ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ
ΠΡΟΓΡΑΜΜΑ ΜΕΤΑΠΤΥΧΙΑΚΩΝ ΣΠΟΥΔΩΝ
ΤΜΗΜΑΤΟΣ ΕΦΑΡΜΟΣΜΕΝΗΣ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΥΛΟΠΟΙΗΣΗ ΤΕΧΝΙΚΩΝ ΚΛΙΜΑΚΩΣΗΣ (SCALING TECHNIQUES)

Διπλωματική Εργασία

του

Κατσαρού Μάριου

Θεσσαλονίκη, Οκτώβριος 2020

ΥΛΟΠΟΙΗΣΗ ΤΕΧΝΙΚΩΝ ΚΛΙΜΑΚΩΣΗΣ (SCALING TECHNIQUES)

Μάριος Κατσαρός

Πτυχίο Εφαρμοσμένης Πληροφορικής, Πανεπιστήμιο Μακεδονίας, 2016

Διπλωματική Εργασία

υποβαλλόμενη για τη μερική εκπλήρωση των απαιτήσεων του

ΜΕΤΑΠΤΥΧΙΑΚΟΥ ΤΙΤΛΟΥ ΣΠΟΥΔΩΝ ΣΤΗΝ ΕΦΑΡΜΟΣΜΕΝΗ ΠΛΗΡΟΦΟΡΙΚΗ

Επιβλέπων Καθηγητής
Σαμαράς Νικόλαος

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την ηη/μμ/εεεε

Σαμαράς Νικόλαος	Χρήστου-Βαρσακέλης Δημήτριος	Σιφαλέρας Άγγελος
.....	.....	.....

Μάριος Κατσαρός

.....

Περίληψη

Η διπλωματική εργασία πραγματεύεται την υλοποίηση των τεχνικών κλιμάκωσης γραμμικών προβλημάτων σε CPU, με την βοήθεια της γλώσσας προγραμματισμού python, για την αποτελεσματικότερη επίλυση γενικών γραμμικών προβλημάτων από τους λύτες. Στην διπλωματική εργασία παρουσιάζονται αναλυτικά 11 τεχνικές κλιμάκωσης, 1)arithmetic mean, 2)de Buchet για την περίπτωση p=1, 3)de Buchet για την περίπτωση p=2, 4)de Buchet για την περίπτωση p=∞, 5) entropy, 6) equilibration , 7) geometric mean, 8) IBM-MPSX, 9) Lp-Norm για την περίπτωση p=1, 10)Lp-Norm για την περίπτωση p=2, 11) Lp-Norm για την περίπτωση p=∞, με ένα αναλυτικό λυμένο αριθμητικό παράδειγμα, ενώ παρουσιάζεται και ο αλγόριθμός τους στην γλώσσα προγραμματισμού Python. Πραγματοποιείται μια εκτενής υπολογιστική μελέτη μεταξύ των τεχνικών κλιμάκωσης προκειμένου να προσδιοριστεί ποια απο τις τεχνικές είναι η γρηγορότερη βάσει των χρόνων που προέκυψαν από την κλιμάκωση γραμμικών προβλημάτων με την Python. Η υπολογιστική μελέτη γίνεται ανάμεσα σε 136 γνωστά γραμμικά προβλήματα. Τέλος, παρατίθενται όλοι οι υπολογιστικοί χρόνοι και τα συμπεράσματα που προέκυψαν απο την υπολογιστική μελέτη, καθώς παρέχονται και προτάσεις για μελλοντική επέκταση.

Περιεχόμενα

1 Εισαγωγή

1.1 Τι είναι επιχειρησιακή έρευνα;

Επιχειρησιακή έρευνα ονομάζεται η επιστήμη που επικεντρώνεται στην αποτελεσματική χρήση της τεχνολογίας για την λήψη σωστών αποφάσεων σε προβλήματα που χρειάζεται η κατανομή διαθέσιμων πόρων. Σύμφωνα με την επιχειρησιακή έρευνα, γίνεται ανάλυση του προβλήματος, προσπάθεια εύρεσης βέλτιστης λύσης, ανάπτυξη μοντέλου επίλυσης του προβλήματος καθώς και έλεγχος και εξέταση των αποτελεσμάτων που βρέθηκαν(Δαρζέντας, 1999). Η επιχειρησιακή έρευνα εφαρμόζεται σε μεγάλο εύρος προβλημάτων. Μια απο τις μεθόδους υλοποίησης της επιχειρησιακής έρευνας είναι και ο γραμμικός προγραμματισμός.

1.2 Γραμμικός Προγραμματισμός (Linear Programming)

Ο γραμμικός προγραμματισμός ή γραμμική βελτιστοποίηση είναι μία ευρέως χρησιμοποιούμενη τεχνική μαθηματικής μοντελοποίησης για τον καθορισμό της βέλτιστης κατανομής των πόρων ανάμεσα σε ανταγωνιστικές απαιτήσεις (Κώστογλου Β., 2015). Είναι μία μαθηματική τεχνική που βοηθά στον υπολογισμό της μέγιστης ή της ελάχιστης τιμής της αντικειμενικής συνάρτησης βάσει των περιορισμών της (The Editors of Encyclopedia Britannica, 1999). Η λέξη προγραμματισμός ορίζει την αναζήτηση της καλύτερης δυνατής επιλογής ανάμεσα σε διαφορετικές εκδοχές. Ο γραμμικός προγραμματισμός εγγυάται την άριστη λύση του διατυπωμένου μοντέλου (Κώστογλου Β., 2015).

Τα προβλήματα γραμμικού προγραμματισμού εκφράζονται στην κανονική μορφή ως εξής:

$$\max/\min z = c_1x_1 + c_2x_2 + \ldots\ldots + c_nx_n$$

Με περιορισμούς

$$a_{11}x_1 + a_{12}x_2 + \ldots..+ a_{1n}x_n \leq b_1$$

$$a_{21}x_1 + a_{22}x_2 + \ldots...+ a_{2n}x_n \leq b_2$$

. . . . . .

$$a_{m1}x_1 + a_{m2}x_2 + \ldots\ldots + a_{mn}x_n \leq b_m$$

$$x_j \geq 0, \ (j = 1,2,\ldots.., n)$$

όπου: $c_j, a_{ij}, b_i \in R$ ($i = 1,2,\ldots.., m$ και $j = 1,2,\ldots\ldots, n$)

και x_j είναι οι άγνωστες μεταβλητές του προβλήματος που πρέπει να υπολογιστούν ώστε η συνάρτηση z να λάβει την ελάχιστη ή την μέγιστη τιμή της αντίστοιχα και να ικανοποιούνται οι περιορισμοί του προβλήματος (Λεμεσιανός Α., 2014).

Ο γραμμικός προγραμματισμός μπορεί να εφαρμοστεί σε πληθώρα πεδίων.Πολλά από τα προβλήματα που χρειάζεται να λύσουμε καθημερινά χαρακτηρίζονται ως προβλήματα γραμμικού προγραμματισμού.
Με λίγα λόγια ο γραμμικός προγραμματισμός ασχολείται με την καλύτερη δυνατή κατανομή πόρων ανάμεσα σε παρόμοιες ή και ανταγωνιστικές δραστηριότητες (Ε.Μ.Π 2006).

1.3 Τι είναι οι τεχνικές κλιμάκωσης

Η κλιμάκωση είναι μια μέθοδος που εφαρμόζεται στα μεγάλα προβλήματα γραμμικού προγραμματισμού για να φέρει το γραμμικό πρόβλημα στην καλύτερη δυνατή μορφή για να μπορεί να επιλυθεί πιο εύκολα από έναν λύτη γραμμικών προβλημάτων(lp-solver)(Ploskas N., Samaras N. ,2013) . Σύμφωνα με τον Tomlin J.A. (1975) η κλιμάκωση εφαρμόζεται σε γραμμικά προβλήματα για 3 λόγους:i) για να γίνει μια συμπαγής αναπαράσταση των ορίων των μεταβλητών, ii) για να γίνει μείωση του αριθμού των επαναλήψεων που απαιτούνται για την επίλυση του γραμμικού προβλήματος, iii)για την βελτίωση της αριθμητικής συμπεριφοράς των αλγορίθμων. Η κλιμάκωση, παρά το γεγονός ότι περιλαμβάνει ένα υπολογιστικό κόστος, καθώς το πρόβλημα για να κλιμακωθεί χρειάζεται χρόνο, βοηθάει στο να λυθούν πιο γρήγορα τα γραμμικά προβλήματα και να παράγουν καλύτερα αποτελέσματα.Υπάρχουν αρκετοί τρόποι με τους οποίους μπορεί να εφαρμοστεί η κλιμάκωση σε ένα πρόβλημα. Οι τρόποι αυτοί ονομάζονται τεχνικές κλιμάκωσης.

1.4 Δομή κεφαλαίων διπλωματικής εργασίας

Στο Κεφάλαιο 2 παρουσιάζονται αναλυτικά οι τεχνικές κλιμάκωσης, με τους μαθηματικούς τύπους τους καθώς παρουσιάζεται και ένα αναλυτικό αριθμητικό παράδειγμα λυμένο για κάθε μέθοδο-τεχνική ξεχωριστά. Στο Κεφάλαιο 3, αναπτύσσονται οι τεχνικές κλιμάκωσης με την γλώσσα προγραμματισμού της python, παραθέτοντας για κάθε τεχνική αναλυτικά τον αλγόριθμο υλοποίησης με τον απαραίτητο σχολιασμο. Στο 4ο κεφάλαιο, αναπτύσσονται τα αποτελέσματα της υπολογιστικής μελέτης, με αναλυτικούς πίνακες που περιέχουν τους χρόνους υλοποίησης κάθε τεχνικής ξεχωριστά για 136 διαφορετικά γραμμικά προβλήματα. Τέλος, στο Κεφάλαιο 5 αναφέρονται τα συμπεράσματα που προέκυψαν απο την υπολογιστική μελέτη καθώς προτείνονται τρόποι για μελλοντική έρευνα.

2 Τεχνικές Κλιμάκωσης

Στο κεφάλαιο αυτό θα αναφερθούν αναλυτικά οι τεχνικές κλιμάκωσης, η μεθοδολογία τους, οι αριθμητικοί τους τύποι καθώς και θα παρουσιαστεί αναλυτικά ένα λυμένο αριθμητικό παράδειγμα για κάθε τεχνική.

