Chapter 2
Introduction to Cellular Automata in Simulation

Val Lowndes, Adrian Bird and Stuart Berry

Cellular Automata and Agents can be used to construct simulations where the movement/change of status in an individual is to be observed. This section introduces both Cellular Automata and Agent-based simulations.

Cellular automata and Agent-based simulations have been used to study:

- Traffic flow investigating the formation of traffic jams, the cellular automata representing vehicles (initially a one-dimensional model). N-S
- Conway's Game of Life. (often two-dimensional models) G-M
- The evolution of epidemics, the cellular automata representing people (often one-dimensional models)
- Bird/fish flocking, the cellular automata representing a single bird/fish
- Evacuating buildings, the cellular automata representing people.

Section 2.2 introduces Conway's Game of Life.

Section 2.3 investigates population changes.

Section 2.4 introduces models for Epidemics and fire evacuations and traffic simulations based around the Nagel and Schreckenberg model for traffic flow.

V. Lowndes
University of Derby, Kedleston Road, Derby DE22 1GB, UK
e-mail: V.P.Lowndes@derby.ac.uk

A. Bird · S. Berry (✉)
College of Engineering and Technology, University of Derby,
Kedleston Road, Derby DE22 1GB, UK
e-mail: s.berry@derby.ac.uk

S. Berry et al. (eds.), *Guide to Computational Modelling for Decision Processes*,
Simulation Foundations, Methods and Applications,
DOI 10.1007/978-3-319-55417-4_2

2.1 Defining the Operation of a Cellular Automata-Based Simulation

A cellular automaton (CA) is a collection of cells arranged in a grid, such that each cell changes state as a function of time according to a defined set of rules that includes the states of neighbouring cells.

For example, applying rules to the system status at time zero produces the system status at time one as shown in Fig. 2.1.

The objective in these simulations is the determination of the behaviour of the system over a period in time. In particular, what is the final or steady state of the system, for example does it achieve a steady state.

The simplest cellular automata are one-dimensional, with cells on a straight line, where each cell can have only two possible states (such as high/low or black/white). But in theory, a CA can have any number of dimensions, and each cell can have any number of possible states. The state of each cell changes in discrete steps at regular time intervals.

The state of a cell at any given time depends on two things:

1. its own state in the previous time step, and
2. the states of its immediate neighbours in the previous time step.

2.2 Conway's Game of Life

This section uses "Conway's Game of Life" to introduce the use of a two-dimensional grid. This game became widely known in 1970 when it was published in the popular journal *Scientific American*. See Gotts, in Griffeath and Moore [1] for background information.

Conway devised the rules for the Game of Life (GOL), aiming to meet the following three criteria:

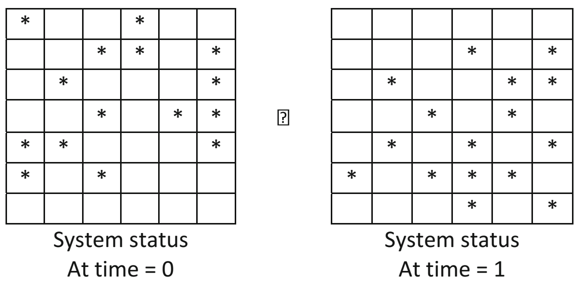


Fig. 2.1 Sample system transition

Criteria for the Game of Life:

1. There should be no initial pattern for which there is a simple proof that the population can grow without limit.
2. There should be initial patterns that *apparently* do grow without limit.
3. There should be simple initial patterns that grow and change for a considerable period of time before coming to an end in three possible ways:
   - Fading away completely (from overcrowding or becoming too sparse)
   - Settling into a stable configuration that remains unchanged thereafter
   - Entering an oscillating phase in which they repeat an endless cycle of two or more periods.

These criteria aimed to ensure that the rules cause the behaviour of the population to be unpredictable. It was, however, later shown by Charles Corderman that Conway's rules did not guarantee that the criteria were satisfied [1, p. 10].

The game itself consists of an infinitely large 2-dimensional grid of squares, in which each represent a cell. The status of each cell at time $t + 1$ depends upon the status of the system of cells at time t. In Fig. 2.2, the status of cell D5 is derived from its own status and that of its 8 surrounding cells at time t:

A cell will be denoted as ***alive*** if it contains a1 (or a*)

The GOL uses rules to generate the system status at time $t + 1$ based on the information (the system status) at time t.

***Rules for Conway's Game of Life*:**
When a cell is alive at time t:

If **two or three** of its eight **neighbours are alive**, **it remains alive**.
If **more than three** of its eight **neighbours are alive**, **it dies** from overcrowding.
If **less than two** of its eight **neighbours are alive**, **it dies** from loneliness.

