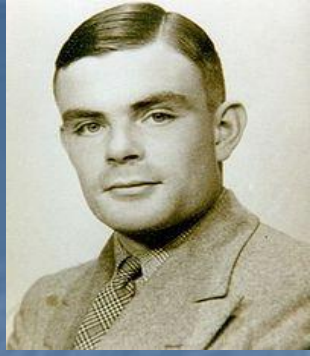
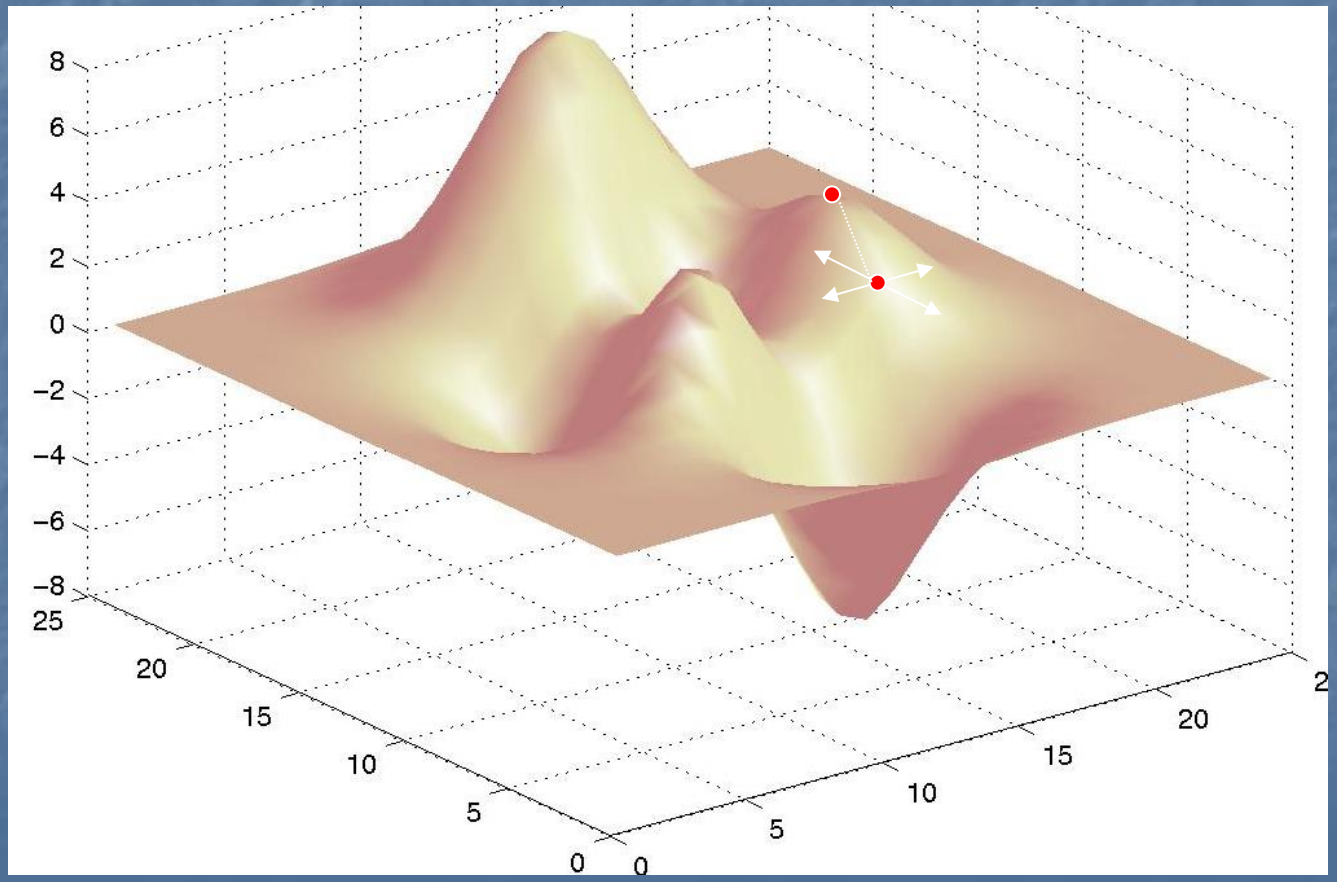
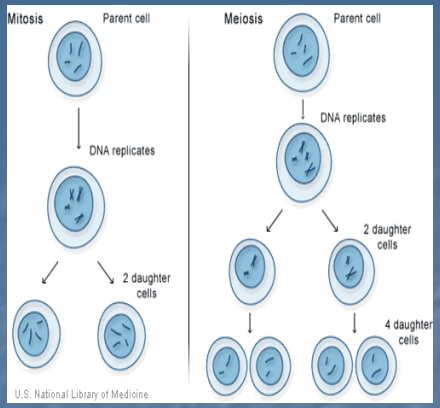
BioComputation

