MATH 5740 Mathematical Modeling: The Villisendo Process for Traffic Analysis *

Vignesh Iyer, Josh Cragun, and Eric Brown

April 16, 2020

Abstract

Given a connected multi-weighted directional graph $G = (\mathcal{V}, E)$ representing a network of streets within a city (the sets of weights representing road length, speed limit and congestion as a function of time), this paper will investigate the following:

1. What is the shortest route in time between vertices $\mathbf{a}, \mathbf{b} \in \mathcal{V}$?
2. What time should one leave in order to be on time for an appointment?
3. At what threshold do side-roads become more viable than main/direct roads?
4. If the city commissioners want to add a road to the network, where should it go in order to reduce overall congestion the most?
5. On which streets should public transportation be built up to reduce congestion the most?

*The name "Villisendo" comes from a random fantasy name generator because we were otherwise incapable of coming up with a good name for this paper.

1 Prerequisites

Let the following square matrices be given:

1. D with entries d_{ij} being the physical distance between nodes i and j (We assume there are no self connecting edges and thus we represent $d_{ij} = 0$ if no edge connects them).

2. V with entries v_{ij} being the speed limit (or maximum speed) of the road connecting i and j (0 if no edge connects them).

3. $C(t)$ with continuous entries $c_{ij}(t) : [0, 24) \to (0, 1]$ being the fluidity of traffic (comparable to measuring road congestion) on the road connecting i and j at time t ($c = 1$ when no cars are on the road, 0 if no edge connects them).

2 The model

2.1 Travel time along an edge

Assume that the actual velocity along some given road is $v_{ij}(t) = v_{ij} \cdot c_{ij}(t)$. Given the physical relation $v = \frac{ds}{dt}$ and since $v_{ij}(t)$ is continuous we can calculate the distance function through the indefinite integral

$$s_{ij}(t) = \int v_{ij}(t)dt = \int v_{ij} \cdot c_{ij}(t)dt = v_{ij} \int c_{ij}(t)dt.$$

Since v_{ij} is a positive constant and $c_{ij}(t)$ is always positive, we know that $s_{ij}(t)$ is a monotone increasing function and therefore injective, hence invertible. We can obtain the travel time T_{ij} from a starting time t_0:

$$d_{ij} = s_{ij}(t_f) - s_{ij}(t_0) \Longrightarrow T_{ij} = s_{ij}^{-1}(d_{ij} + s_{ij}(t_0)) - t_0 = t_f - t_0$$

Remark. *In practical examples it is unlikely that either the form of the congestion functions $c_{ij}(t)$ and/or the (inverse) position function will be explicitly available. But the above will still hold in principle. As such, numerical methods typically will need to be employed in order to find a usable approximate solution.*

We can now construct a matrix containing travel times along each edge of our network. We can rewrite $T_{ij} = T_{ij}(t_0)$ as a function of start time.

2.2 Travel time along a path

Let $W = (a, b, c, ..., z)$ denote a finite directed drive along the edges $(ab, bc, ..., yz)$ of the graph.

Given the travel time relation T_{ij}, one may compute the travel time along the whole walk from a starting time t_a. Since $T_{ab}(t_a) = t_b - t_a$ i.e. the difference in time of being at node b and node a, we can recover the "absolute" time when we reach our first stopping point t_b by simply considering the quantity $T_{ab}(t_a) + t_a$. Since t_b is our input for the next step, if we want to know the magnitude of to arrive at stop c, we simply compute $T_{bc}(T_{ab}(t_a)+t_a)+T_{ab}(t_a) = (t_c - t_b) + (t_b - t_a) = t_c - t_a$.

From here it becomes clear that the general form is

$$\begin{aligned} T_W &= T_{yz}(T_{xy}(\cdots T_{ab}(t_a) + t_a)) + T_{xy}(\cdots T_{ab}(t_a) + t_a) + \cdots + T_{ab}(t_a) \\ &= (t_z - t_y) + (t_y - t_x) + \cdots + (t_c - t_b) + (t_b - t_a) \\ &= t_z - t_a \end{aligned}$$

and so the absolute time of arrival is simply:

$$t_z = T_W + t_a$$

3 Answers to posed questions

3.1 Shortest route between two nodes

The shortest path between two nodes in a graph is typically found using Dijkstra's algorithm. However, that does not work in this case as illustrated by a simple example:

$$T = \begin{bmatrix} 0 & 2 & f(t) \\ & 0 & 2 \\ & & 0 \end{bmatrix} \qquad \text{where} \quad f(t) = \begin{cases} 11 - 5t & t \in [t_0, t_0 + 2) \\ 1 & t \in [t_0 + 2, \infty) \end{cases}$$

The solution found by Dijkstra's algorithm would be the *path* (1,2,3) taking 4 units time. However, the actual fastest route is to wait two units time then take the path (1,3) taking a total of 3 units time.

