Γραμμική και Συνδυαστική Βελτιστοποίηση

Εργασία 2

Ονοματεπώνυμο: Γεώργιος Παπουτσάς

ΑΜ: 1083738

Έτος: 4ο

GitHub repository:
https://github.com/Papiqulos/Ergasia_2_GSB

Περιεχόμενα

Άσκηση 1:

Ερώτημα (α):

Αξιοποιώντας την pulp βιβλιοθήκη στην python γράφουμε το γραμμικό πρόβλημα προγραμματισμού:

```python
# Initialize a maximization problem
prob = pulp.LpProblem("ask1", pulp.LpMaximize)

# Basic Variables
x1 = pulp.LpVariable("x1", lowBound=0, cat=pulp.const.LpContinuous)
x2 = pulp.LpVariable("x2", lowBound=0, cat=pulp.const.LpContinuous)
x3 = pulp.LpVariable("x3", lowBound=0, cat=pulp.const.LpContinuous)
x4 = pulp.LpVariable("x4", lowBound=0, cat=pulp.const.LpContinuous)

# Objective function
prob += 5*x1 + 3*x2 + x3 + 4*x4, "obj"

# Constraints
prob += x1 - 2*x2 + 2*x3 + 3*x4 <= 10, "c1"
prob += 2*x1 + 2*x2 + 2*x3 - x4 <= 6, "c2"
prob += 3*x1 + x2 - x3 + x4 <= 10, "c3"
prob += -x2 + 2*x3 + 2*x4 <= 7, "c4"
```

Λύνοντάς το με την pulp παίρνουμε την βέλτιστη λύση καθώς και τα σκιώδη κόστη των περιορισμών και τις τιμές των μεταβλητών χαλάρωσης στη βέλτιστη λύση:

```
Status: Optimal
objective = 37.8666667
x1 = 0.0
x2 = 5.1333333
x3 = 0.6
x4 = 5.4666667
```

```
Sensitivity Analysis
Constraint                              Shadow Price          Slack
c1 : x1 - 2*x2 + 2*x3 + 3*x4 <= 10          -0.0              2.6666667000000004
c2 : 2*x1 + 2*x2 + 2*x3 - x4 <= 6           0.73333333              -0.0
c3 : 3*x1 + x2 - x3 + x4 <= 10          2.6             -0.0
c4 : -x2 + 2*x3 + 2*x4 <= 7             1.0666667               -0.0
PS D:\ACode\~Python\Linear Programming\Ergasia_2>
```

Από την παραπάνω λύση συμπεραίνουμε ότι βασικές μεταβλητές είναι οι x_2 $έως$ x_5 και οι x_1, x_6, x_7, x_8 οι μη βασικές αφού παίρνουν μηδενικές τιμές στην βέλτιστη λύση.

Ο πίνακας των περιορισμών μαζί με τις μεταβλητές χαλάρωσης:

$$A = \begin{bmatrix} 1 & -2 & 2 & 3 & 1 & 0 & 0 & 0 \\ 2 & 2 & 2 & -1 & 0 & 1 & 0 & 0 \\ 3 & 1 & -1 & 1 & 0 & 0 & 1 & 0 \\ 0 & -1 & 2 & 2 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Άρα έχοντας ως βασικές μεταβλητές τις x_2 $έως$ x_5, ο βέλτιστος βασικός πίνακας είναι:

$$B = \begin{bmatrix} -2 & 2 & 3 & 1 \\ 2 & 2 & -1 & 0 \\ 1 & -1 & 1 & 0 \\ -1 & 2 & 2 & 0 \end{bmatrix}$$

Αντικαθιστώντας την βέλτιστη λύση σε κάθε περιορισμό έχουμε:

- C1: $0 - 2 \cdot 5.13 + 2 \cdot 0.6 + 3 \cdot 5.47 = 7.35$
- C2: $0 + 2 \cdot 5.13 + 2 \cdot 0.6 - 1 \cdot 5.47 = 6$
- C3: $0 + 5.13 - 0.6 + 5.47 = 10$
- C4: $0 - 5.13 + 2 \cdot 0.6 + 2 \cdot 5.47 = 7$

Επομένως δεσμευτικοί είναι οι C2, C3, C4 αφού ισχύει η ισότητα του περιορισμού

Γεωμετρική ερμηνεία:

Η βέλτιστη λύση $(x_1, x_2, x_3, x_4) = (0, 5.133, 0.6, 5.467)$ βρίσκεται στην τομή των επιπέδων που ορίζονται από τους δεσμευτικούς περιορισμούς. Αυτό το σημείο τομής αντιπροσωπεύει μία κορυφή της εφικτής περιοχής που η αντικειμενική συνάρτηση λαμβάνει την μέγιστη τιμή της

Ερώτημα (β):

Έχω:

$$B = \begin{bmatrix} -2 & 2 & 3 & 1 \\ 2 & 2 & -1 & 0 \\ 1 & -1 & 1 & 0 \\ -1 & 2 & 2 & 0 \end{bmatrix}$$

$$N = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 3 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$B^{-1}N = \begin{bmatrix} 1.733 & 0.267 & 0.4 & -0.067 \\ -0.2 & 0.2 & -0.2 & 0.2 \\ 1.067 & -0.067 & 0.4 & 0.267 \\ 1.667 & 0.333 & 0 & -1.333 \end{bmatrix}$$

$$c_B{}^T = (3, 1, 4, 0)$$

$$c_N{}^T = (5, 0, 0,0)$$

Εξετάζουμε διαταραχή δ_2 στην βασική μεταβλητή x_2:

Η βέλτιστη λύση θα παραμείνει στην ίδια κορυφή αν ισχύει:

$$c_N{}^T - (3 + \delta_2, 1, 4, 0)B^{-1}N \leq 0$$

$$(5, 0, 0,0) - (3 + \delta_2, 1, 4, 0)\begin{bmatrix} 1.733 & 0.267 & 0.4 & -0.067 \\ -0.2 & 0.2 & -0.2 & 0.2 \\ 1.067 & -0.067 & 0.4 & 0.267 \\ 1.667 & 0.333 & 0 & -1.333 \end{bmatrix} \leq 0$$

$$(5, 0, 0, 0) - (1.733\delta_2 + 9.267, 0.267\delta_2 + 0.733, 0.4\delta_2 + 2.6, -0.067\delta_2 + 1.0677) \leq 0$$

$$(5 - 1.733\delta_2 - 9.267, -0.267\delta_2 - 0.733, -\ 0.4\delta_2 - 2.6, 0.067\delta_2 - 1.0677\ ) \leq 0$$

$$\Rightarrow -2.462 \leq \delta_2 \leq 15.9$$

Αν ο συντελεστής της x_2 είναι στο διάστημα $[0.538, 18.9]$ τότε η βέλτιστη κορυφή δεν αλλάζει ενώ η μεταβολή που θα παρατηρηθεί στην αντικειμενική συνάρτηση είναι:

$$\Delta z = \delta_2 e_1^T B^{-1} b \Rightarrow$$

$$\Rightarrow \Delta z = 5.135\delta_2$$

Εξετάζουμε διαταραχή δ_1 στη μη βασική μεταβλητή x_1:

Η βέλτιστη λύση θα παραμείνει στην ίδια κορυφή αν ισχύει:

$$[5+\delta_1, 0, 0, 0] - [3, 1, 4, 0]B^{-1}N = [5+\delta_1, 0, 0, 0] - [9.267, 0.733, 2.6, 1.067]$$
$$= [\delta_1 - 4.267, -0.733, -2.6, -1.067] \leq 0$$
$$\delta_1 - 4.267 \leq 0 \Rightarrow$$
$$\delta_1 \leq 4.267$$

Αν ο συντελεστής της x_1 είναι στο διάστημα $[-\infty, 9.267]$ τότε η βέλτιστη κορυφή δεν αλλάζει και επίσης δεν αλλάζει η τιμή της αντικειμενικής συνάρτησης

Ερώτημα (γ):

Έστω ότι εισάγουμε διαταραχή στον C1 περιορισμό (μη δεσμευτικός):

$$x_1 - 2x_2 + 2x_3 + 3x_4 \leq 10 + \gamma$$

Το διάστημα ανοχής βρίσκεται από την σχέση:

$$B^{-1}b + \gamma B^{-1}e_1 \geq 0 \Rightarrow$$

$$(5.1335, 0.6, 5.467, 2.267)^T + \gamma(0, 0, 0, 1)^T \geq 0 \Rightarrow$$

$$2.267 + \gamma \geq 0 \Rightarrow$$

$$\gamma \geq -2.267$$

Στη βέλτιστη λύση ο περιορισμός αυτός δεν είναι δεσμευτικός συνεπώς η αντικειμενική συνάρτηση δεν αλλάζει τιμή για κάποιο διάστημα τιμών της γ

Αντίστοιχα αν εισάγουμε διαταραχή στον C2 περιορισμό (δεσμευτικός):

$$2x_1 + 2x_2 + 2x_3 - x_4 \leq 6 + \gamma$$

Το διάστημα ανοχής βρίσκεται από την σχέση:

$$B^{-1}b + \gamma B^{-1}e_2 \geq 0 \Rightarrow$$

$$(5.1335, 0.6, 5.467, 2.267)^T + \gamma(0.267, 0.2, -0.067, 0.333)^T \geq 0 \Rightarrow$$

$$(5.135\ + 0.267\gamma, 0.6 + 0.2\gamma, 5.467 - 0.067\gamma, 2.267 + 0.333\gamma) \geq 0 \Rightarrow$$

$$-3 \leq \gamma \leq 81.597$$

Σε κάποια περιοχή του γ η αντικειμενική συνάρτηση μεταβάλλεται σταθερά κατά 0.733γ αφού η σκιώδης τιμή είναι 0.733

Ερώτημα (δ):

Γνωρίζουμε ότι σε ένα εφικτό τυπικό π.γ.π. το οποίο έχει βέλτιστη λύση, το σκιώδες κόστος ενός περιορισμού προσήμου υπάρχει και ισούται με το αντίθετο της βέλτιστης τιμής της αντίστοιχης δυϊκής μεταβλητής χαλάρωσης, Ακόμη, αν αυτός ο περιορισμός αναφέρεται σε μη βασική μεταβλητή και το σκιώδες κόστος είναι διάφορο του μηδενός τότε αυτό αντιστοιχεί στην μέγιστη μεταβολή που μπορεί να έχει ο αντίστοιχος συντελεστής στην αντικειμενική συνάρτηση πριν η μη βασική μεταβλητή γίνει βασική.

Άρα θα εξετάσουμε την μη βασική μεταβλητή x_1 και θα αναζητήσουμε την μεταβολή που πρέπει να υποστεί ο συντελεστής της στην αντικειμενική συνάρτηση ώστε να γίνει βασική με τον παραπάνω τρόπο.

