Γραμμική & Συνδυαστική Βελτιστοποίηση

ΕΡΓΑΣΙΑ #1

Κυριάκος Στρατάκος – up1072704

up1072704@upnet.gr

ΑΣΚΗΣΗ1

Στόχος της Άσκησης 1 είναι η γραφική επίλυση ενός προβλήματος Γ.Π.

Οι μεταβλητές απόφασης του συστήματος είναι η x1,x2 με περιορισμούς

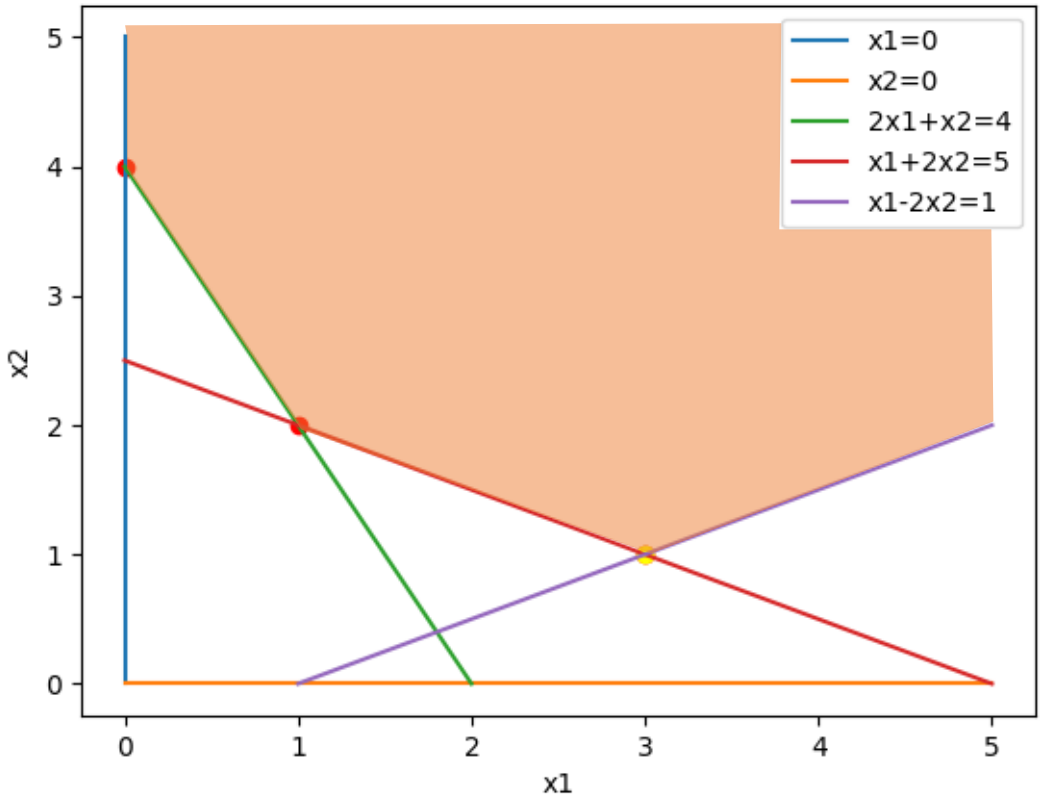
$$2x1 + x2 \geq 4$$

$$x1 + x2 \geq 5$$

$$x1 - 2x2 \leq 1$$

$$x1, x2 \geq 0$$

Με χρήση της python παριστάνουμε την εφικτή περιοχή η οποία όπως παρατηρούμε είναι ανοιχτή. Σχεδιάζονται επιπλέον όλες οι εφικτές κορυφές της εφικτής περιοχής.



Εικόνα 1 Εφικτή περιοχή προβλήματος πάνω απο την πράσινη, κόκκινη και μωβ γραμμή, δεξιά απο την μπλε.

Οι αντικειμενικές συναρτήσεις που θέλουμε να βελτιστοποιήσουμε είναι οι

$$maxZ1 = 2x1 - 5x2$$

$$maxZ2 = 2x1 - 4x2$$

$$maxZ3 = 2x1 - 3x2$$

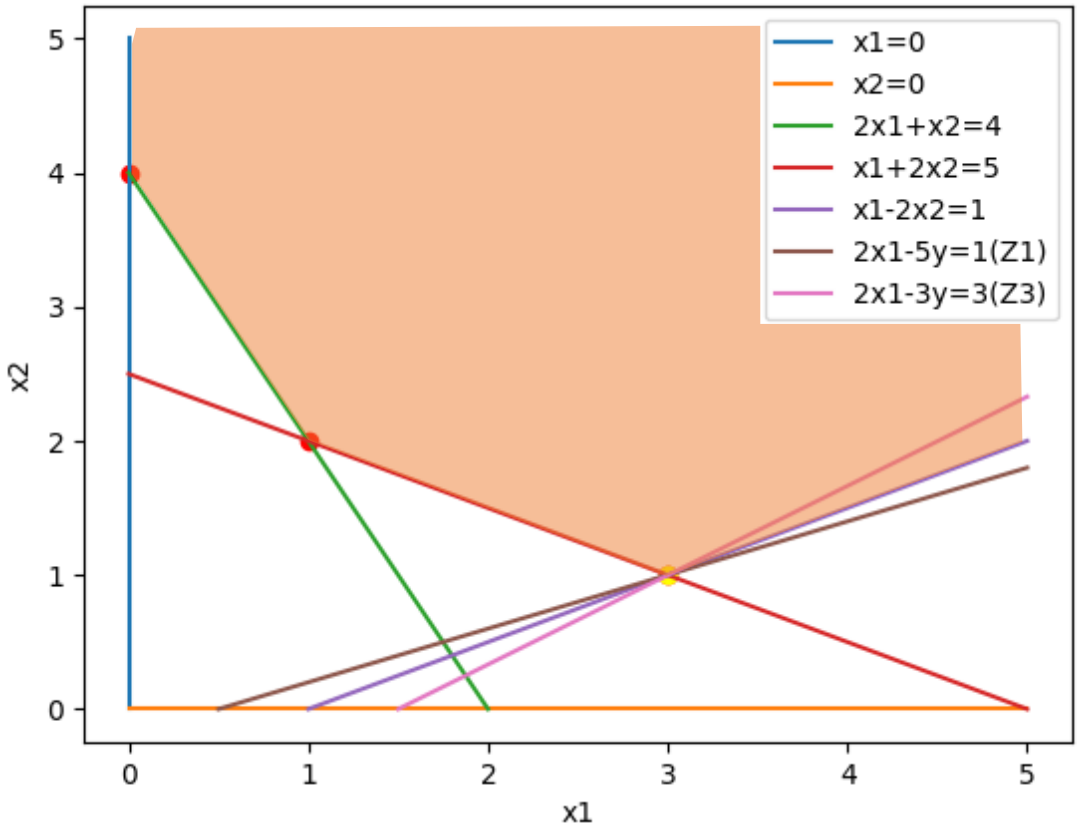
Απ την θεωρία γνωρίζουμε πως η βέλτιστη λύση (ένα υπάρχει) θα βρίσκεται σε κορυφή της εφικτής περιοχής.

