USING A CENTRAL ROUTE PLANNING SYSTEM TO OPTIMIZE TRAVEL TIME IN A NETWORK

ZIANG CHU, ZIYAN LIU, AILIN YU

ABSTRACT. As online route guidance systems have become ubiquitous alongside the proliferation of portable computers with internet connection, the feasibility of central planning to minimize congestion has been significantly increased. In this paper, we model traffic flow with the congestion effects to demonstrate the role of a central planner. We introduce a specific congestion model derived from fluid dynamics and implement an algorithm for optimization, and cost function.

1. INTRODUCTION

The study of traffic flow is concerned with how vehicles interact with each other on the road, in particular, congestion effects, resulting in slower speeds, longer trip times, and increased wait time. In order to minimize congestion and optimize traveling time for individuals and for the system, we consider the role of a central planner. As online route guidance systems have become pervasive and portable computers have become prevalent in everyday life, central planners would have access to various statistics including the origin, destination, current speed and the route each individual is currently on. Gathering all the information about the system, a central planner can suggest a route for everyone in the system.

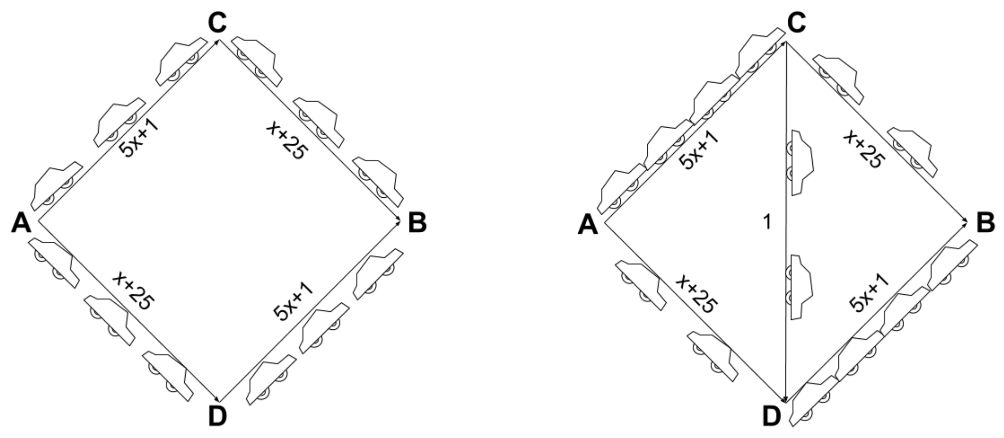

FIGURE 1. LEFT: Original Network, RIGHT: Additional Edge

A well known phenomenon, Braess' Paradox,(see [7] for a detailed explanation) explains how adding a new link to an existing transportation network might not improve the operation of the system since individuals act at the margin and cannot consider global effects. Adding new roads to a congested traffic system may surprisingly increase the total travel time. The paradox can be illustrated through Figure 1, which consists of a simple road network with four locations A, B, C and D and four road connecting these four locations. Vehicles are trying to get from A to B and will always choose the route that is the shortest in terms of travel time which depends on the density of traffic. The number x denote the number of vehicles currently using the roads. For example, assuming the time units are minutes, under the system on the left, since the routes ACB

and ADB are equally attractive, both takes $6x + 26$ minutes to get from A to B. Now suppose we have a total of six vehicles and there will be three vehicles on route ACB and three vehicles on route ADB, so $x = 3$ and the driving time for each of the vehicles would be $6 * 3 + 26 = 44$ minutes. Now we consider the system on the right in which there is an addition of a new road that only has a travel time of one minute. Notice that travel time on this new road is constant, fast and independent of density, so the route $ACDB$ would be highly attractive, but with the number of people on this route increases, the average driving time also increases, in our six vehicles case, if all six of them take this route, the travel time increases to 63 minutes and thus routes ACB and ADB becomes attractive again. Thus after some back and forth, the consequence of the additional road is that two vehicles will choose to travel the route $ACDB$, two vehicles will choose to travel the route ACB and two vehicles will choose to travel the route ADB. Thus, the travel time of these vehicles becomes 57 minutes which is higher than the original 44 minutes without the additional road. Therefore, congestion cannot necessarily be resolved by simply building more roads, and we should consider the problem from other aspects. Braess' paradox also reveals that allowing individuals to choose their own optimal paths might result in a more delayed situation for each individual than if a central planner optimized the routes. Thus, we want to figure out a central planning algorithm to reduce congestion in traffic systems.

In this paper, we draw from various existing models to present a simplified ideal fluid model for congestion. We analyzed various microscopic and macroscopic models such as the hydrodynamic model, the probabilistic model, the kinetic wave model, the Payne-Whitham model, the Aw-Rascle model and many other models [1, 3, 4, 9, 11, 12, 13, 14, 15, 8]. We also examined car-following models that utilizes algorithms that uses GPS to track vehicle locations such as Dijkstra's algorithm and Lee's algorithm [2]. GPS models are time dependent, thus they are more dynamic and complex. One existing model we have closely examined is a macroscopic model of traffic called a fluid dynamics model which involves the conservation law formulation proposed by Lighthill, Whitham and Richards (LWR) [10] [5]. This model involves the density of vehicles, traffic flow on a homogeneous highway and the equilibrium speed. The model is based on the continuity equation that describes the conservation of the number of vehicles on the road without entrances and exits. The LWR model also assumes that velocity is always in equilibrium. From the LWR model, we further assumed that that vehicles in the system behave as an ideal fluid. Thus, we can apply the mass flow rate equation for ideal fluids to model moving vehicle on the roads.

In Section 2, we formally discuss how we define the problem and present key concepts and notations. In Section 3, we explicitly explain how we build up our congestion model and the algorithm we use to optimize total travel cost. We use a simplified model to illustrate how our algorithm works and expand it to apply to more general cases. Finally in Section 4, we present examples that test our algorithms against looping, moving backward, and taking extremely expensive routes. We also tested the algorithm on a road map of Manhattan, which contains 17 stations.

