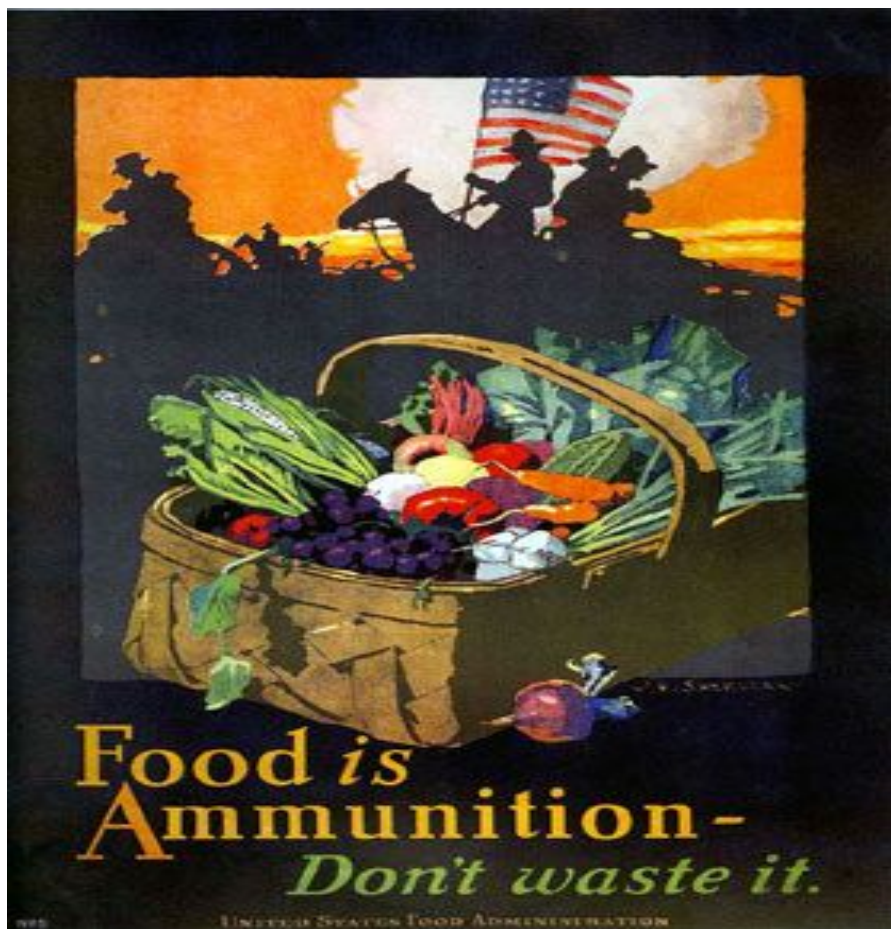


Βελτιστοποίηση

Optimization Problem: The Diet Problem

Εσκιόγλου Μαρία 3237

Βαρταλάμη Σοφία 3133



Τα περισσότερα πρακτικά προβλήματα μπορούν να μειωθούν σε προβλήματα μεγαλύτερης και μικρότερης σημασίας..και μόνο λύνοντας αυτά τα προβλήματα μπορούμε να ικανοποιήσουμε τις απαιτήσεις της πράξης η οποία πάντοτε ψάχνει το καλύτερο, το πιο βολικό.

P.L . Chebyshev (1821–1894)

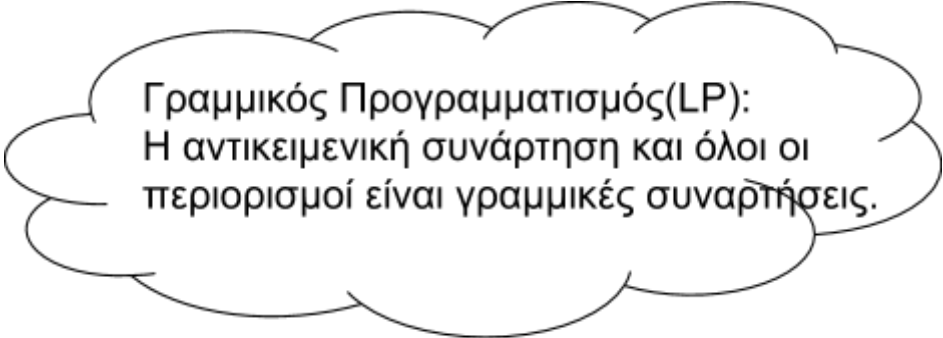
Εισαγωγή και σκοπός εργασίας:

Ως βελτιστοποίηση ορίζουμε τη διαδικασία επίλυσης ενός προβλήματος με τον καλύτερο δυνατό τρόπο. Με μαθηματική προσέγγιση, αυτό επιτυγχάνεται με την εύρεση μεγίστου, ελαχίστου. Μια κατηγορία προβλημάτων βελτιστοποίησης ανήκουν στο γραμμικό προγραμματισμό ή γραμμική βελτιστοποίηση, μέθοδος για την επίτευξη του καλύτερου αποτελέσματος (για παράδειγμα: μέγιστο κέρδος ή ελάχιστο κόστος) σε ένα μαθηματικό υπόδειγμα, του οποίου οι προϋποθέσεις (περιορισμοί) είναι ένα σύνολο γραμμικών σχέσεων των μεταβλητών του.

