BITS Pilani

Pilani Campus

Artificial and Computational Intelligence

AIMLCLZG557

Contributors & Designers of document content : Cluster Course Faculty Team

M2 : : Problem Solving Agent using Search

Presented by
Indumathi V
indumathi.p@wilp.bits-pilani.ac.in

Disclaimer and Acknowledgement

- Few content for these slides may have been obtained from prescribed books and various other source on the Internet
- I hereby acknowledge all the contributors for their material and inputs and gratefully acknowledge people others who made their course materials freely available online.
- .I have provided source information wherever necessary
- This is not a full fledged reading materials. Students are requested to refer to the textbook w.r.t detailed content of the presentation deck that is expected to be shared over e-learning portal - taxilla.
- I have added and modified the content to suit the requirements of the class dynamics & live session's lecture delivery flow for presentation

- **Slide Source / Preparation / Review:**
- From BITS Pilani WILP**:** Prof.Raja vadhana, Prof. Indumathi, Prof.Sangeetha
- From BITS Oncampus & External : Mr.Santosh GSK

Course Plan

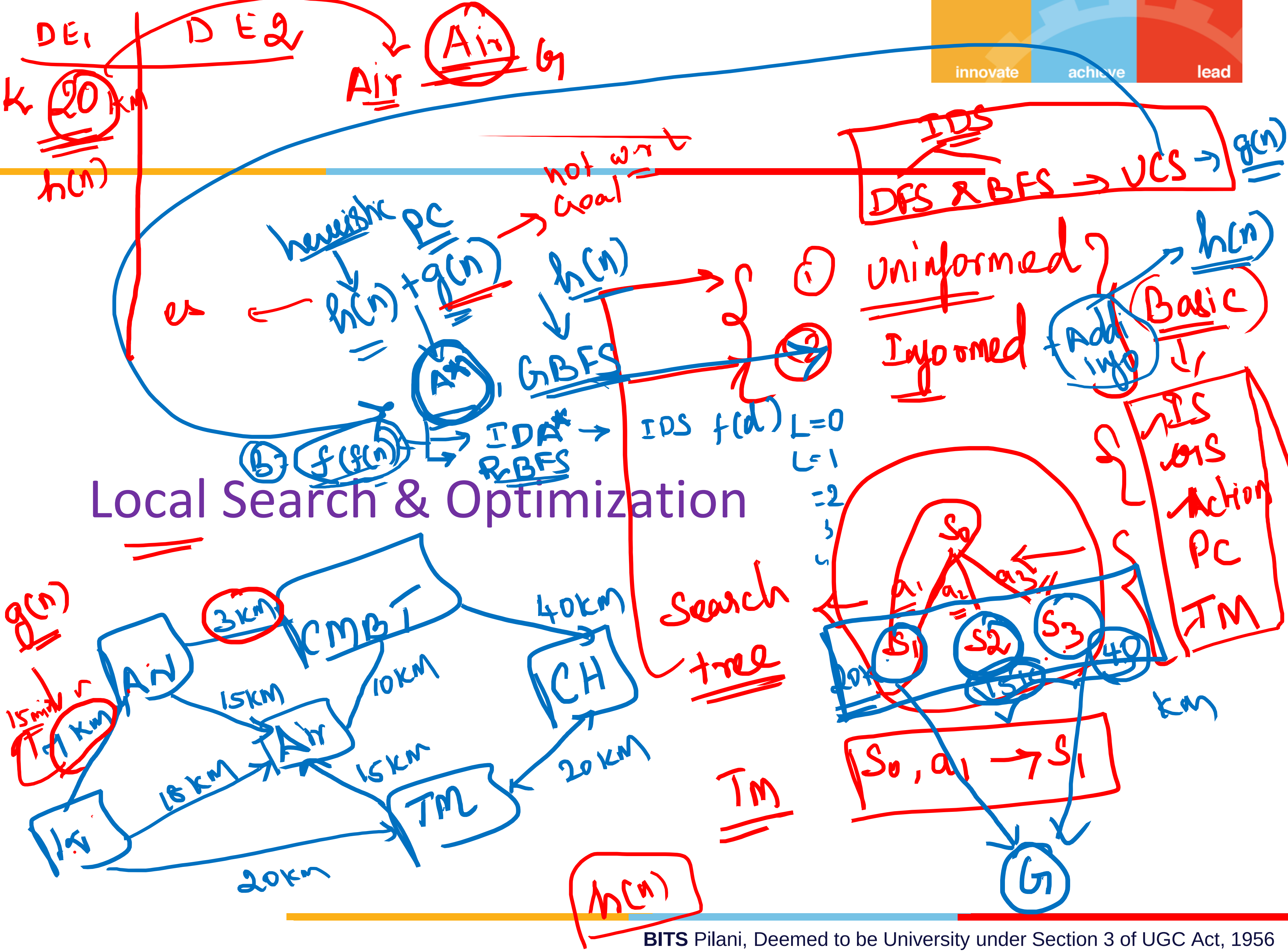
M1	Introduction to AI
M2	Problem Solving Agent using Search
M3	Game Playing
M4	Knowledge Representation using Logics
M5	Probabilistic Representation and Reasoning
M6	Reasoning over time, Reinforcement Learning
M7	Ethics in AI

Learning Objective

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

1. Design fitness function for a problem
2. Construct a search tree
3. 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)
4. Design and show Genetic Algorithm steps for a given problem

Local Search & Optimization



Envi

1 static

2 Fully obs → 1 partial info
↳ sensor
↳ Computation } capability

3 Det

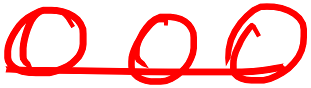
4 Discrete

5 Single

6

2 Path is irrelevant

3 Define Goal or (Expected goal)



Chess board →

← "Greedy choice" → '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 → h(n)

Scope : Local best Successor

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

1 Single Instance Based

- Hill Climbing → RRHC, SHC
- ~~Simulated Annealing~~
- Local Beam Search
- ~~Tabu Search~~

