

Εργασία 1 Γραμμικής βελτιστοποίησης

Ονοματεπώνυμο : Νικόλας Φιλιππάτος

AM: 1072754

Εργασία: 1η

Ημερομηνία: March 25, 2024

Table Of Contents

- [Table Of Contents](#)
 - [Εκφωνήσεις](#)
 - [Exercise 1](#)
 - [Exercise 2](#)
 - [Exercise 3](#)
 - [Exercise 4](#)
 - [Exercise 5](#)
 - [Exercise 6](#)
 - [Solutions](#)
 - [Solution Exercise 01](#)
 - [1a](#)
 - [1b](#)
 - [1γ](#)
 - [Python Scripts](#)
 - [*plt_line.py*](#)
 - [*linearGui.py*](#)
 - [*exercise_01.py*](#)
 - [Solution Exercise 2](#)
 - [2a](#)
 - [2b](#)
 - [Solution Exercise 3](#)
 - [Μοντελοποίηση Προβλήματος](#)
 - [Solution Exercise 4](#)
 - [4b](#)
 - [Solution Exercise 5](#)
 - [5a](#)
 - [Λυσείς Συστηματος](#)
 - [Python Script](#)
 - [*Tools>equation_class.py*](#)
 - [*Tools>equation_solver.py*](#)
 - [*exercise_05.py*](#)
 - [Solution Exercise 6](#)
 - [Περιγραφή Simplex](#)
 - [Βήματα Simplex](#)
-

Εκφωνήσεις

Exercise 1

Δίνεται το παρακάτω πρόβλημα γραμμικού προγραμματισμού:

$$\begin{aligned} \min \quad & Z = 2x_1 - x_2 \\ \text{όταν} \quad & \\ \text{(Π1)} \quad & x_1 + x_2 \geq 10 \\ \text{(Π2)} \quad & -10x_1 + x_2 \leq 10 \\ \text{(Π3)} \quad & -4x_1 + x_2 \leq 20 \\ \text{(Π4)} \quad & x_1 + 4x_2 \geq 20 \\ & x_1, x_2 \geq 0 \end{aligned}$$

- α) Να παραστήσετε γραφικά την εφικτή περιοχή του προβλήματος καθώς και όλες τις κορυφές της. Περιγράψτε τη μορφή της εφικτής περιοχής. Με γραφικό τρόπο βρείτε τη βέλτιστη κορυφή του προβλήματος, εάν υπάρχει.
- β) Ομοίως με γραφικό τρόπο βρείτε τη βέλτιστη λύση, αν υπάρχει, όταν η αντικειμενική συνάρτηση γίνει:

$$\min Z = 11x_1 - x_2$$

και η εφικτή περιοχή παραμένει ίδια με το ερώτημα (α)

- γ) Αν η εφικτή περιοχή είναι όπως περιγράφεται στο ερώτημα (α) και η αντικειμενική συνάρτηση δίνεται ως:

$$\max Z = c_1x_1 - x_2$$

ποιά θα πρέπει να είναι η τιμή του c_1 ώστε η βέλτιστη λύση να βρίσκεται στην τομή των ευθειών που ορίζονται από τους περιορισμούς Π1 και Π4;

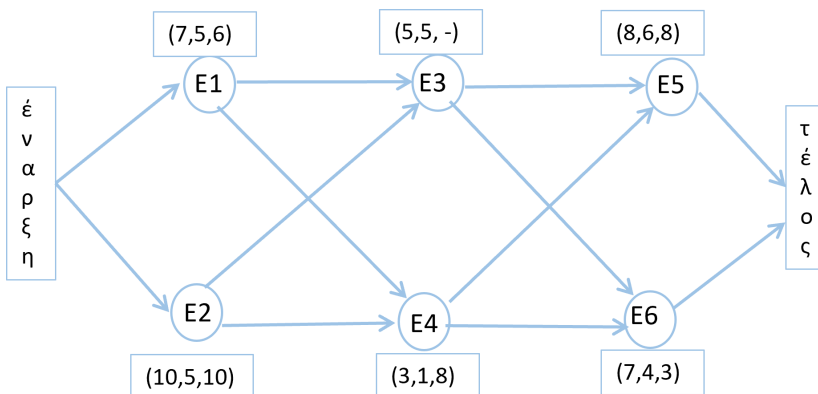
Exercise 2

Ένας φρουτοχυμός που παράγει η εταιρεία FRESH περιέχει το πολύ 3 μέρη χυμό πορτοκαλιού και τουλάχιστον 1 μέρος χυμό μήλου, σύμφωνα με τα αναγραφόμενα στη συσκευασία του. Η εβδομαδιαία δυναμικότητα της εταιρείας για το συγκεκριμένο προϊόν δεν ξεπερνάει τα 500 lit ενώ σύμφωνα με τα δεδομένα των πωλήσεων είναι σίγουρο ότι η αγορά μπορεί να απορροφήσει τουλάχιστον 400 lit κάθε εβδομάδα. Για την επόμενη εβδομάδα η εταιρεία μπορεί να διαθέσει μέχρι 250 lit χυμό μήλου και το τμήμα παραγωγής στοχεύει να παράξει φρουτοχυμό που θα περιέχει τουλάχιστον 40% χυμό πορτοκαλιού. Το κόστος για 1 lit χυμό πορτοκαλιού είναι 2 χ.μ. (χρηματικές μονάδες) και για 1 lit χυμό μήλου είναι 1 χ.μ. ενώ η εταιρεία πουλάει τον φρουτοχυμό (ανεξαρτήτου αναλογίας των δύο φρούτων) προς 5 χ.μ.

- α) Βοηθήστε το τμήμα παραγωγής της εταιρείας FRESH να αποφασίσει για τις ποσότητες χυμού που θα πρέπει να παράγει την επόμενη εβδομάδα όταν αντικειμενικός στόχος είναι η μεγιστοποίηση του κέρδους. Μοντελοποιήστε το παραπάνω σενάριο ως πρόβλημα γραμμικού προγραμματισμού και λύστε το γραφικά. Περιγράψτε αναλυτικά τις μεταβλητές απόφασης, την αντικειμενική συνάρτηση και τους περιορισμούς. Περιγράψτε τη βέλτιστη λύση αναφορικά με τους περιορισμούς, δηλ. ποιους ικανοποιεί οριακά (δεσμευτικοί περιορισμοί) και ποιους χαλαρά (μη δεσμευτικοί περιορισμοί).
- β) Αν αντίθετα το τμήμα παραγωγής στοχεύει να παράξει φρουτοχυμό που θα περιέχει τουλάχιστον 50%/60% χυμό πορτοκαλιού, τι επιπτώσεις θα έχει αυτή η απόφασή τους στη βέλτιστη λύση του προβλήματος;

Exercise 3

Θεωρήστε ένα έργο το οποίο για να ολοκληρωθεί απαιτεί την διεκπεραίωση 6 επί μέρους εργασίες (E1 - E6). Οι εργασίες είναι εξαρτημένες μεταξύ τους και οι εξαρτήσεις δίνονται με το παρακάτω σχήμα:



Σύμφωνα με το σχήμα οι εργασίες E1 και E2 μπορούν να εκτελεστούν παράλληλα, όμως για να εκτελεστεί εργασία E3 θα πρέπει να έχουν ολοκληρωθεί οι εργασίες E1 και E2.

Ομοίως και για τις υπόλοιπες εργασίες. Το έργο ολοκληρώνεται όταν εκτελεστούν (παράλληλα) οι εργασίες E5 και E6. Για κάθε εργασία δίνονται ο κανονικός χρόνος διεκπεραίωσης της εργασίας (σε εβδομάδες), το απόλυτο ελάχιστο για τον χρόνο αυτό, και το κόστος που θα προκύψει αν προσπαθήσουμε να μειώσουμε τον κανονικό χρόνο κατά μία εβδομάδα.

Η εταιρεία που έχει αναλάβει το έργο ενδιαφέρεται να μειώσει τον συνολικό χρόνο διεκπεραίωσης του (αν είναι εφικτό) σε 19 εβδομάδες, επομένως η διάρκεια μίας ή περισσότερων εργασιών θα πρέπει να μειωθεί σε σχέση με την κανονική τους διάρκεια. Προφανώς ο στόχος αυτός θα πρέπει να επιτευχθεί με το μικρότερο δυνατό κόστος.

Μοντελοποιήστε το συγκεκριμένο σενάριο προγραμματισμού εργασιών ως πρόβλημα γραμμικού προγραμματισμού. Ορίστε κατάλληλες μεταβλητές απόφασης και διαμορφώστε τους περιορισμούς όπως περιγράφονται στο σχήμα. Περιγράψτε και μοντελοποιήστε τον αντικειμενικό στόχο της εταιρείας. Δώστε την αντικειμενική συνάρτηση του προβλήματος γραμμικού προγραμματισμού. (Σημ. Η άσκηση δεν ζητάει τη λύση του προβλήματος, μόνο τη μοντελοποίησή του.)

Exercise 4

- (Π1) Η τομή X δύο κυρτών συνόλων X_1 και X_2 είναι κυρτό σύνολο. Ισχύει το ίδιο για την ένωση των κυρτών συνόλων;
- (Π2) Το σύνολο $M = \{(x, y) \in R_{++}^2 | xy \geq k, k \in R\}$ είναι κυρτό σύνολο. (Υπόδειξη. Μπορείτε να χρησιμοποιήσετε ως γνωστό το Λήμμα ότι για θετικούς αριθμούς a και b ισχύει πάντα: $\frac{a}{b} + \frac{b}{a} \geq 2$)

Exercise 5

Ασκήση 5. Θεωρήστε το πρόβλημα γραμμικού προγραμματισμού:

$$\max \quad Z = -2x_1 + x_2 - 4x_3 + 3x_4$$

όταν

$$x_1 + x_2 + 3x_3 + 2x_4 \leq 4$$

$$x_1 - x_3 + x_4 \leq 2$$

$$2x_1 + x_2 \leq 3$$

$$x_1, x_2, x_3, x_4 \geq 0$$

(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.

