

Assignment 4 of Algorithm

Zuyao Chen 201728008629002

zychen.uestc@gmail.com

1. Linear-inequality feasibility

Proof:

suppose that we have an algorithm for linear programming, that means we can solve

$$\begin{array}{ll}\max & \mathbf{c}^T \mathbf{x} \\ \text{s.t.} & \mathbf{Ax} \leq \mathbf{b} \\ & \mathbf{x} \geq \mathbf{0}\end{array}$$

given a linear inequality feasibility problem, our goal is to check whether there exist \mathbf{x}^* ,

$$\mathbf{A}'\mathbf{x}^* \leq \mathbf{b}', \mathbf{x}^* \geq \mathbf{0}$$

which equals to

$$\begin{array}{ll}\max & \mathbf{0} \cdot \mathbf{x} \\ \text{s.t.} & \mathbf{A}'\mathbf{x} \leq \mathbf{b}' \\ & \mathbf{x} \geq \mathbf{0}\end{array}$$

this problem can be solved using the same algorithm.

2. Airplane Landing Problem

Let x_1, x_2, \dots, x_n be the exact landing time of each airplane respectively, the problem can be written as

$$\begin{array}{ll}\max & \min(x_2 - x_1, x_3 - x_2, \dots, x_n - x_{n-1}) \\ \text{s.t.} & s_i \leq x_i \leq t_i, i = 1, 2, \dots, n\end{array}$$

LP form

$$\begin{array}{ll}\max & d \\ \text{s.t.} & s_i \leq x_i \leq t_i, i = 1, 2, \dots, n \\ & x_k - x_{k-1} \geq d, k = 2, 3, \dots, n \\ & d \geq 0\end{array}$$

Now we show that how to obtain the dual form of this question, first we minimize $-d$,

$$\begin{array}{ll}\min & z = -d \\ \text{s.t.} & s_i \leq x_i \leq t_i, i = 1, 2, \dots, n \\ & -x_{k+1} + x_k \leq -d = z, k = 1, 2, \dots, n-1 \\ & z \leq 0\end{array}$$

using Lagrange multiplier,

$$L(z, x, \lambda, \alpha, \varphi, \phi) = z + \sum_{i=1}^n \lambda_i(x_i - t_i) + \sum_{i=1}^n \alpha_i(-x_i + s_i) + \sum_{i=1}^{n-1} \varphi_i(-x_{i+1} + x_i - z) + \phi z$$

$$\frac{\partial L}{\partial z} = 1 - \sum_{i=1}^{n-1} \varphi_i + \phi = 0$$

$$\frac{\partial L}{\partial x_1} = \lambda_1 - \alpha_1 + \varphi_1 = 0$$

$$\frac{\partial L}{\partial x_n} = \lambda_n - \alpha_n - \varphi_{n-1} = 0$$

$$\frac{\partial L}{\partial x_i} = \lambda_i - \alpha_i + \varphi_i - \varphi_{i-1} = 0, i = 2, 3, \dots, n-1$$

thus

$$g(\lambda, \alpha, \varphi, \phi) = \inf_{z, x} L(z, x, \lambda, \alpha, \varphi, \phi)$$

$$\begin{aligned} &= z(1 - \sum_{i=1}^n \varphi_i + \phi) + x_1(\lambda_1 - \alpha_1 + \varphi_1) + x_n(\lambda_n - \alpha_n - \varphi_{n-1}) - \sum_{i=1}^n \lambda_i t_i + \sum_{i=1}^n \alpha_i s_i \\ &= - \sum_{i=1}^n \lambda_i t_i + \sum_{i=1}^n \alpha_i s_i \end{aligned}$$