Διατυπώνουμε πρώτα το δυϊκό πρόβλημα:

$$\min z\ \ 10y_1 + 6y_2 + 10y_3 + 7y_4$$

Όταν

$$y_1 + 2y_2 + 3y_3 \geq 5$$

$$-2y_1 + 2y_2 + y_3 - 4y_4 \geq 3$$

$$2y_1 + 2y_2 - y_3 + 2y_4 \geq 1$$

$$3y_1 - y_2 + y_3 + 2y_4 \geq 4$$

$$y_1, y_2, y_3, y_4 \geq 0$$

Λύνοντας το κατά τα γνωστά με την pulp:

```python
def solver_dual():
    """Solve the dual problem using the PuLP library. The problem is defined as:
    minimize 10*y1 + 6*y2 + 10*y3 + 7*y4
    subject to:
        y1 + 2*y2 + 3*y3 >= 5
        -2*y1 + 2*y2 + y3 - y4 >= 3
        2*y1 + 2*y2 - y3 + 2*y4 >= 1
        3*y1 - y2 + y3 + 2*y4 >= 4
        y1, y2, y3, y4 >= 0

    The function also prints the status of the solved LP, the value of the objective, the value of the variables at the optimum,
    and the shadow price and slack of the constraints.
    """

    # Initialize a minimization problem
    prob = pulp.LpProblem("ask1_dual", pulp.LpMinimize)

    # Basic Variables
    y1 = pulp.LpVariable("y1", lowBound=0, cat=pulp.const.LpContinuous)
    y2 = pulp.LpVariable("y2", lowBound=0, cat=pulp.const.LpContinuous)
    y3 = pulp.LpVariable("y3", lowBound=0, cat=pulp.const.LpContinuous)
    y4 = pulp.LpVariable("y4", lowBound=0, cat=pulp.const.LpContinuous)

    # Objective function
    prob += 10*y1 + 6*y2 + 10*y3 + 7*y4, "obj"

    # Constraints
    prob += y1 + 2*y2 + 3*y3 >= 5, "c1"
    prob += -2*y1 + 2*y2 + y3 - y4 >= 3, "c2"
    prob += 2*y1 + 2*y2 - y3 + 2*y4 >= 1, "c3"
    prob += 3*y1 - y2 + y3 + 2*y4 >= 4, "c4"
```

```
Status: Optimal
objective = 37.866666880000004
y1 = 0.0
y2 = 0.73333333
y3 = 2.6
y4 = 1.0666667

Sensitivity Analysis
Constraint                              Shadow Price            Slack
c1 : y1 + 2*y2 + 3*y3 >= 5              0.0              -4.2666667
c2 : -2*y1 + 2*y2 + y3 - y4 >= 3                5.1333333                -0.0
c3 : 2*y1 + 2*y2 - y3 + 2*y4 >= 1               0.6             -0.0
c4 : 3*y1 - y2 + y3 + 2*y4 >= 4         5.4666667               -0.0
```

Βλέπουμε ότι η βέλτιστη τιμή της μεταβλητής y_5 είναι -4.267. Άρα μεταβολή που πρέπει να υποστεί ο συντελεστής της x_1 στην αντικειμενική συνάρτηση για να μετατραπεί σε βασική είναι 4.267

Άσκηση 2:

Ερώτημα (α):

Το πρωτεύον πρόβλημα:

$$\max 3x_1 - 2x_2 - 5x_3 + 7x_4 + 8x_5$$

Όταν

$$x_2 - x_3 + 3x_4 - 4x_5 = -6$$

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

$$x_1 + 2x_3 - 2x_4 \leq -5$$

$$-x_1 \leq 2$$

$$x_1 \leq 10$$

$$-x_2 \leq -5$$

$$x_2 \leq 25$$

$$x_2, x_3, x_4 \geq 0, x_5 \in \mathbb{R}$$

Το δυϊκό του:

$$min - 6y_1 - 2y_2 - 5y_3 + 2y_4 + 10y_5 - 5y_6 + 25y_7$$

Όταν

$$-2y_2 + y_3 - y_4 + y_5 = 3$$

$$y_1 - 3y_2 - y_6 + y_7 \geq -2$$

$$-y_1 + 3y_2 + 2y_3 \geq -5$$

$$3y_1 + y_2 - 2y_3 \geq 7$$

$$-4y_1 = 8$$

$$y_1 \in \mathbb{R}$$

$$y_2, y_3, y_4\ y_5, y_6, y_7 \geq 0$$

Ερώτημα (β):

Έχω τον πίνακα των συντελεστών των περιορισμών του πρωτεύον προβλήματος:

$$A = \begin{bmatrix} 0 & 1 & -1 & 3 & -4 \\ -2 & -3 & 3 & 1 & 0 \\ 1 & 0 & 2 & -2 & 0 \\ -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Αφού θεωρούμε λύση που έχει ως βασικές μεταβλητές τις $x_2, x_4, x_5, x_8, x_9, x_{10}, x_{11}$ τότε επιλέγουμε τις στήλες

$$I = \{2, 4, 5\}, J = \{3, 4, 5, ,6\}$$

από τον A και τον I_7 αντίστοιχα

Και για το δυϊκό

$$\bar{I} = \{1, 3\}, \bar{J} = \{1, 2, 7\}$$

από τον I_5 και τον A^T αντίστοιχα

Τότε ο βασικός πίνακας:

$$B = \left[A^I I_7^J\right] = \begin{bmatrix} 1 & 3 & -4 & 0 & 0 & 0 & 0 \\ -3 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -2 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

