BITS Pilani
Pilani Campus

Artificial & Computational Intelligence

AIMLCLZG557

M2 : Problem Solving Agent using Search

Indumathi V
Guest Faculty,
BITS - WILP

Course Plan

M1	Introduction to AI
M2	Problem Solving Agent using Search
M3	Game Playing
M4	Knowledge Representation using Logics
M5	Probabilistic Representation and Reasoning
M6	Reasoning over time
M7	Ethics in AI

Module 2 : Problem Solving Agent using Search

A. Uninformed Search

B. Informed Search

C. Heuristic Functions

D. Local Search Algorithms & Optimization Problems

Learning Objective

At the end of this class , students Should be able to:

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

Local Search & Optimization

↓

Greedy choice

↓

next best successor

① Partially observable

② Irrelavant Path

③ Goal Oriented

↓

expected Goal

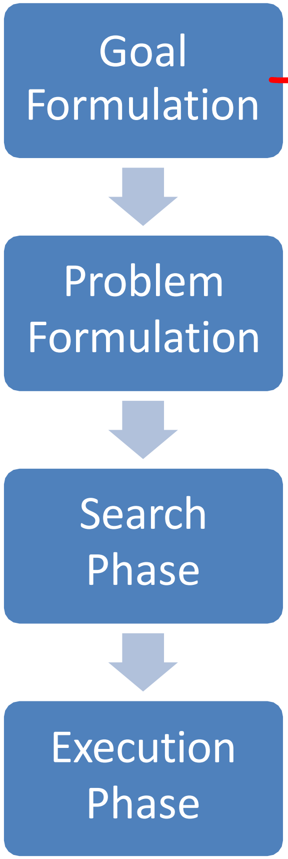
④ Memory limitation

Task Environment

① single instance

② multiple instance

Phases of Solution Search by PSA



→ optimal soln

↓

nearer to my goal

* mini

* max

h(n)

↳ fitness ƒn

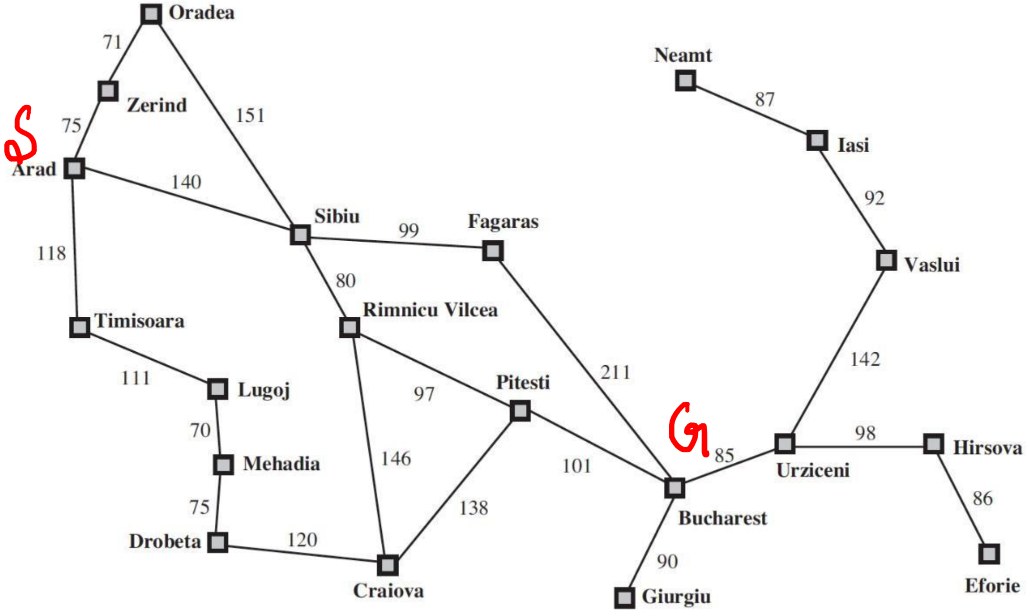
Assumptions – Environment :

Static (4.5)

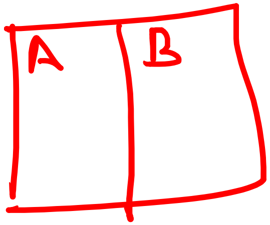
Observable

Discrete (4.4)

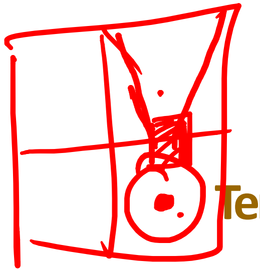
Deterministic (MDP)

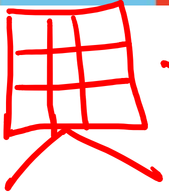


Local Search



fully

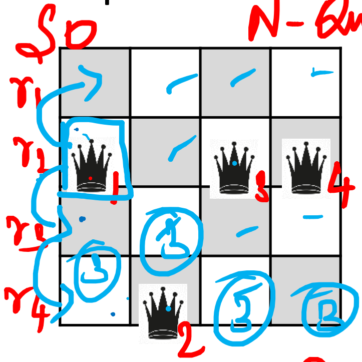


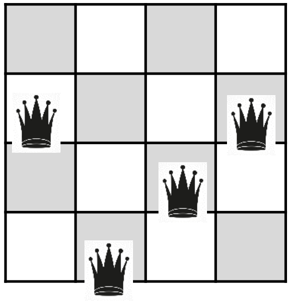

✓

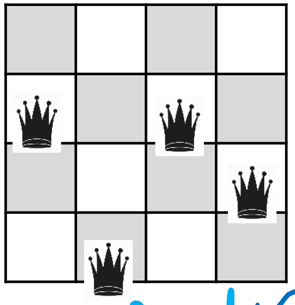
Terminology

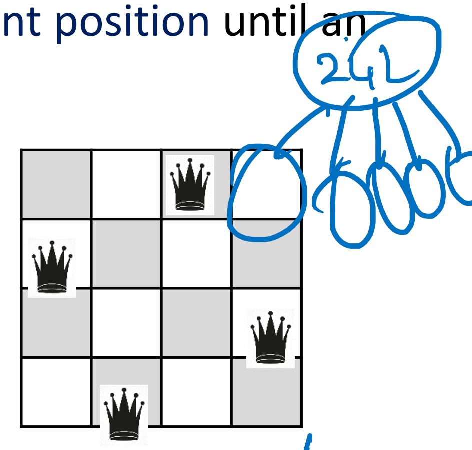
Local Search : Search in the state-space in the neighbourhood of current position until an optimal solution is found

N-Queen → 4









Q1	Q2	Q3	Q4
2	4	r2	r2

S0 [2, 4, 2, 2]

