Traffic Simulation Report (GAMA platform)

Author: Hieu Chu

Email: chc116@uowmail.edu.au

Date: 2019/03/23

GitHub repo: https://github.com/aaazureee/traffic (for source code and how to import the project).

Table of Contents

1. Introduction

The traffic simulation consists of a road network, which is based on a graph of nodes (intersection) and roads connecting these nodes. During the simulation, people will be initialized at a random source node, and is given a random destination node to travel to. People will move towards the destination node after each cycle in the simulation. When the person arrives at the destination node, it is removed from the simulation.

This simulation replicates a real world traffic situation, meaning that vehicles aren't allowed to overtake when they are travelling in the same road (i.e. same lane) by determining equilibrium speed using BPR equation. The model also handles congested road, i.e. traffic jam by preventing people from entering the road when the road is at its max capacity. Moreover, people agents have the possibility to apply a smart rerouting strategy based on the current conditions (for example, if the next road is congested, the agent might want to avoid this and choose a different path). The strategy is based on a neural network implementation, with application of genetic algorithm to find optimal strategy parameters. We refer the reader to (Johan Barthélemy and Timoteo Carletti, 2017) for more detailed documentation.

Moreover, it supports dynamic user interaction such as blocking and un-blocking a chosen road on the network graph and see how the people agent adapts to the situation.

There are also data visualization graphs that will aid user's interpretation of the simulation (running in parallel with the main simulation), and text output data in CSV format concerning accumulated number of people passing through roads, nodes, and origin-destination matrix.

2. Environment

This section will provide an overview of underlying components of the simulation (showing important variables and logic handling).

2.1. Global species (Main.gaml)

Variables

- **seed**
  Type: float
  Description: The random seed can be set by user to reproduce the expected result concerning random variables.

- **shape_file_roads**
  Type: file (.shp)
  Description: The shape file (the road network graph) that will be loaded into the simulation. It is assumed that the shape file has ID, length, and freespeed attribute.

- **strategy_file**
  Type: file (.txt)
  Description: The CSV file that contains alpha and theta weights for neural-network based rerouting strategy. User can customize the weights, add more weights and the simulation will feed these weights to people agent randomly.
  Sample format:
  alpha, theta
  2.53, 0.12
  2.54, 0.16
  ...

- **my_graph**
  Type: graph
  Description: The road network that will be built based on shape file (shape_file_roads) loaded.

- **time_list**
  Type: list of float values (unit in second)
  Description: This list will contains the time needed of each person to reach the immediate next node on their path to destination node.

- **nb_people_init**
  Type: integer
  Description: Number of people initialized at the start of the simulation.

- **min_nb_people_spawn and max_nb_people_spawn**
  Type: integer
  Description: Range of number of people randomly spawn after some interval specified by user (spawn_interval). This variable can be dynamically changed during the simulation.

- **spawn_interval**
  Type: integer

Description: After spawn_interval cycles, a random number of people (between min_nb_people_spawn and max_nb_people_spawn) will be randomly spawn. This variable can be dynamically changed during the simulation.

- **radio_prob**
  Type: float
  Description: probability that a single people agent will have global radio, i.e. is notified of traffic change globally (such as recently blocked or unblocked road, so that it can adapts the shortest path accordingly).
- **smart_strategy_prob**
  Type: float
  Description: probability that a single people agent will have a neural-network based strategy attached, so that it can avoids congested road on the shortest path and adapts accordingly.
- **nb_trips_completed**
  Type: integer
  Description: Total number of trips completed during the simulation.

Functions

- **batch_create_people(int num_people)**
  Return: nil
  Description:
  - Spawn people at random source nodes with randomly given destination nodes.
  - **Compute the shortest path** between the source node and the destination node for each person as well.
  - Whenever a new people agent is created, based on the radio_prob and smart_strategy_prob, it will have both radio and smart reroute strategy, or only have the radio, or only have the smart strategy, or have none. Each type of people agent will be colorized differently on the main simulation display (see People GUI).
  - Initial speed will be set to 0 m/s.
  - Increment accum_traffic_count at location nodes by 1 if people are spawned on top of them.
- **generate_people() when: every(spawn_interval)**
  Return: nil
  Description: Invoke batch_create_people() function above after every spawn_interval cycles, with the num_people parameter randomly generated between min_nb_people_spawn and max_nb_people_spawn.
- **get_equi_speed(float free_speed, int current_volume, int max_capacity)**
  Return: float

Description: Will calculate speed of people agent according to BPR equation (Bureau of Public Roads (BPR), 2019), which prevents people from overtaking each other if they are travelling on the same lane.

- **update_min_time()**

  Return: nil

  Overview:

  This is a **core function of the simulation**. It acts as a function to **change the global cycle's time** to prepare people agents before they are able to move in a correctly specified way.

  The idea of the simulation is that, **after each cycle**, it is guaranteed that one (or more) **people agent with the minimum time to reach its immediate next node on its shortest path, will end up at that node** (which means that each cycle's time maybe different).

  This is needed because if people travel with similar cycle time, people agent might pass through one node on its path and end up in the next road. This might cause problems because each road has different maximum travel speed, and also we can't correctly calculate speed (according to BPR equation) before entering the new road. Also, in order to apply the neural-network based strategy, the agent must be on the node to compute the strategy output correctly.

  Description:

  - **Create a list of selected people**:

    (

    - Can't find the shortest path (cant_find_path = true) but is in the middle of the road (agent will move to the nearest next node before getting stuck, so that it doesn't block future people agents coming into this road).

    OR

    - Able to find the shortest path (cant_find_path = false).

)

    AND

    (

    - Is currently in the blocked road (is_in_blocked_road = true) but the destination is in the same road.

    OR

    - Not currently in the blocked road (is_in_blocked_road = false).

)

  - **Ask selected people above**:

    - If the road they are going to enter is jammed (i.e. current_volume = max_capacity), change people's is_road_jammed to true and set speed to 0 m/s.

    - If people's is_road_jammed is false:

      +) If people are on the node (is_on_node = true):

      ➢ Calculate initial BPR speed using get_equi_speed function.

            - Increments people’s next road current_volume by 1.
            - Set free_flow_time_needed to be distance to next node / speed.

        +) Update travel_time to be distance to next node / speed.

        +) Add travel_time to global time_list.

    - **Find minimum value of time_list, change global cycle step time to be equal to that value.**
    - **Increment people’s real_time_spent by the minimum time just found.**

- **refresh_min_time() (Reflex function, invoked in every cycle)**

  Return: nil

  Description: Empty the time list (time_list <- []) then invoke update_min_time().

