Linear Programming

Ονοματεπώνυμο : Νικόλας Φιλιππάτος
AM: 1072754
Εργασία: 2η
Ημερομηνία: May 19, 2024

Table Of Contents

Ασκηση 1

Θεωρήστε το πρόβλημα γραμμικού προγραμματισμού:

$$\begin{array}{rll} max & 5x1 + 3x2 + x3 + 4x4 & \\ & x1 - 2x2 + 2x3 + 3x4 & \leq 10 \\ & 2x1 + 2x2 + 2x3 - x4 & \leq 6 \\ & 3x1 + x2 - x3 + x4 & \leq 10 \\ & -x2 + 2x3 + 2x4 & \leq 7 \\ & x1, x2, x3, x4 & \geq 0 \end{array}$$

Ασκηση 1 (α)

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

Ασκηση 1 (β)

(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. (Σημ. Οι υπολογισμοί θα πρέπει να γίνουν αναλυτικά)

Ασκηση 1 (γ)

(γ) Επιλέξτε έναν δεσμευτικό και έναν μη δεσμευτικό περιορισμό (ή περιορισμό προσήμου) 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. (Σημ. Οι υπολογισμοί θα πρέπει να γίνουν αναλυτικά)

Ασκηση 1 (δ)

(δ) Για μία μη βασική μεταβλητή της βέλτιστης λύσης βρείτε την μεταβολή που πρέπει να υποστεί ο συντελεστής της στην αντικειμενική συνάρτηση για να μετατραπεί σε βασική η συγκεκριμένη μεταβλητή.

Ασκηση 2

Ασκηση 2. Θεωρήστε το πρόβλημα γραμμικού προγραμματισμού:

$$\begin{array}{rll} max & 3x1 - 2x2 - 5x3 + 7x4 + 8x5 & \\ & x2 - x3 + 3x4 - 4x5 & = -6 \\ & 2x1 + 3x2 - 3x3 - x4 & \geq 2 \\ & x1 + 2x3 - 2x4 & \leq -5 \\ & -2 \leq x1 & \leq 10 \\ & 5 \leq x2 & \leq 25 \\ & x3, x4 \geq 0, x5 \in R & \end{array}$$

Ασκηση 2 (α)

(α) Γράψτε το δυϊκό του παραπάνω προβλήματος.

Ασκηση 2 (β)

(β) Αν συμβολίσουμε με x6 έως x12 τις μεταβλητές χαλάρωσης των περιορισμών του πρωτεύοντος, θεωρήστε τη λύση που έχει ως βασικές μεταβλητές τις x2 , x4 , x5 , x8 , x9 , x10 , x11 .
Γράψτε τον βασικό πίνακα B για το πρωτεύον και τον συμπληρωματικό του B C για το δυϊκό και υπολογίστε τις βασικές λύσεις που αντιστοιχούν στα δύο προβλήματα. Εξετάστε αν ισχύει η συμπληρωματικότητα μεταξύ των δύο λύσεων. Επιπλέον, εξετάστε αν είναι δυνατή η εφαρμογή των θεωρημάτων ασθενούς ή/και ισχυρής δυϊκότητας για τις λύσεις που βρήκατε. Εξηγήστε πλήρως την απάντησή σας.

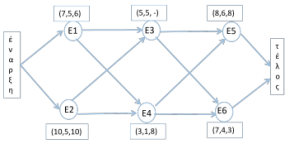
Ασκηση 2 (γ)

(γ) Λύστε τα δύο προβλήματα με κάποιον από τους επιλυτές που διδαχτήκατε.

Ασκηση 3

Θεωρήστε το πρόβλημα του προγραμματισμού εργασιών που σας ζητήθηκε να μοντελοποιήσετε στην Ασκηση 3 της Εργασίας]1 και παρατίθεται για διευκόλυνσή σας πιο κάτω. Χρησιμοποιήστε έναν από τους επιλυτές (solvers) γραμμικού προγραμματισμού που αναφέρθηκαν στις διαλέξεις για την επίλυσή του. (Σημείωση. Σε περίπτωση που η μοντελοποίηση που είχατε προηγουμένως δεν σας ικανοποιεί μπορείτε να την αλλάξετε/βελτιώσετε στην παρούσα Εργασία.)

> Θεωρήστε ένα έργο το οποίο για να ολοκληρωθεί απαιτεί την διεκπεραίωση 6 επί μέρους εργασίες (Ε1 - Ε6). Οι εργασίες είναι εξαρτημένες μεταξύ τους και οι εξαρτήσεις δίνονται με το παρακάτω σχήμα:



Σύμφωνα με το σχήμα οι εργασίες Ε1 και Ε2 μπορούν να εκτελεστούν παράλληλα, όμως για να εκτελεστεί η εργασία Ε3 θα πρέπει να έχουν ολοκληρωθεί οι εργασίες Ε1 και Ε2. Ομοίως και για τις υπόλοιπες εργασίες. Το έργο ολοκληρώνεται όταν εκτελεστούν (παράλ- ληλα) οι εργασίες Ε5 και Ε6. 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 περισσοτέρων 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.

