

# Traffic Simulation Report (GAMA platform)

Author: Hieu Chu

Email: chc116@uowmail.edu.au

Date: 2019/03/28

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

## Table of Contents

1. Introduction.....	2
2. Environment .....	3
2.1. Class diagram.....	3
2.2. Global species (Main.gaml) .....	4
Attributes.....	4
Functions .....	5
2.3. People species (People.gaml) .....	7
Attributes.....	7
Functions .....	7
GUI.....	9
2.4. Road species (Road.gaml).....	11
Attributes.....	11
Functions .....	12
GUI.....	13
2.5. Node species (Node.gaml).....	13
Attributes.....	13
GUI.....	13
3. Simulation.....	14
3.1. Main simulation.....	14
1. Global species initialization .....	16
2. Global species reflex functions (invoked in every cycles) .....	17
3. People species reflex functions (invoked in every cycles).....	18
3.2. User command .....	18
3.3. Parameters View .....	19
3.4. Monitor.....	20

3.5.	Output data files .....	20
3.6.	Further customization .....	21
4.	Data Visualization .....	22
4.1.	Road counts categorized by traffic density level.....	22
4.2.	Top-k populated nodes and roads.....	23
4.3.	Speed chart.....	24
5.	References .....	24

## 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. Class diagram

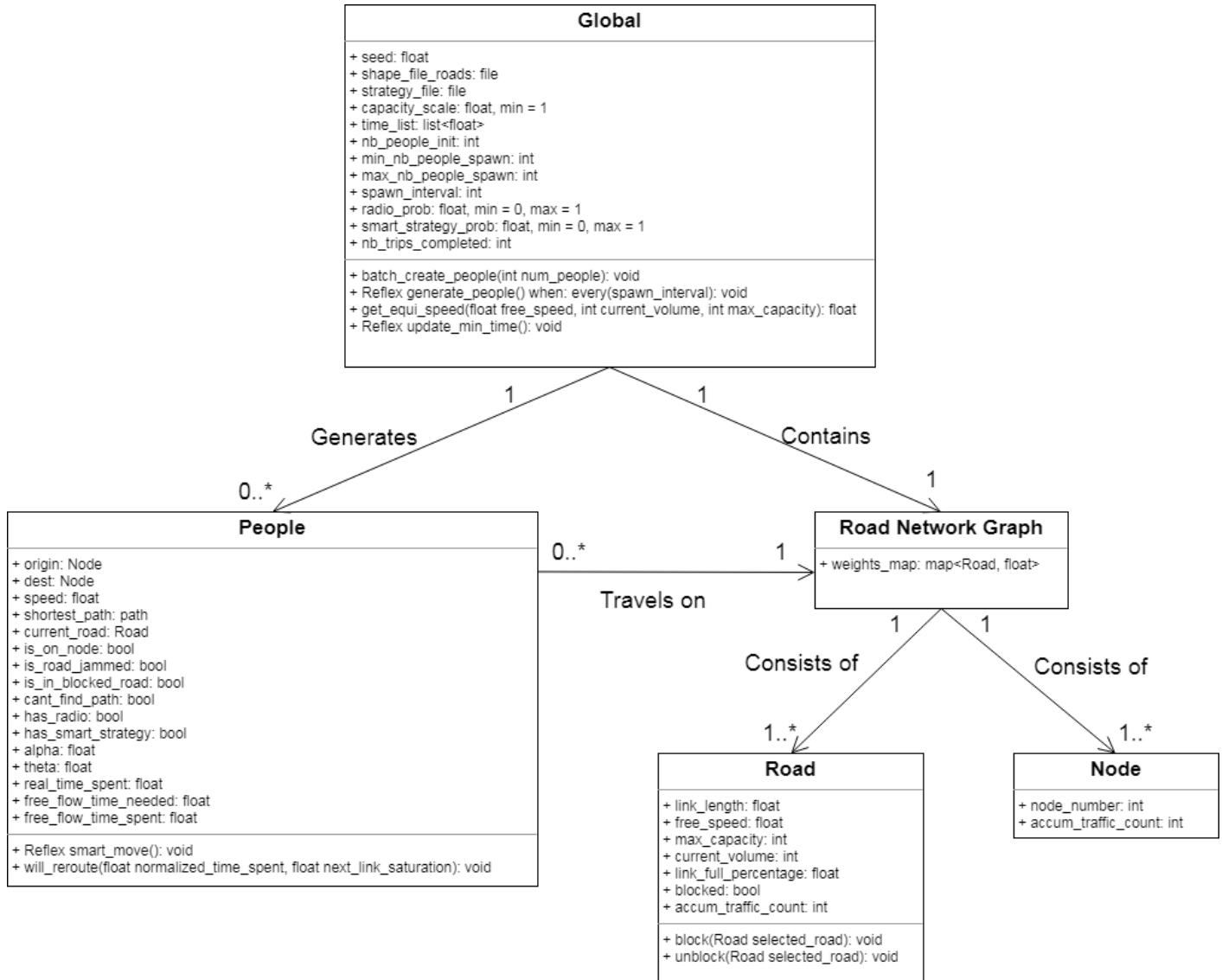


Figure 2.a. Model class diagram

The figure above is the class diagram for the simulation model. The detailed description of each class (species) are documented below.