Exercise 6

Θεωρήστε το πρόβλημα γραμμικού προγραμματισμού

$$\max \quad Z = 5x_1 + 4x_2 - x_3 + 3x_4$$

όταν

$$3x_1 + 2x_2 - 3x_3 + x_4 \leq 24$$

$$3x_1 + 3x_2 + x_3 + 3x_4 \leq 36$$

$$x_1, x_2, x_3, x_4 \geq 0$$

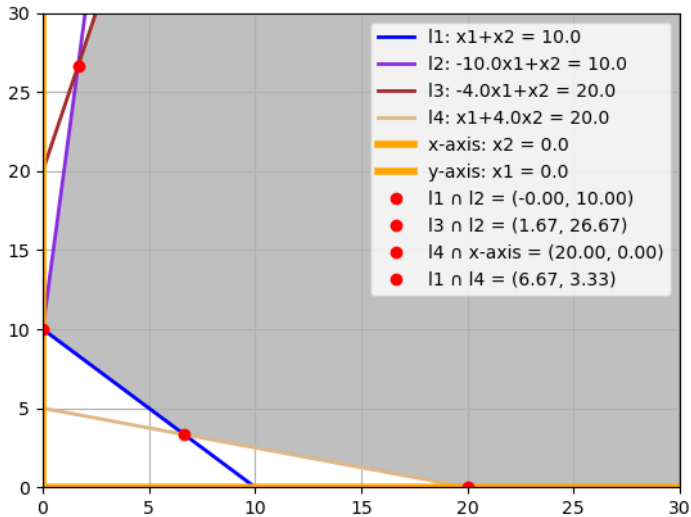
- (α) Εφαρμόστε τον αλγόριθμο Simplex για να βρείτε τη βέλτιστη λύση του, αν υπάρχει. Σε κάθε επανάληψη του αλγορίθμου περιγράψτε συνοπτικά τα βήματα που ακολουθείτε και τις αποφάσεις που παίρνετε μέχρι το επόμενο βήμα.
- (β) Εφαρμόστε όλες τις εναλλακτικές επιλογές που μπορεί να έχετε σε κάθε βήμα επιλογής της εισερχόμενης ή εξερχόμενης μεταβλητής στις επαναλήψεις του αλγορίθμου και δημιουργήστε έναν γράφο με τα βήματα (κορυφές) του αλγορίθμου μέχρι τη βέλτιστη λύση.

Solutions

Solution Exercise 01

1a

1. α) Να παραστήσετε γραφικά την εφικτή περιοχή του προβλήματος καθώς και όλες τις κορυφές της. Περιγράψτε τη μορφή της εφικτής περιοχής. Με γραφικό τρόπο βρείτε τη βέλτιστη κορυφή του προβλήματος, εάν υπάρχει.



Θα σχεδιάσουμε τις ευθειες

$$\begin{aligned}y &= -x + 10 \\y &= 10x + 20 \\y &= 4x + 20 \\y &= -\frac{1}{4}x + 5\end{aligned}$$

Με βάση τους περιορισμούς Π1-Π4 και $x_1, x_2 \geq 0$ η εφικτή περιοχή περιγράφεται από την γραμμοσκιασμένη περιοχή του σχήματος.

Κορυφές της εφικτής περιοχής :

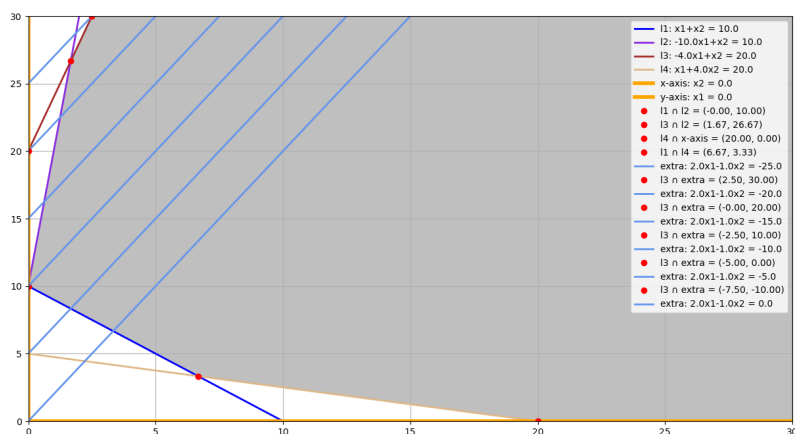
- Σημείο Τομής του Π2 και Π3 [1.67, 26.67]
- Σημείο Τομής του Π1 και Π2 [0, 10]
- Σημείο Τομής του Π1 και Π4 [6.67, 3.33]
- Σημείο Τομής του Π4 και του άξονα x [20, 0]

Παρατηρούμε η εφικτή περιοχή

Είναι bounded από τα αριστερά λόγω των περιορισμών Π3, Π2, Π1, Π4, αλλά από τα δεξιά όσο αυξάνονται τα x βλέπουμε ότι έχουμε μόνο ένα κάτω όριο (ευθεια $x_2 = 0$ και ο Π4)

Για να βρούμε την βελτιστη κορυφη, σχηματίζουμε τις ευθειες $2x_1 - x_2 = c$, όπου $c = [0, -5, -10, -15, -20, -25]$ και κοιτάζουμε εαν συμπίπτει με κάποια κορυφη

Βλέπουμε ότι μειώνοντας το c η αντικειμενική συνάρτηση παραμενει μεσα στην εφικτή περιοχη χωρις να βρισκει καποιο ανω οριο, οποτε δνε υπαρχει η βελτιστη κορυφη .



Το πρόβλημα έχει απεριοριστη λυση.

1b

2. (β) Ομοίως με γραφικό τρόπο βρείτε τη βέλτιστη λύση, αν υπάρχει, όταν η αντικειμενική συνάρτηση γίνει:

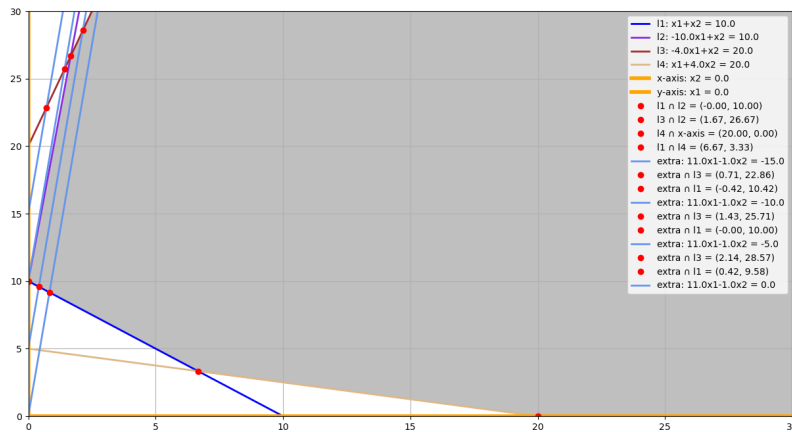
$$\min Z = 11x_1 - x_2$$

και η εφικτή περιοχή παραμένει ίδια με το ερώτημα (α)

Για να λυσουμε γραφικά το πρόβλημα γραφουμε την αντικειμενικη συναρτηση ιση με c :

$2x_1 - x_2 = c$, όπου $c = [0, -5, -10, -15, -20, -25]$

Μειωνουμε την τιμη του c και βλεπουμε να πηγαίνει προς τα αριστερα .



Παρατηρούμε ότι μετά το $c = -10$ η αντικειμενική συνάρτηση βγαίνει εκτός της εφικτής περιοχής. Επομένως η βέλτιστη κορυφή είναι η (0,10) Τομή του περιορισμού Π1 και Π2 και η μικρότερη τιμή που μπορεί να πάρει η αντικειμενική συνάρτηση είναι -10

1γ

3. (γ) Αν η εφικτή περιοχή είναι όπως περιγράφεται στο ερώτημα (α) και η αντικειμενική συνάρτηση δίνεται ως:

$$\max Z = c_1x_1 - x_2$$

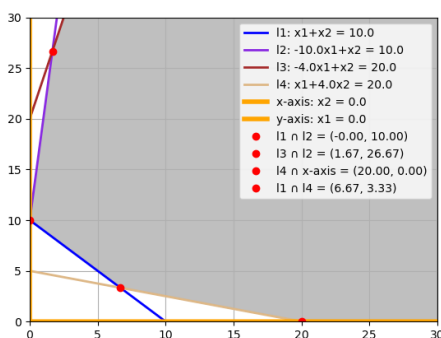
ποιά θα πρέπει να είναι η τιμή του c_1 ώστε η βέλτιστη λύση να βρίσκεται στην τομή των ευθειών που ορίζονται από τους περιορισμούς Π1 και Π4;

Η κορυφή των περιορισμών Π1, Π2

$$y = -x + 10$$

$$y = -\frac{1}{4}x + 5$$

- Σημείο Τομής του Π1 και Π4 [6,67, 3.33]



$$\min Z = 2x_1 - x_2$$

όταν

$$\begin{aligned} \text{(Π1)} \quad x_1 + x_2 &\geq 10 \\ \text{(Π2)} \quad -10x_1 + x_2 &\leq 10 \\ \text{(Π3)} \quad -4x_1 + x_2 &\leq 20 \\ \text{(Π4)} \quad x_1 + 4x_2 &\geq 20 \\ x_1, x_2 &\geq 0 \end{aligned}$$

Η αντικειμενική συνάρτηση του ερωτήματος γ είναι $\min Z = c_1x_1 - x_2$. Αν λυσουμε ως προς x_2 :

$$x_2 = c_1x_1 - Z$$

Πρέπει να βρούμε το εναρτισμω του c_1 ετισωστη βελτιστη λυση να ειναι η τομη των περιορισμων Π1 και Π2 : - Σημείο Τομής του Π1 και Π4 [6, 67, 3.33] Μπορουμε να γραψ

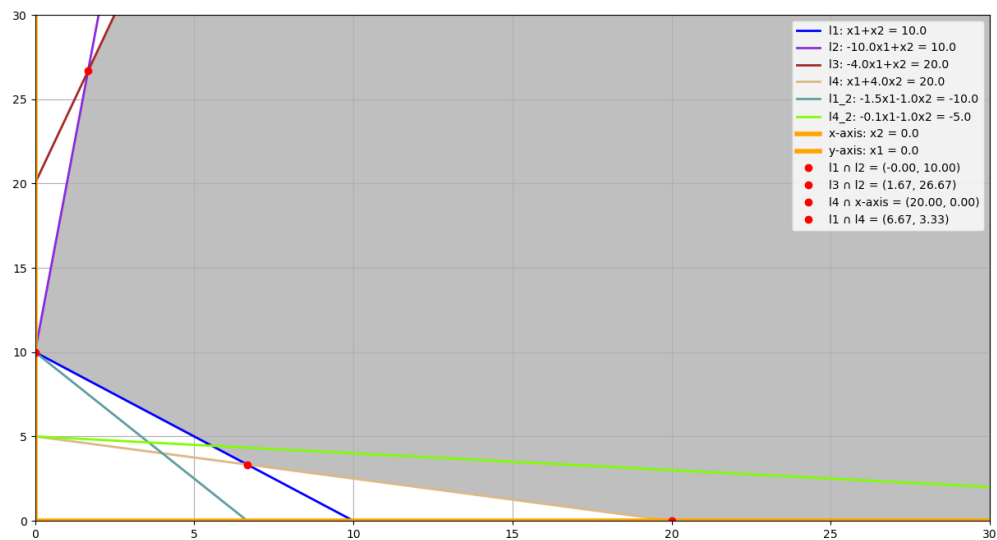
```

\begin{matrix}
-x\{1\}-x\{2\}\leq -10 \\\
\cdot \frac{1}{4} - x_{\{2\}} \leq -5 \\
\end{matrix}
$$$

```

Χωρίς να επηρεάσει το σχήμα.