2. Traffic System Problem

The main focus of the Traffic System Problem (TSP) is to find an approach to plan routes for vehicles such that the system's cost is minimized. In TSP, there is a road map that contains transportation stations and roads connecting theses stations with different width and length. These vehicles all have their own routes towards their destination and these routes are determined by a travel plan. The travel plan can be used to find the time expenses for each vehicle which are corresponding to a cost function defined on the whole system. The objective of TSP is to minimize the cost function subject to the road map and vehicles' travel plans.

Such a problem is continuous and difficult to study, so we make an assumption to discretize the problem. First, we assume that all the vehicles enter the roads at the same time, such that although they arrive at new stations at different times, they would have to wait on each other and enter the new roads together. In this way, we can discretize the problem. We divide this process into *steps* to keep track of the problem; one step starts when all vehicles enter the roads at the same time and ends when the last vehicle reaches the new station.

The input to the TSP is a tuple $\mathcal{S}$ in the following form.

Definition 2.1. $\mathcal{S} = \{G, R, OD, f\}$ consists of:

- $G = (V, E)$ is the road map in the system, where $V = \{s_1, s_2, ..., s_n\}$ is the set of all the transportation stations and E is the set of all the roads connecting these stations. For each road $e \in E$, $e = (s_i, s_j)$, meaning this road starts from s_i and ends at s_j.
- $R = \{r_{12}, r_{13}, ...r_{1n}\}$ is the set of all roads connecting these stations. For all $r_{ij} \in R, i \neq j$ and $(s_i, s_j) \in E$, $r = (v_m, l, d)$ where v_m stands for the maximum velocity that a vehicle can travel on the road subject to the speed limit, l is the width of the road in terms of the number of lanes and d is the length of the road.
- $OD = \{t_{11}, t_{12}, ...t_{nn}\}$ is the set of origin-destination pairs of all the vehicles in the system. For arbitrary $t_{ij} \in OD$, it means that there are t_{ij} vehicles that plan to travel from s_i to s_j.
- $f : \mathbb{P} \to [0, \infty)$ is the cost function where $\mathbb{P}$ is the set of all possible travel plans. f finds the total cost of the system based on the a specific travel plan P.

The output is a travelling plan P for the transportation system:

- $P = \langle P^1, P^2, ..., P^m \rangle$ is the sequence of travel plans in each *step.*

The optimization problem becomes:

$$\begin{aligned} \underset{P}{\text{minimize}} \quad & f(P) \\ \text{subject to} \quad & \text{is\_valid}(P) \end{aligned} \tag{2.1}$$

We will write P explicitly in mathematical form in next section and discuss the valid P based on various assumptions and situations.

3. Optimizing Traffic System Problem

In order to find ways to optimize the Traffic System Problem, we have to express its elements in mathematical formula and define a strict cost function for it. Since the cost function is based on the travelling plan P and is not simple to derive, we will discuss the expression and rationale in the first subsection of this part.
Below is the mathematical expression for each of the rest element in the tuple $\mathcal{S}$:

$$G = \begin{pmatrix} (v_{11}, l_{11}, d_{11}) & (v_{12}, l_{12}, d_{12}) & ... & (v_{1N}, l_{1N}, d_{1N}) \\ (v_{21}, l_{21}, d_{21}) & (v_{22}, l_{22}, d_{22}) & ... & (v_{2N}, l_{2N}, d_{2N}) \\ ... & ... & ... & ... \\ (v_{N1}, l_{N1}, d_{N1}) & (v_{N2}, l_{N2}, d_{N2}) & ... & (v_{NN}, l_{NN}, d_{NN}) \end{pmatrix}$$

where v_{ij} stands for the maximum speed between the point i and j, l_{ij} represents the width and d_{ij} denotes the distance. If i and j are not connected, then all three entries would be set to -1; if $i = j$ then $d_{ij} = 0$.

$$OD = \begin{pmatrix} n_{11} & n_{12} & ... & n_{1N} \\ n_{21} & n_{22} & ... & n_{2N} \\ ... & ... & ... & ... \\ n_{N1} & n_{N2} & ... & n_{NN} \end{pmatrix}$$

where n_{ij} means that number of people staying at point i and want to go to point j.

3.1. **Cost Function.**

To make an abstract TSP problem concrete, we have to strictly define a cost function f based on the travel plan P. To start with, we firstly have to define the travel plan P mathematically. As we have mentioned in our model, P has multiple steps, so we only define one particular step in the whole process and it can be generalized to all the steps. After that, we can explore our cost function. The cost function in TSP, in the most intuitive way, should be related to the time travelling for each vehicle. So, before we come to a concrete formula for cost function, we will first discuss how to find the travel time for each vehicle on the road.

3.1.1. *Travelling Plan.*

As we have defined earlier in the TRAVEL SYSTEM PROBLEM in Section 2, a travelling plan is a step by step sequence of travel plans: $P = \langle P_1, P_2, ..., P_m \rangle$. The plan for one step, say P^t, is

$$P^t = \begin{pmatrix} (P^t_{111}, P^t_{112}, ..., P^t_{11N}) & (P^t_{121}, P^t_{122}, ..., P^t_{12N}) & ... & (P^t_{1N1}, P^t_{1N2}, ..., P^t_{1NN}) \\ (P^t_{211}, P^t_{212}, ..., P^t_{21N}) & . & ... & . \\ (P^t_{311}, P^t_{312}, ..., P^t_{31N}) & . & ... & . \\ . & . & ... & . \\ . & . & ... & . \\ . & . & ... & . \\ (P^t_{N11}, P^t_{N12}, ..., P^t_{N1N}) & (P^t_{N21}, P^t_{N22}, ..., P^t_{N2N}) & ... & (P^t_{NN1}, P^t_{NN2}, ..., P^t_{NNN}) \end{pmatrix}$$

where P_{ijk} means the proportion of people who are currently at s_i with final destination s_j and are sent to s_k in the this step; therefore, $\sum_{k=0}^{N} P_{ijk} = 1$.

