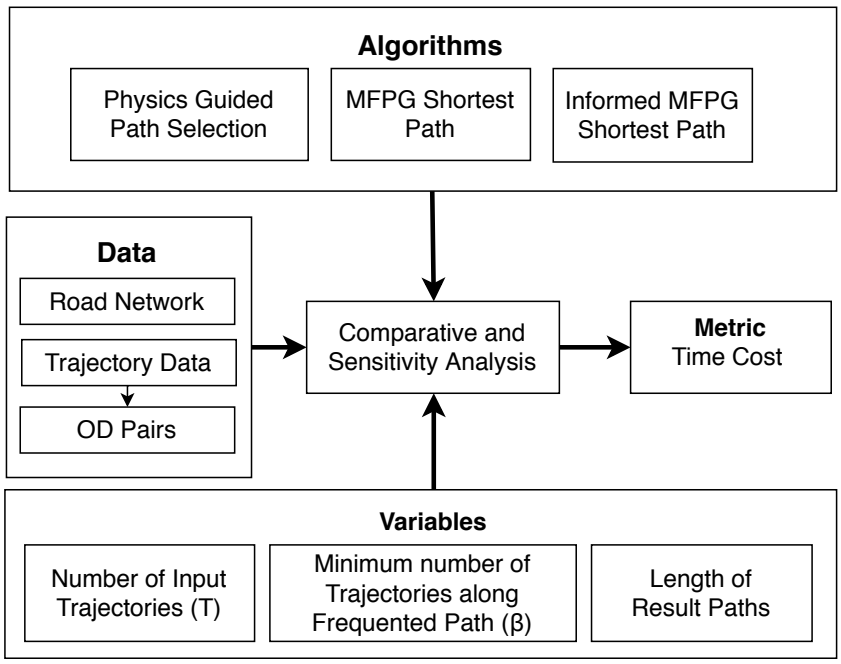



Figure 6.1: Experiment Design

### 6.1.2 Candidate methods and cost metric

We want to compare the results of the proposed methods with the method used in [24]. We will consider that to be the baseline method. To keep other factors from affecting the comparison between the methods, we use the same cost estimation model that was used there. The algorithms compared are: Physics Guided Path Selection [24], MFPG Shortest Path Algorithm, and Informed MFPG Shortest Path Algorithm. The algorithms were compared on their execution time to determine efficiency.

### 6.1.3 Experiment environment

The experiments were run on a machine with Intel(R) Core($^{TM}$) 15-7500 CPU @ 3.40GHz and 64GB memory. The operating system used was Windows 10. The algorithms were implemented using C# (.NET Framework 4.7).

The overall experiment design is shown in Figure 6.1. On the top we see the algorithms that we used for comparison. The left part of the diagram shows the data being used, and the bottom shows the variables that are analysis.

### 6.1.4 Data Set

#### UPS Truck Data Set

Experiments were conducted on a real trajectory dataset with 920 vehicle trajectories from 3 UPS trucks in Fort Worth Texas between 1/1/2017 and 6/30/2019. Each trajectory is the trajectory of a single truck for an entire day. We get time, location and 250 other attributes including status of vehicles power-train system (e.g., energy consumption and stop count) in each trajectory record. The stop count is used to split the entire days trajectories into sub trajectories between one origin and destination (2 delivery stops). This gives us information about the routing preferences on that route and the engine's performance on it. A map matching algorithm from [27] is used to align the trajectory data with a digital map from Open Street Maps. The map from Open Street Maps has 9084 road segments and 6193 intersections.

After splitting up the trajectories based on the stop count we have 10129 sub-trajectories with an average length of 54 road segments. Of the 9084 road segments in the network, 3837 road segments are traversed by at least 1 trajectory. Figure 6.2 shows the map of Fort Worth, TX with the trajectory data. The orange lines show the trajectory data on the map. The darker shades of orange show those paths that have a higher number of trajectories along them.

Making use of the sub trajectories, we use their origins and destinations to get 10129 Origin-Destination (OD) pairs. In Figure 6.2 the origins are shown as circles and the destinations are indicated with triangles. These represent real trips because they are the actual paths travelled by users on delivery trips.

Since we need to test the effect of number of trajectories on performance, we create 2 additional subsets consisting of 75% and 50% of the total trajectories in the original dataset. The first one has 7597 trajectories while the other has 5065 trajectories. The subsets are created using random sampling without replacement.

#### Synthetic Data Set

In this case the spatial graph consists of 5929 spatial nodes, and 9560 spatial edges between these nodes. Each spatial node is a point in geographical space and has an associated latitude and longitude. The nodes are placed in a grid pattern with edges between two nodes based on a probability.

We add 15000 trajectories to the base spatial graph, that are randomly created. The cost associated on each edge here is the travel time on each edge. The speed of the vehicle is assumed to be between 30 and 50 miles per hour. Left turns are given a

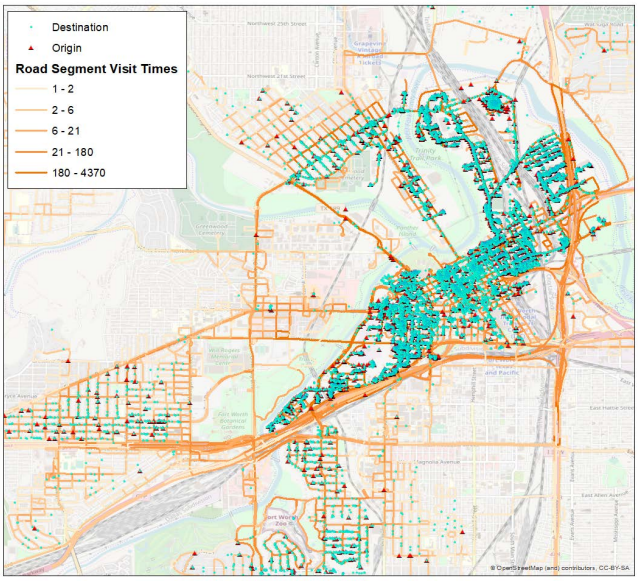



Figure 6.2: Trajectories and Origin-Destination pairs from UPS trucks used for experiment

higher time penalty compared to right turns, to reflect the actual road conditions. Out of the 9560 spatial edges in the graph, 3968 are traversed by at least 1 trajectory.

The origin-destination pairs are generated by using all the 15000 trajectories. In addition to these 10000 additional origin destination pairs are randomly created from within the spatial network.

