# math_464_actual_hw9

March 21, 2024

**#Math 464. HW9. Lazizbek**

## 1 Exercise 4.1

```
min z  =  x1 -  x2  + 0 x3 + 0 x4

s.t.    2 x1 + 3 x2 -   x3 +   x4 <= 0

        3 x1 +   x2 + 4 x3 - 2 x4 >= 3

        - x1 -   x2 + 2 x3 +   x4  = 6

          x1 <= 0

          x2,x3 >= 0

          x4 free
```

**Dual of this LP above is as follows:**

```
max w  = 0 p1 + 3 p2  + 6 p3

s.t.     2 p1 + 3 p2  - 1 p3 >=  1

         3 p1 + 1 p2  - 1 p3 <= -1

        -1 p1 + 4 p2  + 2 p3 <=  0

         1 p1 - 2 p2  + 1 p3  =  0

           p1 <= 0

           p2 >= 0

           p3 free
```

## 2 *Linear Program for the Primal LP*

[ ]:
```
"""
Script to demonstrate how to solve small mixed integer programs
using the python optimize module in the scipy package
"""
"""
All decision variables are from reals:
"""

import pandas as pd
import numpy as np
import scipy.optimize as opt

# The problem we will solve is:

# EXercise 4.1


# min z  =  x1 -  x2  + 0 x3 + 0 x4

# s.t.    2 x1 + 3 x2 -   x3 +   x4 <=  0

#       - 3 x1 -   x2 - 4 x3 + 2 x4 <= -3

#         - x1 -   x2 + 2 x3 +   x4  =  6

#            x1 <= 0

#            x2,x3 >= 0

#            x4 free

#            x1, x2, x3, x4 in R

function_array = list()
solution_array = list()

# First build the objective vector.
c=np.array([ 1, -1,  0,  0])

# Next, create the coefficient array for the inequality constraints.
# Note that the inequalities must be Ax <= b, so some sign
# changes result when converting >= into <=.
# A = None # if no inequality constraints

A = np.array([[   2,   3,  -1,   1],\
```

```
               [ - 3, - 1, - 4,   2]])

# Next the right-hand-side vector for the inequalities
# Sign changes can occur here too.
# b = None # if no inequality constraints
b = np.array([ 0 , - 3])


#The coefficient matrix for the equality constraints and
# the right hand side vector.
# Ae = None # if no equality constraints

Ae = np.array([[ - 1, - 1,   2,    1]])

# be = None # if no equality constraints
be = np.array([ 6 ])

# Next, we provide any lower and upper bound vectors, one
# value for each decision variable.  In this example all
# lower bound are zero and there are no upper bounds.
bounds=(( - np.inf, 0 ), ( 0, np.inf), ( 0, np.inf), ( - np.inf, np.inf))

# Lastly, we can specify which variables are required to be integer.
# If no variables are integer then isint=[];  In our example, only x2
# is integer.
# isint=[]

# The call to the mixed integer solver looks like the following.
# Notice that we pass usual "c" when we have a minimization
# problem, we send "-c" when we have max problem.
# This is because the solver is expecting a minimization.

res=opt.linprog(c,A,b,Ae,be,bounds)

# The result is stored in the dictionary variable "res".
# In particular, to show the optimal objective value and the
# optimal decision variable values:

print("min z = ", res['fun'])
print("at optimal solution x = ", res['x'])

print(res['message'])
# print(res['x'])
# print(res)
# print(np.dot(c, res['x']))

# To download:
```

```
# !sudo apt-get install texlive-xetex texlive-fonts-recommended
↪texlive-plain-generic
# !jupyter nbconvert --to pdf /content/math_464_actual_hw2.ipynb
```

```
min z =  None
at optimal solution x =  None
The problem is unbounded. (HiGHS Status 10: model_status is Unbounded;
primal_status is At upper bound)
```

## 3 *Linear Program for the Dual LP*

**Since the dual LP is a max problem, I'll turn it into min problem during the program by giving negative, -w to make it compatible for the software:**