1 — 1NN → Neighborhood region expansion

var 2NN

3NN

⋮

NNN

4 NNN

level 1

2,4, 2,2

1,1

1,

22

22

2 4, 2 ,

2,4,2,2

1,4,22 3,4,22 4,4,22

3 3 3 3 3

2 1 2 2
2
3

⇒ 12 Successors

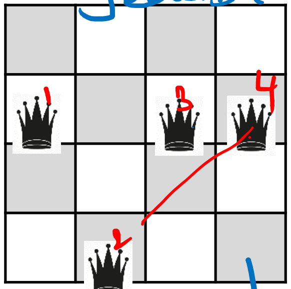
Local Search

INN Assumption

Terminology

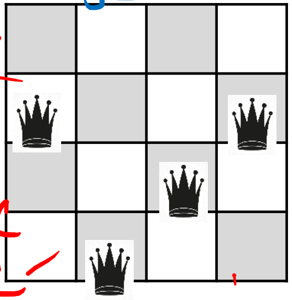
Local Search : Search in the state-space in the neighbourhood of current position until an optimal solution is found

feasible state

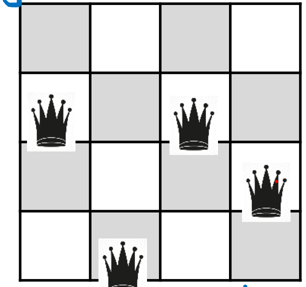


Q1Q2 NC
Q1Q3 C
Q1Q4 C
Q2Q3 NC
Q2Q4 C
Q3Q4 C

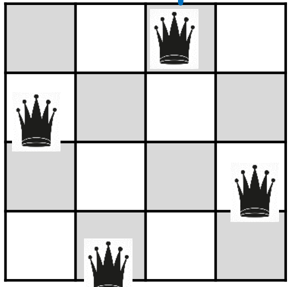
fs -



fs -



optimal



Feasible State/Solution **Neighboring States** **Optimal Solution**

Fitness Value:

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

D1, h(n) = No.of Conflicting **pairs** of queens ↓ min

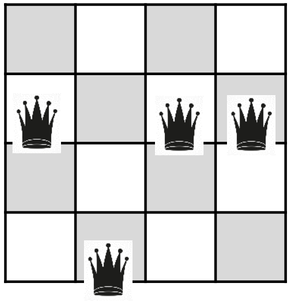
h(n) = 2 h(n) = 2 h(n) = 4 h(n) = 6

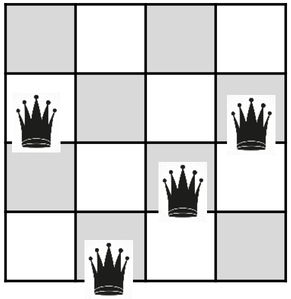
D2 h(n) = No.of.**Non**-Conflicting **pairs** of queens. ↑ max

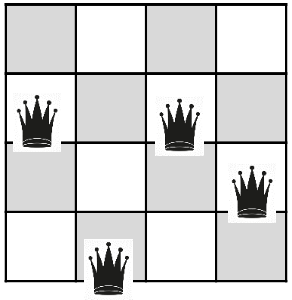
Local Search

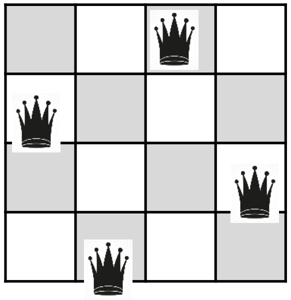
Terminology

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 mix
h(n) = No.of.Conflicting **pairs** of queens			
h(n) = 2	h(n) = 2	h(n) = 4	h(n) = 6 max
h(n) = No.of.**Non**-Conflicting **pairs** of queens.			

Local Search

Terminology

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

Algorithms:

- Choice of Neighbour ✓ 1NN, 2NN
- Looping Condition ✓
- Termination Condition

→ expected Goal → TC1

→ h(n) = 6 → Threshold a 4

↓ expected

2	5	3	2
♛	6	♛	♛
3	5	4	2
4	♛	4	2

Local Search

Optimization Problem

Goal : Navigate through a state space for a given problem such that an optimal solution can be found

Objective : Minimize or Maximize the objective evaluation function value

Scope : Local

Objective Function : Fitness Value evaluates the goodness of current solution

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

Single Instance Based

- ✓ Hill Climbing
- Simulated Annealing
- ✓ Local Beam Search
- Tabu Search

Multiple Instance Based

- ✓ Genetic Algorithm
- Particle Swarm Optimization
- ✓ Ant Colony Optimization

4 Alg

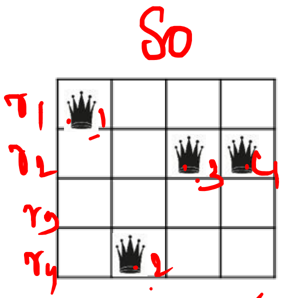
Hill Climbing

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.

NC Q1-Q2
NC Q1-Q3
NC Q1-Q4
NC Q2-Q3
C Q2-Q4
C Q3-Q4



1	4	2	2	4

Note : Steps 3 & 4 in the above algorithm will be a part of variation of 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

