091M4041H - Assignment 4

胡鹏飞 计控学院 201828007329009

1. Linear-inequality feasibility

Given a set of m linear inequalities on n variables x1; x2; …… ; xn, the linear-inequality feasibility problem asks if there is a setting of the variables that simultaneously satisfies each of the inequalities.

Show that if we have an algorithm for linear programming, we can use it to solve the linear-inequality feasibility problem. The number of variables and constraints that you use in the linear-programming problem should be polynomial in n and m.

**The formulation of an LP**

根据题意，现在存在一个线性规划的算法以及一组变量，需要解决的是如何利用这个算法来判断这组变量是否linear-inequality。

假设线性规划的算法为



参照老师在课堂上讲述的内敛法技巧，我们将算法中目标函数改变.即



如果算法能够求得最优解x0=0,则linear-inequality是可行的，否则为不可行。

1. Gas Station Placement

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 between any two gas stations being as small as possible. Suppose there are n towns with distances from one endpoint of the road being d1; d2; ……; dn. 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. d1;……; dn and r have been given and satisfied d1 < d2 < …… < dn, 0 < r < d1 and di + r < di+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..

**The formulation of an LP**

根据题意，假设两个相邻的加油站之间的最大距离为z，第i个加油站的位置为xi，xi距di不能超过r,所以可将上述实际问题建模为：



6 Dual Simplex Algorithm

For the problem

minimize

-7x1 + 7x2 - 2x3 - x4 - 6x5

subject to:

3x1 - x2 + x3 - 2x4 = -3

2x1 + x2 + x4 + x5 = 4

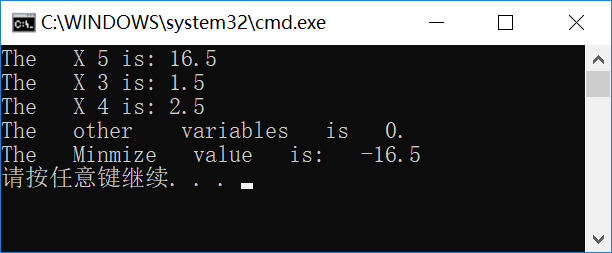
-x1 + 3x2 - 3x4 + x6 = 12

xi > 0; (i = 1;……; 6)

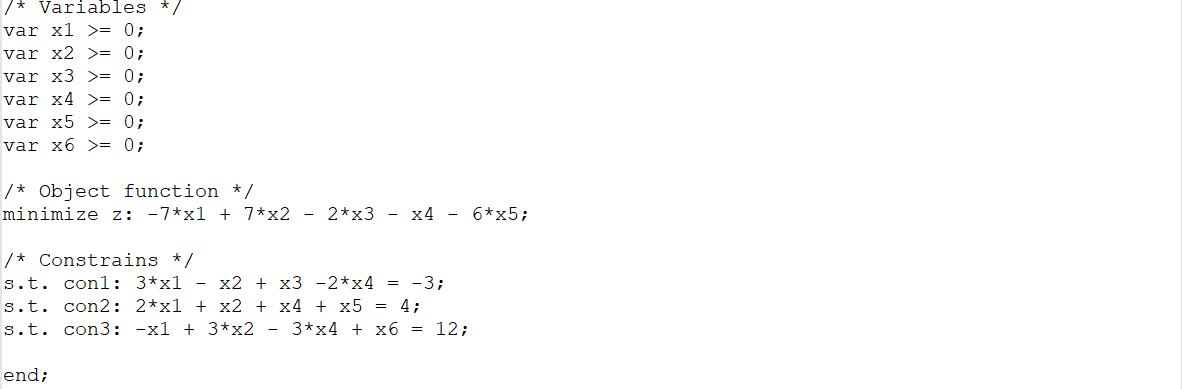
Implement dual simplex algorithm with your favorate language to solve this problem, and make comparison result with GLPK or Gurobi or other similar tools.