## 2.2. Global species (Main.gaml)

### Attributes

**Table 1:** Global species environment attributes

Attribute	Description
<b>seed</b>	The random seed set by user to reproduce the expected result when using random functions.
<b>shape_file_roads</b>	Shape file loaded into the simulation. Shape file must contains attributes: ID, length, freespeed and capacity.
<b>strategy_file</b>	CSV file containing input weights (alpha, theta) for smart reroute strategy. <b>Sample format:</b> alpha, theta 2.53, 0.12 2.54, 0.16 ...
<b>my_graph</b>	The road network that will be built based on the shape file loaded (will contain road species and node species agents).
<b>capacity_scale</b>	The scale down ratio of road agents' capacity in the road network. Float value, min = 1.
<b>time_list</b>	This list will contains the time needed of each person to reach the immediate next node on his / her path to the destination node.
<b>nb_people_init</b>	Number of people initialized in the road network graph.
<b>min_nb_people_spawn</b>	Minimum number of people spawn after specified interval.
<b>max_nb_people_spawn</b>	Maximum number of people spawn after specified interval.
<b>spawn_interval</b>	After spawn_interval cycles, a random number of people will be newly generated in the road network graph.
<b>radio_prob</b>	Probability that a single people agent (when generated) will have a global radio i.e. is notified of traffic change globally (such as recently blocked or unblocked road, so that it can adapts to find a new shortest path accordingly). Min value = 0, max value = 1.
<b>smart_strategy_prob</b>	Probability that a single people agent (when generated) will have a neural-network based strategy attached, so that it can avoids congested road on the shortest path and adapts accordingly. Min value = 0, max value = 1.
<b>nb_trips_completed</b>	Total number of trips completed by people agents.

## 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() (Reflex function, invoked in every cycle)**

**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.

### 2.3. People species (People.gaml)

#### Attributes

**Table 2:** People species attributes

Attribute	Description
<b>origin</b>	People agent's location (origin node).
<b>dest</b>	The destination node.
<b>speed</b>	The speed of people agent (unit in m/s).
<b>shortest_path</b>	The shortest path from origin to destination node on the network graph.
<b>current_road</b>	Current road that people agent is on.
<b>is_on_node</b>	Boolean value to check if people agent is on the node before entering the new road.
<b>is_road_jammed</b>	Boolean value to check if the next road people agent is taking is at max capacity (cannot enter).
<b>is_in_blocked_road</b>	Boolean value to check if people agent is currently inside the blocked road.
<b>cant_find_path</b>	Boolean value to check if people agent cannot find the shortest path from origin to destination on the road network graph.
<b>has_radio</b>	Boolean value to check if people agent has global radio attached.
<b>has_smart_strategy</b>	Boolean value to check if people agent has a smart reroute strategy attached.
<b>alpha</b>	Value of alpha weights (in radians) for smart reroute strategy.
<b>theta</b>	Value of theta weights for smart reroute strategy.
<b>real_time_spent</b>	The amount of time (in second) that people agent has spent in the simulation.
<b>free_flow_time_needed</b>	The assumed free flow travel time (in second) that people agent needs to reach the immediate next node.
<b>free_flow_time_spent</b>	The free flow travel time (in second) the people 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: **will\_reroute** ([normalized\\_time\\_spent](#), [next\\_link\\_saturation](#)), where:
      - [normalized\\_time\\_spent](#) =  $\text{real\_time\_spent} / \text{free\_flow\_time\_spent}$
      - [next\\_link\\_saturation](#) =  $\text{current\_road.current\_volume} / \text{current\_road.max\_capacity}$



- **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<sup>2</sup> time spent from the source up to the current position,  $x_1$ ;
- and the saturation<sup>3</sup> 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

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

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

*Figure 2.b. Neural-network based strategy implementation*

As you can see in Figure 2.b. (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  $x_1$  will be (**real\_time\_spent** divided by **free\_flow\_time\_spent**), and the saturation of the next link  $x_2$  will be (**current\_volume** divided by **max\_capacity**). The output Heaviside step function  $H(n)$ , will take value 1 if  $n \geq 0$ , 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.

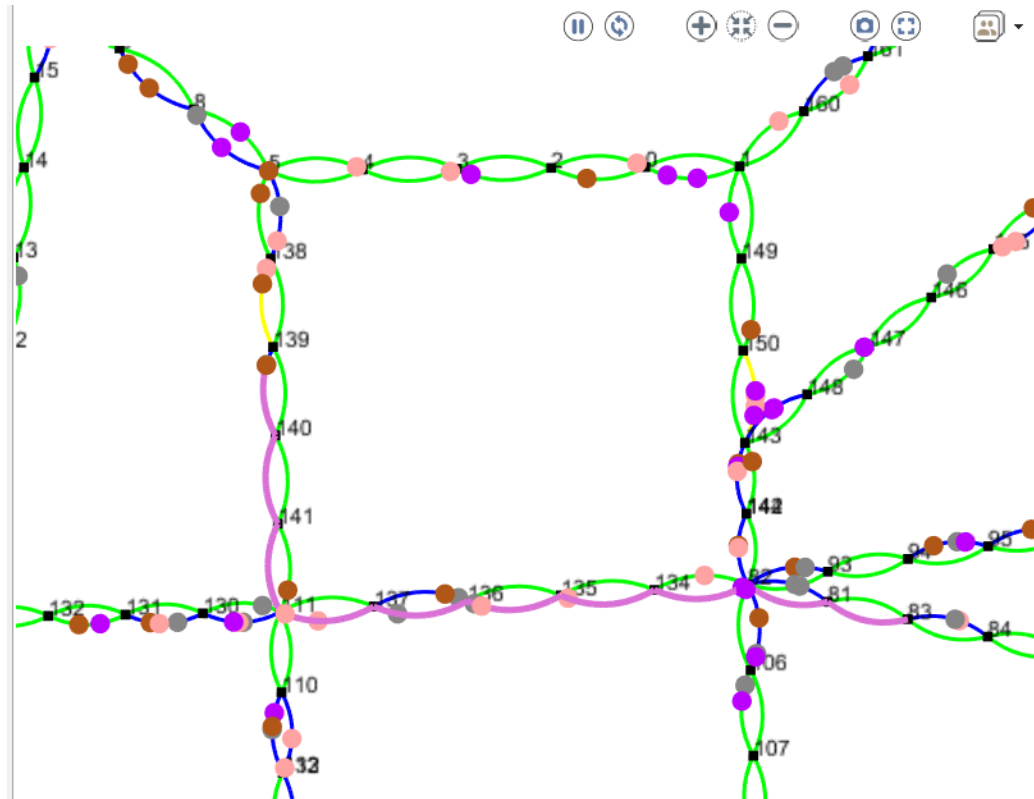


Figure 2.c. Sample display of people GUI and shortest path color in main simulation