Αντικείμενο μελέτης μας είναι το Diet Problem, το οποίο χρονολογείται από τη δεκαετία του 1930 και του 1940 και παρουσιάζει γραμμικό (LP) μοντέλο για την επιλογή ενός συνόλου ελάχιστων δαπανών που θα ικανοποιούν ένα σύνολο καθημερινών διατροφικών απαιτήσεων. Πιο συγκεκριμένα, το πρόβλημα διατυπώνεται ως γραμμικό πρόγραμμα όπου στόχος του είναι η **ελαχιστοποίηση του κόστους και οι περιορισμοί είναι η ικανοποίηση των καθορισμένων διατροφικών απαιτήσεων.**

Μια σύντομη ιστορική αναδρομή:

Το Diet Problem είναι ένα από τα πρώτα προβλήματα βελτιστοποίησης που μελετήθηκαν τη δεκαετία του 1930 και του 1940. Το πρόβλημα προκλήθηκε από την επιθυμία του Στρατού να πετύχει την ελαχιστοποίηση του κόστους διατροφής στο πεδίο σε συνδυασμό με την παροχή μιας υγιεινής διατροφής. Από τους πρώτους ερευνητές, ο George Stigler διατύπωσε το πρόβλημα της διατροφής του Stigler. Η εικασία του για το κόστος μιας βέλτιστης διατροφής ήταν 39,93 \$ ετησίως. Αργότερα, ο Jack Laderman χρησιμοποίησε τη νέα **μέθοδο simplex** (δημοφιλής αλγόριθμος για γραμμικό προγραμματισμό) για την επίλυση του μοντέλου του Stigler. Χρειάστηκαν εννέα υπάλληλοι χρησιμοποιώντας χειροκίνητες αριθμομηχανές γραφείου 120 εργάσιμες ημέρες για την επίλυση της βέλτιστης λύσης 39,69\$. Η εικασία του Στίγκλερ έκλεισε μόλις 0,24 \$ ετησίως!



Γραμμικός Προγραμματισμός(LP):
Η αντικειμενική συνάρτηση και όλοι οι
περιορισμοί είναι γραμμικές συναρτήσεις.

Ανάλυση Προβλήματος:

Ως δεδομένα έχουμε ένα σύνολο φαγητών μαζί με τις πληροφορίες όσον αφορά τα θρεπτικά συστατικά και το κόστος ανά μερίδα για κάθε φαγητό. Στόχος του διατροφικού προβλήματος αποτελεί η επιλογή ενός συγκεκριμένου αριθμού μερίδων κάθε τροφής για αγορά και κατανάλωση με κριτήριο την ελαχιστοποίηση του κόστους αγοράς ενώ ταυτόχρονα θα ικανοποιούνται οι καθορισμένες διατροφικές απαιτήσεις. Οι διατροφικές απαιτήσεις μπορούν να εκφραστούν ως το μέγιστο και το ελάχιστο επιτρεπτό επίπεδο για κάθε θρεπτικό συστατικό. Οι υπόλοιποι περιορισμοί όπως ο ελάχιστος ή μέγιστος αριθμός μερίδων βελτιώνουν ποιοτικά το μενού.

Υποθέτουμε ότι το μενού μας αποτελείται από τα εξής τρόφιμα: μήλο(μερίδα: 1 αποφλοιωμένο μήλο), κοτόπουλο(μερίδα: 4 oz κοτόπουλο), καστανό ρύζι(μερίδα: 1 κούπα ρύζι), πατατάκια(μερίδα: 1 oz) και σπανάκι(μερίδα: 1 κούπα σπανάκι).

Στη συνέχεια , θα ασχοληθούμε με τους εξής περιορισμούς : την ποσότητα των θερμίδων, την περιεκτικότητα σε λιπαρά, νάτριο ,πρωτεΐνη, βιταμίνη Α και βιταμίνη C κάθε μερίδας για κάθε τρόφιμο που θα εμπεριέχεται στο παραπάνω καθορισμένο μενού.

Αυτοί οι περιορισμοί εξασφαλίζουν την καλύτερη ποιότητα του μοντέλου.

Για την κάθε μερίδα των τροφίμων που περιέχονται στο μενού έχουμε:

Αναλυτικός Πίνακας για τη θρεπτική αξία κάθε τροφίμου:

food	calories	fat	sodium	vitamin C	vitamin A	protein
apple	65	0	0	5.7	20.25	0.3
chicken	328	4	512	32	0	17.6
brown rice	216	0	10	0	0	5
chips	137	2	210	9.5	20.27	2.2
spinach	6.9	0	42	8.4	843.9	1

Μαθηματική Υλοποίηση:

Κατά μαθηματική προσέγγιση, το διατροφικό πρόβλημα αποτελεί πρόβλημα γραμμικού προγραμματισμού(-γραμμικής βελτιστοποίησης).

