INTRODUCTION TO ARTIFICIAL INTELLIGENCE

Unit 7

Outline

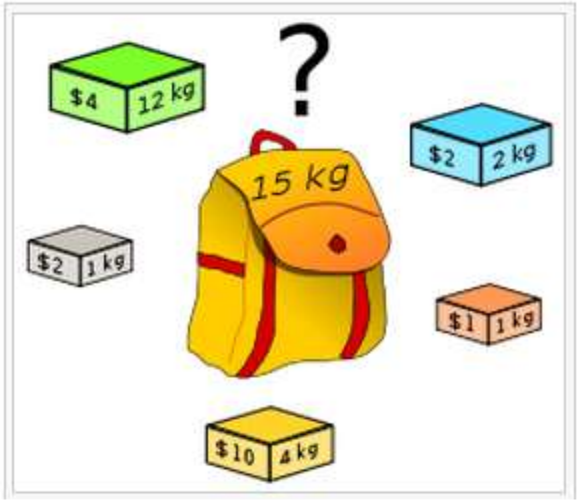
- Evolutionary Computation
  - Evolutionary Algorithm

Local Search / Optimization

- In many problems, you are only concerned about the optimum value and not about the whole path that lead you to that value.
- So goal state is important but the path that lead to the goal state is unimportant.
- Sometimes you do not exactly know about the goal node but you know certain characteristics of the goal node.

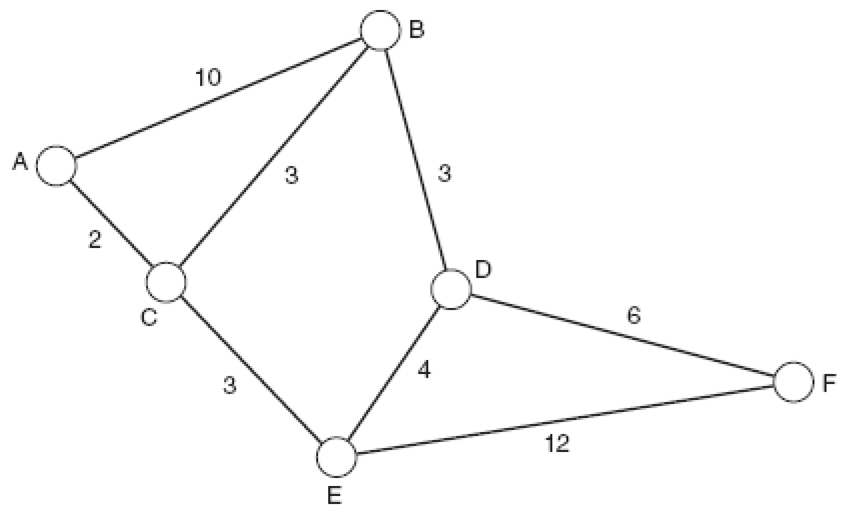
Knapsack Problem

- Given a set of items, each with a weight and a value, determine the number of each item to include in a collection so that the total weight is less than or equal to a given limit and the total value is as large as possible.



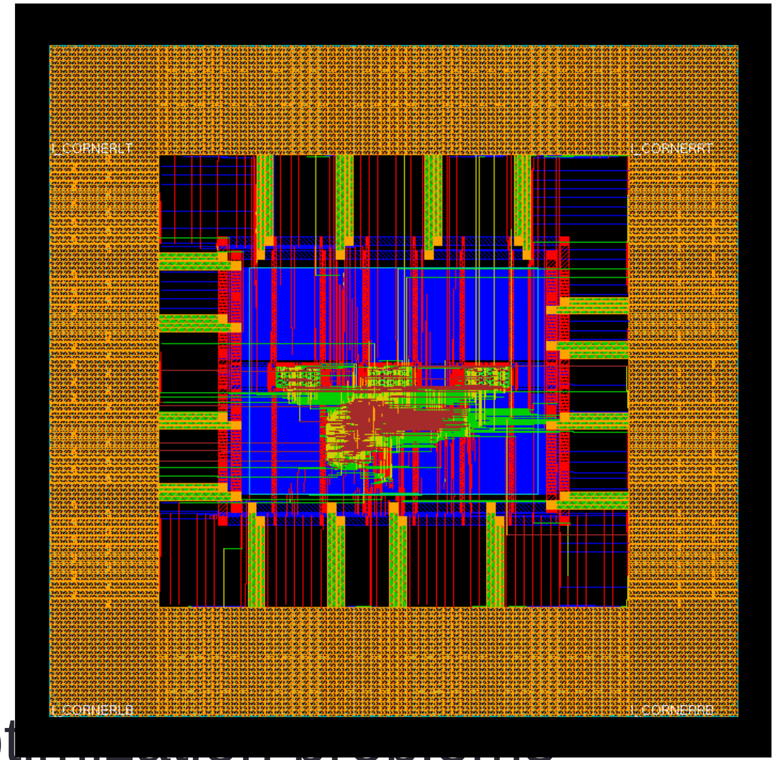
Travelling Salesman Problem

- Given a list of cities and the distances between each pair of cities, what is the shortest possible route that visits each city exactly once and returns to the origin city?



Other similar problems

- Resource Allocations
- Scheduling Problems
- Factory floor layout
- Job-Shop scheduling
- Vehicle routing problem
- VLSI Design
- ....
- And many more combinatorial optimization problems

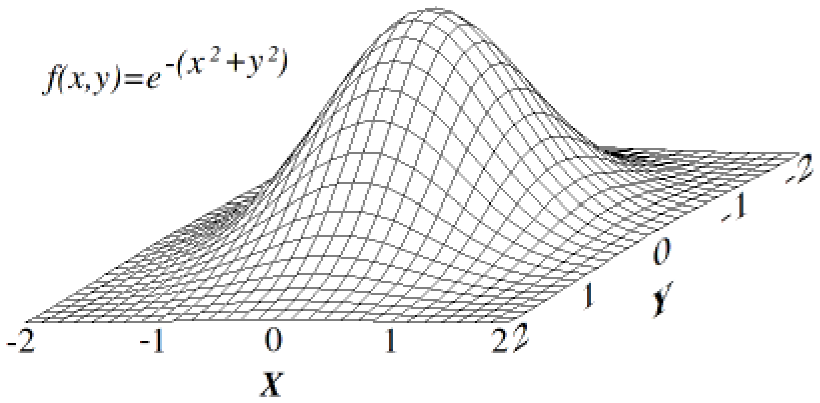


Hill Climbing

- General Idea:
  - Start wherever
  - Always choose the best neighbor
  - If no neighbors have better scores than current, quit
- Why can this be a terrible idea?

Local vs Global Optimum?

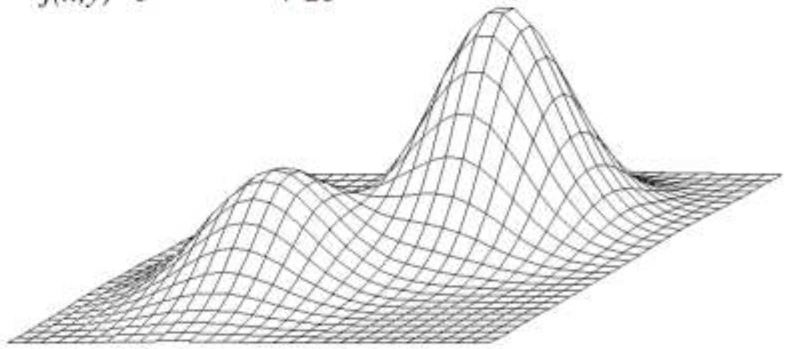
What is the optimum point?



