Basics of cellular automata models

CE 391F

April 2, 2013

ANNOUNCEMENTS

- Homework 3 due Thursday, April 4
- Wednesday office hours rescheduled due to CTR symposium: 9:30–11

REVIEW

The basic car following model

$$\ddot{x}_f(t) = \lambda(\dot{x}_\ell(t-T) - \dot{x}_f(t-T))$$

Local and asymptotic stability

How did λ and T affect these stability values?

What kind of λ values have been observed in experiments?

How did car-following models correspond to continuum models?

OUTLINE

CELLULAR AUTOMATA

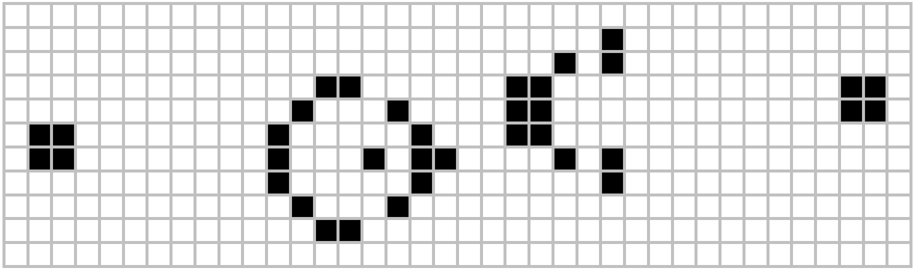
An alternative to continuous car-following are discrete "cellular automata" models.

These were developed in the 1990s by physicists, and are the traffic model used in TRANSIMS.

Cellular automata simplify car-following models in the same way that the CTM simplifies the LWR model. As we will see shortly, it also provides an easy avenue for handling lane-changing behavior.

What is a cellular automaton?

Celluar automata are defined on a discrete grid of cells, at a discrete set of times.



- Each cell exists in one of a finite number of states
- Moving from one timestep to the next, the state of each cell is updated based on the state of nearby cells.

Cellular automata were developed by John von Neumann and Stanislaw Ulam in the 1940s, and have been applied to simulate computer processors, seashell patterns, neurons, fluid dynamics, and many other objectcs.

Conway's Game of Life

The "game of life" is the best-known cellular automaton.

Imagine an infinite grid of cells, which exist in one of two possible states: "alive" or "dead"

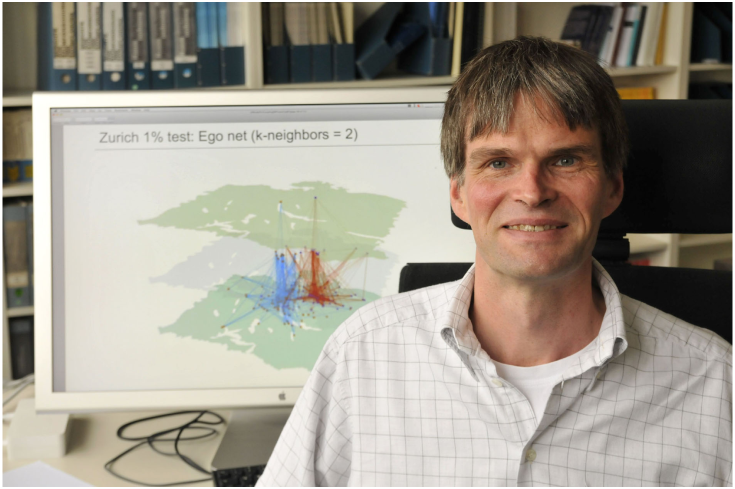
Each cell has eight neighbors, and updates occur based on the following rules:

1. Any live cell with fewer than two live neighbors dies (under-population)
2. Any live cell with two or three live neighbors stays alive
3. Any live cell with more than three live neighbors dies (over-population)
4. Any dead cell with exactly three live neighbors becomes alive (reproduction)

Using only these simple rules, a huge variety of complex patterns can be created.

In a similar way, when applied to traffic flow, cellular automata can replicate complex phenomena with a simple set of rules.

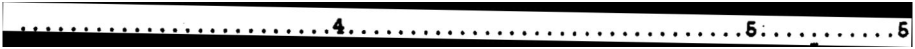
Kai Nagel pioneered the application of CAs to traffic modeling, largely at Los Alamos National Laboratory (although this research started earlier, at the Universität zu Köln).



For now, consider a one-lane roadway, which is represented with a one-dimensional line of cells.

These cells are much smaller than the CTM cells — here, a cell can contain at most one vehicle.

