`EvolveTraffic:`

A Traffic Microsimulation Model
for Bridge Loading

Version 1.0.1

User Manual

(Version 0.91)

University College Dublin, Ireland

Laboratoire Central des Ponts et Chaussées, France

Acknowledgements

<div align="right">

Dr Colin Caprani,

August 2008

</div>

Contents

1. Introduction

1.1 The `EvolveTraffic` *Program*

`EvolveTraffic` is a program that simulates driving behaviour for the purpose of estimating highway bridge loading. Vehicles are read into the program and they are simulated driving along a road which may have speed limit sections and gradient sections. The vehicles interact with one another and may perform lane changes. Various statistics of the traffic can be output at various points along the road. When the vehicles pass the Output Detector they are written to the output file.

The output traffic file contains the same vehicles as the input traffic file, but represents a different configuration (or order) of the traffic, one that depends on the features the user has specified for the traffic and road. This new configuration of traffic is useful for bridge loading because it essentially extends measured data, which is expensive to obtain.

1.2   The User Manual

Purpose

This User Manual has been written to explain the use of the `EvolveTraffic` program, and to explain its capabilities and limitations. It explains the user interface, the required input information, and the output information.

This manual does not explain the Intelligent Driver Model (IDM) used for the traffic microsimulation in the program. Further, it does not explain how the input data is to be prepared or how the program output is to be used.

Conventions

Menus

Menu commands are written as:

        Configuration > Simulation

And mean the user should select Simulation from the Configuration menu as shown:

Keyboard

The use of shortcut keys is indicated as:

        Ctrl+I

And means that the user should press the Control and "I" keys simultaneously.

Notices

Points of significant importance are denoted as:

> **<u>Important!</u>**

Typically, failure to adhere to these points will result in unexpected behaviour or a program crash.

Parameters that may be changed in the EvolveTraffic.ini configuration file are coloured in green as:

> 3.5 tonnes

Values in this text are default values and may be changed by a Developer User.

Glossary

Detector	A 'virtual' loop detector – a location on the road where information on the traffic flow passing that point is abstracted.
IDM	Intelligent Driver Model – the microsimulation model used in the current version of the program.
Invisible Mode	A simulation is run with no output to screen (except for the progress bar indicator) and no user interaction allowed.
MOBIL	Minimize Overall Braking Induced by Lane changes – the lane change model developed by Treiber.
Output Detector	When a vehicle passes this form of detector, it is written to file along with its arrival time. Every road and direction must have these detectors since they are the reason for the program's development.
Visible Mode	A simulation is run so that the vehicles are animated on the screen and user interaction with the program can occur.
Road Feature	Any 'feature' of the road – currently only speed limit sections of road, and sections of road with gradient are considered.
SDI	Single Document Interface – the program operates on one `EvolveTraffic` file at a time, similar to the way NotePad in Windows operates.
Treiber	Dr. Martin Treiber, Technische Universtät Dresden, developer of the IDM traffic microsimulation model [2].

1.3   Release History

`EvolveTraffic` **Program**

Version	Date	Description
1.0	22/8/08	Initial release.
1.0.1	1/9/08	Bug fix: SAFT file format – removed unnecessary GVW kN to kg/100 conversion.

`EvolveTraffic` Manual

Version	Date	Description
0.9	22/8/08	Initial release.
0.91	1/9/08	• Updated EvolveTraffic version number. • Added manual and program release histories.

2. Installing the Program

2.1 System Requirements

The program has been tested on Windows XP. It is expected to run on Windows Vista and on Windows 2000 also.

The program is a single-threaded application and so cannot take advantage of multi-core processors. Therefore for maximum speed, prefer a computer with a fast single processor over a computer with a multi-core slower processor.

The program does not require much memory because:

- the input file is read in line by line as required with a buffer of (default) 10 vehicles being held;
- the memory a vehicle uses is released when the vehicle leaves the road;
- memory used for the storage of the metrics detectors data is released upon writing to file, which occurs at user-specified intervals (usually about 60 seconds of simulation time).

For long roads with a large output time interval a computer with 1 GB of RAM is recommended so that other programs can continue to operate successfully.

2.2   Installation

To install the program, carry out the following steps:

1. Extract the zip-archive to a folder;

2. In that folder, double click on the file `Setup.exe`;

3. Follow the on-screen instructions to install the program.

3. About `EvolveTraffic`

3.1 Introduction

`EvolveTraffic` broadly performs the following steps:

4. A file containing a record of vehicles is opened and linked to (the 'traffic file');

5. The simulation commences;

6. Vehicles are read from the traffic file and enter the road at the time indicated in the traffic file;

7. The vehicles proceed along the road with the driving behaviour simulated user a traffic microsimulation model;

8. When a vehicle arrives at an Output Detector, it is written to the output traffic file, with the time of arrival at the Output Detector.

9. The simulation progresses until all vehicles have been read from the traffic file and have exited the road.

3.2 `EvolveTraffic`*: Capabilities and Limitations*

Capabilities

`EvolveTraffic` is able to:

- Read data from CASTOR or SAFT format vehicle files;

- manipulate vehicles according to the IDM traffic microsimulation model as they progress along a road;

- consider 5 classes of vehicle: cars, small trucks, large trucks, cranes and low-loaders; especially chosen for their relevance to bridge loading.

- model roads of up to 8-lanes in width, 4 per direction;

- model one or two directions at the same time, each of up to 4 lanes;

- take account of speed limit sections of road;

- take account of gradient sections of road;

- consider overlapping gradient and speed limit sections of road;

- output vehicles to file along with their arrival times at the output detector;

- output Flow & Density information for specified locations on the road at specified time intervals, for specified classes of vehicles;

- output Headway information for specified locations on the road at specified time intervals, for specified classes of vehicles;

- output traffic Composition information for specified locations on the road at specified time intervals;

- output Lane Change information, by direction, at specified time intervals, for specified classes of vehicles;

Limitations

`EvolveTraffic` is not able to:

- determine the number of lanes or directions of traffic in the specified input traffic file, in advance of the simulation;

- determine the input traffic file format (whether CASTOR or SAFT);

- calculate load effects on bridges;

- model on-ramps, curves in the road or lane closures;

- model different lengths of lanes or directions;

- Cars-only lanes cannot yet be modelled.

3.3 User Levels

There will be two main types of user of `EvolveTraffic`:

Ordinary User

This type of user is typically interested in running the program to generate new traffic configurations. They will be using a set of IDM parameters previously calibrated, but may be considering different road layouts. Usually, they will not require the traffic metrics detectors output. Indeed once the road layout has been developed, this user will probably run the program in Invisible Mode, to get the output traffic file as quickly as possible.

Developer User

This user is using the program to calibrate it against measured real traffic. This user will be using the traffic flow metrics detectors in order to compare measured traffic properties against the program's output in order to develop a set of IDM parameters that model the measured traffic. This user may also need to change some of the program configuration settings in the `EvolveTraffic.ini` file in order to achieve this goal. This user will mostly work in the Visible Mode for visual verification of the traffic behaviour, and to use the DataTips to see vehicle information during the simulation.

4. The `EvolveTraffic` User Interface

4.1 Introduction

The `EvolveTraffic` program has a Single Document Interface (SDI). This means only a single `EvolveTraffic` file can be opened at any one time – similar to the way NotePad in Windows operates. The menu and toolbar commands, whilst being specific to the program, are operated in the same way as standard Windows programs.

Of note, for ease of use, there is extensive support for keyboard shortcuts in `EvolveTraffic`. This makes it much quicker and easier to use the program features during a simulation, whilst in Visible Mode.

4.2   Menu

File

- File > New:

  Opens a new *.etr file with some default settings.

- File > Open:

  Displays the Open File dialog for choosing an existing *.etr file.

- File > Save:

  Saves the current file.

- File > Save As:

  Allows the current file to be saved as a different name.

- File > Save Image:

  Saves the view currently visible to a *.bmp file specified in the Save dialog. This feature is useful for including configurations in reports/presentations etc. and replaces the need for using the Windows Print Screen command.

- File > Print:

  Prints the current view – this feature is not supported as may be seen from the Print Preview command.

- File > Print Preview:

  Displays a preview of the output that may be sent to the printer.

- File > Print Setup:

  Specify the setup to be used when printing.

- File > Recently Used Files:

  A list of recently used files for ease of access.

- File > Exit:

  Closes the program.

Configuration

- Configuration > Simulation:

  Brings up the Simulation Configuration dialog box – see Section 5.

- Configuration > Traffic Model:

  Brings up the Traffic Model Configuration dialog box – see Section 5.3.

- Configuration > Road Features:

  Brings up the Road Features Configuration dialog box – see Section 5.4.

- Configuration > Traffic Flow Metrics:

  Brings up the Traffic Flow Metrics Configuration dialog box – see Section 5.5.

Run

- Run > As Visible:

  Starts the simulation in the Visible Mode – see Section 6.2. If the simulation is currently paused, this restarts the simulation as per Tools > Pause (Resume).

- Run > As Invisible:

  Starts the simulation in the Invisible Mode – see Section 6.2.

Tools

- Tools > Zoom In:

  Zooms into the road, making the road and vehicles larger on the screen.

- Tools > Zoom Out:

  Zooms out from the road, making the road and vehicles smaller on the screen.

- Tools > Speed Up:

  Speeds the simulation up – the ratio of simulation time to real time increases.

- Tools > Slow Down:

  Slows the simulation – the ratio of simulation time to real time reduces.

- Tools > Pause (Resume):

Once a simulation is running, this pauses it. If a simulation is paused, this command changes to Resume, and resumes the simulation when clicked.

- Tools > Stop:

Stops the current simulation.

- Tools >Preferences:

Displays the Preferences dialog – see Section 6.3.

View

- View > Toolbar:

Choose whether or not to display the toolbar.

- View > Status bar

Choose whether or not to display the status bar at the bottom of the screen, which contains the progress bar during the simulation.

Help

- Help > About:

This shows information about the copyright/developers and version of the program running.

4.3 Toolbar
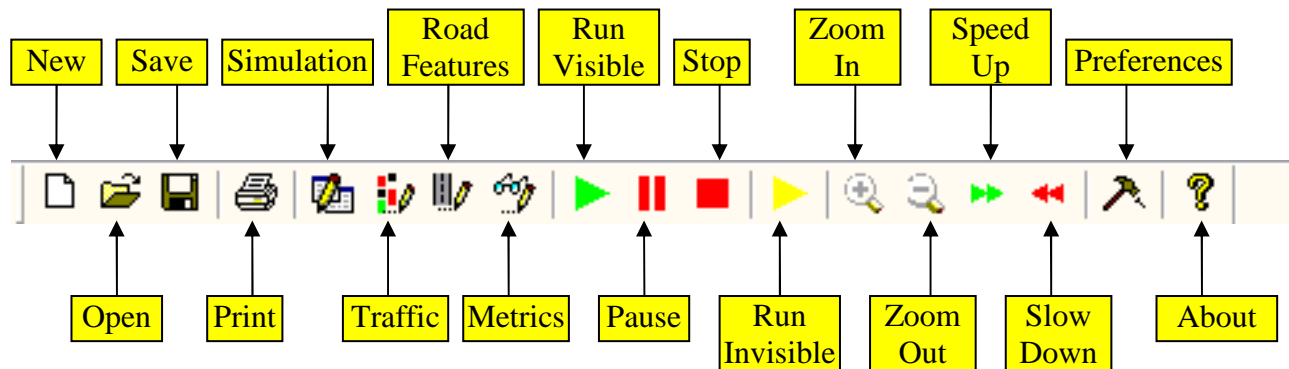
The buttons indicated provide a shortcut to the menu commands as follows:

- **New**: File > New;

- **Open**: File > Open;

- **Save**: File > Save;

- **Print**: File > Print;

- **Simulation**: Configuration > Simulation;

- **Traffic**: Configuration > Traffic Model;

- **Road Features**: Configuration > Road Features;

- **Metrics**: Configuration > Traffic Flow Metrics;

- **Run Visible**: Run > As Visible;

- **Pause**: Tools > Pause (Resume);

- **Stop**: Tools > Stop;

- **Run Invisible**: Run > As Invisible;

- **Zoom In**: Tools > Zoom In;

- **Zoom Out**: Tools > Zoom Out;

- **Speed Up**: Tools > Speed Up;

- **Slow Down**: Tools > Slow Down;

- **Preferences**: Tools >Preferences;

- **About**: Help > About.

