BITS Pilani
Pilani Campus

Artificial & Computational Intelligence

DSE CLZG557

M2 : Problem Solving Agent using Search

Raja vadhana P
Assistant Professor,
BITS - CSIS

Course Plan

M1 Introduction to AI

M2 Problem Solving Agent using Search

M3 Game Playing, Constraint Satisfaction Problem

M4 Knowledge Representation using Logics

M5 Probabilistic Representation and Reasoning

M6 Reasoning over time, Reinforcement Learning

M7 AI Trends and Applications, Philosophical foundations

Module 2 : Problem Solving Agent using Search

A. Uninformed Search

B. Informed Search

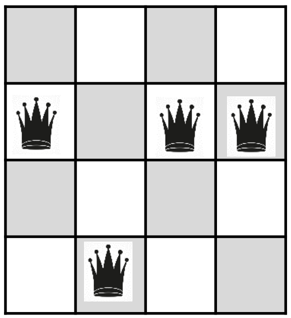
C. Heuristic Functions

D. Local Search Algorithms & Optimization Problems

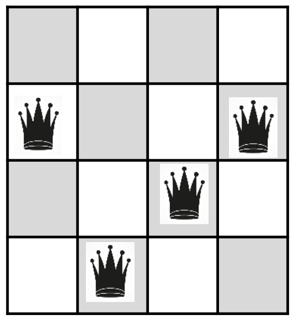
Local Search & Optimization

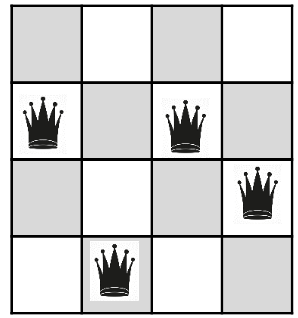
Terminology

Local Search : Search in the state-space in the neighbourhood of current position until an optimal solution is found

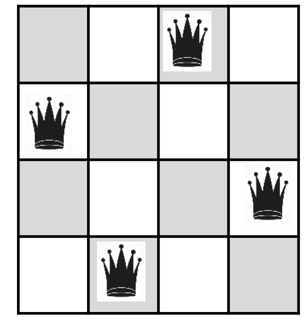


Feasible State/Solution





Neighboring States



Optimal Solution

Fitness Value:

h(n) = 4 h(n) = 4 h(n) = 2 h(n) = 0

Above is an example of h(n) = No.of.Conflicting **pairs** of queens

h(n) = 0 h(n) = 0 h(n) = 1 h(n) = 0

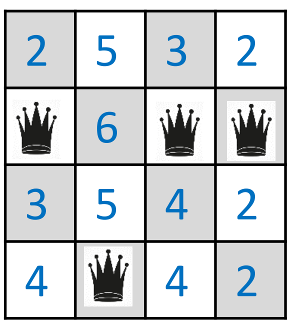
Above is an example of h(n) = No.of.Non-Conflicting Single queens with other queens in the board.

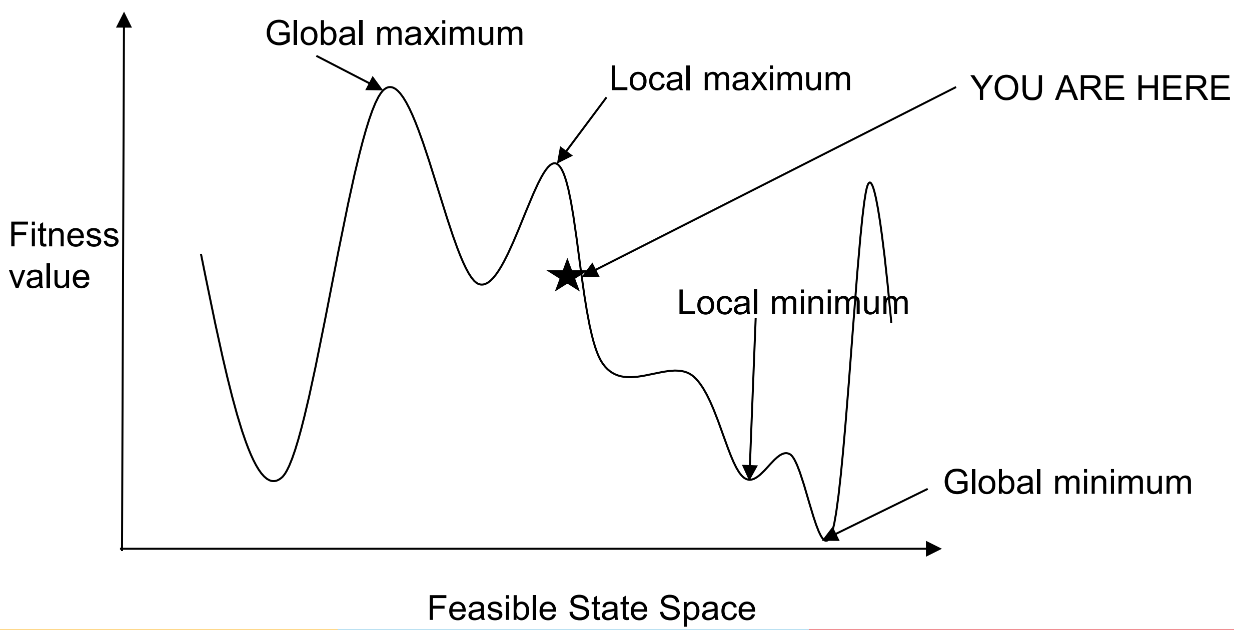
Terminology

Local Search : Search in the state-space in the neighbourhood of current position until an optimal solution is found

Algorithms:

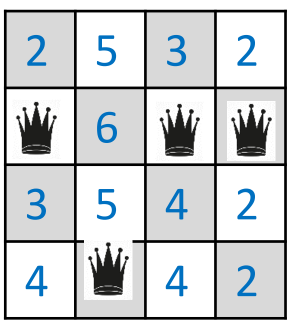
- Choice of Neighbor
- Looping Condition
- Termination Condition

