Artificial Intelligence (CS-401)

Chapter 4: Local Search Algorithm and Optimization

Objectives of the Chapter

- Local Search: Hill Climbing
- Escaping Local Maxima: Local Beam Search
- Simulated Annealing
- Genetic Algorithm

Local Search

- Use single current state and move to the neighboring state.
- **Idea:** Start with an initial guess at a solution and incrementally improve it until it is one.

Hill Climbing Algorithm

(Local Search, Greedy Approach, No backtrack)

- In this algorithm, we don't need to maintain and handle the search tree or graph as it only keeps a single current state
- Hill climbing algorithm continuously moves in the direction of increasing elevation/value to find the peak or best solution to the problem.
- It terminates when it reaches a peak value where no neighbor has a higher value.
- It is also called **greedy local search** as it only looks to its good immediate neighbor state and not beyond that.
- .

Algorithm

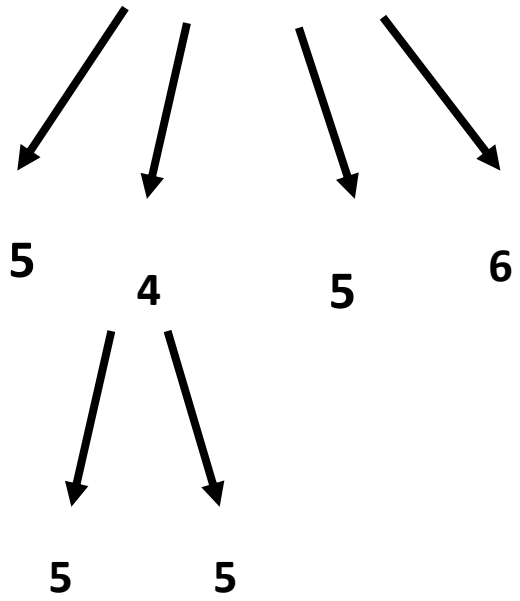
- **Step 1:** Evaluate the initial state, if it is goal state then return success and Stop.
- **Step 2:** Loop Until a solution is found or there is no new operator left to apply.
- **Step 3:** Select and apply an operator to the current state.
- **Step 4:** Check new state:
  - If it is goal state, then return success and quit.
  - Else if it is better than the current state then assign new state as a current state.
  - Else if not better than the current state, then return to step2.
- **Step 5:** Exit.

