HW1 Linear Optimization 2022-2023

Created	@April 10, 2023 12:54 PM
Class	Γραμμικη Βελτιστοποιηση
Type	HW
Materials	
Reviewed	☐

Νικόλας Φιλιππάτος

AM: 1072754

Ασκηση 1

Άσκηση 1. Έστω πρόβλημα γραμμικού προγραμματισμού που έχει ως περιορισμούς τις παρακάτω ανισώσεις:

$$\begin{aligned} (\Pi 1) &\quad 2x_1 + x_2 \geq 4 \\ (\Pi 2) &\quad x_1 + 2x_2 \geq 5 \\ (\Pi 3) &\quad x_1 - 2x_2 \leq 1 \\ &\quad x_1, x_2 \geq 0 \end{aligned}$$

(α) Παραστήστε γραφικά την εφικτή περιοχή του προβλήματος καθώς και όλες τις κορυφές της. Περιγράψτε τη μορφή της εφικτής περιοχής.

(β) Λύστε το παραπάνω πρόβλημα γραφικά με κάθε μία από τις παρακάτω αντικειμενικές συναρτήσεις:

(i) $\max Z = 2x_1 - 5x_2$ $\quad (ii)$ $\max Z = 2x_1 - 4x_2$ $\quad (iii)$ $\max Z = 2x_1 - 3x_2$

Σε κάθε περίπτωση περιγράψτε αναλυτικά τη μορφή της λύσης, εφ' όσον υπάρχει.

ασκ1-εκφωνηση

a) Σχεδιαση Εφικτής περιοχης

Θα σχεδιάσουμε τις ευθείες

$y = -2x + 4$

$y = -0.5x + 2.5$

$y = 0.5x - 0.5$

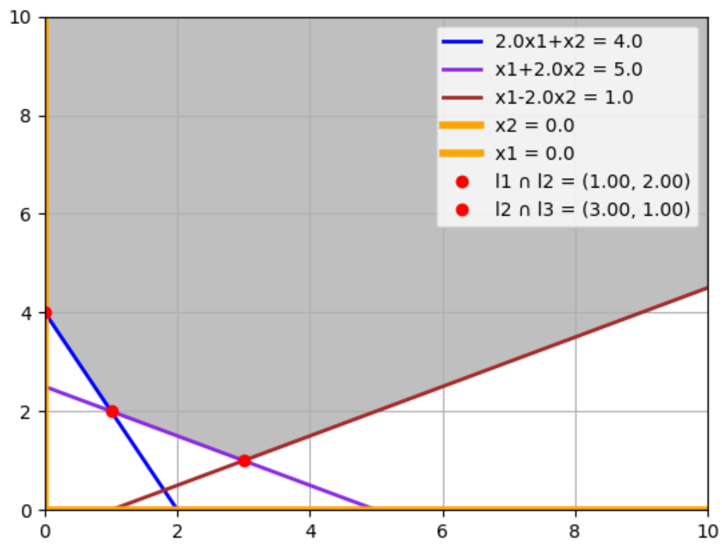
Οι οποίες προκύπτουν από τους περιορισμούς και θεωρησαμε $x_2 = y$ και $x_1 = x$

Η εφικτή περιοχή λύσεων , βρίσκεται πάνω απο τις ευθείες αυτές, και στο σχήμα μας αναπαριστάται απο τη γραμμοσκιασμένη περιοχή.

Παρατηρούμε ότι η εφικτή περιοχή δεν ειναι φραγμενη.

Κορυφες της περιοχης ειναι η τομη της:

- μπλε ευθειας με τον αξονα y στο σημειο (0,4)
- μπλε ευθειας με την μωβ ευθεια στο σημειο (1,2)
- μωβ ευθειας με την καφε ευθεια στο σημειο (3,1)



b) *Επιλυση των max*

$$max\ 2x_1 - 5x_2$$

θεωρουμε την αντικειμενικη συναρτηση $2x_1 - 5x_2 = c$

Και την σχεδιαζουμε για διαφορες τιμες του c. Θελουμε να μεγιστοποιησουμε την τιμη της αντικειμενικης συναρτησης, οποτε θα κοιταξουμε για θετικα c .

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/18a3e756-4683-4abc-b25f-f0caeef6fe3d/plt_line.py

class line for plotting and extra functions

```
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.colors as mcolors


class line:
    """ax1 + bx2 = c"""

    def __init__(self, a, b, c, name, legend_show=True) -> None:
        self.a, self.b, self.c = map(float, (a, b, c))
        self.name = name
        self.legend_show = legend_show

    def __str__(self):
        """returns the equation """
        return f"{str(self.a if self.a !=1 else '')+'x1' if self.a else ''}{'+' if self.a and self.b>0 else ''}{str(self.b if self.b !

    def intersection(self, line, plotting=True):
        """returns the intersection point of two lines, and plots it if plotting is True"""

        x_up = line.c * self.b - self.c * line.b
        x_down = line.a * self.b - self.a * line.b
        if x_down == 0:
            # print(f'{self.name} and {line.name} are parallel')
            return None

        x = x_up / x_down
        y = self.equation(x)

        if plotting:
            # plot the spot ,as a red dot with a label
            plt.plot(
                x, y, 'ro', label=f"{self.name} ∩ {line.name} = ({x:.2f}, {y:.2f})")
            if self.legend_show:
                plt.legend()
        return [x, y]

    def equation(self, x):
        """returns the y value of the line for a given x value"""
        return (-self.a * x + self.c)/self.b

    def reverse_equation(self, y):
        """returns the x value of the line for a given y value"""
        return (-self.b * y + self.c)/self.a

    def plot(self, x, color=None, lw=2, ms=12):
        """plots the line, and the equation as a label"""
        if self.b == 0:
            plt.axvline(x=self.c/self.a, label=f'{self.name}: {self}',
                        color=self.auto_color_chooser(color), linewidth=lw, markersize=ms)
        else:
            plt.plot(x, self.equation(x), label=f'{self.name}: {self}', color=self.auto_color_chooser(
                color), linewidth=lw, markersize=ms)
        # calls the plot settings function to show the legend
        if self.legend_show:
            plt.legend()

    def auto_color_chooser(self, color=None):
        """returns a color from the matplotlib color list, if color is not given, it returns the next color in the list, if color is g
        colors = sorted(mcolors.cnames)
```

```
        if not color and not hasattr(self, 'ind'):
            self.ind = 0
        elif not color and hasattr(self, 'ind'):
            self.ind = (self.ind+1) % len(colors)
        elif color and not hasattr(self, 'ind'):
            self.ind = colors.index(color) if color in colors else 0
        return colors[self.ind]
```