Evolutionary Algorithms 2

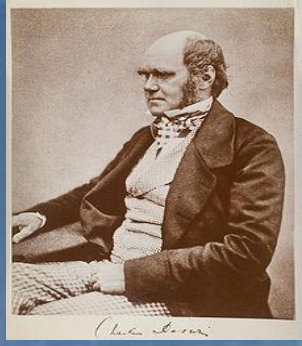
Learning and Search Spaces

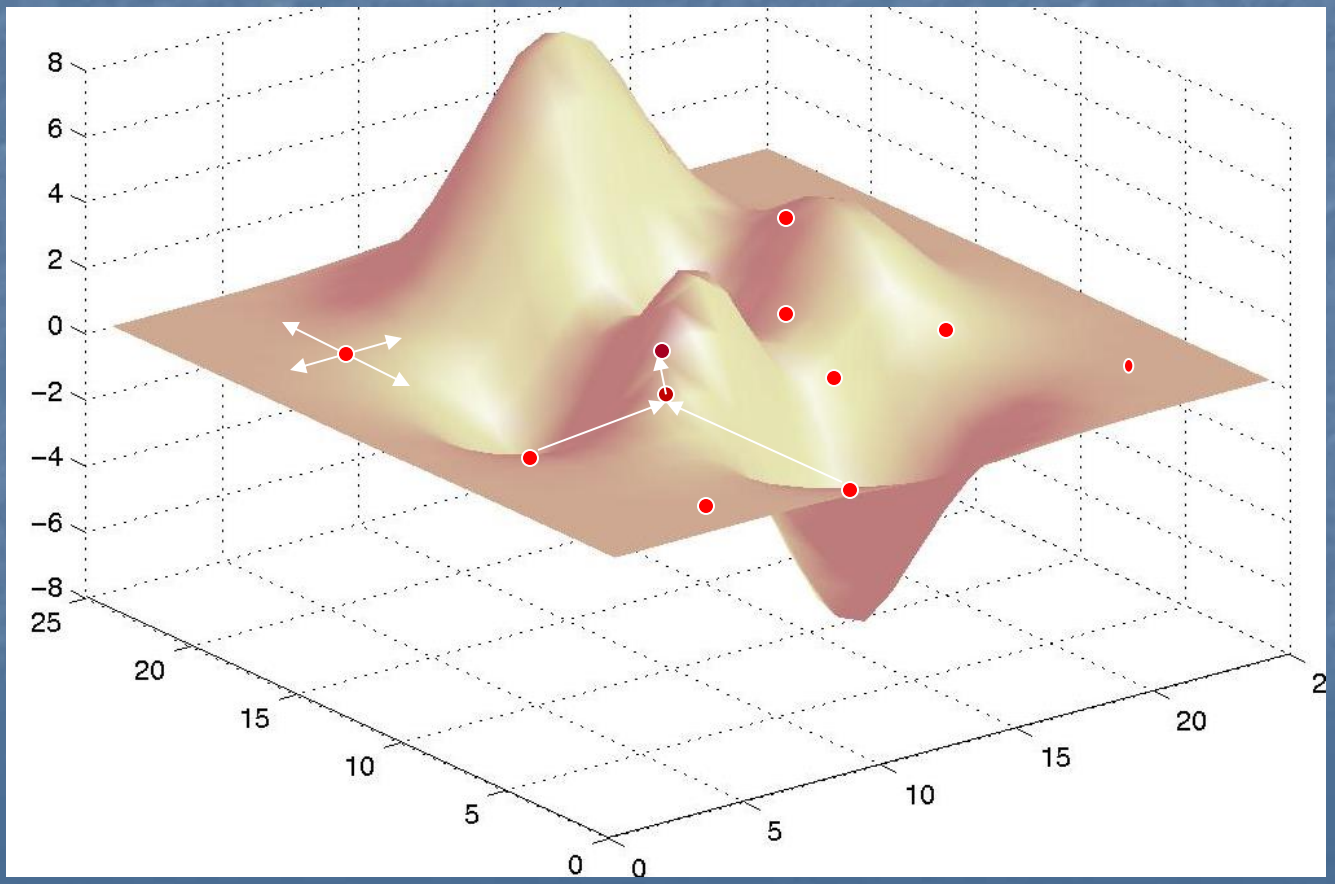




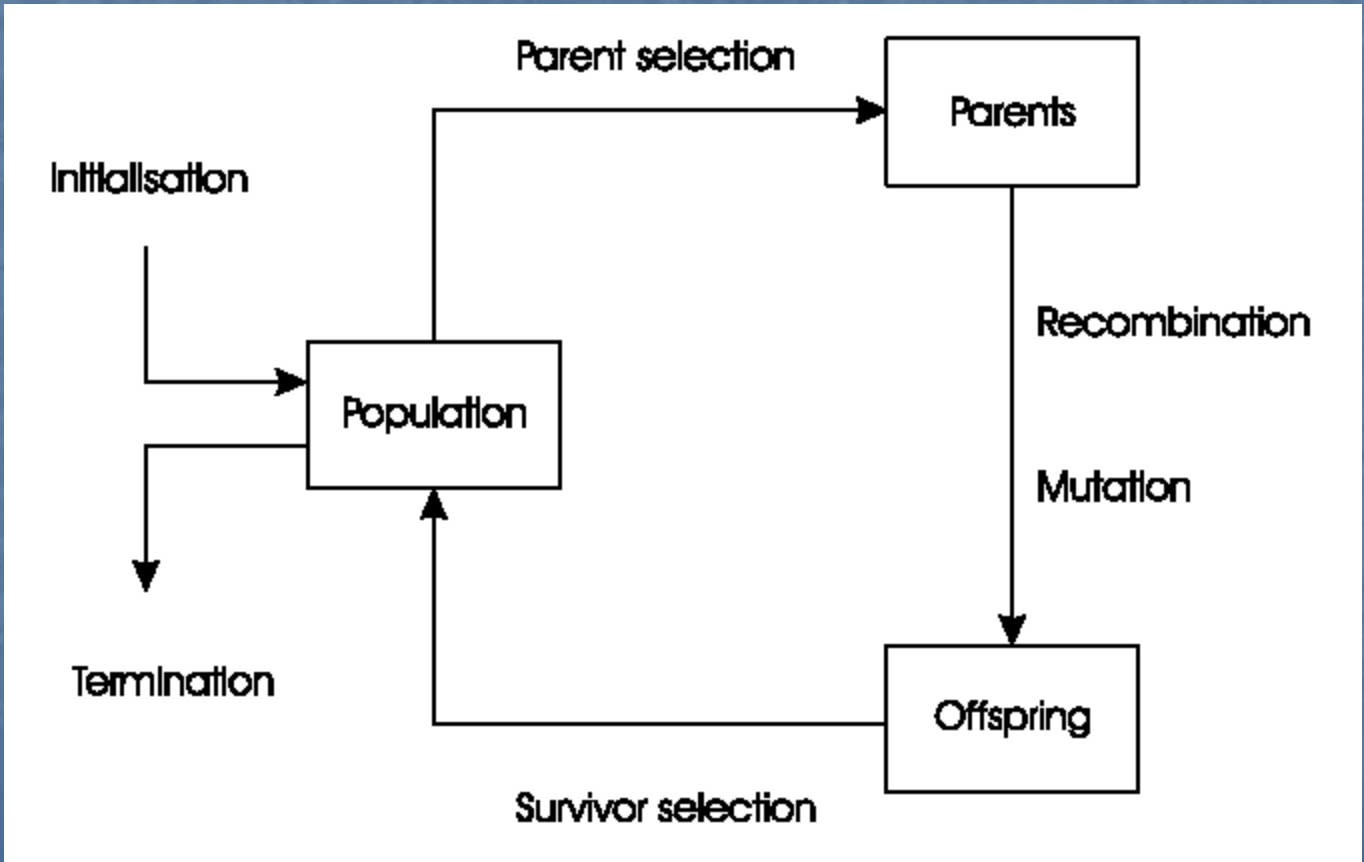


Evolution as Search





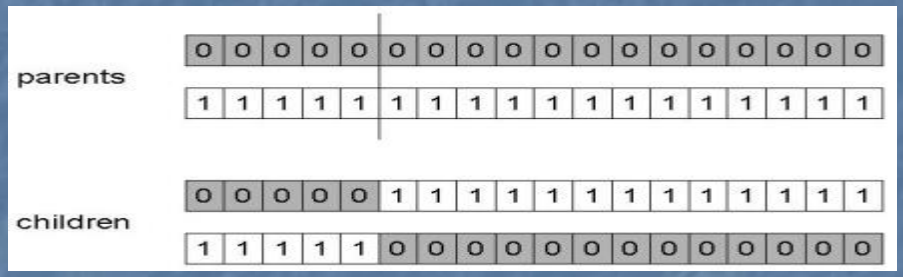
Evolutionary Search Process

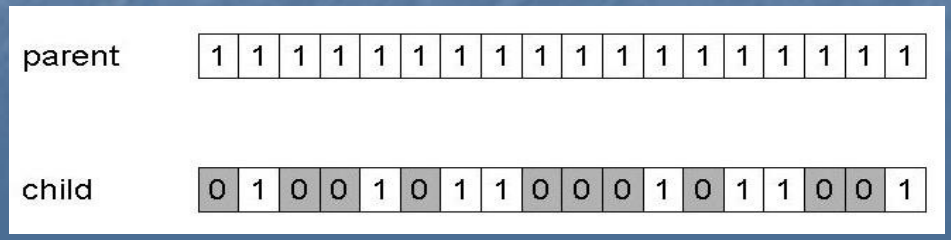


Genetic Algorithm Operations

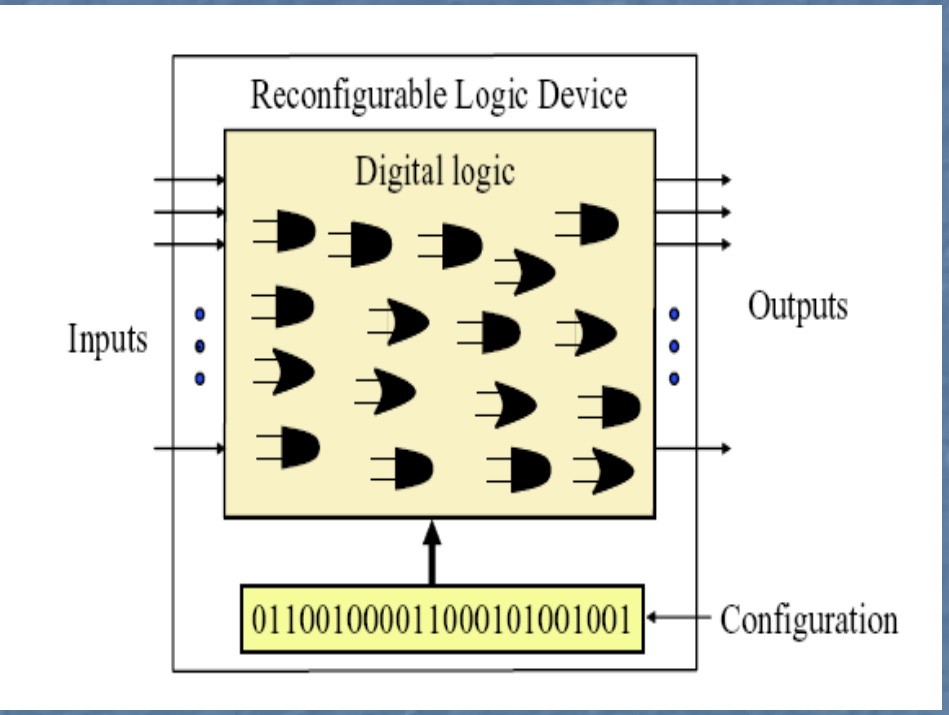
Population	Fitness
1	25.0
2	5.0
3	40.0
4	10.0
5	20.0

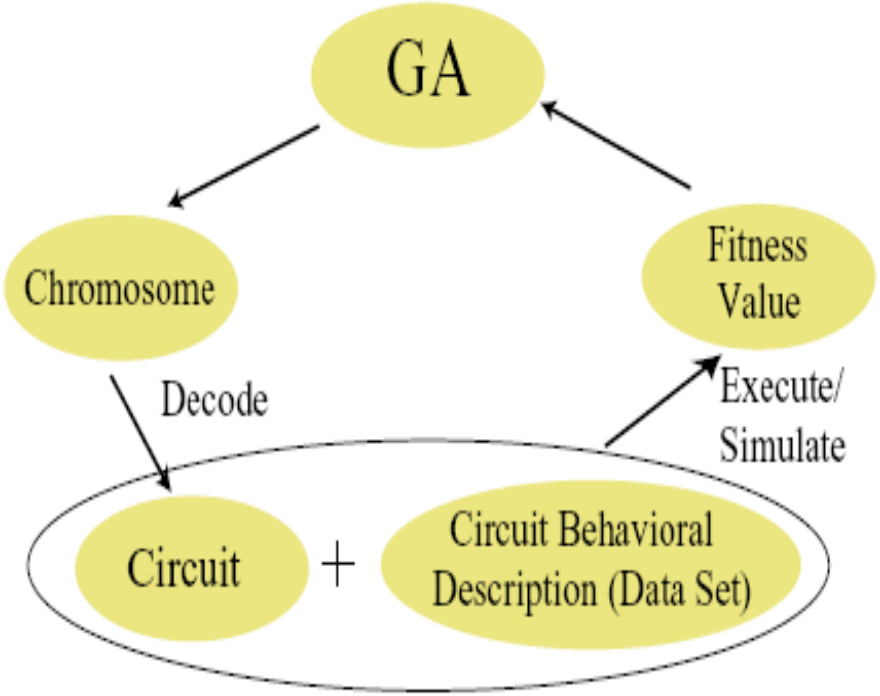
- Selection
  - Fitness Proportionate (Roulette Wheel)
- Recombination
  - One-point
- Mutation
  - Bit Flipping





Example: Hardware Design





Rules

- General knowledge can be represented by condition-action rules: IF <state> THEN <action>
- Standard production system form is *computationally complete*.
- Such rules can represent future predictions, categorize inputs, etc.
- High-level knowledge can be inferred from clusters of rules with similar conditions.
- It is usually the case that such rules are amenable to human understanding and programming.

Evolving Rules

- A Genetic Algorithm can be used to find the set of rules which solves a given task.
