Markov Chain Based Algorithms for City-Scale Pollution-Aware Traffic Routing

Thesis submitted in partial fulfillment
of the requirements for the degree of

*Master of Science in **Computer Science and Engineering** by Research*

by

S. Shreevignesh
2018111019
sshreevignesh.s@research.iiit.ac.in

International Institute of Information Technology
Hyderabad - 500 032, INDIA
May 2023

International Institute of Information Technology
Hyderabad, India

CERTIFICATE

It is certified that the work contained in this thesis, titled "Markov Chain Based Algorithms for City-Scale Pollution-Aware Traffic Routing" by S. Shreevignesh, has been carried out under my supervision and is not submitted elsewhere for a degree.

_____ _____

Date Advisor: Dr. Praveen Paruchuri

_____

Co-Advisor: Dr. Girish Varma

To my family

Acknowledgments

I would like to express my sincere gratitude to all the people who supported me throughout my master's thesis. Firstly, I am deeply indebted to my advisor, Dr. Praveen Paruchuri, for giving me the opportunity to be a part of his group and introducing me to the fields of Artificial Intelligence and Intelligent Transportation. I am grateful for his invaluable guidance, encouragement and feedback. He has been a great mentor and a source of inspiration for me. I also thank my co-advisor, Dr. Girish Varma, for introducing me to the field of Markov Chain Monte Carlo algorithms and for his insightful comments and suggestions. Special thanks to Kohli Center on Intelligent Systems for their generous support.

I would like to acknowledge the support of my colleagues and friends at the Machine Learning Lab. They have provided me with a stimulating and friendly environment to conduct my research. I especially thank Fiza Husain and Sambhav Solanki for their friendship, collaboration, assistance and camaraderie.

I would also like to thank my friends for their friendship, moral support and constructive criticism. They have been a source of inspiration and motivation for me during this challenging journey. I am especially grateful to Adarsh, Aditya, Ashwin, Dolton, Giri, Joseph, Mallika, Sahil, Shiva, Shreeya, Siddharth, Srinath, Srivathsan, Sudhansh, and a few more. I am lucky to have such wonderful friends in my life.

Last but not least, I would like to dedicate this thesis to my family. Words cannot express how much I owe them for their unconditional love, support and sacrifice. They have always been there for me through thick and thin. I am especially grateful to my parents, Aparna and Suriyanarayanan, for their endless encouragement and faith in me. I would also like to thank my sister, Abirami, for her understanding and companionship.

Abstract

Air pollution is a growing concern across the world. Long-term exposure to severe pollution can cause serious health issues. Outdoor air pollution accounts for an estimated 4.2 million deaths per year, primarily due to stroke, heart disease, lung cancer, and chronic respiratory diseases. Low and middle-income countries suffer the most, especially in the Western Pacific and South-East Asia regions. A significant cause of air pollution in urban areas worldwide is the high volume of road traffic. Studies have shown that people are willing to choose greener routes when credible information is provided

One of the approaches to solve this problem is to design a transportation policy that balances multiple objectives of

- Avoiding extreme pollution in any part of the road network
- Enabling short transit times for the vehicles that follow the policy
- Making effective use of the road capacities to route the traffic.

We propose a novel sampling-based approach for this problem. We provide the first construction of a *Markov Chain* that can sample integer max flow solutions of a planar graph, with theoretical guarantees that the probabilities depend on the aggregate transit length. We designed a traffic policy using diverse samples and simulated traffic on real-world road maps using the SUMO traffic simulator. We designed the *MaxFlow-MCMC* algorithm that used the *Markov Chain* to sample max flow solutions and generate a k-optimal set. We observe a considerable decrease in areas with severe pollution when experimented with maps of large cities across the world compared to other approaches.

Extending the above work, we also propose a significantly faster extension to the *MaxFlow-MCMC* algorithm without compromising on the performance involving the following contributions.We provide the first construction of a Markov Chain to sample a set of k-optimal max flow solutions directly from a planar graph. We also provide theoretical guarantees that the stationary distribution probabilities depend on both the k-optimality value and the aggregate transit length. We simulate traffic on large-scale real-world roadmaps using the SUMO traffic simulator. We observe a significant speed improvement in the range of 22 to 242 times in our experiments while obtaining lesser average pollution compared to the *MaxFlow-MCMC* algorithm.

To summarize, we present two novel *Markov Chain* based algorithms to generate a diverse set of max flow solutions from a planar graph, while being scalable to large road networks. We provide theo-

retical guarantees for the *Markov Chain* and also provide simulation results to show that our algorithms perfomred better than existing algorithms.

Contents

List of Figures

List of Tables

Chapter *1*

Introduction

1.1 Motivation

1.1.1 Air Pollution

Air pollution, a matter of growing global concern, poses a peril to human health and the delicate balance of the environment. The World Health Organization (WHO) warns that most of the planet's population lives in areas where the air is more polluted than recommended levels [40]. This contamination's sources vary, including transportation, industry, and energy generation. Amongst these, road traffic is considered a major contributor to air pollution in urban areas [16]. Inhaling these pollutants can lead to various health problems, from respiratory and cardiovascular diseases to certain cancers. Furthermore, air pollution can adversely affect the environment, contaminating soil and water and affecting our planet's biodiversity.

1.1.2 Long Term Exposure to Air Pollution

Long-term exposure to air pollution is especially problematic, as it heightens the risk of chronic health conditions. Studies have revealed that those exposed to elevated levels of pollutants over extended periods are at a higher risk of developing conditions such as heart disease, stroke, lung cancer, and chronic respiratory diseases [31]. Low and middle-income communities, particularly those in the Western Pacific and South-East Asia regions, are especially vulnerable. Research has shown that providing credible information can encourage individuals to make greener choices, such as choosing more eco-friendly modes of transportation [1].

1.1.3 Transportation Policy

In order to prevent the concentration of traffic-based pollution in specific pockets of an area, a transportation policy that can distribute the traffic flow more evenly throughout the road network of an area or a city was proposed in Kamishetty et al. (2020) [24]. The policy also ensured maximum throughput of

traffic throughput via the usage of *maximum flows* [15] to guide the traffic flow. They use the concept of k-optimality, which ensures that any two flows have at most $\leq k$ edges in common, and design a traffic policy using multiple *k-optimal maximum flow solutions* to route traffic. These solutions can then be used to route vehicles on different days or for reasonable timelines, thereby ensuring that the pollution gets distributed over different routes during the timeline. The concept of *k-optimality* [30] is used to capture the diversity among solutions. However, the proposed method could only be used for small areas due to the high computational complexity. We aim to develop more computationally efficient methods that can effectively distribute traffic flow evenly across the network and maintain reasonable path length while also being scalable to large city-scale road networks.

1.2 Overview

This thesis builds upon the previous line of work to develop significantly faster methods for pollution-aware traffic routing using a *Markov Chain Monte Carlo (*MCMC*)* [7, 19, 17, 6] based method to sample integer max flow solutions from a planar graph and generate a k-Optimal set of max flow solutions (hence named *MaxFlow-MCMC*). We extend the algorithm for sampling paths in planar graphs [28] and provide proof of convergence to the stationary distribution, which assigns a higher probability to max flow solutions with shorter aggregate path lengths.

We provide theoretical guarantees that the Markov chain will converge to a unique stationary distribution, where the max flow solutions that have a lesser total path length have a higher probability. We simulated the algorithms on large-scale real-world road networks of multiple cities using *Simulation of Urban Mobility (SUMO)* [26], which is a traffic simulator in combination with *OpenStreetMap (OSM)* [18], an open-source tool that helps to model traffic settings on real-world maps. We used the emission modelling capability of SUMO to compare the pollution levels generated.

We also build upon the *MaxFlow-MCMC* method to design an even faster Markov chain that can directly sample a set of diverse max flow solutions. We provide theoretical guarantees that the *Markov Chain* converges to a unique stationary distribution, where a set of max flow solutions that have lesser path length and more diversity will have a higher probability. We tested this algorithm, named *kOpt-MCMC*, on the same real-world road networks using SUMO and provided a significant improvement in runtime over the *MaxFlow-MCMC* method for parameters value that provide similar or less mean pollution.

1.3 Contributions

In Summary, Our Key contributions are

1. We design a novel *Markov Chain* that can be used to sample a set of integer max flow solutions from a planar graph proportional to its length.

2. We use the *MCMC* method to obtain a set of diverse max flows, which serve as the basis for directing vehicular traffic. Traffic routing using such a set of max flows ensures that

   i) A diverse set of paths are used, which prevents the concentration of pollution,

   ii) Vehicles are routed through shorter paths

   iii) The capacity of the road network is fully utilized as it is a max flow.

3. We design a *Markov Chain* that can be used to sample a set of integer max flow solutions from a planar graph with probabilities proportional to an exponential of a function combining both the total length and the diversity of the solution set

4. We provide proofs of convergences for both of the above *Markov Chains*. These proofs provide us with theoretical guarantees related to the quality of the Max Flow solution set generated.

5. We benchmark our method using the *SUMO* traffic simulator on real-world maps and show improvements in the concentration of pollution as well as order of magnitude improvements in the runtime of the algorithm as compared to previous works

1.4 Thesis Organisation

The rest of the thesis is organized as follows:

- **Chapter 2: Related Work** provides an overview of the related work in the field, including maximum flow problems, pollution-aware routing, Markov Chain Monte Carlo methods etc.

- **Chapter 3: MaxFlow-MCMC method** discusses the theory and design of the MaxFlow-MCMC method, which generates a k-optimal solution by sampling max flow solutions from a Markov Chain, along with the proof of convergence of the Markov Chain along with the analysis of time and memory complexity.

- **Chapter 4: EMF-MCMC method** discusses the theory and design of the EMF-MCMC method, which samples a k-optimal solution set directly from a Markov Chain, along with the proof of convergence of the Markov Chain along with the analysis of time and memory complexity.

- **Chapter 5: Experimental Details** provides details regarding the experimental setup. This section discusses how our methods can be applied in the real world, along with details of the SUMO simulator, which was used to perform the simulations. This section also describes how we measure the pollution to compare the algorithms along with the computational setup.

- **Chapter 6: Results** provides the experimental results for both the Algorithms. The first part of the section compares the *MaxFlow-MCMC* Algorithm with existing work in the field. It shows

how Maxflow-MCMC performs better than existing methods in terms of time and memory complexity. The rest of the section describes experiments conducted on large city-scale real-world roadmaps for both *MaxFlow-MCMC* and *EMF-MCMC*, and shows that *EMF-MCMC* provides similar performance to *MaxFlow-MCMC* in terms of pollution while being significantly faster.

- **Chapter 7: Conclusions and Future Directions** provides the conclusion for the work presented in this thesis and discusses a few of the possible directions in which this work could be extended in the future.

Chapter *2*

Related Works

2.1 Maximum Flow Problem

There are multiple algorithms to obtain a solution for the general maximum flow problem, including the *Ford-Fulkerson* algorithm [15], which can compute the maximum flow between two given points in a network including traffic networks [34]. The Ford-Fulkerson Algorithm computes the maximum flow between a source and a sink by iteratively finding paths that have a residual capacity and adding flows to them until no such paths can be found. However, most existing algorithms provide us with only a single max flow solution for a given source and a sink in a network. We aim to develop an algorithm to obtain multiple diverse max flow solutions between two points.

2.2 k-Optimality

The concept of k-optimality was initially proposed for the Distributed Constraint Optimization (DCOP) problem [30, 5]. The concept of k-similarity in traffic settings for computing the shortest paths between two nodes was introduced in [2]. We use a similar idea to define *k-optimality*, to ensure a certain amount of diversity among the max flow solutions that would help to distribute the pollution over more edges. We define a k-optimal set of max flow solutions as a set of max flow solutions where any two solutions from the set have at most k common edges.

2.3 Pollution routing problem

The literature on the pollution routing problem has proposed mathematical models that account for various factors, including greenhouse gas emissions, vehicle load, speed, and travel distance [3]. Subsequent studies have further expanded upon this concept by incorporating a fleet of vehicles of varying types [25]. However, none of these models has explicitly sought to maximize vehicle flow during routing. The development of solutions for the pollution routing problem has primarily been driven by data

related to the dynamics of individual vehicles, such as the speed and acceleration profiles used by [22] to predict and allocate energy-efficient routes. Other studies, such as those conducted by [12], have developed solutions for deterministic shortest-path problems based on velocity, start time, and load.