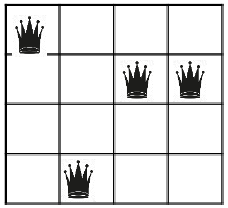
Q1-Q3

Q1-Q4

Q2-Q3

Q2-Q4

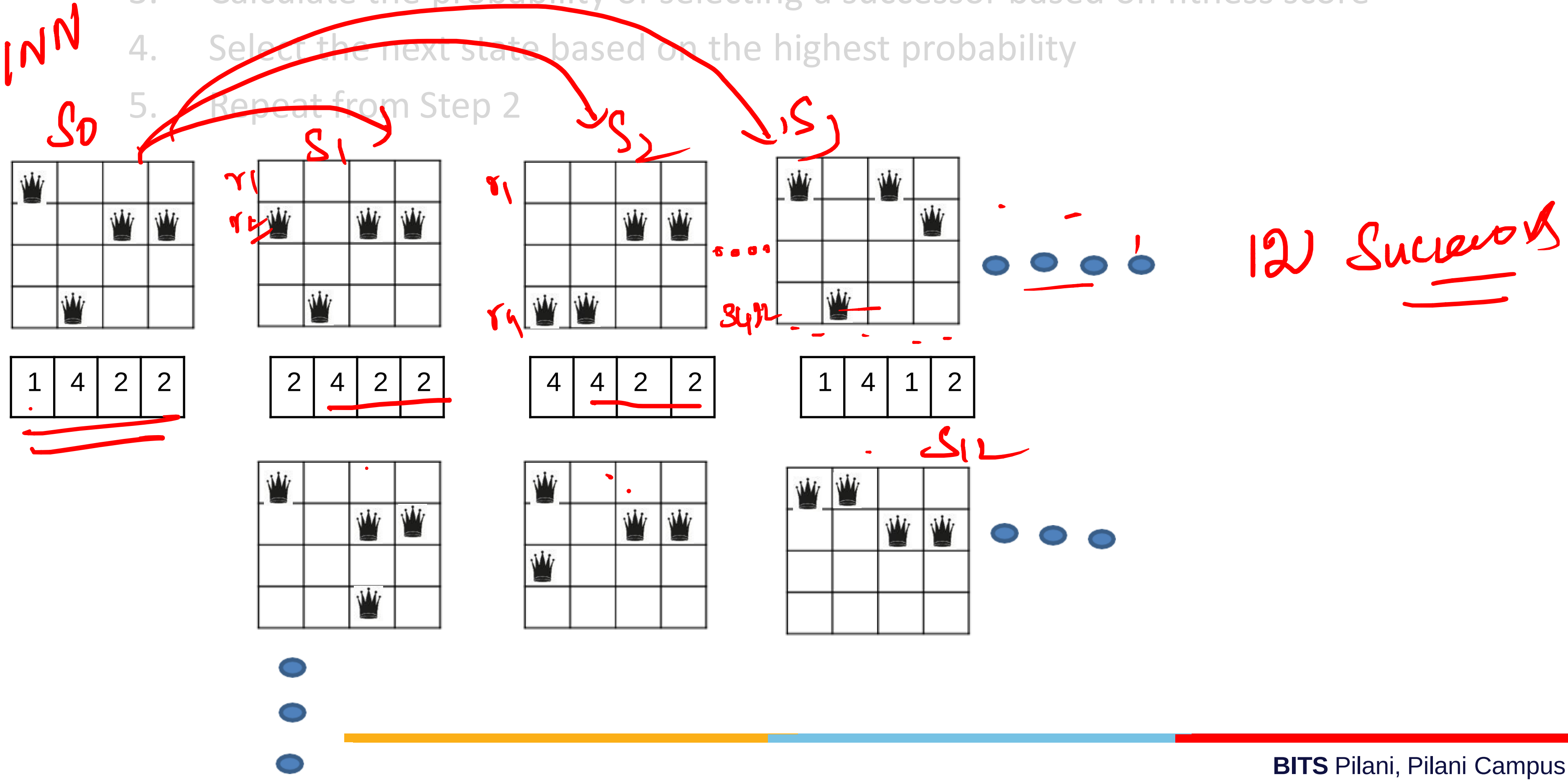
Q3-Q4



1	4	2	2	4	✓

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

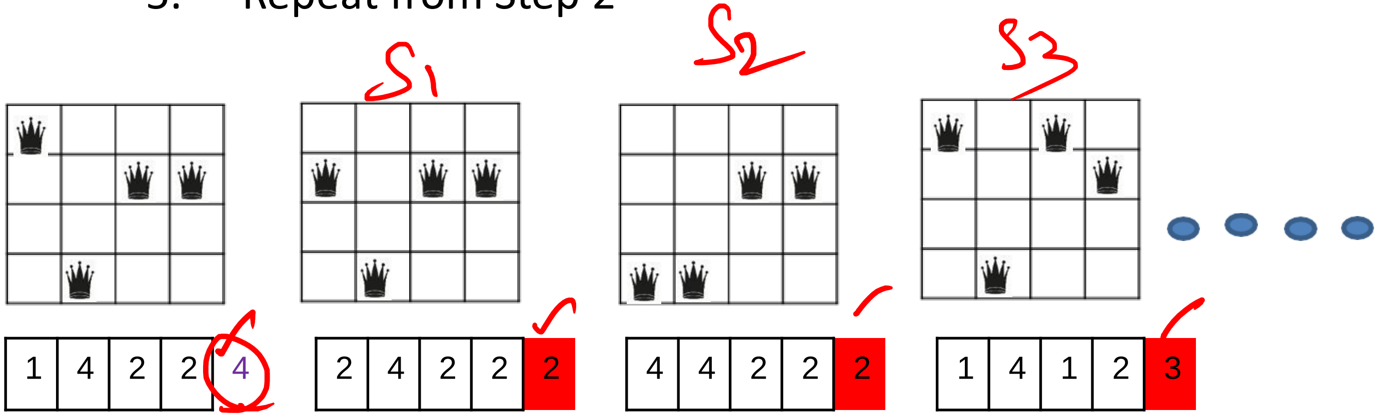
2	4	2	2

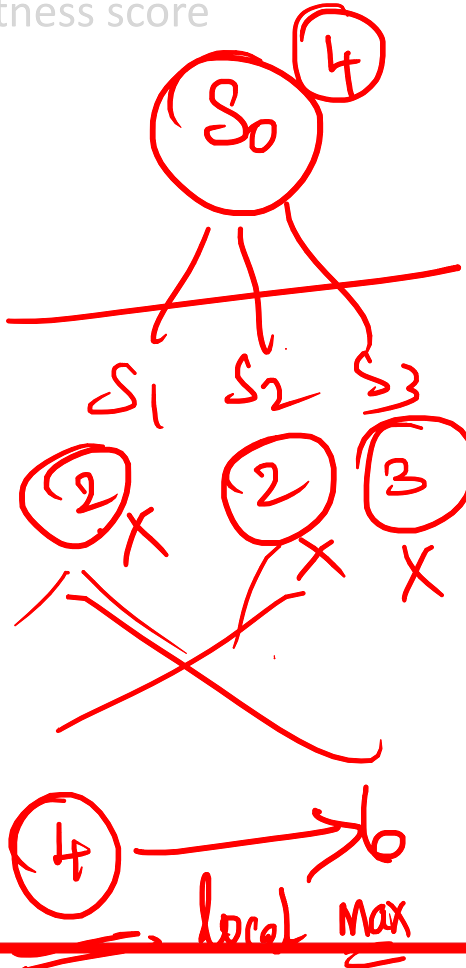
4	4	2	2

1	4	1	2

Hill Climbing

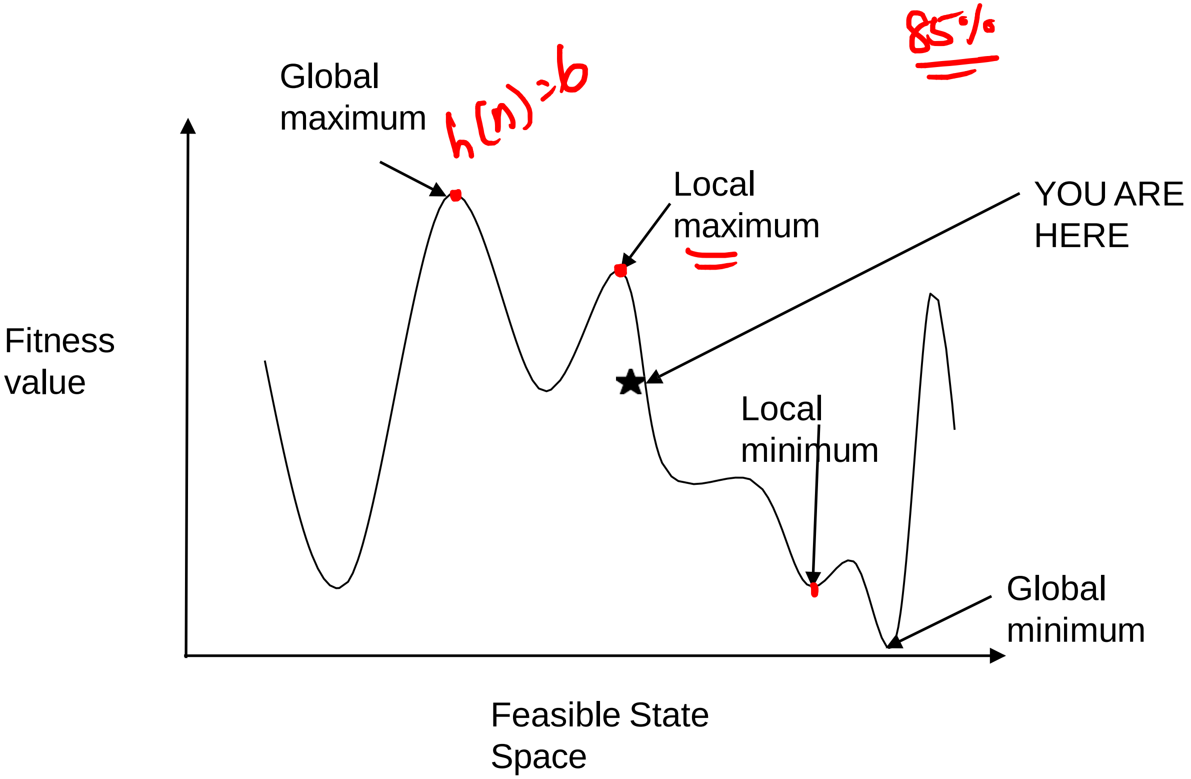
1. Select a random state
2. Evaluate the fitness scores for all the successors of the state
3. Calculate the probability of selecting a successor based on fitness score
4. Select the next state based on the highest probability
5. Repeat from Step 2





