The Trouble with Traffic

An analysis of modern traffic systems and the effects congestion has on them

By Cole Amaratunge, Trece Wicklander-Bryant, Sam Qian, Kris Kwon

[**Introduction**](#_c8nvv4dyl8tt) **2**

[**Model Description**](#_fdg5jst40b2z) **3**

[Agent: Car](#_dd60pgovp5mu) 3

[Field: Map](#_yocqxs6jy4cm) 4

[Intersection Class (Nodes)](#_mba520m7ku05) 4

[Road Class (Edges)](#_4u8lk13arvz7) 4

[Simulation Driver:](#_z0uql5sndt5l) 4

[The Simulation Process](#_b7lusygt7iwd) 5

[List of Assumptions](#_yaeks0ekde0g) 5

[List of Constraints](#_mgfqwu3fzh1v) 5

[Model Q&A](#_hbkp1m94g9gl) 6

[**Analysis**](#_atqmgvbri9zw) **6**

[Verification](#_akqtgzjfthas) 6

[Validation](#_82za8xphpw2b) 7

[**Testing**](#_ypec2c4uwsej) **7**

[**Personnel**](#_sm0th25jjb8x) **8**

[Team Member: Cole](#_jeo9khfm9aed) 8

[Team Member: Trece](#_roabcc1xufk8) 9

[Team Member: Sam](#_gwxyqfbqyqnl) 9

[Team Member: Kris](#_xdwak1esx7gn) 10

[**Technologies**](#_m4kap19hk25b) **10**

[Python](#_1t9v9rj2gvn9) 10

[Sublime Text](#_cmxmxsy8ebz4) 11

[GitHub](#_h6ftssk1ga0t) 11

[Discord](#_ycu8c6naregj) 11

[Google Drive](#_vonhhin8ud3p) 11

[Other Traffic Simulations](#_4ejqdxvj8q8m) 11

[**Benchmarks**](#_gw07dikqbzab) **12**

[**References**](#_pxlb57l2ma9b) **12**

# **Introduction**

The aim of our model is to simulate the effects of congestion on a modern traffic system. Our model draws on a list of concepts and theories: Graph Theory, Queuing theory, Shortest paths algorithms and agent based models.

When it comes to modeling traffic, Graph Theory is quite popular and rightfully so. A Mathematical Graph is described as a set of nodes (points) and edges (lines). While nodes cannot directly interact with another node, we consider two nodes adjacent if they share a single edge between them. Edges of a graph are either directed, one-way relationship from node to node, or undirected, two-way relationship between nodes. Graphs can then be further classified by: the total number of edges and nodes or the number of edges per node. The theory lends itself quite naturally to stop-and-go movement, and so with certain assumptions traffic systems are well described with a graph.

Queuing Theory is also quite common with traffic models, but unlike graph theory, is not for describing a whole traffic system, instead the use is more focused. Often when a system is having issues, it helps to understand the behaviour at a specific point of trouble, that is the goal of queuing theory. The theory focuses on points in a traffic system and aims to mathematically describe the behaviour at those points. Often, in Vehicular traffic system these points tend to be intersections, on/off ramps or accidents.

While Shortest path algorithms can be considered a subset of graph theory, their purpose is a different in traffic models. Again, it is quite standard to see shortest path algorithms come up when discussing vehicular traffic models especially models described with mathematical graphs. However, like queuing theory Shortest paths don’t focus on describing the system, instead they often refer to a proposed solution or a hypothesis. The algorithms, like the name suggests, describe a way to find the shortest path between two points. For our specific model, we plan on using the famous Dijkstra’s algorithm (with some slight modifications). Dijkstra’s algorithm is a proven and standard method for computing shortest paths. The bad news, Dijkstra’s algorithm (and most other shortest path algorithms) is computationally costly but, for now we will avoid worrying about computational performance and scalability.

Lastly, we have Agent-based models and again we are discussing a popular concept for traffic models. The agent based model is based on describing a system with agents and and a field. In vehicular traffic systems, agents are the vehicles and the field is the set of roads, destinations, and any other traffic related stops. Agents, are given a set of rules and are then their behaviour in the designated field is recorded. For traffic simulations these rules vary from model to model but, in essence, the rules describes when cars can and cannot move.

Notice that both Graph theory and Agent-based modeling are fundamentally how our traffic system will be simulated. They do not answer any questions or expose system flaws. However, we hope by manipulating our model at specific points of interest we can use queuing theory to verify our results and then maybe even predict future simulations. Shortest path algorithms will be our way of measuring data from our model. By using a modified Dijkstra’s algorithm we can actually compare different agents, simulations and variables.