Example

5

1	2	4
5		7
3	6	8

State

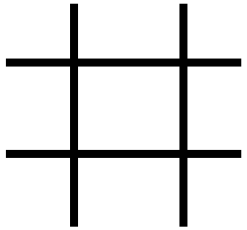


1	4	7
2	5	8
3	6	

goal

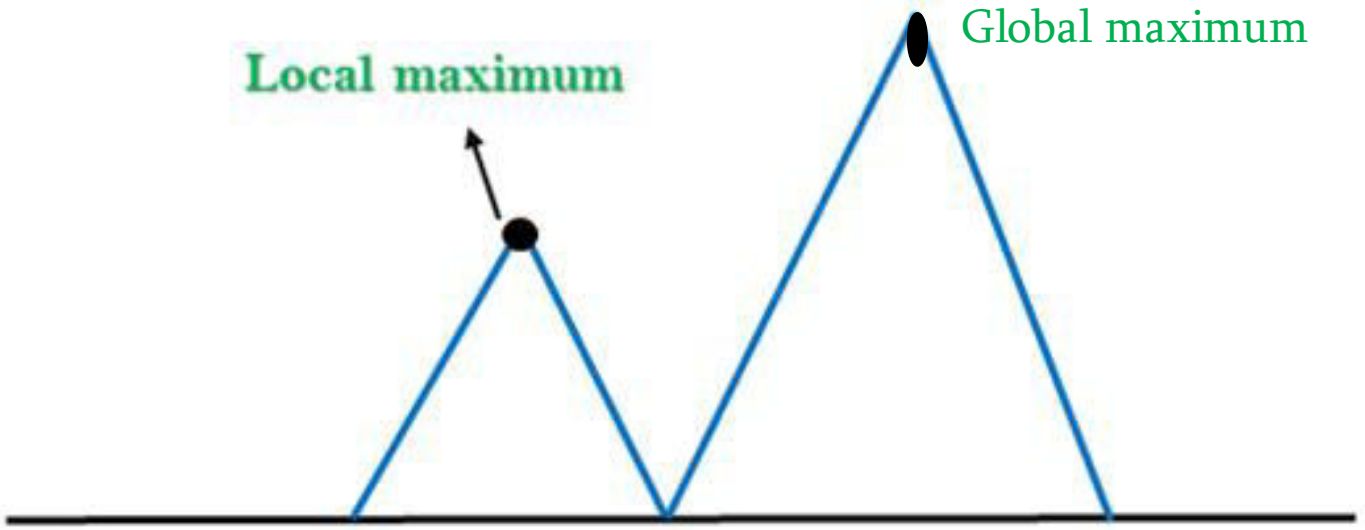
Example

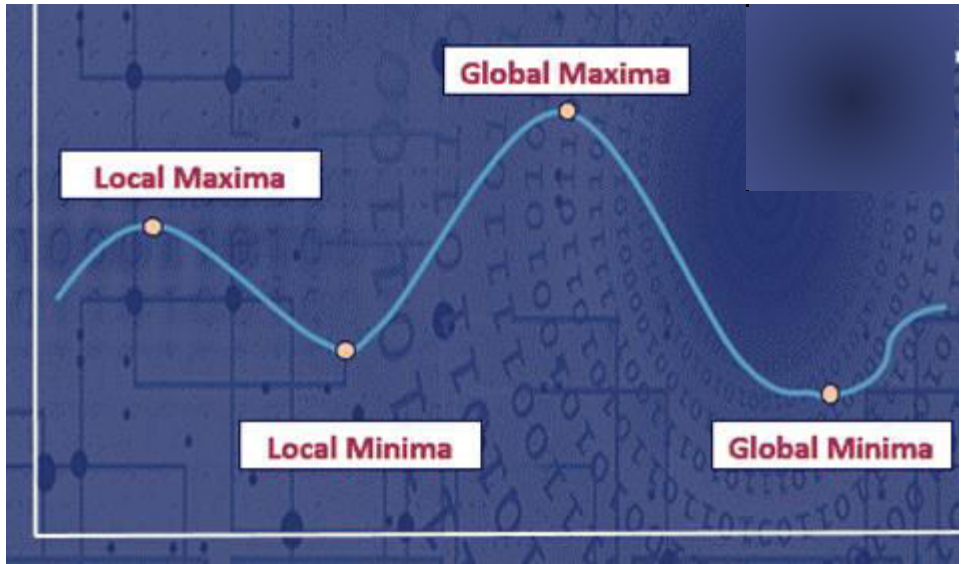
- Tic-Tac-Toe



Maximum number of winning lines

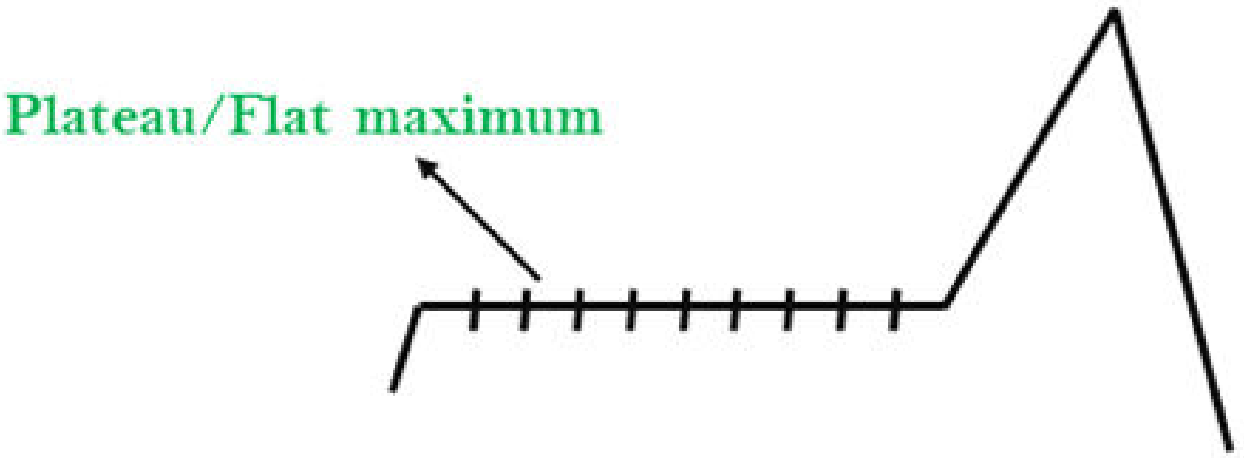
Problems in Hill Climbing





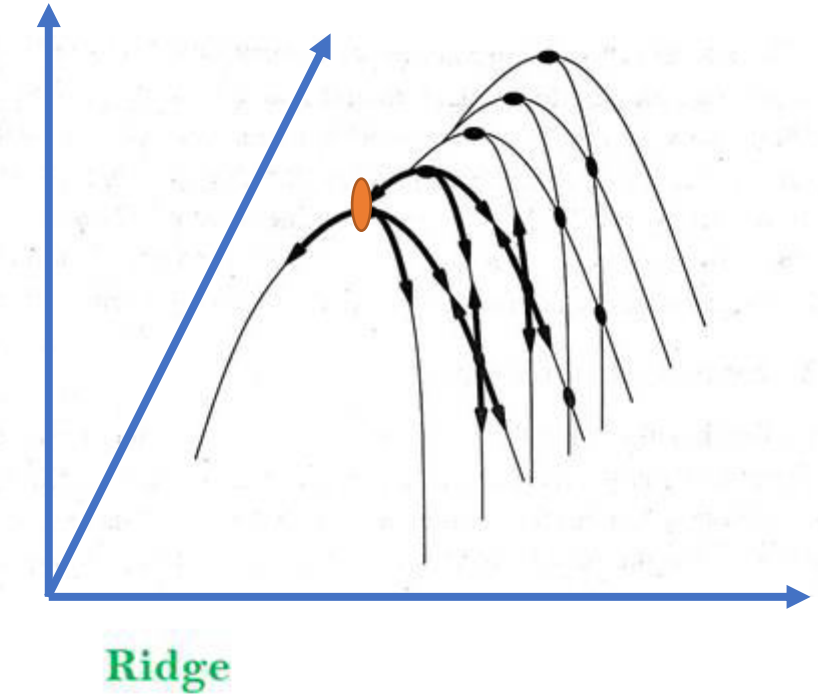
- **1. Local Maximum:** A local maximum is a peak state in the landscape which is better than each of its neighboring states, but there is another state also present which is higher than the local maximum.

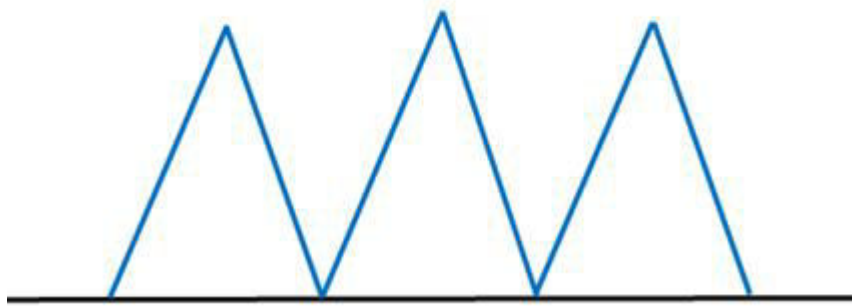