Βλεπουμε οτι με το $c_1 = -1, c_2 = -\frac{1}{4}$ η αντικειμενικη συναρτηση ταυτιζεται με τις ευθειες των περιορισμων Π1 και Π4 αντιστοιχα



Βλεπουμε οτι αμα αυξησουμε το c_1 απο -0.25 σε -0.1 αποκταει περισσοτερες λυσεις και σταματαει η τομη [6.67, 3.33] να ειναι η βελτιστη κορυφη. Αντιστοιχα αμα μειωσουμε το c_1 απο -1 σε -1.5 η αντικειμενικη συναρτηση βγαίνει εκτος περιοχης.

Επομενως το c_1 θα ανηκει :

$$-1 < c_1 < -\frac{1}{4}$$

Δεν περιλαμβανουμε το -1 και -0.25 γιατι τοτε δνε θα ειχαμε μονο το σημειο [6.67,3.33] ως βελτιστη λυση

Python Scripts

Για την επίλυση αξιοποιήθηκαν 3 python scripts:

plt_line.py

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

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

    def __init__(
        self,
        a: float,
        b: float,
        c: float,
        name: str,
        legend_show=True,
        type: str = "line",
    ) -> None:
        """
        Args:
            a (float): _description_
            b (float): _description_
            c (float): _description_
            name (str): _description_
            legend_show (bool, optional): _description_. Defaults to True.
        """
        self.a, self.b, self.c = map(float, (a, b, c))
        self.name = name
        self.legend_show = legend_show
        self.type = type

    def __str__(self):
        """returns the equation"""

        output = 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 !=1 else '')+'x2' if self.b else ''}} = {{self.c}}"
        return output

    def __call__(self, x, reverse=False) -> float:
        if reverse:
            return self.reverse_equation(x)
        return self.equation(x)

    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:
            return None

        if self.b == 0:
            x = self.c / self.a
            y = line.equation(x)
        elif line.b == 0:
            x = line.c / line.a
            y = self.equation(x)
        else:
            x = x_up / x_down
            y = self.equation(x)

        if plotting:
            plt.plot(
                x, y, "ro", label=f"{{self.name}} n {{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"""
        if self.b != 0:
            return (-self.a * x + self.c) / self.b
        return self.c / self.a
```



```

def reverse_equation(self, y):
    """returns the x value of the line for a given y value"""
    if self.a != 0:
        return (-self.b * y + self.c) / self.a
    return self.c / self.b

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,
        )
    elif self.a == 0:
        plt.axhline(
            y=self.c / self.b,
            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 given, it returns the color if it is in the list, else it returns the first color in the list"""
    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]

```

linearGui.py

```

import matplotlib.pyplot as plt
import numpy as np
import matplotlib.colors as mcolors
import sys
from pathlib import Path
from shapely.geometry.polygon import Polygon
from shapely.geometry import Point

sys.path.append(str(Path(__file__).parents[1]))
from Tools.plt_line import line

class LinearGUI:
    def __init__(self, *args, **kwargs) -> None:
        """
        Args:
            plt (module): the matplotlib module
            xAxis (np.array): the x axis
            limit (int): the limit of the graph
            ylim (tuple): the y axis limits
            xlim (tuple): the x axis limits
            lines (list): the lines to plot
            save_images (bool): save the images
            v (list): the feasible points
            name (str): the name of the graph
        """

```

```

self.plt = kwargs.get("plt", plt)

self.limit = kwargs.get("limit", 10)

xAxis = np.linspace(-10, self.limit + 10, 10)
self.xAxis = kwargs.get("xAxis", xAxis)

self.ylim = kwargs.get("ylim", (0, self.limit))
self.xlim = kwargs.get("xlim", (0, self.limit))

self.lines = kwargs.get("lines", [])
self.name = kwargs.get("name", "LinearGUI")
self.figsize = kwargs.get("figsize")
self.step = kwargs.get("step", 1)
self.save_images = kwargs.get("save_images", False)

self.v = []

def show(self):
    self.plt.show()

def plot_equations(self):
    for i, line in enumerate(self.lines):
        if line.type == "line":
            if i == 0:
                color = "blue"
            else:
                color = self.lines[i - 1].auto_color_choser()
            line.plot(self.xAxis, color)

    self.add_axis_lines()

def add_axis_lines(self):
    right_limit = line(1, 0, self.limit, "limit", type="axis")
    x = line(0, 1, 0, "x-axis", type="axis")
    y = line(1, 0, 0, "y-axis", type="axis")

    self.lines.append(right_limit)
    self.lines.append(x)
    self.lines.append(y)

    x.plot(self.xAxis, "orange", lw=4)
    y.plot(self.xAxis, "orange", lw=4)

def find_feasible_points(self):
    pass

def fill_feasible(self):

    self.find_feasible_points()

    # Create the x and y Coordinates for the fill area
    x = [i[0] for i in self.v]
    y = [i[1] for i in self.v]

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

def create_figure(self, name: str = None):
    if name:
        self.name = name
    if self.figsize:
        plt.figure(self.name, figsize=self.figsize)
    else:
        plt.figure(self.name)

    self.plt.ylim(*self.ylim)
    self.plt.xlim(*self.xlim)
    self.plt.grid()
    self.plot_equations()
    self.fill_feasible()

def check_feasible_point(self, points):
    p = Point(points)
    polygon = Polygon(self.v)
    return polygon.contains(p)

def intersections_in_feasible(self, extra):
    for l in self.lines:

```

```

        possible = extra.intersection(l, plotting=False)
        if self.check_feasible_point(possible):
            extra.intersection(l)

def graphical_solution(self, a, b, minlim, maxlim, legend=True):

    counter = minlim
    while counter < maxlim:
        extra = line(a, b, counter, "extra")
        extra.plot(self.xAxis, "cornflowerblue")
        extra.legend_show = False
        self.intersections_in_feasible(extra)
        counter += self.step

def save_image(self, file_name):
    if self.save_images:
        img_folder = Path(self.parent, "img")

        image_file = Path(img_folder, file_name)
        self.plt.savefig(image_file, dpi="figure")

```

exercise_01.py

```

import matplotlib.pyplot as plt
import sys
from pathlib import Path

from plt_line import line
from linearGui import LinearGUI

class Exerc01Gui(LinearGUI):
    def __init__(self, *args, **kwargs) -> None:
        limit = 30

        kwargs["limit"] = limit
        super().__init__(*args, **kwargs)
        self.lines = []
        self.lines.append(line(1, 1, 10, "l1", type="line"))
        self.lines.append(line(-10, 1, 10, "l2", type="line"))
        self.lines.append(line(-4, 1, 20, "l3", type="line"))
        self.lines.append(line(1, 4, 20, "l4", type="line"))

    def find_feasible_points(self):

        # Intersection lines 1 and 2
        self.v.append(self.lines[0].intersection(self.lines[1]))

        # Intersection lines 2 and 3
        self.v.append(self.lines[2].intersection(self.lines[1]))

        # Upper left limit corner
        self.v.append([self.lines[2](self.limit, reverse=True), self.limit])

        # Upper right corner
        self.v.append([self.limit, self.limit])

        # Lower right corner
        self.v.append([self.limit, 0])

        # Intersection x_axis and line 4
        x_axis = [l for l in self.lines if l.name == "x-axis"][0]
        self.v.append(self.lines[3].intersection(x_axis))

        # Intersection line 4 and line 1
        self.v.append(self.lines[0].intersection(self.lines[3]))

    def save_image(self, file_name):
        if self.save_images:
            parent = Path(__file__).parent
            img_folder = Path(parent, "img")

            image_file = Path(img_folder, file_name)
            self.plt.savefig(image_file, dpi="figure")

    def feasible_region(self):
        self.create_figure()
        self.save_image("exerc01_a.png")

```

```

def exerc01_a(self):
    self.name = "Z: min 2x1-x2"
    self.create_figure()
    self.step = 5
    self.graphical_solution(2, -1, -25, 5)
    self.save_image("exerc01_a_1.png")

def exerc01_b(self):
    self.name = "Z: min 11x1-x2"
    self.create_figure()
    self.step = 5
    self.graphical_solution(11, -1, -15, 5)
    self.save_image("exerc01_b.png")

def exerc01_c(self):
    self.name = "Z: min c1x1-x2"

    top = self.lines[0].intersection(self.lines[3], plotting=False)
    self.lines.append(line(-1.5, -1, -10, "l1_2", type="line"))
    self.lines.append(line(-0.1, -1, -5, "l4_2", type="line"))

    self.create_figure()
    self.save_image("exerc01_c.png")

def main(self):

    self.save_images = False
    self.feasible_region()
    self.exerc01_a()
    self.exerc01_b()
    self.exerc01_c()
    self.show()