**4.1 The result of C++**

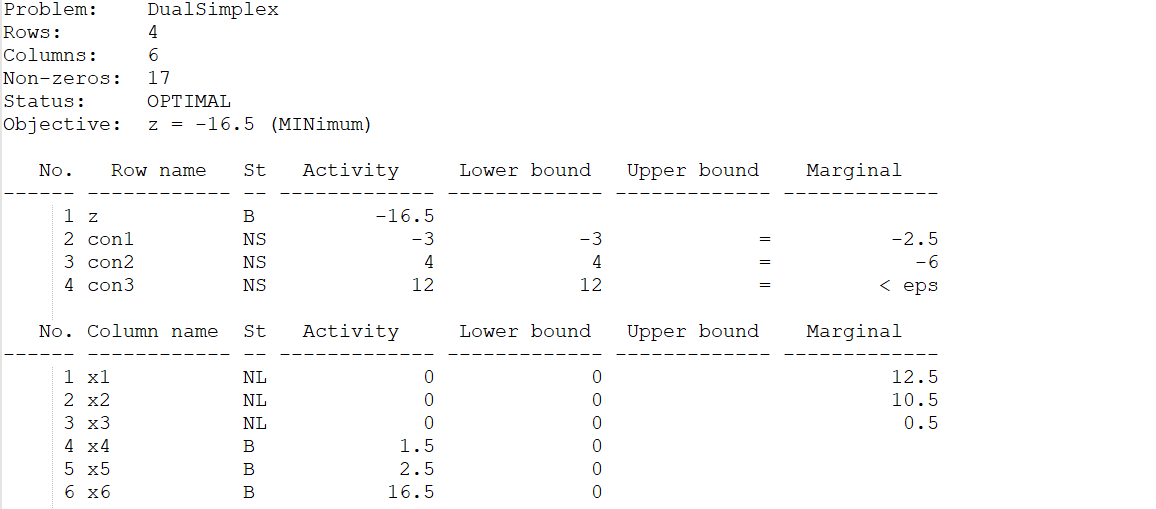
代码见附录，结果如下



**4.2 The result of GLPK**



运行结果为



同样，求得的解为[0,0,0,1.5,2.5,16.5],最终结果为z=-16.5，与刚才计算结果一致。

**附录：**

#include <iostream>

#include <conio.h>

#include <math.h>

#include <stdio.h>

**using** **namespace** std**;**

**typedef** int BOOL**;**

#define TRUE 1

#define FALSE 0

**typedef** double REAL**;**

#define ZERO 1e-10

//矩阵求逆

BOOL Inv**(**REAL **\*\*** a**,** int n**);**

BOOL Inv**(**REAL **\*** a**,** int n**);**

//矩阵相乘

void Damul**(**REAL **\*** a**,** REAL **\*** b**,** size\_t m**,** size\_t n**,** size\_t k**,** REAL **\*** c**);**

//线形规划

BOOL Line\_Optimize**(**REAL **\*** A**,** REAL **\*** B**,** REAL **\*** C**,** int m**,** int n**,**