# **Model Description**

Our model will analyze the effect of traffic due to cars knowing routes of each other.

Structure of the model:

Our model will be agent-based, which allows us to separate our model into the agents and the field, they will act in.

## **Agent: Car**

The car will be our only used agent for the model. However for the sake of our model, physical car properties will not be taken into account. While we admit size of a car could impact traffic systems, we are not interested in measuring that impact for the model. The primary components our Car, is current location and destination. The value of these component will describe a specific node which we will discuss in greater in detail in the field section. However, to account for visualization the car’s current location will be stored in terms of x and y coordinates.

To Implement a car we are going to create a car class which will contain the described fields but, it will also have a set of methods for behaviour.

The find\_next\_Node method will use a modified version of Dijkstra’s to go to the next node. Our modified Dijkstra’s will calculate shortest paths but, will also include the capacity of a node.

Then the Move method will actually move the car from node to node. This will be done by simply updating the field’s data structures.

## Field: Map

The map is what the car agents will be “driving” on, it can be thought of as a graph. That being said the map will consist of two major components intersections(nodes) and roads(edges). Below is a class level view of the graphing components:

### Intersection Class (Nodes)

Member Variables:

* Queue of cars at intersection
* List of roads that are connected to it
* Weight
* Weight trail

Member Functions:

* shortestPath() - Returns the shortest path from current node to destination node and does not consider the congestion level of each road.
* shortestPathTraffic() - Moves the car from one node to the best possible node by considering the traffic congestion levels.

### Road Class (Edges)

Member Variables:

* List of cars on this road
* Road Capacity
* Time to travel road
* Starting Intersection
* Destination Intersection

Member Functions:

* computeTime() - Updates the time to travel road variable with the new time it will take to travel the full distance of the road.

### Simulation Driver:

Member Variables:

* INTERSECTIONS - list of interactions
* ROADS - list of roads
* Map - a matrix of map
* NUM\_CARS - number of cars
* CARS - list of cars

Member Functions:

* \_\_init\_\_map() - load map file and fill INTERSECTIONS and ROADS list
* \_\_init\_\_cars() - fill CARS list
* update() - update intersections, roads, NUM\_CARS, Maps, CARS

## The Simulation Process

The system will call init\_map method first to load map from map file to a matrix. As map section has discussed, we will have edges and stops in the matrix and numbers representing capacities. Next, it will call method init\_cars to generate a list of car agents. All car agents will have the same starting points and destinations Time frame is set to be zero as we put cars on roads at first. At this point, initialization has done. All intersections will be updated as cars being popped out from the queues, we call it a time frame when updating intersections. When the queues are empty, the simulation ends. Total time takes for cars reaching the destination will be the time takes the last car reaching the destination.

## List of Assumptions

* No traffic lights
* Constant speed on roads
* Size of cars will be treated as same
* Road is measured as capacity (cars/road)
* No car accidents
* Reaction time is same for all drivers
* All the roads are one way, bi-way roads can be expressed as an individual layer
* No stopping points on roads (only on nodes, intersections)

## List of Constraints

* Due to computational power limits, map is small
* Data is generated by the simulation, which may be different from real life situation
* Map is a constant for now
* Cars cannot truly move simultaneously

## Model Q&A

Q: What constraints will you have (and do you have data/can you get data to constrain your model)?

A: We will have small size maps due to computational limits. Data is generated by the simulation, which only works with reasonable settings.

Q: What could you model be used for?

A: The model can be used to optimize traffic during busy hours. It can also be helpful with traffic facilities design because the model is capable of calculating the effect from road capacities, node configurations and number of cars.

Q: What kinds of problems could it help solve? What use-case limitations would your model have?

A: It can help properly distribute traffic loads in a city, and reduce traffic congestions. In order to make it work, drivers have to agree on taking the routes provided by the system.

Q: What would you need to overcome those use-case limitations?

A: The idea would be to integrate our model into an already popular mapping service or autonomous car system. That way, we can bypass marketing and just work with improving systems.

# **Analysis**

“Will cars knowing the routes of other cars decrease time to get their destination and ultimately reduce the congestion?”

In our models, we are specifically interested in the time taken for all the cars to finish their trips. Inputs for our models are the number of cars starting from one point to the destination. Starting point and destination is same for all the cars in the model.

## Verification

We will be utilizing two kinds of models, one with normal GPS and another with GPS implemented with greedy algorithms. With normal GPS, every car will take route that was provided in the beginning. Regardless of increasing input size, the route that is taken by all the cars should be same and this phenomenon will be visually represented for verification.

