LPO2

May 3, 2021

#

LINEAR PROGRAMMING AND OPTIMISATION

###

ASSIGNMENT 3

###

THE SIMPLEX METHOD

###

COMPUTER IMPLEMENTATION

Library/module	Description
np	Library for scientific computing in Python
math	Module for performing mathematical tasks
tqdm	instant progress bar/meter while using looping statements
gc	garbage collecter interface

INPUT: > Cost function and its constraints

OUTPUT: >Basic feasible solution, the objective value and zj-cj value

[53]:
```
import numpy as np
import math
import gc
from tqdm import tqdm
```

Passing the linear program coefficients

[54]:
```
c = [1, 1, 0, 0, 0]
A = [[-1, 1, 1, 0, 0],
     [ 1, 0, 0, 1, 0],
     [ 0, 1, 0, 0, 1]]
b = [2, 4, 4]
```

[47]:
```
gc.collect()
```

[47]: 140

Convert the coefficients to a tableau and find pivot

[35]:
```
def simplex(c, A, b):
    tableau = to_tableau(c, A, b)

    while can_be_improved(tableau):
        pivot_position = get_pivot_position(tableau)
        tableau = pivot_step(tableau, pivot_position)

    return get_solution(tableau)
```

[36]:
```
def to_tableau(c, A, b):
    xb = [eq + [x] for eq, x in zip(A, b)]
    z = c + [0]
    return xb + [z]
```

[37]:
```
def can_be_improved(tableau):
    z = tableau[-1]
    return any(x > 0 for x in tqdm(z[:-1]))
```

[38]:
```
def get_pivot_position(tableau):
    z = tableau[-1]
    column = next(i for i, x in enumerate(z[:-1]) if x > 0)

    restrictions = []
    for eq in tqdm(tableau[:-1]):
        el = eq[column]
        restrictions.append(math.inf if el <= 0 else eq[-1] / el)

    row = restrictions.index(min(restrictions))
    return row, column
```

[40]:
```
def pivot_step(tableau, pivot_position):
    new_tableau = [[] for eq in tableau]

    i, j = pivot_position
    pivot_value = tableau[i][j]
    new_tableau[i] = np.array(tableau[i]) / pivot_value

    for eq_i, eq in enumerate(tableau):
        if eq_i != i:
            multiplier = np.array(new_tableau[i]) * tableau[eq_i][j]
            new_tableau[eq_i] = np.array(tableau[eq_i]) - multiplier

    return new_tableau
```

[51]:
```
def is_basic(column):
    return sum(column) == 1 and len([c for c in column if c == 0]) ==␣
 ↪len(column) - 1
```

[52]:
```
def get_solution(tableau):
    columns = np.array(tableau).T
    solutions = []
    for column in tqdm(columns):
        solution = 0
        if is_basic(column):
            one_index = column.tolist().index(1)
            solution = columns[-1][one_index]
        solutions.append(solution)

    return solutions
```

[42]:
```
def simplex(c, A, b):
    tableau = to_tableau(c, A, b)

    while can_be_improved(tableau):
        pivot_position = get_pivot_position(tableau)
        tableau = pivot_step(tableau, pivot_position)

    return get_solution(tableau)
```

[48]:
```
gc.collect()
```

[48]: 20

[49]:
```
solution = simplex(c, A, b)
print('Basic Feasible Solution: ', solution)
```

```
  0%|          | 0/5 [00:00<?, ?it/s]
100%|      | 3/3 [00:00<00:00, 9258.95it/s]
 20%|         | 1/5 [00:00<00:00, 2833.99it/s]
100%|      | 3/3 [00:00<00:00, 15650.39it/s]
100%|      | 5/5 [00:00<00:00, 37249.59it/s]
100%|      | 6/6 [00:00<00:00, 29959.31it/s]

Basic Feasible Solution:  [4.0, 4.0, 2.0, 0, 0, 0]
```

[ ]:
```

```