Problems in Hill Climbing....



2 Plateau: A plateau is the flat area of the search space in which all the neighbor states of the current state contains the same value, because of this algorithm does not find any best direction to move. A hill-climbing search might be lost in the plateau area.

Problems in Hill Climbing…

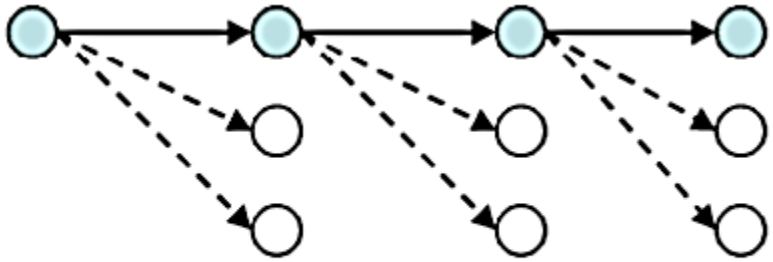




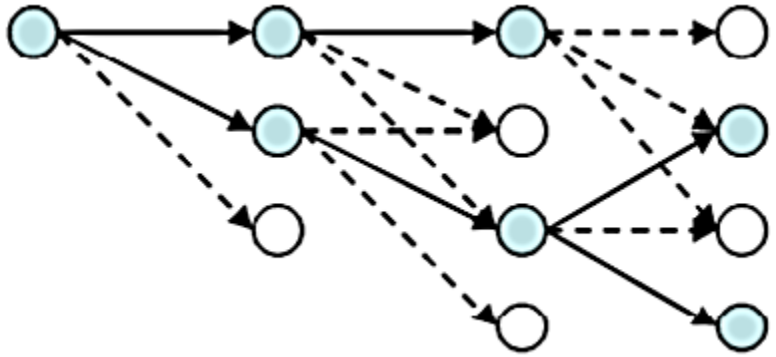
3. Ridges: A ridge is a special form of the local maximum. It has an area which is higher than its surrounding areas, but itself has a slope, and cannot be reached in a single move.