On the other hand, in the model with greedy GPS, each car will optimally reroute its trip based on the congestion information, which is greedy algorithm. In visualization, as input size increases, cars should take different routes based on the calculation at each instance.

## Validation

The output, overall time taken for every cars to finish their trips, will vary based on models and various inputs. Validating various output data requires statistic validation methods. Multiple simulations will produce patterns to mark significance of outputs. By analyzing correlation and variances of the significances, the model will answer our original question.

# **Testing**

We will conduct 3 kinds of tests are before running our model with the “Greedy GPS” to predict the future based on our assumption. The map which will be used for the model will be manually constructed so that the model map resembles the real traffic situations. The 3 tests before that are going to be ran before the “Greedy GPS” are as follows:

1. Running a small number of cars on model with normal GPS and manually constructed map.  
   1. Outputs, the time taken for every cars to finish their trips, will be tested by comparing it to the data provided by “Google map.” In this manner, we assure that the manually constructed map is sophisticated enough to reflect real traffic situation in our models.
2. Running significantly large number of cars on model with normal GPS and manually constructed map.  
   1. Large number of cars as input in the model with normal GPS will produce different, or longer, time as output in comparison to the running small number of cars. We are trying to simulate congestions in our simulation, therefore, inputs of large number of cars should produce longer time taken for every cars to finish their trips. In this way, we assure that our model sophisticatedly reflects real traffic situation, congestions in this case.
3. Running a small number of cars on model with “Greedy GPS” and manually constructed map.  
   1. In this case, the outputs should be similar to the one with the running model with normal GPS with a small number of cars. Without congestions on the road, the GPS will not and does not have make greedy choices for finding paths. In this way, we assure that making greedy choice is only made when the road is congested. Doing so, we assume only factor that is affecting the result is making greedy choices whether resolve congestion problems.

Before running many simulations with significantly large number of cars in the model with “Greedy GPS” and varying traffic maps we need to acknowledge the following:

* The method for making greedy choices will be tested separately. The algorithm needs to be proved that each calculation will produce optimal solutions at each instance. Moreover, implementing algorithm will be verified so that we assure the calculation is appropriate.

Then, we can run multiple simulations of our models with normal GPS and “Greedy GPS.” Furthermore, map can be replaced for representing different traffic situations for more simulations.

# **Personnel**

## Team Member: Cole

Member Bio

My main goal is to apply and verify proven theoretical concepts to our model. This would include coming up with mathematical functions for and from our model. Although I cannot be constantly applying theoretical so, my other task is to help with the model design, deciding on classes and method signatures. Besides that, I will also have my assigned portions to code and documentation to write.

Strengths

* **Code:** Python
* **Mathematics:** Graph Theory, Matrices and differential equations

Responsibilities

* In charge of Class design
* Integrating theoretical concepts

## Team Member: Trece

Member Bio

The main take-away I’d like to gain from this project is an overall better understanding on the entire modeling process. In addition to understanding the modeling process I believe that the problem we’re attempting to solve would be extremely helpful to anyone that commutes. If we were able to solve this problem it could be integrated with GPS and possibly make self-driving cars slightly better.

Strengths

* **CS:** Computer Vision, Neural Networks, Databases
* **Mathematics:** Linear Algebra
* Organizational Skills

Responsibilities

* Create, Format, and Finalize team documentation.
* Analysis of working algorithms.
* Design objects relating to our problem.

## Team Member: Sam

Member Bio

My main interest of this project is to learn project experiences that includes finding the problem, brainstorming possible solutions, implementing solutions. This topic is extremely interesting, if we all use same navigation system, the system should optimize traffic. Although the model seems to be simple compare to real life situation, it is still a good way to study traffic, and we can also build upon this easily.

Strengths

* CS: Python
* Mathematics: Calculus, Tensors, Algebra
* Problem solving skill

Responsibilities

* Implementation of theories
* Module test

## Team Member: Kris

Member Bio

My main interest of this project is not only the interesting result that will answer our questions, but also, model development process. During the development process, brainstorming, implementation, verification, and validation require us to critically think to properly solve the problems and find answers. This experience is invaluable and working with awesome teammates excites me even more. I do not have any previous experience of working for project in team, but this would be the good chance to learn as much as I can.

Strengths

* Object oriented Programming
* Software Architecture
* Software Testing

Responsibilities

* Testing (verification and validation of model)
* Output analysis
* Making sure the team is on track

# **Technologies**

This section outlines the technologies that our team will be utilizing to create our project, specifically the code for the project and all relating documentation for the project.

## Python

Python is an interpreted high level programming language that is very general in purpose. Its importance for our project is of our project will be coded entirely in python.