Παρατηρούμε λοιπόν πως η βέλτιστη λύση για την Z1 βρίσκεται στην κορυφή (x1,x2)=(3,1) με Z1=1.

Η βέλτιστη λύση της Z2 ικανοποιείται απ' όλα τα σημεία της ευθείας x1-2x2=1 και έχει τιμή Z2 = 2.

Τέλος παρατηρούμε πως δεν υπάρχει βέλτιστη κορυφή για την Z3 και πως η μέγιστη τιμή της είναι το άπειρο.

Μπορούμε να δούμε σχηματικά τις Z1,Z2 (ταυτίζεται με την x1-2x2=1) με χρήση της python.



Ο κώδικας python που φτιάχνει τα παραπάνω διαγράμματα παρουσιάζεται παρακάτω:

```
from turtle import color
import matplotlib.pyplot as plt
import numpy as np
# vriskei tin veltisti korifi gia mia Z kai tin xrwmatizei kitrini
def solver(z,xpoints,ypoints):
    maxx = -1000
    for i in range(len(xpoints)):
        if z(xpoints[i],ypoints[i])>maxx:
            maxx = z(xpoints[i],ypoints[i])
            point = xpoints[i],ypoints[i]
    plt.scatter([point[0]],[point[1]],color='yellow')
    return(maxx,point)
```

```
#antikeimenikes synartiseis
def z1(x,y):
    return 2*x-5*y

def z2(x,y):
    return 2*x-4*y

def z3(x,y):
    return 2*x-3*y

plt.plot([0,0],[0,5],label='x1=0')
plt.plot([0,5],[0,0],label='x2=0')


plt.plot([0,2],[4,0],label='2x1+x2=4') #2x1+x2=4
plt.plot([0,5],[2.5,0],label='x1+2x2=5') #x1+2x2-5=0
plt.plot([1,5],[0,2],label='x1-2x2=1') #x1-2x2-1=0
#lyseis
plt.plot([0.5,5],[0,1.8],label='2x1-5y=1(Z1)') #2x1-5y=1
plt.plot([1.5,5],[0,2.33],label='2x1-3y=3(Z3)') #2x1-5y=1
xpoints =[0,1,3]
ypoints = [4,2,1]
plt.scatter(xpoints,ypoints,color='red')
# plt.scatter([3],[1],color="red")
leg = plt.legend(loc='upper right')
plt.xlabel("x1")
plt.ylabel("x2")
# vres tin veltisti korifi gia tin z1 kai xrwmatise tin kitrini
print(solver(z1,xpoints,ypoints))
plt.show()
```

ΑΣΚΗΣΗ2

Στόχος της Άσκησης 2 είναι η μοντελοποίηση και λύση ενός προβλήματος Γ.Π.

Ονομάζουμε x1 και x2 (μεταβλητές απόφασης) την ποσότητα που θα χρειαστούμε απ' τα υλικά G1 και G2 αντίστοιχα για την παραγωγή μιας μονάδας προϊόντος.

Στόχος μας η ελαχιστοποίηση του κόστους παραγωγής μονάδας

$$minZ = 6x_1 + 7.5x_2.$$

Μια μονάδα προϊόντος πρέπει να περιέχει τουλάχιστον 5.1 γρ. φυτικές ίνες, το πολύ 8.4 γρ. λιπαρά και το πολύ 10.8 γρ. πρωτεΐνης. Επιπλέον οι ποσότητες x1,x2 πρέπει να αθροίζονται σε μια μονάδα προϊόντος.

Μετατρέπουμε τους περιορισμούς που μας δίνονται σε ανισώσεις:

$$6x_1 + 4.5x_2 \geq 5.1$$

$$6x_1 + 9x_2 \leq 8.4$$

$$12x_1 + 9x_2 \leq 10.8$$

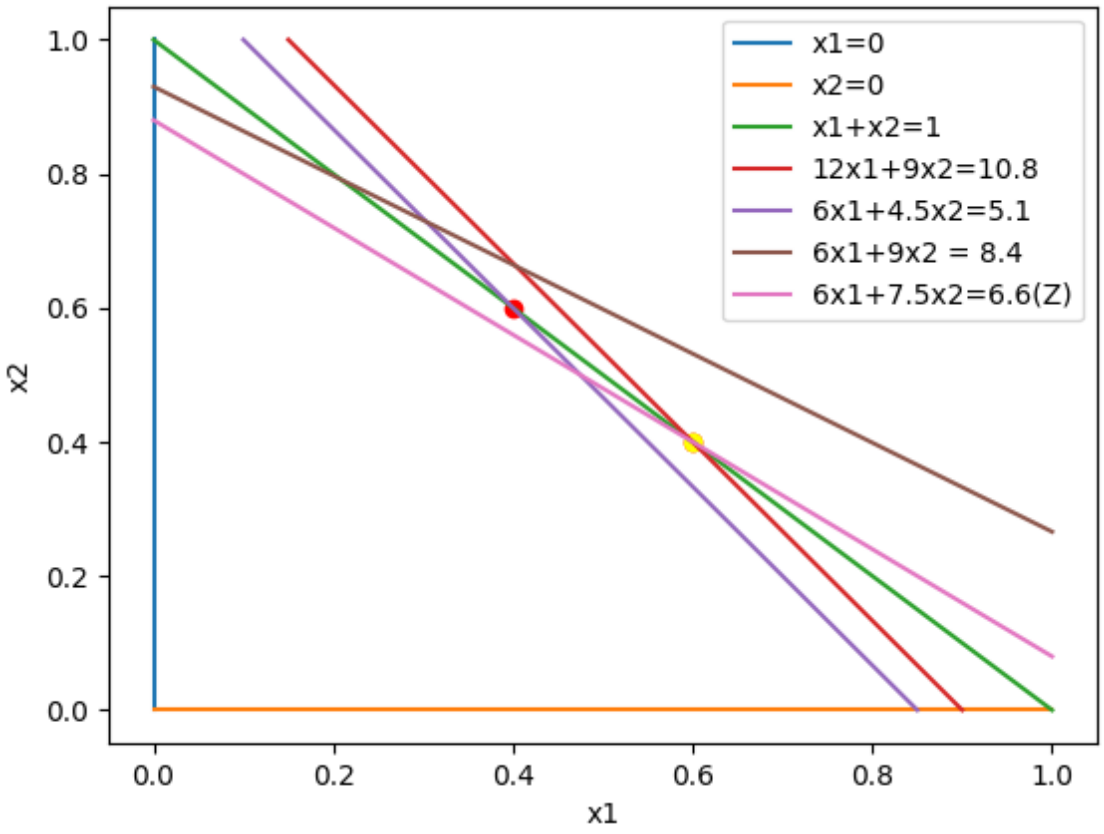
$$x_1 + x_2 = 1$$

$$x_1, x_2 \geq 0$$

Σχεδιάζουμε τις ευθείες των περιορισμών για να βρούμε την εφικτή περιοχή του προβλήματος (βλ. Γράφημα1).

Ο περιορισμός x1 + x2 = 1 είναι μια ευθεία γραμμή. Επομένως η βέλτιστη λύση του προβλήματος πρέπει να βρίσκεται πάνω σε αυτή την ευθεία και να ικανοποιεί και τους υπόλοιπους περιορισμούς. Είναι γνωστό από την θεωρία πως η βέλτιστη λύση θα βρίσκεται πάνω σε μια εφικτή κορυφή των περιορισμών. Εφικτές κορυφές είναι οι κορυφές που ικανοποιούν όλους τους περιορισμούς. Στο συγκεκριμένο παράδειγμα είναι η τομή των ευθειών

$$\begin{Bmatrix} x1 + x2 = 1 \\ 6x1 + 9x2 = 10.8 \end{Bmatrix} \text{ και } \begin{Bmatrix} x1 + x2 = 1 \\ 12x1 + 9x2 = 10.8 \end{Bmatrix}.$$



Εικόνα 2 Ευθείες περιορισμών και εφικτές κορυφές του προβλήματος. Η κίτρινη κορυφή είναι η βέλτιστη.

Παρατηρούμε πως οι συνάρτηση κόστους ελαχιστοποιείται για x1 = 0.6 και x2=0.4. Το τελικό κόστος παραγωγής μιας μονάδας προϊόντος είναι Z = 6.6.

Ο κώδικας python που δημιουργεί το διάγραμμα και υπολογίζει την βέλτιστη λύση:

```
from turtle import color
import matplotlib.pyplot as plt
```

```
def solver(xpoints,ypoints):
    minn = 1000
    #antikeimeniki sinartisi
    def z(x,y):
        return 6*x+7.5*y
    #ypologismos veltistis korifis
    for i in range(len(xpoints)):
        if z(xpoints[i],ypoints[i])<minn:
            minn = z(xpoints[i],ypoints[i])
            point = xpoints[i],ypoints[i]
    plt.scatter([point[0]],[point[1]],color='yellow')
    return(minn,point)


#periorismoi
plt.plot([0,0],[0,1],label="x1=0")
plt.plot([0,1],[0,0],label="x2=0")
plt.plot([0,1],[1,0],label="x1+x2=1")
plt.plot([0.9,0.15],[0,1],label="12x1+9x2=10.8") #12x1+9x2=10.8
plt.plot([0.85,0.1],[0,1],label='6x1+4.5x2=5.1') #6x1+4.5x2=5.1
plt.plot([1,0],[0.266,0.93],label='6x1+9x2 = 8.4') #6x1+9x2 = 8.4
plt.plot([0.9,0.15],[0,1],label="12x1+9x2=10.8") #12x1+9x2=10.8
#lysi
plt.plot([1,0],[0.08,0.88],label="6x1+7.5x2=6.6(Z)") #12x1+9x2=10.8
#efikta simia tomis periorismwn
xpoints =[0.4,0.6]
ypoints = [0.6,0.4]

#labels kai setup gia to diagramma
leg = plt.legend(loc='upper right')
plt.xlabel("x1")
plt.ylabel("x2")
#plot simeiwn korifwn
plt.scatter(xpoints,ypoints,color='red')

print(solver(xpoints,ypoints))
plt.show()
```

ΑΣΚΗΣΗ3

Στόχος της Άσκησης 3 είναι η μοντελοποίηση ενός προβλήματος Γ.Π.