While there exist modified versions of Dijkstra's algorithm which permit for variable edge weights, a fundamental flaw of any Dijkstra-based algorithm is the inability to revisit nodes, which in our model represents waiting for a more opportune departure time. Dijkstra's algorithm in practice queues the current visited node and bases comparisons of weights on adjacent edges, which is why instead of revisiting nodes, the model considers waiting for a practical departure time. The development of such an algorithm to find a solution to this problem is outside of the scope of this paper.

Because we assume our model is deterministic, the time cost curve for a road over the course of an entire day can be calculated beforehand. Because of this, the optimal path can be found analytically.

Lemma 3.1. *Any walk can be reduced to a path taking the same or less time, i.e. the optimal route contains no cycles.*

Proof. Let $W = (a, \cdots, b, c, \cdots d, b, e, \cdots, f)$ and let T_C be the time to complete the cycle $C = (b, c, \cdots, d, b) \subset W$. Now consider the cycle-free path $W' = (a, \cdots, b, e, \cdots, f) \subset W$. If we choose the waiting time at node b to be $w_b = T_C$ then $T_{W'} = T_W$. Therefore,

$$\inf_{w_b \in [0, T_C]} T_{W'} \leq T_W$$

Repeating the same argument for every cycle in W completes the proof. □

In essence, the best path will not contain any extraneous steps because there will always exist a path that is just as good if one simply departs from the same starting node at a later time. Another way of looking at this is to consider the Bellman optimality principle which states that a (time-independent) sub-path of a shortest path is always itself a shortest path. However, taking $C \subset W$ as given above, it is clear the the shortest path from b to itself is no path at all. Hence nothing was gained by going in a circle.

What this means is that we can limit our solution set to be any regular path from our starting point to our destination with a waiting offset for each vertex.

3.1.1 The Best Path

To find the best path, first we need to find a list of candidate paths. This can simply be done by treating our system as an directed, unweighted graph and performing a *Depth First Search* on it. We know that the best path must use one of these paths (with some amount of waiting offset at each node), since as shown above, it is always better to wait than go in circles. For each path found, we need to find the fastest path for each. The total time cost from a starting time t_0 for a given path $P = (v_1, v_2, \cdots v_n)$ is given by the following function: (Here we let $\mathcal{T}_{ij}(t_0) = T_{ij}(t_0) + t_0$ i.e. the absolute time of arrival)

$$\begin{aligned} C_P(t_0, w_1, w_2, \cdots w_{n-1}) = {} & w_1 + T_{1,2}(t_0 + w_1) \\ & + w_2 + T_{2,3}(\mathcal{T}_{1,2}(t_0 + w_1) + w_2) \\ & \vdots \\ & + w_{n-1} + T_{n-1,n}(\mathcal{T}_{n-2,n-1}(\cdots) + w_{n-2}) + w_{n-1}) \end{aligned}$$

Where $w_1, w_2, \cdots w_{n-1}$ are the waiting times at each node v_1 through v_{n-1}. It is apparent that this problem will become complex rather quickly as the number of vertices in the path increases, nonetheless, it is possible to find these solutions as critical points of C as given by its gradient, the boundary of the domain, or as a solution of ϵ-subgradient descent methods (if the time functions are not completely differential). Once every optimal waiting configuration is found, we simply take the configuration with the lowest overall cost. If it is a tie, then it may be selected arbitrarily among the winners.

3.1.2 Example

Let us use the same example as before, with a similar matrix starting at $t_0 = 0$.

$$T = \begin{bmatrix} 0 & 2 & f(t) \\ 2 & 0 & 2 \\ f(t) & 2 & 0 \end{bmatrix} \qquad \text{where} \quad f(t) = \begin{cases} 11 - 5t & t \in [0, 2) \\ 1 & t \in [2, \infty) \end{cases}$$

Note: The matrix has been made symmetrical. This will ultimately play no part because our above lemma proves that there is no point in using a back edge in an optimal route. The back edges were added to illustrate this point.