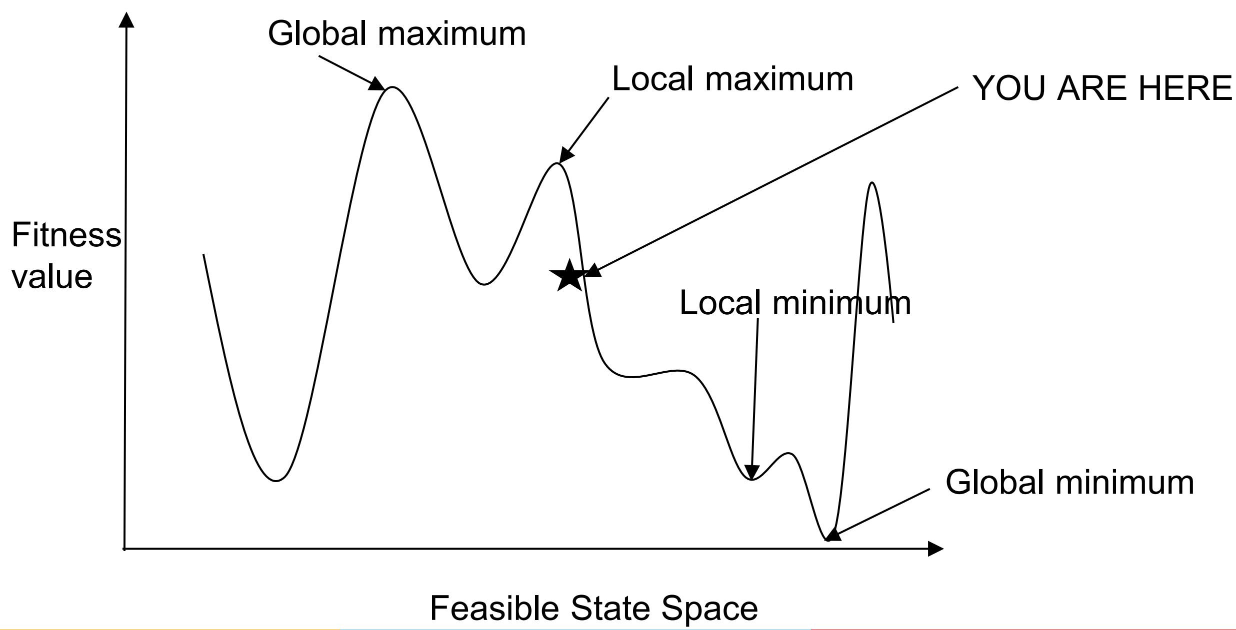




Hill Climbing

Hill Climbing



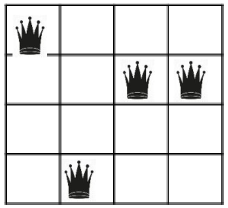


Hill Climbing

1. Select a random state
2. Evaluate the fitness scores for all the successors of the state
3. Calculate the probability of selecting a successor based on fitness score
4. Select the next state based on the highest probability
5. Repeat from Step 2

h(n) = No.of non-conflicting pairs of queens in the board.

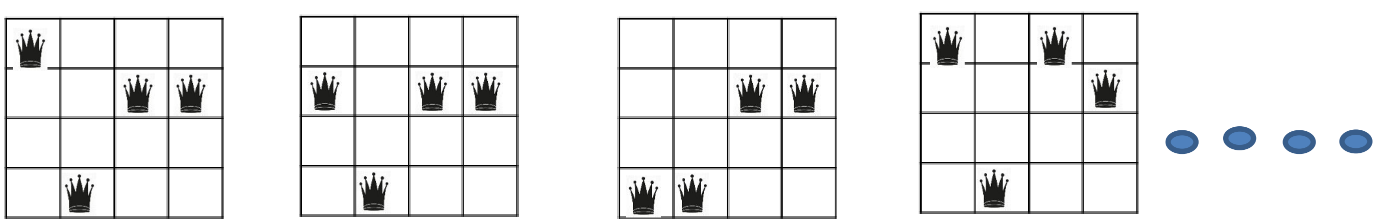
Q1-Q2
Q1-Q3
Q1-Q4
Q2-Q3
Q2-Q4
Q3-Q4



1	4	2	2	4

Stochastic Hill Climbing

1. Select a random state
2. Evaluate the fitness scores for all the successors of the state
3. Calculate the probability of selecting a successor based on fitness score
4. Select the next state based on the highest probability
5. Repeat from Step 2



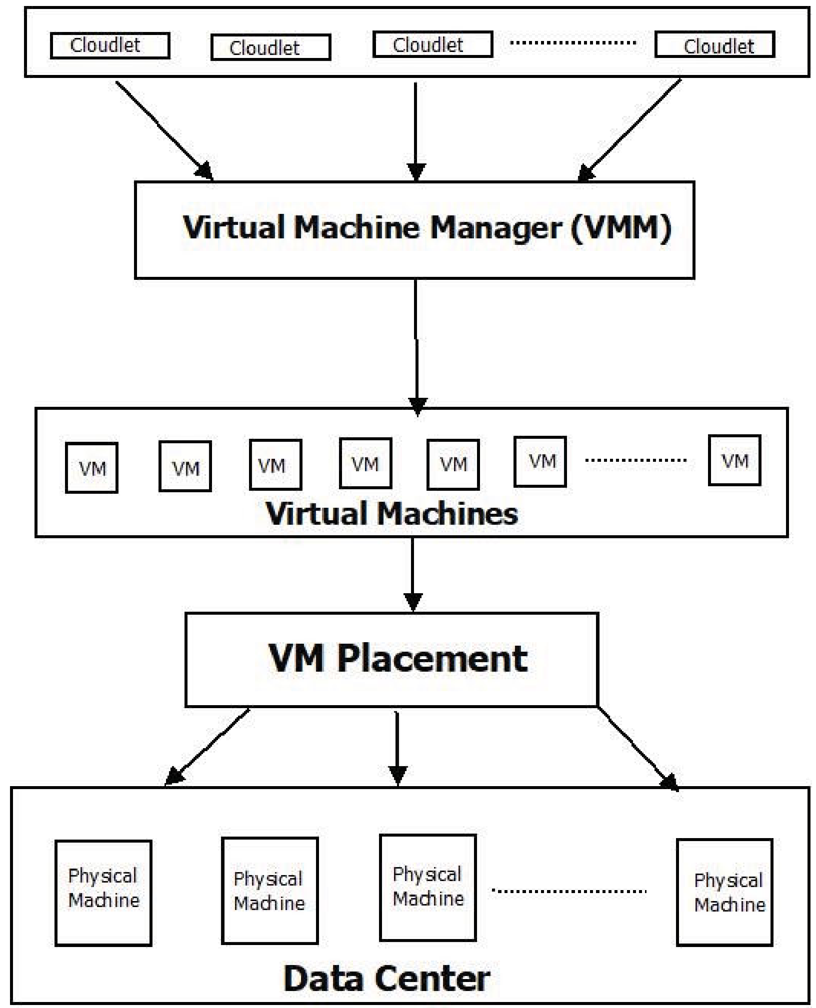
1	4	2	2	4

2	4	2	2	2

4	4	2	2	2

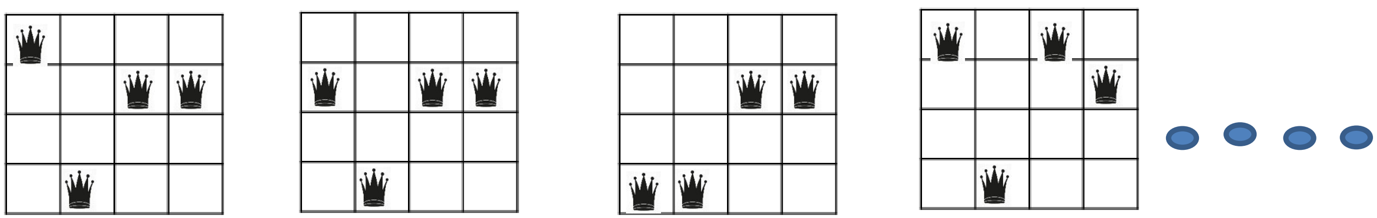
1	4	1	2	3

Stochastic Hill Climbing



Stochastic Hill Climbing

1. Select a random state
2. Evaluate the fitness scores for all the successors of the state
3. Calculate the probability of selecting a successor based on fitness score
4. Select the next state based on the highest probability
5. Repeat from Step 2