Έστω F το πλήθος των τροφίμων και N ο αριθμός των θρεπτικών ουσιών.

Στη συνέχεια ορίζουμε τις παραμέτρους:

A_{ij} : ποσότητα θρεπτικής ουσίας j στο τρόφιμο i , $\forall i \in F$ και $\forall j \in N$

C_i : κόστος κάθε μερίδας φαγητού i , $\forall i \in F$

N_{minj} : ελάχιστο απαιτούμενο επίπεδο θρεπτικών συστατικών j , $\forall j \in N$

N_{maxj} : μέγιστο επιτρεπόμενο επίπεδο θρεπτικών συστατικών j , $\forall j \in N$

Η μεταβλητή x_i δείχνει τον αριθμό των μερίδων φαγητού για αγορά και κατανάλωση

→ Ελαχιστοποίηση του συνολικού κόστους φαγητού: minimize $\sum_{i \in F} C_i x_i$

Περιορισμοί:

➤ Για κάθε θρεπτικό $j \in N$, τουλάχιστον να ικανοποιεί το ελάχιστο απαιτούμενο επίπεδο:

$$\sum_{i \in F} A_{ij} x_i \geq N_{minj}, \forall j \in N$$

➤ Για κάθε θρεπτικό $j \in N$, να μην υπερβαίνει το μέγιστο επιτρεπόμενο επίπεδο:

$$\sum_{i \in F} A_{ij} x_i \leq N_{maxj}, \forall j \in N$$

Ουσιαστικά, έχουμε το παρακάτω πρόβλημα γραμμικής βελτιστοποίησης:

$$\begin{aligned} \min \quad & \sum_{i \in F} C_i x_i \\ \text{subject to} \quad & \sum_{i \in F} A_{ij} x_i - N_{minj} \geq 0 \text{ and } N_{maxj} - \sum_{i \in F} A_{ij} x_i \geq 0, \forall j \in N \end{aligned}$$

Έστω ότι έχουμε τις τροφές:

x_1 (apple) with $C_1=1.5$

x_2 (chicken) with $C_2=2.75$

x_3 (brown ice) with $C_3=0.5$

x_4 (chips) with $C_4=1$

x_5 (spinach) with $C_5=2$

Με το x_i όπως αναφέραμε να υποδηλώνει τον αριθμό των μερίδων φαγητού

- **Αντικειμενική Συνάρτηση**

Ας ορίσουμε $f(x_1, x_2, x_3, x_4, x_5) = 1.5x_1 + 2.75x_2 + 0.5x_3 + x_4 + 2x_5$

- **Περιορισμοί**

Για τις θερμίδες έχουμε: $65x_1 + 328x_2 + 216x_3 + 137x_4 + 6.9x_5 = 2000$

$g1(x_1, x_2, x_3, x_4, x_5) = 65x_1 + 328x_2 + 216x_3 + 137x_4 + 6.9x_5 - 2000 = 0$

Για τα λιπαρά: $4x_2 + 2x_4 \leq 10$

$$g2(x_1, x_2, x_3, x_4, x_5) = -4x_2 - 2x_4 + 10 \geq 0$$

$$\text{Για το νάτριο: } 512x_2 + 10x_3 + 210x_4 + 24x_5 \leq 2200$$

$$g3(x_1, x_2, x_3, x_4, x_5) = -512x_2 - 10x_3 - 210x_4 - 24x_5 + 2200 \geq 0$$

$$\text{Για τη βιταμίνη C: } 5.7x_1 + 32x_2 + 9.5x_4 + 8.4x_5 \geq 100$$

$$g4(x_1, x_2, x_3, x_4, x_5) = 5.7x_1 + 32x_2 + 9.5x_4 + 8.4x_5 - 100 \geq 0$$

$$\text{Για τη βιταμίνη A: } 20.25x_1 + 20.27x_4 + 843.9x_5 \geq 700$$

$$g5(x_1, x_2, x_3, x_4, x_5) = 20.25x_1 + 20.27x_4 + 843.9x_5 - 700 \geq 0$$

$$\text{Για την πρωτεΐνη: } 0.3x_1 + 17.6x_2 + 5x_3 + 2.2x_4 + x_5 \geq 50$$

$$g6(x_1, x_2, x_3, x_4, x_5) = 0.3x_1 + 17.6x_2 + 5x_3 + 2.2x_4 + x_5 - 50 \geq 0$$

$$\text{με } x_1, x_2, x_3, x_4, x_5 \geq 1$$

Συνάρτηση Lagrange:

Εφαρμόζουμε τη **συνθήκη μέθοδο πολλαπλασιασστή Lagrange**

$$L(x_1, x_2, x_3, x_4, x_5, \lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5, \lambda_6) = 1.5x_1 + 2.75x_2 + 0.5x_3 + x_4 + 2x_5 - \lambda_1(65x_1 + 328x_2 + 216x_3 + 137x_4 + 6.9x_5 - 2000) - \lambda_2(-4x_2 - 2x_4 + 10) - \lambda_3(-512x_2 - 10x_3 - 210x_4 - 24x_5 + 2200) - \lambda_4(5.7x_1 + 32x_2 + 9.5x_4 + 8.4x_5 - 100) - \lambda_5(20.25x_1 + 20.27x_4 + 843.9x_5 - 700) - \lambda_6(0.3x_1 + 17.6x_2 + 5x_3 + 2.2x_4 + x_5 - 50)$$

Ο υποψήφιος τοπικός βελτιστοποιητής θα πρέπει να ανήκει στο σύνολο Ω όπου ισχύουν οι παραπάνω ανισοτικοί περιορισμοί αλλά και οι αντίστοιχοι ισοτικοί.

Υπάρχουν 2 καταστάσεις στους ανισοτικούς περιορισμούς, να είναι ενεργοί ($g_i(x) = 0$) ή ανενεργοί ($g_i(x) > 0$).

Σημείωση: το θετικό πρόσημο των πολλαπλασιαστών Lagrange είναι κοινή χαρακτηριστική ιδιότητα των προβλημάτων βελτιστοποίησης με ανισοτικούς περιορισμούς. Όταν κάποιος περιορισμός είναι ενεργός πρέπει το αντίστοιχο λ να είναι θετικό.

Η L είναι παραγωγίσιμη:

$$L_{x1} = 1.5 - 65\lambda_1 - 5.7\lambda_4 - 20.25\lambda_5 - 0.3\lambda_6 = 0$$

$$L_{x2} = 2.75 - 328\lambda_1 + 4\lambda_2 + 512\lambda_3 - 32\lambda_4 - 17.6\lambda_6 = 0$$

$$L_{x3} = 0.5 - 216\lambda_1 + 10\lambda_3 - 5\lambda_6 = 0$$

$$L_{x4} = 1 - 137\lambda_1 + 2\lambda_2 + 210\lambda_3 - 9.5\lambda_4 - 20.27\lambda_5 - 2.2\lambda_6 = 0$$

$$L_{x5} = 2 - 6.9\lambda_1 + 24\lambda_3 - 8.4\lambda_4 - 843.9\lambda_5 - \lambda_6 = 0$$

KKT conditions:

$$1.5-65\lambda_1-5.7\lambda_4-20.25\lambda_5-0.3\lambda_6=0$$

$$2.75-328\lambda_1+4\lambda_2+512\lambda_3-32\lambda_4-17.6\lambda_6=0$$

$$0.5-216\lambda_1+10\lambda_3-5\lambda_6=0$$

$$1-137\lambda_1+2\lambda_2+210\lambda_3-9.5\lambda_4-20.27\lambda_5-2.2\lambda_6=0$$

$$2-6.9\lambda_1+24\lambda_3-8.4\lambda_4-843.9\lambda_5-\lambda_6=0$$

$$65x_1 + 328x_2 + 216x_3 + 137x_4 + 6.9x_5 - 2000 = 0$$

$$-4x_2 - 2x_4 + 10 \geq 0$$

$$-512x_2 - 10x_3 - 210x_4 - 24x_5 + 2200 \geq 0$$

$$5.7x_1 + 32x_2 + 9.5x_4 + 8.4x_5 - 100 \geq 0$$

$$20.25x_1 + 20.27x_4 + 843.9x_5 - 700 \geq 0$$

$$0.3x_1 + 17.6x_2 + 5x_3 + 2.2x_4 + x_5 - 50 \geq 0$$

$$\mu \in x_1, x_2, x_3, x_4, x_5 \geq 1$$

$$\lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5, \lambda_6 \geq 0$$

$$\lambda_1 (65x_1 + 328x_2 + 216x_3 + 137x_4 + 6.9x_5 - 2000) = 0$$

$$\lambda_2 (-4x_2 - 2x_4 + 10) = 0$$

$$\lambda_3 (-512x_2 - 10x_3 - 210x_4 - 24x_5 + 2200) = 0$$

$$\lambda_4 (5.7x_1 + 32x_2 + 9.5x_4 + 8.4x_5 - 100) = 0$$

$$\lambda_5 (20.25x_1 + 20.27x_4 + 843.9x_5 - 700) = 0$$

$$\lambda_6 (0.3x_1 + 17.6x_2 + 5x_3 + 2.2x_4 + x_5 - 50) = 0$$

Σημείωση: λύση του παραπάνω συστήματος μπορεί να γίνει με τη μέθοδος Gauss Jordan.

