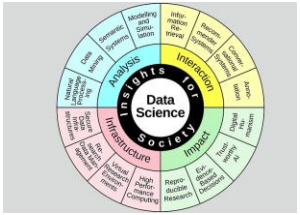
Self-Organising Systems

Rudolf Mayer



Institute of Information Systems Engineering, Data Science Group
(mayer@ifs.tuwien.ac.at)

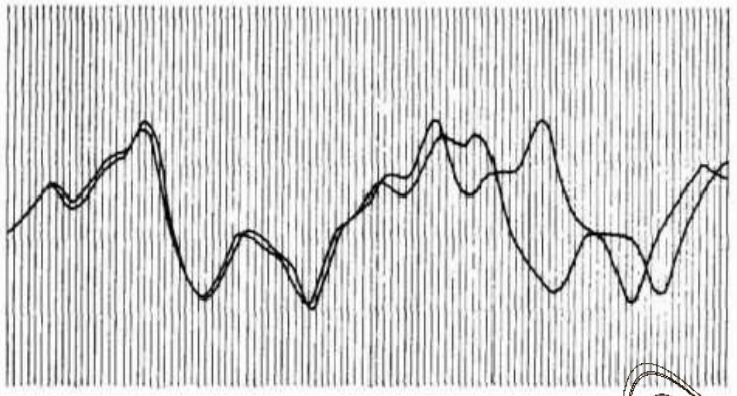
SBA Research
(www.sba-research.org)

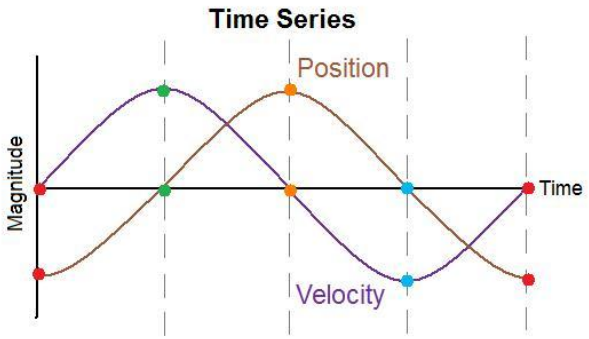
Agenda

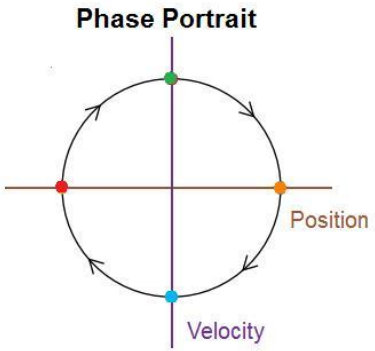
- Introduction & Definitions of Self-organising Systems
- Cellular Automata
- Genetic algorithms & Genetic programming
- Ant Colony Optimisations
  - *to be extended in swarm optimisation lectures*
- Agent Based Systems / Multi Agent Systems

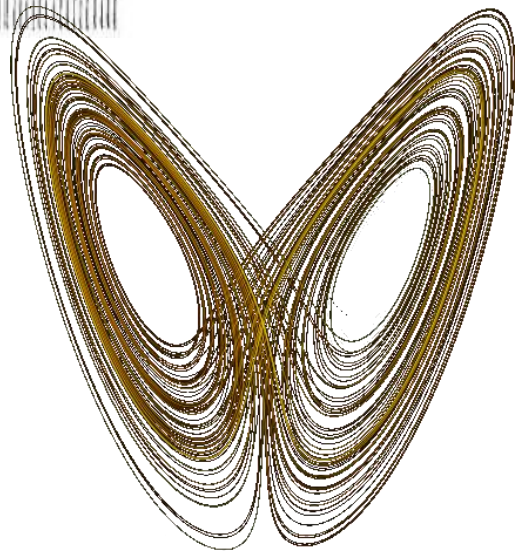
Recap – Self-Organising System

- Self Organisation principles
  - Complex system
  - Multiple interactions
  - Autonomous
  - Emergence









Agenda

- Introduction & Definitions of Self-organising Systems
- Cellular Automata
- Genetic algorithms & Genetic programming
- Ant Colony Optimisations
  - *to be extended in swarm optimisation lectures*
- Agent Based Systems / Multi Agent Systems

Cellular Automata

- *A CA is an array of identically programmed automata, or cells, which interact with one another in a neighbourhood and have definite state*
- ***Microscopic*** modelling approach

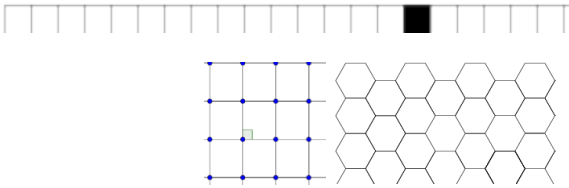
- Topics
  - Roots of the Cellular Automata Idea
  - Building Cellular Automata
  - Examples of well-known CAs
  - Behaviour of CAs
  - Applications

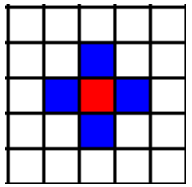
Cellular Automata

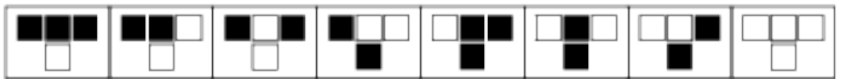
- Origins in the 1940s
  - John von Neumann, Stanislaw Ulam, Norbert Wiener
  - Theory of Computation, Automata Theory
  - Simulation of Complex Systems by Interaction of using “Simple” Rules
- 1970s: Game of Life
  - John Horton Conway
- 1980s – present: Stephen Wolfram & others
  - creator of Mathematica & Wolfram Alpha (http://www.wolframalpha.com/)

Building Cellular Automata

- Basic building blocks
  - The Cell (Node)
    - States
  - The Lattice
    - Cell Arrangement
  - Neighbourhoods
  - Transition Rules







Building Cellular Automata

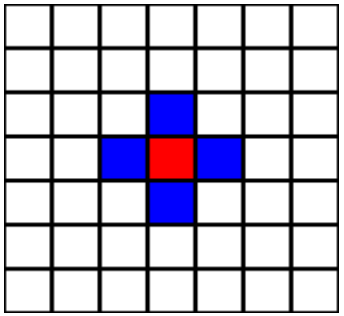
- CA consists of a regular grid of cells (lattice)
  - Each cell is in one of a finite number of states (simplest case: on/off)
- Grid can be in any finite number of dimensions
  - often 1D / 2D (especially if spatial aspect is relevant)
- Neighbourhood of a cell: set relative to a specified cell
- *Generation* creation
  - Initial state t=0 with cell assignment
  - Subsequent states (t=1,…n) created with **rules** that define the state of each cell

  ➔ Discrete modelling

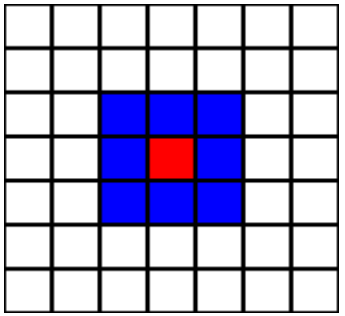
Building Cellular Automata

- Neighbourhoods

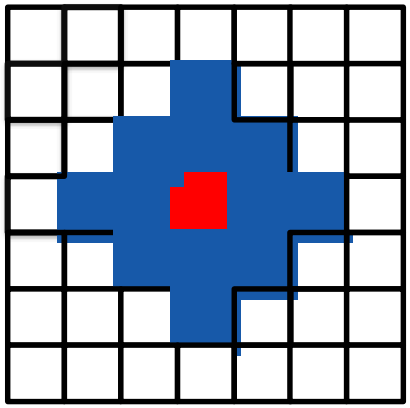
2-D

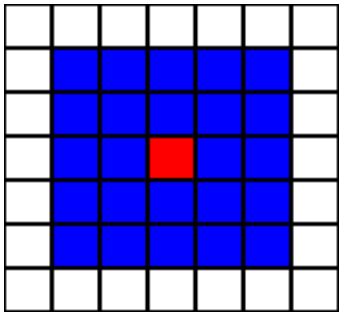


von Neumann Neighbourhood



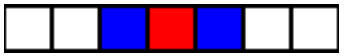
Moore Neighbourhood





Extended Neighbourhood
(2nd order)

1-D



Building Cellular Automata

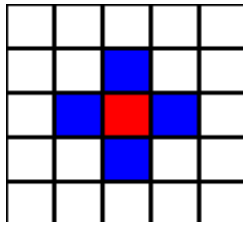
- Rules: mathematical functions; generally:
  - One rule applicable to **all** cells
  - Applied to all cells **at the same time**
    - Order is not important
  - Does not change over time
    - Exceptions, e.g. asynchronous cellular automaton

- Update rule
  - $S_{t+1} = f(s, s_1, s_2, \ldots s_n)$

new state of cell — **state of cell** — **state of the neighbors**

Update rule: simple example

Update rule: $f(s, s_1, s_2, s_3, s_4) = \sum s \bmod 2$



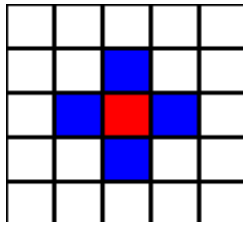
Old state

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

New state

Update rule: simple example

Update rule: $f(s, s_1, s_2, s_3, s_4) = \sum s \bmod 2$



Old state

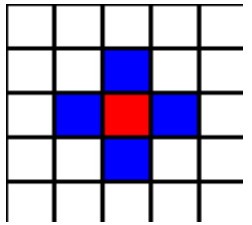
1	0	1	0	0	1	1
0	1	1	0	1	1	1
0	0	1	0	0	0	1
1	0	1	1	1	1	1
0	1	0	0	0	0	0
0	0	1	1	1	1	1
0	0	0	0	1	1	0

New state

			?			

Update rule: simple example

Update rule: $f(s, s_1, s_2, s_3, s_4) = \sum s \bmod 2$



Old state

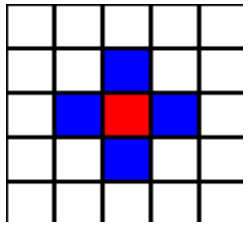
1	0	1	0	0	1	1
0	1	1	0	1	1	1
0	0	1	0	0	0	1
1	0	1	1	1	1	1
0	1	0	0	0	0	0
0	0	1	1	1	1	1
0	0	0	0	1	1	0

New state

			1			

Update rule: simple example

Update rule: $f(s, s_1, s_2, s_3, s_4) = \sum s \bmod 2$



Old state

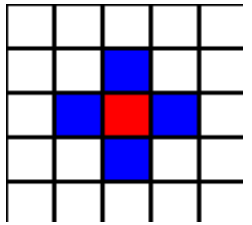
1	0	1	0	0	1	1
0	1	1	0	1	1	1
0	0	1	0	0	0	1
1	0	1	1	1	1	1
0	1	0	0	0	0	0
0	0	1	1	1	1	1
0	0	0	0	1	1	0

New state

			1			
				?		

Update rule: simple example

Update rule: $f(s, s_1, s_2, s_3, s_4) = \sum s \bmod 2$



Old state

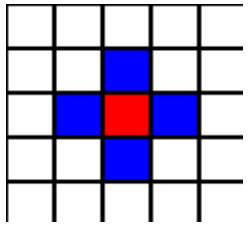
1	0	1	0	0	1	1
0	1	1	0	1	1	1
0	0	1	0	0	0	1
1	0	1	1	1	1	1
0	1	0	0	0	0	0
0	0	1	1	1	1	1
0	0	0	0	1	1	0

New state

			1			
				0		

Update rule: simple example

Update rule: $f(s, s_1, s_2, s_3, s_4) = \sum s \bmod 2$



Old state

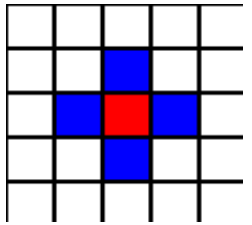
1	0	1	0	0	1	1
0	1	1	0	1	1	1
0	0	1	0	0	0	1
1	0	1	1	1	1	1
0	1	0	0	0	0	0
0	0	1	1	1	1	1
0	0	0	0	1	1	0

New state

						?
			1			
				0		

Update rule: simple example

Update rule: $f(s, s_1, s_2, s_3, s_4) = \sum s \bmod 2$



Old state

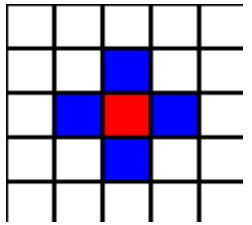
1	0	1	0	0	1	1
0	1	1	0	1	1	1
0	0	1	0	0	0	1
1	0	1	1	1	1	1
0	1	0	0	0	0	0
0	0	1	1	1	1	1
0	0	0	0	1	1	0

New state

						0
			1			
				0		

Update rule: simple example

Update rule: $f(s, s_1, s_2, s_3, s_4) = \sum s \bmod 4$



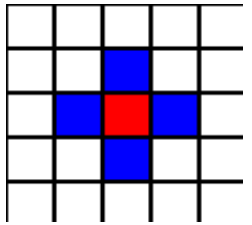
Old state

1	2	1	0	2	1	1
2	3	1	2	3	1	1
0	2	1	2	0	2	1
3	2	1	1	1	5	1
0	1	2	0	2	2	0
2	0	1	1	1	3	1
2	2	0	2	3	1	2

New state

Update rule: simple example

Update rule: $f(s, s_1, s_2, s_3, s_4) = \sum s \bmod 4$



Old state

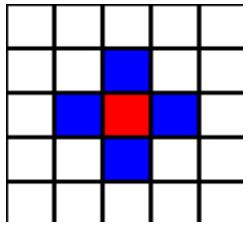
1	2	1	0	2	1	1
2	3	1	2	3	1	1
0	2	1	2	0	2	1
3	2	1	1	1	5	1
0	1	2	0	2	2	0
2	0	1	1	1	3	1
2	2	0	2	3	1	2

New state

			1			

Update rule: simple example

Update rule: $f(s, s_1, s_2, s_3, s_4) = \sum s \bmod 4$



Old state

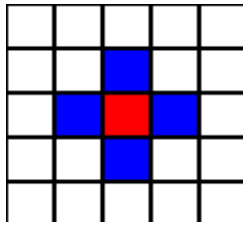
1	2	1	0	2	1	1
2	3	1	2	3	1	1
0	2	1	2	0	2	1
3	2	1	1	1	5	1
0	1	2	0	2	2	0
2	0	1	1	1	3	1
2	2	0	2	3	1	2

New state

			1			
				2		

Update rule: simple example

Update rule: $f(s, s_1, s_2, s_3, s_4) = \sum s \bmod 4$



Old state

1	2	1	0	2	1	1
2	3	1	2	3	1	1
0	2	1	2	0	2	1
3	2	1	1	1	5	1
0	1	2	0	2	2	0
2	0	1	1	1	3	1
2	2	0	2	3	1	2

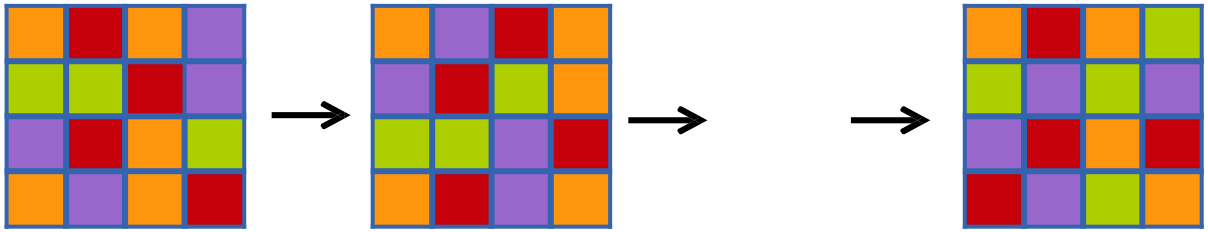
New state

						0
			1			
				2		

Updates

- Updates happen to all cells simultaneously
  - Neighborhoods are all computed from the same system state
  - Update order of cells is irrelevant

- Steps in one iteration
  - Determine neighbors of all cells
  - Compute state updates for all cells (and store them)
  - Apply the updated states



Rule Types

- **Explicit** Rules: every group of states of the neighbourhood cells is related to a state of the core cell
  - E.g. 1-D CA: a rule could be "011 -> x0x"
  - Core cell becomes 0 in the next time step **if left cell** is 0, **right cell** is 1 **and core cell** is 1. *Every possible state has to be described.*

- ***Totalistic*** Rules: state of the core cell only dependent upon a *function of the states* (often: sum) of the neighbourhood cells
  - E.g. If sum of adjacent cells is 4 ➔ state of the core cell is 1; in all other cases the state of the core cell is 0.

- ***Legal*** Rules: Subset of all *possible* rules
  - Those that produce no 1s from 0-state lattices, and provide symmetry (e.g. 110 → 1 and 011 → 1)

Algorithm properties

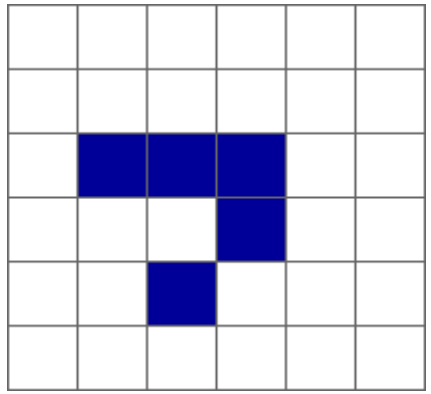
- CAs develop in space and time
- Cells arranged to n-dimensional lattices
- Finite and discrete cell states
- Cells have identical properties and transition rules
- Future state of cell only depending on
  - Neighbourhood of cell and
  - Defined transition rules

- Discrete Simulation Method

Conway's Game of Life

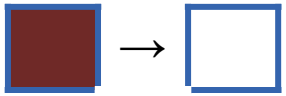
- John Horton Conway, 1970
- Eight neighbours (Moore neighbourhood)

- Rules
  - A cell that is dead becomes alive at time t+1 if exactly three cells are alive (reproduction)
  - A cell that is alive at time t dies at time t+1 if at time t
    - less than two (under-population) or
    - more than three cells are alive (over-crowding)
  - All instances of Game of Life follow same rule
    - Difference is in initial state

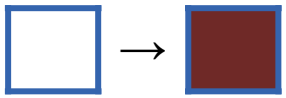


Conway's Game of Life

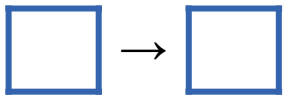
- Rules:



An alive cell with fewer than two or more than three alive neighbors dies (“under-population” or “overcrowding”)



A dead cell with exactly three alive neighbors becomes alive (“reproduction”)



Cells keep their state in any other case

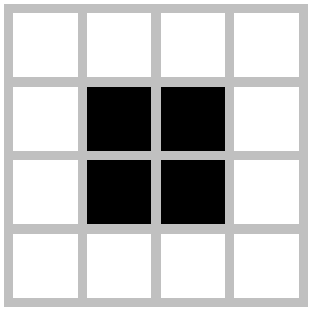
Conway's Game of Life

- A very simple example
  - Well suited to show the **concepts** of CAs
- Well studied
  - Pattern analysis of the Game of Life became its own science

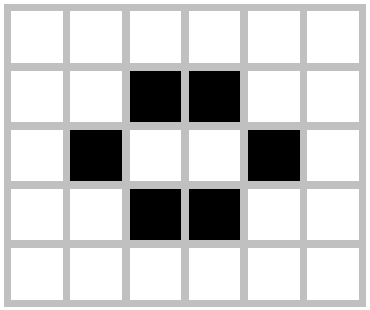
Conway's Game of Life: Examples

- Still / static

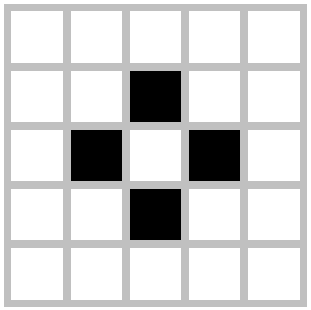
- dead ➔ alive: exactly three cells are alive
- alive ➔ dead: less than two or more than three cells are alive



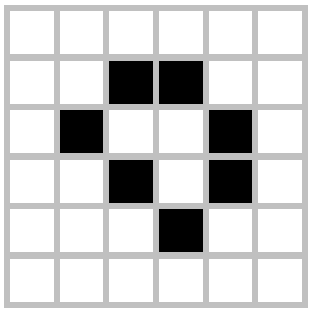
Block



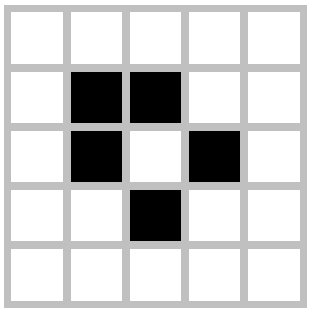
Beehive



Tub



Loaf



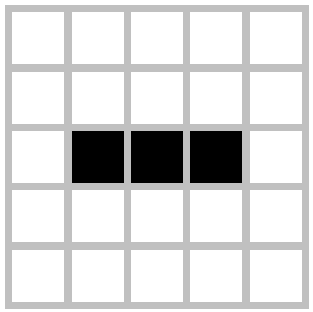
Boat

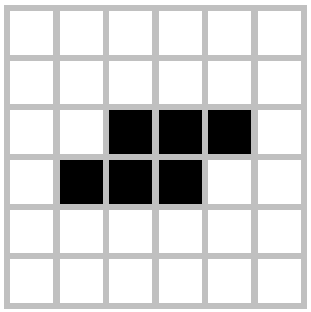
http://en.wikipedia.org/wiki/Conway%27s_Game_of_Life

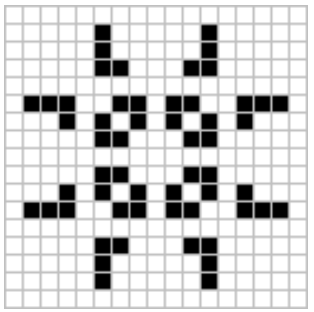
Conway's Game of Life: Examples

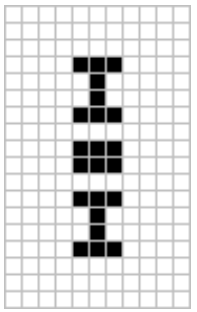
- Oscilating

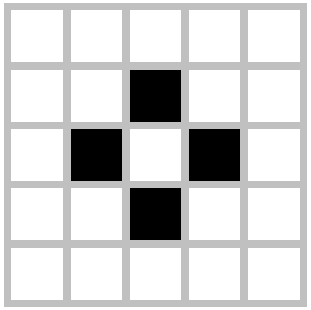
- dead ➔ alive: exactly three cells are alive
- alive ➔ dead: less than two or more than three cells are alive

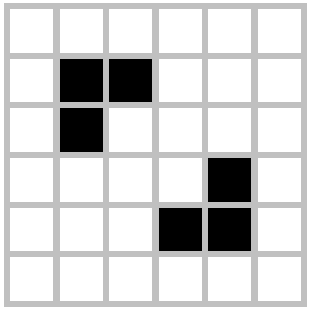












2 cycles

3 cycles

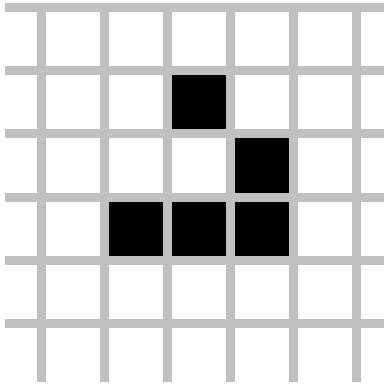
15 cycles

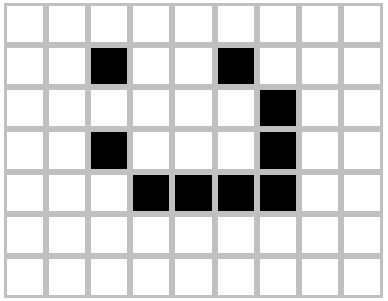
http://en.wikipedia.org/wiki/Conway%27s_Game_of_Life

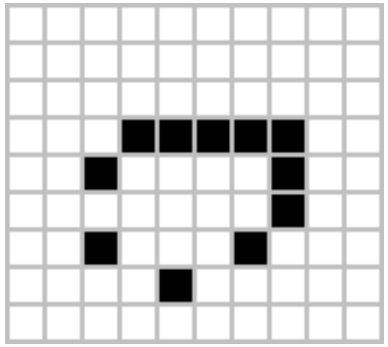
Conway's Game of Life: Examples

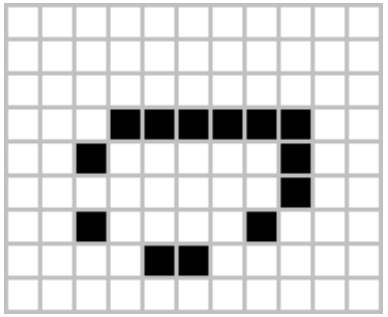
- Spaceships

- **dead ➔ alive: exactly three cells are alive**
- **alive ➔ dead: less than two or more than three cells are alive**