[ ]:
```
"""
Script to demonstrate how to solve small mixed integer programs
using the python optimize module in the scipy package
"""
"""
All decision variables are from reals:
"""

import pandas as pd
import numpy as np
import scipy.optimize as opt

# The problem we will solve is:

# Dual LP of Exercise 4.1


# min - w = - 0 p1 - 3 p2  - 6 p3  # turning to min problem

# s.t.        2 p1 + 3 p2  - 1 p3 >=  1

#             3 p1 + 1 p2  - 1 p3 <= -1

#            -1 p1 + 4 p2  + 2 p3 <=  0

#             1 p1 - 2 p2  + 1 p3  =  0

#               p1 <= 0

#               p2 >= 0

#               p3 free
```

```
#               p1, p2, p3 in R

function_array = list()
solution_array = list()

# First build the objective vector.
c=np.array([ 0, -3,  -6])

# Next, create the coefficient array for the inequality constraints.
# Note that the inequalities must be Ax <= b, so some sign
# changes result when converting >= into <=.
# A = None # if no inequality constraints

A = np.array([[ - 2, - 3,   1],\
              [   3,   1,  -1],\
              [ - 1,   4,   2]])

# Next the right-hand-side vector for the inequalities
# Sign changes can occur here too.
# b = None # if no inequality constraints
b = np.array([ - 1, - 1,  0])


#The coefficient matrix for the equality constraints and
# the right hand side vector.
# Ae = None # if no equality constraints

Ae = np.array([[   1, - 2,   1]])

# be = None # if no equality constraints
be = np.array([ 0 ])

# Next, we provide any lower and upper bound vectors, one
# value for each decision variable.  In this example all
# lower bound are zero and there are no upper bounds.
bounds=(( - np.inf, 0 ), ( 0, np.inf), ( - np.inf, np.inf))

# Lastly, we can specify which variables are required to be integer.
# If no variables are integer then isint=[];  In our example, only x2
# is integer.
# isint=[]

# The call to the mixed integer solver looks like the following.
# Notice that we pass usual "c" when we have a minimization
# problem, we send "-c" when we have max problem.
# This is because the solver is expecting a minimization.
```

```
res=opt.linprog(c,A,b,Ae,be,bounds)

# The result is stored in the dictionary variable "res".
# In particular, to show the optimal objective value and the
# optimal decision variable values:

print("min w = ", res['fun'])
print("at optimal solution x = ", res['x'])
print(res['message'])
# print(res)
# print(np.dot(c, res['x']))
```

```
min w =  None
at optimal solution x =  None
The problem is infeasible. (HiGHS Status 8: model_status is Infeasible;
primal_status is Basic)
```

**By the reulting table of possibilities for Primal and Dual LP's that we created based on Theorem 4.1(Weak duality), and Theorem 4.2(Strong duality),**

**Unbounded Primal LP => Infeasible Dual LP!**

**I'll give the reasoning on a separate page!**

[ ]:
```
# !sudo apt-get install texlive-xetex texlive-fonts-recommended
 ↪texlive-plain-generic
```

[ ]:
```
# !jupyter nbconvert --to pdf /content/math_464_actual_hw8.ipynb
```

MATH 464. HW#9. Lazizbek.

Discussing the results:

LP Primal result:

(1) min $z$ = None

$x$ = None

The problem is unbounded $\Rightarrow$

(2) $z^* = -\infty \Rightarrow$

"min problem"

(3) • I can make $z = c^T x$ as small as possible & stay feasible, then by Theorem 4.1 (Weak duality) theorem says, if a feasible point $p$ exists, then $p^T b \le c^T x = -\infty$! $\Rightarrow$

It is impossible to have $p$. s.t

(4) $p^T b \le -\infty$. $\Rightarrow$

That is given Primal LP is unbundend & if a feasible point $p$ exists, then $p^T b \le -\infty$, impossible happens, therefore there's no feasible point $p \Rightarrow$

(5) Dual program is infeasible as in the result from the software result for Dual LP!