2.1 Arithmetic mean (Τεχνική Αριθμητικού Μέσου)

Στην τεχνική του αριθμητικού μέσου κάθε στοιχείο κάθε γραμμής διαιρείται με τον αριθμητικό μέσο όλων των μη-μηδενικών στοιχείων της γραμμής. Η ακριβής εξίσωση κανονικοποίησης των γραμμών, φαίνεται στην εξίσωση (1) παρακάτω:

$$r_i^{k+1} = n_i / \Sigma|x_{i,j}^{\kappa}| \text{ , } i \in Mj \text{ (1), (Ploskas N., Samaras N. 2013)}$$

Με τον ίδιο τρόπο , κάθε στήλη διαιρείται με τον αριθμητικό μέσο όλων των μη μηδενικών στοιχείων της στήλης όπως φαίνεται στην εξίσωση παρακάτω:

$$s_j^{k+1} = m_j / \Sigma|x_{i,j}^{\kappa+1/2}| \text{ , } j \in Ni \text{ (2),(Ploskas N., Samaras N. 2013)}$$

<u>Αριθμητικό Παράδειγμα:</u>

max	50x1+	120x2+		40x3+		80x4	
s.t	2x1+	x2+		x3+			<= 450
		3x2+		x3+		x4	<= 180
	4x1+			x3+			<= 400
	x1+	x2+				x4	<= 110

xj(j=1...4)>=0

Πίνακας Α:

2	1	1	0
0	3	1	1
4	0	1	0

1	1	0	1

Πίνακας c:

50
120
40
80

Πίνακας b:

450
180
400
110

Υπολογίζεται ο πίνακας r, σύμφωνα με την εξίσωση (1)

Πίνακας r:

3/(2+1+1)=0.75
2/(3+1+1)= 0.4
2/(4+1) = 0.4
3/(1+1+1)=1

Κάθε στοιχείο του πίνακα r πολλαπλασιάζεται με κάθε στοιχείο των πινάκων A,b στην αντίστοιχη γραμμή. Αναλυτικά:

Πίνακας A:

x0.75	1.5	0.75	0.75	0
x0.4	0	1.2	0.4	0.4

x0.4	1.6	0	0.4	0
x1	1	1	0	1

Πίνακας b:

x0.75	337.5
x0.4	72
x0.4	160
x1	110

Στην συνέχεια υπολογίζεται ο πίνακας s σύμφωνα με την εξίσωση (2):

Πίνακας s:

3/(1.5+1.6+1)=0.7317
3/(0.75+1.2+1)= 1.0169
3/(0.75+0.4+0.4) = 1.9355
2/(0.4+0.1)=1.4286

Κάθε στοιχείο του πίνακα s πολλαπλασιάζεται με κάθε στοιχείο των πινάκων A,c στην αντίστοιχη στήλη. Αναλυτικά:

Πίνακας A:

x0.7317	x 1.0169	x1.9355	x1.4286
1.1976	0.7627	1.4516	0
0	1.2203	0.7742	0.5714
1.1707	0	0.7742	0
0.7317	1.0169	0	1.4286

Πίνακας c:

x0.7317	36.585
x 1.0169	122.028
x1.9355	77.42
x1.4286	114.288

Το τελικό γραμμικό πρόβλημα διαμορφώνεται ως εξής:

max 36.585 x1+ 122.028 x2+ 77.42 x3+ 114.288 x4

s.t 1.0976 x1+ 0.7627 x2+ 1.4516 x3+ <= 450

1.2203 x2+ 0.7742 x3+ 0.5714 x4 <= 180

1.1707 x1+ 0.7742 x3+ <= 400

0.7317 x1+ 1.0169 x2+ 1.4286 x4 <= 110

xj(j=1...4)>=0

2.2 de Buchet(p=1,2,∞)

Η τεχνική κλιμάκωσης de Buchet βασίζεται (Ploskas N., Samaras N. 2013) στην εξίσωση:

$$\max_{(r,s>0)} (\Sigma_{i,j \in Z} \{A_{ij} r_i s_j + 1/(A_{ij} r_i s)\}^p)^{1/p}$$

όπου p>0 και Z ο αριθμός των μη μηδενικών στοιχείων του πίνακα A.

Ειδικότερα για τις περιπτώσεις p=1,2, ∞ :

Για την περίπτωση p=1 οι γραμμές κλιμακώνονται ώς εξής:

$$r_i^{k+1} = \{(\Sigma\ 1/|X_{ij}^K|)\ (\Sigma\ |X_{ij}^K|)\}^{½} \quad (3)$$

Με τον ίδιο τρόπο οι στήλες κλιμακώνονται ως εξής:

$$s_j^{k+1} = \{(\Sigma\ 1/|X_{ij}^{K+1/2}|)\ (\Sigma\ 1/|X_{ij}^{K+1/2}|)\}^{½} \quad (4)$$

Για την περίπτωση p=2 οι γραμμές κλιμακώνονται ώς εξής:

$$r_i^{k+1} = \{(\Sigma\ (1/|X_{ij}^K|)^2)\ (\Sigma\ |X_{ij}^K|)^2\}^{¼} \quad (5)$$

Με τον ίδιο τρόπο οι στήλες κλιμακώνονται ως εξής:

$$s_j^{k+1} = \{(\Sigma\ (1/|X_{ij}^{K+1/2}|)^2)\ (\Sigma(|X_{ij}^{K+1/2}|)^2)\}^{¼} \quad (6)$$

Για την περίπτωση p=∞ οι γραμμές κλιμακώνονται ώς εξής:

$$r_i^{k+1} = 1/\{(\max\ |X_{ij}^K|)\ (\min\ |X_{ij}^K|)\}^{½} \quad (7)$$

Με τον ίδιο τρόπο οι στήλες κλιμακώνονται ως εξής:

$$s_j^{k+1} = \{(\max 1/|X_{ij}^{K+1/2}|)\ (\min 1/|X_{ij}^{K+1/2}|)\}^{½} \quad (8)$$

Αριθμητικό Παράδειγμα:

p=1

max	50x1+	120x2+	40x3+	80x4		
s.t	2x1+	x2+	x3+		<=	450
		3x2+	x3+	x4	<=	180
	4x1+		x3+		<=	400
	x1+	x2+		x4	<=	110

xj(j=1...4)>=0

Πίνακας A:

2	1	1	0
0	3	1	1
4	0	1	0
1	1	0	1

Πίνακας c:

50
120
40
80

Πίνακας b:

450
180

400
110

Υπολογίζεται ο πίνακας r, σύμφωνα με την εξίσωση (3):

Πίνακας r:

0.79026
0.6832
0.5
1

Κάθε στοιχείο του πίνακα r πολλαπλασιάζεται με κάθε στοιχείο των πινάκων A,b στην αντίστοιχη γραμμή. Αναλυτικά:

Πίνακας A:

x 0.7903	1.5806	0.7903	0.7903	0
x 0.6832	0	2.0496	0.6832	0.6832
x0.5	2	0	0.25	0
x1	1	1	0	1

Πίνακας b:

x0.7903	355.617
x0.6832	122.976
x0.5	200
x1	110

Στην συνέχεια υπολογίζεται ο πίνακας s σύμφωνα με την εξίσωση (4):

Πίνακας s:

0.6823
0.8465
0.7345
1.2098

Κάθε στοιχείο του πίνακα s πολλαπλασιάζεται με κάθε στοιχείο των πινάκων A,c στην αντίστοιχη στήλη. Αναλυτικά:

Πίνακας A:

x0.6823	x 0.8465	x0.7345	x1.2098
1.0784	0.669	0.5805	0
0	1.735	0.5018	0.8265
1.3646	0	0.1836	0
0.6823	0.8465	0	1.2098

Πίνακας c:

x0.6823	34.115
x 0.8465	101.58
x0.7345	29.38
x1.2098	96.784

Το τελικό γραμμικό πρόβλημα διαμορφώνεται ως εξής:

max	34.115 x1+	101.58 x2+	29.38	x3+	96.784 x4			
s.t	1.0784 x1+	0.669 x2+	0.5805	x3+			<=	355.617
		1.735 x2+	0.5018	x3+	0.8265	x4	<=	122.976
	1.3646 x1+		0.1836	x3+			<=	200
	0.6823 x1+	0.8465 x2+			1.2098	x4	<=	110

xj(j=1...4)>=0

p=2

max	50x1+	120x2+	40x3+	80x4		
s.t	2x1+	x2+	x3+		<=	450
		3x2+	x3+	x4	<=	180
	4x1+		x3+		<=	400
	x1+	x2+		x4	<=	110

xj(j=1...4)>=0

Πίνακας A:

2	1	1	0
0	3	1	1
4	0	1	0
1	1	0	1

Πίνακας c:

50
120
40
80

Πίνακας b:

450
180

400
110

Υπολογίζεται ο πίνακας r, σύμφωνα με την εξίσωση (5):

Πίνακας r:

0.7825
0.6619
0.5
1

Κάθε στοιχείο του πίνακα r πολλαπλασιάζεται με κάθε στοιχείο των πινάκων A,b στην αντίστοιχη γραμμή. Αναλυτικά:

Πίνακας A:

x 0.7825	1.565	0.7825	0.7825	0
x 0.6619	0	1.9857	0.6619	0.6619
x0.5	2	0	0.5	0
x1	1	1	0	1

Πίνακας b:

x0.7825	352.125
x0.6619	119.142
x0.5	200
x1	110

Στην συνέχεια υπολογίζεται ο πίνακας s σύμφωνα με την εξίσωση (6):

Πίνακας s:

0.6869
0.8492
1.5707
1.2291

Κάθε στοιχείο του πίνακα s πολλαπλασιάζεται με κάθε στοιχείο των πινάκων A,c στην αντίστοιχη στήλη. Αναλυτικά:

Πίνακας A:

x0.6869	x 0.8492	x1.5707	x1.2291
1.075	0.6645	1.2290	0
0	1.6863	1.0396	0.8135
1.3738	0	0.7854	0
0.6869	0.8492	0	1.2291

Πίνακας c:

x0.6869	34.345
x 0.8492	101.904
x1.5707	62.828
x1.2291	98.328

Το τελικό γραμμικό πρόβλημα διαμορφώνεται ως εξής:

max	34.345 x1+	101.904 x2+	62.828 x3+	98.328 x4				
s.t	1.075 x1+	0.6645 x2+	1.2290 x3+			<=	352.125	
		1.6863 x2+	1.0396 x3+	0.8135 x4		<=	119.142	
	1.3738 x1+		0.7854 x3+			<=	200	
	0.6869 x1+	0.8492 x2+		1.2291 x4		<=	110	

xj(j=1...4)>=0

p=∞

max	50x1+	120x2+	40x3+	80x4		
s.t	2x1+	x2+	x3+		<=	450
		3x2+	x3+	x4	<=	180
	4x1+		x3+		<=	400
	x1+	x2+		x4	<=	110

xj(j=1...4)>=0

Πίνακας A:

2	1	1	0
0	3	1	1
4	0	1	0
1	1	0	1

Πίνακας c:

50
120
40
80

Πίνακας b:

450

180
400
110

Υπολογίζεται ο πίνακας r, σύμφωνα με την εξίσωση (7):

Πίνακας r:

0.7071
0.5774
0.5
1

Κάθε στοιχείο του πίνακα r πολλαπλασιάζεται με κάθε στοιχείο των πινάκων A,b στην αντίστοιχη γραμμή. Αναλυτικά:

Πίνακας A:

x 0.7071	1.4142	0.7071	0.7071	0
x 0.5774	0	1.7322	0.5774	0.5774
x0.5	2	0	0.5	0
x1	1	1	0	1

Πίνακας b:

x0.7071	318.195
x0.5774	103.932
x0.5	200
x1	110

Στην συνέχεια υπολογίζεται ο πίνακας s σύμφωνα με την εξίσωση (8):

Πίνακας s:

0.7071
0.9036
1.6818
1.316

Κάθε στοιχείο του πίνακα s πολλαπλασιάζεται με κάθε στοιχείο των πινάκων A,c στην αντίστοιχη στήλη. Αναλυτικά:

Πίνακας A:

x0.7071	x 0.9036	x1.6818	x1.316
0.1	0.6389	1.2967	0
0	1.5652	0.9710	0.7599
1.4142	0	0.8409	0
1.316	1.316	0	1.316

Πίνακας c:

x0.7071	35.353
x 0.9036	108.432
x1.6818	67.272
x1.316	105.28

Το τελικό γραμμικό πρόβλημα διαμορφώνεται ως εξής:

max	35.355 x1+	108.432 x2+	67.272 x3+	105.28 x4				
s.t	0.1 x1+	0.6389 x2+	1.2967 x3+			<=	318.195
		1.5652 x2+	0.9710 x3+	0.7599 x4		<=	103.932
	1.4142 x1+		0.8409 x3+			<=	200

1.316 x1+ 1.316 x2+ 1.316 x4 <= 110

xj(j=1...4)>=0

2.3 Entropy

Η τεχνική κλιμάκωσης Entropy, βασίζεται στην ίδια μεθοδολογία με την τεχνική του αριθμητικού μέσου(arithmetic mean)(Ploskas N., Samaras N. 2013).

