Trialling DASM as an Empirical Modelling Tool for Modelling the M6 and M6 Toll Motorway System Congestion

Abstract

This contribution consisted of two main objectives; Testing DASM[1]'s current effectiveness as an Empirical Modelling tool for modelling real world systems and producing an accurate and complete model of the M6 and M6 Toll motorway system to allow a cost effective and safe method of trialling various changes to determine which could potentially relieve congestion and reduce the number of road accidents.

This paper discusses the;

- Model Design
- Current state of the model
- Findings from using DASM for creating the model
- Issues identified with the current version of DASM
- Future work for the model

Model Design

The model was designed to accurately represent individual cars on the M6 and M6 Toll motorway system, heading southbound from junction 12, past junction 11 where the M6 Toll splits off and continuing on to junction 4 where they rejoin.

The distance from Junction 12 to Junction 11 was measured as 3,570m, Junction 11 to Junction 4 was 37,355m via the M6 and 38,900m via the M6 Toll. The measurements were obtained using a Google Maps distance measuring tool[2].

To make the modelling more manageable the road distances were scaled down to 1:10th. As the curvature of the motorway was deemed irrelevant from a modelling perspective and too complicated to implement, the motorways were modelled running horizontally with each other with two vertical connecting junctions connecting them at either end. As the two roads were made to run parallel, the two routes from Junction 11 to Junction 4 had to be changed to equal length. These scaled-down distances were tested and shown to work fine in the model but have since been reduced further to improve the model's effectiveness at representing congestion.

The motorway is represented as an agent with its own attributes. It specifies road conditions such as fog, ice and rain, what notices are shown to the drivers such as "M6 Toll Clear", "Slow" and "40mph", the rate of which cars are joining the motorway at the various junctions, the national speed limit for the motorways and the cost to use the toll road. It also specifies the location of traffic management chevrons that are commonly found on the M6 motorway:

Source: http://farm3.static.flickr.com/2604/3686639455_c306427af6.jpg

The cars on the motorway are also modelled as individual agents. They aim to attain a speed determined by their own ‘target speed’ attribute. They slow down if a car in front is within their preferred ‘time difference’ (‘time difference’ being how long it takes for them to reach the car in front’s current position). If they can save them selves from breaking by changing to a higher lane then they will. Likewise, they will change to a lower lane if they can do so safely. They break and accelerate at rates determined for each of them individually.

The ‘target speed’, ‘preferred time difference’, ‘acceleration’, ‘deceleration’, ‘likeliness to use the toll road’ and other attributes are dependent on three sources; the motorway agent attributes and the vehicle type and drive profile pairs selected with weighted randomness from a small database within the model. The information within the database was sourced from my own knowledge and can be improved with proper research.

Some example data for the vehicle types and driver profiles are included here:

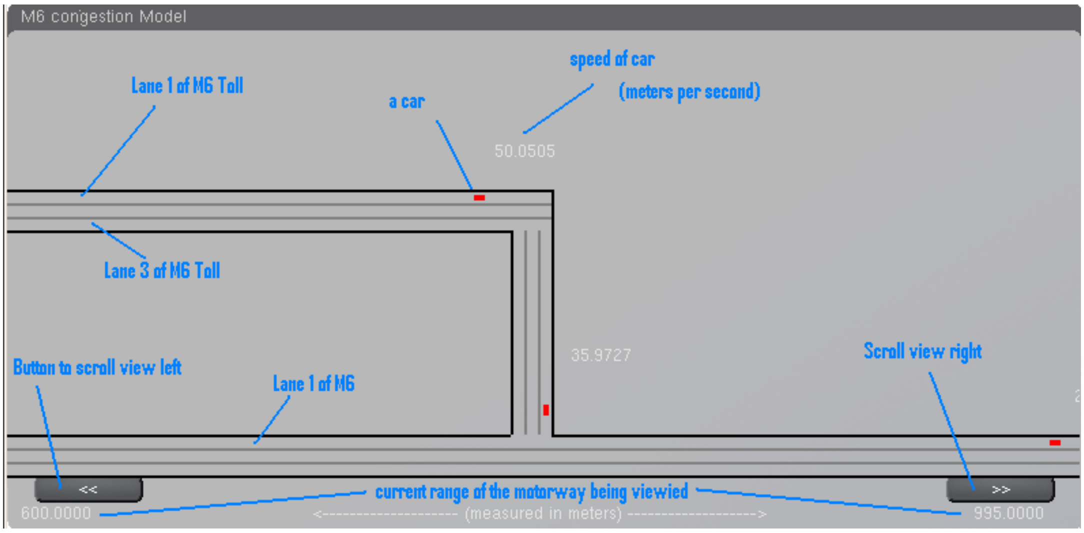
Driver Type: Student
Number of students per 100 drivers = 5
Percentage to exceed the speed limit = 10%
Response time = 800 milliseconds
Preferred time to each car in front = 1500ms
Preferred rate of acceleration = aggressive *(aggressive/normal/gentle)*
Probability of paying to use toll road in normal conditions = 5%
Percentage of driving a ‘4x4’ = 20%
Percentage of driving a ‘hatch-back’ = 75%
Percentage of driving a ‘sports car’ = 5%