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

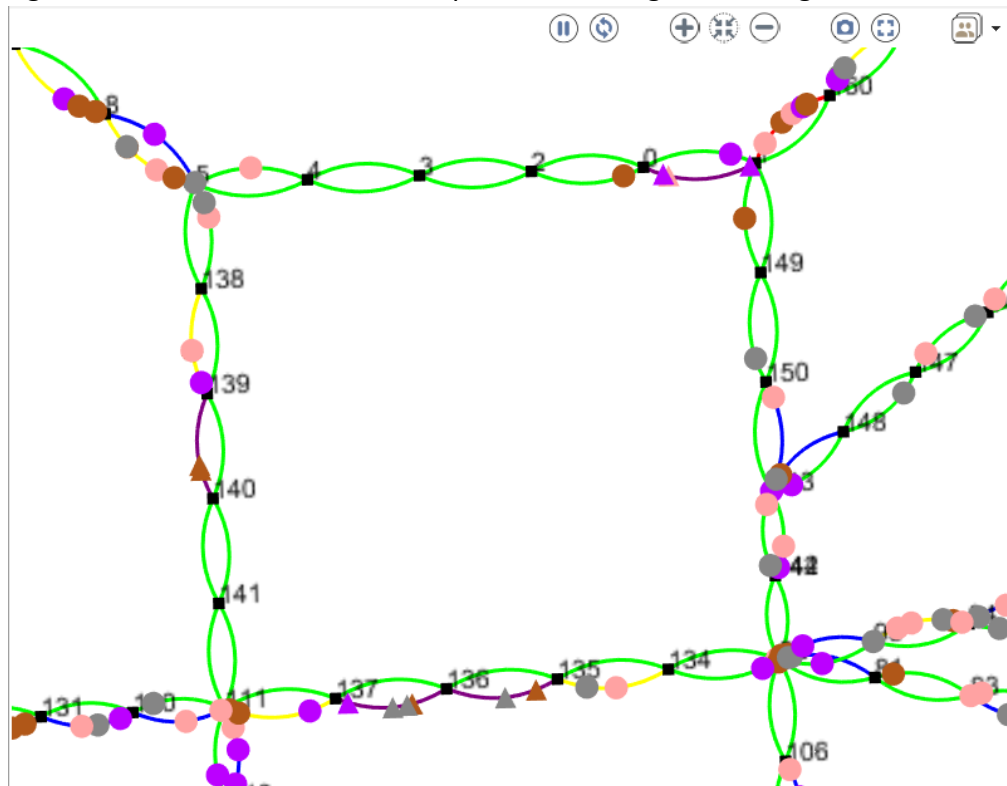


Figure 2.d. Sample display of people GUI when staying inside blocked road

## 2.4. Road species (Road.gaml)

### Attributes

**Table 3:** Road species attributes

Attribute	Description
<b>link_length</b>	Real length of the road (will be loaded from the attribute tables of the shape file). Unit in meters.
<b>free_speed</b>	Maximum speed to travel on this road (will be loaded from the attribute tables of the shape file). Unit in m/s.
<b>max_capacity</b>	Capacity of the road (will be loaded from the attribute tables of the shape file).
<b>current_volume</b>	Current volume (number of people agents) on the road.
<b>link_full_percentage</b>	The value of current road's traffic density (i.e. equals to $\text{current\_volume} / \text{max\_capacity}$ ). There are 5 traffic density levels based on the value of this variable: <ul style="list-style-type: none"><li>- <b>Low:</b> [0, 0.25)</li><li>- <b>Moderate:</b> (0.25, 0.5)</li><li>- <b>High:</b> (0.5, 0.75)</li><li>- <b>Extreme:</b> (0.75, 1)</li><li>- <b>Traffic jam:</b> 1</li></ul>
<b>blocked</b>	Boolean value to check whether the road is currently blocked.
<b>accum_traffic_count</b>	Accumulated number of people agents passing through this road agent.

## Functions

- **block()**

**Return:** nil

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

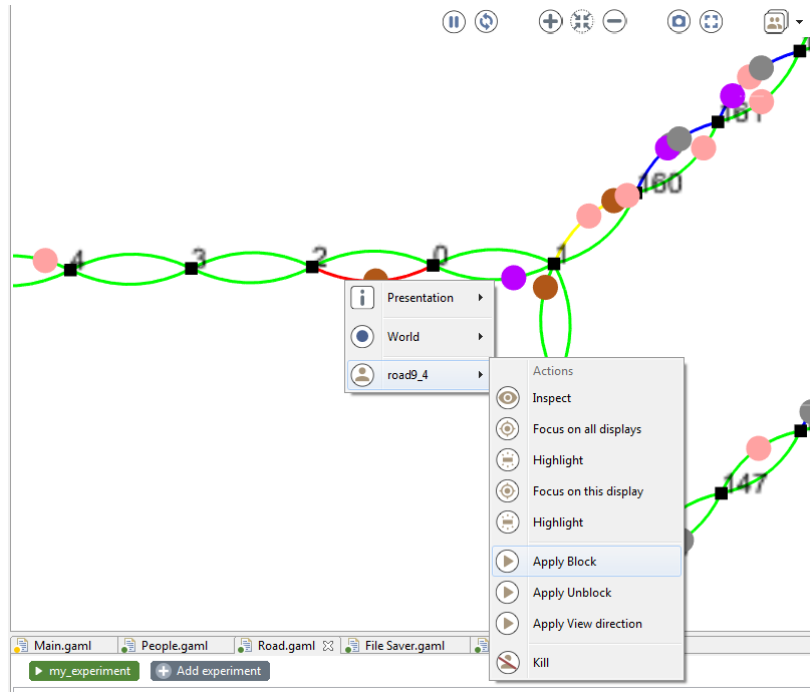


Figure 2.e. Applying block function to selected road in main simulation display

### 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 destination 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 destination 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.