4.4   Shortcut Keys

Extensive support for shortcut keys is included in `EvolveTraffic`. Some commands are only accessible through the keyboard.

Commands only accessibly through the keyboard:

- **Left Arrow**: Moves the portion of the road being viewed, one-tenth the length of the road to the left;
- **Right Arrow**: Moves the viewport one-tenth of the road length to the right;
- **Home**: Moves the viewport to the start of the road;
- **End**: Moves the viewport to the end of the road.

In addition to the standard Windows keyboard selection of menu commands, `EvolveTraffic` provides the following keyboard shortcuts to menu commands:

- **Ctrl+N**: File > New;
- **Ctrl+O**: File > Open;
- **Ctrl+S**: File > Save;
- **Ctrl+I**: File > Save Image;
- **Ctrl+P**: File > Print;
- **Alt+S**: Configuration > Simulation;
- **Alt+T**: Configuration > Traffic Model;
- **Alt+R**: Configuration > Road Features;
- **Alt+M**: Configuration > Traffic Flow Metrics;
- **F5**: Run > Visible;
- **Shift+F5**: Run > As Invisible;
- **F10**: Tools > Zoom In;
- **F11**: Tools > Zoom Out;
- **F6**: Tools > Speed Up;

- **F7**: Tools > Slow Down;

- **F9**: Tools > Pause (Resume);

- **F12**: Tools > Stop;

- **Alt**+**P**: Tools >Preferences.

4.5   Other Interface Elements
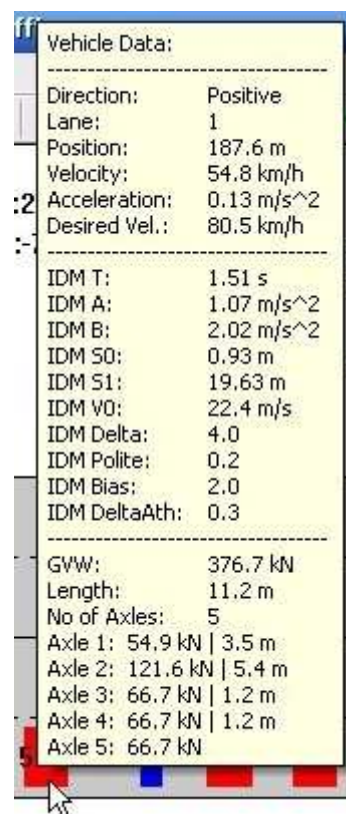
DataTips

All of the relevant information about a vehicle can be obtain through the DataTips.
To use these:

1. A simulation must be running in Visible Mode;
2. The simulation must be paused;
3. Clicking on the vehicle of interest with the left mouse button displays the DataTip for that vehicle.

The DataTip contains all of the relevant data for that vehicle, as shown:

```
Vehicle Data:
------------------------------------
Direction:       Positive
Lane:            1
Position:        187.6 m
Velocity:        54.8 km/h
Acceleration:    0.13 m/s^2
Desired Vel.:    80.5 km/h
------------------------------------
IDM T:           1.51 s
IDM A:           1.07 m/s^2
IDM B:           2.02 m/s^2
IDM S0:          0.93 m
IDM S1:          19.63 m
IDM V0:          22.4 m/s
IDM Delta:       4.0
IDM Polite:      0.2
IDM Bias:        2.0
IDM DeltaAth:    0.3
------------------------------------
GVW:             376.7 kN
Length:          11.2 m
No of Axles:     5
Axle 1: 54.9 kN | 3.5 m
Axle 2: 121.6 kN | 5.4 m
Axle 3: 66.7 kN | 1.2 m
Axle 4: 66.7 kN | 1.2 m
Axle 5: 66.7 kN
```

Progress Bar

The progress bar gives an indication of the amount of simulation complete and remaining. At the start of the simulation, EvolveTraffic counts the number of vehicles in the input traffic file. At each time step the number of vehicles already read from the file is compared to the total number of vehicles to give an estimate of the simulation duration, as shown:
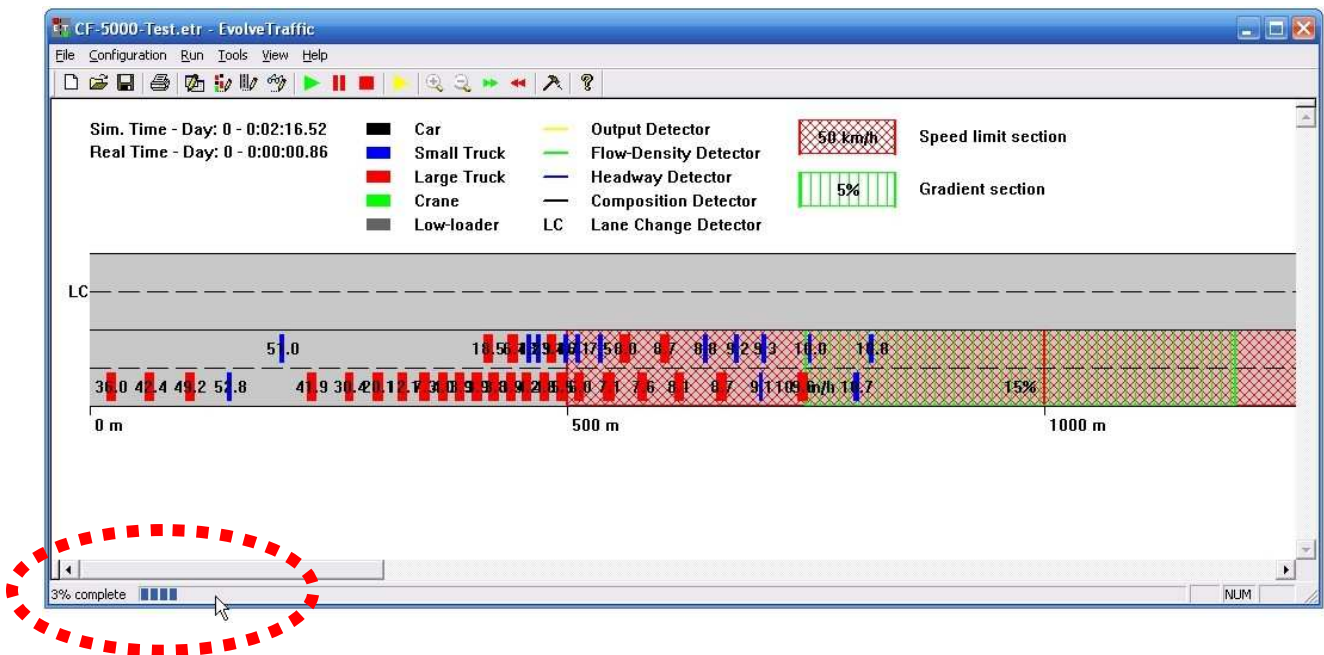
Note that once the last vehicle has entered the road the progress bar will display 100% (since all vehicles have been read), but the simulation is not complete until the last vehicle has left the road.

Clocks

For simulations run in both the Visible and Invisible Modes, two clocks are maintained:

- A simulation clock: which represents time as observed by the vehicles;

- A real clock: representing the elapsed actual time since the start of the simulation.
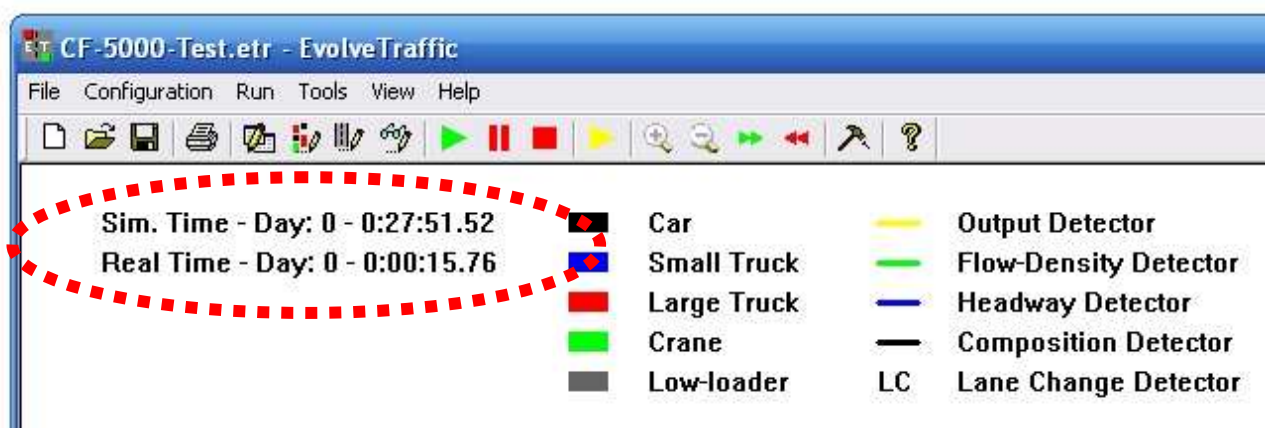
In Visible Mode, when paused, the real clock is also paused. This means that at the end of the simulation an estimate of the average time warp (ratio of simulation to real time) is easily got.

In Invisible Mode, only the final simulation and real times are shown upon completion of the simulation. This enables the time warp to be calculated.

Calculation of the time warp can prove useful to examine the effect of including various detector output, or even just the detector's time intervals (Section 5.5).

5. `EvolveTraffic` Input

5.1 *Introduction*

The output from `EvolveTraffic` is only of use if the input data is representative of the traffic stream of interest. Use of the program is therefore an iterative process: the Developer User must run the program to get results that may be compared to measured traffic statistics and macroscopic properties. Depending on the comparison the Developer User must then change some of the traffic configuration properties as appropriate and re-run to check on the revised comparison. This process may take many iterations.

To minimise effort, all the input data for the program can be saved into `EvolveTraffic` files – file of type `*.etr`.

Once a traffic stream has been calibrated, that traffic may be run over many different road configurations (lengths, speed limits, and gradients). Again a different `*.etr` file may come in useful at this point. Note however, that since the IDM properties may be probabilistic (Section 5.3), running the same traffic over the same road configuration can result in (slightly) different traffic output files.

All of the above aspects mean that the a calibrated site may be used as the basis for many hundreds of different traffic configurations.

5.2   Simulation Configuration

This dialog configures the simulation parameters. The various sections of the dialog (as shown) are explained as follows:
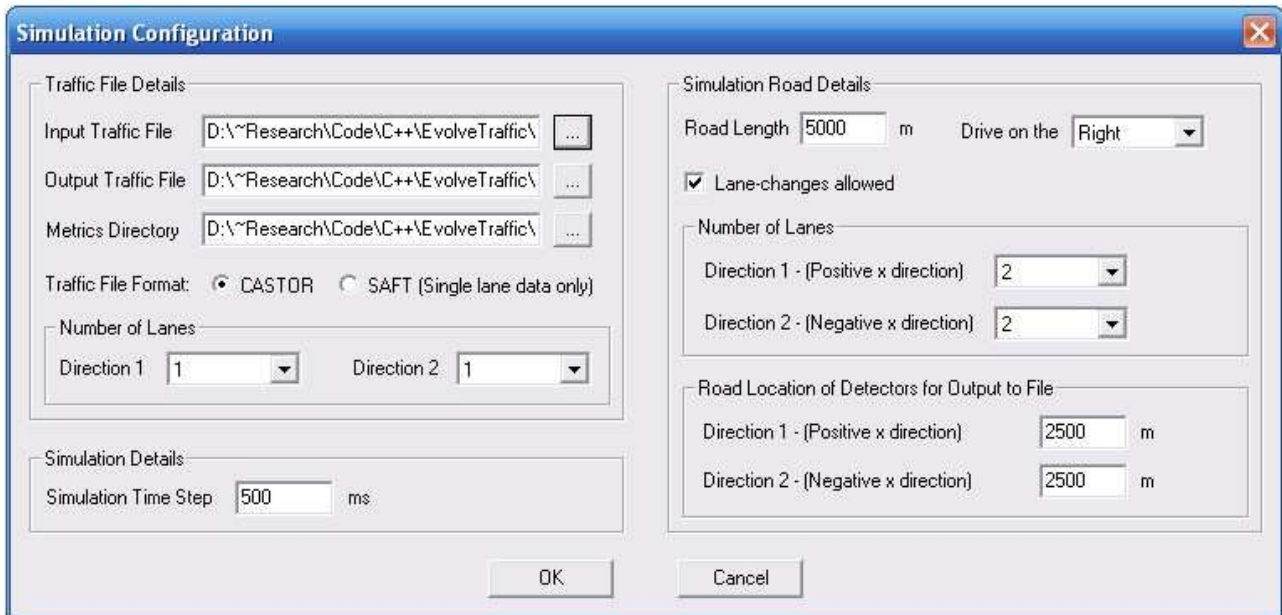
Traffic File Details

- **Input Traffic File**: The file from which the traffic to be placed on the road is read;

- **Output Traffic File**: The file to which the traffic is written as it passes the output detector;

- **Metrics Directory**: The directory to which the metrics detectors output files (if any) will be written;

- **Traffic File Format**: the format of the file to be read in and written out. Note that only CASTOR supports multiple-lane traffic. The SAFT format is for single lane traffic.