2.4 k-opt Pareto Max Flow Algorithm (k-PMFA)

The *k-Opt Pareto Max Flow* algorithm [24] was the first algorithm to compute a set of k-optimal max flow solutions where each max flow solution has at most k common edges with every other max flow solution in the set. They first presented a Pareto Max Flow Algorithm (PMFA), which finds all the possible max flow solutions between a given source and a given sink. They further present the k-opt Pareto Max Flow Algorithm (k-PMFA), which starts with an empty k-optimal set and iterates through the max flow solutions generated by the PMFA algorithm, checking for k-optimality and adding the max flow solutions to the k-optimal set accordingly. However, the algorithm involves the computation of simple paths from a given source to a given destination. A simple path is defined as a path between two nodes where no node is visited more than once [11]. The number of simple paths in a graph can be exponential in the size of the graph. Hence the algorithm requires exponential time. Our proposed algorithm and the *MaxFlow-MCMC method* generate a k-optimal set without requiring us to find and store all the simple paths and, therefore, will be more efficient in time and memory requirements.

2.5 Markov Chain Monte Carlo Methods

The *Markov Chain* Monte Carlo method is a popular method for generating combinatorial structures [23]. While such generation generally gives a uniform distribution over structures, we need to sample from a non-uniform distribution [27]. The MCMC method works by designing a Markov Chain whose stationary distribution is the probability distribution we desire to sample from. The Metropolis-Hastings Algorithm [20] is a popular method for designing a Markov Chain. The algorithm decides the transition probabilities from a state x to a state y based on the ratio of the stationary distribution probabilities of x and y. In this thesis, we present two novel Markov Chains whose transitions were designed similarly.

2.6 MCMC Method for sampling paths in a planar graph

Monatanari et al.(2015) [28] design an *MCMC* method for sampling simple paths in a planar graph. The *MaxFlow-MCMC* method was inspired by them but required additional proof to show the irreducibility of the *Markov Chain*. Their Markov Chain sampled paths from a planar graph proportional to their length. The transitions of the Markov chain were achieved using the faces of the planar graph. In this thesis, we present two novel Markov chains whose design was inspired by them. We present two

novel Markov chains that can sample a max flow solution and a set of max flow solutions directly from a planar graph. We also provide theoretical guarantees that both the Markov Chains will converge to the desired stationary distribution.

Chapter *3*

MaxFlow-MCMC method

In this chapter, we will discuss the design of the *Markov Chain* and the theory behind the *MaxFlow-MCMC* algorithm.

We consider the road network to be a planar graph $G(V, E)$. Edges E of the graph will be the roads, while nodes V will be the junctions where roads intersect or represent the end of roads. Each road in the network has a length and a specific number of lanes used as the edge's capacity since one vehicle can travel through each lane at a particular time. Let s be the source node from which the vehicles travel to the destination node t. We want to ensure that the maximum number of vehicles can travel from s to t at any time and hence represent this as a maximum flow problem.

Our approach starts with an initial integer max flow solution and makes small random modifications to it, resulting in a sample from a distribution over integer max flow solutions. This method is commonly known as *Monte Carlo Markov Chain (MCMC)* method. The *Ford Fulkerson Algorithm* can find the initial max flow or set of paths. The distribution will result in higher probabilities for max flows whose aggregate length of paths is shorter.

Since we are designing a system to route vehicles, we focus on integer max flow solutions where the flow through each edge is a non-negative integer. The *Integrality Theorem* [13], which is a corollary of the *Ford-Fulkerson Algorithm*, states that there exists at least one integer max flow solution in a graph where all the edge capacities are non-negative integers. Hence, we will have at least one integer max flow solution for a road network.

3.1 A Markov Chain for Integer Max Flow

Let us represent an integer max flow solution as a set of mf number of paths, where mf is the maximum flow value from s to t, and each path contains a flow of 1. Note that if an edge has a flow value f, it will be part of exactly f paths. We define a *Markov Chain* M_{flow}, whose state space Ω is the set of all integer max flow solutions. We define the total length of a max flow solution x denoted $|x|$ as the sum of the lengths of all the paths in it. The length of each path will be the sum of the lengths of all the edges in the path. We can reroute a path of the max flow solution using a face if they have one or

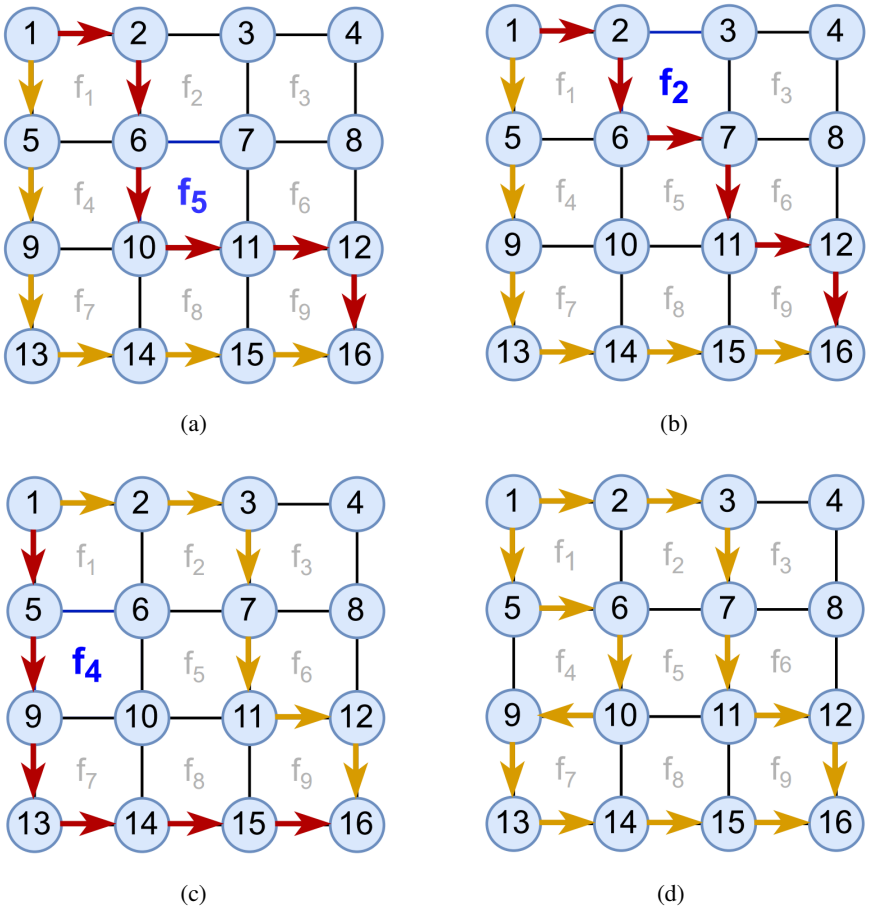

Figure 3.1: Example of a transition in the *Markov chain*. The underlying road network has 16 junctions forming a grid with max flow value from 1 to 16 to be 2. The state space consists of all possible sets of 7 max flows from 1 to 16. Each max flow can also be considered as a pair of $1 - 16$ paths (marked using arrows). The transition in M_{flowset} given in Algorithm 3, involves choosing one max flow solution from the set. Let's say the same max flow solution is selected for three consecutive transitions for the purpose of this example. The transition involves choosing one of the paths p (marked in red) and one of the faces $f \in \{f_1, \cdots f_{10}\}$ (marked as blue) of the planar graph at random and rerouting the path along the face. The faces $\{f_1 \cdots f_9\}$ are marked in the graph and f_{10} is the outer face.

Algorithm 1 $M_{\text{flow}}(x)$ defines a step of the *Markov Chain* on current state x. See Figure 3.1, for an example of its run.

faces = set of all faces in the planar graph
mf = the current integer max flow
paths = set of paths in x
$b \leftarrow$ Uniform($\{0, 1\}$)
if $b == 1$ **then**
    f $\leftarrow$ Uniform(faces), p $\leftarrow$ Uniform(paths)
    if f, p do not share an edge **or** rerouting p through f violates capacity **then**
        return x
    else
        $y \leftarrow$ reroute(p, f)
        return y with probability $\min\{1, \frac{\lambda^{|y|}}{\lambda^{|x|}}\}$ where $|x|$ and $|y|$ are the total lengths of x and y.
        return x
    end if
end if

more common edges, as shown in Figure 3.1. These transitions are inspired by [28], who built a similar *Markov Chain* for sampling paths. The transitions rules of M_{flow} are given in Algorithm 3.

The algorithm has a hyperparameter λ that decides the preference given to the total length of a state. If $\lambda < 1$, transitions to a state with a lesser total length will be more probable. Similarly, if $\lambda > 1$, transitions to a state with a higher total length will be more probable. When $\lambda = 1$, the transition to all neighbouring paths will have the same probability.

3.1.1 Convergence of the Markov Chain

We show that the *Markov chain* M_{flow} converges to a stationary distribution which has the property that the probability of max flow x will be proportional to $\lambda^{|x|}$. Setting $\lambda < 1$, allows us to sample max flows with shorter aggregate lengths. First, we show that the distribution with the above property is a stationary distribution.

Lemma 3.1.1. A stationary distribution of M_{flow} is

$$\pi(x) = \frac{\lambda^{|x|}}{Z} \qquad \text{where} \qquad Z = \sum_{y \in \Omega} \lambda^{|y|}$$

Proof. We show that π satisfies the detailed balance condition. That is, for the transition probabilities $P(x, y)$ (from x to y), and $\forall x, y \in \Omega$,

$$\pi(x) \cdot P(x, y) = \pi(y) \cdot P(y, x)$$

For the case where x and y are states that differ by more than one face, $P(x, y) = 0$ according to our transition rules, and therefore the detailed balance condition will be satisfied. Let us take two states, x, and y, which differ only by one face. Let $P(x, y)$ be the probability of transitioning from x to y. Let F be the total number of faces in the graph. Now,

$$\pi(x) \cdot P(x, y) = \frac{\lambda^{|x|}}{Z} \cdot \frac{1}{F} \cdot \frac{1}{mf} \cdot \min\left\{1, \frac{\lambda^{|y|}}{\lambda^{|x|}}\right\}$$

Similarly,

$$\pi(y) \cdot P(y, x) = \frac{\lambda^{|y|}}{Z} \cdot \frac{1}{F} \cdot \frac{1}{mf} \cdot \min\left\{1, \frac{\lambda^{|x|}}{\lambda^{|y|}}\right\}$$

If $\frac{\lambda^{|x|}}{\lambda^{|y|}} \leq 1$,

$$\begin{aligned}\pi(x) \cdot P(x, y) &= \frac{\lambda^{|x|}}{Z} \cdot \frac{1}{F} \cdot \frac{1}{mf} \cdot 1 = \frac{\lambda^{|y|}}{Z} \cdot \frac{1}{F} \cdot \frac{1}{mf} \cdot \frac{\lambda^{|x|}}{\lambda^{|y|}} \\ &= \pi(y) \cdot P(y, x)\end{aligned}$$

Similarly, If $\frac{\lambda^{|x|}}{\lambda^{|y|}} > 1$,

$$\begin{aligned}\pi(y) \cdot P(y, x) &= \frac{\lambda^{|y|}}{Z} \cdot \frac{1}{F} \cdot \frac{1}{mf} \cdot 1 = \frac{\lambda^{|x|}}{Z} \cdot \frac{1}{F} \cdot \frac{1}{mf} \cdot \frac{\lambda^{|y|}}{\lambda^{|x|}} \\ &= \pi(x) \cdot P(x, y)\end{aligned}$$

Therefore, M_{flow} satisfies the detailed balance condition, and π is its stationary distribution. □

Next, we show that starting from any state, M_{flow} converges to π, by showing that it is Ergodic. An Ergodic *Markov Chain* will have a unique stationary distribution and will converge to it [10]. For Ergodicity, we need to show that it is i.) aperiodic and ii.) irreducible.