When a cell is dead:

If **exactly three** of its eight **neighbours are alive**, **it comes to life**.
In **all other cases**, the **cell remains dead**.

Example 1 The rules are applied to the data given in Fig. 2.3 (status at time 0), to generate the state of the system at time 1 (see Fig. 2.4).

Applying the rules to the data shown in Fig. 2.3 gives:

C4	*C5*	*C6*
D4	*D5*	*D6*
C4	*C5*	*C6*

→

	D5	

Fig. 2.2 Cells relevant to status of cell D5

Fig. 2.3 Status at time 0

	1	2	3	4	5	6	7	8
A								
B		*	*					
C			*					
D								
E			*					
F			*					
G								
H								

Fig. 2.4 Status at time 1

	1	2	3	4	5	6	7	8
A								
B		*	*					
C		*	*					
D								
E								
F								
G								
H								

Applied to *live cells*:

Cells B2, B3 and C3 surrounded by 2 "*live*" cells, this stays *alive*
Cell E3 and F3 surrounded by only one "*live*" cell, this *dies* (now empty)

Applied to *not alive cells*

Cell C2 surrounded by 3 "*live*" cells, this becomes *alive*

Combining these results gives Fig. 2.4: the status at time 1.

Applying the rules again to the data displayed in Fig. 2.4 give the status at time 2 as shown in Fig. 2.5.

Example 2 Consider the starting pattern shown in Fig. 2.6. At each iteration, count the number of non empty cells. A plot of the number of nonzero (live) points at each iteration is given in Fig. 2.7.

Plotting the number of nonzero (live) points at each iteration gives the graph:

Figure 2.7 shows that a steady state is developing after 36 iterations: approximately 15 live cells.

When a GOL simulation is performed, the rules are applied until the population has become 'stagnant', which means that it will have adopted one of the following states:

Fig. 2.5 Status at time 2

	1	2	3	4	5	6	7	8
A								
B		*	*					
C		*	*					
D								
E								
F								
G								
H								

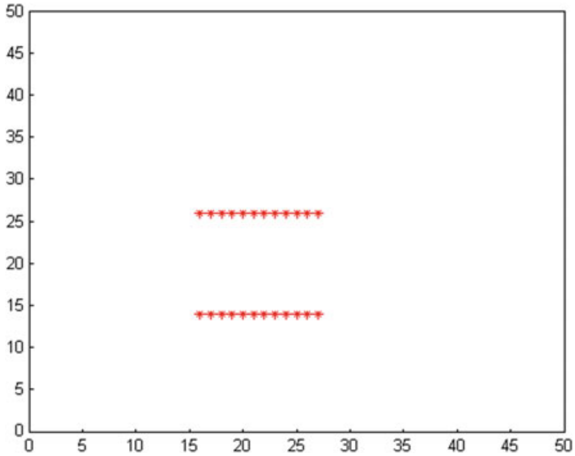


Fig. 2.6 Two parallel lines

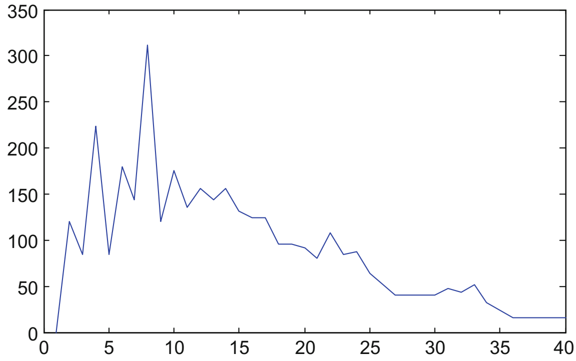


Fig. 2.7 Plot of number of live cells

- There are no more living cells—complete annihilation.
- Cells form a grouping which exhibits no change from one generation to the next —still life.
- Cells form a grouping which mutate through a given number of generations resulting in a repetitive cycle.
- Cells traverse the 'universe' in a repetitive but translated (in position) manner and will never collide with any other cells—Gliders and spaceships.

2.3 Investigating Population Growth and Decay

The initial pattern was randomly generated with the number of live cells selected representing an *average* 15% loading.

Figure 2.8 shows one initial pattern generated with random 15% loading. By carrying out several simulations for a particular loading, the most likely state after a given number of iterations and the terminal state can be identified.