- Traditionally, the possible values of the inputs/concepts are binary encoded and concatenated together to form a rule in conjunctive form:
- IF *A* AND *B* AND *C* AND … THEN *X*

Simple Example

- Temp = {frozen, cold, warm, hot} = {00, 01, 10, 11}
- Weather = {snow, rain, wind, sun }
- Clothing = {jumper, coat }

- IF cold AND rain THEN coat
- IF 0101 THEN 1 => 01011

Generalization

- For large, complex problem spaces it is not feasible to learn rules for each possible combination of variables/states/concepts.
- Rule conditions must be able to identify the descriptive features of the problem and ignore the others.
- In the previous example, a coat would be required if it was either cold, frozen, snowing or raining.

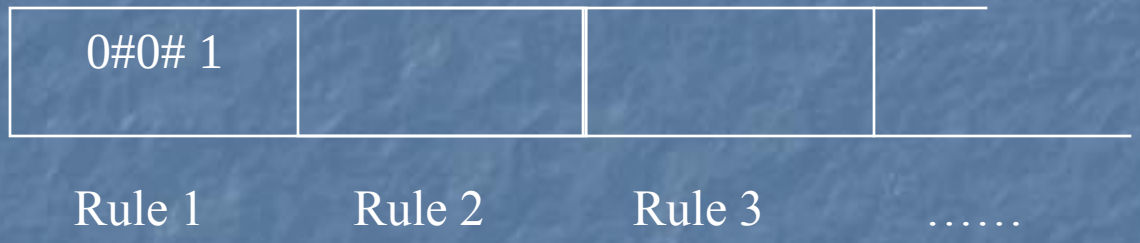
Generalization

- By adding a wildcard symbol ‘#’ to the encoding which means “don’t care” a single rule can be found:
- IF 00 AND 00 THEN 1 (IF frozen AND snow THEN coat)
- IF 01 AND 00 THEN 1
- IF 00 AND 01 THEN 1
- IF 01 AND 01 THEN 1 (IF cold AND rain THEN coat)

- IF 0# AND 0# THEN 1

Representation

- Rules concatenated to form genome.
- Variable length (fitness cost on size).