Aperiodicity

A *Markov chain* is said to be aperiodic if

$$\forall x \in \Omega, \gcd\{t \in \mathbb{N} | P^t(x, x) > 0\} = 1. \tag{3.1}$$

where P^t is the t step transition probabilities. For M_{flow}, we defined transitions in such a way that there is a self-loop with a probability of at least $1/2$. Therefore, M_{flow} is aperiodic.

Irreducibility

A *Markov chain* is said to be irreducible if

$$\forall x, y \in \Omega, \exists t \in N \mid P_t(x, y) > 0. \tag{3.2}$$

That is, every state in the state space of M_{flow} can be reached from every other state with a finite number of steps. It is shown in Montanari et al. (2015) [28] that we can reach any simple $s - t$ path from any other path by rerouting through the faces of the planar graph. However, in our case, we need to ensure that the capacity constraints are not violated. We show this using the following definition and proposition.

Definition 3.1.2 (Outer paths of an Integer Flow)**.** We assume that the line joining s and t to be the direction of positive x axis. We define the *outer path* of an integer flow in a planar graph as a path between s and t with a non-zero flow, such that there is no other path in the flow with a non-zero flow in between the path and the outer face of the graph. We define the *top outer path* as the outer path that contains the node with the highest y-coordinate and the *bottom outer path* as the one that contains the node with the least y-coordinate.

Lemma 3.1.3. *There exists an outer path for every integer flow in a planar graph.*

Proof. Let p be a path with a non-zero flow in the max flow. If p is not the outer path then there exists a path $p1$ with a non-zero flow such that some part of $p1$ is between p and the outer face. We divide the proof into 2 cases

- When $p1$ is completely in between p and the outer face. This means that $p1$ will be the new outer path.
- When a part of $p1$ is between p and the outer face. Since we are working only with planar graphs, the edges cannot cross over each other. Therefore, $p1$ and p must intersect at a node. Let n be the node. Now our new path will be the path which has the edges of p from s to n and edges of $p1$ from n to t. Since both p and $p1$ have non-zero flows, the new path will also have a non-zero flow. We can keep repeating the above two steps until there is no such path $p1$.

□

Lemma 3.1.4. *Every max flow has at most 2 outer paths.*

Proof. We represent the planar graph on the x-y plane such that s is the origin and t lies in the positive x-axis. Let us define the top outer path as the outer path that contains the node with the highest y-coordinate and the bottom outer path as the outer path that contains the node with the least y-coordinate. By the definition of the outer path, we can say that there will not be any path with a non-zero flow below the bottom outer path and above the top outer path. Since the direction of flow is in the direction of the positive x-axis, there cannot be any paths that are completely to the left of s or completely to the right of t. Therefore, there can be a maximum of 2 outer paths. □

Proposition 3.1.5. *We can reach any state in M_{flow} from any other state in a finite number of steps.*

Proof. Let x and y be any two states in Ω and $paths(x), paths(y)$ be the set of mf paths in x, y respectively. Let $op_x \in paths(x)$ and $op_y \in paths(y)$ be the top outer paths of x and y respectively. We transform op_x and op_y to the same s - t path using a sequence of reversible M_{flow} transitions.Let $s, v_1 \cdots, v_r, t$ be common nodes of op_x and op_y. For each of the $r + 1$ subpaths between these nodes, we convert the subpath in the lower one of op_x, op_y to the other. This is possible without violating capacity constraints since there is no flow above the top outer paths by definition. Then we consider the residual flow x', y' given by removing the unit flow through these paths from x, y. Then we repeat the same process on x', y', until the residual flow becomes 0. A diagrammatic example of the above proof is given in Figure 3.2

□

Since our *Markov chain* is both irreducible and aperiodic, it is ergodic and will have a unique stationary distribution to which it will converge irrespective of the initial state. As proved in Lemma 4.1.1, the probability of a max flow solution x is proportional to $\lambda^{|x|}$ in the stationary distribution.

3.2 Traffic Routing using MaxFlow-MCMC

We propose to use the *MaxFlow-MCMC* algorithm (see Algorithm 2) to generate a k-Optimal max flow solution set, which can be used for routing vehicles in such a way that the pollution is evenly spread out. Even though our chain is not rapidly mixing, we can start sampling before the chain has completely mixed as we do not need the solutions to follow the specific distribution for our use case since we are just sampling to generate a k-optimal set of max flow solutions. We define the k-optimality of our solution set similar to the definition of k-similarity in barrett2008engineering. A set of max flow solutions is k-optimal when any two solutions from the set have at most k common edges.

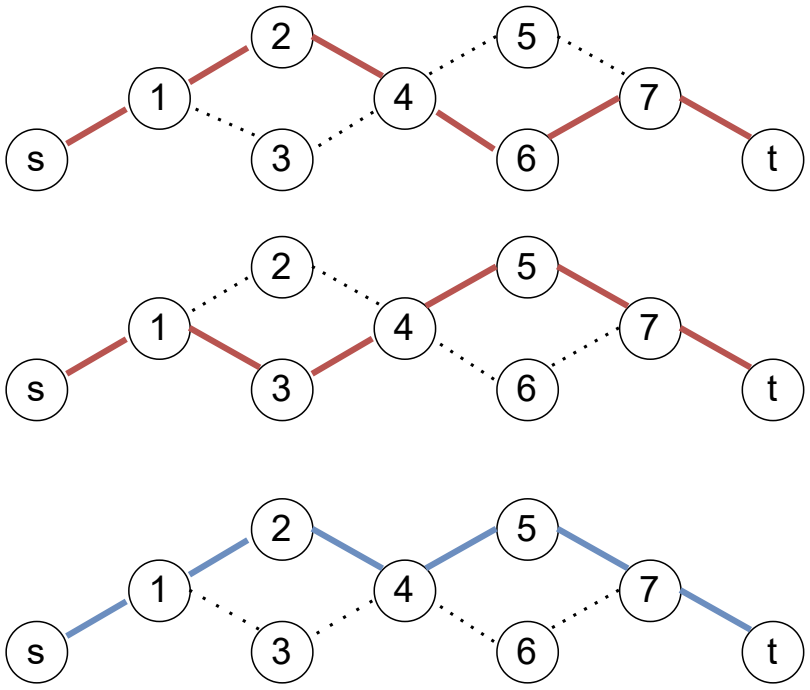

Figure 3.2: Let us consider the above planar graph which has a source node s, a destination node t and seven other nodes labelled $1-7$. Let all the edges have a flow value of 1. The solid lines denote a flow and dotted lines represent an edge without a flow. The first two diagrams represent the top outer paths of two different max flow solutions. The common nodes between the two are $s, 1, 4, 7, t$, which gives us four subpaths. For each of these subpaths, we convert the subpath in the lower one of the two to the other, resulting in both of these paths becoming the path shown in the third diagram. For max flow solutions with more than one path, we create the residual graph and repeat above the steps until both the max flow solutions reach the same state. Since all our steps are reversible, we can reach any state from any other state in a finite number of steps. to the other.

Algorithm 2 x denotes current state of the *Markov chain*, *FFA* denotes Ford Fulkerson Algorithm, `mix_iter` denotes number of iterations for mixing, `num_iter` denotes number of iterations for sampling and sf denotes sampling frequency.

$x \leftarrow$ FFA(s,t), solutionset $\leftarrow \emptyset$
while iter $\leq$ `num_iter` + `mix_iter` **do**
  if iter $>$ `mix_iter` and iter% sf == 0 **then**
    if KOpt(x,solutionset) **then**
      solutionset $\leftarrow$ solutionset + x
    end if
  end if
  x $\leftarrow M_{\text{flow}}(x)$ (See Algorithm 1)
  iter $\leftarrow$ iter+1
end while
return solutionset

We use the *Ford-Fulkerson Algorithm* [15] to find one max flow solution for the starting state. We initially let the *Markov chain* run for a few iterations without sampling to ensure that the initial state is random when we start sampling. We sample a random max flow solution from the current distribution every few iterations and check if it is k-optimal with the solution set. If it is, we append it to the solution set. We keep repeating this until we reach the exit condition. The *MaxFlow-MCMC* Algorithm has multiple parameters we can modify to suit our time and solution constraints, making it scalable for larger maps.

3.3 Parameters for MaxFlow-MCMC

- **Sampling Frequency**: The number of iterations after which we sample a max flow solution from the current distribution. If we want to increase the number of solutions in the solution set, we can sample the solutions more frequently, and if the aim is to improve the runtime of the algorithm, we can sample less frequently.
- **Exit Condition**: The User can decide the exit condition of the algorithm based on requirements. The algorithm can be run for a fixed number of iterations, or it can be run until a solution set of the required size is obtained.
- **Lambda(λ)**: Affects the total length of the max flow solutions. A lesser value of lambda would mean that the solutions obtained will typically have a lesser total path length. In comparison, a

higher value of lambda would mean that the solutions will also include paths of higher length (which can allow sampling a large number of solutions).

- **Value of k**: Increasing the Value of k would decrease the time required to obtain a set of solutions of the same size, but the set would have more common edges, leading to a lesser spread of pollution in general. Similarly, decreasing the value of k will increase the time required to generate the k-optimal set of the same size but the solutions obtained will be more diverse.

3.4 Complexity Analysis of MaxFlow-MCMC

3.4.1 Time Complexity.

The initial *FFA* step is of complexity $O(|E| * mf)$, where $|E|$ is the number of edges and mf is the max flow. Calculating the faces needs $O(kV)$ on average, and $O(V^2)$ in the worst-case [33]. At every step of the algorithm, the following computations need to be done: (a) Randomly choosing a face needs $O(|F|) = O(|E|)$ since $|F| + |E| - |V| = 2$ by Euler's formula [38], where $|F|$ and $|V|$ are the number of faces and nodes respectively. (b) Checking for max flow condition needs reasoning over the number of edges in the path, i.e., $O(|E|)$. (c) Rerouting involves $O(|E|)$. (d) Checking for k-Optimality needs to be done every sf steps needing $O(\frac{num\_sol*|E|^2}{sf})$.

The overall worst-case time complexity would therefore be $O(\frac{num\_iter*num\_sol*|E|^2}{sf} + |V|^2)$. Therefore, worst case runtime of *MaxFlow-MCMC* will be polynomial in $|E|$ and $|V|$, provided that num_iter and num_sol are not very high, both of which can be tuned per need in the algorithm. Note that the runtime for *MaxFlow-MCMC* does not include the time required for calculating the faces of the graph since it was done as a prepossessing step. schneider2019finding proves that even this step can be computed with an average complexity of $O(k|V|)$ and, therefore, will not make a significant change to the overall runtime.

3.4.2 Space Complexity.

Following are the key steps involved in the algorithm, which require: (a) A memory of $O(|E|^2)$ for storing all the faces. (b) A memory of $num\_sol * mf * |E|)$ for storing the solution set. This leads to an overall worst-case space complexity of $O((num\_sol * mf * |E| + |E|^2)$. In summary, our algorithm has a time and space complexity that is tractable in the number of edges and, therefore, scalable to larger networks.

Chapter *4*

EMF-MCMC method

In this chapter, we build upon the previous *Markov Chain* to design a significantly faster Markov chain that can directly sample a set of diverse max flow solutions. The advantage of sampling a max flow solution set directly is as follows:

- We can account for the diversity of the solution set alongside the path lengths within the *Markov Chain* itself. Therefore, we can make it converge towards max flow sets with both higher diversity and lesser path lengths. The previous *Markov Chain* could not account for diversity as we are sampling max flow solutions individually and checking for k-optimality.
- The proposed *EMF-MCMC* method is an anytime algorithm. That is, we are guaranteed to obtain a solution set of a given size n, whose solution quality increases over time. In addition, we can stop the algorithm at any point in time and obtain a k-optimal set whose quality depends on the amount of computation performed. With the previous *Markov Chain* we have to set the value of k beforehand, but we will not know how long it will take for the *Markov Chain* to generate the required number of solutions.
- The algorithm uses a set of n max flow solutions as the initial solution set and increases its quality over time. Hence, we can obtain better-quality solutions if we have a better starting state. We can create a good set of starting states by using max flow solutions obtained from different algorithms.
- Given the significant speed improvement over the *MaxFlow-MCMC* method, the *EMF-MCMC* method will be especially useful when there are frequent changes in the network like roads getting blocked.