Μια εταιρία έχει χωρίσει το 24ωρο σε 10 ζώνες ωρών. Σε κάθε ζώνη ώρας απαιτεί να δουλεύουν συγκεκριμένοι εργαζόμενοι. Για παράδειγμα στην ζώνη 18.00-20.00 πρέπει να δουλεύουν τουλάχιστον 82 εργαζόμενοι. Επιπλέον κάθε εργαζόμενος μπορεί να δουλεύει σε 5 βασικές βάρδιες όπως αναφέρεται στην εκφώνηση της άσκησης. Κάθε βάρδια κοστίζει διαφορετικό ποσό για την εταιρία.

Στόχος είναι η επιλογή των εργαζομένων που δουλεύουν σε κάθε βάρδια ώστε να ελαχιστοποιηθεί το συνολικό ημερήσιο κόστος της εταιρίας.

Επιλέγουμε ως μεταβλητές απόφασης τον αριθμό των εργαζομένων που δουλεύουν σε κάθε βάρδια. Έτσι έχουμε

x1: εργαζόμενοι που δουλεύουν στην βάρδια 6.00- 14.00

x2: εργαζόμενοι που δουλεύουν στην βάρδια 8.00- 16.00

x3: εργαζόμενοι που δουλεύουν στην βάρδια 12.00- 20.00

x4: εργαζόμενοι που δουλεύουν στην βάρδια 16.00- 00.00

x5: εργαζόμενοι που δουλεύουν στην βάρδια 22.00- 6.00

x1, x2, x3, x4, x5 θετικοί ακέραιοι.

Σε κάθε ζώνη ωρών πρέπει να δουλεύει τουλάχιστον ένας αριθμός ατόμων. Έτσι όταν οι βάρδιες επικαλύπτονται πρέπει το άθροισμα των εργαζομένων που δουλεύουν στις 2 βάρδιες να είναι μεγαλύτερο από τον ελάχιστο αριθμό ατόμων.

Έτσι καταλήγουμε στους εξής περιορισμούς:

$$x1 \geq 48$$

$$x1 + x2 \geq \max(79,65) = 79$$

$$x1 + x2 + x3 \geq 87$$

$$x2 + x3 \geq 64$$

$$x3 + x4 \geq \max(73,82) = 82$$

$$x4 \geq 43$$

$$x4 + x5 \geq 52$$

$$x5 \geq 15$$

Στη ζώνη 6-8 πρέπει να βρίσκονται στην εταιρία τουλάχιστον 48 άτομα. Στην ζώνη αυτή δουλεύουν μόνο οι εργαζόμενοι της πρώτης βάρδιας. Έτσι λοιπόν βγαίνει ο πρώτος περιορισμός πως οι εργαζόμενοι της πρώτης βάρδιας πρέπει να είναι τουλάχιστον 48.

Στις ζώνες 8-10, 10-12 δουλεύουν μαζί οι εργαζόμενοι από την πρώτη και την δεύτερη βάρδια. Στις 8-10 χρειάζονται τουλάχιστον 79 εργαζόμενοι και στις 10-12 τουλάχιστον 65. Το άθροισμα x1+x2 πρέπει να είναι μεγαλύτερο από το max(79,65). Το ίδιο γίνεται και σε όλες τις άλλες ζώνες που έχουμε επικαλύψεις βαρδιών.

Η αντικειμενική συνάρτηση κόστους είναι το ημερήσιο κόστος της εταιρίας και προκύπτει από τον πίνακα ημερησίου κόστους ανά εργαζόμενο ανά βάρδια.

$$minZ = 170x1 + 160x2 + 175x3 + 180x4 + 195x5$$

ΑΣΚΗΣΗ4

(Π1) Η τομή δύο κυρτών συνόλων είναι κυρτό σύνολο.

Έστω δύο κυρτά σύνολα $A, B \subset R^n$.

$$A \cap B \in A, B \; \alpha\rho\alpha$$

$$\forall x1, x2 \in A \cap B, \lambda \in [0,1], \lambda x1 + (1-\lambda)x2 \in A$$

$$\forall x1, x2 \in A \cap B, \lambda \in [0,1], \lambda x1 + (1-\lambda)x2 \in B \ \ και\ άρα$$

$$\forall x1, x2 \in A \cap B, \lambda \in [0,1], \lambda x1 + (1-\lambda)x2 \in A \cap B\ .$$

Το σύνολο $A \cap B$ είναι κυρτό.

(Π2) Το σύνολο $\Omega = \{(x,y) \in R^2 | x^2 + y^2 \leq 1\}$ είναι κυρτό .

Έστω P1,P2 εντός του μοναδιαίου κύκλου.

$$P1 = x1\,\hat{\imath} + y1\,\hat{\jmath}$$

$$P2 = x2\,\hat{\imath} + y2\,\hat{\jmath}$$

$$x1^2 + y1^2 \leq 1\ , x2^2 + y2^2 \leq 1$$

Πρέπει

$$P = \lambda P1 + (1-\lambda)P2\ \in \Omega, \lambda \in [0,1]$$

$$P = (\lambda x1 + (1-\lambda)x2)\,\hat{\imath} + (\lambda y1 + (1-\lambda)y2)\,\hat{\jmath}$$

Αρκεί να δείξω $|P| \leq 1$.

$$(\lambda x1 + (1-\lambda)x2)^2 + (\lambda y1 + (1-\lambda)y2)^2 \leq 1$$

$$\lambda^2(x1^2 + y1^2) + (1-\lambda)^2(x2^2 + y2^2) + 2\lambda(1-\lambda)(x1x2 + y1y2) \leq 1$$

όμως

$$\lambda^2(x1^2 + y1^2) + (1-\lambda)^2(x2^2 + y2^2) \leq \lambda^2 + (1-\lambda)^2$$

Επομένως αρκεί

$$\lambda^2 + (1-\lambda)^2 + 2\lambda(1-\lambda)(x1x2 + y1y2) \leq 1$$

$$2\lambda(1-\lambda)(x1x2 + y1y2 - 1) \leq 0$$

Ισχύει ότι:

$$0 \leq 2\lambda(1-\lambda) \leq 1$$

και

$$x1x2 + y1y2 - 1 \leq x1x2 + y1y2 - \frac{x1^2 + y1^2 + x2^2 + y2^2}{2} =$$
$$= -\frac{1}{2}(x1 - x2)^2 - \frac{1}{2}(y1 - y2)^2 \leq 0.$$

Επομένως η πρόταση ισχύει.