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/13970982-1154-407d-b452-1a4742af8f72/exerc-01.py

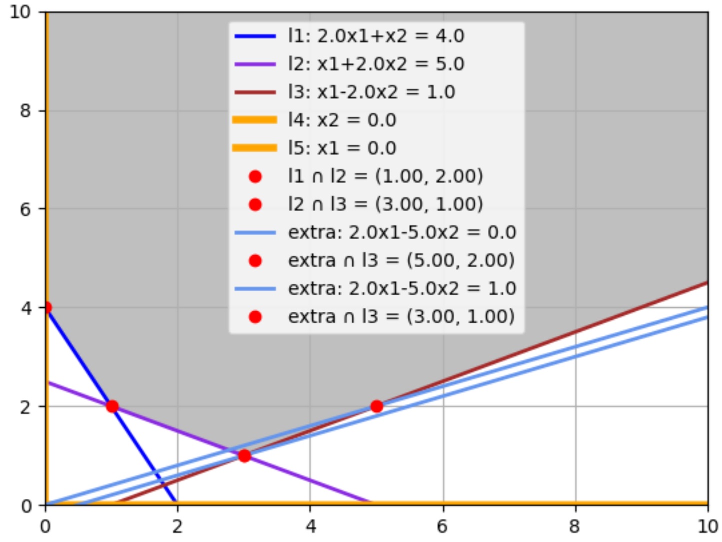
exercise 1 solution written in python

```
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.colors as mcolors

from plt_line import line


def plot_equations():

    # create the x axis points
    xAxis = np.linspace(-100, 100, 100000)

    # create the lines from the constraints
    l1 = line(2, 1, 4, "l1")
    l2 = line(1, 2, 5, "l2")
    l3 = line(1, -2, 1, "l3")
    l4 = line(0, 1, 0, "l4")
    l5 = line(1, 0, 0, "l5")

    # plot the lines
    l1.plot(xAxis, "blue")
    l2.plot(xAxis, l1.auto_color_chooser())
    l3.plot(xAxis, l2.auto_color_chooser())

    l4.plot(xAxis, "orange", lw=4)

    l5.plot(xAxis, "orange", lw=4)

    line_x10 = line(1, 0, 10, "x=10")

    # fill the feasible space, based on the constraints
    # find the vector points (different name)
    v0 = [0, 10]

    v1 = [0, 4]
    # l1 intersection with x=0
    plt.plot(v1[0], v1[1], "o", color="red")

    v2 = l1.intersection(l2)
    v3 = l2.intersection(l3)

    v4 = l3.intersection(line_x10, plotting=False)

    # corner point for the fill
    v5 = [10, 10]

    # create the x and y coordinates for the fill
    x = [i[0] for i in [v0, v1, v2, v3, v4, v5]]
    y = [i[1] for i in [v0, v1, v2, v3, v4, v5]]

    # fill takes the x and y of a polygon and fills it with color
    plt.fill(x, y, color="gray", alpha=0.5)


def graphical_solution_max(a, b, minlim, maxlim, xAxis, legend=True):
    """plots the extra lines of the objective function """
    extra_lines = [line(a, b, i, 'extra') for i in range(minlim, maxlim)]
    for lin in extra_lines:
        lin.legend_show = legend
        lin.plot(xAxis, 'cornflowerblue')
        # we need to find the intersection with the l3 (brown line) to find the max value of the objective function
```

```
        lin.intersection(line(1, -2, 1, "l3"))
        # prints the legend and the intersection points


def plt_settings(limit):
    # adjusts the dimentions of the plot
    plt.ylim(0, limit)
    plt.xlim(0, limit)
    plt.grid()


def main():
    xAxis = np.linspace(-100, 100, 100000)

    plt.figure('feasible region')
    plt_settings(10)
    plot_equations()
    plt.savefig('exerc01-feasible_region.png', dpi='figure')


    plt.figure('a: max 2x1-5x2')
    plt_settings(10)

    plot_equations()
    graphical_solution_max(2, -5, 0, 2, xAxis=xAxis)
    plt.savefig('exerc01-a.png', dpi='figure')

    plt.figure('b: max 2x1-4x2')
    plt_settings(10)

    plot_equations()
    graphical_solution_max(2, -4, 0, 3, xAxis=xAxis)
    plt.savefig('exerc01-b.png', dpi='figure')

    plt.figure('c: max 2x1-3x2')
    plt_settings(10)

    plot_equations()
    graphical_solution_max(2, -3, 0, 20, xAxis=xAxis, legend=False)
    plt.savefig('exerc01-c.png', dpi='figure')

    plt.show()


if __name__ == "__main__":
    main()
```

