City-Scale Pollution Aware Traffic Routing by Sampling Max Flows Using MCMC

Shreevignesh Suriyanarayanan, Praveen Paruchuri, Girish Varma

Machine Learning Lab, IIIT Hyderabad
sshreevignesh.s@research.iiit.ac.in, praveen.p@iiit.ac.in, girish.varma@iiit.ac.in

Abstract

A significant cause of air pollution in urban areas worldwide is the high volume of road traffic. Long-term exposure to severe pollution can cause serious health issues. One approach towards tackling this problem is to design a pollution-aware traffic routing policy that balances multiple objectives of i) avoiding extreme pollution in any area ii) enabling short transit times, and iii) making effective use of the road capacities. 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 observe a considerable decrease in areas with severe pollution when experimented with maps of large cities across the world compared to other approaches.

1 Introduction

Long-term exposure to high amounts of air pollution causes various health issues (Pope III and Dockery 2006). Data from WHO (World Health Organization et al. 2016) shows that 91% of the world's population lives in places where the pollution levels exceed the guideline limits. 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. Road traffic is considered one of the most significant contributors to air pollution in urban environments (Gualtieri et al. 2015). Studies have shown that people are willing to choose greener routes when credible information is provided (Ahmed et al. 2020).

Kamishetty, Vadlamannati, and Paruchuri (2020) propose the development of a transportation policy that distributes the traffic flow more evenly through a city to reduce the concentration of pollution in specific pockets. 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 differently on different days or for reasonable timelines. Here, *k-optimality* is a measure (Pearce and Tambe 2007) that is used to capture distinctness between solutions while *maximum flow* solutions (Ford and Fulkerson 1956) are generated to maintain the throughput of traffic network. However, existing work in this space can only be used for small areas due to computational scalability issues. We build upon this line of work to develop a significantly faster solution for pollution aware traffic routing using a *Markov Chain Monte Carlo (*MCMC*)* (Diaconis 2009; Hastings 1970; Gubernatis 2005; Bubley 2001) 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*). Our *Markov chain* extends the algorithm for sampling paths in planar graphs (Montanari and Penna 2015), and we provide proof of convergence to the stationary distribution, which assigns a higher probability to max flow solutions with shorter aggregate path length.

To compare our algorithm with the previous work, we simulated the algorithm on real-world road networks of mul-

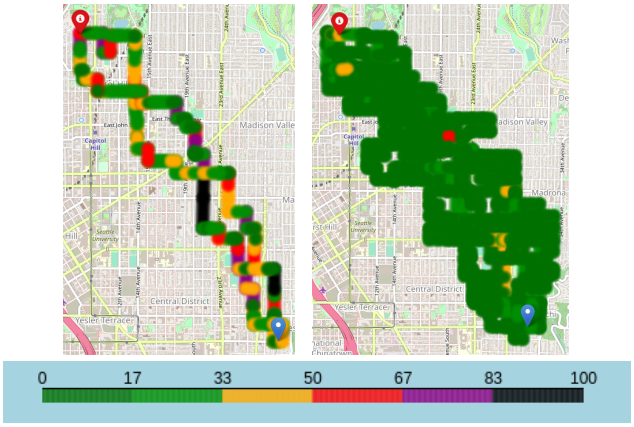

Figure 1: Pollution released on road links for traffic simulation on a large area of Seattle, USA. Colour legend given at bottom is the percentage of the max pollution. Left is obtained using only a single flow (*FFA*) and right is using our *MaxFlow-MCMC* which uses multiple diverse max flow solutions. As can be seen severe pollution (red, purple, black) is prevented.

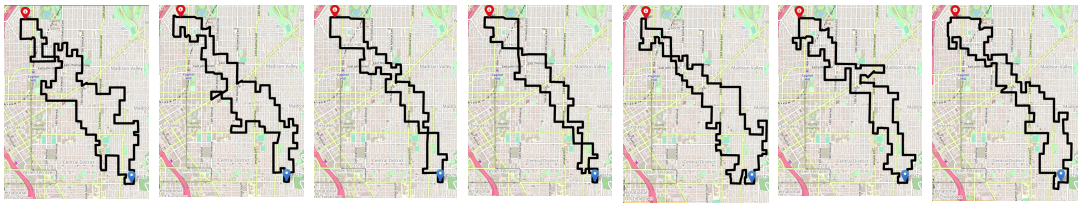

Figure 2: Seven diverse max flows samples from the *MaxFlow-MCMC* algorithm. Traffic is routed according to these and simulated in *SUMO* for equal intervals, resulting in the emission heatmap shown in Figure 1.