Αριθμητικό Παράδειγμα:

max 50x1+ 120x2+ 40x3+ 80x4

s.t 2x1+ x2+ x3+ <= 450

3x2+ x3+ x4 <= 180

4x1+ x3+ <= 400

x1+ x2+ x4 <= 110

xj(j=1...4)>=0

Πίνακας A:

2	1	1	0
0	3	1	1
4	0	1	0
1	1	0	1

Πίνακας c:

50
120
40
80

Πίνακας b:

450
180
400
110

Υπολογίζεται ο πίνακας r, σύμφωνα με την εξίσωση (1)

Πίνακας r:

0.75
0.4
0.4
1

Κάθε στοιχείο του πίνακα r πολλαπλασιάζεται με κάθε στοιχείο των πινάκων A,b στην αντίστοιχη γραμμή. Αναλυτικά:

Πίνακας A:

x0.75	1.5	0.75	0.75	0
x0.4	0	1.2	0.4	0.4
x0.4	1.6	0	0.4	0
x1	1	1	0	1

Πίνακας b:

x0.75	337.5
x0.4	72
x0.4	160
x1	110

Στην συνέχεια υπολογίζεται ο πίνακας s σύμφωνα με την εξίσωση (2):

Πίνακας s:

0.7317
1.0169
1.9355
1.4286

Κάθε στοιχείο του πίνακα s πολλαπλασιάζεται με κάθε στοιχείο των πινάκων A,c στην αντίστοιχη στήλη. Αναλυτικά:

Πίνακας A:

x0.7317	x 1.0169	x1.9355	x1.4286
1.1976	0.7627	1.4516	0
0	1.2203	0.7742	0.5714
1.1707	0	0.7742	0
0.7317	1.0169	0	1.4286

Πίνακας c:

x0.7317	36.585
x 1.0169	122.028
x1.9355	77.42
x1.4286	114.288

Το τελικό γραμμικό πρόβλημα διαμορφώνεται ως εξής:

max 36.585 x1+ 122.028 x2+ 77.42 x3+ 114.288 x4

s.t	1.0976 x1+	0.7627 x2+	1.4516 x3+			<=	450
		1.2203 x2+	0.7742 x3+	0.5714	x4	<=	180
	1.1707 x1+		0.7742 x3+			<=	400
	0.7317 x1+	1.0169 x2+		1.4286	x4	<=	110

xj(j=1...4)>=0

2.4 Equilibration

Σε αυτή την τεχνική κλιμάκωσης, κάθε στοιχείο του πίνακα A και του πίνακα b πολλαπλασιάζεται με τον αντίστροφο από το μεγαλύτερο στοιχείο κάθε γραμμής του πίνακα A. Έπειτα για κάθε στήλη του πίνακα A βρίσκεται το μεγαλύτερο στοιχείο σε απόλυτη τιμη που να μήν είναι το 1 και οι πίνακες A και c πολλαπλασιάζονται με τον αντίστροφο του αντίστοιχα. Με την τεχνική αυτή όλα τα στοιχεία του πίνακα A θα έχουν τιμές από -1 εώς και 1.

Αριθμητικό παράδειγμα:

max	50x1+	120x2+	40x3+	80x4			
s.t	2x1+	x2+	x3+			<=	450
		3x2+	x3+	x4		<=	180
	4x1+		x3+			<=	400
	x1+	x2+		x4		<=	110

xj(j=1...4)>=0

Πίνακας A:

2	1	1	0
0	3	1	1
4	0	1	0
1	1	0	1

Πίνακας c:

50
120
40
80

Πίνακας b:

450
180
400
110

Στην συνέχεια υπολογίζεται ο πίνακας r με τον τρόπο που αναφέρθηκε παραπάνω.

Πίνακας r:

0.5
0.3333
0.3333
1

Κάθε στοιχείο του πίνακα r πολλαπλασιάζεται με κάθε στοιχείο των πινάκων A,b στην αντίστοιχη γραμμή. Αναλυτικά:

Πίνακας A:

x0.5	1	0.5	0.5	0
x0.3333	0	1	0.3333	0.3333
x0.3333	1	0	0.25	0
x1	1	1	0	1

Πίνακας b:

x0.5	225
x0.3333	60
x0.3333	100
x1	110

Στην συνέχεια υπολογίζεται ο πίνακας s με τον τρόπο που αναφέρθηκε:

Πίνακας s:

1
1
0.5
1

Κάθε στοιχείο του πίνακα s πολλαπλασιάζεται με κάθε στοιχείο των πινάκων A,c στην αντίστοιχη στήλη. Αναλυτικά:

Πίνακας A:

x1	x 1	x0.5	x1
1	0.5	0.25	0
0	1	0.1667	0.3333
1	0	0.125	0
1	1	0	1

Πίνακας c:

x1	50
x1	120
x0.5	20

x1	80

Το τελικό γραμμικό πρόβλημα διαμορφώνεται ως εξής:

max	50 x1+	120 x2+	20 x3+	80 x4			
s.t	1. x1+	0.5 x2+	0.25 x3+			<=	225
		1 x2+	0.1667 x3+	0.3333	x4	<=	60
	1 x1+		0.125 x3+			<=	100
	1 x1+	1 x2+		1 x4		<=	110

xj(j=1...4)>=0

2.5 Geometric Mean

Στην τεχνική του γεωμετρικού μέσου, οι γραμμές του προβλήματος κανονικοποιούνται ώς εξής (Ploskas N., Samaras N. 2013):

$r_i^{k+1}=(\max X_{ij}^k \min X_{ij}^k)^{-½}$, $i \in Mj$ (9)

Με παρόμοιο τρόπο οι στήλες κανονικοποιούνται ως εξής:

$s_j^{k+1}=(\max X_{ij}^{k+1/2} \min X_{ij}^{k+1/2})^{-½}$, $j \in Ni$ (10)

Αριθμητικό Παράδειγμα

max	50x1+	120x2+	40x3+	80x4			
s.t	2x1+	x2+	x3+			<=	450
		3x2+	x3+	x4		<=	180
	4x1+		x3+			<=	400
	x1+	x2+		x4		<=	110

xj(j=1...4)>=0

Πίνακας Α:

2	1	1	0

0	3	1	1
4	0	1	0
1	1	0	1

Πίνακας c:

50
120
40
80

Πίνακας b:

450
180
400
110

Υπολογίζεται ο πίνακας r, σύμφωνα με την εξίσωση (9)

Πίνακας r:

0.7071
0.5774
0.5
1

Κάθε στοιχείο του πίνακα s πολλαπλασιάζεται με κάθε στοιχείο των πινάκων A,c στην αντίστοιχη γραμμή. Αναλυτικά:

Πίνακας A:

x0.7071	1.4142	0.7071	0.7071	0
x0.5774	0	1.7322	0.5774	0.5774
x0.5	2	0	0.5	0
x1	1	1	0	1

Πίνακας b:

x0.7071	318.195
x0.5774	103.932
x0.5	200
x1	110

Υπολογίζεται ο πίνακας s, σύμφωνα με την εξίσωση (10)

Πίνακας s:

0.7071
0.9036
1.6818
1.316

Πίνακας A:

x0.7071	x 0.9036	x1.6818	x1.316
1	0.6389	1.1892	0
0	1.5652	0.971	0.7599
1.4142	0	0.8409	0
0.7071	0.9036	0	1.316

Πίνακας c:

x0.7071	35.355
x 0.9036	108.432
x1.6818	67.272
x1.316	105.28

Το τελικό γραμμικό πρόβλημα διαμορφώνεται ως εξής:

max	35.355 x1+	108.432 x2+	67.272 x3+	105.28 x4				
s.t	1 x1+	0.6389 x2+	0.1892 x3+			<=	318.195	
		1.5652 x2+	0.971 x3+	0.7599 x4		<=	103.932	
	1.4142 x1+		0.8409 x3+			<=	200	
	0.7071 x1+	0.9036 x2+		1.316 x4		<=	110	

xj(j=1...4)>=0

2.6 IBM MPSX

Σε αυτήν την τεχνική, (Ploskas N., Samaras N. 2013) εφαρμόζεται η τεχνική του γεωμετρικού μέσου μέχρι και 4 φορές, ή μέχρι να ισχύει η σχέση (11). Στην συνέχεια εφαρμόζεται η τεχνική equilibration.

$$1/|Z|(\Sigma (X_{ij}^{k})^2-(\Sigma (X_{ij}^{k}))^2/|Z|)<\varepsilon,\ i,j \in Z \quad (11),$$

όπου |Z| είναι ο καρτεσιανός αριθμός των μη μηδενικών στοιχείων του X^k και ε είναι μία σταθερά που συνήθως καθορίζεται να είναι μικρότερη του 10.Στην συνέχεια εφαρμόζεται η τεχνική κλιμάκωσης equilibration.

