



Université Constantine 2
جامعة قسنطينة 2



Foundation of Artificial Intelligence

TP 03

Dr. NECIBI Khaled
Faculté des nouvelles technologies
Khaled.necibi@univ-constantine2.dz



Université Constantine 2
جامعة قسنطينة 2



Foundation of Artificial Intelligence

- TP Résolution de problèmes par recherche locale -

Dr. NECIBI Khaled

Faculté des nouvelles technologies

Khaled.necibi@univ-constantine2.dz

Etudiants concernés

Faculté/Institut	Département	Niveau	Spécialité
Nouvelles technologies	IFA	Master 1	SDIA

Résolution de problème par recherche locale

- Exercice 01 - Hill Climbing Steepest Ascent
- On considère un espace de recherche représenté par la matrice suivante (aléatoirement remplie)

12	15	10	7	14	16
13	18	13	20	5	23
17	9	23	24	19	2
7	14	16	12	34	10
20	5	23	13	18	13
24	19	2	17	9	23

Résolution de problème par recherche locale

● Exercice 01

- On veut appliquer l'algorithme Hill Climbing vu pendant le cours (Steepest Ascent)
- Ecrire un programme python qui affiche l'évolution des nœuds visités dans le sens de la montée la plus rapide

● Propriétés de l'algorithme Hill Climbing :

- La recherche se fait localement
- Suit une approche gloutonne
- Il n'y a pas de backtracking

● Type de Hill Climbing

- Hill Climbing simple
- Hill Climbing de type steepest ascent
- Hill Climbing stochastique
- Hill Climbing de type restart aléatoire

Résolution de problème par recherche locale

- Exercice 01 : Solution
- Trouvez le prochain voisin

HillClimbingSA.py > ...

```
1  import numpy as np
2
3
4  def find_neighbours(state, landscape):
5      neighbours = []
6      dim = landscape.shape
7
8      # left neighbour
9      if state[0] != 0:
10         neighbours.append((state[0] - 1, state[1]))
11
12     # right neighbour
13     if state[0] != dim[0] - 1:
14         neighbours.append((state[0] + 1, state[1]))
15
16     # top neighbour
17     if state[1] != 0:
18         neighbours.append((state[0], state[1] - 1))
19
20     # bottom neighbour
21     if state[1] != dim[1] - 1:
22         neighbours.append((state[0], state[1] + 1))
23
```


HillClimbingSA.py > hill_climb

```
24     # top left
25     if state[0] != 0 and state[1] != 0:
26         neighbours.append((state[0] - 1, state[1] - 1))
27
28     # bottom left
29     if state[0] != 0 and state[1] != dim[1] - 1:
30         neighbours.append((state[0] - 1, state[1] + 1))
31
32     # top right
33     if state[0] != dim[0] - 1 and state[1] != 0:
34         neighbours.append((state[0] + 1, state[1] - 1))
35
36     # bottom right
37     if state[0] != dim[0] - 1 and state[1] != dim[1] - 1:
38         neighbours.append((state[0] + 1, state[1] + 1))
39
40     return neighbours
41
```

Résolution de problème par recherche locale

- Exercice 01 : Solution

- Trouvez le prochain voisin avec la meilleur valeur (la plus grande)

HillClimbingSA.py >  __main__

```
43 # Current optimization objective: local/global maximum
44 def hill_climb(curr_state, landscape):
45     neighbours = find_neighbours(curr_state, landscape)
46     bool
47     ascended = False
48     next_state = curr_state
49     for neighbour in neighbours: #Find the neighbour with the greatest value
50         if landscape[neighbour[0]][neighbour[1]] > landscape[next_state[0]][next_state[1]]:
51             next_state = neighbour
52             ascended = True
53
54     return ascended, next_state
55
```

Résolution de problème par recherche locale

- Exercice 01 : Solution
- Définir le programme principal

```
HillClimbingSA.py > _main_  
56  
57 def __main__():  
58     landscape = np.random.randint(1, high=50, size=(10, 10))  
59     print(landscape)  
60     start_state = (3, 6) # matrix index coordinates  
61     current_state = start_state  
62     count = 1  
63     ascending = True  
64     while ascending:  
65         print("\nStep #", count)  
66         print("Current state coordinates: ", current_state)  
67         print("Current state value: ", landscape[current_state[0]][current_state[1]])  
68         count += 1  
69         ascending, current_state = hill_climb(current_state, landscape)  
70  
71     print("\nStep #", count)  
72     print("Optimization objective reached.")  
73     print("Final state coordinates: ", current_state)  
74     print("Final state value: ", landscape[current_state[0]][current_state[1]])  
75  
76  
77     __main__()  
78
```


Résolution de problème par recherche locale

● Exercice 01 : Solution

```
PROBLEMS 10 OUTPUT DEBUG CONSOLE TERMINAL PORTS

PS C:\Users\KHALED\Strategies> & C:/Users/KHALED/AppData/Local/Programs/Python/Python39/python.exe c:/Users/KHALED/Strategies/Hi
llClimbingSA.py
[[33 21 41 25 11 36 25 8 43 44]
 [37 1 29 31 24 27 12 16 28 26]
 [14 25 15 23 32 10 28 48 20 29]
 [41 47 41 16 25 16 45 19 39 7]
 [9 17 20 16 12 10 47 23 1 42]
 [39 18 29 17 41 37 26 37 34 7]
 [18 22 43 43 22 16 30 2 48 10]
 [11 12 47 12 41 43 24 28 36 4]
 [22 25 16 32 26 14 26 41 47 45]
 [7 37 12 23 13 5 49 34 25 34]]

Step # 1
Current state coordinates: (3, 6)
Current state value: 45

Step # 2
Current state coordinates: (2, 7)
Current state value: 48

Step # 3
Optimization objective reached.
Final state coordinates: (2, 7)
Final state value: 48
PS C:\Users\KHALED\Strategies>
```