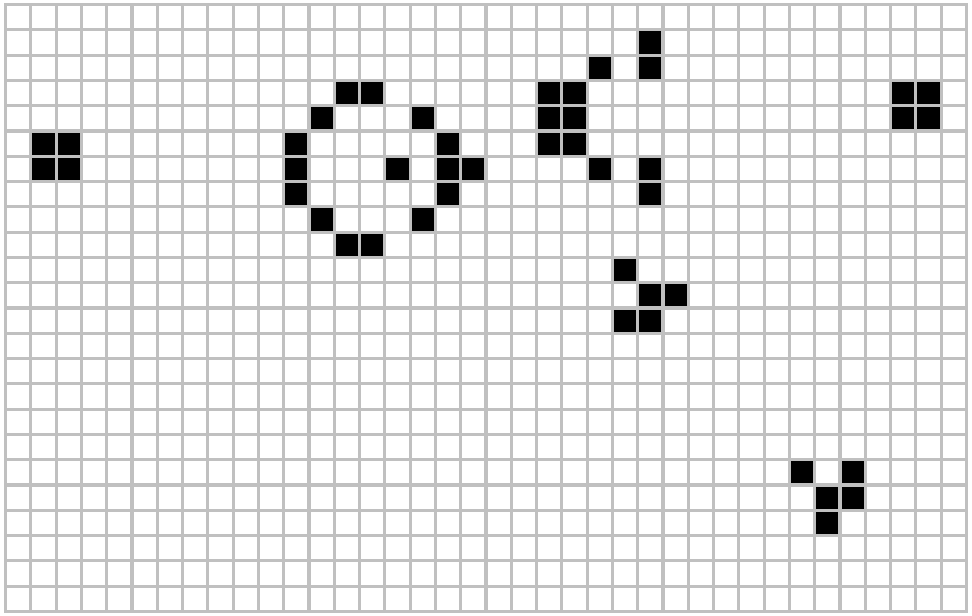


http://en.wikipedia.org/wiki/Conway%27s_Game_of_Life

Conway's Game of Life: gun

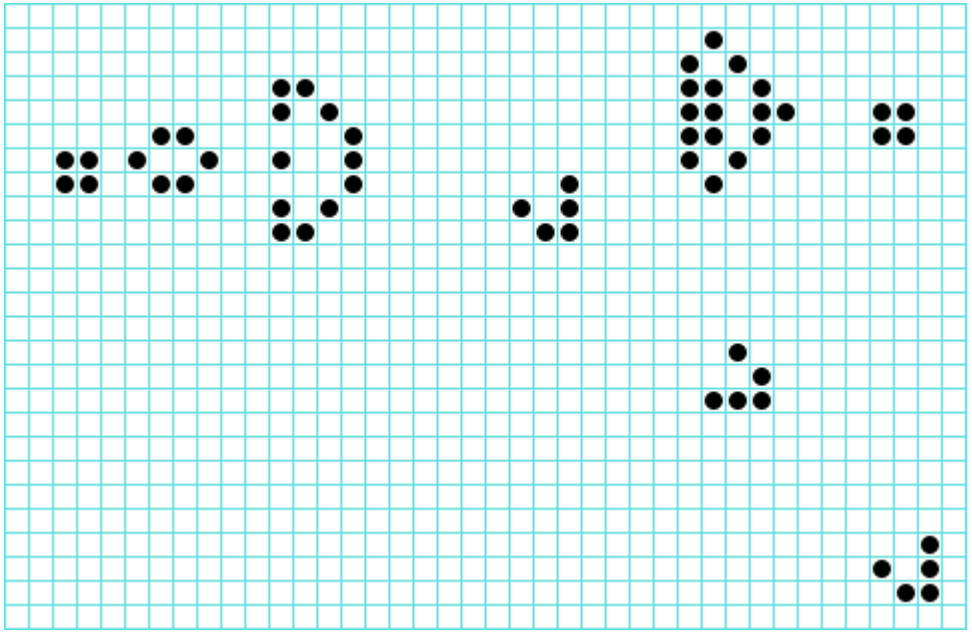
- Infinite growth: pattern that exhibit infinite growth, population is unbounded
  - Conway initially considered that impossible
- First example: Gosper glider gun, found by Bill Gosper (1970)
- Gun: main part that oscillates and periodically emits “spaceships”

Conway's Game of Life: gun



- Source: http://www.conwaylife.com

Conway's Game of Life: gun



- Source: http://www.numericana.com

Conway's Game of Life: “Breeder”

http://en.wikipedia.org/wiki/Conway%27s_Game_of_Life

Conway's Game of Life: “Puffer train”

http://en.wikipedia.org/wiki/Conway%27s_Game_of_Life

Conway's Game of Life: “Rake”

http://en.wikipedia.org/wiki/Conway%27s_Game_of_Life

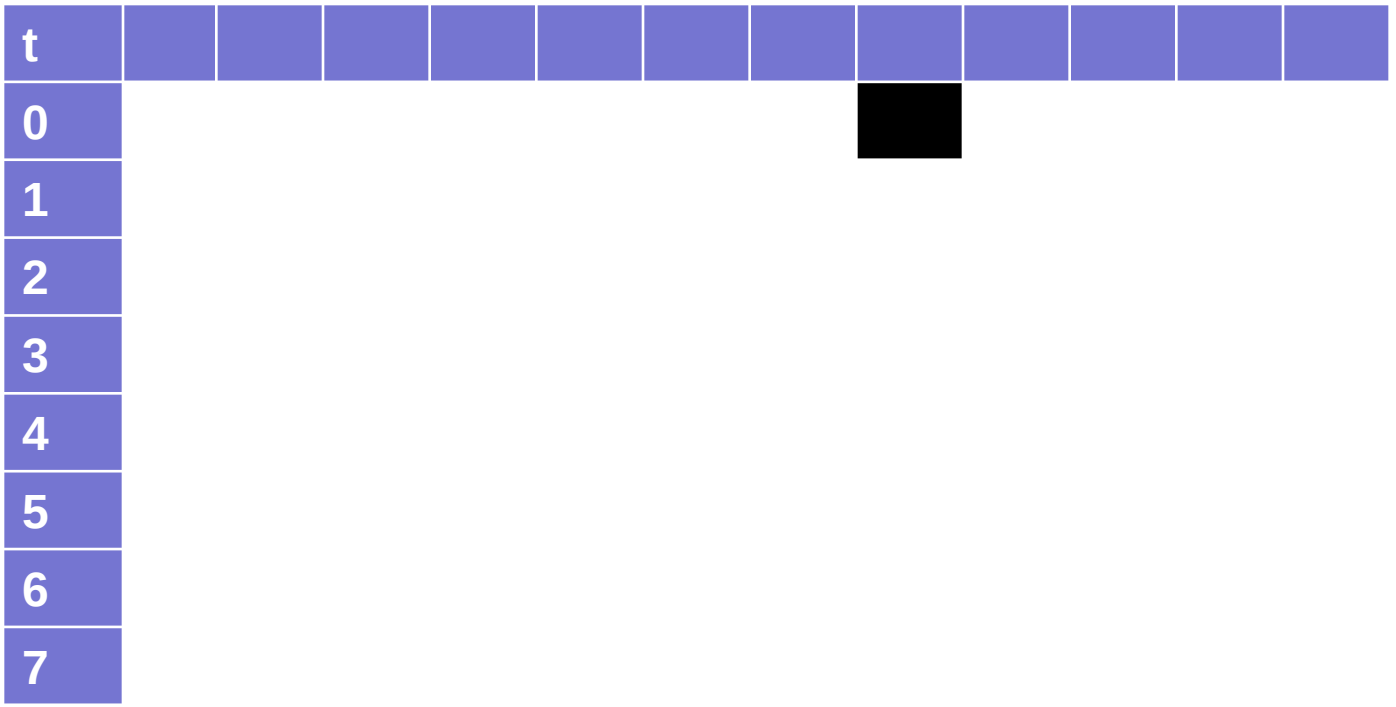
Conway's Game of Life

- A very simple example
  - Well suited to show the **concepts** of CAs
- Well studied
  - Pattern analysis of the Game of Life became its own science
- But - doesn’t demonstrate the **full power** of CAs

Cellular automata: examples

- Rule 110 (Wolfram)
  - 1-dimensional automaton (➔ two neighbours)

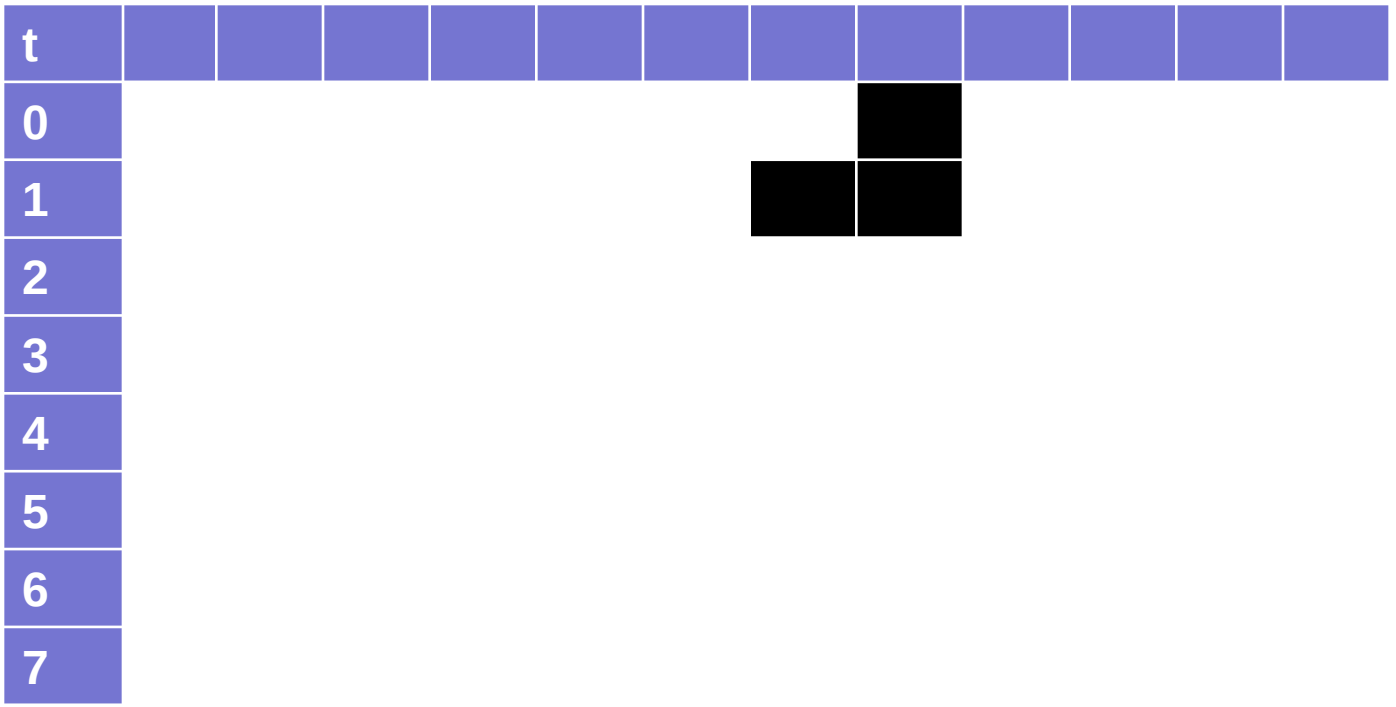
t=0	111	110	101	100	011	010	001	000
t=1	0	1	1	0	1	1	1	0



Cellular automata: examples

- Rule 110 (Wolfram)
  - 1-dimensional automaton (➔ two neighbours)

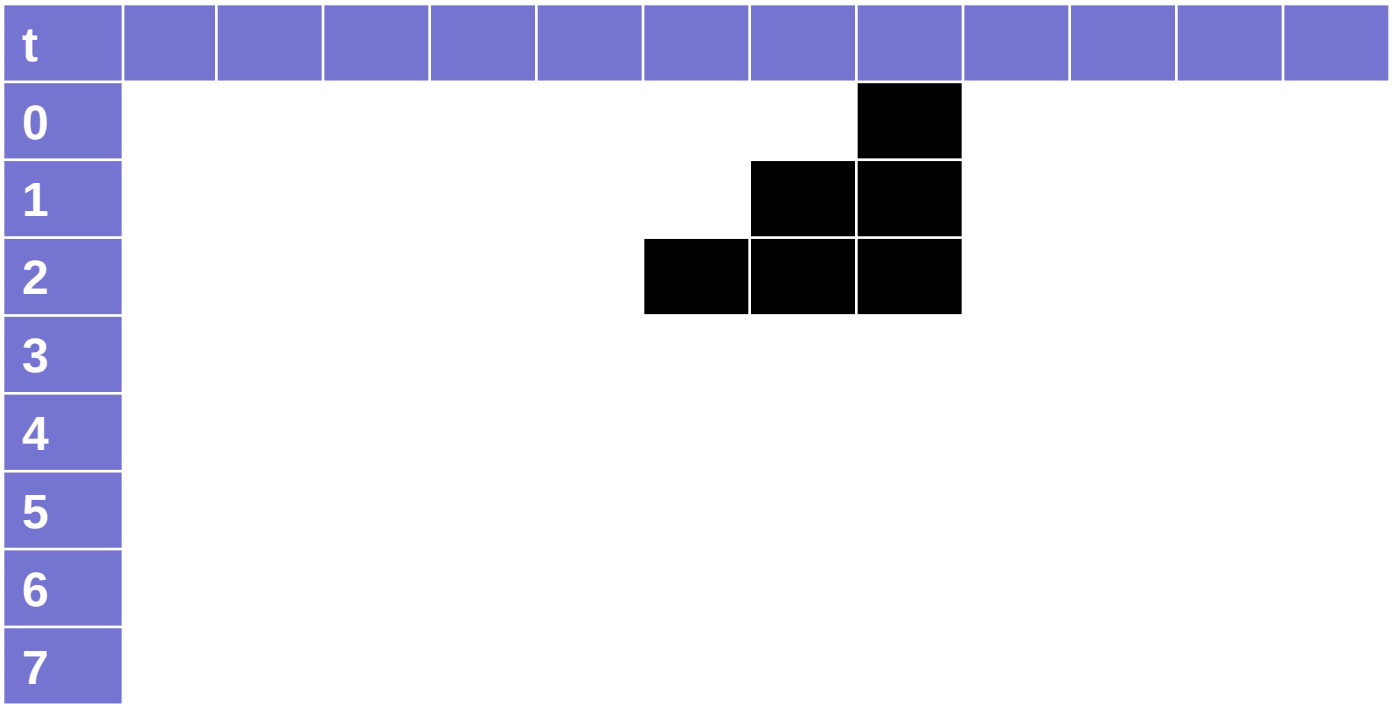
t=0	111	110	101	100	011	010	001	000
t=1	0	1	1	0	1	1	1	0



Cellular automata: examples

- Rule 110 (Wolfram)
  - 1-dimensional automaton (➔ two neighbours)

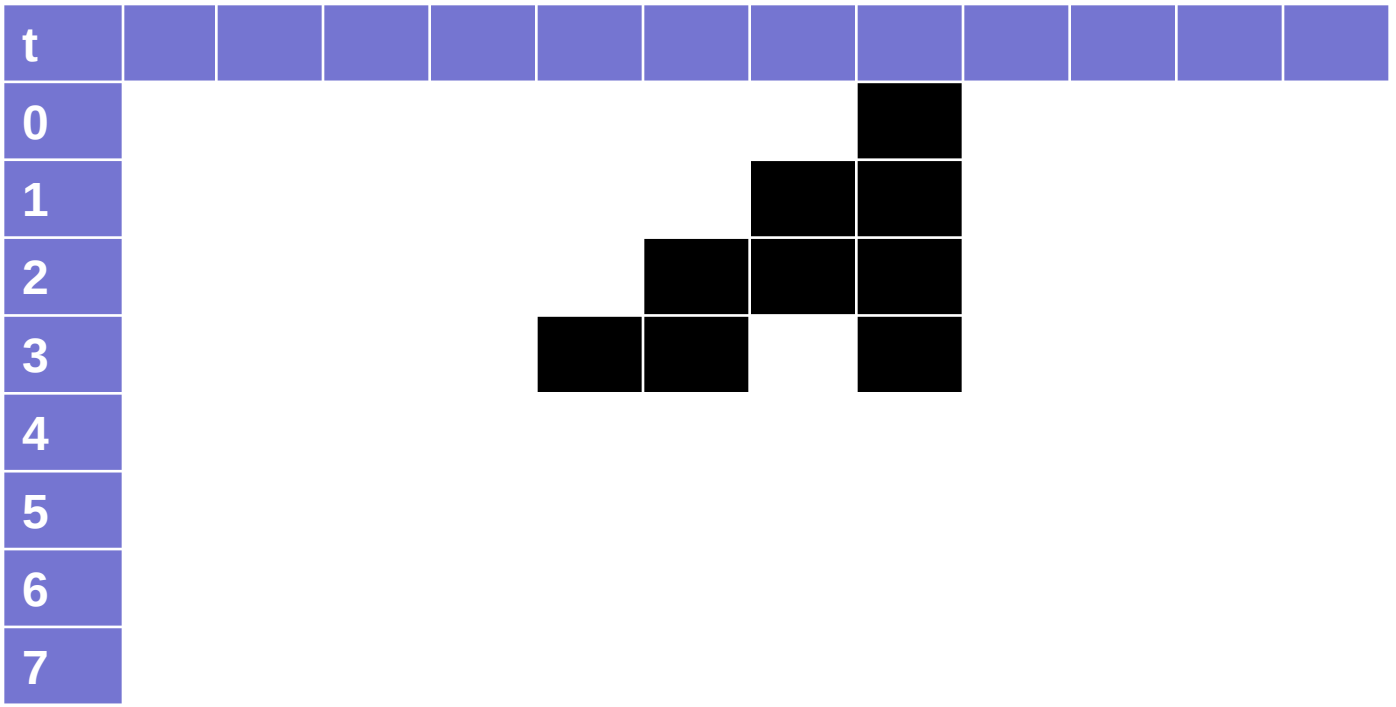
t=0	111	110	101	100	011	010	001	000
t=1	0	1	1	0	1	1	1	0



Cellular automata: examples

- Rule 110 (Wolfram)
  - 1-dimensional automaton (➔ two neighbours)

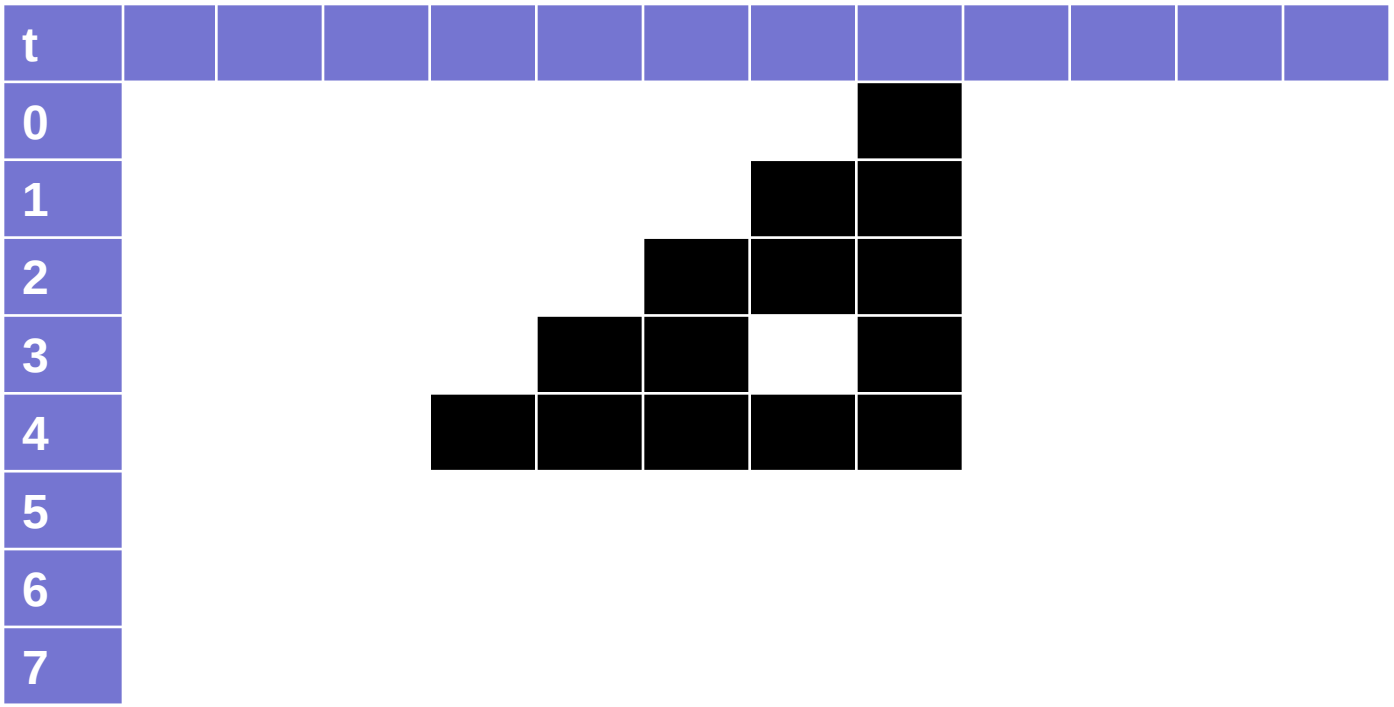
t=0	111	110	101	100	011	010	001	000
t=1	0	1	1	0	1	1	1	0



Cellular automata: examples

- Rule 110 (Wolfram)
  - 1-dimensional automaton (➔ two neighbours)

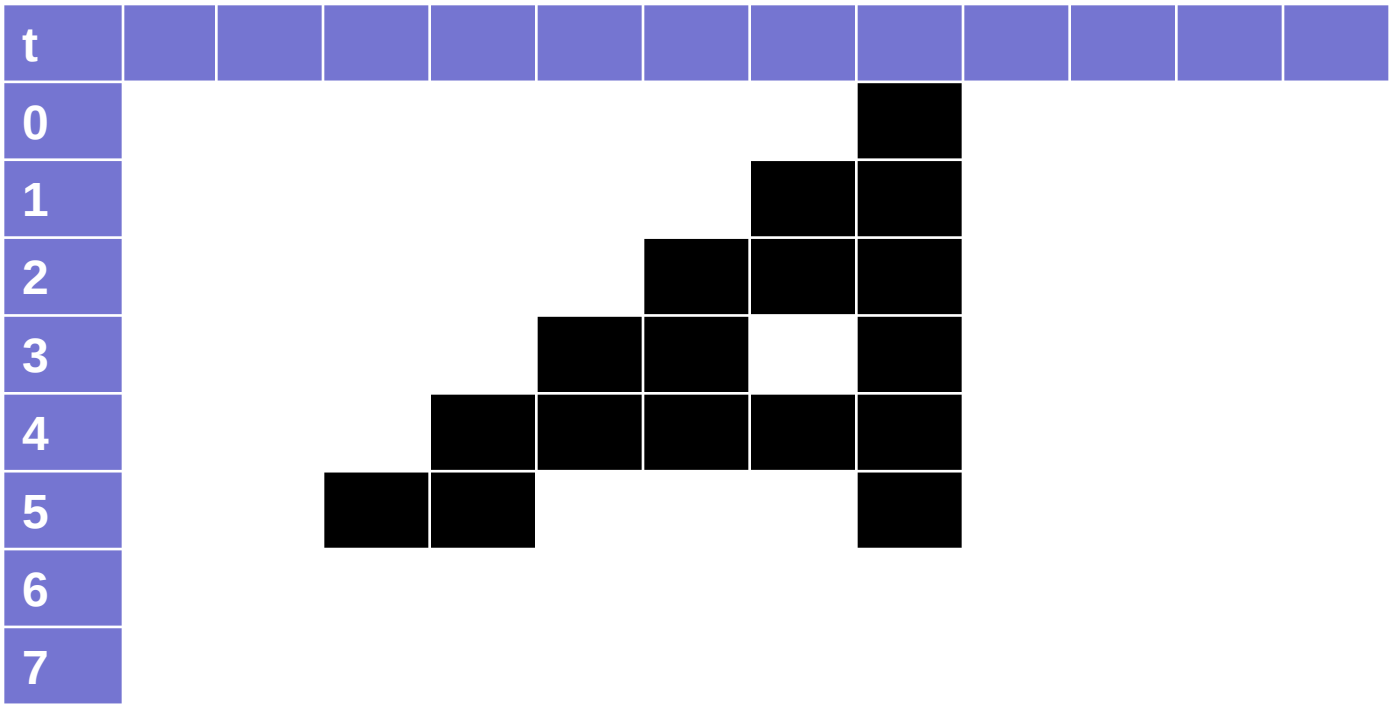
t=0	111	110	101	100	011	010	001	000
t=1	0	1	1	0	1	1	1	0