Από τις 6 τελευταίες σχέσεις διακρίνουμε περιπτώσεις ($= 2^{n=6} = 64$) και λύνουμε το σύστημα που προκύπτει. Εάν κάποια λύση δεν ικανοποιεί κάποιον περιορισμό απορρίπτεται.

Η λύση μας, \mathbf{x}^* , είναι αυτή που πληροί τις συνθήκες KKT.

Για την τιμή \mathbf{x}^* της f που βρήκαμε κατασκευάζουμε τον εσσιανό πίνακα

$$\nabla^2 L(x_1, x_2, x_3, x_4, x_5, \lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5, \lambda_6)$$

Αν $|\nabla^2 L(x_1, x_2, x_3, x_4, x_5, \lambda_1, \lambda_2, \lambda_3, \lambda_4, \lambda_5, \lambda_6)| > 0$ (θετικά ορισμένος στον εφαπτόμενο κώνο) τότε η f έχει στο σημείο \mathbf{x}^* δεσμευμένο τοπικό ελάχιστο.

❖ Ιδανικός αλγόριθμος για την επίλυση γραμμικών προβλημάτων είναι η μέθοδος **Simplex**:

Επειδή όμως έχουμε και περιορισμούς με ισότητες και με ανισότητες τύπου \geq θα πρέπει να χρησιμοποιηθούν τεχνητές μεταβλητές και έτσι θα χρησιμοποιήσουμε τη μέθοδο Big-M.

Χειρισμός ανισοτικών σχέσεων και η μετατροπή τους σε ισότητες:

- Για ανισότητες \geq : Προσθέτουμε μια μεταβλητή -Si που ονομάζεται **surplus variable** (μεταβλητή πλεονάσματος) και προσθέτουμε την **artificial variable** (τεχνητή μεταβλητή) Ai.

Μια τεχνητή μεταβλητή είναι μια μεταβλητή που εισάγεται σε κάθε μία εξίσωση που έχει μια μεταβλητή πλεονάσματος.

- Για ανισότητες \leq : Προσθέτουμε μια μεταβλητή Si που ονομάζεται **slack variable**
- Μ αντιπροσωπεύει έναν αριθμό μεγαλύτερο από οποιοδήποτε άπειρο αριθμό και ονομάζεται big M
- +M για τις τεχνητές μεταβλητές προκειμένου να αποφευχθεί μια τεχνητή μεταβλητή να γίνει μέρος της βέλτιστης λύση στο αρχικό πρόβλημα

$$\text{minimize } z = 1.5x_1 + 2.75x_2 + 0.5x_3 + x_4 + 2x_5 + MA1 + MA2 + MA3 + MA4$$

$$\text{subject to } 65x_1 + 328x_2 + 216x_3 + 137x_4 + 6.9x_5 + A1 = 2000$$

$$4x_2 + 2x_4 + S1 = 10$$

$$512x_2 + 10x_3 + 210x_4 + 24x_5 + S2 = 2200$$

$$5.7x_1 + 32x_2 + 9.5x_4 + 8.4x_5 - S3 + A2 = 100$$

$$20.25x_1 + 20.27x_4 + 843.9x_5 - S4 + A3 = 700$$

$$0.3x_1 + 17.6x_2 + 5x_3 + 2.2x_4 + x_5 - S5 + A4 = 50$$

Step A: initial table

Cj		1.5	2.75	0.5	1	2	0	0	0	0	0	M	M	M	M		
CB	basic variables	x_1	x_2	x_3	x_4	x_5	S1	S2	S3	S4	S5	A1	A2	A3	A4	Solu tion	Ratio
M	A1	65	328	216	137	6.9	0	0	0	0	0	1	0	0	0	2000	6.09
0	S1	0	4	0	2	0	1	0	0	0	0	0	0	0	0	10	2.5
0	S2	0	512	10	210	24	0	1	0	0	0	0	0	0	0	2200	4.29
M	A2	5.7	32	0	9.5	8.4	0	0	-1	0	0	0	1	0	0	100	3.12
M	A3	20.25	0	0	20.27	843.9	0	0	0	-1	0	0	0	1	0	700	

M	A4	0.3	17.6	5	2.2	1	0	0	0	0	-1	0	0	0	1	50	2.84
Zj		91.52M	377.6M	221M	168.97M	860.2M	0	0	-M	-M	-M	M	M	M	M		
Cj-Zj		1.5-91.52M	2.75-377.6M	0.5-221M	1-168.97M	2-860.2M	0	0	M	M	M	0	0	0	0		

$$\mu \epsilon Zj = \sum_{i=1}^6 XiCBi$$

Step B: selection of the entering variable (to the set of basic variables)

Αν το $Cj-Zj \geq 0$ η λύση είναι βέλτιστη

Επιλέγουμε το ελάχιστο $Cj-Zj$, στην περίπτωση μας το 2.75-377.6M και εστιάζουμε σε εκείνη τη στήλη.