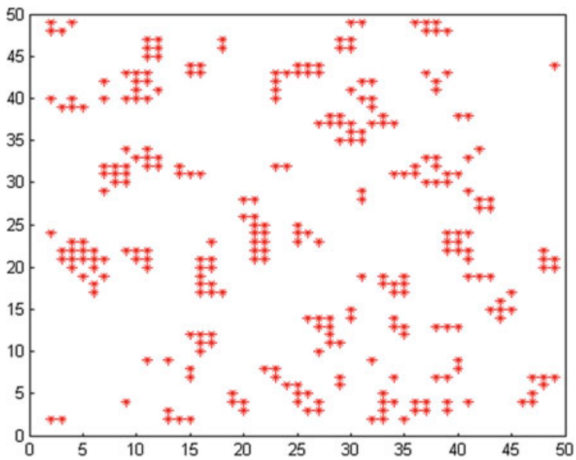
Figure 2.9 shows the number of live cells at each iteration, for the simulation generated by the initial pattern shown in Fig. 2.8.

This simulation was run for 500 cycles and Fig. 2.9 shows that for this particular starting pattern, the system converged to a steady state at approximately 470 cycles.

The pattern of live cells (system after 500 iterations) is as shown in Fig. 2.10.

This consists of both static and cyclical shapes. Remember, orientation is not fixed, so shape X is the same as shape Y.

Fig. 2.8 Initial state 15% loading



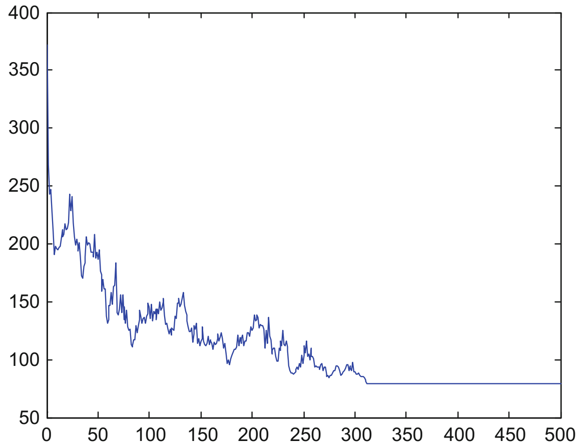


Fig. 2.9 Plot of number of live cells

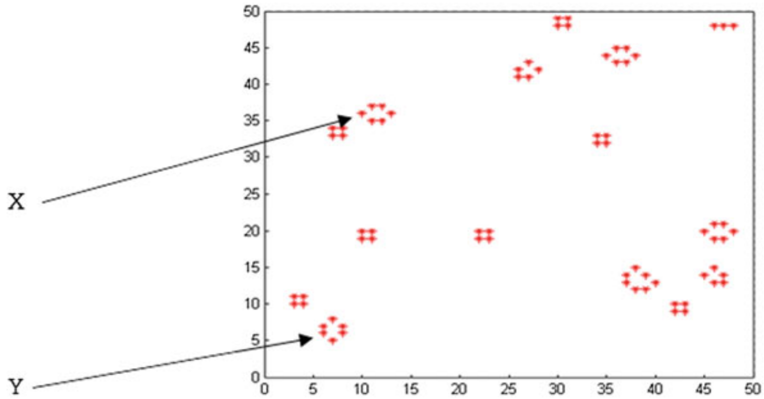


Fig. 2.10 Finishing state, stable shapes

2.4 Applying Cellular Automata and Agents in Modelling

2.4.1 Agent-Based Modelling: Modelling the Spread of Infections

The population is represented on an $n \times n$ where a healthy person is indicted by a cell value of 0 and an infected person by a cell value of 1. Each individual (agent) has an attached record of current duration of their infection.

For example, in Fig. 2.11

Fig. 2.11 Tables A and S showing infected cells and current duration of infection

0	0	0	0	0	0
0	1	0	0	0	0
0	0	0	0	1	0
0	0	1	0	0	0
0	0	0	0	0	0

0	0	0	0	0	0
0	3	0	0	0	0
0	0	0	0	2	0
0	0	5	0	0	0
0	0	0	0	0	0

$\mathrm{A}\,(4,3) = 1$ indicating the cell occupant is infected
$\mathrm{S}\,(4,3) = 5$ indicating that the infection has lasted 5 time periods

Probability of infection
This is dependent upon the status of the surrounding cells and the variables

$$\mathrm{cv}(i,j) = a(i,j+1) + a(i,j-1) + a(i+1,j) + a(i-1,j)$$
$$\mathrm{Tr} = \text{transmission rate}$$
$$P(i,j) = f_I(\mathrm{cv}, \mathrm{Tr})$$

Probability of recovery
This is dependent on the status of the surrounding cells and the duration of the illness

$$\mathrm{cv}(i,j) = a(i,j+1) + a(i,j-1) + a(i+1,j) + a(i-1,j)$$
$$\mathrm{Rr} = \text{Recovery rate}$$
$$P(i,j) = f_R(\mathrm{cv}, \mathrm{Rr})$$

Figure 2.12 shows the normalised spread, number contracting the illness and the duration of the illness, with reducing levels of infection, for a population of 6400.