return S0 optimal sol

Hill Climbing



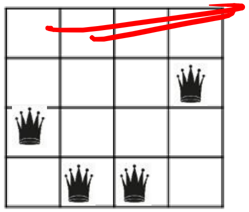
Hill Climbing

Random Restart

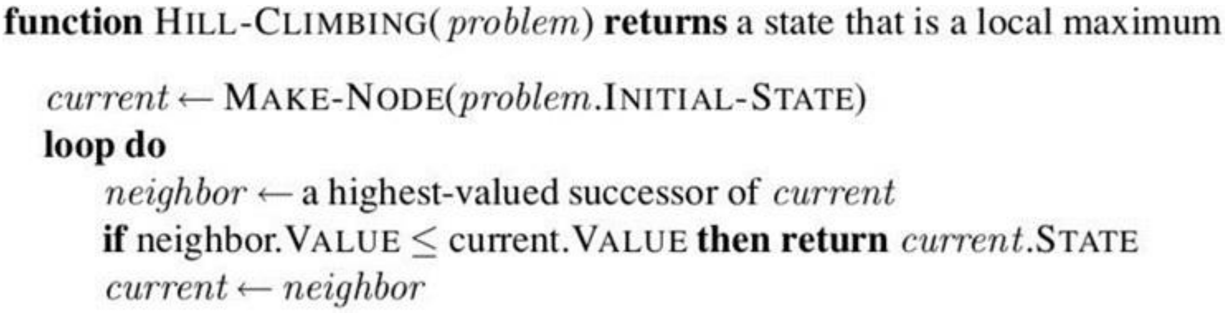
1. Select a random state
2. Evaluate the fitness scores for all the successors of the state
3. Calculate the probability of selecting a successor based on fitness score
4. Select the next state based on the highest probability

5. Repeat from Step 2 6.

another random



3	4	4	2	3



```
function HILL-CLIMBING(problem) returns a state that is a local maximum
  current ← MAKE-NODE(problem.INITIAL-STATE)
  loop do
     neighbor ← a highest-valued successor of current
     if neighbor.VALUE ≤ current.VALUE then return current.STATE
     current ← neighbor
```