και ο συμπληρωματικός

$$B^C = \left[A^{T^{\bar{J}}}(-I_5)^{\bar{I}}\right] = \begin{bmatrix} 0 & -2 & 0 & -1 & 0 \\ 1 & -3 & 1 & 0 & 0 \\ -1 & 3 & 0 & 0 & -1 \\ 3 & 1 & 0 & 0 & 0 \\ -4 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Παίρνοντας τις ορίζουσες του βασικού πίνακα B και του αντίστοιχου του δυικού προβλήματος B^C παρατηρούμε ότι δεν είναι ίσες αρά δεν ισχύει η συμπληρωματικότητα

Άρα η βασική λύση για το πρωτεύον πρόβλημα είναι:

$$b = \begin{bmatrix} -6 \\ -2 \\ -5 \\ 2 \\ 10 \\ -5 \\ 25 \end{bmatrix}$$

$$x^0 = B^{-1}b = \begin{bmatrix} 25 \\ 73 \\ 62.5 \\ 141 \\ 2 \\ 10 \\ 20 \end{bmatrix}$$

η οποία προκύπτει εφικτή

και για το δυϊκό:

$$c = \begin{bmatrix} 3 \\ -2 \\ -5 \\ 7 \\ 8 \end{bmatrix}$$

$$y^0 = B^{C^{-1}}c = \begin{bmatrix} -2 \\ 13 \\ 39 \\ -29 \\ 46 \end{bmatrix}$$

η οποία προκύπτει μη εφικτή

Εφόσον η βασική λύση του δυϊκού είναι μη εφικτή δεν ισχύει ούτε το θεώρημα ασθενής δυϊκότητας ούτε το θεώρημα ισχυρής δυϊκότητας.

Παρατίθεται και ο σχετικός κώδικας python που χρησιμοποιήθηκε για τις διάφορες πράξεις:

```
def helper():

    # Primal problem
    A = np.array([  [0, 1, -1, 3, -4],
                    [-2, -3, 3, 1, 0],
                    [1, 0, 2, -2, 0],
                    [-1, 0, 0, 0, 0],
                    [1, 0, 0, 0, 0],
                    [0, -1, 0, 0, 0],
                    [0, 1, 0, 0, 0]])

    # Right-hand side of the constraints
    b = np.array([-6, -2, -5, 2, 10, -5, 25])
    # Coefficients of the corresponding variables in the objective function
    c = np.array([-2, 7, 8, 0, 0, 0, 0])

    # I = 2,4,5 J = 3,4,5,6
    # x2, x4, x5, x8, x9, x10, x11 are the basic variables
    B = np.array([[1, 3, -4, 0, 0, 0, 0],
                  [-3, 1, 0, 0, 0, 0, 0],
                  [0, -2, 0, 1, 0, 0, 0],
                  [0, 0, 0, 0, 1, 0, 0],
                  [0, 0, 0, 0, 0, 1, 0],
                  [-1, 0, 0, 0, 0, 0, 1],
                  [1, 0, 0, 0, 0, 0, 0]])

    # Dual problem
    A_dual = A.T

    # Right-hand side of the constraints
    b_dual = np.array([3, -2, -5, 7, 8])
    # Coefficients of the corresponding variables in the objective function
    c_dual = np.array([-6, -2, 25, 0, 0])

    # I = 1,3 J = 1,2,7
    # y1, y2, y7, y8, y10 are the basic variables
    Bc = np.array([
                [0, -2, 0, -1, 0],
                [1, -3, 1, 0, 0],
                [-1, 3, 0, 0, -1],
                [3, 1, 0, 0, 0],
                [-4, 0, 0, 0, 0]])

    primal_solution = np.linalg.inv(B) @ b
    dual_solution = np.linalg.inv(Bc) @ b_dual

    objective_primal = c @ primal_solution
    objective_dual = c_dual @ dual_solution
```

```
Objective of Primal: 961.0
Objective of Dual: 961.0
Basic solution of Primal: [ 25.    73.    62.5 141.     2.    10.    20. ]
Basic solution of Dual: [ -2.   13.   39.  -29.   46.]
Determinant of Primal B: 4.00
Determinant of Dual B: -4.00
```

Ερώτημα (γ):

Χρησιμοποιώντας την βιβλιοθήκη pulp στην python παίρνουμε την λύση του primal προβλήματος.

```python
def solver_primal():
    """
    Solve the primal problem using the PuLP library.
    The problem is defined as:
    maximize 3*x1 - 2*x2 - 5*x3 + 7*x4 + 8*x5
    subject to:
        x2 - x3 + 3*x4 - 4*x5 = 6
        2*x1 - 3*x2 + 3*x3 + x4 <= -2
        x1 + 2*x3 - 2*x4 <= -5
        -x1 <= 2
         x1 <= 10
        -x2 <= -5
        x2 <= 25

    The function also prints the status of the solved LP, the value of the objective, the value of the variables at the optimum,
    and the shadow price and slack of the constraints.
    """

    # Initialize a maximization problem
    prob = pulp.LpProblem("Primal", pulp.LpMaximize)

    # Basic Variables
    x1 = pulp.LpVariable("x1", cat=pulp.const.LpContinuous)
    x2 = pulp.LpVariable("x2", lowBound=0, cat=pulp.const.LpContinuous)
    x3 = pulp.LpVariable("x3", lowBound=0, cat=pulp.const.LpContinuous)
    x4 = pulp.LpVariable("x4", lowBound=0, cat=pulp.const.LpContinuous)
    x5 = pulp.LpVariable("x5", cat=pulp.const.LpContinuous)

    # Objective function
    prob += 3*x1 - 2*x2 - 5*x3 + 7*x4 + 8*x5, "obj"

    # Constraints
    prob +=  x2 - x3 + 3*x4 - 4*x5 == -6, "c1"
    prob += -2*x1 - 3*x2 + 3*x3 + x4 <= -2, "c2"
    prob += x1 + 2*x3 - 2*x4 <= -5, "c3"
    prob += -x1 <= 2, "c4"
    prob += x1 <= 10, "c5"
    prob += -x2 <= -5, "c6"
    prob += x2 <= 25, "c7"
```

```
Status: Optimal
objective = 1251.0
x1 = 10.0
x2 = 25.0
x3 = 0.0
x4 = 93.0
x5 = 77.5

Sensitivity Analysis
Constraint                                Shadow Price           Slack
c1 : x2 - x3 + 3*x4 - 4*x5 = -6           -2.0           -0.0
c2 : -2*x1 - 3*x2 + 3*x3 + x4 <= -2              13.0            -0.0
c3 : x1 + 2*x3 - 2*x4 <= -5               -0.0           171.0
c4 : -x1 <= 2            -0.0             12.0
c5 : x1 <= 10            29.0             -0.0
c6 : -x2 <= -5           -0.0             20.0
c7 : x2 <= 25            39.0             -0.0
```