tiple cities. We use *Simulation of Urban Mobility (SUMO)* (Lopez et al. 2018), which is a traffic simulator in combination with *OpenStreetMap (OSM)* (Haklay and Weber 2008), an open-source tool that helps to model traffic settings on a real-world map. We also use the emission modeling capability of SUMO to evaluate the pollution levels generated by the different solution approaches, namely *MaxFlow-MCMC*, *k-optimal Pareto Max Flow Algorithm (k-PMFA)* (Kamishetty, Vadlamannati, and Paruchuri 2020) and *Ford-Fulkerson Algorithm (FFA)* (Ford and Fulkerson 1956). Our results show that *MaxFlow-MCMC* performs as well as the *k-PMFA* algorithm in terms of pollution severities while being scalable to large-scale cities. In particular, we obtained a 79% decrease in normalized mean pollution compared to *FFA* on a larger map corresponding to Seattle (with 18699 edges).

Our Contributions

1. We design a *Markov Chain* that can be used to sample integer max flow solutions from a planar graph with probabilities proportional to an exponential of the length of the solution (see Section 3).
2. We use the *MCMC* method to obtain a set of diverse max flows (k-optimal) and use it to give the first approach to designing a large-scale transport policy that avoids severe pollution while maintaining reasonable transit times. Traffic routing using such a set of max flows ensures that i) a diverse set of paths are used, which prevents concentration of pollution ii) vehicles are routed through shorter paths, and iii) capacity of the road network is fully utilized as it is a max flow (see Section 4).
3. We test our *MaxFlow-MCMC* traffic policies using the *SUMO* traffic simulator on real-world maps.The performance was compared with *k-PMFA* and *FFA* algorithms. We also demonstrate the scalability of *MaxFlow-MCMC* on large real-world road maps (see Section 6).

2 Related Works

Maximum Flow Problem. The *Ford-Fulkerson* algorithm (Ford and Fulkerson 1956), is a popular algorithm to compute the maximum flow between two given points in a network, including traffic networks (Schrijver 2002). However, it provides us with only a single max flow solution. We aim to develop an algorithm that will provide multiple max flow solutions between two points.

Pollution aware routing. There has been some work on the topic of pollution aware routing (Alam, Perugu, and McNabola 2018), (Boriboonsomsin et al. 2012). However, most of the previous work is focused on minimising the exhaust emissions and fuel consumption of a vehicle. In recent times, Google Maps has introduced eco-friendly routing, which suggests a path in a similar manner (Dicker 2021). Our work focuses on improving the health of residents who live near frequently used roads by reducing their long term exposure to high levels of pollution i.e., we do not explicitly aim to reduce the pollution but aim to distribute the pollution better.

k-opt Pareto Max Flow Algorithm (k-PMFA). The *k-Opt Pareto Max Flow* Algorithm (Kamishetty, Vadlamannati, and Paruchuri 2020), computes 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. One of the steps in the algorithm involves computing all the simple paths from the source to the destination. A simple path between two nodes is defined as a path where no node appears more than once (Easley, Kleinberg et al. 2012). Counting all the simple paths from a source to a destination was proved to be #P complete (Valiant 1979), which implies that finding all the simple paths would be an NP-hard problem. Our algorithm generates a k-optimal set without requiring us to find and store all the simple paths, and hence it will be more efficient in terms of time and memory requirements.

MCMC Method for sampling paths in a planar graph. There is extensive literature on using *MCMC* method for generating combinatorial structures (Jerrum, Valiant, and Vazirani 1986). Typically such generation gives a uniform distribution over structures. In our case, we need to sample from a non-uniform distribution (Martin and Randall 2000). Montanari and Penna (2015) designs an *MCMC* method for sampling simple paths in a planar graph. We construct a *Markov Chain* over integer Max Flows of a graph, with specific guarantees on stationery distribution. Our approach for sampling max flow solutions from a planar graph is inspired by them but requires additional proofs for showing the irreducibility of the *Markov chain*, which ensures that capacity constraints are not violated.