Ασκηση 4

Θεωρήστε το πρόβλημα γραμμικού προγραμματισμού:

$$
\begin{aligned}
max \quad z = x1 + 2x2 + x3 - 3x4 + x5 + x6 - x7 & \\
x1 + x2 - x4 + 2x6 - 2x7 & \leq 6 \\
x2 - x4 + x5 - 2x6 + 2x7 & \leq 4 \\
x2 + x3 + x6 - x7 & \leq 2 \\
x2 - x4 - x6 + x7 & \leq 1 \\
x1, \ldots, x7 & \geq 0
\end{aligned}
$$

Χρησιμοποιήστε τη δυϊκή θεωρία για να εξετάσετε αν η λύση $x = (\frac{15}{2}, 0, \frac{11}{4}, 0, \frac{5}{2}, 0, \frac{3}{4})$ είναι βέλτιστη για το παραπάνω πρόβλημα γ.π. χωρίς όμως την εφαρμογή του αλγορίθμου Simplex (ή οποιουδήποτε επιλυτή) είτε στο ίδιο το πρόβλημα είτε στο δυϊκό του.

Ασκηση 5

Θεωρήστε το πρόβλημα του σακιδίου:

$$
\begin{aligned}
max \quad z = 23x1 + 17x2 + 30x3 + 14x4 + 9x5 & \\
6x1 + 5x2 + 10x3 + 7x4 + 5x5 & \leq 14 \\
xi \in 0, 1 i = 1, 2, 3, 4, 5 &
\end{aligned}
$$

Ασκηση 5 (α)

(α) Εφαρμόστε τον αλγόριθμο Branch & Bound για την επίλυσή του. Φροντίστε η επιλογή του επόμενου κόμβου για διακλάδωση να γίνεται κάθε φορά με κριτήριο την καλύτερη τιμή της αντικειμενικής συνάρτησης (Jumbtracking or Best First).

Ασκηση 5 (β)

(β) Για το παραπάνω πρόβλημα υπάρχει δυνατότητα να συσφίξουμε τον βασικό περιορισμό του προβλήματος ώστε να διευκολυνθεί η αλγοριθμική διαδικασία εύρεσης της βέλτιστης τιμής;

Ασκηση 5 (γ)

(γ) Χρησιμοποιώντας τον περιορισμό του παραπάνω προβλήματος προτείνετε επίπεδα αποκοπής από τρεις διαφορετικές ελάχιστες καλύψεις του. (Σημ. Υπάρχουν περισσότερες από τρεις). Πως διαμορφώνεται το δένδρο της Branch & Bound μετά την εισαγωγή
των επιπέδων αποκοπής;

Solution Exercise 1

> ## Ασκηση 1
>
> Θεωρήστε το πρόβλημα γραμμικού προγραμματισμού:
>
> $$\begin{array}{rcl} max & 5x1+3x2+x3+4x4 & \\ & x1-2x2+2x3+3x4 & \leq 10 \\ & 2x1+2x2+2x3-x4 & \leq 6 \\ & 3x1+x2-x3+x4 & \leq 10 \\ & -x2+2x3+2x4 & \leq 7 \\ & x1,x2,x3,x4 & \geq 0 \end{array}$$
>
> ### Ασκηση 1 (α)
>
> Λύστε το πρόβλημα (με κάποιον από τους επιλυτές που αναφέραμε στις διαλέξεις) και περιγράψτε τη βέλτιστη λύση, δηλ. δώστε τις τιμές των μεταβλητών απόφασης και την τιμή της αντικειμενικής συνάρτησης. Δώστε τις βασικές / μη-βασικές μεταβλητές και τον βέλτιστο βασικό πίνακα. Ξεχωρίστε τους δεσμευτικούς από τους μη δεσμευτικούς περιορισμούς και μέσω αυτών περιγράψτε γεωμετρικά τη βέλτιστη κορυφή
>
> ### Ασκηση 1 (β)
>
> (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. (Σημ. Οι υπολογισμοί θα πρέπει να γίνουν αναλυτικά)
>
> ### Ασκηση 1 (γ)
>
> (γ) Επιλέξτε έναν δεσμευτικό και έναν μη δεσμευτικό περιορισμό (ή περιορισμό προσήμου) 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. (Σημ. Οι υπολογισμοί θα πρέπει να γίνουν αναλυτικά)
>
> ### Ασκηση 1 (δ)
>
> (δ) Για μία μη βασική μεταβλητή της βέλτιστης λύσης βρείτε την μεταβολή που πρέπει να υποστεί ο συντελεστής της στην αντικειμενική συνάρτηση για να μετατραπεί σε βασική η συγκεκριμένη μεταβλητή.