(Π3) Αν $x^1, x^2, \dots, x^m \in X, όπου\ X \subset R^n κυρτό$, τότε κάθε κυρτός συνδυασμός τους ανήκει στο Χ.

m=2:

$$x1, x2 \in X \Leftrightarrow\ x =\ \lambda x1 + (1-\lambda)x2\ \in X, \lambda \in [0,1]$$

Συνεπώς για $\lambda 1 = \lambda\ , \lambda 2 = 1 - \lambda,$

$$x = \lambda 1 x1 + \lambda 2 x2 \in X, \lambda 1, \lambda 2 \geq 0 \text{ και } \lambda 1 + \lambda 2 = 1.$$

m=3:

$$x1, x2 \in X \text{ άρα } x' = \lambda x1 + (1 - \lambda)x2 \in X, \lambda \in [0,1]$$

$$x', x3 \in X \text{ άρα } x = \lambda' x' + (1 - \lambda')x3 \in X, \lambda' \in [0,1]$$

$$x = \lambda' \lambda\, x1 + \lambda'(1 - \lambda)x2 + (1 - \lambda')x3 =$$

$$= \lambda 1 x1 + \lambda 2 x2 + \lambda 3 x3 \in X, \lambda 1, \lambda 2, \lambda 3 \geq 0$$

και

$$\lambda 1 + \lambda 2 + \lambda 3 = \lambda' \lambda + \lambda'(1 - \lambda) + 1 - \lambda' = 1.$$

Έστω πως η πρόταση ισχύει για $m = k - 1.$

Θα δείξουμε πως η πρόταση ισχύει για m = k.

$$x = \lambda 1 x_1 + \lambda 2 x_2 + \cdots + \lambda_{k-1} x_{k-1} \in X\ , \sum_{i=1}^{k-1} \lambda_i = 1.$$

$$x_k \in X \text{ και άρα}$$

$$x' = \lambda x + (1 - \lambda)x_k \in X, \lambda \in [0,1].$$

$$x' = \lambda\, \lambda 1\, x_1 + \lambda\, \lambda 2\, x_2 + \cdots + \lambda\, \lambda_{k-1} x_{k-1} + (1 - \lambda)x_k.$$

$$\lambda\, \lambda 1 + \lambda\, \lambda 2 + \cdots + \lambda\, \lambda_{k-1} + 1 - \lambda = \lambda \left(\sum_{i=1}^{k-1} \lambda_i \right) - \lambda + 1 = 1.$$

Επομένως η πρόταση ισχύει για κάθε κυρτό συνδυασμό..

ΑΣΚΗΣΗ5

Στόχος της Άσκησης 5 είναι η επίλυση του παρακάτω προβλήματος ΓΠ:

$$\min Z = 12x1 - 10x2 - 30x3$$

όταν

$$-3x1 + 2x2 + 8x3 \leq 17$$

$$-x1 + x2 + 3x3 \leq 9$$

$$-2x1 + x2 + 8x3 \leq 16$$

$$x1, x2, x3 \geq 0.$$

Αρχικά, ζητείται να βρεθούν όλες οι κορυφές του που δημιουργούνται από τις τομές των υπερεπιπέδων του πολύτοπου των περιορισμών και να ξεχωρίσουμε ποιες απ' αυτές ανήκουν στο πολύτοπο των εφικτών λύσεων.

Μια κορυφή δημιουργείται από 3 επίπεδα. Για τον λόγο αυτό λύνουμε όλους τους πιθανούς συνδυασμούς τριών από τις έξι εξισώσεις (20 συνδυασμοί). Αν τουλάχιστον 2

τριάδες δώσουν την ίδια κορυφή τότε η κορυφή είναι εκφυλισμένη, δημιουργείται δηλαδή από περισσότερα από 3 επίπεδα.

Ακολουθεί ο κώδικας python που εκτελεί αυτή την διαδικασία.

Ορίζουμε τους πίνακες των εξισώσεων των επιπέδων του πολύτοπου:

```
# −3x1 + 2x2 + 8x3 ≤ 17
# −x1 + x2 + 3x3 ≤ 9
# −2x1 + x2 + 8x3 ≤ 16
# x1, x2, x3 ≥ 0

A = np.array([[-3,2,8],[-1,1,3],[-2,1,8]])
I = np.eye(3)
A = np.concatenate((A,-I))
B = np.array([17,9,16,0,0,0])
```

Οι εξισώσεις

$$x1, x2, x3 \geq 0,$$

μετατρέπονται σε

$$-x1, -x2, -x3 \leq 0,$$

Ώστε να ισχύει:

$$Ax \leq B.$$

Βρίσκουμε όλες τις κορυφές που δημιουργούν τα υπερεπίπεδα:

```
it = 0
for i in range(len(A)):
    for j in range(i+1,len(A)):
        for k in range(j+1,len(A)):
            it+=1
            print(it)
            print(A[[i,j,k]],B[[i,j,k]])
            try:
                x = linalg.solve(A[[i,j,k]],B[[i,j,k]])
                y = np.dot(A,x)
                if np.all(y<=B):
                    print('valid')
                else:
                    print('invalid')
                print(x)

            except:
                print("adinato sistima")
            print("\n")
```

Με τις 3 επαναλήψεις τα i,j,k θα «σαρώσουν» όλους τους πιθανούς συνδυασμούς (i,j,k).

Η δομή try-except μας προφυλάσσει απ' το ενδεχόμενο το σύστημα να είναι αδύνατο. Σε αυτή την περίπτωση το πρόγραμμα θα έβγαζε error. Με το try-except όταν το

πρόγραμμα βγάλει error εκτελώντας τις εντολές του try,διακόπτει την διαδικασία και εκτελεί τις εντολές που βρίσκονται στο except, δηλαδή, τυπώνει «αδύνατο σύστημα».