Hill Climbing

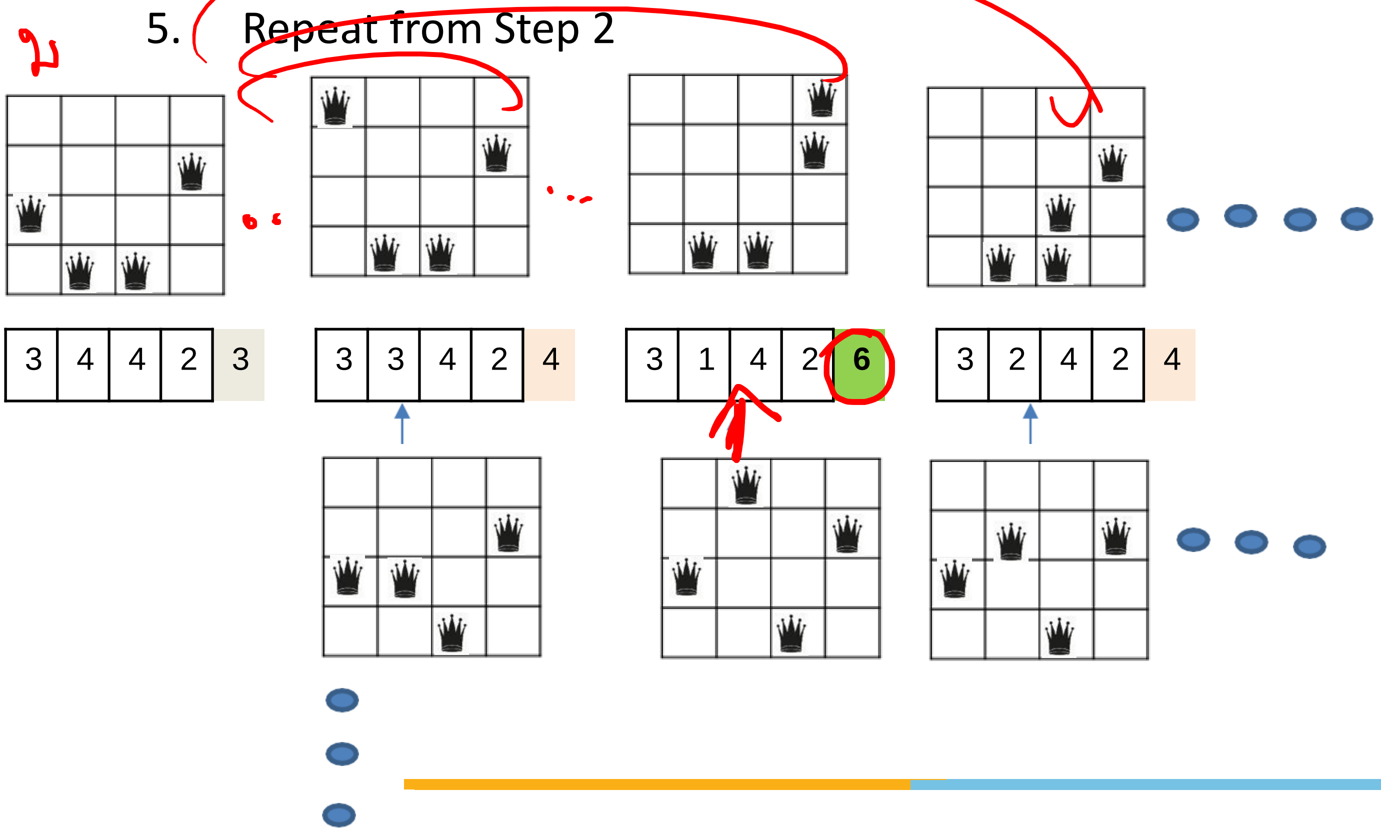
K= 10

K=100

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

3

4

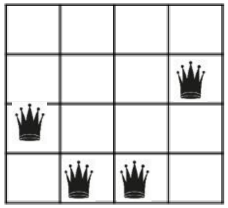


Hill Climbing

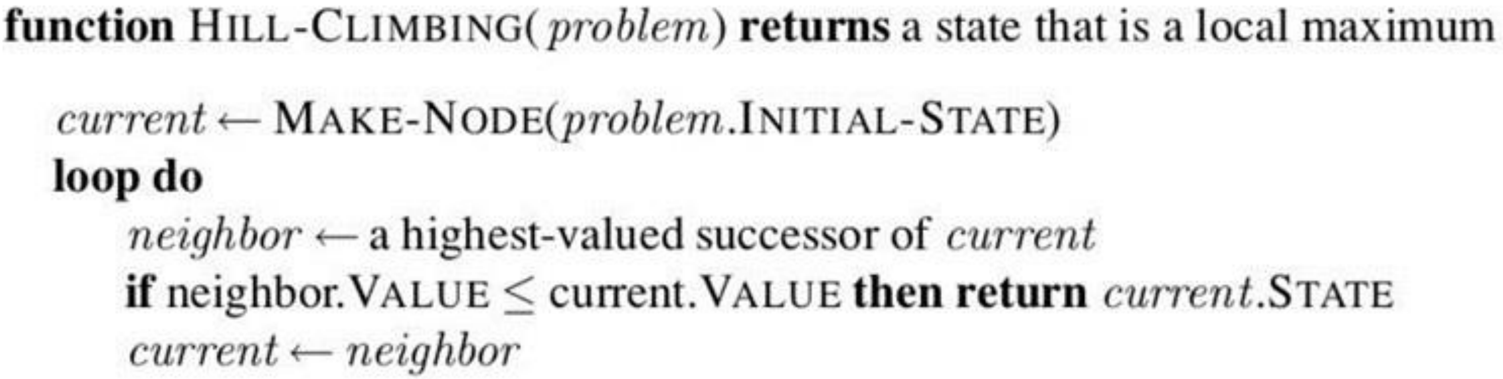
Random Restart

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 6.



3	4	4	2	3



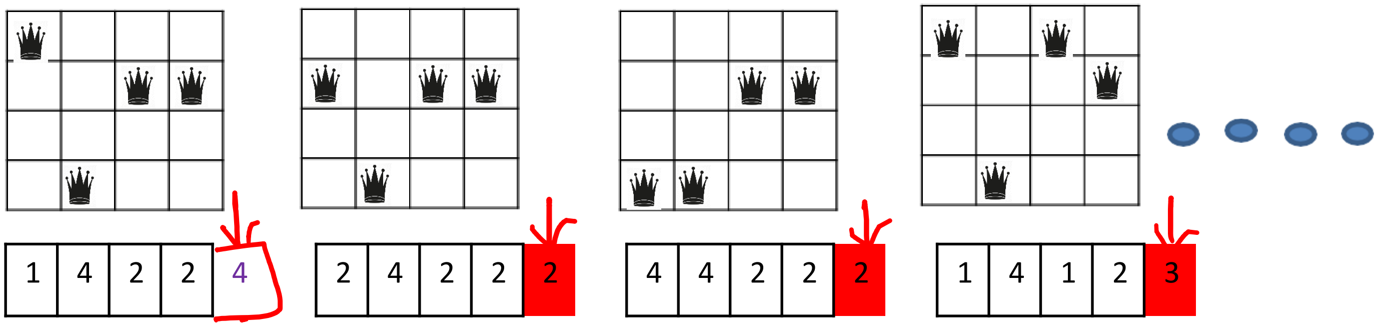
```
function HILL-CLIMBING(problem) returns a state that is a local maximum

  current ← MAKE-NODE(problem.INITIAL-STATE)
  loop do
     neighbor ← a highest-valued successor of current
     if neighbor.VALUE ≤ current.VALUE then return current.STATE
     current ← neighbor
```

Stochastic Hill Climbing

$f(n) \rightarrow \boxed{T} \rightarrow$ Probability

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

① way max

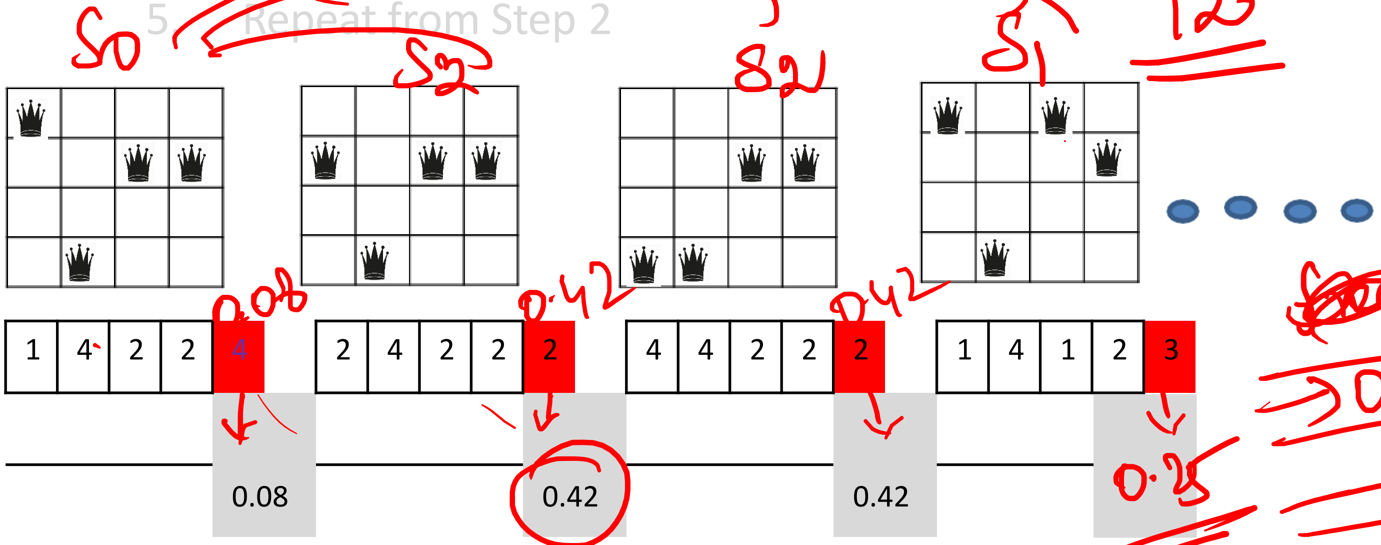
② P=0.2

0.2 < 0.25

INN

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

12

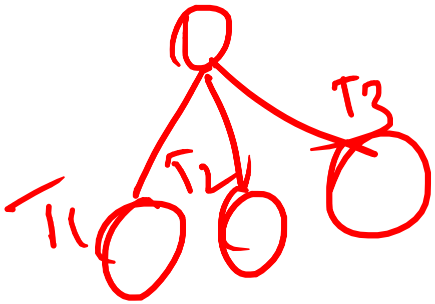


~~[illegible]~~	freq		
0	2	2/12	0.16
1	1	1/12	0.08
2	5	5/12	0.42
3	3	3/12	0.25
4		1/12	0.08

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

f(n)

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



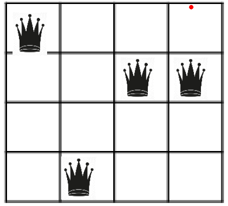
Local Beam Search

Beam Search

k=2. k 3

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 probabilit
6. Repeat from Step 2

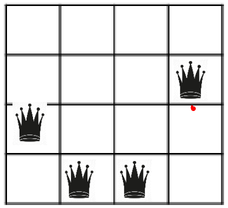
S_0



1	4	2	2	4

Hill | Sto (ra

S_0'