## 2.5. Node species (Node.gaml)

### Attributes

**Table 4:** Node species attributes

Attribute	Description
<b>node_number</b>	ID of the node to be represented on GUI interface, range from 0 to (total number of nodes – 1).
<b>accum_traffic_count</b>	Accumulated number of people agents passing through this node agent.

## 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

This section will serve as a high-level procedure of the simulation that may not include all detailed informations. For such information, please see [2. Environment](#) part, concerning attributes and actions of agents.

**These are the general cases involving how people agent would react in the simulation:**

- At each cycle, we will find the global minimum time (considering all people agents) to reach the immediate next node. After that, in the “moving” phase, all people agents will move according to the time just found (meaning that each cycle’s time is different, and this ensures that people agent will end up at one node before entering the new road).
- If the people agent can find shortest path, he/she can travel normally.
- If the people agent can’t find the shortest path, but is already in the middle of the road, he/she will move to the nearest next node before getting stuck (to prevent future people agents from entering the same road).
- If people agent is not in the blocked road, he/she can travel normally.
- If people agent is in the blocked road, but the destination is the other end of that same road, he/she can travel normally.
- If people agent cannot find the shortest path and is currently on the node’s position, at every cycle, he/she will attempt to find a new shortest path.
- If the next road of the shortest path is not full, the people agent can travel normally.
- If the next road of the shortest path is blocked, the people agent will attempt to find a new shortest path avoiding this blocked road.
- If people agent **has smart reroute strategy attached**, is on one node’s position and can find the shortest path, he/she will attempt to find a new shortest path avoiding the next congested link if possible. If it’s not possible (all shortest paths must include the congested link), he/she will travel normally like before.
- If an interactive blocking action has been applied, only people agent which are not in a blocked road, **has global radio attached** or is nearby the blocked road, can attempt to find the new shortest path. This means that people agent **without global radio** are not notified of the traffic changes globally, and must travel to the blocked road first (then attempt to reroute) before knowing that such change happened.

Here is a diagram to show an overview of the traffic simulation steps:

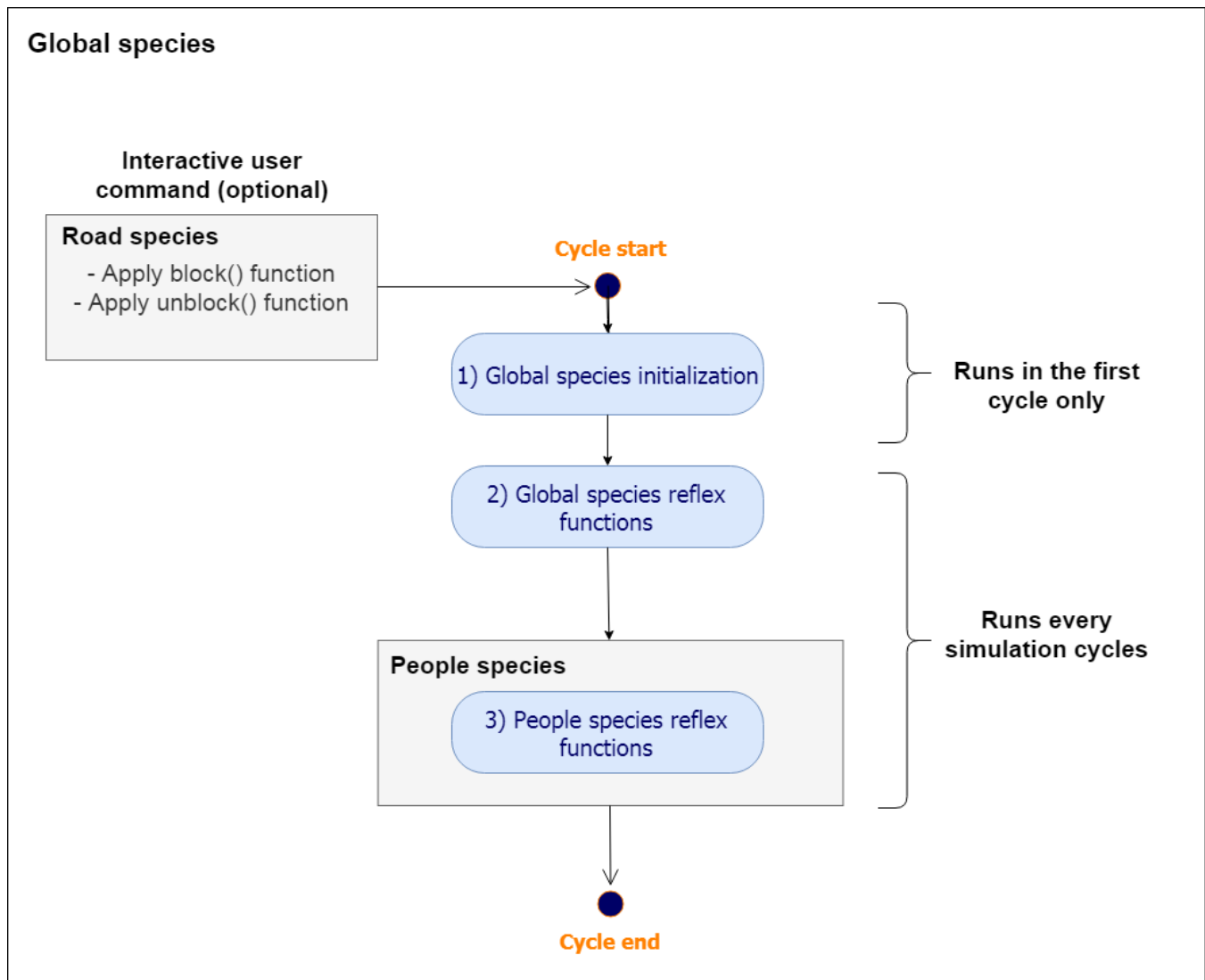


Figure 3.a. Overview of simulation steps

The figure above shows the steps of the main simulation. The standard simulation includes **3 main parts: Global initialization phase, Global species reflex actions and People species reflex actions**. They are sequentially invoked in the order above.