2.2. People species (People.gaml)

Variables

- **dest**

  Type: node

  Description: Destination node.

- **speed**

  Type: float

  Description: Speed of people agent (unit in m/s).

- **shortest_path**

  Type: path

  Description: Shortest path from source node the destination node

- **current_road**

  Type: road

  Description: Current road that people is taking.

- **is_on_node**

  Type: boolean

  Description: People are currently on one node before entering the new road.

- **is_road_jammed**

  Type: boolean

  Description: The next road people going to take is at maximum capacity.

- **is_in_blocked_road**

  Type: boolean

  Description: People are currently on a blocked road.

- **cant_find_path**

  Type: boolean

  Description: People cannot find the shortest path on the graph.

- **has_radio**
  Type: boolean
  Description: People agent has radio_prob percentage to have global radio.
- **has_smart_strategy**
  Type: boolean
  Description: People agent has smart_strategy_prob percentage to have neural-network based reroute strategy.
- **alpha**
  Type: float
  Description: Value of alpha weight (in radians) for neural-network based reroute strategy.
- **theta**
  Type: float
  Description: Value of theta weight for neural-network based reroute strategy.
- **real_time_spent**
  Type: float
  Description: The amount time people agent spent.
- **free_flow_time_needed**
  Type: float
  Description: The assumed free flow travel time that agent needs to reach the next node.
- **free_flow_time_spent**
  Type: float
  Description: The free flow travel time that agent has spent.

Functions

- **smart_move() (Reflex function, invoked in every cycle)**
  Return: nil
  Overview:
  This is the function that will define how people agent moves after setting the minimum global cycle time (in global's update_min_time function).
  Description:
  - The function is only invoked in every cycle **for selected people agents** who:

  (
  - Can't find the shortest path (cant_find_path = true) but is in the middle of the road (agent will move to the nearest next node before getting stuck, so that it doesn't block future people agents coming into this road).

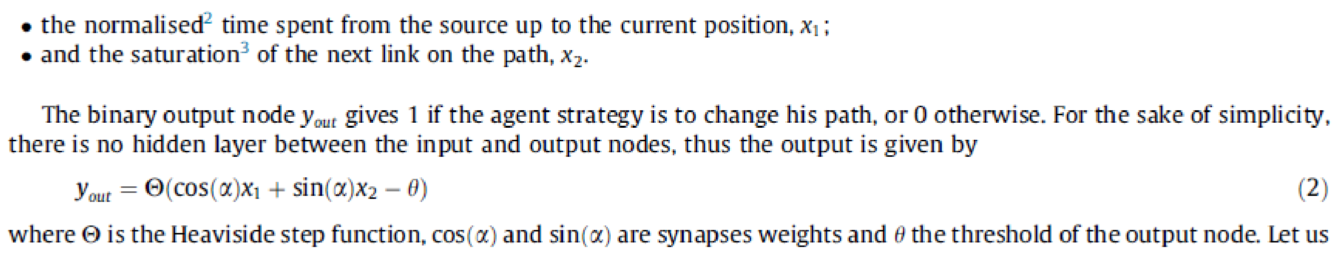
  OR
  - Able to find the shortest path (cant_find_path = false).

)
  AND

The current road is not full (is_road_jammed = false).
AND
(
- Is currently in the blocked road (is_in_blocked_road = true) but the destination is in the same road.
OR
- Not currently in the blocked road (is_in_blocked_road = false).
)

- **Follow the shortest path specified** (computed in batch_create_people function), according to **BPR speed** (computed in update_min_time function), with time travelled equals to **minimum global's cycle time** (computed in update_min_time function).
- Change is_on_node to false.
- If people agent arrives at the next node on their shortest path:
  - Decrease current road's volume by 1 (current_road.current_volume).
  - Increase traffic count at next_node by 1 (next_node.accum_traffic_count)
  - Increment free_flow_time_spent by free_flow_time_needed (accumulate previous free flow time calculated from previous node).
  - If it's the final node (i.e. destination node):
    +) Increase global's number of trips completed by 1 (nb_trips_completed).
    +) Make this agent disappear from the simulation.
  - Else:
    +) Change is_on_node to true.
    +) Change current_road to next road on the shortest path.
    +) If current_road is blocked, the agent will try to reroute by recomputing the shortest path from location to destination. If it's not possible, change agent's cant_find_path to true.
    +) If the agent has smart strategy (has_smart_strategy = true) and is able to find the shortest path (cant_find_path = false), the agent will invoke:
    do will_reroute(normalized_time_spent, next_link_saturation), where:
    normalized_time_spent = real_time_spent / free_flow_time_spent
    next_link_saturation = current_road.current_volume / current_road.max_capacity

- **will_reroute(float normalized_time_spent, float next_link_saturation)**
  Return: nil
  Description: This function will determine if the agent will reroute to avoid a congested road on its shortest path, based on neural-network implementation.



- the normalised[2] time spent from the source up to the current position, x_1;
- and the saturation[3] of the next link on the path, x_2.

The binary output node y_{out} gives 1 if the agent strategy is to change his path, or 0 otherwise. For the sake of simplicity, there is no hidden layer between the input and output nodes, thus the output is given by

$$y_{out} = \Theta(\cos(\alpha)x_1 + \sin(\alpha)x_2 - \theta) \qquad (2)$$

where Θ is the Heaviside step function, $\cos(\alpha)$ and $\sin(\alpha)$ are synapses weights and θ the threshold of the output node. Let us

Figure 2.a. Neural-network based strategy implementation

As you can see in Figure 2.a. (Johan Barthélemy and Timoteo Carletti 2017, p.26), this is the function that will determine the agent's decision whether to reroute or not. The normalized time x1 will be (real_time_spent divided by free_flow_time_spent), and the saturation of the next link x2 will be (current_volume divided by max_capacity). The output Heaviside step function H(n), will take value 1 if n >= 0, and will take value 0 otherwise. If the output is 1, the agent will re-route, otherwise no.

If the output value is 1, the people agent will try to compute the shortest path from its location to destination without the congested road (the next road). If there is no such path (no shortest path not including the congested road), the agent will function normally, i.e. wait until the road is no longer congested and travel normally according to the original shortest path. If there is such path, the agent will make the decision to reroute and change its shortest path avoiding the congested road.