3.1.2. *Congestion Model.*

In order to calculate the travel time for each vehicle on the road, we have to know both the distance and the speed as shown in equation (3.1). Here we derive a simple congestion model to find the speed. We can consider a continuous model for congestion based on the mass flow rate equation (3.2) for ideal fluids, namely

$$t = \frac{d}{v} \tag{3.1}$$

$$\rho A v = c \tag{3.2}$$

where

- ρ is the density,
- A is the cross sectional area,
- v is the velocity,
- c is the constant.

This can be applied to traffic flow where ρ corresponds to the density of cars per lane and v refers to the average velocity of vehicles travelling. Also, notice that A is constant because the number of lanes stays the same on all roads. Thus for any ρ, v, we have a constant c':

$$\rho v = \frac{c}{A} = c'$$

Therefore, speed has an inverse relationship with ρ.

$$v = \frac{c'}{\rho} \tag{3.3}$$

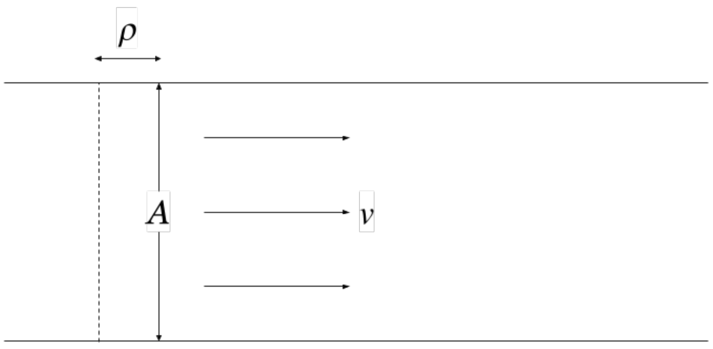

FIGURE 2. Congestion Model

From the number of vehicles on the road, y, and number of lanes, l, on each road, we could derive ρ as

$$\rho = \frac{ky}{l} \tag{3.4}$$

where k is some constant. Ideally, when $y \leq l$, i.e. there are fewer than one vehicle on each lane, the road is not congested and all the vehicles should move at the maximum speed, v_m. For $y > l$, from equations (3.3) and (3.4), we have $v = \frac{c'}{k}\frac{l}{y}$. By continuity, $\frac{c'}{k} = v_m$. Thus v becomes a piecewise function as follows,

$$v = \begin{cases} v_m & y \leq l \\ v_m \frac{l}{y} & y > l \end{cases} \tag{3.5}$$

Notice we can write v with the function $f(x)$, s.t. $v = v_m \cdot f(x)$

$$f(x) = \begin{cases} 1 & 0 \leq x \leq 1 \\ \frac{1}{x} & x > 1 \end{cases} \tag{3.6}$$

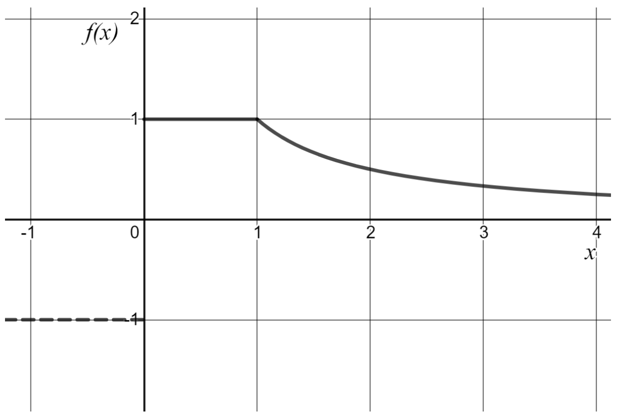

FIGURE 3. $f(x)$

From (3.6), we can get that the instant average speed of each road, which changes with respect to the number of vehicles on that road and thus we can derive cost of travel in terms of time spend traveling on that road.

However in later work, we found out that due to the in-continuity of the congestion function at $x = 1$, our algorithm to find travel plans does not give us optimal results. The optimization process was extremely slow and reported negative values which caused errors in the result. By Theorem 7 of [6], under the assumption that the gradient is

Lipschitz continuous, the derived solution of regularized problem converges globally to the original problem. Thus in order to allow our problem to converge to an optimal solution, we have to regularize (3.6) to make it $C^{1,1}$. We regularize (3.6) through the use of cubic splines to obtain the function in equation (3.7) such that the function's gradient is Lipschitz continuous. Since we have two fixed points, $(0,1)$ and $(1,1)$, and we want specific slopes at these two points, we need 4 degrees of freedom. Therefore, we use the polynomial $y = ax^3 + bx^2 + cx + d$ to be the regularized function. We know that $c = 0$ and $d = 1$ by plugging in $f(0) = 1$ and $f'(0) = 0$. To avoid the function from growing over 1, we want the cubic function to reach maximum when $x \leq 0$, and from this constraint, we get $b = 0$. Then, by setting the cubic function and $\frac{1}{x}$ equal and their respect derivatives equal, we obtain the value of a. From the function and Theorem 7 of [6], our results we obtain from our algorithm will be optimal.

$$g(x) = \begin{cases} -1 & x < 0 \\ -\frac{(\frac{3}{4}^3)}{4}x^3 + 1 & 0 \leq x \leq \frac{4}{3} \\ \frac{1}{x} & x > \frac{4}{3} \end{cases} \tag{3.7}$$

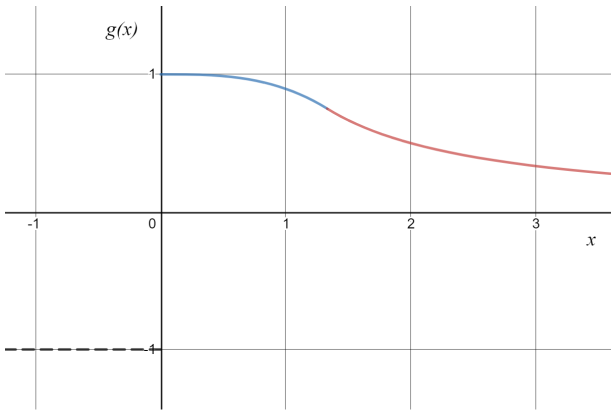

FIGURE 4. $g(x)$

3.1.3. *Cost Function.*

Now we have a strict definition for a travel plan in one step P^t and the congestion model, we can derive our cost function based on it. Using OD and P^t, we can derive the new origin-destination matrix OD_{new} and travelling N for this step, where OD_{new} stands for the new OD matrix after the step and X is the matrix representing number of vehicles traveling on each road during this step.