Solution Exercise 1 (a)

> ### Ασκηση 1 (α)
>
> Λύστε το πρόβλημα (με κάποιον από τους επιλυτές που αναφέραμε στις διαλέξεις) και περιγράψτε τη βέλτιστη λύση, δηλ. δώστε τις τιμές των μεταβλητών απόφασης και την τιμή της αντικειμενικής συνάρτησης. Δώστε τις βασικές / μη-βασικές μεταβλητές και τον βέλτιστο βασικό πίνακα. Ξεχωρίστε τους δεσμευτικούς από τους μη δεσμευτικούς περιορισμούς και μέσω αυτών περιγράψτε γεωμετρικά τη βέλτιστη κορυφή

Προσθέτουμε τις μεταβλητές χαλάρωσης :

$$\begin{array}{rcl} max & 5x1+3x2+x3+4x4 & \\ & x1-2x2+2x3+3x4+s_1 & = 10 \\ & 2x1+2x2+2x3-x4+s_2 & = 6 \\ & 3x1+x2-x3+x4+s_3 & = 10 \\ & -x2+2x3+2x4+s_4 & = 7 \\ & x1,x2,x3,x4 & \geq 0 \end{array}$$

Τρέχοντας το πρόγραμμα βρίσκουμε $x = (0, 5.13333, 0.6, 5.46667)$, και το max της αντικειμενικής συνάρτησης $ Z =37.86667$

Οι μεταβλητές χαλάρωσης :

s1 = -2.666650000000004
s2 = -9.999999999621423e-06
s3 = 0.0
s4 = 9.999999997845066e-06

Απο στρογγυλοποίηση :

$$\begin{array}{c} s_1 = -2.67 \\ s_2 = 0 \\ s_3 = 0 \\ s_4 = 0 \end{array}$$

Βασικές μεταβλητές : x_2, x_3, x_4, s_1
Μη βασικές μεταβλητές : x_1, s_2, s_3, s_4

Βέλτιστος βασικός πίνακας :

$$B = [A \quad I_m]$$

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

Δεσμευτικοι Περιορισμοι: 2,3,4
Μη Δεσμευτικοι Περιορισμοι : 1

Solution exercise 1 (b)

> ## Ασκηση 1 (β)
>
> (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. (Σημ. Οι υπολογισμοί θα πρέπει να γίνουν αναλυτικά)

Εισάγουμε διαταραχη δ στη βασική μεταβλητή x_2 :

$$c_B + \delta e_k$$

$$max\ 5x1 + (3+\delta)x2 + x3 + 4x4$$

Ψαχνω το διαστημα ανοχης για το δ για να παραμεινει βελτιστη η λυση που εχουμε βρει.

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

$$B^{-1} = \begin{pmatrix} 0 & \frac{4}{15} & \frac{2}{5} & \frac{-1}{15} \\ 0 & \frac{1}{5} & \frac{-1}{5} & \frac{1}{5} \\ 0 & \frac{-1}{15} & \frac{2}{5} & \frac{4}{15} \\ 1 & \frac{1}{3} & 0 & \frac{-4}{3} \end{pmatrix}$$

Πινακας N μη βασικων μεταβλητων :

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

$$c_B^T = (3 \quad 1 \quad 4 \quad 0)$$

$$c_N^T = (5 \quad 0 \quad 0 \quad 0)$$

Για να παραεινει ιδια η βελτιστη λυση παρα τη διαταραχη του συντελεστη της x_2 θα πρεπει να ισχυει :

$$(c_N^T (c_B + \delta e_k)^T B^{-1} N)_\alpha \leq 0 \quad , \alpha \in \{1, \dots, n\} \implies$$

Η βελτιση βαση εχει τις μεταβλητες x_2, x_3, x_4, s_1

$$(c_B + \delta e_k)^T = \begin{pmatrix} 3 \\ 1 \\ 4 \\ 0 \end{pmatrix}$$

$$(5 \quad 0 \quad 0 \quad 0) - (3+\delta \quad 1 \quad 4 \quad 0) \begin{pmatrix} 0 & \frac{4}{15} & \frac{2}{5} & \frac{-1}{15} \\ 0 & \frac{1}{5} & \frac{-1}{5} & \frac{1}{5} \\ 0 & \frac{-1}{15} & \frac{2}{5} & \frac{4}{15} \\ 1 & \frac{1}{3} & 0 & \frac{-4}{3} \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 & 0 \\ 2 & 1 & 0 & 0 \\ 3 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \leq 0 \implies$$

$$(5 \quad 0 \quad 0 \quad 0) - (3+\delta \quad 1 \quad 4 \quad 0) \begin{pmatrix} \frac{26}{15} & \frac{4}{15} & \frac{2}{5} & \frac{-1}{15} \\ \frac{-1}{5} & \frac{1}{5} & \frac{-1}{5} & \frac{1}{5} \\ \frac{16}{15} & \frac{-1}{15} & \frac{2}{5} & \frac{4}{15} \\ \frac{5}{3} & \frac{1}{3} & 0 & \frac{-4}{3} \end{pmatrix} \leq 0 \implies$$