1	4	2	2	4	0.08
2	4	2	2	2	0.42
4	4	2	2	2	0.42
1	4	1	2	3	0.25

12 N = {4,2,2,3,3,2,2,0,2,1,3,0}

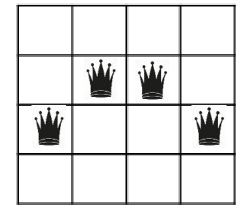
$next \leftarrow$ a randomly selected successor of $current$
$\Delta E \leftarrow next.\text{VALUE} - current.\text{VALUE}$
if $\Delta E > 0$ **then** $current \leftarrow next$
else $current \leftarrow next$ only with probability $e^{\Delta E/T}$

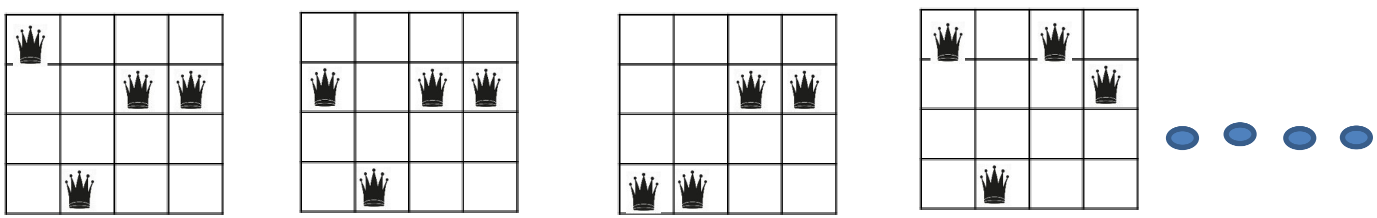
S

Simulated Annealing

Simulated Annealing

1. Select a random state
2. Evaluate the fitness scores for all the successors of the state
3. Calculate the probability of selecting a successor based on fitness score
4. Select the next state based on the highest probability
5. Repeat from Step 2





1	4	2	2	4	0.55
2	4	2	2	2	0.5
4	4	3	3	2	0.5
1	4	1	2	3	0.525

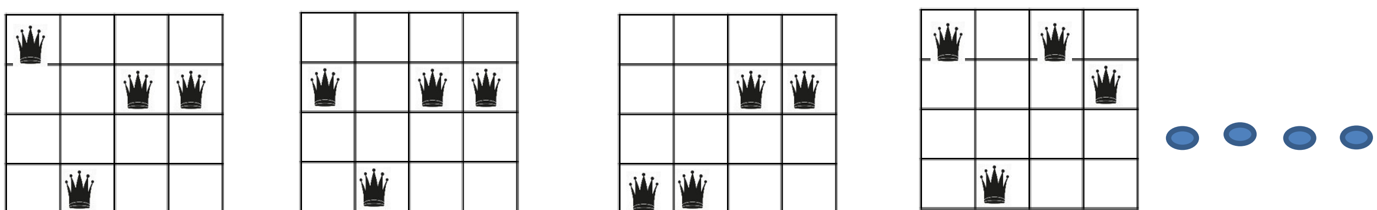
12 N = {4,2,2,3,3,2,1,3,2,1,3,2}

Init = 2

Simulated Annealing

function SIMULATED-ANNEALING(*problem*, *schedule*) **returns** a solution state
  inputs: *problem*, a problem
    schedule, a mapping from time to "temperature"

  current ← MAKE-NODE(*problem*.INITIAL-STATE)
  for $t = 1$ **to** ∞ **do**
    $T \leftarrow schedule(t)$
    if $T = 0$ **then return** *current*
    next ← a randomly selected successor of *current*
    $\Delta E \leftarrow next.\text{VALUE} - current.\text{VALUE}$
    if $\Delta E > 0$ **then** *current* ← *next*
    else *current* ← *next* only with probability $e^{\Delta E/T}$



1 4 2 2 4

2 4 2 2 2

4 4 2 2 2

1 4 1 2

Next Value	ΔE	ΔE/t	$e^{\Delta E/t}$	$\frac{1}{1+e^{\Delta E/t}}$
1	-1	-0.1	0.904	0.525
2	0	0	1	0.5
3	1	0.1	1.105	0.47
4	2	0.2	1.221	0.45

Simulated Annealing

Current Value = 4 (Local Maxima)

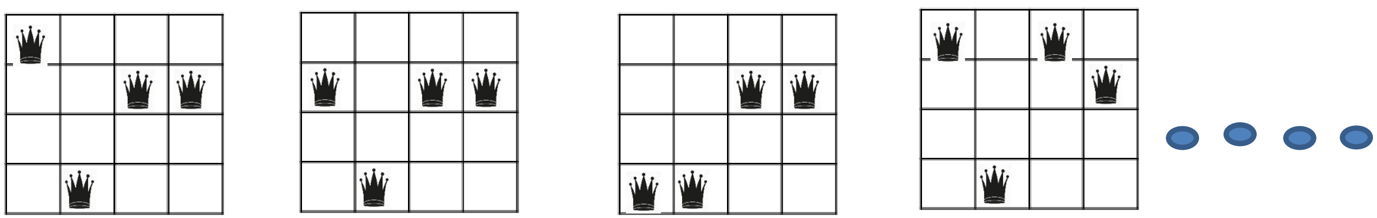
Global Maxima = 6

Next Value	ΔE	ΔE/t	$e^{\Delta E/t}$	$\frac{1}{1+e^{\Delta E/t}}$	ΔE/t	$e^{\Delta E/t}$	$\frac{1}{1+e^{\Delta E/t}}$
2	2	0.1	1.12	0.47	0.4	1.49	0.40
3	1	0.05	1.05	0.49	0.2	1.22	0.45
5	-1	-0.05	0.95	0.51	-0.2	0.82	0.55

Simulated Annealing

function SIMULATED-ANNEALING(*problem*, *schedule*) **returns** a solution state
  inputs: *problem*, a problem
    schedule, a mapping from time to "temperature"

  current ← MAKE-NODE(*problem*.INITIAL-STATE)
  for t = 1 **to** ∞ **do**
    T ← *schedule*(t)
    if T = 0 **then return** *current*
    next ← a randomly selected successor of *current*
    ΔE ← *next*.VALUE – *current*.VALUE
    if $\Delta E > 0$ **then** *current* ← *next*
    else *current* ← *next* only with probability $e^{\Delta E/T}$