Cellular automata: examples

- Rule 110 (Wolfram)
  - 1-dimensional automaton (➔ two neighbours)

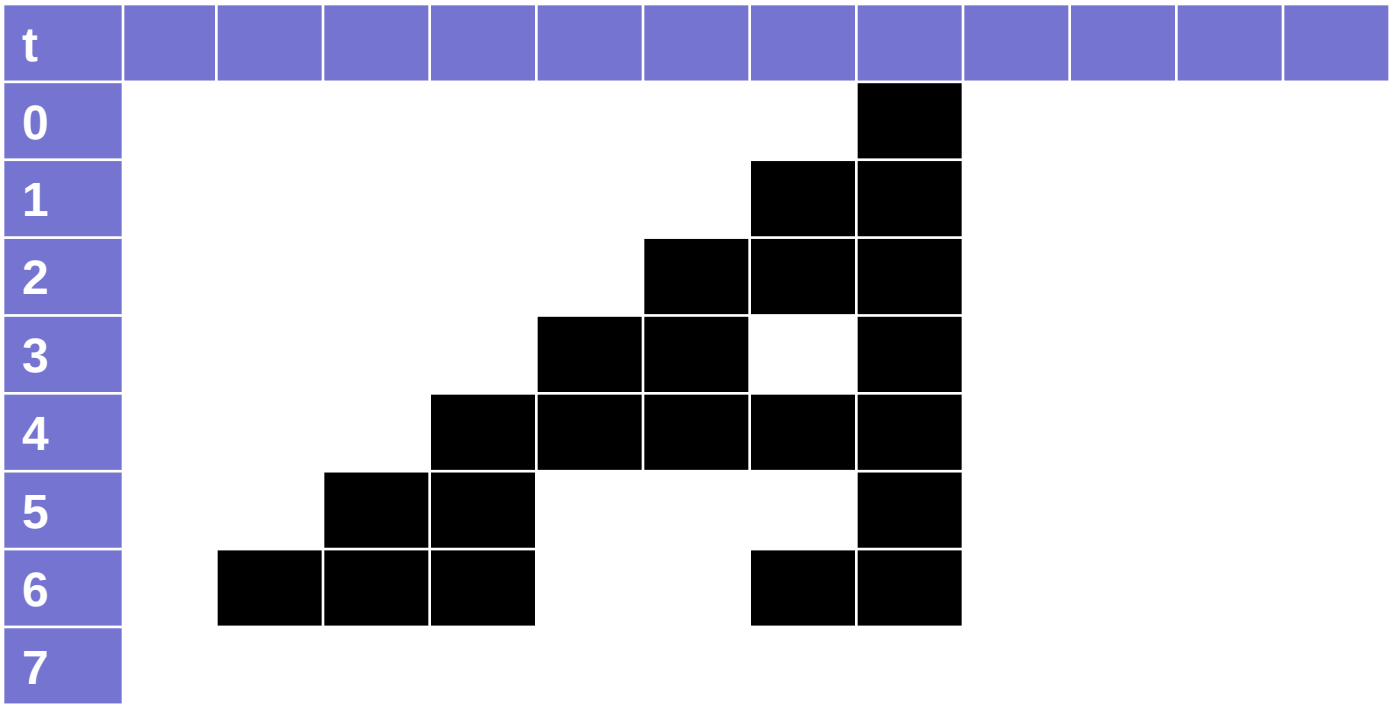
t=0	111	110	101	100	011	010	001	000
t=1	0	1	1	0	1	1	1	0



Cellular automata: examples

- Rule 110 (Wolfram)
  - 1-dimensional automaton (➔ two neighbours)

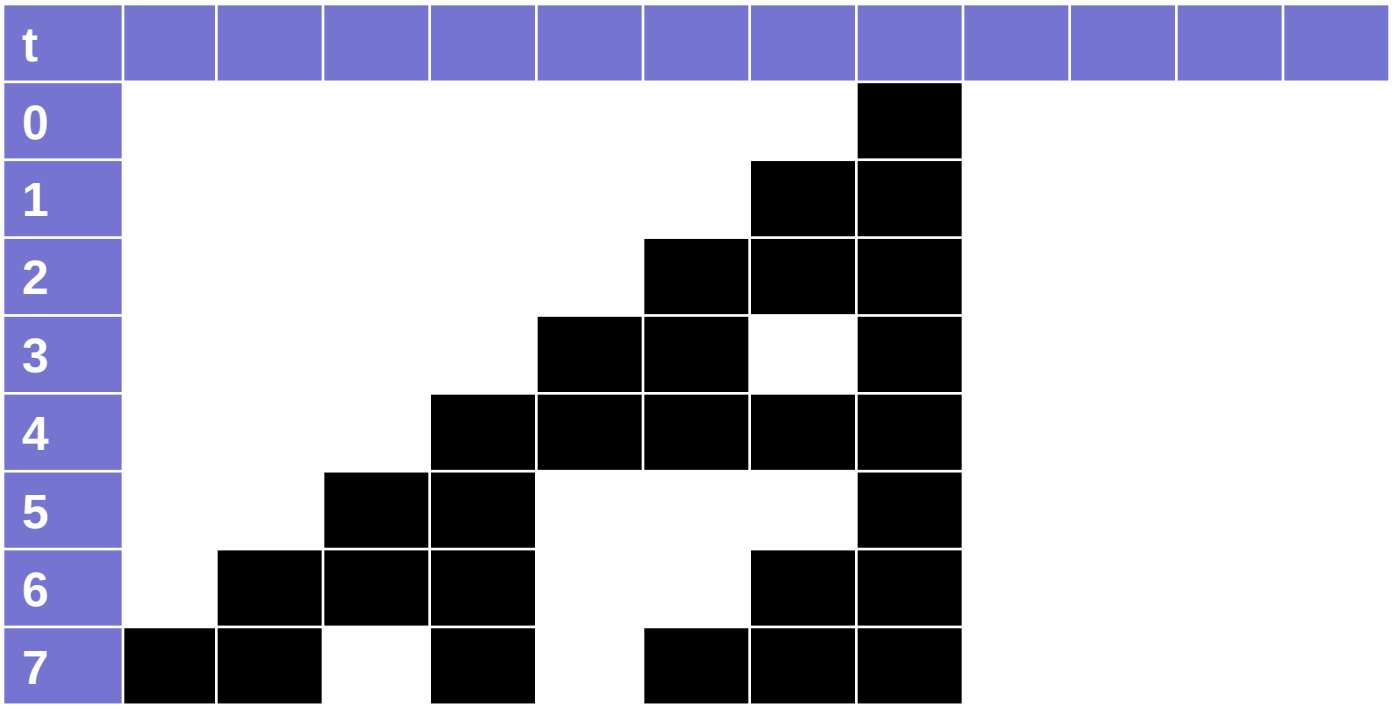
t=0	111	110	101	100	011	010	001	000
t=1	0	1	1	0	1	1	1	0



Cellular automata: examples

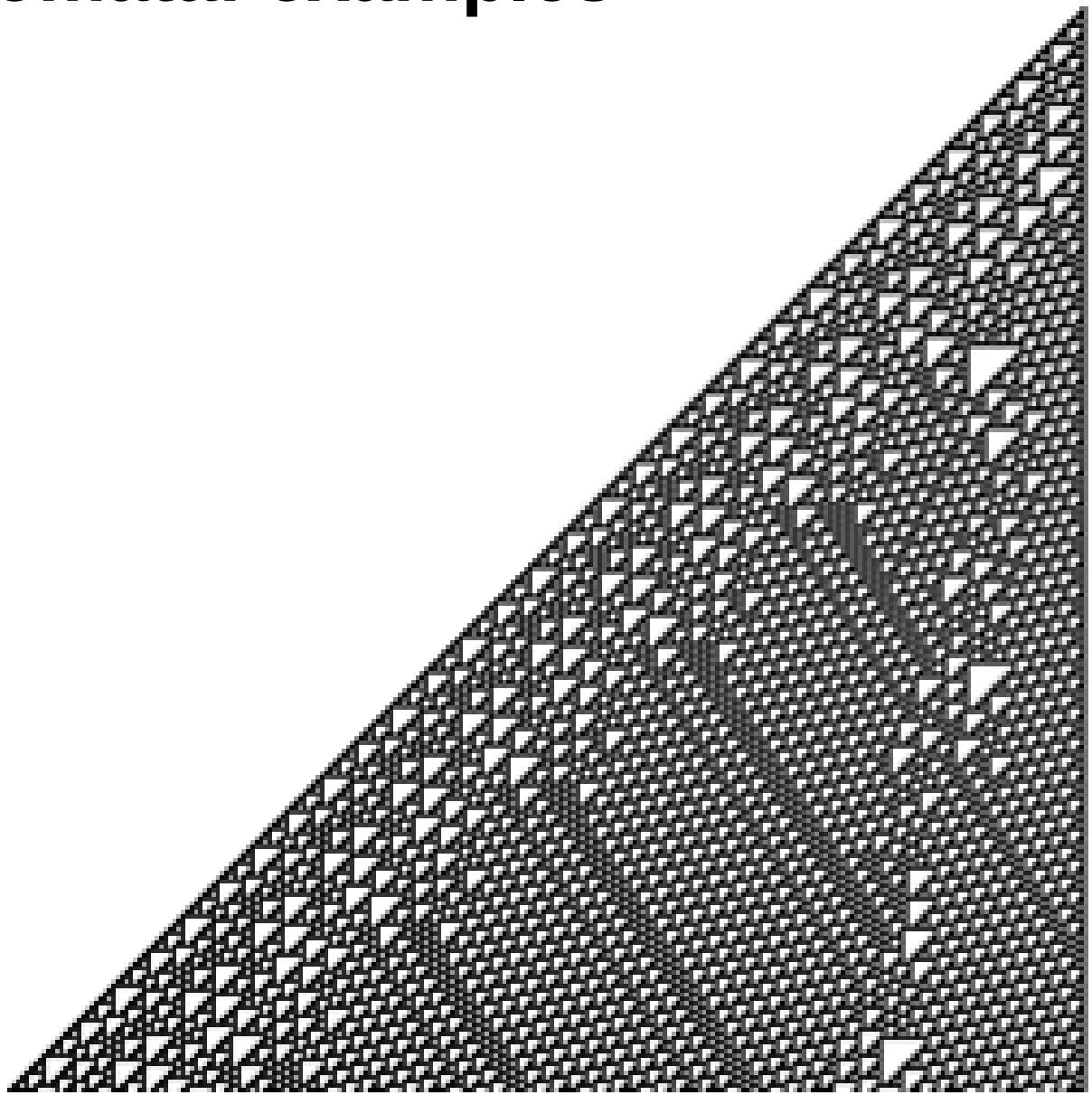
- Rule 110 (Wolfram)
  - 1-dimensional automaton (➔ two neighbours)

t=0	111	110	101	100	011	010	001	000
t=1	0	1	1	0	1	1	1	0



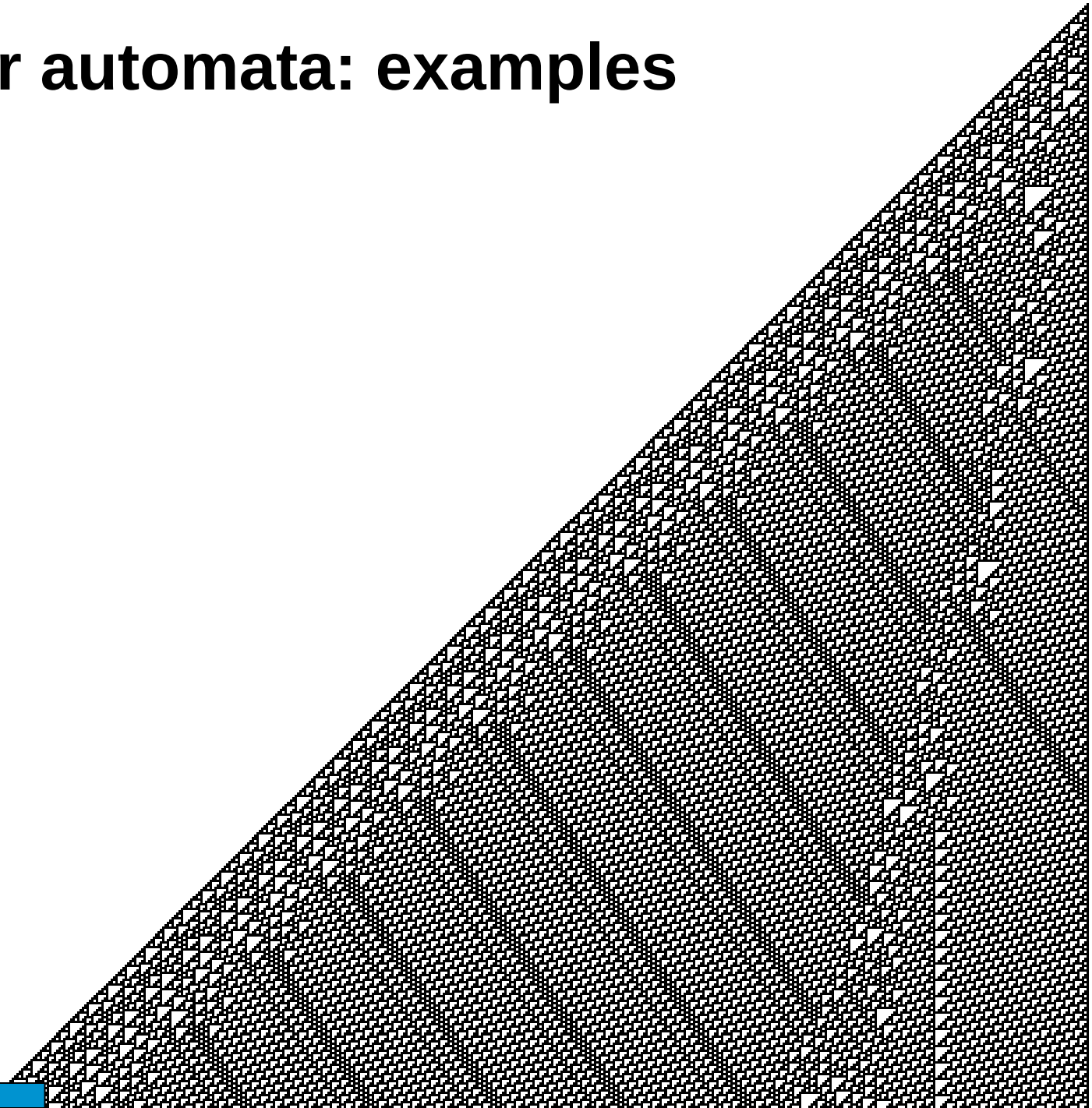
Cellular automata: examples

- Rule 110



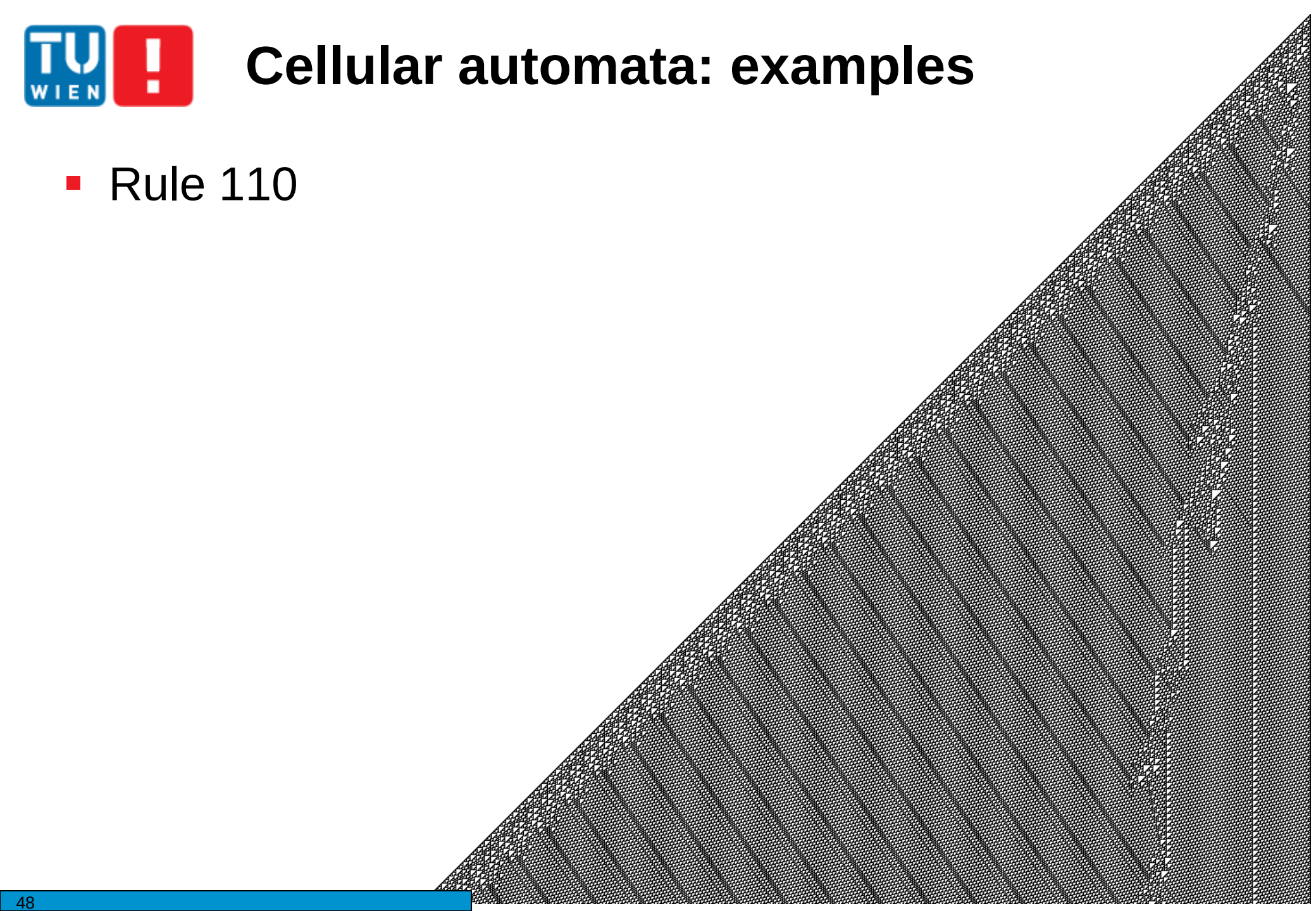
Cellular automata: examples

- Rule 110



Cellular automata: examples

- Rule 110



Cellular automata: examples

- Rule 110 (Wolfram)
  - Name ?

t=0	111	110	101	100	011	010	001	000
t=1	0	1	1	0	1	1	1	0

Cellular automata: examples

- Rule 110 (Wolfram)
  - Name ?

t=0	111	110	101	100	011	010	001	000
t=1	0	1	1	0	1	1	1	0

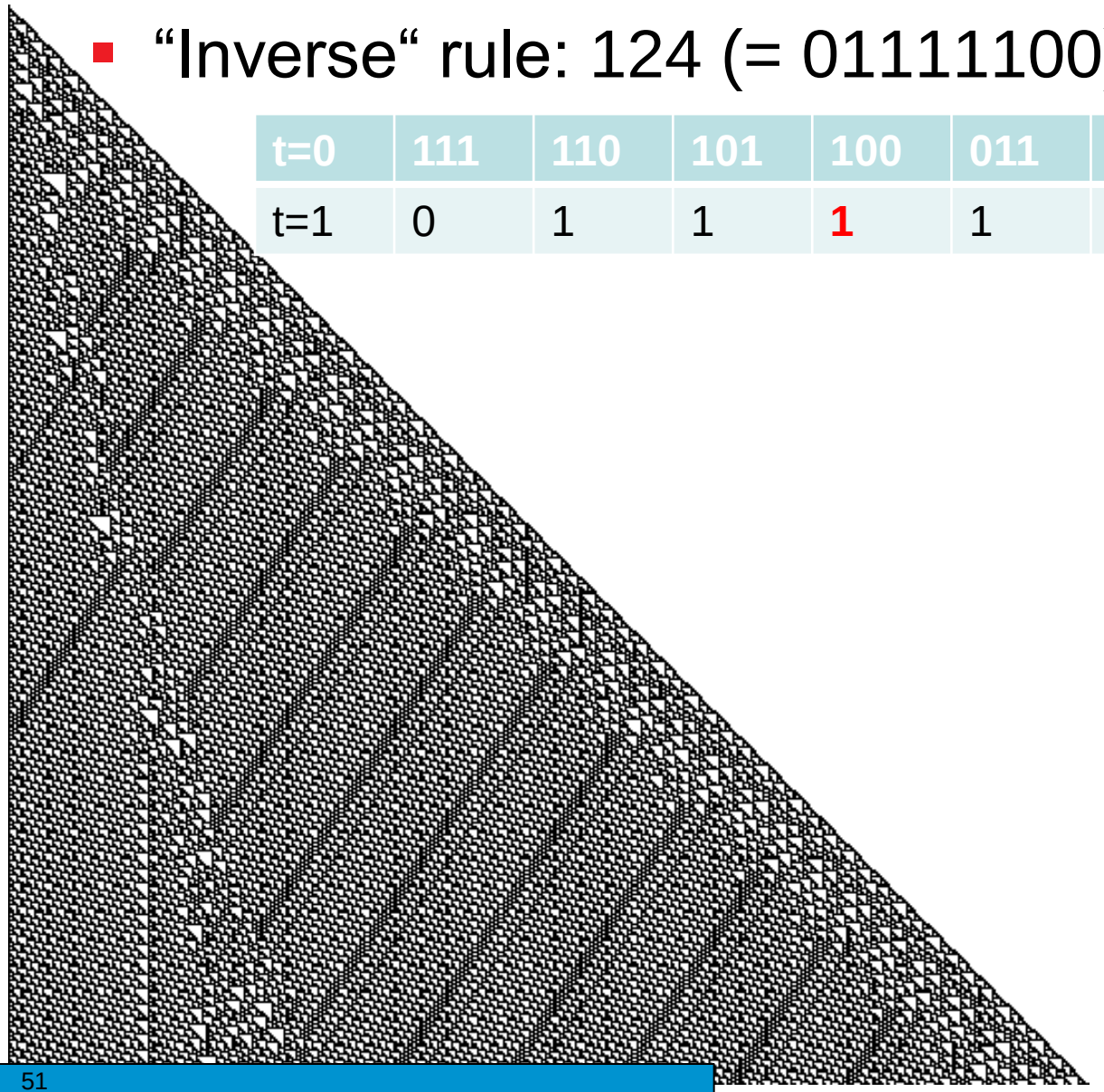
    - 01101110 binary == 110 decimal

  - "Neither completely stable nor completely chaotic”

Cellular automata: examples

- “Inverse“ rule: 124 (= 01111100)

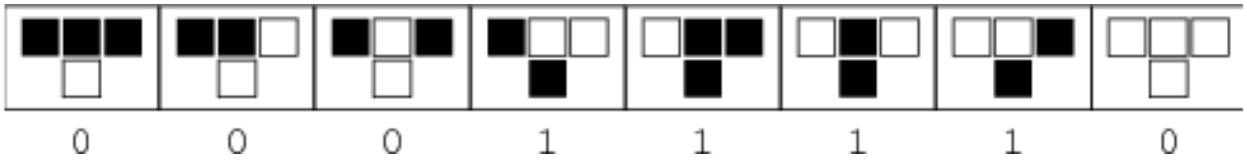
t=0	111	110	101	100	011	010	001	000
t=1	0	1	1	**1**	1	1	**0**	0