if __name__ == "__main__":
    try:
        gui = Exerc01Gui(plt, name="Feasible Region", figsize=(15, 8))
        gui.main()
    except KeyboardInterrupt:
        pass
    except Exception as e:
        print(e)

```

Solution Exercise 2

Exercise 2

Ενας φρουτοχυμός που παράγει η εταιρεία FRESH περιέχει το πολύ 3 μέρη χυμό πορτοκαλιού και τουλάχιστον 1 μέρος χυμό μήλου, σύμφωνα με τα αναγραφόμενα στη συσκευασία του. Η εβδομαδιαία δυναμικότητα της εταιρείας για το συγκεκριμένο προϊόν δεν ξεπερνάει τα 500 lit ενώ σύμφωνα με τα δεδομένα των πωλήσεων είναι σίγουρο ότι η αγορά μπορεί να απορροφήσει τουλάχιστον 400 lit κάθε εβδομάδα. Για την επόμενη εβδομάδα η εταιρεία μπορεί να διαθέσει μέχρι 250 lit χυμό μήλου και το τμήμα παραγωγής στοχεύει να παράξει φρουτοχυμό που θα περιέχει τουλάχιστον 40% χυμό πορτοκαλιού. Το κόστος για 1 lit χυμό πορτοκαλιού είναι 2 χ.μ. (χρηματικές μονάδες) και για 1 lit χυμό μήλου είναι 1 χ.μ. ενώ η εταιρεία πουλάει τον φρουτοχυμό (ανεξαρτήτου αναλογίας των δύο φρούτων) προς 5 χ.μ.

1. a) Βοηθήστε το τμήμα παραγωγής της εταιρείας FRESH να αποφασίσει για τις ποσότητες χυμού που θα πρέπει να παράγει την επόμενη εβδομάδα όταν αντικειμενικός στόχος είναι η μεγιστοποίηση του κέρδους. Μοντελοποιήστε το παραπάνω σενάριο ως πρόβλημα γραμμικού προγραμματισμού και λύστε το γραφικά. Περιγράψτε αναλυτικά τις μεταβλητές απόφασης, την αντικειμενική συνάρτηση και τους περιορισμούς. Περιγράψτε τη βέλτιστη λύση αναφορικά με τους περιορισμούς, δηλ. ποιους ικανοποιεί οριακά (δεσμευτικοί περιορισμοί) και ποιους χαλαρά (μη δεσμευτικοί περιορισμοί).
2. b) Αν αντίθετα το τμήμα παραγωγής στοχεύσει να παράξει φρουτοχυμό που θα περιέχει τουλάχιστον 50%/60% χυμό πορτοκαλιού, τι επιπτώσεις θα έχει αυτή η απόφασή τους στη βέλτιστη λύση του προβλήματος;

2a

Μεταβλητές Αποφασής :

- x : Ποσότητα χυμου πορτοκαλιου σε lit
- y : Ποσότητα χυμου μηλου σε lit

Το κόστος για την παραγωγή τους εκφράζεται απο την σχέση :

$$2 \cdot x + y$$

καθώς 1l χυμο πορτοκαλι κοστιζει 2 χ.μ. και 1l χυμος μηλου κοστιζει 1 χμ

Απο τον εβδομαδιαιο περιορισμο παραγωγης, βγαζουμε οτι

$$x + y \leq 500$$

Επισης βλεπουμε οτι εφοσον το υποστηριζει η αγορα, πρεπει να παραξουμε τουλαχιστον 400l αρα :

$$x + y \geq 400$$

Τα εσοδα απο τον χυμο ειναι 5 χ.μ. επι την ποσοτητα του χυμου που παραγουμε, αρα

$$5 \cdot (x + y)$$

Θελουμε να μεγιστοποιησουμε το κερδος, το οποιο ειναι η διαφορα των εσοδων απο τα εξοδα :

$$5(x + y) - (2x + y) = 3x + 4y$$

Ακομη γραφει οτι η ο φρουτοχυμος πρεπει να περιεχει το πολυ 3 μερη χυμο πορτοκαλιου και τουλαχιστον 1 μερος χυμο μηλου

Για να βρουμε το ποσοστο του χυμου πορτοκαλιου χρησησιμοποιουμε τον τυπο :

$$\frac{x}{x + y}$$

Οποτε :

$$\frac{x}{x + y} \leq \frac{3}{4} \implies x - 3y \leq 0$$

Και για το μηλο :

$$\frac{y}{x + y} \geq \frac{1}{4} \implies x - 3y \leq 0$$

το μηλο για την επομενη εβδομαδα μπορει να ειναι μεχρι 250l

$$y \leq 250$$

Τελος μας δινει οτι εχουν στοχο το φρουτοποτο να περιεχει τουλαχιστον 40% χυμο πορτοκαλι :

$$\frac{x}{x + y} \geq 0.4 \implies 0.6x - 0.4y \geq 0 \implies 6x - 4y \geq 0 \implies 3x - 2y \geq 0$$

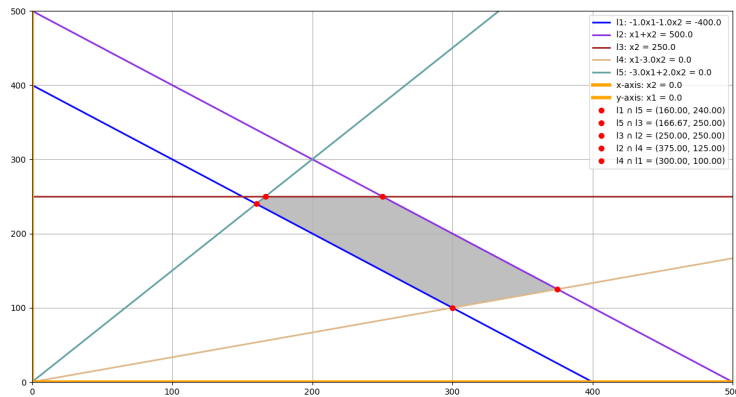
Το μοντελο γραμμικου προγραμματισμου ειναι :

$$\begin{array}{llll} \max Z : & 3x & + & 4y \\ \text{st} & & & \\ & x & + & y \geq 400 \quad (I1) \\ & x & + & y \leq 500 \quad (I2) \\ & & & y \leq 250 \quad (I3) \\ & x & - & 3y \leq 0 \quad (I4) \\ & 3x & - & 2y \geq 0 \quad (I5) \end{array}$$

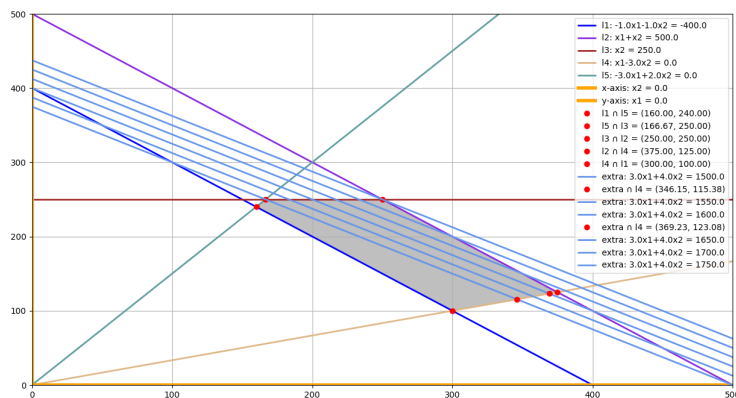
Το οποιο το μετατρεπουμε σε :

$$\begin{aligned}
 \max Z : \quad & 3x + 4y \\
 \text{st} \quad & \\
 & -x - y \leq 400 \quad (I1) \\
 & x + y \leq 500 \quad (I2) \\
 & y \leq 250 \quad (I3) \\
 & x - 3y \leq 0 \quad (I4) \\
 & -3x + 2y \leq 0 \quad (I5)
 \end{aligned}$$

Παίρνουμε την εφικτή περιοχή, και για τη γραφική αναπαράσταση έχουμε $x_1 = x, x_2 = y$



Ψαχνοντας να το λυσουμε γραφικα :



Βλεπουμε οτι η βελτιστη κορυφη ειναι το σημειο τομης του Π3 και Π2 [250,250]

Αυτο σημαίνει οτι χρησιμοποιωντας 50% χυμο πορτοκαλι και 50% χυμο μηλο θα εχουν το μεγαλυτερο κερδος στις 1750 χ.μ.

Πληθρουνται ολοι οι περιορισμοι.

2b

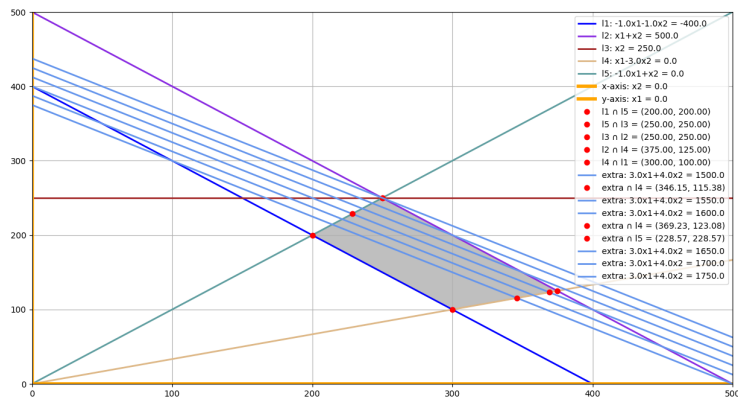
2. b) Αν αντίθετα το τμήμα παραγωγής στοχεύσει να παράξει φρουτοχυμό που θα περιέχει τουλάχιστον 50%/60% χυμό πορτοκαλιού, τι επιπτώσεις θα έχει αυτή η απόφασή τους στη βέλτιστη λύση του προβλήματος;

Για να έχουμε τουλάχιστον 50% χυμο πορτοκαλι αλλαζουμε τον περιορισμο Π5. Βλεπουμε οτι το 50% ειναι κατω απο το 75% του περιορισμου Π4. Οποτε εχουμε :

$$\frac{x}{x+y} \geq 0.5 \implies \frac{x}{x+y} \geq \frac{1}{2} \implies x - y \geq 0 \implies -x + y \leq 0$$

$$\begin{aligned}
 \max Z : \quad & 3x + 4y \\
 \text{st} \quad & \\
 & -x - y \leq 400 \quad (I1) \\
 & x + y \leq 500 \quad (I2) \\
 & y \leq 250 \quad (I3) \\
 & x - 3y \leq 0 \quad (I4) \\
 & -x + y \leq 0 \quad (I5)
 \end{aligned}$$

Κανουμε τη γραφικη παρασταση :



Παρατηρούμε ότι η βελτιστή κορυφή δεν έχει αλλάξει και είναι η τομή του περιορισμού Π3 και Π2 [250,250]

Για να έχει τουλάχιστον 60% χυμο πορτοκαλι αλλάζουμε πάλι τον περιορισμό Π5

$$\frac{x}{x+y} \geq 0.6 \implies x \geq 0.6x + 0.6y \implies -2x + 3y \leq 0$$

$$\begin{aligned} \max Z : \quad & 3x + 4y \\ \text{st} \end{aligned}$$

$$-x - y \leq -400 \quad (\Pi 1)$$

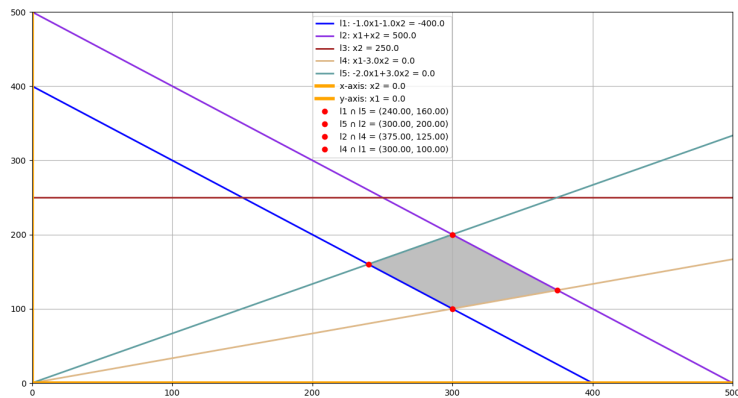
$$x + y \leq 500 \quad (\Pi 2)$$

$$y \leq 250 \quad (\Pi 3)$$

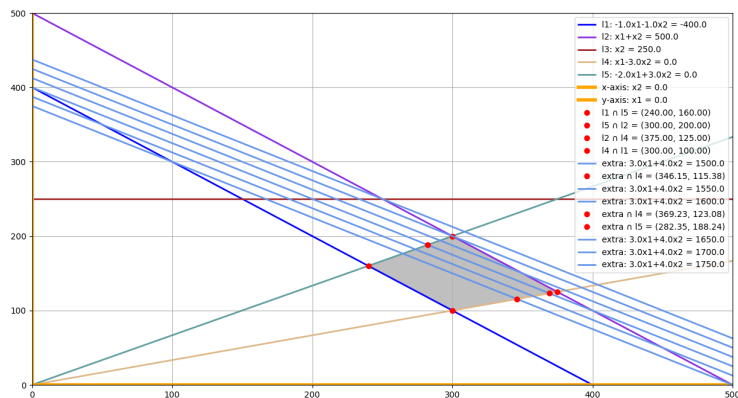
$$x - 3y \leq 0 \quad (\Pi 4)$$

$$-2x + 3y \leq 0 \quad (\Pi 5)$$

Η εφικτή περιοχή είναι πλέον :



Εφαρμόζουμε την αντικειμενική συνάρτηση :