3 Sampling Integer Max Flows Using MCMC

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

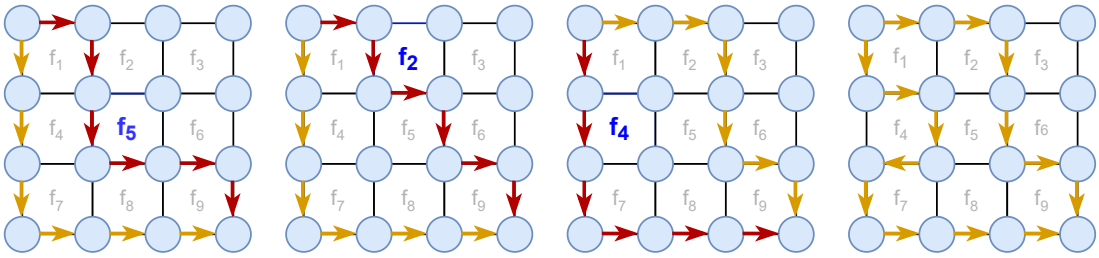

Figure 3: Example of a transition in the *Markov chain*. The underlying road network has 16 junctions forming a grid with max flow value from 1 to 16 to be 2. The state space consist of all max flows from 1 to 16, which can also be considered as a pair of $1-16$ paths (marked using arrows). The transition in M_{flow} given in Algorithm 1, 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.

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* (Erickson 1999), 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 more common edges, as shown in Figure 3. These transitions are inspired by (Montanari and Penna 2015), who built a similar *Markov Chain* for sampling paths. The transitions rules of M_{flow} are given in Algorithm 1.

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.

3.2 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 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. The proof uses standard arguments and is provided in appendix. [1]

□

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 (Durrett 2019). For Ergodicity, we need to show that it is i.) aperiodic and ii.) irreducible.

Aperiodicity. A *Markov chain* is said to be aperiodic if

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

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

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

That is, every state in state space can be reached from every other state with a finite number of steps. It is shown in Montanari and Penna (2015) 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.

[1] The extended version of the paper and code can be accessed at https://sshreevignesh.github.io/MCMCProjectPage/

Algorithm 1: $M_{\text{flow}}(x)$ defines a step of the *Markov Chain* on current state x. See Figure 3, 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

Definition (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.

Proposition. *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 proof is given in the appendix for clarity.

□

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 1, in the stationary distribution, the probability of a max flow solution x is proportional to $\lambda^{|x|}$.

4 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 Barrett et al. (2008). A set of max flow solutions is k-optimal when any two solutions from the set have at most k common edges.

We use the *Ford-Fulkerson Algorithm* (Ford and Fulkerson 1956) 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.

4.1 Implementing Policy in Simulation and Real World

There are multiple ways by which the *MaxFlow-MCMC* algorithm can be implemented in real world scenarios: (a) As described in Kamishetty, Vadlamannati, and Paruchuri (2020), traffic police can use a different solution to route the traffic on different days. For example, by identifying a set of 7 solutions, traffic can be routed using one solution per day of the week. A solution can be implemented by restricting the set of paths available for a vehicle to take and the number of vehicles on each path but does not restrict a vehicle's choice among the allowed ones. (b) GPS software(s) like Google maps can use to suggest different set of paths to different users on different days (Dicker 2021). (c) Autonomous cars can be routed using our method, where vehicles can be instructed to use different paths.

4.2 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 comes 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 modeling, congestion, etc., resulting in realistic traffic modeling.

4.3 Parameters for MaxFlow-MCMC

Sampling Frequency: The number of iterations after which we sample a max flow solution from the current distribution. To increase the number of solutions, we can sample the solutions more frequently, and if the aim is to improve the algorithm runtime, 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.