Important! `EvolveTraffic` does not check the file format prior to reading the file. Therefore setting the wrong file format here will result in a program crash.

- **Number of Lanes**: Specify the number of lanes of traffic in each direction, as contained in the traffic file.

Important! `EvolveTraffic` does not check the number of lanes in the traffic file prior to reading the file. Therefore setting the wrong numbers here may result in a program crash. In particular:

  - Setting more lanes than there are on the traffic file does not affect the program execution;
  - Setting fewer lanes than there are in the traffic file results in a program crash.

Simulation Details

- **Simulation Step**: The time step, in milliseconds, that the simulation progresses against. Vehicle driving properties are only updated once per time step. Too small a time step is inefficient; too large may result in inaccuracies. For the IDM, Treiber uses 250 ms typically, stating that smaller time steps do not exhibit significantly different results [3], [4].

Simulation Road Details

- **Road Length**: The length of the road in metres that the vehicles are to be driven along. All lanes are this length.
- **Drive on the**: Choose the drive-side for the vehicles, European or British-Irish. This setting only affects the graphical display of the road and vehicles.

- **Lane Changes Allowed**: Choose whether to allow lane changes or not.

- **Number of Lanes**: Specify the number of lanes in each direction on the simulation road, up to a maximum of 4. If there is only one direction choose 0 lanes for the Direction 2 (choosing zero lanes for Direction 1 means the program will swap Direction 1 and Direction 2 – single direction roads always run in the positive x direction). Note that the user cannot set fewer lanes in the simulation road than there are in the input traffic file since the vehicles cannot arrive to non-existent lanes.

- **Road Location of Detectors for Output To File**: Specify the position of the Output Detectors which, upon a vehicle's arrival, write the vehicle to file along with its new time-of-arrival.

5.3   Traffic Model Configuration

This dialog configures the IDM parameters that govern the driving behaviour for each of the 5 classes of vehicle that the program considers.

IDM Parameters for Vehicle Class Class 0 - Cars ▼ << Copy from << ▼				
	Distribution	Location	Scale	Shape
Safe time headway, T (s)	Normal ▼	1.20	0.05	0.00
Maximum acceleration, a (m/s^2)	Normal ▼	1.00	0.05	0.00
Comfortable decceleration, b (m/s^2)	Normal ▼	2.00	0.10	0.00
Minimum jam distance, s0 (m)	Normal ▼	1.00	0.05	0.00
Elastic jam distance, s1 (m)	Normal ▼	10.00	0.70	0.00
Desired velocity, v0 (km/h)	Normal ▼	110.00	7.00	0.00
Acceleration exponent, delta	Constant ▼	4.00	2.00	0.00
Lane change politeness factor, p	Constant ▼	0.20	2.00	0.00
Outside lane bias factor, deltaAbias	Constant ▼	1.00	2.00	0.00
Lane change threshold, deltaAth (m/s^2)	Constant ▼	0.30	2.00	0.00

OK   Cancel

For an explanation of the parameters, refer to Treiber's work [1], [3].

Classes of Vehicle

The program considers 5 classes of vehicle:

1. **Class 0 – Cars**: any vehicle under 3.5 tonnes;
2. **Class 1 – Small Trucks**: any vehicle with less than 4 axles (i.e. 2- and 3-axle vehicles) over 3.5 tonnes;
3. **Class 2 – Large Trucks**: any vehicle that is not a car, small truck, crane or low-loader;
4. **Class 3 – Crane**: any vehicle that is not a car, and has a maximum axle spacing of 4.5 m and an average axle spacing less than 2.5 m;

5. **Class 4** – Low-loader: any vehicle that is not a car and has a maximum axle spacing not less than 7.5 m.

`EvolveTraffic` considers these particular classes of vehicles due to their importance in highway bridge traffic loading.

Parameter Values

For a particular class of vehicle, there are 10 parameters to be set for the IDM and MOBIL (lane-changing) models [1], [3]. In all of Treiber's work to-date these parameters have had the same value of each vehicle class he considers. `EvolveTraffic` extends this by allowing vehicles within a class to have varying parameters, that is, the program allows probabilistic parameters. However, the program retains the ability to specify deterministic (or constant) parameter values.

Each parameter can be distributed according to a number of distributions. Since distributions require different numbers of parameters, the most general parameter set of location, scale and shape is available. However, setting a parameter (consider shape, for example) for a distribution that does not require it (consider the Normal distribution, for example), has no effect. It is up to the user to decide upon the distribution that best suits the parameter being modelled. The supported distributions, with parameters and some comments are:

1. **Exponential**: Location and Scale parameters – may be useful for some parameters such as Safe Time Headway. Not suitable for most parameters, for example, Desired Velocity;

2. **Log-Normal**: Location and Scale parameters;

3. **Gamma**: Location and Scale parameters;

4. **Gumbel**: Location and Scale parameters;

5. **Poisson**: Location and Scale parameters – provides a continuous approximation to the discrete distribution only valid when the location parameter is greater than 10;

6. **GEV**: Location, Scale and Shape parameters – although strictly an extreme value distribution, its flexibility (due to its three parameters) allows many phenomena to be modelled accurately;

7. **Normal**: Location and Scale – the standard Gaussian distribution of parameters mean and standard deviation;

8. **Constant**: Location – this is the deterministic option as no random numbers are generated and the parameter takes the value Location for all vehicles in the class.

Important! `EvolveTraffic` does not check the validity of the distribution values entered. It is up to the user to verify that the values and distribution chosen are realistic and appropriate both to the distribution and the parameter. Choosing inappropriate values or distributions will have unexpected consequences which may include the program crashing.

For calibration purposes it is recommended to use Constant parameters initially and then use appropriate distributions with increasing values of Scale. This allows the impact of the probabilistic parameter values to be assessed. In any case, since most parameters cannot be directly measured, the Scale or Shape parameters must be based upon judgement and the effect upon the macroscopic traffic statistics.

Defining a Parameter Set

Choose the Vehicle Class from the drop-down box labelled IDM Parameters for Vehicle Class. There are two methods of entering data:

1. Using the mouse double click the left button on the cell to edit and type in the value.

2. Use the Tab and the arrow buttons to move between cells. When the appropriate cell is selected, typing a value places that value in the cell.

Copying Parameter Sets Between Vehicle Classes

Since it is likely that most parameter values will be the same between classes, the dialog allows the user to copy parameters sets between vehicle classes. For example, to copy the parameter set from vehicle Class 2 – Large Truck to Class 3 – Crane:

1. Be currently defining parameters for vehicle Class 3 – Crane vehicle by choosing the Vehicle Class from the drop-down box labelled IDM Parameters for Vehicle Class;

2. In the second (right) drop-down box, select Class 2 – Large Truck;

3. Press the "<< Copy From <<" button, where the arrows indicate the direction in which the copying is taking place.

This is as shown below:

IDM Traffic Model Configuration

IDM Parameters for Vehicle Class   Class 3 - Crane   << Copy from <<   Class 2 - Large Truck

	Distribution		Location	Scale	Shape
Safe time headway, T (s)	Normal	▼	1.50	0.05	0.00
Maximum acceleration, a (m/s^2)	Normal	▼	1.00	0.05	0.00
Comfortable decceleration, b (m/s^2)	Normal	▼	2.00	0.10	0.00
Minimum jam distance, s0 (m)	Normal	▼	1.00	0.05	0.00
Elastic jam distance, s1 (m)	Normal	▼	20.00	0.70	0.00
Desired velocity, v0 (km/h)	Normal	▼	78.00	2.00	0.00
Acceleration exponent, delta	Constant	▼	4.00	2.00	0.00
Lane change politeness factor, p	Constant	▼	0.20	2.00	0.00
Outside lane bias factor, deltaAbias	Constant	▼	2.00	2.00	0.00
Lane change threshold, deltaAth (m/s^2)	Constant	▼	0.30	2.00	0.00

OK   Cancel

5.4   Road Features Configuration

This dialog allows the user to define the features that a road may have – the so-called Road Features. Currently, speed limit and gradient sections of road are available.
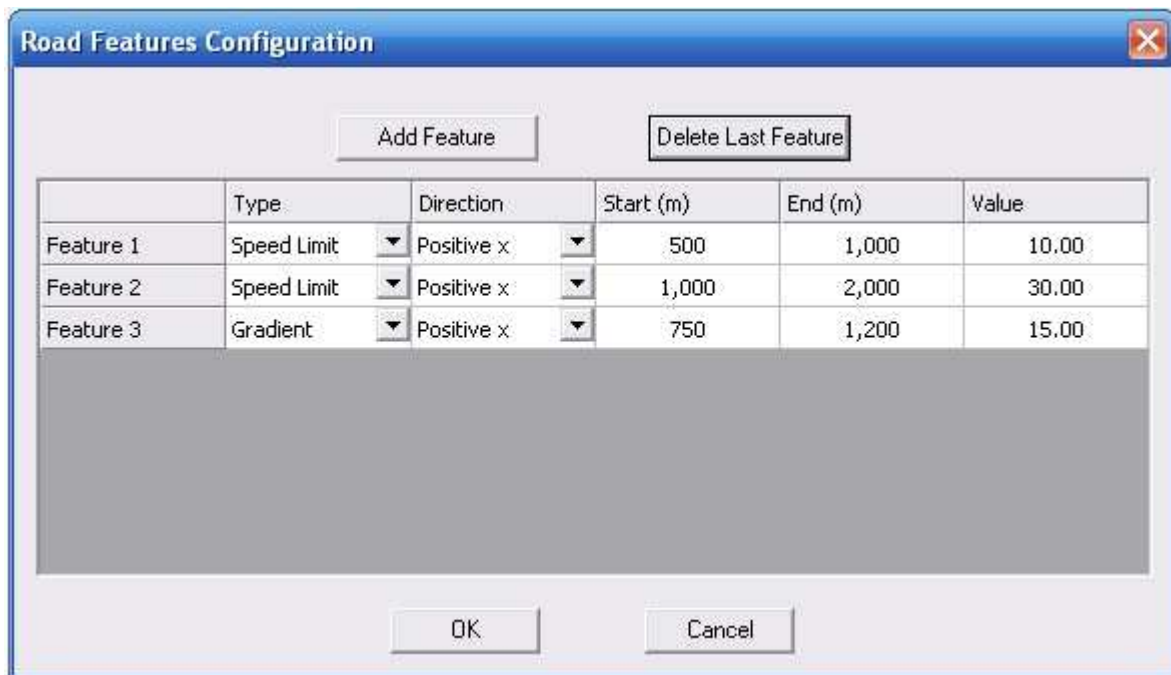
Adding and Deleting Road Features

To add a Road Feature, press the *Add Feature* button and then edit the details of the newly added feature. Currently only the Road Feature that is last in the list can be deleted. Pressing the *Delete Last Feature* button deletes this last feature.

