Abstract
This report describes how the Nagel-Schreckenberg (NaSch) cellular automata model is applied to a single road for the purpose of looking at flux and density in the fundamental diagram. An analytical model is developed in parallel to complement and verify the numerical model. The NaSch model is then expanded to model two roads merging into one, with the development of suitable rules for merging and boundary conditions. The numerical efficiency of this cellular automata model is demonstrated.
Acknowledgments

I would like to firstly thank my project supervisor Mauro Ferreira for providing me with such an interesting project and for the useful and indeed essential discussion and feedback throughout. My thanks also to Dr. René Meier and Vinny Reynolds both from the Distributed Systems Group of the Department of Computer Science, Trinity College Dublin, and to Dr Michael Markowski of the University of Delaware for all their expertise in traffic modelling which they shared with me. I am indebted to Pat, Brid, Nuala, Lorna and Caitriona for their proof-reading throughout Christmas. And finally thank you to Sían for her continued support throughout the course of the project.
Contents

Abstract ........................................................................................................................................i
Acknowledgments ..........................................................................................................................ii
Contents ...........................................................................................................................................iii

1 Introduction ..................................................................................................................................1
  1.1 Motivation .................................................................................................................................1
  1.2 Goals .........................................................................................................................................1

2 Background ....................................................................................................................................2
  2.1 Traffic Jams; Definition, Causes, and the Phenomenon of Phantom Traffic Jams.................................2
  2.2 The Fundamental Diagram ........................................................................................................3
  2.3 Types of Traffic ..........................................................................................................................4
  2.4 Summary of Available Models ...................................................................................................4
    2.4.1 Macroscopic – fluid-dynamics .................................................................................................4
    2.4.2 Mesoscopic – gas-kinetic models ............................................................................................5
    2.4.3 Microscopic – car-following model .......................................................................................5
    2.4.4 Microscopic – cellular automata model ..................................................................................6
  2.5 NaSch cellular automata model ..................................................................................................6
  2.6 Analytical Model .......................................................................................................................8
    2.6.1 A car-following analytical model ..........................................................................................8
    2.6.2 Stochastic factor ...................................................................................................................10

3 Experimental Method ..................................................................................................................13
  3.1 Numerical single road ................................................................................................................13
    3.1.1 Programme structure .............................................................................................................13
    3.1.2 Measuring density and flux ..................................................................................................14
    3.1.3 Random numbers ..................................................................................................................14
  3.2 Two Roads Merging into One ....................................................................................................15
    3.2.1 Discussion of Model and Merging Rules ..............................................................................15
    3.2.2 Closed loop / open loop .......................................................................................................16
  3.3 Empirical Data ..........................................................................................................................18

4 Results ..........................................................................................................................................19
4.1 Single Lane Road ................................................................. 19
  4.1.1 Position-time graphs ...................................................... 19
  4.1.2 Effect of measurement parameters ................................. 21
  4.1.3 Effects of speed limit and stochastic factor ...................... 22
4.2 Reconciling Analytical & Numerical Models .................................. 23
  4.2.1 Deterministic limit ....................................................... 23
  4.2.2 Stochastic factor .......................................................... 24
  4.2.3 Empirical Data ............................................................. 25
4.3 Two Roads Merging into One .................................................. 26
4.4 Numerical Efficiency .............................................................. 28
5 Conclusions .............................................................................. 29
6 Further Work ............................................................................ 29
7 References ............................................................................... 30
Appendix A The source code ..................................................... A-1
1 Introduction

1.1 Motivation

“The volume of vehicular traffic in the past several years has rapidly outstripped the capacities of the nations’ highways. It has become increasingly necessary to understand the dynamics of traffic flow and obtain a mathematical description of the process.”

Quotation from H. Greenberg (US) in 1959. [1]

Traffic is a major problem in all developed countries around the world. As economies grow there is an ever increasing level of traffic, more goods to be transported, more people travelling to work and to leisure activities. Traffic jams are a drain on the economy and do damage to the environment. Building more roads and adding extra lanes to existing roads does help to alleviate the problem but this is both expensive and detrimental to the environment. As an alternative there would be considerable economic and environmental benefits in making better and more efficient use of existing roads. The ultimate goal would be to keep roads at maximum throughput, but to achieve this we need to gain a better understanding of traffic dynamics and how different factors affect the flux (flow) and capacity of a road.

1.2 Goals

In this project my goal is to take a phenomenon in physics and apply a suitable model to it so that its characteristics may be reproduced and predicted analytically and numerically with a computer programme. The physical phenomena I have chosen is vehicular traffic, traffic flow and traffic jams.

I aim to make a brief review of existing available traffic models, to select a suitable numerical model and implement it with a computer programme. In parallel to this, I aim to create an analytical model that will complement and validate the numerical model. I intend to use this model to look at a 1D straight road with periodic boundary conditions and get the fundamental diagram of density versus flux. I then intend to make a comparison of the
model with empirical data to see if the model has in fact reproduced real characteristics.

Based on the foregoing I then aim, “as a first step”, to reproducing complex road networks, to introduce a junction in the form of two roads merging into one, looking at the effects this has on the dynamics of traffic and the fundamental diagram.

2 Background

2.1 Traffic Jams; Definition, Causes, and the Phenomenon of Phantom Traffic Jams

Traffic jams, defined as “a situation where all road traffic is stationary or very slow” can be caused by a number of factors including reduction in the capacity of the roadway, by excess demand for the given capacity, or by a phenomenon called phantom traffic jams. Traffic jams caused by a reduction in capacity can originate from the presence of a toll-booth, a reduction in the number of lanes or an accident. Traffic jams caused by excess demand occur because of an increase in volume from an on-ramp onto a motorway or the merging of two or more roadways. Traffic jams can also be formed for no apparent reason, resulting in so-called “phantom traffic jams”. They are due to imperfect driving such as over-braking, tailgating, loss of concentration, rubbernecking, slow traffic switching lanes, or the presence of a speed camera. Their concentration and frequency grows with density. Phantom traffic jams grow from small fluctuations in traffic behaviour rather than being triggered by large, exceptional events.

As vehicles join the back of phantom traffic jams and others leave the front, the jams propagate upstream against the flow of the traffic, whereas the front end of traffic jams caused by other factors such as toll booths remain stationary.
2.2 The Fundamental Diagram

One of the fundamental parameters used when analysing traffic dynamics is flux $\Phi$, referred to as current or flow in some papers. It is the number of vehicles passing a fixed point per unit time. The other important parameter is density $\rho$, the number of vehicles per unit length.

The relationship between vehicle flux and vehicle density gives a good understanding of what is happening to traffic. The Fundamental Diagram in traffic modelling is a plot of traffic flux and local time averaged density as shown in fig 1. At low densities when the traffic is in a free flowing phase, flux increases linearly with density. At high densities the traffic is in a congested state and flux decreases with density to a limit of no vehicles moving on a full road. Near the point of maximum flux there is a phase transition from free traffic to congested traffic.

