Γραμμική & Συνδυαστική Βελτιστοποίηση

Εργασία #2

Μαρία-Νίκη Ζωγράφου
ΑΜ: 1096060

Περιεχόμενα

Άσκηση 1.

Θεωρήστε το πρόβλημα γραμμικού προγραμματισμού:

$$\max\ 3x_1 + 11x_2 + 9x_3 - x_4 - 29x_5$$

όταν

$$\begin{aligned} x_2 + x_3 + x_4 - 2x_5 &\leq 4 \\ x_1 - x_2 + x_3 + 2x_4 + x_5 &\geq 0 \\ x_1 + x_2 + x_3 - 3x_5 &\leq 1 \\ x_1 \in \mathbb{R}, x_2, x_3, x_4, x_5 &\geq 0 \end{aligned}$$

(α) Λύστε το πρόβλημα (με κάποιον από τους επιλυτές που αναφέραμε στις διαλέξεις) και περιγράψτε τη βέλτιστη λύση, δηλ. τιμές των μεταβλητών απόφασης και τιμή αντικειμενικής συνάρτησης. Δώστε τις βασικές / μη-βασικές μεταβλητές και τον βέλτιστο βασικό πίνακα. Επίσης ξεχωρίστε τους δεσμευτικούς από τους μη δεσμευτικούς περιορισμούς και μέσω αυτών περιγράψτε τη βέλτιστη κορυφή.

Με χρήση pulp:

```
# Αντικειμενική συνάρτηση
model += 3*x1 + 11*x2 + 9*x3 - x4 - 29*x5, "Objective"

# Περιορισμοί
model += x2 + x3 + x4 - 2*x5 <= 4, "Constraint_1"
model += x1 - x2 + x3 + 2*x4 + x5 >= 0, "Constraint_2"
model += x1 + x2 + x3 - 3*x5 <= 1, "Constraint_3"
```

Βέλτιστη Λύση: **Objective value: 28.0** με x1 = -3.0, x2 = 0.5, x3 = 3.5, x4 = 0.0, x5 = 0.0

Βασικές Μεταβλητές: x_1=-3.0, x_2=0.5, x_3=3.5

Μη βασικές Μεταβλητές: x_4=0.0, x_5=0.0

Βασικός Πίνακας:

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

Δεσμευτικοί Περιορισμοί: όλοι όσοι ικανοποιούν τις ισότητες

Slack του περιορισμού 1 (<=): 0.0

Surplus του περιορισμού 2 (>=): 0.0

Slack του περιορισμού 3 (<=): 0.0

Περιγραφή Βέλτιστης Κορυφής:

Η βέλτιστη λύση βρίσκεται στη **κορυφή του πολυέδρου εφικτών λύσεων** που ορίζεται από **τη συνάντηση των 3 δεσμευτικών περιορισμών** δηλαδή είναι η τομή των υπερεπιπέδων των δεσμευτικών περιορισμών.

(β) Επιλέξτε μία βασική και μία μη-βασική μεταβλητή. Περιγράψτε τι θα συμβεί εάν ο συντελεστής της καθεμιάς στην αντικειμενική συνάρτηση (ξεχωριστά) διαταραχθεί κατά ένα ποσό γ. Βρείτε τα διαστήματα ανοχής για τους συγκεκριμένους συντελεστές ώστε να παραμείνει η βέλτιστη λύση στην ίδια κορυφή.

Έστω ότι επιλέγω την x_2 και την x_4.

Διαστήματα ανοχής: όσο παραμένουν στα διαστήματα ανοχής η βέλτιστη κορυφή παραμένει η ίδια.

Μεταβολή συντελεστή x_4: Max(3x1 +11x2 +9x3 – 1 x4 –29x5).

C_B^T=[3 11 9], C_N^T=[-1 -29]

Η λύση στην οποία βρισκόμαστε θα παραμείνει βέλτιστη μόνον εφ'όσον ισχύει:

$$(\mathbf{c}_N^T + \delta \mathbf{e}_k^T - \mathbf{c}_B^T \mathbf{B}^{-1}\mathbf{N})_\alpha \leq \mathbf{0} \quad \text{όπου } \alpha \in \{1, \dots, n\}$$

N= [[1 -2], [2 1], [0 -3]] και $\mathbf{B}^{-1}$=[[-1. 0. 1.], [0. -0.5 0.5],[1. 0.5 -0.5]]

k=0: e_k^T=[1 0] άρα: [-1 -29] + [δ 0] - [3 11 9] $B^{-1}N$ = [δ-1 -29]- [4. -25.]

Άρα δ-1-4<=0 οπότε δ<=5, οπότε: **δ∈(-∞,5).**

Επομένως ο συντελεστής της x_4 θα πρέπει να είναι στο διάστημα **x_4∈ (-∞,4).**

Μεταβολή συντελεστή x_2:

Η λύση στην οποία βρισκόμαστε θα παραμείνει βέλτιστη μόνον εφόσον συνεχίζει να ισχύει:

$$(\mathbf{c}_N^T - (\mathbf{c}_B + \delta \mathbf{e}_k)^T \mathbf{B}^{-1}\mathbf{N})_\alpha \leq \mathbf{0} \quad \text{όπου } \alpha \in \{1, \dots, n\}$$

C_N^T=[-1 -29], C_B^T=[3 11 9]

e_k^T=[0 1 0]

[-1 -29] – [3 11+δ 9] $B^{-1}N$ με $B^{-1}N$ =[[-1. -1.], [-1. -2.], [2. 0.]]

Επομένως [-1 -29] – [-3-11+18-δ -3-22-2δ] <= 0

Άρα πρέπει δ<=5 και δ<=2. Επομένως πρέπει **δ<=2**.

δ∈(-∞,2) και **x_2∈ (-∞,13).**

Ερμηνεία:

Αν ο συντελεστής της μεταβλητής **x_4** (μη-βασική) στην αντικειμενική συνάρτηση διαταραχθεί κατά ένα ποσό γ μέσα στα όρια **(-∞,5),** η λύση παραμένει στην ίδια κορυφή και η αντικειμενική συνάρτηση παραμένει στην ίδια τιμή, δηλαδή η τελική λύση είναι η βέλτιστη.

Αν ο συντελεστής της μεταβλητής **x_2** (βασική) διαταραχθεί κατά γ μέσα στα όρια **(-∞,2)** η βέλτιστη λύση παραμένει στην ίδια κορυφή και η τιμή της αντικειμενικής συνάρτησης μεταβάλλεται κατά:

$$\Delta z = \delta_1 \mathbf{e}_2^T \mathbf{B}^{-1}\mathbf{b} = (9/2)\delta_1$$

(γ) Επιλέξτε έναν δεσμευτικό και έναν μη δεσμευτικό περιορισμό και περιγράψτε τι θα συμβεί εάν το δεξιό μέρος του καθενός από αυτούς διαταραχθεί κατά ένα ποσό γ. Βρείτε τα διαστήματα ανοχής που αντιστοιχούν στους δυο αυτούς περιορισμούς.

Όλοι οι περιορισμοί είναι δεσμευτικοί (εκτός από τους περιορισμούς προσήμου).