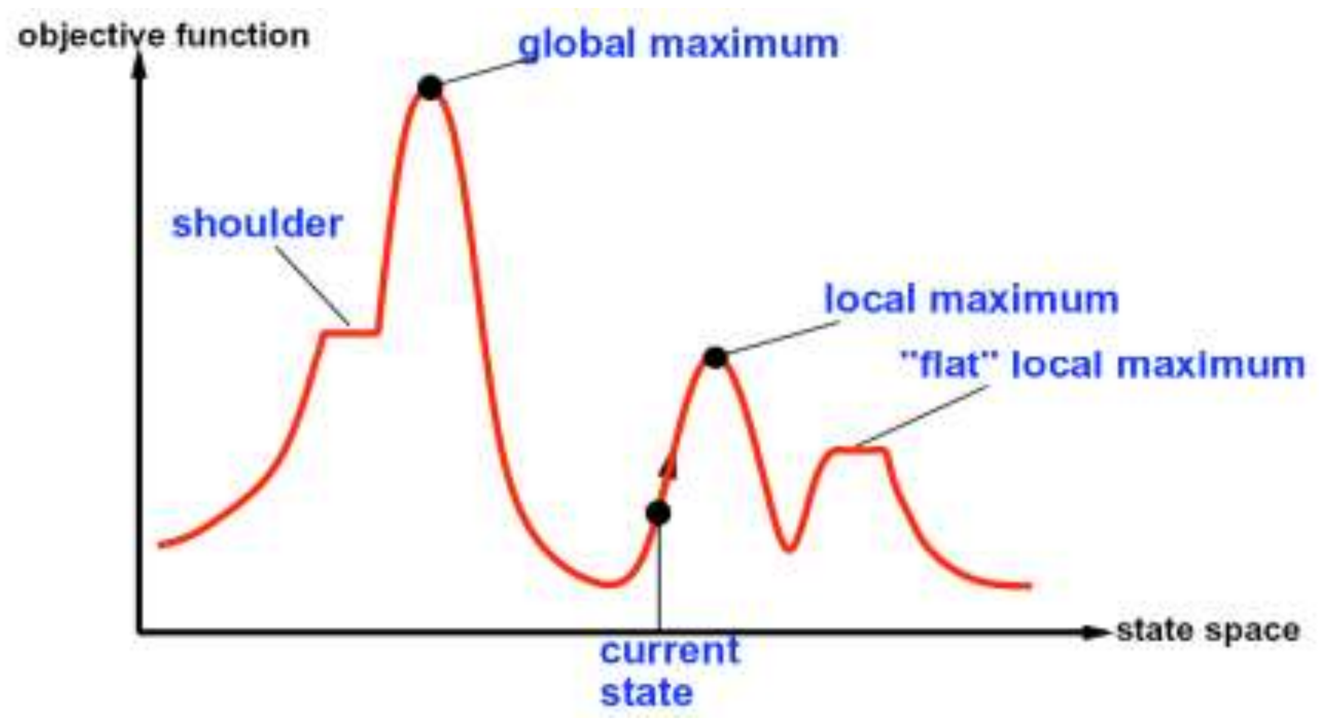
Local vs Global Optimum?

What do you think about the optimum point now?

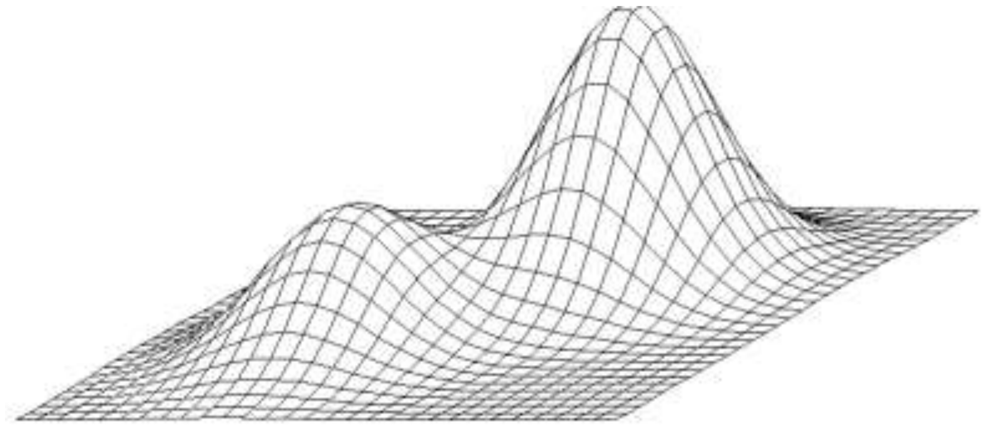
$$f(x,y)=e^{-(x^2+y^2)}+2e^{-((x-1.7)^2+(y-1.7)^2)}$$