Αριθμητικό Παράδειγμα

max	50x1+	120x2+	40x3+	80x4		
s.t	2x1+	x2+	x3+		<=	450
		3x2+	x3+	x4	<=	180

4x1+		x3+		<=	400
x1+	x2+		x4	<=	110

xj(j=1...4)>=0

Πίνακας A:

2	1	1	0
0	3	1	1
4	0	1	0
1	1	0	1

Πίνακας c:

50
120
40
80

Πίνακας b:

450
180
400
110

Σύμφωνα με την τεχνική IBM MPSX στο προβλημα εφαρμόζεται η τεχνική του γεωμετρικού μέσου 4 φορές ή μέχρι να ισχύει η σχέση (11). Άρα με την εφαρμογή της τεχνικής του γεωμετρικού μέσου για πρώτη φορά έχουμε:

Πίνακας r:

0.7071
0.5774
0.5
1

Πίνακας A:

x0.7071	1.4142	0.7071	0.7071	0
x0.5774	0	1.7322	0.5774	0.5774
x0.5	2	0	0.5	0
x1	1	1	0	1

Πίνακας b:

x0.7071	318.195
x0.5774	103.932
x0.5	200
x1	110

Πίνακας s:

0.7071
0.9036
1.6818
1.316

Πίνακας A:

x0.7071	x 0.9036	x1.6818	x1.316

1	0.6389	1.1892	0
0	1.5652	0.971	0.7599
1.4142	0	0.8409	0
0.7071	0.9036	0	1.316

Πίνακας c:

x0.7071	35.355
x 0.9036	108.432
x1.6818	67.272
x1.316	105.28

Στην συνέχεια γίνεται έλεγχος αν ισχύει η σχέση:

1/11*(12.547943720000001-(127.84372624000002/11)=0.08416243537190078<10

Από την στιγμή που ισχύει η σχέση θα εφαρμοστεί η τεχνική equilibration. Άρα:

Πίνακας r:

0.8409
0.6389
0.7071
0.7599

Πίνακας A:

x 0.8409	0.8409	0.5372	1	0
x 0.6389	0	1	0.6204	0.4855
x 0.7071	1	0	0.5946	0
x 0.7599	0.5373	0.6866	0	1

Πίνακας b:

x 0.8409	268.1756
x 0.6389	66.4021
x 0.7071	141.42
x 0.7599	83.589

Πίνακας s:

1
1
1
1

Πίνακας A:

x1	x 1	x1	x1
0.8409	0.5372	1	0
0	1	0.6204	0.4855
1	0	0.5946	0
0.5373	0.6866	0	1

Πίνακας c:

x1	35.355
x1	108.432
x1	67.272
x1	105.28

Το τελικό γραμμικό πρόβλημα διαμορφώνεται ως εξής:

max 35.355 x1+ 108.432 x2+ 67.272 x3+ 105.28 x4

s.t	0.8409	x1+	0.5372	x2+	1	x3+			<=	268.1756
			1	x2+	0.6204	x3+	0.4855	x4	<=	66.4021
	1	x1+			0.5946	x3+			<=	141.42
	0.5373	x1+	0.6866	x2+			1	x4	<=	83.589

xj(j=1...4)>=0

2.7 LP Norm(p=1,2,∞)

Η τεχνική κλιμάκωσης de Buchet βασίζεται στην εξίσωση:

$$\max_{(r,s>0)} (\Sigma_{i,j \in Z} |\log(A_{ij} r_i s_{j)}|^p)^{1/p}$$

όπου p>0 και Z ο αριθμός των μη μηδενικών στοιχείων του A (Ploskas N., Samaras N. 2013).

Ειδικότερα για τις περιπτώσεις p=1,2, ∞:

Για την περίπτωση p=1 οι γραμμές κλιμακώνονται ώς εξής.:

$$r_i^{k+1} = 1/\text{median}\{|X_{ij}^K|), j \in N_{i\,(12)}$$

Με τον ίδιο τρόπο οι στήλες κλιμακώνονται ως εξής:

$$s_j^{k+1} = 1/\text{median}\{|X_{ij}^{K+1/2}|), i \in M_{j\,(13)}$$

Για την περίπτωση p=2 οι γραμμές κλιμακώνονται ώς εξής:

$$r_i^{k+1} = 1/\Pi(X_{ij}^K)^{1/nι} \ , j \in N_{i(14)}$$

Με τον ίδιο τρόπο οι στήλες κλιμακώνονται ως εξής:

$$s_j^{k+1} = 1/\Pi(X_{ij}^{K+1/2})^{1/mjι} \ , i \in M_{j\,(15)}$$

Για την περίπτωση p=∞ οι γραμμές και οι στήλες κλιμακώνονται σύμφωνα με την τεχνική de Buchet(περίπτωση p=∞) και γεωμετρικού μέσου.

Αριθμητικό παράδειγμα

p=1

max	50x1+	120x2+	40x3+	80x4		
s.t	2x1+	x2+	x3+		<=	450
		3x2+	x3+	x4	<=	180
	4x1+		x3+		<=	400

x1+ x2+ x4 <= 110

xj(j=1...4)>=0

Πίνακας A:

2	1	1	0
0	3	1	1
4	0	1	0
1	1	0	1

Πίνακας c:

50
120
40
80

Πίνακας b:

450
180
400
110

Υπολογίζεται ο πίνακας r, σύμφωνα με την εξίσωση (12):

Πίνακας r:

1
1
2.5
1

Κάθε στοιχείο του πίνακα s πολλαπλασιάζεται με κάθε στοιχείο των πινάκων A,c στην αντίστοιχη γραμμή. Αναλυτικά:

Πίνακας A:

x1	2	1	1	0
x 1	0	3	1	1
x 2.5	10	0	2.5	0
x 1	1	1	0	1

Πίνακας b:

x 1	450
x 1	180
x 2.5	1000
x 1	110

Υπολογίζεται ο πίνακας s, σύμφωνα με την εξίσωση (13)

Πίνακας s:

2
1
1
1

Κάθε στοιχείο του πίνακα s πολλαπλασιάζεται με κάθε στοιχείο των πινάκων A,c στην αντίστοιχη γραμμή. Αναλυτικά:

Πίνακας A:

x2	x 1	x1	x1
4	1	1	0
0	3	1	1
20	0	2.5	0
2	1	0	1

Πίνακας c:

x2	100
x1	120
x1	40
x1	80

Το τελικό γραμμικό πρόβλημα διαμορφώνεται ως εξής:

max 100 x1+ 120 x2+ 40 x3+ 80 x4

s.t 4 x1+ 1 x2+ 1 x3+ <= 450

3 x2+ 1 x3+ 2.5 x4 <= 180

20 x1+ 2.5 x3+ <= 1000

2 x1+ 1 x2+ 1 x4 <= 110

xj(j=1...4)>=0

p=2

max 50x1+ 120x2+ 40x3+ 80x4

s.t	2x1+	x2+	x3+		<=	450
		3x2+	x3+	x4	<=	180
	4x1+		x3+		<=	400
	x1+	x2+		x4	<=	110

xj(j=1...4)>=0

Πίνακας A:

2	1	1	0
0	3	1	1
4	0	1	0
1	1	0	1

Πίνακας c:

50
120
40
80

Πίνακας b:

450
180
400
110

Υπολογίζεται ο πίνακας r, σύμφωνα με την εξίσωση (14):

Πίνακας r:

0.7937

0.6934
0.5
1

Κάθε στοιχείο του πίνακα s πολλαπλασιάζεται με κάθε στοιχείο των πινάκων A,c στην αντίστοιχη γραμμή. Αναλυτικά:

Πίνακας A:

x0.7937	1.5874	0.7937	0.7937	0
x 0.6934	0	3	0.6934	2.0802
x 0.5	5	0	1.25	0
x 1	1	1	0	1

Πίνακας b:

x0.7937	357.165
x 0.6934	124.812
x 0.5	200
x 1	110

Υπολογίζεται ο πίνακας s, σύμφωνα με την εξίσωση (15)

Πίνακας s:

0.5013
0.7489
1.1328
0.693

Κάθε στοιχείο του πίνακα s πολλαπλασιάζεται με κάθε στοιχείο των πινάκων A,c στην αντίστοιχη γραμμή. Αναλυτικά:

Πίνακας A:

x0.5013	x 0.7489	x1.1328	x0.693
0.7958	0.5944	0.8991	0
0	2.2467	0.7855	2.0802
2.5065	0	1.416	0
0.5013	0.7489	0	1.4416

Πίνακας c:

x0.5013	25.065
x0.7489	89.868
x1.1328	45.312
x0.693	55.44

Το τελικό γραμμικό πρόβλημα διαμορφώνεται ως εξής:

max 25.065 x1+ 89.868 x2+ 45.312 x3+ 55.44 x4

s.t 0.7958 x1+ 0.5944 x2+ 0.8991 x3+ <= 357.165

2.2467 x2+ 0.7855 x3+ 2.0802 x4 <= 124.812

2.5065 x1+ 1.416 x3+ <= 200

0.5013 x1+ 0.7489 x2+ 1.4416 x4 <= 110

xj(j=1...4)>=0

p=∞

max 50x1+ 120x2+ 40x3+ 80x4

s.t	2x1+	x2+	x3+		<=	450
		3x2+	x3+	x4	<=	180
	4x1+		x3+		<=	400
	x1+	x2+		x4	<=	110

xj(j=1...4)>=0

Πίνακας A:

2	1	1	0
0	3	1	1
4	0	1	0
1	1	0	1

Πίνακας c:

50
120
40
80

Πίνακας b:

450
180
400
110

Πίνακας r:

0.7071

0.5774
0.5
1

Κάθε στοιχείο του πίνακα s πολλαπλασιάζεται με κάθε στοιχείο των πινάκων A,c στην αντίστοιχη γραμμή. Αναλυτικά:

Πίνακας A:

x 0.7071	1.4142	0.7071	0.7071	0
x 0.5774	0	1.7322	0.5774	0.5774
x0.5	2	0	0.5	0
x1	1	1	0	1

Πίνακας b:

x0.7071	318.195
x0.5774	103.932
x0.5	200
x1	110

Πίνακας s:

0.7071
0.9036
1.6818
1.316

Κάθε στοιχείο του πίνακα s πολλαπλασιάζεται με κάθε στοιχείο των πινάκων A,c στην αντίστοιχη γραμμή. Αναλυτικά:

Πίνακας A:

x0.7071	x 0.9036	x1.6818	x1.316
0.1	0.6389	1.2967	0
0	1.5652	0.9710	0.7599
1.4142	0	0.8409	0
1.316	1.316	0	1.316

Πίνακας c:

x0.7071	35.353
x 0.9036	108.432
x1.6818	67.272
x1.316	105.28

Το τελικό γραμμικό πρόβλημα διαμορφώνεται ως εξής:

max	35.355 x1+	108.432 x2+	67.272 x3+	105.28 x4		
s.t	0.1 x1+	0.6389 x2+	1.2967 x3+		<=	318.195
		1.5652 x2+	0.9710 x3+	0.7599 x4	<=	103.932
	1.4142 x1+		0.8409 x3+		<=	200
	1.316 x1+	1.316 x2+		1.316 x4	<=	110

xj(j=1...4)>=0

3 Τεχνικές κλιμάκωσης και Python

Στο κεφάλαιο αυτό, παρουσιάζεται αναλυτικά ο αλγόριθμος για κάθε τεχνική κλιμάκωσης ξεχωριστά. Για την καλύτερη κατανόηση των αλγορίθμων παρατίθενται τα ονόματα των δομών που είναι κοινά σε όλους τους αλγορίθμους:

A = λίστα που περιλαμβάνει τα στοιχεία του πίνακα A του πορβλήματος από το αρχείο mps.

c=λίστα που περιλαμβάνει τα στοιχεία του πίνακα c του προβλήματος από το αρχείο mps.

b=λίστα που περιλαμβάνει τα στοιχεία του πίνακα b του προβλήματος από το αρχείο mps.