![Fundamental Diagram](image)

*fig 1. Sketch of the fundamental diagram showing the expected relationship between density and flux*

Flux is a time-based parameter while density is a space-based parameter. When analysing both real traffic and modelled traffic, flux is measured by averaging the vehicles passing a fixed point over a time window, often 5 minutes [2]. The associated density is measured “local” to that point and averaged over the same time window.
Another important parameter used when describing traffic dynamics is the **distance headway** $d$: it is the distance from the front of one vehicle to the back of the next vehicle. In traffic this is the predominant factor that drivers use when determining the appropriate speed at which to travel.

### 2.3 Types of Traffic

For a traffic model to be complete and give a true representation of real roads, account would have to be taken of the different types of vehicles using that road; cars, trucks, busses and motorbikes. Each type would have a different length, acceleration and maximum velocity. The aim of my project is to model the main characteristics of traffic and so I have elected to confine all of my models to a single vehicle type, a ‘car’. All cars in the model have equal length, rate of acceleration and maximum velocity. The generalisation to include more vehicle types has been left for further work after this project.

### 2.4 Summary of Available Models

#### 2.4.1 Macroscopic – fluid-dynamics

At a macroscopic scale traffic on a road appears to flow like a fluid in a stream. A macroscopic theory can be developed by adapting the hydrodynamic theory of fluids. Traffic can be thought of as essentially a one-dimensional comprisable fluid. Away from junctions no vehicles enter or leave the system so there is a conservation of vehicles. Lighthill and Whitham were the first to propose the continuum model in 1955 [1]. The traffic at position $x$ and time $t$ has density $\rho(x,t)$ and average velocity $v(x,t)$. The continuum model is given by

$$\frac{\partial \rho(x,t)}{\partial t} + \frac{\partial \Phi(x,t)}{\partial x} = 0$$

where $\Phi(x,t) = \rho(x,t)v(x,t)$ is the traffic flux. Kerner and Konhauser (1993, 1994) investigated this and gave the complete continuum model as

$$\frac{\partial \rho(x,t)}{\partial t} + \frac{\partial (\rho v)}{\partial x} = 0$$

$$\rho \frac{\partial v}{\partial t} + \rho v \frac{\partial v}{\partial x} = \frac{\rho}{\tau} \left[ V(\rho) - v \right] - c_0^2 \frac{\partial \rho}{\partial x} + \mu \frac{\partial^2 v}{\partial x^2}$$

where $\tau, c_0^2$ and $\mu$ are phenomenological constants [3].
This model continues to be under active research, and is constantly being improved to better match physical phenomenon of traffic such as density waves and driver anticipation. This is a numerically efficient model that gives good agreement with empirical data and is suitable for analytical evaluation.

2.4.2 Mesoscopic – gas-kinetic models

The kinetic theory treats vehicles as a gas of interacting particles where each particle represents a vehicle. Prigogine and Herman in 1971 used the Boltzmann equation to describe traffic

\[
\frac{\partial f(x,v,t)}{\partial t} + v \frac{\partial f(x,v,t)}{\partial x} = -f(x,v,t) + \frac{\rho(x,t)F_{\text{des}}(v)}{\tau_{\text{rel}}} + \left( \frac{\partial f(x,v,t)}{\partial t} \right)_{\text{int}}
\]

(4)

where the first term on the right-hand side represents the relaxation of the velocity distribution function \(f(x,v,t)\) to the desired velocity distribution \(\rho(x,t)F_{\text{des}}(v)\) with the relaxation time \(\tau_{\text{rel}}\) in the absence of the interactions of vehicles and the second-term on the right-hand side takes into account the change arising from the interactions among vehicles [3].

Kinetic gas theory model also continues to be under active research, and it has been expanded from a single road model to a two-dimensional city traffic model [1].

2.4.3 Microscopic – car-following model

In the “follow-the-leader” model individual cars on a road can be thought of as interacting particles obeying Newton’s equations of motion. The motion of a vehicle is affected only by the motion of the vehicle in front of it. In a sense cars are dragged along the road by those in front of them. The acceleration of the n'th vehicle is determined by the difference in velocity between the n+1'th vehicle and it:

\[
\ddot{x}_n(t) = \frac{\dot{x}_{n+1}(t) - \dot{x}_n(t)}{\tau}
\]

(5)

where \(\tau\) sets the time scale and the “sensitivity” of the model [2]. Differentiating both sides with respect to time gives the distance headway:

\[
\Delta x_n(t) = x_{n+1}(t) - x_n(t) = d_0 + \tau \dot{x}_n(t)
\]

(6)
This equation gives vehicles a minimum safe distance $d_0$ to the vehicle in front as well as giving increasing distance-headway with increasing velocity to avoid collisions.

This microscopic model computes the positions and velocities of individual vehicles closer than all of the other types of models but it is computationally slow requiring the computation of floating-point numbers. It is best suited to small scale simulations of individual junctions or small road networks.

### 2.4.4 Microscopic – cellular automata model

Cellular automata (CA) models are an idealization of physical systems by using discrete time and space and each unit can only have a finite number of discrete states. The best known CA model is the 1970’s Conway’s game of life. CA models are popular because in general they are highly computationally efficient due to their use of integer values for time, space and state.

The best known CA model for traffic was developed by Nagel and Schreckenberg (NaSch) in 1992 [4]. This is the model I have chosen to base my numerical simulation on and it is discussed in more detail in the next section below.

### 2.5 NaSch cellular automata model

In the Nagel-Schreckenberg model (NaSch) [1,2,4], the road is divided into discrete cells or lattice sites; each cell is either empty or occupied by exactly one car at each time step. Position, velocity, acceleration and time are treated as discrete integer variables making this a computationally efficient model. The n’th vehicle can have a velocity $v_n=0,1,2,\ldots,v_{\text{max}}$. As a first step the road is initialised randomly as being occupied 1 or unoccupied 0. At each occupied site, a car is assigned an integer velocity $v \leq v_{\text{max}}$. Car positions are updated in parallel. Their velocities are set according to the following rules.
i. **Accelerate**: All cars are accelerated by one until a maximum of $v_{\text{max}}$. This represents how drivers like to drive at the speed limit where possible.

$$v_n \rightarrow \min(v_{n+1}, v_{\text{max}}) \quad (7)$$

ii. **Decelerate**: If there is a car closer than $v_n$, decelerate so as not to crash.

$$v_n \rightarrow \min(v_n, d_{n+1}) \quad (8)$$

iii. **Randomization**: With a given probability $q$, randomly decrease the velocity by one. This represents imperfect driving and over-braking.

$$v_n \rightarrow \max(v_n - 1, 0) \text{ with probability } q \quad (9)$$

iv. **Move**: With all the velocities set, the position of each car is updated.

$$x_n \rightarrow x_n + v_n \quad (10)$$

These steps can be best understood with a visual example in fig 2.

To avoid creating and destroying cars at either end of the road periodic boundary conditions can be used. For $x_n + v_n > L$, where $L$ is the length of the road:

$$x_n \rightarrow x_n + v_n - L \quad (11)$$

With this approach cars are conserved and an infinite road is simulated.