The Speed Limit Road Feature

To include a speed limit section of road, do the following:

1. Add a Road Feature as above;

2. Choose *Speed Limit* from the *Type* drop-down menu.

3. Choose the direction that the speed limit is to be applied to from the *Direction* drop-down box.

4. Input the locations of the start and end of the speed limit section, as measured from the start of the road, in the positive *x* direction.

5. Type the *Value* of the speed limit; that is, the actual speed limit in km/h. Though up to two decimal places is allowed, this is only enabled for defining gradients – speed limits should be integer valued. Non-integer speed limits will be rounded by the program.

Notes:

- `EvolveTraffic` will exchange the start and end values if it needs to;
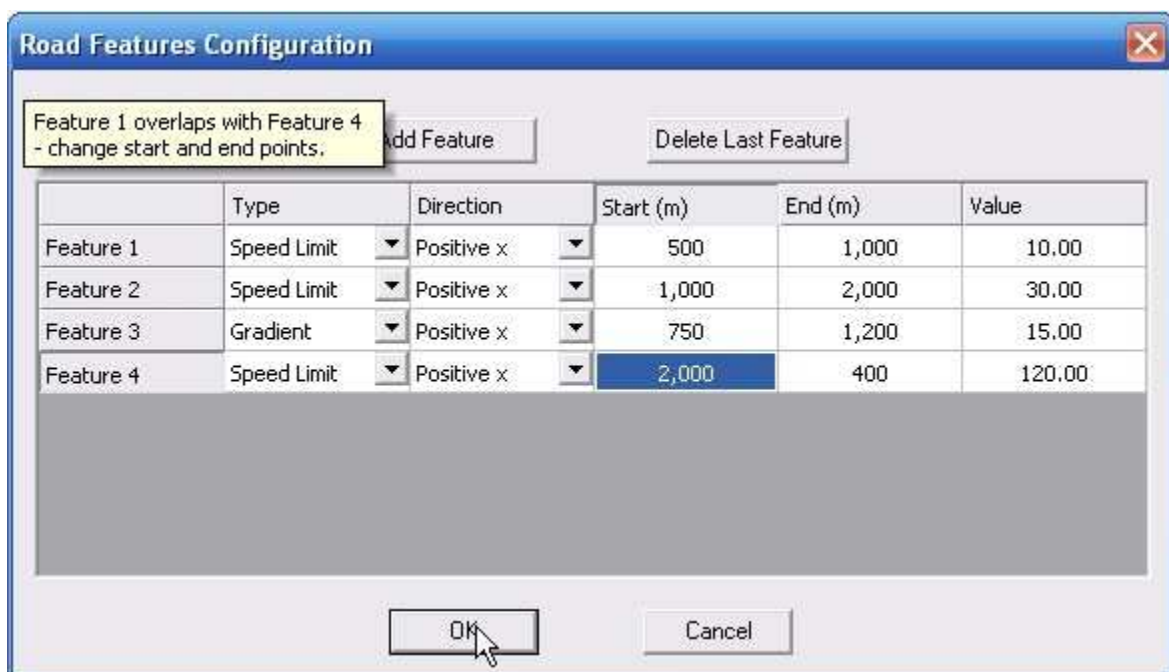
- There cannot be two speed limits for the same section of road, and so EvolveTraffic checks for, and prevents, the user defining overlapping speed limit sections, as shown below where the Feature 4 speed limit overlaps with the Feature 1 and Feature 2 speed limits, as shown above;

- To prevent unrealistic speed limits, the program checks that the speed limits defined are between 10 and 150 km/h;

Effect of a Speed Limit on a Vehicle

When a vehicle enters a speed limit region the program carries out the following steps:

1. Checks to see if the speed limit is less than the vehicle's desired velocity, if it is it proceeds to the next step;
2. Stores the desired velocity of the vehicle before it entered the speed limit region;
3. Sets the vehicle a new desired velocity by keeping the ratio of desired velocity to speed limit the same as the initial desired velocity to average desired velocity (i.e. the Location value of the IDM parameter V0 distribution).

This last step means that vehicles going faster than the average for their class remain faster when in the speed limit region and similarly for slower vehicles.

When a vehicle leaves a speed limit region of road, its initial desired velocity is restored. Note that a vehicle can enter multiple speed limit regions and still have its initial desired velocity restored.

The Gradient Road Feature

To specify a section of road with gradient, follow the steps as per the Speed Limit Road Feature. The *Value* of a Gradient Road Feature is the slope of the road, measured as a percentage:
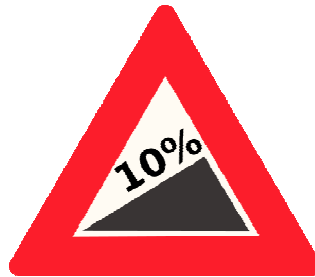
`EvolveTraffic` checks that gradient sections of road do not overlap and that the maximum slope of 15% is not exceeded.

Effect of a Gradient on a Vehicle

The `EvolveTraffic` configuration file has several parameters that specify how a gradient affects a vehicle. In Treiber's work, he models the effect of a gradient by reducing the desired velocity [1], [5] only. `EvolveTraffic` allows the Developer User to change more IDM parameters than this, though care must be taken.

Please refer to the comments in the configuration file for further information on how a gradient can affect vehicles.

5.5   Traffic Flow Metrics Configuration

This dialog allows the user to define locations on the road where information regarding the traffic flow is abstracted:

	Type		Direction		Vehicle Types		Time Interval (s)	Location / Step (m)
Detector 1	Flow & Density	▼	Positive x	▼	All Vehicles	▼	60	2,000
Detector 2	Flow & Density	▼	Positive x	▼	All Vehicles	▼	60	4,000
Detector 3	Flow & Density	▼	Negative x	▼	All Vehicles	▼	60	2,000
Detector 4	Flow & Density	▼	Negative x	▼	All Vehicles	▼	60	4,000
Detector 5	Headway	▼	Positive x	▼	All Vehicles	▼	60	2,000
Detector 6	Headway	▼	Positive x	▼	All Vehicles	▼	60	4,000
Detector 7	Headway	▼	Negative x	▼	All Vehicles	▼	60	2,000
Detector 8	Headway	▼	Negative x	▼	All Vehicles	▼	60	4,000
Detector 9	Composition	▼	Positive x	▼	All Vehicles	▼	60	2,500
Detector 10	Composition	▼	Negative x	▼	All Vehicles	▼	60	2,500
Detector 11	Lane Changes	▼	Negative x	▼	All Vehicles	▼	60	200

Macroscopic Traffic Flow Metrics Detectors — dialog with Add Detector, Delete Last Detector, OK and Cancel buttons.

To add or delete a detector, follow the same steps as for adding or deleting a Road Feature, explained in Section 5.4. For all types of detector the following parameters are the same:

- Type – the type of detector;
- Direction – the road direction the detector covers;
- Vehicle Types – the types of vehicles that the detector examines.

The types of detectors available, and their properties, are:

Flow & Density

This detector extracts the flow and density of the traffic as follows:

- The flow is determined by counting the number of vehicles (of the selected type) passing during the time interval. This figure is then changed to the standard vehicles per hour measure.

- The density is determined from the flow per hour (previously found) divided by the average velocity of the vehicles during the time interval. That is, `EvolveTraffic` uses the relation:

$$\text{Flow} = \text{Average Velocity} \times \text{Density}$$

The time interval directly affects the calculation of the above properties, and so is of importance in the results. Thus, the last two parameters are:

- Time Interval – the interval (in simulation time) between the detector outputs.
- Location/Step –this value specifies the location of the detector.

Refer to Section 7.3 for information on the data output by this detector.

Headway

This detector assembles the headway data for the chosen vehicle type(s).

`EvolveTraffic` considers headway to be the time between two successive vehicles, in the same lane, arriving at the detector.

Since `EvolveTraffic` is run in discrete simulation steps, the time of arrival at the detector is based on an interpolation which is based on constant velocity over the simulation time-step.

The program records headways calculated in this manner for each lane of the specified direction, and for the specified vehicle types. The last two parameters are:

- Time Interval – the interval (in simulation time) between the detector outputs of the recorded data. This value can be reasonably large to avoid the slow process of accessing the hard drive.
- Location/Step –this value specifies the location of the detector.

Refer to Section 7.4 for information on the data output by this detector.

Composition

This detector assembles the composition of the traffic stream at the detector location. Since this statistic only makes sense if every type of vehicle is accounted for, it only performs output if the user selects *All Vehicles* in the *Vehicle Type* drop-down box.

The program outputs the number of each type of vehicle to pass the detector during the previous time interval. Thus, the last two parameters are:

- Time Interval – the interval (in simulation time) between the detector outputs.
- Location/Step – the location of the detector.

Refer to Section 7.5 for information on the data output by this detector.

Lane Changes

This detector assembles Lane Change statistics. A Lane Change Detector takes account of lane changes that occur anywhere in the specified road direction and so it is only necessary to specify one Lane Change Detector per direction. Specifying more Lane Change Detectors will only slow the program's execution. The last two parameters are:

- Time Interval – the interval (in simulation time) between the detector outputs.
- Location/Step – for a Lane Change Detector the 'Step' is the spatial discretization required for outputting the lane change rate, as defined by Trieber [3]. This value should not be more than 1000 m.

Refer to Section 7.6 for information on the data output by this detector.

6. Running EvolveTraffic

6.1 Introduction

The usual procedure followed when using the program is:

1. Define the input data either by using the dialogs (Section 5) or by opening an `*.etr` file (Section 5.1);

2. Run the program in Visible Mode (Section 6.2) to view the behaviour of the traffic (Section 4) and to obtain intermediate information (Section 4.5);

3. Examine the various detector output files using a `*.csv` file viewer, such as MS Excel.

The user preferences and configuration preferences required for the program are explained next.

6.2   Simulation Modes

Introduction

Two modes of running the simulations are provided to meet the aims of the program.

Visible Mode

In this mode, the user can interact with the program at any stage of the simulation and change visualisation parameters via the User Preferences dialog box (Section 6.3). This mode displays the vehicles as they progress along the road. The animation can be paused or stopped at any time. Whilst paused, information on vehicles can be got though the DataTips (Section 4.5). Also, at any point during the simulation, a screenshot of the simulation may be taken using the File > Save Image command (or the Ctrl+I keyboard shortcut). These features make the Visible Mode an interactive environment for the calibration of a traffic model.

The features described for Visible Mode above come at the expense of execution time. Both rendering the simulation to the screen at 20 frames per second, and interrupting the simulation every time the user interacts with the program, mean that results cannot be obtained as fast as possible. Further, this mode requires that the program 'sleep' for a period during each simulation step in order that the animation of vehicles is done at a slow enough speed for human eyes to perceive. Invisible Mode rectifies these issues as it is optimised for speed of execution.

Invisible Mode

When a simulation is run in invisible mode the program only does two things:

1. Runs the simulation as fast as possible;
2. Updates the progress bar.

The progress bar is therefore the only indication the user has that the program is running. This mode can run very fast with simulation time to real ratios of over 5000 easily achieved.

When running in Invisible Mode, the program will require the full use of the processor it is being run on. Users with multi-core processors will be able to run other programs as `EvolveTraffic` runs in the background. However users with single-core processers will not be able to use other programs and the computer will appear frozen until the simulation is complete. For such users, to optimise simulation times, it is advised that no other programs be running at the same time.