REAL **\*** Result**,** REAL **\*** X**,** int **\*** Is**);**

template **<**class T**>**

inline void ExChange**(**T**&** a**,** T**&** b**)**

**{**

T temp **=** a**;**

a **=** b**;**

b **=** temp**;**

**}**

BOOL Inv**(**REAL **\*\*** a**,** int n**)**

**{**

REAL d**;**

int i**,** j**,** k**;**

int success **=** FALSE**;**

int **\*** is **=** **new** int**[**n**];**

int **\*** js **=** **new** int**[**n**];**

**for** **(**k **=** 0**;** k **<**n**;** k**++)** **{**

d **=** 0.0**;**

**for** **(**i **=** k**;** i **<**n**;** i**++)** **{**

**for** **(**j **=** k**;** j **<**n**;** j**++)** **{**

**if** **(**fabs**(**a**[**i**][**j**])>** d**)** **{**

d **=** fabs**(**a**[**i**][**j**]);**

is**[**k**]** **=** i**;**

js**[**k**]** **=** j**;**

**}**

**}**

**}**

**if** **(**d **<**ZERO**)** **goto** Clear**;**

**for** **(**j **=** 0**;** j **<**n**;** j**++)**ExChange**(**a**[**k**][**j**],** a**[**is**[**k**]][**j**]);**

**for** **(**i **=** 0**;** i **<**n**;** i**++)**ExChange**(**a**[**i**][**k**],** a**[**i**][**js**[**k**]]);**

a**[**k**][**k**]** **=** 1 **/** a**[**k**][**k**];**

**for** **(**j **=** 0**;** j **<**n**;** j**++)** **{**

**if** **(**j **!=** k**)**a**[**k**][**j**]** **\*=** a**[**k**][**k**];**

**}**

**for** **(**i **=** 0**;** i **<**n**;** i**++)** **{**

**if** **(**i **!=** k**)** **{**

**for** **(**j **=** 0**;** j **<**n**;** j**++)**

**if** **(**j **!=** k**)**a**[**i**][**j**]** **-=** a**[**i**][**k**]** **\*** a**[**k**][**j**];**

**}**

**}**

**for** **(**i **=** 0**;** i **<**n**;** i**++)** **{**

**if** **(**i **!=** k**)** **{**

a**[**i**][**k**]** **\*=** **((-**1.0**)\***a**[**k**][**k**]);**

**}**

**}**

**}** //end for

**for** **(**k **=** **(**n **-** 1**);** k **>=** 0**;** k**--)**

**{**

**for** **(**j **=** 0**;** j **<**n**;** j**++)**

ExChange**(**a**[**k**][**j**],** a**[**js**[**k**]][**j**]);**

**for** **(**i **=** 0**;** i **<**n**;** i**++)**

ExChange**(**a**[**i**][**k**],** a**[**i**][**is**[**k**]]);**

**}**

success **=** TRUE**;**

Clear**:**

**delete[]** is**;**

**delete[]** js**;**

**return** success**;**

**}**

BOOL Inv**(**REAL **\*** a**,** int n**)**

**{**

REAL **\*\***kma **=** **new** REAL**\*[**n**];**

**for** **(**int i **=** 0**;** i **<**n**;** i**++)** **{**

kma**[**i**]** **=** a **+** i**\***n**;**

**}**

BOOL ret **=** Inv**(**kma**,** n**);**

**delete[]** kma**;**

**return** ret**;**

**}**

void Damul**(**REAL **\*** a**,** REAL **\*** b**,** size\_t m**,** size\_t n**,** size\_t k**,** REAL **\*** c**)**

**{**

unsigned int i**,** j**,** l**,** u**;**

**for** **(**i **=** 0**;** i **<=** **(**m **-** 1**);** i**++)**

**{**

**for** **(**j **=** 0**;** j **<=** **(**k **-** 1**);** j**++)**

**{**

u **=** i**\***k **+** j**;**

c**[**u**]** **=** 0.0**;**

**for** **(**l **=** 0**;** l **<=** n **-** 1**;** l**++)**

**{**

c**[**u**]** **+=** a**[**i**\***n **+** l**]** **\*** b**[**l**\***k **+** j**];**

**}**

**}**

**}**

**return;**

**}**

BOOL Line\_Optimize**(**REAL **\*** A**,** REAL **\*** B**,** REAL **\*** C**,** int m**,** int n**,**