The state of a cell is either “empty” (if there is no vehicle present), or a nonnegative integer v expressing the vehicle’s speed (in units of cells per tick).

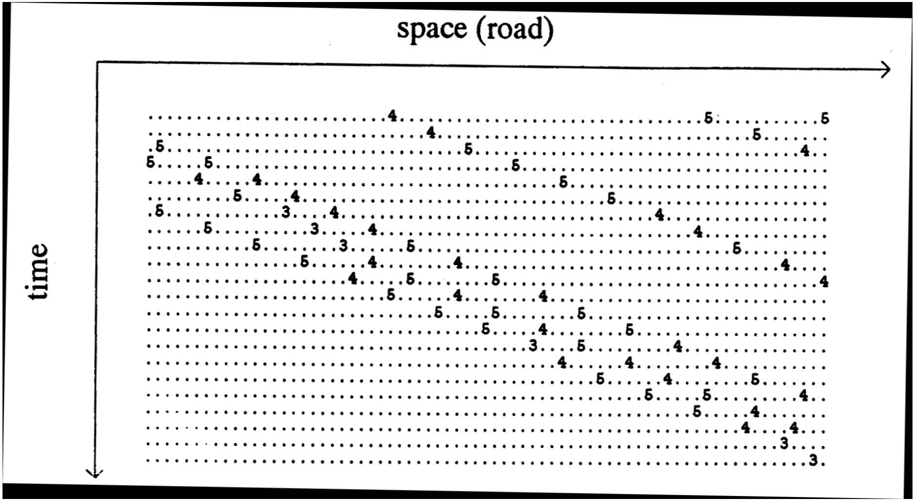


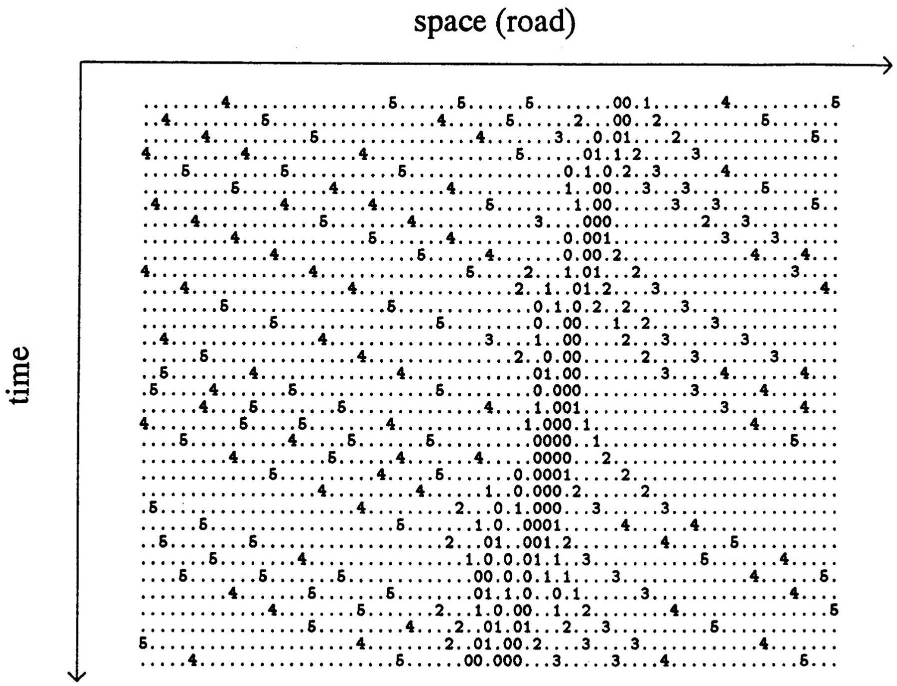
```
.........................4...................................5...........5
```

The system is governed by the following rules, all four of which are applied to each vehicle in the stated order:

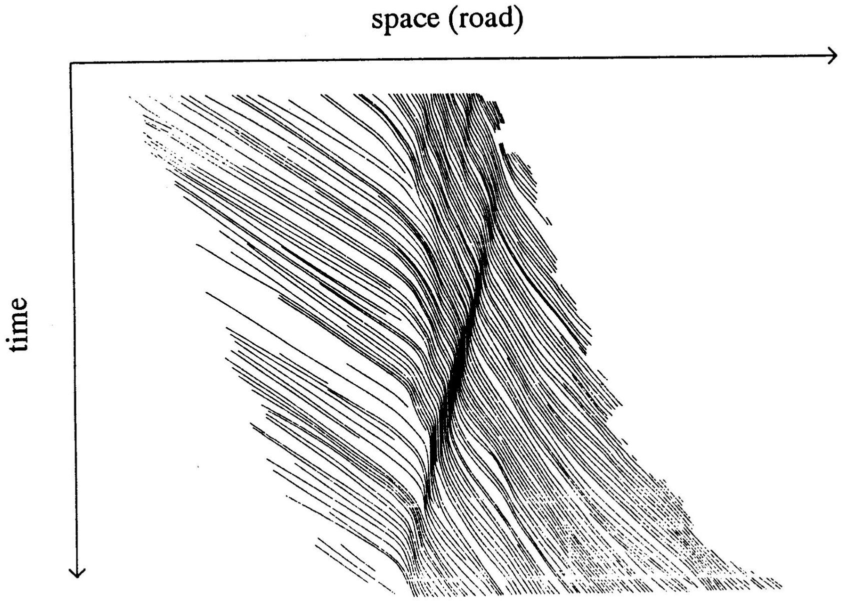
- **Acceleration:** If the velocity v is less than v_{max}, and the distance to the next car ahead is greater than $v+1$, the speed increases by 1.
- **Car-following:** If the distance to the next vehicle is j and $j \leq v$, the speed decreases to $j-1$.
- **Randomization:** If the velocity is positive, it decreases by 1 with probability p
- **Motion:** The vehicle advances v cells.

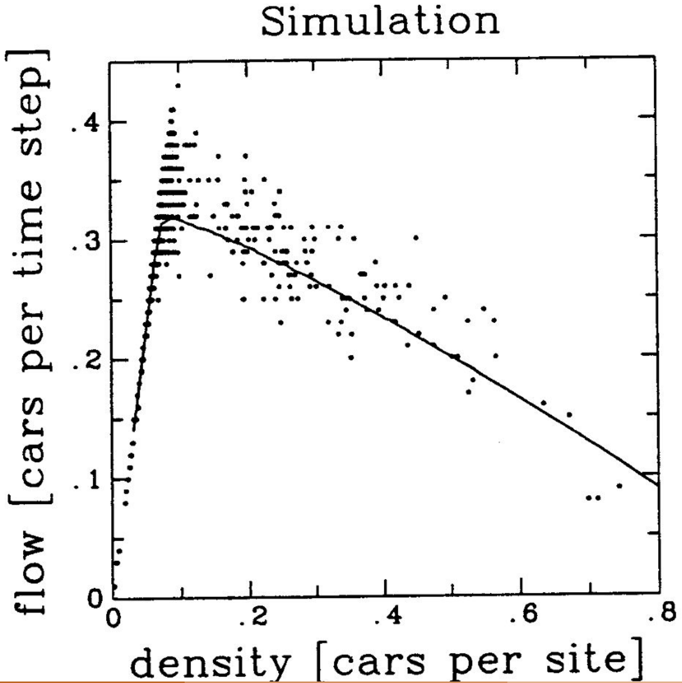
These steps are performed in parallel for each vehicle.