The road network is specified by a graph $G(V, E)$, where the nodes V correspond to road intersections or the endpoint of roads, while the edges E represent roads. Each edge has a length, and the number of lanes is specified as its capacity. One of our goals is to maximize the number of vehicles that can travel from the source node s to the destination node t at any given time, and hence this objective is represented as a maximum flow problem. Similar to the *MaxFlow-MCMC* method, we also want the traffic/pollution to be distributed across multiple edges, for which we try to build a diverse set of

k-Optimal integer max flows. We define a k-optimal set of max flow solutions as a set of max flow solutions where any two solutions from the set have at most k common edges.

4.1 A Markov Chain for sampling a set of Integer Max Flow solutions

As opposed to the *MaxFlow-MCMC* method in the previous chapter that uses a single max flow solution, the *EMF-MCMC* method in this chapter expands the starting state to a set of n integer max flow solutions. We make small random modifications to the initial set, which results in a distribution over all the possible sets of n integer max flow solutions over time. Our proposed method starts by initializing the algorithm with a set of solutions obtained from different max flow algorithms, such as the Ford Fulkerson Algorithm (FFA), with augmenting paths found using Breadth-First Search (BFS) and Dijkstra's Algorithm.

The stationary distribution of our *Markov Chain* will set higher probabilities for max flows with lesser aggregate path length and higher diversity based on the parameters we set. Similar to the previous chapter, we limit our focus to integer max flow solutions as we are designing a system to route vehicles, where the flow through each edge is a non-negative integer.

The integer max flow solutions are represented as a collection of mf paths, where mf is the maximum flow value from the source node s to the destination node t. Each path within the collection is assigned a flow value of 1. It is important to note that if a particular edge has a flow value of $f > 1$, it will be present in exactly f paths.

We define a *Markov Chain*, denoted as M_{flowset}, that has a state space Ω consisting of all feasible max flow solution sets of a predefined size n. The total length of a max flow solution set x, represented as $|x|$, is calculated as the aggregate of the lengths of all the paths in x, where the length of each path is determined by the sum of the lengths of the edges composing it. Let λ be a predefined constant and x be the current state (a set of n integer max flows). We can reroute a path of a max flow solution using a face that shares one or more common edges with the path [28]. Figure 3.1 provides an example of a transition in the *Markov Chain*. The transition rules of M_{flowset} are given in Algorithm 3. The transition rules are designed using the Metropolis-Hastings method [20] so that the stationary distribution of M_{flowset} assigns probabilities for each state x proportional to $\lambda^{f(x)}$ where λ is a hyperparameter and $f(x)$ is a cost function.

We use a function of total path length and the diversity of the solution set to decide the stationary distribution probability of a given state. The function is defined as follows.

$$f(x) = \alpha * |x| + (1 - \alpha) * k^2 \tag{4.1}$$

Here, k indicates the k-optimality value of the solution set, and α is a parameter between 0 and 1 that decides the priority trade-off between the total path length and the diversity of the solution set. Increasing the value of α will result in a max flow solution set with a lesser path length at the cost of a

higher k value, indicating that it will be less diverse. Similarly, decreasing the value of α will result in a max flow solution set with a lesser value of k at the cost of a higher total path length.

We wanted to make sure that the value of k-optimality is not very high and hence use the square of it in the equation to give a higher preference towards minimising it. Using a higher power of k resulted in extremely large values, due to which we decided to use k^2. The value of λ will decide the preference given to the $f(x)$ of a state x. If $\lambda < 1$, transitions to a state x with a lesser value of $f(x)$ will be more probable. Similarly, if $\lambda > 1$, transitions to a state x with a higher value of $f(x)$ will be more probable. When $\lambda = 1$, $\frac{\lambda^{f(y)}}{\lambda^{f(x)}} = 1$ and hence the transition to all neighbouring states will have the same probability.

4.1.1 Properties of the Markov Chain.

We show that the *Markov Chain* M_{flowset} converges to a stationary distribution which has the property that the probability of max flow set x will be proportional to $\lambda^{f(x)}$. Setting $\lambda < 1$, allows us to sample max flows with shorter aggregate lengths and lesser value of k.

First, we show that the distribution with the above property is a stationary distribution.

Lemma 4.1.1. A stationary distribution of M_{flowset} is

$$\pi(x) = \frac{\lambda^{f(x)}}{Z} \qquad \text{where} \qquad Z = \sum_{y \in \Omega} \lambda^{f(y)}$$

Proof. We show that π satisfies the detailed balance condition. That is, for the transition probabilities $P(x, y)$ (from x to y), and $\forall x, y \in \Omega, \pi(x) \cdot P(x, y) = \pi(y) \cdot P(y, x)$.

For the case where x and y are states that differ by more than one face in a single max flow solution or differ in more than one max flow solution, $P(x, y) = 0$ according to our transition rules, and therefore the detailed balance condition will be satisfied. Let us take two states, x, and y, which differ only by one face in one max flow solution. Let $P(x, y)$ be the probability of transitioning from x to y. Let F be the total number of faces in the graph. Now,

$$\pi(x) \cdot P(x, y) = \frac{\lambda^{f(x)}}{Z} \cdot \frac{1}{n} \cdot \frac{1}{F} \cdot \frac{1}{mf} \cdot \min\left\{1, \frac{\lambda^{f(y)}}{\lambda^{(x)}}\right\}$$

If $\frac{\lambda^{f(x)}}{\lambda^{f(y)}} \leq 1$,

$$\begin{aligned}\pi(x) \cdot P(x, y) &= \frac{\lambda^{f(x)}}{Z} \cdot \frac{1}{n} \cdot \frac{1}{F} \cdot \frac{1}{mf} \cdot 1 = \frac{\lambda^{f(y)}}{Z} \cdot \frac{1}{n} \cdot \frac{1}{F} \cdot \frac{1}{mf} \cdot \frac{\lambda^{f(x)}}{\lambda^{f(y)}} \\ &= \pi(y) \cdot P(y, x)\end{aligned}$$

Algorithm 3 $M_{\text{flowset}}(x)$ defines a step of the Markov Chain on current state x.

faces = set of all internal faces in the planar graph
mf_set = the current set of integer max-flows
$b \leftarrow$ Uniform($\{0, 1\}$)
if $b == 1$ **then**
    $mf \leftarrow$ Uniform(mf_set)
    paths = set of paths in mf
    f $\leftarrow$ Uniform(faces), p $\leftarrow$ Uniform(paths)
    if f, p do not share an edge **or** rerouting p through f violates capacity **then**
        return x
    else
        $y \leftarrow$ reroute(p, f)
        return y with probability $\min\{1, \frac{\lambda^{f(y)}}{\lambda^{f(x)}}\}$ where $f(x)$ and $f(y)$ are calculated for x and y according to equation 4.1
        return x
    end if
end if

Similarly, If $\lambda^{f(x)}/\lambda^{f(y)} > 1$,

$$\begin{aligned}\pi(x) \cdot P(x, y) &= \frac{\lambda^{f(x)}}{Z} \cdot \frac{1}{n} \cdot \frac{1}{F} \cdot \frac{1}{mf} \cdot \frac{\lambda^{f(y)}}{\lambda^{f(x)}} = \frac{\lambda^{f(y)}}{Z} \cdot \frac{1}{n} \cdot \frac{1}{F} \cdot \frac{1}{mf} \cdot 1 \\ &= \pi(y) \cdot P(y, x)\end{aligned}$$

Therefore, M_{flowset} satisfies the detailed balance condition, and π is its stationary distribution. □

Next, we show that starting from any state, M_{flowset} converges to π by showing that it is Ergodic. We need to show that the chain is both aperiodic and irreducible.

Aperiodicity

For M_{flowset}, we defined transitions in such a way that there is a self-loop with a probability of at least $1/2$. Therefore, M_{flowset} is aperiodic as per the definition in Equation 3.1..

Irreducibility

In the previous chapter on the *MaxFlow-MCMC Algorithm*, we proved that the *Markov Chain* M_{flow} is irreducible. This implies that we can reach from any integer max flow to any other integer max flow in a finite number of steps. Since M_{flowset} has a set of n integer max flows, and all of them have a non-zero probability of being selected for a transition, we can say that M_{flowset} is irreducible. This is because we can serially do transitions between two states x and y such that the first max flow in x reaches the first max flow in y, the same for the second, third till the last max flow of the set x reaches that of y. Therefore, our *Markov Chain* is irreducible as per the definition in Equation 3.2.

Since our *Markov Chain* is both irreducible and aperiodic, it is ergodic and will have a unique stationary distribution to which it will converge irrespective of the initial state. As proved in Lemma 4.1.1, in the stationary distribution, the probability of a max flow solution x is proportional to $\lambda^{f(x)}$.

4.2 Traffic Routing using EMF-MCMC

We propose to use the M_{flowset} algorithm to generate a k-optimal max flow solution set, which can be used for routing vehicles in a manner that promotes even distribution of pollution. The concept of k-optimality is established by drawing parallels with the definition of k-similarity in [2]. A set of max flow solutions is considered k-optimal if any two solutions from the set have no more than k common edges.

We introduce two adaptations of the classic Ford-Fulkerson Algorithm [15] to initialize the *Markov Chain* M_{flowset}. The first version, referred to as FFA(BFS), employs Breadth-First Search [41] to identify augmenting paths in the network. The second version, referred to as FFA(Dijkstra), utilizes Dijkstra's Algorithm [9] to discover augmenting paths. These two versions of the algorithm serve as a starting point for generating a set of max flow solutions in the *Markov Chain*.

We create our initial set of 7 max flows by using three solutions of FFA(BFS) and four solutions of FFA(Dijkstra). We simulate our *Markov Chain* until the exit condition is reached. As the *Markov Chain* converges to the stationary distribution over time, we are guaranteed to obtain better solutions over time. We can modify the parameters for M_{flowset} based on our constraints.

In the *MaxFlow-MCMC* method, whenever a solution was sampled from the *Markov Chain*, the k-optimality value with all the solutions in the set was checked. In contrast, the *EMF-MCMC* method maintains a running tally of the k-optimality values between all pairs of max flow solutions in the set. It updates them by considering only those edges that have been modified in the transition instead of all the edges. This helps to optimize the computational efficiency and helps to avoid recalculating the k-optimality at every step of the simulation.

4.3 Parameters for EMF-MCMC

- **Alpha (α)**: Affects the trade-off between total path length and the k-optimality value of the max flow set. A higher value of α would give a higher priority towards reducing the path length, and a lower value of α will give a higher priority towards reducing the k-optimality value, thereby improving the distribution of pollution.
- **Exit Condition**: The user can decide the exit condition of the algorithm based on requirements. The algorithm can be run for a fixed number of iterations or a fixed amount of time based on the user's constraints.
- **Lambda(λ)**: Affects the importance given to the value of the function f for the probabilities of the max flow solution sets. A lesser value of λ would mean that the solutions obtained will typically have a lesser value of f. In comparison, a higher value of λ would mean that the solutions will also include sets with a higher value of f (which can allow the *Markov Chain* to explore over a larger number of solutions).

4.4 Complexity Analysis of EMF-MCMC

4.4.1 Time Complexity.

The initial *FFA* step is of complexity $O(|E| * mf)$, where $|E|$ is the number of edges and mf is the max flow. Calculating the faces needs $O(kV)$ on average, and $O(V^2)$ in the worst-case [33]. At every step of the algorithm, the following computations need to be done:

1. Randomly choosing a face needs $O(|F|) = O(|E|)$ since $|F| + |E| - |V| = 2$ by Euler's formula. [38], where $|F|$ and $|V|$ are the number of faces and nodes respectively.
2. Randomly choosing for a path needs $O(\text{max\_flow} * \text{num\_sol})$
3. Checking for max flow condition needs reasoning over the number of edges in the path, i.e., $O(|E|)$.
4. Rerouting involves $O(|E|)$.
5. Updating the k-optimality value needs $O(|E| * \text{num\_sol})$ since we only need to update the values using those edges that are part of the face we chose.

The overall worst-case time complexity of one step of *EMF-MCMC* would therefore be $O(|E| * (\text{num\_sol} + \text{max\_flow}) * \text{num\_sol})$. Therefore, worst case runtime of *EMF-MCMC* will be $O(\text{num\_iter} *$

$(|E| + \text{max\_flow}) * \text{num\_sol})$, which is lesser than the complexity of *MaxFlow-MCMC*, which was $O(\frac{\text{num\_iter}*\text{num\_sol}*|E|^2}{sf} + |V|^2)$.