2 Multiple Instance Based

- Genetic Algorithm
- Particle Swarm Optimization
- Ant Colony Optimization

Local Search

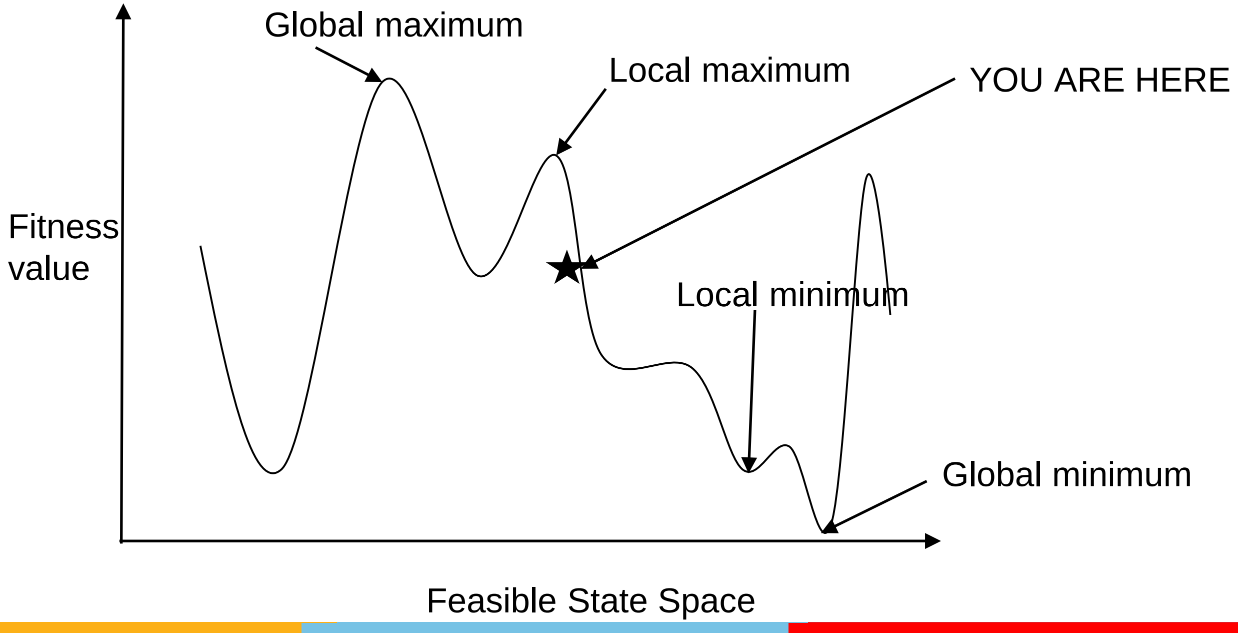
Terminology

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

Algorithms:

- Choice of Neighbor
- Looping Condition
- Termination Condition

2	5	3	2
♛	6	♛	♛
3	5	4	2
4	♛	4	2



Local Search Terminology

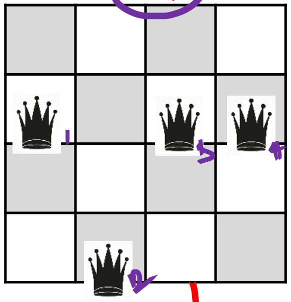
Inf un | Local
Empty | completely filled BC
PFBC
Empty

PEAS ← I/P

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

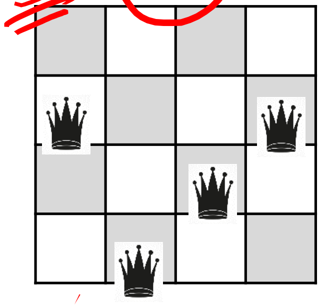
IS

S1

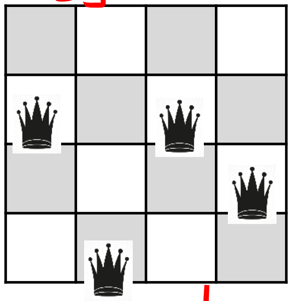


Q1 Q2
Q1 Q3
Q1 Q4
Q2 Q3
Q2 Q4

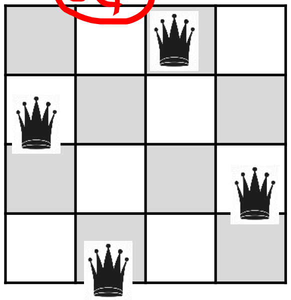
IS S2



S3



S4



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

Fitness Value: h

D1 h(n) = 4 | h(n) = 4 | h(n) = 2 | h(n) = 0 ✓

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

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

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

Local Search

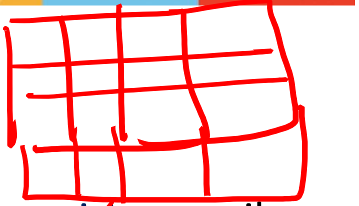
Inf un — Empty

Local — completely filled BC

PFBC

Empty

Terminology

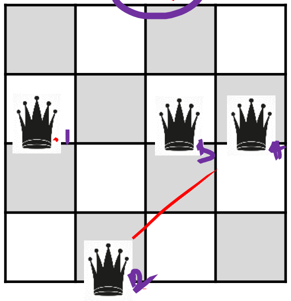


PEAS ← I/P

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

IS

S1



6 pairs

Q1 Q2 — NC

Q1 Q3 → C

Q1 Q4 — C

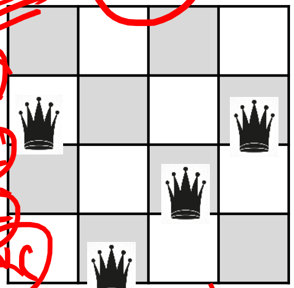
Q2 Q3 — NC

Q2 Q4 C

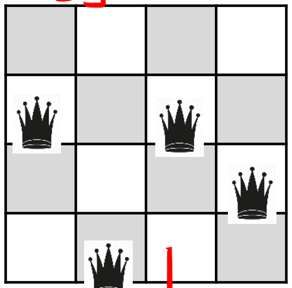
Q3 Q4 → C

IS

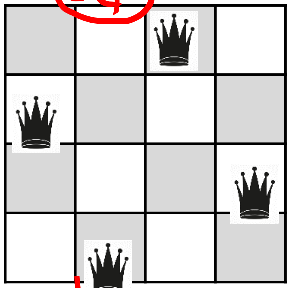
S2



S3



S4



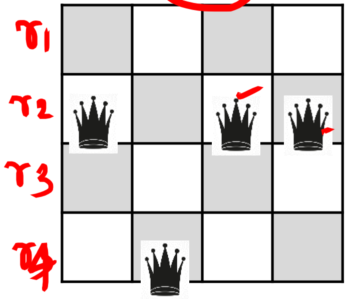
Feasible State/Solution	**Neighboring States**		**Optimal Solution**
Fitness Value:			
h(n) = 4	h(n) = 4	h(n) = 2	h(n) = 0
h(n) = No.of.Conflicting **pairs** of queens			
h(n) = 2	h(n) = 2	h(n) = 4	h(n) = 6
h(n) = No.of.**Non**-Conflicting **pairs** of queens.			