The worst case, no treatment and highest rate of infection lead to:

Peak number of patients	2460 (normalised to have value 1)
Percentage of population	38.4%
Maximum duration of epidemic	615 time units

The effect of improved levels of treatment for each level of infection is shown in Fig. 2.13 where the treatment has improved from that used to create Fig. 2.12.

Peak number of patients	2052 (normalised value 2460 as above)
Percentage of population	32.0%
Maximum duration of epidemic	555 time units

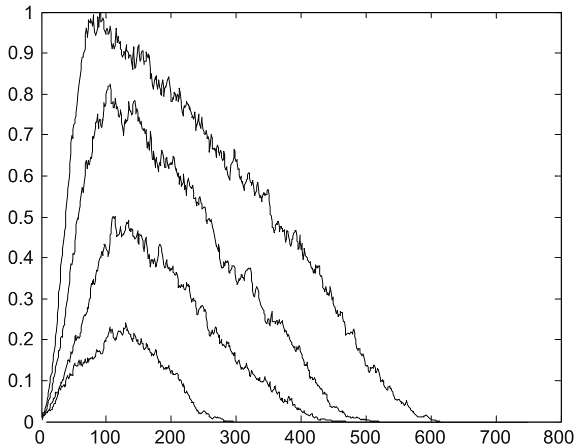


Fig. 2.12 Modelling the effect treatment on infection spread

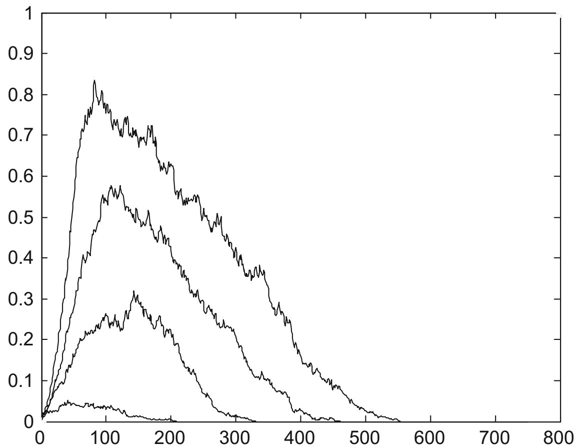


Fig. 2.13 Modelling the effect improvement in treatment

In each case both the duration of the epidemic and the maximum number infected has been reduced.

This agent-based model demonstrates the effect of both preventative and remedial treatment regimes in restricting the spread of an infectious disease.

2.4.2 Traffic Flow

This application is used to introduce the use of cellular automata because it employs a simple one-dimensional model, representing a single carriageway road (Note: this is assuming a model where initially cars are not allowed to overtake).

This section shows that although this is the simplest cellular automata-based model, it is a model where the results can have an immediate consequence, to the understanding of traffic flow and hence to the derivation of traffic management policies. The models presented here use the same principles as those in Nagel and Schreckenberg but the analysis of the results is by way of the average velocity.

2.4.2.1 Notation

v_{ij} the velocity of a car j at time i
$v_{i+1,j}$ the velocity of a car j at time $i + 1$
$d_{i,j}$ the distance to the next car $j + 1$, bumper to bumper, at time i
$v_{\max}$ the maximum attainable velocity (the speed limit)
x_{ij} the position of car j at time i.

2.4.2.2 Basic Model

In this model, each cell represents a section of road and each cell will either contain a car or be empty. The system is updated at regular time intervals, using a set of rules, the effect of the update being to allow the cars to move along the road.

In Fig. 2.14, the coloured markers show the progress of the cars with time.

Notice that

- the "blue" car travels a total distance of 4 units in two time steps (average velocity 2)
- the "red" car travels a total distance of 2 units (average velocity of 1).

so a traffic jam is forming at time 2.