By updating car positions in parallel as opposed to sequentially or randomly, chain overreactions can lead to spontaneous phantom traffic jams similar to those observed in real traffic [2].
An addition to the NaSch model is the **“slow to start”** probability $p$. When cars come to a complete stop, there is a probability $p$ that they do not move / accelerate at all, in that time step. This reproduces the phenomenon of how drivers are slow to move off from a standing start. In real situations this further adds to the problem of phantom traffic jams and the size of the tail back associated with them.

### 2.6 Analytical Model

#### 2.6.1 A car-following analytical model

Taking a special case of the car-following model in which cars are evenly spaced along the road and move with uniform velocity. With $d_0$, equal to the
car length plus a minimum gap between cars, and the distance headway $d$ between cars (fig 3), a road segment with $N$ cars would have a length

$$L = N(d_0 + d)$$  \hspace{1cm} (12)

with a density

$$\rho = \frac{N}{L}$$  \hspace{1cm} (13)

$$= \frac{1}{(d_0 + d)}$$  \hspace{1cm} (14)

**fig 3. Car spacing.** Where $d_0$ is the car length plus a minimum gap and $d$ is the distance to the next car.

The flux $\Phi$ is a function of density and is defined by

$$\Phi(\rho) = \nu\rho$$  \hspace{1cm} (15)

where $\nu$ is each car’s velocity and is function of $d$. A choice has to be made to decide the safe velocity for a given distance. One approach could be that a car must be able to come to a complete stop within that distance.

$$\nu = \sqrt{-2ad}$$  \hspace{1cm} (16)

where $a$ is the cars rate of deceleration ($a<0$). However this approach is inappropriate because driver reaction time isn’t taken into account and drivers generally don’t drive in this manner. A more appropriate approach is to use the road safety “Two-second rule”, in which a constant temporal distance $t_0$ is left between each car, ie. two seconds.

$$\nu = \frac{d}{t_0}$$  \hspace{1cm} (17)

Speed limit now needs to be taken into account, otherwise, as the density goes to zero, velocities would go to infinity.

$$\nu = \min\left(\nu_{\text{max}}, \frac{d}{t_0}\right)$$  \hspace{1cm} (18)
This naturally gives rise to the two phases in the fundamental diagram, the free flow phase \( \frac{d}{t_0} > v_{\text{max}} \) and the jammed phase \( \frac{d}{t_0} < v_{\text{max}} \).

The resulting flux is then given by

\[
\Phi(d) = v_{\text{max}} \quad \text{free-flow} \tag{19}
\]

\[
\Phi(d) = \frac{d}{t_0 (d_0 + d)} \quad \text{jammed} \tag{20}
\]

or

\[
\Phi(\rho) = \begin{cases} 
  v_{\text{max}} \rho & \text{for } \rho \leq (v_{\text{max}} t_0 + d_0)^{-1} \\
  1 - \frac{d_0 \rho}{t_0} & \text{otherwise}
\end{cases} \tag{21}
\]

Fig 4 shows a plot of these equations for a set of chosen variables.

2.6.2 Stochastic factor

This analytical model is not complete without a stochastic factor comparable to the slow down factor in the NaSch CA model. This factor should have the effect of reducing the flux due to imperfect driving, particularly in the jammed phase.
A car’s velocity is found by the time $t$ it takes to travel a distance $L$.

$$v_0 = \frac{L}{t} \quad (22)$$

where $v_0$ is the deterministic velocity. Assuming that a car will stop with a given probability for a time $\tau_i$. In sufficiently dense traffic any car ahead of the subject car that stops will also cause the subject car to stop (fig 5). $\tau$ is then the sum of all of these stops over a distance $L$

$$\tau = \sum \tau_i \quad (23)$$

As shown in fig 6 a car travels with a velocity $v_0$ for a total time $t$ and is stopped for a total time $\tau$. The reduced velocity $v'$ is given by

$$v' = \frac{L}{t + \tau} \quad (24)$$

and from (22)

$$v_0 t = v' (t + \tau) \quad (25)$$
The change in flux is given by
\[
\Delta \Phi = (v_0 - v') \rho \tag{26}
\]
\[
\Delta \Phi = \left(1 - \frac{t}{t + \tau}\right) v_0 \rho \tag{27}
\]

Let
\[
\gamma = \left(1 - \frac{t}{t + \tau}\right) \tag{28}
\]
where \(\gamma\) is a constant describing the degree of randomness or “imperfect driving”.
\[
\Delta \Phi = \gamma v_0 \rho \tag{29}
\]
and the reduced flux is
\[
\Phi' = (1 - \gamma) \Phi_0 \tag{30}
\]

In the low density free flow phase cars experience very little effect from traffic ahead slowing down, there is little or no cumulative effect of the slow-down and so the factor \(\gamma = 0\).

From (21) and (30)
\[
\Phi'(\rho) = \begin{cases} 
  v_{\text{max}} \rho & \text{for } \rho \leq \left(\frac{v_{\text{max}} t_0}{(1 - \gamma)} + d_0\right)^{-1} \\
  \frac{(1 - d_0 \rho)(1 - \gamma)}{t_0} & \text{otherwise}
\end{cases} \tag{31}
\]

Suitable values for the dimensionless scalar \(\gamma\) and comparison with the NaSch CA \(q\) factor is given in section 4.2.

Also of note, by a Taylor expansion for \(\tau \ll t\)
\[
\gamma \approx \frac{\tau}{t} \tag{32}
\]
3 Experimental Method

3.1 Numerical single road

Using just the NaSch CA rules from section 2.5 I wrote a programme in C. The basis of the programme is an array representing the road with discrete cells. Each cell is a structure containing information about the vehicle currently occupying it. The road has periodic boundary conditions so that when a car reaches the end of the array it is repositioned back to the start.

3.1.1 Programme structure

Information held for each road cell:
- Site occupancy, 0 or 1
- Car velocity $v_n$
- Journey Time, set to zero at the start of the road and incremented each time step.
- Car Number, to aid visualisation
- Origin, used in the 2 roads merging model

Parameters given to the programme:
- Road length
- Number of cars
- Speed limit, $v_{\text{max}}$
- Slow down probability q
- Slow to start probability p
- Number of steps
- Size of the time window over which flux and density measurements are averaged
- Area over which to take density measurements
- Relaxation time, after initial “seeding” before measurements start

Default values for these parameters are coded in, but customisation can be made through the command line.
The road is then initialised by seeding it with the specified number of cars in random positions with random initial velocities. The simulation then runs, setting the velocity of each vehicle using the 3 rules (i, ii and iii from section 2.5) with one function and then updating the positions with another function. A relaxation time is allowed to pass before any measurements are taken so that any artefacts caused by the random seeding are allowed to dissipate.

### 3.1.2 Measuring density and flux

For convenience the “measuring pole” is positioned at the end of the road. When a car passes this point the flux count is incremented by one. Flux is measured repeatedly over the given time window. The density measured is a “local time averaged density”. This is done by counting the cars either side of the flux point for a given length, as shown in fig 7. Density is calculated at every time step and averaged over the same time window as its associated flux. At the end of this time window the counts are normalised, printed to a file for plotting and reset to zero.