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 ← FFA(s,t), solutionset ← ∅
while iter ≤ num_iter + mix_iter do
    if iter > mix_iter and iter% sf == 0 then
        if KOpt(x,solutionset) then
            solutionset ← solutionset + x
        end if
    end if
    x ← M_flow(x) (See Algorithm 1)
    iter ← iter+1
end while
return solutionset
```

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.

4.4 Complexity Analysis of MaxFlow-MCMC

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 (Schneider and Sbalzarini 2015). 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 (Trudeau 2013), 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. Schneider and Sbalzarini (2015) 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.

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 a 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.

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 1: Parameter values used for experiments. The first four are used for ablation studies and the last two are for large scale experiments.

5 Experimental Details

The parameters for experiments on large scale graphs named *MaxFlow-MCMC Large*, are presented in Table 1. Since the *MaxFlow-MCMC* method is applicable only for planar graphs, 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 and sink nodes respectively, using infinite capacity edges (Borradaile et al. 2011). Algorithms were then used with vehicles traveling from the virtual source to the virtual sink.

We tested *MaxFlow-MCMC* for different values of k and simulated traffic using the *SUMO* simulator (Lopez et al. 2018). We used Sage Math (The Sage Developers 2021) to find the faces of the roadmap. Vehicles were released in waves with an interval of 5 seconds each for the small map and an interval of 15 seconds each for the larger map. They are released in waves so that a reasonable distance is created before the next wave of vehicles is allowed into the simulation. The number of vehicles per wave was equal to the maximum flow from the source to the destination. We increased the interval for the large map since it involved roads with different speed limits. Vehicles also have to slow down at junctions for turning, and a larger path can potentially involve a larger number of turns. We use NOx values (i.e., NO and NO_2 values) to measure the pollution due to their high dependency on traffic flow and their adverse impact on human health (Tomàs Vergés 2013). The plots were generated using matplotlib (Hunter 2007).

Due to the randomness involved in our algorithm, different runs using the same parameters may provide us with a different number of solutions. Hence, we use each solution of the generated k-optimal set for 1 hour and compute *Normalised Mean pollution* (NOx_{nm}) to compare the different runs of the algorithm accurately.

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

Map name	Sq. km	Edges	Nodes	s, t Pairs	k
Small map	0.03	37	25	1	9
Seattle	25.16	18699	14939	2	170
Berkeley	82.90	30808	24864	1	440
Kanpur	18.50	29956	20707	1	350
Islington	14.90	5382	2367	2	300

Table 2: Details of the maps used for the experiments. We use a small map to compare with *k-PMFA* since it is infeasible to run for larger maps.

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 total number of solutions 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 decrease 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.

For *MaxFlow-MCMC*, we set λ close to 1 since we measure the lengths of paths in meters. Given that the probability of our *Markov chain* moving from one state to another depends on the difference in their length, and if we set lambda to a smaller value like 0.5, the probability of moving to a state that is just 5 meters longer would be close to 0.03.

For comparison with previous work on *k-PMFA* Kamishetty, Vadlamannati, and Paruchuri (2020), we needed to make some modifications since the algorithm was designed for directed graphs. Hence, it can become inefficient to distribute the pollution in road networks containing lanes in both directions. This is because solutions generated by it can include flow in both directions of the road as part of two different paths, e.g., a max flow solution can contain one path that has a flow of 1 from node p to node q, whereas another path can have a flow of 1 from node q to node p. This will lead to increased pollution with no effective increase in capacity, which can be avoided. Hence, we extend the *k-PMFA* algorithm to undirected graphs by using *Breadth-First Search (BFS)* on the flow graph of the max flow solutions to generate the individual paths. This can help avoid the above situation while retaining the max flow solutions where the flows do not cancel each other.