Ab= λίστα που περιλαμβάνει το σύμβολο ισότητας ή ανισότητας κάθε περιορισμού

metav= λίστα που περιλαμβάνει τα ονόματα των μεταβλητών του προβλήματος.

3.1 Arithmetic mean (Τεχνική Αριθμητικού Μέσου)

```
r=[] #αρχικοποίηση πίνακα r
s=[] #αρχικοποίηση πίνακα s
for i in A:
    count=0 #αριθμός μη μηδενικών στοιχείων σε κάθε σειρά του A
    su=0 # το άθροισμα των μη μηδενικών στοιχείων σε κάθε σειρά
    for j in i:
        if j!=0:
            count+=1 #αύξηση αθροίσματος μη μηδενικών στοιχείων
            su+=float(j) # προσθήκη της τιμής του στοιχείου στο άθροισμα
    if su!=0.0:
        r.append(float(count)/float(su))#υπολογισμός του πίνακα r
    else:
        r.append(0.0)
for i in range(len(A)):
    for j in range(len(A[i])):
        A[i][j]=float(A[i][j])*float(r[i]) #γέμισμα πίνακα A μετά τον πολλαπλασιασμό
κάθε στοιχείου του με το αντίστοιχο στοιχείο του πίνακα r
    b[i]*=r[i]# γέμισμα πίνακα b μετά τον πολλαπλασιασμό με το αντίστοιχο r
```

```
    for i in range(len(metav)):
        count=0 #αριθμός μη μηδενικών στοιχείων σε κάθε στήλη του A
        su=0 # το άθροισμα των μη μηδενικών στοιχείων σε κάθε στήλη του A
        for j in range((len(rows)-1)):
            if A[j][i]!=0.0 or A[j][i]!=- 0.0 :
                count+=1#αύξηση αθροίσματος μη μηδενικών στοιχείων
                su+=float(A[j][i])# προσθήκη της τιμής του στοιχείου στο άθροισμα
        if su!=0.0:
            s.append(float(count)/float(su)) #υπολογισμός του πίνακα s
        else:
            s.append(0.0)
    for i in range(len(metav)):
        for j in range((len(rows)-1)):
            A[j][i]=float(A[j][i])*float(s[i] #γέμισμα πίνακα A μετά τον πολλαπλασιασμό
με το αντίστοιχο s
        c[i]*=s[i]# γέμισμα πίνακα c μετά τον πολλαπλασιασμό με το αντίστοιχο s
    print(cost," z=",end="")
    for i in range(len(c)):
        if c[i]==0.0:
            continue
        else:
            print("+",c[i],"*",metav[i],end="") # εκτύπωση της αντικειμενικής συνάρτησης
    print("\n""s.t: ",end="")
    for i in range(len(A)):
        if sum(A[i])==0:
            continue
        for j in range(len(metav)): # εκτύπωση των περιορισμών.
            if A[i][j]==0.0:
                continue
            else:
                print("+",A[i][j],"*",metav[j],end="")
        print(Ab[i] ," ",b[i]),
        print("\n")
```

3.2 de Buchet(p=1,2,oo)

Περίπτωση p=1

```
for i in A:
        count=0 # άθροισμα των αντιστρόφων όλων  μη μηδενικών στοιχείων σε κάθε γραμμή
    su=0 # άθροισμα των τιμών των μη μηδενικών στοιχείων σε κάθε γραμμή
    for j in i:
      if j!=0:
        count= count +1/float(j)#αύξηση αθροίσματος
        su+=float(j)#αύξηση αθροίσματος
    if su!=0.0:
      r.append(sqrt(abs(float(count)/float(su))))#υπολογισμός του πίνακα r
    else:
      r.append(0.0)
  for i in range(len(A)):
    for j in range(len(A[i])):
        A[i][j]=float(A[i][j])*float(r[i])  #γέμισμα του πίνακα A πολλαπλασιάζοντας κάθε στοιχείο κάθε γράμμης του πίνακα A με το αντίστοιχο στοιχείο του πίνακα r
    b[i]*=r[i]# refill the b array
  for i in range(len(metav)):
      count=0 #  άθροισμα των αντιστρόφων όλων  μη μηδενικών στοιχείων σε κάθε κάθε στήλη του A
    su=0 # άθροισμα των τιμών όλων των μη μηδενικών στοιχείων κάθε στήλης
    for j in range((len(rows)-1)):
      if A[j][i]!=0.0 or A[j][i]!=- 0.0 :
        count=count+1/A[j][i]
        su+=float(A[j][i])
    if su!=0.0:
      s.append(sqrt(abs(float(count)/float(su)))) #υπολογισμός του πίνακα s
    else:
      s.append(0.0)
  for i in range(len(metav)):
    for j in range((len(rows)-1)):
```

```
            A[j][i]=float(A[j][i])*float(s[i]) #γέμισμα του πίνακα A πολλαπλασιάζοντας κάθε στοιχείο κάθε στήλης του πίνακα A με το αντίστοιχο στοιχείο του πίνακα s
        c[i]= float(c[i]*s[i])
    print(cost," z=",end="")
    for i in range(len(c)):
        if float(c[i])==0.0:
            continue
        else:
            print("+",c[i],"*",metav[i],end="") # Εκτύπωση της αντικειμενικής συνάρτησης
    print("\n""s.t: ",end="")
    for i in range(len(A)):
        if sum(A[i])==0:
            continue
        for j in range(len(metav)): # εκτύπωση των περιορισμών
            if A[i][j]==0.0:
                continue
            else:
                print("+",A[i][j],"*",metav[j],end="")
        print(Ab[i] ," ",b[i])
        print("\n")
```

Περίπτωση p=2

```
for i in range(len(A)):
        count=0
        su=0
          a1 =[abs(float(element)) for element in A[i]] # αντικατάσταση όλων των τιμών του πίνακα A με την απόλυτη τιμή τους
        A[i]=a1
             Anonzero=list(filter(lambda ele: ele !=0.0,a1)) # πίνακας που περιέχει τα στοιχεία της σειράς A[i] χώρις τα μηδενικά στοιχεία
           if not Anonzero:#σε περίπτωση που δεν υπάρχει μη μηδενικό στοιχείο σε μια γραμμή προστίθεται 0 στον πίνακα r
            r.append(0.0)
```

```
            continue
        for j in Anonzero: #από τα μη μηδενικά στοιχεία κάθε γραμμής
            count+=pow(j,-2) # αύξηση του αθροίσματος υψώνοντας κάθε στοιχείο εις την -2
                su+=pow(j,2) # αύξηση του αθροίσματος υψώνοντας κάθε στοιχείο στο τετράγωνο.
            r.append(pow(count/su,1/4)) #υπολογισμός του αντίστοιχου r διαιρώντας τα 2 αθροίσματα και υψώνοντας το αποτέλεσμα εις την 1/4.
    for i in range(len(A)):
        for j in range(len(A[i])):
                A[i][j]=float(A[i][j])*float(r[i]) #υπολογισμός πίνακα A έπειτα από τον πολλαπλασιασμό των στοιχείων με τα αντίστοιχα του πίνακα r
            b[i]*=r[i] #πολλαπλασιασμός κάθε στοιχείου του πίνακα b με το αντίστοιχο στοιχείο του πίνακα r.
    for j in range(len(metav)):
        count=0 #μηδενισμός του αριθμητή
        su=0 #μηδενισμός του παρανομαστή
        collumn=[item[j] for item in A] #η στήλη που εξετάζεται σε κάθε επανάληψη
     Anonzero=list(filter(lambda ele: ele !=0.0,collumn))#πίνακας στήλης j του A χωρίς τα μηδενικά στοιχεία
            if not Anonzero:#σε περίπτωση που δεν υπάρχει μη μηδενικό στοιχείο σε μια στήλη προστίθεται 0 στον πίνακα s
            s.append(0.0)
            continue #συνέχεια στην επόμενη στήλη
        for i in Anonzero:
            count+=pow(i,-2) # αύξηση του αθροίσματος υψώνοντας κάθε στοιχείο εις την -2
                su+=pow(i,2) # αύξηση του αθροίσματος υψώνοντας κάθε στοιχείο στο τετράγωνο.
            s.append(pow(count/su,1/4))#υπολογισμός του αντίστοιχου s διαιρώντας τα 2 αθροίσματα και υψώνοντας το αποτέλεσμα εις την 1/4.

        for i in range(len(metav)):#οπου metav, ο πίνακας με όλες τις μεταβλητές του
```

```
προβλήματος
        for j in range((len(rows)-1)):
                A[j][i]=float(A[j][i])*float(s[i]) #υπολογισμός πίνακα A έπειτα από τον πολλαπλασιασμό των στοιχείων με τα αντίστοιχα του πίνακα s
            c[i]= float(c[i]*s[i]) #πολλαπλασιασμός κάθε στοιχείου του πίνακα c με το αντίστοιχο στοιχείο του πίνακα s.
    print(cost," z=",end="")#εκτύπωση της αντικειμενικής συνάρτησης.
    for i in range(len(c)):
      if float(c[i])==0.0:
        continue
      else:
        print("+",c[i],"*",metav[i],end="")
    print("\n""s.t: ",end="")
    for i in range(len(A)):
      if sum(A[i])==0:
         continue
      for j in range(len(metav)): # εκτύπωση των περιορισμών
        if A[i][j]==0.0:
          continue
        else:
          print("+",A[i][j],"*",metav[j],end="")
      print(Ab[i] ," ",b[i])
      print("\n")
```