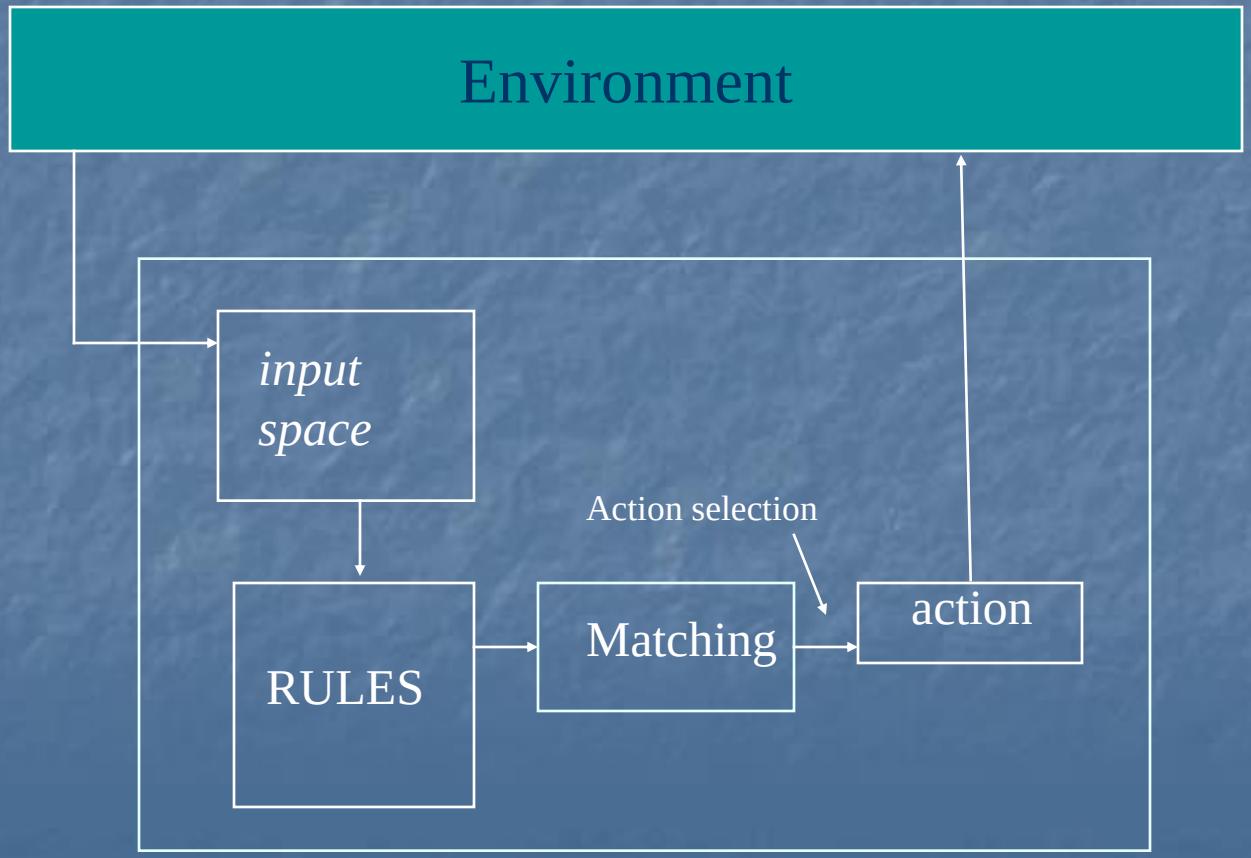


Reminder: Evolution Pseudo Code

Build rule-base from genome and try on the problem to calculate its fitness.

```
BEGIN
  INITIALISE population with random candidate solutions;
  EVALUATE each candidate;
  REPEAT UNTIL ( TERMINATION CONDITION is satisfied ) DO
    1 SELECT parents;
    2 RECOMBINE pairs of parents;
    3 MUTATE the resulting offspring;
    4 EVALUATE new candidates;
    5 SELECT individuals for the next generation;
  OD
END
```

Production System Schematic



Cycle

- Rulebase of just three rules:
  - 1. IF 001#001 THEN 001
  - 2. IF #####11 THEN 111
  - 3. IF 11#111# THEN 101
- Receives binary encoded input from environment: 1111011
- All rules compare their conditions with the current input.
- Must have same value in each position of input or a #.
- Rule 2 matches in this case.
- The output/decision of the system is the rule's corresponding action, e.g., whatever 111 encodes for in this case.

Contd.

- If more than one rule matches, and they have different actions, must have a conflict resolution strategy.
- Typical scheme uses a roulette-wheel using the number of rules which propose each action.
- Output given.
- Then get next input, etc. until end of task.
- Fitness of system assigned to the set of rules.

Rule Discovery

- Fitness typically averaged over a number of trials.
- Alignment used before crossover.
- Crossover within and/or between rules.
- Mutation altered to include #
- Inversion also used (esp. for crossover):

R1:R2:R3:R4:R5 ⟶ R1:R4:R3:R2:R5

Road Traffic Junction Control

- These evolutionary learners have been applied with some success in a number of real-world domains.
- Responsive, distributed control of road traffic is sought for many reasons.
- Allowing machine learning techniques to control each individual junction or sets of locally coupled junctions represents one way forward.

Inputs/Outputs

- Sensor technology at most inner-city junctions is typically still very crude: at best, approximations of approach lane occupancy some distance from the junction is available.
- Per cycle, the evo learner receives this % occupancy encoded as a binary string for each of the four approach lanes.
- It then returns the split time of red and green between the two directions of the junction.