$$N = \begin{pmatrix} x_{11} & x_{12} & \dots & x_{1N} \\ x_{21} & x_{22} & \dots & x_{2N} \\ \dots & \dots & \dots & \dots \\ x_{N1} & x_{N2} & \dots & x_{NN} \end{pmatrix}$$

s.t. $x_{ij} = \sum_{k=0}^{n} n_{ik} \cdot p_{ikj}$. Where n_{ik} is in OD which means the number of people at s_i planning to travel to s_k and p_{ikj} is the proportion of these people who head to s_j in this step.

$$OD_{new} = \begin{pmatrix} n'_{11} & n'_{12} & \dots & n'_{1N} \\ n'_{21} & n'_{22} & \dots & n'_{2N} \\ \dots & \dots & \dots & \dots \\ n'_{N1} & n'_{N2} & \dots & n'_{NN} \end{pmatrix}$$

Where $n'_{ij} = \sum_{k=0}^{n} n_{kj} \cdot p_{kji}$.

Use N to determine the cost, we have multiple choices for our cost function

1. Total cost for all vehicles.
2. Total cost for the system.
3. Maximum cost for all vehicles.
4. L2 norm cost for all vehicles.

...

However, in our paper, we currently only consider *total cost for all vehicles*, where in each step

$$cost(P) = \sum_{i,j=1}^{N} x_{ij} \cdot max\left(\frac{d_{ij}}{(v_{ij} f\left(\frac{x_{ij}}{l_{ij}}\right)}, c\right)$$

where c is a preset constant value representing the minimum cost for each step, i.e. even when vehicles wait at a station at a particular step, they still have some costs.

3.2. **2-Step Simplified Model.**

While considering to apply the model to a more complex system, we decided to first consider a 2-step model to illustrate our optimization algorithm more clearly. By 2-step model, it means that all of the people travel at most two *steps* in the whole process, thus we have following assumptions in this subsection:

Assumption 1. *The graph G is complete.*

Assumption 2. *People travel at most 2-steps before arriving at their destinations.*

Based on these assumptions, we could build the 2-step model naturally since every vertex can achieve another in no more than 2 steps. We will mathematically define the special case below and solve it accordingly.

3.2.1. *Formulate the Problem.*

In this special setting, we require a two-step travelling plan, with P^1 and P^2, to complete our two-step model. Notice that P^2 depends solely on P^1 because after the first step, all people have to get to their destination at the second step, thus we can find P^2 with P^1 and OD. For each vehicle, based on its origin s_i and destination s_j, we pick an intermediate station s_k for it in P^1, and this vehicle goes from s_i to s_k in the first step and then s_k to s_j. Since the travel plan P^1 can solely determine the system's operation, we set $P = P^1$ in this special case.

The travelling matrix N^1, N^2 in both steps with each entry represented as x^1_{ij}, x^2_{ij} can be found by:

$$\begin{aligned} x^1_{ik} &= \sum_j p_{ijk} \cdot n_{ij} \\ x^2_{kj} &= \sum_i p_{ijk} \cdot n_{ij} \end{aligned} \tag{3.8}$$

Then the costs C^1, C^2 in two steps become

$$\begin{aligned} C^1 &= \sum_{i,k=1}^{N} x^1_{ij} \cdot max\left(\frac{d_{ik}}{v_{ik} f\left(\frac{x^1_{ik}}{l_{ik}}\right)}, c\right) \\ C^2 &= \sum_{k,j=1}^{N} x^1_{kj} \cdot max\left(\frac{d_{kj}}{v_{kj} f\left(\frac{x^2_{kj}}{l_{kj}}\right)}, c\right) \end{aligned} \tag{3.9}$$

Then we can calculate the total cost $C = C^1 + C^2$.

3.2.2. *Finding Optimal Travel Plan.*

Now that we have found our cost which depends on the preset G and OD and the variable P, we can proceed to find the optimal travel plan utilizing a similar algorithm as in [6]. In order to optimize our travel plan P, in each step, we have to solve N quadratic programming problems to find the direction for optimization and then change our P in this direction by a small step α until P converges.

Before performing the quadratic programming step, we have to first find the gradient of the cost with respect to P, say $\nabla_P C$. Due to the high order of G, multi-dimensional summation, as well as the addition of the congestion model, it becomes extremely complex to find the gradient of the cost function. Therefore, we decide to use automatic differentiation which computes derivatives at each arithmetic operation and can be used to update derivatives automatically as we change the objective function. Its flexibility allows us to use different congestion models or other cost functions conveniently in the future. By applying automatic differentiation, we have the partial differentiation of C with respect to our plan P, $\nabla_P C$, which is a $N \times N \times N$ matrix such that

$$\nabla_P C_{ijk} = \frac{\partial C}{\partial p_{ijk}}$$

Then, we can use $\nabla_P C$ to set up a Quadratic Programming formulation and update the optimum P in each step. Below we will discuss how we formulate the quadratic programming problem and how we proceed after this. The pseudo-code below shows our algorithm.