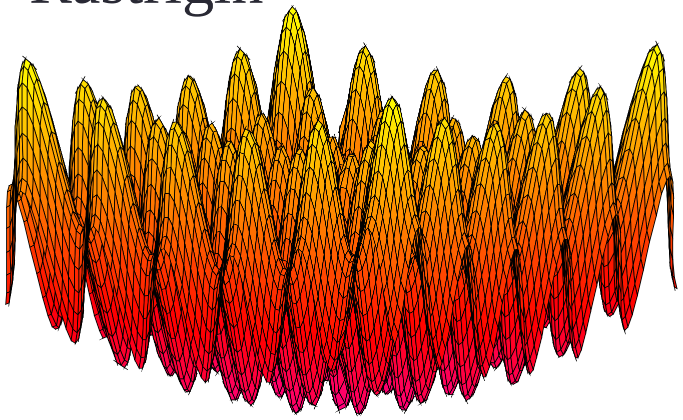
Solution Space Landscape



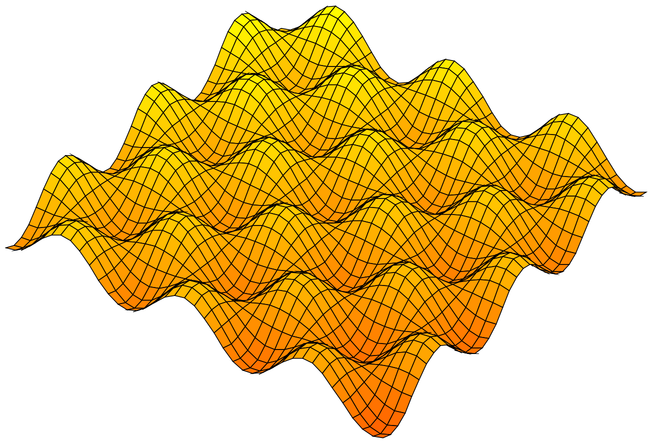
Complex Functions



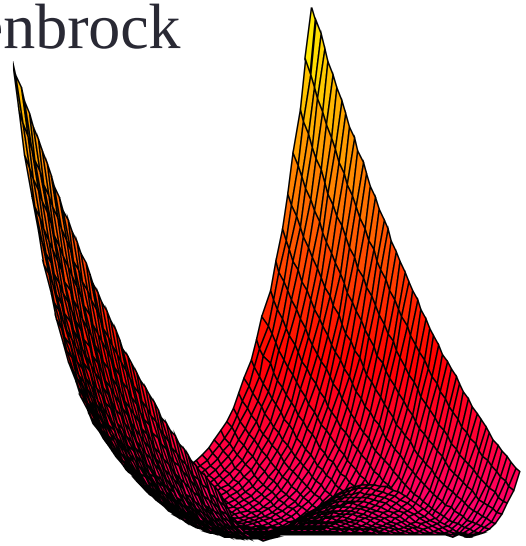
Rastrigin



Griewank

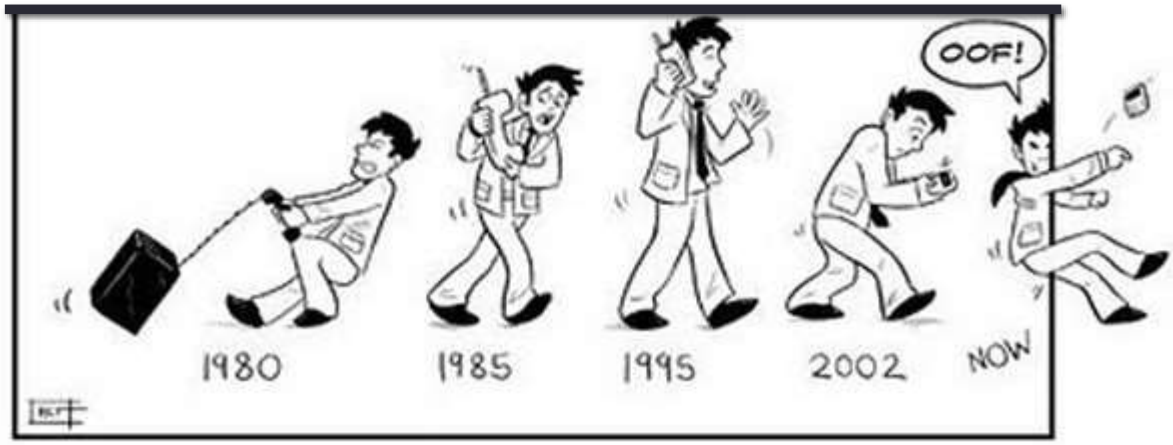


Rosenbrock



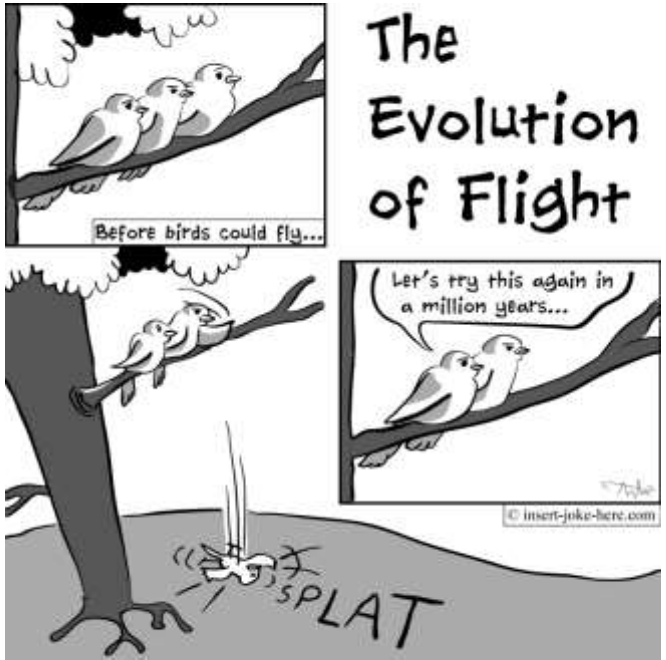
Evolution

- **Evolution** is change in the heritable characteristics of biological populations over successive generations. Evolutionary processes give rise to biodiversity at every level of biological organization, including the levels of species, individual organisms, and molecules.



Evolution for Problem Solving

- Evolution in itself is a mechanism of incremental search, whereby more fit solutions to problems propagate to future generations, and less fit solutions gradually fade away.
- This process of natural selection provides a wonderful vehicle for finding solutions to difficult multivariate optimization problems.


The Evolution of Flight
Before birds could fly...
SPLAT
Let's try this again in a million years...
© insert-joke-here.com

Evolutionary Algorithm (Source: Wikipedia)

- In artificial intelligence, an evolutionary algorithm (EA) is a subset of evolutionary computation, a generic population-based meta-heuristic optimization algorithm.
- An EA uses some mechanisms inspired by biological evolution: reproduction, mutation, recombination, and selection.
- Candidate solutions to the optimization problem play the role of individuals in a population, and the fitness function determines the environment within which the solutions "live" .
- Evolution of the population then takes place after the repeated application of the above operators.

Overview of Evolutionary Algorithms

- A parallel search scheme that is inspired by biological evolution
- EAs have been used successfully for optimization problems in several fields
- They make little assumptions about the landscape
- Works for problem where classical schemes fail such as
  - When derivate of a function does not exist
  - The function is discontinuous

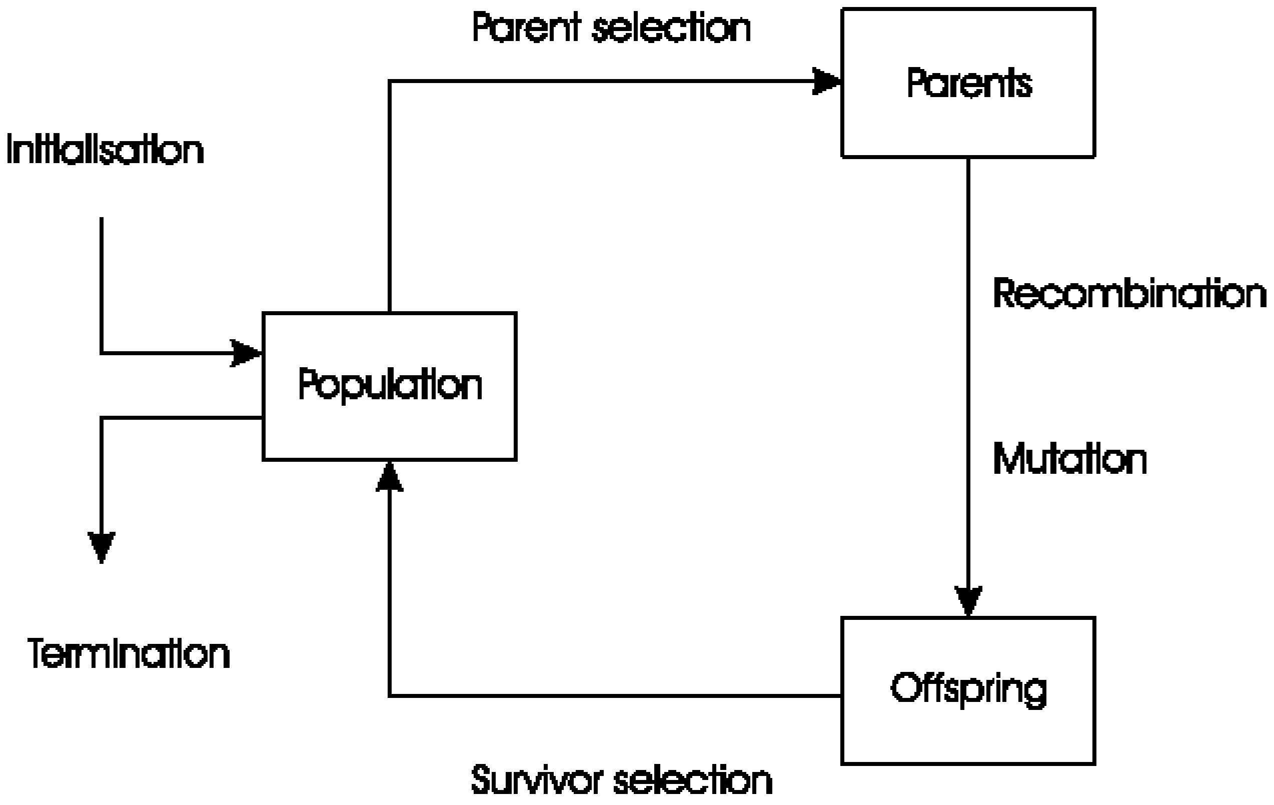
History

- Several efforts were started in parallel during 1960s to solve a variety of problems using more or less the same approach
  - Evolutionary Strategies (Berlin Technical University)
  - Genetic Algorithms (University of Michigan)
  - Evolutionary Programming (UCLA)
- During 1990s the above communities agreed to the term "Evolutionary Algorithms".