Και για το δυϊκό:

```python
def solver_dual():
    """
    Solve the dual problem using the PuLP library.
    The problem is defined as:
    minimize -6*y1 - 2*y2 - 5*y3 + 2*y4 + 10*y5 - 5*y6 + 25*y7
    subject to:
        2*y2 + y3 + y4 + y5 = 3
        y1 + 3*y2 + y6 - y7 >= -2
        -y1 - 3*y2 + 2*y3 >= -5
        3*y1 - y2 - 2*y3 >= 7
        -4*y1 == 8

    The function also prints the status of the solved LP, the value of the objective, the value of the variables at the optimum,
    and the shadow price and slack of the constraints.
    """

    # Initialize a minimization problem
    prob = pulp.LpProblem("Dual", pulp.LpMinimize)

    # Basic Variables
    y1 = pulp.LpVariable("y1", cat=pulp.const.LpContinuous)
    y2 = pulp.LpVariable("y2", lowBound=0, cat=pulp.const.LpContinuous)
    y3 = pulp.LpVariable("y3", lowBound=0, cat=pulp.const.LpContinuous)
    y4 = pulp.LpVariable("y4", lowBound=0, cat=pulp.const.LpContinuous)
    y5 = pulp.LpVariable("y5", lowBound=0, cat=pulp.const.LpContinuous)
    y6 = pulp.LpVariable("y6", lowBound=0, cat=pulp.const.LpContinuous)
    y7 = pulp.LpVariable("y7", lowBound=0, cat=pulp.const.LpContinuous)

    # Objective function
    prob += -6*y1 - 2*y2 - 5*y3 + 2*y4 + 10*y5 - 5*y6 + 25*y7, "obj"

    # Constraints
    prob += -2*y2 + y3 - y4 + y5 == 3, "c1"
    prob += y1 - 3*y2 - y6 + y7 >= -2, "c2"
    prob += -y1 + 3*y2 + 2*y3 >= -5, "c3"
    prob += 3*y1 + y2 - 2*y3 >= 7, "c4"
    prob += -4*y1 == 8, "c5"
```

```
Status: Optimal
objective = 1251.0
y1 = -2.0
y2 = 13.0
y3 = 0.0
y4 = 0.0
y5 = 29.0
y6 = 0.0
y7 = 39.0

Sensitivity Analysis
Constraint                              Shadow Price          Slack
c1 : -2*y2 + y3 - y4 + y5 = 3           10.0            -0.0
c2 : y1 - 3*y2 - y6 + y7 >= -2          25.0            -0.0
c3 : -y1 + 3*y2 + 2*y3 >= -5            0.0             -46.0
c4 : 3*y1 + y2 - 2*y3 >= 7              93.0            -0.0
c5 : -4*y1 = 8          77.5            -0.0
```

Άσκηση 3:

Το πρόβλημα είχε μοντελοποιηθεί ως εξής:

$$min\ Z = 6x_1 + 10x_2 + 8x_3 + 8x_3 + 3x_5$$

Όταν

$$\sum_{i=1}^{5} x_i = 23 - 19 = 4\ (\Pi 1)$$

$$x_1 \leq 2\ (\Pi 2)$$

$$x_2 \leq 5\ (\Pi 3)$$

$$x_3 \leq 2\ (\Pi 4)$$

$$x_4 \leq 2\ (\Pi 5)$$

$$x_5 \leq 3\ (\Pi 6)$$

Χρησιμοποιώντας την βιβλιοθήκη pulp στην python παίρνουμε την λύση του προβλήματος.

```
Status: Optimal
objective = 3.0
x1 =  2.00
x2 = -5.00
x3 =  2.00
x4 =  2.00
x5 =  3.00

Sensitivity Analysis
Constraint                        Shadow Price          Slack
c1 : x1 + x2 + x3 + x4 + x5 = 4        10.00         -0.00
```