REAL **\*** Result**,** REAL **\*** X**,** int **\*** Is**)**

**{**

REAL r**;**

int i**,** j**,** k**;**

int Success **=** FALSE**;**

REAL**\*** b **=** **new** REAL**[**m**\***m**];**

REAL**\*** MatTmp **=** **new** REAL**[**m**\***m**];**

REAL**\*** Mat1 **=** **new** REAL**[**m**];**

REAL**\*** Mat2 **=** **new** REAL**[**m**];**

REAL**\*** E **=** **new** REAL**[**m**\***m**];**

**for** **(**i **=** 0**;** i **<**m**;** i**++)** **{**

**for** **(**j **=** 0**;** j **<**m**;** j**++)** **{**

b**[**i**\***m **+** j**]** **=** A**[**i**\***n **+** Is**[**j**]];**

**}**

**}**

**if** **(!**Inv**(**b**,** m**))** **{**

**goto** Release**;**

**}**

Damul**(**b**,** B**,** m**,** m**,** 1**,** X**);**

**for** **(;;)** **{**

**for** **(**i **=** 0**;** i **<**m**;** i**++)** **{**

Mat2**[**i**]** **=** C**[**Is**[**i**]];**

**}**

Damul**(**Mat2**,** b**,** 1**,** m**,** m**,** Mat1**);**

**for** **(**i **=** 0**;** i **<**n**;** i**++)** **{**

**for** **(**j **=** 0**;** j **<**m**;** j**++)** **{**

Mat2**[**j**]** **=** A**[**j**\***n **+** i**];**

**}**

Damul**(**Mat1**,** Mat2**,** 1**,** m**,** 1**,** **&**r**);**

r **=** C**[**i**]** **-** r**;**

**if** **(**r **<-**ZERO**)** **{**

**break;**

**}**

**}**

**if** **(**i **>=** n**)**

**{**

**\***Result **=** 0**;**

**for** **(**i **=** 0**;** i **<**m**;** i**++)** **{**

**\***Result **+=** C**[**Is**[**i**]]** **\*** X**[**i**];**

**}**

Success **=** TRUE**;**

**goto** Release**;**

**}**

Damul**(**b**,** Mat2**,** m**,** m**,** 1**,** Mat1**);**

r **=** 1E10**;**

j **=** **-**1**;**

**for** **(**k **=** 0**;** k **<**m**;** k**++)** **{**

**if** **(**Mat1**[**k**]>** ZERO**)** **{**

REAL temp **=** X**[**k**]** **/** Mat1**[**k**];**

**if** **(**temp **<**r**)** **{**

r **=** temp**;**

j **=** k**;**

**}**

**}**

**}**

**if** **(**j **<** 0**)** **{**

Success **=** FALSE**;**

**goto** Release**;**

**}**

**for** **(**k **=** 0**;** k **<**m**\***m**;** k**++)** **{**

E**[**k**]** **=** 0**;**

**}**

**for** **(**k **=** 0**;** k **<**m**;** k**++)** **{**

E**[**k**\***m **+** k**]** **=** 1**;**

**}**

**for** **(**k **=** 0**;** k **<**m**;** k**++)** **{**

E**[**k**\***m **+** j**]** **=** **-**Mat1**[**k**]** **/** Mat1**[**j**];**

**}**

E**[**j**\***m **+** j**]** **=** 1 **/** Mat1**[**j**];**

Is**[**j**]** **=** i**;**

Damul**(**E**,** b**,** m**,** m**,** m**,** MatTmp**);**

Damul**(**E**,** X**,** m**,** m**,** 1**,** Mat2**);**

**for** **(**i **=** 0**;** i **<**m**\***m**;** i**++)** **{**

b**[**i**]** **=** MatTmp**[**i**];**

**}**

**for** **(**i **=** 0**;** i **<**m**;** i**++)** **{**

X**[**i**]** **=** Mat2**[**i**];**

**}**

**}**

Release**:**

**delete[]** E**;**

**delete[]** Mat2**;**

**delete[]** Mat1**;**

**delete[]** MatTmp**;**

**delete[]** b**;**

**return** Success**;**

**}**

void main**()**

**{**

REAL A**[]** **=** **{**

3**,** **-**1**,** 1**,** **-**2**,** 0**,** 0**,**

2**,** 1**,** 0**,** 1**,** 1**,** 0**,**

**-**1**,** 3**,** 0**,** **-**3**,** 0**,** 1**,**

**};**

REAL B**[]** **=** **{**

**-**3**,**4**,**12

**};**

REAL C**[]** **=** **{**

**-**7**,** 7**,** **-**2**,** **-**1**,** **-**6**,** 0

**};**

REAL RESULT**;**

REAL x**[**3**];**

int Is**[]** **=** **{**

0**,**1**,**3

**};**

**if** **(!**Line\_Optimize**(**A**,** B**,** C**,** 3**,** 6**,** **&**RESULT**,** x**,** Is**))** **{**

cout **<<** "Calculate Wrong! " **<<** endl**;**

**return;**

**}**

**for** **(**int i **=** 0**;** i **<**3**;** i**++)** **{**

cout **<<** "The X " **<<** Is**[**i**]** **<<** " is: " **<<** x**[**i**]** **<<** endl**;**

**}**

cout **<<** "The other variables is 0. " **<<** endl**;**

cout **<<** "The Minmize value is: " **<<** RESULT **<<** endl**;**

**}**