Université Polytechnique HAUTS-DE-FRANCE

ALGORITHME DE RÉSOLUTION DU PROBLÈME DU SAC À DOS MULTIDIMENSIONNEL

Ali Abakar Oumar

Adediran Flore

Aw Amadou Tidiane

Mahaman Bachir Attouman Ousmane

Yves Fernand Nga Ngono

Enseignant : Christophe WILBAUT

PRÉSENTATION DU PROBLÈME

- PROBLÈME DU SAC À DOS MULTIPLE

$$(\text{SADM}) \begin{cases} \max & \sum_{j=1}^{n} c_j x_j & \\ \text{s.c. :} & \sum_{j=1}^{n} a_{ij} x_j \leq b_i, & i \in M = \{1, \ldots, m\} \\ & x_j \in \{0, 1\} & j \in N = \{1, \ldots, n\} \end{cases}$$

```
for j in range(m):
    if sum( x*weight[j] ) >constraints[j]:
        return 0

return 1
```

ALGORITHME DE RÉSOLUTION :

- LA RECHERCHE TABOU

Algorithme de base

- Engendrer une configuration initiale X

-T = [ ]

Répéter

- Calculer m = le meilleur mouvement parmi les mouvements non tabous .

-X = X (+) m

- Mettre T à jour

Jusqu'à condition fin

Retourner X

CALCUL DU MOUVEMENT

CONTRAINTES DE SUBSTITUTION

1. Recalcule de nouvelles contraintes

```
for i in range(m):
    b[i] = constraints[i] - np.sum(weight[i]*x)
```

2 . Calcul de coefficients

```
if b[i] > 0 :
    w[i] = (1/b[i])
else:
    w[i] = (2 + abs(b[i]))
```

```
for i in range(m):
    s[i] = w[i]*weight[i]
```

3. Contrainte de substitution

```
new_constraints[i]= w[i] * constraints[i]
```

```
s = np.sum(s,axis=0)
s0 = sum(new_constraints)
```

CALCUL DU MOUVEMENT

CHOIX DU MOUVEMENT

```
values_over_weight = C/S

mov_index = values_over_weight.argsort()
mov_index = values_over_weight.argsort()[::-1]

for tabu in tabu_list:
        if tabu in mov_index:
            mov_index.remove(tabu)
  i=0
  while x[ mov_index[i] ] == 0:
      i+=1

  x[mov_index[i]] = 1
  x[mov_index[i]] = 0
```

LISTE TABOU

```
d_tabu_list.append(latest_move)
```

```
c_tabu_list.append(latest_move)
```

```
if iteration%7==0:
    c_tabu_list=[]
    d_tabu_list=[]
```