Παρατίθεται παρακάτω και ο σχετικός κώδικας:

```python
def solver():

    # Initialize a minimization problem
    prob = pulp.LpProblem("ask3", pulp.LpMinimize)

    # Basic Variables
    x1 = pulp.LpVariable("x1", upBound=2, cat=pulp.const.LpContinuous)
    x2 = pulp.LpVariable("x2", upBound=5, cat=pulp.const.LpContinuous)
    x3 = pulp.LpVariable("x3", upBound=2, cat=pulp.const.LpContinuous)
    x4 = pulp.LpVariable("x4", upBound=2, cat=pulp.const.LpContinuous)
    x5 = pulp.LpVariable("x5", upBound=3, cat=pulp.const.LpContinuous)


    # Objective function
    prob += 6*x1 + 10*x2 +8*x3 + 8*x4 + 3*x5, "obj"

    # Constraints
    prob += x1 + x2 + x3 + x4 + x5 == 4, "c1"


    # Solve the problem using the default solver
    prob.solve()

    # Print the status of the solved LP
    print("--------------------------------------")
    print("Status:", pulp.LpStatus[prob.status])

    # Print the value of the objective
    print("objective =", pulp.value(prob.objective))

    # Print the value of the variables at the optimum
    for v in prob.variables():
        print(f'{v.name} = {v.varValue:5.2f}')

    # Print the shadow price and slack of the constraints
    print("\nSensitivity Analysis\nConstraint\t\t\t\tShadow Price\t\tSlack")
    for name, c in prob.constraints.items():
        print(f'{name} : {c}\t\t{c.pi:.2f}\t\t{c.slack:.2f}')
```

Άσκηση 4:

Αρχικά διατυπώνουμε το πρωτεύον πρόβλημα και το δυϊκό του:

$$\max z \ \ x_1 + 2x_2 + x_3 - 3x_4 + x_5 + x_6 - x_7$$

Όταν

$$x_1 + x_2 - x_4 + 2x_6 - 2x_7 \leq 6$$

$$x_2 - x_4 + x_5 - 2x_6 + 2x_7 \leq 4$$

$$x_2 + x_3 + x_6 - x_7 \leq 2$$

$$x_2 - x_4 - x_6 + x_7 \leq 1$$

$$x_1, x_2, x_3, x_4, x_5, x_6, x_7 \geq 0$$

και το δυϊκό:

$$\min z \ \ 6y_1 + 4_2 + 2y_3 + y_4$$

Όταν

$$y_1 \geq 1$$

$$y_1 + y_2 + y_3 + y_4 \geq 2$$

$$y_3 \geq 1$$

$$-y_1 - y_2 - y_4 \geq -3$$

$$y_2 \geq 1$$

$$2y_1 - 2y_2 + y_3 - y_4 \geq 1$$

$$-2y_1 + 2y_2 - y_3 + y_4 \geq -1$$

$$y_1, y_2, y_3, y_4 \geq 0$$

Έχοντας την λύση του πρωτεύον η οποία είναι και εφικτή:

$$x = (\frac{15}{2}, 0, \frac{11}{4}, 0, \frac{5}{2}, 0, \frac{3}{4})$$

Παρατηρούμε ότι οι βασικές μεταβλητές είναι οι x_1, x_3, x_5, x_7

Άρα πηγαίνουμε στους αντίστοιχους περιορισμούς του δυϊκού προβλήματος και βάζουμε το ίσον για να βρούμε μία πιθανή λύση για το δυϊκό:

$$y_1 = 1$$

$$y_3 = 1$$

$$y_2 = 1$$

$$-2y_1 + 2y_2 - y_3 + y_4 = -1$$

Και παίρνουμε την πιθανή λύση η οποία είναι εφικτή:

$$y = (1, 1, 1, 0)$$

Τώρα θα βρούμε τις τιμές των αντικειμενικών συναρτήσεων για τις εκάστοτε λύσεις στο πρωτεύον και το δυϊκό και θα ελέγξουμε αν ισχύει το θεώρημα της ισχυρής δυϊκότητας αφού έχουμε βρει εφικτές λύσεις για τα πρωτεύον και το δυϊκό:

$$c^T x = 12$$

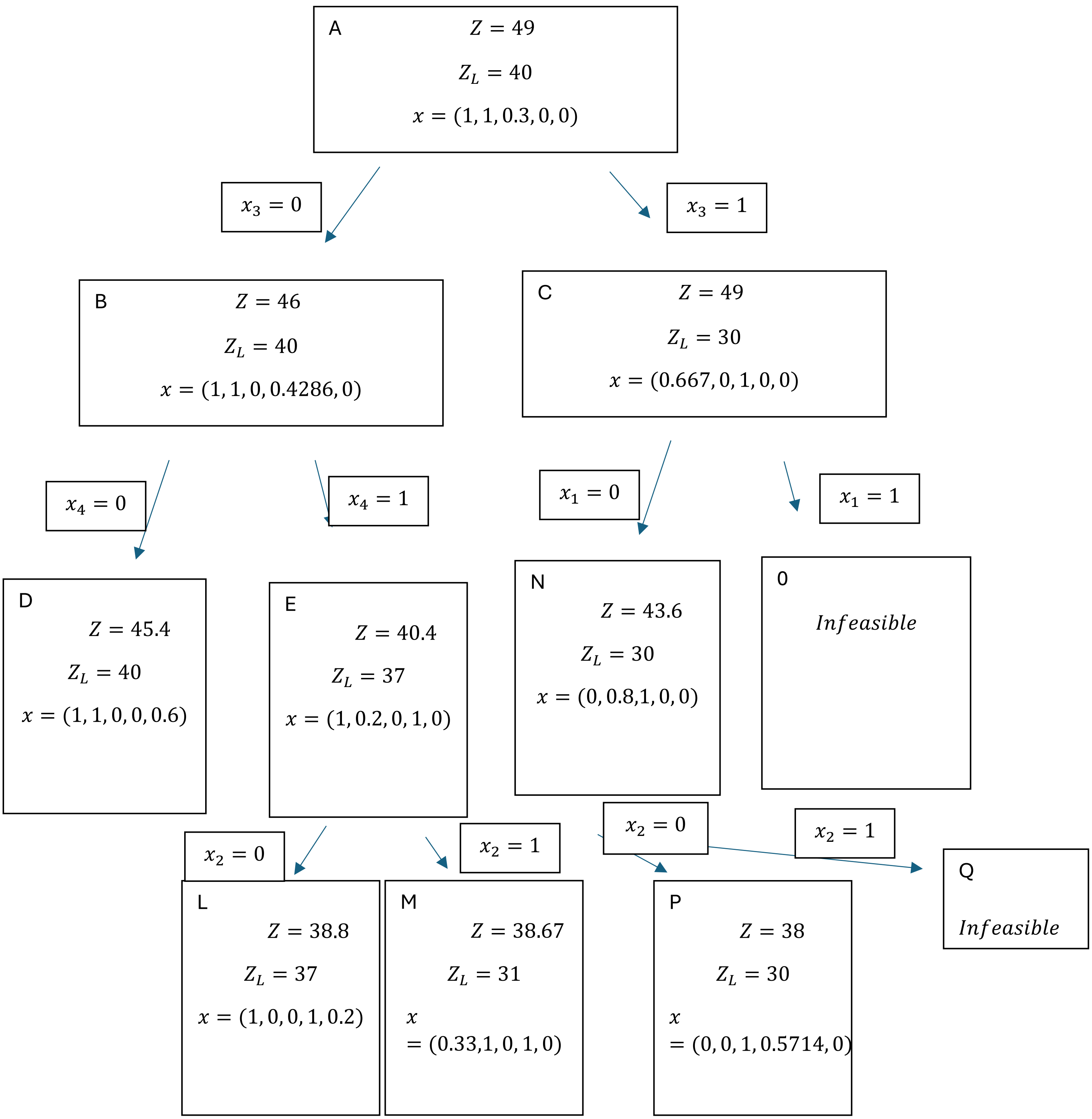
$$b^T y = 12$$

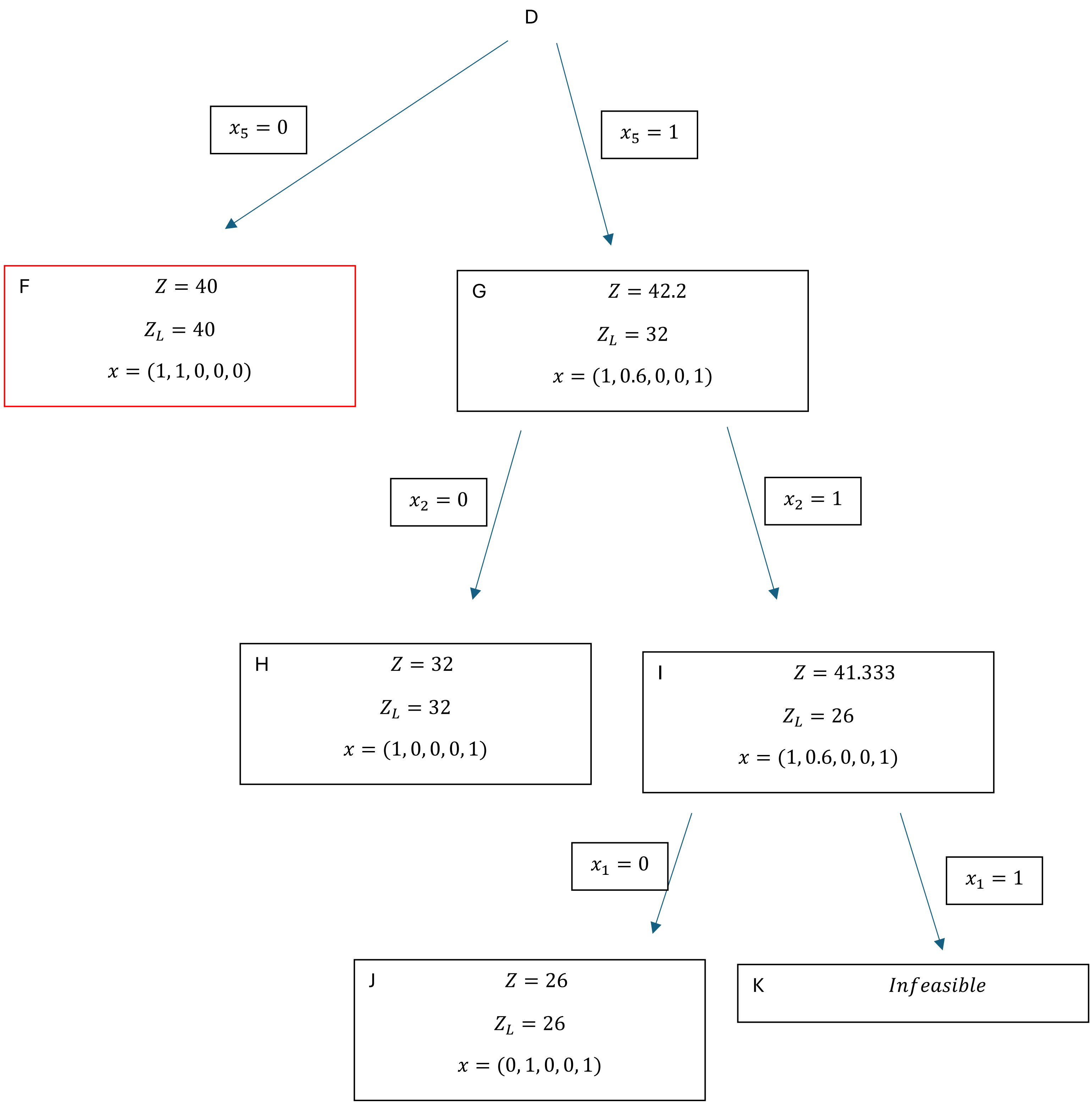
Εφόσον οι τιμές των αντικειμενικών συναρτήσεων είναι ίσες τότε η λύση της εκφώνησης είναι και η βέλτιστη.