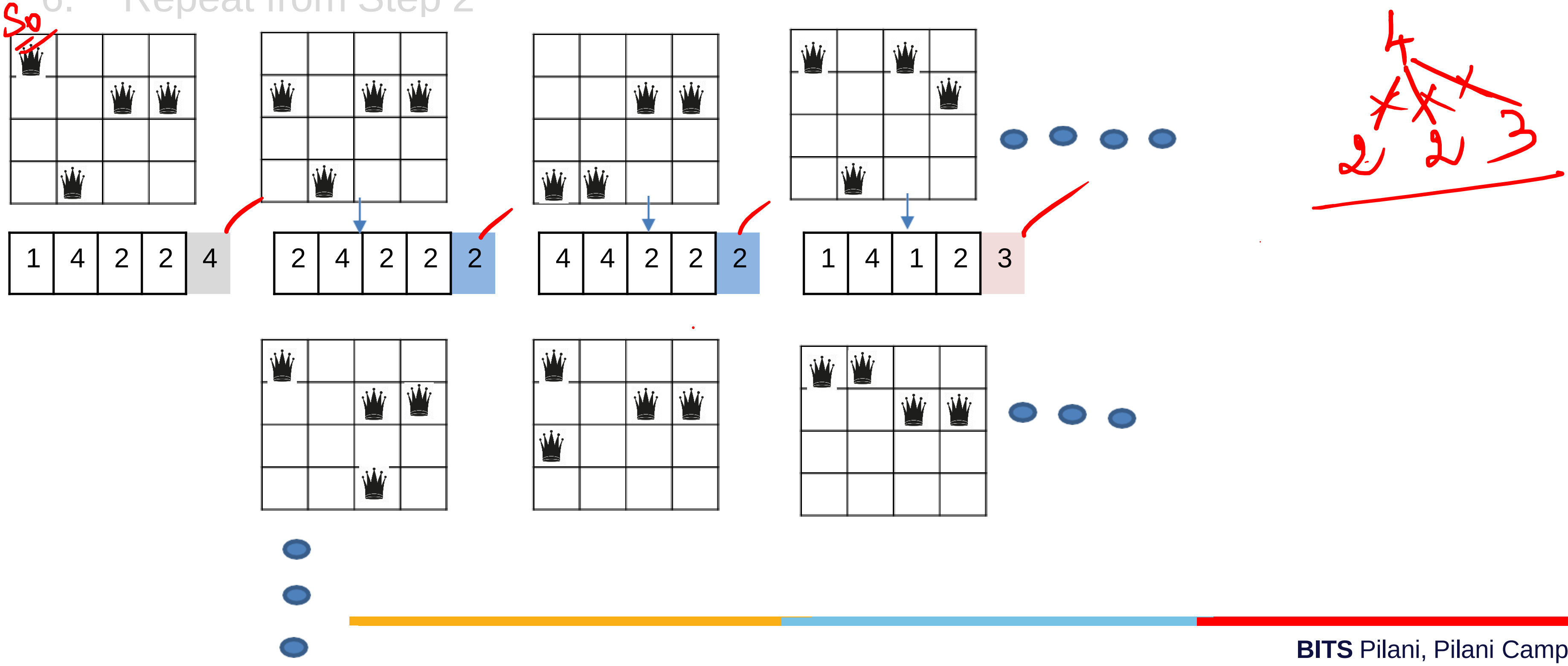


3	4	4	2	3

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



1	4	2	2	4

2	4	2	2	2

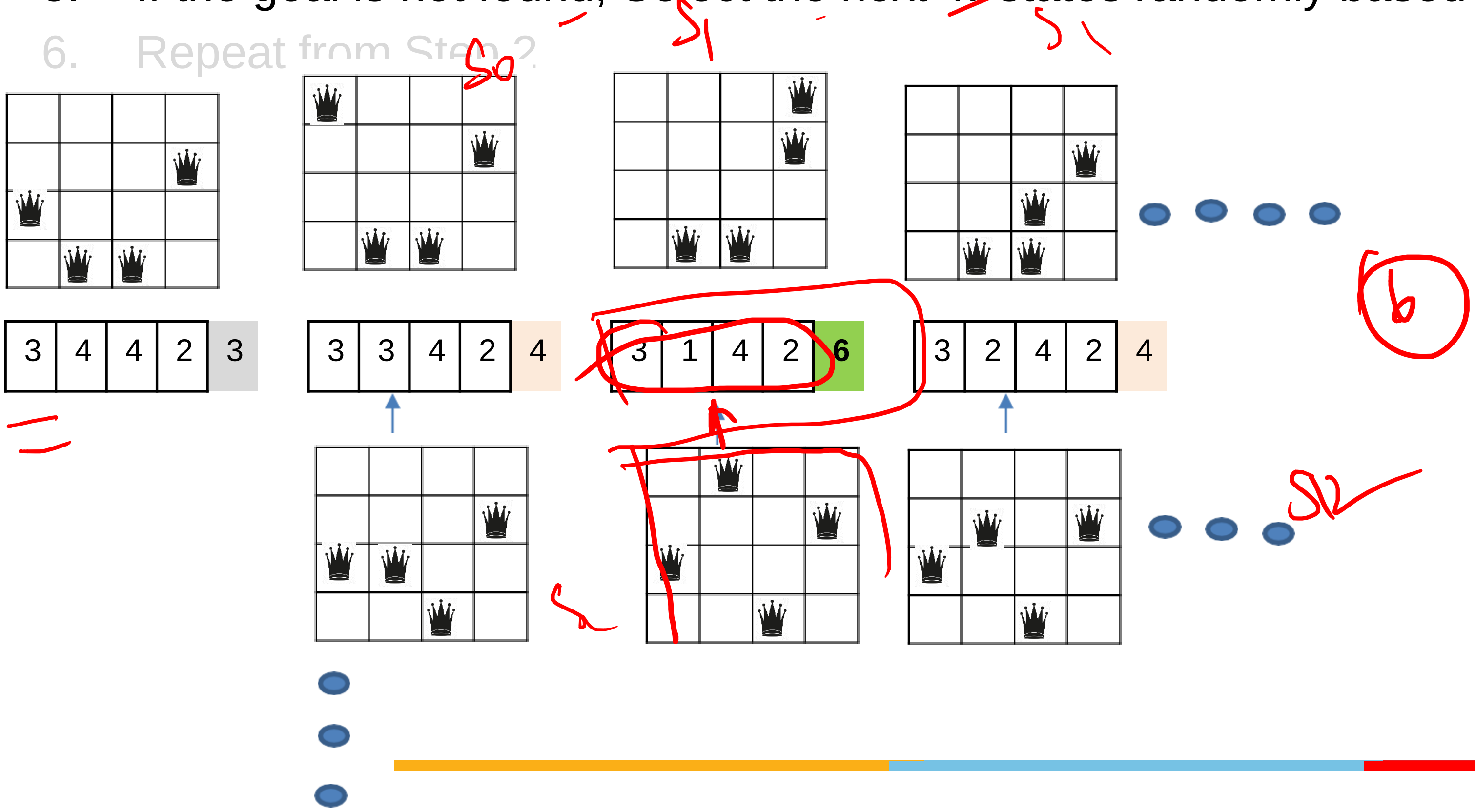
4	4	2	2	2

1	4	1	2	3

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



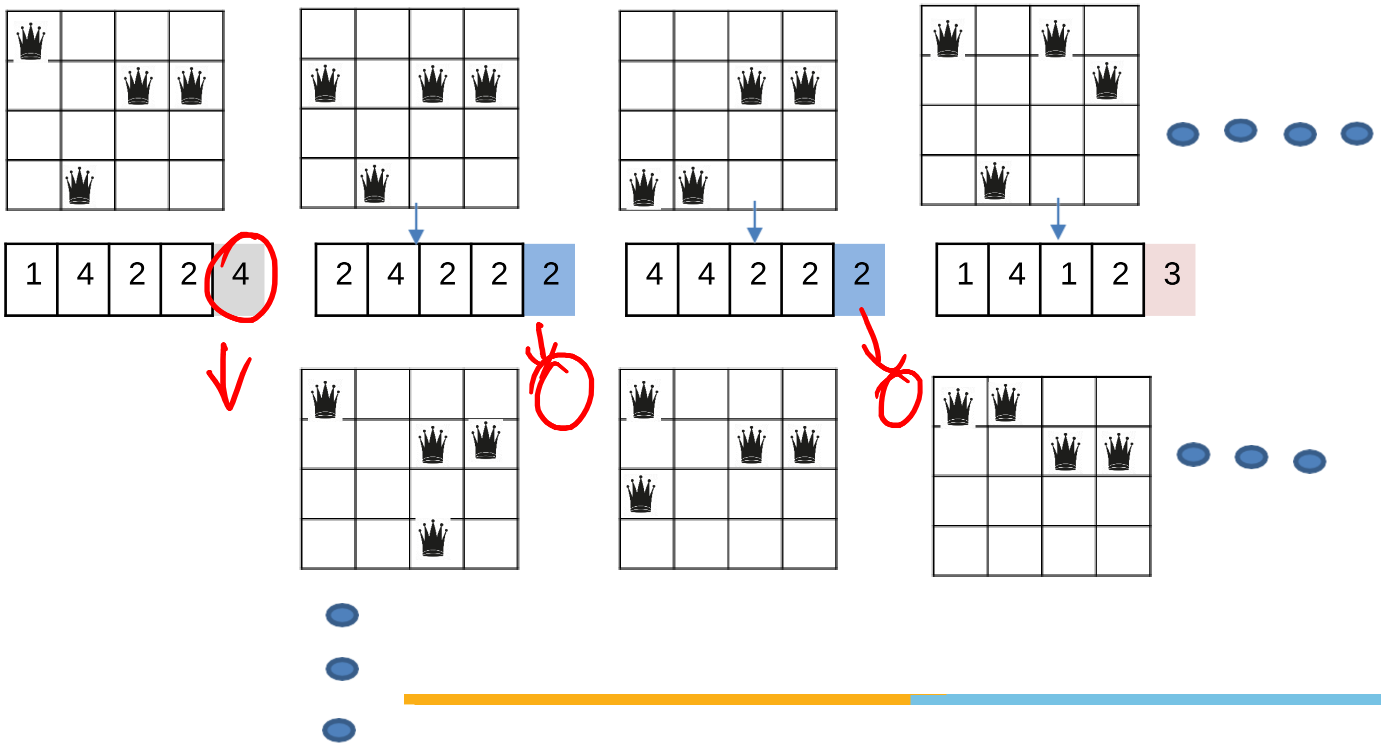
3	4	4	2	3
3	3	4	2	4
3	1	4	2	6
3	2	4	2	4

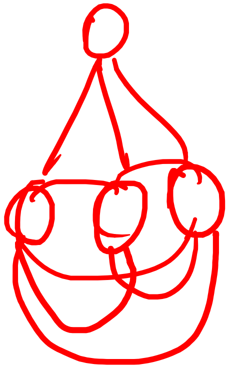
Stochastic Beam Search

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

2 ↓ 1st





Communicate → No cooperation

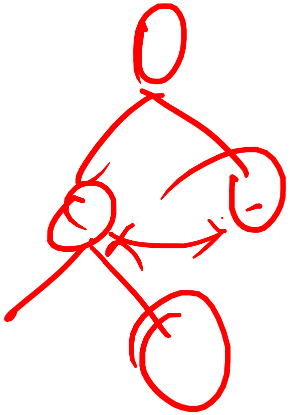
Genetic Algorithm

4 Selection f(n)

Randamization

Heridity

Mutation



offspring.

Par

Genetic Algorithm

f1	f2	f3	f4
1	0	1	0

binary

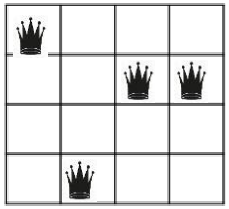
1010 INT

Chromosom

1. Select ‘k’ random states – **Initialization : k=4**
2. Evaluate the fitness value all states
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. elect
6. ces n ssor)
7. ver S allowed to mutate