Local Beam Search

- Start with k randomly generated states (ß= 2 or 3)
- At each iteration, all the successors of all k states are generated
- If any one is a goal state, stop; else select the k best successors from the complete list and successors from the complete list and repeat



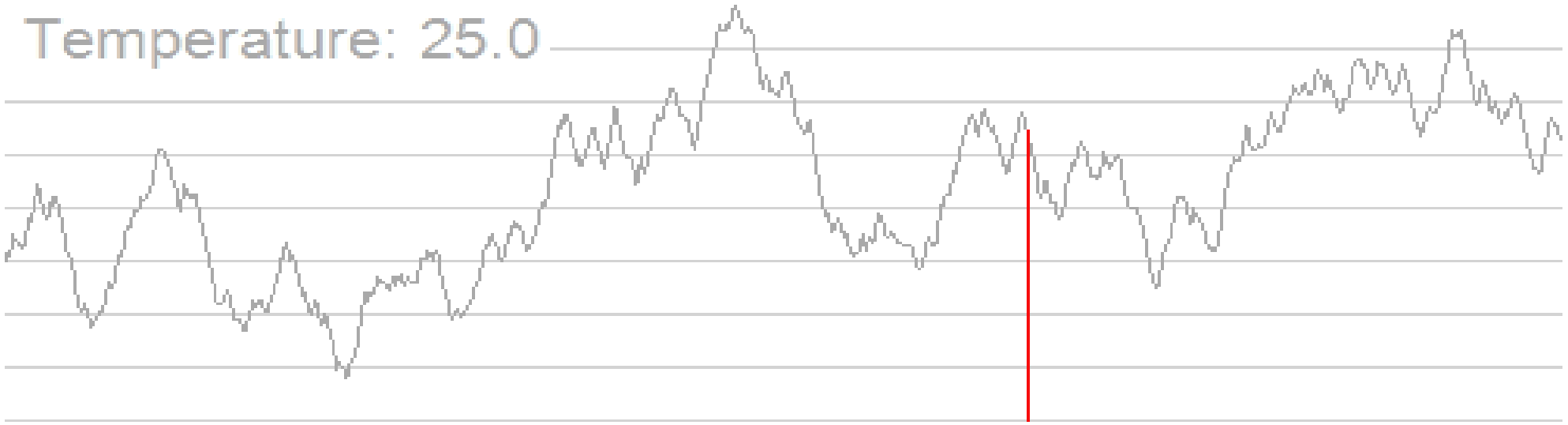
Greedy Local



Local Beam

Simulated Annealing

- Conceptually SA exploits an Analogy between Annealing and the Search for a minimum in a more general system
- Explore Successors wildly randomly: High temp
- As time goes by , explore less wildly: Cool Down
- Until there's a time when things settle: Cold



Genetic Algorithm

Introduced in the 1970s by John Holland at University of Michigan

- begin with *k* randomly generated states (population)
- each state (individual) is a string over some alphabet (chromosome)
- fitness function (bigger number is better)
- crossover
- mutate (evolve?)

Nature of Computer Mapping

Nature	Computer
Population	Set of solutions.
Individual	Solution to a problem.
Fitness	Quality of a solution.
Chromosome	Encoding for a Solution.
Gene	Part of the encoding of a solution.
Reproduction	Crossover

Computational Model

Step 1: Representing individuals.

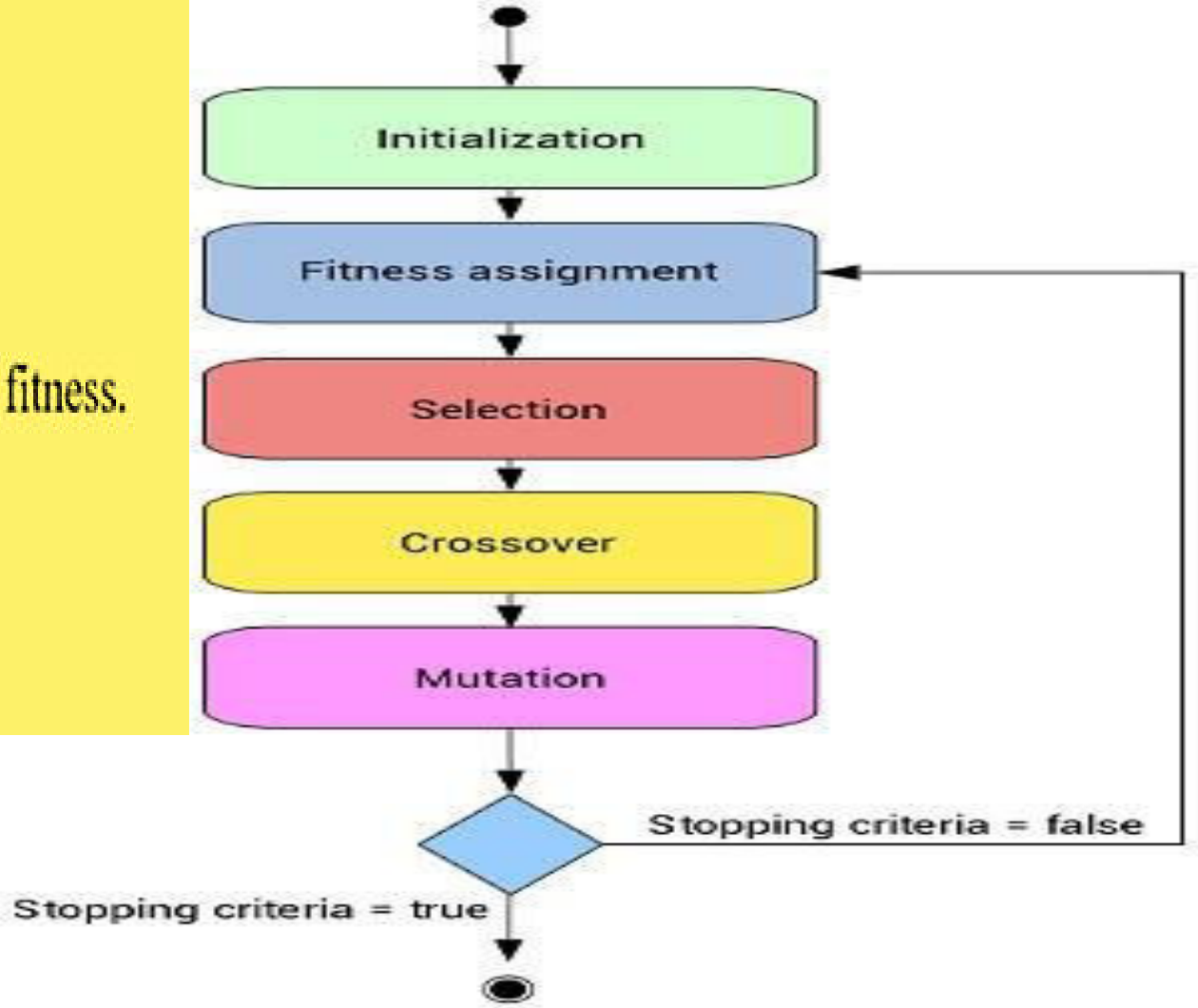
Step 2: Generating an initial Population.