Step C : selection of the leaving variable

Έπειτα υπολογίζουμε το RATIO που το καθένα είναι το αποτέλεσμα της διαίρεσης της λύσης στη γραμμή i διά του του αντίστοιχου στοιχείου στην στήλη j που επιλέξαμε και στην γραμμή που βρισκόμαστε i και επιλέγουμε το μικρότερο από αυτά επικεντρώνοντας σε αυτή τη γραμμή. Στην περίπτωση μας, αυτό σημαίνει ότι εστιάζουμε στην πορτοκαλί γραμμή και έτσι φεύγει η μεταβλητή S1.

Το κόκκινο τετραγωνάκι ονομάζεται pivot και η μεταβλητή που θα εισάγουμε θα είναι η x_2 με βάση τη στήλη που ήμασταν.

Step D : pivot

Έπειτα ξαναδημιουργείται ένας πίνακα μόνο που τώρα αντί για S1 θα έχουμε την x_2 .

Στη γραμμή του x_2 το 0 θα γίνει 2.75 και στη συνέχεια τα νέα στοιχεία θα είναι τα στοιχεία της προηγούμενης διά το pivot ενώ το RATIO θα ναι κενό.

Για τα υπόλοιπα δεδομένα θα τροποποιηθούν με βάση την παρακάτω φόρμουλα:

New Value= Old Value-(element on the line of pivot*element on the column of the pivot)/pivot

Μόλις συμπληρώσουμε τα στοιχεία αυτά, υπολογίζουμε τα Zj , $Cj-Zj$.

Εάν το $Cj-Zj$ είναι ≥ 0 τότε σταματάμε τη διαδικασία-αλγόριθμο και έχουμε βρει βέλτιστη λύση με τα x_1, x_2, x_3, x_4, x_5 να έχουν ως τιμές αυτές που αντιστοιχούν στη στήλη Solution αλλιώς επαναλαμβάνουμε τη διαδικασία.

Προγραμματιστικά:

Resources	1 (x)	2 (y)	Requirement
Vitamin A	2	1	8
Vitamin C	1	2	10
Cost	50	70	$Z=50x+70y$

Από το παραπάνω πρόβλημα προκύπτουν οι παρακάτω ανισοτικοί περιορισμοί:

$$2x+1y \geq 8 \quad (\text{Σχέση 1})$$

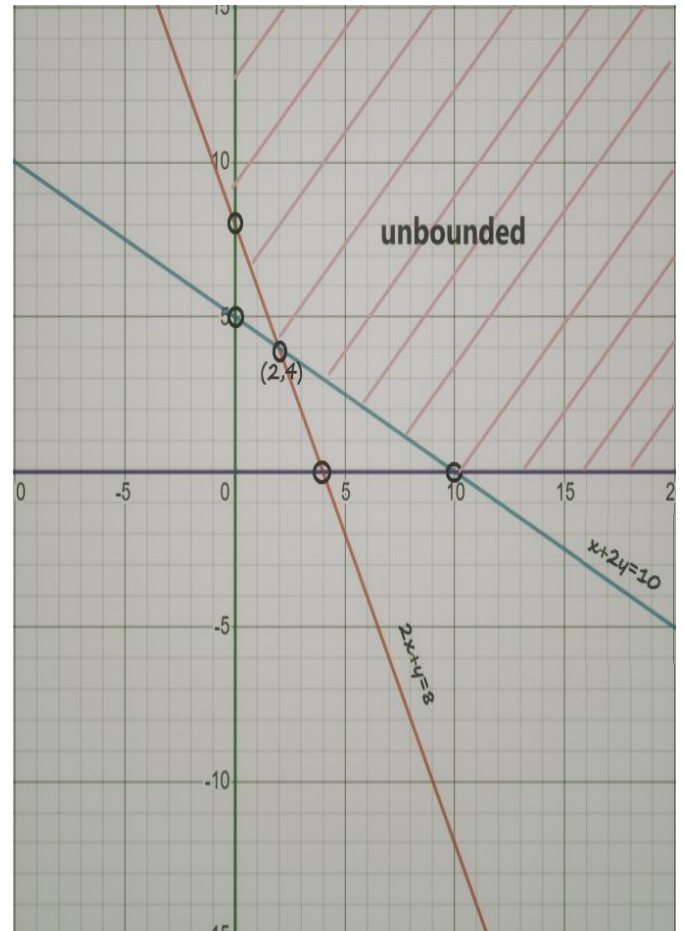
$$1x+2y \geq 10 \quad (\text{Σχέση 2})$$

$$x \geq 0 \quad (\text{Σχέση 3})$$

$$y \geq 0 \quad (\text{Σχέση 4})$$

Όπως φαίνεται στο διπλανό σχήμα, επιλέγουμε τα ακριανά σημεία:

Σημεία	$Z=50x+70y$
A(0,8)	560
B(2,4)	380
C(10,0)	500



Άρα το ελάχιστο κόστος που προκύπτει από το συγκεκριμένα δεδομένα είναι το 380.