Algorithm 1: Finding Optimal Travel Plan

```
  Input: G, OD ,P
  Output: $P_{optimal}$
1 d ← [1,0,0,...0];
2 while $max(d) > \sigma$ do
3     $P_{new}$ ← $qp$(P);
4     d←$P_{new}$ - P;
5     $\alpha$← $findStepSize$(P,d);
6     P ← P+ $\alpha \cdot d$;
7 end
8 $P_{optimal}$ ← P ;
```

3.2.3. *Quadratic Programming.*

As long as we find $\nabla_P C$, we can use it to define quadratic programming formulations to proceed our algorithm. P is the target tensor which has dimension $N \times N \times N$ and in the quadratic programming problem, we study each P_{ij} separately, s.t. $|P_{ij}| = n$. Here, P_{prev} stands for the optimized P found at last step. We define a quadratic programming formulation for $\forall i, j \in [N]$, it can be expressed as:

$$\begin{aligned} \underset{P_{ij}}{\text{minimize}} \quad & \nabla_P C_{ij} P_{prev.ij} + \|P_{prev.ij} - P_{ij}\|_2 \\ \text{subject to} \quad & \sum_{k=1}^{n} P_{ijk} = 1 \\ & P_{ijk} \geq 0, \forall k \in [N] \end{aligned} \tag{3.10}$$

3.2.4. *Update Travel Plan.*

From each of the quadratic programming problem, there exists a solution $\hat{P_{ij}}$; put them in one matrix and call it $\hat{P}$. Find our search direction by $d = \hat{P} - P_{prev}$ and we should update our P in the new step using $P_{new} = P_{prev} + \alpha d$. We want to find $\alpha \in [0, 1]$ that minimizes $cost(P + \alpha d)$. We provide pseudo-code for our algorithm below:

```
Algorithm 2: Find Step Size
  Input: P, d
  Output: $\alpha$
1 $\alpha \leftarrow 1$ ;
2 $\tau, \theta \leftarrow 0.9, 0.5$ ;
3 while $cost(P + \alpha d) \geq cost(P) + \tau \cdot \alpha d cost'(P)$ do
4     $\alpha \leftarrow \alpha \cdot \theta$;
5 end
```

Then we can update the optimal travelling plan P and do another round of optimization as shown above.

3.3. **Multi-Step Model.**

Since real life travel routes are not constrained in the number of roads that a vehicle may take, in order for our model to fit better in the realistic situations, we expand our model to apply to more complex travel plans. Also, if some roads in the system are extremely congested, in order to avoid congestion, it might be worth it to travel more steps in order to obtain the minimum travel time. Thus we propose the multi-step model which is more applicable and practical.

Majority of our model and algorithm follows exactly from our simplified model. And still, we assume the graph to be complete. Similar to the simplified model, a *k-step* model requires $k - 1$ steps of travel plans to start with and the k^{th} plan will depends on the $k-1$ plans. Therefore, our plan will be $(k-1) \times N \times N \times N$. $P = (P^1, P^2, ..., P^{k-1})$, where P^i is in the same form as P^1 in the previous section.

In our optimization process, the initial plan is that each vehicle directly travels to their destinations in the first step and stays in the remaining steps.

3.4. **General Graph.**

In the above models, we assumed that our graph is fully connected so that destinations could be reached very easily, even directly. However in the real world, road systems are usually not fully connected. Cars and buses need to go through a series of intermediate stops to arrive at their final destinations. Further, the situations when some roads are sometimes under construction or blocked also force us to consider the problem of incompleteness.

Since the graph is incomplete, we cannot directly send every vehicle to its destination and thus need to change the way we generate the initial travel plan that we use in the optimization problem. Here, we find the shortest path for each vehicle and then follow the shortest path to assign where it goes in each step in initial P. For instance, for those leaving from B to D in Figure 5, the shortest paths may be $B - A - D$ and $B - C - D$ based on their length, width and maximum velocity. In the initial plan, the vehicles cannot be sent to D directly, but need to be sent to either A or C in the first step, and then to D.

Now that the graph is incomplete and we have to send all vehicles to their destinations in limited steps, we still have to put an assumption here:

Assumption 3. *The diameter of graph G, meaning the longest step-distance between any two points in the graph, must be less or equal to the number of steps in the plan.*

The non-negativity and simplex constraints for quadratic programming still hold, but we must add conditions to make sure that when an edge does not exist in the graph,

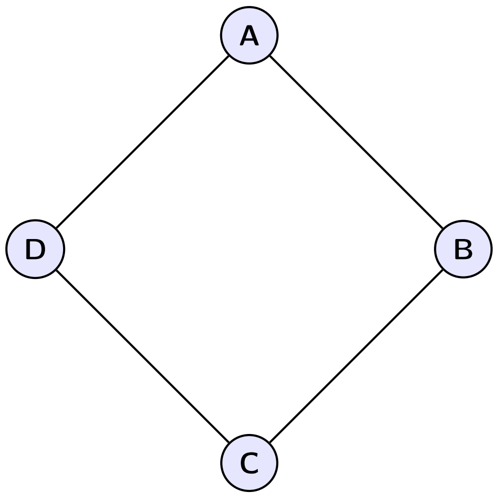

FIGURE 5. Incomplete $n = 4$

or the edge cannot send vehicles to destination in the remaining steps, vehicles do not take that edge. So now, the quadratic programming problem formulation for each $t \in [MaxStep], i, j \in [N]$, where t stands for the current step in the travel plan P, becomes

$$\begin{aligned} \underset{P^t_{ij}}{\text{minimize}} \quad & \overline{C_{ij}} P^t_{prev.ij} + \|P^t_{prev.ij} - P^t_{ij}\|_2 \\ \text{subject to} \quad & \sum_{k=1}^{n} P^t_{ijk} = 1 \\ & P^t_{ijk} \geq 0, \forall k \in [N], \forall t \in [MaxStep] \\ & P^t_{ijk} = 0 \text{ if } (s_i, s_k) \notin E \\ & P^t_{ijk} = 0 \text{ if shortestPath}(s_k, s_j) > \text{MaxStep} - t \end{aligned} \tag{3.11}$$

Different from above situations, we cannot put the quadratic programming problem in to standard form by directly adding more constraints. Rather, in order to do this, we have to change the dimension of the quadratic programming optimization formulation. For entries where $P^t_{ijk} = 0$, we choose not to put them in the formulation and thus for different t, i, j, k, we have to solve a quadratic programming problem in different sizes. For the rest parts, do exactly same thing as we did in the previous sections.