The rules to enable this simulation (passage of time) are defined in Nagel and Schreckenberg and are given by the four steps **N-S Rules**:

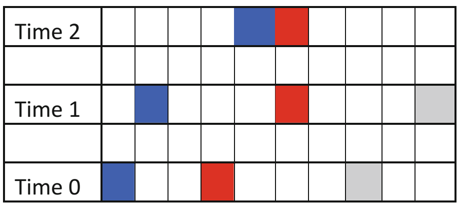


Fig. 2.14 Sample car movements

Step 1: acceleration All cars that have not already reached the maximal velocity v_{max} accelerate by one unit: $v_{i+1} \Rightarrow v_i + 1$
Step 2: safety distance If a car has d empty cells in front of it and if its velocity v_{i+1}(after step 1) is larger than d, then the velocity reduces to d: $v_{i+1} \Rightarrow \min\{d, v_{i+1}\}$
Step 3: randomisation With probability p, the velocity is reduced by one unit (if v_{i+1} after step 2): $v_{i+1} \Rightarrow v_{i+1} - 1$
Step 4: driving After steps 1–3, the new velocity v_{i+1} for each car j has been determined. Car j moves forward by $v_{\ i+1}$ cells: $x_{i+1,j} \Rightarrow x_{i,j} + v_{i+1}$.

Example 3: Illustrating the operation of the N-S Rules The section of road shown in Fig. 2.15a contains 4 cars with the given velocities, at Time = 0. Work through each of the steps (work through the four steps for each of the 4 cars, checking your velocity calculations against each of the Fig. 2.15b and e. Figure 2.15e gives the final state at Time = 1 after completion of the N-S Rules. Notice that the road is in effect a loop, with car 4 in Fig. 2.15a having one empty cell between itself and car 1.

Step 1: For car 1: $v_1 := v_0 + 1 = 3$; $v_{\max} = 5$
Step 2: For car 1: $v_1 := \min\ (d_0,\ v_1) = \min(2,\ 3) = 2$
Step 3: Random reduction: if rand < p then $v_1 := v_{1-1}$
Step 4: *Drive forward*: For car 1: $x_{1,1} \Rightarrow x_{0,1} + v_{1,1} = 1 + 2 = 3$

2.4.2.3 Simulation of Traffic Flow Using MATLAB

The following analysis was completed using a set of MATLAB programs. Each MATLAB program required the following information to be defined:

- the road length
- traffic density, as a percentage,
- the probability of a car slowing.

At each time step, the simulation records:

- the status of each cell in array a,
  - $a(i) = 1 \Rightarrow$ presence of a car,
  - $a(i) = 0 \Rightarrow$ empty cell, no car

(a)

Road position Time t=0	Car 1			Car 2						Car 3		Car 4	
v_{max} = 5, p = 0.5													
v_0	2			4						3		1	
d_0	2			5						1		1	

(b) ***Step 1:*** For car 1: $v_1 := v_0 + 1 = 3$; $v_{max} = 5$

Road position at time t = 0	Car 1			Car 2						Car 3		Car 4	
v_1	3			5						4		2	
d_0	2			5						1		1	

(c) ***Step 2:*** For car 1: $v_1 :=$ $min\ (d_0, v_1)$ = min(2, 3) = 2

Road position at time t = 0	Car 1			Car 2						Car 3		Car 4	
Velocity V_1	2			5						1		1	
d_0	2			5						1		1	

(d) ***Step 3:*** Random reduction: if rand<p then $v_1 := v_1 - 1$

Road position at time t = 0	Car 1			Car 2						Car 3		Car 4	
P=0.5: rand	0.83			0.62						0.31		0 .54	
Velocity v_1	2			5						0		1	
d_0	2			5						1		1	

(e) ***Step 4:*** *Drive forward*: For car 1: $x_{1,1} \Rightarrow x_{0,1} + v_{1,1} = 1 + 2 = 3$

Road position at time t= 1			Car 1						Car 2	Car 3			Car 4
Velocity at time 1 v_1			2						5	0			1
d_1			5						0	2			2

Fig. 2.15 **a** Sample data at time = 0. **b** Sample new velocities at time = 0, after step 1. **c** Sample new velocities at time = 0, after step 2. **d** Sample new velocities at time = 0, after step 3. **e** Final velocities and positions for sample at time t = 1

- the status of each car in array *vl*
  - $v(i)$ gives the current velocity of the car in cell i; $v_i \in (0, 5)$.

Figures 2.16 and 2.17 are typical graphical outputs from this program. For cellular automata, conventionally the vertical axis represents time units and the horizontal axis distance units. The results displayed in these graphs were generated using p = 0.5 (probability of a car randomly slowing) with 10% cell loading (traffic density) (Fig. 2.16) and then with 20% cell loadings (Fig. 2.17).

Notice

(a) In this and the following figures that the vertical axis measures time and the horizontal axis distance.