Παρατηρουμε οτι η αντικειμενικη συναρτηση τεμνει την καφε ($x_1 - 2x_2 = 1$ σε δυο σημεια:

- στο σημειο τομης της l2 και l3 (3,1)
- στο σημειο τομης της l3 με την $2x_1 - 5x_2 = 1$ (5,2)

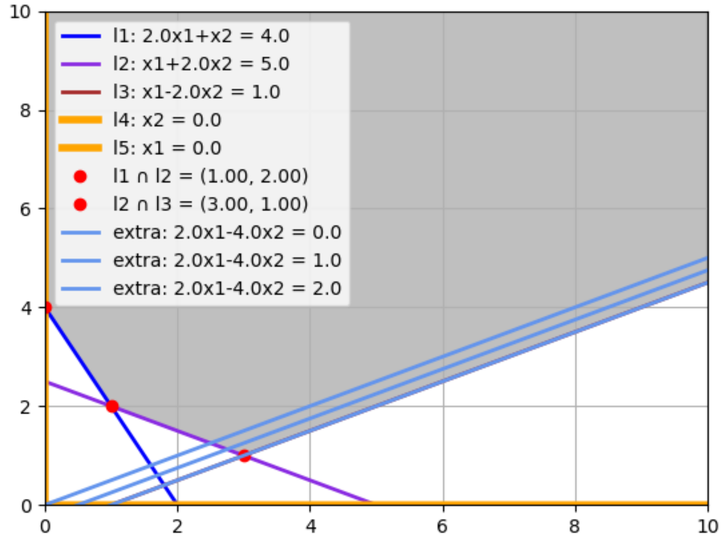


exerc01-a.png

Επομενως το κερδος μεγιστοποιειται στο σημειο $(x_1, x_2) = (5, 2)$ με τιμη 1 .

$max\ 2x_1 - 4x_2$

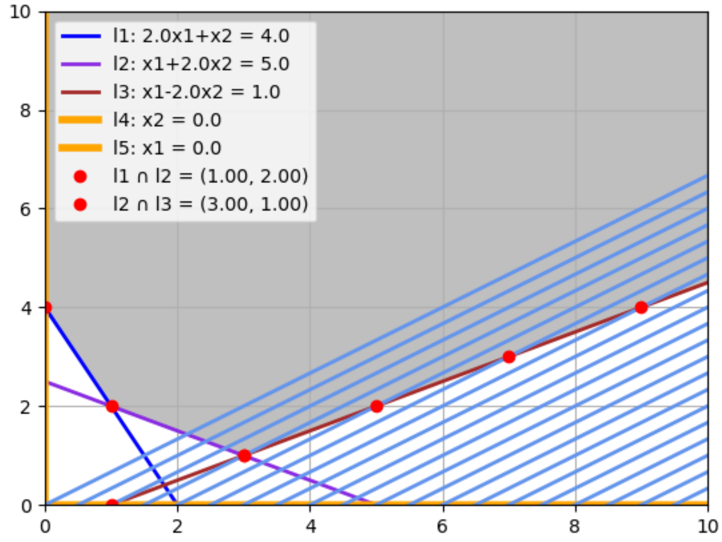
Θα ακολουθησουμε την ιδια διαδικασια και απο τον κωδικα θα παρουμε την παρακατω γραφικη παρασταση



exerc01-b.png

Εδω βλεπουμε οτι για c = 2 ($2x_1 - 4x_2 = 2$) η ευθεια extra ταυτιζεται πανω στην l3 . Αυτο σημαινει οτι εχει απειρες λυσεις , και μεγιστη τιμη 2 .

$$max\ 2x_1 - 3x_2$$



exerc01-c.png

Στην τριτη γραφικη παρασταση βλεπουμε οτι η ευθεια $2x_1 - 4x_2 = c$ δνε ειναι φραγμενη.

(περισσοτερα ??)

Ασκηση 2

Άσκηση 2. Εταιρεία τροφίμων σχεδιάζει ένα καινούργιο προϊόν (snack) με χαμηλά λιπαρά. Συγκεκριμένα, οι προδιαγραφές των τεχνολόγων τους απαιτούν κάθε 1 μονάδα του προϊόντος να περιέχει τουλάχιστον 5.1 γρ. φυτικές ίνες, το πολύ 8.4 γρ. λιπαρά και το πολύ 10.8 γρ. πρωτεΐνης. Για την παρασκευή του προϊόντος θα χρειαστεί η μίξη δύο δημητριακών, G_1 και G_2. Τα δύο δημητριακά έχουν διαφορετικά θρεπτικά χαρακτηριστικά και τα οποία δίνονται στον Πίνακα 1.

Πίνακας 1: Θρεπτικά χαρακτηριστικά των δημητριακών.

	Ποσότητα (γρ. ανά μονάδα)		
	Φυτικές ίνες	Λιπαρά	Πρωτεΐνη
G_1	6	6	12
G_2	4.5	9	9

Αν το κόστος μίας μονάδας των δημητριακών G_1 και G_2 είναι 6 και 7.5 χρηματικές μονάδες αντίστοιχα, προσδιορίστε την ποσότητα που πρέπει να χρησιμοποιηθεί από το καθένα εξ' αυτών, έτσι ώστε να δημιουργηθεί 1 μονάδα του ζητούμενου προϊόντος με τον οικονομικότερο δυνατό τρόπο.

(α) Δώστε ένα μοντέλο γραμμικού προγραμματισμού για το παραπάνω πρόβλημα σχεδιασμού προϊόντος.

(β) Λύστε το πρόβλημα γραφικά.

