Artificial intelligence 1

Anders Brodin

Evolutionary Ecology

Artificial life / Artificial intelligence / Computational intelligence:

- Cellular automata

- Evolutionary Programming

- Genetic Algorithms

- Genetic Programming

- Neural networks

$$x = x + 1$$

$x = x + 1$

$x{+}{+}$

Artificial life / Artificial intelligence / Computational intelligence:

- Cellular automata

- Evolutionary Programming

- Genetic Algorithms

- Genetic Programming

- -

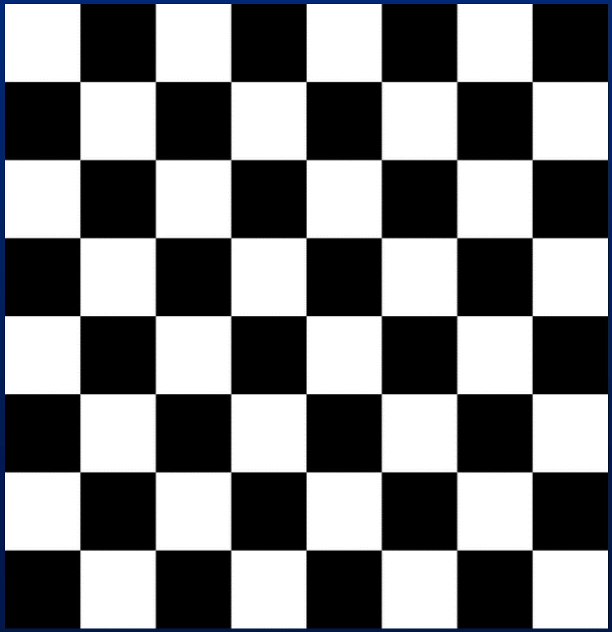
- Neural networks

Cellular automata - global effects from local rules:

- Visually represented as a grid of cells (1D, 2D, 3D)
- Each cell contains a variable e.g., 0 or 1 (white or black, alive or dead).
- At the same level all cells update synchronously.
- All cells subject to the same rule
- Time advances in discrete steps

A one dimensional example:

1
2
3
…

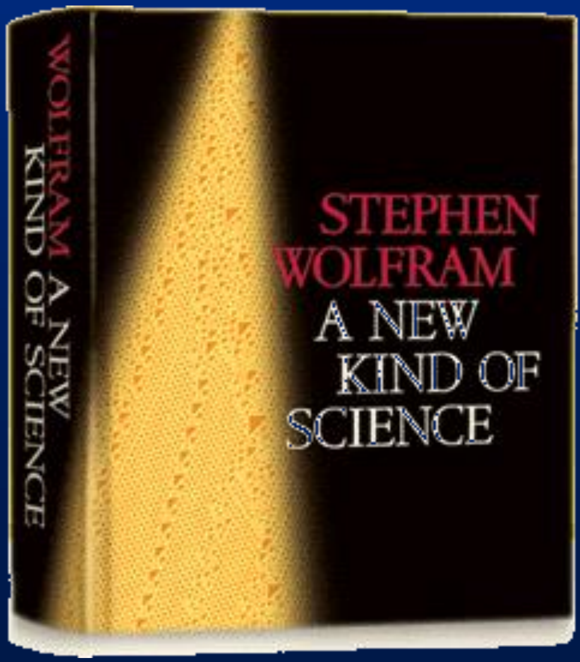


"A New Kind of Science" Wolfram's ideas:

– Complex behaviour is often the result of simple computational rules.

– The proof: simple cellular automata can produce any complex behaviour.

– Traditional mathematics is not enough.



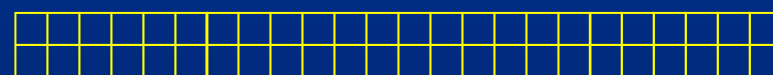
“When I made my first discoveries about cellular automata in the early 1980s I suspected that I had seen the beginning of something important. But I had no idea just how important it would all ultimately turn out to be. And indeed over the past twenty years I have made more discoveries than I ever thought possible. And a new kind of science that I have spent so much effort building has seemed an ever more central and critical direction for future intellectual development.”

