Caleb Logemann
MATH 566 Discrete Optimization
Homework 4

1. Write a program that will solve an instance of linear programming by enumerating basic feasible solutions and keeping the best one. Input to the program is matrix, $A \in \mathbb{R}^{m \times n}$ and vectors $\mathbf{b} \in \mathbb{R}^m$ and $\mathbf{c} \in \mathbb{R}^n$

   The following script implements this function that solves a linear program by trying all basic feasible solutions. The function acheives the same values of the objective function as Sage's built in linear program solver.

```
# https://wiki.sagemath.org/quickref?action=AttachFile&do=get&target=
    ↪ quickref-linalg.pdf

# I also suggest to google itertools.combinations
#
# This HW is about writing a program that solves linear programming
    ↪ problems where
# the input is A,b,c and you want to find x >= 0 that minimizes c^Tx if
    ↪ Ax = b.
#
# You should write a solver that enumerates all basic feasible solutions
    ↪  and picks
# the best one. Recall that you get a basic feasible solution by picking
    ↪  m columns
# of A that are linearly independent and solve for x.
#
# The same is also implemented using sage solver so you can check if
    ↪ your program
# produced the correct result.
#
import itertools as it

# This is where you will write your program
def solve_linear_program(A,b,c):
    m = len(b)
    n = len(c)
    if A.nrows() != m or A.ncols() != n or m > n:
        print "Invalid input"
        return [None,None]

    min_value = None
    min_x     = None

    # get all possible combinations of columns
    combinations = list(it.combinations(range(n), m))
    for v in combinations:
        # vector of zeroes
        x = vector(QQ, n)
        # solve subsystem
        if A[:,v].is_invertible():
```

```
            u = A[:,v]\b
            # use constraint that all variables must be nonnegative
            if min(u) >= 0:
                # put nonzero values into zeros vector
                for i in range(m):
                    x[v[i]] = u[i]
                # calculate value of objective function
                value = c.dot_product(x)
                # test if better than previous minimum value
                if value < min_value or min_value == None:
                    min_value = value
                    min_x = x

    return [min_value,min_x]

# This is solver using sage
def solve_using_sage(A,b,c):
    m = len(b)
    n = len(c)
    p = MixedIntegerLinearProgram(maximization=False);
    x = p.new_variable(nonnegative=True);
    for i in range(m):
        p.add_constraint( b[i] == sum([A[i,j]*x[j] for j in range(n)]) )
    p.set_objective(sum( [ c[j]*x[j] for j in range(n)] ))
    return p.solve()

# This is running your and also sage solver and prints the results
def test_solver(A,b,c):
    min_value, min_x = solve_linear_program(A,b,c)
    min_value_sage = solve_using_sage(A,b,c)
    print "Optimal solution has value", min_value_sage, "You computed",
        ↪ min_value, "for x=",min_x

######### Several test data

# Input
A = matrix(QQ,[[1,0,1,0,0],[1,1,0,1,0],[-1,1,0,0,1]])
b = vector(QQ,[4,5,1])
c = vector(QQ,[-1,-2,0,0,0])

test_solver(A,b,c)


# Input
A = matrix(QQ,[[1,-2,1,0,0],[1,1,0,1,0],[-1,1,0,0,1]])
b = vector(QQ,[4,2,4])
c = vector(QQ,[-1,2,0,0,0])

test_solver(A,b,c)
```

```
# Input
A = matrix(QQ
    ↪ ,[[1,-2,6,6,1,1,0,0],[1,1,2,2,-2,0,1,0],[-1,1,3,3,6,0,0,1]])
b = vector(QQ,[4,2,5])
c = vector(QQ,[-1,2,-2,-2,1,0,0,0])

test_solver(A,b,c)
```

This is the output of the script.