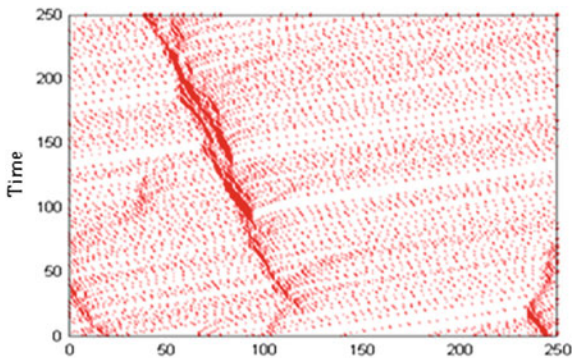


Fig. 2.16 10% cell loading, free-forming traffic jam

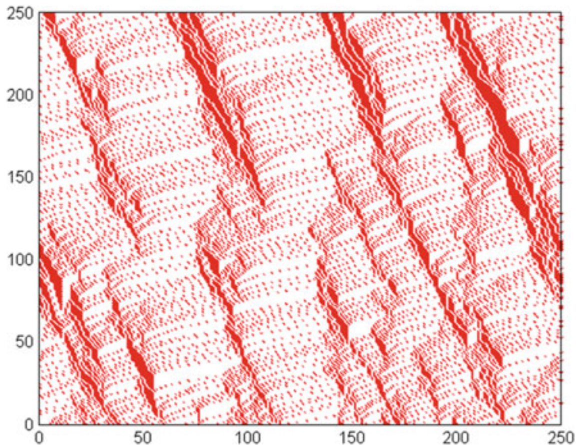


Fig. 2.17 20% cell loading. Higher density implies more traffic jams

(b) The deeper shading indicates the presence of a jam.
(c) A free-forming jam "moves" backwards in time.

Notice that the jams, deeper shaded areas" are parallel.

Extending the "road" length in the second simulation replaces Fig. 2.17 with Fig. 2.18.

The plot shown in Fig. 2.18 suggests the existence of many regular "self-forming" queues of traffic in this model and a congested traffic flow.

The program uses two parameters to define the simulation (CA model) to generate this plot

Density of traffic = 0.20	20% of cells filled	Traffic demand
Random slowing = 0.50	50% of cars slow down	Driver behaviour

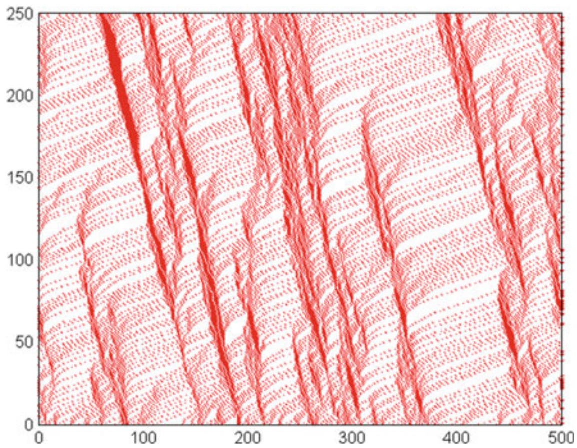


Fig. 2.18 20% cell loading extended road length

2.4.2.4 Analysis of Simulation Results

The traffic simulation program was run several times varying the traffic density, but keeping p (the probability of a random reduction in speed), constant (at 10%). The results generated have been used to produce the plot of average car velocity v traffic density in Fig. 2.19a.

Within the results shown in Fig. 2.19a, there seems to be two phases, traffic density (road loading x) up to 12% and over 12% giving the models for travel time T as:

Road loading less than 12%	travel time	$T = 4.9027 - 0.0044x$	$r^2 = 0.94$
Road loading more than 12%	travel time	$T = 8.2325e^{-0.046x}$	$r^2 = 0.99$

for the results shown in Fig. 2.19b, these models become:

Road loading (x) less than 5%	travel time	$T = 4.4048 - 0.0119x$	$r^2 = 0.92$
Road loading more than 5%	travel time	$T = 68.208x^{-1.427}$	$r^2 = 0.97$

and for the results shown in Fig. 2.19c, these models become:

Road loading (x) less than 15%	travel time	$T = 4.9979 - 0.0008x$	$r^2 = 0.84$
Road loading more than 15%	travel time	$T = 11.379e^{-0.05x}$	$r^2 = 0.98$

Figure 2.19d displays all three models for comparison showing that at normal loadings, up to 40%, random slowing has a great effect on a roads carrying capacity. Thus, indicating that at times of high traffic density, the “random” behaviour of drivers becomes less important, with respect to travel time.

Note 1: when there is no random slowing and all cars are equally spaced, the implied road loading will be 16.7%.