Stephen Wolfram, A New Kind of Science 2002

Wolfram’s rule table:

- A systematic way of naming 1-d automata
- Not important to learn, just understand
- Describes what happens at second row
- One cell in focus

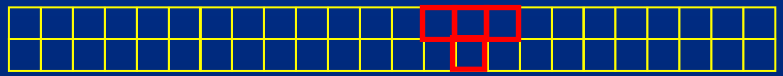
First row (generation 1)
Second row (generation 2)



Wolfram’s rule table:

- A systematic way of naming 1-d automata
- Not important to learn, just understand
- Describes what happens at second row
- One cell in focus

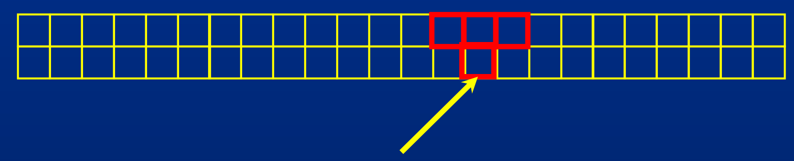
First row (generation 1)
Second row (generation 2)



Wolfram’s rule table:

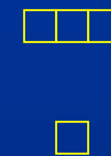
- A systematic way of naming 1-d automata
- Not important to learn, just understand
- Describes what happens at second row
- One cell in focus

First row (generation 1)
Second row (generation 2)



Cell in focus
but the same rule applies to all cells in row 2 simultaneously

Wolfram’s rule table:



Wolfram's rule table, binary numbers:

Base 2	Base 10
0	0
1	1
10	2
?	3

Fill in the following missing numbers:

Base 2	Base 10
0	0
1	1
10	2
11	3
?	4
?	5
?	6

Wolfram’s rule table, binary numbers:

Base 2		Base 10
0	0	0
1	2^0	1
10	2^1	2
11		3
100	2^2	4
101		5
110		6
111		7
1000	2^3	8
1001		9
1010		10

Wolfram's rule table, binary numbers:

Base 2	Base 10
0	0
1	1
10	2
11	3
100	4
101	5
110	6
111	7
1000	8
1001	9
1010	10

Maximum binary number eight digits:

_ _ _ _ _ _ _ _

Wolfram's rule table, binary numbers:

Base 2	Base 10
0	0
1	1
10	2
11	3
100	4
101	5
110	6
111	7
1000	8
1001	9
1010	10

Maximum binary number eight digits:

1 1 1 1 1 1 1 1

Wolfram's rule table, binary numbers:

Base 2	Base 10
0	0
1	1
10	2
11	3
100	4
101	5
110	6
111	7
1000	8
1001	9
1010	10

Maximum binary number eight digits:

1	1	1	1	1	1	1	1
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0

Wolfram's rule table, binary numbers:

Base 2	Base 10
0	0
1	1
10	2
11	3
100	4
101	5
110	6
111	7
1000	8
1001	9
1010	10

Maximum binary number eight digits:

1	1	1	1	1	1	1	1
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
128	64	32	16	8	4	2	1

Wolfram's rule table, binary numbers:

Base 2	Base 10
0	0
1	1
10	2
11	3
100	4
101	5
110	6
111	7
1000	8
1001	9
1010	10

Maximum binary number eight digits:

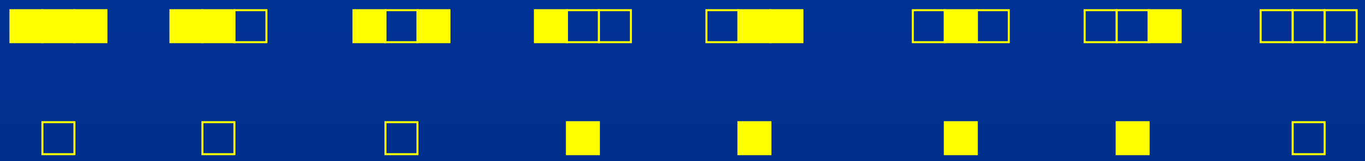
1	1	1	1	1	1	1	1		
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0		
128	64	32	16	8	4	2	1	=	255

Wolfram's rule table:

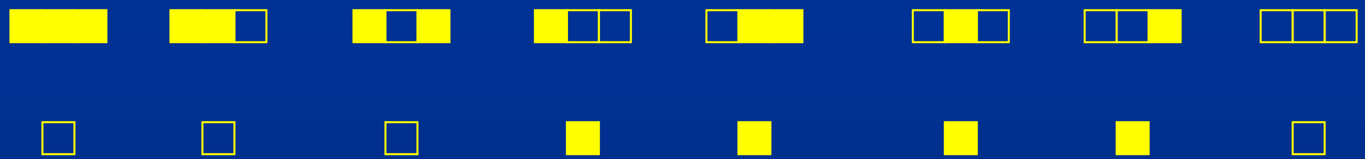
Filled square = 1 or ”yellow” or ”alive”