Furthermore, when the user decides to apply a block / unblock function to a selected road, and steps through the next cycle, the standard simulation cycle will run as normal.

Please see details of the mentioned steps in the next page.

## 1. Global species initialization

**Step 1:** Generate graph from shape file specified by user ([shape\\_file\\_roads](#)):

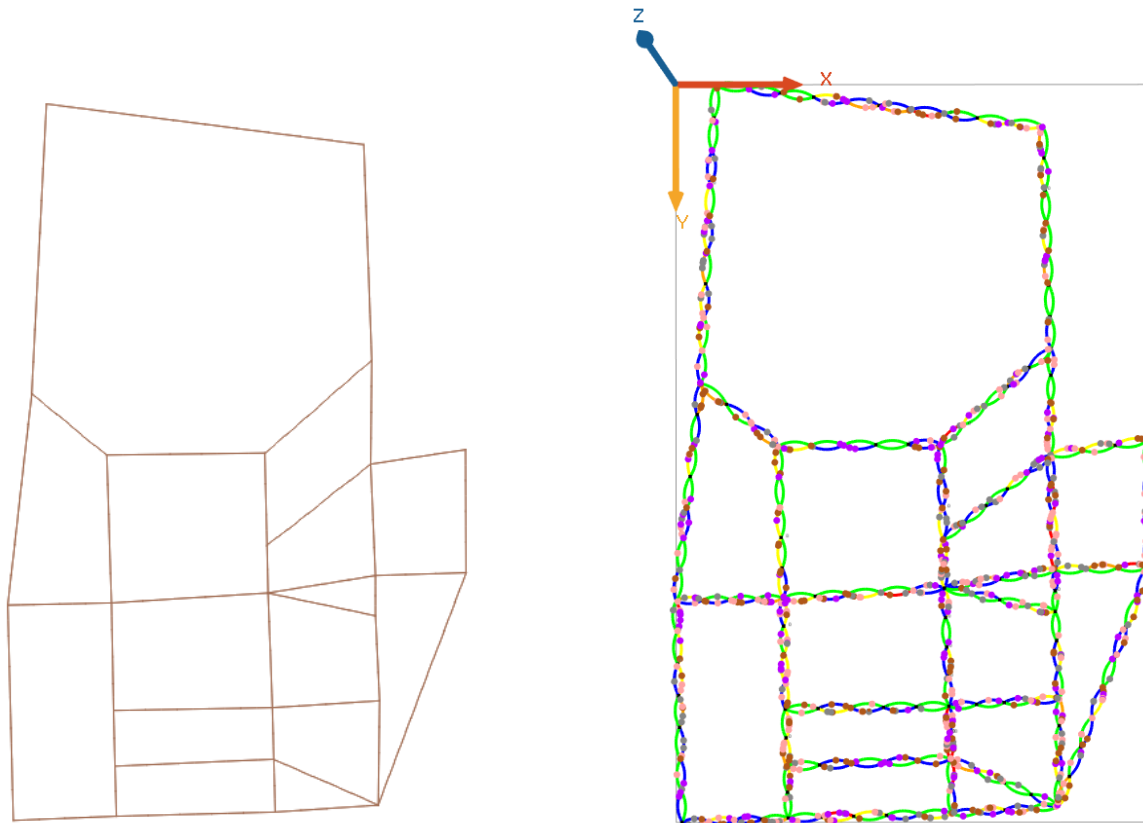


Figure 3.b. Road network from shape file – Road network in GAMA

As we can see in the figure above, the sample shape file (located at `~/input_data/network_links.shp`) is visualized in QGIS on the left (a software to visualize shape file). We will load the data from this shape file to create our road network in GAMA platform. Each link in the shape file's road network corresponds to one agent of our **road species**, and at every intersection of the road agents, an agent of our **node species** will be placed. Moreover, attributes from the shape file will also be loaded into our road agents. Here is a sample portion of the attributes table of the shape file:





- ▲  `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.d. Sample attribute data of shape file

When we generate road agents, the agents will load data from the attribute table and store these attributes to correctly reflect the original characteristics of the road network. **The road network graph** will consist of road agents and node agents created, **with edges' weights equals to each road agent's link length**.

Note that the "capacity" attribute will be scaled down based on [capacity\\_scale](#) global attribute (can be configured by user), so that different scenarios can be tested for smart rerouting strategy.

### Step 2: Generate people agents in the road network (see [batch\\_create\\_people](#))

After generating the road network graph, we will create random people agents at random nodes in the road network. Each people agent created will have a random origin, destination and a specified shortest path for the agent to follow. Moreover, the people agent will have a probability to **have a global radio attached** (to receive blocking and unblocking traffic changes globally), or **a smart reroute strategy attached**.

If the shortest path is not found between the random origin and destination, the agent will be removed from the simulation. If the shortest path is found, and if the people agent **has a smart reroute strategy attached**, we will load strategy weights (alpha and theta) from the input strategy file.

## 2. Global species reflex functions (invoked in every cycles)

### Step 1: Run [update\\_min\\_time](#) function:

- Select a list of people agent from the population that satisfy the condition to travel normally.
- Ask the list of selected people agent just found:
  - +)- Calculate the equilibrium speed that the people agent will travel at (this ensures that later people agents who enter populated road will have a lower speed, so that they cannot overtake others in a same-lane road).
  - +)- Calculate the real time to for the people agent to reach its immediate next node.
- Find the global minimum time (considering all people agents) to reach the immediate next node, and set the current cycle's time to be equal to that. This ensures that after the "moving" phase of people species, people agent must end up at one node before entering the new road).

### Step 2: Generate random people agents in the road network after some cycles (see [generate\\_people](#)).

### 3. People species reflex functions (invoked in every cycles)

**Step 1:** People agents, who cannot find the shortest path and is currently stuck at one node's position, will attempt to find new shortest path between the origin and destination.