Schematic

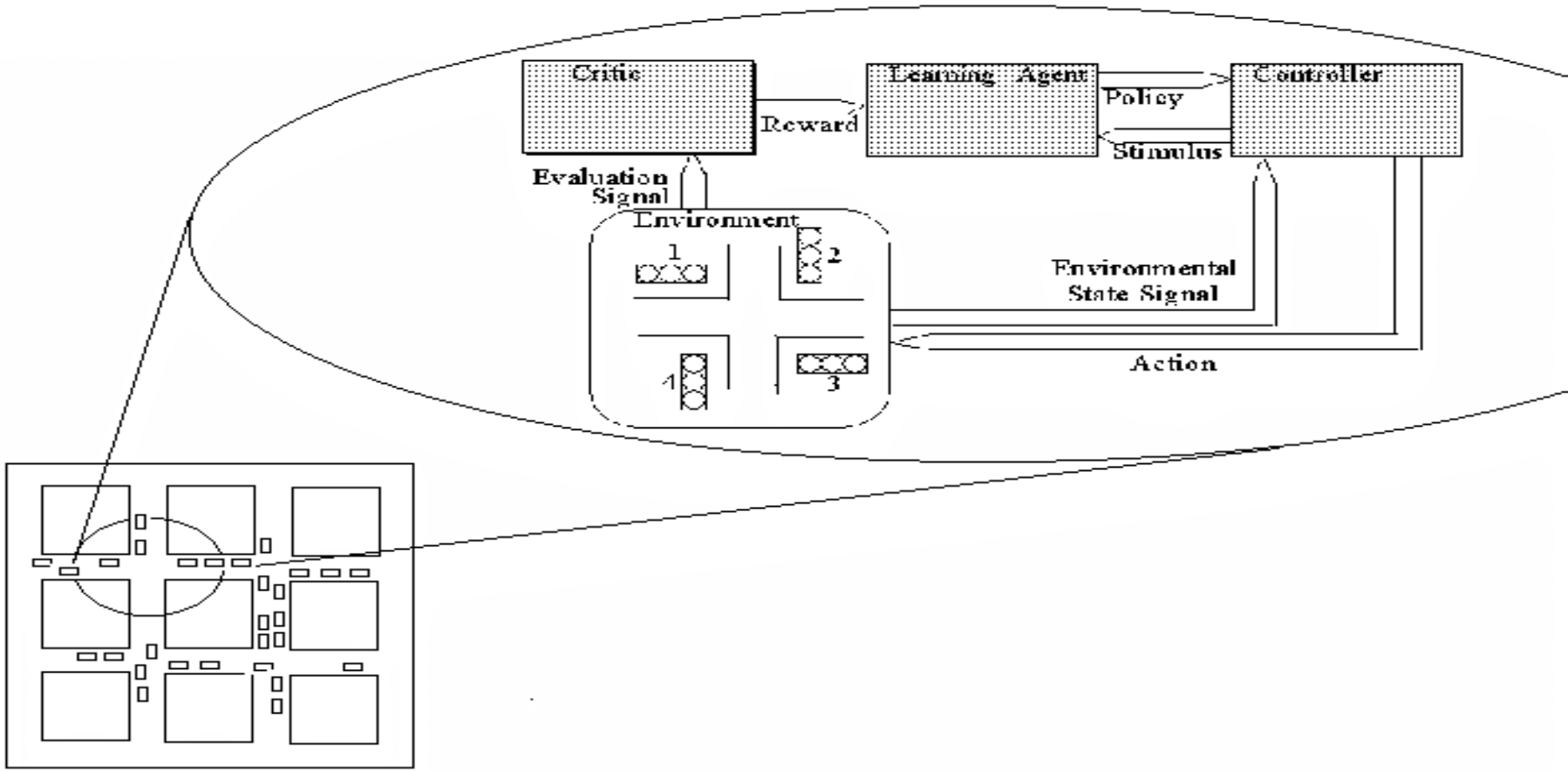
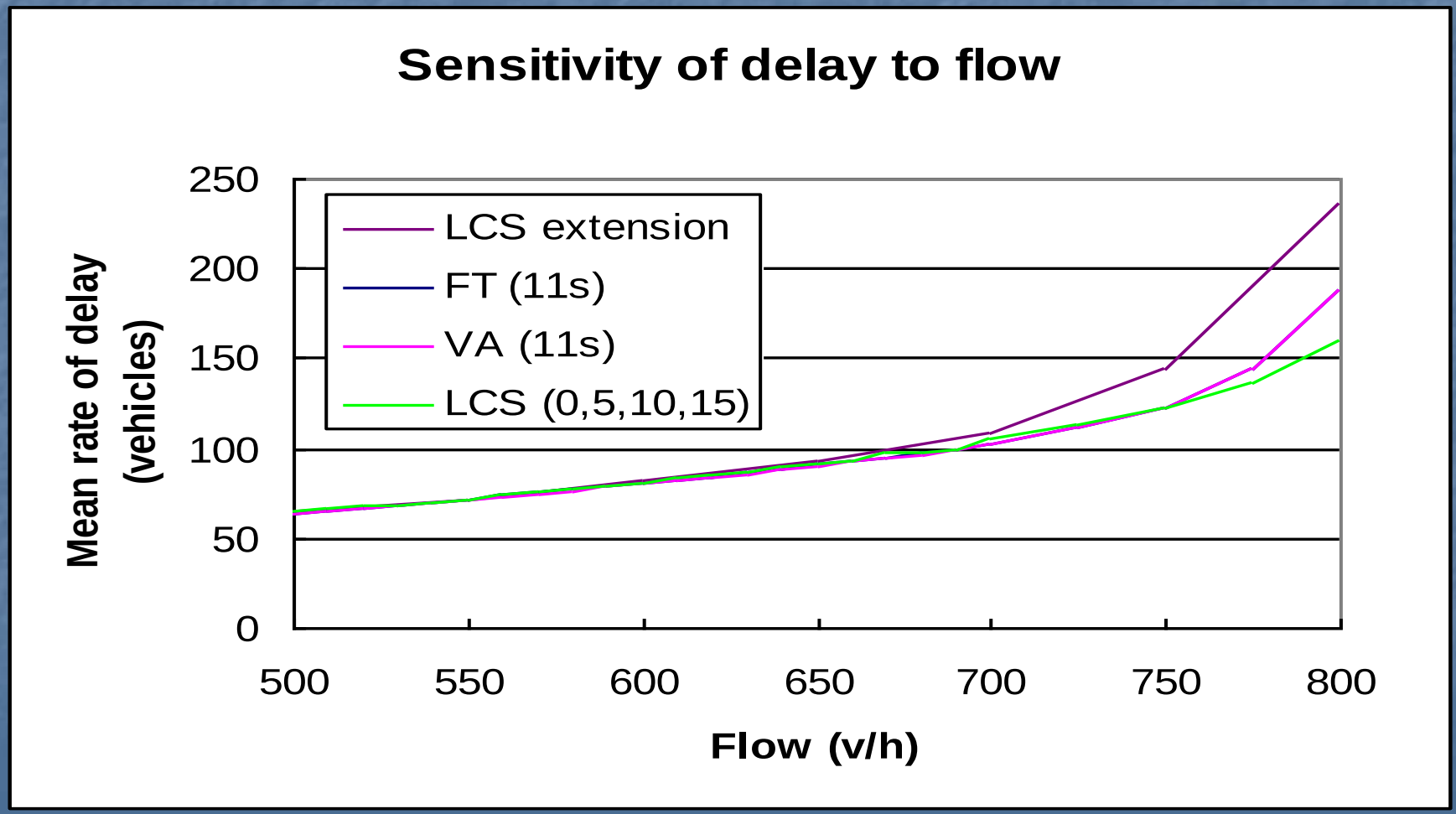


Figure 1: 4-junction network simulation (inset is blow-up of local junction controller signal flow)

Fitness

- Before each new cycle, a fitness increment is calculated for the junction controller with respect to the longest queue detected on any approach lane.
- Each junction controller acts independently of the others in the network: fully distributed, local control.
- The evo learner was found to be as good as or better than a number of traditional control strategies for road traffic junctions.

