Genetic Algorithms

1. Genetics – and genetic algorithms
2. The travelling salesman
3. Resource optimisation problems
4. Clustering
5. Rule generation

Genetics and genetic algorithms

- Nature has found effective ways to solve many problems – we can mimic its approaches in “biologically-inspired” computer systems
  - Neural networks
  - Genetic algorithms (GAs)
  - Artificial life
  - Swarm intelligence

Optimisation problems

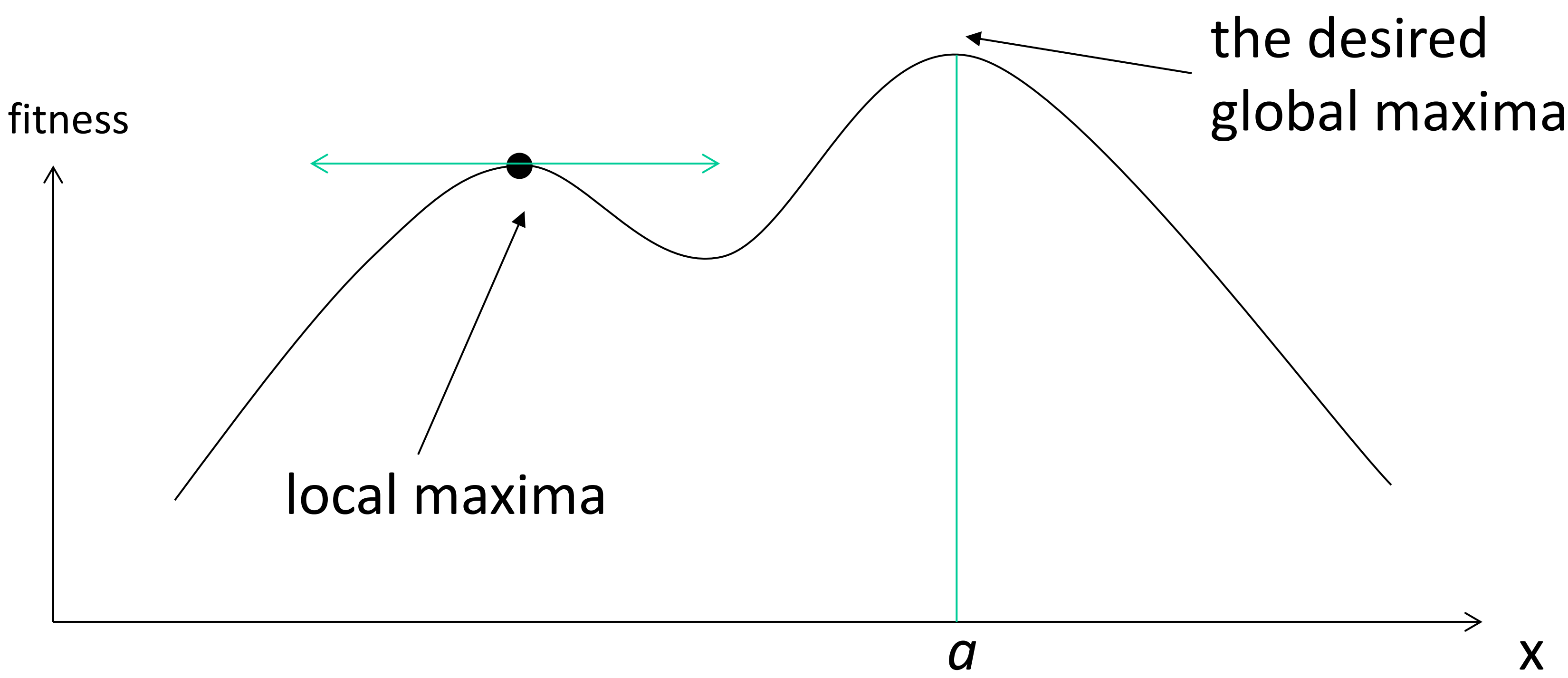
- Optimisation problems are about finding the best …
- e.g.,
  - The best allocation of classes to rooms
  - The best clustering
  - The best prediction of …
- Many data mining problems can be rephrased as optimisation problems

Optimisation problems

- Are phrased as:
  - A set of parameters (input variables)
  - A fitness function defined on these parameters (which measures "goodness")
  - A set of constraints (on the parameters)
- The problem is then to find a (or the) set of parameter values that satisfy the constraints & maximise the fitness function