Since the Maximum flow value will usually be lesser than the total number of edges in the graph (since we are dealing with large-scale road networks, where the number of edges will be high and the max flow value will be small), for all practical purposes, we can assume the complexity of one step of *EMF-MCMC* to be $O(|E| * \text{num\_sol})$. Hence, the worst case runtime of *EMF-MCMC* will be $O(\text{num\_iter} * |E| * \text{num\_sol})$,

Note that the runtime mentioned for both the *EMF-MCMC* and *MaxFlow-MCMC* does not include the time required for calculating the faces of the graph since it was performed as a prepossessing step. Schneider et al. (2015) [33] provide proof that even this step can be computed with an average complexity of $O(k|V|)$. Hence, the inclusion of this step will not make a significant change to the overall runtime.

4.4.2 Space Complexity.

Following are the critical steps of the algorithm in terms of memory, which require:

1. A memory of $O(|E|^2)$ for storing all the faces.

2. A memory of $O(\text{num\_sol} * mf * |E|)$ for storing the current state of the *Markov Chain* (i.e., a set of max flows).

This leads to an overall worst-case space complexity of $O(\text{num\_sol} * mf * |E| + |E|^2)$, which is the same as that of *MaxFlow-MCMC*. In summary, our *EMF-MCMC* algorithm has a time complexity lesser than that of *MaxFlow-MCMC* and a memory complexity similar to that of *MaxFlow-MCMC*. The reduced time complexity, combined with the fact that the *Markov Chain* in *EMF-MCMC* explicitly optimizes for the k-optimality value along with the path length, should lead to significant improvements in efficiency as compared to *MaxFlow-MCMC*.

Chapter *5*

Experimental Details

In this chapter, we describe the details of the experiments performed. First, we mention the different ways our experiments can be translated into real-world scenarios. We then describe the SUMO traffic simulator, which we used to perform the simulations. We also describe the parameters used to evaluate both the *MaxFlow-MCMC* and *EMF-MCMC* algorithms.

5.1 Implementing Policy in Simulation and Real World

By offering a set of diverse, k-optimal max flow solutions, both the *MaxFlow-MCMC* and *EMF-MCMC* algorithms can be implemented in a variety of real-world scenarios.

- As described in [24], traffic police could utilize different max flow solutions for routing on different days of the week.
- GPS software like Google Maps [8] could also utilize the algorithm to provide unique path suggestions to different users, thus promoting an even distribution of vehicular traffic.
- The algorithm's utilization can also be extended to autonomous vehicles, where vehicles could be instructed to follow different max flow solutions.

5.2 Traffic Simulation using SUMO

Simulation of Urban MObility (SUMO) is an open source, microscopic and continuous traffic simulation package designed to handle large networks. SUMO is equipped with a large set of tools for scenario creation and simulation and can handle multiple aspects of traffic flow generation, including computation of acceleration, deceleration, emission modelling, congestion, the distance between vehicles, etc., resulting in realistic modelling.

Parameter	Value
Dimensions (length x width x height)	5 m x 1.8 m x 1.5 m
Minimum gap between vehicles	2.5 m
Maximum acceleration	2.6 $\mathrm{m/s^2}$
Maximum deceleration	4.5 $\mathrm{m/s^2}$
Emergency deceleration	9 $\mathrm{m/s^2}$
Maximum speed	200 km/h
Emission Class	HBEFA3/PC_G_EU4

Table 5.1: Default vehicle parameters used by SUMO. Our simulations were performed using vehicles with the above parameters

Map name	Sq. km	Edges	Nodes	s, t Pairs
Seattle	25.16	18699	14939	2
Berkeley	82.90	30808	24864	1
Kanpur	18.50	29956	20707	1
Islington	14.90	5382	2367	2

Table 5.2: Details of the maps used for the experiments.

Table 5.1 contains the default vehicle parameters used by SUMO. We used the same parameters for our simulation. Besides the ones mentioned in Table 5.1, SUMO lets us assign departure and arrival-related data for each individual vehicle. Each vehicle has a unique identifier, the route the vehicle should take, the departure time etc. SUMO also allows us to modify other parameters related to the speed, arrival and departure properties etc.

5.3 Experimental Setup

Both of the algorithms proposed in this thesis *MaxFlow-MCMC* method and the *EMF-MCMC* method are applicable only for planar graphs. Therefore, we modified the real-world network to make it planar by removing the non-planar components such as bridges. Since there can be cases of multiple sources and destinations for vehicles, we performed experiments with more than one source and destination also. We introduce a virtual source and a virtual destination node, connected to all the source

Name	λ	num_iter	sf
MaxFlow-MCMC1	0.95	50000	25
MaxFlow-MCMC2	0.95	25000	25
MaxFlow-MCMC3	0.90	50000	25
MaxFlow-MCMC4	0.95	50000	50
MaxFlow-MCMCLarge	0.99	until 7 sol	25
MaxFlow-MCMCKanpur	0.95	until 7 sol	25

Table 5.3: Parameter values used for the *MaxFlow-MCMC* algorithm in the experiments. The first four are used for comparison with k-*PMFA* and the last two are for large city-scale experiments.

Name	λ	num_iter	α
EMF-MCMC_Berkeley	0.99	3000000	0.9
EMF-MCMC_Islington	0.99	500000	0.9
EMF-MCMC_Kanpur	0.95	5000000	0.9
EMF-MCMC_Seattle	0.99	500000	0.9

Table 5.4: Parameter values used for the *EMF-MCMC* algorithm in the large city-scale experiments.

and sink nodes, respectively, using infinite capacity edges [4]. Algorithms were then used with vehicles travelling from the virtual source to the virtual sink. For the large-scale experiments, we used four real-world maps corresponding to Seattle, Berkeley, Kanpur and Islington. The details of the maps are given in Table 5.2.

We tested both the algorithms for different values of the parameters and simulated traffic using the *SUMO* simulator [26]. We used Sage Math [36] to find the faces of the roadmap. Vehicles were released in waves, with a 5 second interval for the small map and a 15 second interval for the larger map, to create a reasonable distance between vehicles. The number of vehicles per wave was equal to the maximum flow from the source to the destination. We used the longer interval for the larger map due to roads with different speed limits and to account for a potentially higher number of turns. In addition, vehicles have to slow down at junctions for turns, and a longer path would potentially involve a larger number of turns. We used NOx values to measure the pollution levels, as they have a high dependency on traffic flow and adversely impact human health [37]. The results were visualized using matplotlib [21] and seaborn [39].

Due to the randomness involved in the *MaxFlow-MCMC* algorithm, different runs using the same parameters may provide us with a different number of solutions. Hence, we used each solution of the generated k-optimal set for 1 hour, which can correspond to one day in reality, and computed the *Normalised Mean pollution* (NOx_{nm}) to compare the different runs of the algorithm accurately. While the *EMF-MCMC* method always provides us with a fixed number of max flow solutions, we use the same metric to evaluate the performance as we want to compare it with that of *MaxFlow-MCMC*.

$$NOx_{nm} = \frac{\textit{Mean amount of NOx released per hour}}{\textit{Total number of edges with non-zero emissions}}$$

SUMO provides us with the amount of NOx released for every edge with non-zero emissions over the simulation period. We calculate NOx_{nm} by taking the mean pollution over all the edges and dividing it by the size of the solution set we used, as we use one solution per hour. NOx_{nm} helps us understand how well the pollution is distributed over an area. This is due to the insight that to decrease NOx_{nm}, we need to reduce the mean amount of NOx released, or we have to increase the number of edges being used. Since we are releasing the same number of vehicles per hour for all the simulations, NO_xnm will be highly dependent on the pollution spread.

Tables 5.3 and 5.4 show the different sets of parameters used in the experiments for *MaxFlow-MCMC* and *EMF-MCMC* respectively. The values of λ were set to be close to 1, as the lengths of the paths were measured in meters. The rationale behind this is that the probability of transitioning from one state to another in the *Markov Chain* depends on the difference in the lengths of these states. If we set λ to a lower value, such as 0.5, the probability of transitioning to a state with a length only 5 meters longer would be close to 0.03 for the *MaxFlow-MCMC* algorithm. The same will be true *EMF-MCMC* algorithm with $\alpha = 1$.

The runtime results presented for all the algorithms were measured on an Intel E5-2640 processor on an HP SL230s compute node. All the results were obtained by averaging over 30 runs of the algorithms to represent the performance better.

Chapter *6*

Results

In this chapter, We benchmark the performance of both the *MaxFlow-MCMC* algorithm and the *EMF-MCMC* algorithm. We first conduct experiments on a small map comparing the *MaxFlow-MCMC* algorithm with the k*-PMFA* algorithm using the SUMO simulator. We also simulated our *EMF-MCMC* and *MaxFlow-MCMC* algorithms on large-scale maps of many cities and compared them with the Ford Fulkerson Algorithm (FFA). All the networks were obtained from *OpenStreetMap* [18].

6.1 Comparision of MaxFlow-MCMC with k-PMFA

Normalised Mean Pollution

City	Traffic Policy	Length avg (m)	Pollution (NOx_{nm} in mg) avg	max	total
Small Map	FFA(BFS)	213	4387	9644	61428
	FFA(Dij)	198	4616	19884	73853
	k-PMFA(7)	208	1853	11255	72262
	MCMC(7)	207	1784	11776	69104

Table 6.1: Comparing the pollution values of k*-PMFA* and *MCMC* for the small map with $k = 9$. We can observe that *MaxFlow-MCMC* gives a performance comparable to that of k*-PMFA* while still being scalable to larger maps.

k*-PMFA* [24] is a pollution-aware routing algorithm that we compare within Table 6.1. The table shows the performance of three algorithms, namely *MaxFlow-MCMC*, k*-PMFA*, and *FFA*. For both the

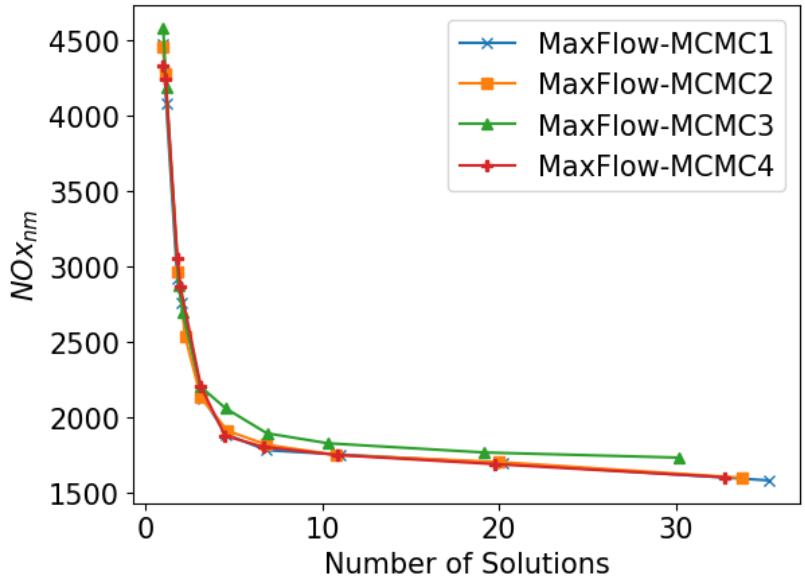

Figure 6.1: Plot of Normalised mean pollution vs The number of max flow solutions used. The x-axis represents the number of solutions in the k-optimal set, while the y-axis represents the normalised mean pollution obtained by using all the solutions in the k-optimal set. The change in normalised mean pollution reduces as we increase the number of solutions (obtained by increasing the value of k). While there is a significant reduction up to 10, the benefits are lesser later. This indicates that we can obtain good performance with a small solution set.

k-PMFA and *MaxFlow-MCMC* algorithm, we used a k value of 9. As shown, the *Maxflow-MCMC* gives comparable results while being scalable.