GUI

- People agent species is displayed on main simulation with a shape of circle. There are 4 types of people, each colorized differently for visual aid:
  - Normal people agent: dim gray color.
  - People agent who has both global radio and smart reroute strategy: light coral color.
  - People agent who only have global radio: dark violet color.
  - People agent who only have smart reroute strategy: brown color.
- When people agents are clicked on main display simulation, the shortest path for that agent will be highlighted in orchid color.

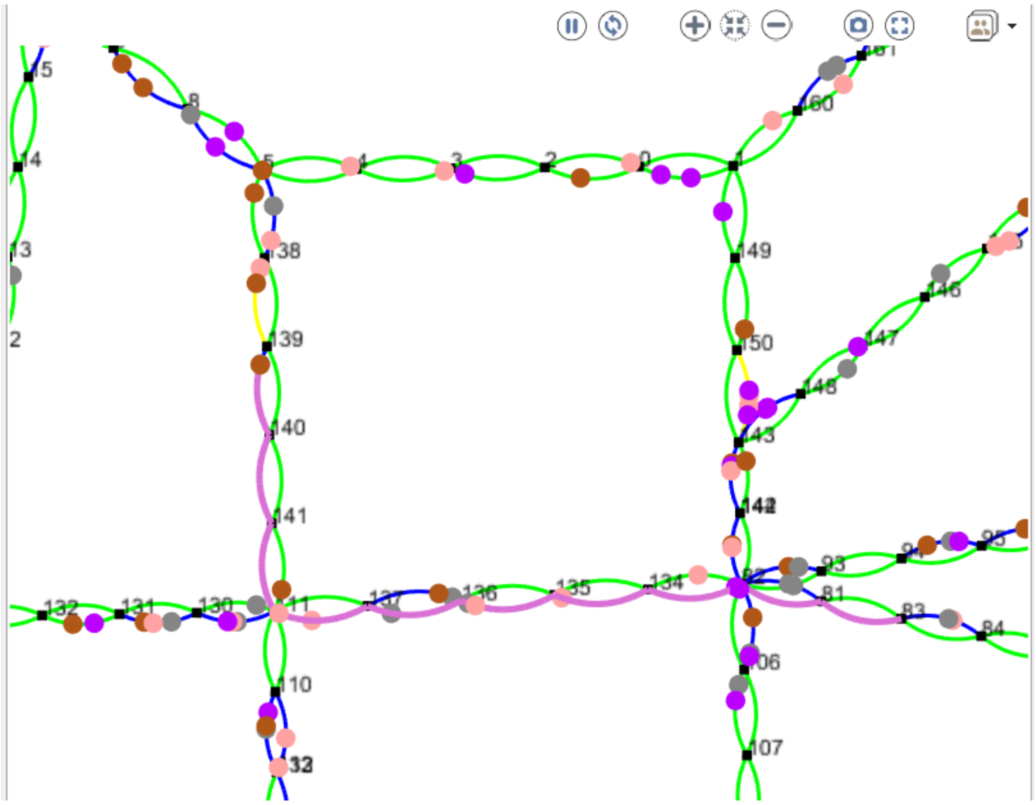


Figure 2.b. Sample display of people GUI and shortest path color in main simulation

- If people agents are in blocked road, its shape will be changed to triangle for visual aid.

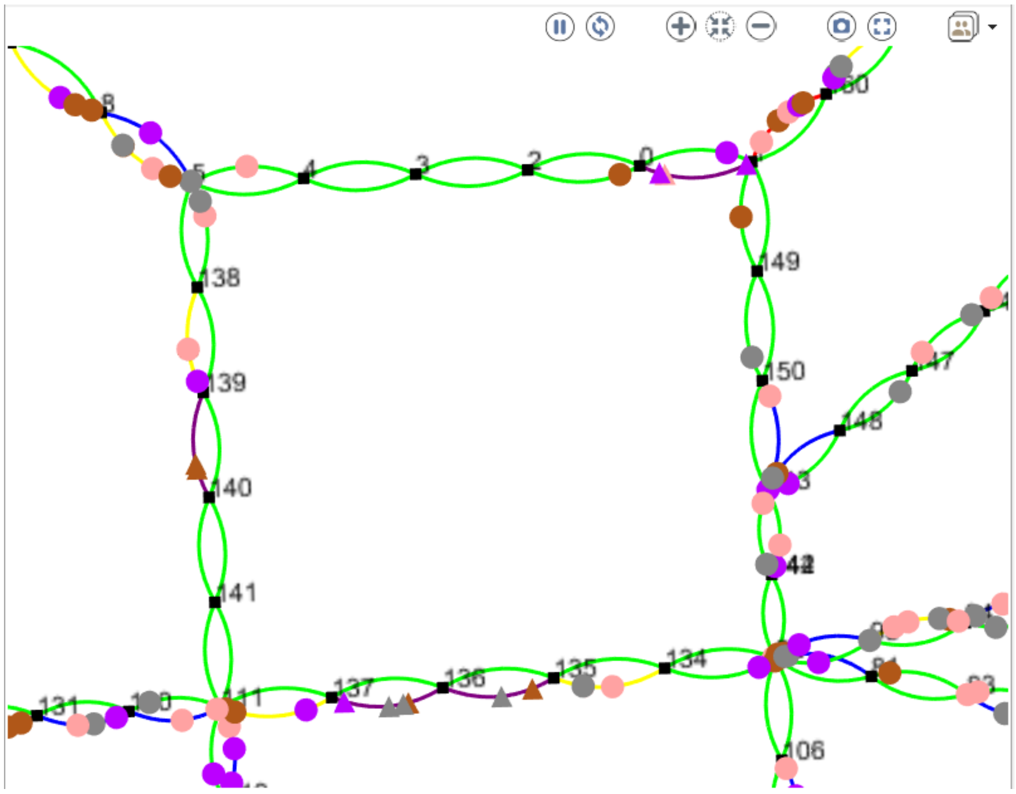


Figure 2.c. Sample display of people GUI when staying inside blocked road

2.3. Road species (Road.gaml)

Variables

- **link_length**
  Type: float (unit in meters)
  Description: Length of the road.
- **free_speed**
  Type: float (unit in meters/second)
  Description: Maximum speed to travel on this road (when there is no one on the road).
- **max_capacity**
  Type: integer
  Description: Maximum capacity of the road (i.e. maximum number of people on the road at the current moment).
- **current_volume**
  Type: integer
  Default: 0
  Description: Current volume of the road (i.e. current number of people on the road at the current moment).
- **link_full_percentage**
  Type: float
  Default: 0
  Description: The value of current road's traffic density (i.e. equals to current_volume / max_capacity). There are 5 traffic density levels based on the value of this variable:
  - Low: [0, 0.25)
  - Moderate: (0.25, 0.5)
  - High: (0.5, 0.75)
  - Extreme: (0.75, 1)
  - Traffic jam: 1
- **blocked**
  Type: boolean
  Default: false
  Description: Block status of the current road.
- **accum_traffic_count**
  Type: integer
  Default: 0
  Description: Accumulated traffic count (number of people agents passing through this road).