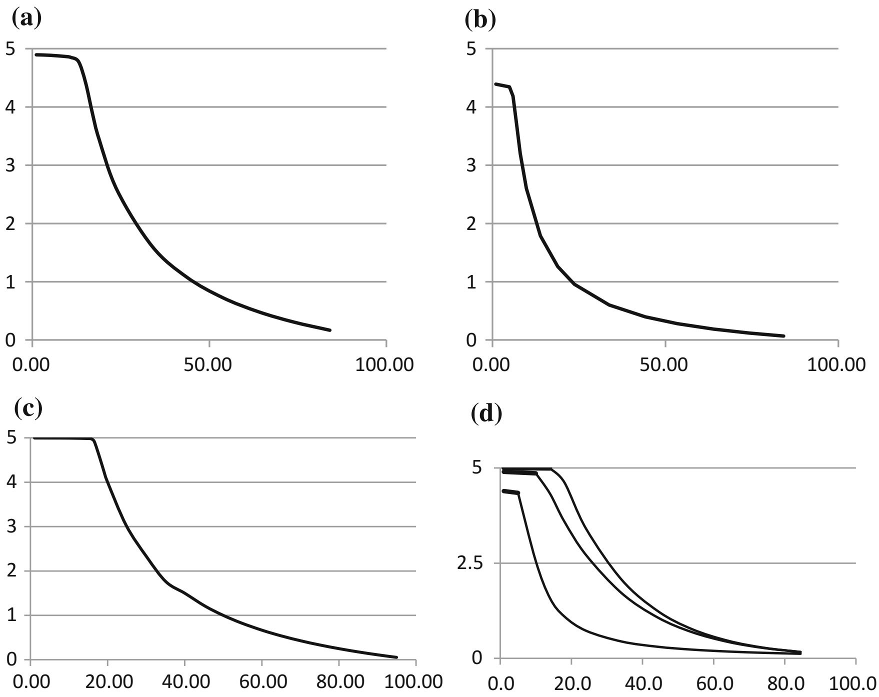


Fig. 2.19 Average speed against traffic density. **a** 10% random slowing. **b** 60% random slowing. **c** Average speed against traffic density, no random slowing. **d** Average speeds against traffic density, all cases

Note 2: the stopping distance s (minimum separation in feet) in terms of the vehicles velocity v (in miles per hour) is given by

$$s = 0.05v^2 + 2v + 15,$$

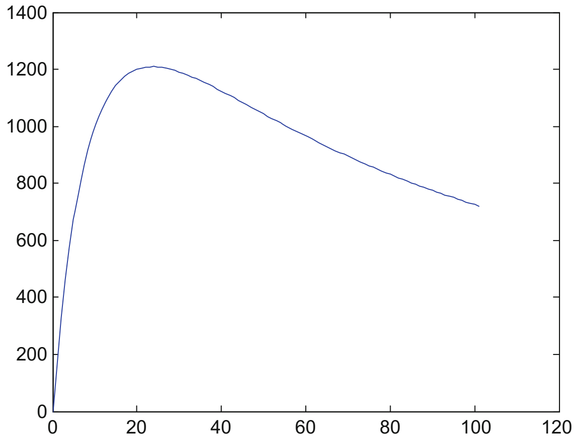
and as the average car length is 13 feet, thus the loading per mile for free-flowing traffic, model 7c, is given by

$$\mathrm{MF} = \frac{5280}{((0.05v^2 + 2v + 15) + 13)}$$

The loading per hour is therefore given by

$$\mathrm{MFH} = \frac{5280v}{((0.05v^2 + 2v + 15) + 13)}$$

Fig. 2.20 Road capacity per hour against traffic speed



At 70 mph	MF = 16.1 per mile	MFH = 1127
At 30 mph	MF = 39.7 per mile	MFH = 1191
At 24 mph	MF = 50.4 per mile	MFH = 1209

Note the optimal velocity is 24 mph; a plot of "flow per hour" against velocity is shown in Fig. 2.20. This demonstrates that low speeds, less than 15 mph are very inefficient, low capacity per hour, as are speeds greater than 100 mph.

Note 3: loadings within 10% of the optimal occur at speeds between 12.5 and 44.5 mph.

2.4.2.5 Analysing Traffic Flows

An area of interest is the frequency of occurrence of self-forming jams. Autocorrelating the historic data can be used to investigate the frequency of jams within free-flowing traffic. Figure 2.21a shows a plot obtained from correlating the position data at time t with the position data at time (t–k) over a time span of T time units. When the data has a perfect pattern, jams occur at regular intervals have the same length and traffic flows freely between jams the plot of the aurocorrelations will have this form the jam frequency being indicated by the spacing of the peak values in the plot.

Figure 2.21b shows the plot obtained for a more normal case when the jams are not (quite) evenly spaced and not (quite) the same length, but although the peaks do not have the same magnitude, the plot does give an indication of the jam frequency; here, the data can be analysed using a Hilbert–Huang transform, see Bird [2] for details.