Γνωρίζουμε ότι:

Το διάστημα ανοχής για τον περιορισμό i ορίζεται από την ανισότητα:

$$\mathbf{B}^{-1}\mathbf{b} + \gamma \mathbf{B}^{-1}\mathbf{e}_i \geq 0$$

Άρα για τον **δεσμευτικό περιορισμό 1**: x2 +x3 +x4 −2x5 ≤ 4 έχουμε: ei= e1 = $[1,0,0]^T$ και b= [4 0 1]

Έχουμε ήδη το $x_B = B^{-1}b$= [-3. 0.5 3.5]

$$\begin{bmatrix} -3 \\ 0.5 \\ 3.5 \end{bmatrix} + \gamma \begin{bmatrix} -1 \\ 0 \\ 1 \end{bmatrix} \geq 0 \rightarrow \begin{Bmatrix} \gamma \leq -3 \\ 0.5 \geq 0 \\ \gamma \geq -3.5 \end{Bmatrix}$$

Καταλήγουμε ότι $\boxed{\gamma \in [-3.5,3].}$

Το δεξί μέλος του 1ου περιορισμού **μπορεί να μεταβληθεί μόνο κατά γ∈[−3.5,−3],** ώστε να παραμείνει **η ίδια κορυφή** (ίδια βάση). Η Z(γ) θα μεταβάλλεται με σταθερό ρυθμό που ισούται με τη βέλτιστη τιμή της αντίστοιχης δυικής (σκιώδης τιμής).

Για τον **μη δεσμευτικό περιορισμό x2>=0**:

Διάστημα ανοχής: $(\mathbf{B}^{-1}\mathbf{b})_j \geq \gamma \mathbf{e}_j$

Οπότε πρέπει: $\boxed{0.5 \geq \gamma.}$

(δ) Καταστρώστε και λύστε (με κάποιον από τους επιλυτές που αναφέραμε στις δια λέξεις) το δυϊκό του παραπάνω προβλήματος. Ελέγξτε κατά πόσο ισχύουν οι συνθήκες συμπληρωματικής χαλαρότητας για τα δύο αυτά προβλήματα.

Πίνακας Α πρωτεύοντος:

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

Δυικό: $\mathrm{Z} = \min(4y_1 + 0y_2 + 1y_3)$

$$\begin{Bmatrix} y_2 + y_3 = 3 & (x_1 \in \mathrm{R}) \\ y_1 - y_2 + y_3 \geq 11 & (x_2 \geq 0) \\ y_1 + y_2 + y_3 \geq 9 & (x_3 \geq 0) \\ y_1 + 2y_2 \geq -1 & (x_4 \geq 0) \\ -2y_1 + y_2 - 3y_3 \geq -29 & (x_5 \geq 0) \end{Bmatrix} \;\kappa\alpha\iota\; y_1, y_3 \geq 0 \;\kappa\alpha\iota\; y_2 \leq 0$$

Επίλυση με Pulp

Status: Optimal
y1 = 6.0
y2 = -1.0
y3 = 4.0

Objective = 28.0

Έλεγχος Συμπληρωματικής Χαλαρότητας:

Οι y1,y2,y3 είναι μη μηδενικές επομένως οι τρεις περιορισμοί του πρωτεύοντος πρέπει να είναι δεσμευτικοί. Επιπλέον πρέπει y1,y3>0 και y2<0. Όλα τα παραπάνω ικανοποιούνται και οι τιμές των αντικειμενικών συναρτήσεων είναι ίδιες.

Κώδικας

```
import pulp
import numpy as np


def exercise1_model():
    # Δημιουργία μοντέλου
    model = pulp.LpProblem("LP1", pulp.LpMaximize)

    # Δημιουργία μεταβλητών
    x1 = pulp.LpVariable('x1', cat='Continuous')  # x1 ∈ R
    x2 = pulp.LpVariable('x2', lowBound=0)
    x3 = pulp.LpVariable('x3', lowBound=0)
    x4 = pulp.LpVariable('x4', lowBound=0)
    x5 = pulp.LpVariable('x5', lowBound=0)

    # Αντικειμενική συνάρτηση
    model += 3*x1 + 11*x2 + 9*x3 - x4 - 29*x5, "Objective"

    # Περιορισμοί
    model += x2 + x3 + x4 - 2*x5 <= 4, "Constraint_1"
    model += x1 - x2 + x3 + 2*x4 + x5 >= 0, "Constraint_2"
    model += x1 + x2 + x3 - 3*x5 <= 1, "Constraint_3"

    # Επίλυση
    model.solve(pulp.PULP_CBC_CMD(msg=False))
    return model

def extract_basic_matrix(A, b, variable_values, var_to_index):
    basic_vars = [v for v, val in variable_values.items() if abs(val) > 1e-8]
    nonbasic_vars = [v for v in variable_values if v not in basic_vars]

    B = A[:, [var_to_index[v] for v in basic_vars]]
    N = A[:, [var_to_index[v] for v in nonbasic_vars]]

    xB = np.linalg.solve(B, b)

    print("Βασικές μεταβλητές:", basic_vars)
    print("Μη βασικές μεταβλητές:", nonbasic_vars)
```

```
    print("Πίνακας B:")
    print(B)
    print("B_inv")
    B_inv = np.linalg.inv(B)
    print(B_inv)

    print("b=", b)

    print("xB =", xB)

    #e=np.array([1.,0.,0.])
    #print("DF",B_inv @ e)
    return B, xB, N, basic_vars, nonbasic_vars


def answer_a(model):
    print("Status:", pulp.LpStatus[model.status])
    print("Objective value:", pulp.value(model.objective))
    for var in model.variables():
        print(f"{var.name} = {var.varValue}")

    variable_values = {var.name: var.varValue for var in model.variables()}

    print("Μεταβλητή χαλάρωσης (slack) του περιορισμού 1 (<=):",
model.constraints["Constraint_1"].slack)
    print("Μεταβλητή πλεονάσματος (surplus) του περιορισμού 2 (>=):",
model.constraints["Constraint_2"].slack)
    print("Μεταβλητή χαλάρωσης (slack) του περιορισμού 3 (<=):",
model.constraints["Constraint_3"].slack)

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

    var_to_index = {var: i for i, var in enumerate(variable_values.keys())}
    c = {'x1': 3,    'x2': 11,     'x3': 9,    'x4': -1,    'x5': -29    }

    B, xB, N, basic_vars, nonbasic_vars = extract_basic_matrix(A, b,
variable_values, var_to_index)

def answer_d():
    # Create LP
    dual = pulp.LpProblem("Dual", pulp.LpMinimize)

    # Variables
    y1 = pulp.LpVariable('y1', lowBound=0)
```

```python
    y2 = pulp.LpVariable('y2', upBound=0)
    y3 = pulp.LpVariable('y3', lowBound=0)

    # Objective
    dual += 4*y1 + 0*y2 + 1*y3

    # Constraints
    dual += y2 + y3 == 3
    dual += y1 - y2 + y3 >= 11
    dual += y1 + y2 + y3 >= 9
    dual += y1 + 2*y2 >= -1
    dual += -2*y1 + y2 - 3*y3 >= -29

    # Solve
    dual.solve(pulp.PULP_CBC_CMD(msg=False))
    print("Status:", pulp.LpStatus[dual.status])
    for var in dual.variables():
        print(f"{var.name} = {var.varValue}")
    print("Objective =", pulp.value(dual.objective))

if __name__ == "__main__":
    model = exercise1_model()
    answer_a(model)
    # Μετά τη λύση του μοντέλου:
    variable_values = {var.name: var.varValue for var in model.variables()}
    var_to_index = {var: i for i, var in enumerate(variable_values.keys())}
    basic_vars = [v for v, val in variable_values.items() if abs(val) > 1e-8]
    nonbasic_vars = [v for v in variable_values if v not in basic_vars]

    # Πίνακας A και b
    A = np.array([
        [0, 1, 1, 1, -2],
        [1, -1, 1, 2, 1],
        [1, 1, 1, 0, -3]
    ])
    b = np.array([4, 0, 1])

    # Δημιουργία B και N
    B = A[:, [var_to_index[v] for v in basic_vars]]
    N = A[:, [var_to_index[v] for v in nonbasic_vars]]
    B_inv = np.linalg.inv(B)
    print("-----------------------------------------")
    print("                  DUAL")
    print("-----------------------------------------")
    answer_d()
```