In order to test the effect of the number of trajectories on performance, the two additional subsets created consist of 7500 trajectories, and 11250 trajectories, created using random sampling without replacement.

### 6.1.5 Admissible Heuristic

#### UPS Truck Data Set

In the experiments, we consider energy consumption as the cost measure. The amount of energy consumed by an electric vehicle on a network is a function of the friction coefficient, air resistance, mass, area and velocity of the vehicle, as well as the length of the path traversed, as shown in [24, 2, 33, 21]. In [24], they model the energy consumption of a vehicle moving along a path using a lower order physics model as:

$$Work = \frac{t}{\eta}(mav + c_{rr}mgv + \frac{1}{2}c_{air}A\rho v^3)$$

| Symbol | Physical Interpretation |
|---|---|
| $t$ | time |
| $\eta$ | vehicle's power-train system efficiency |
| $m$ | vehicle's mass |
| $a$ | acceleration |
| $v$ | velocity |
| $c_{rr}$ | rolling resistance constant |
| $g$ | acceleration due to gravity |
| $c_{air}$ | air resistance constant |
| $A$ | vehicle's front surface area |
| $\rho$ | air density |
| $l(n,G)$ | Euclidean distance between n and G |

Table 6.1: Physical interpretations of symbols used in vehicle energy consumption model

The physical interpretation of the symbols used is shown in Table 6.1. Here the term $mav$ corresponds to the energy consumption due to acceleration, $c_{rr}mgv$ corresponds to the energy consumption due to rolling friction, and $\frac{1}{2}c_{air}A\rho v^3$ corresponds to the energy consumption due to air resistance. Since we are only looking for an underestimation of the actual energy consumption, and we do not consider energy regeneration, we can assume the acceleration ($a$) to be 0. We can also assume the efficiency of the power-train system ($\eta$) to be 1. Now the admissible heuristic we use in the spatial graph is $h(n) = c_{rr}mgl(n,G) + \frac{1}{2}c_{air}A\rho v^2 l(n,G)$. To ensure that this heuristic is admissible, we make use of the lowest values of $c_{rr}$, $c_{air}$, and $\rho$ from literature. This would ensure that the heuristic would always underestimate the cost from $n$ to the goal($G$). Also since $l(n,G)$ is the Euclidean distance between $n$ and the goal, any path between $n$ and $G$ would be at least this value. Hence the heuristic is admissible. In the experiments below the values used were $m = 3000kg$, $g = 9.8m/s^2$, $A = 2m^2$, $s = 20miles/hour$, $\rho = 1.14kg/m^3$, $c_{air} = 0.4$ [2], $c_{rr} = 0.4$ [20].

**Synthetic Data Set**

In the synthetic data set we consider the time cost to be the cost measure. In this case calculating an admissible heuristic is easier. Since we know the maximum speed of a

vehicle in the graph to be 50 miles per hour, we can make use of this to come up with an admissible heuristic. We know that $time = distance/speed$. Since we only want an underestimation of the time taken by a vehicle in the graph, we can use the Euclidean distance between a given spatial node and the destination in the computations. Out admissible heuristic in the spatial graph now becomes $h(n) = l(n,G)/50$. Since there cannot be any path between the spatial node and the destination with a distance less than the Euclidean distance and the vehicle cannot travel at a speed faster than 50 miles per hour in the graph, this is an admissible heuristic in the spatial graph.

## 6.2 Experiment Results

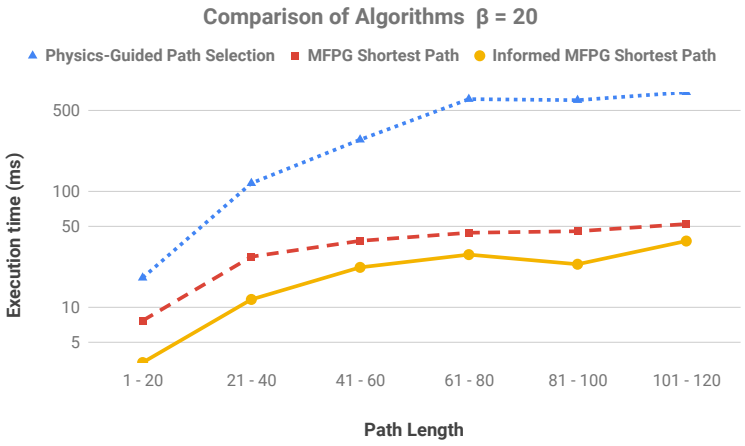
**Comparative Analysis:**

*How do the different methods compare to the baseline approach?*

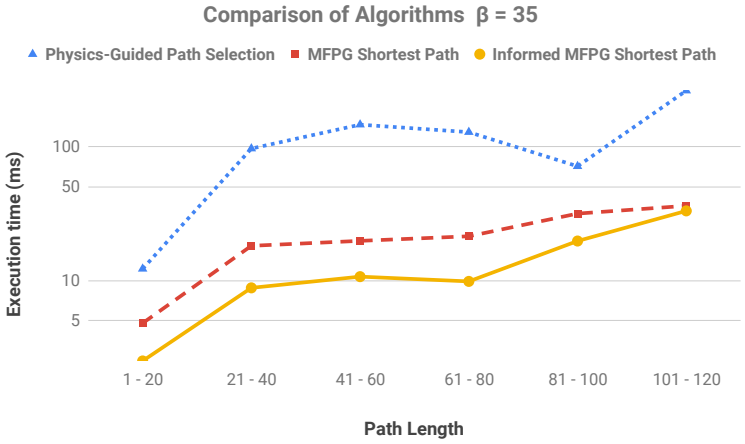
As we have seen, the number of $FPs$ and $UFPs$ depends upon the minimum number of trajectories along a $FP(\beta)$, and also the total number of input trajectories. The number of road segments in the lowest-cost path in the result also affects the number of iterations each algorithm takes to find the path. Hence to compare these algorithms, we use various values of $\beta$ (20, 35, 50 for the UPS truck data, and 25, 50, 75 for the synthetic data) and compare their performance on different result path lengths.