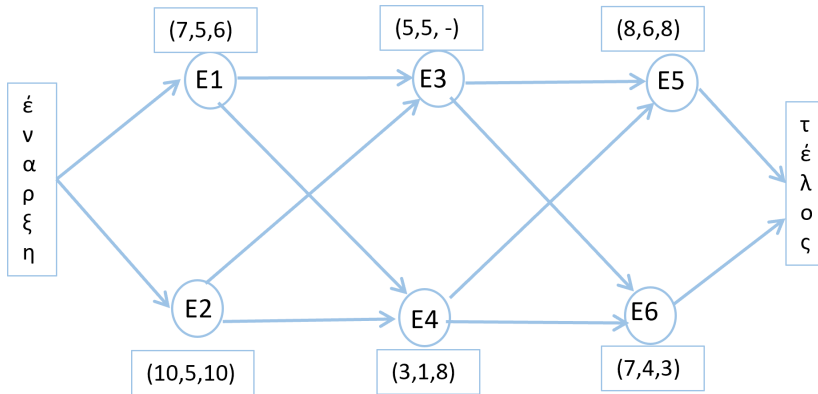


Και η βελτιστή κορυφή βρίσκεται στην τομή των περιορισμών Π5 και Π2 [300,200]

Solution Exercise 3

Exercise 3

Θεωρήστε ένα έργο το οποίο για να ολοκληρωθεί απαιτεί την διεκπεραίωση 6 επί μέρους εργασίες (E1 - E6). Οι εργασίες είναι εξαρτημένες μεταξύ τους και οι εξαρτήσεις δίνονται με το παρακάτω σχήμα:



Σύμφωνα με το σχήμα οι εργασίες E1 και E2 μπορούν να εκτελεστούν παράλληλα, όμως για να εκτελεστεί η εργασία E3 θα πρέπει να έχουν ολοκληρωθεί οι εργασίες E1 και E2.

Ομοίως και για τις υπόλοιπες εργασίες. Το έργο ολοκληρώνεται όταν εκτελεστούν (παράλληλα) οι εργασίες E5 και E6. Για κάθε εργασία δίνονται ο κανονικός χρόνος διεκπεραίωσης της εργασίας (σε εβδομάδες), το απόλυτο ελάχιστο για τον χρόνο αυτό, και το κόστος που θα προκύψει αν προσπαθήσουμε να μειώσουμε τον κανονικό χρόνο κατά μία εβδομάδα.

Η εταιρεία που έχει αναλάβει το έργο ενδιαφέρεται να μειώσει τον συνολικό χρόνο διεκπεραίωσης του (αν είναι εφικτό) σε 19 εβδομάδες, επομένως η διάρκεια μίας ή περισσότερων εργασιών θα πρέπει να μειωθεί σε σχέση με την κανονική τους διάρκεια. Προφανώς ο στόχος αυτός θα πρέπει να επιτευχθεί με το μικρότερο δυνατό κόστος.

Μοντελοποιήστε το συγκεκριμένο σενάριο προγραμματισμού εργασιών ως πρόβλημα γραμμικού προγραμματισμού. Ορίστε κατάλληλες μεταβλητές απόφασης και διαμορφώστε τους περιορισμούς όπως περιγράφονται στο σχήμα. Περιγράψτε και μοντελοποιήστε τον αντικειμενικό στόχο της εταιρείας. Δώστε την αντικειμενική συνάρτηση του προβλήματος γραμμικού προγραμματισμού. (Σημ. Η άσκηση δεν ζητάει τη λύση του προβλήματος, μόνο τη μοντελοποίησή του.)

Απο τις πληροφορίες του προβλήματος κρατάμε ότι :

Τα ζευγη (E1,E2), (E3,E4), (E5,E6) μεταξύ τους σαν ζευγη μπορούν να τρεξουν ταυτοχρονα. Για να ξεκινήσει το επομενο ζευγος στην σειρα πρεπει να εχουν τελειώσει και τα δυο μελη του προηγουμενου ζευγους.

Αναπαριστούμε με t τους χρόνους της κάθε διεργασίας, και βρίσκουμε το μέγιστο και ελάχιστο χρόνο που μπορούν να διαρκέσουν

$$\begin{aligned}7 &\geq t_1 \geq 5 \\10 &\geq t_2 \geq 5 \\t_3 &= 5 \\3 &\geq t_4 \geq 1 \\8 &\geq t_5 \geq 6 \\7 &\geq t_6 \geq 4\end{aligned}$$

Με c θεωρούμε το βάρος κόστους της κάθε βελτίωσης του χρόνου διεργασίας :

$$\begin{aligned}c_1 &= 6 \\c_2 &= 10 \\c_3 &= 0 \\c_4 &= 8 \\c_5 &= 8 \\c_6 &= 3\end{aligned}$$

Με s αναπαριστούμε τον χρόνο που μπορούμε να αφαιρέσουμε από τον κανονικό

$$\begin{aligned}0 &\leq s_1 \leq 2 \\0 &\leq s_2 \leq 5 \\0 &\leq s_4 \leq 2 \\0 &\leq s_5 \leq 2 \\0 &\leq s_6 \leq 3\end{aligned}$$

Η διεργασία 3 πάντα θα παίρνει 5 βδομάδες και δεν μπορεί να ελαχιστοποιηθεί.

Το κόστος για την κάθε μια βελτίωση είναι :

$$\begin{aligned}cost_1 &= s_1 \cdot c_1 \\cost_2 &= s_2 \cdot c_2 \\cost_4 &= s_4 \cdot c_4 \\cost_5 &= s_5 \cdot c_5 \\cost_6 &= s_6 \cdot c_6\end{aligned}$$

Μεσα στους περιορισμούς πρεπει να βαλουμε και την ολοκληρωση ολης της διαδικασιας μεσα σε 19 εβδομαδες.

Θεωρουμε τους χρονους t_{12}, t_{34}, t_{56} που παιρνει το καθε ζευγος για να ολοκληρωθει. Μαζι με πιθανες βελτιωσεις οι ελαχιστοι και μεγιστοι χρονοι που παιρνει για να ολοκληρωθουν ειναι :

$$\begin{aligned} 5 &\leq t_{12} \leq 10 \\ t_{34} &= 5 \\ 6 &\leq t_{56} \leq 8 \end{aligned}$$

Επομενως θα θεωρησουμε σαν μεταβλητες αποφασης τους χρονους των βελτιωσεων s και θα παμε να μειωσουμε το συνολικο κοστος τους :

Μοντελοποιηση Προβληματος

$$\begin{aligned} \min \quad & Z : s_1 \cdot + s_1 \cdot 6 + s_2 \cdot 10 + s_3 \cdot 0 + s_4 \cdot 8 + s_5 \cdot 8 + s_6 \cdot 3 \\ \text{st} \quad & s_1 \leq 2 \\ & s_2 \leq 5 \\ & s_4 \leq 2 \\ & s_5 \leq 2 \\ & s_6 \leq 3 \\ & \max(7 - s_1, 10 - s_2) + 5 + \max(8 - s_5, 7 - s_6) \leq 19 \\ & s_1, s_2, s_3, s_4, s_5, s_6 \geq 0 \end{aligned}$$

Ο στοχος της αντικειμενικης συναρτησης ειναι να επιτευχθουν οι 19 βδομαδες εκτελεσης, με το λιγοτερο δυνατο κοστος.
Οι μεταβλητες αποφασης ειναι οι χρονοι που βελτιωνουν την διαρκεια εκτελεσης της καθε διαδικασιας.
Για το ζευγος (E3,E4) παντα ο χρονος θα ειναι 5, καθως το E3 δεν παιρνει βελτιωση και το E4 θα εκτελειται σε κανονικο χρονο 3, το οποιο θα ειναι παντα μικροτερο του 5.

Solution Exercise 4

4b

- (Π2) Το σύνολο $M = \{(x, y) \in R_{++}^2 | xy \geq k, k \in R\}$ είναι κυρτό σύνολο. (Υπόδειξη. Μπορείτε να χρησιμοποιήσετε ως γνωστό το Λήμμα ότι για θετικούς αριθμούς a και b ισχύει πάντα: $\frac{a}{b} + \frac{b}{a} \geq 2$)

Εχουμε το συνολο $M = \{(x, y) \in R_{++}^2 | xy \geq k, k \in R\}$

Θα παρουμε δυο σημεια $(x_1, y_1), (x_2, y_2)$ τα οποια ανηκουν στο συνολο M, δηλαδη $x_1 y_1 \geq k, x_2 y_2 \geq k$

Για να δειξουμε οτι το συνολο ειναι κυρτο συνολο θελουμε να δειξουμε οτι το σημειο $(\lambda x_1 + (1 - \lambda)x_2, \lambda y_1 + (1 - \lambda)y_2) \in M$

Για να ανηκει στο M θα πρεπει :

$$(\lambda x_1 + (1 - \lambda)x_2) \cdot (\lambda y_1 + (1 - \lambda)y_2) \geq k$$

Θα πιασουμε το αριστερο μελος :

$$\begin{aligned} & (\lambda x_1 + (1 - \lambda)x_2) \cdot (\lambda y_1 + (1 - \lambda)y_2) \implies \\ & \lambda^2 x_1 y_1 + \lambda(1 - \lambda)(x_1 y_2 + x_2 y_1) + (1 - \lambda)^2 x_2 y_2 \implies \\ & \lambda^2 x_1 y_1 + \lambda(1 - \lambda)(x_1 y_2 + x_2 y_1) + (1 - \lambda)^2 x_2 y_2 \end{aligned}$$

Θα εξετασουμε τα τρια μερη ξεχωριστα :

Ξεκινουμε απο την ιδιοτητα του σημειου (x_1, y_1) που ανηκει στο συνολο M και πολλαπλασιαζουμε με τον θετικο αριθμο λ^2

$$\begin{aligned} x_1 y_1 \geq k & \implies \\ \lambda^2 x_1 y_1 \geq \lambda^2 k & \end{aligned}$$

Ξεκινουμε απο την ιδιοτητα του σημειου (x_2, y_2) που ανηκει στο συνολο M και πολλαπλασιαζουμε με τον θετικο αριθμο $(1 - \lambda)^2$