Περίπτωση p=∞

```
for i in range(len(A)):
        a1 =[abs(float(element)) for element in A[i]]# αντικατάσταση όλων των τιμών του πίνακα A με την απόλυτη τιμή τους
      A[i]=a1
          Anonzero=list(filter(lambda ele: ele !=0.0,a1))#αποθηκεύονται στον πίνακα Anonzero όλα τα μη μηδενικά στοιχεία της γραμμής i του πίνακα A
          if not Anonzero:#σε περίπτωση που δεν υπάρχει μη μηδενικό στοιχείο σε μια γραμμή προστίθεται 0 στον πίνακα r
            r.append(0.0)
```

```
            continue
                    r.append(pow((max(Anonzero)*min(Anonzero)),-1/2))#υπολογισμός r πολλαπλασιάζοντας το μεγαλύτερο και το μικρότερο μη μηδενικό στοιχείο κάθε γραμμής και υψώνοντας τα στην -1/2
        for i in range(len(A)):
            for j in range(len(A[i])):
                    A[i][j]=float(A[i][j])*float(r[i]) #υπολογισμός πίνακα A έπειτα από τον πολλαπλασιασμό των στοιχείων με τα αντίστοιχα του πίνακα r
                b[i]*=r[i]#πολλαπλασιασμός κάθε στοιχείου του πίνακα b με το αντίστοιχο στοιχείο του πίνακα r.
        for j in range(len(metav)):
            collumn=[item[j] for item in A]#η στήλη που εξετάζεται σε κάθε επανάληψη
                Anonzero=list(filter(lambda ele: ele !=0.0,collumn))#πίνακας στήλης j του A χωρίς τα μηδενικά στοιχεία
                if not Anonzero:#σε περίπτωση που δεν υπάρχει μη μηδενικό στοιχείο σε μια στήλη προστίθεται 0 στον πίνακα s
                s.append(0.0)
                continue
                        s.append(pow((max(Anonzero)*min(Anonzero)),-1/2))#υπολογισμός s πολλαπλασιάζοντας το μεγαλύτερο και το μικρότερο μη μηδενικό στοιχείο κάθε στήλης.
        for i in range(len(metav)):
            for j in range((len(rows)-1)):
                    A[j][i]=float(A[j][i])*float(s[i]) #υπολογισμός πίνακα A έπειτα από τον πολλαπλασιασμό των στοιχείων με τα αντίστοιχα του πίνακα s
                c[i]= float(c[i]*s[i])#πολλαπλασιασμός κάθε στοιχείου του πίνακα c με το αντίστοιχο στοιχείο του πίνακα s.
        print(cost," z=",end="")
        for i in range(len(c)):
            if float(c[i])==0.0:
                continue
            else:
                print("+",c[i],"*",metav[i],end="") #εκτύπωση της αντικειμενικής συνάρτησης.
        print("\n""s.t: ",end="")
```

```
    for i in range(len(A)):
        if sum(A[i])==0:
            continue
        for j in range(len(metav)): #εκτύπωση των περιορισμών
            if A[i][j]==0.0:
                continue
            else:
                print("+",A[i][j],"*",metav[j],end="")
        print(Ab[i] ," ",b[i])
        print("\n")
```

3.3 Entropy

```
    r=[] #αρχικοποίηση πίνακα r
    s=[] #αρχικοποίηση πίνακα s


    for i in A:
        count=0 #αριθμός μη μηδενικών στοιχείων σε κάθε σειρά του A
        su=0 # το άθροισμα των μη μηδενικών στοιχείων σε κάθε σειρά
        for j in i:
            if j!=0:
                count+=1 #αύξηση αθροίσματος μη μηδενικών στοιχείων
                su+=float(j) # προσθήκη της τιμής του στοιχείου στο άθροισμα
        if su!=0.0:
            r.append(float(count)/float(su))#υπολογισμός του πίνακα r
        else:
            r.append(0.0)
    for i in range(len(A)):
        for j in range(len(A[i])):
            A[i][j]=float(A[i][j])*float(r[i]) #γέμισμα πίνακα A μετά τον πολλαπλασιασμό
κάθε στοιχείου του με το αντίστοιχο στοιχείο του πίνακα r
        b[i]*=r[i]# γέμισμα πίνακα b μετά τον πολλαπλασιασμό με το αντίστοιχο r
    for i in range(len(metav)):
        count=0 #αριθμός μη μηδενικών στοιχείων σε κάθε στήλη του A
        su=0 # το άθροισμα των μη μηδενικών στοιχείων σε κάθε στήλη του A
        for j in range((len(rows)-1)):
            if A[j][i]!=0.0 or A[j][i]!=- 0.0 :
                count+=1#αύξηση αθροίσματος μη μηδενικών στοιχείων
                su+=float(A[j][i])# προσθήκη της τιμής του στοιχείου στο άθροισμα
        if su!=0.0:
            s.append(float(count)/float(su)) #υπολογισμός του πίνακα s
        else:
            s.append(0.0)
    for i in range(len(metav)):
        for j in range((len(rows)-1)):
```

```
            A[j][i]=float(A[j][i])*float(s[i] #γέμισμα πίνακα Α μετά τον πολλαπλασιασμό με το αντίστοιχο s
        c[i]*=s[i]# γέμισμα πίνακα c μετά τον πολλαπλασιασμό με το αντίστοιχο s
    print(cost," z=",end="")
    for i in range(len(c)):
        if c[i]==0.0:
            continue
        else:
            print("+",c[i],"*",metav[i],end="") # εκτύπωση της αντικειμενικής συνάρτησης
    print("\n""s.t: ",end="")
    for i in range(len(A)):
        if sum(A[i])==0:
            continue
        for j in range(len(metav)): # εκτύπωση των περιορισμών.
            if A[i][j]==0.0:
                continue
            else:
                print("+",A[i][j],"*",metav[j],end="")
        print(Ab[i] ," ",b[i]),
        print("\n")
```

3.4 Equilibration

```
for i in range(len(A)):
        a1 =[abs(float(element)) for element in A[i]]#αντικατάσταση κάθε στοιχείου του A με την απόλυτη τιμή του.
        A[i]=a1
            Anonzero=list(filter(lambda ele: ele !=0.0,a1))#αποθηκεύονται στον πίνακα Anonzero όλα τα μη μηδενικά στοιχεία της γραμμής i του πίνακα A
        if not Anonzero:
           r.append(0.0)
           continue
            r.append(1/max(Anonzero))#γέμισμα r με το αντίστροφο του μεγαλύτερου μη μηδενικού στοιχείου κάθε γραμμής
    for i in range(len(A)):
        for j in range(len(A[i])):
            A[i][j]=float(A[i][j])*float(r[i]) #γέμισμα πίνακα A μετά τον πολλαπλασιασμό κάθε στοιχείου του με το αντίστοιχο στοιχείο του πίνακα r
        b[i]*=r[i]# γέμισμα πίνακα b μετά τον πολλαπλασιασμό με το αντίστοιχο r
    for j in range(len(metav)):
        collumn=[item[j] for item in A]#η στήλη που εξετάζεται σε κάθε επανάληψη
            Anonzero=list(filter(lambda ele: ele !=0.0,collumn))#πίνακας στήλης j του A χωρίς τα μηδενικά στοιχεία
        if not Anonzero:
           s.append(0.0)
           continue
            s.append(1/max(Anonzero))#γέμισμα s με το αντίστροφο του μεγαλύτερου μη μηδενικού στοιχείου κάθε στήλης.
    for i in range(len(metav)):
        for j in range((len(rows)-1)):
            A[j][i]=float(A[j][i])*float(s[i])#γέμισμα πίνακα A μετά τον πολλαπλασιασμό κάθε στοιχείου του με το αντίστοιχο στοιχείο του πίνακα s
              c[i]= float(c[i]*s[i])# γέμισμα πίνακα c μετά τον πολλαπλασιασμό με το αντίστοιχο s
    print(cost," z=",end="")#εκτύπωση της αντικειμενικής συνάρτησης
```

```
    for i in range(len(c)):
        if float(c[i])==0.0:
            continue
        else:
            print("+",c[i],"*",metav[i],end="")
    print("\n""s.t: ",end="")#εκτύπωση των περιορισμών
    for i in range(len(A)):
        if sum(A[i])==0:
            continue
        for j in range(len(metav)):
            if A[i][j]==0.0:
                continue
            else:
                print("+",A[i][j],"*",metav[j],end="")
        print(Ab[i] ," ",b[i])
        print("\n")
```

3.5 Geometric Mean

```
for i in range(len(A)):
        a1 =[abs(float(element)) for element in A[i]]#αντικατάσταση κάθε στοιχείου του A με την απόλυτη τιμή του.
        A[i]=a1
            Anonzero=list(filter(lambda ele: ele !=0.0,a1))#αποθηκεύονται στον πίνακα Anonzero όλα τα μη μηδενικά στοιχεία της γραμμής ί του πίνακα A
        if not Anonzero:
            r.append(0.0)
            continue
                r.append(pow((max(Anonzero)*min(Anonzero)),-1/2))#υπολογισμός του r πολλαπλασιάζοντας το μεγαλύτερο με το μικρότερο στοιχείο κάθε γραμμής και υψώνοντας το γινόμενο στην -1/2)
    for i in range(len(A)):
        for j in range(len(A[i])):
                    A[i][j]=A[i][j]*r[i] #γέμισμα πίνακα A μετά τον πολλαπλασιασμό κάθε
```

```
στοιχείου του με το αντίστοιχο στοιχείο του πίνακα r
        b[i]*=r[i]# γέμισμα πίνακα b μετά τον πολλαπλασιασμό με το αντίστοιχο r
    for j in range(len(metav)):
        collumn=[item[j] for item in A]#η στήλη που εξετάζεται σε κάθε επανάληψη
            Anonzero=list(filter(lambda ele: ele !=0.0,collumn))#πίνακας στήλης j του A
χωρίς τα μηδενικά στοιχεία
        if not Anonzero:
            s.append(0.0)
            continue
                s.append(pow((max(Anonzero)*min(Anonzero)),-1/2))#υπολογισμός του r
πολλαπλασιάζοντας το μεγαλύτερο με το μικρότερο στοιχείο κάθε στήλης και υψώνοντας
το γινόμενο στην -1/2)
    for i in range(len(metav)):
        for j in range((len(rows)-1)):
                A[j][i]=float(A[j][i])*float(s[i])γέμισμα πίνακα A μετά τον πολλαπλασιασμό
κάθε στοιχείου του με το αντίστοιχο στοιχείο του πίνακα s
                c[i]= float(c[i]*s[i])# γέμισμα πίνακα c μετά τον πολλαπλασιασμό με το
αντίστοιχο s
    print(cost," z=",end="")#εκτύπωση της αντικειμενικής συνάρτησης
    for i in range(len(c)):
        if float(c[i])==0.0:
            continue
        else:
            print("+",c[i],"*",metav[i],end="")
    print("\n""s.t: ",end="")#εκτύπωση των περιορισμών
    for i in range(len(A)):
        if sum(A[i])==0:
            continue
        for j in range(len(metav)):
            if A[i][j]==0.0:
                continue
            else:
                print("+",A[i][j],"*",metav[j],end="")
```

```
        print(Ab[i] ," ",b[i])
        print("\n")
```