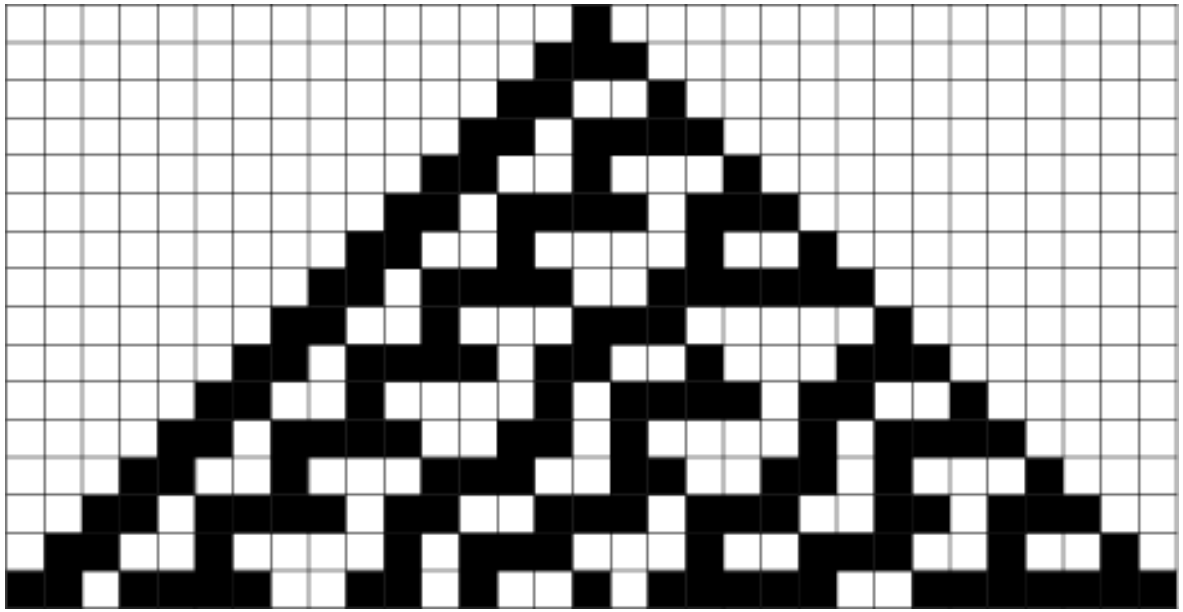


Cellular automata: examples

- Rule 30

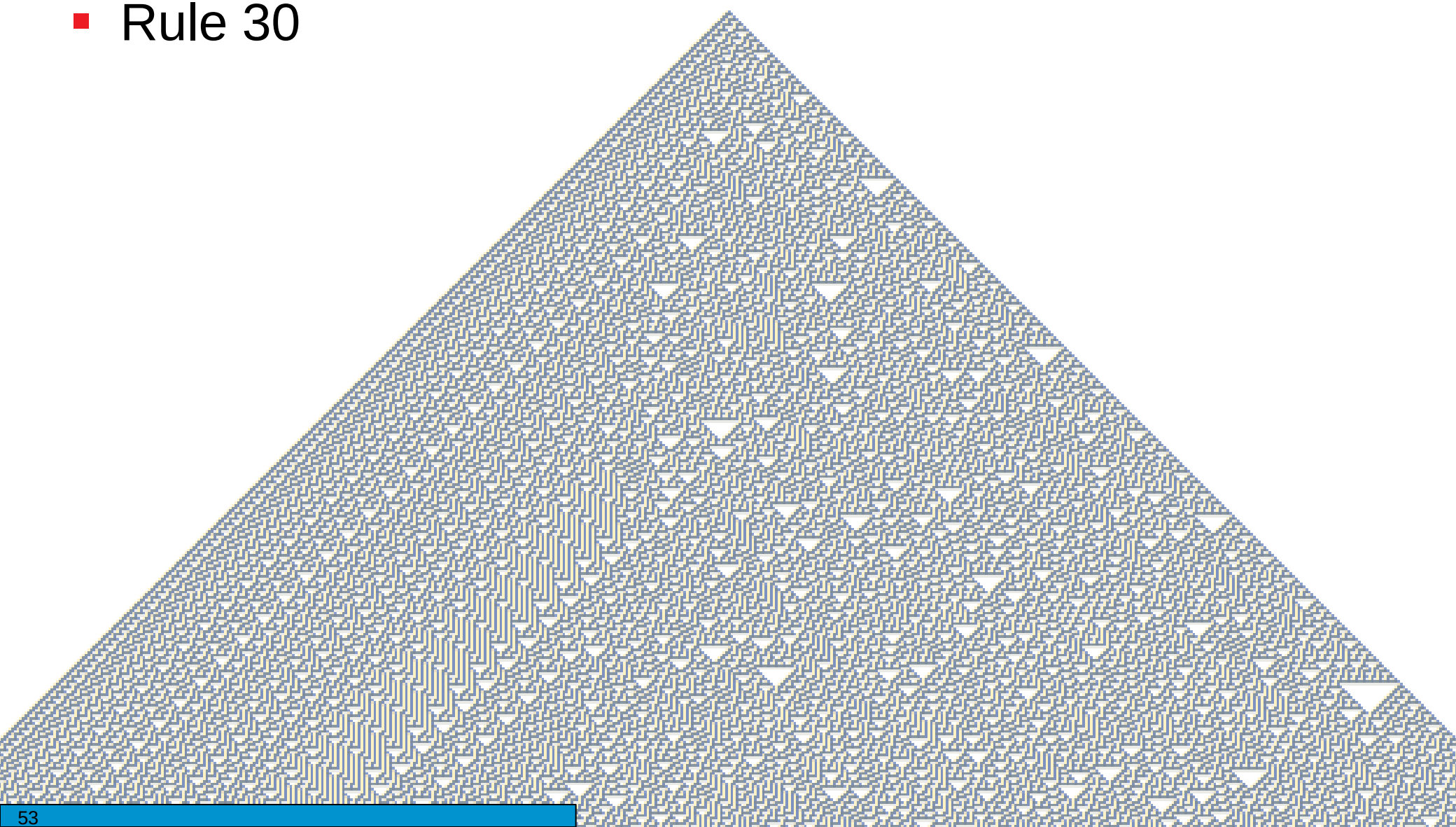
t=0	111	110	101	100	011	010	001	000
t=1	0	0	0	1	1	1	1	0





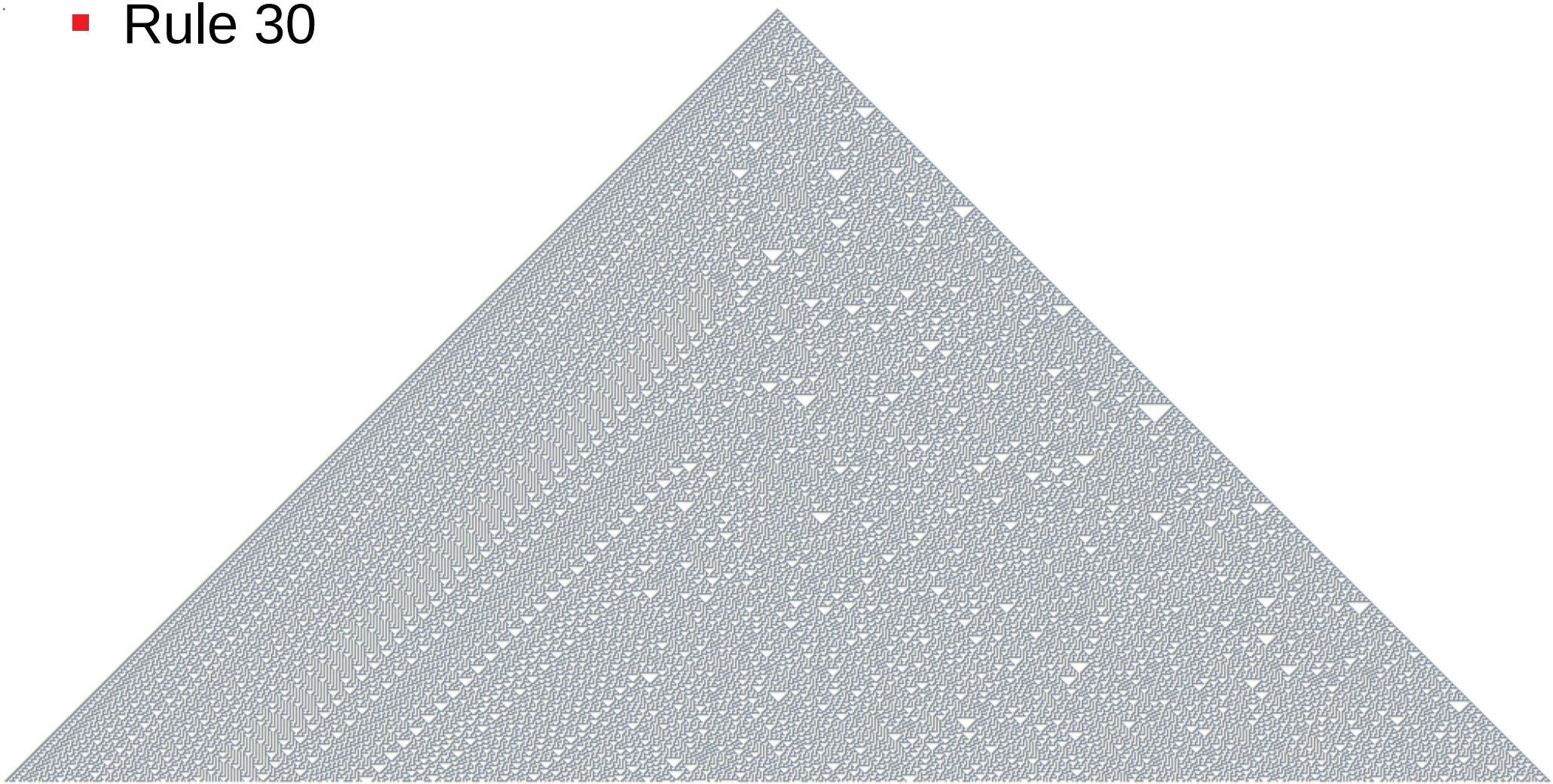
Cellular automata: examples

- Rule 30



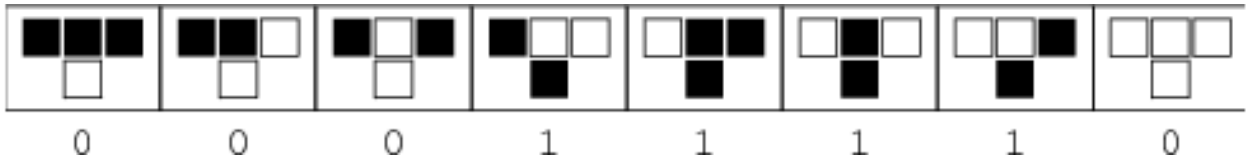
Cellular automata: examples

- Rule 30



Cellular automata: examples

- Rule 30



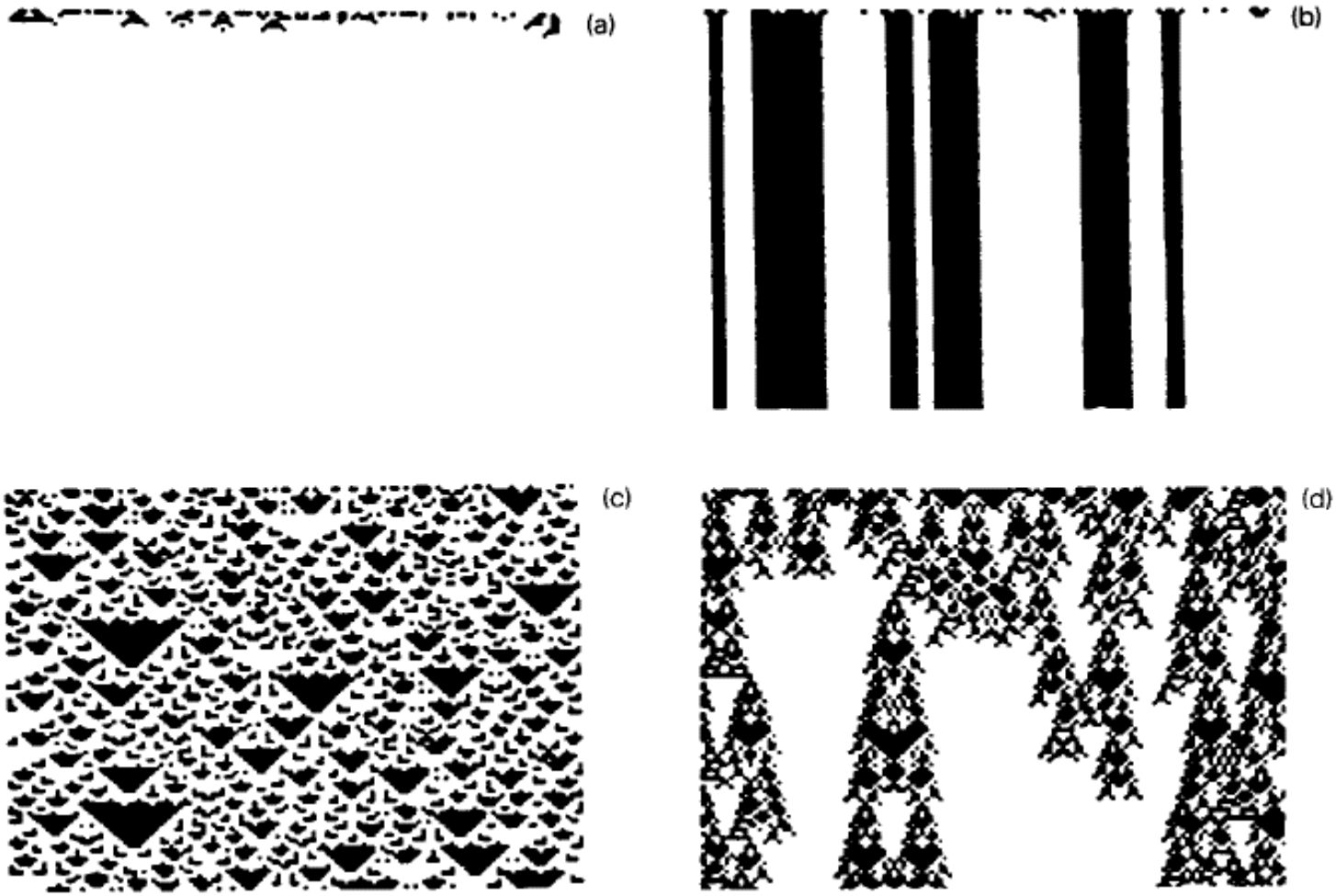
  - Generates seemingly randomness
  - Wolfram proposed center column as pseudo-random number generator
    - Used in Mathematica for random integers
  - Displays sensitive dependence on initial conditions (starting configuration) – butterfly effect

Behaviour of CAs

- Universal Computation (capable to perform any finite algorithm)

- Classification by Wolfram (L = probability of cell alive at next cycle)
  - Class 1: converges quickly into stable, homogeneous state, randomness in the initial pattern disappears
  - Class 2: Limit Cycles (0<< L < 0,3)
    - Stable/oscillating structures, some of the randomness remains
  - Class 3: Chaotic/Strange Attractors (L ~ 0,5)
    - fractal structure, stable structures are quickly destroyed by noise
  - Class 4: More Complex Behaviour (Univ. Comp.) (L ~ 0,3)
    - local structures that survive for long periods; Class 2 structures may be outcome, large number of iterations required

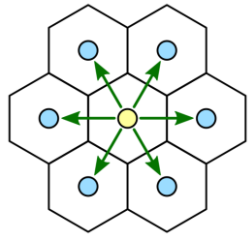
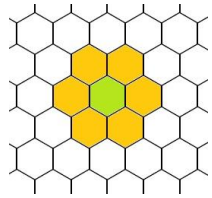
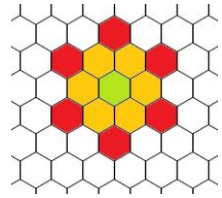
Behaviour – Illustration



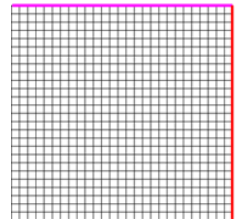
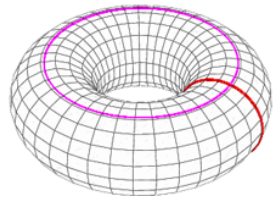
(Stephen Wolfram)

Cellular Automata: variants & extensions

- Hexagonal grid

- Torus architecture – neighbours at edges
  - Larger or asymmetric neighborhoods

- Longer memory
  - Regard state of neighbors not only in t, but also in t − 1, etc.
- More complex rules (not only count / sum)
  - **Stochastic** rules
- Multiple states (values)
  - Additional attributes (i.e. state variables)

Cellular Automata – Application Example

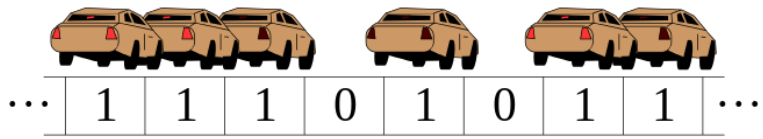
- Model for simulation of traffic flow
  - Implemented as CA
  - *Any ideas how?*

Cellular Automata – Application Example

- Model for simulation of traffic flow
  - Implemented as CA
- Rule 184 (Wolfram code)

t=0	111	110	101	100	011	010	001	000
t=1	1	0	1	1	1	0	0	0

  - *If a cell with value 1 has a cell with value 0 immediately to its right, the 1 "moves rightwards", leaving a 0 behind.*
  - Primitive model; "particle-hopping model"
  - Predicts some emergent features of real traffic
    - Clusters of freely moving cars separated by stretches of open road
    - Waves of stop-and-go traffic when traffic is heavy



  - *More advanced model?*

Cellular Automata – Application Example

- Nagel–Schreckenberg model for simulation of traffic
- Proposed in the early 1990s
- Idea: model the emergence of traffic jams
  - Also those that emerge when there is no capacity overload

- Implemented as CA
  - *How?*

- Road segments as cells
- Cars as state (occupied / empty)
- *More sophisticated update rules*

Cellular Automata – Application Example

- Road divided into segments
  - Length: one car + space between cars (~7.5 metres)
  - Initial model: road as a ring ➔ start & end segment connected
- Step: response time of a driver, e.g. 1 second

- Movement: 1-5 cells ahead (e.g. 27 – 135 km/h)
- Update rules:
  1. Accelerate vehicle speed by 1 (if not at max speed 5 cells)
  2. Break if there are not enough free cells ahead
  3. Reduce speed by 1, by a probability *p*
  4. Move vehicles forward

Cellular Automata – Application Example

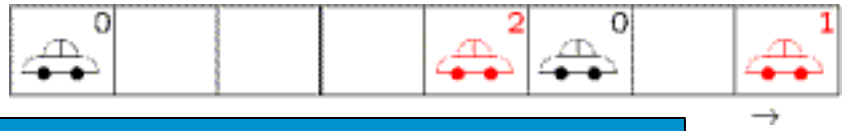
- Initial configuration at t=0

1. Acceleration by 1 – if road ahead is clear

2. Breaking if needed

3. Random deceleration

4. Movement – new state at t=1



Cellular Automata – Application Example

- Reduction of speed by random probability *p ?*
  - *Why is that needed / useful?*

- Models several phenomena
  - Drivers don't accelerate to maximum speed – at least not fast enough…
  - Drivers don't keep their speed constant at max
  - Drivers that need to break overreact – decelerate too much

➔ Makes the model non-deterministic

➔Allows emergence of traffic jam when not over capacity

Cellular Automata – Application Example

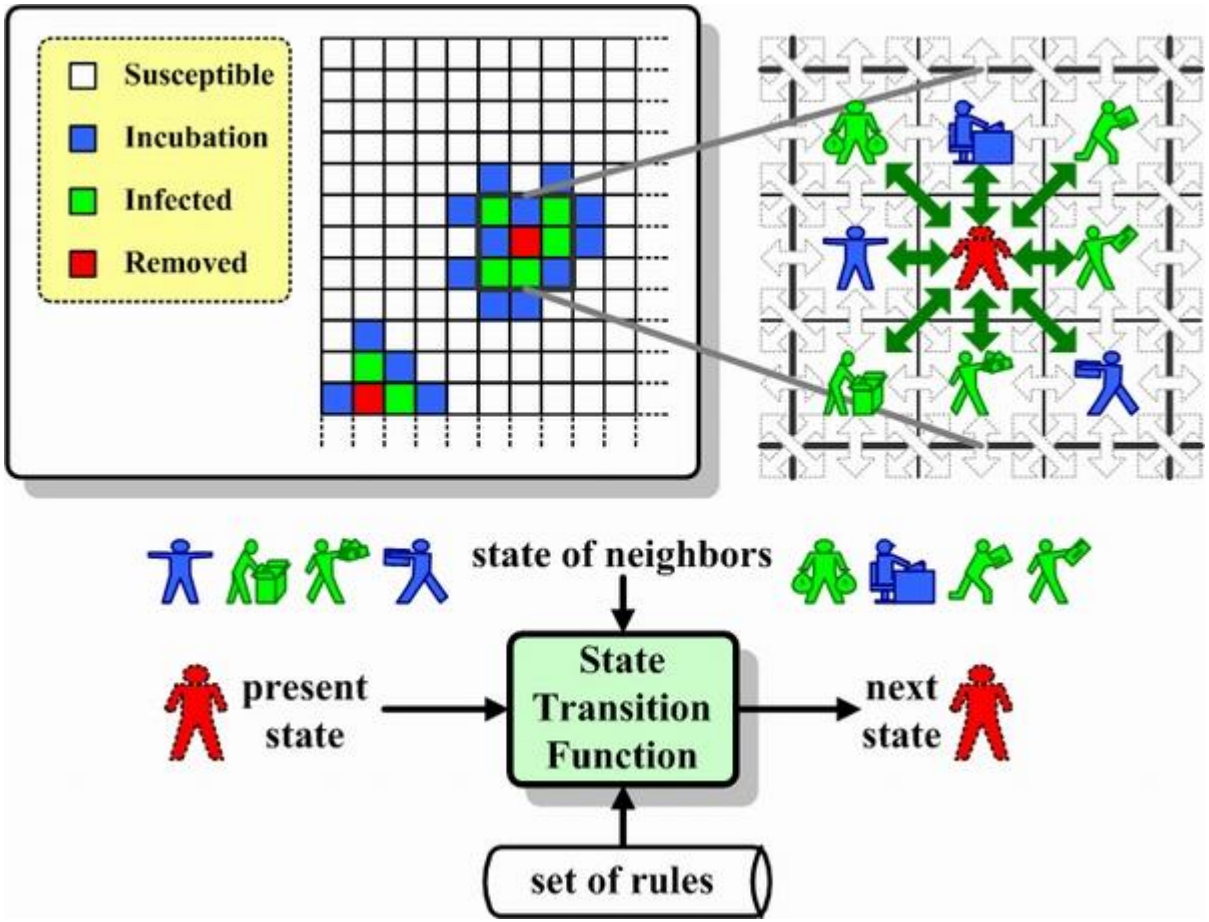
- Basic model extended and applied in:

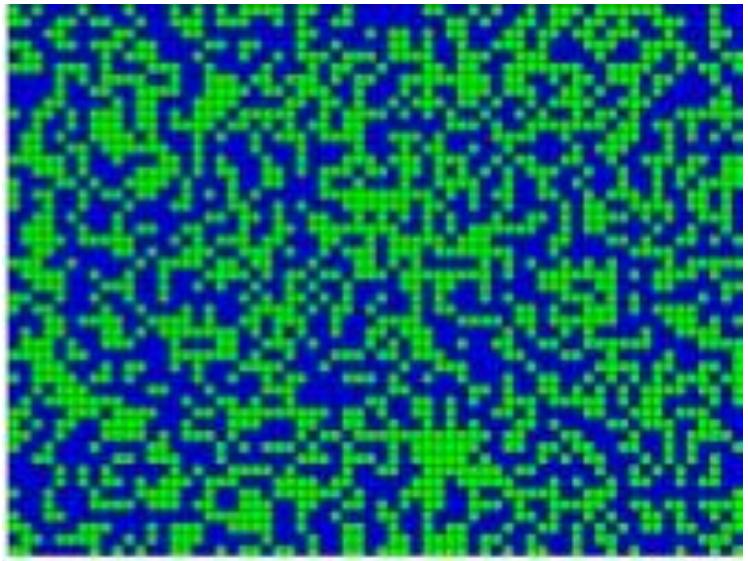
- German state of Nordrhein-Westfalen
  - Traffic forecast on motorways, http://www.autobahn.nrw.de/

- USA: Transportation Analysis and Simulation System (TRANSIMS)
  - http://code.google.com/p/transims/