1	4	2	2	4

2	4	2	2	2

4	4	2	2

Next Value	ΔE	ΔE/t	$e^{\Delta E/t}$	$\frac{1}{1+e^{\Delta E/t}}$	$e^{-\Delta E/t}$	$\frac{1}{1+e^{-\Delta E/t}}$
1	-1	-0.1	0.904	0.525	1.105	0.47
2	0	0	1	0.5	0	0.5
3	1	0.1	1.105	0.47	0.904	0.525
4	2	0.2	1.221	0.45	0.819	0.55

Simulated Annealing

Maximization problem design to achieve global minima

Set Temp to very high temp t

Set n as number of iteration to be performed at a particular t

L1: Randomly select a random neighbour

Calculate Energy barrier E = f(N)-f(C)

If **E > 0** then its a good move

    Move ahead for next tree search level

Else

    Create a random number r:[0-1]

    If **r< $e^{-E/t}$**

        Choose this bad state & move downhill

    Else

        Go to L1.

If Goal is reached or {acceptable goal(set criteria to check)node is reached & t is small END}

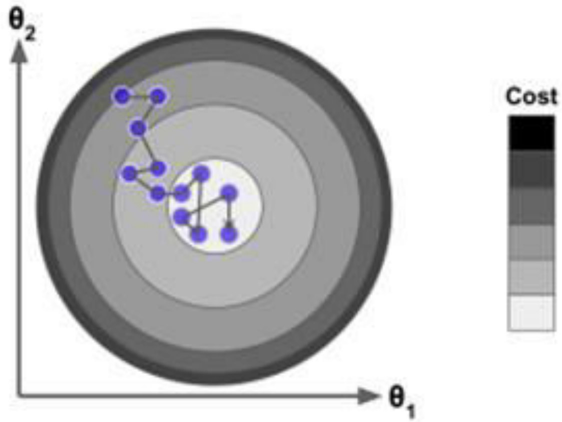
Else

    If no.of.neighbors explored has reached a threshold >=n

        then Lower t and go to L1.

Machine Learning

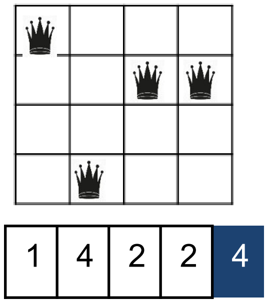
Examples

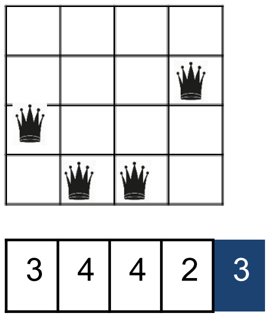


Local Beam Search

Beam Search

1. Initialize k random state
2. Evaluate the fitness scores for all the successors of the k states
3. Calculate the probability of selecting a successor based on fitness score
4. Select the next state based on the highest probability
5. If the goal is not found, Select the next ‘k’ states randomly based on the probability
6. Repeat from Step 2