Terminology

C_1 C_2 C_3 C_4

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

S_0



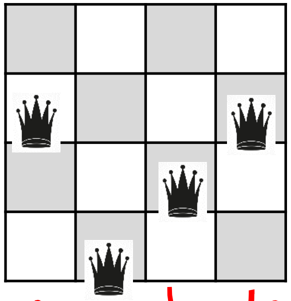
vector

col

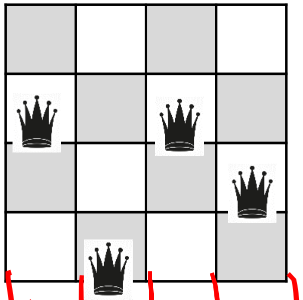
IS

V1	V2	V3	V4
2	4	2	2

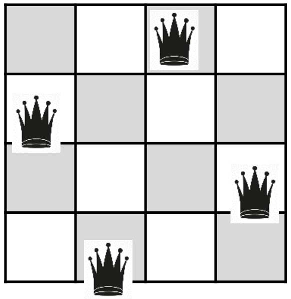
S_1



V1	V2	V3	V4
2	4	3	2



V1	V2	V3	V4
2	4	2	3



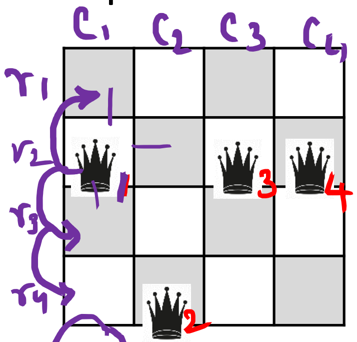
2	4	1	3

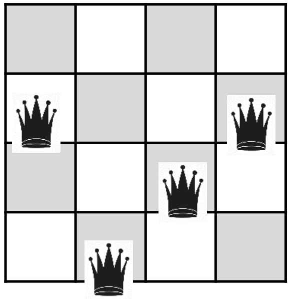
Local Search

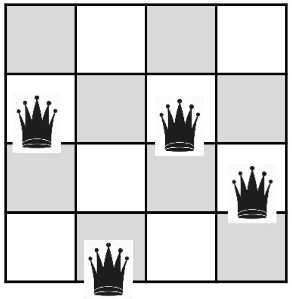
V1 → position of res Q on these rows
Q in the 1st col positioned at row 2

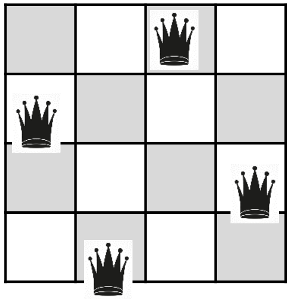
Terminology

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







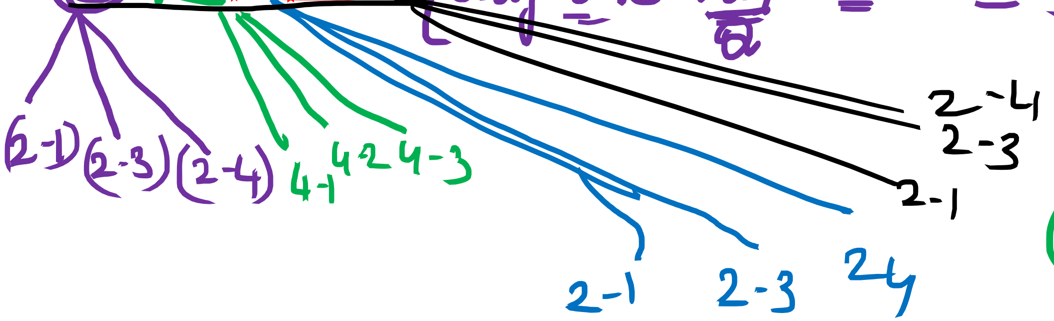


V1	V2	V3	V4
2	4	2	2

S0

LS change [only one Var at a time]

Q



9

1 NN

1 Neighborhood
region expansion

12 Succes

2 NN

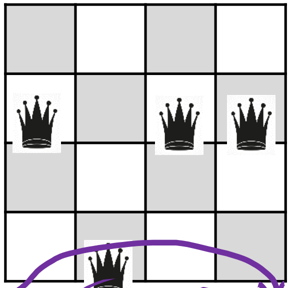
Local Search

3 NN → ↑ed

4 NN → ↑cd

Terminology

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



(1,3,4) (1,2,3)

2NN

[2,1,2,2]