## Sublime Text

Sublime Text is a text editor with a specific focus on program files of all types. We will be using Sublime Text as the environment in which we write our python files.

## GitHub

GitHub is a hosting service for version control commonly utilized by coding development teams. Its use in our project will be the same, that is, we will be utilizing GitHub in order to enable team development for our code as well as all documents related to our project.

## Discord

Discord is a freeware voice chat application that specializes in text and audio communication between users. We are using discord for communication purpose(i.e. general communication, meeting times, tasks to complete, etc.).

## Google Drive

Google Drive is a file storage and synchronization service. Google Drive’s role in our project is the workspace that we create documents before pushing them to our repository. Google drive is great because we have access to it anywhere since it is cloud-based, more importantly all members can remotely and synchronously work on the same document while see all changes made by other members in real time.

## Other Traffic Simulations

Modelling traffic is a common simulation, and there is an abundance of simulations available. While we have not discovered any models quite like ours, we believe there are elements we can learn from by understanding these other models. There is even a chance to use these other models to verify some results of our model, for example checking shortest paths or queue delays.

# **Benchmarks**

A list and description of benchmarks, with dates as to when each benchmark will be finished. Of these benchmarks, one will be graded. Call that benchmark to be graded "Milestone" in your Development Plan.

* + For Milestone
    - Abstraction
      * Define classes
      * Define methods
      * Interactions between classes.
    - Testing plans
    - Implementation
      * Map construction
      * Greedy Algorithm (for both normal and Greedy GPS)
      * Test implementation
    - Testing (verification)
    - Running simulations
  + After Milestone
    - Data Analysis
    - Modifications
    - Validation
    - Powerpoint
    - Visualization
    - Documentation

# **References**

“Agent-Based Model.” *Wikipedia*, Wikimedia Foundation, 15 Apr. 2018,

en.wikipedia.org/wiki/Agent-based\_model.

“Fundamentals of Transportation/Queueing.” *Music Theory/Modes - Wikibooks, Open*

*Books for an Open World*,

en.wikibooks.org/wiki/Fundamentals\_of\_Transportation/Queueing#Real\_Life\_Ca

uses\_of\_Queue\_Generation.

“Graph (Abstract Data Type).” *Wikipedia*, Wikimedia Foundation, 10 Apr. 2018,

en.wikipedia.org/wiki/Graph\_(abstract\_data\_type).

“Graph Theory.” *Wikipedia*, Wikimedia Foundation, 20 Apr. 2018,

en.wikipedia.org/wiki/Graph\_theory#Graph.

Hoyle, Mark. “Random Walks in MATLAB - File Exchange - MATLAB Central.” *MATLAB*

*& Simulink*, 1 Sept. 2016,

www.mathworks.com/matlabcentral/fileexchange/22003-random-walks-in-matlab

?focused=6784479&tab=example.

IEEE. “An Agent Based Approach for Modeling Traffic Flow - IEEE Conference

Publication.” *Design and Implementation of Autonomous Vehicle Valet Parking*

*System - IEEE Conference Publication*, Wiley-IEEE Press, 6 May 2010,

ieeexplore.ieee.org/document/5461811/.

Karima, Benhamza & Ellagoune, Salah & Seridi, Hamid & Akdag, Herman. (2018).

Agent-based modeling for traffic simulation.

https://www.researchgate.net/publication/228778032\_Agent-based\_modeling\_for

\_traffic\_simulation

Mastio, Matthieu, et al. “Distributed Agent-Based Traffic Simulations - IEEE Journals &

Magazine.” *Design and Implementation of Autonomous Vehicle Valet Parking System - IEEE Conference Publication*, Wiley-IEEE Press, 18 Jan. 2018,

ieeexplore.ieee.org/document/8263443/?reload=true.

Tan, Yong. “Analyzing Traffic Problem Model with Graph Theory Algorithms.” Arxiv,

28 July 2015. https://arxiv.org/ftp/arxiv/papers/1406/1406.4828.pdf

Tutorials Point. “Graph Theory Types of Graphs.” *Www.tutorialspoint.com*, Tutorials

Point, 8 Jan. 2018, www.tutorialspoint.com/graph\_theory/types\_of\_graphs.htm.

Yang, Shuguo, and Xiaoyan Yang. *The Application of the Queuing Theory in the Traffic*

*Flow of Intersection*. 6th ed., vol. 8, Word Academy of Science, Engineering and

Technology, 2014,

waset.org/publications/9999252/the-application-of-the-queuing-theory-in-the-traffi

c-flow-of-intersection.