2. Convert the following program to equational form and solve it using the simplex method.

$$(P)\begin{cases} \text{maximize} & x_1 + 2x_2 & & \\ \text{subject to} & x_1 & \leq & 4 \\ & x_1 + x_2 & \leq & 5 \\ & -x_1 + x_2 & \leq & 1 \\ & x_1, x_2 \geq 0 & & \end{cases}$$

Use Bland's rule for selecting pivots. That is, pivot on variable with lowest possible index. Use the last tableau to argue that the solution is indeed optimal. Plot the set of feasible solutions of (P) and mark the solutions obtained after each iteration of the simplex method.

The new linear program in equational form with slack variables is

$$(P)\begin{cases} \max & x_1 + 2x_2 & & & & & & = & 4 & \\ \text{s.t.} & x_1 & + x_3 & & & & & = & 4 & \\ & x_1 + x_2 & & + x_4 & & & & = & 5 & \\ & -x_1 + x_2 & & & + x_5 & & & = & 1 & \\ & & & & & & x_i & \geq & 0 & 1 \leq i \leq 5 \end{cases}$$

An initial basic feasible solution with $x_3 = 4$, $x_4 = 5$ and $x_5 = 1$.

$$\begin{aligned} x_3 &= 4 - x_1 \\ x_4 &= 5 - x_1 - x_2 \\ x_5 &= 1 + x_1 - x_2 \\ z &= 0 + x_1 + 2x_2 \end{aligned}$$

Increasing x_1 by 4 results in the following.

$$\begin{aligned} x_1 &= 4 - x_3 \\ x_4 &= 1 - x_2 + x_3 \\ x_5 &= 5 - x_2 - x_3 \\ z &= 4 + 2x_2 - x_3 \end{aligned}$$

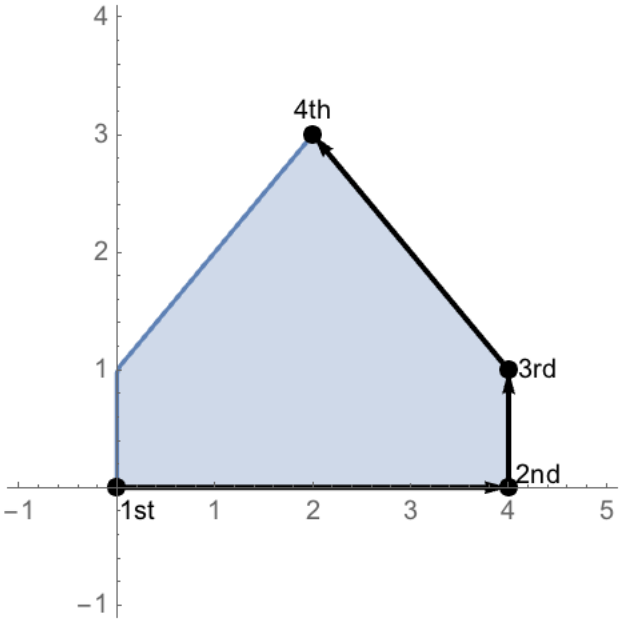
Now x_2 can be increased by 1.

$$\begin{aligned} x_1 &= 4 - x_3 \\ x_2 &= 1 + x_3 - x_4 \\ x_5 &= 4 - 2x_3 - x_4 \\ z &= 6 + x_3 - 2x_4 \end{aligned}$$

Lastly x_3 can be increased by 2.

$$\begin{aligned} x_1 &= 2 + (1/2)x_4 + x_5 \\ x_2 &= 3 - (3/2)x_4 - x_5 \\ x_3 &= 2 - (1/2)x_4 - x_5 \\ z &= 8 - (5/2)x_4 - x_5 \end{aligned}$$

This is the optimal solution because all the coefficients of the objective function are negative.



3. Use simplex method on the following program:

$$(P)\begin{cases} \text{maximize} & x_1 & & & & & \\ \text{subject to} & x_1 & - & x_2 & \leq & 1 \\ & -x_1 & + & x_2 & \leq & 2 \\ & & & x_1, x_2 \geq 0 & & \end{cases}$$

What is happening in the computation?