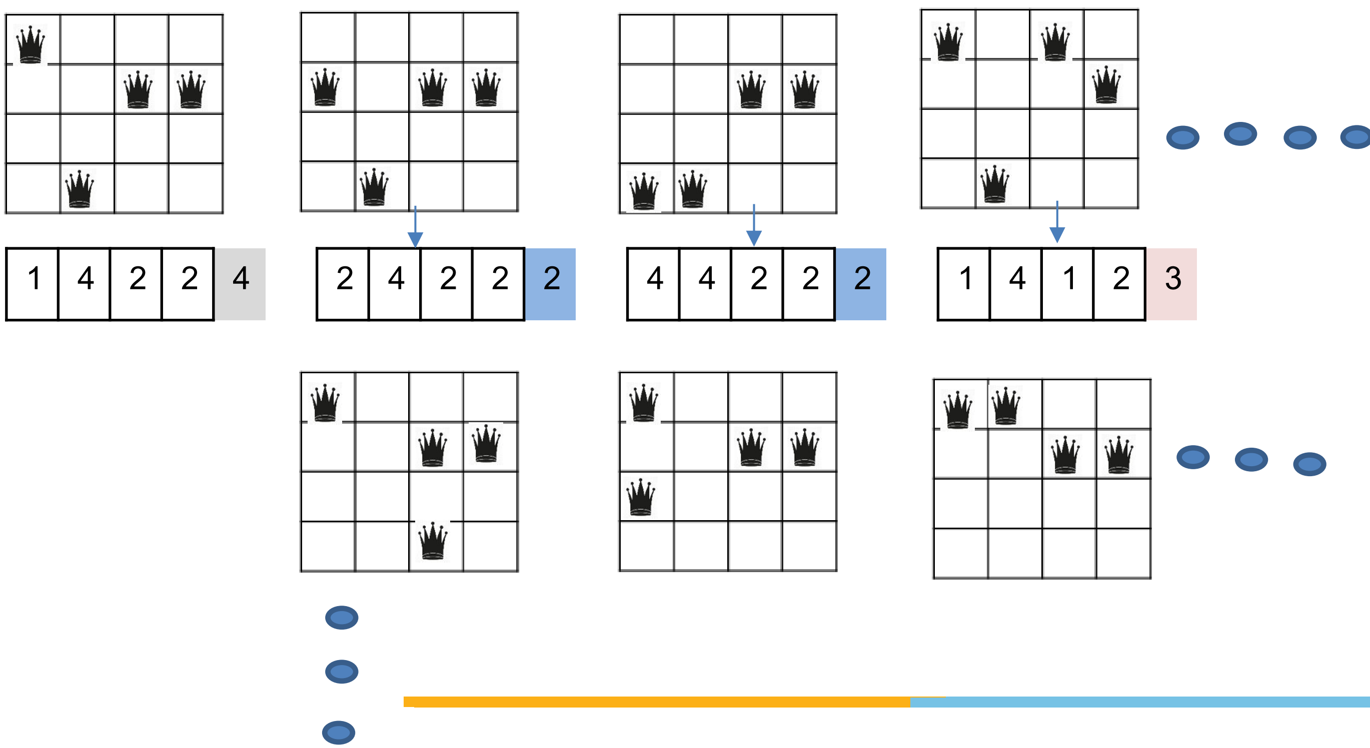




Beam Search

1st State

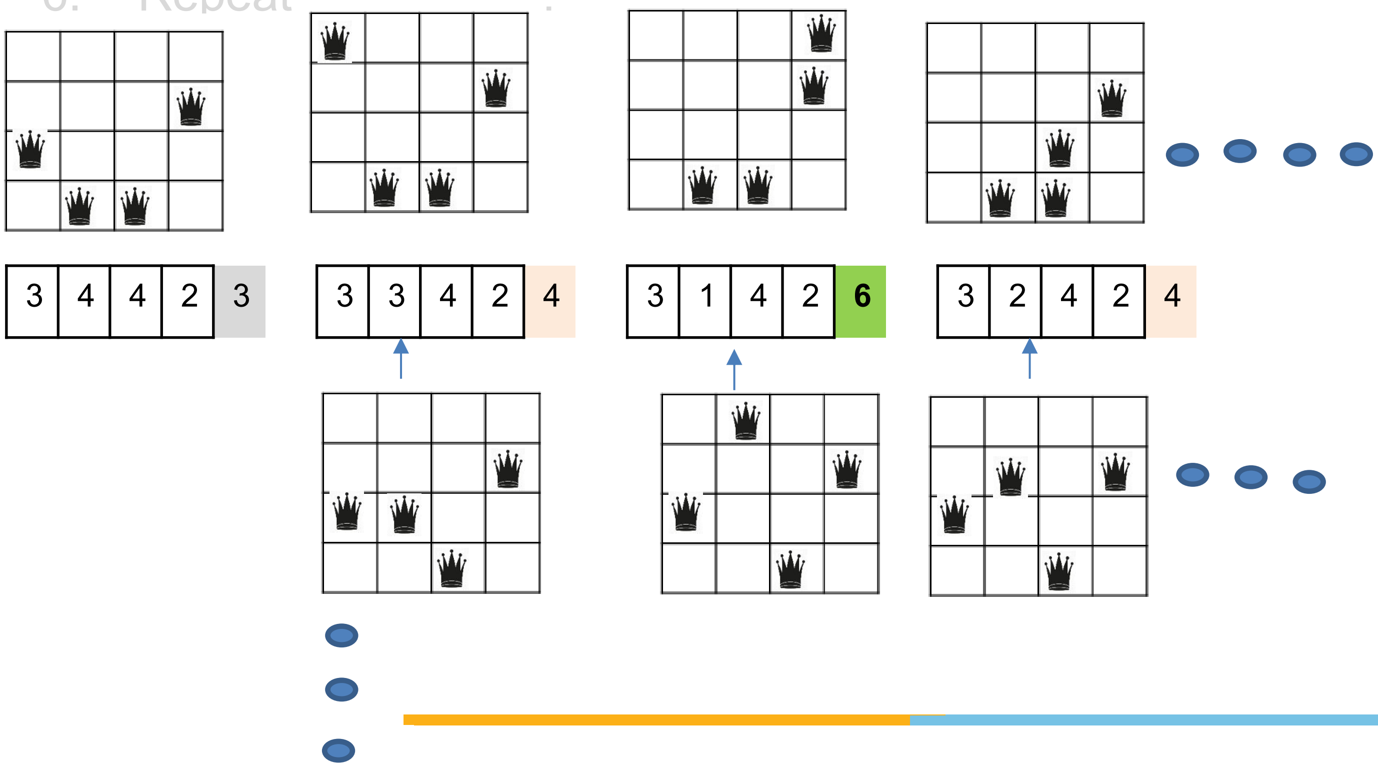
1. Initialize k random state
2. Evaluate the fitness scores for all the successors of the k states
3. Calculate the probability of selecting a successor based on fitness score
4. Select the next state based on the highest probability
5. If the goal is not found, Select the next ‘k’ states randomly based on the probability
6. Repeat from Step 2



Beam Search

2nd State

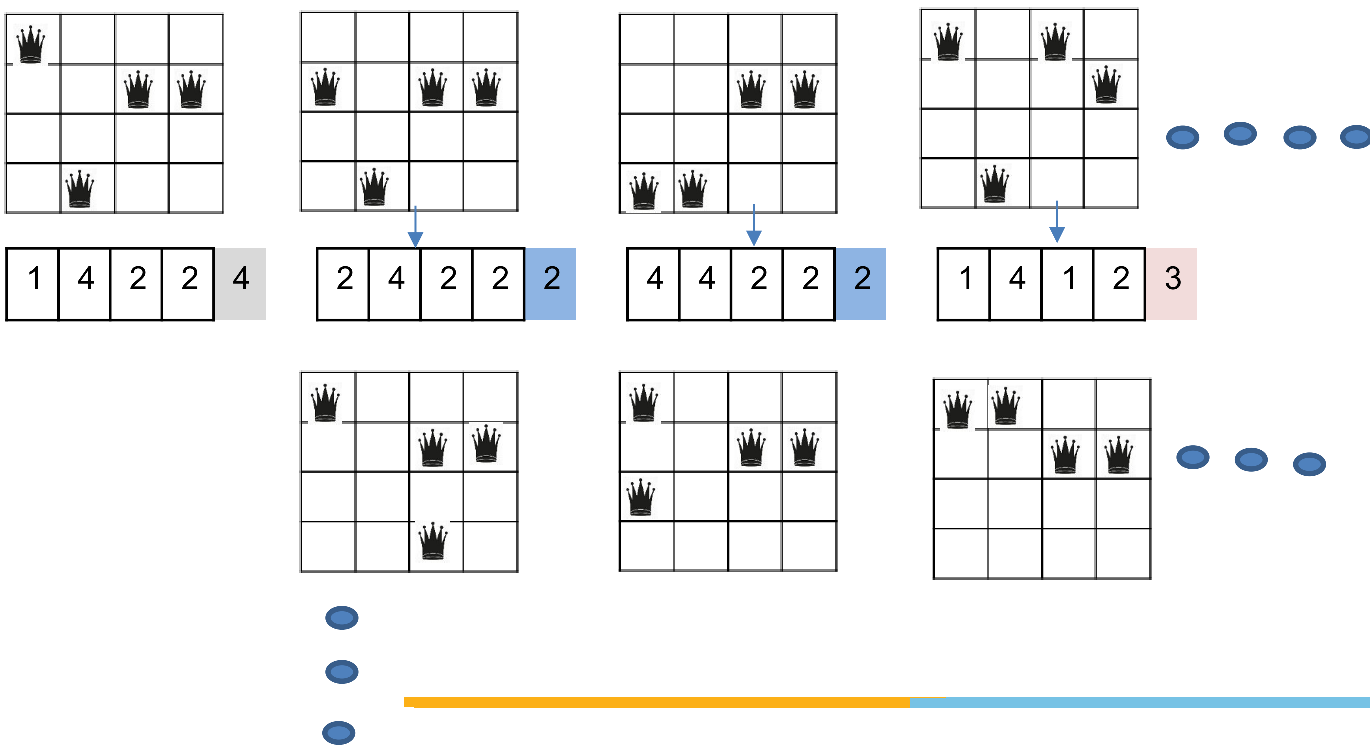
1. Initialize k random state
2. Evaluate the fitness scores for all the successors of the k states
3. Calculate the probability of selecting a successor based on fitness score
4. Select the next state based on the highest probability
5. If the goal is not found, Select the next ‘k’ states randomly based on the probability
6. Repeat from Step 2