EVÈNEMENT CRITIQUE

PHASE CONSTRUCTIVE

ZONE INFAISABLE

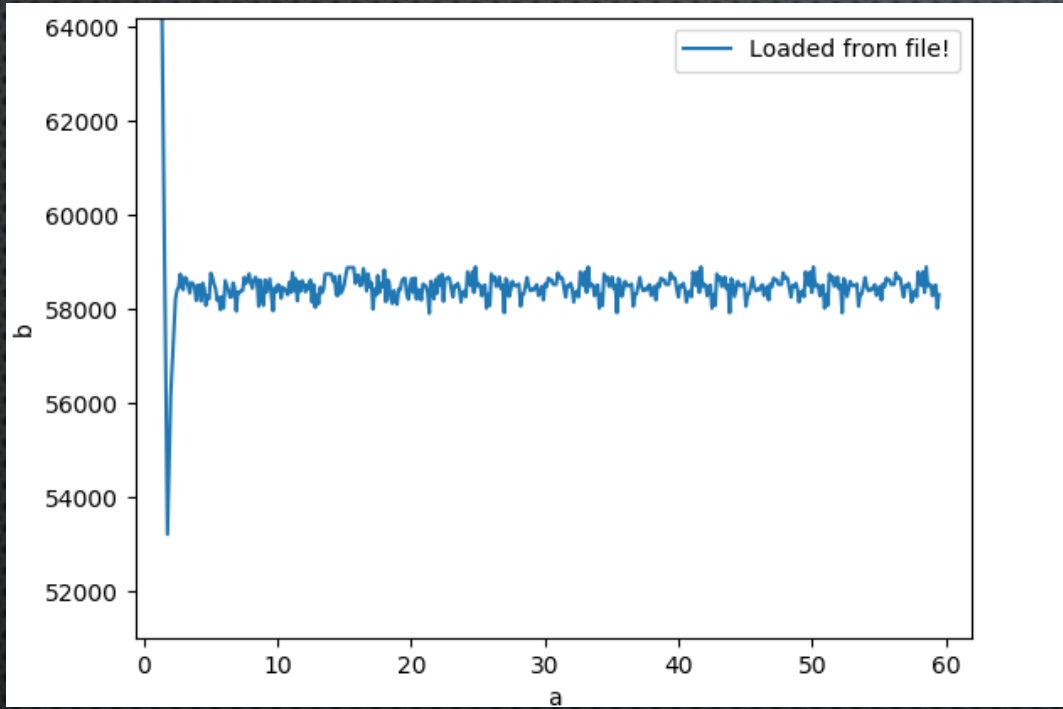
PHASE DESTRUCTIVE

ZONE FAISABLE

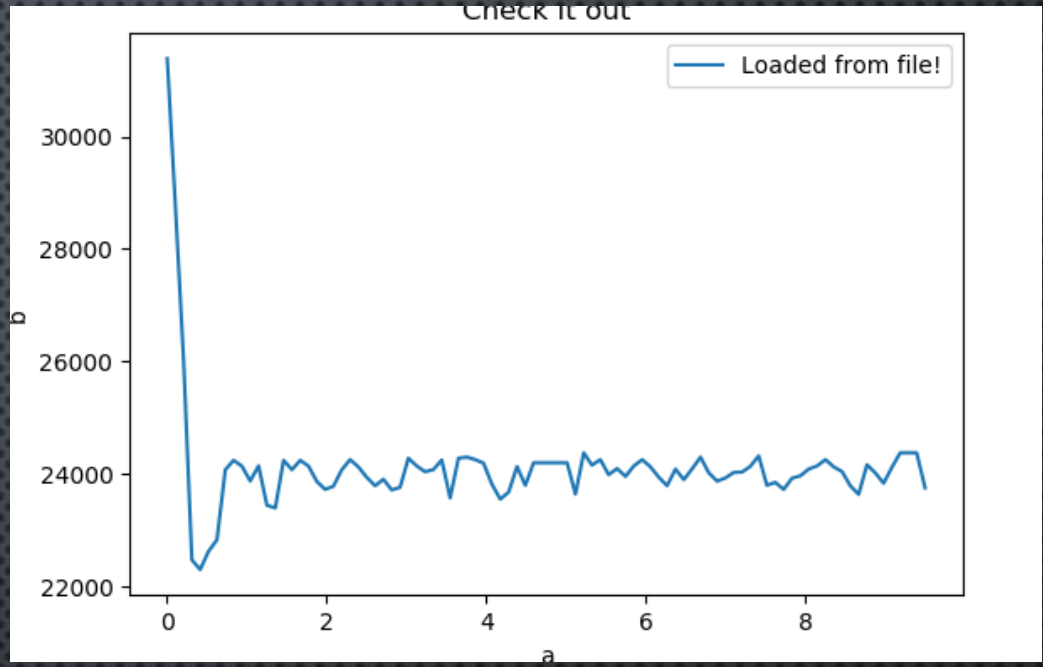
ALGORITHME GÉNÉRAL

```
while((time.time() -start_time) < exec_time-0.5):
    iteration+=1

    for _ in range(5):
        move_choice=best_move()
        latest_move=remove(move_choice,c_tabu_list)

    feasible = real_evaluate()
    d_tabu_list.append(latest_move)

    while feasible  :
        move_choice = best_move()
        latest_move = put(move_choice,d_tabu_list)
        feasible=real_evaluate()

    c_tabu_list.append(latest_move)
    critical_constructive_proc()

    if iteration%7==0:
        c_tabu_list=[]
        d_tabu_list=[]
```

RÉSULTATS



250M5_1.DAT



100M5_1.DAT