Step 3: Applying a Fitness Function.

Step 4: Selecting parents for mating in accordance to their fitness.

Step 5: Crossover of parents to produce new generation.

Step 6: Mutation of new generation to bring diversity.



Encoding

*The process of representing the solution in the form of a **string** that conveys the necessary information.*

- **Binary Encoding** – Most common method of encoding. Chromosomes are strings of 1s and 0s and each position in the chromosome represents a particular characteristic of the problem.

  Permutation Encoding – Useful in ordering problems such as the Traveling Salesman Problem (TSP). Example. In TSP, every chromosome is a string of numbers, each of which represents a city to be visited.

- **Value Encoding** – Used in problems where complicated values, such as real numbers, are used and where binary encoding would not suffice.

Crossover

It is the process in which two chromosomes (strings) combine their genetic material (bits) to produce a new offspring which possesses both their characteristics.

- *Two strings are picked from the mating pool at random to cross over.*
- *The method chosen depends on the Encoding Method.*

Crossover

- **Single Point Crossover-** A random point is chosen on the individual chromosomes (strings) and the genetic material is exchanged at this point.

Chromosome1	11011 | 00100110110
Chromosome 2	11011 | 11000011110
Offspring 1	11011 | 11000011110
Offspring 2	11011 | 00100110110

Crossover

- **Two-Point Crossover-** Two random points are chosen on the individual chromosomes (strings) and the genetic material is exchanged at these points.

Chromosome1	11011 | 00100 | 110110
Chromosome 2	10101 | 11000 | 011110
Offspring 1	10101 | 00100 | 011110
Offspring 2	11011 | 11000 | 110110

NOTE: These chromosomes are different from the last example.