Άσκηση 2.

Θεωρήστε το πρόβλημα γραμμικού προγραμματισμού:

$$\min z = x_1 + x_2$$

όταν

$$\begin{aligned} 2x_1 + 3x_2 + x_3 + x_4 &\leq 0 \\ -x_1 + x_2 + 2x_3 + x_4 &= 6 \\ 3x_1 + x_2 + 4x_3 + 2x_4 &\geq 3 \\ x_1 \leq 0,\ x_2,\ x_4 \geq 0,&\ x_3 \in \mathbb{R} \end{aligned}$$

(α) Καταστρώστε το δυϊκό του παραπάνω προβλήματος. Λύστε το δυϊκό πρόβλημα με τη βοήθεια κατάλληλης βιβλιοθήκης γραμμικού προγραμματισμού της python. Περιγράψτε (αλγεβρικά και γεωμετρικά) τη βέλτιστη λύση, εφ΄ όσον υπάρχει (Σημ. δώστε τιμή αντικειμενικής συνάρτησης, τιμές μεταβλητών, κορυφή, δεσμευτικοί/μη δεσμευτικοί περιορισμοί.)

Μετατροπή σε τυπική μορφή του πρωτεύοντος και ύστερα εύρεση του δυϊκού:

$$\left.\begin{cases} 2y_1 - y_2 + 3y_3 \geq 1 & (x_1 \leq 0) \\ 3y_1 + y_2 + y_3 \leq 1 & (x_2 \geq 0) \\ y_1 + 2y_2 + 4y_3 = 0 & (x_3 \ \in \mathrm{R}) \\ y_1 + y_2 + 2y_3 \leq \ 0 & (x_4 \geq 0) \end{cases}\right\} \kappa\alpha\iota\ y_1 \leq 0\ \kappa\alpha\iota\ y_3 \geq 0\ \ \kappa\alpha\iota\ y_2 \in \mathrm{R}$$

Λύση με Pulp:

Status: Optimal
Objective value: -1.8
y1 = 0.0
y2 = -0.4
y3 = 0.2

y1=0: άρα η λύση **του πρωτεύοντος** θα βρίσκεται σε τομή **3 περιορισμών**, **εκτός** του 1ου, καθώς δεν θα είναι δεσμευτικός. Για το **δευτερεύον** παρατηρούμε ότι για τον περιορισμό $y_1 \leq 0$ θα ισχύει η ισότητα, δηλαδή η κορυφή θα βρίσκεται στο y2-y3 επίπεδο 2D υπερεπίπεδο -η λύση **βρίσκεται πάνω στο επίπεδο του περιορισμού** y1=0.

Π1: $2y_1 - y_2 + 3y_3 = 1\ \delta\varepsilon\sigma\mu\varepsilon\upsilon\tau\iota\kappa\acute{o}\varsigma$

Π2: $3y_1 + y_2 + y_3 = -0.2\ \ \leq 1\ \mu\eta\ \delta\varepsilon\sigma\mu\varepsilon\upsilon\tau\iota\kappa\acute{o}\varsigma$

Π3: $y_1 + 2y_2 + 4y_3 = 0$

Π4: $y_1 + y_2 + 2y_3 = 0\ \delta\varepsilon\sigma\mu\varepsilon\upsilon\tau\iota\kappa\acute{o}\varsigma$

(β) Χρησιμοποιήστε τη λύση του δυϊκού που βρήκατε στο ερώτημα (α) και τη δυϊκή θεωρία για να βρείτε αλγεβρικά τη συμπληρωματική λύση του πρωτεύοντος που σας δόθηκε παραπάνω. Περιγράψτε λεπτομερώς τις διαδικασίες που ακολουθείτε και τις αποφάσεις που παίρνετε. Περιγράψτε κι εδώ τη βέλτιστη λύση, όπως και στο προηγούμενο ερώτημα.

Optimal Z=-1.8

1. Αν ο περιορισμός i του δυϊκού (πρωτεύοντος) ικανοποιείται ως γνήσια ανισότητα τότε η i μεταβλητή του πρωτεύοντος (δυϊκού) είναι υποχρεωτικά μηδέν,
2. Αν η j μεταβλητή του δυϊκού (πρωτεύοντος) είναι θετική τότε ο j περιορισμός του πρωτεύοντος (δυϊκού) ικανοποιείται υποχρεωτικά ως ισότητα.

y3 = 0.2 άρα $3x1+x2+4x3+2x4 = 3$

y2 = -0.4 άρα $-x1+x2+2x3+x4 = 6$

y1 = 0.0 άρα $2x1+3x2+x3+x4 \leq 0$

και Π_2 *μη δεσμευτικός* άρα $\boldsymbol{x_2 = 0}$

$$Z = x_1 + x_2 = -1.8 \ \rightarrow \boldsymbol{x1 = -1.8}$$

Λύνουμε τις εξισώσεις και ανισώσεις:

$$\boldsymbol{2x_3 + x_4 = 4.2}$$

$$\boldsymbol{x_3 + x_4 < 3.6}$$

Άρα: $\boldsymbol{x_3 \geq \ 0.6\ , x_4 \leq 3\ , x_1 = -1.8, x_2 = 0\ \textit{και}\ Z = -1.8}$

Σημείωση:

Βέλτιστη λύση πρωτεύοντος		**Βέλτιστη λύση δυϊκού**
Πολλαπλές λύσεις	$\Rightarrow$	Εκφυλισμένη λύση
Μοναδική και μη-εκφυλισμένη	$\Rightarrow$	Μοναδική και μη-εκφυλισμένη
Πολλαπλές και μη-εκφυλισμένες	$\Rightarrow$	Μοναδική και εκφυλισμένη
Μοναδική και εκφυλισμένη	$\Rightarrow$	Πολλαπλές λύσεις

Το δυικό έχει εκφυλισμένη λύση, επομένως το πρωτεύον έχει πολλαπλές βέλτιστες λύσεις.