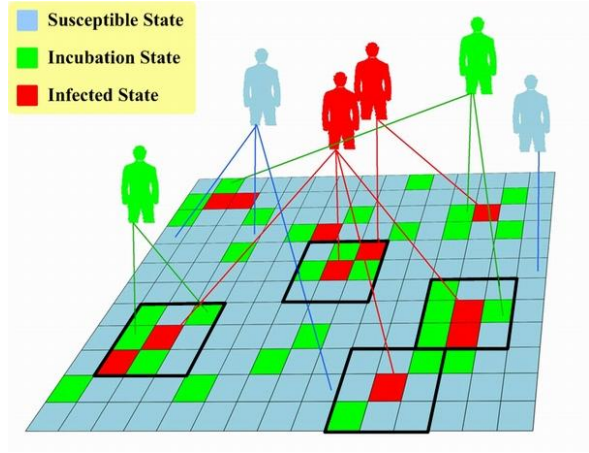
Cellular Automata – Application Example

- Modelling of disease epidemic



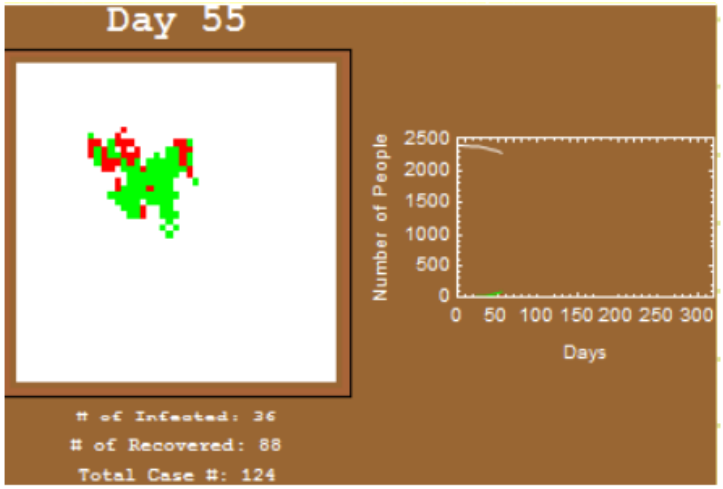


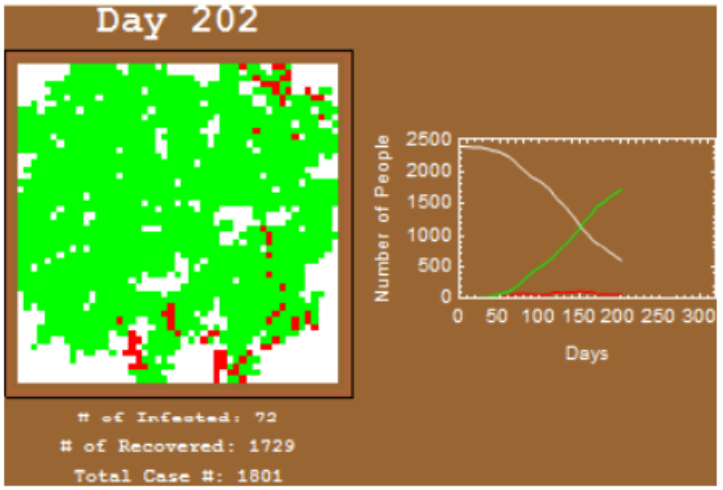
Cellular Automata – Application Example

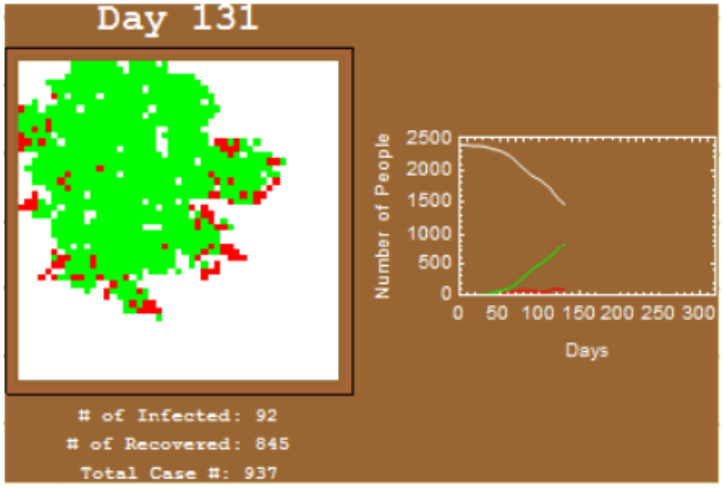


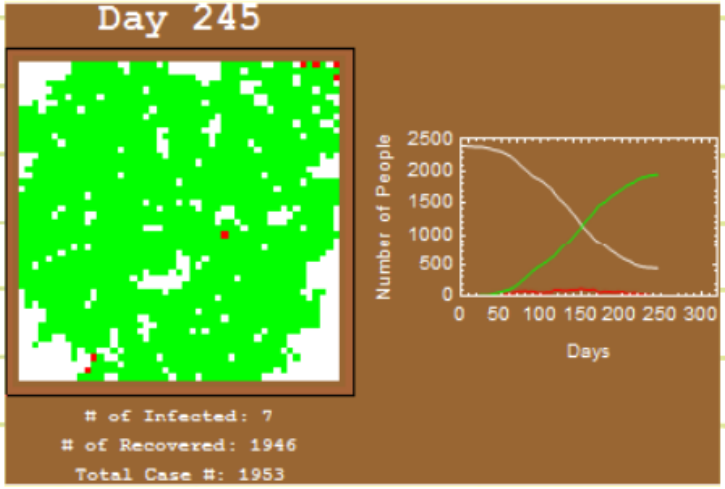
- Modelling of disease epidemic
  - Multiple cell states (in this example: four)
  - Transition rules e.g.:
    - incubation if a neighbouring cell is infected
  - Other variations:
    - No “incubation” status
    - Different levels (duration/states) of incubation
    - Leading to a recovered/“immune” status for that disease (if not fatal)
    - Also: vaccinated status
  - SIR (susceptible | infected | recovered)

Cellular Automata – Application Example



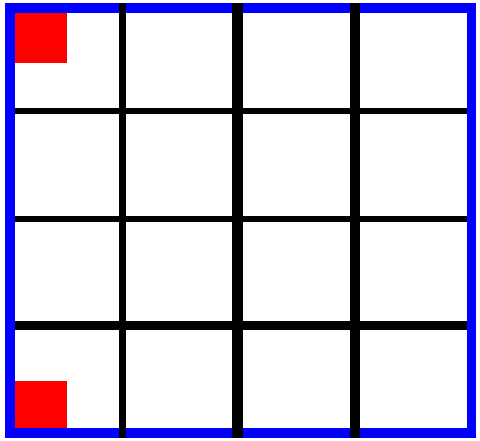






Other Application areas

- HPP, FHP - Gas Models
  - Lattice Gas Cellular Automata, simulate fluid flows
- Ising Model
  - ferromagnetism in statistical mechanics: magnetic dipole moments of atomic spins
- Self replication/reproduction
  - e.g. biological cells via cell division
- Chemical Waves
  - Belusov-Zhabotinsky Reaction (chemical oscillator/ chemical clock)
- Reaction-Diffusion Systems
  - Chemical reactions: transfer substances; diffusion: spread over a surface



Lattice Gas Cellular Automata (LGCA)

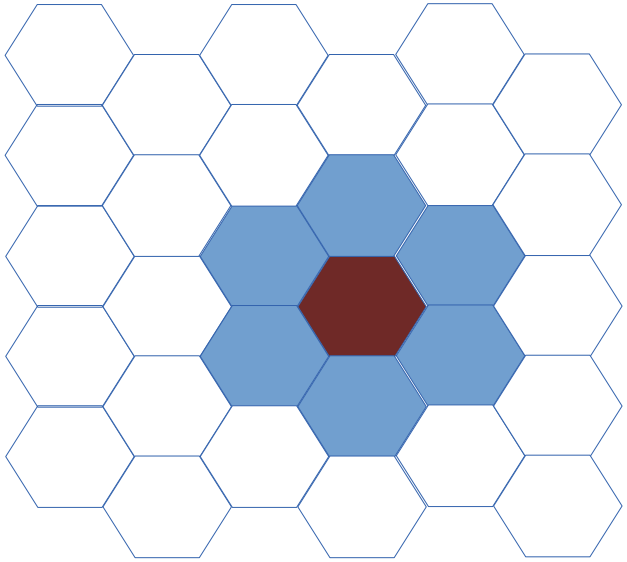
- Extension of the CA concept
- Intention: Simulate fluids and gases
  - Hardy, Pomeau & de Pazzis (HPP, square lattice), 1973
  - Frisch, Hasslacher & Pomeau (FHP, hexagonal grid), 1986

- Ideas
  - Cells do not have states but instead can contain particles
  - A particle can only proceed to a cell in the neighborhood
  - Instead of state updates, particles move to other cells
  - Particles represent the fluid or the gas

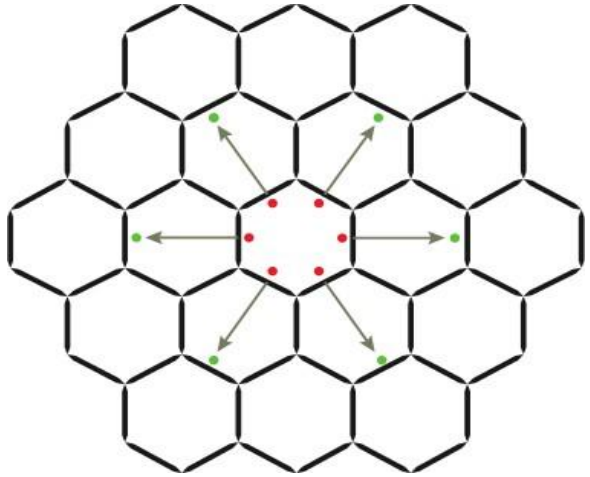
Lattice Gas Cellular Automata

- HPP
  - Square grid, Von-Neumann neighborhood
    - max. 4 particles per cell so that max. 1 particle goes to each neighbor
  - Several issues when it comes to real interpretations (comparison with real fluids, validation)
- FHP
  - Hexagonal grid
  - Neighborhood = surrounding cells
  - Max. 6 particles per cell, each going into a different direction → consistent definition
  - Corresponds to the Navier-Stokes-Equations → valid representation of fluid dynamics

FHP Model



neighbourhood

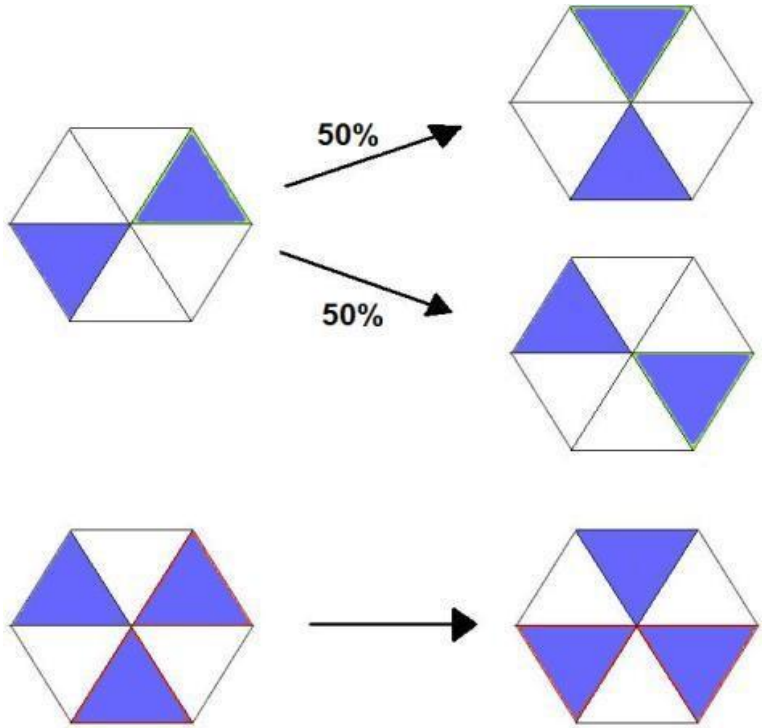


particles and directions

- Particle movements consist of two phases
  - Rotation of cells for special configurations
  - Movements of particles into their direction
- Further Developed by Wolf-Gladrow (2000)
  - Different variations (FHP-I, FHP-II, FHP-III)

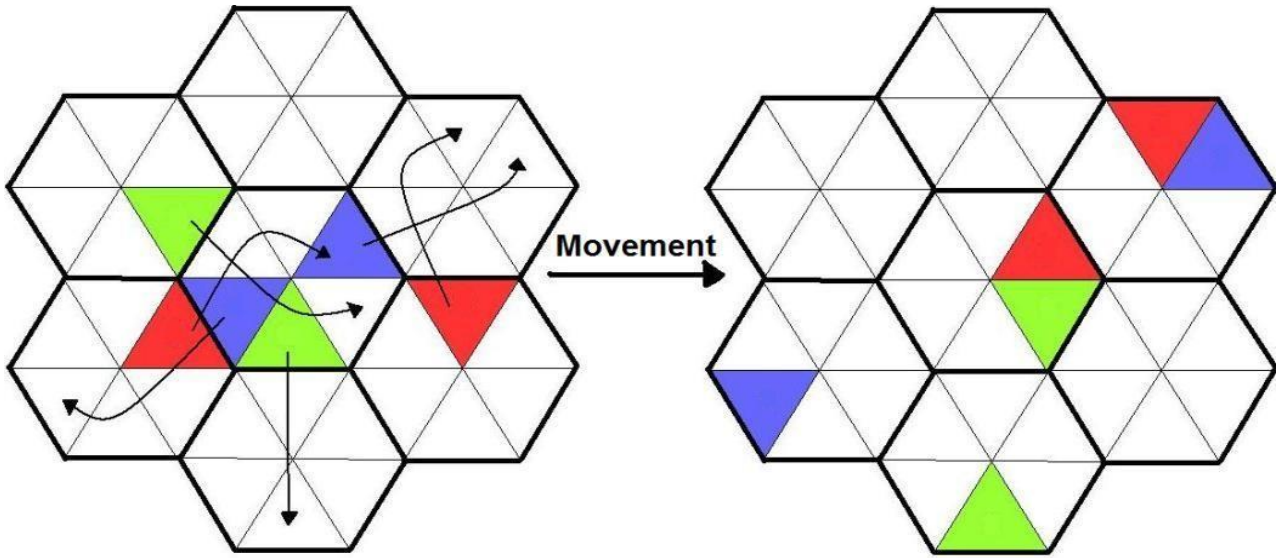
FHP Model

- Rotations
  - In the most simple case of FHP-I only for two situations
  - Provide a randomness



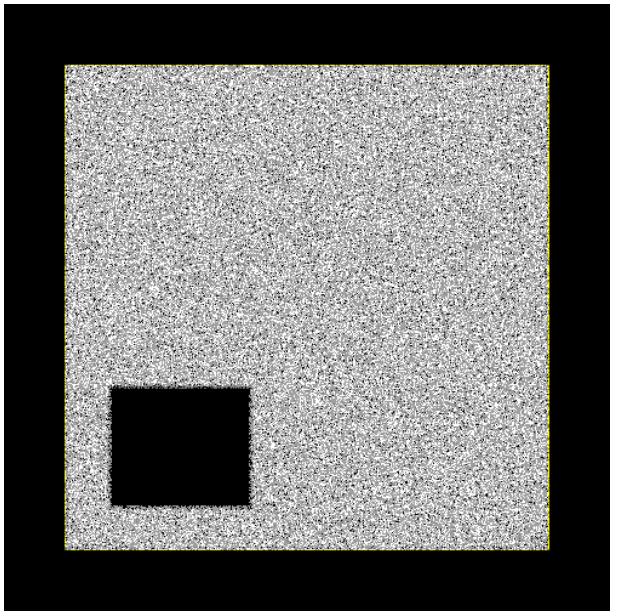
FHP Model

- Movements into designated directions



LGCAs: Simulations & Visualizations

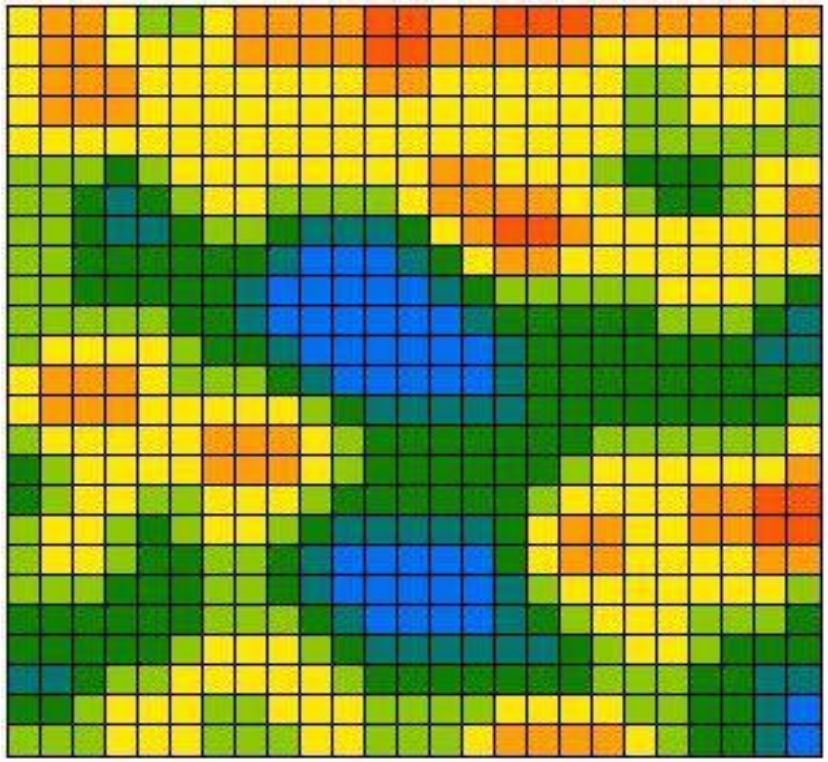
- HPP
  - http://en.wikipedia.org/wiki/File:Gas_velocity.gif



- FHP
  - http://www.youtube.com/watch?v=HluQpDFOceg
  - http://www.youtube.com/watch?v=00W6H7BGZ94

Simulation of social phenomena

- Opinion Dynamics via checkerboards

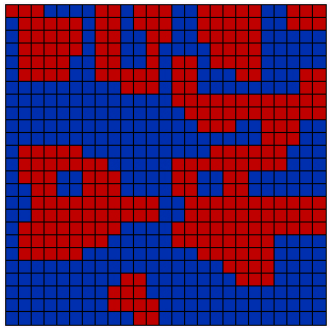


- Cells (individuals) have opinions $u_i(t) \in [0, 1]$
  - States of the cells
  - von Neuman neighborhood N_i
- Individuals update opinions by averaging over opinions of all their neighbours
  - Same weight is given to each neighbour
  - $u_i(t+) = 1/\#N_i \; \Sigma \; j \in N_i \; u_j(t)$
    - Discretized by step-function to # states
- Stewise updating of randomly selected cells
  - I.e. not cells at once, just one!

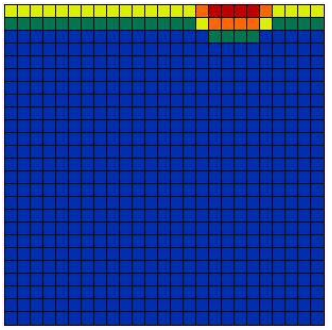
Rainer Hegselmann & Andreas Flache. “Understanding Complex Social Dynamics: A Plea For Cellular Automata Based Modelling”. Journal of Artificial Societies and Social Simulation. 1998

Simulation of social phenomena

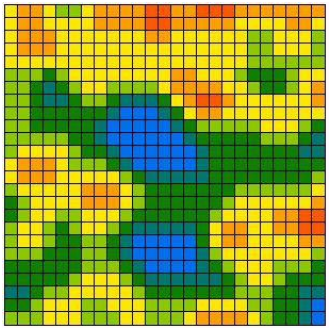
- Opinion Dynamics: Typical (stable) Results



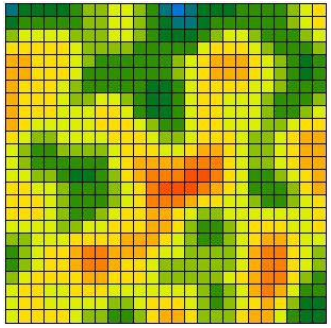
2 possible opinions



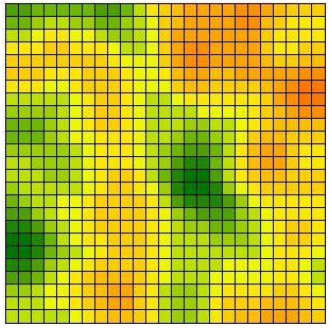
5 possible opinions



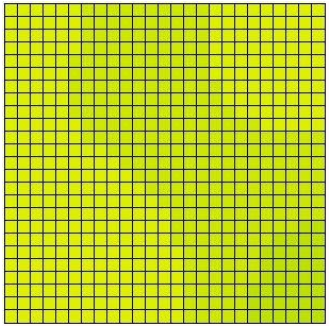
10 possible opinions



15 possible opinions



30 possible opinions

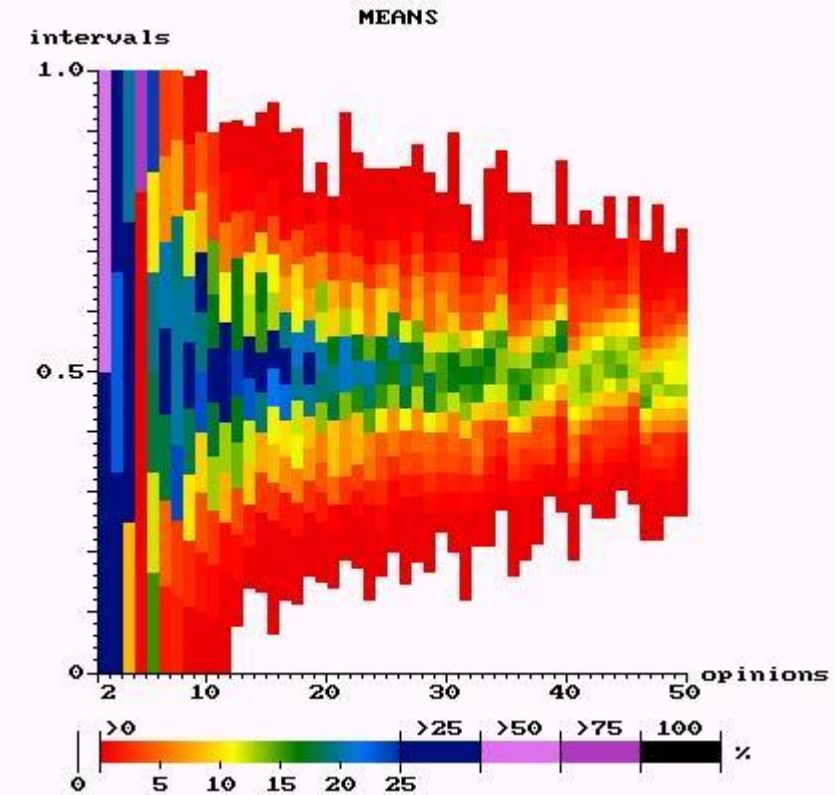


continuous opinions

- The more possible opinions: the closer stable results get to universal consensus

Simulation of social phenomena

- CA reach stable state
- Micro rules lead to macro patterns
- More options: more consensus
- Extreme opinions disappear
- Discretisation matters



Simulation of social phenomena

- CA based modelling of social dynamics
- CA models cover basic features of a significant class of social processes in the real world
  - Over a period of time numerous people interact, interactions and information are local and the neighbourhoods of interaction overlap
  - System that is based on locality and overlapping interaction structures
- Shows how certain macro effects are dynamic consequences of decisions and mechanisms operating only at micro level
- Order, structure, clustering and segregation are all generated by such micro level rules
- Essential feature of CA: discretisation of space, time and state

- Qualitative behaviour of discrete dynamics may be quite different from continuous case
  - If carried out incautiously, CA based modelling may produce artefacts!