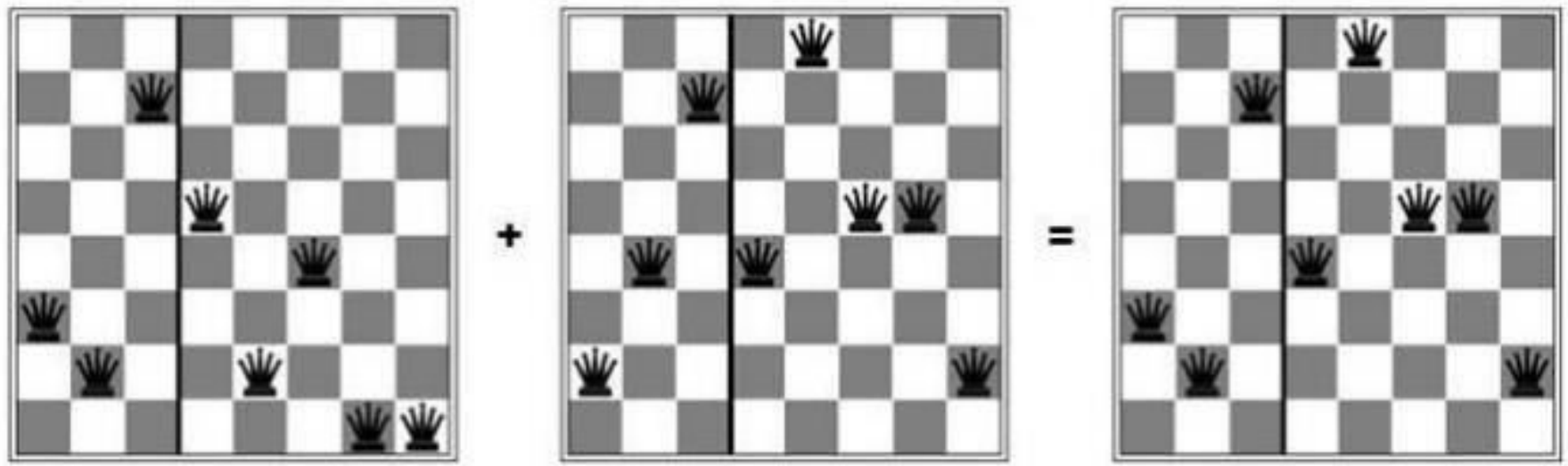
Crossover

- **Uniform Crossover-** Each gene (bit) is selected randomly from one of the corresponding genes of the parent chromosomes.

Chromosome1	11011 | 00100 | 110110
Chromosome 2	10101 | 11000 | 011110
Offspring	10111 | 00000 | 110110

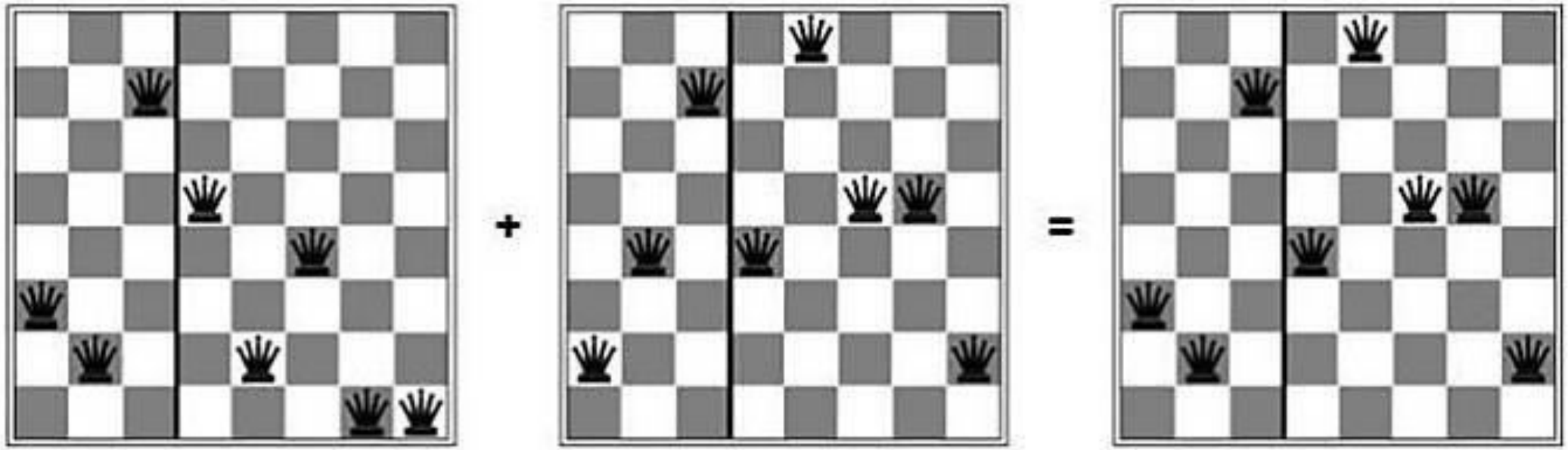
NOTE: Uniform Crossover yields ONLY 1 offspring.

Genetic algorithms:8-queens



Genetic Algorithm

- Fitness function= Pair of non-attacking queens
- That way higher scores are better

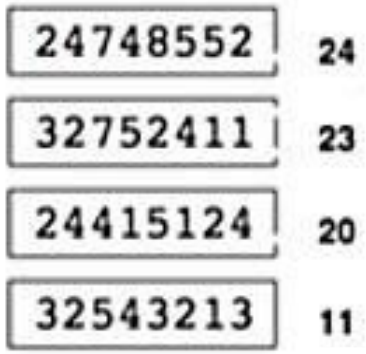


23 fitness

24748552 string

Fitness function

Represent states and compute fitness function.



(a)
Initial Population

Probability= 24+23+20+11= 78

Probability

Compute probability of being chosen (from fitness function).