(γ) Σχολιάστε τη λύση που βρήκατε σε σχέση με τους περιορισμούς του προβλήματος.

ασκ2-εκφωνηση

α) Επιλυση

max ..

st

x>0

Απο τους περιορισμους , μπορουμε να βγαλουμε το μοντελο :

$$min\,\{\begin{bmatrix}6\ 7.5\end{bmatrix}\begin{bmatrix}x_1\\x_2\end{bmatrix} \mid \begin{bmatrix}-6\ -4.5\\ 6\ 9\\ 12\ 9\\ 1\ 1\\ -1\ -1\end{bmatrix}\ x \leq \begin{bmatrix}-5.1\\ 8.4\\ 10.8\\ 1\\ -1\end{bmatrix}, x \geq 0\}$$

οπου $x = [x_1, x_2]^T$ ειναι η ποσοτητα που χρησιμοποιουμε απο τα δυο δημητριακα G1, G2

Επειδη πρεπει να παρουμε ακριβως μια μοναδα προιοντος, μετατρεπουμε την ισοτητα $x_1 + x_2 = 1$ σε δυο ανισοτητες : $x_1 + x_2 \leq 1$ και $-x_1 - x_2 \leq -1$

---

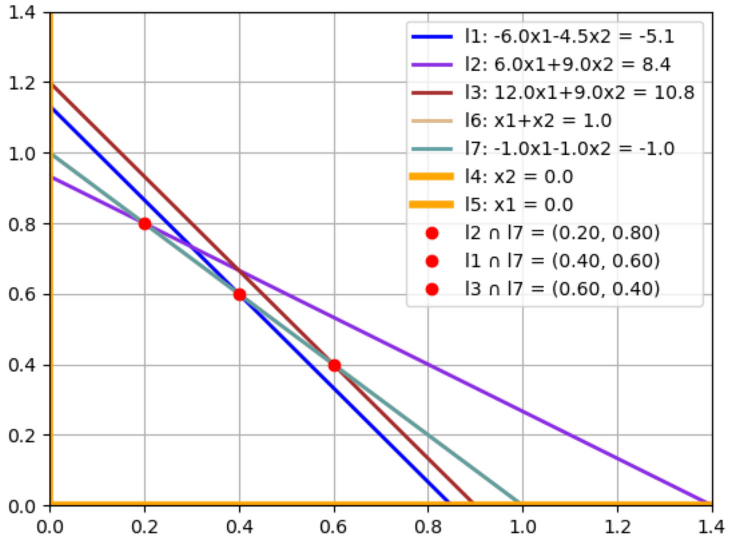
b) Γραφικη επιλυση

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/dfce6276-3414-40d9-8a0c-080306aa2ede/exerc-02.py

python code for the second exercise

```
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.colors as mcolors

from plt_line import line


def plot_equations():

    # create the x axis points
    xAxis = np.linspace(-100, 100, 100000)

    # create the lines from the constraints
    l1 = line(-6, -4.5, -5.1, "l1")
    l2 = line(6,  9, 8.4, "l2")
    l3 = line(12, 9, 10.8, "l3")
    l4 = line(0, 1, 0, "l4")
    l5 = line(1, 0, 0, "l5")
    l6 = line(1, 1, 1, "l6")
    l7 = line(-1, -1, -1, "l7")


    # plot the lines
    l1.plot(xAxis, "blue")
    l2.plot(xAxis, l1.auto_color_chooser())
    l3.plot(xAxis, l2.auto_color_chooser())
    l6.plot(xAxis, l3.auto_color_chooser())
    l7.plot(xAxis, l6.auto_color_chooser())

    l4.plot(xAxis, "orange", lw=4)

    l5.plot(xAxis, "orange", lw=4)


    v1 = l2.intersection(l7)

    v2 = l1.intersection(l7)
```

```
    # this is the point we want
    v3 = l3.intersection(l7)


def plt_settings(limit):
    plt.ylim(0, limit)
    plt.xlim(0, limit)
    plt.grid()


def main():

    plt.figure('min 6x1-7.5x2')
    plt_settings(1.4)
    plot_equations()
    plt.savefig('exerc02.png', dpi='figure')


    plt.show()


if __name__ == "__main__":
    main()
```



Απο την γραφικη παρασταση , η εφικτη περιοχη ειναι μονο κατω απο τις ευθειες και πανω στην $x_1 + x_2 = 1$. Οποτε παιρνουμε τα σημεια τομης της l7 με τις l2 (μωβ) και l3(καφε)

- [0.2 , 0.8]
- [0.4 , 0.6]
- [0.6 , 0.4]

Η $6x_1 + 7.5x_2$ ελαχιστοποιειται στο σημειο [0.6 , 0.4] με τιμη 6.6

Ασκηση 3

Άσκηση 3. Αεροπορική εταιρεία σχεδιάζει τη στελέχωση του τμήματος εξυπηρέτησης πελατών ανάλογα με το ημερήσιο πρόγραμμα των πτήσεων της. Σύμφωνα με αυτό το πρόγραμμα ο Πίνακας 2 δίνει τον ελάχιστο αριθμό ατόμων που θα πρέπει να εργάζονται σε κάθε ώρα του 24-ώρου καθώς και το ημερήσιο κόστος ανά εργαζόμενο ανάλογα με τη βάρδια στην οποία απασχολείται.

Πίνακας 2: Απαιτούμενος Αριθμός Εργαζομένων.