CA & Checkerboard models

- CA
  - Individuals with heterogeneous states
  - But homogenous transition rules
  - No movement possible, only passive “interaction” between individuals
- Checkerboard models (Hegselmann 1998)
  - Advancement of CA models, precursor/simple form of Agents
  - Movement of individuals is possible
    - Checkerboard migration models vs. steady-site models (CA)
  - Multiple, varying neighborhoods possible
    - e.g. small interaction, medium migration, larger information windows

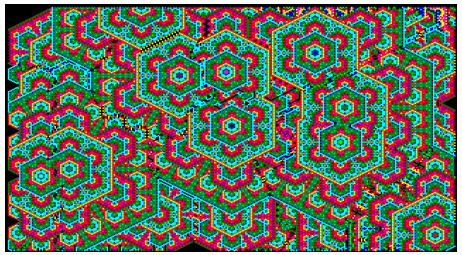
Application areas: categories

- Some problem-solving areas
  - Random number generation
  - Cryptography (rule 30 as block cipher)
  - (Traffic modelling)
  - ...
- Modelling of all kinds of natural phenomena
  - Especially in the earlier days, when micro-processors became available (thus e.g. popularity of game of life)
- Modelling of social phenomena

CA – Significance and Usage

- “Good at investigating the outcomes at the macro scale of millions of simple micro-scale events“
  (Gilbert, Troitzsch)

- Can be calculated very fast due to simple structure
  - Computational power no issue anymore
- More elaborate techniques available (Agents)
  - Differentiation not always clear
- Still used in a number of sciences
  - Mathematics, Informatics, Biology
- Computer-generated Art



CAs today

- Rich Theoretical Results (Automata Theory)
- Several advantages, but – did CAs replace Differential Equations in Modelling... ?
- Disadvantages of CAs
  - Simple Rules but: How to find the ***Right Ones?***
  - Scaling Problems
  - Various Practical Problems (Constant # of Elements...)

Agent Based Systems

- (Autonomous) Software Agents

- New Software Engineering Paradigma
- Agent Modelling

- Differences to CAs
  - Not the Lattice is in the centre but
    - The Individual and the
    - Interaction, which can be more Complex and “Realistic”

Further reading

- „It‘s Alive“: The Coming Convergence of Information, Biology, and Business“, Christoph Meyer, Stan Davis

- „Complexity: The Emerging Science at the Edge of Order and Chaos“, Michael Mitchell Waldrop

Further reading

- Gerhardt M., Schuster H. (1995), Das digitale Universum - Zelluläre Automaten als Modelle der Natur, Vieweg, Braunschweig/Wiesbaden
- Gardner M. (April 1970), The Fantastic Combinations of John Conway's New Solitaire Game of "Life", Scientific American, 223:4, 120-123
- Dewdney A.K. (August 1989), A Cellular Universe of Debris, Droplets, Defects and Demons, Scientific American, 261:2, 102-105
- Dewdney A.K. (January 1990), The Cellular Automata Programs That Create Wireworld, Rugworld and Other Diversions, Scientific American, 262:1, 146-149
- Riedl R. (1985), Evolution und Erkenntnis, Piper, München
- Riedl R. (1986), Die Strategie der Genesis, Piper, München
- Eigen M., Winkler R., Das Spiel. Naturgesetze steuern den Zufall, Piper, München (1985)
- Kauffman S., At Home in the Universe: The Search for Laws of Self-Organization and Complexity, Oxford University Press (1995)

Agenda

- Introduction & Definitions of Self-organising Systems
- Cellular Automata
- Genetic algorithms & Genetic programming
- Ant Colony Optimisations
  - *to be extended in swarm optimisation lectures*
- Agent Based Systems / Multi Agent Systems

About Optimisation & Learning

- Learning as Optimisation

- Optimisation-Methods
  - *Analytic*
  - Deterministic
  - Random
  - Brute Force
  - ***Evolutionary***

Optimisation examples

- Minimize Effort
  - Travelling Salesman (GA & Ant)
- Calculate optimal parameters of mathematical models
  - Time-series analysis
  - Pattern matching
- Maximize Yield

- Finding models
  - Curve fitting
  - Traffic routing
  - Pattern matching

Main Categories of Optimisation Problems

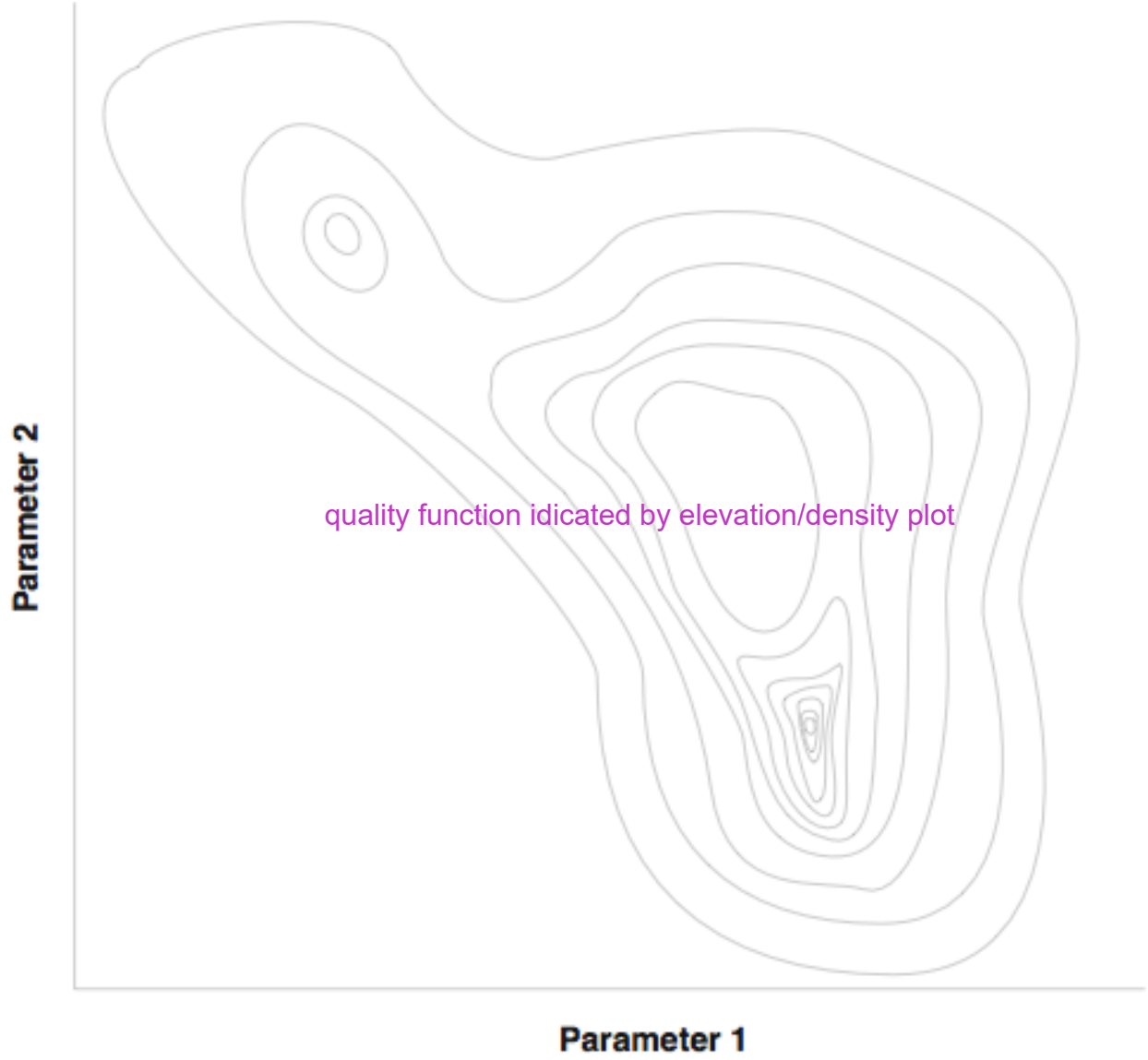
- Parameter Estimation
  - Mathematical models
- Subset Selection
  - Select a subset of a larger set
- Combinatorial problems (Sequencing problems)
  - Travelling Salesman
- Model Finding

Quality Function and Phase Space

- A ***Quality Function:*** a function that allows to assign a quantitative quality measure to a specific set of parameters, or a specific configuration or status of a system

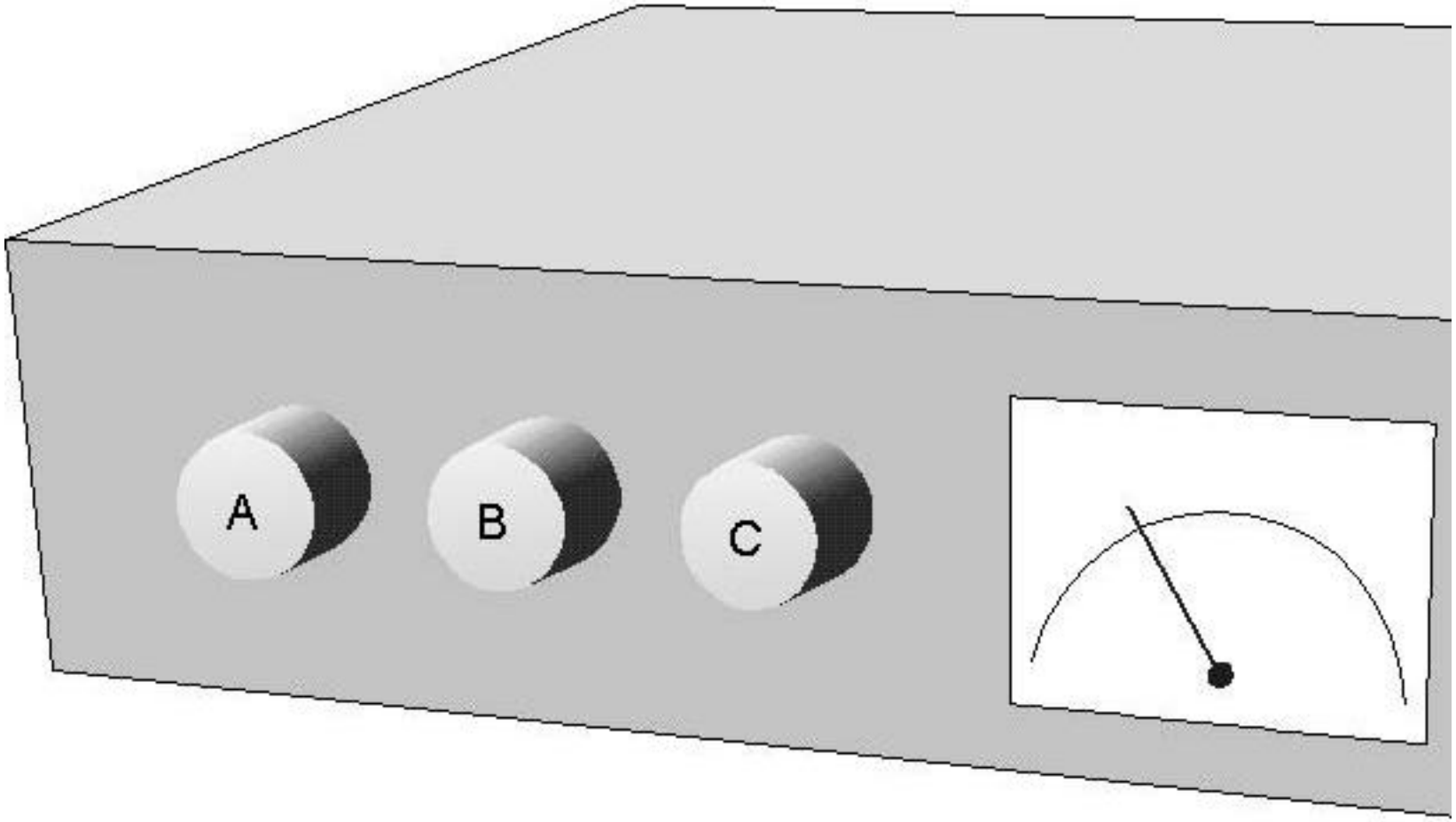
- *Phase Space is the n-dimensional space that is spanned by the relevant parameters of a system*

Phase space: simple example



„Black Box“

- In optimisation, model often treated as black box



Various Optimisation Methods

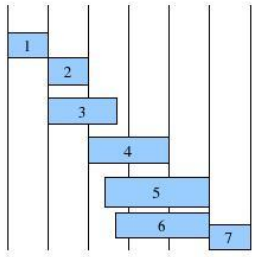
- Full Grid Search (Brute Force)
- Hill Climbing (deterministic)
  - Simplex
  - Gradient Method (e.g. Gradient descent)
- Radom Search (non-deterministic)
  - Monte-Carlo Methods
- Combinations
  - Evolutionary Algorithms
    - Genetic Algorithms

Genetic Algorithms Applications

- Generally heuristic optimisation / search tasks
  - Often NP (hard) problems

NP = wherer brute force is not possible

  - Scheduling (job scheduling) – ideal assignments of jobs to resources at particular times
  - Timetabling



  - Travelling Salesman – finding the optimal route

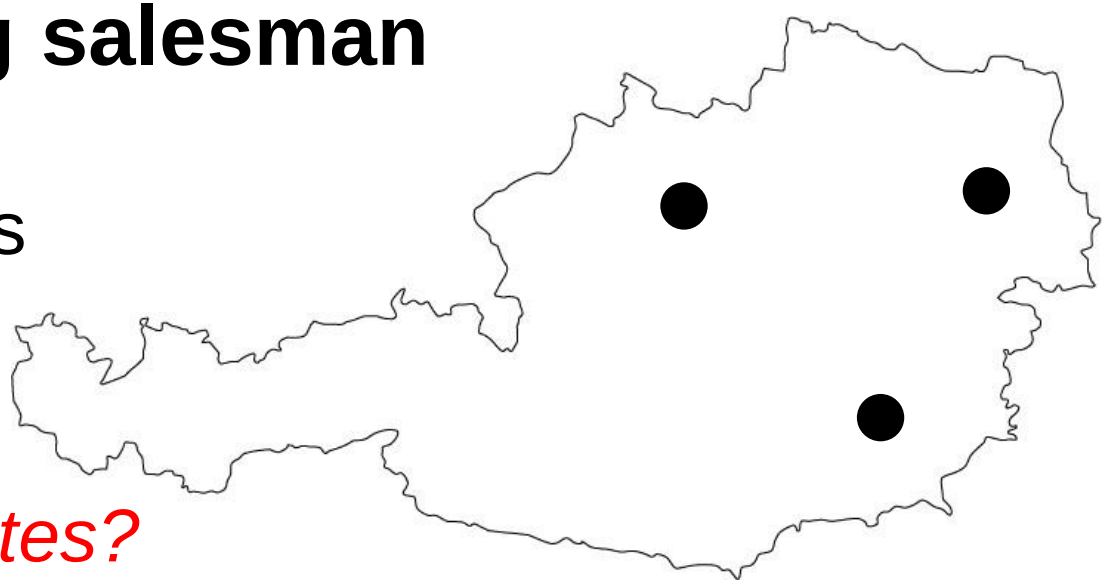
  - Economics (e.g. supply/demand model, using GAs to determine decisions for next period)

GA: Application example

- Travelling salesman problem
- Task
  - Given a set of locations, visit all locations in the optimal route
  - Optimal: shortest distance / fastest travel time, ...

- *Why is this difficult?*

GA: Travelling salesman

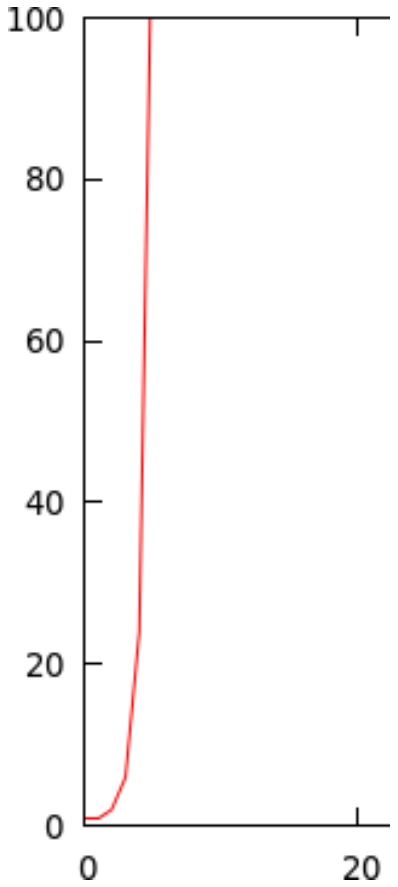


- Simple example: 3 cities
  - E.g. Vienna, Graz, Linz
- *How many possible routes?*
  - 3 starting places (not a given starting point)
    - 2 options for second stop
    - (1 option for last stop)

  ➔ 6 different routes
    - (Would be less if starting & end city are fixed)
- *General number of routes?*
  - Factorial: $n! = n \times (n-1)!$
    - $3! = 3 \times 2 \times 1 = 6$

GA: Travelling salesman

n	n!
1	1
2	2
3	6
4	24
5	120
6	720
7	5,040
8	40,320
9	362,880
10	3,628,800
20	2,432,902,008,176,640,000
50	$3.041409320 \times 10^{64}$



- Exhaustive search becomes infeasible
  - np-hard
- Heuristic solvers applied
  - Evaluation of goodness is fast
  - Find “good enough” solution, fast -> quality function
  - Can’t verify if optimal

GA: Travelling salesman

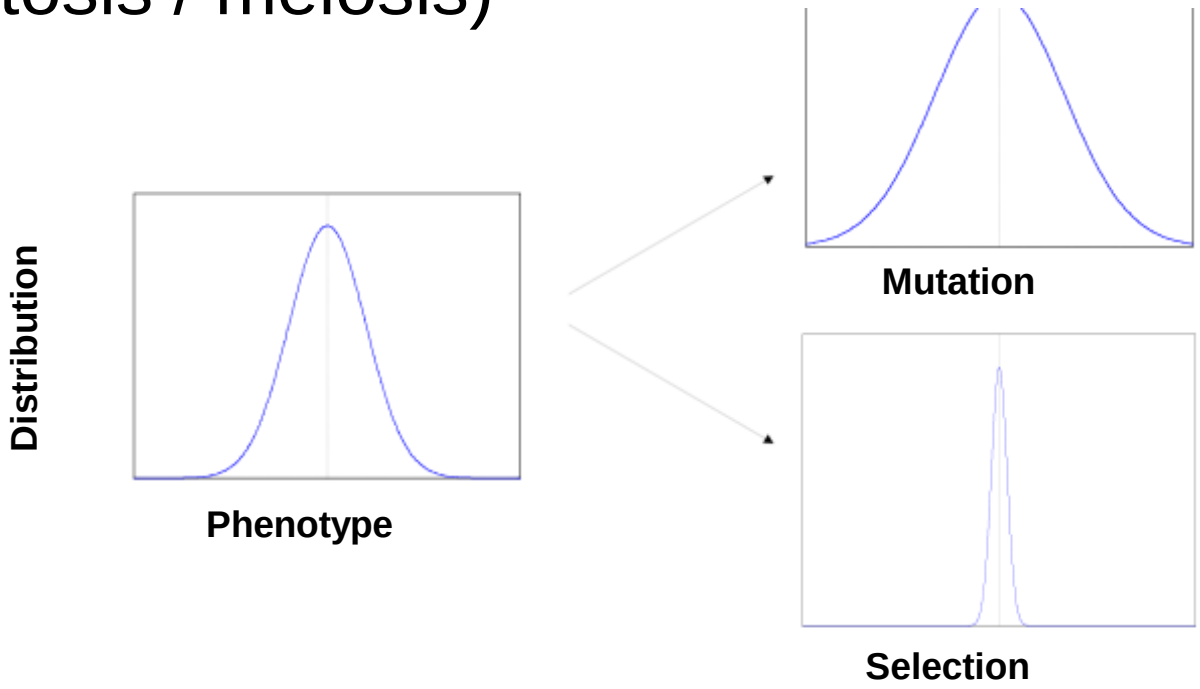
- Formulating Travelling salesman as GA
  - *Encoding?*
    how to present as genetic alogrythm
  - *Initialisation?*
    whats the first solution
  - *Fitness function?*
    makes sebnse in my scenario
  - *Restrictions on genetic operators?*

Genetic Principles

- Phenotype (observable characteristics) determined by the information coded in genes (plus the environment)

- Cell division (mitosis / meiosis)
  - Reproduction
  - Crossover
  - Mutation

- Selection



Genetic Algorithms: principles

need to create

- Candidate solutions (individuals, phenotypes)
  - Set of properties (chromosomes, genotype) represented by genes
  - ➔ Coding

- Algorithm principles improve candidate solutions by phenomenes abserved in biology:
  - Selection
  - Mating
  - Crossover
  - Mutation

- Evaluation of goodness: fitness function

Genotype and Phenotype

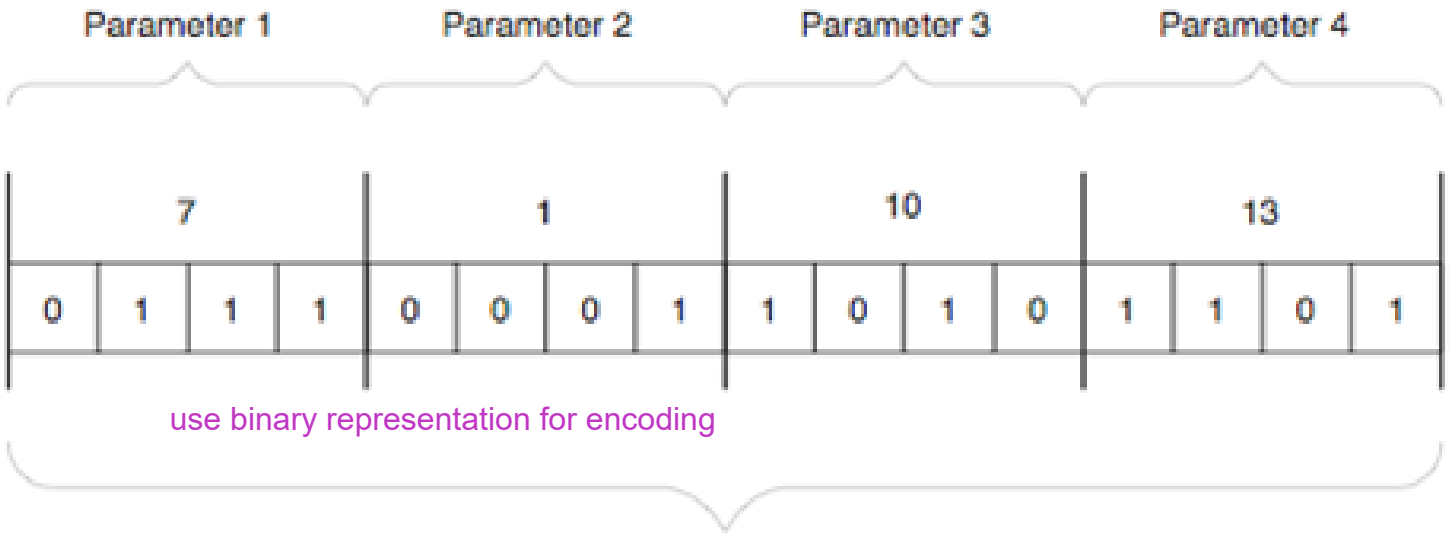
- "Genotype" is an organism's full hereditary information
  - The set of genes representing the chromosome

- "Phenotype" is an organism's actual observed properties
  - The actual physical representation of the chromosome

Phenotype representation and...

Parameter 1	**0,07**	**(7)**
Parameter 2	**0,01**	**(1)**
Parameter 3	**0,1**	**(10)**
Parameter 4	**0,13**	**(13)**

-> 0-16 /4bits for binary representation



-> 1 candidate solution

caveat of binary representation:
big step from 7 -> 8
0111 vs 1000. instead use greycode

-->create multiple candidate solutions and improve

... Coding Options like ...