Repeat from Step 2

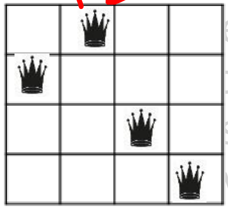
P1 ✓



1	4	2	2

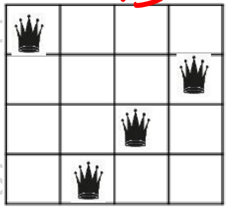
Chromosom

P2



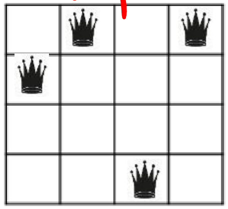
2	1	3	4

P3



1	4	3	2

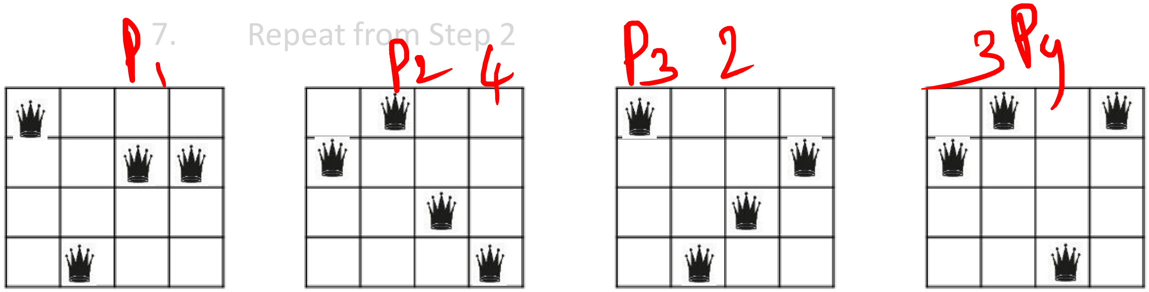
P4



2	1	4	1

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

4

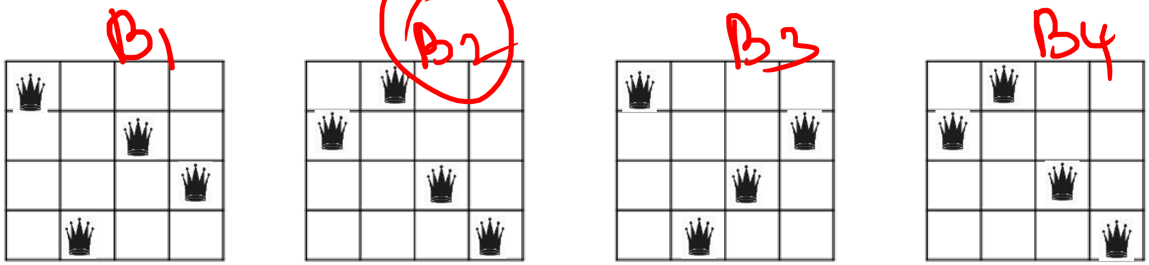
2	1	3	4	4

1	4	3	2	2

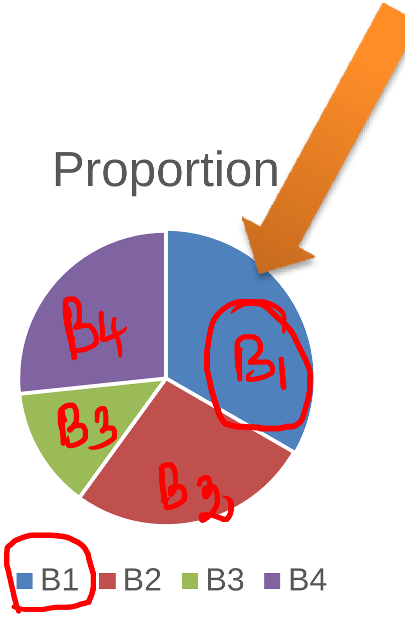
2	1	4	1	3

Genetic Algorithm

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



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



B4 B1

B1, B3

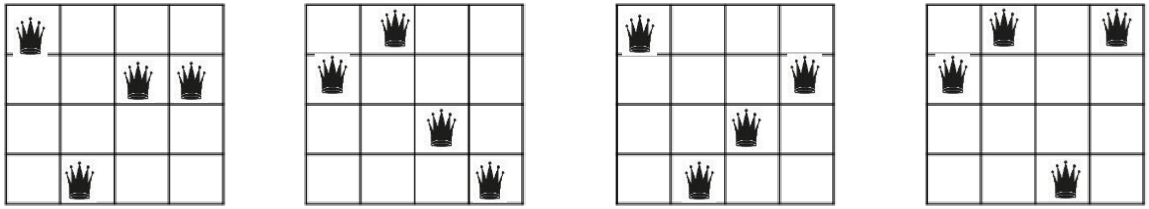
1

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

2 times

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



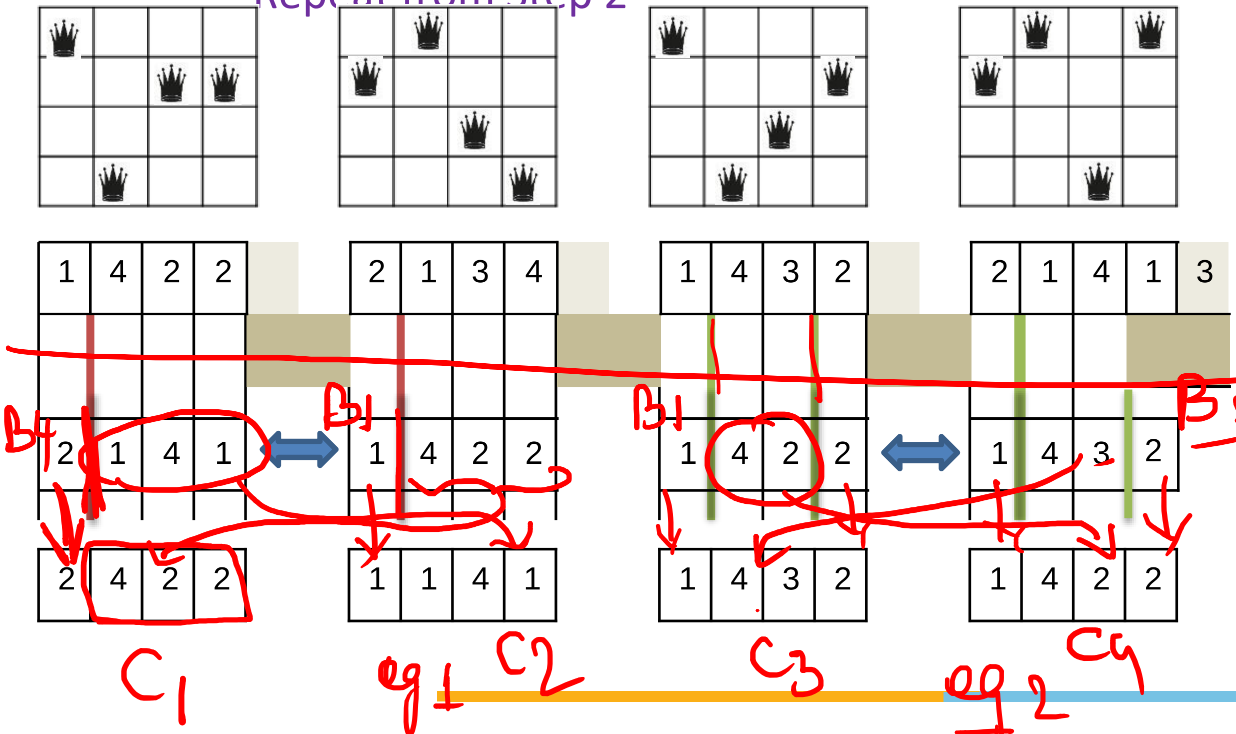
1	4	2	2	4	2	1	3	4	4	1	4	3	2	2	2	1	4	1	3
				0.31					0.31					0.15					0.23

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 Successor is
6. allowed to mutate
7. Repeat from Step 2



1 split

2 split

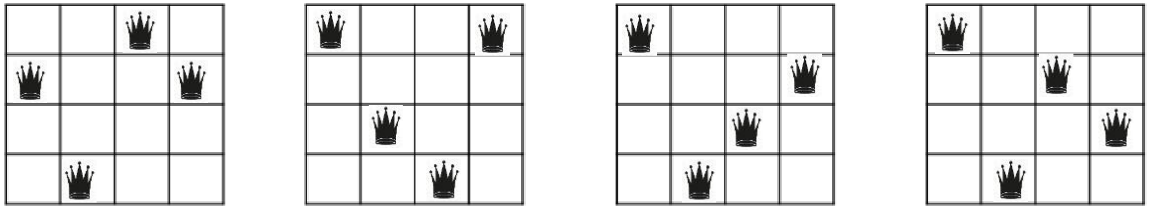
2|141 1|422

2422 1141

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 =
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 → Nm



2	4	2	2

C1

2	4	1	2

C1

1	1	4	1

C2

1	3	4	1

C2

1	4	3	2

C3

No

1	4	3	2

C.

1	4	2	2

C4

1	4	2	3

50% ✓

30%.

1st One val

2nd

3rd

4th

1st Iteration

Genetic Algorithm

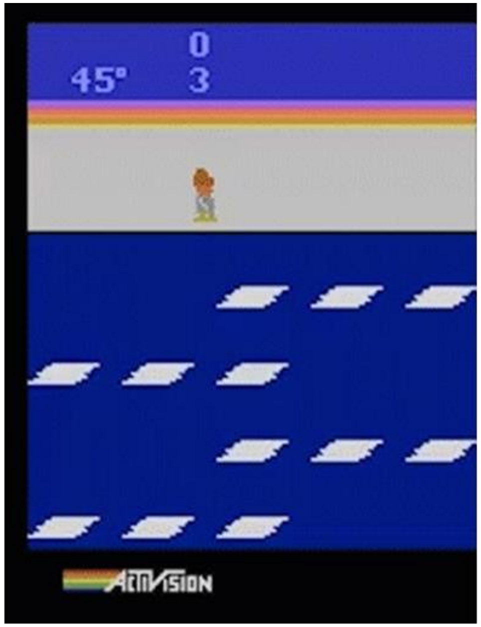
Techniques:

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 - Application in Games



Source Credit:

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

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