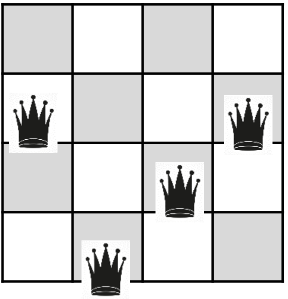
9 9

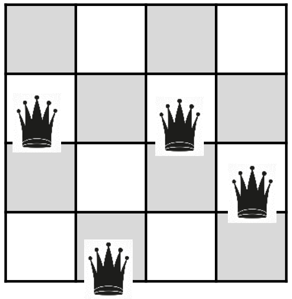
1 1
1 2
1 3

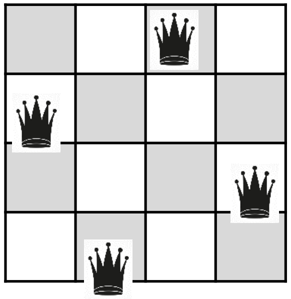
3 1
3 2
3 3

4 1
4 2
4 3

9







$v_1 v_2 \to 9$

$v_1 v_3 \to 9$

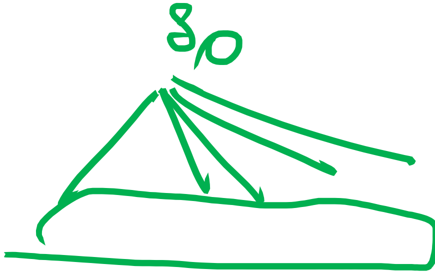
$v_1 v_4 \to 9$

$v_2 v_3 \to 9$

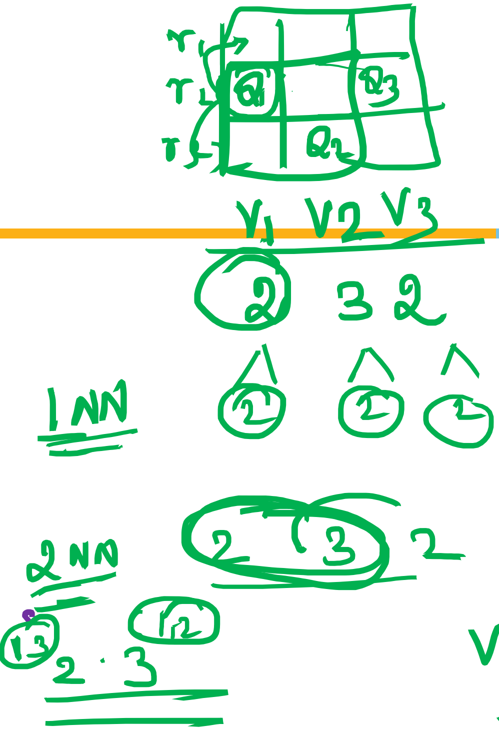
$v_2 v_4 \to 9$

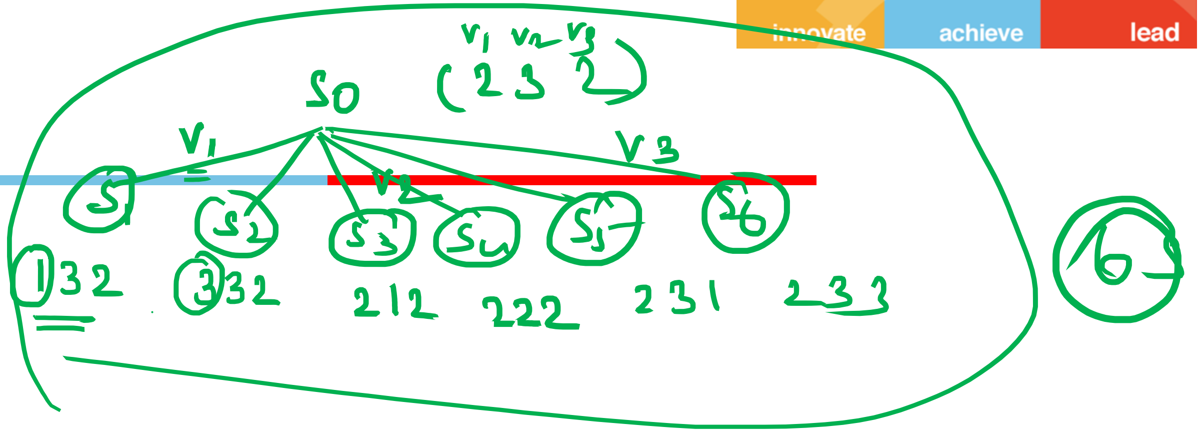
$v_3 v_4 \to 9$

6 x 9



54





Hill Climbing

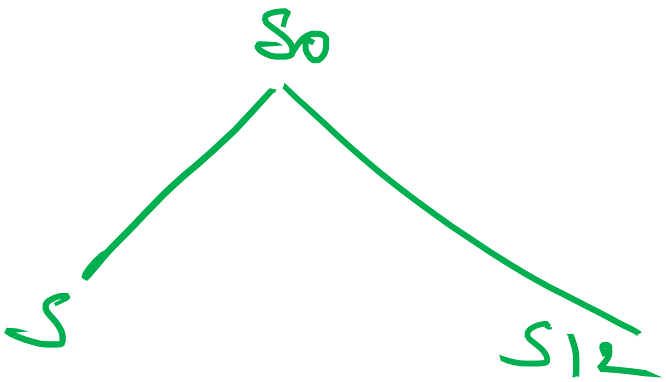
$V_1 V_2 \rightarrow 4$

$V_1 V_3 \rightarrow 4 \Rightarrow \underline{12/8}$

$V_2 V_3 \rightarrow 4$

1, 1
1, 2
3, 1
3, 2

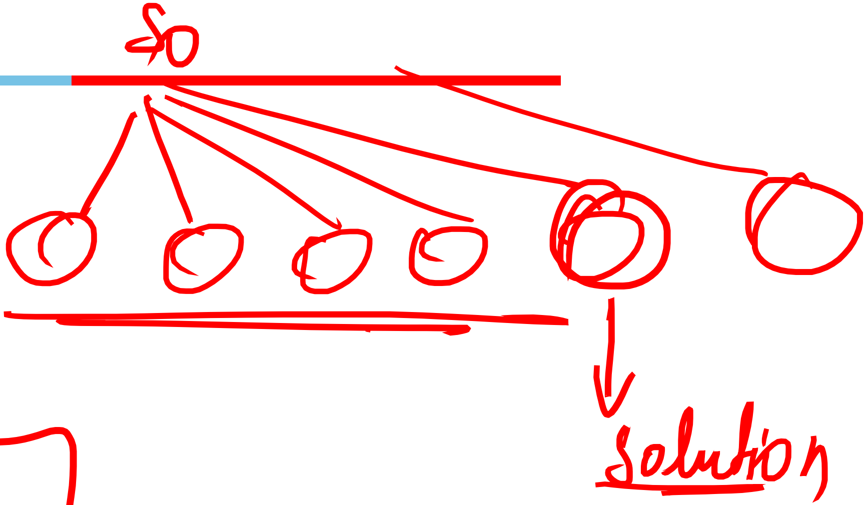
2 3 2



(1,3) (1,2) (1,3)

2 3 2

1 1 1

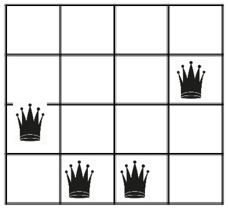


1 NN

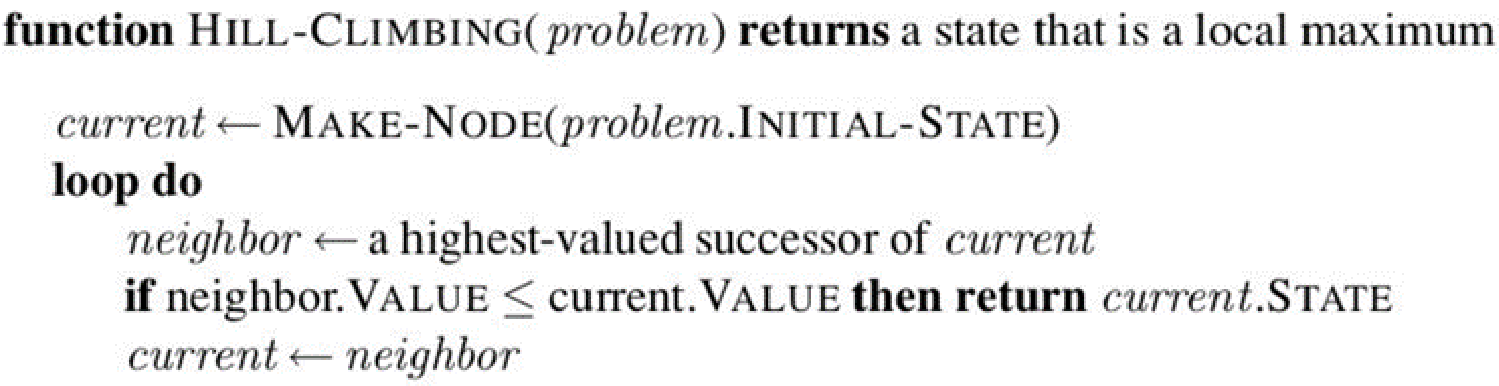
Hill Climbing

Hill Climbing

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



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

1. ✓ Select a random state IS
2. Evaluate the fitness scores for all the successors of the state