$$(5 \quad 0 \quad 0 \quad 0) - \left(\frac{139+26\delta}{15} \quad \frac{11+4\delta}{15} \quad \frac{13+2\delta}{5} \quad \frac{16-\delta}{15}\right) \leq 0 \implies$$

$$\left(\frac{-64-26\delta}{15} \quad \frac{-11-4\delta}{15} \quad \frac{-13-2\delta}{5} \quad \frac{-16+\delta}{15}\right) \leq 0 \implies$$

$$-64 - 26\delta \leq 0 \implies \delta \geq -\frac{32}{13}$$
$$-11 - 4\delta \leq 0 \implies \delta \geq -\frac{11}{4}$$
$$-13 - 2\delta \leq 0 \implies \delta \geq -\frac{13}{2}$$
$$-16 + \delta \leq 0 \implies \delta \leq 16$$

Το δ ανηκει στο διάστημα

$$\delta \in \left(-\frac{32}{13}, 16\right)$$

Απο τις μη βασικες μεταβλητες επιλεγω την x_1, και θα πρεπει να ισχυει :

$$(c_N^T + \delta e_k^T - C_B^T B^{-1} N \leq 0 \implies$$

$$(5+\delta \quad 0 \quad 0 \quad 0) - (3 \quad 1 \quad 4 \quad 0) \begin{pmatrix} \frac{26}{15} & \frac{4}{15} & \frac{2}{5} & \frac{-1}{15} \\ \frac{-1}{5} & \frac{1}{5} & \frac{-1}{5} & \frac{1}{5} \\ \frac{16}{15} & \frac{-1}{15} & \frac{2}{5} & \frac{4}{15} \\ \frac{5}{3} & \frac{1}{3} & 0 & \frac{-4}{3} \end{pmatrix} \leq 0 \implies$$

$$(5+\delta \quad 0 \quad 0 \quad 0) - \begin{pmatrix} \frac{139}{15} & \frac{11}{15} & \frac{13}{5} & \frac{16}{15} \end{pmatrix} \leq 0 \implies$$

$$5 + \delta - \frac{139}{15} \leq 0 \leftrightarrow \delta \leq \frac{64}{15}$$

Επομένως το διαστημα ανοχης για την μεταβολη του συντελεστη της x_1 ειναι το :

$$\delta \leq \frac{64}{15}$$

Solution Exercise 1 (c)

Ασκηση 1 (γ)

(γ) Επιλέξτε έναν δεσμευτικό και έναν μη δεσμευτικό περιορισμό (ή περιορισμό προσήμου) 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. (Σημ. Οι υπολογισμοί θα πρέπει να γίνουν αναλυτικά)

Διαλεγοντας τον δεσμευτικο περιορισμο (4)

Το διαστημα ανοχης βρισκεται απο την σχεση :

$$B^{-1}b + \delta B^{-1} e_i \geq 0$$

Αρα για τον περιορισμο 4 :

$$B^{-1}b + \delta B^{-1} e_4 \geq 0$$

$$b = \begin{pmatrix} 10 \\ 6 \\ 10 \\ 7 \end{pmatrix}$$

$$B^{-1}e_4 = \begin{pmatrix} \frac{-1}{15} \\ \frac{1}{5} \\ \frac{4}{15} \\ \frac{-4}{3} \end{pmatrix}$$

$$B^{-1}b = \begin{pmatrix} \frac{77}{15} \\ \frac{3}{5} \\ \frac{82}{15} \\ \frac{8}{3} \end{pmatrix}$$

$$B^{-1}b - \delta B^{-1} e_4 = \begin{pmatrix} \frac{77}{15} \\ \frac{3}{5} \\ \frac{82}{15} \\ \frac{8}{3} \end{pmatrix} + \delta \begin{pmatrix} \frac{-1}{15} \\ \frac{1}{5} \\ \frac{4}{15} \\ \frac{-4}{3} \end{pmatrix} \geq 0 \implies$$

$$\begin{pmatrix} \frac{77}{15} + \delta\frac{-1}{15} \\ \frac{3}{5} + \delta\frac{1}{5} \\ \frac{82}{15} + \delta\frac{4}{15} \\ \frac{8}{3} + \delta\frac{-4}{3} \end{pmatrix} \geq 0 \implies$$

$$\begin{matrix} \frac{77}{15} + \delta\frac{-1}{15} \geq 0 \\ \frac{3}{5} + \delta\frac{1}{5} \geq 0 \\ \frac{82}{15} + \delta\frac{4}{15} \geq 0 \\ \frac{8}{3} + \delta\frac{-4}{3} \geq 0 \end{matrix} \implies \begin{matrix} 77 - \delta \geq 0 \\ 3 + \delta \geq 0 \\ 82 + 4\delta \geq 0 \\ 8 - 4\delta \geq 0 \end{matrix} \implies \begin{matrix} \delta \leq 77 \\ \delta \geq -3 \\ \delta \geq -10.5 \\ \delta \leq 2 \end{matrix} \implies \delta \in [-3, 2]$$