4. Experiments and Results

In order to test the performance of our algorithms, we conduct some experiments within both artificial and real-world context. In the following examples, we will show some interesting facts, compare our optimal travelling plan versus the naive route choices.

4.1. Experiments and Tests.

Before we conduct our experiments, we have to define what we call as a *naive route plan.* Basically it means that each individual chooses the most cost-efficient route in the graph as if no one else is travelling.

Definition 4.1. Define *naive route plan* as a sequence of stations in the graph, $S = \langle s_1, s_2, ..., s_p \rangle$, s.t.

$$\begin{aligned} \underset{S}{\text{minimize}} \quad & \sum_{i=1}^{p-1} d(s_i, s_{i+1}) / v(s_i, s_{i+1}) \\ \text{subject to} \quad & (s_i, s_{i+1}) \in E, \forall i \in [p-1] \end{aligned} \tag{4.1}$$

The *naive cost* means the total cost of travelling where everyone picks *naive route plan.*

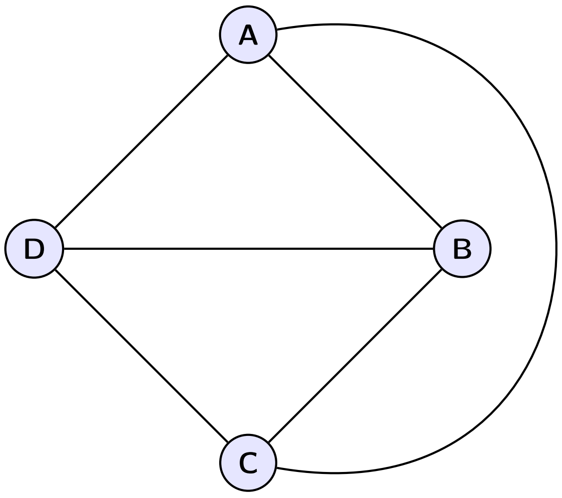

FIGURE 6. Complete $n = 4$

Example 1. *Divide up Vehicles*

We first test our algorithm on a small complete graph where $n = 4$ as in Figure 6. The purpose of this example is to see whether our plan tries to divide up the vehicles onto multiple routes. We set up the graph with the property such that each of the roads are weighted equally in terms of distance, number of lanes and maximum velocity. In this way, we could expect that vehicles would go to their destinations via the direct route in most cases. We set OD to be a matrix with diagonals being zeros. The non-zero entries in the same row are equal so that the result becomes more explicit to interpret. So from each location, we sent the same number of people to the other locations.

$$OD = \begin{pmatrix} 0 & 5 & 5 & 5 \\ 0 & 0 & 5 & 5 \\ 0 & 0 & 0 & 5 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

$$P^1 = \begin{pmatrix} (1,0,0,0) & (\frac{2}{3},\frac{1}{3},0,0) & (\frac{2}{3},0,\frac{1}{3},0) & (\frac{2}{3},0,0,\frac{1}{3}) \\ (1,0,0,0) & (0,1,0,0) & (0,\frac{2}{3},\frac{1}{3},0) & (0,\frac{2}{3},0,\frac{1}{3}) \\ (1,0,0,0) & (0,1,0,0) & (0,0,1,0) & (0,0,\frac{2}{3},\frac{1}{3}) \\ (1,0,0,0) & (0,1,0,0) & (0,0,1,0) & (0,0,0,1) \end{pmatrix}$$

For all the vehicles starting from A and going to D, our results indicate that $\frac{1}{3}$ of them are sent in each step. Since having all of them go in the same step will result in heavy congestion, it makes sense for the travel plan to split up the vehicles to go in separate steps. And from this fraction, we can notice that the proportion of people sent through each step is inversely related to the step size that we set up. In this model, we set up the cost of waiting as the minimum cost of travelling in one step and the total cost of the optimized travel plan is 66.6, while the *naive cost* is 125.0.

Example 2. *Avoid High Cost Passages*

We then tested our algorithm on the same graph as in Figure 6. But this time, we want to discourage vehicles to take AC edge by setting the edge to have an extremely high cost. For all vehicles leaving from A to C, we expect our algorithm to divide them up into two streams, going to B and D instead of taking the route to C directly. The goal of this example is to test if our algorithm creates a plan that can avoid high cost routes. We set the OD matrix to be a matrix such that all the vehicles have origin A and destination C. When we set the number of steps to be 2, we found that the vehicles will be diverted into two different paths B and D, and avoiding the high cost route AC as expected. When the number of steps are larger than 2, the plan still divides all the vehicles by half onto B and D, and also reduces travel cost by sending a proportion of them to their final destination in each remaining step.

Example 3. *Avoid Cycles*

Next, we tested our algorithm on a triangular graph. The goal of this example is to test if the algorithm creates a plan that involves cycles. By assigning the number of steps a large number, m, we wish to see that the result of our algorithm will send the vehicles in batches rather than taking them into cycles. We found that our result simply send a proportion of vehicles, $\frac{1}{m}$, at each step.

Example 4. *Incomplete Graph*

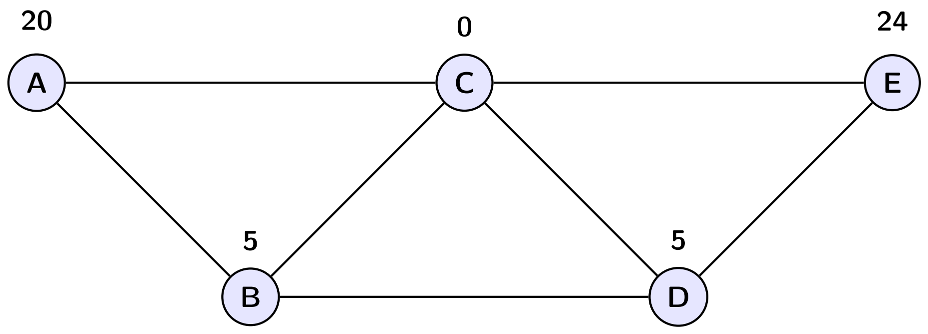

Figure 7. Example 4 Initial

Then, we test an incomplete graph with 5 nodes that have relatively the same cost edges. The graph is defined to be as in Figure 7. Since the diameter of the graph is 2, i.e. every vertex can reach the other in 2 steps, we set the maximum step to be 2. As we see in the results, for all vehicles leaving from A and going to D, most of them ($\frac{7}{10}$), are assigned to B in the first step. Our algorithm also assign all (A, E) vehicles to C since ACE is the only 2-step path. We can also see that the algorithm takes both steps into consideration. It does not split the stream from B to E in the first step, because if it does, the road CE will be very congested in step 2. The total cost for the 2-step example from our optimization process is 135.68 while the naive solution will cost up to 195.20.