$$\begin{aligned} x_2 y_2 \geq k & \implies \\ (1 - \lambda)^2 x_2 y_2 \geq (1 - \lambda)^2 k & \end{aligned}$$

Για το μεσαιο κομματι θα ξεκινήσουμε απο τις ιδιοτητες των σημειων :

$$\begin{aligned} x_1 y_1 \geq k & \implies x_1 \geq \frac{k}{y_1} \implies x_1 y_2 \geq k \cdot \frac{y_2}{y_1} \\ x_2 y_2 \geq k & \implies x_2 \geq \frac{k}{y_2} \implies x_2 y_1 \geq k \cdot \frac{y_1}{y_2} \end{aligned}$$

Μπορουμε να πολλαπλασιασουμε και να διαιρεσουμε με αυτα, καθως μιλαμε για αυστηρα θετικους αριθμους $(x, y \in R_{++})$

Προσθετουμε τις δυο σχεσεις μεταξυ τους :

$$\begin{aligned} x_1 y_2 + x_2 y_1 & \geq k \cdot \frac{y_2}{y_1} + k \cdot \frac{y_1}{y_2} \implies \\ (x_1 y_2 + x_2 y_1) & \geq k \left(\frac{y_2}{y_1} + \frac{y_1}{y_2} \right) \implies \\ \lambda(1 - \lambda)(x_1 y_2 + x_2 y_1) & \geq \lambda(1 - \lambda)k \left(\frac{y_2}{y_1} + \frac{y_1}{y_2} \right) \implies \\ \lambda(1 - \lambda)(x_1 y_2 + x_2 y_1) & \geq \lambda(1 - \lambda)k \left(\frac{y_2}{y_1} + \frac{y_1}{y_2} \right) \geq \lambda(1 - \lambda) \cdot k \cdot 2 \implies \\ \lambda(1 - \lambda)(x_1 y_2 + x_2 y_1) & \geq \lambda(1 - \lambda) \cdot k \cdot 2 \end{aligned}$$

Αξιοποιουμε την ιδιοτητα

$$\begin{aligned} \frac{y_1}{y_2} + \frac{y_2}{y_1} & \geq 2 \implies \\ k \cdot \frac{y_1}{y_2} + \frac{y_2}{y_1} & \geq 2k \implies \\ k \cdot \frac{y_1}{y_2} + \frac{y_2}{y_1} & \geq 2k \geq k \implies \\ k \cdot \frac{y_1}{y_2} + \frac{y_2}{y_1} & \geq k \end{aligned}$$

Καταληγουμε στις παρακατω 3 ανισοτητες, τις οποιες προσθετουμε :

$$\begin{aligned} \lambda^2 x_1 y_1 & \geq \lambda^2 k \\ \lambda(1 - \lambda)(x_1 y_2 + x_2 y_1) & \geq \lambda(1 - \lambda) \cdot k \cdot 2 \\ (1 - \lambda)^2 x_2 y_2 & \geq (1 - \lambda)^2 k \end{aligned}$$

$$\begin{aligned} \lambda^2 x_1 y_1 + \lambda(1 - \lambda)(x_1 y_2 + x_2 y_1) + (1 - \lambda)^2 x_2 y_2 & \geq \lambda^2 k + \lambda(1 - \lambda) \cdot k \cdot 2 + (1 - \lambda)^2 k \implies \\ \lambda^2 x_1 y_1 + \lambda(1 - \lambda)(x_1 y_2 + x_2 y_1) + (1 - \lambda)^2 x_2 y_2 & \geq k \cdot (\lambda^2 + \lambda(1 - \lambda) \cdot 2 + (1 - \lambda)^2) \implies \\ \lambda^2 x_1 y_1 + \lambda(1 - \lambda)(x_1 y_2 + x_2 y_1) + (1 - \lambda)^2 x_2 y_2 & \geq k \cdot (\lambda^2 + 2\lambda - 2\lambda^2 + 1 - 2\lambda + \lambda^2) \implies \\ \lambda^2 x_1 y_1 + \lambda(1 - \lambda)(x_1 y_2 + x_2 y_1) + (1 - \lambda)^2 x_2 y_2 & \geq k \end{aligned}$$

Επομενως το σημειο $(\lambda x_1 + (1 - \lambda)x_2, \lambda y_1 + (1 - \lambda)y_2) \in M$ για οποιαδηποτε δυο $(x_1, y_1), (x_2, y_2)$ το συνολο M ειναι κυρτο

Solution Exercise 5

Exercise 5

Ασκηση 5. Θεωρήστε το πρόβλημα γραμμικού προγραμματισμού:

$$\max_{\text{όταν}} \quad Z = -2x_1 + x_2 - 4x_3 + 3x_4$$

$$x_1 + x_2 + 3x_3 + 2x_4 \leq 4$$

$$x_1 - x_3 + x_4 \leq 2$$

$$2x_1 + x_2 \leq 3$$

$$x_1, x_2, x_3, x_4 \geq 0$$

- (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.

5a

Θελουμε να βρουμε ολες τις κορυφες που δημιουργονται απο τις τομες των υπερπιπεδων του

Είμαστε στο R^4 , αρα μια κορυφη ειναι τομη 4 υπερεπιπεδων. Εχουμε 7 υπερεπιπεδα αρα:

$$\binom{7}{4} = \frac{7!}{4!(7-4)!} = \frac{7!}{4!3!} = \frac{7 \cdot 6 \cdot 5}{3 \cdot 2 \cdot 1} = 35$$

Θα εχουμε 35 πιθανες κορυφες

Θα παρουμε τα παρακατω συστηματα :

$$\begin{aligned} \min \quad & Z = 12x_1 - 10x_2 \\ \text{st} \quad & x_1 + x_2 + 3x_3 + 2x_4 \leq 4 \\ & x_1 - x_3 + x_4 \leq 2 \\ & 2x_1 + x_2 \leq 3 \\ & x_1, x_2, x_3, x_4 \geq 0 \end{aligned}$$

Εχουμε 7 Περιορισμους, οποτε θα βρουμε ολους τους συνδυασμους που μπορουν να κανουν οι εξισωσεις, θα βρουμε την κοινή τους λύση και θα ελεγχουμε ποιες απο αυτες τις λυσεις εκπληρωνουν τους περιορισμους

Ελεγχουμε τα παρακατω συστηματα και παρνουμε τις λυσεις τους :

Λυσεις Συστηματος

	$x_1+x_2+3x_3+2x_4 = 4$	
	$x_1+0x_2-3x_3+4x_4 = 2$	
	$2x_1+x_2+0x_3+0x_4 = 3$	
	$x_1+0x_2+0x_3+0x_4 = 0$	

	Solution 0: [0. 3. 0. 0.5 0. 0. 0.]	Feasible: True

	$x_1+x_2+3x_3+2x_4 = 4$	
	$x_1+0x_2-3x_3+4x_4 = 2$	
	$2x_1+x_2+0x_3+0x_4 = 3$	
	$0x_1+x_2+0x_3+0x_4 = 0$	

	Solution 1: [1.5 0. 0.5 0. 0.5 0. 0.]	Feasible: True

	$x_1+x_2+3x_3+2x_4 = 4$	
	$x_1+0x_2-3x_3+4x_4 = 2$	
	$2x_1+x_2+0x_3+0x_4 = 3$	
	$0x_1+0x_2+x_3+0x_4 = 0$	

	Solution 2: [0. 3. 0. 0. 0. 0.5 0.]	Feasible: True

	$x_1+x_2+3x_3+2x_4 = 4$	
	$x_1+0x_2-3x_3+4x_4 = 2$	
	$x_1+0x_2+0x_3+0x_4 = 0$	
	$0x_1+x_2+0x_3+0x_4 = 0$	

| Solution 4: [0. -0. 0. 0.66666667 1. 0. 0.] | Feasible: False |

	$x_1+x_2+3x_3+2x_4 = 4$	
	$x_1+0x_2-3x_3+4x_4 = 2$	
	$x_1+0x_2+0x_3+0x_4 = 0$	
	$0x_1+0x_2+x_3+0x_4 = 0$	

| Solution 5: [0. 3. 0. 0. 0. 0.5 0.] | Feasible: True |

	$x_1+x_2+3x_3+2x_4 = 4$	
	$x_1+0x_2-3x_3+4x_4 = 2$	
	$x_1+0x_2+0x_3+0x_4 = 0$	
	$0x_1+0x_2+0x_3+x_4 = 0$	

| Solution 6: [0. 6. 0. -0.66666667 0. 0. 0.] | Feasible: False |

	$x_1+x_2+3x_3+2x_4 = 4$	
	$x_1+0x_2-3x_3+4x_4 = 2$	
	$0x_1+x_2+0x_3+0x_4 = 0$	
	$0x_1+0x_2+x_3+0x_4 = 0$	

| Solution 7: [6. -0. 0. 0. -0. -1. 0.] | Feasible: False |

	$x_1+x_2+3x_3+2x_4 = 4$	
	$x_1+0x_2-3x_3+4x_4 = 2$	
	$0x_1+x_2+0x_3+0x_4 = 0$	
	$0x_1+0x_2+0x_3+x_4 = 0$	

| Solution 8: [3. -0. 0. 0. 0.33333333 0. 0.] | Feasible: False |

	$x_1+x_2+3x_3+2x_4 = 4$	
	$x_1+0x_2-3x_3+4x_4 = 2$	
	$0x_1+0x_2+x_3+0x_4 = 0$	
	$0x_1+0x_2+0x_3+x_4 = 0$	

| Solution 9: [2. 2. 0. 0. 0. 0. 0.] | Feasible: False |

	$x_1+x_2+3x_3+2x_4 = 4$	
	$2x_1+x_2+0x_3+0x_4 = 3$	
	$x_1+0x_2+0x_3+0x_4 = 0$	
	$0x_1+0x_2+x_3+0x_4 = 0$	

| Solution 11: [0. 0. 3. 0. 0. 0.5 0.] | Feasible: False |

	$x_1+x_2+3x_3+2x_4 = 4$	
	$2x_1+x_2+0x_3+0x_4 = 3$	
	$x_1+0x_2+0x_3+0x_4 = 0$	
	$0x_1+0x_2+0x_3+x_4 = 0$	

| Solution 12: [0. 0. 3. 0.33333333 0. 0. 0.] | Feasible: False |

