**Modeling High Volume Networks With Dynamically Assigned Node Structures**

### Introduction

- Current transportation networks utilize inflexible algorithms to determine how to route vehicles.
  - This is why buses and trains operate on specific, rigid schedules.
  - This is also why red lights may halt several cars, even when no cars are using the intersecting road.
- Current algorithms cannot be used in networks such as automobile systems because:
  - They fully define the path of the vehicle before runtime.
  - They cannot adapt to changes in the network (drivers changing destination, or the addition or removal of roads).
- Dynamic algorithms, which route vehicles while the network is running, are needed for automobile systems.
- Dynamic algorithms can account for vehicles changing destination and roads being added or blocked, because they operate in runtime.
- Dynamic algorithms would trade optimality of routes for computation speed and flexibility.

### Engineering Goals

1. Create a model of a network in Java using paths, nodes, and vehicles.
2. Create a program which accepts an initial network and outputs network data.
3. Create an algorithm which can schedule multiple vehicles to a single node without collision.
4. Create an algorithm which can route a vehicle incrementally closer to a given node, using path weights given by the algorithm in goal 3.
5. Using the algorithms created, use the program created in goal 2 to model a small network and route vehicles, displaying network data in text format.
6. Create an animated model for the program.
7. Extend the routing algorithm to create a partial route of multiple nodes.
8. Determine the program’s ability to respond to dynamic changes.
9. Test the program’s efficiency using several randomly generated networks.

### Program Overview

- The program is written in Java.
- Networks are composed of nodes, roads, and vehicles.
- Roads are the intersection points of roads – All vehicle routing and scheduling is done at nodes.
- Roads are undirected paths between two nodes. Vehicles are points which travel on roads. They originate at a specific node and have a destination node.
- Low-level algorithms which control the path of the vehicles are schedule and route, each of which are used when a vehicle enters a node.
- The problem runs each vehicle along roads and schedules vehicles at nodes until all vehicles are at their destinations.
- Schedule is used to determine how long it will take for a vehicle to traverse a road, using the length of the road between the nodes, the maximum speed that the vehicle can travel on the road, and times during which the output node is occupied.
- Schedule returns the soonest time that the vehicle may reach the node.
- Route determines all of the sequences of roads that consist of a path route traversals from the origin node to the destination node. It then determines how long it would take to fully traverse each sequence, and determines the best sequence to take by comparing the distance travelled towards (absolute geographic position) the end destination and deciding this by the time it would take to traverse the sequence. It ignores routes that it has already taken.
- Dynamic changes during the “run” method allow vehicle re-scheduling, and clear all pre-determined route data from nodes. This allows other vehicles to use the time slots otherwise used by the vehicle.

### Results

- A program was created which can model networks with real-world constraints and dynamically route vehicles.
- The program was demonstrated as being capable of routing and running several large and complex networks.
- No collisions occurred on the network, as demonstrated by data from the program.
- Random networks were used for testing. With a coe of at least 5, almost all networks could be routed successfully.
- The program responded successfully to dynamic changes in the destination of vehicles.

### Methods and Algorithms

**Program**

- Create the Network and Network Element
- Display the Network
- Runstep
- Are all vehicles at their destinations?
- Output Network Data

**Runstep**

- For all vehicles
- How routes are calculated is further detailed below (methods are as follows):
  - Minimum time = NETTIME + destination
  - Find the maximum speed along the road

**Route**

- Let current be the current distance to the destination
- Let diff be the amount of time it would take to traverse the road sequence (according to schedule)
- Update Network Time

**Schedule**

- Minimum time = NETTIME + destination
- Find the first scheduled time less than the minimum time
- Find the first pair of times after this time separated by 1 + BUFF
- Schedule the vehicle at the lower of these times

**Methods**

- Is the node in the list of scheduled times empty?
- Yes
- No
- Schedule the vehicle at the minimum time
- Find the first scheduled time less than the minimum time
- Find the first pair of times after this time separated by 1 + BUFF
- Schedule the vehicle at the lower of these times

### Summary

Current traffic and rail networks depend largely on pre-defined routes which are optimal for given start conditions. As networks become more complex, and as network parameters are liable to change during runtime (rail breakdowns, wrecks, change of destination, etc.), future networks will need flexible algorithms which can adapt to changes in network parameters. This research attempts to address the flexibility problem of networks by demonstrating a network with dynamically routed and run packets, using a simple greedy algorithm to determine to which nodes to travel. The algorithm demonstrates capabilities to adapt each vehicle’s route in accordance with traffic patterns, which static algorithms are incapable of doing. The algorithm is tested using a Java program which allows for network modeling and testing. Output from the network is represented textually and graphically, and demonstrates collision-free operation of the network.

### Algorithm Correctness and Complexity

As vehicles may temporarily occupy the same space on a road in real life applications (such as a car overtaking another car), collisions on roads were not taken into account in the algorithm correctness. Correctness was determined as representing no potential for collisions on nodes. The scheduling algorithm therefore must be correct, as a collision could only happen if two vehicles are scheduled for the same node simultaneously. This can occur because schedule) separates all vehicles by time. For a network with n nodes and r roads, let c be the coe of each car on the network, and r be the number of vehicles on the network. Thus, each vehicle must check an average of $\frac{r}{c}$ times.

Thus for the entire network, the amount of time, T, that it takes a particular computer to determine how to route all of the vehicles is

$$T = \frac{nr}{c}$$

This shows that algorithm complexity increases exponentially with c.

### Program Limitations and Assumptions

- Routes are often non-optimal, as vehicles follow a greedy algorithm.
- Distribution of vehicles on roads may slow down an unacceptable amount of time. It may occur if a node receives a large amount of traffic from a different input road, causing the schedule algorithm to route these vehicles far later than their minimum time.
- The program assumes properly formatted input, and that all nodes and roads referenced when vehicles, nodes, and roads will be created.
- In rare cases (closed loops), vehicles may be incapable of escaping a certain set of nodes. This is not a problem at higher c values.

### Future Work

- Implement a user interface which can accept dynamic changes to the network.
- Create a random network generator which outputs more realistic networks than the truly random networks currently generated.
- Statistically determine complexity and computation time variance with respect to each network parameter.

### Works Cited