Example


Sensitivity of delay to flow
Mean rate of delay (vehicles)
250
200
150
100
50
0
500
550
600
650
700
750
800
Flow (v/h)
LCS extension
FT (11s)
VA (11s)
LCS (0,5,10,15)

Evolutionary Algorithms out there

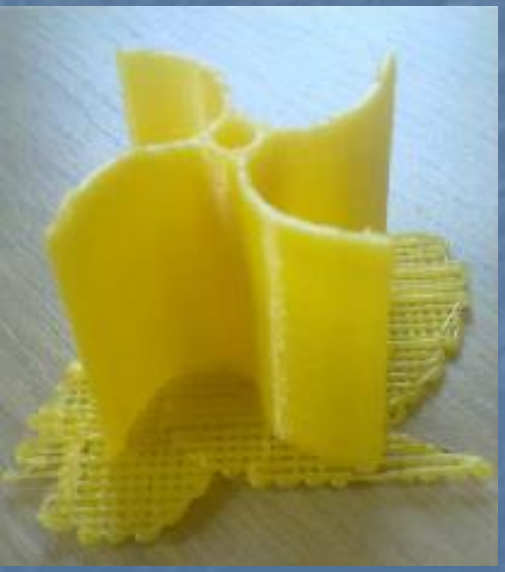
- When applying EAs to real-world problems a number of modifications must typically be made to the canonical EA:
  - Non-binary representations
  - Permutation problems
  - More sophisticated search operators

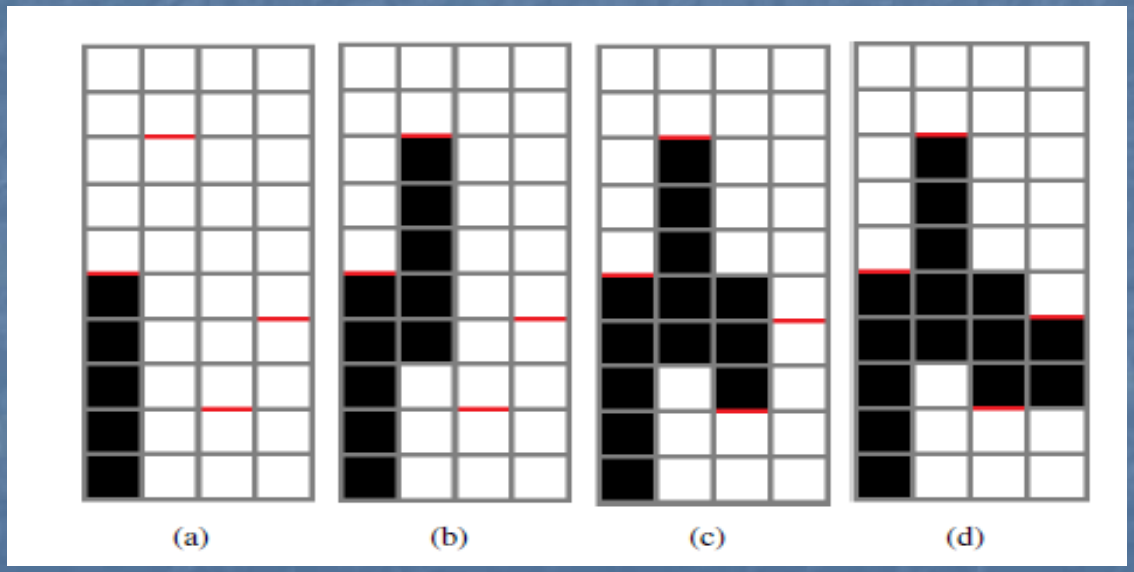
Integer representations

- Some problems naturally have integer variables, e.g., image processing light/colour parameters
- Others take *categorical* values from a fixed set, e.g., {blue, green, yellow, pink}
- Crossover works as standard
- Extend bit-flipping mutation to
  - "creep" i.e., more likely to move to similar value (+/- 1)
  - Random choice (esp. categorical variables)


Wind

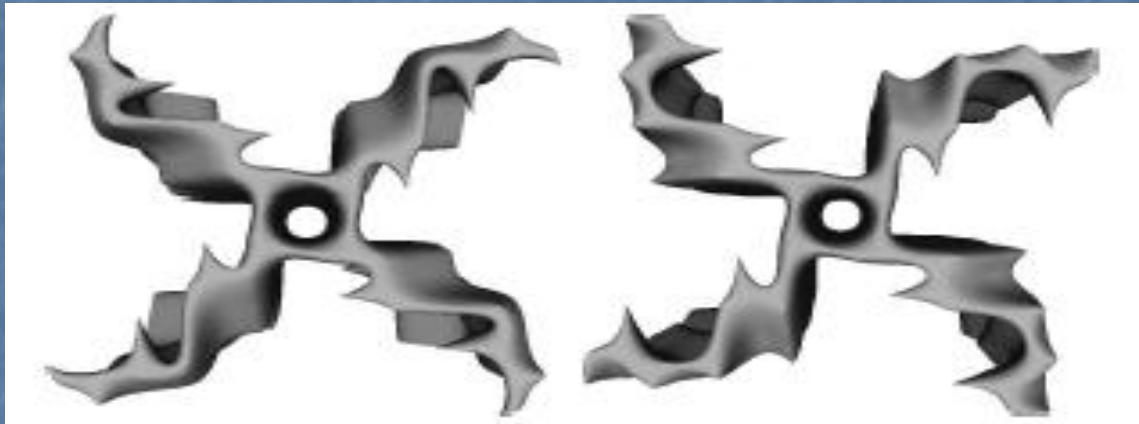

- Confined design space to a four-blade Savonius Vertical Axis Wind turbine with alterations in blade shape (profile and twist) possible.



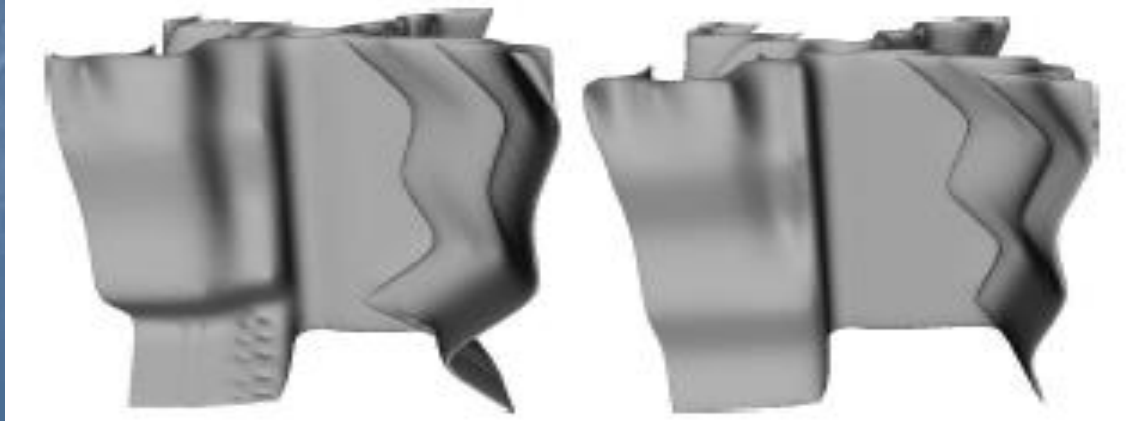


- E.g., genome [5,8,2,4] defines four offsets from central spindle for turbine blade.