Κώδικας

```
import pulp

# Δημιουργία προβλήματος
model = pulp.LpProblem("Dual_LP", pulp.LpMaximize)

# Μεταβλητές: y1 ≤ 0, y3 ≥ 0, y2 ελεύθερη
y1 = pulp.LpVariable("y1", upBound=0)
y2 = pulp.LpVariable("y2")  # unrestricted
y3 = pulp.LpVariable("y3", lowBound=0)

# Αντικειμενική συνάρτηση: max 6y2 + 3y3
model += 6*y2 + 3*y3, "Objective"

# Περιορισμοί
model += 2*y1 - y2 + 3*y3 >= 1    # (x1 ≤ 0)
model += 3*y1 + y2 + y3 <= 1      # (x2 ≥ 0)
model += y1 + 2*y2 + 4*y3 == 0    # (x3 ∈ ℝ)
model += y1 + y2 + 2*y3 <= 0      # (x4 ≥ 0)

# Επίλυση
model.solve(pulp.PULP_CBC_CMD(msg=False))

# Αποτελέσματα
print("Status:", pulp.LpStatus[model.status])
print("Objective value:", pulp.value(model.objective))
for var in model.variables():
    print(f"{var.name} = {var.varValue}")
```

Άσκηση 3.

Θεωρήστε το πρόβλημα γραμμικού προγραμματισμού:

$$\max z = 6x_1 + x_2 - x_3 - x_4$$

όταν

$$\begin{aligned} x_1 + 2x_2 + x_3 + x_4 &\leq 5 \\ 3x_1 + x_2 - x_3 &\leq 8 \\ x_2 + x_3 + x_4 &= 1 \\ x_1, x_2 \in \mathbb{R} \text{ και } x_3, x_4 &\geq 0 \end{aligned}$$

Χρησιμοποιήστε τη δυϊκή θεωρία για να εξετάσετε αν η λύση x = (3,−1,0,2) είναι βέλτιστη για το παραπάνω πρόβλημα γ.π., όμως χωρίς την εφαρμογή του αλγορίθμου Simplex (ή οποιουδήποτε επιλυτή) είτε στο ίδιο το πρόβλημα είτε στο δυϊκό του.

Αρχικά ελέγχουμε αν ικανοποιούνται οι περιορισμοί:

Π1: $2 \leq 5$

Π2: $8 \leq 8$ δεσμευτικός

Π3: $1 = 1$ δεσμευτικός

Επίσης x1, x2 ∈ R και x3,x4 ≥ 0 ισχύουν

Άρα το διάνυσμα είναι εφικτό και δίνει Z=15.

Δυϊκό:

$$min = 5y_1 + 8y_2 + y_3$$

$$\Pi 1 \leq 5 \text{ άρα } y_1 \geq 0, \Pi 2 \leq 8 \text{ άρα } y_2 \geq 0, \Pi 3 = 1 \text{ άρα } y_3 \in \mathrm{R}$$

$$x_1 \in \mathrm{R} \text{ άρα } y_1 + 3y_2 = 6$$

$$x_2 \in \mathrm{R} \text{ άρα } 2y_1 + y_2 + y_3 = 1$$

$$x_3 \geq 0 \text{ άρα } y_1 - y_2 + y_3 \geq -1$$

$$x_4 \geq 0 \text{ άρα } y_1 + y_3 \geq -1$$

Συνθήκες συμπληρωματικής χαλαρότητας:

Ο Π1 ικανοποιείται σαν γνήσια ανισότητα, επομένως το $\boldsymbol{y1=0}$.

Π2 και Π3 είναι δεσμευτικοί ⇒ καμιά δέσμευση για y2,y3.

Επίσης

2. Αν η j μεταβλητή του δυϊκού (πρωτεύοντος) είναι θετική τότε ο j περιορισμός του πρωτεύοντος (δυϊκού) ικανοποιείται υποχρεωτικά ως ισότητα.

Άρα αφού $x_4 = 2 \Rightarrow y_1 + y_3 = -1$

Επομένως $\boldsymbol{y_3} = \boldsymbol{-1}$ και από $2y_1 + y_2 + y_3 = 1$ *έχουμε ότι* $\boldsymbol{y_2} = \mathbf{2}$.

Ο περιορισμός 3 όμως: $y_1 - y_2 + y_3 = -3 \geq -1$ αποτυγχάνει.

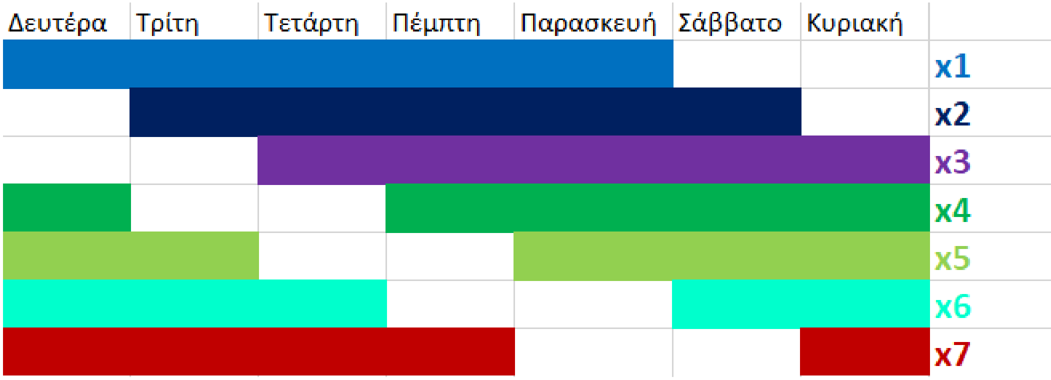
Επομένως δεν υπάρχει εφικτό δυικό y που να ικανοποιεί τις συμπληρωματικές συνθήκες με το x.

Άσκηση 4.

Ένα μεγάλο εστιατόριο ορίζει τις βάρδιες εργασίας για τους σερβιτόρους του έτσι ώστε ο καθένας να εργάζεται για 5 συνεχόμενες ημέρες της εβδομάδας και να παίρνει 2 συνεχόμενες ημέρες ρεπό. Για παράδειγμα οι σερβιτόροι μιας βάρδιας μπορεί να εργαστούν από Κυριακή έως Πέμπτη και στη συνέχεια να πάρουν ρεπό την Παρασκευή και το Σάββατο. Έστω ότι απαιτείται να βρίσκονται στο εστιατόριο κατ΄ελάχιστον 8 σερβιτόροι κάθε Δευτέρα, Τρίτη, Τετάρτη και Πέμπτη, 15 σερβιτόροι κάθε Παρασκευή και Σάββατο και 10 σερβιτόροι κάθε Κυριακή. Ο στόχος του μάνατζερ του εστιατορίου είναι να ικανοποιήσει αυτήν την απαίτηση με το μικρότερο δυνατό πλήθος σερβιτόρων.

(α) Μοντελοποιήστε το παραπάνω πρόβλημα ως πρόβλημα ακέραιου γραμμικού προγραμματισμού.

Έστω x_i οι σερβιτόροι που ξεκινάνε να δουλεύουν μια συγκεκριμένη μέρα.