Evolutionary Computing

- Evolutionary Algorithms
  - Genetic Algorithms
  - Evolutionary Programming
  - Evolutionary Strategies
  - Genetic Programming
- Swarm Intelligence
  - Particle Swarm Optimization
  - Ant Colony Optimization
- Other Nature Inspired Models
  - Artificial Immune Systems
  - Differential Evolution
  - Honey Bee Optimization
  - Coevolution

General Scheme of EAs



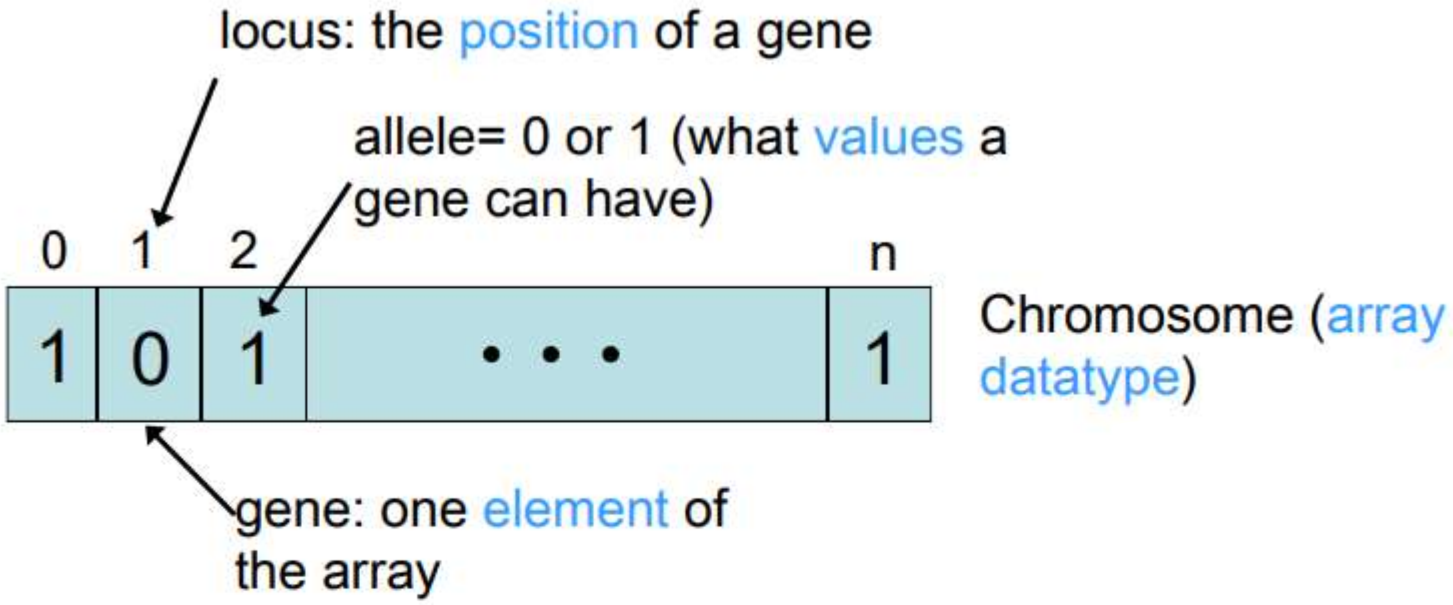
A Typical Evolutionary Algorithm Cycle

- Step 1: Initialize the population randomly or with potentially good ***solutions***.
- Step 2: Compute the ***fitness*** of each individual in the population.
- Step 3: Select parents using a ***selection procedure***.
- Step 4: Create offspring by ***crossover*** and ***mutation*** operators.
- Step 5: Compute the ***fitness*** of the new offspring.
- Step 6: Select members of population to die using a ***selection procedure***.
- Step 7: Go to Step 2 until termination criteria are met.

Important Concepts

- Chromosome (Candidate Solution)
- Fitness Function
- Parent Selection (Pre-selection)
- Crossover (Exploration)
- Mutation (Exploitation)
- Survivor Selection (Post-selection)

Representation: EA Terms

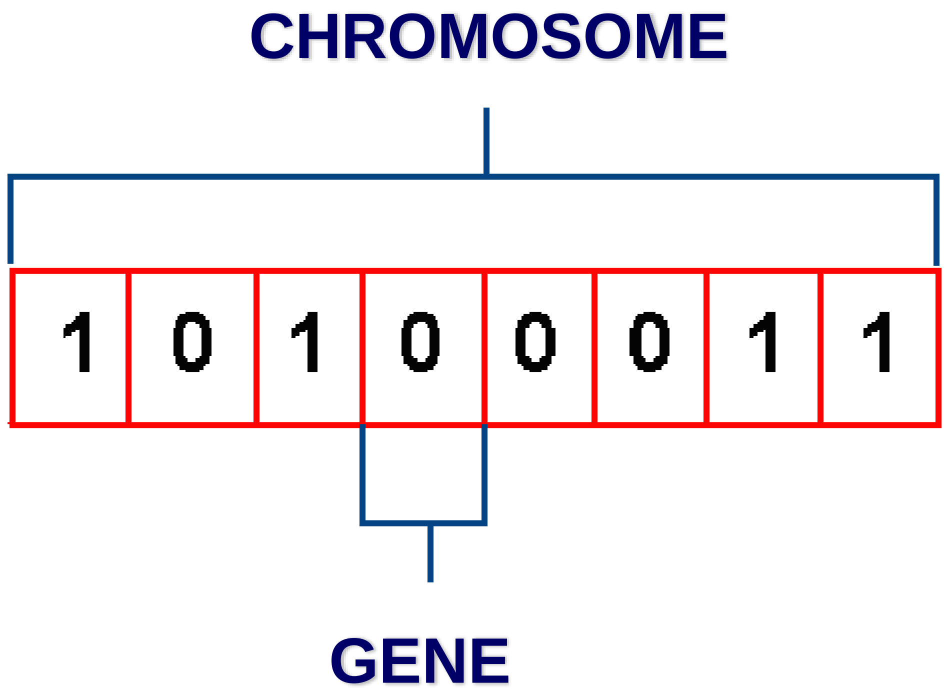

locus: the position of a gene
allele= 0 or 1 (what values a gene can have)
0
1
2
n
1
0
1
• • •
1
Chromosome (array datatype)
gene: one element of the array

Most common representation of Chromosomes:

- – Binary
- – Integer
- – Real-Valued or Floating-Point
- – Permutation
- – Tree

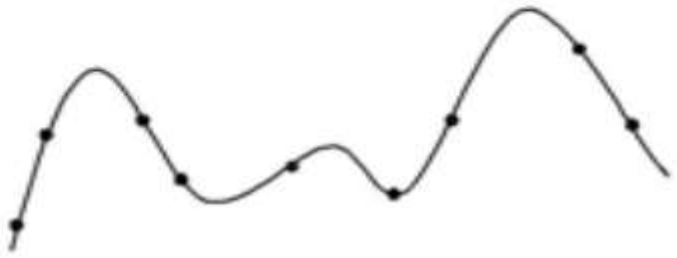
Example: Discrete Representation (Binary alphabet)

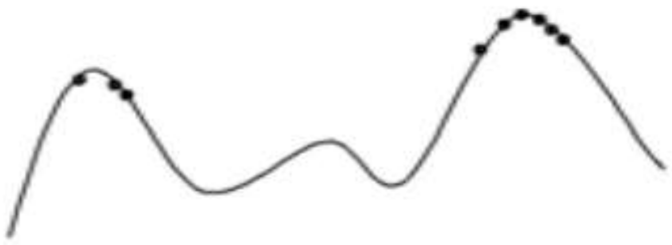
- Representation of an individual can be using discrete values (binary, integer, or any other system with a discrete set of values).
- Following is an example of binary representation.



Exploitation vs. Exploration

- Any efficient optimization algorithm must use two techniques to find a global maximum
- Exploration:
  - to investigate new and unknown areas in the search space
- Exploitation:
  - to make use of knowledge found at points previously visited to help find better points
- These two requirements are contradictory, and a good search algorithm must find a tradeoff between the two.