The new linear program with slack variables is

$$(P)\begin{cases} \text{maximize} & x_1 & & & & & & & & \\ \text{subject to} & x_1 & - & x_2 & + & x_3 & & & = & 1 \\ & -x_1 & + & x_2 & & & + & x_4 & = & 2 \\ & & & & & & & x_i & \geq & 0 \end{cases}$$

The initial basic feasible is $x_3 = 1$ and $x_4 = 2$.

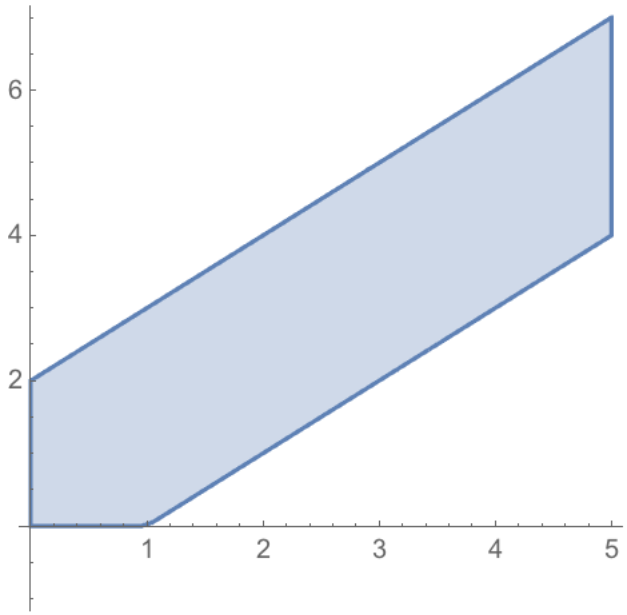
$$\begin{aligned} x_3 &= 1 - x_1 + x_2 \\ x_4 &= 2 + x_1 - x_2 \\ z &= 0 + x_1 \end{aligned}$$

First increase x_1 by 1.

$$\begin{aligned} x_1 &= 1 + x_2 - x_3 \\ x_4 &= 3 - x_3 \\ z &= 1 + x_2 - x_3 \end{aligned}$$

Now we see that x_2 should be increased however there is no condition limiting the increase of x_2. This implies that the feasible region is unbounded and the objective function can be increased indefinitely.

As can be seen in the following image the feasible region continues out to infinity to the right.



4. How would you solve a program $(P) =$ maximize $\mathbf{c}^T\mathbf{x}$ s.t. $A\mathbf{x} \leq \mathbf{b}, \mathbf{x} \geq \mathbf{0}$ using the ellipsoid method with an $\varepsilon > 0$ error? (Suppose both (P) and its dual (D) are superconsistent.)

   I will create a new linear program of higher dimension that satisfies both the primal and the dual. Consider the set $S = \left\{\mathbf{z} = [\mathbf{x}, \mathbf{y}]^T : A\mathbf{x} \leq \mathbf{b}, A^T\mathbf{y} \geq \mathbf{c}, \mathbf{x} \geq 0, \mathbf{y} \geq 0, \mathbf{c}^T\mathbf{x} \leq \mathbf{b}^T\mathbf{y} + \epsilon\right\}$. This set of vectors is a polytope that contains feasible solutions to both the primal and the dual linear program. Since (P) and (D) are superconsistent there exists at least one vector $\mathbf{x}$ that optimizes (P) and one vector $\mathbf{y}$ that optimizes (D). Therefore there is at least one vector $\mathbf{z} \in S$. The last constraint in this set, $\mathbf{c}^T\mathbf{x} \leq \mathbf{b}^T\mathbf{y} + \epsilon$ allows for other solutions that are within ϵ of the optimal solution. This allows guarantees that there are multiple vectors in S, or that S is an actual polytope and not a single point.

   In order to solve the initial linear program, the ellipsoid method can be applied to the polytope S. Once the ellipsoid method has found a feasible solution, that solution must be within ϵ of the optimal solution.