Global max

↓

6 pairs

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

4

Q1-Q2 – NC

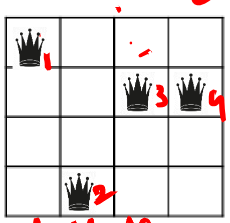
Q1-Q3 – NC

Q1-Q4 – NC

Q2-Q3 – NC

Q2-Q4 – C

Q3-Q4 – C



v1 v2 v3 v4

1	4	2	2	4

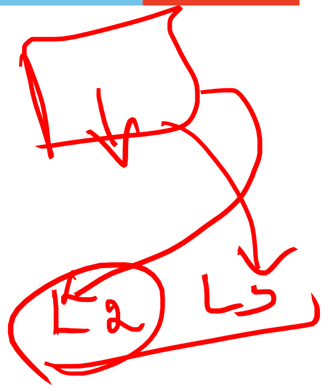
local max

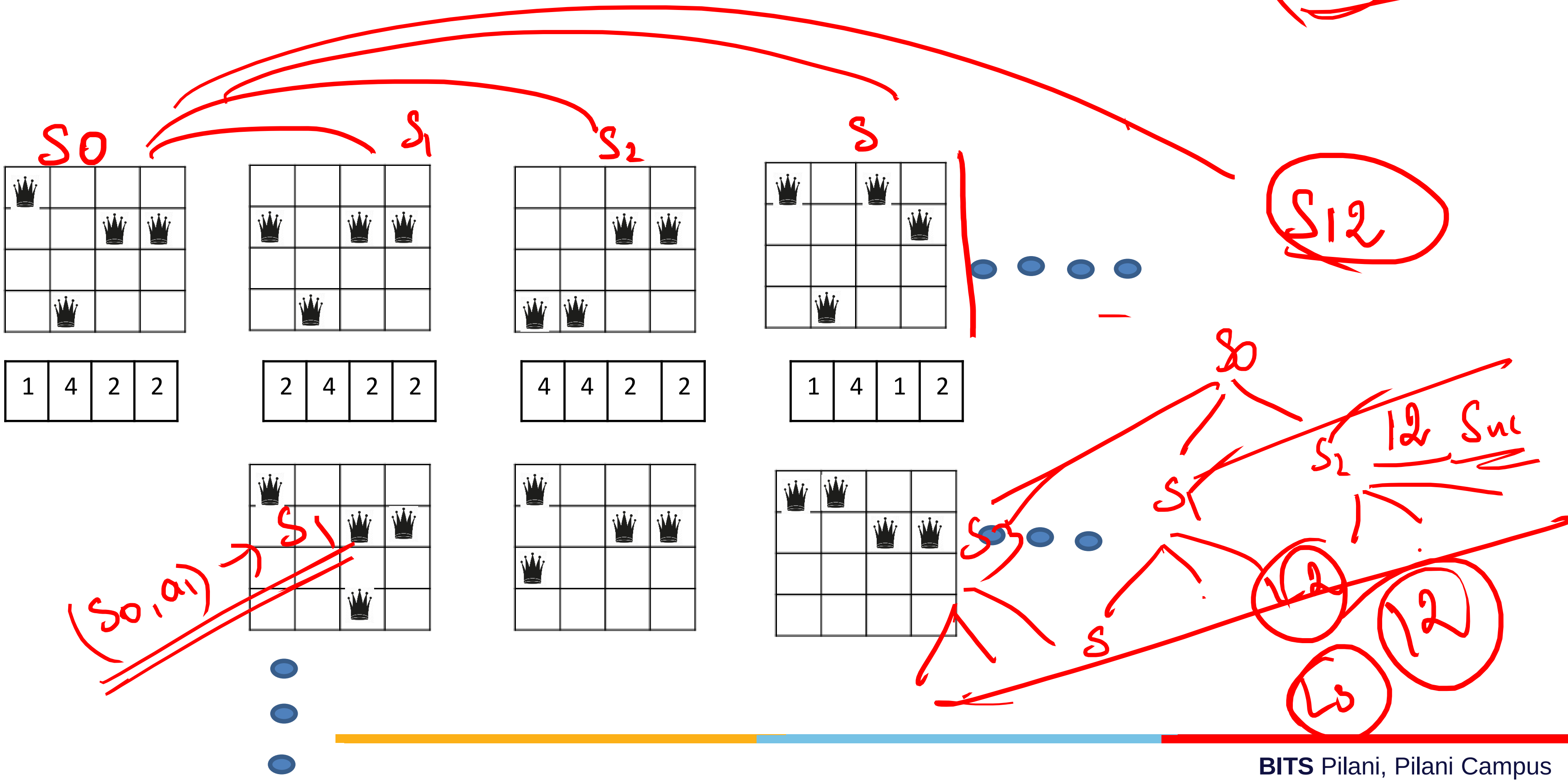
4 < 6

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

Hill Climbing

1. Select a random state INN
2. Evaluate the fitness scores for all the successors of the state





1	4	2	2

2	4	2	2

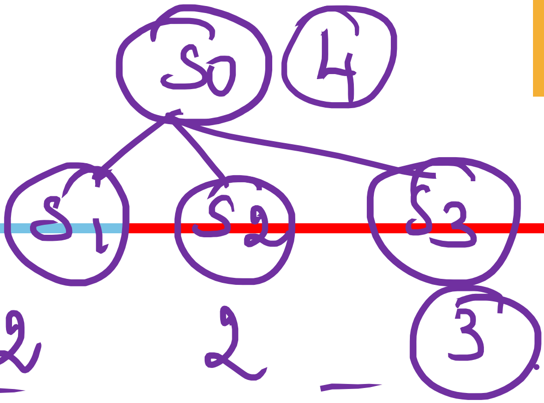
4	4	2	2

1	4	1	2

Hill Climbing

I1

4 ≠ 2
4 ≠ 2
4 ≠ 3

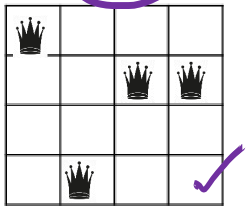


4 ≠ 3

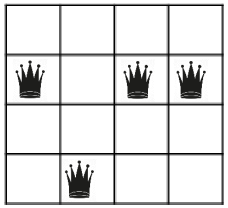
STOP

