算法设计与分析第四次作业 - 线性规划

201628013229058 洪鑫

1 Airplane Landing Problem

问题描述

With human lives at stake, an air traffic controller has to schedule the airplanes that are landing at an airport in order to avoid airplane collision. Each airplane i has a time window $[s_i, t_i]$ during which it can safely land. You must compute the exact time of landing for each airplane that respects these time windows. Furthermore, the airplane landings should be stretched out as much as possible so that the minimum time gap between successive landings is as large as possible.
For example, if the time window of landing three airplanes are [10:00-11:00], [11:20-11:40], [12:00-12:20], and they land at 10:00, 11:20, 12:20 respectively, then the smallest gap is 60 minutes, which occurs between the last two airplanes.
Given n time windows, denoted as $[s_1, t_1]$, $[s_2, t_2]$, … , $[s_n, t_n]$ satisfying
$s_1 < t_1 < s_2 < t_2 < \ldots < s_n < t_n$, you are required to give the exact landing time of each airplane, in which the smallest gap between successive landings is maximized.
Please formulate this problem as an LP, construct an instance and use GLPK or Gurobi or other similar tools to solve it.

表达式

$$
\begin{aligned}
max \quad & y \\
s.t. \quad & x_{i+1} - x_i \geq y \quad \text{for each } i \in \{1, 2, \ldots, n-1\} \\
& s_i \leq x_i \leq t_i \quad \text{for each } i \in \{1, 2, \ldots, n\}
\end{aligned}
$$

其中，$x_i, x \in \{1, 2, \ldots, n\}$表示第$i$架飞机的具体降落时间，$y$表示最小时间间隔。

实例

[10, 11], [11, 12], [12, 13]

使用 GLPK 求解

执行命令：`glpsol --cpxlp airplane.lp -o airplane_result.txt`

输入文件：*airplane.lp*（CPLEX格式）

```
Maximize
    min_interval: y
Subject To
    c1: x2 - x1 - y >= 0
    c2: x3 - x2 - y >= 0
Bounds
    10 <= x1 <= 11
    11 <= x2 <= 12
    12 <= x3 <= 13
End
```

输出文件：*airplane_result.txt*

```
Problem:
Rows:       2
Columns:    4
Non-zeros:  6
Status:     OPTIMAL
Objective:  min_interval = 1.5 (MAXimum)

   No.   Row name   St   Activity     Lower bound   Upper bound    Marginal
------ ------------ -- ------------ ------------ ------------ -------------
     1 c1           NL             0            0                      -0.5
     2 c2           NL             0            0                      -0.5

   No. Column name  St   Activity     Lower bound   Upper bound    Marginal
------ ------------ -- ------------ ------------ ------------ -------------
     1 y            B            1.5            0
     2 x2           B           11.5           11           12
     3 x1           NL            10           10           11          -0.5
     4 x3           NU            13           12           13           0.5
...
End of output
```

实例的解为：$x_1 = 10, x_2 = 11.5, x_3 = 13$，最小时间间隔的最大值为：$1.5h$。
所以安排第一架飞机在10:00降落，第二架飞机在11:30降落，第三架飞机在13:00降落。r

2 Gas Station Placement

问题描述

> Let's consider a long, quiet country road with towns scattered very sparsely along it. Sinopec, largest oil refiner in China, wants to place gas stations along the road. Each gas station is assigned to a nearby town, and the distance be- tween any two gas stations being as small as possible. Suppose there are n towns with distances from one endpoint of the road being $d_1, d_2, \cdots, d_n$. n gas stations are to be placed along the road, one station for one town. Besides, each station is at most r far away from its correspond town. $d_1, \cdots, d_n$ and r have been given and satisfied $d_1 < d_2 < \cdots < d_n, 0 < r < d_1$ and $d_i + r < d_{i+1} - r$ for all i. The objective is to find the optimal placement such that the maximal distance between two successive gas stations is minimized.
> Please formulate this problem as an LP.