Καταληγουμε στο διαστημα ανοχης

$$\delta \in [-3, 2]$$

Για τον δεσμευτικο περιορισμο 4

Διαλεγουμε τον μη δεσμευτικο περιορισμο (1)

Θα πρεπει να ισχυει :

$$B^{-1}b - B^{-1}N\delta e_i \geq 0$$

$$B^{-1}Ne_1 = \begin{pmatrix} \frac{26}{15} \\ \frac{-1}{5} \\ \frac{16}{15} \\ \frac{5}{3} \end{pmatrix}$$

$$B^{-1}b - B^{-1}N\delta e_i \geq 0 \implies \begin{pmatrix} \frac{77}{15} \\ \frac{3}{5} \\ \frac{82}{15} \\ \frac{8}{3} \end{pmatrix} - \begin{pmatrix} \frac{26}{15} \\ \frac{-1}{5} \\ \frac{16}{15} \\ \frac{5}{3} \end{pmatrix} \delta \geq 0 \implies$$

$$\begin{pmatrix} \frac{77}{15} - \delta\frac{26}{15} \\ \frac{3}{5} - \delta\frac{-1}{5} \\ \frac{82}{15} - \delta\frac{16}{15} \\ \frac{8}{3} - \delta\frac{5}{3} \end{pmatrix} \geq 0 \implies$$

$$\begin{matrix} \frac{77}{15} - \delta\frac{26}{15} \geq 0 \\ \frac{3}{5} - \delta\frac{-1}{5} \geq 0 \\ \frac{82}{15} - \delta\frac{16}{15} \geq 0 \\ \frac{8}{3} - \delta\frac{5}{3} \geq 0 \end{matrix} \implies \begin{matrix} 77 - 26\delta \geq 0 \\ 3 + \delta \geq 0 \\ 82 - 16\delta \geq 0 \\ 8 - 5\delta \geq 0 \end{matrix} \implies \begin{matrix} \delta \leq \frac{77}{26} \\ \delta \geq -3 \\ \delta \leq \frac{41}{13} \\ \delta \leq \frac{8}{5} \end{matrix} \implies \delta \in \left[-3, \frac{8}{5}\right]$$

Python File

Exercise_01.py

```python
import sys
from pathlib import Path
import numpy as np

sys.path.append(str(Path(__file__).parents[3]))
from Tools.pulp_solver import PulpSolver


class Exerc01(PulpSolver):
    def __init__(self, *args, **kwargs) -> None:
        super().__init__(*args, **kwargs)

    def exerc_01_a(self):
        self.b = [
            [1, -2, 2, 3],
            [2, 2, 2, -1],
            [3, 1, -1, 1],
            [0, -1, 2, 2],
        ]
        self.c = [10, 6, 10, 7]

        # can be written better
        self.inequality = ["<=", "<=", "<=", "<="]
        self.obj = [5, 3, 1, 4]
        self.sense = "max"
        self.create_components()
        self.presenting_results()

    def exercise_01_b_1(self):
        """
        Description:
            This method solves a matrix multiplication
        """
        from scipy.optimize import linprog
        import fractions

        g = 1
        cn = [5, 0, 0, 0]
        cb = [3, 1, 4, 0]

        B = [
            [-2, 2, 3, 1],
            [2, 2, -1, 0],
            [1, -1, 1, 0],
            [-1, 2, 2, 0],
        ]

        N = [
            [1, 0, 0, 0],
            [2, 1, 0, 0],
            [3, 0, 1, 0],
            [0, 0, 0, 1],
        ]

        B = np.array(B)
        N = np.array(N)

        np.set_printoptions(
            formatter={"all": lambda x: str(fractions.Fraction(x).limit_denominator())}
        )

        B_inv = np.linalg.inv(B)
        B_inv_N = np.dot(B_inv, N)

        print(f"\n{'-'*30}\nB_inv_N:\n{'-'*30}")
        print(B_inv_N)

        # Add a variable g to the cb = [3 + g, 1, 4, 0] and multiply it with the B_inv_N
        # to get the new cn

        cn = np.array(cn)
        cb = np.array(cb)

        B_inv_N_cb = np.dot(cb, B_inv_N)
        print(f"\n{'-'*30}\nB_inv_N_cb:\n{'-'*30}")
```

```
        print(B_inv_N_cb)

        cn = cn - B_inv_N_cb

        print(f"\n{'-'*30}\ncn:\n{'-'*30}")
        print(cn)

        # Solve the linear programming problem

    def exercise_01_b_2(self):
        disturbances = [d / 13 for d in range(-32, 16 * 13)]
        # list(map(lambda x:x/13, range(-32, 16*13))

        for d in disturbances:
            print(f"\n\n{'-'*30}\nDisturbance: {d}\n{'-'*30}")
            self.add_disturbance(d, 0)
            self.presenting_results()
            print(f"\n{'-'*30}\n{'-'*30}\n\n")

    def exercise_01_b_3(self):
        disturbances = [d / 13 for d in range(-32, 16 * 13)]
        # list(map(lambda x:x/13, range(-32, 16*13))

        for d in disturbances:
            print(f"\n\n{'-'*30}\nDisturbance: {d}\n{'-'*30}")
            self.add_disturbance(d, 0)
            self.presenting_results()
            print(f"\n{'-'*30}\n{'-'*30}\n\n")

    def add_disturbance(self, d, index=0):
        self.b = [
            [1, -2, 2, 3],
            [2, 2, 2, -1],
            [3, 1, -1, 1],
            [0, -1, 2, 2],
        ]
        self.c = [10, 6, 10, 7]

        # can be written better
        self.inequality = ["<=", "<=", "<=", "<="]
        self.obj = [
            item + d if i == index else item for i, item in enumerate([5, 3, 1, 4])
        ]
        self.sense = "max"
        self.create_components()

    def main(self):
        self.exerc_01_a()
        self.exercise_01_b_1()
        self.exercise_01_b_2()


if __name__ == "__main__":
    exerc = Exerc01(name="Exercise1a")
    exerc.main()
```