6.3   Program Preferences

Ordinary User Preferences

User preferences can be specified though the User Preferences dialog, shown here:
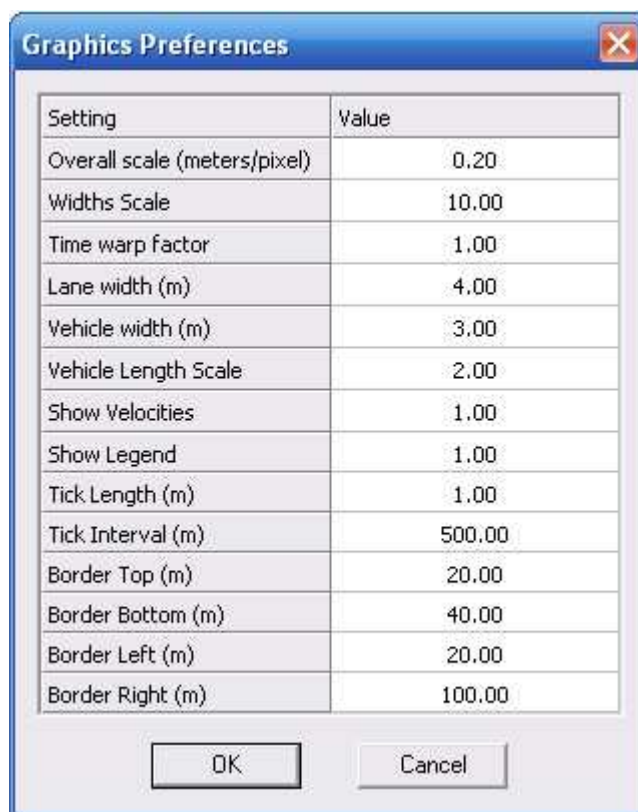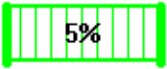
The parameters are:

- **Overall Scale**: the current scale of the drawing;
- **Widths Scale**: the exaggeration applied to the road and vehicle widths so that it is easier to visualize the vehicles;
- **Time Warp Factor**: the ratio between simulation time and real time;
- **Lane Width**: the width of a lane as drawn prior to application of the Widths Scale factor;
- **Vehicle Width**: the width of a vehicle as drawn prior to application of the widths scale factor;

- **Vehicle Length Scale**: a factor that allows the length of a vehicle to be exaggerated in order that a more reasonable aspect ration is kept once the Widths Scale factor has been applied. Note that when this is not unity it can appear that vehicles overlap on the road when very close (bumper-to-bumper traffic), set this to unity to observe that this is not the case. Note also that the vehicle length is exaggerated from the front of the vehicle, that is, the extra length is applied to the back of the vehicle always.

- **Show Velocities**: set this value to zero to turn off (any other value turns it on) the display of the velocities of each vehicle (shown as a text output overlaid):
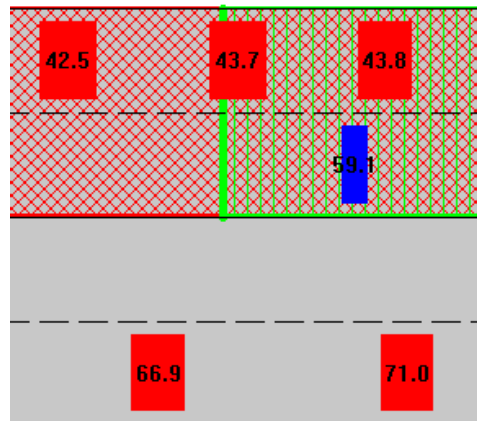
- **Show Legend**: set this value to zero to turn off (any other value turns it on) the display of the legend at the top of the screen:

- **Tick Length**: the length of the ruler ticks;

- **Tick Interval**: the interval between ruler ticks and distance displays;

- **Border Top**: the border between the top of the screen and the legend and clocks;

- **Border Bottom**: currently obsolete;

- **Border Left**: the border between the start of the road and the edge of the window;

- **Border Right**: the border between the end of the road and the edge of the window.

Developer User Preferences

These are as specified in the `EvolveTraffic.ini` configuration file found in the program folder. Refer to Appendix 9.1 for the default version of this file, and for the extensive explanatory comments.

Note that the user can add their own comments (similar to C++ style single line comments) to this file by preceding the line with two forward slashes: //.

Some of the configuration parameters may be brought into the main interface in future versions for ease of use.

6.4   Examples

Included with the installation are three example applications of `EvolveTraffic` in the Examples folder of the program installation folder. Open the relevant `*.etr` file and run in Visible Mode to see the examples, or in Invisible Mode to get straight to the detector output.

Example 1 – `Ex 1.etr`

This example represents a congested truck-only traffic file, run on a road of 5 km length, with several speed limit and gradient regions:
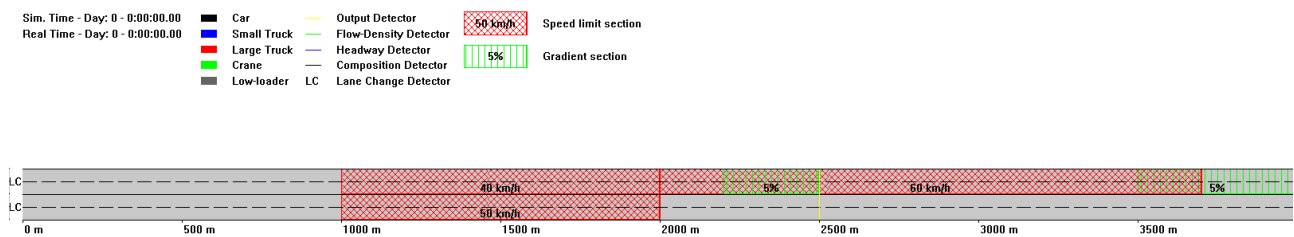
Lane change detectors are specified for both directions.

In this file, congestion can be seen to start occurring in each direction at the first Road Feature the vehicles arrive at. High levels of lane changes occur at this point, as may be observed in the Lane Change Detector Output.

Example 2 – `Ex 2.etr`

This example illustrates a free-flowing truck-only traffic file, run on the same road as Example 1. Various detectors are specified along the length of the positive direction:
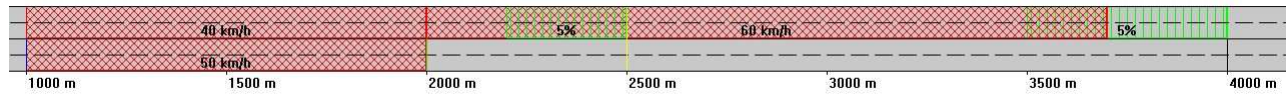
Example 3 – `Ex 3.etr`

This file illustrates a SAFT input file from the RN4 in France. A speed limit section of 50 km/h is specified for 500 m along the 3 km road:

7. `EvolveTraffic` Output

7.1   Introduction

`EvolveTraffic` has the ability to produce large amounts of output, especially for long or heavily congested traffic files. The program accesses the hard drive at every time interval specified for each detector (Section 5.5) and this can significantly slow the program's execution. Therefore it is recommended that statistical detector output is only used when required for calibrating the IDM traffic parameters. For execution of the program for the sole purpose of obtaining another randomized traffic file, based on calibrated IDM parameters, it is better to have no detector output.

For calibration purposes, the statistical detectors' output is comprehensive and should be sufficient to obtain good matches between measured and simulated traffic. This process is not trivial – only macroscopic statistics can be used as the basis for comparison of measured against simulated traffic.

Besides the output traffic file, all detector output files are in Comma-Separated Values (`*.csv`) file format which can be read by any spreadsheet viewer, such as MS Excel. They are output to the directory as specified by the user in the Simulation Configuration dialog (Section 5.2).

7.2   Traffic File

For every simulation run, each vehicle is output to file at the time it hits the location of the Output Detector (Section 5.2) for its direction. The physical properties of the vehicle are maintained through the simulation, and these are written out to file and will be the same as those read in. The arrival time at the Output Detector is output to this file as the vehicle's new time stamp. It is this time stamp that is the reason behind EvolveTraffic, since it represents the driving behaviour as the vehicle moves along the road.

The output traffic file is in the same format as the input traffic file (Section 5.2). For reference, the CASTOR and SAFT file formats are given.

The output traffic file is saved with the name, extension and location as specified by the user in the Simulation Configuration dialog (Section 5.2).

7.3 Flow & Density

File

The output for a Flow Density Detector is made to the following file:

    FD_vvv_ddd_xxxx.csv

Where:

- `FD` – means Flow Density detector file;
- `vvv` – the vehicle type for which the output is made, it can be:
    - All – for All Vehicles output;
    - Car – for Car only output;
    - ST – for Small Truck only output;
    - LT – for Large Truck only output;
    - Crane – for Crane only output;
    - LL – for Low-loader only output;
- `ddd` – the direction identifier:
    - Pos – for the positive x direction;
    - Neg – for the negative x direction;
- `xxxxx` – a number representing the location of the detector in metres.

Output

The format of the Flow Density output is:

- The first column, contains an index representing the time interval for which the output was made;

Next, for each lane in the direction that the detector considers, for the particular time interval considered, we have three columns:

- Density (veh/km) – the measured traffic density of the vehicles considered;
- Flow (veh/hr) – the flow of the vehicles considered;
- Ave. Vel. (km/h) – the average velocity of the vehicles considered.

Finally, for the direction as a whole, we have the three columns of Density, Flow and Ave. Vel. as before.

An example output for a two-lane direction is:

	A	B	C	D	E	F	G	H	I	J
1		Lane 1			Lane 2			Totals		
2	Interval No.	Density (veh/km)	Flow (veh/hr)	Avg. Vel. (km/h)	Density (veh/km)	Flow (veh/hr)	Avg. Vel. (km/h)	Density (veh/km)	Flow (veh/hr)	Avg. Vel. (km/h)
3	1	-1.#IND	0	-1.#IND	-1.#IND	0	-1.#IND	-1.#IND	0	-1.#IND
4	2	4.55368	360	79.057	0.673806	60	89.0464	5.22748	420	168.103
5	3	11.0702	780	70.4591	2.18971	180	82.2028	13.26	960	152.662
6	4	14.3979	960	66.6763	3.10109	240	77.3921	17.499	1200	144.068
7	5	14.6639	960	65.467	3.65787	300	82.0149	18.3218	1260	147.482
8	6	14.8398	960	64.691	3.76076	300	79.771	18.6005	1260	144.462
9	7	14.0666	900	63.9813	5.60274	420	74.9633	19.6694	1320	138.945
10	8	15.1851	960	63.2197	3.85794	300	77.7618	19.0431	1260	140.982
11	9	15.1839	960	63.2249	3.92699	300	76.3944	19.1109	1260	139.619

Note that the -1.#IND means that no number is present – i.e. no vehicles passed the detector during the time interval considered. This can be clearly seen from the Flow information, since it is zero.

7.4 Headway

File

The output for a Headway Detector is made to the following file:

`Head_vvv_ddd_xxxx.csv`

In which `vvv`, `ddd`, and `xxxx` have the same meanings as for the Flow Density Detector file (Section 7.3).

Output

	A	B
1	Headways (s)	
2	Lane 1	Lane 2
3	521.373	
4	32.2436	552.235
5	43.4141	
6	13.4481	80.4202
7	9.89299	
8	13.8559	
9	21.329	
10	16.6086	
11	14.9764	
12	17.6644	
13	21.3549	124.5
14	16.2593	
15	15.4452	
16	17.4007	
17	27.9952	40.4083
18	18.0831	
19	18.8952	
20	11.8103	71.362

At each time interval, `EvolveTraffic` outputs the vectors of headways (in seconds), for the vehicle types considered, for each lane for the direction which the detector covers. An example is shown to the left.

Note from the example that the first vehicles have large headways (since there is no vehicle in front). Also notice that Lane 1 is more heavily trafficked than Lane 2.

7.5 Composition

File

The output for a Composition Detector is made to the following file:

```
Comp_vvv_ddd_xxxx.csv
```

In which vvv, ddd, and xxxx have the same meanings as for the Flow Density Detector file (Section 7.3).

Output

The output for each time interval is simply the count of the number of vehicles of the vehicle type that passed the detector during the previous time interval. Each time interval writes to a row and each vehicle type has a column. All best seen in the example output (taken from a traffic file without cars):

	A	B	C	D	E
	No. Cars	No. Small Trucks	No. Large Trucks	No. Cranes	No. Low-Loaders
1					
2	0	0	0	0	0
3	0	0	0	0	0
4	0	5	9	0	0
5	0	3	12	1	0
6	0	6	12	0	0
7	0	5	12	0	0
8	0	5	11	0	0
9	0	2	4	1	0
10	0	1	6	1	0
11	0	1	4	1	0
12	0	0	5	0	0
13	0	1	4	0	0
14	0	2	3	1	0
15	0	0	5	0	0

7.6 Lane Changes

File

The output for a Lane Change Detector is made up of the following files:

1. LC_All_vvv_ddd.csv;

2. LC_Comp_vvv_ddd.csv;

3. LC_Rate_vvv_ddd.csv;

4. LC_ST_vvv_ddd.csv;

In which vvv, and ddd have the same meanings as for the Flow Density Detector file (Section 7.3).