Περίοδος 24-ώρου	Ελάχιστος Αριθμός Εργαζομένων
06:00 - 08:00	48
08:00 - 10:00	79
10:00 - 12:00	65
12:00 - 14:00	87
14:00 - 16:00	64
16:00 - 18:00	73
18:00 - 20:00	82
20:00 - 22:00	43
22:00 - 24:00	52
24:00 - 06:00	15

Σύμφωνα με τους κανονισμούς κάθε εργαζόμενος θα πρέπει να να εργάζεται συνεχές 8-ωρο που εκτείνεται στη διάρκεια μια βάρδιας. Οι βάρδιες είναι 5, δηλ. 6:00 π.μ.-2:00 μ.μ, 8:00 π.μ.-4:00 μ.μ, 12:00 π.μ.-8:00 μ.μ, 4:00 μ.μ.-12:00 μ.μ, και 10:00 μ.μ.-6:00 π.μ. Τέλος, το ημερήσιο κόστος ανά εργαζόμενο στις βάρδιες εξαρτάται από τη δημοτικότητα της κάθε βάρδιας αλλά και τα ειδικά επιδόματα που πιθανόν να προσφέρονται, όπως φαίνεται στον Πίνακα 3.

Πίνακας 3: Ημερήσιο κόστος ανά εργαζόμενο και βάρδια.

Βάρδια	Περίοδος	Ημερήσιο κόστος ανά εργαζόμενο
1	06:00 - 14:00	170
2	08:00 - 16:00	160
3	12:00 - 20:00	175
4	16:00 - 24:00	180
5	22:00 - 06:00	195

Μοντελοποιήστε το παραπάνω πρόβλημα προγραμματισμού ανθρώπινου δυναμικού με τη βοήθεια του γραμμικού προγραμματισμού και αντικειμενικό σκοπό την ελαχιστοποίηση του συνολικού ημερήσιου κόστους της εταιρείας σε μισθούς για το συγκεκριμένο τμήμα.

εκφωνηση ασκησης 3

Επιλυση

Θα θεωρησουμε τις μεταβλητες αποφασης: $x_1, x_2, x_3, x_4, x_5, x_6, x_7, x_8, x_9, x_{10}$

οπου x_i ειναι ο ελαχιστος αριθμος

ωρες	ελαχιστος αριθμος	μεταβλητη	Βαρδια
06:00-08:00	48	x1	1
08:00-10:00	79	x2	1,2

ωρες	ελαχιστος αριθμος	μεταβλητη	Βαρδια
10:00-12:00	65	x3	1,2
12:00-14:00	87	x4	1,2,3
14:00-16:00	64	x5	2,3
16:00-18:00	73	x6	3,4
18:00-20:00	82	x7	3,4
20:00-22:00	43	x8	4
22:00-24:00	52	x9	4,5
24:00-06:00	15	x10	5

Βαρδια	περιοδος βαρδιας	κοστος ανα ημερα	αριθμος υπαλλληλων
1	06:00-14:00	170	x1+x2+x3+x4
2	08:00-16:00	160	x2+x3+x4+x5
3	12:00-20:00	175	x4+x5+x6+x7
4	16:00-24:00	180	x6+x7+x8+x9
5	22:00-06:00	195	x9+x10

Οποτε εχουμε για καθε βαρδια, αναλογα με την περιοδο της βαρδιας μπορουμε να υπολογισουμε τον αριθμο των υπαλληλων που θα δουλευουν (x_i)

Επομενως μπορουμε να συνδεσουμε τις στηλες κοστος ανα ημερα με τον αριθμο υπαλληλων καθε βαρδιας

Εμεις θελουμε να εχουμε το ελαχιστο καθημερινο κοστος της εταιριας.

$min\ (x_1 + x_2 + x_3 + x_4) * 170 + (x_2 + x_3 + x_4 + x_5) * 160 + (x_4 + x_5 + x_6 + x_7) * 175 + (x_6 + x_7 + x_8 + x_9) * 180 + (x_9 + x_{10}) * 195$

Οι περιορισμοι μας st :

$$
\begin{gathered}
x_1 \geq 48 \\
x_2 \geq 79 \\
x_3 \geq 65 \\
x_4 \geq 87 \\
x_5 \geq 64 \\
x_6 \geq 73 \\
x_7 \geq 82 \\
x_8 \geq 43 \\
x_9 \geq 52 \\
x_{10} \geq 15
\end{gathered}
$$

Μπορουμε να περιγραψουμε το προβλημα με τη παρακατω σχεση:

$$min\ \{cx \mid I_{10}x \geq b\ \}$$

Οπου:

$$c = \begin{bmatrix} 170, & 330, & 330, & 505, & 335, & 355, & 355, & 180, & 375, & 195 \end{bmatrix}$$

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \end{bmatrix}$$

I_{10} ειναι ο μοναδιαιος πινακας

$$I_{10} = \begin{bmatrix} 1,0,0,0,0,0,0,0,0,0 \\ 0,1,0,0,0,0,0,0,0,0 \\ 0,0,1,0,0,0,0,0,0,0 \\ 0,0,0,1,0,0,0,0,0,0 \\ 0,0,0,0,1,0,0,0,0,0 \\ 0,0,0,0,0,1,0,0,0,0 \\ 0,0,0,0,0,0,1,0,0,0 \\ 0,0,0,0,0,0,0,1,0,0 \\ 0,0,0,0,0,0,0,0,1,0 \\ 0,0,0,0,0,0,0,0,0,1 \end{bmatrix}$$

$$b = \begin{bmatrix} 48, 79, 65, 87, 64, 73, 82, 43, 52, 15 \end{bmatrix}^T$$

$$b = \begin{bmatrix} 48 \\ 79 \\ 65 \\ 87 \\ 64 \\ 73 \\ 82 \\ 43 \\ 52 \\ 15 \end{bmatrix}$$

Ασκηση 4

Άσκηση 4. Αποδείξτε τις παρακάτω προτάσεις.