Number of solutions

Figure 6.3 provides comparison results for the number of max flow solutions provided by the k-PMFA and MaxFlow-MCMC algorithms. The x-axis of the figure shows the value of k while the y-axis represents the average number of solutions in the k-optimal set. There are 5 lines in the figure that corresponds to the four MCMC variants (as presented in Table 5.3) along with k-PMFA. The figure shows that the MaxFlow-MCMC method serves as a very good approximation for k-PMFA. For example, when the value of k is set to 10, k-PMFA provided a solution set of size 12, while the 4 variants of MaxFlow-MCMC provided 11.03, 10.8, 10.33 and 10.83 solutions on average, with a maximum of 14, 13, 12, 12 and a minimum of 9, 9, 9 and 9 solutions respectively.

The graph also confirms the effect of the frequency of sampling, i.e., when we started sampling more frequently while keeping the other parameters constant, the number of solutions obtained were increased. We can see that MaxFlow-MCMC1 has a higher number of solutions as compared to MaxFlow-

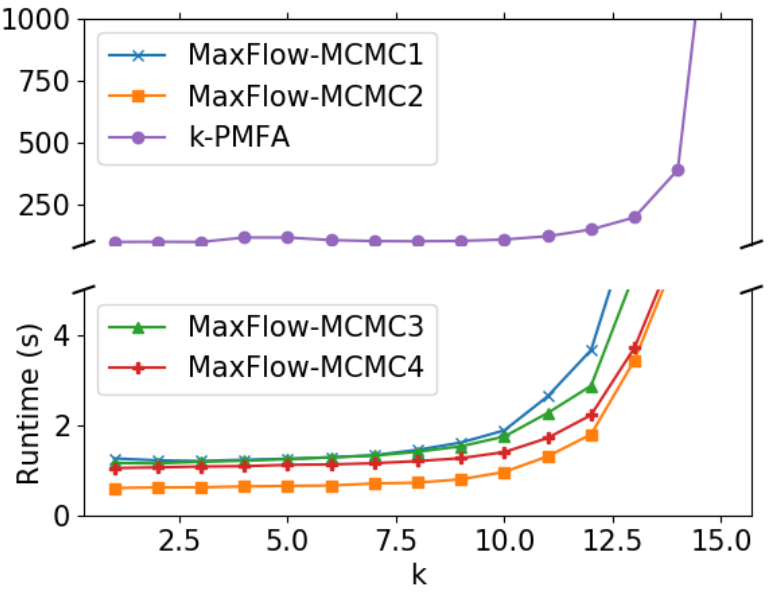

Figure 6.2: Runtime comparison of *MaxFlow-MCMC* vs k*-PMFA*. The x-axis represents the k-optimality value used while the y axis represents the runtime of the algorithm in seconds. We can observe that the runtime of k*-PMFA* is two orders of magnitude higher than *MCMC*, making it infeasible for road networks over a few sq km. Increasing the value of k increases the diversity while also increasing the running time, especially after $k = 10$.

MCMC4 since we sample more frequently. Similarly, when the number of iterations were reduced while keeping the other parameters constant, the number of solutions obtained for the same value of k decreased. We can see that the number of solutions obtained by MaxFlow-MCMC2 is less than that of MaxFlow-MCMC1 due to the lesser number of iterations.

Effect of hyperparameters

Figure 6.1 studies the effect of using multiple max flow solutions on the pollution level. We also experiment with four different hyperparameter values of *MaxFlow-MCMC* as given in Table 5.3, denoted as *MaxFlow-MCMC* 1-4. It shows that increasing the number of solutions after 7-10 does not significantly change the NOx_{nm} value across the different sets of parameters tested. A key insight from this experiment is that for practical purposes, we can identify a set of, say 7 solutions and use one solution per day of the week to route the vehicles and obtain a better distribution of pollution compared to the *FFA* solution.

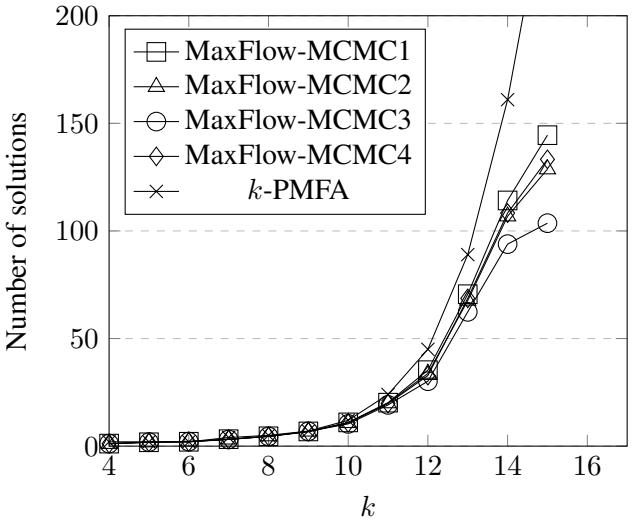

Figure 6.3: Comparison of k vs Number of solutions. The x-axis represents the maximum number of common edges between two max flow solutions and the y-axis represents the number of max flow solutions obtained in the k-optimal set. Parameters for the MaxFlow-MCMC$1-4$ are provided in Table 5.3. At k=0, the size of the solution set will be 1 since no common edges are allowed between two max flow solutions. The number of max flow solutions increases as we increase k. Plot also shows that the number of solutions provided by MaxFlow-MCMC is very close to that of k-PMFA when the number of solutions is small. Varying the parameters of MaxFlow-MCMC changes the size of the k-optimal set obtained

Runtime

Figure 6.2 shows the runtime comparison results between *MaxFlow-MCMC* and k*-PMFA*. Although *MaxFlow-MCMC* provides approximately the same number of solutions as k*-PMFA*, it is significantly faster. For example, for k = 10 the runtime needed for k*-PMFA* is 110.07 seconds while the slowest *MCMC* version has a runtime of 1.89 seconds. For values of k from 1 to 15, the slowest version of *MaxFlow-MCMC* (MaxFlow-MCMC1) showed a speedup of 65x on average, and the fastest version (MaxFlow-MCMC2) showed a speedup of 130x on average.

6.2 Pollution reduction in City Scale Road Networks

To demonstrate the scalability of our algorithms, we used four large real-world road networks that were used to evaluate the *MaxFlow-MCMC* and *EMF-MCMC* Algorithms, namely Seattle, Berkeley, Kanpur, and Islington (see Table 6.2). Table 5.2 provides details of the maps used. We used the respective parameters from Table 5.4 for all the simulations. For the Kanpur map, we set the value of

City	Traffic Policy	Length avg (m)	Pollution (NOx_{nm} in mg) avg	max * 10^{-3}	total * 10^{-4}
Seattle	FFA(BFS)	3270	2402	18.2	57.6
	FFA(Dij)	2133	1999	10.0	55.4
	MCMC(3)	3914 ± 27	856 ± 11	12.8 ± 0.9	77.3 ± 1.2
	MCMC(7)	4041 ± 24	512 ± 9	12.3 ± 0.8	83.6 ± 1.1
	k-MCMC(7)	3533 ± 12	432 ± 5	9.7 ± 0.5	85.9 ± 0.9
Kanpur	FFA(BFS)	4628	1152	8.8	48.0
	FFA(Dij)	4248	922	8.8	44.2
	MCMC(3)	4513 ± 4	597 ± 4	6.6 ± 0.1	50.6 ± 0.2
	MCMC(7)	4512 ± 4	444 ± 2	5.9 ± 0.1	50.7 ± 0.1
	k-MCMC(7)	4364 ± 6	266 ± 1	4.9 ± 0.1	53.9 ± 0.1
Berkeley	FFA(BFS)	8528	945	10.2	57.4
	FFA(Dij)	8001	718	8.5	54.3
	MCMC(3)	9457 ± 40	462 ± 4	6.7 ± 0.2	66.5 ± 0.4
	MCMC(7)	9456 ± 19	316 ± 2	5.7 ± 0.2	66.7 ± 0.2
	k-MCMC(7)	8628 ± 18	242 ± 1.2	6.9 ± 0.3	67.8 ± 0.2
Islington	FFA(BFS)	4723	2048	13.7	62.7
	FFA(Dij)	4930	2047	13.7	65.3
	MCMC(3)	4976 ± 10	1369 ± 5	8.0 ± 0.1	67.5 ± 0.2
	MCMC(7)	4945 ± 7	1150 ± 4	7.6 ± 0.0	67.0 ± 0.1
	k-MCMC(7)	4926 ± 6	1013 ± 3	8.4 ± 0.0	74.7 ± 0.1

Table 6.2: Comparison of lengths and pollution generated for different traffic policies and cities simulated in *SUMO*. Since our *MaxFlow-MCMC* and *EMF-MCMC* policy is randomized, we provide mean value and the standard error [14] from 30 runs. Avg. pollution decreases significantly in all cases. Max pollution also reduces in most cases, helping our objective of distributing the pollution better. There is some increase in distance travelled and hence the total pollution (summed up over all the links).

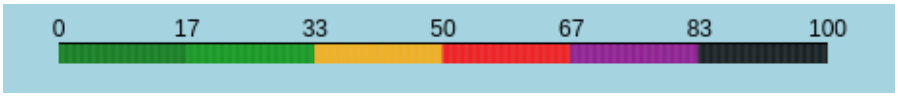

Figure 6.4: Legend for the heatmaps. Each edge is assigned a colour based on its pollution value relative to the highest pollution value in any edge among all the algorithms for the given map

Map	Maxflow-MCMC(s)	EMF-MCMC(s)	Speed Improvement
Berkeley	80726 ± 8083	3565 ± 24	22.64 times
Islington	18462 ± 2578	76 ± 0	242.92 times
Seattle	17600 ± 3698	163 ± 1	107.97 times
Kanpur	53086 ± 6609	2292 ± 9	23.16 times

Table 6.3: Runtime comparison between MaxFlow-MCMC and EMF-MCMC. We can observe that the EMF-MCMC algorithm is significantly faster than the MaxFlow-MCMC algorithm for all four maps

$\lambda = 0.95$ since it was used originally to test *MaxFlow-MCMC*, as $\lambda = 0.99$ was giving us longer paths. We use two versions of *FFA*, which use *Breadth First Search (BFS)* and *Dijkstra's* shortest path algorithm, respectively, to find the augmenting paths. The difference between the two versions of *FFA* is that *BFS* finds the shortest paths in terms of the number of edges, and *Dijkstra's* finds them in terms of total path length. We also use two versions of *MaxFlow-MCMC*, which samples 3 and 7 max flow solutions, respectively.

We can observe in Table 6.2 that *MaxFlow-MCMC* and *EMF-MCMC* distribute pollution better than both versions of the *FFA* algorithm. We can also observe that the performance of *EMF-MCMC* is similar to that of the *MaxFlow-MCMC* method but only at a fraction of the runtime as observed from Table 6.3.

6.3 Heatmaps

Figures 6.5 and 6.6 show the NOx heatmap for all four algorithms for the multi-source and single-source simulations, respectively. The heatmaps show that *EMF-MCMC* is similar to *MaxFlow-MCMC* and distributes pollution over a larger area. In addition, it also reduces the concentration of pollution in specific areas compared to *FFA*. In summary, the number of edges with higher values of pollution decrease with the usage of the *EMF-MCMC* solution.

6.4 Kernel Density Estimation plots

Kernel Density Estimation plots, also known as Parzen–Rosenblatt window method [29, 32], is a method that estimates the probability density of a distribution using samples. We use this method to plot the probability density of the pollution values of the edges in Figure 6.8. Note that the figure is plotted in Logarithmic scale. A higher probability density indicates that there are more edges with the given pollution value. We observe from the figure that *FFA* has a higher probability density at higher pollution values, and *EMF-MCMC* and *MaxFlow-MCMC* have a higher probability density at lower pollution values. This indicates that our algorithms are better at distributing pollution more evenly and preventing a high concentration of pollution in any specific area. *EMF-MCMC* and *MaxFlow-MCMC* also have a similar distribution, indicating that *EMF-MCMC* produces similar results while being significantly faster. .

6.5 Effect of hyperparameters

Figure 6.7 shows the relation between the k-optimality value and path length for *EMF-MCMC* algorithm for the four large city-scale maps. The data points were obtained by modifying the value of α. Initially, α was set to 1 and was reduced in intervals of 0.1 to obtain the data points. For all four maps, the experimental results supported our theoretical proof by showing that as α increases, the value of path length increases while the value of k-optimality decreases. We performed experiments with four different values for the number of iterations. The graph shows that we obtain better results as we increase the number of iterations, showing that our algorithm is indeed an anytime algorithm. Each colour represents a certain number of iterations in the figure, according to the legend at the top of the figure.