pulp_solver.py

```
import pulp


class PulpSolver:
    def __init__(self, *args, **kwargs) -> None:
        """
        Description:
        ------------
        A class that creates a linear programming model using the pulp library.

        Parameters:
        -----------
        *args: int
            The number of variables to be created.
        **kwargs: dict
            name: str
                The name of the model.
            sense: max or min
                The sense of the model. Default is None.
            b: list
```

```
            The b vector of the model. Default is [].
        c: list
            The c vector of the model. Default is [].
        obj: list
            The objective function of the model. Default is [].
        inequality: list
            The inequality of the model. Default is [].
    """
    self.name = kwargs.get("name", "PulpSolver")
    self.sense = kwargs.get("sense", None)

    self.b = kwargs.get("b", [])
    self.c = kwargs.get("c", [])
    self.obj = kwargs.get("obj", [])
    self.inequality = kwargs.get("inequality", [])

def sensing(self):
    if self.sense == "max":
        self.sense = pulp.LpMaximize
    elif self.sense == "min":
        self.sense = pulp.LpMinimize
    elif self.sense is None:
        self.sense = pulp.LpMaximize
    else:
        raise ValueError("Sense must be either 'max' or 'min'.")

def create_components(self):
    print(self.b, self.c, self.obj, self.inequality)
    self.sensing()

    self.model = pulp.LpProblem(self.name, sense=self.sense)
    self.build_model()
    self.solver = pulp.GLPK_CMD()

    self.status = self.model.solve(self.solver)

def build_model(self):
    """
    Description:
    """

    # Takes the number of variables and creates them.
    self.x = [
        pulp.LpVariable(name=f"x{i}", lowBound=0)
        for i in range(1, len(self.b[0]) + 1)
    ]

    for i, row in enumerate(self.b):

        self.model += (
            sum([self.x[j] * row[j] for j in range(len(row))]) <= self.c[i]
        )

    self.model += sum([self.x[i] * self.obj[i] for i in range(len(self.obj))])

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

    for name, constraint in self.model.constraints.items():
        print(f"{name}: {constraint.value()}")
```

Solution 5

Ασκηση 5

Θεωρήστε το πρόβλημα του σακιδίου:

$$\begin{aligned} max \quad & z = 23x1 + 17x2 + 30x3 + 14x4 + 9x5 \\ & 6x1 + 5x2 + 10x3 + 7x4 + 5x5 \quad \leq 14 \\ & xi \in 0, 1 i = 1, 2, 3, 4, 5 \end{aligned}$$

Ασκηση 5 (α)

(α) Εφαρμόστε τον αλγόριθμο Branch & Bound για την επίλυσή του. Φροντίστε η επιλογή του επόμενου κόμβου για διακλάδωση να γίνεται κάθε φορά με κριτήριο την καλύτερη τιμή της αντικειμενικής συνάρτησης (Jumbtracking or Best First).