Functions

- ### block()

Return: nil

Description: This function will trigger when user right clicks a road and choose Block action.

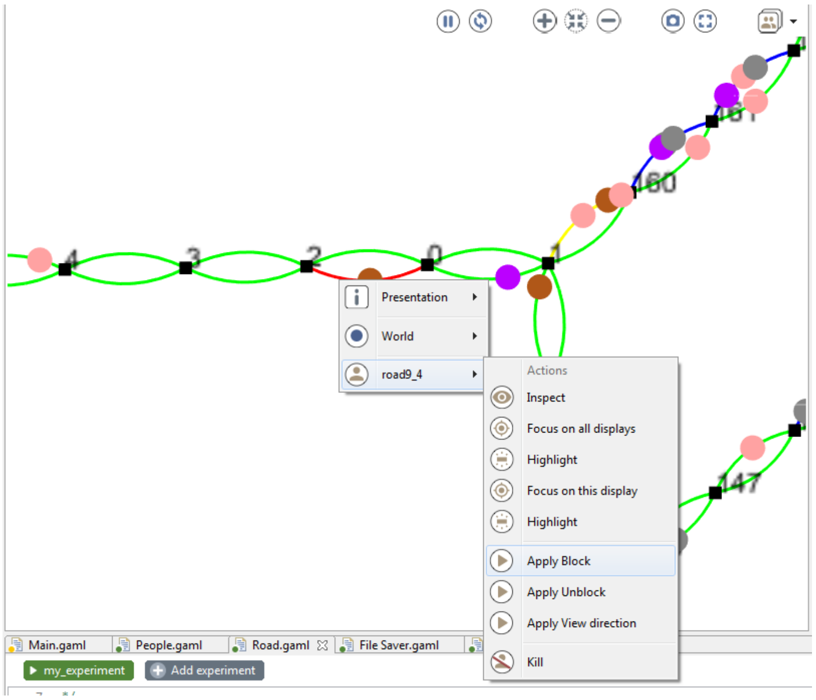


Figure 2.d. Applying block function to selected road in main simulation display

Here is the procedure:

- Change the selected road's block status (blocked) to true.
- Rebuild the graph by choosing roads agents that have blocked value of false (i.e. exclude the road just selected).
- Ask people:

  - If they are currently on the excluded road, and are currently not on one node (is_on_node = false), i.e. in the middle of the excluded road:

    +) Change people's is_in_blocked_road to true.

  - Select people who are not currently in the blocked road (is_in_blocked_road = false), and either have global radio (has_radio = true) or have their next road blocked:

    +) Re-compute the new shortest path between location and dest node. (based on new graph just rebuilt).

    +) If a possible path is not found:

    ➢ Change people's cant_find_path to true.

    ➢ Change speed to 0 m/s.

- ### unblock()

Return: nil

Description: This function will trigger when user right clicks a road and choose Unblock action. Here is the procedure:

- Change the selected road's block status (blocked) to false.
- Rebuild the graph by choosing roads agents that have blocked value of false.
- Ask people:

  - If they are currently on the newly unblocked road, and are currently not on one node (is_on_node = false), i.e. in the middle of the road:

    +) Change people's is_in_blocked_road to false.

- Select people who are not currently in the blocked road (is_in_blocked_road = false), and either have global radio (has_radio = true) or are currently on the starting point of the recently unblocked road:

  +) Re-compute the shortest path between location and dest node. (based on new graph just rebuilt).

  +) If a possible path is not found:

  - ➢ Change people's cant_find_path to true.
  - ➢ Change speed to 0 m/s.

GUI

The road has a shape of line connecting two nodes. Its color will change depending on the value of link_full_percentage to reflect current traffic density on the road:

- Low: Lime color
- Moderate: Blue color
- High: Yellow color
- Extreme: Orange color
- Traffic jam: Red color

Also, if the road is blocked, the road's color will be changed to purple.

2.4. Node species (Node.gaml)

Variables

- **node_number**

  Type: integer

  Description: ID of the node to be represented on GUI interface, from 0 to (total number of nodes - 1).

- **accum_traffic_count**

  Type: integer

  Description: Accumulated traffic count whenever new people spawn at or pass through the node's location.

GUI

The node will be represented on GUI interface as node ID number on top of the node's location.

3. Simulation

3.1. Main simulation

Simulation procedure

- Global species functions will run first:
    - +) Generate graph from shape file specified by user (shape_file_roads).
    - +) Spawn people initially using batch_create_people function
    - +) After every spawn_interval cycles, invoke generate_people function.
    - +) Run refresh_min_time function to generate a list of selected people agent who will be able to travel in the next simulation, update people agent's variable (such as distance, speed, traffic jam status, etc.) and most importantly, **change global's cycle time to be the minimum time of all people agent to reach the immediate next node on its shortest path.**
- People species functions will run next:
    - +) smart_move function will be invoked every cycle for selected people.
    - +) Additionally, if people agent has smart reroute strategy (has_smart_strategy = true), the agent will invoke will_reroute function, which will determine whether the agent will choose a different shortest path, avoiding the congested link, based on current local conditions.
    - +) People will move according to the shortest path specified, with initial BPR speed computed together with global's minimum cycle time.