The LC_All File

This file outputs the raw information relating to lane change events. The format of the file is best seen by example:

	A	B	C	D	E	F
1	LANE CHANGE - VERBOSE OUTPUT OF EVENTS					
2	Time	Position	From Lane	To Lane	Right/Left	Type
3						
4	82.52	4031	4	3	Left	SmallTr
5	95.02	4022	4	3	Left	SmallTr
6	115.02	4032	4	3	Left	SmallTr
7	134.52	4058	4	3	Left	SmallTr
8	142.52	4021	4	3	Left	SmallTr
9	170.02	4036	4	3	Left	SmallTr
10	177.52	3993	4	3	Left	SmallTr
11	190.02	4023	4	3	Left	SmallTr
12	200.02	1987	4	3	Left	LargeTr
13	207.52	1992	4	3	Left	LargeTr

The columns are explained as:

- **Time**: the time at which the lane change vene took place;

- **Position**: the location (as measured from the start of the road) of the lane change;

- **From Lane**: the lane number that the vehicle moved from;

- **To Lane**: the lane number that the vehicle moved to;

- **Right/Left**: whether the vehicle moved left or right;

- **Type**: the vehicle type that changed lane.

Note that lanes are numbered from the outside lane of the positive x direction to the outside lane of the negative x direction. So for a European road:
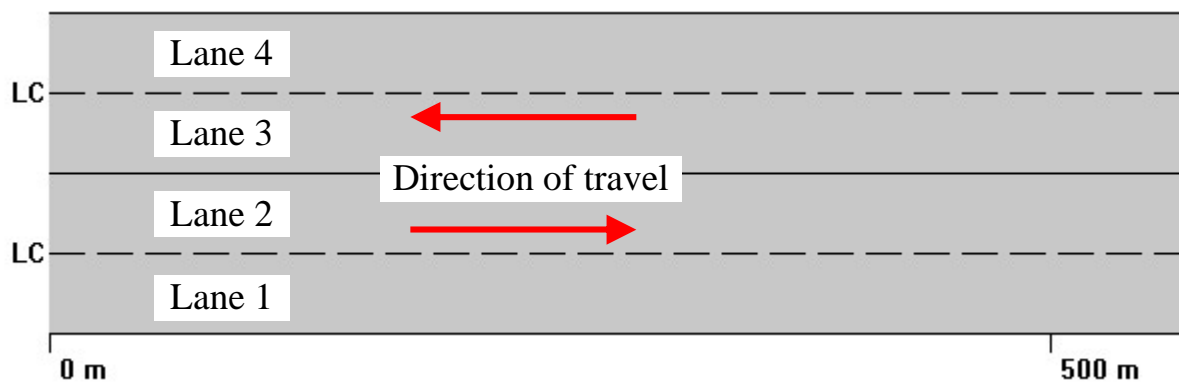
And for a left-hand drive roads:
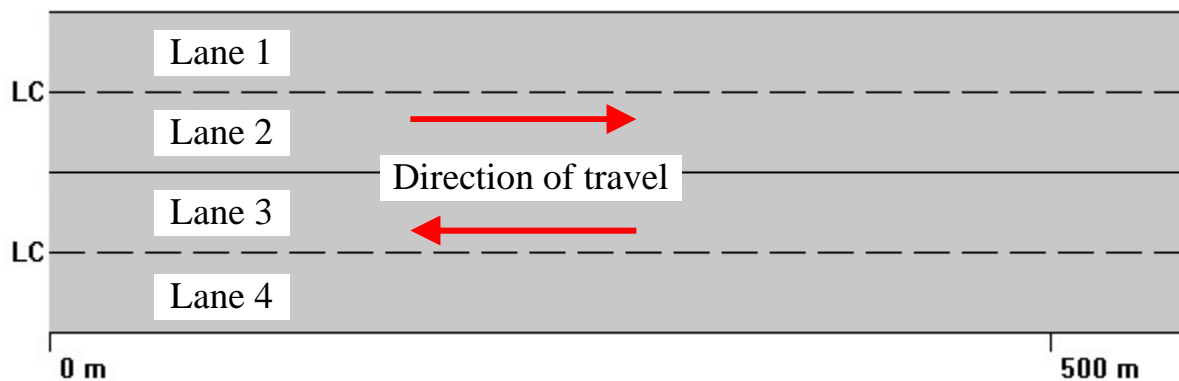
The LC_Comp File

This file gives information pertaining to the composition of lane change events by vehicle type. Since this output only makes sense if the vehicle type for this detector is *All Vehicles*, no output is performed for other vehicle types.

An example of this file's output for a traffic stream with no cars is:

	A	B	C	D	E	F
1	LANE CHANGE - COMPOSITION OF EVENTS					
2	Time Interval	Car Changes	SmallTr Changes	LargeTr Changes	Crane Changes	LowLoad Changes
3	60	0	0	0	0	0
4	120	0	3	0	0	0
5	180	0	4	0	0	0
6	240	0	10	6	0	0
7	300	0	9	2	0	0
8	360	0	8	4	1	0
9	420	0	4	3	0	0
10	480	0	7	0	1	0
11	540	0	10	1	0	0
12	600	0	6	0	0	0
13	660	0	7	4	1	0

The number of lane changes that occurred in the previous time interval of each vehicle type is as indicated.

© UCD/LCPC 2008

The LC_Rate File

This file contains the lane change rate for each time interval for each 'Step' (see Section 5.5) along the road length. This metric is as defined in the paper on the lane change model, MOBIL [3]. The lane change rate is:

$$r(\rho, x, t) = \frac{n_{LC}}{\Delta t \cdot \Delta x}$$

In which:

- $r(\rho, x, t)$ is the lane change rate, and is a function of both time, distance and traffic density;

- n_{LC} is the number of lane changes that occurred in the time interval, Δt;

- Δt and Δx are the temporal and spatial discretizations.

The lane change rate is measure in units of number of lane changes per hour per kilometre (1/hr/km). Taking $\Delta t = 1$ min. and $\Delta x = 200$ m, a single lane change is equivalent to a lane change rate of:

$$\frac{1}{(1/60)(200/1000)} = 300 \text{ /hr/km}$$

Therefore the lane change rate values in the file will tend to be round numbers due ot the above calculations. Further, this metric is calculated for each time interval at each step along the road, leading to a large matrix of lane change rates. An example is:

	A	B	C	D	E	F	G	H	I	Y	Z	AA
1	LANE CHANGE - RATES BY LOCATION AND TIME											
2	Time (s)	Location (m)										
3		0	200	400	600	800	1000	1200	1400	4600	4800	5000
4	60	0	0	0	0	0	0	300	0	0	0	0
5	120	0	0	0	0	0	900	600	0	0	0	0
6	180	0	0	0	0	0	1200	300	0	0	0	0
7	240	0	0	0	0	0	900	600	0	0	0	0
8	300	0	0	0	0	0	1500	0	0	0	0	0
9	360	0	0	0	0	0	900	0	0	0	0	0
10	420	0	0	0	0	0	600	0	0	0	0	0
11	480	0	0	0	0	0	2100	0	0	0	300	0
12	540	0	0	0	0	0	1200	0	0	0	0	0
13	600	0	0	0	0	0	1200	0	0	0	0	0
14	660	0	0	0	0	0	1500	300	0	0	0	0
15	720	0	0	0	0	0	600	0	0	0	0	0
16	780	0	0	0	0	0	1500	0	0	0	0	0
17	840	0	0	0	0	0	1500	0	0	0	0	0
18	900	0	0	0	0	0	600	0	0	0	0	0
19	960	0	0	0	0	0	1800	0	0	0	0	0
20	1020	0	0	0	0	0	1800	0	0	0	0	0
21	1080	0	0	0	0	0	900	0	0	0	0	0
22	1140	0	0	0	0	0	900	300	0	0	0	0
23	1200	0	0	0	0	0	900	600	0	0	0	0
24	1260	0	0	0	0	0	300	0	0	0	0	0
25	1320	0	0	0	0	1500	0	0	0	0	0	0

Note that in this example the columns referring to Steps 1600 m to 4400 m have been hidden.

From this example it is apparent that there is significant lane changing at 1000 m. This is caused by a speed limit section that starts at this point in the road.

By keeping the Lane Change Detector time interval the same as that of the Flow Density Detector, the lane change rates and density at a particular point on the road can be plotted together. As explained in the MOBIL paper [3], this curve should display low rates of lane changing at low flows, high rates at medium flows and low rates at fully congested flows.

The LC_ST File

This file outputs the time and position of each lane change event for both Left to Right changes, and for Right to Left changes, for the direction that the detector is applied to. An example is:

	A	B	C	D
1	LANE CHANGE - SPACE TIME OF EVENTS			
2	Left To Right		Right To Left	
3	Time (s)	Location (m)	Time (s)	Location (m)
4			56.02	1009
5			66.52	1015
6	66.52	1023		
7			68.52	989
8			82.52	961
9			90.52	987
10	119.02	1689		
11			119.52	1679
12			127.02	918
13			136.52	963
14	144.52	1996		
15			147.52	946
16			154.52	1012
17	162.02	2124		

The plot that results from a large amount of such data can be very informative:
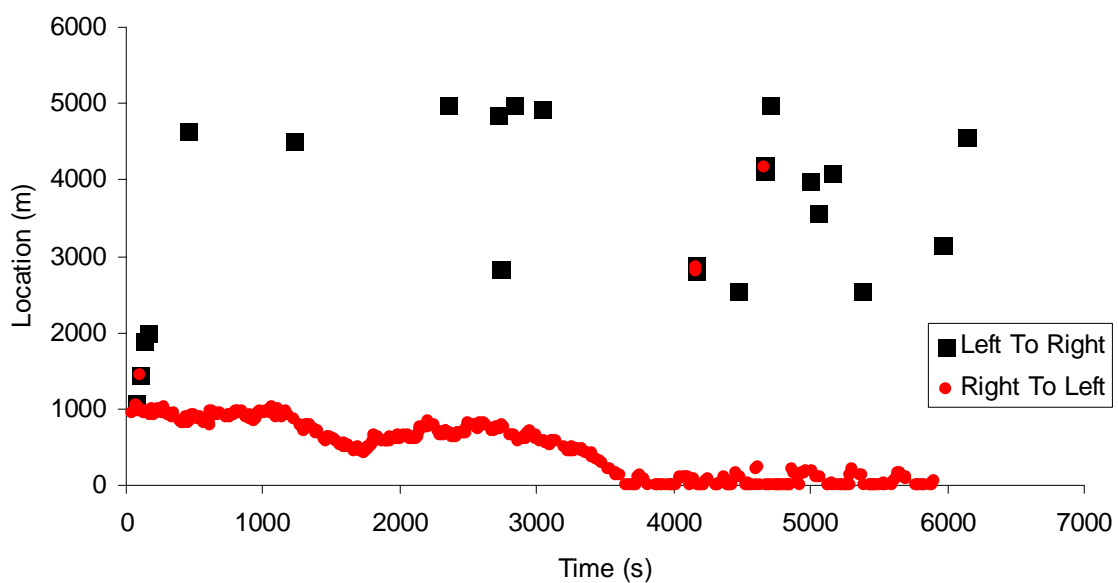
Noting that each change is represented by a point, this plot shows that a large number of lane change events occur at about 1000 m and as time progresses this point moves backwards towards the start of the road (at 0 m). This is explained by the presence of the speed limit region previously mentioned (and as shown below), that causes congestion that progresses backwards, necessitating earlier lane changes. Further along the road (beyond say 2000 m) there are a number of vehicles returning to the outside lane, and very few overtaking events.
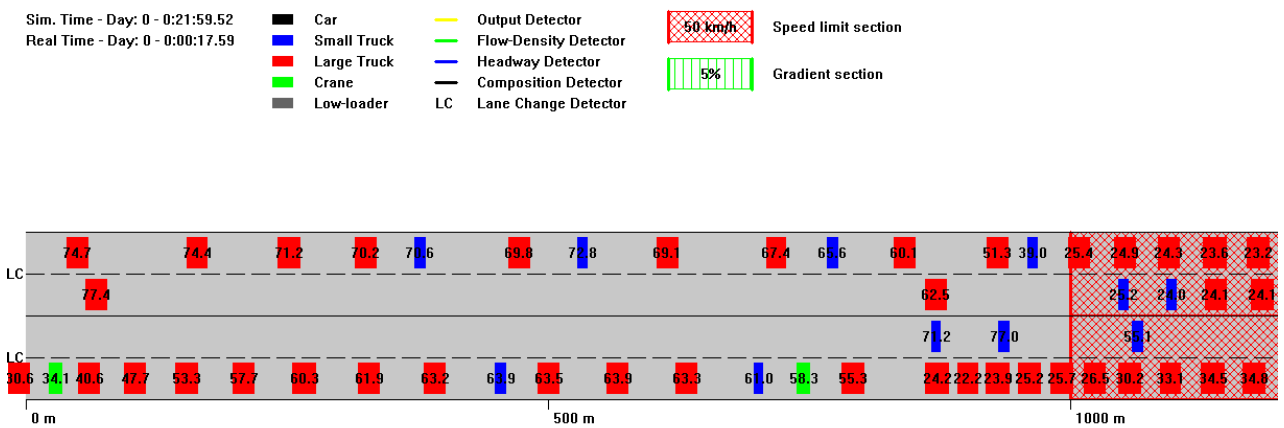
8. FAQs and TroubleShooting