(Π1) Η τομή X δύο κυρτών συνόλων X_1 και X_2 είναι κυρτό σύνολο.

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

(Π3) Αν $\mathbf{x^1}, \mathbf{x^2}, \ldots, \mathbf{x^m}$ είναι σημεία ενός κυρτού συνόλου $X \subset \mathbb{R}^n$, τότε κάθε κυρτός συνδυασμός τους ανήκει επίσης στο σύνολο X. (**Υπόδειξη**. Δείξτε πρώτα ότι ισχύει για $m = 2$ και $m = 3$. Μετά με επαγωγή μπορούμε να δείξουμε ότι θα ισχύει για κάθε m.)

ασκηση 4 εκφωνηση

Π1 Επιλυση

Απο την θεωρια , γνωριζουμε οτι ενα συνολο X ειναι κυρτο εαν και μονο αν για $\forall x, y \ και \ \lambda \in [0,1]$ ισχυει οτι :

$$\lambda x + (1-\lambda)y \ \in X$$

Θελουμε το ευθυγραμμο τμημα που ενωνει δυο οποιαδηποτε σημεια του X να βρισκεται ολοκληρο μεσα στο X .

Για να δειξουμε οτι η τομη δυο κυρτων συνολων ειναι κυρτο συνολο, πρεπει να δειξουμε οτι οποιοδηποτε ευθυγραμμο τμημα μεταξυ δυο σημειων ανηκει και αυτο στην τομη.

Θεωρουμε δυο οποιαδηποτε σημεια x,y της τομης των $X_1 \cap X_2$

Οποτε ισχυει

$$\begin{gathered} x, y \ \in X_1 \cap X_2, \\ x, y \ \in X_1, \\ x, y \ \in X_2 \end{gathered}$$

Εαν τα x,y ανηκουν στην τομη των συνολων , αναγκαστικα περιεχονται και στα συνολα .

Εφοσον τα X_1, X_2 ειναι κυρτα συνολα ισχυει $\lambda x + (1-\lambda)y \ \in X_1$ και $\lambda x + (1-\lambda)y \ \in X_2$

Οποιοσδηποτε συνδιασμος των x,y θα ειναι μεσα και στα δυο συνολα X_1, X_2. Επομενως και το $\lambda x + (1-\lambda)y \ \in X_1 \cap X_2$

Αρα η τομη δυο κυρτων συνολων ειναι και αυτο κυρτο συνολο.

Π2 Επιλυση

Εχουμε το συνολο $\Omega = \{(x,y) \in \Re^2 \mid x^2 + y^2 \leq 1\}$

Θα παρουμε δυο σημεια $(x_1, y_1), (x_2, y_2)$ τα οποια ανηκουν στο συνολο Ω, δηλαδη $x_1^2 + y_1^2 \leq 1, x_2^2 + y_2^2 \leq 1$

Για να δειξουμε οτι το συνολο Ω ειναι κυρτο, θα πρεπει και τα $\lambda x_1 + (1-\lambda)x_2 \ \in \Omega, \ \ \lambda y_1 + (1-\lambda)y_2 \ \in \Omega$

Οποτε θελουμε να ισχυει για $x = \lambda x_1 + (1-\lambda)x_2$, $\ y = \lambda y_1 + (1-\lambda)y_2$: $x^2 + y^2 \leq 1$

$$
\begin{gathered}
x^2 + y^2 \le 1 \implies \\
(\lambda x_1 + (1-\lambda)x_2)^2 \; + \; (\lambda y_1 + (1-\lambda)y_2)^2 \le 1 \implies \\
\lambda^2(x_1^2 + y_1^2) + (1-\lambda)^2(x_2^2 + y_2^2) + 2\lambda(1-\lambda)(x_1x_2 + y_1y_2) \le 1 \implies \\
\Theta\varepsilon\omega\rho o\upsilon\mu\varepsilon \\
c \; = \; \lambda^2(x_1^2 + y_1^2) + (1-\lambda)^2(x_2^2 + y_2^2) + 2\lambda(1-\lambda)(x_1x_2 + y_1y_2)
\end{gathered}
$$

Οπου εφοσον ισχυει $x_1^2 + y_1^2 \le 1, x_2^2 + y_2^2 \le 1$

$$
\begin{gathered}
x_1^2 + y_1^2 \le 1 \implies \\
\lambda^2(x_1^2 + y_1^2) \le \lambda^2
\end{gathered}
$$

$$
\begin{gathered}
x_2^2 + y_2^2 \le 1 \implies \\
(1-\lambda)^2(x_2^2 + y_2^2) \le (1-\lambda)^2
\end{gathered}
$$

Τελος

$$
\{x_1^2 + y_1^2 \le 1, x_2^2 + y_2^2 \le 1\}
$$

$$
\begin{gathered}
\frac{x_1^2 + y_1^2}{2} + \frac{x_2^2 + y_2^2}{2} \le 1 \implies \\
\frac{x_1^2 + y_1^2}{2} + \frac{x_2^2 + y_2^2}{2} - x_1x_2 - y_1y_2 \le 1 - x_1x_2 - y_1y_2 \implies \\
\frac{x_1^2 - 2x_1x_2 + x_2^2}{2} + \frac{y_1^2 - 2y_1y_2 + y_2^2}{2} \le 1 - x_1x_2 - y_1y_2 \implies \\
\frac{(x_1 - x_2)^2}{2} + \frac{(y_1 - y_2)^2}{2} \le 1 - x_1x_2 - y_1y_2 \implies \\
0 \le \frac{(x_1 - x_2)^2}{2} + \frac{(y_1 - y_2)^2}{2} \le 1 - x_1x_2 - y_1y_2 \implies \\
0 \le 1 - x_1x_2 - y_1y_2 \implies \\
x_1x_2 + y_1y_2 \le 1 \implies \\
\lambda(1-\lambda)(x_1x_2 + y_1y_2) \le \lambda(1-\lambda) \implies
\end{gathered}
$$