- Binary: chromosome represented as a string of 0 & 1s
  - Most common → first works about GA used this encoding
  - Binary vs. Gray Code
- Permutation: Useful for ordering problems such as travelling salesman problem
  - String of numbers, representing a number in a sequence.
- Value: The actual value is encoded as it is
  - For complex information, e.g. real numbers, weights of a NN, ..
- Tree

... Graycode

Decimal (phenotype)	Binary (genotype)	Gray (genotype)
0	000	000
1	001	001
2	010	011
3	011	010
4	100	110
5	101	111
6	110	101
7	111	100

Advantages of Gray Code? changes depicted in graycode similar effect as phenotype. bit by bit vs many bits in binary

Graycode

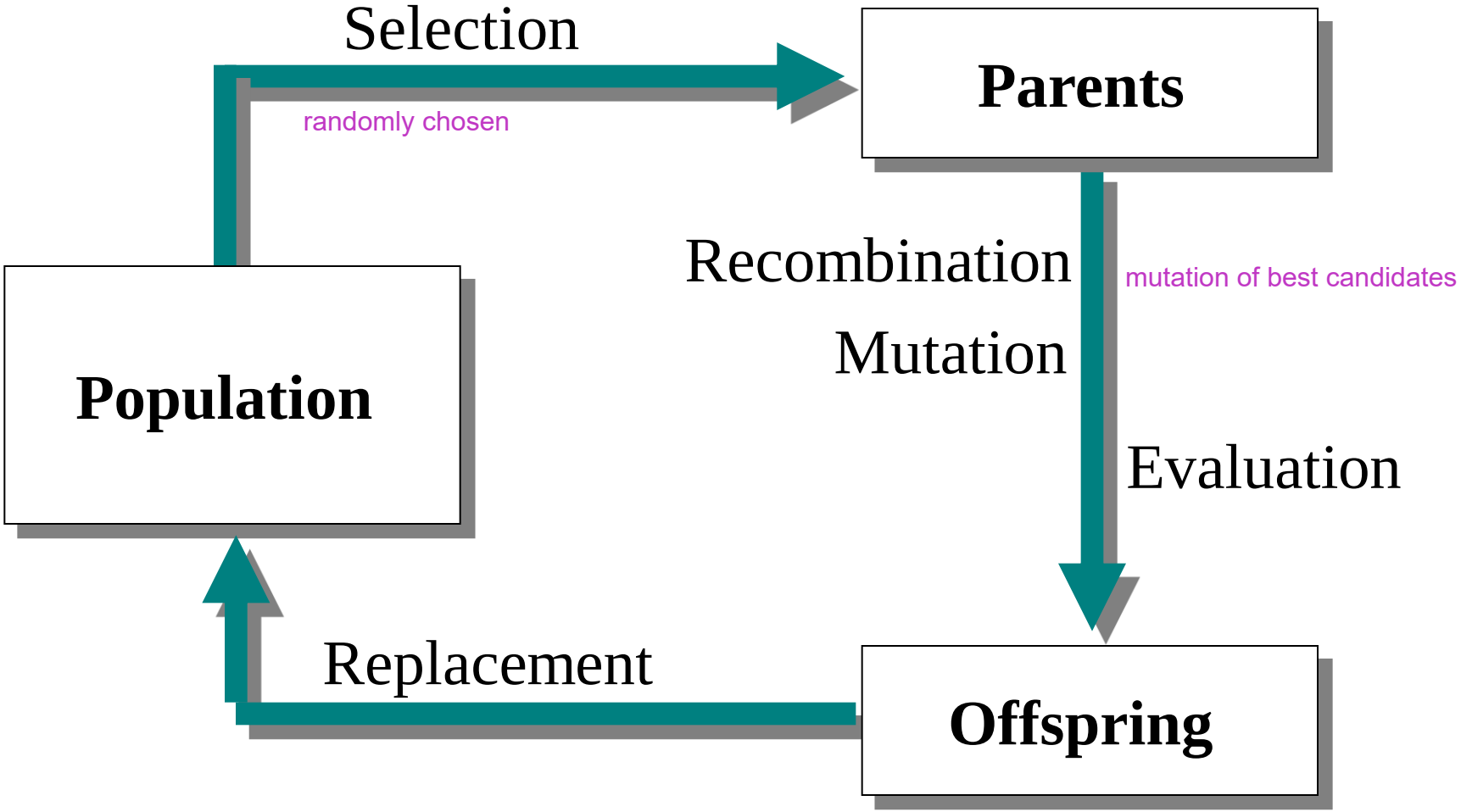
- Reflected binary code
  - changing only one bit at a time
- Thus better to trace transition between states
- Hamming distance of 1 between adjacent codes
  - *Hamming distance: number of positions at which symbols are different*

Representation: Example

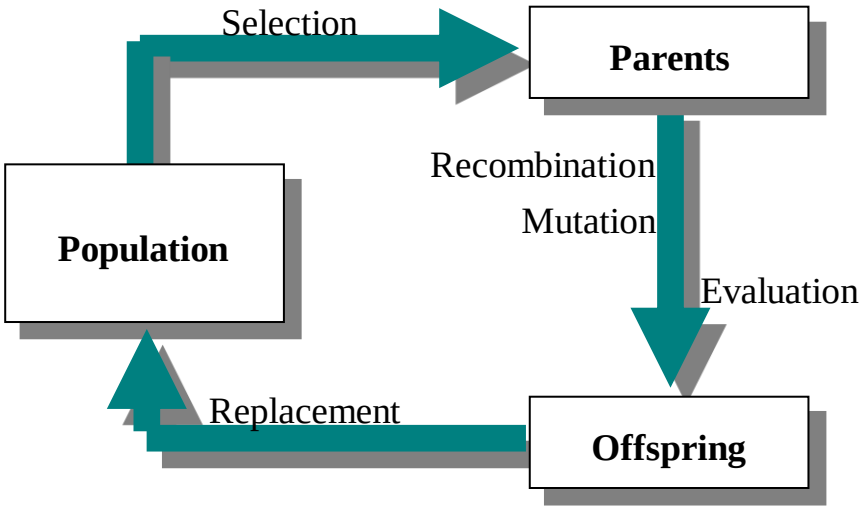
- Function optimization
  - maximize function $f(x,y) = x^2 + y^2$ = fitness function
  - on the interval of integers $\langle 0, 31 \rangle$
  - x and y are coded on 5 bits
    - *(could be solved analytically, but an easy example to illustrate..)*

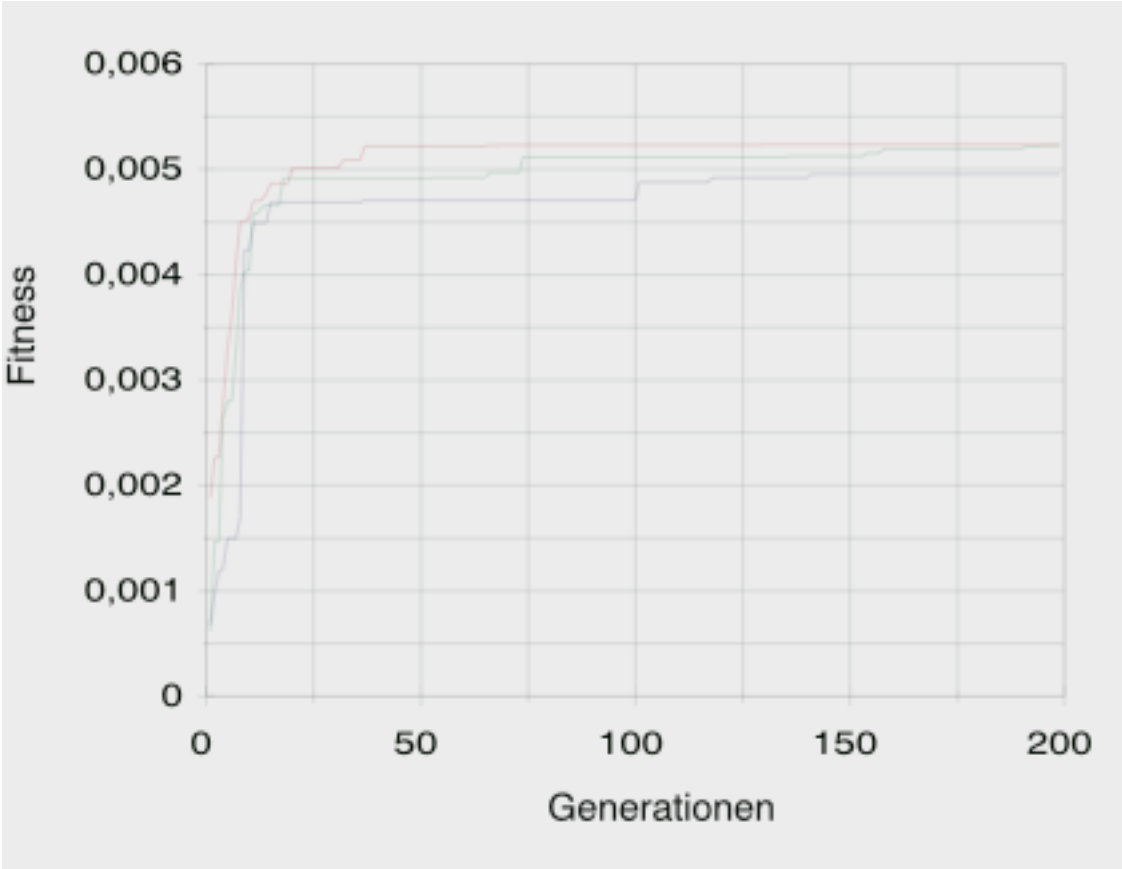
genotype	phenotype	fitness
0 0 0 0 0, 0 1 0 1 0	0, 10	100
0 0 0 0 1, 1 1 0 0 1	1, 25	625 + 1 = 626
0 1 0 1 1, 0 0 0 1 1	11, 3	121 + 9 = 130
1 1 0 1 1, 1 0 0 1 0	27, 18	729 + 324 = 1053

Evolutionary cycle



Evolutionary cycle





Mutation = random

- Alters one (or more) gene values
  - Can help GA to avoid local minima
    - Preventing population of becoming too similar to each other

    ➔ preserving and introducing diversity

- Bit string mutation
  - Flip bit at random positions

i fin graycode, one flip is a small change in phenotype

1 1 1 0 1 0 1

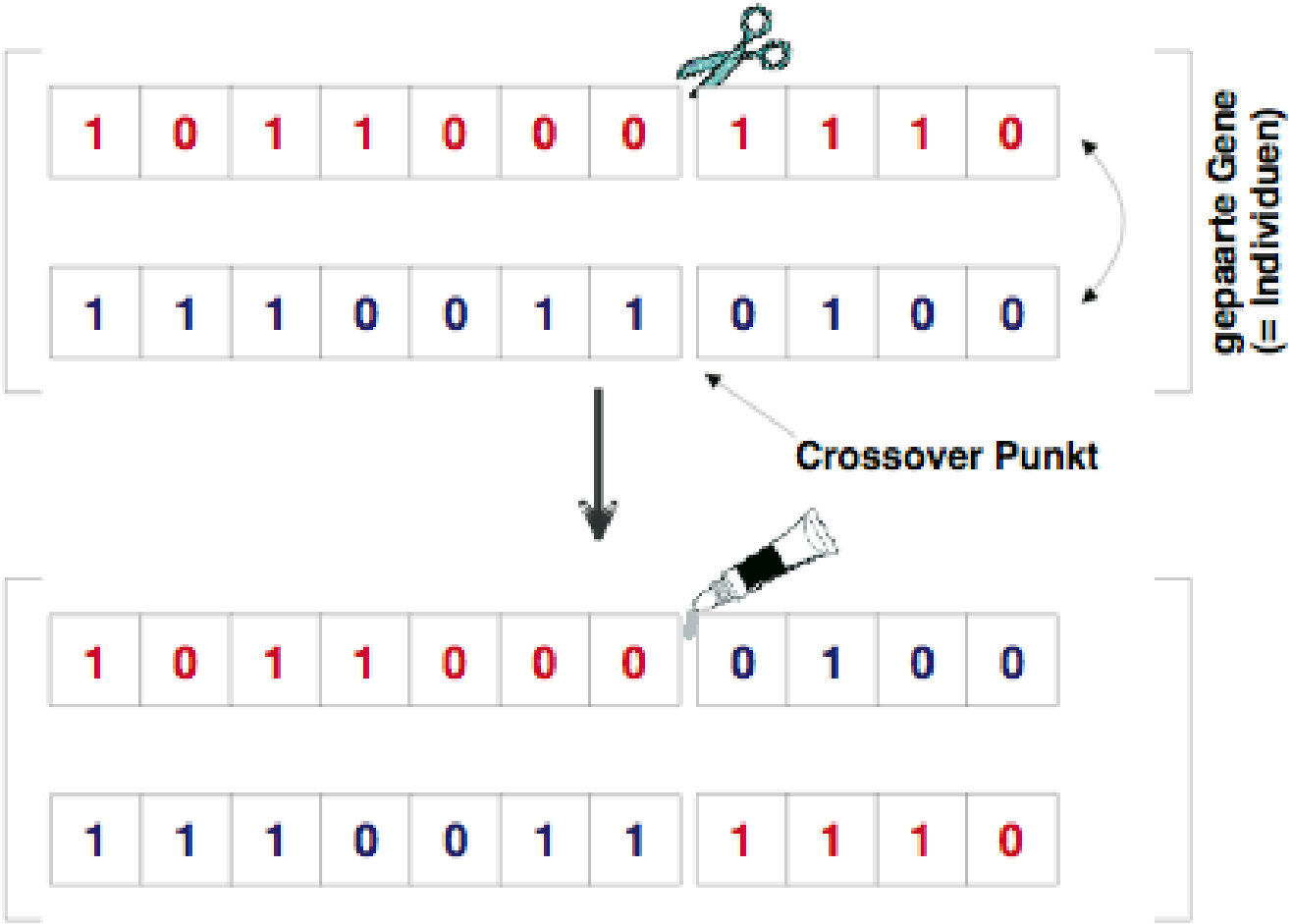
↓

1 0 1 0 1 0 1

Crossover / reporoduction

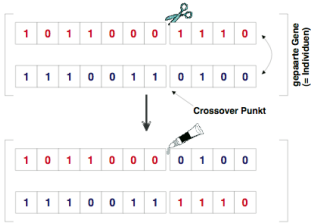
- One Point Crossover

crossover of two different individuals



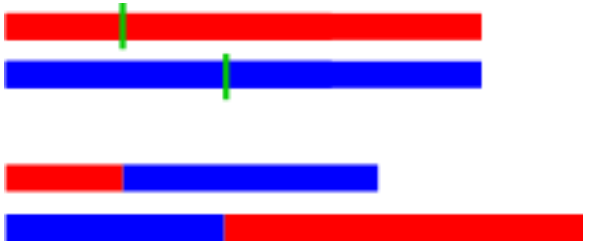
Crossover variations

- One Point Crossover



- Two Point Crossover

- Cut and splice
  - Changes length

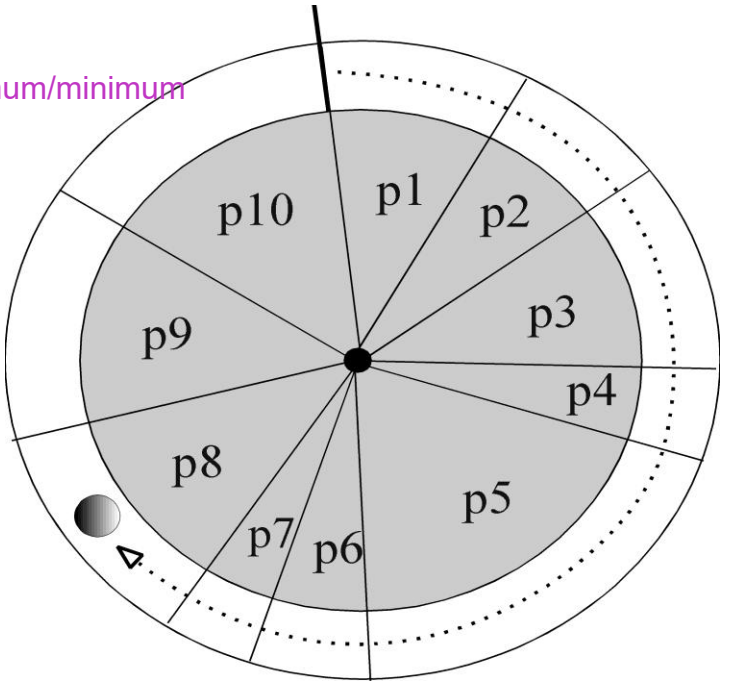


- Uniform Crossover
  - Mixing ratio

Selection

- Models the principle of „surviving the fittest“ observed in nature
  - favours the fitter individuals, suppresses the weak ones
  - each individual should have at least some chance to pass to the next generation
- Roulette wheel <avoid caveat. might reach a local maximum/minimum
  - probability P_i of selecting i-th individual is proportional to its fitness f_i

$$P_i = \frac{f_i}{\sum_{j=1}^{PopSize} f_j}$$



- Other selection schemes
  - Stochastic universal sampling, ***Tournament selection***, Reminder stoch. sampling, Rank-based sel.

GA: Travelling salesman

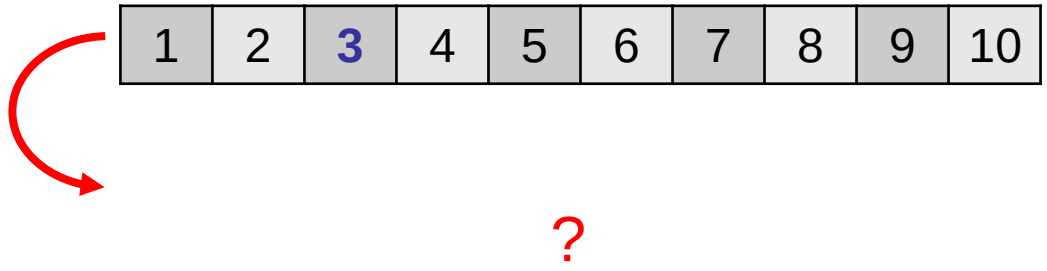
- Formulating Travelling salesman as GA
  - *Encoding?*
    - Simple approach: list of cities (incremental identifiers)
  - *Initialisation?*
    - Random order of cities
    - Needs to **contain all** cities!
  - *Fitness function?*
    - Inverted distance of route
    - E.g. 1/distance + roulett wheel
  - *Restrictions on genetic operators?*
    - New solution still needs to **contain all** cities

      ➔ Mutation?

      ➔ Crossover?

GA: Travelling salesman

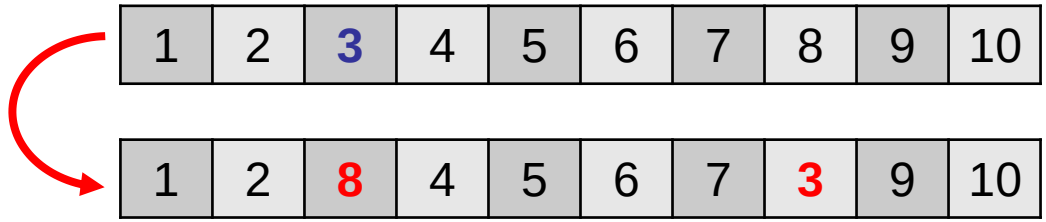
- New solution still needs to contain all cities
- Mutation?
  - ?



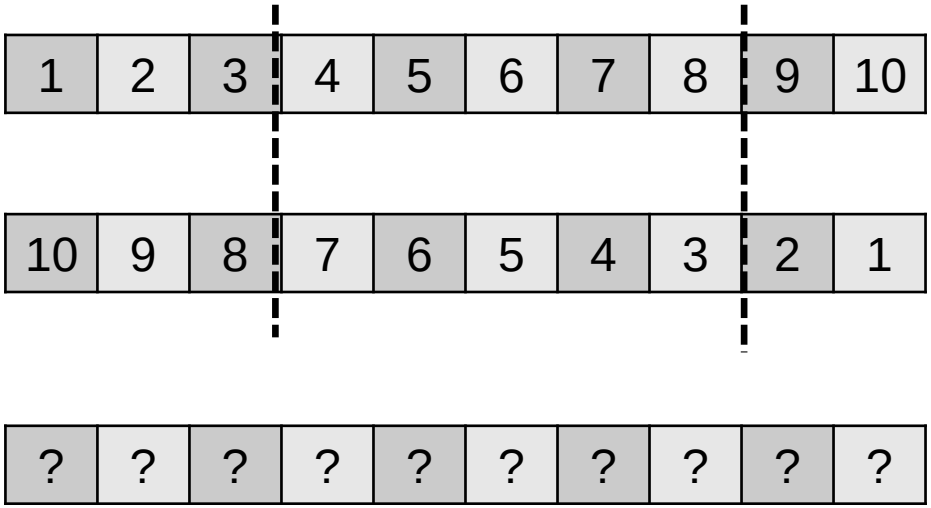
- Crossover?
  - ?

GA: Travelling salesman

- New solution still needs to contain all cities
- Mutation?
  - E.g. swapping positions

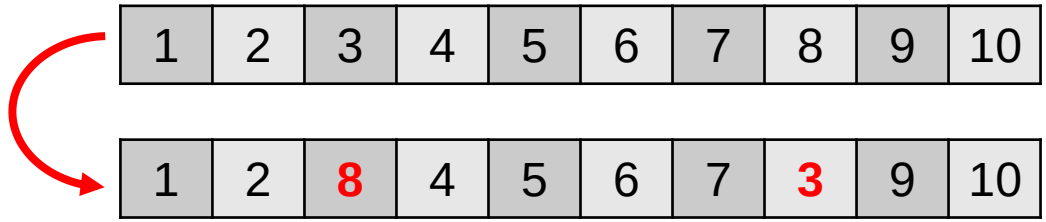


- Crossover?
  - ?

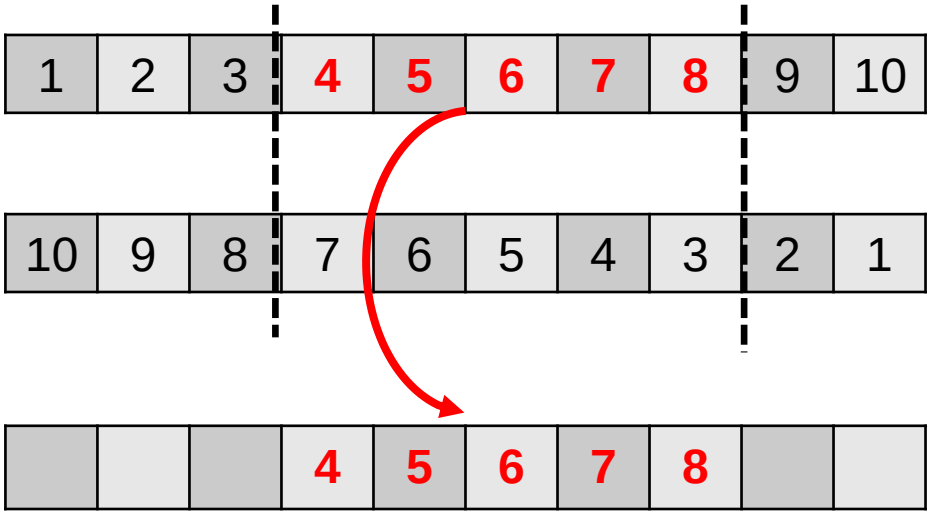


GA: Travelling salesman

- New solution still needs to contain all cities
- Mutation?
  - E.g. swapping positions

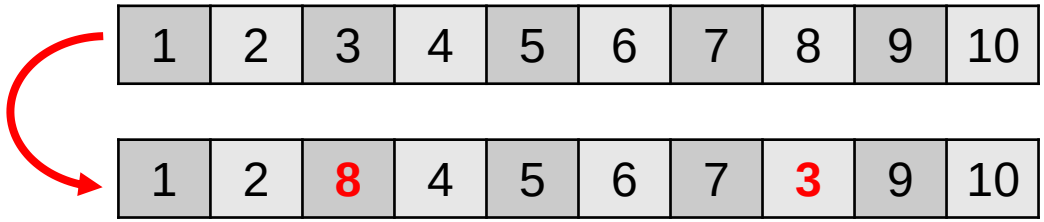


- Crossover?
  - Take part of first parent

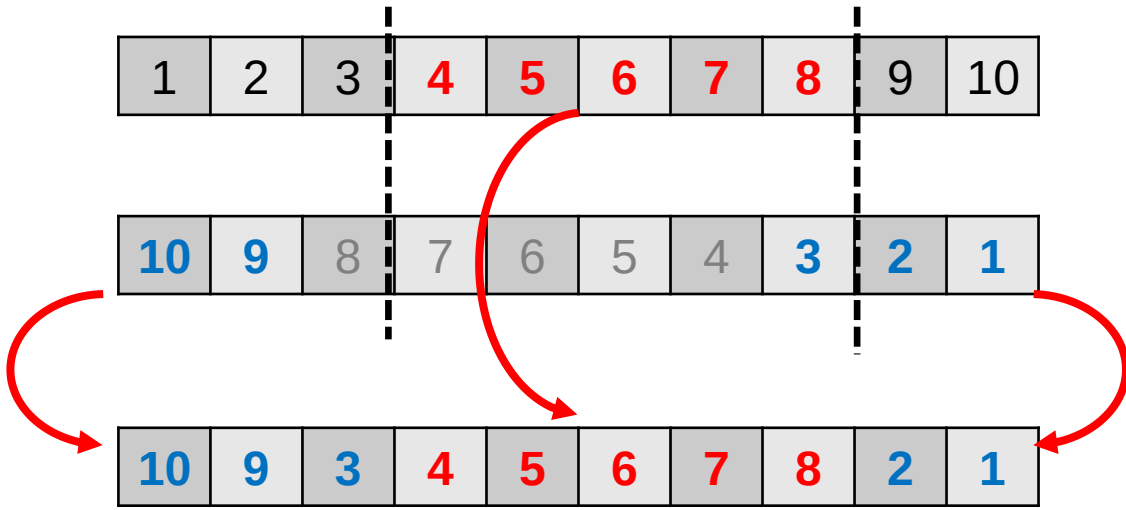


GA: Travelling salesman

- New solution still needs to contain all cities
- Mutation?
  - E.g. swapping positions