6.6 Runtime

Table 6.3 presents the runtime comparison results between *EMF-MCMC* and *MaxFlow-MCMC* for the results presented in Table 6.2. We can observe a significant improvement in runtime for all the maps with *EMF-MCMC* while the results are also favourable in terms of the mean pollution for all the maps. The *EMF-MCMC* algorithm is a minimum of 22 times faster and a maximum of 242 times faster in our experiments as observed from Table 6.3

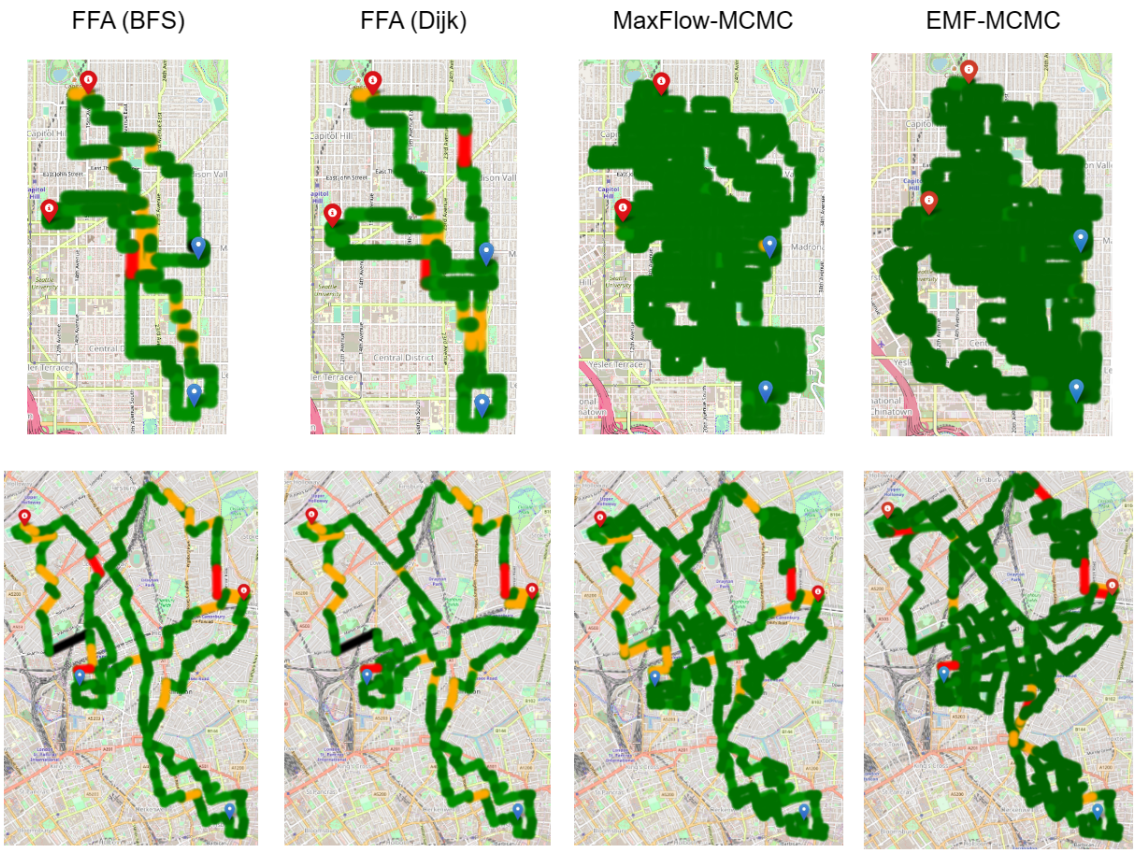

Figure 6.5: NOx heatmaps for different algorithms with multiple sources and sinks for Seattle and Islington. The colour for each edge is based on the legend in Figure 6.4. For both *MaxFlow-MCMC* [35] and *EMF-MCMC*, a set of 7 max flow solutions was used. We can see that *EMF-MCMC*, similar to *MaxFlow-MCMC*, distributes the pollution more evenly throughout the area compared to the FFA algorithms. Figure 6.8 provides the corresponding Kernel Density Estimation plots for this Figure, which shows the distribution of pollution values among the edges.

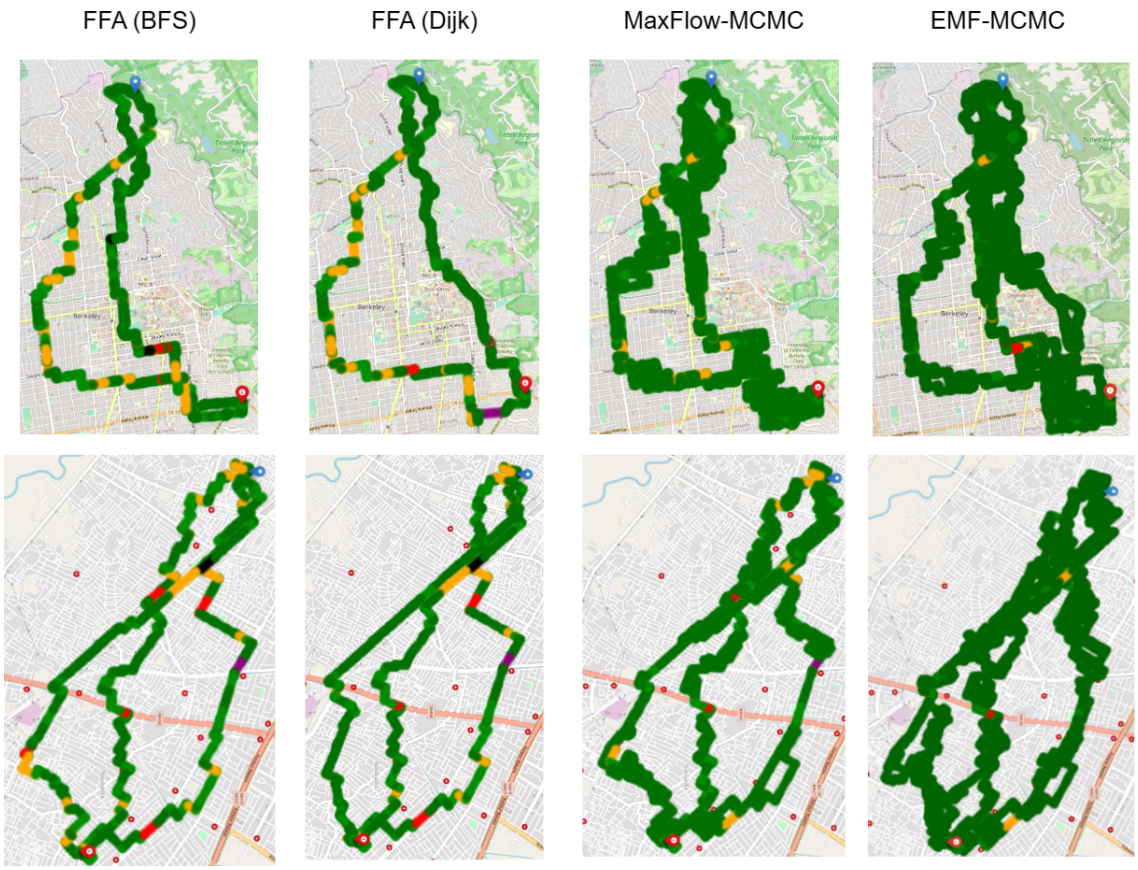

Figure 6.6: NOx heatmaps for different algorithms with a single source and sink for Berkeley and Kanpur. The colour for each edge is based on the legend in Figure 6.4. For both *MaxFlow-MCMC* [35] and *EMF-MCMC*, a set of 7 max flow solutions were used. The figure shows *EMF-MCMC*, similar to *MaxFlow-MCMC*, distributes the pollution more evenly throughout the area compared to the FFA algorithms. Figure 6.8 provides the corresponding Kernel Density Estimation plots for this Figure, which shows the distribution of pollution values among the edges.

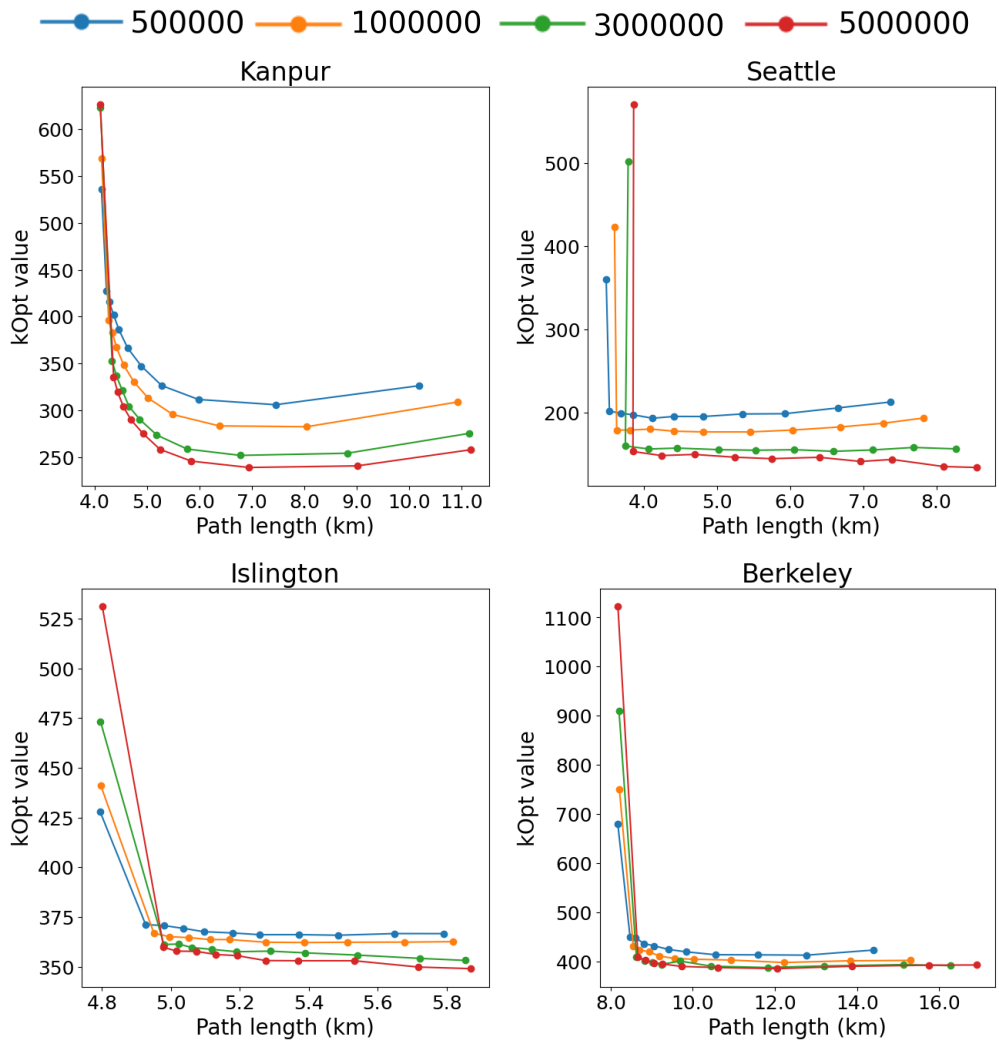

Figure 6.7: Plot of k-optimality value vs the Path Length for different values of the number of iterations. The x-axis represents path length in kilometres, and the y-axis represents the k-optimality value multiplied by a factor of $10^{-}2$. The colours represent the number of iterations according to the legend at the top of the figure. The data points (from left to right) were obtained by changing the α values from 1 in decrements of 0.1. We can see that as the value of α decreases, the k-optimality value decreases, and the path length increases. We can also see a significant reduction in k-optimality for a small increase in path length when α is decreased from 1 to 0.9.