Runtime results presented were measured on an Intel E5-2640 processor on an HP SL230s compute node. All the results were averaged over 30 runs.

6 Results

We benchmark the performance of *Maxflow-MCMC* policy on large-scale maps of many cities along with baseline algorithms. We also compare it with the previous work of *k-PMFA* (Kamishetty, Vadlamannati, and Paruchuri 2020) on a small map due to its scalability issues. All the networks were obtained from *OpenStreetMap* (Haklay and Weber 2008).

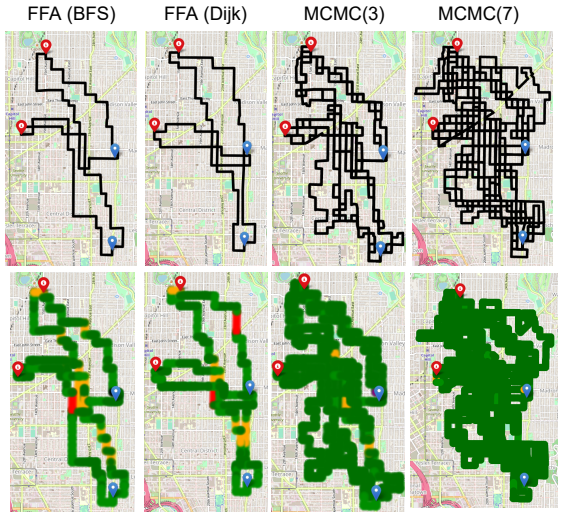

Figure 4: Maxflows (top) and pollution (bottom) generated for the different policies on the map of Seattle with two s, t-pairs. MCMC(k) denotes routing using k samples from *Markov Chain*. *MaxFlow-MCMC* with 7 samples prevents severe pollution (red, orange) from happening in any area. Legend of colors is same as Figure 1.

Pollution reduction in City Scale Road Networks. We benchmarked *MaxFlow-MCMC* on four larger real-world road networks, namely Seattle, Berkeley, Kanpur, and Islington, to demonstrate scalability (see Table 3). Table 2 provides details of the maps used. We used MaxFlow-MCMCLarge from Table 1 for all the simulations. For Kanpur map, we set the value of $\lambda = 0.95$ since $\lambda = 0.99$ 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 in terms of total path length. We also use two versions of *MaxFlow-MCMC*, which samples 3 and 7 max flow solutions, respectively.

In Table 3, the average pollution per link has decreased for all maps while there is some increase in the average distance traveled and total pollution summed up over all the edges. This is because the samples can be slightly longer than the shortest paths. However, the pollution is spread out because of the diverse samples, decreasing the max pollution in any link, which was our objective. It is also clear that increasing the number of max flow solutions sampled from 3 to 7 further prevents severe pollution. Figure 4 shows the NOx heatmap and the paths used by the algorithms for the multi-source multi-sink scenario of Seattle. The heatmap shows that while *Maxflow-MCMC* distributes pollution over a much larger area, it also reduces the concentration of pollution in specific areas compared to *FFA*. Plots for the other cities are provided in the Appendix.

Comparison with k-PMFA. *k-PMFA* (Kamishetty, Vadlamannati, and Paruchuri 2020) is a pollution-aware routing

City	Traffic Policy	Length avg(m)	Pollution (NOx_{nm} in g) avg	max	total
Berkeley	FFA(BFS)	8528	0.945	10.2	57.4
	FFA(Dij)	8001	0.718	8.5	54.3
	MCMC(3)	9457	0.462	6.7	66.5
	MCMC(7)	9456	0.316	5.7	66.7
Islington	FFA(BFS)	4723	2.048	13.7	62.7
	FFA(Dij)	4930	2.047	13.7	65.3
	MCMC(3)	4976	1.369	8.0	67.5
	MCMC(7)	4945	1.150	7.6	67.0
Seattle	FFA(BFS)	3270	2.402	18.2	57.6
	FFA(Dij)	2133	1.999	10.0	55.4
	MCMC(3)	3914	0.856	12.8	77.3
	MCMC(7)	4041	0.512	12.3	83.6
Kanpur	FFA(BFS)	4628	1.152	8.8	48.0
	FFA(Dij)	4248	0.922	8.8	44.2
	MCMC(3)	4513	0.597	6.6	50.6
	MCMC(7)	4512	0.444	5.9	50.7

Table 3: Comparison of path lengths and pollution for different traffic policies and cities simulated in *SUMO*. Since our *MaxFlow-MCMC* policy is randomized, we provide mean value from 30 runs. The standard error (Siddharth Kalla 2022) for all values was less than 7%. 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).

City	Traffic Policy	Length avg (m)	Pollution (NOx_{nm} in g) avg	max	total
Small Map	FFA(BFS)	213	4.387	9.6	61.4
	FFA(Dij)	198	4.6	19.9	73.9
	k-PMFA(7)	208	1.9	11.3	72.3
	MCMC(7)	207	1.8	11.8	69.1

Table 4: Comparing the pollution values of *k-PMFA* and *MCMC* for the small map with $k = 9$.

algorithm that we compare within Table 4. The table shows the performance of three algorithms, namely *MaxFlow-MCMC*, *k-PMFA*, and *FFA*. As shown, the *Maxflow-MCMC* gives comparable results while being scalable.

Effect of hyperparameters. Figure 5 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 1, 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.

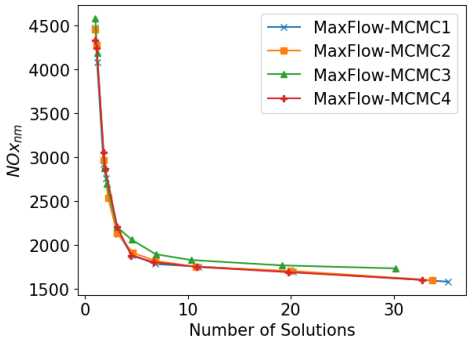

Figure 5: The change in normalised mean pollution reduces as we increase the number of solutions (obtained by increasing value of k). While there is significant reduction upto 10, the benefits are lesser later.