Stochastic Beam Search

Sample from 1st State

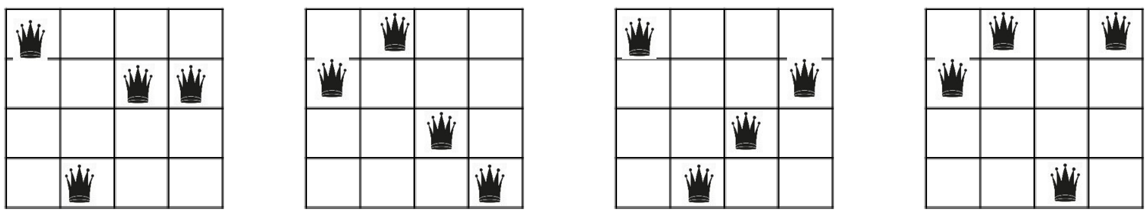
1. Initialize k random state
2. Evaluate the fitness scores for all the successors of the k states
3. Calculate the probability of selecting a successor based on fitness score
4. Select the next state based on the highest probability
5. If the goal is not found, Select the next ‘k’ states randomly based on the probability
6. Repeat from Step 2



Genetic Algorithm

Genetic Algorithm

1. Select ‘k’ random states – **Initialization : k=4**
2. Evaluate the fitness value all states : Maximizing function : No.of.Non-attacking pairs Queens → Threshold = 6



1	4	2	2	4

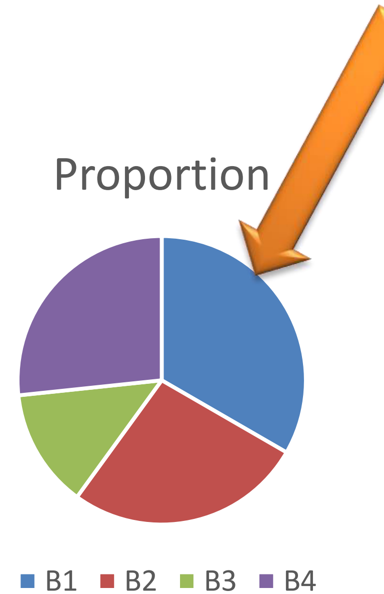
2	1	3	4	4

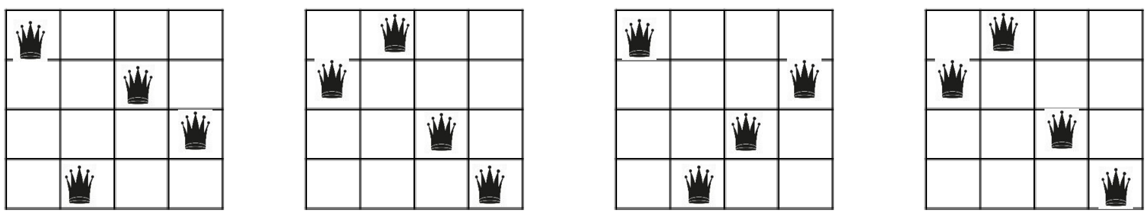
1	4	3	2	2

2	1	4	1	3

Genetic Algorithm

Eg., use roulette wheel mechanism to select pair/s





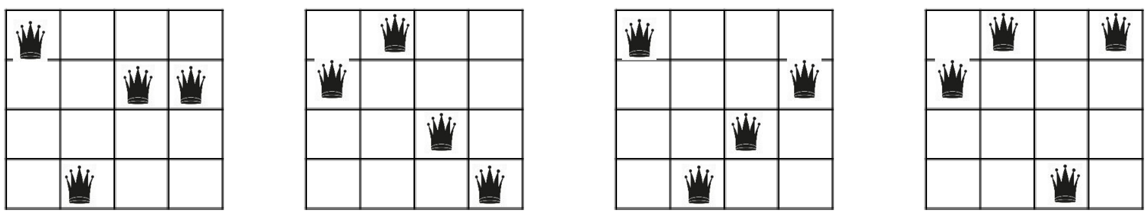
				Fitness	Proportion
1	4	2	3	5	0.33
2	1	3	4	4	0.27
1	4	3	2	2	0.13
2	1	3	4	4	0.27

2	1	3	4
1	4	2	3
1	4	2	3
1	4	3	2

Sample winners of game -1 ,2,3,4 : B4, B1, B1, B3

Genetic Algorithm

1. Select ‘k’ random states – **Initialization : k=4**
2. Evaluate the fitness value all states : Maximizing function : No.of.Non-attacking pairs Queens → Threshold = 6
3. ~~If anyone of the state’s has achieved the threshold fitness value or threshold new states or no change is seen than previous iteration then the algorithm stops~~
4. Else, use roulette wheel mechanism to select pair/s
5. Pairs selected produces new state (successor) by crossover
6. Successor is allowed to mutate
7. Repeat from Step 2



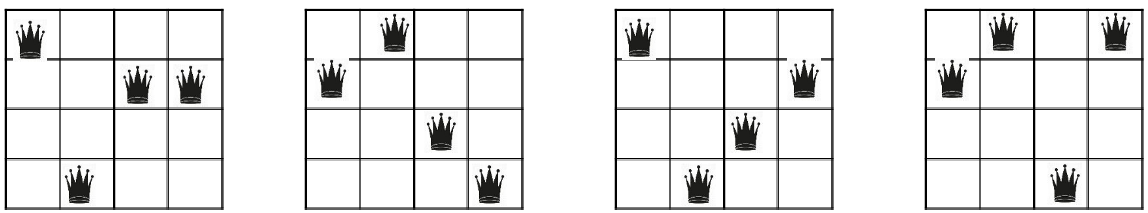
1	4	2	2	4	0.31
2	1	3	4	4	0.31
1	4	3	2	2	0.15
2	1	4	1	3	0.23

2	1	4	1
1	4	2	2
1	4	2	2
1	4	3	2

Sample winners of game -1 ,2,3,4 : B4, B1, B1, B3

Genetic Algorithm

1. Select ‘k’ random states – **Initialization : k=4**
2. Evaluate the fitness value all states : Maximizing function : No.of.Non-attacking pairs Queens → Threshold = 6
3. ~~If anyone of the state’s has achieved the threshold fitness value or threshold new states or no change is seen than previous iteration then the algorithm stops~~
4. Else, use roulette wheel mechanism to select pair/s
5. Pairs selected produces new state (successor) by crossover
6. Successor is allowed to mutate
7. Repeat from Step 2



1	4	2	2	4	0.31
2	1	3	4	4	0.31
1	4	3	2	2	0.15
2	1	4	1	3	0.23

Crossover pairs:

2	1	4	1	⟺	1	4	2	2
1	4	2	2	⟺	1	4	3	2

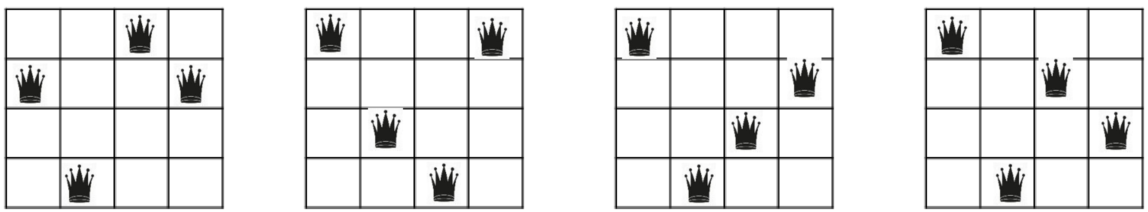
Successors:

2	4	2	2
1	1	4	1
1	4	3	2
1	4	2	2

http://ictactjournals.in/paper/IJSC_V6_I1_paper_4_pp_1083_1092.pdf

Genetic Algorithm

1. Select 'k' random states – **Initialization : k=4**
2. Evaluate the fitness value all states : Maximizing function : No.of.Non-attacking pairs Queens → Threshold = 6
3. ~~If anyone of the state's has achieved the threshold fitness value or threshold new states or no change is seen than previous iteration then the algorithm stops~~
4. Else, use roulette wheel mechanism to select pair/s
5. Pairs selected produces new state (successor) by crossover
6. Successor is allowed to mutate
7. Repeat from Step 2



2	4	2	2

1	1	4	1

1	4	3	2

1	4	2	2

2	4	1	2

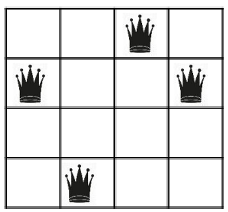
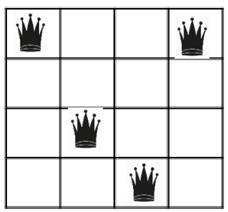
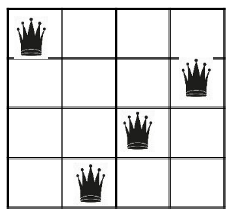
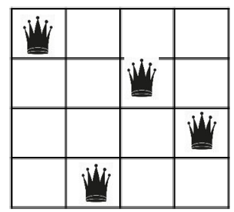
1	3	4	1

1	4	3	2

1	4	2	3

Genetic Algorithm

1. Select 'k' random states – **Initialization : k=4**
2. Evaluate the fitness value all states : Maximizing function : No.of.Non-attacking pairs Queens → Threshold = 6
3. If anyone of the state's has achieved the threshold fitness value or threshold new states or no change is seen than previous iteration then the algorithm stops
4. Else, use roulette wheel mechanism to select pair/s
5. Pairs selected produces new state (successor) by crossover
6. Successor is allowed to mutate
7. Repeat from Step 2

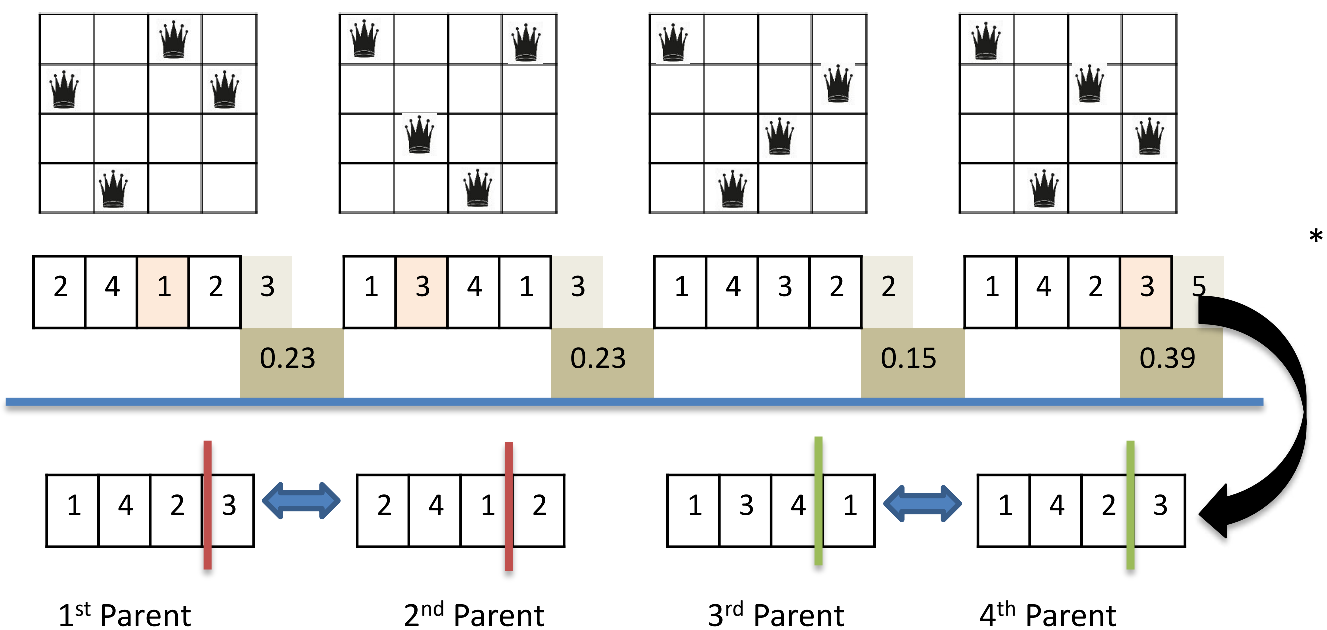





2	4	2	2
1	1	4	1
1	4	3	2
1	4	2	2

2	4	1	2	3	0.23
1	3	4	1	3	0.23
1	4	3	2	2	0.15
1	4	2	3	5	0.39

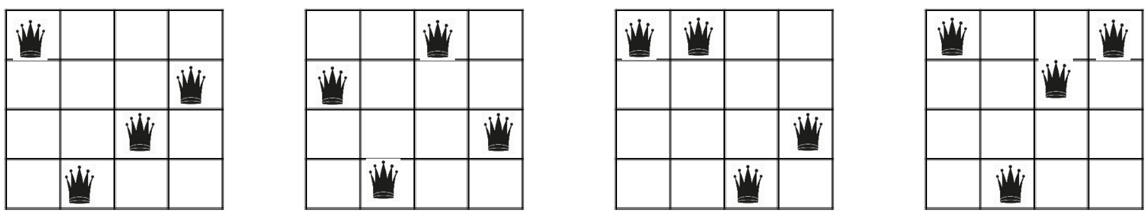
Genetic Algorithm

1. Select 'k' random states – **Initialization : k=4**
2. Evaluate the fitness value all states : Maximizing function : No.of.Non-attacking pairs Queens → Threshold = 6
3. If anyone of the state's has achieved the threshold fitness value or threshold new states or no change is seen than previous iteration then the algorithm stops
4. Else, use roulette wheel mechanism to select pair/s
5. Pairs selected produces new state (successor) by crossover
6. Successor is allowed to mutate
7. Repeat from Step 2