Συνάρτηση κόστους: $Z = \min(\sum x_i)$

Περιορισμοί:

$$Δευτέρα\ x_1 + x_4 + x_5 + x_6 + x_7 \geq 8$$

$$Τρίτη\ x_1 + x_2 + x_5 + x_6 + x_7 \geq 8$$

$$Τετάρτη\ x_1 + x_2 + x_3 + x_6 + x_7 \geq 8$$

$$Πέμπτη\ x_1 + x_2 + x_3 + x_4 + x_7 \geq 8$$

$$Παρασκευή\ x_1 + x_2 + x_3 + x_4 + x_5 \geq 15$$

$$Σάββατο\ x_6 + x_2 + x_3 + x_4 + x_5 \geq 15$$

$$Κυριακή\ x_6 + x_7 + x_3 + x_4 + x_5 \geq 10$$

(β) Λύστε το πρόβλημα με τη βοήθεια του αλγορίθμου Branch & Bound. Για την επίλυση των προβλημάτων γραμμικού προγραμματισμού στους κόμβους του γράφου χρησιμοποιήσετε κάποιον από τους solvers γραμμικού προγραμματισμού που αναφέρθηκαν στις διαλέξεις μας. Σε περίπτωση πολλαπλών βέλτιστων λύσεων βρείτε τουλάχιστον τρεις και σχολιάστε τις διαφορές τους.

Ξεκινάμε επίλυση με pulp.

Βήμα 1:

Βέλτιστη Τιμή: 15.33333326
x1 = 0.0
x2 = 0.33333333
x3 = 0.0
x4 = 7.3333333
x5 = 0.0
x6 = 7.3333333
x7 = 0.33333333

Ας κάνουμε branch στη x4

$x_4 \geq 8$:

Βέλτιστη Τιμή: 16.0
x1 = 0.0
x2 = 1.0
x3 = 0.0
x4 = 8.0
x5 = 0.0
x6 = 6.0
x7 = 1.0

Βέλτιστη Λύση!
Σταματάμε εδώ

$x_4 \leq 7$:

Βέλτιστη Τιμή: 15.33333332
x1 = 0.0
x2 = 0.33333333
x3 = 0.33333333
x4 = 7.0
x5 = 0.33333333
x6 = 7.0
x7 = 0.33333333

Μη ακέραιες τιμές!
Συνεχίζουμε

$x_2 \geq 1$:

Βέλτιστη Τιμή: 16.0
x1 = 0.0
x2 = 1.0
x3 = 0.0
x4 = 7.0
x5 = 1.0
x6 = 6.0
x7 = 1.0

Βέλτιστη Λύση!
Σταματάμε εδώ

$x_2 \leq 0$:

Βέλτιστη Τιμή: 15.5
x1 = 0.0
x2 = 0.0
x3 = 5.5
x4 = 2.0
x5 = 5.5
x6 = 2.0
x7 = 0.5

Μη ακέραιες τιμές!
Συνεχίζουμε

$x_3 \geq 6$:

Βέλτιστη Τιμή: 16.0
x1 = 0.0
x2 = 0.0
x3 = 6.0
x4 = 1.0
x5 = 7.0
x6 = 1.0
x7 = 1.0

Βέλτιστη Λύση!
Σταματάμε εδώ

$x_3 \leq 5$:

Βέλτιστη Τιμή: 15.5
x1 = 0.0
x2 = 0.0
x3 = 5.0
x4 = 2.5
x5 = 5.0
x6 = 2.5
x7 = 0.5

Μη ακέραιες τιμές!
Συνεχίζουμε …

Έχουν βρεθεί τρεις βέλτιστες λύσεις ήδη οπότε σταματάμε:

$x_3 \geq 6$:	$x_2 \geq 1$:	$x_4 \geq 8$:
Βέλτιστη Τιμή: 16.0	*Βέλτιστη Τιμή: 16.0*	*Βέλτιστη Τιμή: 16.0*
x1 = 0.0	*x1 = 0.0*	*x1 = 0.0*
x2 = 0.0	*x2 = 1.0*	*x2 = 1.0*
x3 = 6.0	*x3 = 0.0*	*x3 = 0.0*
x4 = 1.0	*x4 = 7.0*	*x4 = 8.0*
x5 = 7.0	*x5 = 1.0*	*x5 = 0.0*
x6 = 1.0	*x6 = 6.0*	*x6 = 6.0*
x7 = 1.0	*x7 = 1.0*	*x7 = 1.0*

Κώδικας με branches στα σχόλια

```python
import pulp

model = pulp.LpProblem("Waiter_Scheduling_Integer", pulp.LpMinimize)

# Δημιουργία μεταβλητών x1 έως x7
x = [pulp.LpVariable(f"x{i+1}", lowBound=0, cat='Continuous') for i in range(7)]

# Συνάρτηση κόστους: ελαχιστοποίηση συνολικού αριθμού σερβιτόρων
model += pulp.lpSum(x), "Total_Waiters"

# Περιορισμοί ανά ημέρα
model += x[0] + x[3] + x[4] + x[5] + x[6] >= 8   # Δευτέρα
model += x[0] + x[1] + x[4] + x[5] + x[6] >= 8   # Τρίτη
model += x[0] + x[1] + x[2] + x[5] + x[6] >= 8   # Τετάρτη
model += x[0] + x[1] + x[2] + x[3] + x[6] >= 8   # Πέμπτη
model += x[1] + x[2] + x[3] + x[4] + x[5] >= 15  # Παρασκευή
model += x[2] + x[3] + x[4] + x[5] + x[6] >= 15  # Σάββατο
model += x[0] + x[3] + x[4] + x[5] + x[6] >= 10  # Κυριακή

#branch A
#model += x[3] <= 7

#branch A1
# κραταμε το model += x[3] <= 7  # από A
#model += x[1] <= 0  # A1 = x2 <= 0

#branch A11
#model+=x[2]>=6 # x3 >= 6 #optimal solution

#branch A12
#model+=x[2]<=5

# #branch A2
# model += x[3] <= 7  # από A
# model += x[1] >= 1   #optimal solution

# #branch B
# model += x[3] >= 8 #optimal solution

# Λύση με solver CBC

solver = pulp.PULP_CBC_CMD(msg=True)
model.solve(solver)

# Εμφάνιση αποτελεσμάτων
print("Status:", pulp.LpStatus[model.status])
print("Βέλτιστη Τιμή:", pulp.value(model.objective))
```

```
for var in x:
    print(f"{var.name} = {var.varValue}")
```

Κώδικας για απευθείας εύρεση της τελικής λύσης

```
import pulp

# Δημιουργία μοντέλου
model = pulp.LpProblem("Waiter_Scheduling", pulp.LpMinimize)

# Μεταβλητές: x1 έως x7 (ακέραιες, μη αρνητικές)
x = [pulp.LpVariable(f"x{i+1}", lowBound=0, cat='Integer') for i in range(7)]

# Συνάρτηση κόστους: ελαχιστοποίηση του συνολικού αριθμού σερβιτόρων
model += pulp.lpSum(x), "Total_Waiters"

# Περιορισμοί (από πίνακα εικόνας)
model += x[0] + x[3] + x[4] + x[5] + x[6] >= 8   # Δευτέρα
model += x[0] + x[1] + x[4] + x[5] + x[6] >= 8   # Τρίτη
model += x[0] + x[1] + x[2] + x[5] + x[6] >= 8   # Τετάρτη
model += x[0] + x[1] + x[2] + x[3] + x[6] >= 8   # Πέμπτη
model += x[1] + x[2] + x[3] + x[4] + x[5] >= 15  # Παρασκευή
model += x[2] + x[3] + x[4] + x[5] + x[6] >= 15  # Σάββατο
model += x[0] + x[3] + x[4] + x[5] + x[6] >= 10  # Κυριακή

# Επίλυση με χρήση του solver CBC (που χρησιμοποιεί Branch & Bound)
solver = pulp.PULP_CBC_CMD(msg=1)
status = model.solve(solver)

# Εκτύπωση αποτελεσμάτων
print(f"Status: {pulp.LpStatus[status]}")
print(f"Βέλτιστη Τιμή: {pulp.value(model.objective)}")

for var in x:
    print(f"{var.name} = {var.varValue}")
```

Άσκηση 5.