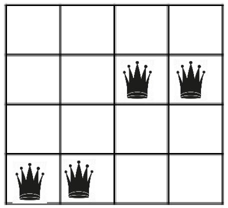
TC 1

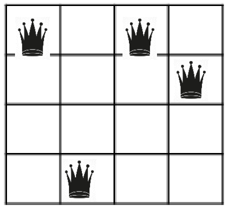
✓ S0



S1







● ● ● ●

1	4	2	2	4 ✓

2	4	2	2	2 ✓

4	4	2	2	2 ✓

1	4	1	2	3 ✓

NC pair ↑

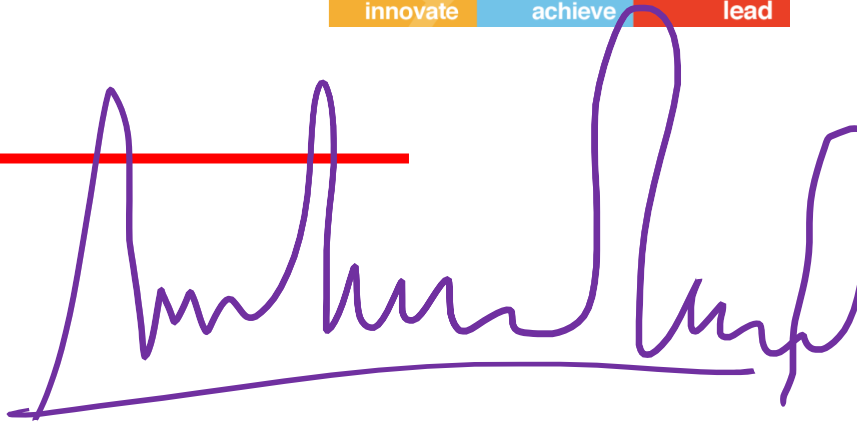
Local Maxima → Random Restart

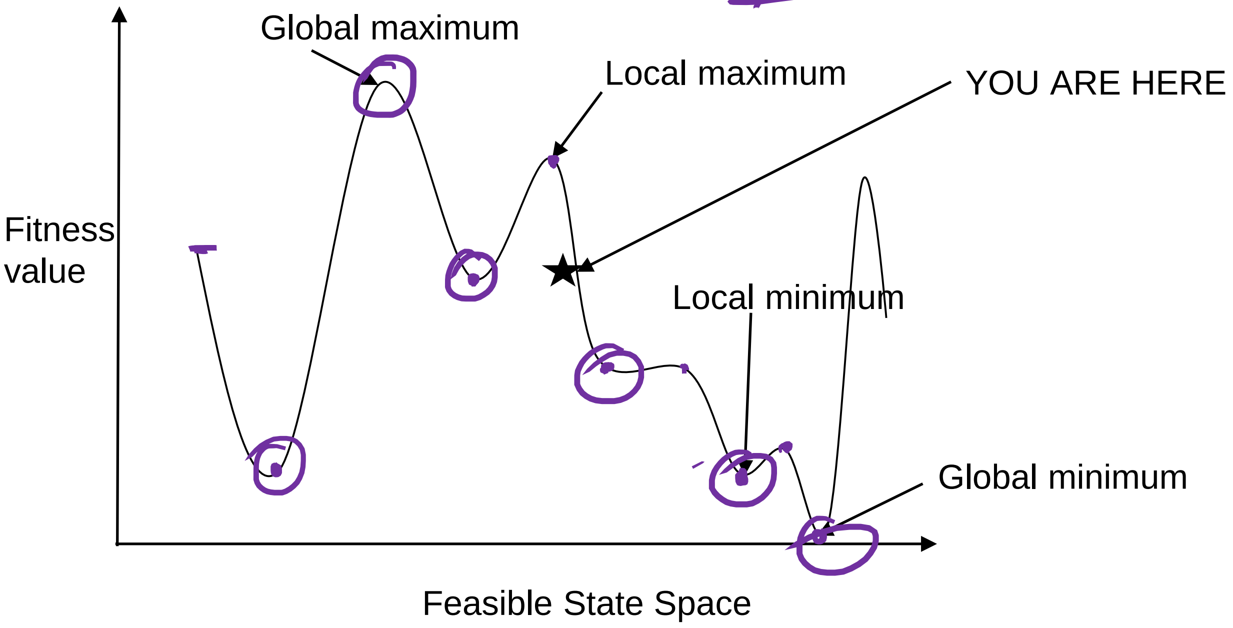
Global maxima → 6

6 pairs NC

Hill Climbing

Countable R, RHA





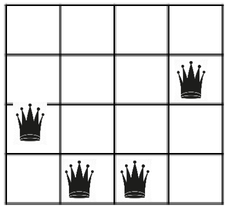
Hill Climbing

Random Restart

another

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

I2



v1 v2 v3 v4

3	4	4	2	3

NC

```
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

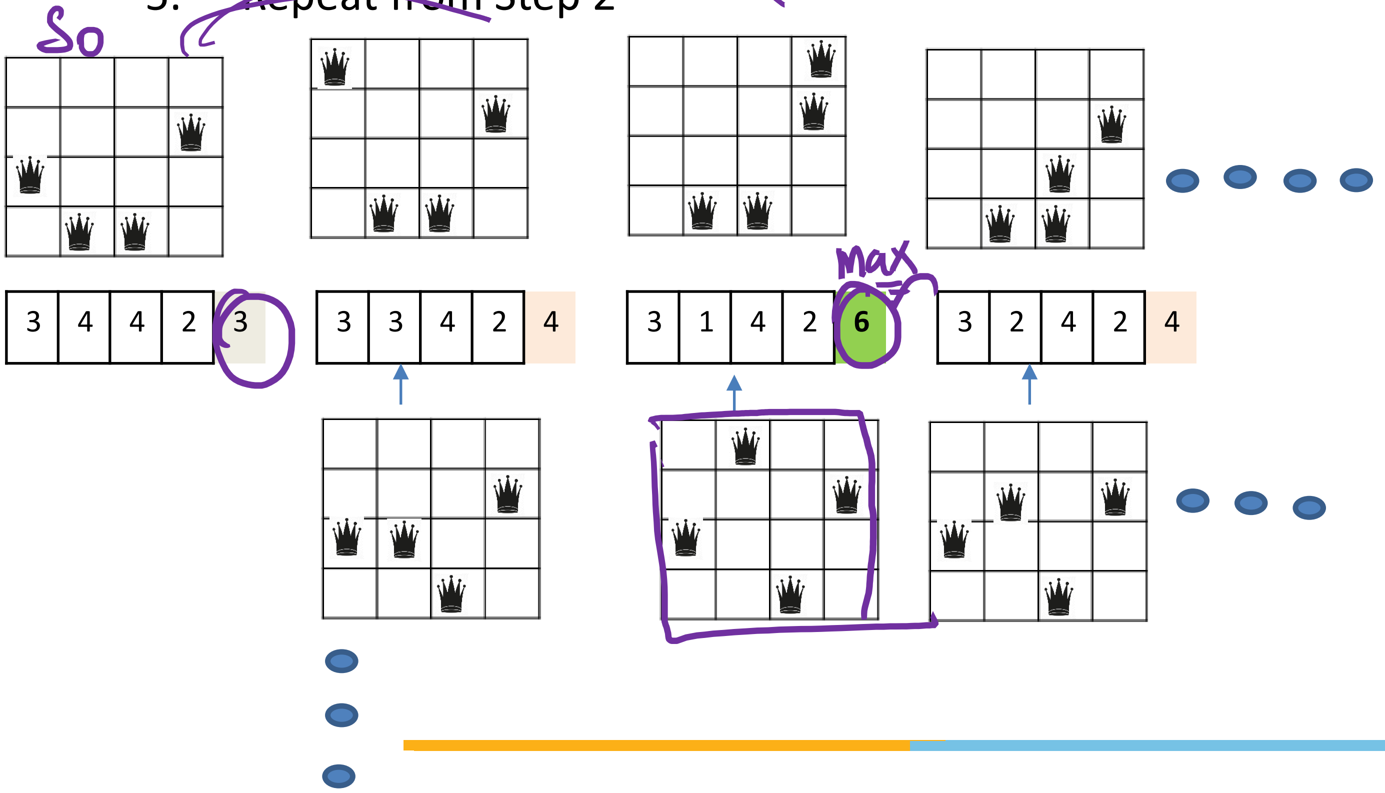
Random Restart

TC1

4 ≮ 2
2
≤ 3 STOPS

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