Ασκηση 5 (β)

(β) Για το παραπάνω πρόβλημα υπάρχει δυνατότητα να συσφίξουμε τον βασικό περιορισμό του προβλήματος ώστε να διευκολυνθεί η αλγοριθμική διαδικασία εύρεσης της βέλτιστης τιμής;

Ασκηση 5 (γ)

(γ) Χρησιμοποιώντας τον περιορισμό του παραπάνω προβλήματος προτείνετε επίπεδα αποκοπής από τρεις διαφορετικές ελάχιστες καλύψεις του. (Σημ. Υπάρχουν περισσότερες από τρεις). Πως διαμορφώνεται το δένδρο της Branch & Bound μετά την εισαγωγή
των επιπέδων αποκοπής;

Solution 5 a

Ασκηση 5 (α)

(α) Εφαρμόστε τον αλγόριθμο Branch & Bound για την επίλυσή του. Φροντίστε η επιλογή του επόμενου κόμβου για διακλάδωση να γίνεται κάθε φορά με κριτήριο την καλύτερη τιμή της αντικειμενικής συνάρτησης (Jumbtracking or Best First).

$$\begin{aligned} max \quad & z = 23x1 + 17x2 + 30x3 + 14x4 + 9x5 \\ & 6x1 + 5x2 + 10x3 + 7x4 + 5x5 \quad \leq 14 \\ & xi \in 0, 1 i = 1, 2, 3, 4, 5 \end{aligned}$$

Πρώτη λυση ειναι το x:$(1, 1, 0.3, 0, 0)$ με τιμη αντικειμενικης συναρτησης $z = 23 + 17 + 30 * 0.3 = 49$.

Θα δημιουργησω δυο κλαδους για το x_3: $x_3 = 1$ και $x_3 = 0$. Προσθετω τους κλαδους αυτους στην λιστα με τα nodes και ελεγχω τις τιμες των μεταβλητων και την λυση τους.

Επαναλαμβανω την διαδικασια και περναω απο ολα τα nodes μεχρι να εξαντληθουν οι επιλογες .

```
Final solution: [1. 1. 0. 0. 0.], value: 40.0
```

Program Results :

```
Initial best fun: 49.0, Initial best x: [1.  1.  0.3 0.  0. ]
--------------------------------------------------
Previous solution: None, value: None
Current solution [1.  1.  0.3 0.  0. ], value: 49.0
--------------------------------------------------

--------------------------------------------------
Previous solution: [1.  1.  0.3 0.  0. ], value: 49.0
Current solution [0.66666667 0.         1.         0.         0.        ], value: 45.33333333333333
--------------------------------------------------

--------------------------------------------------
Previous solution: [0.66666667 0.         1.         0.         0.        ], value: 45.33333333333333
Current solution [0.  0.8 1.  0.  0. ], value: 43.6
--------------------------------------------------

--------------------------------------------------
Previous solution: [0.  0.8 1.  0.  0. ], value: 43.6
Current solution [0.         0.         1.         0.57142857 0.        ], value: 38.0
--------------------------------------------------

--------------------------------------------------
Previous solution: [0.         0.         1.         0.57142857 0.        ], value: 38.0
Current solution [0.  0.  1.  0.  0.8], value: 37.2
--------------------------------------------------

--------------------------------------------------
Previous solution: [0.  0.  1.  0.  0.8], value: 37.2
Current solution [0. 0. 1. 0. 0.], value: 30.0
New best solution
--------------------------------------------------

--------------------------------------------------
Previous solution: [0. 0. 1. 0. 0.], value: 30.0
Current solution [1.         1.         0.         0.42857143 0.        ], value: 46.0
--------------------------------------------------

--------------------------------------------------
Previous solution: [1.         1.         0.         0.42857143 0.        ], value: 46.0
Current solution [1.  0.2 0.  1.  0. ], value: 40.4
--------------------------------------------------

--------------------------------------------------
Previous solution: [1.  0.2 0.  1.  0. ], value: 40.4
Current solution [0.33333333 1.         0.         1.         0.        ], value: 38.666666666666664
--------------------------------------------------

--------------------------------------------------
Previous solution: [0.33333333 1.         0.         1.         0.        ], value: 38.666666666666664
Current solution [0.  1.  0.  1.  0.4], value: 34.6
--------------------------------------------------

--------------------------------------------------
Previous solution: [0.  1.  0.  1.  0.4], value: 34.6
Current solution [0. 1. 0. 1. 0.], value: 31.0
New best solution
--------------------------------------------------

--------------------------------------------------
Previous solution: [0. 1. 0. 1. 0.], value: 31.0
Current solution [1.  0.  0.  1.  0.2], value: 38.8
--------------------------------------------------

--------------------------------------------------
Previous solution: [1.  0.  0.  1.  0.2], value: 38.8
Current solution [1. 0. 0. 1. 0.], value: 37.0
New best solution
```

```
--------------------------------------------------

--------------------------------------------------
Previous solution: [1. 0. 0. 1. 0.], value: 37.0
Current solution [1.  1.  0.  0.  0.6], value: 45.4
--------------------------------------------------

--------------------------------------------------
Previous solution: [1.  1.  0.  0.  0.6], value: 45.4
Current solution [1. 1. 0. 0. 0.], value: 40.0
New best solution
--------------------------------------------------

--------------------------------------------------
Previous solution: [1. 1. 0. 0. 0.], value: 40.0
Current solution [1.  0.6 0.  0.  1. ], value: 42.2
--------------------------------------------------

--------------------------------------------------
Previous solution: [1.  0.6 0.  0.  1. ], value: 42.2
Current solution [0.66666667 1.         0.         0.         1.        ], value: 41.33333333333333
--------------------------------------------------

Final solution: [1. 1. 0. 0. 0.], value: 40.0
```