(Note: See more detailed description of functions mentioned above in 2. Environment, functions part)

Sample illustration

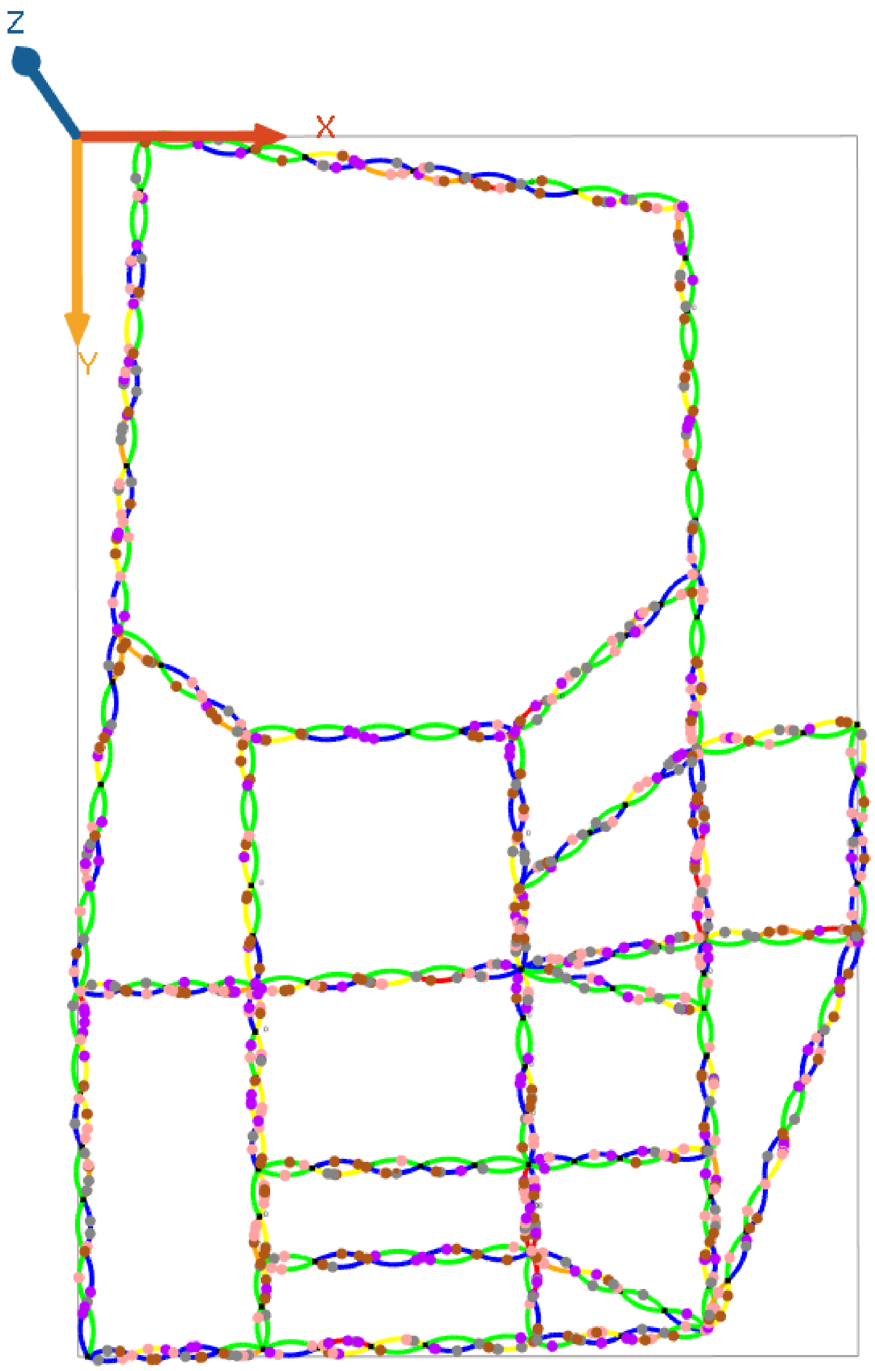


Figure 3.a. Main simulation display

In Figure 3.a., you can see that people agents are circles colorized differently based on their characteristics that are moving around on different roads, which also have different colors based on the traffic density level (see People species GUI and Road species GUI).

3.2. User command

- During the simulation, people can right click road on main display and choose to Block (block function) or Unblock (unblock function) to see how people agent changes their path.
- In summary, these are the possible cases for re-routing strategy of people agent after applying Block function to selected road;

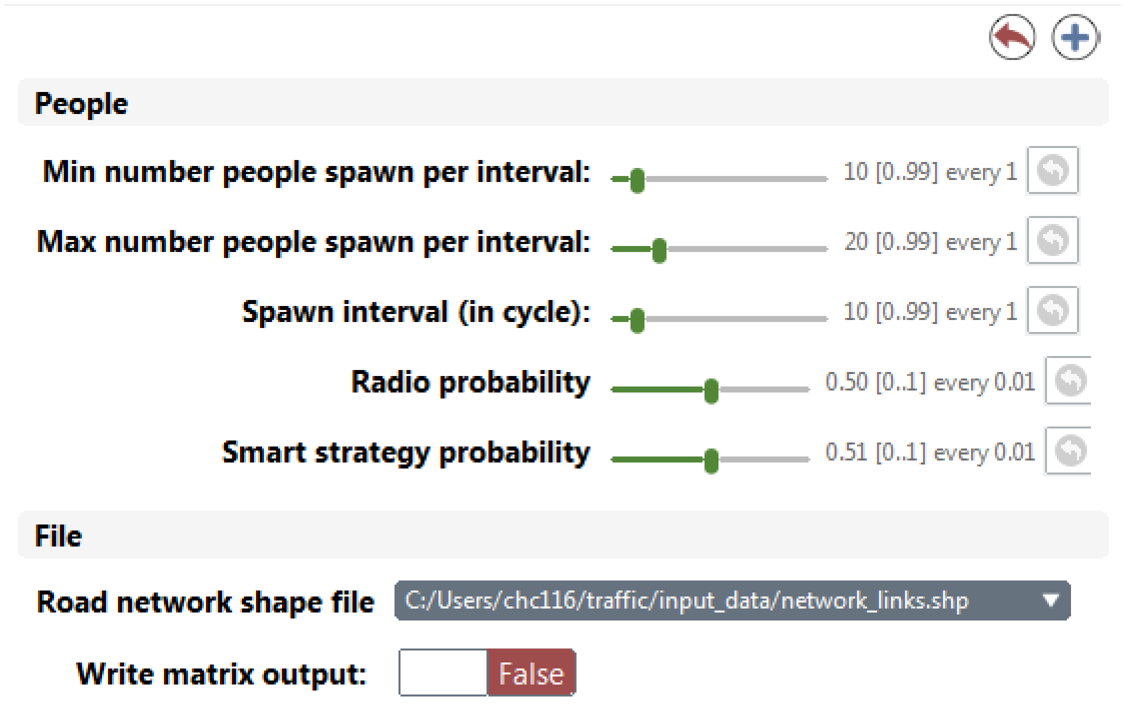
1) People who are inside blocked road, but has destination node on the same road, will continue to travel normally.

2) People who are inside blocked road, but has destination node outside of the current road, will get stuck.

3) People who are outside blocked road, and either have global radio to be notified of recent traffic changes or have their next road blocked, will try to re-route by computing new shortest path.