Open (blue) square = 0 or ”blue” or ”dead”

Wolfram’s rule table:



Wolfram's rule table:



Binary representation:

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

Wolfram's rule table:

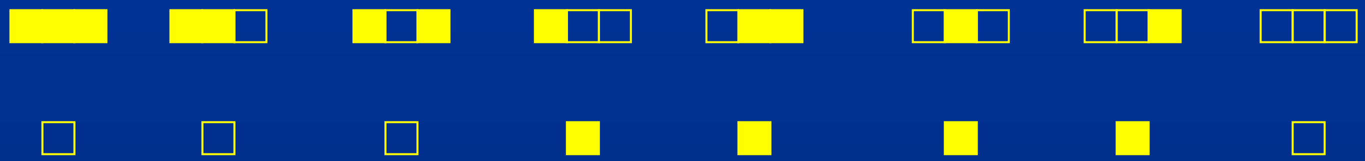
■■■	■■□	■□■	■□□	□■■	□■□	□□■	□□□
□	□	□	■	■	■	■	□

Binary representation:

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

$$2^4 + 2^3 + 2^2 + 2^1$$

Wolfram's rule table:



Binary representation:

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

$$2^4 + 2^3 + 2^2 + 2^1$$

Same thing, different representation

Wolfram's rule table:

■■■	■■□	■□■	■□□	□■■	□■□	□□■	□□□
□	□	□	■	■	■	■	□

Binary representation:

111	110	101	100	011	010	001	000	
0	0	0	1	1	1	1	0	
$0 +$	$0 +$	$0 +$	$2^4 +$	$2^3 +$	$2^2 +$	$2^1 +$	0	
			$16 +$	$8 +$	$4 +$	2		$= 30$

Rule 30

Wolfram’s classes:

Class 1: A fixed, homogeneous, state is eventually reached (e.g., rules 0, 8, 128, 136, 160, 168). 136: 10001000, seed with rows

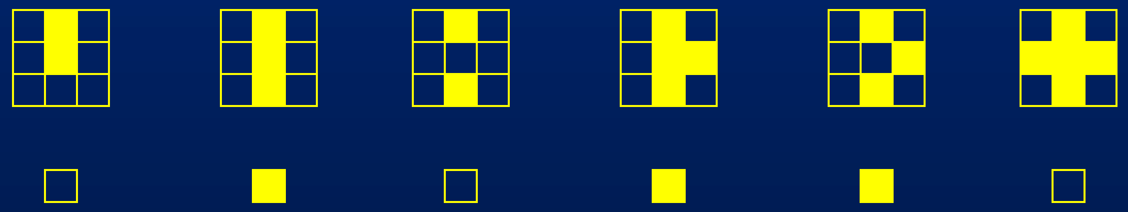
Class 2: A pattern consisting of separated periodic regions is produced (e.g., rules 4, 37, 56, 73). 37: 00100101, seed with 1

Class 3: A chaotic, aperiodic, pattern is produced (e.g., rules 18, 45, 105, 126).
18: 00010010, seed with 1

Class 4: Complex, localized structures are generated (e.g., rules 30, 110).
30: 00011110, seed with 1

Game of Life: each cell is “alive” or “dead”

- Two dimensional CA with rules based on number of live neighbours among 8
  - N = 1 death (loneliness)
  - N = 2 no change
  - N = 3 birth
  - N = 4 death (overcrowding)



Applications:

- Biological systems
- Art and design
- Computer graphics
- Image processing
- Games

Evolutionary Computation:

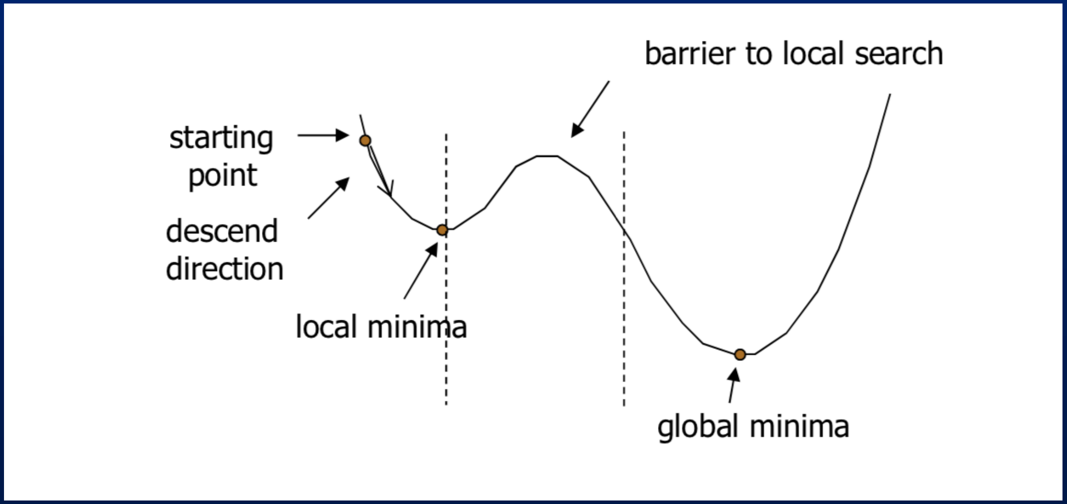
- algorithmic models that use Darwinian-like evolution
- iterative optimization

Different types:

- Evolutionary strategy models
- Genetic algorithms
- Genetic programming

Genetic algorithms, GA:

- robust and global compared to calculus and enumerative solutions
- easier to use when little is known about the problem
- GA's can search a large solution space and find a global optimum
- use only for complex problems
- alternative: simulated annealing



A genetic algorithms step by step

1. Represent your problem as genes in a binary value chromosome (= solution)

Example:

Find the maximum of a 10 digits base 2 number:

```
MyChromosome <- c(1, 0, 0, 0, 1, 0, 1, 1, 0, 0 )
```

2. Random generation of a ”population” of chromosomes

individual	representation
1	0 0 1 0 1 1 1 0 1 0
2	1 0 1 1 1 0 1 0 1 0
3	0 0 0 1 1 1 1 0 0 1
4	1 0 0 0 0 1 1 1 1 0

In reality start with 50 chromosomes (=solutions)

3. Evaluate ”fitness” of individual chromosomes

individual	representation	fitness
1	0 0 1 0 1 1 1 0 1 0	186
2	1 0 1 1 1 0 1 0 1 0	746
3	0 0 0 1 1 1 1 0 0 1	121
4	1 0 0 0 0 1 1 1 1 0	532

Fitness = how good is a chromosome as solution

4. Sort solutions after rank

individual	representation	fitness
1	1 0 1 1 1 0 1 0 1 0	746
2	1 0 0 0 0 1 1 1 1 0	532
3	0 0 1 0 1 1 1 0 1 0	186
4	0 0 0 1 1 1 1 0 0 1	121

5. Pairing, recombination

individual	representation										fitness
1	1	0	1	1	1	0	1	0	1	0	746
2	1	0	0	0	0	1	1	1	1	0	532
3	0	0	1	0	1	1	1	0	1	0	186
4	0	0	0	1	1	1	1	0	0	1	121

Elitistic pairing, random cut

5. Pairing, recombination → offspring

individual	representation	fitness
1	1 0 1 1 1 0 1 0 1 0	746
2	1 0 0 0 0 1 1 1 1 0	532
3	0 0 1 0 1 1 1 0 1 0	186
4	0 0 0 1 1 1 1 0 0 1	121
5	1 0 1 1 1 1 1 1 1 0	767

High quality offspring = good solution

6. Insert offspring in population, discard worst solutions

individual	representation	fitness
1	1 0 1 1 1 1 1 1 1 0	767
2	1 0 1 1 1 0 1 0 1 0	746
3	1 0 0 0 0 1 1 1 1 0	532
4	0 0 1 0 1 1 1 0 1 0	186
5	~~0 0 0 1 1 1 1 0 0 1~~	121

”killed”! (keep the number of chromosmes at 50)

Why cannot the GA solve this problem?

individual	representation	fitness
1	1 0 1 1 1 1 1 1 1 0	767
2	1 0 1 1 1 0 1 0 1 0	746
3	1 0 0 0 0 1 1 1 1 0	532
4	0 0 1 0 1 1 1 0 1 0	186

7. Mutation

individual	representation	fitness
1	1 0 1 1 1 1 1 1 1 0	767
2	1 1 1 1 1 0 1 0 1 0	874
3	1 0 0 0 0 1 1 1 1 0	532
4	0 0 1 0 1 1 1 0 1 0	186

7. Mutation

individual	representation	fitness
1	1 0 1 1 1 1 1 1 1 0	767
2	1 1 1 1 1 0 1 0 1 0	874
3	1 0 0 0 0 1 1 1 1 0	532
4	0 0 1 0 1 1 1 0 1 0	186

Now you can make a genetic algorithm!