Επεξήγηση Κώδικα:

Υπάρχουν πολλές βιβλιοθήκες στην Python για την επίλυση προβλημάτων βελτιστοποίησης. Στην συγκεκριμένη περίπτωση, χρησιμοποιήθηκε η PuLP, η οποία είναι ανοιχτού κώδικα και κατάλληλη για γραμμικό προγραμματισμό. Η PuLP μπορεί να εγκατασταθεί με τους παρακάτω τρόπους:

```
$ sudo pip install pulp # PuLP
$ sudo apt-get install glpk-utils # GLPK
$ sudo apt-get install coinor-cbc # CoinOR
```

Στην συνέχεια, εισάγουμε τα πάντα από την βιβλιοθήκη:

```
from pulp import *
```

Αρχικά, δημιουργούμε ένα πρόβλημα γραμμικού προγραμματισμού με την μέθοδο LpProblem της PuLP. Σκοπός της

```
prob = LpProblem("Minimizing Cost",LpMinimize)
```

Στη συνέχεια αρχικοποιούμε τις επιλογές που απαρτίζουν τα διαφορετικά στοιχεία που έχουμε στην διάθεση μας. Θέτουμε το lowBound=1, έτσι ώστε για κάθε φαγητό να υπολογιστεί τουλάχιστον μία μερίδα. Το lowBound είναι μία παράμετρος που μπορεί να μεταβληθεί ανάλογα με τις ανάγκες του εκάστοτε χρήστη. Για παράδειγμα, μεταβάλλοντας την τιμή της lowBound σε 3 για το σπανάκι, το αποτέλεσμα που προκύπτει θα έχει τουλάχιστον 3 μερίδες από το συγκεκριμένο προϊόν. Για τις ανάγκες του παραδείγματος, το μενού αποτελείται από τις εξής τροφές: μήλα, κοτόπουλα, ρύζι, πατάτες, σπανάκι. Για τις συγκεκριμένες τροφές βρίσκουμε τις αντίστοιχες τιμές από ορισμένα θρεπτικά συστατικά, όπως είναι οι θερμίδες, τα λίπη, το νάτριο, η βιταμίνη A, η βιταμίνη C και οι πρωτεΐνες. Αυτές οι τιμές αντικατοπτρίζουν τις πραγματικές και έχουν παρθεί από την ιστοσελίδα:

<https://nutritionfacts.org/>.

Έπειτα, ξεκινάμε την μοντελοποίηση του προβλήματος προσθέτοντας την objective function, χρησιμοποιώντας την μέθοδο lpSum. Πριν προχωρήσουμε στην δημιουργία των constraints, πολλαπλασιάζουμε την ποσότητα κάθε επιλογής με την αντίστοιχη τιμή σε ευρώ κάθε προϊόντος. Έτσι, επιλέχθηκαν τυχαία οι τιμές 1.5£ για μήλα, 2.75£ για το κοτόπουλο, 0.5£ για το ρύζι, 1£ για τις πατάτες και 2£ για το σπανάκι. Οι τιμές αυτές, όπως προαναφέρθηκε, είναι τυχαίες και αλλάζουν αναλόγως το πρόβλημα.

```
prob += 1.50 * choice1 + 2.75*choice2 + .5*choice3 + 1*choice4  
+ 2*choice5
```

Μετά την δημιουργία της αντικειμενικής συνάρτησης, χρησιμοποιούμε την lpSum μέθοδο από την βιβλιοθήκη PuLP και προσθέτουμε τους περιορισμούς για τα διάφορα θρεπτικά συστατικά. Στο παράδειγμα, που επιδεικνύεται, έχουμε θέσει τις θερμίδες να είναι ίσες με 2000, τα λίπη κάτω από 10, το νάτριο κάτω από 2200, οι βιταμίνες A και C να είναι άνω ή ίσα με τις τιμές 100 και 700 αντίστοιχα και τις πρωτεΐνες πάνω από τα 50.

```
prob += pulp.lpSum([calories[i]*choices[i] for i in range(len(choices))]) == 2000  
prob += pulp.lpSum([fat[i]*choices[i] for i in range(len(choices))]) <= 10  
prob += pulp.lpSum([sodium[i]*choices[i] for i in range(len(choices))]) <= 2200  
prob += pulp.lpSum([vitC[i]*choices[i] for i in range(len(choices))]) >= 100  
prob += pulp.lpSum([vitA[i]*choices[i] for i in range(len(choices))]) >= 700
```

```
prob += pulp.lpSum([protein[i]*choices[i] for i in range(len(choices))]) >= 50
```

