



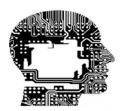
# Foundation of Artificial Intelligence TP 03

Dr. NECIBI Khaled

Faculté des nouvelles technologies

Khaled.necibi@univ-constantine2.dz





## 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

Université Constantine 2 2023/2024. Semestre

- Exercice 01 Hill Climibing 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

Université Constantine 2 © Dr. NECIBI Khaled

## • Exercice 01

- On veut appliquer l'algorithme Hill Climbing vu pendant le cours (Steapest 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

4

- Exercice 01: Solution
  - Trouvez le prochain voisin

```
HillClimbingSA.py > ...
      import numpy as np
      def find_neighbours(state, landscape):
          neighbours = []
          dim = landscape.shape
          # left neighbour
          if state[0] != 0:
              neighbours.append((state[0] - 1, state[1]))
11
12
          # right neighbour
          if state[0] != dim[0] - 1:
13
              neighbours.append((state[0] + 1, state[1]))
          # top neighbour
          if state[1] != 0:
              neighbours.append((state[0], state[1] - 1))
          # bottom neighbour
          if state[1] != dim[1] - 1:
22
              neighbours.append((state[0], state[1] + 1))
```

```
# HillClimbingSA.py >  hill_climb
          # top left
          if state[0] != 0 and state[1] != 0:
              neighbours.append((state[0] - 1, state[1] - 1))
          # bottom left
          if state[0] != 0 and state[1] != dim[1] - 1:
              neighbours.append((state[0] - 1, state[1] + 1))
          # top right
          if state[0] != dim[0] - 1 and state[1] != 0:
              neighbours.append((state[0] + 1, state[1] - 1))
          # bottom right
          if state[0] != dim[0] - 1 and state[1] != dim[1] - 1:
              neighbours.append((state[0] + 1, state[1] + 1))
          return neighbours
```

- Exercice 01: Solution
  - Trouvez le prochain voisin avec la meilleur valeur (la plus grande)

```
HillClimbingSA.py > 
  _main_
      # Current optimization objective: local/global maximum
43
      def hill climb(curr state, landscape):
44
          neighbours = find neighbours(curr state, landscape)
45
          boo1
46
          ascended = False
47
          next_state = curr_state
          for neighbour in neighbours: #Find the neighbour with the greatest value
49
              if landscape[neighbour[0]][neighbour[1]] > landscape[next state[0]][next state[1]]:
51
                  next state = neighbour
52
                  ascended = True
54
          return ascended, next state
```

Université Constantine 2 © Dr. NECIBI Khaled 6

- Exercice 01: Solution
  - Définir le programme principal

```
HillClimbingSA.py >  main_
57
      def main ():
          landscape = np.random.randint(1, high=50, size=(10, 10))
          print(landscape)
          start state = (3, 6) # matrix index coordinates
          current state = start state
62
          count = 1
          ascending = True
          while ascending:
              print("\nStep #", count)
              print("Current state coordinates: ", current_state)
              print("Current state value: ", landscape[current state[0]][current state[1]])
              count += 1
              ascending, current state = hill climb(current state, landscape)
70
          print("\nStep #", count)
71
          print("Optimization objective reached.")
72
          print("Final state coordinates: ", current state)
          print("Final state value: ", landscape[current state[0]][current state[1]])
76
       main ()
```

Université Constantine 2 © Dr. NECIBI Khaled

• 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
 11ClimbingSA.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>
```

Université Constantine 2 © Dr. NEOBI Khaled

8