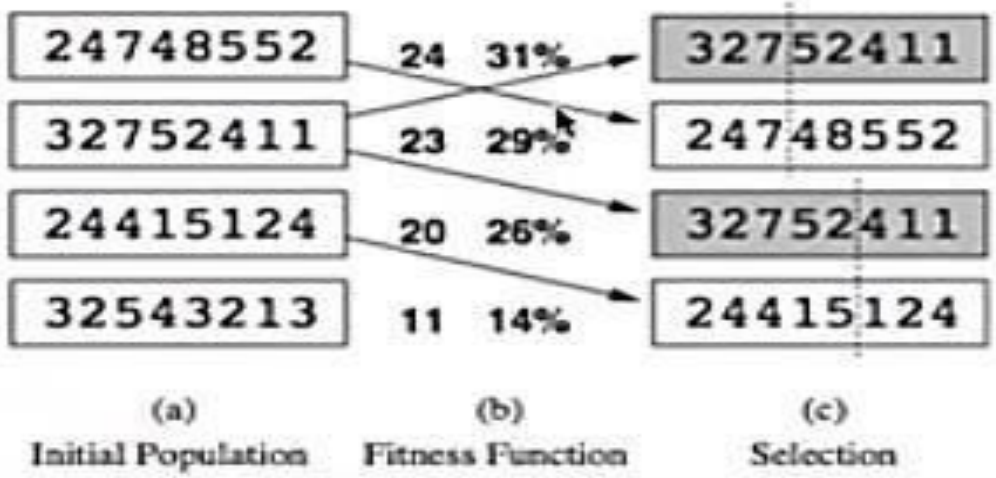
24748552	24	31%
32752411	23	29%
24415124	20	26%
32543213	11	14%

(a)
Initial Population

24/78= 0.307 Normalize 0.307× **100 = 30.7%**
chance of being chosen probably

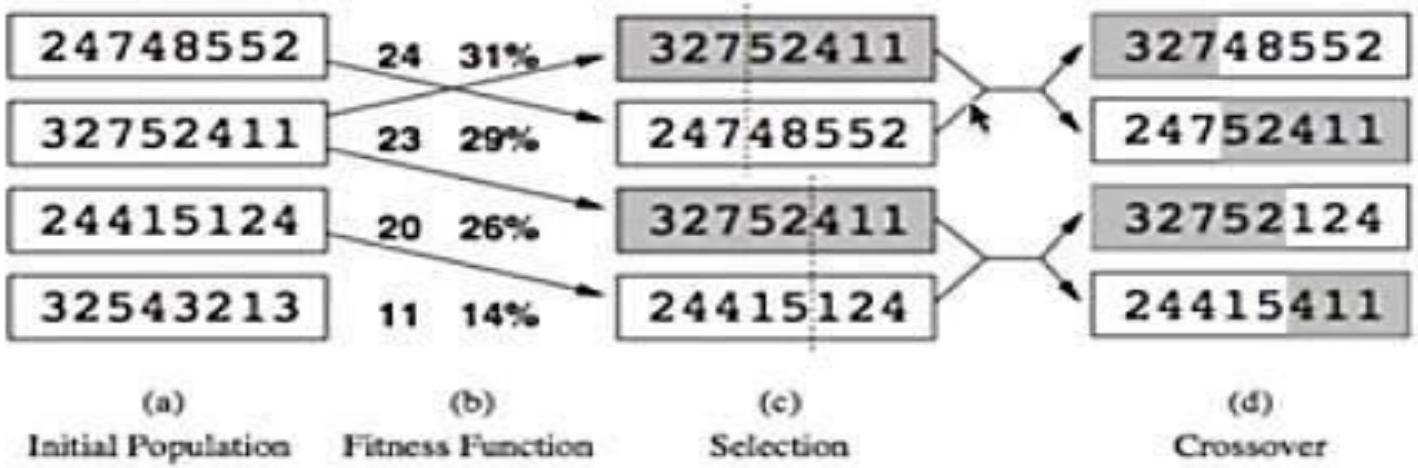
Reproduction

Randomly choose two pairs to reproduce based on probabilities. Pick a crossover point per pair.



Crossover

Crossover, produce offspring.



Mutation

May mutate.

Knapsack Problem

- Let's say, you are going to spend a month in the wilderness. Only thing you are carrying is the backpack which can hold a maximum weight of **30 kg**. Now you have different survival items, each having its own "Survival Points" (which are given for each item in the table). So, your objective is maximize the survival points.
- Here is the table giving details about each item.

ITEM	WEIGHT	SURVIVAL POINTS
SLEEPING BAG	15	15
ROPE	3	7
POCKET KNIFE	2	10
TORCH	5	5
BOTTLE	9	8
GLUCOSE	20	17

- Choose an initial population of four. Then choose best three subsequently (Encoding hint: 1 if item is chosen 0 otherwise)
- Decide crossover point.
- Chance of mutation = 50 items in 100 generations.
- Step by step solution ahead

① Initial population (Initially I am taking 4 states chosen randomly which have weight ≤ 30)

100110
000011
011101
111100

② Second step : Selection / Fitness function : The one with highest survival point is to be selected (i.e best 3 are the ones which have highest survival point).