- If a possible new shortest path as found, travel normally
- If it's not possible (no new shortest path found), will travel to the immediate next node before getting stuck (it is certain that people agent can only be stuck on a node instead of on the middle of an unblocked road, to prevent blocking other vehicles from travelling through the same road).

3.3. Parameters View



People

Min number people spawn per interval:	10	[0..99] every 1
Max number people spawn per interval:	20	[0..99] every 1
Spawn interval (in cycle):	10	[0..99] every 1
Radio probability	0.50	[0..1] every 0.01
Smart strategy probability	0.51	[0..1] every 0.01

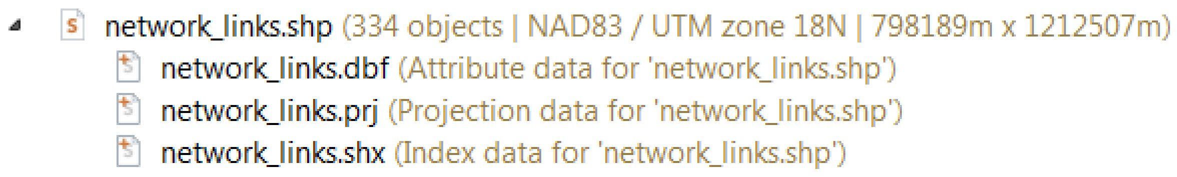
File

Road network shape file: C:/Users/chc116/traffic/input_data/network_links.shp

Write matrix output: False

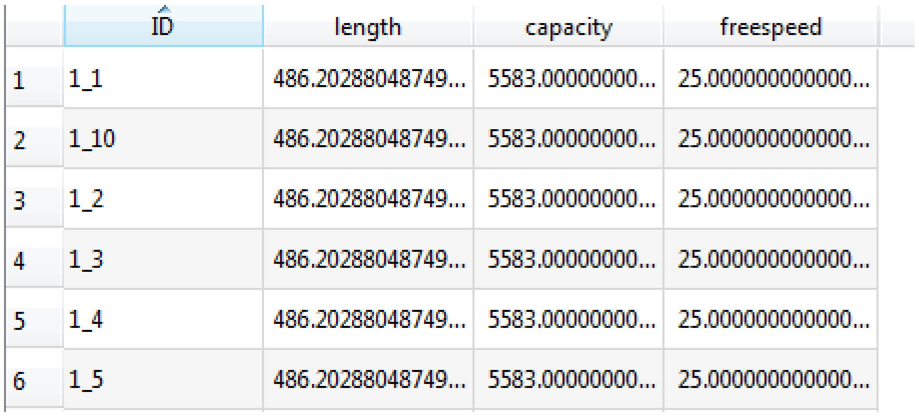
Figure 3.b. Inspect variables section

As we can see in Figure 3.b., the user can dynamically change the variables during the simulation to see how people agent adjusts their behavior. For file section, the user can specify the shape file (.shp) to be loaded into the simulation. Note that the shape file must have these accompanied files and attributes in order for the model to work correctly as specified.



network_links.shp (334 objects | NAD83 / UTM zone 18N | 798189m x 1212507m)
network_links.dbf (Attribute data for 'network_links.shp')
network_links.prj (Projection data for 'network_links.shp')
network_links.shx (Index data for 'network_links.shp')

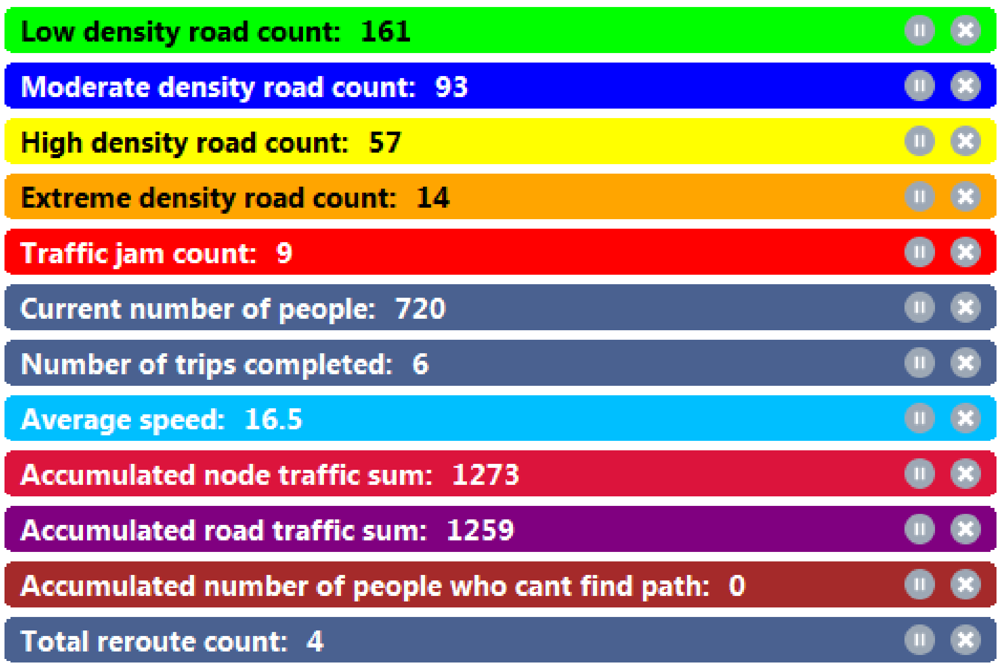
Figure 3.c. Sample input shape file



	ID	length	capacity	freespeed
1	1_1	486.20288048749...	5583.00000000...	25.000000000000...
2	1_10	486.20288048749...	5583.00000000...	25.000000000000...
3	1_2	486.20288048749...	5583.00000000...	25.000000000000...
4	1_3	486.20288048749...	5583.00000000...	25.000000000000...
5	1_4	486.20288048749...	5583.00000000...	25.000000000000...
6	1_5	486.20288048749...	5583.00000000...	25.000000000000...

Figure 3.c. Sample attribute data of shape file

3.4. Monitor



Low density road count: 161
Moderate density road count: 93
High density road count: 57
Extreme density road count: 14
Traffic jam count: 9
Current number of people: 720
Number of trips completed: 6
Average speed: 16.5
Accumulated node traffic sum: 1273
Accumulated road traffic sum: 1259
Accumulated number of people who cant find path: 0
Total reroute count: 4

Figure 3.d. Monitor variables section

- During the simulation, these information will be continuously updated in parallel with the simulation cycles to show user's important information to pay attention to:
+) Road counts categorized by 5 level of traffic density (low, moderate, high, extreme and traffic jam).
+) Current number of people.
+) Total number of trips completed.
+) Average speed value of people agent.
+) Accumulated number of people passing through roads.
+) Accumulated number of people passing through nodes.
+) Accumulated number of people who can't find the shortest path.
+) Accumulated number of times the reroute strategy has been applied (agent decide to take different path).