Code

Exercise_05.py

```python
import sys
from pathlib import Path
import numpy as np

sys.path.append(str(Path(__file__).parents[3]))
from Tools.knapsack import Knapsack


class Exerc05(Knapsack):
    def __init__(self, *args, **kwargs) -> None:
        super().__init__(*args, **kwargs)

    def exerc_05_a(self):
        self.c = np.array([23, 17, 30, 14, 9])
        self.A = np.array([[6, 5, 10, 7, 5]])
        self.b = np.array([14])
        self.bounds = [(0, 1) for _ in range(len(self.c))]

        self.main()


if __name__ == "__main__":
    exerc = Exerc05(name="Exercise5a")
    exerc.exerc_05_a()
```

knapsack.py

```python
from scipy.optimize import linprog
import numpy as np


class Knapsack:
    def __init__(self, *args, **kwargs):
        """
        Description: Initialization of the class

        Args:
            A (np_array, optional) : Defaults to None
            b (np_array, optional) : Defaults to None
            c (np_array, optional) : Defaults to None
            bounds (list, optional): Defaults to None
        """
        self.A = kwargs.get("A", None)
        self.b = kwargs.get("b", None)
        self.c = kwargs.get("c", None)
        self.bounds = kwargs.get("bounds", None)
        if self.c and self.bounds is None:
            self.bounds = [(0, 1) for _ in range(len(self.c))]
        # Initialize  final variables
        self.final_variables = None
        self.final_fun = -np.inf

    def solve(self, bounds):
        """Solves the knapsack problem

        Args:
            bounds (list): Bounds for the variables

        Args Used :
            self.c (np_array)
            self.A (np_array)
            self.b (np_array)

        Returns:
            function(float), x variables(list) : Returns the function value and the x variables
        """
        result = linprog(
            -self.c, A_ub=self.A, b_ub=self.b, bounds=bounds, method="highs"
        )
        if result.success:
            return result.fun, result.x
        return None, None

    def adding_nodes(self, bounds):
```

```
        """
        Description: Adds nodes to the list self.nodes according to the bounds given, if the value is greater than the final value

        Args:
            bounds (list): bounds for the variables
        """
        # Solve the problem for the specific bounds
        fun, variables = self.solve(bounds)
        # Check if the solution is valid
        if fun is not None and -fun > self.final_fun:
            # Append to the node list for checking
            self.nodes.append((fun, bounds, variables))

    def initial(self):
        """
        Description:
            Initial Solution of the branch and bounds problem
        """
        first_fun, first_variables = self.solve(self.bounds)

        if first_fun is None:
            print("No solution found")
            return

        # Inverting the sign to convert from minimization to maximization
        # self.best_value = -self.best_value
        print(f"Initial best fun: {-first_fun}, Initial best x: {first_variables}")

        # Creating the node list
        self.nodes = [(first_fun, self.bounds, first_variables)]

    def check_integer_solution(self, current_solution, current_bounds, current_value):
        for i, item in enumerate(current_solution):
            if item not in [0, 1]:

                # Create two new nodes
                new_bounds_zero = current_bounds.copy()
                new_bounds_one = current_bounds.copy()

                new_bounds_zero[i] = (0, 0)
                new_bounds_one[i] = (1, 1)

                self.adding_nodes(new_bounds_zero)
                self.adding_nodes(new_bounds_one)

                return False

        return True

    def main(self):
        # Find the initial solution
        self.initial()
        current_variables = None
        current_fun = None
        # Check all the solutions
        while self.nodes:
            # Choose the best node with sorting
            self.nodes.sort(key=lambda x: x[0])
            previous_variables = current_variables
            previous_fun = current_fun

            current_fun, current_bounds, current_variables = self.nodes.pop()
            current_fun = -current_fun

            current_integer_solution = self.check_integer_solution(
                current_variables, current_bounds, current_fun
            )

            print(
                f"{'-'*50}\nPrevious solution: {previous_variables}, value: {previous_fun}\nCurrent solution {current_variables},
value: {current_fun} "
            )

            if current_integer_solution and current_fun > self.final_fun:
                self.final_variables = current_variables
                self.final_fun = current_fun
                print("New best solution")

            print(f"{'-'*50}\n")

        print(f"Final solution: {self.final_variables}, value: {self.final_fun}")
```

```
if __name__ == "__main__":
    # Initial data
    c = np.array([23, 17, 30, 14, 9])
    A = np.array([[6, 5, 10, 7, 5]])
    b = np.array([14])
    bounds = [(0, 1) for _ in range(len(c))]

    # Initial solution
    knapsack = Knapsack(A=A, b=b, c=c, bounds=bounds)
    knapsack.main()
```