TC2 GM/GMin



3	4	4	2	3

3	3	4	2	4

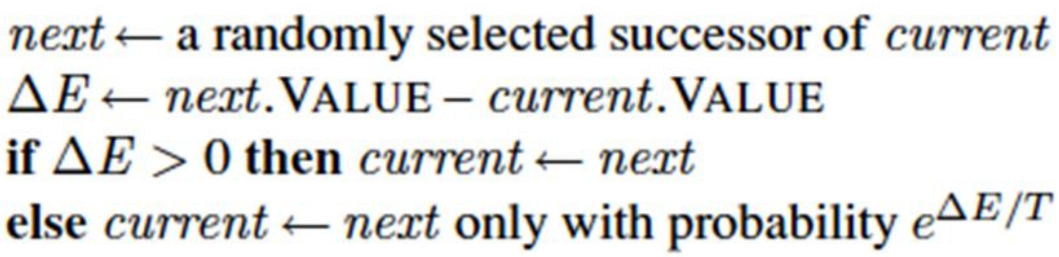
3	1	4	2	6

3	2	4	2	4

12

TC3 K=50

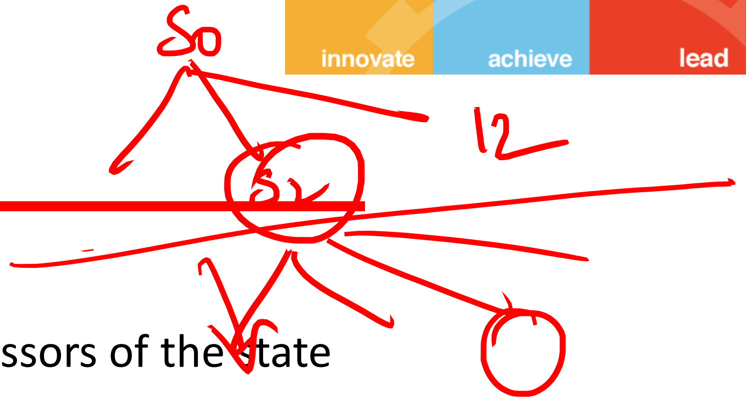
Stochastic Hill Climbing



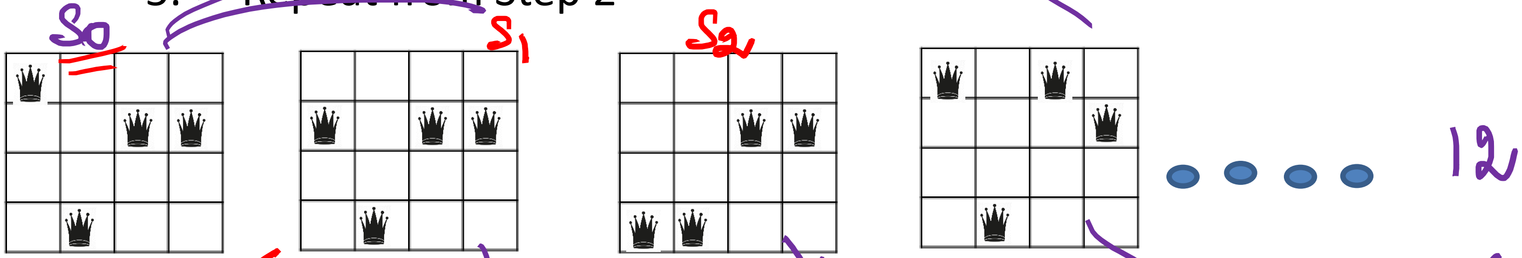
```
$next \leftarrow$ a randomly selected successor of $current$
$\Delta E \leftarrow next.\text{VALUE} - current.\text{VALUE}$
if $\Delta E > 0$ then $current \leftarrow next$
else $current \leftarrow next$ only with probability $e^{\Delta E/T}$
```

f(n) -h(n) NC pairs

Stochastic Hill Climbing 4Q →



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	3	3	2	0.42
1	4	1	2	3	0.33

f(n)	fa	Prob	
1	2	2/12	0.16
2	5	5/12	0.42
3	4	4/12	0.33
4	1	1/12	0.08

h(n) $\boxed{T}$ → Prob