Οι περιορισμοί αυτοί είναι ενδεικτικοί και αποτελούν τα άνω και κάτω όρια που θέτει ο χρήστης ανάλογα με τις ανάγκες της διατροφής του. Έτσι, για παράδειγμα, στο νάτριο, τιμές κάτω από 2200 είναι αποδεκτές, ενώ στις πρωτεΐνες οι τιμές πρέπει να είναι αυστηρά πάνω από 50. Ωστόσο, ο περιορισμός στις θερμίδες είναι ισοτικός, οπότε οι θερμίδες πρέπει να είναι υποχρεωτικά 2000 χωρίς να υπάρχουν αυξομειώσεις. Τέλος, μπορούμε να θέσουμε ταυτόχρονα άνω και κάτω όριο για οποιοδήποτε από τα θρεπτικά συστατικά θέλουμε όπως φαίνεται και στο παρακάτω κομμάτι κώδικα.

```
prob += lpSum([fat[f] *  
food_vars[f] for f in food_items])  
>= 20.0, "FatMinimum"
```

```
prob += lpSum([fat[f] * food_vars[f] for  
f in food_items]) <= 50.0, "FatMaximum"
```

Σε οποιοδήποτε σενάριο βελτιστοποίησης, το δύσκολο κομμάτι είναι η διατύπωση του προβλήματος σε ένα άρτια δομημένο τρόπο ώστε να είναι ευπαρουσίαστο σε αυτόν που πρόκειται να λύσει το πρόβλημα.

Η βιβλιοθήκη PuLP έχει αρκετές επιλογές για αλγόριθμους που λύνουν τέτοια προβλήματα (COIN_MP, Gurobi, CPLEX, κτλ). Για το συγκεκριμένο πρόβλημα δεν θα επιλέξουμε έναν συγκεκριμένο αλγόριθμο, αλλά θα αφήσουμε το πρόγραμμα να επιλέξει το δικό του προκαθορισμένο ανάλογα με την δομή του προβλήματος με την εντολή:

```
prob.solve()
```

Η PuLP δίνει την δυνατότητα εκτύπωσης της κατάστασης του της λύσης. Στην περίπτωση του παραδείγματος που έχουμε υλοποιήσει η κατάσταση θα είναι Optimal, δηλαδή η καλύτερη δυνατή. Σε περίπτωση όμως που το πρόβλημα δεν είναι σωστά διατυπωμένο ή δεν υπάρχουν επαρκή στοιχεία η λύση μπορεί να είναι unbounded ή infeasible.

```
# The status of the solution is printed to the screen  
print("Status:", LpStatus[prob.status])  
>> Status: Optimal
```

Μετά την επίλυση του προβλήματος, ασχοληθήκαμε με την έξοδο του προγράμματος στην κονσόλα. Οι πληροφορίες που θα εμφανιστούν θα είναι οι μερίδες από κάθε είδος τροφής, το συνολικό κόστος και οι συνολικές ποσότητες από τα θρεπτικά συστατικά όπως προκύπτουν μετά την επίλυση της αντικειμενικής συνάρτησης.

Για την εμφάνιση των μερίδων θα χρειαστούμε την τιμή των αντίστοιχων μεταβλητών των choices. Αυτό μπορεί να επιτευχθεί με δύο τρόπους:

- 1) `format(choice.value())`
- 2) `format(choice.varValue)`

Στην συνέχεια εκτυπώνουμε το συνολικό κόστος στρογγυλοποιημένο στα 5 δεκαδικά ψηφία. Ο υπολογισμός του κόστους πραγματοποιείται με την εντολή:

```
total_cost = pulp.value(prob.objective)
```

Το τελευταίο κομμάτι που θα εμφανίσουμε είναι οι συνολικές ποσότητες από τα θρεπτικά συστατικά σε σχέση με αυτά που έχουμε θέσει ως περιορισμούς. Όπως παρατηρούμε από τα αποτελέσματα υπακούν σε όλα τα constraints που έχουμε θέσει σε προηγούμενο βήμα. Σε ανισοτικούς περιορισμούς παρατηρούνται αυξομειώσεις ενώ οι ισοτικοί ικανοποιούνται πλήρως. Οι τιμές αυτές έχουν στρογγυλοποιηθεί ώστε να είναι ακέραιο με την εντολή `round()`. Αν αφαιρεθεί αυτή τροποποιούνται και εμφανίζονται οι πραγματικές τιμές των ποσοτήτων.

Παρακάτω εμφανίζονται screenshots από τις εξόδους του προγράμματος για συγκεκριμένες τιμές.

```
Servings of apples = 1.0  
Servings of chicken = 1.8625  
Servings of brown rice = 5.4  
Servings of chips = 1.0  
Servings of spinach = 3.0
```

```
Total Calories: 2000/2000  
Total Fat: 9/10  
Total Sodium: 1290/2200  
Total Vitamin C: 100/100  
Total Vitamin A: 2572/700  
Total Protein: 65/50
```

```
The total cost is $16.32187 for us to get the required daily nutrients
```

Status: Optimal