100110	→	15+0+0+5+8+0 = 28 ✓
000011	fitness →	0+0+0+0+8+17 = 25
011101	function →	0+7+10+5+0+17 = 39 ✓
111100	→	15+7+10+5+0+0 = 37 ✓

Best three are states 1, 3 & 4

③ Selection : 1, 3 & 4 states

Combination : [1 & 3] & [3 & 4]

1 & 3

1 0 0 1 1 0

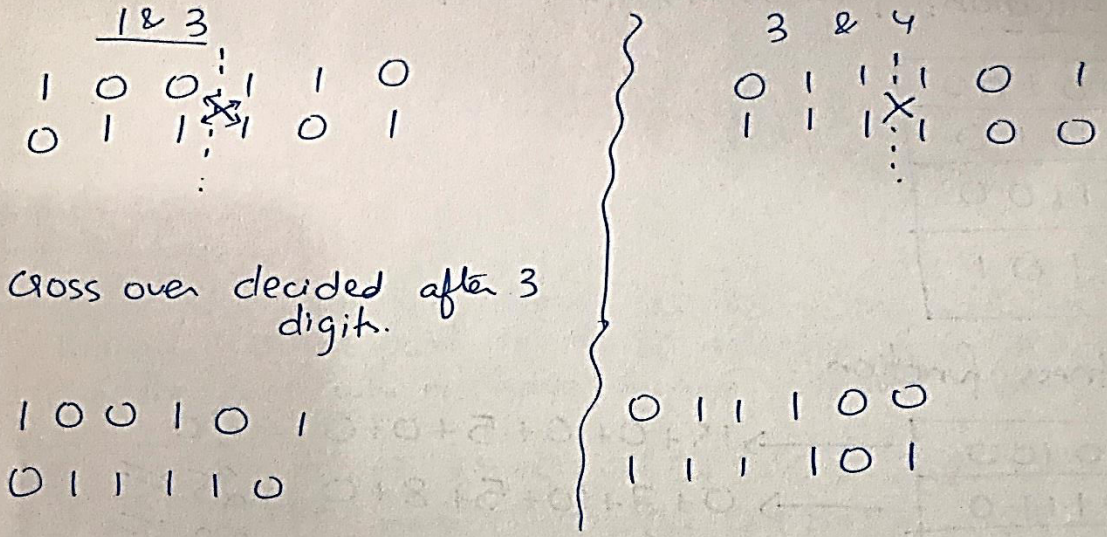
0 1 1 1 0 1

0 1 1 1 0 1

1 1 1 1 0 0

④ ~~Done~~

④ Now cross over will be applied.



Cross over decided after 3 digit.

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

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

⑤ Now mutation (we had 50 items in 100 generations).

Because in question we are told to perform 3 iterations; one of them is initialization iteration & 2 are generation thereby.

50 items —— 100 generation

n —— 2 "

$n = 1$

Thereby one bit mutation will only take place in this entire 3 iteration phase.

Our cross over brought 4 new states.

mutate to

1 0 0 1 0 (1) 0.

0 1 1 1 1 0

0 1 1 1 0 0

1 1 1 1 0 1

Now if you want you can mutate one bit in this iteration, else do it in any ~~two~~ other iteration. I have mutated in this iteration thereby output of first iteration is

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

Now this output of first iteration will be the input for second iteration.

Second iteration

① Initialization: Previous iteration output.

100100
011110
011100
111101

② Fitness function:

100100	→ 15+0+0+5+0+0 = 20
011110	→ 0+7+10+5+8+0 = 25 =
011100	→ 0+7+10+5+0+0 = 22 =
111101	→ 15+7+10+5+0+17 = 54 =

③ Selection 2 & 3 & 4

④ Cross over

2 & 3

011 | 110
011 | 100

011100
011110

3 & 4

011 | 100
111 | 101

011101
111100

Crossover results in

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

No mutation now because I have already done in previous iteration & only one bit mutation was allowed in the question

* Point to notice: See state 3 in this crossover that's exactly holding weight 30 & having survival 39 which is till now the best state as we can also hold more items & survival is also more.

Now as no mutation is done we will input the crossover output of this iteration to third iteration.

That means.

(3) Third iteration

Initialization

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

→ Now rest of the steps will be done & the end.

You can do the third iteration on your own now. First two iterations have been done by me.

Properties

- Work well for mixed (continuous and discrete) problems
- They are less suceptible to get stuck at local optima
- Computationally expensive
- However, easy to perform in parallel
- No math in the process. The objective (fitness) function may be hard