Figure 6.3 and Figure 6.4 shows the execution time in milliseconds of the various algorithms. In the figure, the vertical axis depicts the execution time in milliseconds, while the horizontal axis shows the result path length. The dotted line (blue triangles) show the execution time for the Physics-Guided Path selection from [24], the baseline approach, the dashed line (red squares) show the execution time of the MFPG Shortest Path Algorithm, and the solid line (yellow circles) show the execution time for the Informed MFPG Shortest Path Algorithm. Figure 6.3 compares the run times of the algorithms on the UPS truck data set, for $\beta$ = 20, 35, and 50 while Figure 6.4 compares the run times of the algorithms on the synthetic data set, for $\beta$ = 25, 50, and 75.

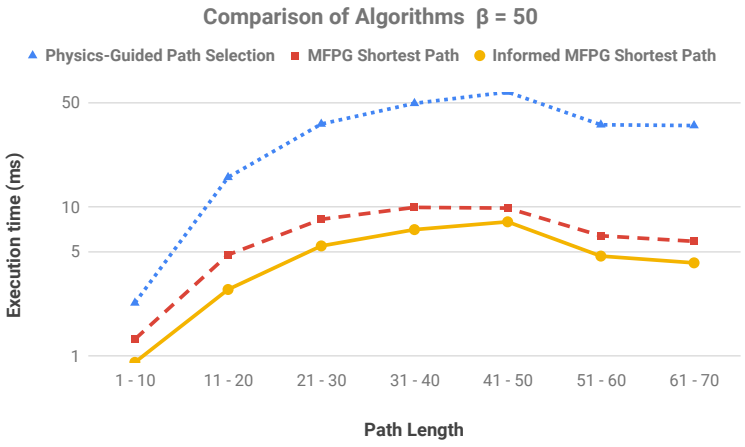
We can see that in all cases the MFPG Shortest Path Algorithm and the Informed MFPG Shortest Path Algorithm are faster than the baseline method. The difference in their performance is more evident as the path length increases. This is because as the path length increases, the baseline method would require more iterations to traverse a given path as compared to the MFPG Shortest Path Algorithm and the Informed MFPG Shortest Path Algorithms, which would extend paths by appending paths rather than edges. Also, we see that using the admissible heuristic, reduces the time taken by the




(a) $\beta = 20$

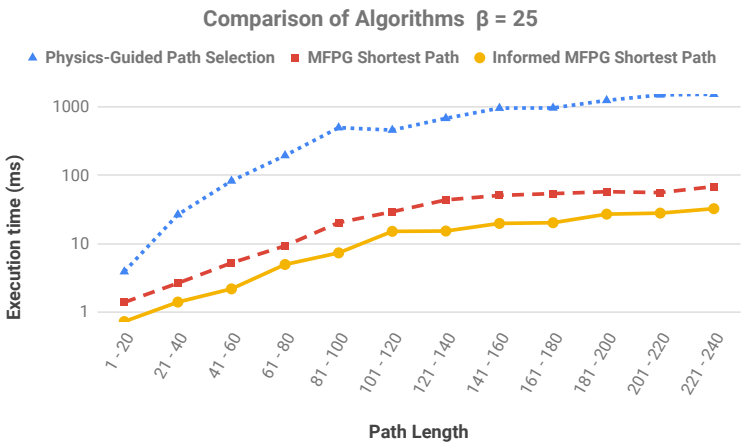



(b) $\beta = 35$

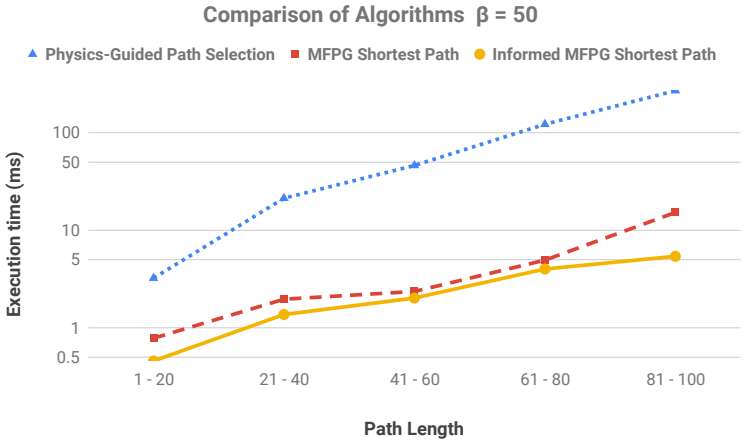



(c) $\beta = 50$

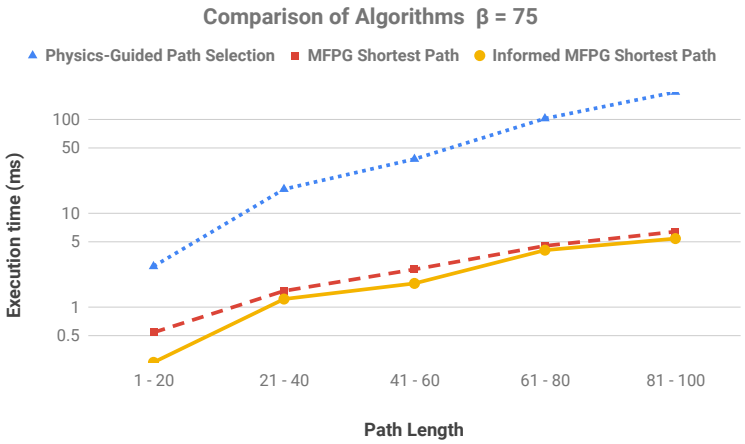
Figure 6.3: Comparison of execution time of algorithms for different values $\beta$, using 10129 trajectories, using UPS truck data




(a) $\beta = 25$




(b) $\beta = 50$




(c) $\beta = 75$

Figure 6.4: Comparison of execution time of algorithms for different values $\beta$, using 15000 trajectories, using synthetic data

| Number of Trajectories | 5065 | 7597 | 10129 |
|---|---|---|---|
| Number of Road Segments | 864 | 961 | 1112 |

(a) Number of road segments with more than 20 trajectories along it, using UPS truck data

| Number of Trajectories | 7500 | 11250 | 15000 |
|---|---|---|---|
| Number of Road Segments | 3029 | 3388 | 3538 |

(b) Number of trajectories with more than 25 trajectories along it, using synthetic data

Table 6.2: Effect of number of input trajectories on distribution of trajectories in road network

| Number of Trajectories | 5065 | 7597 | 10129 |
|---|---|---|---|
| % of reachable OD pairs | 13.4% | 26.3% | 40.1% |

(a) Percentage of origin-destination pairs reachable by varying values of $T$, using UPS truck data ($\beta = 35$)

| Number of Trajectories | 7500 | 11250 | 15000 |
|---|---|---|---|
| % of reachable OD pairs | 10.3% | 38.3% | 46.1% |