Genetic Algorithm

1. Select 'k' random states – **Initialization : k=4**
2. Evaluate the fitness value all states : Maximizing function : No.of.Non-attacking pairs Queens → Threshold = 6
3. If anyone of the state's has achieved the threshold fitness value or threshold new states or no change is seen than previous iteration then the algorithm stops
4. Else, use roulette wheel mechanism to select pair/s
5. Pairs selected produces new state (successor) by crossover
6. Successor is allowed to mutate
7. Repeat from Step 2



1 4 2 3	2 4 1 2	1 3 4 1	1 4 2 3
1 4 2 2	2 4 1 3	1 3 4 3	1 4 2 1
1 4 3 2 | 2	2 4 1 3 | 6	1 1 4 3 | 3	1 4 2 1 | 4

Genetic Algorithm

Techniques:

1. Design of the fitness function
2. Diversity in the population to be accounted
3. Randomization

Application:

- Creative tasks
- Exploratory in nature
- Planning problem
- Static Applications

Genetic Algorithm

```
function GENETIC-ALGORITHM(population, FITNESS-FN) returns an individual
   inputs: population, a set of individuals
           FITNESS-FN, a function that measures the fitness of an individual

   repeat
      new_population ← empty set
      for i = 1 to SIZE(population) do
         x ← RANDOM-SELECTION(population, FITNESS-FN)
         y ← RANDOM-SELECTION(population, FITNESS-FN)
         child ← REPRODUCE(x, y)
         if (small random probability) then child ← MUTATE(child)
         add child to new_population
      population ← new_population
   until some individual is fit enough, or enough time has elapsed
   return the best individual in population, according to FITNESS-FN
```

```
function REPRODUCE(x, y) returns an individual
   inputs: x, y, parent individuals

   n ← LENGTH(x); c ← random number from 1 to n
   return APPEND(SUBSTRING(x, 1, c), SUBSTRING(y, c + 1, n))
```

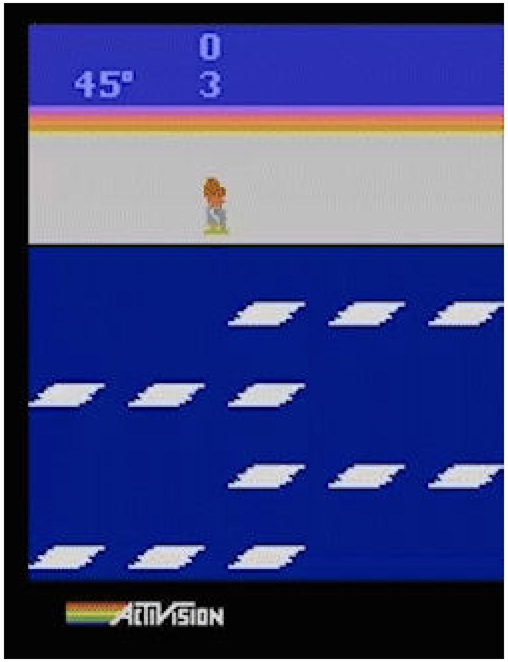
Hyper Parameter Optimization

Examples

- Parameter
- Hyper Parameter
- HP Optimization or Tuning

- K = No.of.Clusters
- C = Regularization , in LR
- Penalty {L1, L2} & class_weight in LogR
- Loss in SGD
- Learning Rate in GD
- Maximum Depth, No.of.Instances at Leaf , No.of.Trees in DT & RF
- No.of.Neurons, No.of.Layers in NN

Genetic Algorithm



Source Credit:

https://ai.googleblog.com/2018/03/using-evolutionary-automl-to-discover.html

https://eng.uber.com/deep-neuroevolution/

Task Environment

Phases of Solution Search by PSA

Goal Formulation

Problem Formulation

Search Phase

Execution Phase

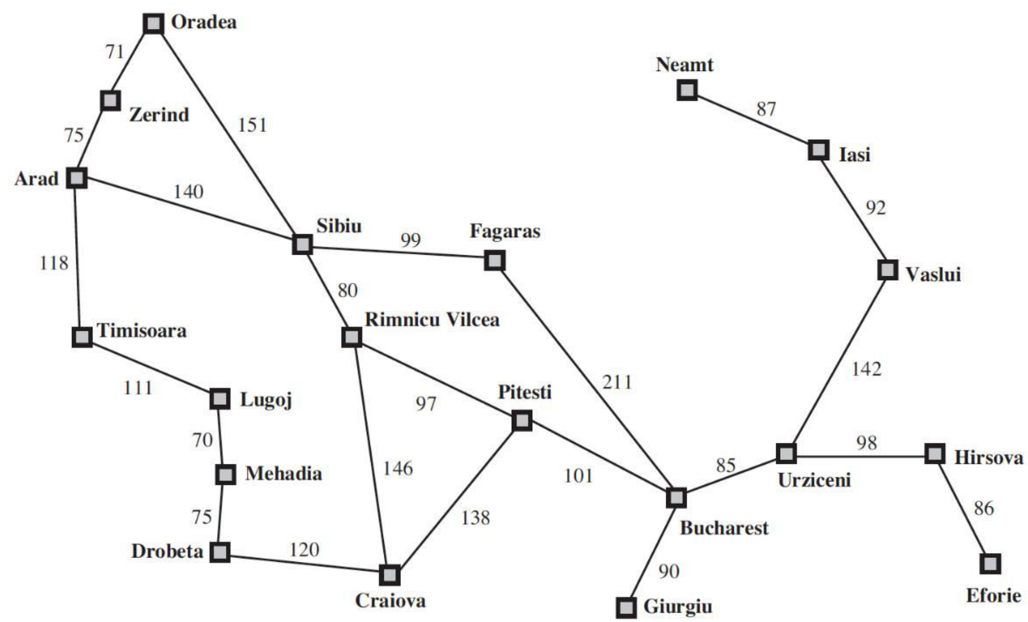
Assumptions – Environment :

Static (4.5)

Observable

Discrete (4.4)

Deterministic (MDP)



Learning Outcome

1. Differentiate which local search is best suitable for given problem
2. Design fitness function for a problem
3. Construct a search tree for finite successors & evaluate the goodness
4. Apply appropriate local search and show the working of algorithm at least for first 2 iterations with at least four next level successor generation(if search tree is large)
5. Design and show Genetic Algorithm steps for a given problem

Note:
In your upcoming webinar 2 Genetic algorithm implementation in python will be demonstrated.
Next module game will also be demo'd. We shall try to provide sufficient introduction for the same during the webinar.
Detailed Min-Max algorithm for games will be covered in next Saturday class

Required Reading: AIMA - Chapter #4.2 , #4.3

Thank You for all your Attention

Note : Some of the slides are adopted from AIMA TB materials