**Step 2:** People agents, who satisfy the conditions to travel normally, will follow the shortest path as specified (see [smart\\_move](#)):

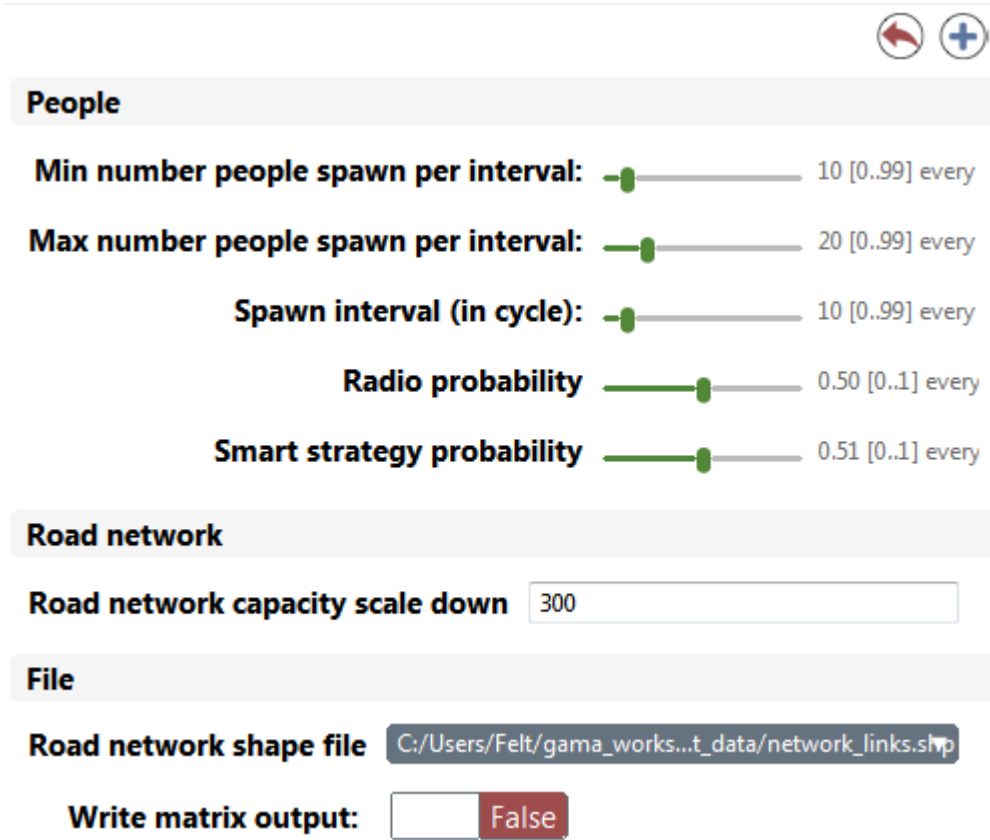
- Move according to the shortest path specified, with speed and time pre-computed in global species update\_min\_time function.
- If people agent reaches the final node, it will be removed from the simulation
- If people agents reaches a non-final node, and the next road is blocked, the agents will attempt to reroute by finding a new shortest path between origin and destination.

**Step 3:** People agents, who **have smart reroute strategy attached**, is on one node's position and can find the shortest path, will attempt to find a new shortest path avoiding the next congested link if possible. If it's not possible (all shortest paths must include the congested link), he/she will travel normally like before (see [will\\_reroute](#) for more details).

#### 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 (this means that people agent **without global radio** are not notified of the traffic changes globally, and must travel to the blocked road first (then attempt to reroute) before knowing that such change happened).
- 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



The Parameters View interface is divided into three main sections: People, Road network, and File. At the top right, there are two circular icons: a red arrow pointing left and a blue plus sign. The 'People' section contains five sliders: 'Min number people spawn per interval' (value 10, range [0..99]), 'Max number people spawn per interval' (value 20, range [0..99]), 'Spawn interval (in cycle)' (value 10, range [0..99]), 'Radio probability' (value 0.50, range [0..1]), and 'Smart strategy probability' (value 0.51, range [0..1]). The 'Road network' section has a text input field for 'Road network capacity scale down' with the value 300. The 'File' section has a text input field for 'Road network shape file' with the value 'C:/Users/Felt/gama\_works...t\_data/network\_links.shp' and a checkbox for 'Write matrix output' which is currently set to 'False'.

Section	Parameter	Value	Range	Unit
People	Min number people spawn per interval	10	[0..99]	every
	Max number people spawn per interval	20	[0..99]	every
	Spawn interval (in cycle)	10	[0..99]	every
	Radio probability	0.50	[0..1]	every
	Smart strategy probability	0.51	[0..1]	every
Road network	Road network capacity scale down	300		
File	Road network shape file	C:/Users/Felt/gama_works...t_data/network_links.shp		
	Write matrix output	False		

Figure 3.e. Inspect variables section

As we can see in the figure above, the user can dynamically change the variables during the simulation to see how people agent adjusts their behavior. For the road network graph, user can specify the scale down ratio of road agent's capacity so that different scenarios can be simulated (note that if you change this value, do reset the simulation to rebuild the network graph). 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 (see [Figure 3.c.](#) and [Figure 3.d.](#)).