8. Cycle until terminantion

1. Represent your problem as genes in a chromosome
2. Random generation of a population of chromosomes
3. Evaluate fitness of individual chromosomes
4. Sort after rank
5. Pairing, recombination
6. Insert offspring
7. Mutation

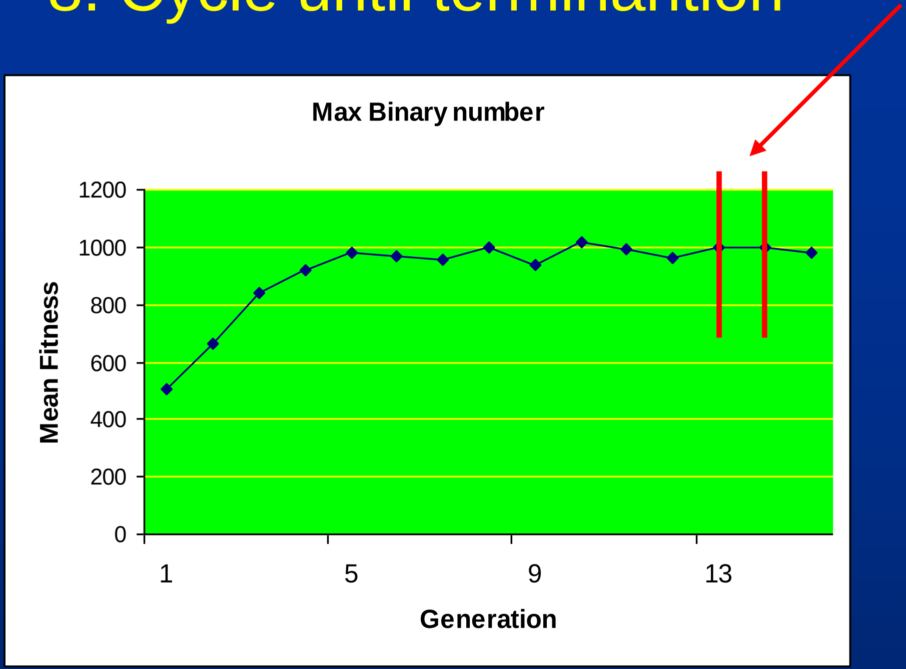
8. Cycle until terminantion

1. Represent your problem as genes in a chromosome
2. Random generation of a population of chromosomes
3. Evaluate fitness of individual chromosomes
4. Sort after rank
5. Pairing, recombination
6. Insert offspring
7. Mutation

Termination?