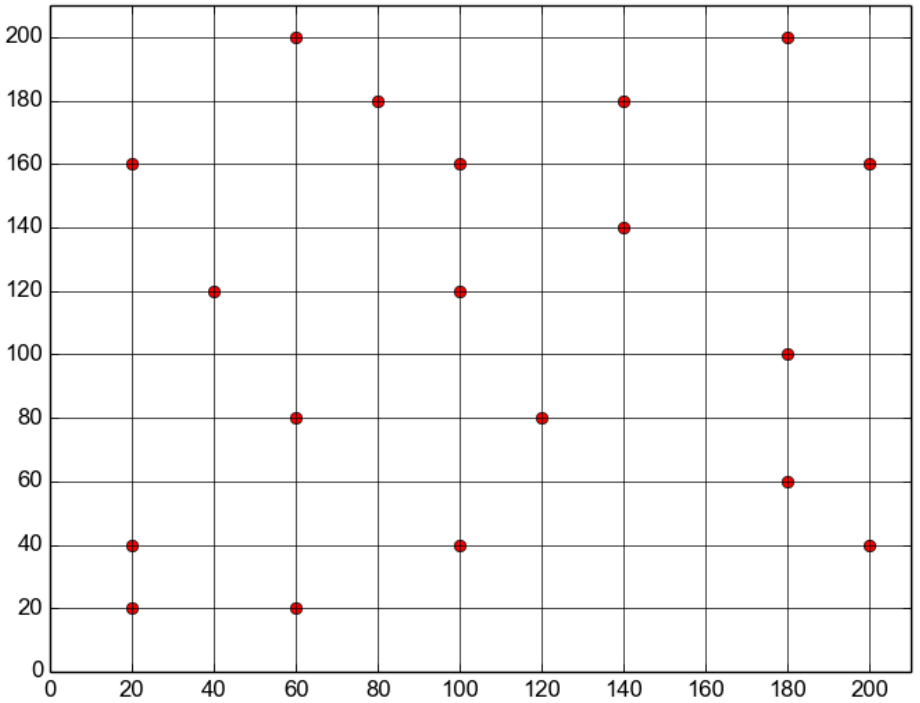


- Crossover?
  - Take part of first parent
  - Fill up unused values from second parent



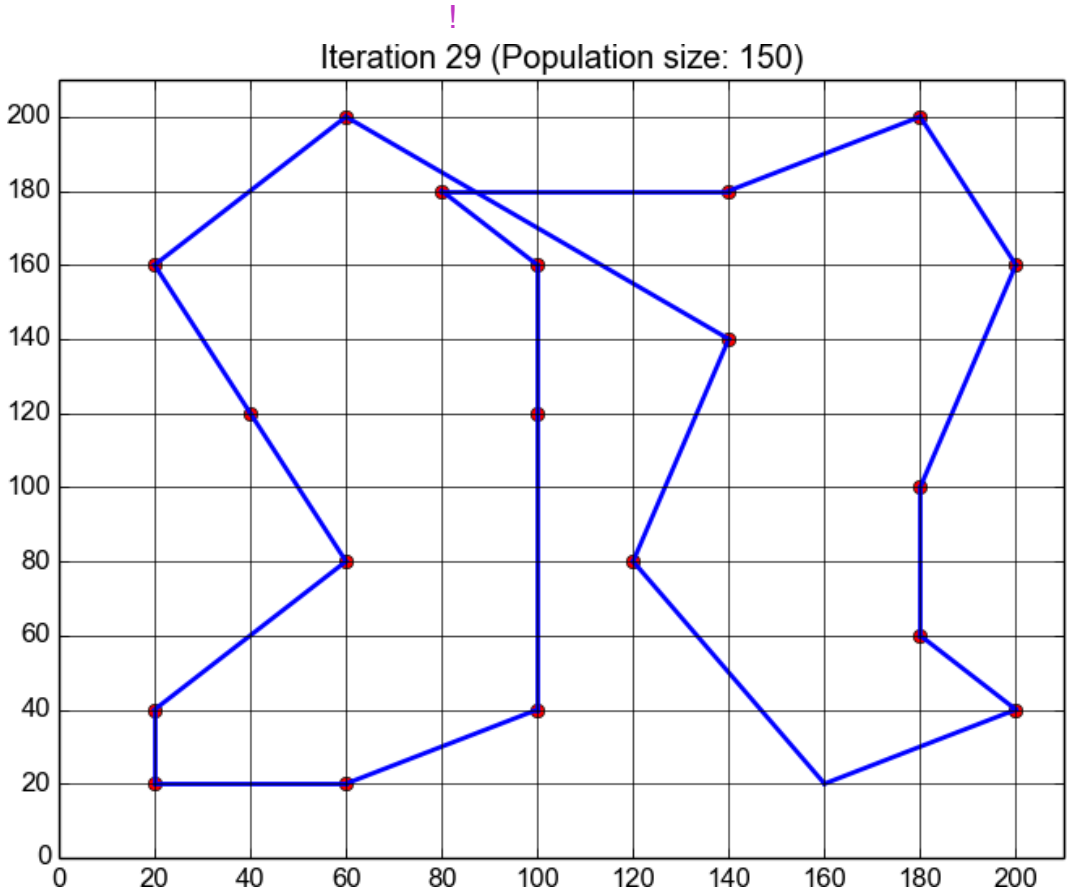
GA: Travelling salesman

- Example
  - 20 cities
  - Distance: Euclidean
  - Initialisation: random



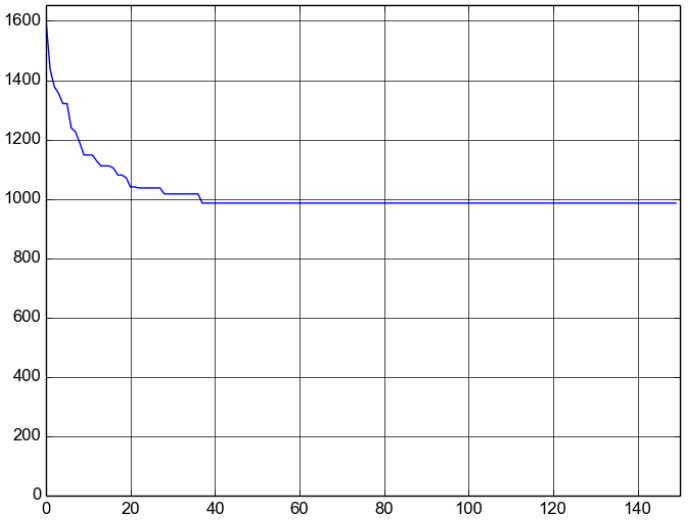
GA: Travelling salesman

- Run 1 29
  - Population size: 150, Iterations: 150



GA: Travelling salesman

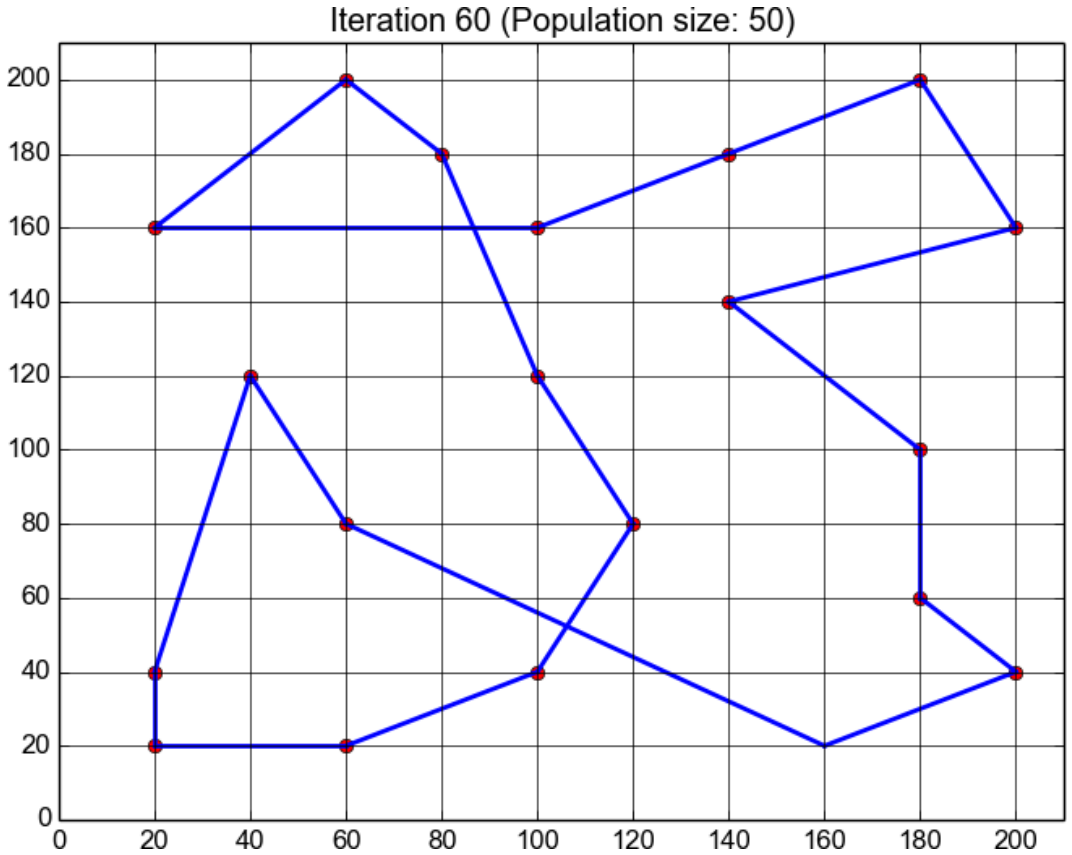
- Run 1
  - Population size: 150, Iterations: 150
  - Final solution in Iteration 38



Fitness (distance)

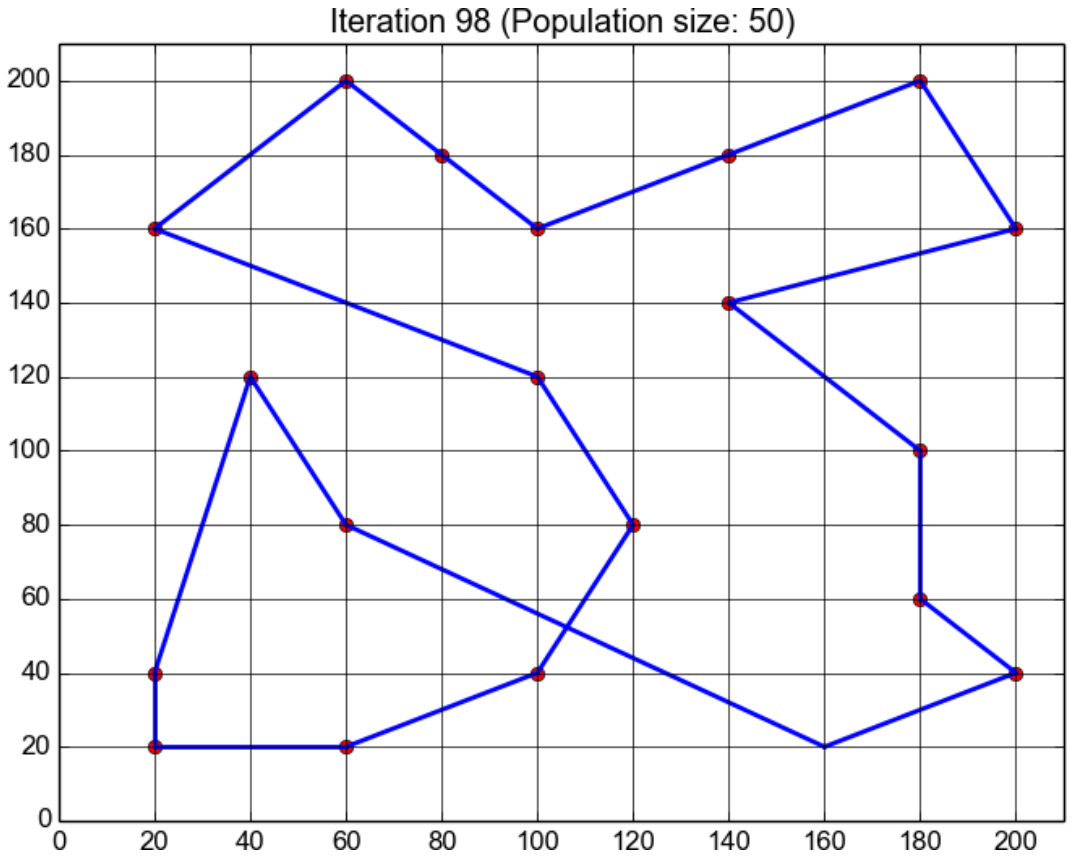
GA: Travelling salesman

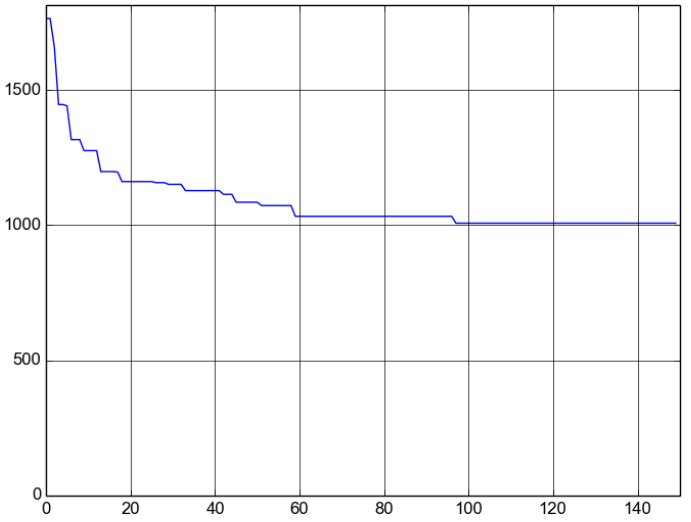
- Run 2
  - Population size: 50, Iterations: 150



GA: Travelling salesman

- Run 2
  - Population size: 50, Iterations: 150
  - Final solution in Iteration 98





Fitness (distance)

GA: Travelling salesman

- Encoding
- Adaptations of mutation & crossover operators
  - To ensure valid solutions
- Relatively quick convergence
- *Optimal solution found?*

Objective (fitness) function

- Objective function
  - The only information about where to look for good solutions (no derivatives/gradients/… available)
  - Should include as much as possible of the knowledge about the sought solution
  - Must be defined for every possible chromosomes that can be expressed by the chosen representation scheme

- Search space can be
  - Multidimensional
  - Nonlinear
  - Multimodal
  - Discrete

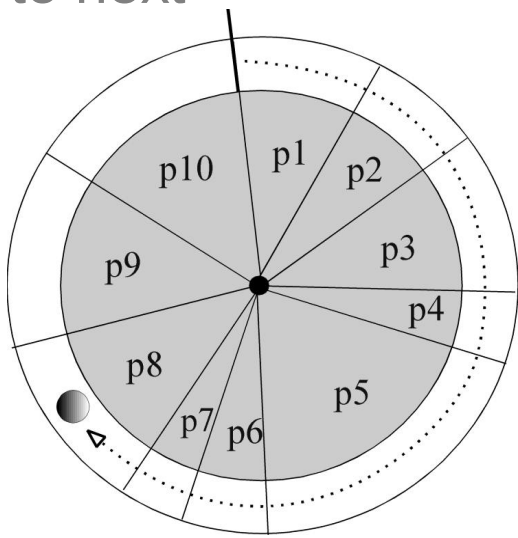
Initialization of the starting population

- Random
  - Pure random sampling (a random generator with P 0/1 = 0.5)
  - No prior knowledge about the sought solution
  - Relies just on lucky sampling of the whole search space by limited sample size
- Informed
  - Uses a prior knowledge about the sought optimal solution
  - **Can improve** the quality of the evolved best solution
  - **Can speed up** the evolutionary process
  - **Can bias** the evolutionary search towards regions with suboptimal solutions
- Preprocessing of the starting population
  - ex.: (multiple) short GA runs to identify promising regions of the search space

Selection

- Models the principle of „surviving the fittest“ observed in nature
- Simple approach: only select the ***n*** fittest
  - Favours the fitter individuals, suppresses the weak ones
  - Each individual should have some chance to pass to next generation
- Roulette wheel
  - Probability P_i of selecting *i*-th individual is proportional to its fitness f_i
    - *(Fitness proportionate selection)*

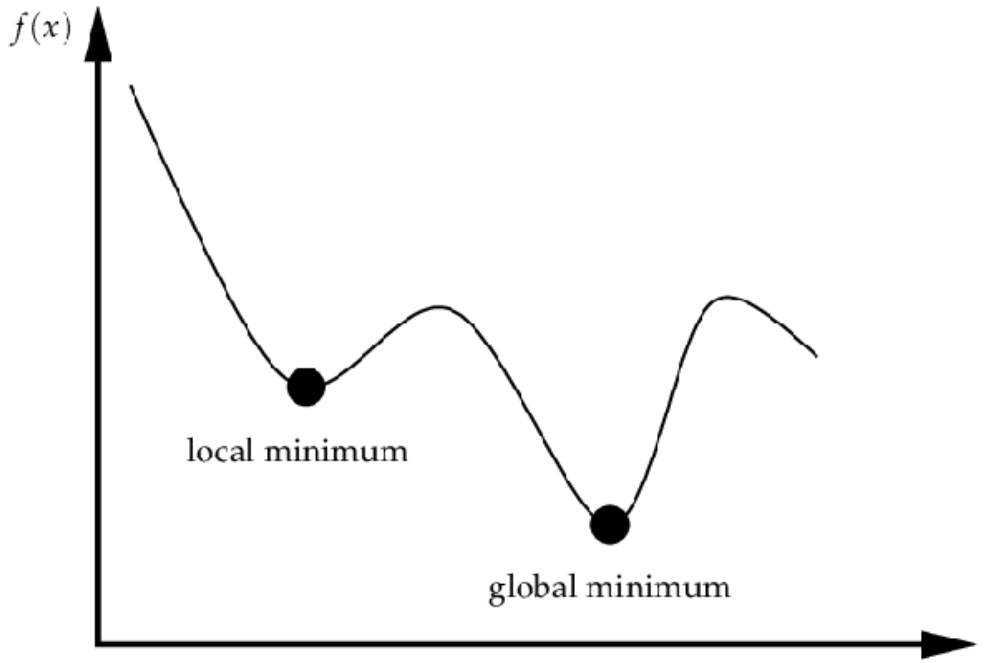
$$P_i = \frac{f_i}{\sum_{j=1}^{PopSize} f_j}$$



- Other selection schemes
  - Stochastic universal sampling, ***Tournament selection***, Reminder stoch. sampling, Rank-based sel.

Population diversity

- Diversity is crucial
  - Early homogenization of genetic material causes premature convergence
    - ➔ Disables valuable exploration of the search space
    - ➔ May lead to local optimum



Population diversity

- Crowding, Fitness sharing
  - The more dense a region of the search space (more individuals of the current population) ➔ the less fit they become
- Modified **selection**, e.g. restricted tournament selection
  - Select individual from various subsets
    - Subsets contain only similar individuals, achieved e.g. by clustering
    - More diversity, as also individuals from sub-optimal groups get chosen
- Adaptation of **mutation** & **crossover** genetic operators
  - When the population converges: change (increase) mutation and/or crossover parameters and/or recombination operators

  ➔ Exploration is increased

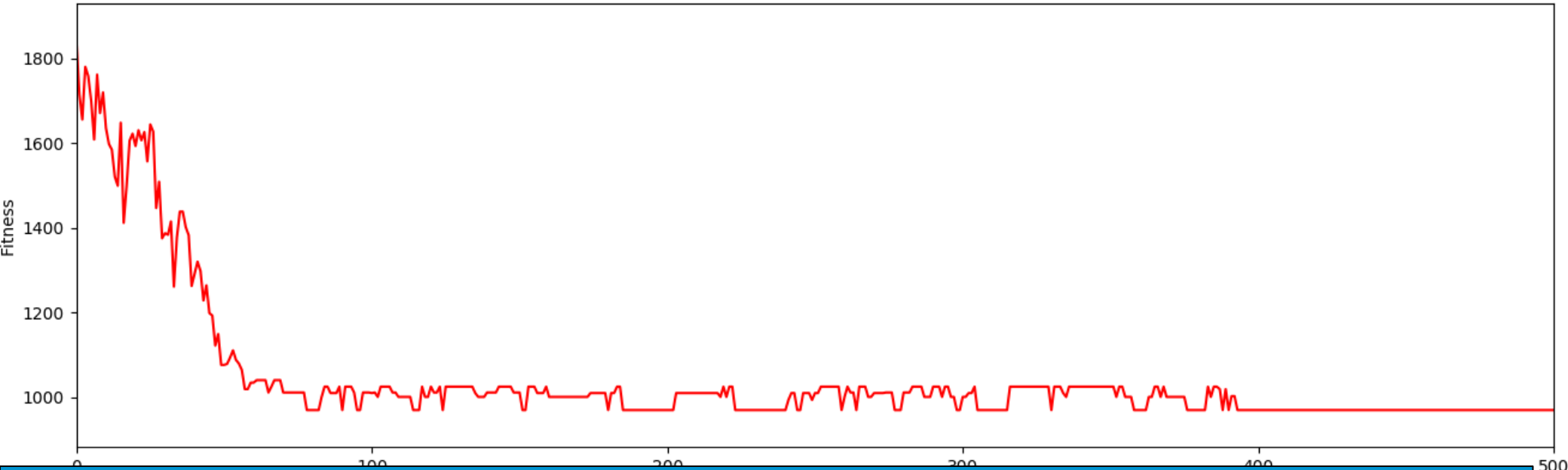
GAs: Recap general ideas

- Ideas are:
  - Adaption
  - Bring new „ideas“ into a system (Mutation)
  - Select among competitors (Selection)
  - Bring together successful individuals (Mating)

„It‘s Alive“: The Coming Convergence of Information, Biology, and Business“, Christoph Meyer, Stan Davis

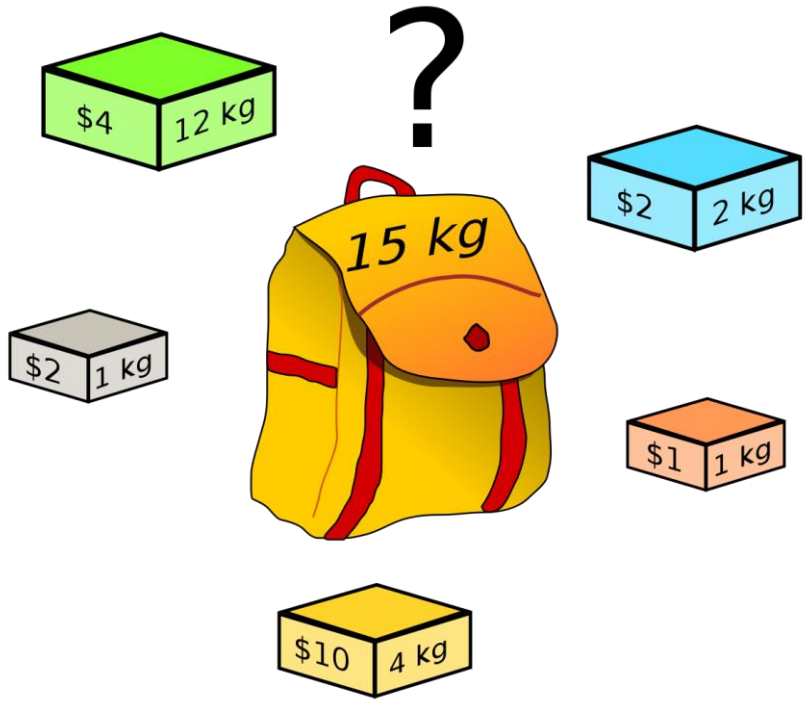
GA: elitist algorithm

- Elitist algorithm variation:
  - During selection: always keep best-known solution so far
  - *Why?*
    - Avoid regress in optimal solution found so far
    - If population big enough: not influencing population diversity too much
    - Alternative: Pocket algorithm – keep best known solution as ***result***



GA: Other example

- Given a set of items, each with a weight and a value
- Determine which items to include in a collection so
  - That the total weight is less than or equal to a given limit
  - And the total value is maximised



- *How to represent as GA?*

Questions ?

Outlook

- Evolutionary Algorithms
  - Genetic Programming
- Swarm intelligence
  - Ant colony optimisation
    - Swarm optimisation to be extended in subsequent lectures
- Multi Agent systems / Agent Based Modelling