Οποτε

$$
\begin{gathered}
c \; = \; \lambda^2(x_1^2 + y_1^2) + (1-\lambda)^2(x_2^2 + y_2^2) + 2\lambda(1-\lambda)(x_1x_2 + y_1y_2) \implies \\
\lambda^2(x_1^2 + y_1^2) + (1-\lambda)^2(x_2^2 + y_2^2) + 2\lambda(1-\lambda)(x_1x_2 + y_1y_2) \le \lambda^2 + (1-\lambda)^2 + 2\lambda(1-\lambda)
\end{gathered}
$$

$$
\lambda^2 + (1-\lambda)^2 + 2\lambda(1-\lambda) = (\lambda + 1 - \lambda)^2 = 1 \implies
$$

$$
\lambda^2(x_1^2 + y_1^2) + (1-\lambda)^2(x_2^2 + y_2^2) + 2\lambda(1-\lambda)(x_1x_2 + y_1y_2) \le 1
$$

Αρα εφοσον για δυο οποιαδηποτε σημεια ισχυει, το συνολο Ω ειναι κυρτο συνολο

Π3 Επιλυση

Ασκηση 5

Άσκηση 5. Θεωρήστε το πρόβλημα γραμμικού προγραμματισμού:

$$\min\ Z = 12x_1 - 10x_2 - 30x_3$$

όταν

$$\begin{aligned} -3x_1 + 2x_2 + 8x_3 &\leq 17 \\ -x_1 + x_2 + 3x_3 &\leq 9 \\ -2x_1 + x_2 + 8x_3 &\leq 16 \\ x_1, x_2, x_3 &\geq 0 \end{aligned}$$

(α) Θεωρήστε το πολύτοπο των εφικτών λύσεων του παραπάνω προβλήματος γραμμικού προγραμματισμού. Βρείτε όλες τις κορυφές που δημιουργούνται από τις τομές των υπερεπιπέδων του και ξεχωρίστε ποιες από αυτές είναι κορυφές του πολύτοπου των εφικτών λύσεων. Εντοπίστε, αν υπάρχουν, τις εκφυλισμένες κορυφές.

(β) Προσθέστε μεταβλητές χαλάρωσης στο σύστημα ανισώσεων και βρείτε όλες τις βασικές (εφικτές και μη-εφικτές) λύσεις για το μη ομογενές σύστημα εξισώσεων που δημιουργείται. Εντοπίστε (αν υπάρχουν) τις εκφυλισμένες βασικές λύσεις.

(γ) Αντιστοιχίστε τις βασικές λύσεις που βρήκατε στο (β) ερώτημα με τις κορυφές του ερωτήματος (α) και τέλος υποδείξτε τη βέλτιστη λύση και βέλτιστη κορυφή του προβλήματος.

ασκηση 5 εκφωνηση

a) Επιλυση

Εχουμε 3 μεταβλητες και 6 ανισοτητες

Οποτε θα εχουμε $\binom{6}{3} = \frac{6!}{3!(6-3)!} = \frac{4*5*6}{3*2} = 20$ πιθανες κορυφες

Θελουμε να βρουμε ολους τους δυνατους συνδυασμους 3 περιορισμων και να ψαξουμε εκει λυσεις

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/e02bad8b-4d64-4f10-bbe8-3fad3e98120c/exerc-05.py

exerc05 code for finding corners

```
import numpy as np
from itertools import combinations

def inequality(li, result):
    return sum([x*y for x,y in zip(li[:3],result)]) <= li[3]

def presenting(li):
    """Presenting the equation in a nice way"""

    result = ''
    # for i in range(3):
```

```
    if li[0] != 0:
        if li[0] == -1:
            result += '-'
        elif li[0] != 1:
            result += str(li[0])
        result += 'x1'
        if li[1]>0:
            result += '+'

    if li[1] != 0:
        if li[1] == -1:
            result += '-'
        elif li[1] != 1:
            result += str(li[1])
        result += 'x2'
        if li[2]>0:
            result += '+'

    if li[2] != 0:
        if li[2] == -1:
            result += '-'
        elif li[2] != 1:
            result += str(li[2])
        result += 'x3'

    return f'{result} = {li[3]}'

def main():
    # constraints from equations 1:3 and the second part of the equation
    A = [   [-3,2,8,17],
            [-1,1,3,9],
            [-2,1,8,16],
            [1,0,0,0],
            [0,1,0,0],
            [0,0,1,0],
         ]

    # get all combinations of 3 equations
    c = list(combinations(A,3))

    results = []

    print(f'\n\n{"-"*15}All{"-"*15}\n\n')

    for inde,value in enumerate(c) :
        a = np.array([x[:3] for x in value])
        b = np.array([x[3] for x in value])

        # solve the system of equations
        result = np.linalg.solve(a,b)
        results.append(result)

        print()
        print('\n'.join([f'{presenting(value[0]):17s} |',f'{presenting(value[1]):17s} | => {result} ',f'{presenting(value[2]):17s} |\n


    print(f'\n\n{"-"*15}valid{"-"*15}\n\n')

    for inde,value in enumerate(c) :
        # check if the result is valid with all 6 equations
        if all([inequality(x,results[inde]) for x in A[:3]]) and all([x>=0 for x in results[inde]]):
            print()
            print('\n'.join([f'{presenting(value[0]):17s} |',f'{presenting(value[1]):17s} | => {results[inde]} ',f'{presenting(value[2



if __name__ == "__main__":
    main()
```

Φτιαχνουμε πινακα A με τα βαρη των μεταβλητων x_1, x_2, x_3 και το αποθηκευουμε σαν λιστα A .