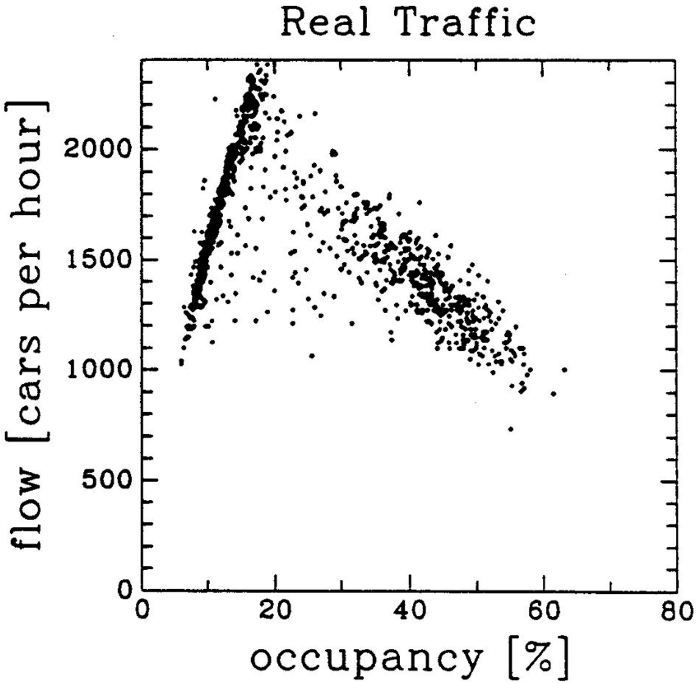

space (road)
time



```
........4................5......5......5.......00.1.......4..........5
..4.........5.................4......5.....2...00..2..........5.......
......4..........5................4......3...0.01....2.............5..
4.........4...........4...............5.....01.1.2.....3..............
....5.........5...........5...............0.1.0.2..3......4...........
.........5.........4...........4..........1..00...3...3.......5.......
.4............4........4...........5.......1.00......3...3.........5..
.....4............5........4............3...000.........2...3.........
.........4.............5.......4..........0.001...........3....3......
.............4..............5......4......0.00.2.............4....4...
4................4...............5....2...1.01...2...............3....
....4................4...............2..1..01.2....3................4.
........5................5.............0.1.0.2..2.....3...............
.............5...............5.........0..00...1..2......3............
..4...............4................3...1..00....2...3.......3.........
......5...............4..............2..0.00......2....3.......3......
..5........4..............4............01.00........3.....4.......4...
.5.....4.......5.............5.........0.000...........3......4.......
......4....5........5..............4...1.001..............3.......4...
4.........5.....5........4............1.000.1................4........
....5..........4.....5......5..........0000..1...................5....
.........4.........5......4.......4....0000...2.......................
.............5..........4....5.......0.0001.....2.....................
..................4.........4.....1..0.000.2......2...................
.5....................4........2...0.1.000...3......3.................
......5...................5......1.0..0001......4......4..............
..5........5..................2...01..001.2.........4......5..........
.......5........4...............1.0.0.01.1..3...........5.......4.....
....5.......5.......5............00.0.0.1.1....3.............4......5.
.........4.......5.......5.......01.1.0..0.1......3..............4....
.............4........5......2...1.0.00..1..2........4...............5
.................5.........4...2..01.01...2...3..........5............
5.....................4.......2..01.00.2....3....3............4.......
.....4...................5......00.000...3.....3....4.............5...
```


space (road)
time


Simulation
flow [cars per time step]
density [cars per site]
0
.1
.2
.3
.4
0
.2
.4
.6
.8


Real Traffic
flow [cars per hour]
occupancy [%]
0
500
1000
1500
2000
0
20
40
60
80

LANE CHANGING

To accommodate lane changing, we add a second row to the grid. As before, cells are either empty or contain the velocity of the vehicle in that cell.

The previous rules are called the “single-lane update rules.”

With lane changing, we allow vehicles to move laterally before applying the single-lane update rules.

Some questions to consider:

- **Symmetry:** Should the rules for changing from left-to-right be the same as those for changing right-to-left?
- **Stochasticity:** Is there any randomness involved in the decision to change lanes?
- **Anisotropy:** Drivers presumably need to look upstream before deciding whether or not to change lanes. Will this cause problems?

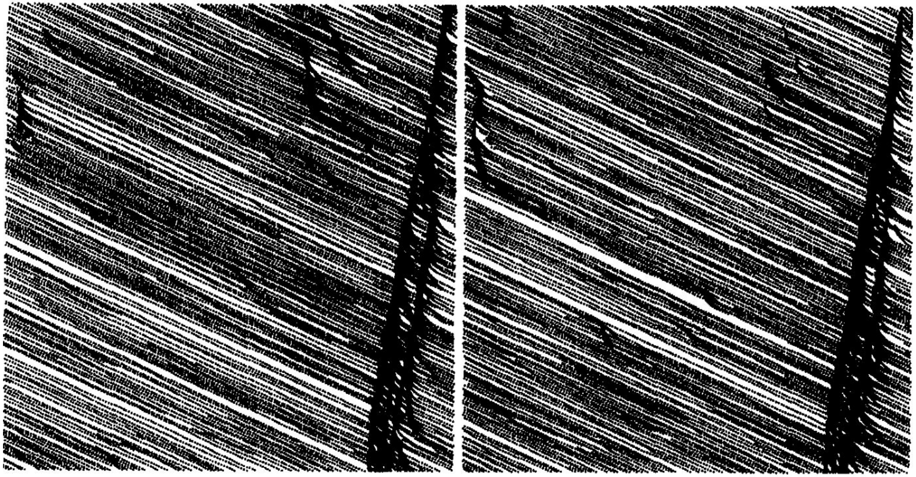
One candidate set of rules... a vehicle changes lanes if all of the following conditions are satisfied:

1. Distance to next vehicle in current lane is less than l
2. Distance to next vehicle in other lane is *greater than* l_o
3. Distance to previous vehicle in other lane is greater than $l_{o,back}$

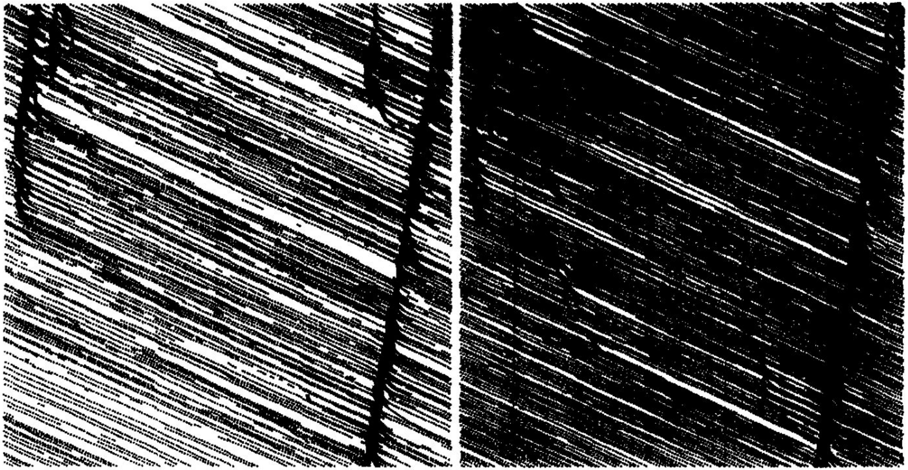
We can construct variations of these rules to describe different scenarios:

- Ignore rule 1 for left-to-right move (asymmetry)
- In addition to the above, only make the lane change with some probability (stochasticity)
- Set $l_{o,back} = 0$ (complete anisotropy)

Symmetric rules



Asymmetric rules



Ping-pong Effect

The “ping-pong” effect occurs when a platoon of vehicles continually switches from one lane to the next.

It can happen with both symmetric and asymmetric lane-switching behaviors

How can we address this?

IMPLEMENTING CELLULAR AUTOMATA

Cellular automata models are fairly easy to implement in programming language (and, with a bit more effort, in a spreadsheet).

Method 1: Explicitly simulate the state of every cell

Method 2: Only keep track of the vehicles, storing the loation and speeds.

What are some advantages and disadvantages of these methods?

Do you move vehicles all at once, or sequentially?

How do you perform a certain action with some probability?

RANDOM NUMBER GENERATION

What does it mean to generate a random number?

Most computers produce *pseudorandom* numbers: they give the appearance of randomness, while being generated by a formula.

A few historical options for generating random numbers in scientific work...

- Roll dice, draw cards, cast lots...
- Draw balls from a “well-stirred urn”
- Table of 40,000 digits “taken at random from census reports”
- Atmospheric noise

Middle-square method

Let's say we want to generate a sequence of random two-digit numbers.

Begin by picking a *seed value* 1234

The first random number is the middle two digits: 23

Square 23, and pick the middle two digits as the next random number:
$23^2 = 0\mathbf{52}9$

Square 52, and get $2\mathbf{70}4$.

So, the sequence begins 23, 52, 70, 90, and so forth.

Even though this sequence is completely deterministic, it gives an appearance of randomness.

Unfortunately, this simple method tends to get stuck in a loop:

23,52,70,90,10,10,10,10,...

Choosing a different seed gives a different sequence:
85,22,48,30,90,10,10,10,...
42,76,77,92,46,11,12,14,19,36,29,84,5,25,62,84,5,25,62,84,...

Researchers have developed much better ways of generating random numbers (and for quantifying how "random" a sequence appears

For now, we'll focus on generating a random real number from a uniform(0,1) distribution.

We can use this to simulate a wide variety of random processes. How can we use this to perform an action with probability p?

How can we convert the middle-square method into a uniform(0,1), approximately?

Stochastic desiderata

A pseudorandom $U(0,1)$ sequence would ideally pass the following tests:

- Frequency test (histograms with any bin width should show approximately equal frequency)
- Serial test (correlation should not be evident; equivalently the random number should not be easily predictable)
- Gap test (the sequence should not "avoid" particular intervals for long stretches)
- Poker test (bin data, check frequency of pairs, three-of-a-kind, full house, etc. to "true" $U(0,1)$ probabilities)
- Coupon collector test
- Run test
- Birthday spacings test