Σε κάθε επανάληψη το πρόγραμμα προσπαθεί να λύσει το σύστημα των εξισώσεων που σχηματίζουν οι i,j,k γραμμές του πίνακα [A|B]. Σε περίπτωση επιτυχίας εκτελείται η πράξη Ax και ελέγχεται αν όλες οι τιμές του παραγόμενου πίνακα είναι μικρότερες απ' τις αντίστοιχες τιμές του B. Ελέγχεται δηλαδή αν η κορυφή είναι κορυφή του πολύτοπου των εφικτών λύσεων.

```
A:
[[-3.  2.  8.]
 [-1.  1.  3.]
 [-2.  1.  8.]]
B:
[17  9 16]
valid
x:
[6.33333333 7.33333333 2.66666667]
```

```
2
A:
[[-3.  2.  8.]
 [-1.  1.  3.]
 [-1. -0. -0.]]
B:
[17  9  0]
invalid
x:
[-0.  10.5 -0.5]
```

```
3
A:
[[-3.  2.  8.]
 [-1.  1.  3.]
 [-0. -1. -0.]]
B:
[17  9  0]
invalid
x:
[21. -0. 10.]
```

```
4
A:
[[-3.  2.  8.]
 [-1.  1.  3.]
 [-0. -0. -1.]]
B:
[17  9  0]
valid
x:
[ 1. 10. -0.]
```

```
5
A:
[[-3.  2.  8.]
 [-2.  1.  8.]
 [-1. -0. -0.]]
B:
[17 16  0]
invalid
x:
[-5.92118946e-16  1.00000000e+00  1.875
```

```
6
A:
[[-3.  2.  8.]
 [-2.  1.  8.]
 [-0. -1. -0.]]
B:
[17 16  0]
invalid
x:
[-1.   -0.    1.75]
```

```
7
A:
[[-3.  2.  8.]
 [-2.  1.  8.]
 [-0. -0. -1.]]
B:
[17 16  0]
invalid
x:
[-15. -14.  -0.]
```

```
8
A:
[[-3.  2.  8.]
 [-1. -0. -0.]
 [-0. -1. -0.]]
B:
[17  0  0]
invalid
x:
[-0.    -0.     2.125]
```

```
9
A:
[[-3.  2.  8.]
 [-1. -0. -0.]
 [-0. -0. -1.]]
B:
[17  0  0]
valid
x:
[-0.   8.5 -0. ]
```

```
10
A:
[[-3.  2.  8.]
 [-0. -1. -0.]
 [-0. -0. -1.]]
B:
[17  0  0]
invalid
x:
[-5.66666667 -0.         -0.        ]
```

```
11
A:
[[-1.  1.  3.]
 [-2.  1.  8.]
 [-1. -0. -0.]]
B:
[ 9 16  0]
invalid
x:
[-4.4408921e-16  4.8000000e+00  1.4000000e+00]
```

```
12
A:
[[-1.  1.  3.]
 [-2.  1.  8.]
 [-0. -1. -0.]]
B:
[ 9 16  0]
invalid
x:
[-12.  -0.  -1.]
```

```
13
A:
[[-1.  1.  3.]
 [-2.  1.  8.]
 [-0. -0. -1.]]
B:
[ 9 16  0]
invalid
x:
[-7.  2. -0.]
```

```
14
A:
[[-1.  1.  3.]
 [-1. -0. -0.]
 [-0. -1. -0.]]
B:
[9 0 0]
invalid
x:
[-0. -0.  3.]
```

```
15
A:
[[-1.  1.  3.]
 [-1. -0. -0.]
 [-0. -0. -1.]]
B:
[9 0 0]
invalid
x:
[-0.  9. -0.]
```

```
16
A:
[[-1.  1.  3.]
 [-0. -1. -0.]
 [-0. -0. -1.]]
B:
[9 0 0]
invalid
x:
[-9. -0. -0.]


17
A:
[[-2.  1.  8.]
 [-1. -0. -0.]
 [-0. -1. -0.]]
B:
[16  0  0]
valid
x:
[-0. -0.  2.]


18
A:
[[-2.  1.  8.]
 [-1. -0. -0.]
 [-0. -0. -1.]]
B:
[16  0  0]
invalid
x:
[-0. 16. -0.]


19
A:
[[-2.  1.  8.]
 [-0. -1. -0.]
 [-0. -0. -1.]]
B:
[16  0  0]
invalid
x:
[-8. -0. -0.]


20
A:
[[-1. -0. -0.]
 [-0. -1. -0.]
 [-0. -0. -1.]]
B:
[0 0 0]
valid
x:
[-0. -0. -0.]
```

Παρατηρούμε πως μόνο οι κορυφές 20,17,9,4 και 1 είναι κορυφές του πολύτοπου των εφικτών λύσεων.

Παρατηρώντας τις λύσεις προσεχτικά βλέπουμε ότι και η 5 είναι εφικτή λύση απλώς το x1 = -5.92x10^-16 < 0 και άρα αδύνατο ενώ ισχύει ότι x1 = 0. Απλώς ο υπολογιστής έχει κάνει λάθος στρογγυλοποίηση.