### 3.4. Monitor

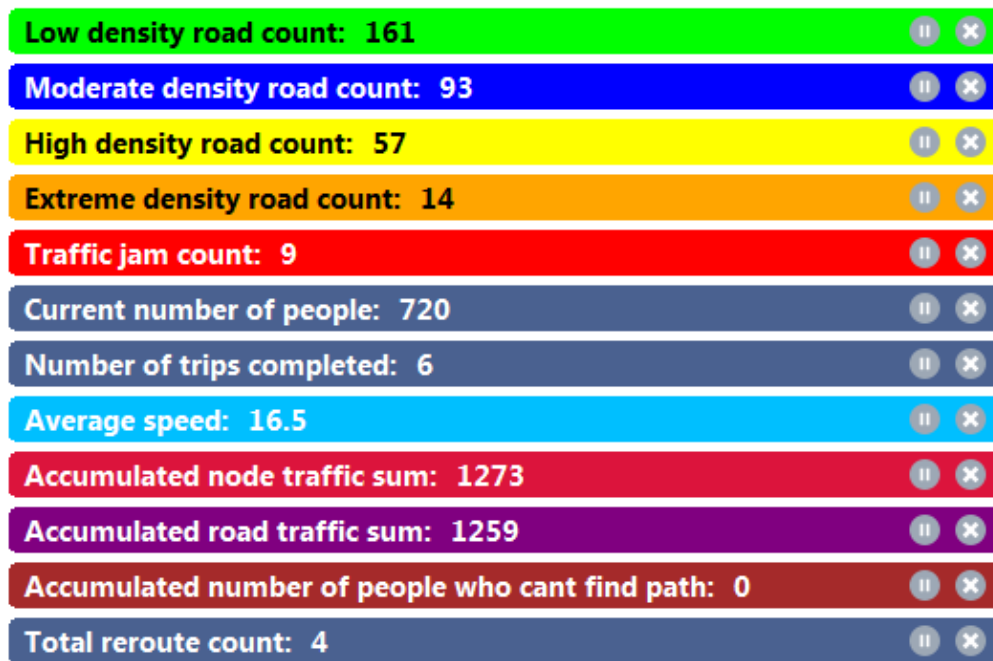


Figure 3.f. Monitor variables section

- 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

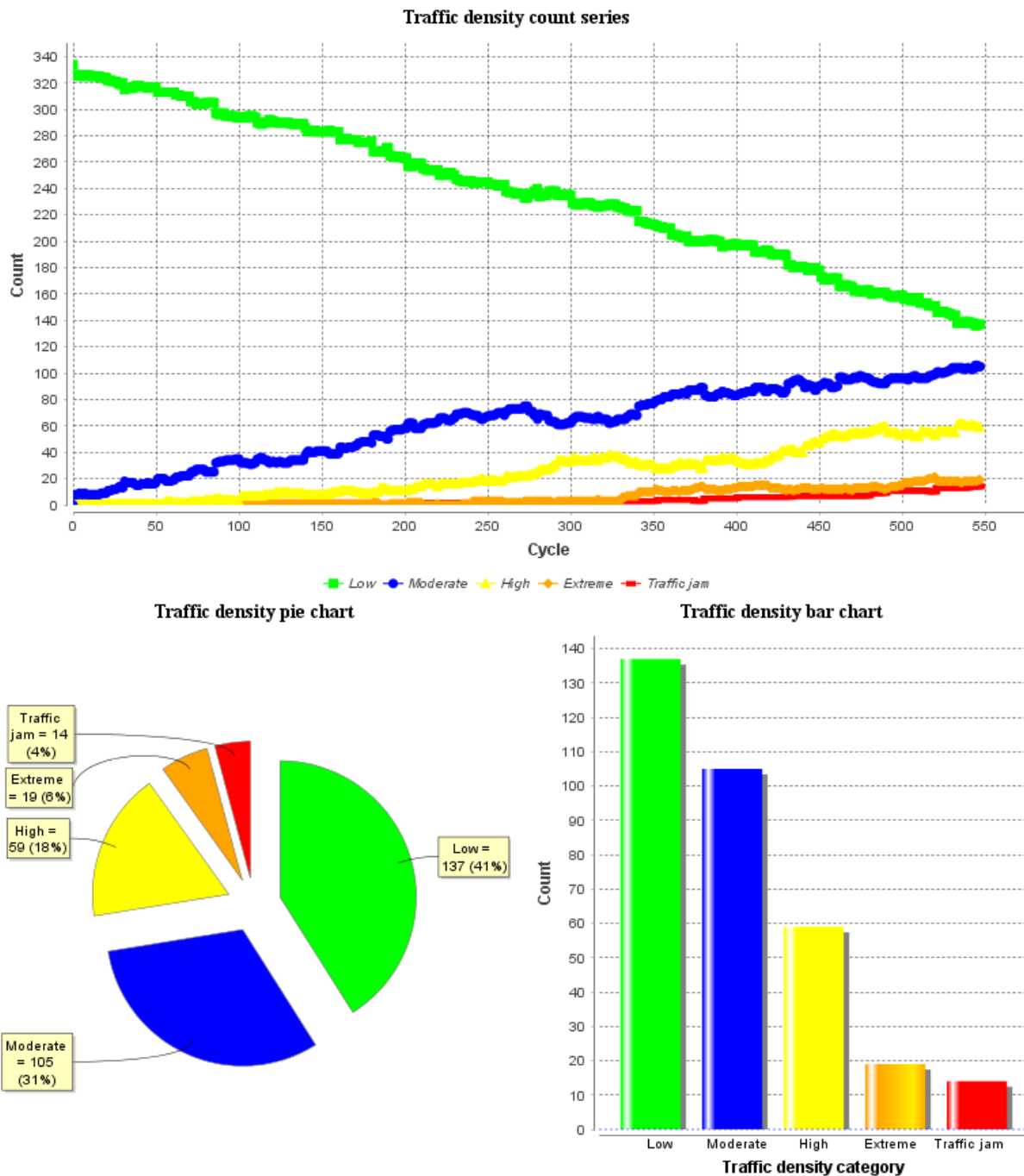


Figure 4.a. Road counts by traffic density level visualized in graphs

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

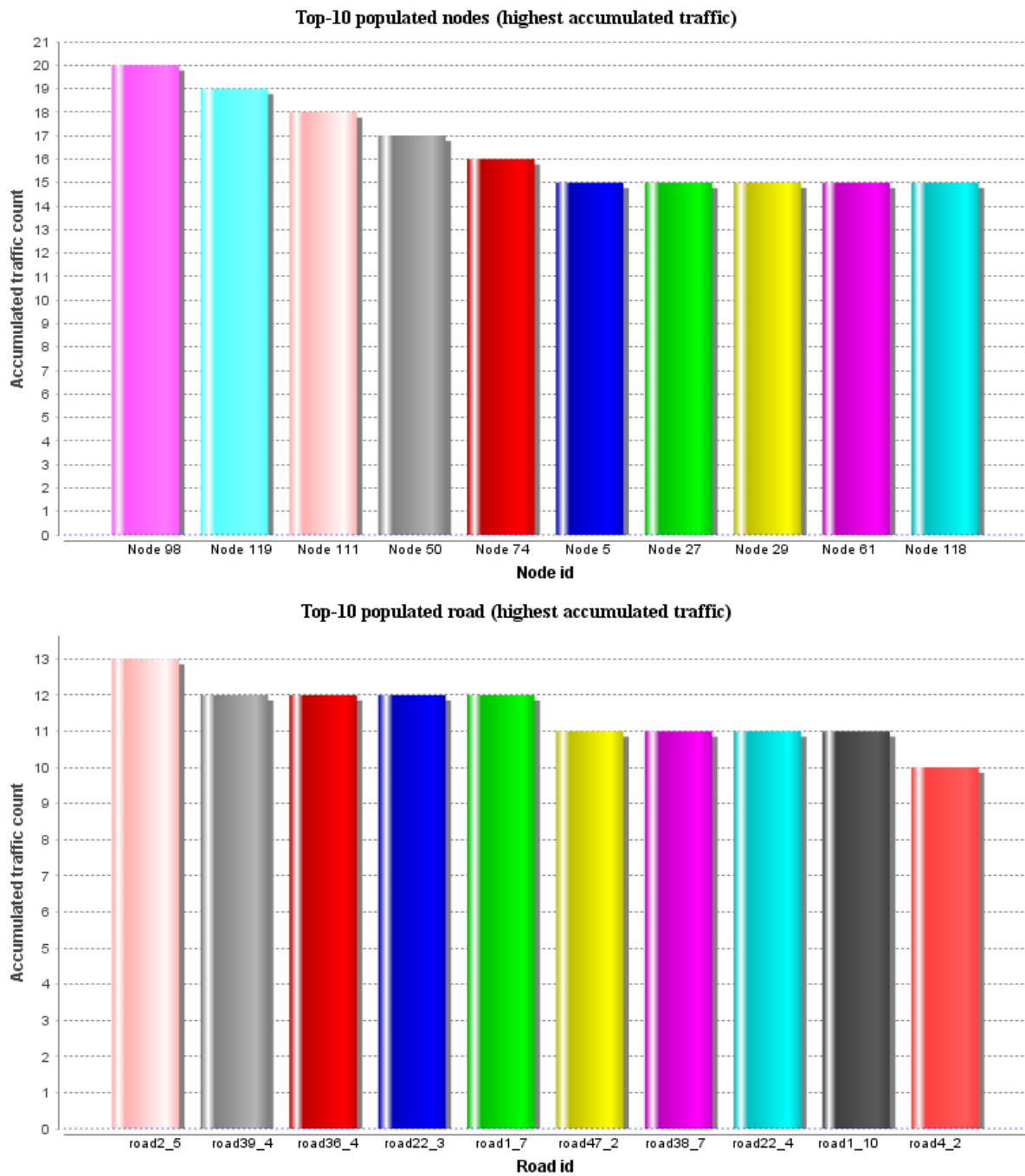