8.1 FAQs

Since this is the first release of this program there are no frequently asked questions!

8.2   TroubleShooting

As this is the first release of this program there bound to be some bugs and inconsistent behavior. Please check below for some possible solutions. If the problem persists, please send a copy of the `*.etr` and input traffic file that causes the problem to colin.caprani@ucd.ie explaining the steps required to reproduce the problem.

It is envisaged that this section of the document will be updated frequently as the program matures.

Problems

The program crashes

- Please check this document for the **Important!** labels and confirm that you have adhered to these requirements.
- Restore the default configuration file values if you have changed any.

The driving is not realistic

- Check the IDM parameters for realism;
- Use Treiber's IDM parameters ([1], [3]) with a small traffic file to verify realism;
- Slowly build the extra complexity into the IDM parameters, verifying the realism at each stage.

9. Appendices

9.1 Appendix 1 – CASTOR and SAFT File Formats

CASTOR File Format

In the table below, the Format column gives the storage type of the data. IX refers to an integer of X number of digits, including leading or trailing zeros.

Record	Unit	Format
Head		I4
Day		I2
Month		I2
Year		I2
Hour		I2
Minute		I2
Second		I2
Second/100		I2
Speed	dm/s	I3
Gross Vehicle Weight - GVW	kg/100	I4
Length	dm	I3
Number of Axles		I1
Direction		I1
Lane		I1
Transverse Location In Lane	dm	I3
Weight Axle 1	kg/100	I3
Spacing Axle 1 - Axle 2	dm	I2
Weight Axle 2	kg/100	I3
Spacing Axle 2 - Axle 3	dm	I2
⋮	⋮	⋮
Spacing Axle 8 - Axle 9	dm	I2
Weight Axle 9	kg/100	I3

SAFT File Format

As for the CASTOR table, the Format column gives the storage type of the data. IX refers to an integer of X number of digits, including leading or trailing zeros. Note that since the SAFT format does not contain direction or lane identifiers, this format is suitable for single lane traffic only.

Record	Unit	Format
Vehicle order		I5
20000 unused number		I5
Day		I2
Month		I2
Year		I2
Hour		I2
Minute		I2
Second		I2
Second/100		I2
Speed	dm/s	I3
Gross Vehicle Weight - GVW	kN	I4
Length	dm	I3
Number of Axles		I1
Weight Axle 1	kN	I3
Spacing Axle 1 - Axle 2	dm	I2
⋮	⋮	⋮
Spacing Axle 8 – Axle 9	dm	I3
Weight Axle 9	kN	I2

9.2   Appendix 2 – Further Documentation

The source code documentation gives further detailed information on the algorithms used in the program, along with explanations of the source code structure. An example output below illustrates the Detector Classes hierarchy. The source code documentation is included in the installation package. To open:

In the program's installation folder, open the \Documents\html\ folder and double click on the index.html file. Alternatively, you can view a PDF of the documentation in the \Documents\ folder: SourceDoc.pdf.