A Breadth first search will indicate that there are two possible paths from 1 to 3: (1, 2, 3) or (1, 3). The time cost functions for each path is like so:

$$\begin{aligned} C_{1,2,3}(0, w_1, w_2) &= w_1 + T_{1,2}(w_1) + w_2 + T_{2,3}(\mathcal{T}_{1,2}(w_1) + w_2) \\ &= w_1 + w_2 + 4 \\ C_{1,3}(0, w_1) &= w_1 + T_{1,3}(w_1) = w_1 + f(w_1) \\ C_{1,3}(0, w_1) &= \begin{cases} 11 - 4w_1 & w_1 \in [0, 2) \\ w_1 + 1 & w_1 \in [2, \infty) \end{cases} \end{aligned}$$

It is clear that $C_{1,2,3}$ will be smallest when $w_1 = w_2 = 0$. And hence the best waiting time for that path $(1, 2, 3)$ would be to go immediately to each stop for a total travel time of 4. For $C_{1,3}$, we obtain the following curve:

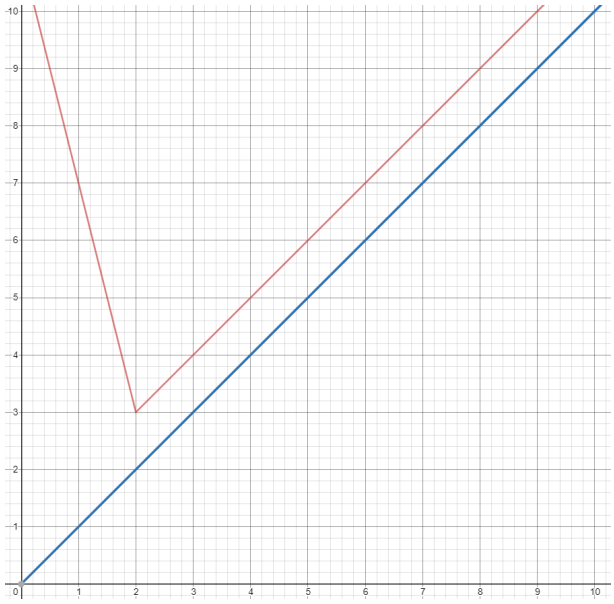

Figure 1: Time cost curve of $C_{1,3}(0, w_1)$

Here, the blue line is the time cost *baseline*, which is the lower bound for all time cost functions. If there are $n-1$ dimensions for waiting times, then the baseline is defined by the following surface in $\mathbb{R}^n$:

$$\text{Baseline}(w_1, \cdots, w_{n-1}) = \sum_{i=1}^{n-1} w_i$$

In any case, it is apparent that the minimum of the time cost curve is at $w_1 = 2$ for a total trip cost of 3 units of time, providing the answer we expected– the best path is (1,3) in which the driver waits at stop 1 for 2 units of time before proceeding to stop 3, and should expect to arrive at his or her destination in 3 units of time.

3.2 When to leave for an appointment

A naïve way of answering this question is to turn time backwards: replace all values of t with $-t$ and find the quickest route from the destination to the origin with initial time t_0 being the desired arrival time.

This approach works only if there are no one-way streets in the model, i.e. if T is symmetric. Consider the following graph:

$$T = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

The quickest (only) route from 1 to 3 is (1,3) in 1 unit time but the quickest (only) route from 3 to 1 is (3,2,1) in 2 units time. An easy way to account for this would simply be to create a mirror graph $V' = V^T$ in which the direction of every road is reversed and then perform the above computation (reflect the time functions by transforming $T'_{ij}(t) = T_{ij}(-(t+24))$). So in our example we would have

$$T' = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$$

And immediately we see that best (only) route from 3 to 1 is simply through (3, 1) in one unit of time (as desired). To then recover the correct answer we simply reverse the optimal path for T'.

3.3 When to use side-roads vs main roads

The solution to this question drops out once a method to calculate the travel time of a particular path exists, since one should use a side road if the travel time of the path is less than the travel time of the main road. Since such a method is provided in 2.2, one can safely say that in a choice between two roads (main or not) $A = (a_1, a_2, \cdots, a_n)$ and $B = (b_1, b_2, \cdots, b_m)$, one should take the route A over route B if $T_A < T_B$. This all being said, it would be worthwhile to formalize the notion of a "main road" versus a regular one.