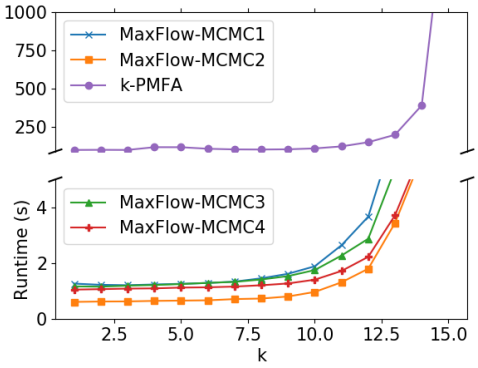

Figure 6: The runtime of *k-PMFA* is two orders of magnitude higher than *MCMC*, making it infeasible for road networks over a few sq km. Increasing the value of k increases the diversity but also the runtime, especially after $k = 10$.

Runtime. Figure 6 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.

7 Conclusion

We design a *Markov Chain* to sample integer max flow solutions for a planar graph. We used it to build an urban traffic routing policy that prevents the concentration of pollution. Since *MCMC* algorithms are very efficient, we are able to scale our experiments to large cities. We believe that the sampling method for integer max flows can be of independent interest in solving other combinatorial optimization problems. For future work, the policy can be expanded to non-planar graphs since real-world maps can contain flyovers and traffic between multiple sources and destinations.

References

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

Alam, M.; Perugu, H.; and McNabola, A. 2018. A comparison of route-choice navigation across air pollution exposure, CO2 emission and traditional travel cost factors. *Transportation Research Part D: Transport and Environment*, 65: 82–100.

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

Boriboonsomsin, K.; Barth, M. J.; Zhu, W.; and Vu, A. 2012. Eco-routing navigation system based on multisource historical and real-time traffic information. *IEEE Transactions on Intelligent Transportation Systems*, 13(4): 1694–1704.

Borradaile, G.; Klein, P. N.; Mozes, S.; Nussbaum, Y.; and Wulff-Nilsen, C. 2011. 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*, 170–179.

Bubley, R. 2001. *Randomized algorithms - approximation, generation and counting*. Distinguished dissertations. Springer. ISBN 978-1-85233-325-6.

Diaconis, P. 2009. The Markov chain Monte Carlo revolution. *Bull. Amer. Math. Soc. 46, 179-205*.

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

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

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

Erickson, J. 1999. *Algorithms*. Citeseer. ISBN 1792644833.

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

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

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

Haklay, M.; and Weber, P. 2008. OpenStreetMap: User-Generated Street Maps. *Pervasive Computing*, 7(4): 12–18.

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

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

Jerrum, M.; Valiant, L.; and Vazirani, V. 1986. Random Generation of Combinatorial Structures from a Uniform Distribution. *Theor. Comput. Sci.*, 43: 169–188.

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

Lopez, P. A.; Behrisch, M.; Bieker-Walz, L.; Erdmann, J.; Flötteröd, Y.-P.; Hilbrich, R.; Lücken, L.; Rummel, J.; Wagner, P.; and Wießner, E. 2018. Microscopic Traffic Simulation using SUMO. In *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE.

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

Montanari, S.; and Penna, P. 2015. On Sampling Simple Paths in Planar Graphs According to Their Lengths. In Italiano, G. F.; Pighizzini, G.; and Sannella, D. T., eds., *Mathematical Foundations of Computer Science 2015*, 493–504. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-662-48054-0.

Pearce, J. P.; and Tambe, M. 2007. Quality Guarantees on k-Optimal Solutions for Distributed Constraint Optimization Problems. In *IJCAI*, 1446–1451.

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

Schneider, S.; and Sbalzarini, I. F. 2015. 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. (Accessed on 2023-03-20).

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

Siddharth Kalla, L. T. W. 2022. Standard Error of the Mean. https://explorable.com/standard-error-of-the-mean.

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

Tomàs Vergés, J. 2013. *Analysis and simulation of traffic management actions for traffic emission reduction*. Ph.D. thesis, TU Berlin.

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

Valiant, L. G. 1979. The complexity of enumeration and reliability problems. *SIAM Journal on Computing*, 8(3): 410–421.

World Health Organization; et al. 2016. *Ambient air pollution: A global assessment of exposure and burden of disease*. World Health Organization. ISBN 9789241511353.