Some 3D Printed Designs



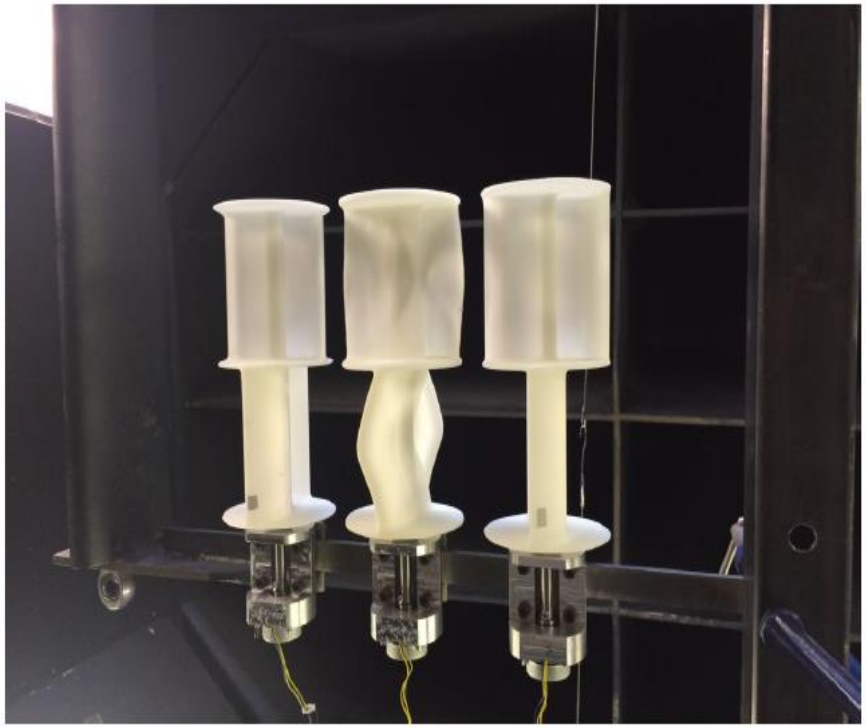
(a) 4th Gen — top view

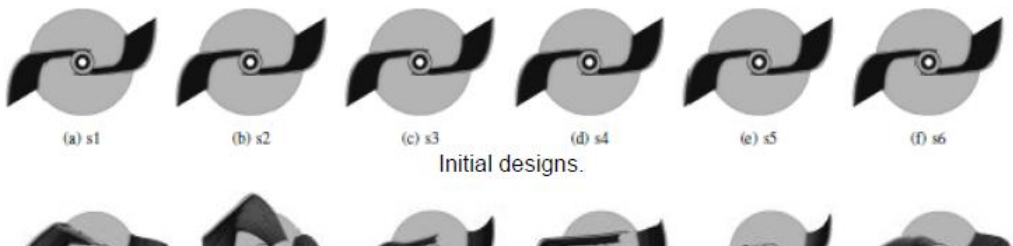


(b) 4th Gen — side view

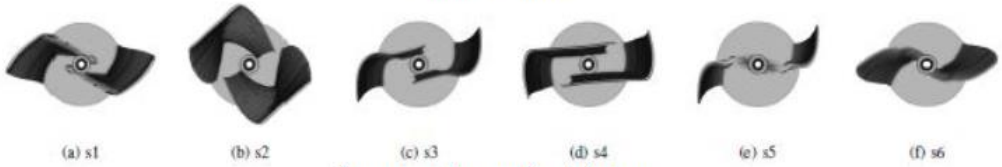
Turbine Arrays

- Performance can be increased by exploitation of inter-turbine flow effects in arrays of VAWT.
- Very costly (impossible?) to accurately model turbine arrays where multiple interactions exist.
- View wind farm as an energy-capture ecosystem and coevolve heterogeneous, interacting VAWT.





Initial designs.



Corresponding evolved designs.

Bench testing.

Wind tunnel testing.

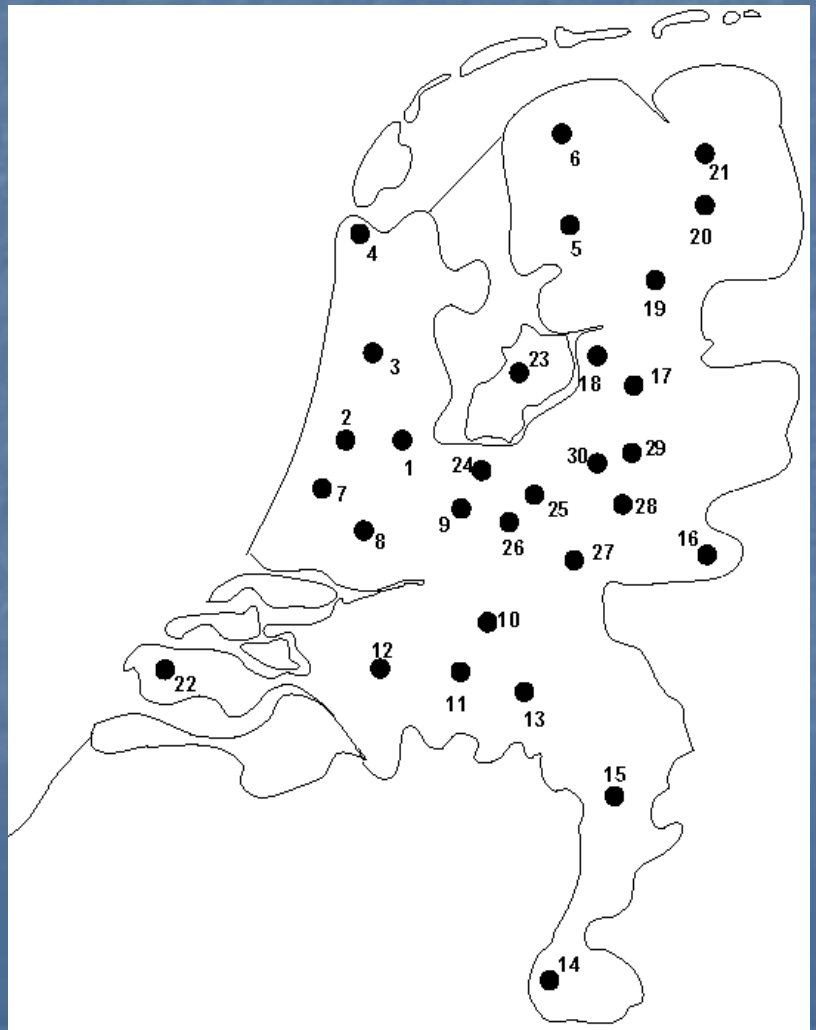
Permutation Representations

- Ordering/sequencing problems form a special class
- Task is (or can be solved by) arranging some objects in a certain order, e.g.:
  - Sort algorithm: important thing is which elements occur before others (order)
  - Travelling Salesman Problem (TSP) : important thing is which elements occur next to each other (adjacency)
- These problems are generally expressed as a permutation:
  - if there are *n* variables then the representation is as a list of *n* encodings, each of which occurs exactly once