Με αντίστοιχο τρόπο δουλεύουμε όταν προσθέτουμε τις μεταβλητές χαλάρωσης στο πρόγραμμα μας. Αυτή την φορά ο πίνακας A έχει διαστάσεις 3x6 (3 ανισώσεις άρα 3 slack variables και 3 μεταβλητές χαλάρωσης).

```
                xyz = np.zeros(6)
                B = np.dot(linalg.inv(A[:,[i,j,k]]),b)
                print(i,j,k)
                xyz[[i,j,k]] = B
                xyz = xyz.reshape(-1,1)
                print(xyz[[i,j,k]])
                val = np.dot(C,xyz)
                print('value:',val)
                for num in B:
                    if num < 0 :
                        print("invalid")
                        break
                else:
                    print("valid!")
                    if(val<minn):
                        minn = val
                        x = xyz[0][0]
                        y = xyz[1][0]
                        z = xyz[2][0]
```

Στον πίνακα xyz αποθηκεύονται οι τιμές που παίρνουν όλες οι μεταβλητές. Ορίζουμε τον πίνακα B ως:

$$B = A^{-1}b,$$

Όπου ο A είναι πίνακας 3x3 με στήλες τις μεταβλητές της βάσης μας. Αν κάποια απ' αυτές τις μεταβλητές είναι μικρότερη του 0 τότε η λύση δεν είναι εφικτή. Αλλιώς βρίσκουμε την τιμή του Z ως:

$$Z = C^T xyz$$

και κρατάμε την μεγαλύτερη.

Στο συγκεκριμένο πρόβλημα το Z ελαχιστοποιείται για

$$(x1, x2, x3) = (1,10,0), minZ = -88$$

Παρατηρούμε πως κάθε λύση του προβλήματος αντιστοιχίζεται σε μια κορυφή. Στην άσκηση δεν έχουμε εκφυλισμένες κορυφές.

```
iteration 1 : basis: 0 1 2
[x1,x2,x3]= [6.33333333 7.33333333 2.66666667]
value= -77.33333333333336
valid!
iteration 2 : basis: 0 1 3
[x1,x2,x3]= [-7.  2.  0.]
value= -104.0
invalid
iteration 3 : basis: 0 1 4
[x1,x2,x3]= [-15. -14.   0.]
value= -39.99999999999994
invalid
iteration 4 : basis: 0 1 5
[x1,x2,x3]= [ 1. 10.  0.]
value= -88.0
valid!
iteration 5 : basis: 0 2 3
[x1,x2,x3]= [-12.   0.  -1.]
value= -114.0
invalid
iteration 6 : basis: 0 2 4
[x1,x2,x3]= [-1.    0.    1.75]
value= -64.5
invalid
iteration 7 : basis: 0 2 5
[x1,x2,x3]= [21.  0. 10.]
value= -48.000000000000085
invalid
iteration 8 : basis: 0 3 4
[x1,x2,x3]= [-8.  0.  0.]
value= -96.0
invalid
iteration 9 : basis: 0 3 5
[x1,x2,x3]= [-9.  0.  0.]
value= -108.0
invalid
iteration 10 : basis: 0 4 5
[x1,x2,x3]= [-5.66666667  0.          0.        ]
value= -68.0
invalid
iteration 11 : basis: 1 2 3
[x1,x2,x3]= [0.  4.8 1.4]
value= -90.00000000000001
invalid
iteration 12 : basis: 1 2 4
[x1,x2,x3]= [0.    1.    1.875]
value= -66.25
valid!
iteration 13 : basis: 1 2 5
[x1,x2,x3]= [ 0.  10.5 -0.5]
value= -90.0
invalid
iteration 14 : basis: 1 3 4
[x1,x2,x3]= [ 0. 16.  0.]
value= -160.0
invalid
iteration 15 : basis: 1 3 5
[x1,x2,x3]= [0. 9. 0.]
value= -90.0
invalid
iteration 16 : basis: 1 4 5
[x1,x2,x3]= [0.  8.5 0. ]
value= -85.0
valid!
iteration 17 : basis: 2 3 4
[x1,x2,x3]= [0. 0. 2.]
value= -60.0
valid!
iteration 18 : basis: 2 3 5
[x1,x2,x3]= [0. 0. 3.]
value= -90.0
invalid
iteration 19 : basis: 2 4 5
[x1,x2,x3]= [0.    0.    2.125]
value= -63.75
invalid
iteration 20 : basis: 3 4 5
[x1,x2,x3]= [0. 0. 0.]
value= 0.0
valid!
minimum solution:
(x1,x2,x3) = ( 1.0 10.0 0.0 )
Z= -88.0
```

Εικόνα 3 Έξοδος του προγράμματος. Σε κάθε επανάληψη μας δείχνει τις μεταβλητές βάσης μας (x1=0,x2=1,..) τις τιμές των μεταβλητών απόφασης καθώς και την τιμή της αντικειμενικής συνάρτησης

ΑΣΚΗΣΗ6

Στόχος της άσκησης είναι η επίλυση ενός προβλήματος Γ.Π. με χρήση του αλγορίθμου Simplex.

$$\min Z = -x1 - 4x2 - 5x3$$

όταν

$$x1 + 2x2 + 3x3 \leq 2$$

$$3x1 + x2 + 2x3 \leq 2$$

$$2x1 + 3x2 + x3 \leq 4$$

$$x1, x2, x3 \geq 0$$

Για την επίλυση αυτού του προβλήματος υλοποιήθηκε ένα πρόγραμμα python που αυτοματοποιεί την διαδικασία του Simplex.

$$\min Z = \max(-Z) = x1 + 4x2 + 5x3$$

Αρχικά γίνεται η αρχικοποίηση των πινάκων C,b καθώς και προστίθενται οι slack variables στον πίνακα B. B = [A|I] (στο πρόγραμμα ο πίνακας BIb έχει ονομαστεί A). Ο πίνακας z αποθηκεύει τα κέρδη κάθε μεταβλητής και ο πίνακας C_z = C-z είναι ο πίνακας βάση του οποίου επιλέγεται ποια μεταβλητή θα γίνει βασική. Ξεκινάμε με βάση BI = {x4,x5,x6}

```
 #sintelestes antikeim synart
C = np.array([1,4,5,0,0,0])
b = np.array([2,2,4])

#tablaue sintelestwn
A = np.array([[1,2,3],
              [3,1,2],
              [2,3,1],
             ])
#[A|I|b]
A = np.concatenate((A,np.eye(len(A)),b.reshape(len(b),-1)),axis=1)


z = np.zeros(len(A[0])).reshape(1,-1)

#basic vars
basis = [3,4,5]
```