3.6 IBM MPSX

```
iterations=0#αριθμός επαναλήψεων
      while iterations<4: #βρόγχος ελέγχου για αριθμό επαναλήψεων
        sumc=0
        sumk=0
        for i in range(len(A)):
            a1 =[abs(float(element)) for element in A[i]]#αντικατάσταση κάθε στοιχείου του Α με την απόλυτη τιμή του.
          A[i]=a1
          Anonzero=list(filter(lambda ele: ele !=0.0,a1))#πίνακας γραμμής i του Α χωρίς τα μηδενικά στοιχεία
          if iterations==0:
            Z+=len(Anonzero)#αριθμός μη μηδενικών στοιχείων στον πίνακα Α
          if not Anonzero:
            r.append(0.0)
            continue
             r.append(pow((max(Anonzero)*min(Anonzero)),-1/2))#υπολογισμός του r πολλαπλασιάζοντας το μεγαλύτερο με το μικρότερο στοιχείο κάθε γραμμής και υψώνοντας το γινόμενο στην -1/2)


        for i in range(len(A)):
          for j in range(len(A[i])):
                          A[i][j]=float(A[i][j])*float(r[i])#γέμισμα πίνακα Α μετά τον πολλαπλασιασμό κάθε στοιχείου του με το αντίστοιχο στοιχείο του πίνακα r
          b[i]*=r[i]# γέμισμα πίνακα b μετά τον πολλαπλασιασμό με το αντίστοιχο r
        for j in range(len(metav)):
          collumn=[item[j] for item in A]#η στήλη που εξετάζεται σε κάθε επανάληψη
            Anonzero=list(filter(lambda ele: ele !=0.0,collumn))#πίνακας στήλης j του Α χωρίς τα μηδενικά στοιχεία
          if not Anonzero:
            s.append(0.0)
```

```
                continue
                 s.append(pow((max(Anonzero)*min(Anonzero)),-1/2))#υπολογισμός του s πολλαπλασιάζοντας το μεγαλύτερο με το μικρότερο στοιχείο κάθε στήλης και υψώνοντας το γινόμενο στην -1/2)
        for i in range(len(metav)):
           for j in range((len(rows)-1)):
                                A[j][i]=float(A[j][i])*float(s[i]) #γέμισμα πίνακα A μετά τον πολλαπλασιασμό κάθε στοιχείου του με το αντίστοιχο στοιχείο του πίνακα s
              sumc+=pow(float(A[j][i]),2)
              sumk+=abs(float(A[j][i]))
           c[i]= float(c[i]*s[i])

          if 1/Z*(sumc-pow(sumk,2)/Z)<10:#Ελεγχος αν ισχύει η συνθήκη. Αν ισχύει το πρόγραμμα βγαίνει από την επανάληψη.
              break
        iterations+=1#αύξηση του μετρητή των επαναλήψεων
#συνέχεια με την τεχνική equilibration.
      r=[] #r array
      s=[] #s array
 for i in range(len(A)):
          a1 =[abs(float(element)) for element in A[i]]#αντικατάσταση κάθε στοιχείου του A με την απόλυτη τιμή του.
        A[i]=a1
            Anonzero=list(filter(lambda ele: ele !=0.0,a1))#αποθηκεύονται στον πίνακα Anonzero όλα τα μη μηδενικά στοιχεία της γραμμής ί του πίνακα A
        if not Anonzero:
           r.append(0.0)
           continue
            r.append(1/max(Anonzero))#γέμισμα r με το αντίστροφο του μεγαλύτερου μη μηδενικού στοιχείου κάθε γραμμής
      for i in range(len(A)):
        for j in range(len(A[i])):
            A[i][j]=float(A[i][j])*float(r[i]) #γέμισμα πίνακα A μετά τον πολλαπλασιασμό
```

```
κάθε στοιχείου του με το αντίστοιχο στοιχείο του πίνακα r
        b[i]*=r[i]# γέμισμα πίνακα b μετά τον πολλαπλασιασμό με το αντίστοιχο r
    for j in range(len(metav)):
        collumn=[item[j] for item in A]#η στήλη που εξετάζεται σε κάθε επανάληψη
            Anonzero=list(filter(lambda ele: ele !=0.0,collumn))#πίνακας στήλης j του A
χωρίς τα μηδενικά στοιχεία
        if not Anonzero:
            s.append(0.0)
            continue
            s.append(1/max(Anonzero))#γέμισμα s με το αντίστροφο του μεγαλύτερου μη
μηδενικού στοιχείου κάθε στήλης.
    for i in range(len(metav)):
        for j in range((len(rows)-1)):
            A[j][i]=float(A[j][i])*float(s[i])#γέμισμα πίνακα A μετά τον πολλαπλασιασμό
κάθε στοιχείου του με το αντίστοιχο στοιχείο του πίνακα s
                c[i]= float(c[i]*s[i])# γέμισμα πίνακα c μετά τον πολλαπλασιασμό με το
αντίστοιχο s
    print(cost," z=",end="")#εκτύπωση της αντικειμενικής συνάρτησης
    for i in range(len(c)):
        if float(c[i])==0.0:
            continue
        else:
            print("+",c[i],"*",metav[i],end="")
    print("\n""s.t: ",end="")#εκτύπωση των περιορισμών
    for i in range(len(A)):
        if sum(A[i])==0:
            continue
        for j in range(len(metav)):
            if A[i][j]==0.0:
                continue
            else:
                print("+",A[i][j],"*",metav[j],end="")
        print(Ab[i] ," ",b[i])
```

```
        print("\n")
```

3.7 LP Norm(p=1,2,oo)

p=1

```
Aabs=[[0 for j in range(len(metav))] for i in range(len(rows)-1)]
    for i in range(len(A)):
        for j in range(len(A[i])):
            Aabs[i][j]=abs(float(A[i][j]))
    for i in Aabs:
        medians=[]#πίνακας που θα αποθηκεύονται οι median κάθε γραμμής
        for j in i:
            if float(j)!=0.0:
                medians.append(j)#προσθήκη στον πίνακα κάθε μη μηδενικό στοιχείο της γραμμής i
        if not medians:
            r.append(0.0)
        else:
            r.append(1/median(medians))#εισαγωγή στον πίνακα r τον αντίστροφο του median κάθε γραμμής.
    for i in range(len(A)):
        for j in range(len(A[i])):
            A[i][j]=float(A[i][j])*float(r[i])γέμισμα πίνακα A μετά τον πολλαπλασιασμό κάθε στοιχείου του με το αντίστοιχο στοιχείο του πίνακα r
            b[i]*=r[i]γέμισμα πίνακα b μετά τον πολλαπλασιασμό του με το αντίστοιχο στοιχείο του πίνακα r
    for i in range(len(A)):
        for j in range(len(A[i])):
            Aabs[i][j]=abs(float(A[i][j]))
    for i in range(len(metav)):
        medians=[]
        for j in range(len(rows)-1):
            if float(Aabs[j][i])==0.0:
```

```
                continue
            else:
                medians.append(Aabs[j][i])
        s.append(1/median(medians)) #εισαγωγή στον πίνακα s τον median κάθε στήλης του A.
    for i in range(len(metav)):
        for j in range((len(rows)-1)):
            A[j][i]=float(A[j][i])*float(s[i])γέμισμα πίνακα A μετά τον πολλαπλασιασμό κάθε στοιχείου του με το αντίστοιχο στοιχείο του πίνακα s
        c[i]= float(c[i]*s[i])γέμισμα πίνακα c μετά τον πολλαπλασιασμό κάθε στοιχείου του με το αντίστοιχο στοιχείο του πίνακα s
    print(cost," z=",end="")
    for i in range(len(c)):
        if float(c[i])!=0.0:
                print("+",c[i],"*",metav[i],end="") # εκτύπωση της αντικειμενικής συνάρτησης
    print("\n""s.t: ",end="")
    for i in range(len(A)):
        if sum(A[i])==0:
            continue
        for j in range(len(metav)): # εκτύπωση των περιορισμών
            if A[i][j]==0.0:
                continue
            else:
                print("+",A[i][j],"*",metav[j],end="")
        print(Ab[i] ," ",b[i])
        print("\n")
```

p=2

```
Aabs=[[0 for j in range(len(metav))] for i in range(len(rows)-1)]
    for i in range(len(A)):
        for j in range(len(A[i])):
            Aabs[i][j]=abs(float(A[i][j]))#πίνακας με τις απόλυτες τιμές του πίνακα A
```

```
for i in Aabs:
    count=0
    su=1
    if not i:
        r.append(0.0)
        continue
    for j in i:
        if float(j)!=0.0:
            count+=1
            su*=abs(float(j))

    if count!=0 and su!=0.0:
        r.append(pow(su,-1/count))#υπολογισμός πίνακα r
    else:
        r.append(0.0)
for i in range(len(A)):
    for j in range(len(A[i])):
        Aabs[i][j]=float(Aabs[i][j])*float(r[i])
    b[i]*=r[i]# πολλαπλασιασμός του πίνακα b με το αντίστοιχο στοιχείο του πίνακα r

for i in range(len(metav)):
    count=0
    su=1
    for j in range(len(rows)-1):
        if float(Aabs[j][i])==0.0:
            continue
        else:
            count+=1
            su*=Aabs[j][i] #πολλαπλασιαμός με κάθε στοιχείο της στήλης j
    if count!=0 and su!=0.0:
        s.append(pow(su,-1/count))#υπολογισμός πίνακα s
    else:
        s.append(0.0)
```

```
    for i in range(len(metav)):
        for j in range((len(rows)-1)):
            Aabs[j][i]=float(Aabs[j][i])*float(s[i]) #πολλαπλασιασμός πίνακα Aabs με τον πίνακα s
        c[i]= float(c[i]*s[i])
    print(cost," z=",end="")#εκτύπωση της αντικειμενικής συνάρτησης.
    for i in range(len(c)):
        if float(c[i])!=0.0:
            print("+",c[i],"*",metav[i],end="")
    print("\n""s.t: ",end="")#εκτύπωση των περιορισμών
    for i in range(len(A)):
        if not Aabs[i]:
            continue
        for j in range(len(metav)):
            if Aabs[i][j]==0.0:
                continue
            else:
                print("+",Aabs[i][j],"*",metav[j],end="")
        print(Ab[i] ," ",b[i])
        print("\n")
```

p=∞

```
for i in range(len(A)):
        a1 =[abs(float(element)) for element in A[i]]#λίστα που περιέχει την απόλυτη τιμή των στοιχείων του A σε κάθε γραμμή
        A[i]=a1
            Anonzero=list(filter(lambda ele: ele !=0.0,a1))#αποθηκεύονται στον πίνακα Anonzero όλα τα μη μηδενικά στοιχεία της γραμμής ί του πίνακα A
        if not Anonzero:
            r.append(0.0)
            continue
            r.append(pow((max(Anonzero)*min(Anonzero)),-1/2))          #υπολογισμός r
```

```
πολλαπλασιάζοντας το μεγαλύτερο και το μικρότερο μη μηδενικό στοιχείο κάθε γραμμής και υψώνοντας τα στην -1/2
    for i in range(len(A)):
        for j in range(len(A[i])):
            A[i][j]=A[i][j]*r[i]
              b[i]*=r[i]#πολλαπλασιασμός κάθε στοιχείου του πίνακα b με το αντίστοιχο στοιχείο του πίνακα r.
    for j in range(len(metav)):
        collumn=[item[j] for item in A]
        Anonzero=list(filter(lambda ele: ele !=0.0,collumn))
        if not Anonzero:
            s.append(0.0)
            continue
             s.append(pow((max(Anonzero)*min(Anonzero)),-1/2))        #υπολογισμός s πολλαπλασιάζοντας το μεγαλύτερο και το μικρότερο μη μηδενικό στοιχείο κάθε στήλης και υψώνοντας τα στην -1/2
    for i in range(len(metav)):
        for j in range((len(rows)-1)):
            A[j][i]=float(A[j][i])*float(s[i])
              c[i]= float(c[i]*s[i])#πολλαπλασιασμός κάθε στοιχείου του πίνακα c με το αντίστοιχο στοιχείο του πίνακα s.
    print(cost," z=",end="")#εκτύπωση της αντικειμενικής συνάρτησης
    for i in range(len(c)):
        if float(c[i])==0.0:
            continue
        else:
            print("+",c[i],"*",metav[i],end="")
    print("\n""s.t: ",end="")#εκτύπωση των περιορισμών
    for i in range(len(A)):
        if sum(A[i])==0:
            continue
        for j in range(len(metav)):
            if A[i][j]==0.0:
```

```
            continue
        else:
            print("+",A[i][j],"*",metav[j],end="")
    print(Ab[i] ," ",b[i])
    print("\n")
```

4 Υπολογιστική Μελέτη

Στο κεφάλαιο αυτό, παρατίθενται αναλυτικά οι χρόνοι υλοποίησης κάθε τεχνικής κλιμάκωσης για κάθε ένα απο τα 136 γραμμικά προβλήματα ξεχωριστά. Επειδή, η τεχνική του αριθμητικού μέσου και η τεχνική της εντροπίας ακολουθούν την ίδια μεθοδολογία, οι χρόνοι υλοποίησης τους έχουν ενοποιηθεί. Το ίδιο ισχύει και για τις τεχνικές de Buchet για την περίπτωση p=∞ , του γεωμετρικού μέσου και LP norm για την περίπτωση του p=∞. Οι χρόνοι είναι σε δευτερόλεπτα.