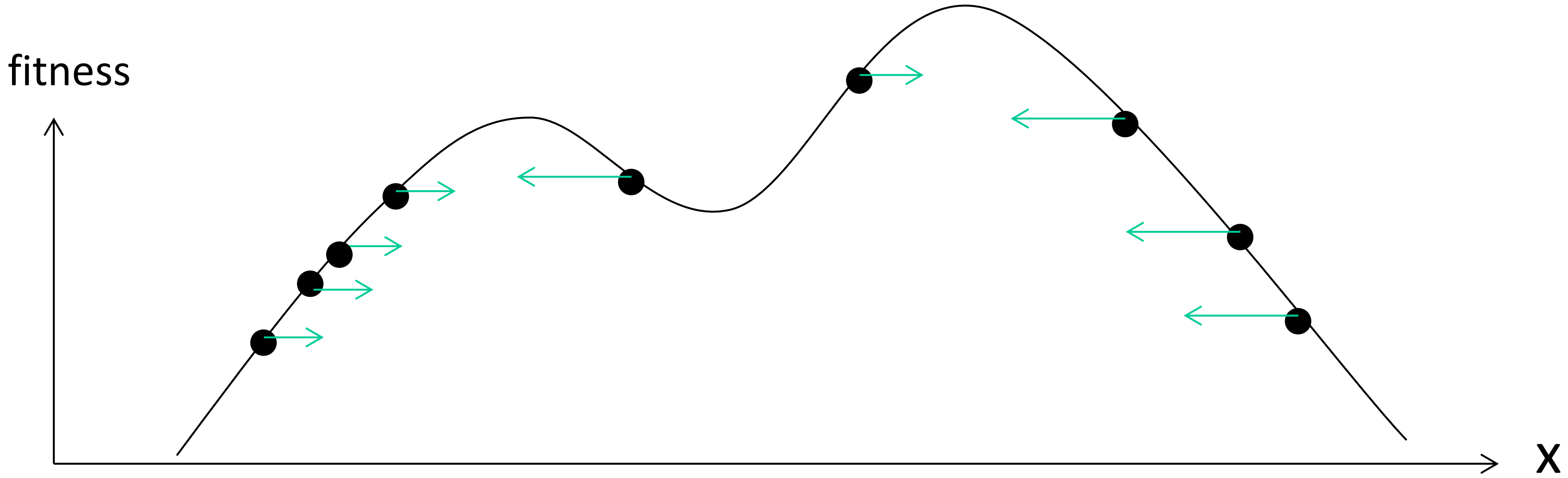
Here we have only one parameter x and no constraints

Optimisation and hill climbing



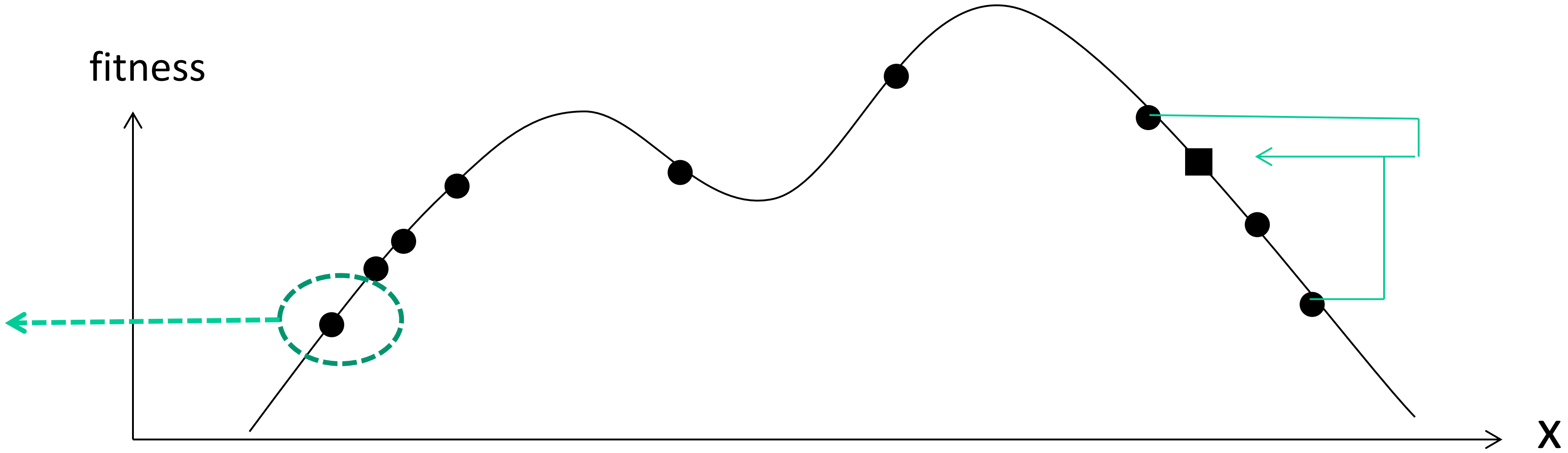
A hill climbing attempt to find the global maxima (i.e., to find *a*) can easily get stuck at the local maxima – because at each step it only makes a local analysis