Detector Class Reference
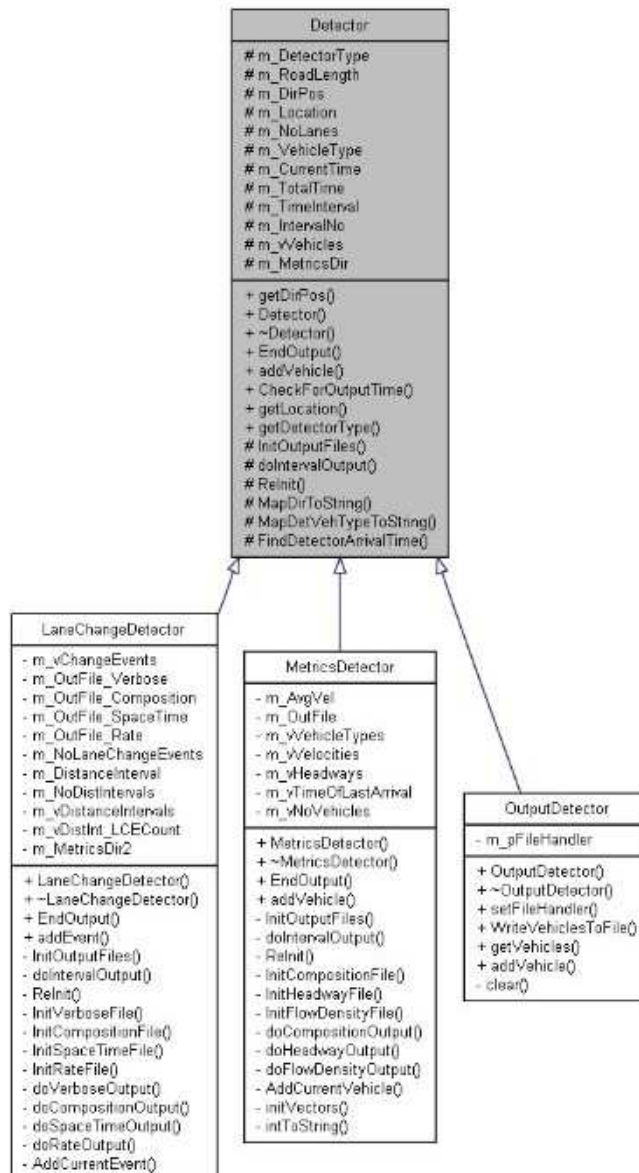
9.3 Appendix 3 - `EvolveTraffic.ini` *Configuration File*

```
//------------------------------------------------------------------------
//------------- EVOLVETRAFFIC PROGRAM CONFIGURATION FILE -------------
//------------------------------------------------------------------------
//
// Notes:
// 1. This file is used by EvolveTraffic to set environment variables
//    that may be altered by the user.
// 2. Before altering this file make a back up copy - even though the
//    default values for each parameter are given in the description.
// 3. The user can insert their own comments in this file, once they
//    are preceded by "//".
// 4. No checking of the validity of this file is done by the program.
//    If, after a change to this file, the program does not run, or
//    exhibits strange behaviour, please use your back up.
// 5. You can email colin.caprani@ucd.ie with any questions, or for
//    a copy of the default file, to save reinstallation.
// 6. This file is in a format particular to EvolveTraffic and is not
//    in standard *.ini format.
// 7. Do NOT alter the sequence of the variables - the program assumes
//    the sequence as given below.
//
//------------------------------------------------------------------------
//
//
//------------------------------------------------------------------------
// IDM - General parameters for the IDM microsimulation model
//------------------------------------------------------------------------
// Please refer to Treiber's references for more information on these
// parameters.
//
// IDM.LANECHANGE_MIN_GAP = 2.0;
//------------------------------------------------------------------------
// The minimum gap to the rear of the vehicle in front, in the target
// lane, or from the front of the rear vehicle in the target lane, to
// the rear of the current vehicle.
// A lane change is not considered if both these gaps are not larger
// than this parameter specifies.
// UNITS: m
// NUMBER TYPE: real
// VALUE:
2.0
//
// IDM.MIN_SPACE_FOR_NEXT_VEHICLE = 27.0;
//------------------------------------------------------------------------
// The minimum clear space in front of the next vehicle to enter the
// road, before it is allowed onto the road.
// UNITS: m
// NUMBER TYPE: real
// VALUE:
27.0
//
// IDM.SAFE_BRAKING = -12.0;
//------------------------------------------------------------------------
// The maximum permissible deceleration any vehicle can undergo.
// This number provides a check on the realism of the random braking
// ability generated according to the user-defined IDM parameters.
// UNITS: m/s^2
```

```
// NUMBER TYPE: real - always negative
// VALUE:
-12.0
//
// IDM.T_DELAY = 1.6;
//----------------------------------------------------------------------
// This specifies the time gap between consideration of lane change
// events for a particular vehicle. Once a lane change event has
// occurred for this vehicle, the timer is reset and a subsequent
// lane change can only occur once the specified time delay has been
// passed.
// UNITS: s
// NUMBER TYPE: real
// VALUE:
1.6
//
//----------------------------------------------------------------------
//
//
//----------------------------------------------------------------------
// ROAD - Parameters affecting the road and its contents
//----------------------------------------------------------------------
//
// Road.ROAD_END_BUFFER = 1000;
//----------------------------------------------------------------------
// This is the distance that the front vehicle in any lane considers
// as being clear road (no vehicles) in front of them, even as it
// approaches the end of the defined road length. This means that
// vehicles approach the end of the road realistically.
// UNITS: m
// NUMBER TYPE: integer
// VALUE:
1000
//
// Road.RoadFeatures.SpeedLimit.SPEEDLIMIT_MIN = 10;
//----------------------------------------------------------------------
// This is the minimum permissible speed limit that the user can
// specify. Included to keep things more realistic.
// This must be greater than zero.
// UNITS: km/h
// NUMBER TYPE: integer
// VALUE:
10
//
// Road.RoadFeatures.SpeedLimit.SPEEDLIMIT_MAX = 200;
//----------------------------------------------------------------------
// This is the maximum permissible speed limit that the user can
// specify. Included to keep things more realistic.
// UNITS: km/h
// NUMBER TYPE: integer
// VALUE:
200
//
// Road.RoadFeatures.Gradient.MAX_SLOPE = 15;
//----------------------------------------------------------------------
// This is the maximum permissible slope of gradient that the user can
// specify. Included to keep things more realistic, and to conform
// better with the assumptions made in modifying the vehicles' driver
// model.
// UNITS: %
// NUMBER TYPE: integer
```

```
// VALUE:
15
//
// Road.RoadFeatures.Gradient.IDM_Modifiers.A_FLAG = true;
//----------------------------------------------------------------------
// This flag specifies if a vehicles acceleration ability is to be
// be modified to account for the gradient it is on.
// Included because Treiber does not do this, but basic physics
// suggests that a vehicles acceleration ability is lessened.
// This ability is not modified for downhill gradients.
// UNITS: N/A
// NUMBER TYPE: boolean (use 1 or 0)
// VALUE:
1
//
// Road.RoadFeatures.Gradient.IDM_Modifiers.A_MIN = 0.1;
//----------------------------------------------------------------------
// The minimum acceleration ability that any vehicle can have after
// having its ability modified to take account of a gradient it may
// be on.
// This number must be greater than zero. The program will crash due to
// a divide by zero, or a square root of a negative number if it is not.
// UNITS: N/A
// NUMBER TYPE: real
// VALUE:
0.1
//
// Road.RoadFeatures.Gradient.IDM_Modifiers.B_FLAG = true;
//----------------------------------------------------------------------
// This flag specifies if a vehicles braking ability is to be
// be modified to account for the gradient it is on.
// Included because Treiber does not do this, but basic physics
// suggests that a vehicle can brake easier going uphill.
// This ability is not modified for downhill gradients.
// UNITS: N/A
// NUMBER TYPE: boolean (use 1 or 0)
// VALUE:
1
//
// Road.RoadFeatures.Gradient.IDM_Modifiers.T_SLOPE = 10.0;
//----------------------------------------------------------------------
// This specifies how the IDM parameter, T, the safe time headway,
// changes according to the gradient the vehicle is on. This value is
// the number of percent of gradient, per second increase in the safe
// time headway.
// It is increased for the gradient, whether uphill or downhill.
// UNITS: %/s
// NUMBER TYPE: real
// VALUE:
10.0
//
// Road.RoadFeatures.Gradient.IDM_Modifiers.V0_MIN = 3.0;
//----------------------------------------------------------------------
// This specifies the minimum desired velocity any vehicle can have
// after its desired velocity has been modified to account for the
// gradient the vehicle is on.
// THis number must be greater than zero, since every vehicle should
// want to make progress.
// UNITS: m/s
// NUMBER TYPE: real
// VALUE:
```

```
3.0
//
// Road.RoadFeatures.Gradient.IDM_Modifiers.V0_SLOPE = 0.2;
//-----------------------------------------------------------------------
// This specifies how the IDM parameter, V0, the desired velocity,
// varies with gradient. It is measured in the % gradient per 1 km/h
// reduction in the desired velocity.
// The desired velocity is reduced for downhill gradients and not
// modified for uphill gradients.
// UNITS:
// NUMBER TYPE: integer
// VALUE:
0.2
//
//-----------------------------------------------------------------------
//
//
//-----------------------------------------------------------------------
// TIME - Parameters defining the time output of the vehicles
//-----------------------------------------------------------------------
// Since we usually consider there to be 5 working days per week for
// 50 weeks per year - which allows for holidays - we have 250 days
// per year. We thus consider there to be 10 working "months" per year
// of 25 working "days" each.
// A vehicle with a time stamp of 3 500 000 seconds thus is written
// out to file as: 10/02/08 12:13.20
//
// Time.DAYS_PER_MT = 25;
//-----------------------------------------------------------------------
// The number of days per simulation month.
// UNITS: N/A
// NUMBER TYPE: integer
// VALUE:
25
//
// Time.MTS_PER_YR = 10;
//-----------------------------------------------------------------------
// The number of simulation months per year.
// UNITS: N/A
// NUMBER TYPE: integer
// VALUE:
10
//
//-----------------------------------------------------------------------
//
//
//-----------------------------------------------------------------------
// VEHICLEID - Parameters defining the identification of vehicles
//-----------------------------------------------------------------------
// The program defines 5 types of vehicles, and these parameters
// govern the vehicle class assigned to a vehicle.
//
// VehicleID.CRANE_AVERAGE_SPACING = 2.5;
//-----------------------------------------------------------------------
// The average spacing of a crane's axles.
// UNITS: m
// NUMBER TYPE: real
// VALUE:
2.5
//
// VehicleID.CRANE_MAX_SPACING = 4.5;
```

```
//------------------------------------------------------------------------
// The maximum single axle spacing possible for a vehicle to still be
// considered a crane.
// UNITS: m
// NUMBER TYPE: real
// VALUE:
4.5
//
// VehicleID.LOWLOADER_MIN_MAX_SPACING = 7.5;
//------------------------------------------------------------------------
// If a vehicle has a maximum axle spacing greater than this value,
// and has more than 2 axles, it is considered a low-loader.
// UNITS: m
// NUMBER TYPE: real
// VALUE:
7.5
//
// VehicleID.SMALL_TRUCK_NO_AXLES = 3;
//------------------------------------------------------------------------
// The maximum number of axles a small truck can be considered to
// have.
// UNITS: N/A
// NUMBER TYPE: integer
// VALUE:
3
//
// VehicleID.TRUCK_WEIGHT_THRESHOLD = 35;
//------------------------------------------------------------------------
// A vehicle with a weight less than this value is a car, otherwise
// it is a truck.
// UNITS: kg/100
// NUMBER TYPE: integer
// VALUE:
35
//
//------------------------------------------------------------------------
//
//
//------------------------------------------------------------------------
// VIEW - Parameters governing the screen display.
//------------------------------------------------------------------------
//
// View.MIN_VIEW_SCALE = 0.1;
//------------------------------------------------------------------------
// The minimum value the user can zoom out to. Setting this value too
// low will result in the program crashing.
// UNITS: N/A
// NUMBER TYPE: real
// VALUE:
0.1
//
// View.DATATIP_DELAY = 20.0;
//------------------------------------------------------------------------
// The time in seconds that the vehicle datatip stays visible after
// the user clicks on a vehicle in pause mode. Must be greater than
// zero or program will crash.
// UNITS: N/A
// NUMBER TYPE: real
// VALUE:
20.0
//
```

```
// View.Colours.Vehicles.VEH_COLOR_CAR = RGB(0,0,0);
//----------------------------------------------------------------------
// The display colour for a car.
// UNITS: RGB( 0 to 255, 0 to 255, 0 to 255 )
// NUMBER TYPE: integer, integer, integer
// VALUE:
0,0,0
//
// View.Colours.Vehicles.VEH_COLOR_SMALLTRUCK = RGB(0,0,255);
//----------------------------------------------------------------------
// The display colour for a small truck.
// UNITS: RGB( 0 to 255, 0 to 255, 0 to 255 )
// NUMBER TYPE: integer, integer, integer
// VALUE:
0,0,255
//
// View.Colours.Vehicles.VEH_COLOR_LARGETRUCK = RGB(255,0,0);
//----------------------------------------------------------------------
// The display colour for a large truck.
// UNITS: RGB( 0 to 255, 0 to 255, 0 to 255 )
// NUMBER TYPE: integer, integer, integer
// VALUE:
255,0,0
//
// View.Colours.Vehicles.VEH_COLOR_CRANE  = RGB(0,255,0);
//----------------------------------------------------------------------
// The display colour for a crane.
// UNITS: RGB( 0 to 255, 0 to 255, 0 to 255 )
// NUMBER TYPE: integer, integer, integer
// VALUE:
0,255,0
//
// View.Colours.Vehicles.VEH_COLOR_LOWLOADER = RGB(100,100,100);
//----------------------------------------------------------------------
// The display colour for a low-loader.
// UNITS: RGB( 0 to 255, 0 to 255, 0 to 255 )
// NUMBER TYPE: integer, integer, integer
// VALUE:
100,100,100
//
// View.Colours.DrawElements.CLR_BACKGRND = RGB(255,255,255);
//----------------------------------------------------------------------
// The display colour for the background - usually white.
// UNITS: RGB( 0 to 255, 0 to 255, 0 to 255 )
// NUMBER TYPE: integer, integer, integer
// VALUE:
255,255,255
//
// View.Colours.DrawElements.CLR_ROAD_SURFACE = RGB(200,200,200);
//----------------------------------------------------------------------
// The display colour for the road surface.
// UNITS: RGB( 0 to 255, 0 to 255, 0 to 255 )
// NUMBER TYPE: integer, integer, integer
// VALUE:
200,200,200
//
//View.Colours.DrawElements.CLR_ROAD_LINES = RGB(0,0,0);
//----------------------------------------------------------------------
// The display colour for the road line markings.
// UNITS: RGB( 0 to 255, 0 to 255, 0 to 255 )
// NUMBER TYPE: integer, integer, integer
```

```
// VALUE:
0,0,0
//
//View.Colours.DrawElements.CLR_RULER_LINES = RGB(0,0,0);
//----------------------------------------------------------------------
// The display colour for the ruler tick lines.
// UNITS: RGB( 0 to 255, 0 to 255, 0 to 255 )
// NUMBER TYPE: integer, integer, integer
// VALUE:
0,0,0
//
//View.Colours.DrawElements.CLR_RULER_TEXT = RGB(0,0,0);
//----------------------------------------------------------------------
// The display colour for the ruler text.
// UNITS: RGB( 0 to 255, 0 to 255, 0 to 255 )
// NUMBER TYPE: integer, integer, integer
// VALUE:
0,0,0
//
//View.Colours.DrawElements.CLR_LEGEND_TEXT = RGB(0,0,0);
//----------------------------------------------------------------------
// The display colour for the legend text.
// UNITS: RGB( 0 to 255, 0 to 255, 0 to 255 )
// NUMBER TYPE: integer, integer, integer
// VALUE:
0,0,0
//
//View.Colours.DrawElements.CLR_DET_OUTPUT = RGB(255,255,0);
//----------------------------------------------------------------------
// The display colour for the output detectors.
// UNITS: RGB( 0 to 255, 0 to 255, 0 to 255 )
// NUMBER TYPE: integer, integer, integer
// VALUE:
255,255,0
//
//View.Colours.DrawElements.CLR_DET_FLOWDENSITY = RGB(0,255,0);
//----------------------------------------------------------------------
// The display colour for the flow-density detectors.
// UNITS: RGB( 0 to 255, 0 to 255, 0 to 255 )
// NUMBER TYPE: integer, integer, integer
// VALUE:
0,255,0
//
//View.Colours.DrawElements.CLR_DET_HEADWAY = RGB(0,0,255);
//----------------------------------------------------------------------
// The display colour for the headway detectors.
// UNITS: RGB( 0 to 255, 0 to 255, 0 to 255 )
// NUMBER TYPE: integer, integer, integer
// VALUE:
0,0,255
//
//View.Colours.DrawElements.CLR_DET_COMPOSITION = RGB(0,0,0);
//----------------------------------------------------------------------
// The display colour for traffic-flow composition detectors.
// UNITS: RGB( 0 to 255, 0 to 255, 0 to 255 )
// NUMBER TYPE: integer, integer, integer
// VALUE:
0,0,0
//
//View.Colours.DrawElements.CLR_FEAT_SPEEDLIMIT = RGB(255,0,0);
//----------------------------------------------------------------------
```

```
// The display colour for the speed limit regions.
// UNITS: RGB( 0 to 255, 0 to 255, 0 to 255 )
// NUMBER TYPE: integer, integer, integer
// VALUE:
255,0,0
//
//View.Colours.DrawElements.CLR_FEAT_GRADIENT = RGB(0,255,0);
//-------------------------------------------------------------------
// The display colour for the gradient regions.
// UNITS: RGB( 0 to 255, 0 to 255, 0 to 255 )
// NUMBER TYPE: integer, integer, integer
// VALUE:
0,255,0
//
//-------------------------------------------------------------------
//------------ END OF EVOLVETRAFFIC CONFIGURATION FILE -------------
//-------------------------------------------------------------------
```

© UCD/LCPC 2008

9.4   Appendix 4 – References

[1]   Treiber, M., Hennecke, A., and Helbing, D. (2000), 'Microscopic Simulation of Congested Traffic' in: *Traffic and Granular Flow '99*, Eds. D. Helbing, H.J. Herrmann, M. Schreckenberg, and D.E. Wolf, Springer, Berlin, pp 365-376.

[2]   http://www.mtreiber.de/

[3]   Kesting, A., Treiber, M., Helbing, D., (2007) 'General lane-changing model MOBIL for car-following models', *Transportation Research Record*: *Journal of the Transportation Research Board*, No. 1999, Transportation Research Board of the National Academies, Washington, D.C., pp. 86–94.

[4]   Treiber, M. and Helbing, D., (1999), 'Explanation of observed features of self-organization in traffic flow', preprint cond-mat/9901239. arXiv:cond-mat/9901239v1.

[5]   Treiber, M. and Helbing, D. (2001) 'Microsimulations of freeway traffic including control measures', Automatisierungstechnik **49**, pp. 478-484. http://www.arxiv.org/abs/cond-mat/0210096.