Figure 4.b. Traffic count at nodes visualized in graphs

In Figure 4.b., 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

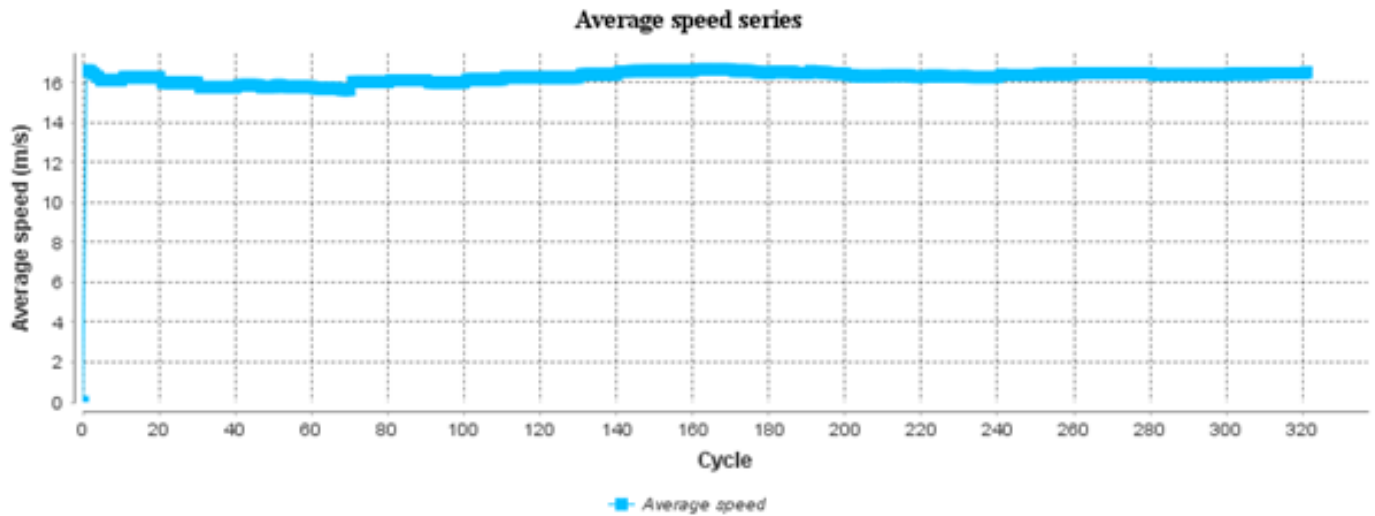


Figure 4.c. Speed parameter visualized in graphs

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](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.