![fig 7. Showing how the local time averaged density is measured either side of the flux measurement with a length of 60 units for a road of length 100 units (say).](image)

For the purpose of getting the fundamental diagram, starting with a near empty road the programme is run repeatedly adding more cars each time when initialising the road. The number of cars used can be increased in steps of 1, 2, 5, 10 or 20 depending on the quality versus computation speed required.

### 3.1.3 Random numbers

Random numbers are used throughout the program;

- in initialising the positions and velocities of cars
- in the randomisation step iii (9)
- in the “slow to start” rule.
The ISO C pseudo-random number generator is used to create numbers in a suitable range. The seed is taken from the clock time so that each run of the programme is unique. When setting the initial position of the cars the programme repeatedly loops over the road with a low probability until all of the cars have been distributed. This ensures a good even spread.

3.2 Two Roads Merging into One

3.2.1 Discussion of Model and Merging Rules

As a first step to developing the model to reproduce a real road network I chose to have two roads merge into one. This is analogous to an on-ramp merging onto a single lane motorway. As shown in fig 8 the two roads exist in parallel, independent of each other. Then there is a merging zone in which cars on road B must try and merge onto road A before road B comes to an end.

![Diagram of 2 road merging model](fig 8. diagram of 2 road merging model)

Each road is updated independently, alternating between each, for every time step.

The system is very dependent on the rules chosen for merging traffic. Priority should be given to those already on the main road A and cars on road B should filter in when there is space for them to do so.

My initial set of merging rules allowed B cars to move over whenever there was space. This led to virtually no flow on A while B cars were free flowing. This was because a car from B might slot in directly in front of an A car causing it to come to a complete stop in the next time step and thus create a traffic jam on road A and thereby making it even easier for even more B cars
to move in. It is not physically realistic or desirable that on-ramp traffic would get priority over the main motorway traffic, so a revision of the merging rules was necessary. I imposed the rule that a car on road B can only move over if there are two empty spaces free, i.e. the empty space that the car is moving into and an empty space behind the one that it is moving into. This ensures that cars on road A can at least maintain a velocity of 1.

Merging rules, for a car on road B in the “merging zone”:

1. The car’s velocity is still set by the road that it is on.
2. The car will try and move over at the earliest possible time when there are at least two spaces available on road A.
3. If the car succeeds in moving over it will move up on that new road as much as it can, until it catches up with another car or to the limit that its velocity allows.
4. If it cannot move over, its velocity is reduced by one and it moves forward on its own road to the limit where it reaches the end of the road, at which point it must stop.

It should be noted that the length of the merging zone can be set to one if it is required to model a T junction with a yield sign.

3.2.2 Closed loop / open loop

In this “two roads merging model”, that comprises of one exit and two entrances, the closed loop (periodic boundary conditions) used for the “single road model” no longer seemed appropriate. Therefore, I first tried an open system in which cars are created at the start of both of the roads and allowed to run off and disappear at the end of the one road. However this presented
two problems. First, at the end of the road there was nothing to slow the cars down; they flowed away freely as if there was a motorway with infinite capacity ahead of them. This had the knock-on effect of all cars on the combined stretch moving freely at their maximum speed. Second, cars couldn't be added at the start of both roads as quickly as they were leaving at the end of the combined road. The net result was that higher densities could not be modelled.

Researching this subject on the internet I discovered a website [7] with a similar CA model in Java that didn't appear to have this problem. In their model some cars towards the end of the road slowed down for no apparent reason and created the same flow characteristics as the single closed loop road. I emailed the author Dr Michael Markowski and got a very helpful response, he told me that he had experienced the exact same problems as I had when developing his programme. His solution was that when setting the velocity for the car at the end of the road, he had the programme look at the distance headway of a few cars behind it and take the average as the headway used in setting its speed. I implemented this in my code and got some satisfaction from it. It worked well at times and I was able to maintain high densities, but it tended to become unstable if the density was to drop for a short period, due to fluctuations. In this situation the average headway used by the last car would be large, thus allowing it to move away more quickly, this in turn would cause a further drop in density with a corresponding increase in headway for the next car thus ultimately leading to a runaway effect.

Another suggestion Dr Michael Markowski made was to take a semi-closed approach. When measuring the headway for the final car on the road, he suggested looping around to the start, but when updating the positions allow the final car to disappear when it reaches the end. This also wouldn't work for the two roads, just as the closed approach didn't, but it did give me the idea of using a stretch of road, say 20 spaces back for establishing the headway for the last car, as if projecting an image of the final stretch of road beyond the end of the road, creating a pseudo periodic boundary condition. This also worked partially and did allow higher densities to be modelled, but it was even
more unstable than previous attempts. If the programme was run over a substantial time span the road became completely blocked. This occurs because a case would arise in which a spontaneous phantom traffic jam would occur at the site from which the projected image was taken and cause the end of the road to jam, leading to the whole end section of the road being full of cars, with no cars been able to move anywhere.

The successful solution I discovered was to label each car with their road of origin and do a closed loop based on this. Cars originating from road A go back to the start of road A, if there is space for them, and cars originating from B loop back to the start of road B. This simple solution works very well and gives the expected results. A benefit from this is that cars are added to the start of both roads at the same rate that they leave, giving a relatively constant density.

3.3 Empirical Data

In the hope of obtaining some empirical data in which to fit my model to, I made contact with Dr. René Meier from the Distributed Systems Group (DSG) of the Department of Computer Science, Trinity College Dublin. He has done extensive work on traffic, specifically developing more intelligent traffic light systems. He had extensive data from detector loops at traffic lights across Dublin city over a period of approximately five years. This data gave the flux through a junction for each cycle of the lights. Unfortunately I was unable to use this data to make a fundamental diagram because densities were not recorded.

René directed me to Vinny Reynolds, also from DSG, who had worked on a traffic model of the streets surrounding Trinity College for the purpose of simulating intelligent traffic lights. We found that it wasn’t practical to make a comparison of the results from the two models but we did have an interesting discussion. His model is based on a car following model which worked well for accurately reproducing a junction but does not scale well. The simulation “ground to a halt” when more then a few roads and junctions were modelled.
He was impressed with the speed of my model and expressed an interest in using a cellular automata model for a city wide simulation.

Vinny also gave me a number of useful suggestions that I could try with my simulation, such as driver anticipation for cars approaching a jam.

I was unable to obtain empirical data directly but I was able to use a fundamental diagram made from empirical data given in [2], the results from this are shown in section 4.2.3

4 Results

4.1 Single Lane Road

Once I had the working programme I started experimenting with it to see how the model would react to different variables. It is generally assumed [1,2,4] to use $v_{ca, max}=5$ and $q=0.5$ for motorway traffic.

4.1.1 Position-time graphs

While developing and testing the code I found it necessary to see what was happening to the vehicles in different situations. I did this by printing out an ASCII representation of the road, with each line representing the road at one time step. Using velocity dependent characters it was easy to visualise the movement and jams of the cars (fig 10). With a simple switch it was also possible to print out instantaneous velocities or car numbers allowing one to track the progress of an individual car.