(b) Percentage of origin-destination pairs reachable by varying values of $T$, using synthetic data ($\beta = 50$)

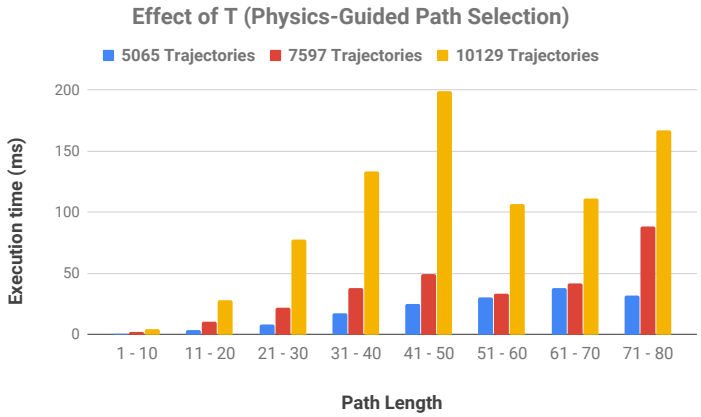
Table 6.3: Percentage of origin-destination pairs that are reachable by varying values of $T$

Informed MFPG Shortest Path Algorithm, as the search space is guided towards the destination.

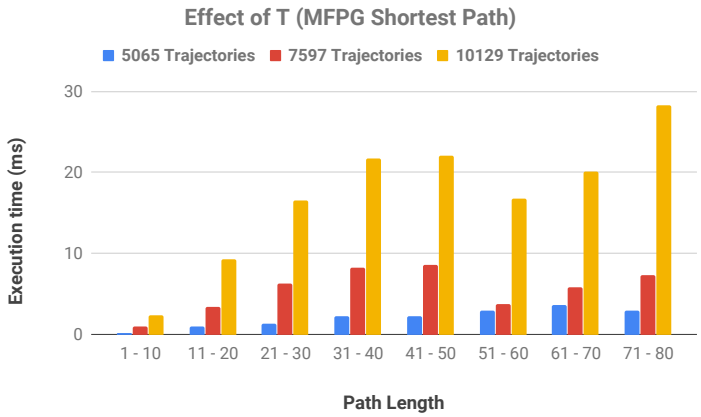
**Sensitivity Analysis:**

*How are the proposed methods affected by the number of input trajectories $(T)$?*

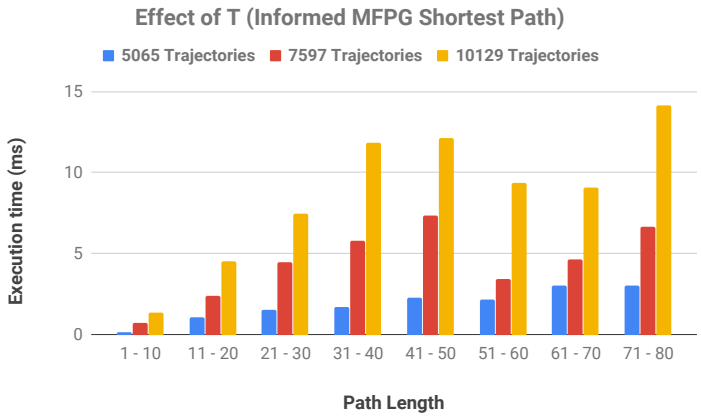
Increasing the number of input trajectories $(T)$ , increases the computational time of the algorithms. With a larger number of trajectories, there would be more paths in the spatial graph that have a large number of trajectories along it increasing the number of $FPs$ and $UFPs$. This increases the search space of the path selection algorithms. To analyze the effect of $T$ on the different algorithms, we make use of 100%, 75%, and 50% of the trajectories in the dataset. The trajectories are selected by random sampling without replacement. Table 6.2 shows how the number of input trajectories $(T)$ affects the number of road segments having a given number of trajectories moving along it.




(a) Physics Guided Path Selection

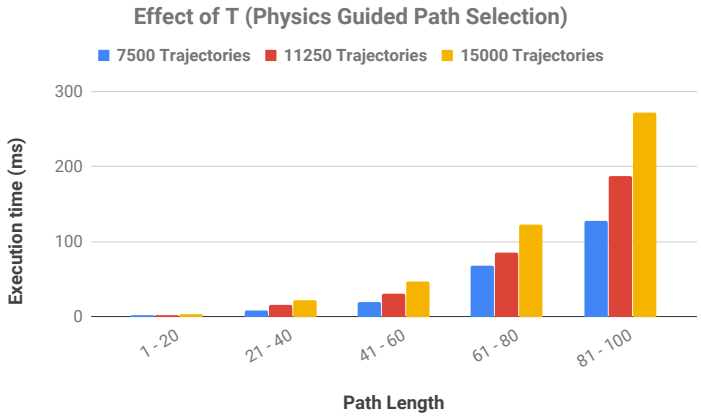



(b) MFPG Shortest Path Algorithm

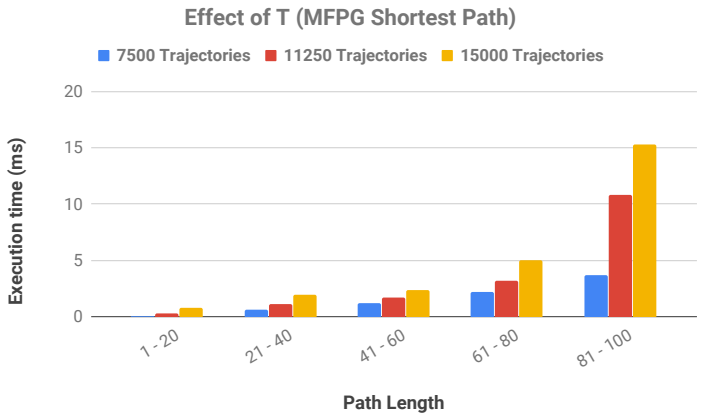



(c) Informed MFPG Shortest Path Algorithm

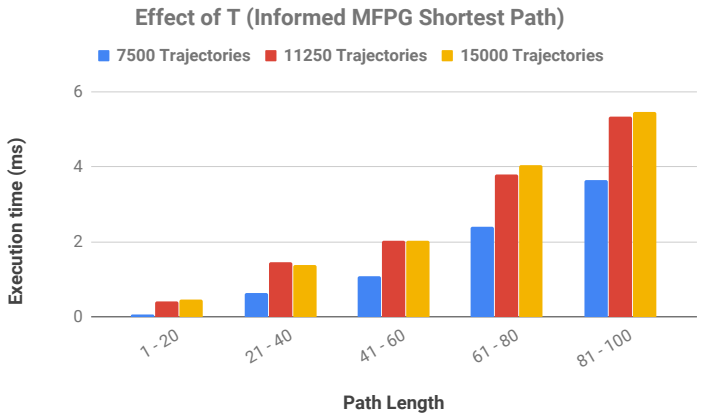
Figure 6.5: Effect of $T$ on execution time, $\beta = 35$, using UPS truck data