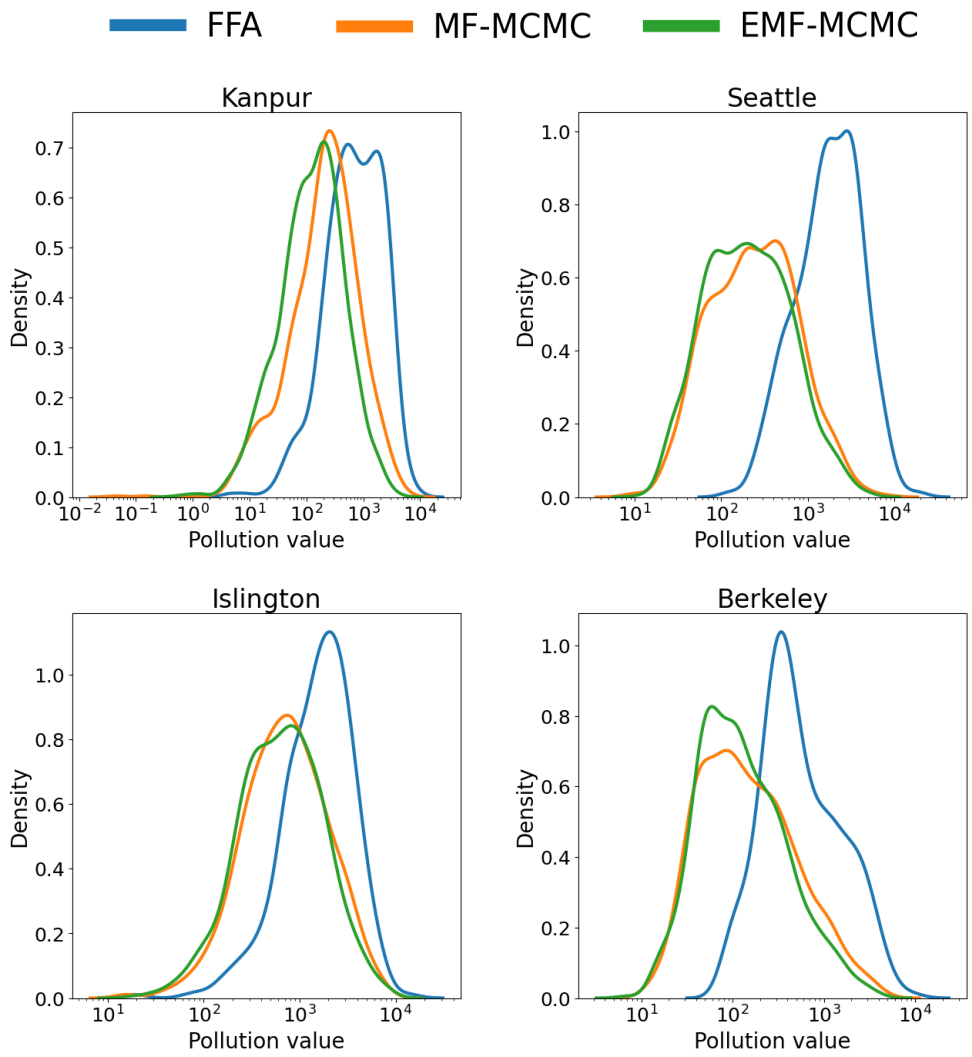

Figure 6.8: KDE plots in logarithmic scale for the Pollution values in edges for FFA(BFS), *MaxFlow-MCMC* [35] and *EMF-MCMC* corresponding to the Heatmaps shown in Figures 6.5 and 6.6. The x-axis represents the value of pollution, and the y-axis represents the probability density. We can see from the plots that *EMF-MCMC* and *MaxFlow-MCMC* have more edges with lesser pollution values, and FFA has more edges with higher pollution values.

Chapter 7

Conclusions and Future directions

7.1 Conclusions

In this thesis, we design two *Markov Chain* based algorithms that can prevent high concentrations of pollution in any area. We first design a Markov Chain to sample integer max flow solutions for a planar graph. We prove that the *Markov Chain* is ergodic and provide theoretical guarantees that the *Markov Chain* converges to a stationary distribution that gives a higher probability to max flow solution with a low path length. Since MCMC algorithms are very efficient, we are able to scale our experiments to large cities. We also design a *Markov Chain* that can directly sample a set of k-opt integer max flow solutions from a planar graph. We prove that the *Markov Chain* is ergodic and provide theoretical guarantees that the *Markov Chain* converges to a stationary distribution that contains max flow solution sets with a low path length and k-optimality value, based on the weights provided. We believe that the sampling method for integer max flows can be of independent interest in solving other combinatorial optimization problems.

We used both of the *Markov Chains* to build an urban traffic routing policy that prevents the concentration of pollution in specific areas. Unlike the existing methods, we show that the *MaxFlow-MCMC* method is scalable to larger graphs.We also show that the *EMF-MCMC* algorithm is a significantly faster method to generate a k-optimal solution set as compared to *MaxFlow-MCMC* method without compromising on the performance. We then tested the algorithm on 4 city-scale real-world road maps. We observed that both the *MaxFlow-MCMC* and *EMF-MCMC* algorithms perform better than existing methods when it comes to distributing the pollution over an area and preventing high concentration in any area. We also show that the *EMF-MCMC method* was 22 to 242 times faster than the *MaxFlow-MCMC* method.

7.2 Future Work

There are a few ways in which the world in this thesis can be expanded in the future.

- The policy can be expanded to non-planar graphs since real-world maps can contain flyovers
- The problem can also be formulated as a multi-commodity max flow problem where we have multiple sources and destinations with a fixed value of flow between each source and destination
- The methods can be modified with additional constraints. For example, public transport like buses require to have multiple stops between the source and destination

Related Publications

1. **Shreevignesh Suriyanarayanan**, Praveen Paruchuri, Girish Varma, *City-Scale Pollution Aware Traffic Routing by Sampling Max Flows Using MCMC* [35]. (Association for the Advancement of Artificial Intelligence - AAAI 2023, accepted as a full paper.) Link to paper: `https://arxiv.org/abs/2302.14442`

Bibliography

[1] S. Ahmed, M. Adnan, D. Janssens, and G. Wets. A route to school informational intervention for air pollution exposure reduction. *Sustainable Cities and Society*, 53:101965, 2020.

[2] C. Barrett, K. Bisset, M. Holzer, G. Konjevod, M. Marathe, and D. Wagner. Engineering label-constrained shortest-path algorithms. In *International conference on algorithmic applications in management*, pages 27–37. Springer, 2008.

[3] T. Bektaş and G. Laporte. The pollution-routing problem. *Transportation Research Part B: Methodological*, 45(8):1232–1250, 2011.

[4] G. Borradaile, P. N. Klein, S. Mozes, Y. Nussbaum, and C. Wulff-Nilsen. Multiple-source multiple-sink maximum flow in directed planar graphs in near-linear time. In *2011 IEEE 52nd Annual Symposium on Foundations of Computer Science*, pages 170–179, 2011.

[5] E. Bowring, J. P. Pearce, C. Portway, M. Jain, and M. Tambe. On k-optimal distributed constraint optimization algorithms: new bounds and algorithms. In *AAMAS (2)*, pages 607–614, 2008.

[6] R. Bubley. *Randomized algorithms - approximation, generation and counting*. Distinguished dissertations. Springer, 2001.

[7] P. Diaconis. The markov chain monte carlo revolution. *Bull. Amer. Math. Soc. 46, 179-205*, 2009.

[8] R. Dicker. 3 new ways to navigate more sustainably with maps. `https://blog.google/products/maps/3-new-ways-navigate-more-sustainably-maps/`, 2021. (Accessed on 07/24/2022).

[9] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[10] R. Durrett. *Probability: theory and examples*, volume 49. Cambridge university press, 2019.

[11] D. Easley, J. Kleinberg, et al. Networks, crowds, and markets: Reasoning about a highly connected world. *Significance*, 9(1):43–44, 2012.

[12] J. F. Ehmke, A. M. Campbell, and B. W. Thomas. Data-driven approaches for emissions-minimized paths in urban areas. *Computers & Operations Research*, 67:34–47, 2016.

[13] J. Erickson. *Algorithms*. 1999.

[14] M. Evans and J. Rosenthal. *Probability and Statistics: The Science of Uncertainty*. Macmillan Learning, 2010.

[15] L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian journal of Mathematics*, 8:399–404, 1956.

[16] G. Gualtieri, A. Crisci, M. Tartaglia, P. Toscano, and B. Gioli. A statistical model to assess air quality levels at urban sites. *Water, Air, & Soil Pollution*, 226(12):1–15, 2015.

[17] J. E. Gubernatis. Marshall Rosenbluth and the Metropolis algorithm). *Physics of Plasmas*, 12(5):057303, May 2005.

[18] M. Haklay and P. Weber. Openstreetmap: User-generated street maps. *Pervasive Computing*, 7(4):12–18, 2008.

[19] W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 04 1970.

[20] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[21] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[22] M. Hunter, A. Biswas, B. Chilukuri, A. Guin, R. Fujimoto, R. Guensler, J. Laval, H. Liu, S. Neal, P. Pecher, et al. Energy-aware dynamic data-driven distributed traffic simulation for energy and emissions reduction. *Handbook of dynamic data driven applications systems*, pages 467–487, 2018.

[23] M. Jerrum, L. Valiant, and V. Vazirani. Random generation of combinatorial structures from a uniform distribution. *Theor. Comput. Sci.*, 43:169–188, 1986.

[24] S. Kamishetty, S. Vadlamannati, and P. Paruchuri. Towards a better management of urban traffic pollution using a pareto max flow approach. *Transportation Research Part D: Transport and Environment*, 79:102194, 2020.

[25] Ç. Koç, T. Bektaş, O. Jabali, and G. Laporte. The fleet size and mix pollution-routing problem. *Transportation Research Part B: Methodological*, 70:239–254, 2014.

[26] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner. Microscopic traffic simulation using sumo. In *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018.

[27] R. Martin and D. Randall. Sampling adsorbing staircase walks using a new markov chain decomposition method. In *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 492–502, 2000.

[28] S. Montanari and P. Penna. On sampling simple paths in planar graphs according to their lengths. In G. F. Italiano, G. Pighizzini, and D. T. Sannella, editors, *Mathematical Foundations of Computer Science 2015*, pages 493–504, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[29] E. Parzen. On Estimation of a Probability Density Function and Mode. *The Annals of Mathematical Statistics*, 33(3):1065 – 1076, 1962.

[30] J. P. Pearce and M. Tambe. Quality guarantees on k-optimal solutions for distributed constraint optimization problems. In *IJCAI*, pages 1446–1451, 2007.

[31] C. A. Pope III and D. W. Dockery. Health effects of fine particulate air pollution: lines that connect. *Journal of the air & waste management association*, 56(6):709–742, 2006.

[32] M. Rosenblatt. Remarks on Some Nonparametric Estimates of a Density Function. *The Annals of Mathematical Statistics*, 27(3):832 – 837, 1956.

[33] S. Schneider and I. F. Sbalzarini. Finding faces in a planar embedding of a graph. https://www.semanticscholar.org/paper/Finding-faces-in-a-planar-embedding-of-a-graph-Schneider-Sbalzarini/7c01368fc62a3cad8ff7d5693aa37c3a66793d08, 2015. (Accessed on 2023-03-20).

[34] A. Schrijver. On the history of the transportation and maximum flow problems. *Mathematical programming*, 91(3):437–445, 2002.

[35] S. Suriyanarayanan, P. Paruchuri, and G. Varma. City-scale pollution aware traffic routing by sampling max flows using mcmc. (to appear in AAAI 2023), 2023.

[36] The Sage Developers. *SageMath, the Sage Mathematics Software System (Version 9.4)*, 2021. `https://www.sagemath.org`.

[37] J. Tomàs Vergés. *Analysis and simulation of traffic management actions for traffic emission reduction*. PhD thesis, TU Berlin, 2013.

[38] R. J. Trudeau. *Introduction to graph theory*. Courier Corporation, 2013.

[39] M. L. Waskom. seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60):3021, 2021.

[40] World Health Organization et al. *Ambient air pollution: A global assessment of exposure and burden of disease*. World Health Organization, 2016.

[41] R. Zhou and E. A. Hansen. Breadth-first heuristic search. *Artificial Intelligence*, 170(4):385–408, 2006.