![Position-time graphs](image)
In the position time graph below (fig 11) each dot represents a car at each time slice. We can see that dense areas represent compacted and jammed traffic. As the cars leave the front of the jam moving forward in time and space, towards the top right, they have a curved trajectory due to their acceleration. The jammed regions move backwards in space as cars leave the front and others join at the back.

![Position-time graph](image)

*fig 11. A detail from fig 12, showing position-time plot vehicles with q=0.5 at 25% density*

The left plot of fig 12 shows how the model behaves with no stochastic factor. All cars react to the jams in the exact same way, and so the jams move back along the road in a regular fashion. The irregularity of jam size and spacing comes from the random initial seeding. In the right plot with q=0.5 we can see spontaneous traffic jams forming as others dissipate. The plots are both taken

![Position-time plots](image)

*fig 12. A position-time plot of a road of length 400 units at 25% density, the left image is q=0 and the right is q=0.5. Dense areas highlight jams.*
from 1000 time steps into the simulation to allow artefacts from the initial seeding to dissipate. One can also see the periodic boundary conditions in effect. The jam leaving around the 1050 mark is the same one reappearing at the top.

Fig 13 illustrates increasing congestion with increasing density. The jams propagate backwards at a regular rate and can be easily measured from the graph as shown in fig 14.

**4.1.2 Effect of measurement parameters**

Suitable choices need to be made for the density length and the size of the flux time window in order to obtain a sharply defined fundamental diagram.
Reducing the length that the density is measured over and increasing the flux time window gives closer correlation of the points in the diagram. With experimentation I found that a length of 20 cells and a time of 300 to 1000 time-steps gave good results. Using a road of length 500 cells or more was suitably long.

4.1.3 Effects of speed limit and stochastic factor

The effect of speed limit on the fundamental diagram can be seen in fig 15. For very low densities a high speed limit allows the few cars on the road to flow along quickly, producing a high flux, but for densities higher than approximately 10% a speed limit greater than 5 has no affect since the predominant factor limiting velocity is the distance to the car in front. Also we can see that with the low limit of $V_{\text{max}}=2$ or 40km/h, there is a reduction in flux up to about 30% density.

Another adjustable variable in the model is the slow down probability which has the effect of creating the jammed phase earlier and reducing flux in this phase as shown in fig 16.
4.2 Reconciling Analytical & Numerical Models

The NaSch CA model is based on scaleless units (unitary cell length and timestep size). This is where the analytical model can be put to use. By fitting the graphs from both models, SI units can be assigned to the output of the CA model, making it suitable for comparison to empirical data.

4.2.1 Deterministic limit

Using the deterministic limits ($q=0$ and $\gamma = 0$) of both models a fit can be made by choosing suitable values for the variables $t_0$, $v_{\text{max}}$ and $d_0$. The reaction time $t_0$ from the analytical model is equivalent to the step size used in the NaSch model. Generally the variables $t_0=1\text{second}$ and the cell size $d_0=7.5\text{m}$ are chosen [1,2,4] implying $v_{\text{max}}=37.5\text{ms}^{-1}$ or $135\text{km/h}$. Using these values in (21) gives a perfect fit to my numerical model with $q=0.0$. I have had to assume that these values were chosen by fitting to some suitable empirical data set.

I consider that these variables should not be fixed, but instead should be adjustable to fit different driving styles. Driving styles can differ from one country or region to the next such that vehicles may drive closer in one then the other. The variables should be chosen such that the proportion
\( \frac{t_0 v_{\text{max}}}{d_0} = v_{\text{cmax}} \) holds for the deterministic limit. In several countries, including Ireland, the non-motorway speed limit is 100km/h so \( v_{\text{max}} = 27.78 \text{ms}^{-1} \) would be more appropriate, for \( t_0 \) we could use 1.2 seconds and this would give \( d_0 = 6.67 \text{m} \). Fig 17 shows this possible set of variables and the conventional set. Both choices may be appropriate for different types of roads and when it is required to model a particular road careful comparison with empirical data from that road type is necessary.

![Fig 17. Plots of numerical and analytical methods combined for two different scales. Both plots use the same numerical data. The left plot is scaled for 100km/h with a gap of 1.2 seconds between cars, the right is scaled for 135km/h and a gap of 1 second.](image)

### 4.2.2 Stochastic factor

Once suitable values for \( t_0 \), \( v_{\text{max}} \) and \( d_0 \) are chosen for the deterministic limit in the previous section a comparison can be made between the numerical \( q \) and the analytical \( \gamma \) (28). Fig 18 and 19 show this relationship.
4.2.3 Empirical Data

When dealing with empirical data with only flux and density information there is no distinction between $t_0$ and $(1-\gamma)$ when fitting equation (31). One can only find the ratio $\frac{t_0}{1-\gamma}$. Therefore the characteristic temporal head-way $t_0$ must be measured by some other means to explicitly obtain both $t_0$ and $\gamma$.

fig 19. Plot of $q$ vs $\gamma$ relationship taken from fig 18

fig 20. Black points taken from [2] "Empirical data for flow and occupancy [density]. The data has been collected by counting loops on a Canadian highway. Both the occupancy and the flow have been directly measured by the detector. Each point in the diagram corresponds to an average over a time interval of 5 min. The red line fit and per second scale added. Peak of fit is at $\Phi=0.580\, s^{-1}, \rho=15.00\%$. 
Due to the unavailability of empirical data I have used a plot (fig 20) from [2] and fitted it with the red lines. Peak of fit is at $\Phi=0.580s^{-1}$, $\rho=15.00\%$. The inscription states that the data is from a Canadian highway. The speed limit for most Canadian highways is 100km/h [8]. Since $\gamma \approx 0$ in the free flow region it is best to fit this region first.

from $\rho = \frac{\rho(\%)}{d_0}$ and (31)

$$d_0 = \frac{v_{max}\rho(\%)}{\Phi}$$

$$= \frac{27.7(15\%)}{0.580}$$

$$= 7.18m$$

(33)

and

$$t_0 = \frac{1 - \rho(\%)}{(1 - \gamma)} \frac{\Phi}{\Phi}$$

$$= \frac{0.850}{0.580}$$

$$= 1.466s$$

(34)

letting $t_0=1s$ (say)

$$\gamma = 0.318$$

(35)

or

letting $\gamma = 0.5$ (say)

$$t_0 = 0.733$$

(36)

All of these variables are close to the expected values, and hence verify my model.

4.3 Two Roads Merging into One

When measuring the flux and density for the merging roads three sets of measurements are made, on both road A and B before they enter the merging zone and at the end of combined road. Due to the movement of vehicles from one road to another it is no longer appropriate to measure the density either side of the flux “measuring post”, measurements must now be made before the post as illustrated in fig 21.
Fig 22 shows road B merging into road A and coming to an end. It also shows that jammed traffic exists upstream of the junction. In general the position of the jam does not move but we can see small perturbations rippling back. The fundamental diagram for this merger is shown in fig 23, from it we can see that the merging acts like a valve on the road, allowing cars through at a constant rate independent of the global density. Increasing the density only adds to the tail-back before the jam and does not affect the local density and flux around the junction.