(a) Physics Guided Path Selection




(b) MFPG Shortest Path Algorithm




(c) Informed MFPG Shortest Path Algorithm

Figure 6.6: Effect of $T$ on execution time, $\beta = 50$, using synthetic data

Table 6.2a shows the number of road segments with more than 20 trajectories, for $T$ = 5565, 7597, and 10129 using the UPS data. Similarly, Table 6.2b shows the number of road segments with more than 25 trajectories, for $T$ = 7500, 11250, and 15000 using the synthetic data. In both cases we can see as $T$ increases, there would be more road segments having large number of trajectories along them.

Figure 6.5 and Figure 6.6 shows the effect of $T$ on the computational time for all the algorithms. In the figure, the vertical axis depicts the execution time in milliseconds, while the horizontal axis shows the result path length. The first bar (blue), shows the execution time when using 50% of the input trajectories. In case of the UPS dataset, this is 5065 trajectories, while in case of the synthetic dataset, this is 7500 trajectories. The second bar (red), shows the execution time when using 75% of the input trajectories. In case of the UPS dataset, this is 7597 trajectories, while in case of the synthetic dataset, this is 11250 trajectories. The last bar (yellow), shows the execution time when using 100% of the input trajectories. In case of the UPS dataset, this is 10129 trajectories, while in case of the synthetic dataset, this is 1500 trajectories.

From the bar graphs, it is clear that for all algorithms increasing the number of input trajectories ($T$), increases the execution time, as with a higher number of trajectories, the number of $FPs$ and $UFPs$ in the graph would increase, increasing the search space of the path selection algorithm. However, since the number of $FPs$ in the graph decreases with decreasing $T$, a lot of origin-destination pairs become unreachable. Table 6.3 shows the number of origin-destination pairs that are reachable for different values of $T$, using both the UPS and synthetic data sets. In all cases we can see that when the value of $T$ is low, the percentage of reachable origin-destination pairs is also lower.

*How are the proposed methods affected by the minimum number of trajectories along a $FP(\beta)$ ?*

The minimum number of trajectories along a $FP(\beta)$, determines how many paths in the spatial graph are $FPs$. Increasing the value of $\beta$ will decrease the number of $FPs$. Table 6.4 shows the number of road segments in the spatial graph with more than $\beta$ trajectories along it. Table 6.4a shows the number of road segments from the UPS truck data that have more than 20, 35, and 50 trajectories along them. Table 6.4b shows the number of road segments from the synthetic dataset that have more than 25, 50, and 75 trajectories along them. It is clear from the table table that as the value of $\beta$ increases, the number of road segments that have at least $\beta$ trajectories along them decrease. Hence increasing $\beta$ reduces the number of $FPs$. Since increasing $\beta$ decreases

| $\beta$ | 20 | 35 | 50 |
|---|---|---|---|
| Number of Road Segments | 1112 | 858 | 731 |

(a) UPS truck data using 10129 trajectories

| $\beta$ | 25 | 50 | 75 |
|---|---|---|---|
| Number of Road Segments | 3538 | 2988 | 2166 |

(b) Synthetic data set using 15000 trajectories

Table 6.4: Number of Road segments with more than $\beta$ trajectories along it

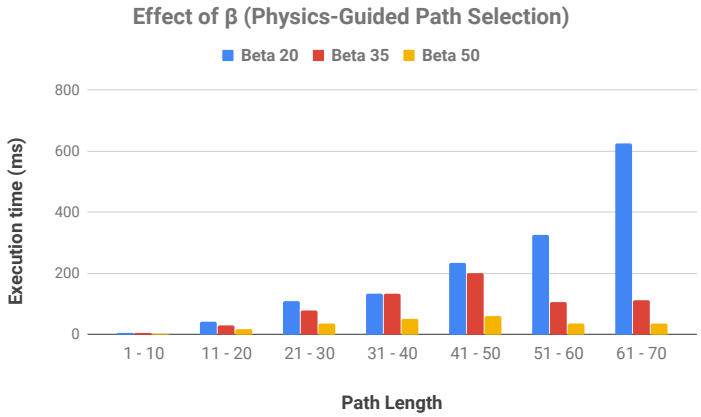
| $\beta$ | 20 | 35 | 50 |
|---|---|---|---|
| % of reachable OD pairs | 58.1% | 40.1% | 24.6% |

(a) Percentage of origin-destination pairs reachable by varying values of $\beta$, using UPS truck data ($T = 10129$)

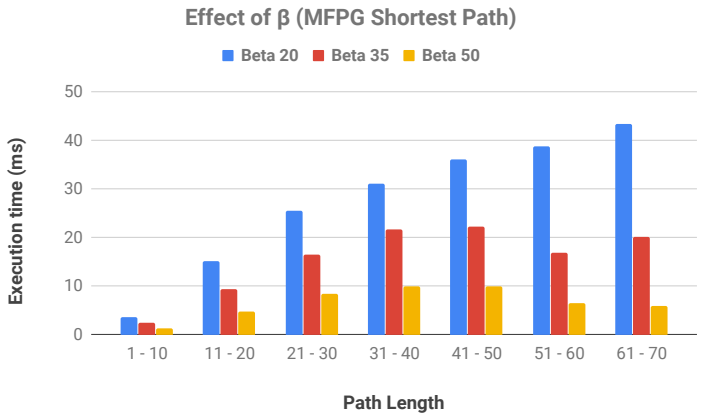
| $\beta$ | 25 | 50 | 75 |
|---|---|---|---|
| % of reachable OD pairs | 60.8% | 46.1% | 19.3% |

(b) Percentage of origin-destination pairs reachable by varying values of $\beta$, using synthetic data ($T = 15000$)