3.5. Output data files

- There are 3 output data files, located at **~/models/traffic-results** folder:
+) node-stats.txt:
Format: cycle,node0,node1,node2,...node-n
Description: This is a CSV file that will log the accumulated number of people passing through each node throughout the simulation.
+) road-stats.txt:
Format: cycle,road0,road1,...road-n
Description: This is a CSV file that will log the accumulated number of people passing through each road throughout the simulation.
+) matrix-stats.txt
Format:

```
        (to) node1 node2 ... node-n
(from)
node1
node2
...
node-n
```

Description: This is a 2D matrix representing origin – destination count whenever a new people agent spawns.

3.6. Further customization

- **Global seed** can be configured by user to reproduce the expected result concerning random variables (located in global species, Main.gaml file).
- The **strategy weights input** file (located at **~/input_data/strategies.txt**) can be modified to feed people agent different alpha and theta weights.
- The origin – destination matrix output is disabled by default (since large road network graph will have too many nodes to be processed, which slows down simulation time). However, for gathering data, user can turn this on in the Parameters view.

- The output data files are currently being saved every 5 cycles, this can be modified in FileSaver.gaml for personal preference.

4. Data Visualization

4.1. Road counts categorized by traffic density level

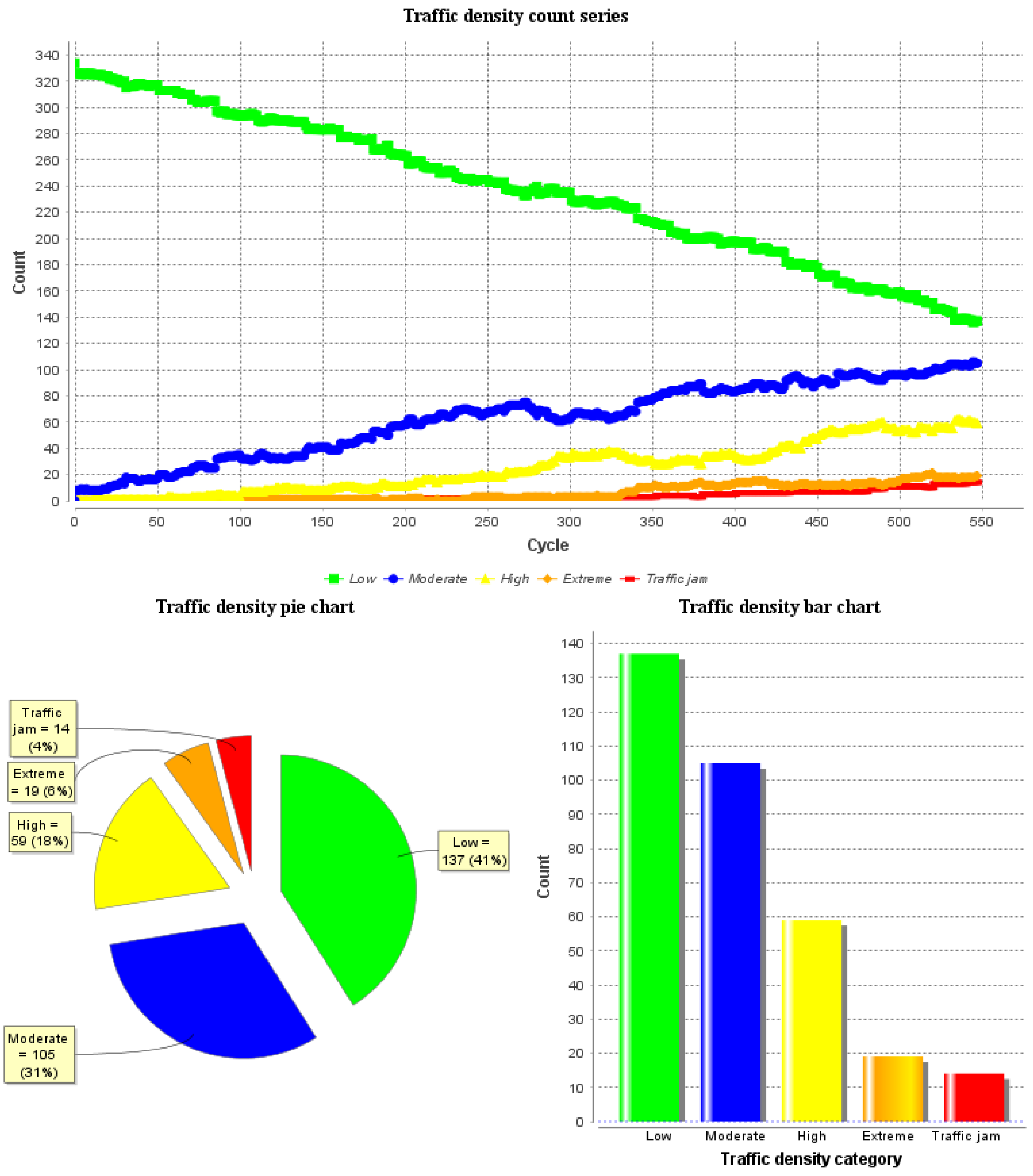


Figure 4.a. Road counts by traffic density level visualized in graphs

In Figure 4.a., at the top it’s the series showing traffic density road counts by time (in cycles). At the bottom left there is a pie chart showing percentage and value of road counts. Alternatively, at the bottom right, there is a bar chart for each type of road based on the traffic density level.