Driver Type: Parent with children
Number per 100 drivers = 2
Percentage to exceed the speed limit = 0%
Response time = 1500 milliseconds
Preferred time to each car in front = 2000ms
Preferred rate of acceleration = gentle *(aggressive/normal/gentle)*
Probability of paying to use toll road in normal conditions = 15%
Percentage of driving a ‘4x4’ = 65%
Percentage of driving a ‘hatch-back’ = 35%
Percentage of driving a ‘sports car’ = 0%

Car Type: 4x4
Optimal deceleration = 13 meters/second
Optimal acceleration = 4 meters/second
Safety Rating = 4 *(0 to 5) ← may effect whether driver takes more risks*

Current State of the Model

Due to issues discussed later in the report the model was unable to be completed. Several features mentioned in the design have however been implemented and these are shown bellow. Firstly here is an annotated screenshot of the model's interface:



Each meter is represented by 2 pixels; a 2 by 4 meter car for example is represented with 4 by 8 pixels.

The model currently handles four cars. Each car can be manually given a target speed to achieve, a preferred time difference from the car in front, a rate of acceleration, rate of deceleration, lane to drive in and a true or false value stating whether it will use the M6 Toll. The model cars realistically drive and react as they would in a similar real world situation given the same attribute values. If a car pulls in front of another and the behind car cannot break fast enough or if a car breaks faster than a car behind is capable of then both cars will enter a 'crash' state. 'Crashed' cars instantly come to a stop and start flashing, additional cars behind will attempt to stop to avoid from joining the pile-up. The 'crashing' of cars is not realistic as even the slightest touch will cause both cars to stop instantly.

When a car leaves the motorway it re-initialises at the start with zero speed after a user-defined delay for that particular car. To avoid constant pile-ups, cars are prevented from joining the motorway unless the first 10 meters of Lane 1 is free.

Implementations for allowing cars to change lane were almost complete but were halted by a difficult to identify bug with DASM. DASM was not upholding some dependencies in a predictable 'one and the same' manner. A car would move to the next lane, and before other code could re-determine the distance to the car in front in the new lane, DASM incremented the lane attribute several more times. The code can be re-activated by un-commenting the code found within the 'car1' agent.

Findings with using DASM for creating the model

DASM provides IF and Switch statements but 'For' loops, indexable arrays (or linked lists) and a pseudorandom number generator are required to make this model scalable to more than a few cars and allow the aforementioned 'internal database' to be implemented so it can randomly generate new driver-vehicle pairs for each car.

Four cars were chosen as a manageable number but this can be increased with additional programming. It does however require excessive duplicating of code and variable re-naming which becomes highly error prone as more cars are added. If N is the number of cars in the model then each car agent requires NK if-statements to determine the distance to the car in front and in adjacent lanes.

The tool provides powerful and easy to use features for developing user interfaces. Its dependency tree is also very well implemented and perfectly handled all but one of my many complex dependencies.

Issues identified with the current version of DASM

DASM is currently a prototype language so some bugs are to be expected. Listed here are the particular ones which I discovered and issues that proved to be significant factors in preventing me from completing the model in time.

When a parsing error occurs, incorrect line numbers are given because empty and commented-out lines are ignored. Much time was spent simply finding which lines the parser was referring too.

There is no variable name checking so a spelling error can generate a new variable with a null value and cause erroneous behaviour in a model without warning.

Operator precedence is wrong with logical ‘or’ taking precedence over ‘and’.

Parsing flaws mean:
`“(3==3)”` evaluates to false, `“(3 == 3)”` evaluates to true,
`“(false or (4 == 4 and 5 == 5 and 3 > 0))”` evaluates to false,
`“(false or (4 == 4 and (5 == 5) and (3 > 0)))”` evaluates to true.
As these flaws occur silently, it can take hours to determine the source of sporadic model behaviour.

`“((3 == 3) and (2 == 2)) or ((1 == 1) and (4 == 4))”` produces a parser error as two opening brackets are used together “((“. This bug makes it extremely difficult to write condition statements as much time must be spent re-arranging the expressions to work without multiple adjacent brackets which themselves are heavily required due to the parsing flaw mentioned above.

Future Work for the Model

Future work for the model can be done on resolving the previously mentioned lane changing issue, introducing chevrons which request drivers to keep a set distance, developing a more sustainable way of introducing additional cars to the model, implementing the pseudorandom driver-vehicle pair selection from an internal database and finally using the model to determine possible ways to reduce congestion and traffic accidents on the notoriously busy M6 motorway.

References

[1] http://www.doste.co.uk/thesis/6dasm
[2] http://www.daftlogic.com/projects-google-maps-distance-calculator.htm