Crossover

- Merges information from parents into offspring
- Choice of what information to merge is stochastic
- Most offspring may be worse, or the same as the parents
- Hope is that some are better by combining elements of genotypes that lead to good traits

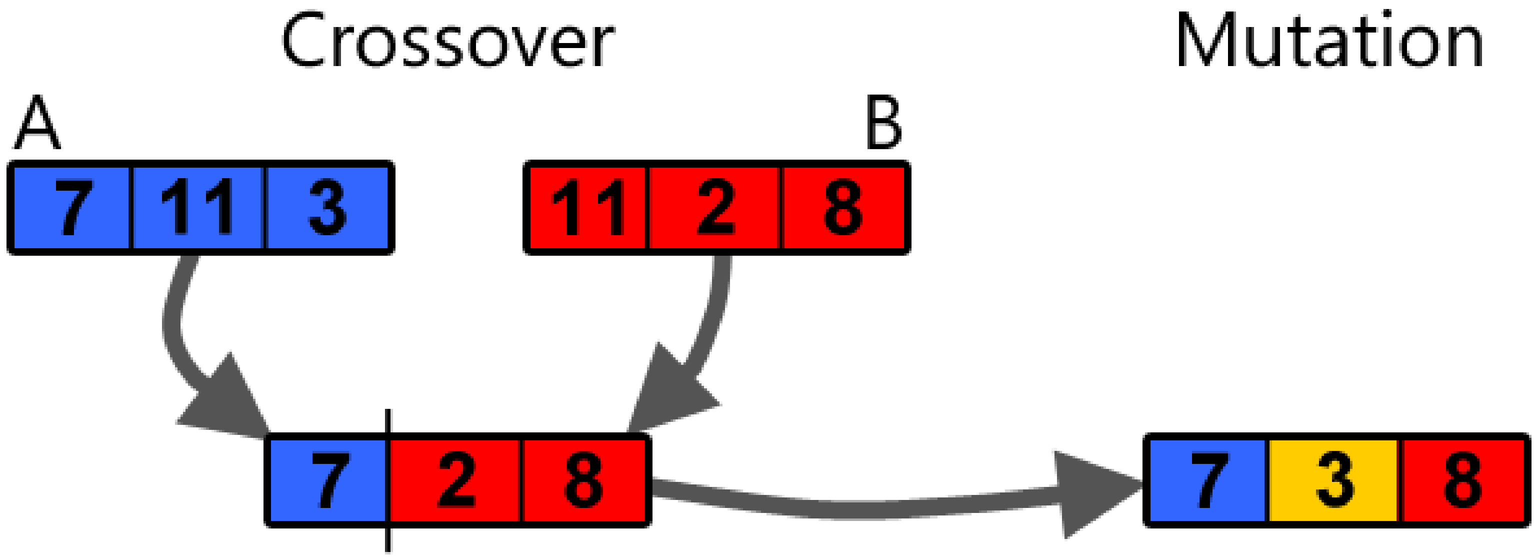
Mutation

- After creation of new individuals via crossover, mutation is applied usually with a low probability to introduce random changes into the population
  - Replace gene values lost from the population or not initially present
  - Evaluate more regions of the search space
  - Avoid premature convergence
  - Makes the entire search space reachable
  -
- If applied at high probability, the offspring will lose their resemblance to their parents and the ability of the algorithm to learn from the parents will be lost

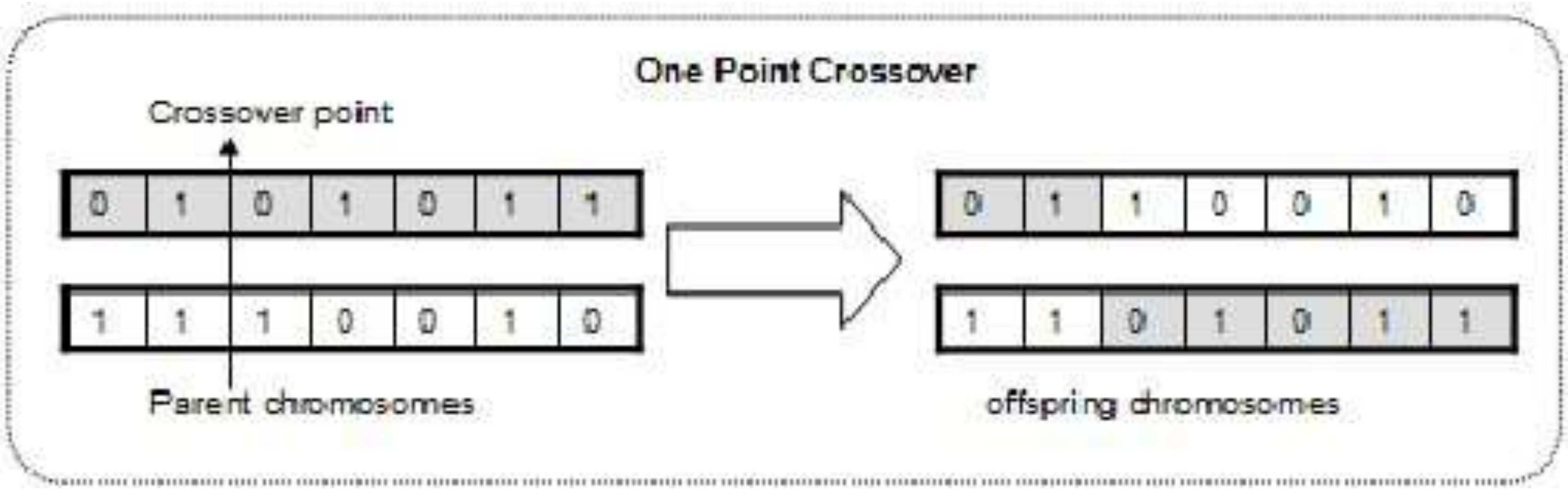
Crossover OR Mutation?

- Exploration: Discovering promising areas in the search space, i.e. gaining information on the problem
- Exploitation: Optimizing within a promising area, i.e. using information
- There is co-operation AND competition between them
- Crossover is explorative, it makes a big jump to an area somewhere “in between” two (parent) areas
- Mutation is exploitative, it creates random small diversions, thereby staying near (in the area of) the parent
- Only crossover can combine information from two parents
- Only mutation can introduce new information (alleles)

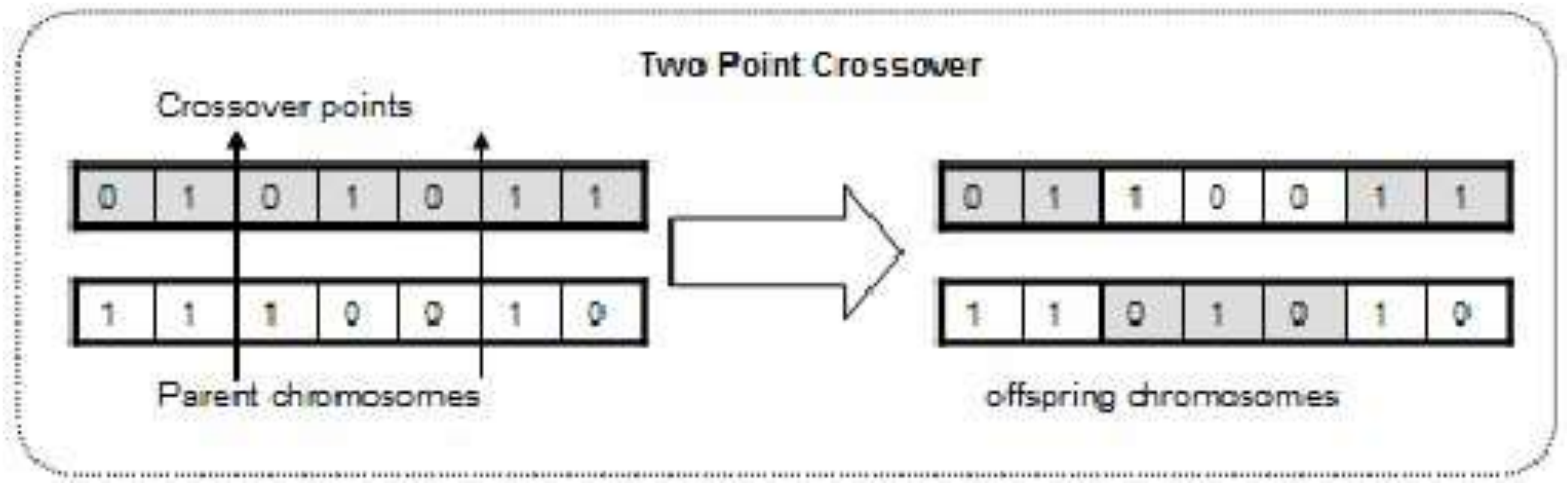
How to produce new solutions?