A parallel attempt using hill climbing



Because of the parallelism, at least some of the partial solutions will (hopefully) reach the global maxima – even though some others get stuck at the local maxima

A parallel attempt using natural selection



Evolution/natural selection creates new individuals from old by breeding, the hope being that “good” individuals will have good children; less good individuals have a lower chance of survival (and hence of breeding)

Genetics

- DNA consists of a sequence of nucleotides, each being based on one of 4 sub-bases
  - adenine (A); cytosine (C); guanine (G); and thymine (T)
- Triples of the above represent amino acids
  - e.g., AAG represents Lysine
- A gene is a portion of the DNA string that
  - encodes a particular function
  - is the unit of heredity

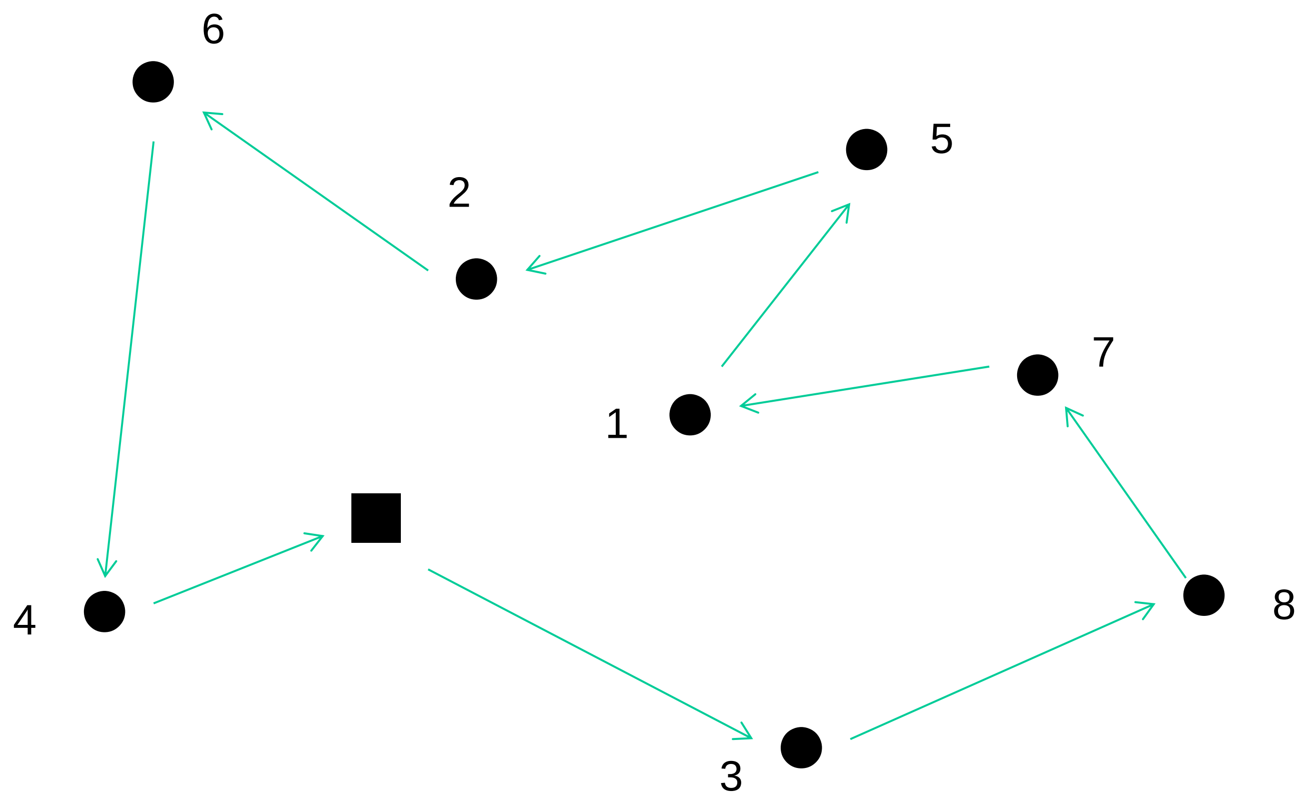
Genetics

- *Mutation* is where one of the genes is modified slightly in some random fashion
- *Crossover* (breeding) is the construction of a new DNA string – with some of the genes coming from the first parent and the rest coming from the other parent

Genetic algorithms

- Code up the problem as a DNA string
  - Each potential/partial solution is coded up as a particular DNA string (i.e., as an individual in our artificial world)
  - This coding is the creative bit
- Define fitness function
  - how good an individual DNA string is
- Define crossover and mutation operators
- Define initial population … and run: in each generation allow some of the individuals to breed/mutate and remove some of the less good individuals
- When we reach stability – i.e., we don’t seem to be improving our goodness of fit, . . .

2. The travelling salesman



■ = office ● = site to be visited

The travelling salesman problem

- For each pair of sites, there is a distance between them. The problem is simple:
  - what's the shortest route?
- Brute-force search: exponential complexity
- Important because:
  - A very wide class of computational problems are known to be equivalent to this problem
  - Unknown if there is any polynomial-time algorithm … general belief is "no"

Encoding the travelling salesman

- Number the sites
- Each potential solution is simply a sequence of site numbers
  - e.g., with 8 sites: (3,8,7,1,5,2,6,4)
- Fitness function is travel distance in the given order (low distance = high fitness)

Encoding the travelling salesman

- Mutation
  - Each value must appear once in the DNA string, therefore modifying one of the values does not make sense
  - Therefore mutation consists of interchanging two values

$$(\underline{3},8,7,\underline{1},5,2,6,4) \rightarrow (1,8,7,3,5,2,6,4)$$

Encoding the travelling salesman

- Crossover
  - Each value must appear once in the DNA string: some crossovers will not “make sense”
  - Given 2 individuals with a “common initial segment”, take the initial segment of one with the latter part of the other