The most striking thing one can take from this plot is that vehicles after the junction travel with the ideal density with optimum flux, independent of the global density. These results were not quite what were expected and closer examination of the merging rules may be necessary along with comparison to empirical data.
Fig 23. Fundamental diagram for merged roads. The red and blue points are for cars on roads A and B respectfully before the merger, and the blue points are cars after the merger. Density is measured over the 20 cells before the merger / end or road and flux is averaged over 1000 time steps. Global density is adjusted from zero to 50%.

4.4 Numerical Efficiency

The CA model by its nature is very efficient numerically. To quantify this I chose the following benchmark settings; two road merging model for 1 million cars at 50% density for one hour simulation time (3,600 time steps). This includes three million road cells or 16,680km of road, measuring flux and density at every time step in each of the three sections and printing them out every 300 steps. The programme was compiled with gnu gcc c99 compiler set to level 3 optimisation.

<table>
<thead>
<tr>
<th>Computer</th>
<th>Run time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single 1GHz Pentium III</td>
<td>1hr 5min 23sec</td>
</tr>
<tr>
<td>Dual 3GHz Xeon (with hyperthreading and EM64T)</td>
<td>10min 19sec</td>
</tr>
</tbody>
</table>

The Dual Xeon computer is a reasonably standard high-end server by today’s standards and being able to do 24 hours of simulation for one million cars in approximately four hours computer time shows how viable it is to do large scale simulations for extended periods of time.
5 Conclusions

This report has briefly reviewed a number of available traffic models. Using the Nagel-Schreckenberg (NaSch) cellular automata model I have produced the fundamental diagram comparable to those published in [1,2,3] and shown how small perturbations in traffic can lead to phantom traffic jams. In parallel I have devised an analytical model with the assumption that all cars are evenly spaced along the road and with this I was able to apply SI units to the scaleless NaSch model. I fitted my analytical model to empirical data from a Canadian highway.

I have expanded the single road NaSch CA model into a simulation of two roads merging into one and encountered some interesting challenges to achieve this including challenges as to the rules to be used for boundary conditions and merging traffic. To resolve these I devised a set of rules for merging that only allowed a car to move over if there are two spaces free and used a semi-closed loop for the boundary conditions in which cars looped around to the road they originally came from. This gave good physical results but further work is required to make them more robust and flexible.

I also demonstrated the numerical efficiency of the program by completing a 1 hour simulation of 1 million cars in 10 minutes computer time. This shows the feasibility of modelling large road networks such as exist in cities.

6 Further Work

My supervisor, Mauro Ferreira, has suggested that this project, and in particular the analytical approach, would be suitable for development into a paper for a Physics teaching journal.

I would like to improve the boundary conditions to allow greater flexibility in the creation and removal of cars, possibly by using a “buffer” approach as suggested by [5]. This would allow the density to be increased and decreased during the simulation and hence observe phenomena such as a traffic jam lingering after the density has dropped below the critical point giving
hysteresis in the fundamental diagram. I would also like to better tune the merging rules so that cars integrate better such as incorporating some of the ideas from [6] which suggests using a “shadow car” to help traffic merge in a physical way.

I would also like to generalise the model to include different vehicle types, some of which would occupy a greater number of cell sites and have lower rate of acceleration and deceleration.

Due to its numerical efficiency the NaSch model could be used effectively in Trinity’s DSG Dublin City model.

7 References


Appendix A The source code

// 2 Roads Merging NaSch CA model
// By David Clarke ( dave@maths.tcd.ie )
// Original source available at www.maths.tcd.ie/~dave/traffic/
//Compile with cc -o traffic traffic.c

#include<stdio.h>
#include<stdlib.h>
#include<math.h>

#define leadin_d 500
#define overlap_d 10
#define leadout_d 500
#define max_length 2100  // greater then leadin+overlap+leadout
#define speed_limit1_d 5
#define speed_limit2_d 5
#define steps_d 20000  //Number of Steps updates
#define cars1_d 0  // cars to start with (0 is empty road)
#define cars2_d 0
#define cars0_d 0  // leadout cars
#define prob_d 0.5  // probability of stopping
#define slow_start_d 0.0  // steps ignore before measuring
#define relax_d 600  // steps flux measured over
#define flux_time_d 2000
#define print_d 0
#define split 10000000
#define once 0  // if 1, run for only one density, for scatter plot.
#define max_density 1  // limiter 0->1, can stop program for a lower density

typedef struct
{    int car;
    int vel;
    int moved;
    int pre_vel;  //previous velocity, used in slow to start
    int jtime;
    int origin;
    int num;
} road_t;
FILE *out1;
FILE *outScatter1, *outScatter2;
FILE *out_jtime1, *out_jtime2;

//Function definitions
void print_road (int);
void print_jams ();
void print_flow_density();
void print_scatter (int c, int tstep);
void setup (int);
void setup_leadout ();
void setspeed (int);
void update (int c, int tstep);
void measure_density();
void variables (int argc, char *argv[]);

//global variables
road_t Road1[max_length],Road2[max_length];
int leadin,overlap,leadout, speed_limit[3], steps, cars[3], relax, density_width, flux_time,tot_length,print;
double prob, slow_start;
int count[3], density_start[3], density_stop[3], time_count, density[3], jtimeMax[3], jtimeMin[3];

int main(int argc, char *argv[])
{
    /* Start up */
    int i, j, n, d, p;
    road_t *roadP;
    variables(argc, argv); //take in variables from command line

tot_length=leadin+overlap+leadout;
//density measurement parameters.
density_start[0]=leadin+overlap+leadout/2;
density_stop[0]=leadin+overlap+leadout;
out1 = fopen("density_flow.dat", "w");
outScatter1 = fopen("trafficYScatter1.dat", "w");
outScatter2 = fopen("trafficYScatter2.dat", "w");
out_jtime1 = fopen("jtime1.dat", "w");
out_jtime2 = fopen("jtime2.dat", "w");
srand(time(0)); // random seed

while(cars[1]<leadin*max_density && cars[2]<leadin*max_density){ // Looping for increasing density
    /* Set Up */
    n = 0;
    setup(1); // initial positioning of cars
    setup(2); // initial positioning of cars
    setup_leadout();
    if(print){
        // initial print
        print_road(1);
        print_road(2);
    }
    /* Relax */
    while(n < relax) {n++; //allow system to settle
        setspeed(1);
        update(1, -1);
        setspeed(2);
        update(2, -1);
    }
    // Give each car a number:
    for(i=0; i<leadin+overlap+leadout; i++){
        if(Road1[i].car==1){Road1[i].num=count[Road1[i].origin];
            count[Road1[i].origin]++;
        }
        if(Road2[i].car==1){Road2[i].num=count[Road2[i].origin];
            count[Road2[i].origin]++;
        }
    }

time_count = flux_time;