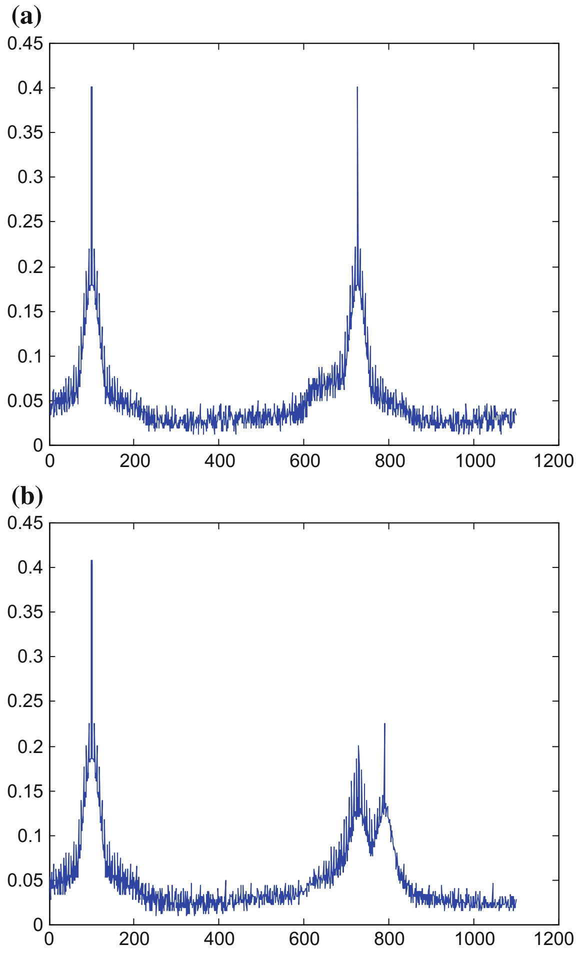


Fig. 2.21 **a** Autocorrelation perfect flow pattern. **b** Autocorrelation not (quite) perfect flow

Discussion Points:
Average Velocity and Traffic Flow

- Why would the maximum average velocity, in this model, be 4.9?
- What would be the highest traffic density that would enable traffic to flow freely without any need to reduce velocity.

Extending the Model
Obvious extensions are:

- Allow overtaking on a single lane road
- Allow for more lanes
- Include different type of vehicles.

One lane overtaking model N-S Rules (overtaking)
To enable overtaking, on a single carriageway, the model is amended so that each vehicle carries two possible velocities

Step 1: acceleration All cars that have not already reached the maximal velocity $v_{\max}$ accelerate by one unit: $v_{i+1} \Rightarrow v_i + 1$
Step 2: safety distance If a car has d empty cells in front of it and if its velocity v_{i+1}(after step 1) is larger than d, then the velocity reduces to d: $v^1{}_{i+1} \Rightarrow \min\{d, v_{i+1}\}$ $v^2{}_{i+1} \Rightarrow v_{i+1}$
Step 3: randomisation With probability p, the velocity is reduced by one unit (if v_{i+1} after step 2): $v^1{}_{i+1} \Rightarrow v^1_{i+1} - 1$ $v^2{}_{i+1} \Rightarrow v^2_{i+1}$
Step 4: driving After steps 1–3, the new velocity v_{i+1} for each car j has been determined. Car j moves forward by $v^k{}_{i+1}$ cells: If the space is free move with velocity v^2 to $x_{i+1,j} \Rightarrow x_{i,j} + v^2_{i+1}$. otherwise $x_{i+1,j} \Rightarrow x_{i,j} + v^1_{i+1}$.

Multi Lane Models
There are two cases:

Two Lane Model, here there are additional rules to

Allow a vehicle in the inside lane to move out
Allow a vehicle in the outside lane to move in
Prohibit "undertaking".

Three, or more, lane model, here there are the road rule sets

Inside lane, cars can move out
Middle lane, cars can move in or out
Outside lane, cars can move in
No undertaking

Note: having developed a three lane model, a multilane model follows using the same rule sets.

References

1. Griffeaths D, Moore C (2003) New constructions in cellular automata. Oxford
2. Bird A (2011) Early detection of aero engine damage from acoustic emission signatures, PhD thesis, Derby

Springer

http://www.springer.com/978-3-319-55416-7

Guide to Computational Modelling for Decision Processes
Theory, Algorithms, Techniques and Applications
Berry, S.; Lowndes, V.; Trovati, M. (Eds.)
2017, XII, 396 p. 170 illus., 101 illus. in color., Hardcover
ISBN: 978-3-319-55416-7