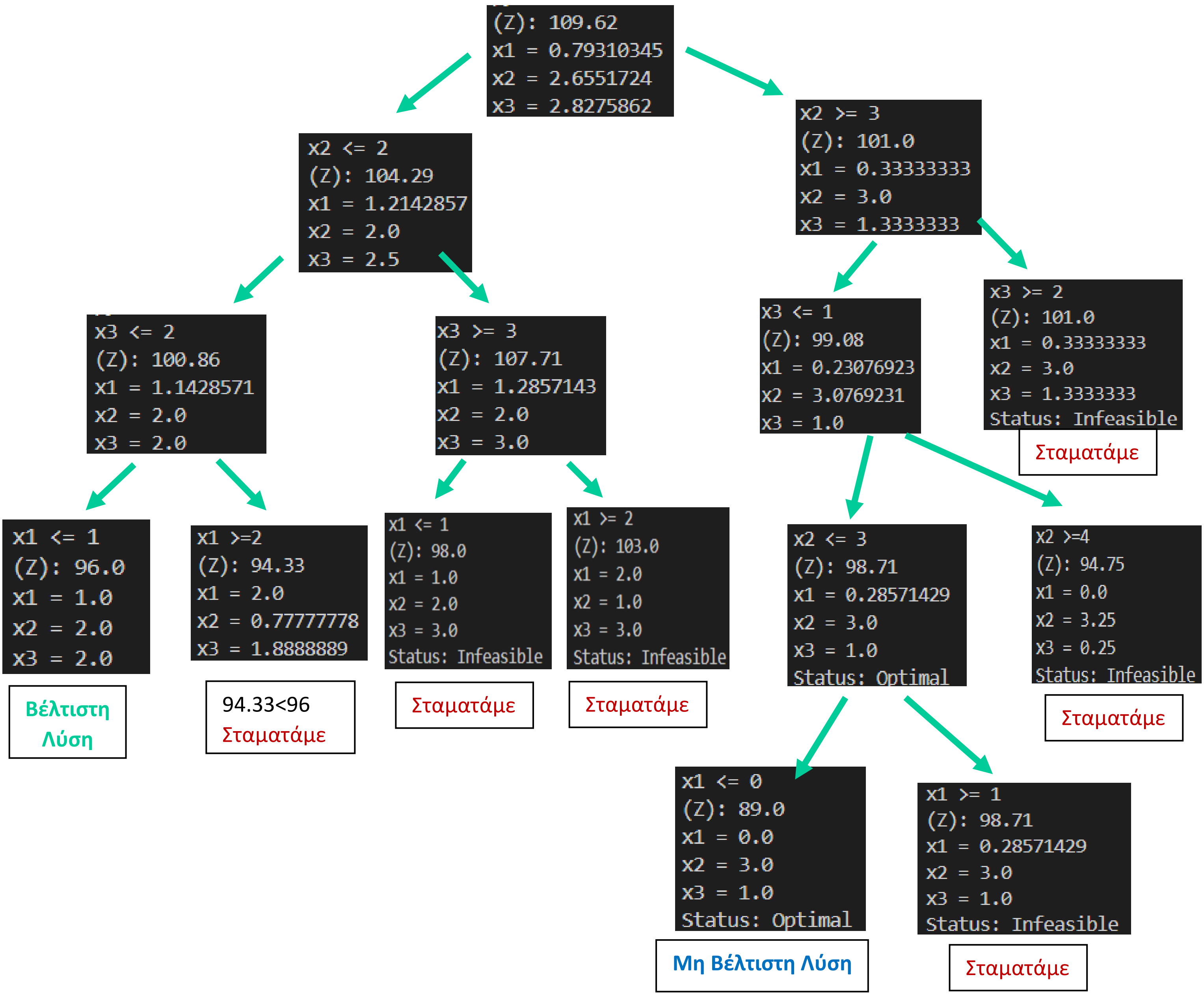
Θεωρήστε το πρόβλημα ακέραιου γραμμικού προγραμματισμού:

$$\max\ 34x_1 + 29x_2 + 2x_3$$

όταν

$$\begin{aligned} 7x_1 + 5x_2 - x_3 &\leq 16 \\ -x_1 + 3x_2 + x_3 &\leq 10 \\ -x_2 + 2x_3 &\leq 3 \\ x_1, x_2, x_3 &\geq 0 \quad \text{και ακέραιοι} \end{aligned}$$

Λύστε το με τη βοήθεια του του αλγορίθμου Branch & Bound. Για την επίλυση των προβλημάτων γραμμικού προγραμματισμού στους κόμβους του γράφου χρησιμοποιήσετε κάποιον από τους solvers γραμμικού προγραμματισμού που αναφέρθηκαν στις διαλέξεις μας.



Υπάρχουν 2 εφικτές λύσεις με βέλτιστη την:

(Z): 96.0

x1 = 1.0

x2 = 2.0

x3 = 2.0

Status: Optimal

Κώδικας με branches στα σχόλια

```
import pulp
# Ορισμός προβλήματος μεγιστοποίησης
model = pulp.LpProblem("Integer_LP_BranchBound", pulp.LpMaximize)
# Μεταβλητές (χαλάρωση: συνεχείς, όχι ακέραιες)
x1 = pulp.LpVariable("x1", lowBound=0, cat='Continuous')
x2 = pulp.LpVariable("x2", lowBound=0, cat='Continuous')
x3 = pulp.LpVariable("x3", lowBound=0, cat='Continuous')
# x1 = pulp.LpVariable("x1", lowBound=0, cat='Integer')
# x2 = pulp.LpVariable("x2", lowBound=0, cat='Integer')
# x3 = pulp.LpVariable("x3", lowBound=0, cat='Integer')

# Αντικειμενική συνάρτηση
model += 34 * x1 + 29 * x2 + 2 * x3

# Περιορισμοί
model += 7 * x1 + 5 * x2 - x3 <= 16
model += -x1 + 3 * x2 + x3 <= 10
model += -x2 + 2 * x3 <= 3

# # branch A
model += x2 <= 2
# print("x2 <= 2")

# # branch A1
model += x3 <= 2
# print("x3 <= 2")

# # branch A11
model += x1 <= 1 ## optimal solution
# print("x1 <= 1") ## optimal solution

# # branch A12
# model += x1>=2 ## optimal solution
# print("x1 >=2") ## optimal solution

# # branch A2
```

```python
#model += x3 >=3
#print("x3 >= 3")

# branch A21
#model += x1 <= 1
#print("x1 <= 1")

# branch A22
# model += x1 >= 2
# print("x1 >= 2 ")

# branch B
#model += x2 >= 3
# print("x2 >= 3")

# # branch B1
#model += x3 <= 1
#print("x3 <= 1")

# # branch B11
#model += x2 <= 3
#print("x2 <= 3")

# # branch B111
# model += x1 <= 0
# print("x1 <= 0")

# # branch B112
# model += x1 >= 1
# print("x1 >= 1")

# # branch B12
# model += x2 >=4
# print("x2 >=4")

# # branch B2
# model += x3 >= 2
# print("x3 >= 2")

# Επίλυση
solver = pulp.PULP_CBC_CMD(msg=False)
model.solve(solver)
# Εμφάνιση αποτελεσμάτων
print("(Z):", round(pulp.value(model.objective),2))
print(f"x1 = {x1.varValue}")
print(f"x2 = {x2.varValue}")
print(f"x3 = {x3.varValue}")
print("Status:", pulp.LpStatus[model.status])
```