One Point Crossover

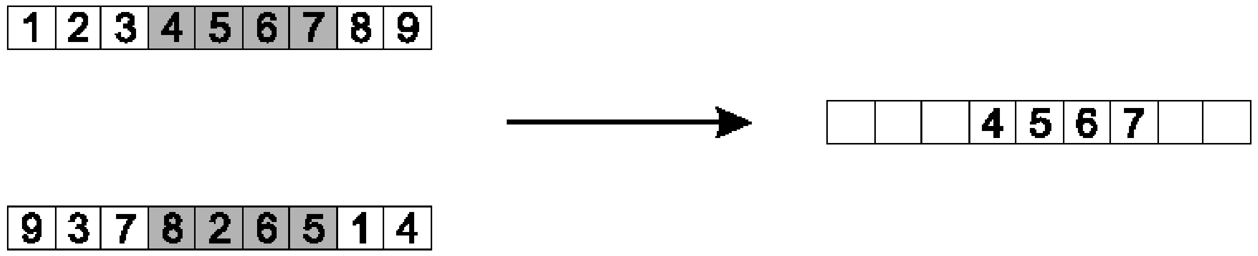


Two Point Crossover

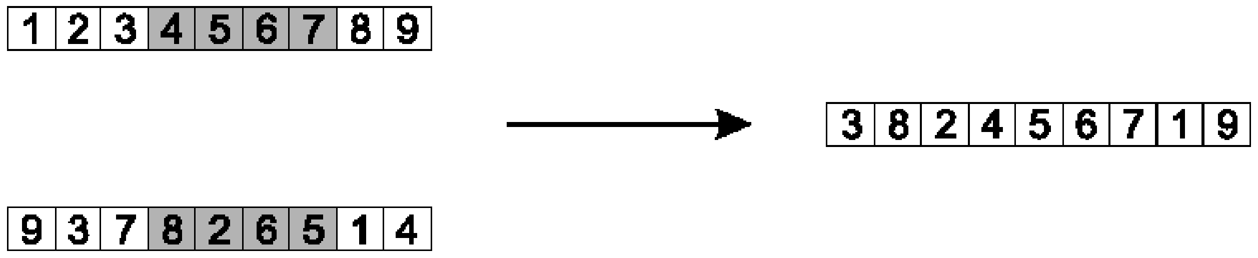


Two Point Crossover – Example

- Copy randomly selected set from first parent



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

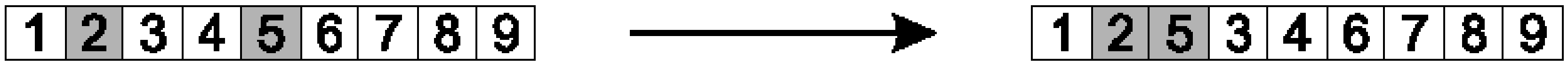


Permutation Representations

- Useful in ordering/sequencing problems
- Task is (or can be solved by) arranging some objects in a certain order.
- Examples:
  - Production scheduling: important thing is which elements are scheduled before others (order)
  - Travelling Salesman Problem (TSP) : important thing is which elements occur next to each other (adjacency)
- If there are n variables then the representation is as a list of n integers, each of which occurs exactly once

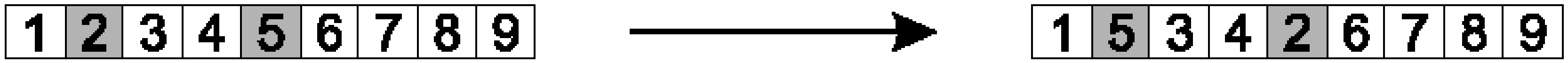
Insert Mutation for Permutations

- 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



Swap Mutation for Permutations

- Pick two alleles at random and swap their positions
- Preserves most of adjacency information (4 links broken), disrupts order more



Selection Procedure

- Selection in evolutionary algorithms is the process of choosing which individuals reproduce offspring and which individuals survive to the next generation
- When selection is used to choose which individuals reproduce, the process is referred to as ***pre-selection*** (parent(s) selection)
- When it is used to select the individuals that survive to the next generation it is called ***post-selection*** (survival selection)

SELECTION SCHEMES

Ahh, the first day teaching Natural Selection is always the best day.

Fitness Proportional

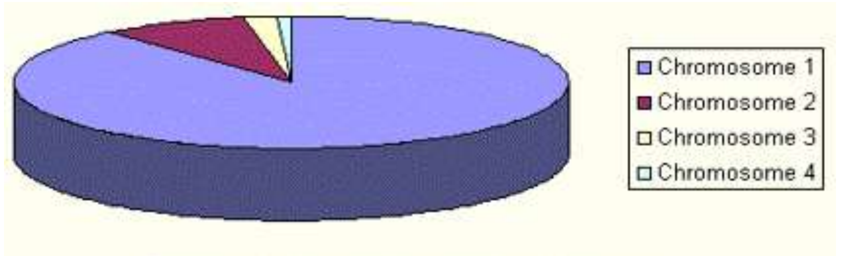
- Individuals are selected based on their fitness in proportion to the other individuals in the population.
- Suffers from selection pressure if an individual dominates the population.

Fitness Proportional Selection (continued)

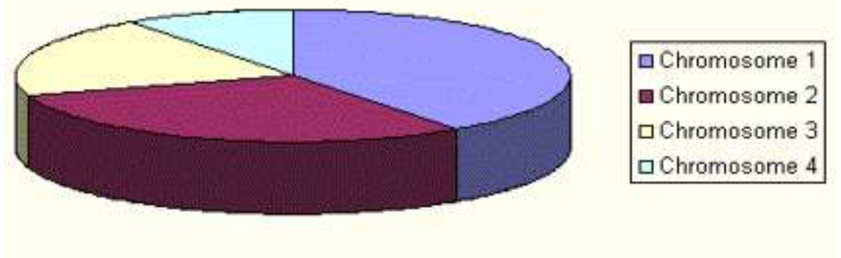
- Main idea: better individuals get higher chance
  - Chances proportional to fitness
  - Implementation: roulette wheel technique
    - Assign to each individual a part of the roulette wheel
    - Spin the wheel n times to select n individuals

Rank Based Selection

- Individuals are selected based on their rank rather than fitness to reduce the bias large fitness values might have (reduce the selection pressure)



Situation before ranking (graph of fitnesses)



Situation after ranking (graph of order numbers)

Rank Based Selection (continued)

- First sort the Fitness value of the Population.
- Then if the Population number is 10 then give the probability of selection to the Population like 0.1,0.2,0.3,…,1.0 .
- Then calculate cumulative Fitness and make roulette wheel.
- And the next steps is same as roulette wheel.