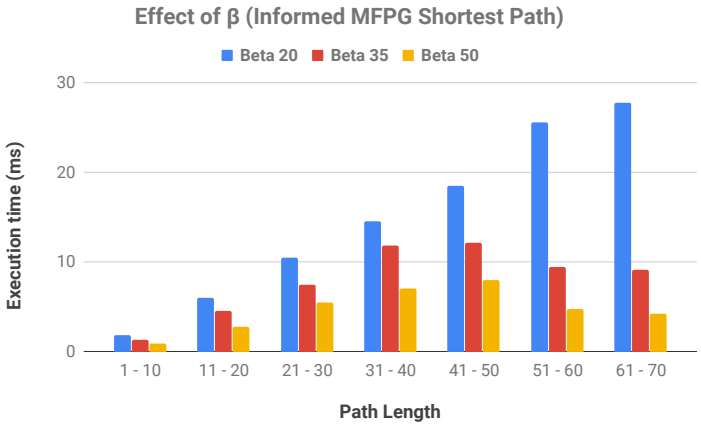
Table 6.5: Percentage of origin-destination pairs that are reachable by varying values of $\beta$




(a) Physics Guided Path Selection

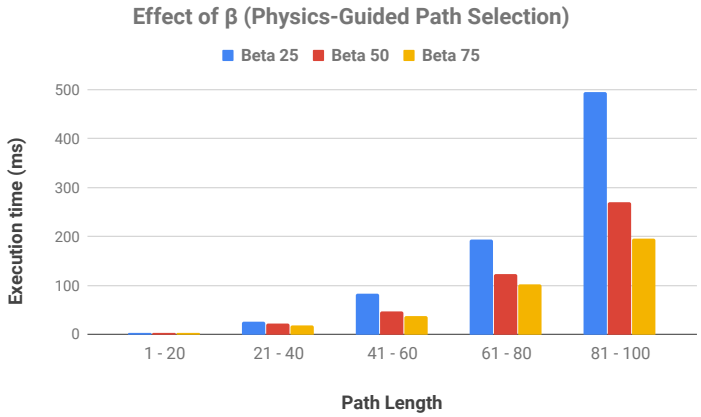



(b) MFPG Shortest Path Algorithm

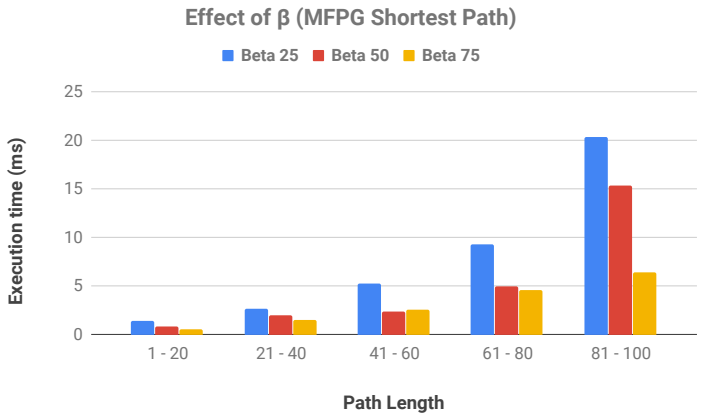



(c) Informed MFPG Shortest Path Algorithm

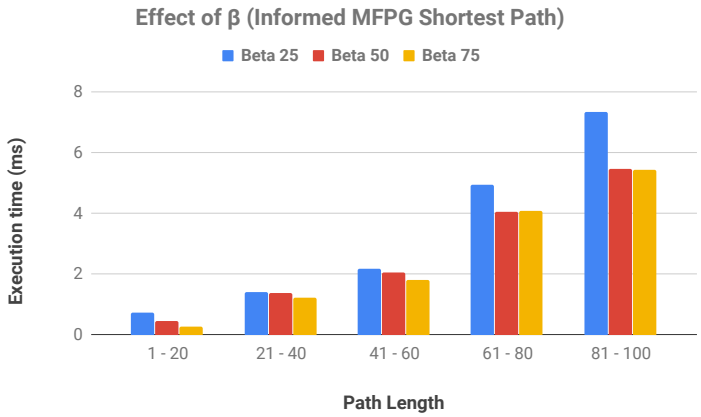
Figure 6.7: Effect of $\beta$ on execution time, $T = 10129$, using UPS truck data




(a) Physics Guided Path Selection




(b) MFPG Shortest Path Algorithm




(c) Informed MFPG Shortest Path Algorithm

Figure 6.8: Effect of $\beta$ on execution time, $T = 15000$, using synthetic data

the number of $FPs$, it reduces the search space and reduces computational time.

Figure 6.7 and Figure 6.8 shows the effect of $\beta$ on the algorithms. In the figure, the vertical axis depicts the execution time in milliseconds, while the horizontal axis shows the result path length.In Figure 6.7, the first bar (blue) shows the execution time of the algorithms when $\beta = 20$, the second bar (red) shows the execution time of the algorithms when $\beta = 35$, and the last bar (yellow) shows the execution time of the algorithms when $\beta = 50$. Similarly, in Figure 6.8, the first bar (blue) shows the execution time of the algorithms when $\beta = 25$, the second bar (red) shows the execution time of the algorithms when $\beta = 50$, and the last bar (yellow) shows the execution time of the algorithms when $\beta = 75$.

From the figures, it is clear that increasing the value of $\beta$ decreases the execution time of the algorithms, by reducing the number of $FPs$ in the graph that need to be explored. In case of the MFPG Shortest Path Algorithm and the Informed MFPG Shortest Path Algorithms, the value of $\beta$ is even more significant, as the size of the MFPG is directly related to the number of $FPs$. Hence choosing an appropriate value of $\beta$ is essential to make these algorithms run efficiently. In selecting the value of $\beta$, there is a trade off between the execution time and the number of reachable origin-destination pairs. Since the number of $FPs$ in the graph decreases with increasing $\beta$, a lot of origin-destination pairs become unreachable. Table 6.5 shows the number of origin-destination pairs that are reachable for different values of $\beta$, using both the UPS and synthetic data sets. We can see that when the value of $\beta$ is low, the percentage of reachable origin-destination pairs is also higher and decreases as the value of $\beta$ increases.