    /* Run Model */
    while (n < steps) {
        n++;
        time_count--; // keep track of time
        setspeed(1);
        update(1, n);
        setspeed(2);
        update(2, n);
        if(print){printf("r1"); print_road(1);}
    }
}
printf("r2"); print_road(2);

if (once){
    print_scatter(1,n); // Print Scatter diagram
    print_scatter(2,n);
}

measure_density();
if (time_count == 0) print_flow_density();

//    print_jams ();

fclose(out1);
fclose(outScatter1);
fclose(outScatter2);
fclose(out_jtime1);
fclose(out_jtime2);
fprintf(stderr,"\nprogram complete\n");
exit(0);
while (cars_used > 0) {
    for (i = 0; i < leadout; i++) {
        if (((int) (0 + (1.0*leadout/cars_used)))*(double)rand() /
            (RAND_MAX + 1.0)) == 1
            && cars_used > 0 && (roadP+i)->car == 0) {
            (roadP+i)->car = 1;
            (roadP+i)->vel = (int)(0+(speed_limit[1]+1)*
                (double)rand()/(RAND_MAX + 1.0));
        }
        cars_used--;
        (roadP+i)->time=-split;
        if(cars[1]==0)
            (roadP+i)->origin=1;
        else if(cars[2]==0)
            (roadP+i)->origin=0;
        else if((int)((1+(1.0*cars[2])/cars[1])*
            (double)rand()/(RAND_MAX + 1.0))==0)
            (roadP+i)->origin=1;
        else
            (roadP+i)->origin=2;
    }
}

void
setspeed (int c){
    int i, j, dist, k, d, p,length;
    road_t *roadP, *roadO;
    switch(c){
        case 1: length=leadin+overlap+leadout;
            roadP=&Road1[0];
            break;
        case 2: length=leadin+overlap;
            roadP=&Road2[0];
            break;
        default: fprintf(stderr,"Error! Unknown case in setspeed\n");
    }
    for (i = 0; i < length; i++) {
        if ((roadP+i)->car == 1) {
            (roadP+i)->pre_vel = (roadP+i)->vel;
            //1. accelerate
            if ((roadP+i)->vel < speed_limit[c])((roadP+i)->vel++;
            dist = 0;
            // Find distance to next car (up to speed limit)
            for(j=1;j<=speed_limit[c];j++){
                if (i+j)>=length) {
                    if(c==1){ //Main branch -off the edge
                        // Find average
                        if((roadP+i)->origin==0)roadO=&Road1[0];
                        else roadO=&Road2[0];
                        if((roadO+i+j)->length)car==0) dist++;
                        else { break;}
                    }
                }
            }
        }
    }
}
void update (int c, int tstep) {
    int i, n = 0, d, length, v, k, m;
    road_t *roadP, *roadO;
    switch (c) {
        case 1: length = leadin + overlap + leadout;
            roadP = &Road1[0];
            break;
        case 2: length = leadin + overlap;
            roadP = &Road2[0];
            break;
        default: fprintf(stderr, "Error! Unknown case in setspeed\n");
    }
    roadO = roadP;
    for (i = 0; i < length; i++) roadP[i]->moved = 0; // Don't move cars twice

    for (i = length - 1; i >= 0; i--) { // move backwards because of merging lane
        if ((roadP[i]->car == 0 && (roadP[i]->moved == 0)) { // if car & not moved
            k = m = 0;

            /* ### Merging ### */
            /* ### Rules ### */

            /* Move over at the earliest opportunity, then move up lane as much as possible ie. min(vel,dist to next car) */

            if (c == 2 && i >= leadin) { // in merging zone
                v = Road2[i].vel;
                for (k = 0; k <= v; k++) { // what k can we move over at?
                    if (Road1[i+k].car == 0 && Road1[i+k].car == 0) {
                        m++; // succeeded in moving over (and 1 space to spare)
                        d = k; // max advancement
                        if (m > 0 && Road1[i+k].car == 1) break; // moved over and behind next car
                    }
                }
                if (m == 0) { // couldn't move, move in lane at reduced speed
                    if (v > 1) {
                        v--; //
                        if (Road2[i+v].car == 1) fprintf(stderr, "Crash while not merging!!\n");
                        Road2[i+v].car = 1;
                        Road2[i].car = 0;
                        Road2[i+v].vel = v;
                        Road2[i+v].jtime = Road2[i].jtime + 1;
                        Road2[i+v].origin = 2;
                        Road2[i+v].num = Road2[i].num;
                    }
                }
            }
        }
    }
}
else
  if(Road1[i+d].car==1)fprintf(stderr,"Crash while merging!!");
  Road1[i+d].car=1;
  Road1[i+d].vel=v;
  Road1[i+d].jtime=Road2[i].jtime+1;
  Road1[i+d].num=Road2[i].num;
  Road1[i+d].origin=2;
  Road2[i].car=0;
}

} /* ### Other motion ### */
else
  if((i+=length)){ // car loops.
    count[0]++;
    d=(roadP+i)->vel+i=length;//car now loops around
    if((roadP+i)->origin==2){
      roadO=&Road2[0];

      /* calculate its journey time */
      // if time <0 car came from middle,ignore
      if(((roadP+i)->jtime>=0&&tstep!=--1){
        if((roadP+i)->origin==1)
          fprintf(out_jtime1,"1\t%d\t%d\n",tstep,(roadP+i)->jtime);
        else fprintf(out_jtime2,"2\t%d\t%d\n",tstep,(roadP+i)->jtime);
      }
      if((roadO+d)->car==1)fprintf(stderr,"\n ** ### Car crash! ### **\n");
      (roadO+d)->car = 1;
      (roadO+d)->vel = (roadP+i)->vel;
      (roadO+d)->jtime = 0;
      (roadO+d)->num = (roadP+i)->num;
      (roadO+d)->origin = (roadP+i)->origin;
      (roadO+d)->moved = 1;
      (roadP+i)->car = 0;
    }
  }

} // 4. Update & Move
else {
  d = (roadP+i)->vel + i;

  if((roadP+d)->car==1)fprintf(stderr,"\n ** ### Car crash! ### **\n");
  (roadP+d)->car = 1;
  (roadP+d)->vel = (roadP+i)->vel;
  (roadP+d)->jtime = (roadP+i)->jtime+1;
  (roadP+d)->num = (roadP+i)->num;
  (roadP+d)->origin = (roadP+i)->origin;
  (roadP+d)->moved = 1;
  (roadP+i)->car = 0;
  if(i<leadin&&d==leadin){count[c]++;} //car has left the zone

}

} //end of update()

/* ### Other motion ### */
else
  if((roadP+i)->vel == 0)(roadP+i)->jtime++; // car doesn't move
else if(i+(+(roadP+i)->vel>=length){ // car loops.
  count[0]++;
  d=(roadP+i)->vel+i=length;//car now loops around
  if((roadP+i)->origin==2){
    roadO=&Road2[0];