$$(\underline{3,8,7},1,5,4,6,2)\ \&\ (8,7,3,\underline{2,5,1,6,4})$$

$$(3,8,7,2,5,1,6,4)$$

3. Resource allocation

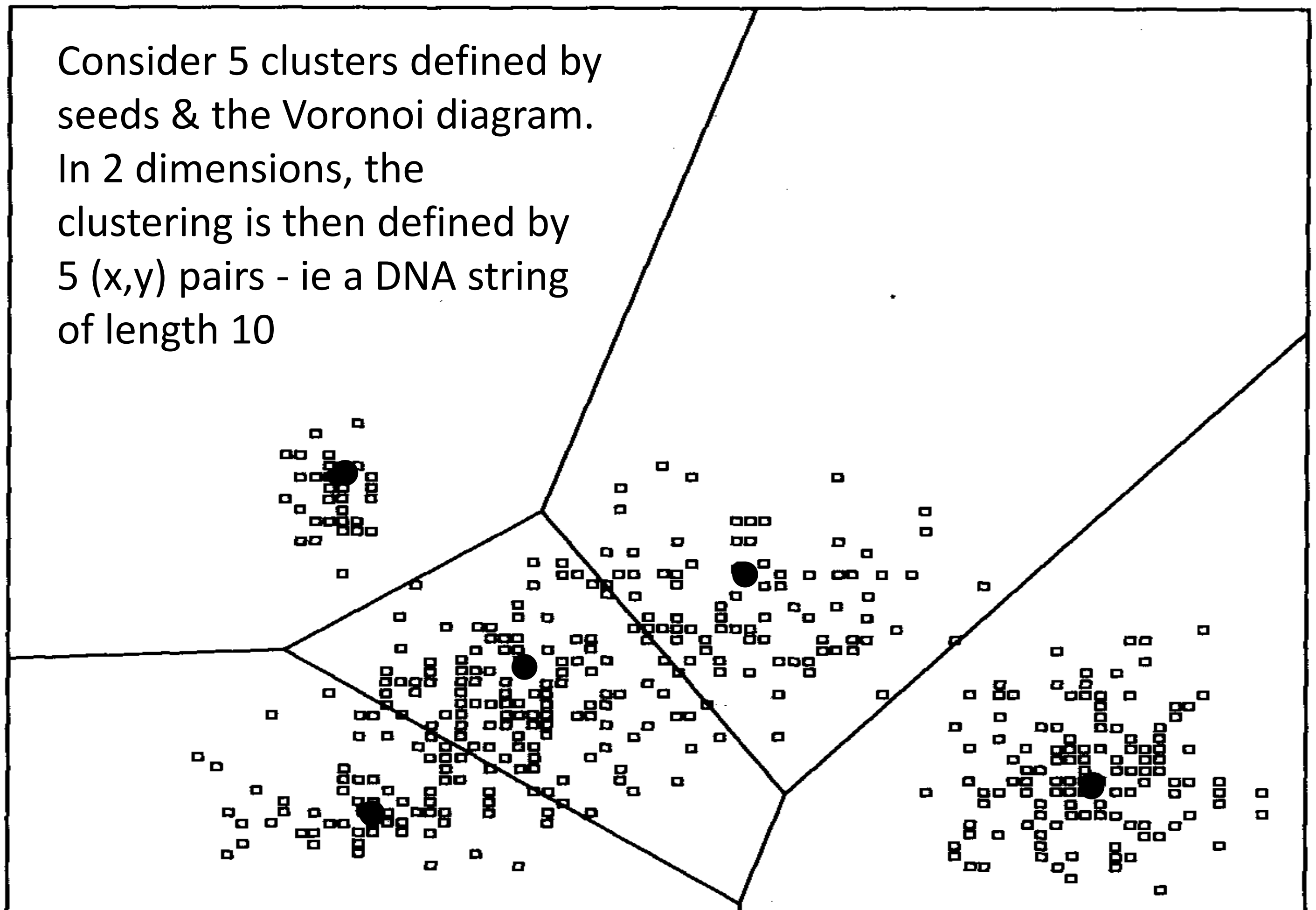
- Consider the timetabling problem
- There are a set of classes and rooms
- Constraints and preferences are obvious
  - Rooms should be bigger than classes
  - No double booking
  - All required classes are timetabled
  - Students & teachers should have at most X hours/day
  - Other teacher preferences

DNA construction

- All DNA strings are an array containing an entry for each “room slot”
  - Room; Date; Time
- The *value in* the room slot is the name of the class allocated
- Fitness can be defined in terms of the violation of preferences (soft constraints)