Σε κάθε επανάληψη:

Επιλέγουμε ποια μεταβλητή θα γίνει βασική. Επιλέγουμε το στοιχείο με την μέγιστη τιμή στο C-z.

```
col = list(C_z[0]).index(max(C_z[0]))
```

Στην συνέχεια επιλέγουμε ποια μεταβλητή θα αντικαταστήσουμε. Επιλέγουμε αυτή με τον μικρότερο λόγο

$$\frac{(B^{-1}b)k}{(B^{-1}N)ka}.$$

Δηλαδή την μεταβλητή με το μικρότερο θετικό λόγο b[i]/A[i][col].

```
    #find the var that will be taken out
    minn = 100
    for i in range(len(A)):
        if(A[i][col]>0 and A[i][-1]/A[i][col]<minn):
```

```
            minn = A[i][-1]/A[i][col]
            row = i
```

Στην συνέχεια εκτελούμε την αλλαγή μεταβλητών και τις γραμμοπράξεις που χρειάζονται:

```
#allagi twn vasikwn metavlitwn
    basis[row] = col
    pivot = A[row,col]

    #grammoprajeis
    A[row,:] = A[row,:]/pivot

    for i in range(len(A)):
        div = A[i,col]
        if(i==row):continue
        A[i,:] = A[i,:]-div*A[row,:]
```

Βρίσκουμε τα νέα κέρδη των μεταβλητών:

```
    #evresi kerdwn z = C.T * A
    z = np.dot(C[basis].reshape(1,-1),A)
```

Αν όλα τα κέρδη είναι μικρότερα ή ίσα του μηδενός έχουμε φτάσει σε τελική λύση. Αλλιώς επαναλαμβάνουμε επιλέγοντας το στοιχείο με το μεγαλύτερο κέρδος για να μπει στην βάση.

```
    #sinthiki termatismou
    if(np.all(C_z<=0)):
        print("done")
        break
```

Το αποτέλεσμα του αλγορίθμου όταν τρέχουμε το πρόγραμμα παρουσιάζεται παρακάτω:

Στην αρχή κάθε iteration τυπώνεται το tableau, οι βασικές μεταβλητές, οι τιμές τους, το Z καθώς και το C-Z.

Επανάληψη 1:

BI = {x4,x5,x6}, Z=0 , (x1,x2,x3) = (0,0,0)

Επανάληψη 2:

BI = {x2,x5,x6}, Z=-3.33 , (x1,x2,x3) = (0,0,0.66)

Επανάληψη 3:

BI = {x1,x5,x6}, Z= -4, (x1,x2,x3) = (0,1,0)

```
iteration: 1
tablaue:
[[1. 2. 3. 1. 0. 0. 2.]
 [3. 1. 2. 0. 1. 0. 2.]
 [2. 3. 1. 0. 0. 1. 4.]]
basic vars
[3, 4, 5]
[2. 2. 4.]
profit:
0.0
C-z:
[[1. 4. 5. 0. 0. 0.]]
iteration: 2
tablaue:
[[ 0.33333333  0.66666667  1.          0.33333333  0.          0.
   0.66666667]
 [ 2.33333333 -0.33333333  0.         -0.66666667  1.          0.
   0.66666667]
 [ 1.66666667  2.33333333  0.         -0.33333333  0.          1.
   3.33333333]]
basic vars
[2, 4, 5]
[0.66666667 0.66666667 3.33333333]
profit:
3.333333333333333
C-z:
[[-0.66666667  0.66666667  0.         -1.66666667  0.          0.        ]]
iteration: 3
tablaue:
[[ 0.5  1.   1.5  0.5  0.   0.   1. ]
 [ 2.5  0.   0.5 -0.5  1.   0.   1. ]
 [ 0.5  0.  -3.5 -1.5  0.   1.   1. ]]
basic vars
[1, 4, 5]
[1. 1. 1.]
profit:
4.0
C-z:
[[-1.  0. -1. -2.  0.  0.]]
done
```

Figure 1 Έξοδος προγράμματος.

Πιο συγκεκριμένα μπορούμε να εξερευνήσουμε το adjacency graph των εφικτών λύσεων:

1: B = {3,4,5}, (x1,x2,x3) = (0,0,0), Z=0

2: B = {2,4,5}, (x1,x2,x3) = (0,0,0.66),

Z=-3.33

3: B = {1,4,5}, (x1,x2,x3) = (0,1,0), Z=-4

4: B = {3,0,5}, (x1,x2,x3) = (0.66,0,0),

Z=-0.66

5: B = {1,0,5}, (x1,x2,x3) = (0.4,0.8,0), Z=-3.6

6: B = {2,0,5}, (x1,x2,x3) = (0.28,0,0.57),

Z=-3.142

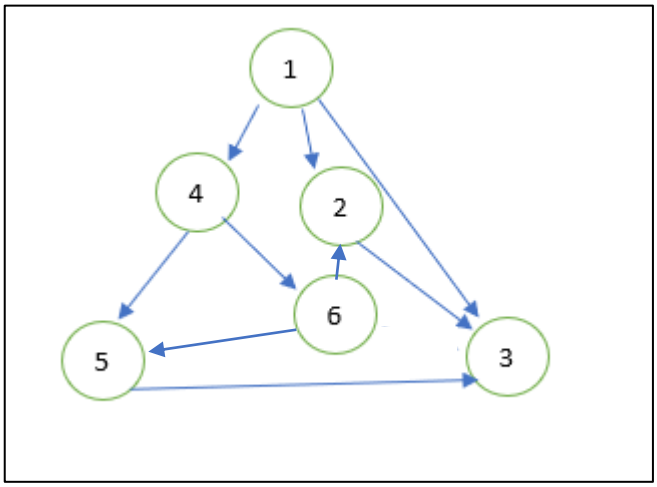


Figure 2 Adjacency graph εφικτών λύσεων