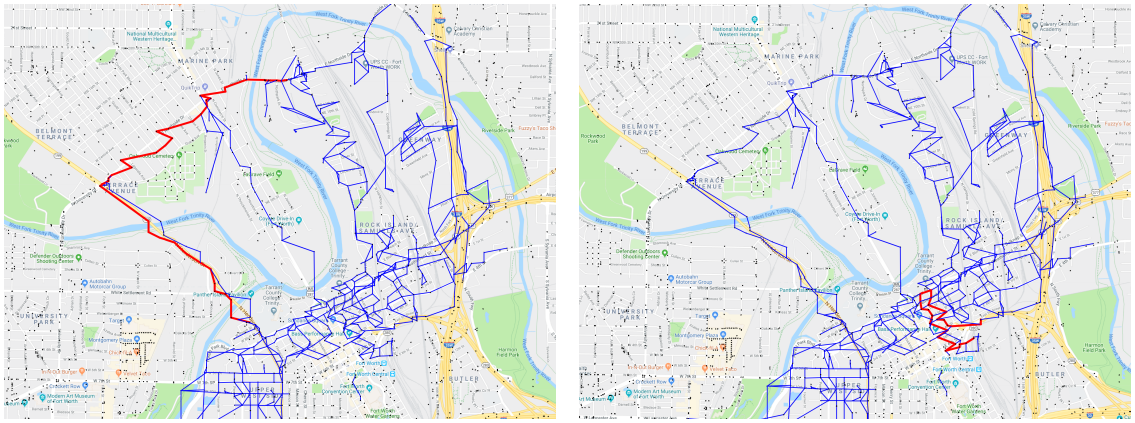
*How are the proposed methods affected by the length of the result path ?*

The length of the result path determines the number of iteration the path selection algorithm needs to go through to find a path. In most cases, with an increasing in the length of the result path, the execution time should increase.

In Figure 6.3 and Figure 6.4 we can see how the algorithms perform based on varying result path lengths. In the figures, the horizontal axis shows the length of the result paths, and the vertical axis shows the execution time in milliseconds.

Since we are using trajectory data, the trajectories might not be evenly distributed throughout the spatial graph. This leads to some longer paths having smaller execution times, as seen in the Figure 6.3b, between road segment length 81-100. This is due to some longer paths going along areas that have fewer $FPs$. This leads to a smaller number of nodes being expanded along these paths.

Figure 6.9 shows two such paths from the UPS truck data. The lines in blue indicate



(a) Path with length 88 road segments (b) Path with length 16 road segments

Figure 6.9: Paths in UPS truck data with abnormal computational time

the road segments that have at least 35 trajectories along them. These would be the paths that are $FPs$. The red lines indicate the path found for given origin-destination pairs.

As seen in the Figure, a lot of the $FPs$ are concentrated in a few regions in the road network. Figure 6.9a shows a path between a origin destination pair of path length 88. This path does not pass through a region of high $FPs$, hence it expands a relatively fewer number of nodes. The execution time for this path was only 45ms, and it expanded a total of 123 nodes. Figure 6.9b however shows a path between a origin destination pair that passes through a region of dense $FPs$. Although this has a path length of just 16, because it passes through so many $FPs$, it expands a large number of nodes. The execution time for this path is 246ms, and it expanded a total of 527 nodes.

# Chapter 7

# Conclusion and Future Work

With the increase in the volume and variety of trajectory data, we need newer routing algorithms that can leverage this rich data to answer interesting questions. The differences in the variety of of data means that rather than having one algorithm to answer all queries, we would need to have multiple specialized algorithms that can work together to answer all these queries. In this thesis we explored Trajectory based Routing using Frequented Paths. We proposed a new representation, "Maximal Frequented Path Graph (MFPG)" that combines the spatial graph and trajectory data into a single representation. We used introduced the MFPG Shortest Path Algorithm that can make use of the MFPG to find the lowest cost path between a origin-destination pair. To further speed up the algorithm, we make use of an admissible heuristic in the MFPG, that allows us to use A* like algorithms in the new representation. The Informed MFPG Shortest Path Algorithm makes use of this heuristic to further improve the computational efficiency of the method. We proved that this approach is both correct and complete. Extensive experiments were conducted using both real-world trajectory data as well as synthetically generated data to test the performance of the algorithms. We also show both experimentally and theoretically that this approach is computationally faster than the baseline approach used in [24].

## 7.1 Future Work

One of the main assumptions of our algorithm is that the path costs are always non-negative. We make use of this assumption to help in the termination of the MFPG Shortest Path algorithm, so that when all candidate paths have a cost greater than the cost of the lowest-cost path found, we can terminate. Negative costs are not uncommon in road networks. For example, if the cost to be minimized is energy consumption of

electric vehicles, then due to regenerative braking, there can be cases in which paths would have a negative energy consumption (the vehicle gains energy as it travels along a path). We would need to modify our algorithm to be able to handle this case as well.

Trajectory data is ever increasing, and in a given spatial graph, the number of trajectories would increase on a daily basis. In the current implementation of the algorithm, the entire MFPG would have to be constructed from scratch each time new trajectory data is received. Using an incremental approach, the MFPG could be modelled such that only the few edges and nodes that have updated need to be changed, rather than reconstructing the entire graph each time.

In the current implementation, the MFPG is created dynamically as the path selection algorithm runs. We could also explore the benefits of precomputing the graph to reduce computation time.

# References


[1] Sabeur Aridhi et al. “A MapReduce-based approach for shortest path problem in large-scale networks”. In: *Engineering Applications of Artificial Intelligence* 41 (2015), pp. 151–165.

[2] Andreas Artmeier et al. “The shortest path problem revisited: Optimal routing for electric vehicles”. In: *Annual conference on artificial intelligence*. Springer. 2010, pp. 309–316.

[3] Jørgen Bang-Jensen and Gregory Z Gutin. *Digraphs: theory, algorithms and applications*. Springer Science & Business Media, 2008.

[4] Favyen Bastani et al. “Machine-assisted map editing”. In: *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM. 2018, pp. 23–32.

[5] Richard Bellman. *On a routing problem*. 1958. URL: http://www.ams.org/journals/qam/1958-16-01/S0033-569X-1958-0102435-2/.

[6] *Big data: The next frontier for innovation, competition, and productivity*. URL: https://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/big-data-the-next-frontier-for-innovation.

[7] Siddhartha Sankar Biswas, Bashir Alam, and MN Doja. “Generalisation of Dijkstra’s algorithm for extraction of shortest paths in directed multigraphs”. In: *Journal of Computer Science* 9.3 (2013), pp. 377–382.