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

6 6

6 Questions

4 to 5 Question

LOS

4

Gen

3 ACO

LOM

2 Search

a) ST

b) PC + h(n)

c) UCS, IDFS, DFS
A*, IDA*

Any Alg

1 S1

↳ PEAS

Theory ↳ TE 5 dim + justi

↳ PSA con

IS, GS, TM, PC

A

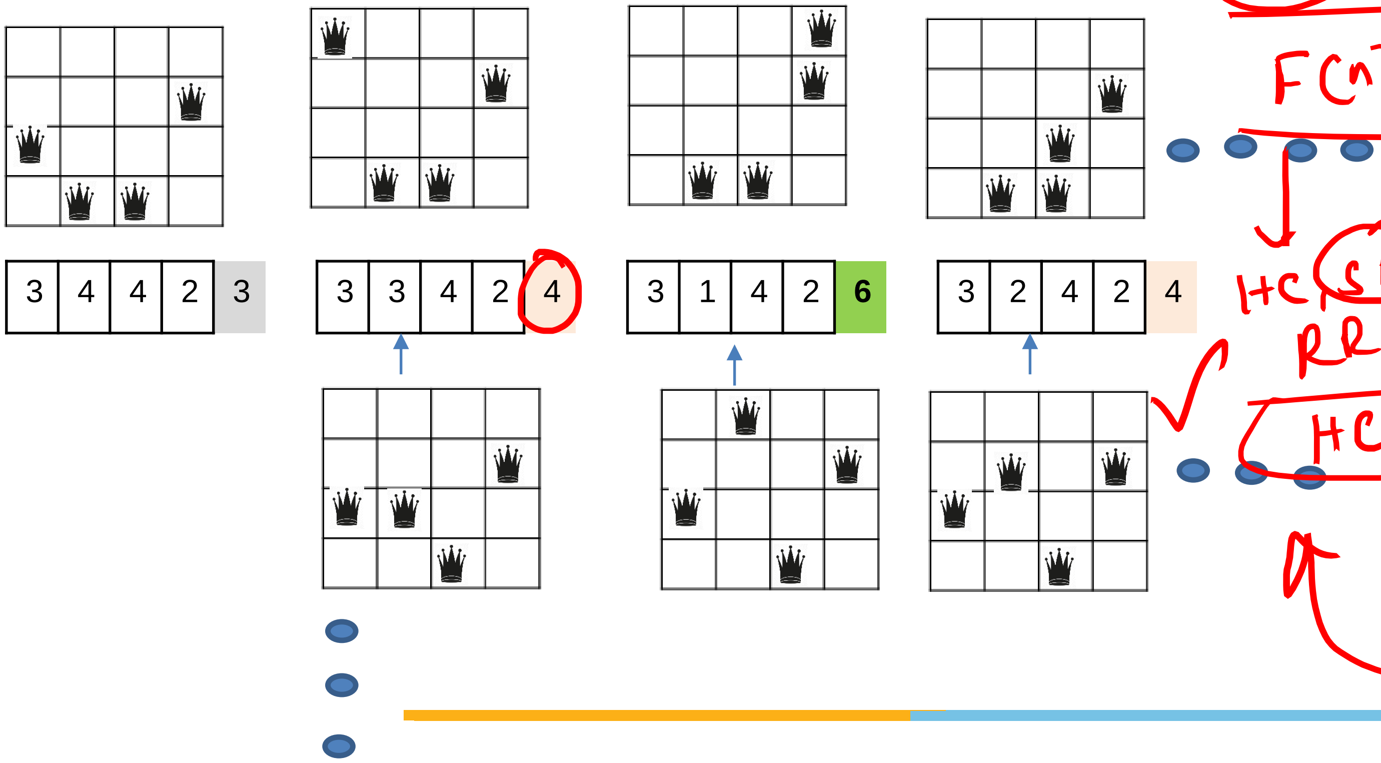
Local Beam Search

5 HD

Beam Search

One 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

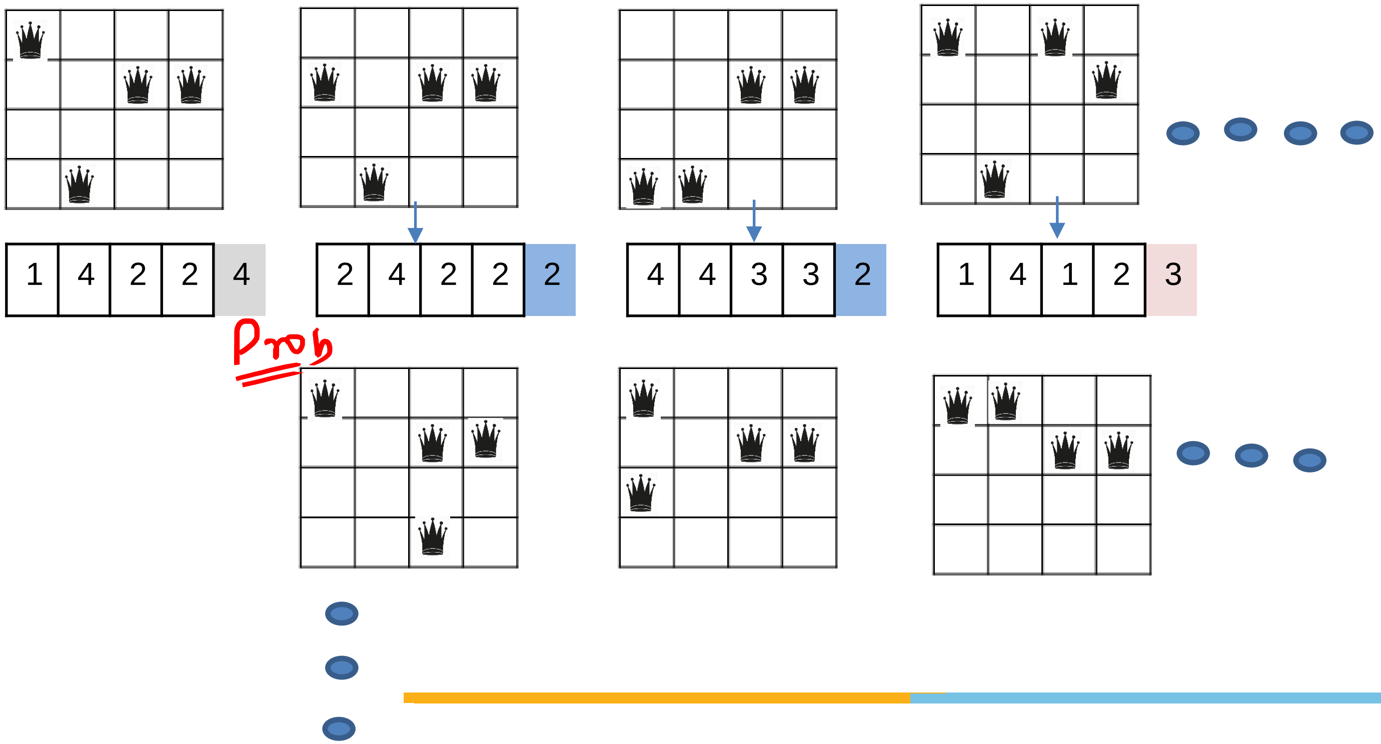


IS1	IS2	IS3
F(n)	F(n)	F(n)
HC (SHC) RR	~	-
HC	SHC	RRS

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



1 4 2 2 4

2 4 2 2 2

4 4 3 3 2

1 4 1 2 3

Prob

Required Reading: AIMA - Chapter # 4.1, #4.2

Thank You for all your Attention

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