The TSP

- Problem:
  - Given n cities
  - Find a complete tour with minimal length
- Encoding:
  - Label the cities 1, 2, … , *n*
  - One complete tour is one permutation (e.g., for n =4 [1,2,3,4], [3,4,2,1] are OK)
- Search space is BIG:

  for 30 cities there are 30! ≈ 10^{32} possible tours



Mutation operators for permutations

- Normal mutation operators lead to inadmissible solutions
  - e.g. bit-wise mutation : let gene *i* have value *j*
  - changing to some other value *k* would mean that *k* occurred twice and *j* no longer occurred
- Therefore must change at least two values
- Mutation parameter now reflects the probability that some operator is applied once to the whole string, rather than individually in each position

Insert Mutation for permutations

- Pick two allele values at random
- Move the second to follow the first, shifting the rest along to accommodate
- Note that this preserves most of the order and the adjacency information

Scramble mutation for permutations

- Pick a subset of genes at random
- Randomly rearrange the alleles in those positions

(note subset does not have to be contiguous)

Demonstration: magic square

- Given a 10x10 grid with a small 3x3 square in it
- Problem: arrange the numbers 1-100 on the grid such that
  - all horizontal, vertical, diagonal sums are equal (505)
  - a small 3x3 square forms a solution for 1-9

Contd.

Typical approach to solving this puzzle:

- Creating random begin arrangement
- Make *N* mutants of given arrangement
- Keeping the mutant (child) with the least error
- Stop when error is zero

Contd.

- Software by M. Herdy, TU Berlin
- Interesting parameters:
  - Step1: small mutation, slow & hits the optimum
  - Step10: large mutation, fast & misses (“jumps over” optimum)
  - Mstep: mutation step size modified on-line, fast & hits optimum
- Start: double-click on icon below
- Exit: click on TUBerlin logo (top-right)

Application

Crossover operators for permutations

- "Normal" crossover operators will often lead to inadmissible solutions

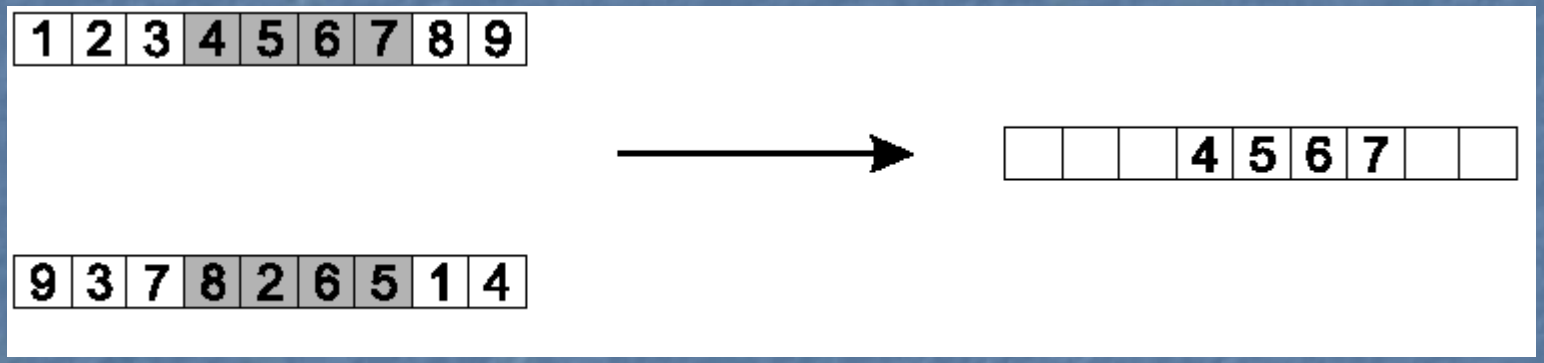
Many specialised operators have been devised which focus on combining order or adjacency information from the two parents

Order 1 crossover

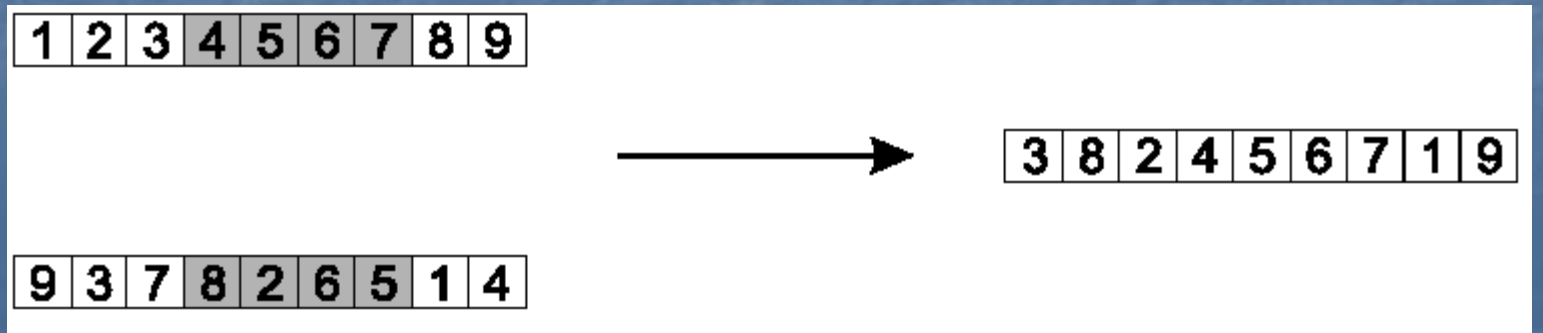
- Idea is to preserve relative order that elements occur
- Informal procedure:
  1. Choose an arbitrary part from the first parent
  2. Copy this part to the first child
  3. Copy the numbers that are not in the first part, to the first child:
     - starting right from cut point of the copied part,
     - using the **order** of the second parent
     - and wrapping around at the end
  4. Analogous for the second child, with parent roles reversed

Order 1 crossover example

- Copy randomly selected set from first parent



- Copy rest from second parent in order 1,9,3,8,2



Conclusions

- Simulated evolution can be used to design many different kinds of things.
- The variables of a problem may lend themselves better to different kinds of representation.
- The basic cycle remains the same but the search operators may need changing to generate new solutions in useful ways.