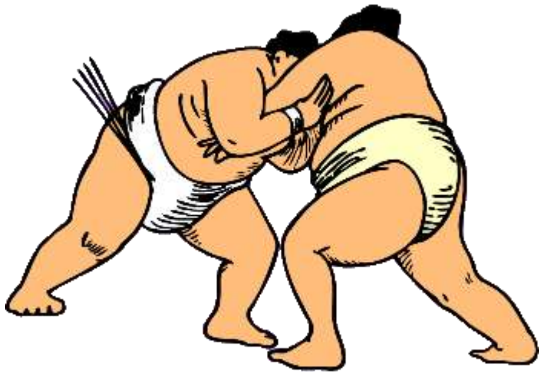
Rank Based Selection (continued)

- Rank-based selection schemes can avoid premature convergence.
- But can be computationally expensive because it sorts the populations based on fitness value.
- But this method can lead to slower convergence, because the best chromosomes do not differ so much from other ones

Solve Example

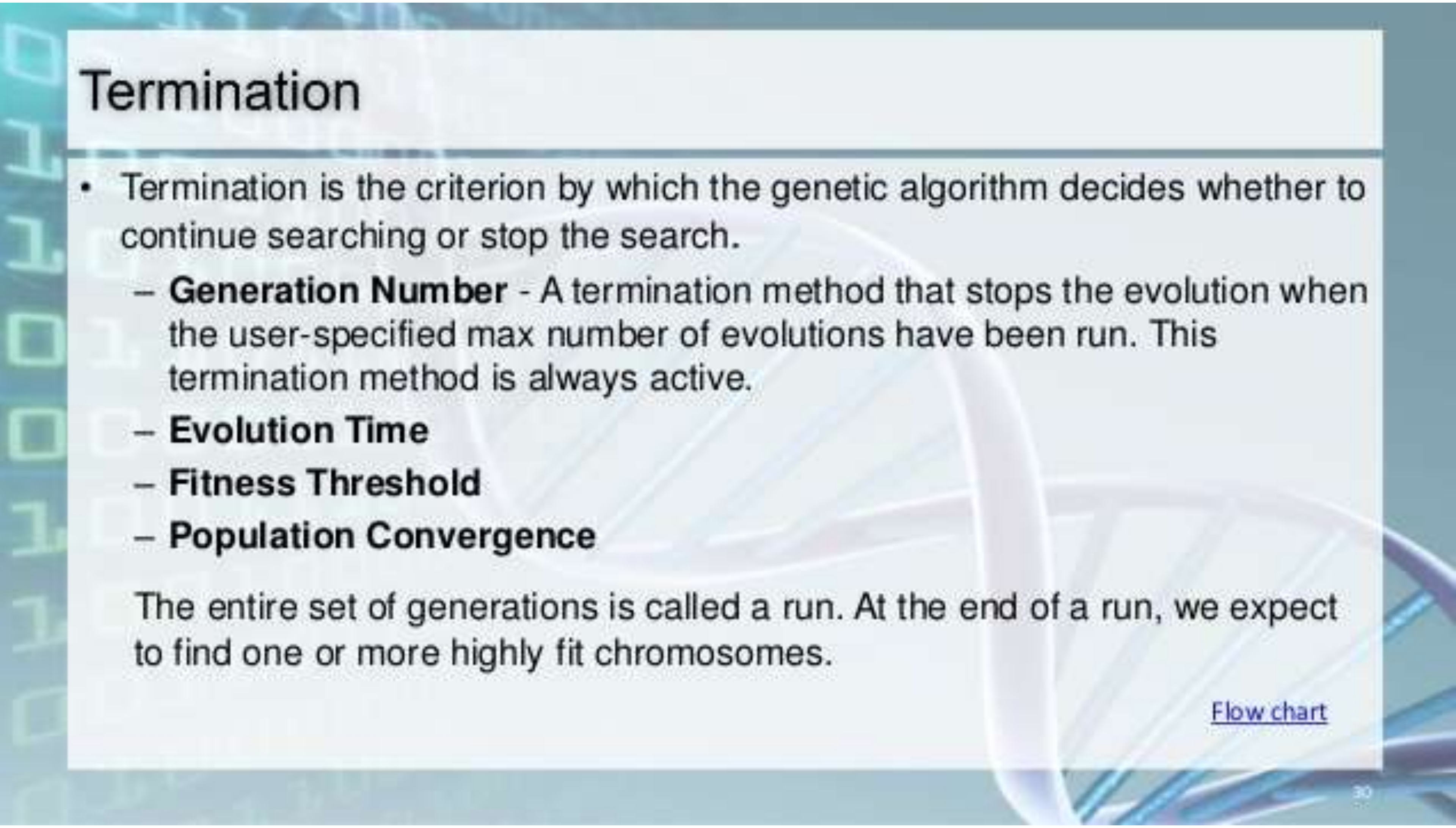
Binary Tournament

- Binary Tournament
  - Two individuals are randomly selected from the population and compared.
  - The one with the highest fitness is selected for reproduction.
  - Then another two individuals are randomly selected and the best fit is kept as the mate to the first parent.



Truncation

- Least useful selection strategy.
- Truncation selection simply retains the fittest x% of the population. These fittest individuals are duplicated so that the population size is maintained.
    - For example, we might select the fittest 25% from a population of 100 individuals. In this case we would create four copies of each of the 25 candidates in order to maintain a population of 100 individuals.
- This is an easy selection strategy to implement but it can result in premature convergence as less fit candidates are ruthlessly culled without being given the opportunity to evolve into something better. Nevertheless, truncation selection can be an effective strategy for certain problems.



Termination

- Termination is the criterion by which the genetic algorithm decides whether to continue searching or stop the search.
  - **Generation Number** - A termination method that stops the evolution when the user-specified max number of evolutions have been run. This termination method is always active.
  - **Evolution Time**
  - **Fitness Threshold**
  - **Population Convergence**

The entire set of generations is called a run. At the end of a run, we expect to find one or more highly fit chromosomes.

Flow chart

30

Source: https://www.slideshare.net/FatemehKarimi/genetic-algorithm-55733368

Traveling Salesman Problem

- Given a number of cities and the costs of traveling from one city to any other city, what is the cheapest round-trip route that visits each city exactly once and then returns to the starting city?

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30

Traveling Salesman Problem – Representation

- Problem:
  - Given n cities
  - Find a complete tour with minimal length
- Encoding:
  - Label the cities 1, 2, … , n
  - One complete tour is one permutation
  - For n =4 [1,2,3,4], [3,4,2,1] are possible tours

Traveling Salesman Problem – Representation

- Start with A
  - A – D – H – F – C – B – G – E
  - Cost = ?
- Start with E
  - E – H – F – C – A – D – B – G
  - Cost = ?
- Start with G
  - G – B – F – H – E – A – D – C
  - Cost = ?
- Search space is BIG
  - For 30 cities there are 30! ≈ 1032 possible tours

	A	B	C	D	E	F	G	H
A	0	8	3	1	4	9	3	6
B	8	0	5	10	11	4	3	6
C	3	5	0	8	7	1	5	12
D	1	10	8	0	9	11	6	4
E	4	11	7	9	0	5	17	3
F	9	4	1	11	5	0	4	1
G	3	3	5	6	17	4	0	7
H	6	6	12	4	3	1	7	0

Step 1 – Initialize The Population

- Suppose the population size is 6

Candidate Solutions
A C B F H D E G
H B G E A C D F
A H G C B D F E
E G B C D H F A
F H A D C B E G
C D B A H E G F

Step 2 – Compute Fitness

Candidate Solutions	Fitness
A C B F H D E G	43
H B G E A C D F	52
A H G C B D F E	49
E G B C D H F A	47
F H A D C B E G	49
C D B A H E G F	56

	A	B	C	D	E	F	G	H
A	0	8	3	1	4	9	3	6
B	8	0	5	10	11	4	3	6
C	3	5	0	8	7	1	5	12
D	1	10	8	0	9	11	6	4
E	4	11	7	9	0	5	17	3
F	9	4	1	11	5	0	4	1
G	3	3	5	6	17	4	0	7
H	6	6	12	4	3	1	7	0

Step 3 – Parent Selection (Binary Tournament)

- First Parent
  - 1 vs. 4 (1 is the winner as its fitness is better)

Candidate Solutions	Fitness
A C B F H D E G	43
H B G E A C D F	52
A H G C B D F E	49
E G B C D H F A	47
F H A D C B E G	49
C D B A H E G F	56

Parent 1	**A C B F H D E G**	**43**

Step 3 – Parent Selection (Binary Tournament)

- Second Parent
  - 2 vs. 5 (5 is the winner as its fitness is better)

Candidate Solutions	Fitness
A C B F H D E G	43
H B G E A C D F	52
A H G C B D F E	49
E G B C D H F A	47
F H A D C B E G	49
C D B A H E G F	56

Parent 1I	F H A D C B E G	49

Step 4a – Crossover

- Crossover to produce two offspring – assume crossover points are 3 and 5

Parent 1	**A C | B F H | D E G**	**43**
Parent 2	**F H | A D C | B E G**	**49**

Offspring 1	D C B F H E G A
Offspring 2	F H A D C E G B

Step 4b: Mutation

- Swap mutation on the two offspring produced

Offspring 1	D C B F H E G A
Offspring 2	F H A D C E G B

- For Offspring 1, swap 2nd and 4th gene.
- For Offspring 2, swap 2nd and 6th gene.