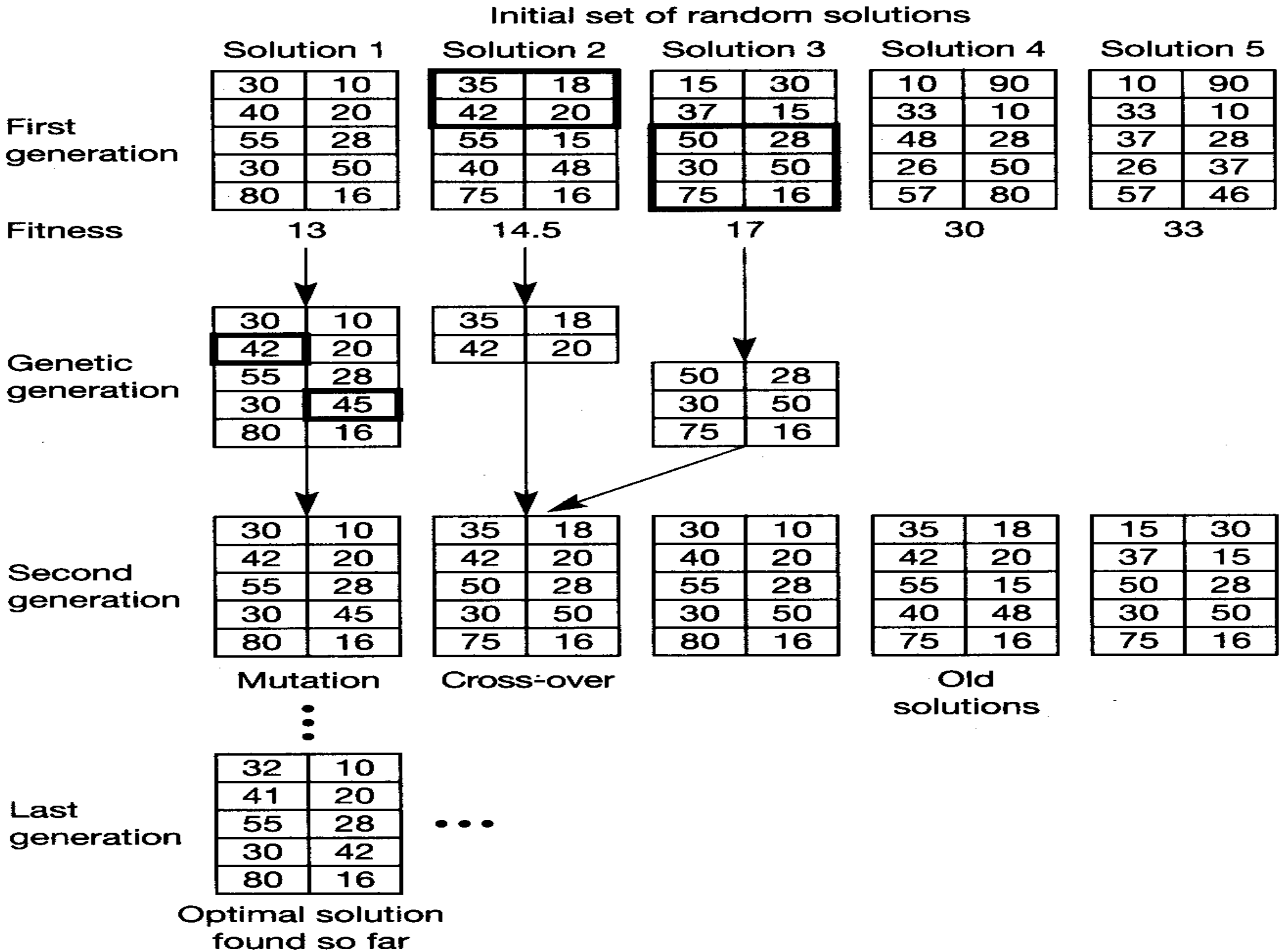
4. Clustering

Consider 5 clusters defined by seeds & the Voronoi diagram. In 2 dimensions, the clustering is then defined by 5 (x,y) pairs - ie a DNA string of length 10



Genetic Algorithms

- Fitness function as before
  - internal cohesion, external separation
- Cross over: take 3 points in one sequence and 2 in another - combine
- Mutation: slightly perturb one or more figures


Initial set of random solutions
Solution 1
Solution 2
Solution 3
Solution 4
Solution 5
First generation
30 10
40 20
55 28
30 50
80 16
35 18
42 20
55 15
40 48
75 16
15 30
37 15
50 28
30 50
75 16
10 90
33 10
48 28
26 50
57 80
10 90
33 10
37 28
26 37
57 46
Fitness
13
14.5
17
30
33
Genetic generation
30 10
42 20
55 28
30 45
80 16
35 18
42 20
50 28
30 50
75 16
Second generation
30 10
42 20
55 28
30 45
80 16
35 18
42 20
50 28
30 50
75 16
30 10
40 20
55 28
30 50
80 16
35 18
42 20
55 15
40 48
75 16
15 30
37 15
50 28
30 50
75 16
Mutation
Cross-over
Old solutions
Last generation
32 10
41 20
55 28
30 42
80 16
Optimal solution found so far

5. Rule generation

- For classification, a rule has the form

  $Cond_1$ & $Cond_2$ & … & $Cond_N$ → Class=C

- Conditions are typically of the form

  $A=v$; $A \neq v$; $A>v$; $A<v$

- Whatever the type of conditions allowed, the DNA string is obvious:

  (C, $Cond_1$, $Cond_2$, …, $Cond_N$, null, null, …)

Rule generation

- The fitness of a DNA string/rule

  $Cond_1$ & $Cond_2$ & … & $Cond_N$ $\rightarrow$ Class=C

  is clearly its predictive accuracy
- Mutation is straightforward
- Crossover is as previously defined – take some elements of one parent and some elements of the other