表达式

$$
\begin{array}{lrcll}
min & & y & & \\
s.t. & x_{i+1} - x_i & \leq y & & \text{for each } i \in \{1, 2, \ldots, n-1\} \\
& d_i - r \leq & x_i & \leq d_i + r & \text{for each } i \in \{1, 2, \ldots, n\}
\end{array}
$$

实例

$d = [4, 10, 16, 23], r = 2$

使用GLPK求解

执行命令：`glpsol --cpxlp station.lp -o station_result.txt`

输入文件：*station.lp*（CPLEX格式）

```
Minimize
    max_interval: y
Subject To
    c1: x2 - x1 - y <= 0
    c2: x3 - x2 - y <= 0
    c3: x4 - x3 - y <= 0
Bounds
    2 <= x1 <= 6
    8 <= x2 <= 12
    14 <= x3 <= 18
    21 <= x4 <= 25
End
```

输入文件：*station_result.txt*

```
Problem:
Rows:       3
Columns:    5
Non-zeros:  9
Status:     OPTIMAL
Objective:  max_interval = 5 (MINimum)

   No.   Row name   St   Activity     Lower bound   Upper bound    Marginal
------ ------------ -- ------------ ------------ ------------ -------------
     1 c1           NU             0                         0     -0.333333
     2 c2           NU             0                         0     -0.333333
     3 c3           NU             0                         0     -0.333333

   No. Column name  St   Activity     Lower bound   Upper bound    Marginal
------ ------------ -- ------------ ------------ ------------ -------------
     1 y            B              5            0
     2 x2           B             11            8           12
     3 x1           NU             6            2            6     -0.333333
     4 x3           B             16           14           18
     5 x4           NL            21           21           25      0.333333
...
End of output
```

实例的解为：$x_1 = 6, x_2 = 11, x_3 = 16, x_4 = 21$，最大距离间隔的最小值为：5.
所以应该在距离起始点6,11,16,21处设置加油站。

3 Simplex Implementation


```python
#!/usr/bin/python3
# Simplex method
# Author: HongXin
# 2016.11.17

import numpy as np


def xlpsol(c, A, b):
    """
    Solve linear programming problem with the follow format:
    min     c^Tx
    s.t.    Ax <= b
            x >= 0
    (c^T means transpose of the vector c)
```

```
    :return: x - optimal solution, opt - optimal objective value
    """
    (B, T) = __init(c, A, b)
    (m, n) = T.shape
    opt = -T[0, 0]  # -T[0, 0] is exactly the optimal value!
    v_c = T[0, 1:]
    v_b = T[1:, 0]
    v_A = T[1:,1:]
    inf = float('inf')

    while True:
        if all(T[0, 1:] >= 0):  # c >= 0
            # just get optimal solution by manipulating index and value
            x = map(lambda t: T[B.index(t) + 1, 0] if t in B else 0,
                    range(0, n - 1))
            return x, opt
        else:
            # choose fist element of v_c smaller than 0
            e = next(x for x in v_c if x < 0)
            delta = map(lambda i: v_b[i]/v_A[i, e]
                        if v_A[i, e] > 0 else inf,
                        range(0, m-1))
            l = delta.index(min(delta))
            if delta[l] == inf:
                print("unbounded")
                return None, None
            else:
                __pivot(B,T,e,l)

def __init(c, A, b):
    """
    0   c   0
    b   A   I
    """
    # transfer to vector and matrix
    (c, A, b) = map(lambda t: np.array(t), [c, A, b])
    [m, n] = A.shape
    if m != b.size:
        print('The size of b must equal with the row of A!')
        exit(1)
    if n != c.size:
        print('The size of c must equal with the column of A!')
        exit(1)
    part_1 = np.vstack((0, b.reshape(b.size, 1)))
    part_2 = np.vstack((c, A))
    part_3 = np.vstack((np.zeros(m), np.identity(m)))
    return range(n, n + m), np.hstack((np.hstack((part_1, part_2)), part_3))

```

```
64
65  def __pivot(B,T,e,l):
66      v_A = T[1:,1:]
67      T[l+1,:] = np.divide(T[l+1,:], v_A[l,e])
68      for i in range(0, T.shape[1]):
69          if i == l+1:
70              continue
71          T[i,:] -= T[l+1,:] * T[i, e+1]
72      B.remove(B[l])
73      B.add(e+1)
74
75
76  if __name__ == '__main__':
77      c = [-1, -14, -6]
78      A = [[1, 1, 1], [1, 0, 0], [0, 0, 1], [0, 3, 1]]
79      b = [4, 2, 3, 6]
80      [x, opt] = xlpsol(c, A, b)
```

Compare with GLPK

$$
\begin{array}{lrcrcrcl}
\min & -x_1 & - & 14x_2 & - & 6x_3 & & \\
s.t. & x_1 & + & x_2 & + & x_3 & \leq & 4 \\
 & x_1 & & & & & \leq & 2 \\
 & & & & & x_3 & \leq & 3 \\
 & & & 3x_2 & + & x_3 & \leq & 6 \\
 & x_1 & , & x_2 & , & x_3 & \geq & 0
\end{array}
$$

解得的最优解均为(0,1,3)，最优值为-32