Offspring 1	D F B C H E G A
Offspring 2	F E A D C H G B

Step 5: Compute Fitness

	A	B	C	D	E	F	G	H
A	0	8	3	1	4	9	3	6
B	8	0	5	10	11	4	3	6
C	3	5	0	8	7	1	5	12
D	1	10	8	0	9	11	6	4
E	4	11	7	9	0	5	17	3
F	9	4	1	11	5	0	4	1
G	3	3	5	6	17	4	0	7
H	6	6	12	4	3	1	7	0

Offspring 1	D F B C H E G A	55
Offspring 2	F E A D C H G B	40

Step 6: Survivor Selection

Parents	Fitness
A C B F H D E G	
H B G E A C D F	
A H G C B D F E	
E G B C D H F A	
F H A D C B E G	
C D B A H E G F	

Children	Fitness
D F B C H E G A	
F E A D C H G B	

Survivor Selection

- Most EAs use fixed population size so need a way of going from (parents + offspring) to next generation
- Suppose the process shown in the previous slides is repeated twice to generate 4 more offspring.
- We have now a pool of these 10 solutions (6 parents + 4 offspring) and can run binary tournament to select 6 candidates for the next generation.
- These 6 selected candidates will form the parents pool in the 2nd generation.

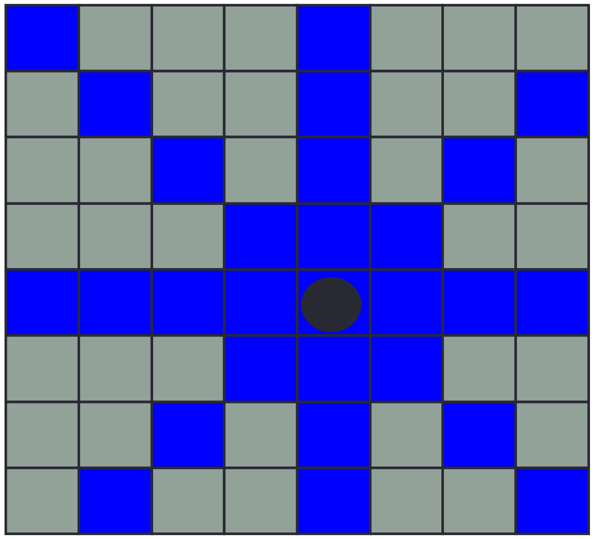
Evolutionary Algorithm Cycle

- Step 1: Initialize the population randomly or with potentially good ***solutions***
- Step 2: Compute the ***fitness*** of each individual in the population
- Step 3: Select parents using a ***selection procedure***
- Step 4: Create offspring by ***crossover*** and ***mutation*** operators
- Step 5: Compute the ***fitness*** of the new offspring
- Step 6: Select members of population to die using a ***selection procedure***
- Step 7: Go to Step 2 until termination criteria are met

Algorithm Design

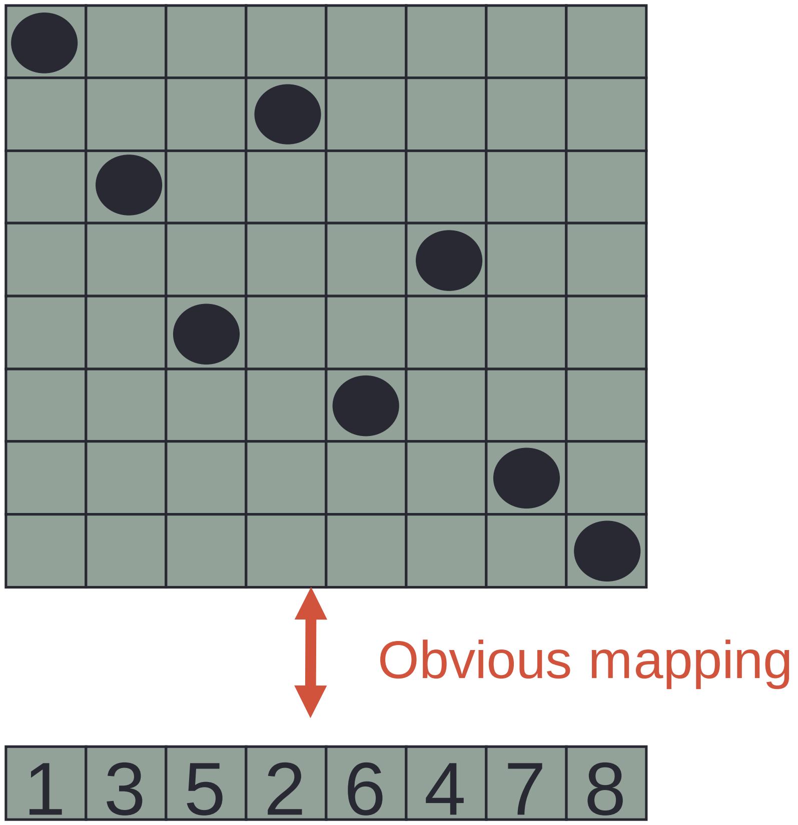
- Design a representation
- Design a way of evaluating an individual
- Design suitable recombination operator(s)
- Design suitable mutation operator(s)
- Decide how to select individuals to be parents
- Decide how to select individuals for the next generation (how to manage the population)
- Decide how to stop: termination criterion

The 8-Queen Problem



Place 8 queens on an 8x8 chessboard in such a way that they cannot check each other

The 8-Queen Problem – Representation

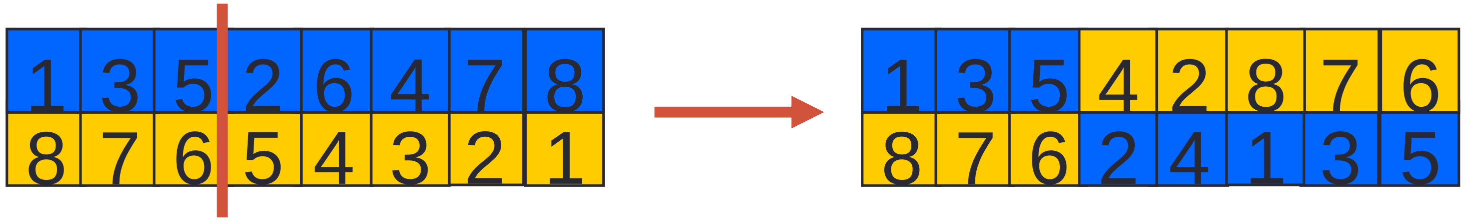


The 8-Queen Problem – Fitness Evaluation

- Penalty of one queen
  - Number of queens she can check
- Penalty of a configuration
  - Sum of the penalties of all queens
- Penalty is to be minimized
- Fitness of a configuration
  - Inverse penalty to be maximized

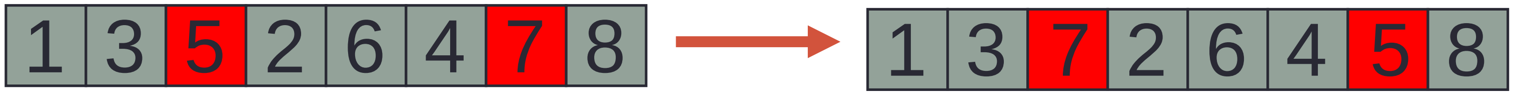
The 8-Queenss Problem – Crossover

- Combining two permutations into two new permutations:
  - choose random crossover point
  - copy first parts into children
  - create second part by inserting values from other parent:
    - in the order they appear there
    - beginning after crossover point
    - skipping values already in child



The 8-Queens Problem – Mutation

- Small variation in one permutation
  - swapping values of two randomly chosen positions


1 3 5 2 6 4 7 8
1 3 7 2 6 4 5 8