[8] *Cell phone sales worldwide 2007-2017*. URL: https://www.statista.com/statistics/263437/global-smartphone-sales-to-end-users-since-2007/.

[9] Yuche Chen et al. “Data-driven fuel consumption estimation: A multivariate adaptive regression spline approach”. In: *Transportation Research Part C: Emerging Technologies* 83 (2017), pp. 134–145.

[10] Jian Dai et al. “Personalized route recommendation using big trajectory data”. In: *2015 IEEE 31st International Conference on Data Engineering*. IEEE. 2015, pp. 543–554.

[11] Daniel Delling et al. “PHAST: Hardware-accelerated shortest path trees”. In: *Journal of Parallel and Distributed Computing* 73.7 (2013), pp. 940–952.

[12] Yong Deng et al. “Fuzzy Dijkstra algorithm for shortest path problem under uncertain environment”. In: *Applied Soft Computing* 12.3 (2012), pp. 1231–1237.

[13] Edsger W Dijkstra. “A note on two problems in connexion with graphs”. In: *Numerische mathematik* 1.1 (1959), pp. 269–271.

[14] Daniel Duque, Leonardo Lozano, and Andrés L Medaglia. “An exact method for the biobjective shortest path problem for large-scale road networks”. In: *European Journal of Operational Research* 242.3 (2015), pp. 788–797.

[15] Jochen Eisner, Stefan Funke, and Sabine Storandt. “Optimal Route Planning for Electric Vehicles in Large Networks”. In: *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*. AAAI’11. San Francisco, California: AAAI Press, 2011, pp. 1108–1113. URL: http://dl.acm.org/citation.cfm?id=2900423.2900599.

[16] *Energy Use for Transportation*. URL: https://www.eia.gov/energyexplained/?page=us_energy_transportation#tab1.

[17] Yuan Gao. “Shortest path problem with uncertain arc lengths”. In: *Computers & Mathematics with Applications* 62.6 (2011), pp. 2591–2600.

[18] P. E. Hart, N. J. Nilsson, and B. Raphael. “A Formal Basis for the Heuristic Determination of Minimum Cost Paths”. In: *IEEE Transactions on Systems Science and Cybernetics* 4.2 (1968), pp. 100–107. ISSN: 0536-1567. DOI: 10.1109/TSSC.1968.300136.

[19] Songtao He et al. “RoadRunner: improving the precision of road network inference from GPS trajectories”. In: *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM. 2018, pp. 3–12.

[20] Heinz Heisler. “14 - Vehicle body aerodynamics”. In: *Advanced Vehicle Technology (Second Edition)*. Ed. by Heinz Heisler. Second Edition. Oxford: Butterworth-Heinemann, 2002, pp. 584–634. ISBN: 978-0-7506-5131-8. DOI: https://doi.org/10.1016/B978-075065131-8/50015-4. URL: http://www.sciencedirect.com/science/article/pii/B9780750651318500154.

[21] Xianan Huang and Huei Peng. “Eco-routing based on a data driven fuel consumption model”. In: *arXiv preprint arXiv:1801.08602* (2018).

[22] Lebeaucarnews. *Traffic jams cost US $87 billion in lost productivity in 2018, and Boston and DC have the nation’s worst*. 2019. URL: https://www.cnbc.com/2019/02/11/americas-87-billion-traffic-jam-ranks-boston-and-dc-as-worst-in-us.html.

[23] Yaguang Li et al. “Multi-task representation learning for travel time estimation”. In: *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. ACM. 2018, pp. 1695–1704.

[24] Yan Li et al. “Physics-guided Energy-efficient Path Selection: A Summary of Results”. In: *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. SIGSPATIAL ’18. Seattle, Washington: ACM, 2018, pp. 99–108. ISBN: 978-1-4503-5889-7. DOI: 10.1145/3274895.3274933. URL: http://doi.acm.org/10.1145/3274895.3274933.

[25] Joel Lovell. *Left-Hand-Turn Elimination*. Dec. 2007. URL: https://www.nytimes.com/2007/12/09/magazine/09left-handturn.html.

[26] Wuman Luo et al. “Finding Time Period-based Most Frequent Path in Big Trajectory Data”. In: *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*. SIGMOD ’13. New York, New York, USA: ACM, 2013, pp. 713–724. ISBN: 978-1-4503-2037-5. DOI: 10.1145/2463676.2465287. URL: http://doi.acm.org/10.1145/2463676.2465287.

[27] Paul Newson and John Krumm. “Hidden Markov Map Matching Through Noise and Sparseness”. In: *Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. GIS ’09. Seattle, Washington: ACM, 2009, pp. 336–343. ISBN: 978-1-60558-649-6. DOI: 10.1145/1653771.1653818. URL: http://doi.acm.org/10.1145/1653771.1653818.

[28] Daniele Quercia, Rossano Schifanella, and Luca Maria Aiello. “The Shortest Path to Happiness: Recommending Beautiful, Quiet, and Happy Routes in the City”. In: *Proceedings of the 25th ACM Conference on Hypertext and Social Media*. HT ’14. Santiago, Chile: ACM, 2014, pp. 116–125. ISBN: 978-1-4503-2954-5. DOI: 10.1145/2631775.2631799. URL: http://doi.acm.org/10.1145/2631775.2631799.

[29] Christian Sommer. “Shortest-path Queries in Static Networks”. In: *ACM Comput. Surv.* 46.4 (Mar. 2014), 45:1–45:31. ISSN: 0360-0300. DOI: 10.1145/2530531. URL: http://doi.acm.org/10.1145/2530531.

[30] *U.S. Energy Information Administration - EIA - Independent Statistics and Analysis*. URL: https://www.eia.gov/todayinenergy/detail.php?id=32432.

[31] Dong Wang et al. “When will you arrive? estimating travel time based on deep neural networks”. In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

[32] Bin Yang et al. “PACE: A PAth-CEntric Paradigm for Stochastic Path Finding”. In: *The VLDB Journal* 27.2 (Apr. 2018), pp. 153–178. ISSN: 1066-8888. DOI: 10.1007/s00778-017-0491-4. URL: https://doi.org/10.1007/s00778-017-0491-4.

[33] L. Zhu et al. “Green routing fuel saving opportunity assessment: A case study using large-scale real-world travel data”. In: *2017 IEEE Intelligent Vehicles Symposium (IV)*. 2017, pp. 1242–1248. DOI: 10.1109/IVS.2017.7995882.