Άσκηση 5:

Ερώτημα (α):

Ο αλγόριθμος έχει ως εξής.:




D
$x_5 = 0$
$x_5 = 1$
F
$Z = 40$
$Z_L = 40$
$x = (1, 1, 0, 0, 0)$
G
$Z = 42.2$
$Z_L = 32$
$x = (1, 0.6, 0, 0, 1)$
$x_2 = 0$
$x_2 = 1$
H
$Z = 32$
$Z_L = 32$
$x = (1, 0, 0, 0, 1)$
I
$Z = 41.333$
$Z_L = 26$
$x = (1, 0.6, 0, 0, 1)$
$x_1 = 0$
$x_1 = 1$
J
$Z = 26$
$Z_L = 26$
$x = (0, 1, 0, 0, 1)$
K
$Infeasible$

Καταλήγουμε στην λύση F ως την βέλτιστη

Ερώτημα (β):

Ακολουθώντας την αλγοριθμική διαδικασία για σύσφιξη περιορισμού για τον μόνο περιορισμο του προβλήματος παρατηρούμε ότι δεν μπορεί να υποστεί βελτίωση.

Αλγοριθμική διαδικασία:

Έστω ο περιορισμός $a_1x_1 + \ldots + a_nx_n \leq b$.

Βήμα 1 Υπολογίζουμε S = άθροισμα των a_i με $a_i > 0$.

Βήμα 2 Βρίσκουμε ένα $a_j \neq 0$ έτσι ώστε $S < b + |a_j|$

- 2.1 Αν δεν υπάρχει, Stop; ο περιορισμός δεν βελτιώνεται.
- 2.2 Αν $a_j > 0$ πήγαινε στο **Βήμα 3**.
- 2.3 Αν $a_j < 0$ πήγαινε στο **Βήμα 4**.

Βήμα 3 ($a_j > 0$) Θέτουμε $\overline{a_j} = S - b$ και $\overline{b} = S - a_j$.
Αντικαθιστούμε: $a_j \leftarrow \overline{a_j}$ και $b \leftarrow \overline{b}$. Πίσω στο **Βήμα 1**.

Βήμα 4 ($a_j < 0$) Αυξάνουμε το a_j σε $\overline{a_j} = b - S$. Πίσω στο **Βήμα 1**.

Παράδειγμα: Επιβεβαιώστε ότι: $2x_1 + 3x_2 \leq 4 \quad \longmapsto \quad x_1 + x_2 \leq 1$

Υλοποίηση σε python για τον περιορισμό του προβλήματος:

```python
# Coefficients of the constraint
a = [6, 5, 10, 7, 5]
# Right-hand side of the constraint
b = 14
S = sum(a)


# Constraint compression
aj_=  0
b_ = 0

for j in range(len(a)):
    if a[j] != 0 and S < b + abs(a[j]):
        if a[j] > 0:
            aj_ = S - b
            b_ = S - a[j]

            a[j] = aj_
            b = b_
        else:
            a[j] = b - S

print(a)
```

```
a_2/Code/askhsh_5.
[6, 5, 10, 7, 5]
```

Ερώτημα (γ):

Τρεις διαφορετικές ελάχιστες καλύψεις είναι:

- $C = \{x_1, x_3, x_4\}$
- $C = \{x_1, x_2, x_3\}$
- $C = \{x_1, x_2, x_4\}$

Και από αυτές τα επίπεδα αποκοπής είναι:

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

Παρατίθεται βοηθητικός κώδικας:

```python
def constraint(x1=0, x2=0, x3=0, x4=0, x5=0):
    C = []
    if x1 != 0:
        # print("x1", end=" ")
        C.append("x1")
    if x2 != 0:
        # print("x2", end=" ")
        C.append("x2")
    if x3 != 0:
        # print("x3", end=" ")
        C.append("x3")
    if x4 != 0:
        # print("x4", end=" ")
        C.append("x4")
    if x5 != 0:
        # print("x5", end=" ")
        C.append("x5")
    return 6*x1 + 5*x2 + 10*x3 + 7*x4 + 5*x5, f"{C} <= {len(C) - 1}"

def generate_cutting_planes():
    print("Cutting planes:")
    _, c1 = constraint(x1=1, x3=1, x4=1)
    _, c2 = constraint(x1=1, x2=1, x3=1)
    _, c3 = constraint(x1=1, x2=1, x4=1)

    print(c1)
    print(c2)
    print(c3)
```

Χρησιμοποιώντας το online solver AtoZmath.com παίρνουμε τα δένδρα για κάθε καινούριο περιορισμό:

- $x_1 + x_3 + x_4 \leq 2$ **Παραμένει ίδιο**
- $x_1 + x_2 + x_3 \leq 2$ **Παραμένει ίδιο**
- $x_1 + x_2 + x_4 \leq 2$ **Αλλάζει ως εξής:**

The branch and bound diagram