8. Cycle until terminantion

diff = $y_1 - y_2$

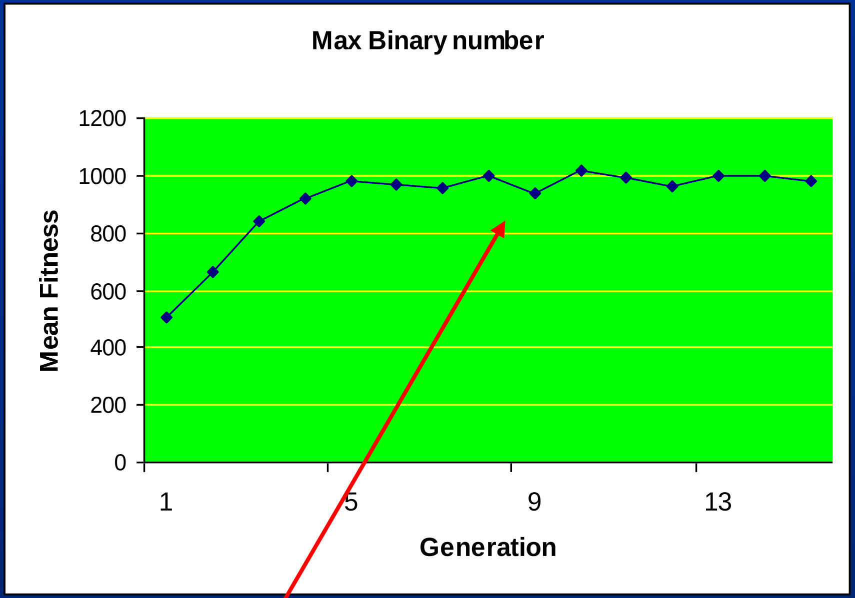


for (g in 1:15)
{cycle algorithm}

g = generation

while (diff > 10)
{diff = $y_1 - y_2$;
cycle algorithm
}

8. Cycle until terminantion



Why not a straight line? It is a very simple problem.

Genetic algorithm

Example 1, binary (= discreet) GA:

Find maximum of a 10 digits base 2 number:

```
MyChromosome <- c(1, 0, 0, 0, 1, 0, 1, 1, 0, 1)
```

2. find best solution of an equation with two unknowns

$f(x, y) = x \sin(4x) + y \sin(2y)$

```
MyChromosome <- c( 3.43537, -1.0002345)
```

Continuous genetic algorithms:

- real, direct values can be used (no goal function)

- decide range for gene values (e.g. $-5 \leq x \leq 5$)

Best solution of equation with two unknowns:

$f(x, y) = x \sin(4x) + y \sin(2y)$

individual	values (start with randomly created numbers)
1	[3.435, -1.003]
2	[-2.331, - 4.234]
3	[-0.877, - 4.954]
4	[3.537, 1.645]
5	[2.479, - 1.541]

Continuous genetic algorithms:

- problems with recombination:

ind	values
1	[3.435, -1.003]
2	[-2.331, - 4.234]
3	[-0.877, - 4.954]
4	[3.537, 1.645]
5	[2.479, - 1.541]

Question1: Why is direct crossover not good?

Continuous genetic algorithms:

- problems with recombination:

ind	values
1	[3.435, -1.003]
2	[-2.331, - 4.234]
3	[-0.877, - 4.954]
4	[3.537, 1.645]
5	[2.479, - 1.541]

Question 1: Why is direct crossover not good?
Question 2: Why is averaging better but not good?

Continuous genetic algorithms:

- blending

ind	values
p_1	[3.435, -1.003]
p_2	[-2.331, - 4.234]

Three solutions:

i) proportional blending: $b * p_1 + (1 - b) * p_2$ where $0 \geq b \geq 1$

ii) linear blending gives three offspring:

offsp1 = $0.5p_1 + 0.5p_2$

offsp2 = $1.5p_1 - 0.5p_2$

offsp3 = $1.5p_2 - 0.5p_1$

iii) combination between crossover and blending

Continuous genetic algorithms:

Question: What problem may this blending process create?
(Think about the min and max values -5, 5)

Suggest a solution to this problem

Continuous genetic algorithms:

- individuals can be ”out of range”

offsp2 = $1.5p_1 - 0.5p_2$

p1 = 4.952, p2 = -4.213 gives offspring 9.535

Three solutions:

i) chop at limit

ii) ”gene repair”

iii) give low fitness

Continuous vs discrete genetic algorithms:

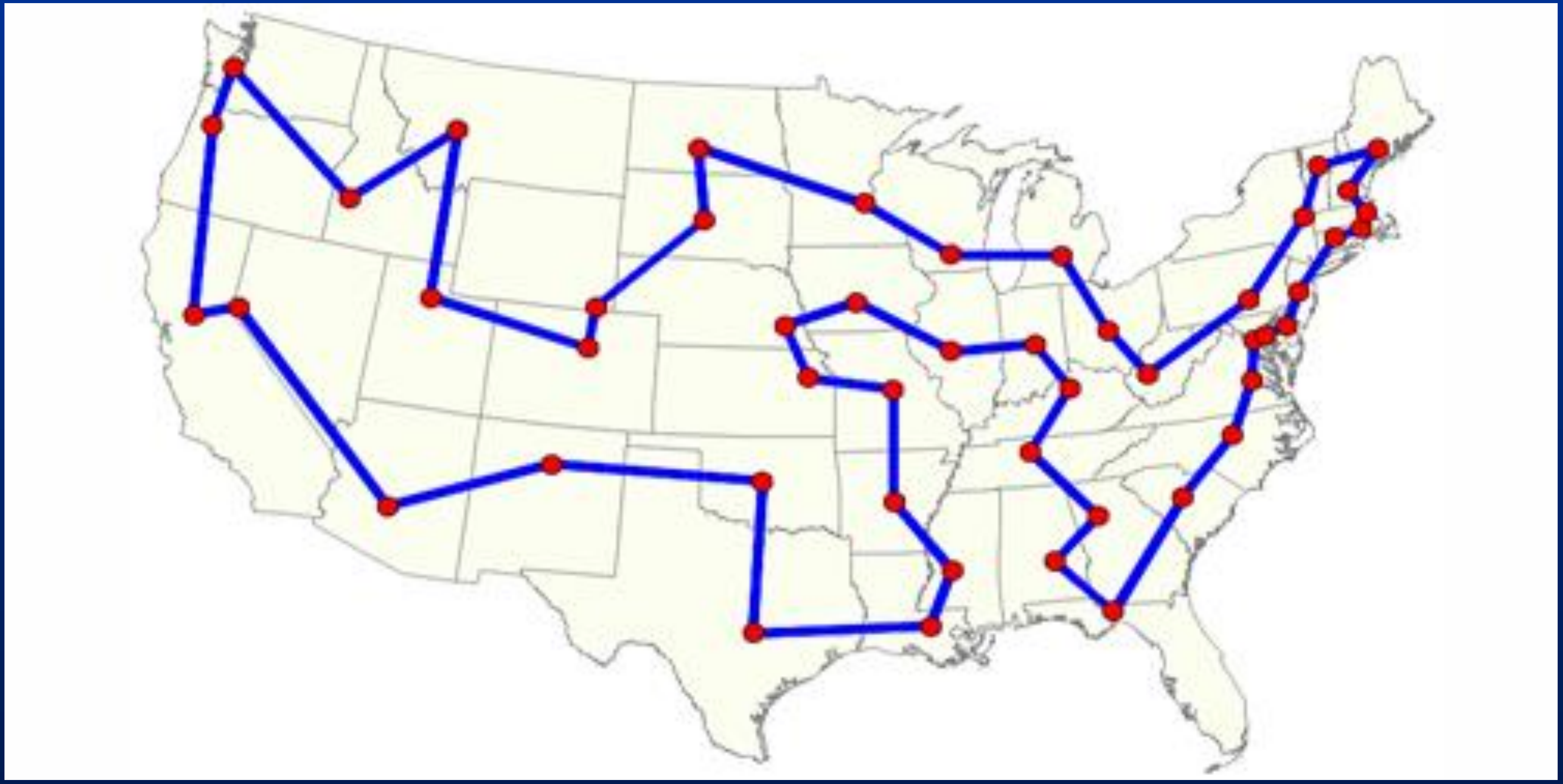
Discrete (0 1 1 0 1)	Continuous (0.654, 3.564…)
binary	real numbers
goal function	direct use of numbers
simple crossing over	complex mating
traditional, original	newer

Traveling salesman problem:

You have to visit: Göteborg, Borås, Norrköping, Kristianstad, start in Lund

Traveling salesman problem:

You have to visit a number of geographic positions, calculate the shortest route



Traveling salesman problem:

You have to visit: Göteborg, Borås, Norrköping, Kristianstad, start in Lund

Chromosome 1	Lund	Göteborg	Borås	N-köping	K-stad
Chromosome 2	Lund	Göteborg	N-köping	K-stad	Borås
Chromosome 3	Lund	K-stad	Göteborg	Borås	N-köping

Etc.

Traveling salesman problem:

You have to visit: Göteborg, Borås, Norrköping, Kristianstad, start in Lund

Chromosome 1	Lund	Göteborg	Borås	N-köping	K-stad
Chromosome 2	Lund	Göteborg	N-köping	K-stad	Borås
Chromosome 3	Lund	K-stad	Göteborg	Borås	N-köping

Etc.

What is fitness?

Traveling salesman problem:

You have to visit: Göteborg, Borås, Norrköping, Kristianstad, start in Lund

	Lund	Göteborg	Borås	N-köping	K-stad
Lund	0	262	275	431	77
Göteborg	262	0	63	311	264
Borås	275	63	0	250	264
N-köping	431	311	250	0	392
K-stad	77	264	264	392	0