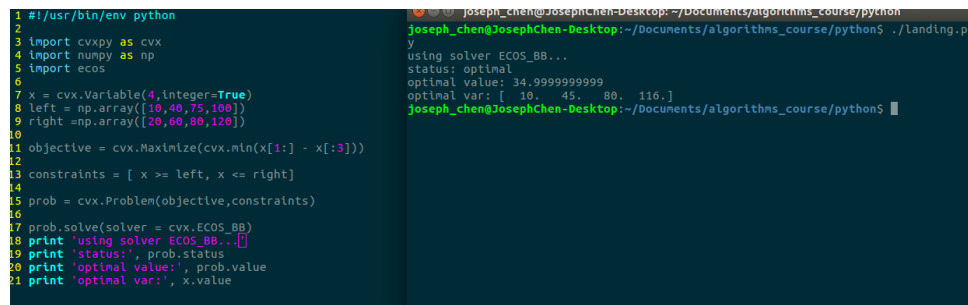
Minimizing $-d$ is equivalent to

$$\begin{aligned} \max \quad & - \sum_{i=1}^n \lambda_i t_i + \sum_{i=1}^n \alpha_i s_i \\ \text{s.t.} \quad & \lambda_i \geq 0, \alpha_i \geq 0, i = 1, 2, \dots, n \\ & \varphi_i \geq 0, i = 1, 2, \dots, n-1 \\ & \lambda_1 - \alpha_1 + \varphi_1 = 0 \\ & \lambda_n - \alpha_n - \varphi_{n-1} = 0 \\ & \lambda_i - \alpha_i + \varphi_i - \varphi_{i-1} = 0, i = 2, 3, \dots, n-1 \\ & 1 - \sum_{i=1}^{n-1} \varphi_i \leq 0 \end{aligned}$$

Hence the original problem that Maximizing d is equivalent to

$$\begin{aligned} \min \quad & \sum_{i=1}^n \lambda_i t_i - \sum_{i=1}^n \alpha_i s_i \\ \text{s.t.} \quad & \lambda_i \geq 0, \alpha_i \geq 0, i = 1, 2, \dots, n \\ & \varphi_i \geq 0, i = 1, 2, \dots, n-1 \\ & \lambda_1 - \alpha_1 + \varphi_1 = 0 \\ & \lambda_n - \alpha_n - \varphi_{n-1} = 0 \\ & \lambda_i - \alpha_i + \varphi_i - \varphi_{i-1} = 0, i = 2, 3, \dots, n-1 \\ & 1 - \sum_{i=1}^{n-1} \varphi_i \leq 0 \end{aligned}$$

for instance, we have $n = 4$, $[10, 20]$, $[40, 60]$, $[75, 80]$, $[100, 120]$ (here the minute is the metric of time), using tool cvxpy we can obtain the optimal solution 35 with optimal variables $x_1 = 10, 45, 80, 116$.



```

1 #!/usr/bin/env python
2
3 import cvxpy as cvx
4 import numpy as np
5 import ecos
6
7 x = cvx.Variable(4, integer=True)
8 left = np.array([10, 40, 75, 100])
9 right = np.array([20, 60, 80, 120])
10
11 objective = cvx.Maximize(cvx.min(x[1:] - x[:3]))
12
13 constraints = [ x >= left, x <= right]
14
15 prob = cvx.Problem(objective, constraints)
16
17 prob.solve(solver = cvx.ECOS_BB)
18 print 'using solver ECOS_BB...'
19 print 'status:', prob.status
20 print 'optimal value:', prob.value
21 print 'optimal var:', x.value

```

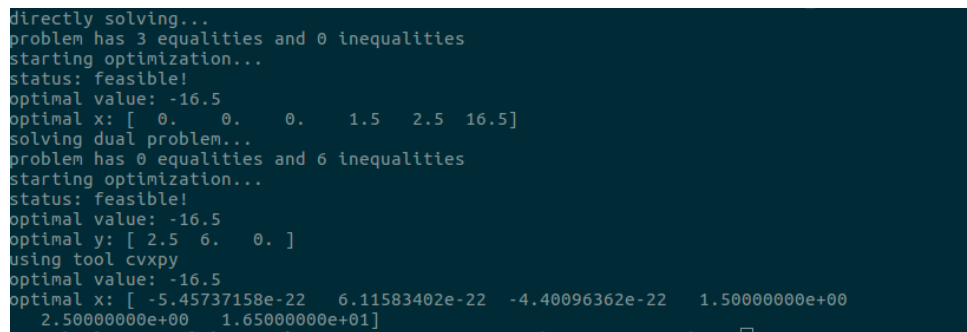
```

Joseph_chen@JosephChen-Desktop:~/Documents/algorithms_course/python$ ./landing.p
y
using solver ECOS_BB...
status: optimal
optimal value: 34.9999999999
optimal var: [ 10.  45.  80. 116.]
Joseph_chen@JosephChen-Desktop:~/Documents/algorithms_course/python$

```

3. Dual Simplex Algorithm

the result is showed below.



```

Directly solving...
problem has 3 equalities and 0 inequalities
starting optimization...
status: feasible!
optimal value: -16.5
optimal x: [ 0.  0.  0.  1.5  2.5 16.5]
solving dual problem...
problem has 0 equalities and 6 inequalities
starting optimization...
status: feasible!
optimal value: -16.5
optimal y: [ 2.5  6.  0. ]
using tool cvxpy
optimal value: -16.5
optimal x: [ -5.45737158e-22  6.11583402e-22 -4.40096362e-22  1.50000000e+00
  2.50000000e+00  1.65000000e+01]

```