```
-----
|          x1+x2+3x3+2x4 = 4          |
|          2x1+x2+0x3+0x4 = 3          |
|          0x1+x2+0x3+0x4 = 0          |
|          0x1+0x2+x3+0x4 = 0          |
|-----|
```

```
| Solution 13: [1.5 0. 0. 0. 1.25 0. ] | Feasible: True |
|-----|
```

```
-----
|          x1+x2+3x3+2x4 = 4          |
|          2x1+x2+0x3+0x4 = 3          |
|          0x1+x2+0x3+0x4 = 0          |
|          0x1+0x2+0x3+x4 = 0          |
|-----|
```

```
| Solution 14: [1.5 0. 0. 0. 0.83333333 0. 0. ] | Feasible: True |
|-----|
```

```
-----
|          x1+x2+3x3+2x4 = 4          |
|          2x1+x2+0x3+0x4 = 3          |
|          0x1+0x2+x3+0x4 = 0          |
|          0x1+0x2+0x3+x4 = 0          |
|-----|
```

```
| Solution 15: [-1. 0. 5. 0. 0. 0. 0.] | Feasible: False |
|-----|
```

```
-----
|          x1+x2+3x3+2x4 = 4          |
|          x1+0x2+0x3+0x4 = 0          |
|          0x1+x2+0x3+0x4 = 0          |
|          0x1+0x2+x3+0x4 = 0          |
|-----|
```

```
| Solution 16: [ 0. 0. 0. -0. -0. 2. 0.] | Feasible: True |
|-----|
```

```
-----
|          x1+x2+3x3+2x4 = 4          |
|          x1+0x2+0x3+0x4 = 0          |
|          0x1+x2+0x3+0x4 = 0          |
|          0x1+0x2+0x3+x4 = 0          |
|-----|
```

```
| Solution 17: [ 0. 0. 0. -0. 1.33333333 0. 0. ] | Feasible: True |
|-----|
```

```
-----
|          x1+x2+3x3+2x4 = 4          |
|          x1+0x2+0x3+0x4 = 0          |
|          0x1+0x2+x3+0x4 = 0          |
|          0x1+0x2+0x3+x4 = 0          |
|-----|
```

```
| Solution 18: [0. 0. 0. 4. 0. 0. 0.] | Feasible: False |
|-----|
```

```
-----
|          x1+x2+3x3+2x4 = 4          |
|          0x1+x2+0x3+0x4 = 0          |
|          0x1+0x2+x3+0x4 = 0          |
|          0x1+0x2+0x3+x4 = 0          |
|-----|
```

```
| Solution 19: [4. 0. 0. 0. 0. 0. 0.] | Feasible: False |
|-----|
```

```
-----
|          x1+0x2-3x3+4x4 = 2          |
|          2x1+x2+0x3+0x4 = 3          |
|          x1+0x2+0x3+0x4 = 0          |
|          0x1+0x2+x3+0x4 = 0          |
|-----|
```

| Solution 21: [0. 0. 3. 0. 0. 0.5 0.] | Feasible: False |

	$x_1 + 0x_2 - 3x_3 + 4x_4 = 2$	
	$2x_1 + x_2 + 0x_3 + 0x_4 = 3$	
	$x_1 + 0x_2 + 0x_3 + 0x_4 = 0$	
	$0x_1 + 0x_2 + 0x_3 + x_4 = 0$	

| Solution 22: [0. 0. 3. -0.66666667 0. 0. 0.] | Feasible: False |

	$x_1 + 0x_2 - 3x_3 + 4x_4 = 2$	
	$2x_1 + x_2 + 0x_3 + 0x_4 = 3$	
	$0x_1 + x_2 + 0x_3 + 0x_4 = 0$	
	$0x_1 + 0x_2 + x_3 + 0x_4 = 0$	

| Solution 23: [0. 1.5 0. 0. -0. 0.125 0.] | Feasible: True |

	$x_1 + 0x_2 - 3x_3 + 4x_4 = 2$	
	$2x_1 + x_2 + 0x_3 + 0x_4 = 3$	
	$0x_1 + x_2 + 0x_3 + 0x_4 = 0$	
	$0x_1 + 0x_2 + 0x_3 + x_4 = 0$	

| Solution 24: [0. 1.5 0. 0. -0.16666667 0. 0.] | Feasible: True |

	$x_1 + 0x_2 - 3x_3 + 4x_4 = 2$	
	$2x_1 + x_2 + 0x_3 + 0x_4 = 3$	
	$0x_1 + 0x_2 + x_3 + 0x_4 = 0$	
	$0x_1 + 0x_2 + 0x_3 + x_4 = 0$	

| Solution 25: [0. 2. -1. 0. 0. 0. 0.] | Feasible: False |

	$x_1 + 0x_2 - 3x_3 + 4x_4 = 2$	
	$x_1 + 0x_2 + 0x_3 + 0x_4 = 0$	
	$0x_1 + x_2 + 0x_3 + 0x_4 = 0$	
	$0x_1 + 0x_2 + x_3 + 0x_4 = 0$	

| Solution 26: [0. 0. 0. 0. 0. 0.5 0.] | Feasible: True |

	$x_1 + 0x_2 - 3x_3 + 4x_4 = 2$	
	$x_1 + 0x_2 + 0x_3 + 0x_4 = 0$	
	$0x_1 + x_2 + 0x_3 + 0x_4 = 0$	
	$0x_1 + 0x_2 + 0x_3 + x_4 = 0$	

| Solution 27: [0. 0. 0. 0. -0.66666667 0. 0.] | Feasible: True |

	$x_1 + 0x_2 - 3x_3 + 4x_4 = 2$	
	$0x_1 + x_2 + 0x_3 + 0x_4 = 0$	
	$0x_1 + 0x_2 + x_3 + 0x_4 = 0$	
	$0x_1 + 0x_2 + 0x_3 + x_4 = 0$	

| Solution 29: [0. 2. 0. 0. 0. 0. 0.] | Feasible: True |

	$2x_1 + x_2 + 0x_3 + 0x_4 = 3$	
	$x_1 + 0x_2 + 0x_3 + 0x_4 = 0$	

	0x1+0x2+x3+0x4 = 0	
	0x1+0x2+0x3+x4 = 0	

	Solution 32: [0. 0. 0. 3. 0. 0. 0.]	Feasible: False

	2x1+x2+0x3+0x4 = 3	
	0x1+x2+0x3+0x4 = 0	
	0x1+0x2+x3+0x4 = 0	
	0x1+0x2+0x3+x4 = 0	

	Solution 33: [0. 0. 1.5 0. 0. 0. 0.]	Feasible: False

	x1+0x2+0x3+0x4 = 0	
	0x1+x2+0x3+0x4 = 0	
	0x1+0x2+x3+0x4 = 0	
	0x1+0x2+0x3+x4 = 0	

	Solution 34: [0. 0. 0. 0. 0. 0. 0.]	Feasible: True

Βλεπουμε οτι μονο οι κορυφες :

$$K_0, K_1, K_2, K_5, K_{13}, K_{14}, K_{16}, K_{17}, K_{23}, K_{24}, K_{26}, K_{27}, K_{29}, K_{34}$$

Ικανοποιουν ολους τους περιορισμους

Solution 0	:	[0.	3.	0.	0.5	0.	0.	0.]	Feasible: True
Solution 1	:	[1.5	0.	0.5	0.	0.5	0.	0.]	Feasible: True
Solution 2	:	[0.	3.	0.	0.	0.	0.5	0.]	Feasible: True
Solution 5	:	[0.	3.	0.	0.	0.	0.5	0.]	Feasible: True
Solution 13	:	[1.5	0.	0.	0.	0.	1.25	0.]	Feasible: True
Solution 14	:	[1.5	0.	0.	0.	0.83	0.	0.]	Feasible: True
Solution 16	:	[0.	0.	0.	-0.	-0.	2.	0.]	Feasible: True
Solution 17	:	[0.	0.	0.	-0.	1.33	0.	0.]	Feasible: True
Solution 23	:	[0.	1.5	0.	0.	-0.	0.125	0.]	Feasible: True
Solution 24	:	[0.	1.5	0.	0.	-0.167	0.	0.]	Feasible: True
Solution 26	:	[0.	0.	0.	0.	0.	0.5	0.]	Feasible: True
Solution 27	:	[0.	0.	0.	0.	-0.667	0.	0.]	Feasible: True
Solution 29	:	[0.	2.	0.	0.	0.	0.	0.]	Feasible: True
Solution 34	:	[0.	0.	0.	0.	0.	0.	0.]	Feasible: True

Δεν υπαρχουν εκφυλισμενες κορυφες.

Python Script

Tools>equation_class.py

```
class equation:
    def __init__(self, *args, **kwargs):
        self.name = kwargs.get("name", "equation")
        self.coefficients = args[:-1]
        self.b = args[-1]

    def __str__(self):
        output = []

        for i, coef in enumerate(self.coefficients):
            if coef == 1:
                temp = f"x{i+1}"
            elif coef == -1:
                temp = f"-x{i+1}"
            else:
                temp = f"{str(coef)}x{i+1}"
            output.append(temp)

        result = output[0]

        for item in output[1:]:
            if item[0] == "-":
                result += item
            else:
                result += "+" + item
        result += f" = {self.b}"

        return result

    def __call__(self, *args, reverse=False):
        return self.equation(*args)

    def equation(self, *args):
        return sum([coef * x for coef, x in zip(self.coefficients, args)])

if __name__ == "__main__":
    l1 = equation(1, 1, 1, 1, name="l1")
    l2 = equation(1, 1, 1, 2, name="l2")
    print(l1)
```

Tools>equation_solver.py

```
import sys
from pathlib import Path
from itertools import combinations
from scipy.special import binom
import numpy as np

sys.path.append(str(Path(__file__).parent))
from equation_class import equation

class EquationSolver:
    def __init__(self, *args, **kwargs) -> None:
        """
        Description:

        Args:
            equations (list): the equations
            coefficients (list): the coefficients of the equations
        """
        self.equations = kwargs.get("equations", [])
        self.coefficients = kwargs.get("coefficients", [])
        self.current_system = None
        self.build_all_equations()
        self.parent = kwargs.get("parent", Path(__file__).parent)

    def build_all_equations(self):
        if self.equations:
            self.coefficients = [eq.coefficients for eq in self.equations]

        if self.coefficients:
            self.equations = [
```



```

        equation(*coef, name=f"line{i}")
        for i, coef in enumerate(self.coefficients)
    ]

    n = len(self.equations[0].coefficients) + 1

    positive_constraints = [
        [0 if i != j else 1 for i in range(n)] for j in range(n - 1)
    ]

    positive_constraints_eq = [
        equation(*item, name=f"pos_x{i}")
        for i, item in enumerate(positive_constraints)
    ]

    self.equations.extend(positive_constraints_eq)

    self.equations = np.array(self.equations)

    self.num_coefficients = len(self.equations[0].coefficients)
    self.num_equations = len(self.equations)

def checker(self, *args):
    results = []
    for eq in self.equations:
        if eq.name.startswith("pos_x"):
            results.append(eq(*args) >= 0)
        else:
            results.append(eq(*args) <= eq.b)
    # results = [eq(*args) <= eq.b for eq in self.equations]
    return all(results)

def number_of_tops(self):
    """
    Description:
        Returns the number of tops that can be found using binomial theorem
    Returns:
        int: The number of tops that can be found
    """

    # In the constraints we also have that x_i >= 0
    return binom(self.num_coefficients + self.num_equations, self.num_equations)

def get_combinations(self):
    """
    Description:
        Get the combinations of the equations that can be used to solve the system using itertools combination
    """
    self.num_coefficients = len(self.equations[0].coefficients)
    self.num_equations = len(self.equations)
    self.combination_index = list(range(self.num_equations))
    self.combos = [
        x for x in combinations(self.combination_index, self.num_coefficients)
    ]

def get_system_solution(self, system: list):
    """
    Description:
        Get the solution of the system of equations given the indexes of the equations to be used

    Args:
        system (list): The indexes of the equations to be used

    Returns:
        list: The solutions of the system equations or None if it's not solvable
    """

    # System, contains the indexes of the equations that were chosen
    system = np.array(system)
    self.current_system = system
    # Get a list of the equations chosen
    equations = list(np.array(self.equations)[system])

    # Build the matrix A, with the coefficients of the equations used
    a = np.array([np.array(eq.coefficients) for eq in equations])

    # Stack the equations in a row
    a_stacked = np.row_stack(a)

    # Check if the determinant of the matrix is 0
    if np.linalg.det(a_stacked) == 0:

```

```

        # It is non Solvable
        return None

    # Build the matrix B, with the b values of the equations used
    b = np.array([np.array(eq.b) for eq in equations])

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

    # Saving the variables that were used
    result_all_var = np.zeros((len(self.equations)))

    # Assign values to the variables that were used in the correct order
    result_all_var[system] = result

    return result_all_var

def tops(self):
    self.get_combinations()
    num_of_vars = len(self.equations[0].coefficients)
    for lis in self.combos:
        solution = self.get_system_solution(lis)
        if solution is not None:
            print(
                f"{'-'*10}\n{solution} | {self.checker(*solution[:num_of_vars])}\n{'-'*10}"
            )

def str_single(self, *args, **kwargs):
    """
    Description:
        Returns the string representation of the solution

    Args:
        *args: list
            The solution to be printed
        **kwargs: dict
            solution: list
                The solution to be printed
            combo: list
                The indexes of the equations that were used to solve the system

    Returns:
        str: The string representation of the solution
    """

    if args:
        solution = args[0]
    else:
        solution = kwargs.get("solution", None)
    output = []

    equations_combos = kwargs.get("combo", None)
    ind = kwargs.get("ind", "")

    if solution is not None:
        # Creates a string for the solution.
        solution_ack = f"| Solution {ind}: {str(solution).replace(' ', ' ').replace(',', ' ').replace(' ', ' ')} |"
        solution_ack += (
            f" Feasible: {self.checker(*solution[:self.num_coefficients])} |"
        )

    # Get the length of the solution string to prettify the output
    length = len(solution_ack)

    # If the equation combo is provided it will be printed
    if equations_combos:
        equations = [str(self.equations[i]) for i in equations_combos]
        lengths = [len(eq) + 4 for eq in equations]
        lengths.append(length)
        max_length = max(lengths)

        output.append("-" * max_length)

        equations = [f"| {eq:^{max_length-4}} |" for eq in equations]
        output.append("\n".join(equations))

    output.append("-" * max_length)
    output.append(solution_ack)
    output.append("-" * max_length + "\n")

```

```

        return "\n".join(output)

def feasible_tops(self):
    self.get_combinations()
    output = []
    for i, lis in enumerate(self.combos):

        # Get the solution of the system
        solution = self.get_system_solution(lis)

        # Check if the solution exists
        if solution is not None:
            # If it exists, it will be printed
            if self.checker(*solution[: self.num_coefficients]):
                output.append(f"Solution {i} : {solution} | Feasible: True")

    print("\n".join(output))

def main(self):

    self.get_combinations()
    output = []
    for i, lis in enumerate(self.combos):

        # Get the solution of the system
        solution = self.get_system_solution(lis)

        # Check if the solution exists
        if solution is not None:
            # If it exists, it will be printed
            output.append(self.str_single(solution, combo=lis, ind=i))

    print("\n".join(output))
    file = Path(self.parent, "output_exerc_05_a.txt")
    with open(file, "w") as f:
        f.write("\n".join(output))

if __name__ == "__main__":
    try:
        coef = [
            [3, 2, 0, 5],
            [2, 1, 2, 5],
        ]
        eq = EquationSolver(coefficients=coef)
        eq.main()

    except KeyboardInterrupt:
        pass
    except Exception as e:
        print(e)

```

exercise_05.py

```

import matplotlib.pyplot as plt
import sys
from pathlib import Path

sys.path.append(str(Path(__file__).parents[2]))
from Tools.equation_solver import EquationSolver

class Exerc05(EquationSolver):
    def __init__(self, *args, **kwargs) -> None:
        super().__init__(*args, **kwargs)
        self.parent = Path(__file__).parent

if __name__ == "__main__":
    try:
        obj = ["max", -2, 1, -4, 3]
        coef = [ [1, 1, 3, 2, 4], [1, 0, -3, 4, 2], [2, 1, 0, 0, 3] ]
        exerc = Exerc05(coefficients=coef)
        exerc.main()
    except KeyboardInterrupt:
        pass
    except Exception as e:

```


Solution Exercise 6

Περιγραφή Simplex

1. Επιλογές στηλών πίνακα A των συντελεστών
2. Μετασχηματισμούς για να βρεθούν οι αντίστοιχες βασικές εφικτές λύσεις :
 1. στον πίνακα
 2. στην αντικειμενική συνάρτηση
 3. στα διανύσματα δεξιού μέρους
3. Εκκίνηση του αλγορίθμου από μια κορυφή της εφικτής περιοχής, συνήθως αρχή αξόνων (εφόσον είναι εφικτή λύση)
4. Μετακινείται μετά σε μια γειτονική κορυφή που δίνει καλύτερη (οχι χειρότερη) τιμή της αντικειμενικής συνάρτησης
5. Επαναλαμβάνεται για έναν πεπερασμένο αριθμό επαναληψεων και σταματάει όταν η αντικειμενική συνάρτηση δεν μπορεί να βελτιωθεί περαιτέρω

Βήματα Simplex

Βήματα Simplex

1. Αρχικοποίηση
 1. Εισαγώνται μεταβλητές χαλάρωσης και δημιουργείται μια πρώτη βασική εφικτή λύση του προβλήματος ($x=0, x_s = b$)
2. Απαλοιφή Gauss
 1. Μια βασική εφικτή λύση $\begin{bmatrix} x_B \\ x_N \end{bmatrix}$ αντιστοιχεί στον βασικό πίνακα B για τον οποίο $B^{-1}b \geq 0$ Με απαλοιφή μετασχηματίζονται όλοι οι πίνακες :
 1. $[B \quad N \quad b] \rightarrow [I_m^B \quad B^{-1}N \quad B^{-1}b]$
 2. $\begin{bmatrix} c_B^T & c_N^T \end{bmatrix} \rightarrow \begin{bmatrix} 0^T & c_N^T - c_B^T B^{-1}N \end{bmatrix}$
 3. Η αντικειμενική Συνάρτηση παίρνει την καινούργια τιμή $z = c_B^T B^{-1}b$ και πηγαίνουμε στο επομένο βήμα.
 4. Αν δεν υπάρχει πίνακας B για τον οποίο $B^{-1}b \geq 0$ σταματάμε, δεν υπάρχουν εφικτές λύσεις
 3. Επιλογή νέας βασικής μεταβλητής
 1. Επιλέγεται μεταβλητή εστω x_a με θετικό συντελεστή στην αντικειμενική συνάρτηση που ονομάζεται εισερχόμενη βασική μεταβλητή. Αν δεν υπάρχουν θετικοί αντικειμενικοί συντελεστές σταματάμε. Η λύση είναι βελτιστή
 4. Κριτήριο ελαχίστου λόγου
 1. στην στήλη του $B^{-1}N$ που αντιστοιχεί στην x_a , βρίσκουμε εκείνον τον δείκτη k για τον οποίο:
$$\frac{(B^{-1}b)_k}{(B^{-1}N)_{ka}} = \min \left\{ \frac{(B^{-1}b)_j}{(B^{-1}N)_{ja}} \mid j \in \{1, \dots, m\}, (B^{-1}N)_{ja} > 0 \right\}$$
 2. Εστω β ο δείκτης της στήλης για την οποία $(I_m^B)_{k\beta} = 1$. Η μεταβλητή x_β ονομάζεται εξερχόμενη μεταβλητή
 3. Αν $(B^{-1}N)_{k\alpha} \leq 0, \forall k$ σταματάμε. Το πρόβλημα δεν είναι φραγμένο. Πηγαίνουμε στο επομένο βήμα
 5. Ανταλλαγή Μεταβλητών
 1. Τα σύνολα των βασικών και μη βασικών μεταβλητών ανταλλάσσουν μεταβλητές. Η εισερχόμενη γίνεται βασική και η εξερχόμενη γίνεται μη βασική. Εφαρμόζεται απαλοιφή Gauss και όλοι οι πίνακες ανανεώνονται, όπως στην αρχή του προβλήματος.
 2. Πηγαίνουμε στο βήμα επιλογής νέας βασικής μεταβλητής και επαναλαμβάνουμε τα 3 τελευταία βήματα μέχρι κάποιο κριτήριο να τερματίσει τον αλγόριθμο
 3. Η Διαδικασία της απαλοιφής ονομάζεται οδήγηση και το στοιχείο $(B^{-1}N)_{k\alpha}$ που χρησιμοποιείται για τις αλλαγές οδηγό στοιχείο