3.4 Where to add roads

The first question we must address is which road addition would be most "important" or "useful" to the city. There are a multitude of ways to do this, and each way largely depends on the model itself. In all cases however, the task of identifying a main road corresponds to identifying which roads are the most central to the graph by either directly connecting to the most locations or containing edges that overlap with most other paths. In any case, it is clear that there can be no objective dichotomy for which a road may be considered "main" or not. Thus any solution would essentially involve implementing a path scoring spectrum to characterize the road's general utility and connectedness in which a"main road" is then any path that reaches or surpasses an arbitrary score threshold. Call any such metric a *Utility Score* $U : \mathbb{P}(\mathcal{V}, E) \to \mathbb{R}^+$ where $\mathbb{P}(\mathcal{V}, E)$ is the set of all paths in $G = (\mathcal{V}, E)$.

Remark. *The the precise nature of such a score is left vague intentionally so as to allow for flexibility for city planners. For example, in some cases it might be preferable to penalize cutting through existing landmarks where as in other cases that might not be a big issue. Utility scores could be primarily concerned with congestion or connectivity (and thereby encouraging commerce). In any case, once such a definition is provided, the following remarks will hold.*

Let the utility score $U(P)$ be as defined above. We can define the *global utility* of a network $U_g(\mathcal{V}, E)$ to be

$$U_g(\mathcal{V}, E) = \sum_{P \in \mathbb{P}(\mathcal{V}, E)} U(P)$$

over all paths P.

If the city were to add a new road e_{ij} then we can define $E_{ij} = E \cup \{e_{ij}\}$. If the new road costs p_{ij} (including both monetary and non-monetary costs) to construct then we can calculate

$$\Delta U_{ij} = \frac{U_g(\mathcal{V}, E_{ij}) - U_g(\mathcal{V}, E)}{p_{ij}}$$

which is the greatest increase in utility per dollar spent. The city commissioners should choose the maximum of these to build the new road. It is also necessary to ensure that this change will not result in Braess' paradox.

3.5 Where to implement public transportation

This solution is very similar to that of the previous section. Instead of adding new edges, however, we are decreasing the congestion (i.e. increasing the value of $C(t)$) along certain edges of the graph according to some cost involved in adding said transportation. Generally this will involve introducing public transportation along roads with high utility scores and optimizing our selection based on what improves the global network utility minus the cost associated with the selection the most.

4 Limitations of the model

1. One critical limitation is that our congestion functions never allows for traffic to come to a complete stop. If this was the case, then a great deal of our premises would no longer hold, and a new approach to the task would be in order. To allow for this, some assumptions would need to be made about the cases in which $c_{ij}(t) = 0$.
   If we assume that c_{ij} is 0 only over finite and convex subsets of $[0, 24)$ (call the union of these subsets O), then it would still possible to compute the arrival time, as $s_{ij}^{-1}(t)$ would be exist over $\mathbb{R}^+ \setminus s_{ij}(O)$ and the image $s_{ij}(t)$ over $[0, 24)\setminus O$ would only be missing countably many distinct points. Hence it would be possible to find a suitable arrival time for any given distance d, even if d is not a member of $\mathbb{R}^+ \setminus s_{ij}(O)$, since one could find an arbitrarily accurate arrival time $s^{-1}(d^*)$ from some other d^* in a neighborhood of d.

2. For each edge, we assume that the traffic is uniform and designate a continuous periodic function – assuming that on a day to day basis, the congestion follows the same pattern i.e. every $[0, 24)$ is uniform. That being said, the exact domain needs to be fixed. These functions could be periodic over weeks, months, or years. In some ways, the periodicity is insignificant to our analysis; however it is necessary for any practical application to be able to make predictions about shortest paths, etc.

3. Lemma 3.1 assumes that it is possible to wait, but many cities have "no stopping any time" zones that essentially force drivers to move in circles. Additionally, one may have to pay to park for more than a few minutes so this may be uneconomical.

4. Stoplights are not factored into our analysis. Doing so would be reasonably straightforward however, as it would generally entail adding an expected delay to each edge based on a standard Bernoulli distribution for the number of stoplights tweaked according to their (possibly unique) probabilities of being green/yellow or red and how long each light remains red. It would also introduce a degree of error that would need to be propagated through each of our original calculations.

5. Multi-lane roads are not factored into our analysis. Accounting for this might be done as simply as extending our congestion functions to account for congestion across all lanes. Doing so would generally allow for the above calculations to hold.