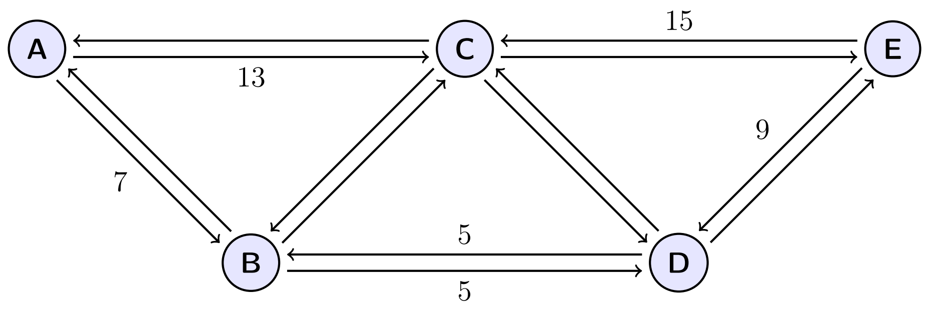

Figure 8. Example 4 Two-Step Step 1

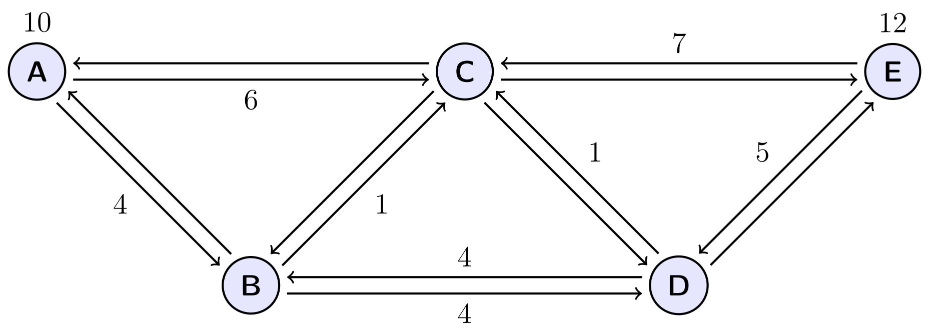

FIGURE 10. Example 4 Three-Step Step 1

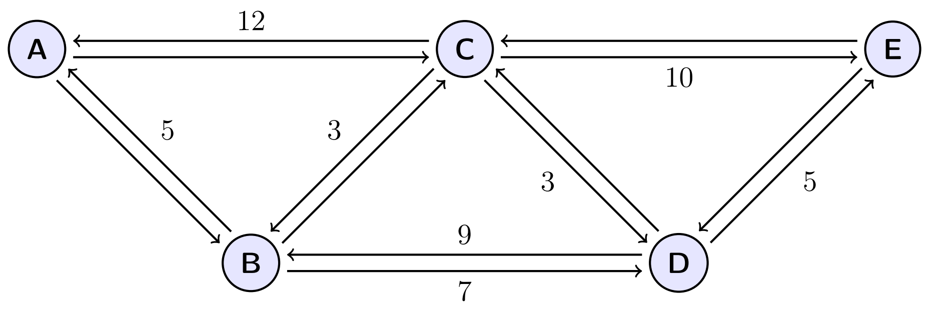

FIGURE 9. Example 4 Two-Step Step 2

We then increase the maximum step to be 3. We can see from the results that some of the vehicles will stay where they are in the first step. This is actually a critical way to lower the total travel time of the system. If people could pick slightly different time to depart, it would be helpful to reduce congestion.

4.2. **Discussion of Results.**

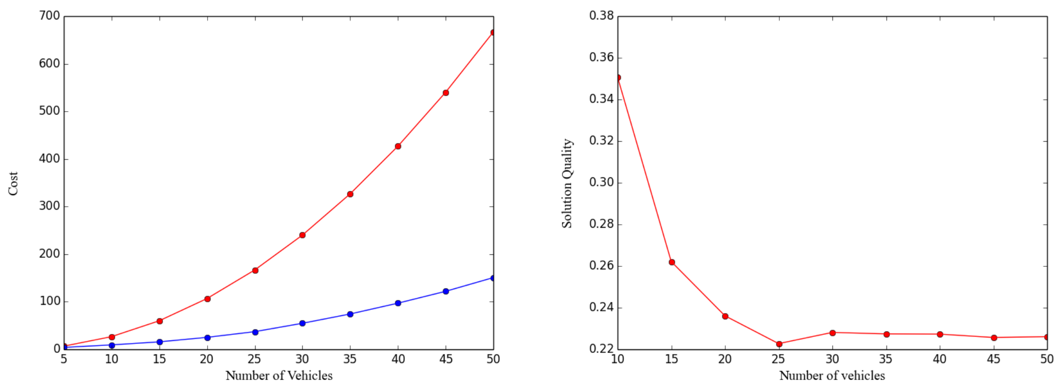

FIGURE 11. Comparison

Using the incomplete graph of 5 nodes in Figure 7, we can conduct multiple tests on the graph with different number of vehicles running on it. We assign different numbers of people to go from A to E, calculate the *naive cost* and optimal cost from our algorithm and plot the data in the Figure 11.