Άσκηση 6.

Μια εταιρεία courier θέλει να μεγιστοποιήσει τα συνολικά ημερήσια έσοδά της. Για την παράδοση των δεμάτων, η εταιρεία διαθέτει ένα αυτοκίνητο με όγκο έντεκα (μονάδες όγκου). Υπάρχουν τα εξής δέματα για παράδοση: το δέμα 1 με όγκο δύο, το δέμα 2 με όγκο τρία, το δέμα 3 με όγκο τέσσερα, το δέμα 4 με όγκο έξι, και το δέμα 5 με όγκο οκτώ. Τα έσοδα από την παράδοση των δεμάτων είναι 10,14,31,48, και 60, αντίστοιχα.

(α) Μοντελοποιήστε αυτό το πρόβλημα με τη βοήθεια ενός μοντέλου ακέραιου γραμμικού προγραμματισμού. Τι τύπος προβλήματος προκύπτει; Λύστε το με τη μέθοδο branch-and-bound και σχεδιάστε το δέντρο που προκύπτει.

Συνάρτηση Κέρδους:

$$Z = 10x_1 + 14x_2 + 31x_3 + 48x_4 + 60x_5$$

Περιορισμός:

$$11 \geq 2x_1 + 3x_2 + 4x_3 + 6x_4 + 8x_5$$

Με

$$x_i \in \{0,1\}$$

Το πρόβλημα είναι τύπου **0-1 Knapsack Problem.** Το x δείχνει αν το δέμα μπαίνει στον σάκο ή όχι (1 ή 0).

Επίλυση με κώδικα

```
├── x = [0.    0.    1.    1.    0.125], val = 86.50
│   ├── x = [0.  0.  0.  0.5 1. ], val = 84.00
│   │   ├── [infeasible or undefined]
│   │   ├── x = [0.    0.    0.75 0.    1.   ], val = 83.25
│   │   │   ├── [infeasible or undefined]
│   │   │   ├── x = [1.    0.333 0.    0.    1.    ], val = 74.67
│   │   │   │   ├── x = [0. 1. 0. 0. 1.], val = 74.00 ✓
│   │   │   │   ├── x = [1. 0. 0. 0. 1.], val = 70.00 ✓
│   ├── x = [0.5 0.  1.  1.  0. ], val = 84.00
│   │   ├── x = [1.    0.    0.75 1.    0.   ], val = 81.25
│   │   │   ├── x = [1.    0.    1.    0.833 0.    ], val = 81.00
│   │   │   │   ├── [infeasible or undefined]
│   │   │   │   ├── x = [1. 1. 1. 0. 0.], val = 55.00 ✓
│   │   │   ├── x = [1. 1. 0. 1. 0.], val = 72.00 ✓
│   │   ├── x = [0.    0.333 1.    1.    0.    ], val = 83.67
│   │   │   ├── x = [0.  1.  0.5 1.  0. ], val = 77.50
│   │   │   │   ├── x = [0.    1.    1.    0.667 0.    ], val = 77.00
│   │   │   │   │   ├── [infeasible or undefined]
│   │   │   │   │   ├── x = [0. 1. 1. 0. 0.], val = 45.00 ✓
│   │   │   │   ├── x = [0. 1. 0. 1. 0.], val = 62.00 ✓
│   │   │   ├── x = [0. 0. 1. 1. 0.], val = 79.00 ✓
```

=== Τελικό αποτέλεσμα ===
Βέλτιστο Z: 79.0 (=31+48)
Επιλεγμένα δέματα: [0, 0, 1, 1, 0] (δέμα 3 και δέμα 4)

(β) Διατυπώστε το δυϊκό του αρχικού χαλαρωμένου προβλήματος μαζί και τις αντίστοιχες σχέσεις συμπληρωματικής χαλαρότητας (complementary slackness) για τα δύο προβλήματα. Χρησιμοποιήστε τις σχέσεις αυτές και τη βέλτιστη λύση του (χαλαρωμένου) προβλήματος από το (α) για να προσδιορίσετε μία βέλτιστη λύση του δυϊκού προβλήματος.

Χαλαρωμένο Πρωτεύον:

$$Z = 10x_1 + 14x_2 + 31x_3 + 48x_4 + 60x_5$$

Περιορισμός:

$$11 \geq 2x_1 + 3x_2 + 4x_3 + 6x_4 + 8x_5$$

Με

$$0 \leq x_i \leq 1 \ \forall\ x_i$$

Δυϊκό:

$$\boldsymbol{min}\ 11y_1 + y_2 + \cdots + y_6$$

$$2y_1 + y_2 \geq 10$$

$$3y_1 + y_3 \geq 14$$

$$4y_1 + y_4 \geq 31$$

$$6y_1 + y_5 \geq 48$$

$$8y_1 + y_6 \geq 60$$

$$y_1, \dots, y_n \geq 0$$

Βέλτιστη Λύση Χαλαρωμένου Προβλήματος:

x1 = 0.0, x2 = 0.0, x3 = 1.0, x4 = 1.0, x5 = 0.125, $\text{Objective} = 86.5$

capacity filled = 11.0

Σχέσεις Συμπληρωματικής Χαλαρότητας:

1. Αν ο περιορισμός i του δυϊκού (πρωτεύοντος) ικανοποιείται ως γνήσια ανισότητα τότε η i μεταβλητή του πρωτεύοντος (δυϊκού) είναι υποχρεωτικά μηδέν,

Όταν ένας περιορισμός *i* είναι μη δεσμευτικός στο πρωτεύον τότε η μεταβλητή *i* του δυϊκού θα είναι υποχρεωτικά μηδέν. Ο περιορισμός 1 είναι δεσμευτικός. Οι περιορισμοί $x_i \leq 1\ \text{για}\ i = 1{,}2{,}5$ δεν είναι δεσμευτικοί επομένως έχουμε $y_2 = y_3 = y_6 = 0$.

2. Αν η j μεταβλητή του δυϊκού (πρωτεύοντος) είναι θετική τότε ο j περιορισμός του πρωτεύοντος (δυϊκού) ικανοποιείται υποχρεωτικά ως ισότητα.