4.2. Top-k populated nodes and roads

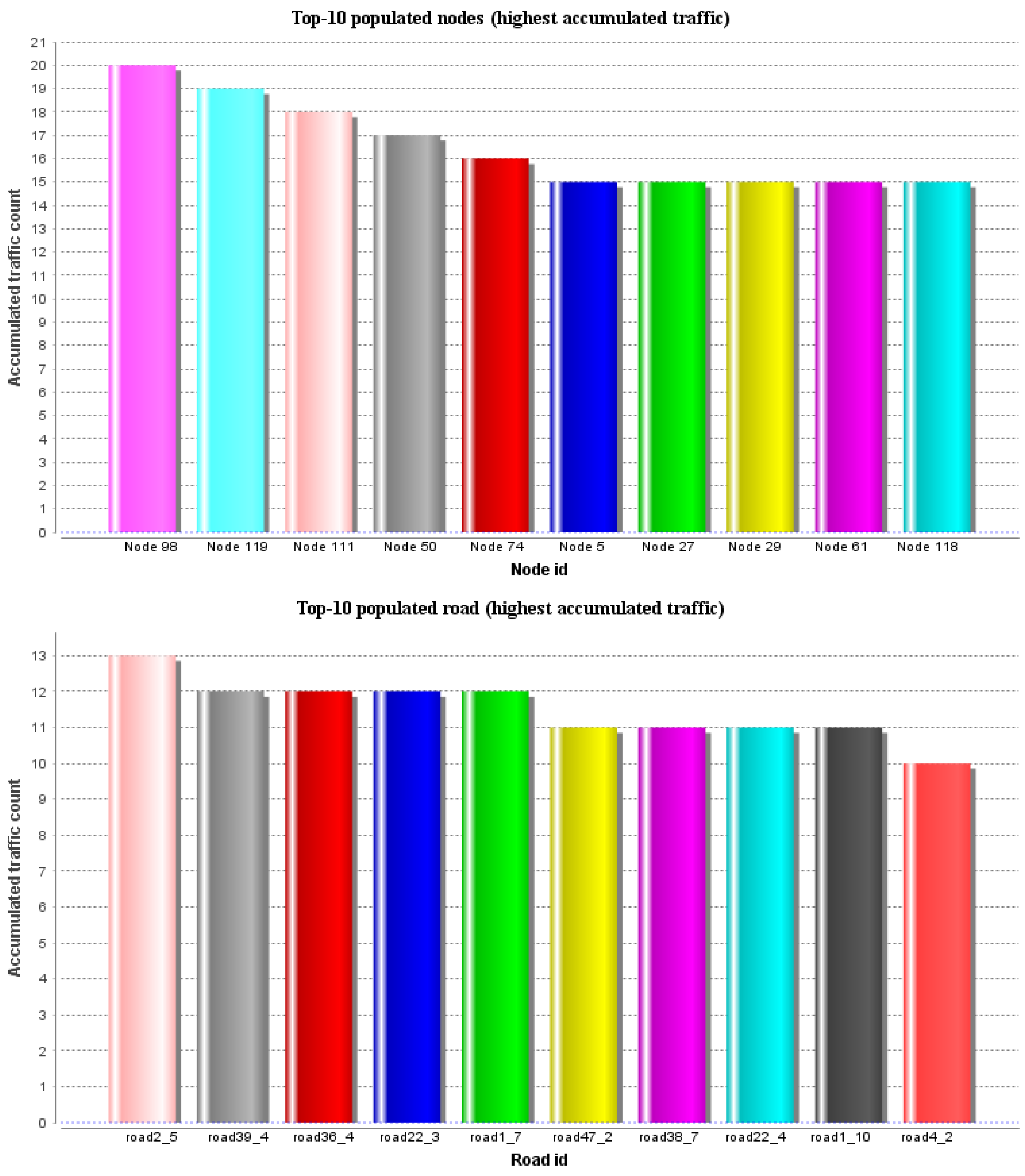


Figure 4.b. Traffic count at nodes visualized in graphs

In Figure 4.c., at the top it's the traffic count histogram illustrating the distribution of nodes according to their current traffic count. At the bottom there is a bar chart showing top-k nodes with highest current traffic count (the k parameter can be configured by user). These information can be used to find areas or intersections that have a high volume of traffic.

4.3. Speed chart

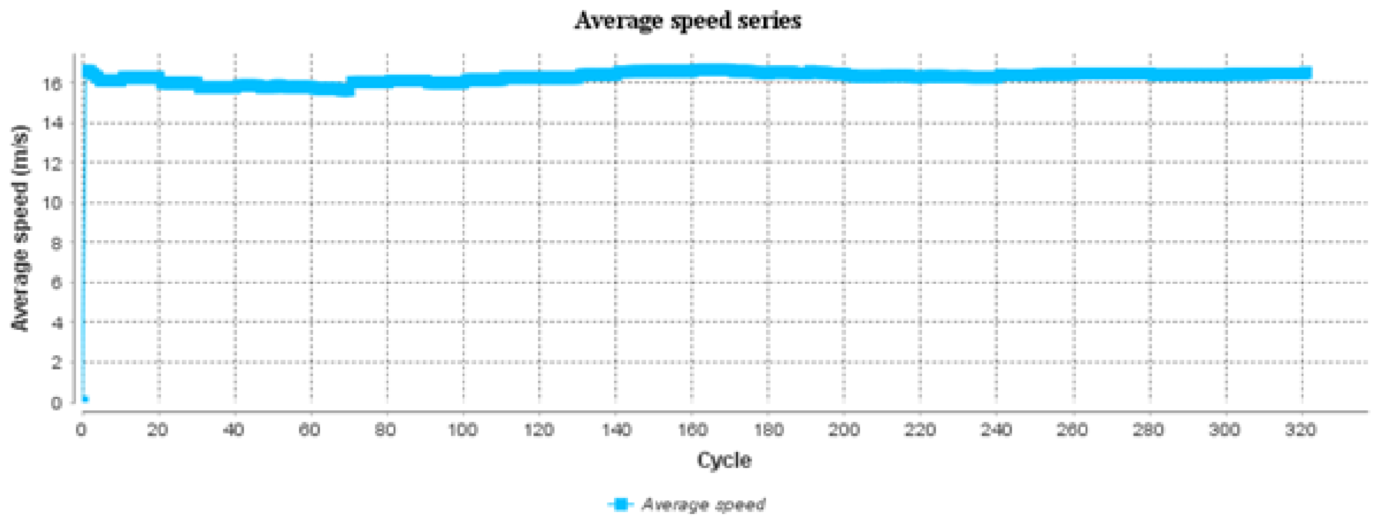


Figure 4.c. Speed parameter visualized in graphs

Figure 4.c. illustrates the average speed of all people agent over time.

5. References

Bureau of Public Roads (BPR), 2019. Route assignment - Wikipedia. [Online]
Available at: https://en.wikipedia.org/wiki/Route_assignment#Frank-Wolfe_algorithm
[Accessed 15 02 2019].

Johan Barthélemy, Timoteo Carletti, 2017. A dynamic behavioural traffic assignment model with strategic agents. *Transportation Research Part C: Emerging Technologies*, Volume 85, pp. 23-46.