    /* calculate its journey time */
    // if time <0 car came from middle,ignore
    if(((roadP+i)->jtime>=0&&tstep!=--1){
      if((roadP+i)->origin==1)
        fprintf(out_jtime1,"1\t%d\t%d\n",tstep,(roadP+i)->jtime);
      else fprintf(out_jtime2,"2\t%d\t%d\n",tstep,(roadP+i)->jtime);
    }
    if((roadO+d)->car==1)fprintf(stderr,"\n ** ### Car crash! ### **\n");
    (roadO+d)->car = 1;
    (roadO+d)->vel = (roadP+i)->vel;
    (roadO+d)->jtime = 0;
    (roadO+d)->num = (roadP+i)->num;
    (roadO+d)->origin = (roadP+i)->origin;
    (roadO+d)->moved = 1;
    (roadP+i)->car = 0;
  }

} // 4. Update & Move
else {
  d = (roadP+i)->vel + i;

  if((roadP+d)->car==1)fprintf(stderr,"\n ** ### Car crash! ### **\n");
  (roadP+d)->car = 1;
  (roadP+d)->vel = (roadP+i)->vel;
  (roadP+d)->jtime = (roadP+i)->jtime+1;
  (roadP+d)->num = (roadP+i)->num;
  (roadP+d)->origin = (roadP+i)->origin;
  (roadP+d)->moved = 1;
  (roadP+i)->car = 0;
  if(i<leadin&&d==leadin){count[c]++;} //car has left the zone

}

} //end of update()
case 0: roadP=&Road1[0]; break;
case 1: roadP=&Road1[0]; break;
case 2: roadP=&Road2[0]; break;
default: fprintf(stderr,"Error measuring density");}
for (i=density_start[j];i<density_stop[j];i++){
  if (((roadP+i)->car == 1) density[j]++;
}
if(print) printf("D%d:%d ",j,density[j]);
}
void print_road (int c){
  int i,length;
  road_t *roadP;
  switch(c){
    case 1: roadP=&Road1[0];
      length=leadin+overlap+leadout;
      break;
    case 2: roadP=&Road2[0];
      length=leadin+overlap;
      break;
    default: fprintf(stderr,"Error! Unknown case in print road
    ");
  }
  for (i = 0; i < tot_length; i++) {
    if((i==0) || (i==leadin) || (i==leadin+overlap) || (i==leadin+overlap+leadout)
      printf("|");
    if (((roadP+i)->car == 0) { printf("-");
    } else {
      printf("%d", (roadP+i)->vel); // print velocities
      printf("%d", (roadP+i)->origin); // print origin
      printf("%d", (roadP+i)->num%10); //print car number mod 10
    }
  }
  printf ("\n");
}
void print_scatter(int c,int tstep){
  int i,length;
  road_t *roadP;
  FILE *Scatter;
  switch(c){
    case 1: roadP=&Road1[0];
      length=leadin+overlap+leadout;
      Scatter=outScatter1;
      break;
    case 2: roadP=&Road2[0];
      length=leadin+overlap;
      Scatter=outScatter2;
      break;
    default: fprintf(stderr,"Error! Unknown case in print scatter
    ");
  }
  for (i = 0; i < length; i++) {
    if (((roadP+i)->car == 1) && ((roadP+i)->num%10 == 0) {
      if(c==2)fprintf(Scatter,"%d\t%d\n",tstep,i);
      /* Separate for plotting different colours based on origin */
      else if(Road1[i].origin==1)fprintf(Scatter,"%d\t%d\n",tstep,i);
      else if(Road1[i].origin==2)fprintf(Scatter,"%d\t%d\n",tstep,i);
```c
void print_flow_density(){
    int j;
    for(j=0; j<3; j++){
        fprintf(out1,"%lf	%lf	%lf
",density[j]*1.0/(flux_time*(density_stop[j]-density_start[j])),(count[j]*1.0)/flux_time);
        time_count = flux_time;
        count[j] = 0;
        density[j] = 0;
    }
    fprintf(out1,"
");
}

//Take in variables from command line
void variables (int argc, char *argv[]){
    if (argc == 1) {
        leadin = leadin_d;
        overlap = overlap_d;
        leadout = leadout_d;
        speed_limit[1] = speed_limit1_d;
        steps = steps_d;
        cars[1] = cars1_d;
        cars[2] = cars2_d;
        cars[0] = cars0_d;
        relax = relax_d;
        flux_time = flux_time_d;
        prob = prob_d;
        slow_start = slow_start_d;
        leadin = leadin_d;
        print = print_d;
    } else if(argc == 14) {
        fprintf(stderr,"Usage: %s <leadin> <overlap> <leadout> <cars1> <cars2> <cars leadout> <speed1> <speed2> <steps> <relax> <prob> <fluxTime> <print>\n\n",argc[0],leadin_d,overlap_d,leadout_d,cars1_d,cars2_d,cars0_d,relax_d,flux_time_d,print_d);
        exit(1);
    } else {
        if (sscanf(argv[1], "%d", &leadin) != 1) {
            fprintf(stderr,\n\narg1: road length not an int\n\n");
            exit(1);
        }
        if (sscanf(argv[2], "%d", &overlap) != 1) {
            fprintf(stderr,\n\narg2: road length not an int\n\n");
            exit(1);
        }
        if (sscanf(argv[3], "%d", &leadout) != 1) {
            fprintf(stderr,\n\narg3: road length not an int\n\n");
            exit(1);
        }
        if(leadin+overlap+leadout>max_length){
            fprintf(stderr,\nTotal length greater then max length\n\n");
            exit(1);
        }
        if (sscanf(argv[4], "%d", &cars[1]) != 1) {
            fprintf(stderr,\narg4:Cars1 not an int\n\n");
    }
```

exit (1); }
if (sscanf (argv[5], "%d", &cars[2]) != 1) {
  fprintf(stderr,"\narg5:Cars2 not an int\n\n");
  exit (1); }
if (sscanf (argv[6], "%d", &cars[0]) != 1) {
  fprintf(stderr,"\narg5:Cars leadout not an int\n\n");
  exit (1); }
if (sscanf (argv[7], "%d", &speed_limit[1]) != 1) {
  fprintf(stderr,"\narg7:Speed Limit 1 not an int\n\n");
  exit (1); }
if (sscanf (argv[8], "%d", &speed_limit[2]) != 1) {
  fprintf(stderr,"\narg8:Speed Limit 2 not an int\n\n");
  exit (1); }
if (sscanf (argv[9], "%d", &steps) != 1) {
  fprintf(stderr,"\narg9: Steps not an int\n\n");
  exit (1); }
if (sscanf (argv[10], "%d", &relax) != 1) {
  fprintf(stderr,"\narg10: relax not an int\n\n");
  exit (1); }
if (sscanf (argv[11], "%lf", &prob) != 1 || prob > 1) {
  fprintf(stderr,"\narg11:Slowing prob not less then 1\n\n");
  exit (1); }
  fprintf(stderr,"\nToo many cars for road length\n\n");
  exit(1); }
if (sscanf (argv[12], "%d", &flux_time) != 1) {
  fprintf(stderr,"\narg12:Flux time not an int\n\n");
  exit (1); }
if (sscanf (argv[13], "%d", &print) != 1 && (print==0||print==1)) {
  fprintf(stderr,"\narg13:Print is not 0 or 1\n\n");
  exit (1); }
}