From the graph, we can see that both *naive cost* and optimized costs grow as we increase the number of people travelling on the roads. However, the two costs differ dramatically in the following way. The naive cost grows at a much faster rate than the optimized cost and reaches 2670 when there are 100 people going simultaneously from A to E. In contrast, our optimized cost remains at a relatively small number of 590 under the same conditions.

Moreover, from the right side of Figure 11, we see that the solution quality converges with the increase in the number of vehicles. When all vehicles can run smoothly without much congestion on the roads, our algorithm gives a travel plan that is only 35% of the naive cost. This percentage quickly converges to 22.6% as we place more vehicles onto the graph.

4.3. **Application.**

Finally, we would like to test our model on a real life example, so we have gathered data from New York City and picked its five boroughs as nodes and some federal interstates as edges. We create the graph as in Figure 12. We also extract the road information like the speed limits, the number of lanes and lengths of the roads from real data.

By setting the maximum step to be 3 and OD to only contain people who wish to travel from Manhattan to Staten Island, we hope to see that people will disperse onto various routes. Our result shows that 48% of the vehicles go to Brooklyn first since Manhattan-Brooklyn-Staten Island is the shortest path. 13% goes to Queens in the first step. We believe that this is due to the system wishes to reduce congestion and total travel time by all means, although the routes might be longer.

We can also see from the results that the optimized cost of the model is 7.95, while the naive cost is 16.18, which is twice as large as the optimized one. We can conclude that indeed the routes proposed by a central planner can plan routes that take the least amount of time for all vehicles in the system even if some of the routes are not the most direct or the shortest. .

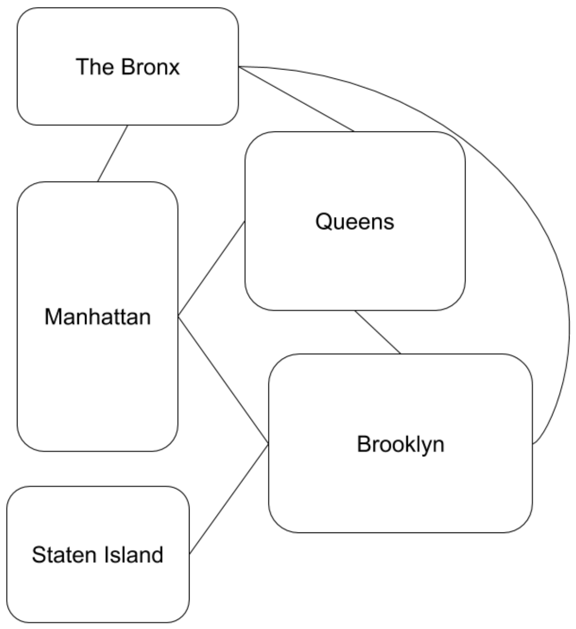

FIGURE 12. New York City Lower Manhattan

5. Future Directions

Our paper emphasize on the total cost for all the vehicles and one aspect of this optimization is that a few individuals may have to endure much higher costs in order for the total cost of everyone in the system to be the minimum. Other modes of optimization may be to minimize costs in terms of total cost for the system, minimum cost for the sum of travelling on all roads in the system, or L2 norm for costs of all the vehicles. From the comparison results, we can see that even with a simple ideal flow rate model, the results were promising. With the application of such a simple model, certain aspects of the model had to be assumed to be fixed such as the number of lanes and the speed limit. Perhaps with more complex models, the results will be even more drastic. Our model also depends on that all the vehicles must enter the roads at the same time, which is not very reflective of ongoing traffic.

Moreover, to simulate a real system where not every vehicle on the road is routed by a central planner, we introduce a new parameter that fix a certain proportion of vehicles in the system to be under our control. For those that are not under our control, since they do not have any information regarding the level of congestion and the traffic condition on the road, we assume that they would just take the naive path to their destination. If multiple shortest plans are available, we assume that an equal proportion would take each route. By manipulating the proportions of vehicles we can control, we can examine how that would affect the overall congestion.

References

[1] C. F. Daganzo, *The cell transmission model: A dynamic representation of highway traffic consistent with the hydrodynamic theory*, Transportation Research Part B: Methodological, 28 (1994), pp. 269 – 287.

[2] J. Fawcett and P. Robinson, *Adaptive routing for road traffic*, Computer Graphics and Applications, IEEE, 20 (2000), pp. 46–53.

[3] F. A. Haight, *Mathematical theories of traffic flow*, tech. rep., 1963.

[4] D. Helbing, *Traffic and related self-driven many-particle systems*, Reviews of modern physics, 73 (2001), p. 1067.

[5] P. I. Richards, *Shock waves on the highway*, Operations Res., 4 (1956), pp. 42–51.

[6] A. Karakitsiou, A. Mavrommati, and A. Migdalas, *Efficient minimization over products of simplices and its application to nonlinear multicommodity network problems*, Operational Research, 4 (2004), pp. 99–118.

[7] F. Kelly, *The mathematics of traffic in networks*, The Princeton companion to mathematics, 1 (2008), pp. 862–870.

[8] B. S. Kerner, *Traffic Congestion, Modeling Approaches to*, Springer New York, New York, NY, 2009, pp. 9302–9355.

[9] W. Leutzbach, *Introduction to the theory of traffic flow*, vol. 47, Springer, 1988.

[10] M. J. Lighthill and G. B. Whitham, *On kinetic waves, ii . a theory of traffic flow on long crowded roads*, 1955.

[11] R. Mahnke, J. Kaupužs, and I. Lubashevsky, *Probabilistic description of traffic flow*, Physics Reports, 408 (2005), pp. 1–130.

[12] A. D. May, *Traffic flow fundamentals*, 1990.

[13] T. Nagatani, *The physics of traffic jams*, Reports on progress in physics, 65 (2002), p. 1331.

[14] K. Nagel, P. Wagner, and R. Woesler, *Still flowing: Approaches to traffic flow and traffic jam modeling*, Operations research, 51 (2003), pp. 681–710.

[15] G. B. Whitham, *Linear and nonlinear waves*, vol. 42, John Wiley & Sons, 2011.