Χρησιμοποιωντας την numpy.linalg.solve(a,b) λυνουμε τις εξισωσεις που προκυπτουν απο τις ανισοτητες .

Ελεγχουμε το αποτελεσμα μας οτι ισχυει για τους περιορισμους και τυπωνουμε το συστημα και την λυση του με την συναρτηση presenting .

```
---------------All---------------



-3x1+2x2+8x3 = 17 |
-x1+x2+3x3 = 9    | => [6.33333333 7.33333333 2.66666667]
-2x1+x2+8x3 = 16  |


-3x1+2x2+8x3 = 17 |
-x1+x2+3x3 = 9    | => [-0.  10.5 -0.5]
x1 = 0            |


-3x1+2x2+8x3 = 17 |
-x1+x2+3x3 = 9    | => [21.  0. 10.]
x2 = 0            |


-3x1+2x2+8x3 = 17 |
-x1+x2+3x3 = 9    | => [ 1. 10.  0.]
x3 = 0            |


-3x1+2x2+8x3 = 17 |
-2x1+x2+8x3 = 16  | => [-2.96059473e-16  1.00000000e+00  1.87500000e+00]
x1 = 0            |


-3x1+2x2+8x3 = 17 |
-2x1+x2+8x3 = 16  | => [-1.    0.    1.75]
x2 = 0            |


-3x1+2x2+8x3 = 17 |
-2x1+x2+8x3 = 16  | => [-15. -14.   0.]
x3 = 0            |


-3x1+2x2+8x3 = 17 |
x1 = 0            | => [-0.     0.     2.125]
x2 = 0            |


-3x1+2x2+8x3 = 17 |
x1 = 0            | => [-0.   8.5  0. ]
x3 = 0            |


-3x1+2x2+8x3 = 17 |
x2 = 0            | => [-5.66666667  0.          0.        ]
x3 = 0            |


-x1+x2+3x3 = 9    |
-2x1+x2+8x3 = 16  | => [-4.4408921e-16  4.8000000e+00  1.4000000e+00]
x1 = 0            |


-x1+x2+3x3 = 9    |
-2x1+x2+8x3 = 16  | => [-12.   0.  -1.]
x2 = 0            |


-x1+x2+3x3 = 9    |
-2x1+x2+8x3 = 16  | => [-7.  2.  0.]
x3 = 0            |


-x1+x2+3x3 = 9    |
x1 = 0            | => [-0.  0.  3.]
x2 = 0            |


-x1+x2+3x3 = 9    |
x1 = 0            | => [-0.  9.  0.]
x3 = 0            |


-x1+x2+3x3 = 9    |
x2 = 0            | => [-9.  0.  0.]
x3 = 0            |


-2x1+x2+8x3 = 16  |
```

```
x1 = 0           | => [-0.  0.  2.]
x2 = 0           |


-2x1+x2+8x3 = 16  |
x1 = 0           | => [-0. 16.  0.]
x3 = 0           |


-2x1+x2+8x3 = 16  |
x2 = 0           | => [-8.  0.  0.]
x3 = 0           |


x1 = 0           |
x2 = 0           | => [0. 0. 0.]
x3 = 0           |



---------------valid---------------



-3x1+2x2+8x3 = 17 |
-x1+x2+3x3 = 9    | => [6.33333333 7.33333333 2.66666667]
-2x1+x2+8x3 = 16  |


-3x1+2x2+8x3 = 17 |
-x1+x2+3x3 = 9    | => [ 1. 10.  0.]
x3 = 0           |


-3x1+2x2+8x3 = 17 |
x1 = 0           | => [-0.   8.5  0. ]
x3 = 0           |


-2x1+x2+8x3 = 16  |
x1 = 0           | => [-0.  0.  2.]
x2 = 0           |


x1 = 0           |
x2 = 0           | => [0. 0. 0.]
x3 = 0           |
```

Οποτε εχουμε 5 κορυφες του πολυτοπου της εφικτης περιοχης

Ασκηση 6

Άσκηση 6. Θεωρήστε το πρόβλημα γραμμικού προγραμματισμού:

$$\min Z = -x_1 - 4x_2 - 5x_3$$

όταν

$$\begin{aligned} x_1 + 2x_2 + 3x_3 &\leq 2 \\ 3x_1 + x_2 + 2x_3 &\leq 2 \\ 2x_1 + 3x_2 + x_3 &\leq 4 \\ x_1, x_2, x_3 &\geq 0 \end{aligned}$$

(α) Εφαρμόστε τον αλγόριθμο Simplex για να βρείτε τη βέλτιστη λύση του, αν υπάρχει. Σε κάθε επανάληψη του αλγορίθμου θα πρέπει να περιγράφετε συνοπτικά τα βήματα που ακολουθείτε και τις αποφάσεις που παίρνετε μέχρι το επόμενο βήμα.

(β) Εφαρμόστε όλες τις εναλλακτικές επιλογές που μπορεί να έχετε σε κάθε βήμα επιλογής της εισερχόμενης ή εξερχόμενης μεταβλητής σε κάθε επανάληψη του αλγορίθμου και δημιουργήστε έναν γράφο με τα βήματα (κορυφές) του αλγορίθμου μέχρι τη βέλτιστη λύση.

εκφωνηση ασκηση 6