Έχουμε $\mathrm{x}_3 = 1.0$, $\mathrm{x}_4 = 1.0$, $\mathrm{x}_5 = 0.125$, επομένως Π3, Π4, Π5 θα ικανοποιούνται ως ισότητες.

$4y_1 + y_4 = 31$ και $6y_1 + y_5 = 48$ και $8y_1 + y_6 = 60$.

Άρα: $\boxed{y_1 = \frac{60}{8} = 7.5, y_2 = 0, y_3 = 0, y_4 = 1, y_5 = 3, y_6 = 0}$

Αυτό δίνει $\boxed{\text{Objective} = 86.5.}$

Οι λύσεις ταυτίζονται επομένως είναι **βέλτιστες**.

(γ) Με τη βοήθεια κάποιου γνωστού solver για ακέραιο γραμμικό προγραμματισμό προσδιορίστε τη βέλτιστη ακέραια λύση του δυϊκού που βρήκατε στο ερώτημα (β). Ελέγξτε αν ισχύουν οι σχέσεις συμπληρωματικής χαλαρότητας για τις βέλτιστες ακέραιες λύσεις του πρωτεύοντος και του αντίστοιχου δυϊκού προβλήματος.

Status: Optimal

Objective value (dual): 88.0

y1 = 8.0000

y2 = 0.0000

y3 = 0.0000

y4 = 0.0000

y5 = 0.0000

y6 = 0.0000

Η ακέραιη λύση του δυϊκού είναι εφικτή και έχει objective = 88.0 ενώ η αντίστοιχη primal βέλτιστη είχε objective = 79.0.

Επομένως αφού οι τιμές τους είναι διαφορετικές δεν ισχύουν οι συνθήκες συμπληρωματικής χαλαρότητας για τις βέλτιστες ακέραιες λύσεις του πρωτεύοντος και του αντίστοιχου δυϊκού προβλήματος..

Κώδικας 6a

```
import pulp
import numpy as np

def is_integral(solution, tol=1e-5):
    return all(abs(x - round(x)) <= tol for x in solution)

def branch_and_bound_knapsack():
    profits = [10, 14, 31, 48, 60]
    volumes = [2, 3, 4, 6, 8]
    capacity = 11
```

```
    n = len(profits)

    best_value = -np.inf
    best_solution = None
    stack = []
    seen = set()

    x_vars = [pulp.LpVariable(f"x{i}", lowBound=0, upBound=1, cat='Continuous') for
i in range(n)]

    def create_model(extra_constraints=None):
        model = pulp.LpProblem("Knapsack_BnB", pulp.LpMaximize)
        model += pulp.lpSum([profits[i] * x_vars[i] for i in range(n)])
        model += pulp.lpSum([volumes[i] * x_vars[i] for i in range(n)]) <= capacity
        if extra_constraints:
            for c in extra_constraints:
                model += c
        return model

    stack.append(([], 0))

    while stack:
        constraints, depth = stack.pop()
        indent = "│   " * depth + "├── "

        model = create_model(constraints)
        glpk_path =
r"C:\Users\USER\Documents\_LinearIntegerOptimization\SETS\set2\glpk-
4.65\w64\glpsol.exe"
        status = model.solve(pulp.GLPK_CMD(path=glpk_path, msg=0))

        if status != pulp.LpStatusOptimal:
            print(f"{indent}[infeasible or undefined]")
            continue

        sol = [x.varValue for x in x_vars]
        val = pulp.value(model.objective)
        key = tuple(round(v, 3) for v in sol)
        if key in seen:
            print(f"{indent} Skipping duplicate")
            continue
        seen.add(key)

        status_note = "✓" if is_integral(sol) else ""
        print(f"{indent}x = {np.round(sol, 3)}, val = {val:.2f} {status_note}")

        if val <= best_value:
            continue
```

```
        if is_integral(sol):
            if val > best_value:
                best_value = val
                best_solution = [int(round(v)) for v in sol]
            continue

        for i in range(n):
            if abs(sol[i] - round(sol[i])) > 1e-5:
                floor_val = np.floor(sol[i])
                ceil_val = np.ceil(sol[i])
                stack.append((constraints + [x_vars[i] <= floor_val], depth + 1))
                stack.append((constraints + [x_vars[i] >= ceil_val], depth + 1))
                break

    return best_value, best_solution

# Εκτέλεση
best_value, best_solution = branch_and_bound_knapsack()

if best_solution is None:
    print("Καμία λύση δεν βρέθηκε.")
else:
    print("\n=== Τελικό αποτέλεσμα ===")
    print("Βέλτιστο Z:", best_value)
    print("Επιλεγμένα δέματα:", best_solution)
```

Κώδικας 6b

```
import pulp
import numpy as np

profits = [10, 14, 31, 48, 60]
volumes = [2, 3, 4, 6, 8]
capacity = 11
n = len(profits)
x_vars = [pulp.LpVariable(f"x{i}", lowBound=0, upBound=1, cat='Continuous') for i in
range(n)]

model = pulp.LpProblem("Knapsack_BnB", pulp.LpMaximize)
model += pulp.lpSum([profits[i] * x_vars[i] for i in range(n)])
model += pulp.lpSum([volumes[i] * x_vars[i] for i in range(n)]) <= capacity

model.solve(pulp.PULP_CBC_CMD(msg=0))
print("Βέλτιστη ακέραια λύση του προβλήματος:")
for i in range(n):
```

```python
    print(f"x{i+1} =", x_vars[i].varValue, end=", ")
print("Objective =", pulp.value(model.objective))
print("capacity filled =", sum(volumes[i] * x_vars[i].varValue for i in range(n)))
print("status =", pulp.LpStatus[model.status])
```

Κώδικας 6c

```python
import pulp

# Model
# Δημιουργία του μοντέλου (ελαχιστοποίηση)
dual = pulp.LpProblem("Dual_LP", pulp.LpMinimize)

# Dual μεταβλητές: y1 για τον περιορισμό βάρους, y2-y6 για τα upper bounds
y_vars = []
for i in range(1, 7):
    y_vars.append(pulp.LpVariable(f'y{i}', lowBound=0, cat='Integer'))

# Αντικειμενική συνάρτηση: min 11*y1 + y2 + y3 + y4 + y5 + y6
y1, y2, y3, y4, y5, y6 = y_vars
dual += 11*y1 + y2 + y3 + y4 + y5 + y6, "Objective"

# Dual constraints για κάθε x_i
dual += 2*y1 + y2 >= 10   # για x1
dual += 3*y1 + y3 >= 14   # για x2
dual += 4*y1 + y4 >= 31   # για x3
dual += 6*y1 + y5 >= 48   # για x4
dual += 8*y1 + y6 >= 60   # για x5

# Επίλυση
dual.solve(pulp.PULP_CBC_CMD(msg=False))

# Εμφάνιση αποτελεσμάτων
print("Status:", pulp.LpStatus[dual.status])
print("Objective value (dual):", pulp.value(dual.objective))
for var in [y1, y2, y3, y4, y5, y6]:
    print(f"{var.name} = {var.varValue:.4f}")
```