Πρόβλημα	Arithmetic mean-entropy	de Buchet p=1	de Buchet p=2	equilibration	de Buchet p=∞ geometricmean-LP norm p=∞	LP norm p=1	LP norm p=2	IBMX
AFIRO	0.0781	0.1719	0.1417	0.1094	0.1422	0.1887	0.0910	0.0937
adlittle	0.2198	0.437	0.4755	0.2355	0.3689	0.4888	0.2355	0.2343
agg	2.5855	2.6719	2.7714	2.8200	3.2183	3.0330	3.3375	2.9124
BANDM	1.8508	3.5348	3.6463	2.0556	3.4977	3.3932	2.2758	3.1953
BNL1	6.7527	12.682	13.341	7.6839	10.472	13.256	9.5637	9.1138
stair	2.0733	4.2330	4.2342	2.4852	2.9529	3.8770	2.5746	3.5905
stocfor1	0.1290	0.2816	0.3286	0.1354	0.1406	0.2372	0.1260	0.1727
STOCFOR 2	23.185	43.756	44.343	26.786	37.934	48.035	33.448	36.906
agg2	2.4830	4.2948	4.6008	2.6027	3.6310	4.4881	3.3791	3.1805
agg3	2.4510	4.1507	4.7274	2.5458	3.5752	4.0744	3.2608	3.1576
stocfor3	2374.5	2537.2	2716.5	2521.8	2526.9	3351.4	4357.3	3329.1
beaconfd	0.7417	1.5981	1.4870	0.8041	1.1134	1.2229	1.1385	1.2514
brandy	0.6405	1.1134	1.1543	0.6537	0.9425	1.1020	0.8448	0.8393

Πρόβλημα	Arithmetic mean-entropy	de Buchet p=1	de Buchet p=2	equilibration	de Buchet p=∞ geometricmean-LP norm p=∞	LP norm p=1	LP norm p=2	IBMX
degen2	2.2666	3.7845	4.4340	2.5283	3.6816	4.2474	3.2120	3.1094
degen3	36.428	68.303	65.863	42.818	62.719	69.550	48.876	49.105
e226	0.8617	1.4368	1.6254	0.8450	1.2136	1.4765	1.1588	1.1089
fffff800	3.9200	7.4336	7.3741	4.3356	6.4188	7.3534	5.2315	5.5246
israel	0.4978	0.9270	1.0388	0.5080	0.7034	1.0111	0.7781	0.8115
lotfi	0.3477	0.7423	0.6478	0.3293	0.4720	0.6015	0.4512	0.5010
maros	10.054	18.228	18.674	11.151	16.778	19.354	13.490	13.992
sc50a	0.0469	0.0822	0.1388	0.0500	0.0627	0.1009	0.0727	0.0661
sc50b	0.0312	0.0468	0.0904	0.0312	0.0468	0.0468	0.0468	0.0468
sc105	0.0781	0.1577	0.2580	0.0781	0.1250	0.1730	0.2198	0.1249
sc205	0.2354	0.5949	0.6049	0.2668	0.3918	0.4507	0.3351	0.3917
scagr7	0.1527	0.3174	0.3283	0.1928	0.2200	0.3361	0.3343	0.2173
scagr25	1.5084	2.3836	2.8528	1.5640	2.0486	2.9335	1.8440	2.1806
scfxm1	1.2535	2.0871	2.4036	1.3059	1.9723	2.3838	1.6651	1.7422
scfxm2	4.3285	8.0534	8.4368	4.7212	7.3171	8.2906	6.0810	6.3010
scfxm3	9.4060	17.799	17.559	10.586	15.918	18.346	12.830	13.838
scorpion	1.0462	1.7471	2.0806	1.0851	1.4015	1.8659	1.2963	1.4832
sctap1	1.0342	1.7386	2.0257	1.1240	1.5564	1.7980	1.2670	1.4656
sctap2	12.499	23.390	23.430	13.961	20.029	24.777	16.546	18.636
sctap3	21.916	41.711	39.917	24.874	43.776	43.993	30.903	33.378
ship04l	6.6942	12.939	12.286	7.7547	11.025	13.327	9.1876	9.3954
ship04s	3.9819	7.7478	7.2621	4.5054	6.8071	8.2493	5.9076	5.9019
ship08l	25.530	48.482	45.823	29.849	47.167	51.319	34.202	37.335
ship08s	11.616	21.951	21.584	13.412	20.754	24.469	15.976	17.581
ship12l	45.763	84.009	80.030	52.859	84.752	92.413	59.962	66.203
ship12s	18.827	35.183	33.236	21.146	31.268	39.068	25.570	28.261
tuff	1.9440	3.7771	3.6801	2.1592	3.2532	3.9140	2.7493	4.0100
wood1p	39.407	77.014	70.045	47.942	69.875	76.666	48.793	53.167

Πρόβλημα	Arithmetic mean-entropy	de Buchet p=1	de Buchet p=2	equilibration	de Buchet p=∞ geometricmean-LP norm p=∞	LP norm p=1	LP norm p=2	IBMX
woodw	97.974	171.53	158.43	108.59	172.25	183.60	119.01	154.20
aa3	187.07	312.97	290.04	198.28	327.56	315.51	204.98	206.64
aa4	95.256	166.79	159.52	108.48	159.54	170.46	111.02	110.57
aa5	164.11	275.17	260.48	178.94	253.34	286.88	188.47	188.08
aa6	102.76	187.32	176.85	122.91	161.61	195.27	130.79	128.73
aircraft	149.71	263.68	248.18	172.67	229.50	298.50	203.17	235.98
car4	3866.3	7523.8	7571.6	4446.9	5399.2		5039.8	6844.8
farm	31.529	60.299	36.411	22.216	22.485	0.0312	49.180	37.471
jendrec1	203.32	377.47	344.15	235.50	229.57	280.71	258.37	320.23
lp22	445.42	759.14	675.35	475.48	455.03	554.99	546.96	569.93
p05	458.01	779.90	685.34	491.08	474.89	564.41	567.36	622.41
pgp2	188.88	342.53	212.73	204.92	193.21	253.25	248.14	297.86
pltexpa2-6	6.2110	11.191	6.9155	6.7287	6.4877	9.5137	8.3477	9.8054
pltexpa2-16	35.449	65.191	40.501	40.064	37.564	50.760	47.797	59.879
pltexpa3-6	243.30	438.34	285.98	275.61	261.77	333.42	319.10	399.98
rosen1	13.944	26.137	15.761	15.719	15.336	18.831	18.498	16.829
rosen7	2.5456	4.9136	3.0112	3.0176	2.7592	3.2447	3.3174	3.3319
rosen8	9.5107	17.816	10.872	10.937	10.477	12.520	12.126	12.110
sc205-2r-4	0.0927	0.2402	0.1118	0.0973	0.0946	0.1093	0.1042	0.1233
sc205-2r-8	0.2187	0.5011	0.2510	0.2199	0.1719	0.2668	0.2511	0.3292
sc205-2r-16	0.6886	1.3178	0.8328	0.7599	0.6262	0.8806	0.8446	1.1238
sc205-2r-27	1.7208	3.3966	2.0707	1.8294	1.6239	2.2718	2.1774	2.8441
sc205-2r-32	2.3575	4.4105	2.7571	2.5193	2.2613	3.1492	3.0237	3.9183
sc205-2r-50	5.5609	9.7135	6.3189	5.8274	5.4028	7.2159	6.9060	9.3574
sc205-2r-64	8.4606	15.489	10.140	9.4321	8.6847	11.518	11.072	14.721
sc205-2r-100	20.415	36.826	23.006	22.158	20.833	27.889	27.000	34.464
sc205-2r-200	81.123	147.18	90.511	88.412	83.125	112.97	106.44	138.01

Πρόβλημα	Arithmetic mean-entropy	de Buchet p=1	de Buchet p=2	equilibration	de Buchet p=∞ geometricmean-LP norm p=∞	LP norm p=1	LP norm p=2	IBMX
sc205-2r-400	355.52	445.89	393.54	388.31	363.30	485.49	466.89	617.27
scagr7-2b-4	1.4046	1.3972	1.2336	1.3336	1.2625	3.3028	2.4490	1.7616
scagr7-2b-16	2.4781	2.5309	2.7898	2.6710	2.3104	2.9934	2.9097	3.7851
scagr7-2b-64	591.91	651.13	649.66	652.87	709.22	764.64	745.25	952.13
scagr7-2c-4	1.7868	1.7651	1.5764	1.6774	2.1015	3.3706	2.5499	2.2947
scagr7-2c-16	2.4622	2.5085	2.8425	2.7058	3.1805	3.0369	2.9598	3.7379
scagr7-2c-64	32.455	36.737	37.565	36.753	40.786	44.882	43.007	52.834
scagr7-2r-4	0.3079	0.3440	0.3404	0.3137	0.3929	0.4545	0.4197	0.4250
scagr7-2r-8	0.7193	0.7354	0.8477	0.7866	0.7896	0.8795	0.8718	1.0990
scagr7-2r-16	2.4772	2.5527	2.7589	2.5694	2.7230	3.1380	3.0631	3.7430
scagr7-2r-27	6.1805	6.7232	7.1964	6.7723	8.1931	8.2580	7.9827	9.7766
scagr7-2r-32	8.6903	9.3537	9.9420	9.4956	10.836	11.605	11.552	13.558
scagr7-2r-54	23.360	26.310	26.795	26.786	31.277	32.605	31.111	37.062
scagr7-2r-64	32.740	37.289	37.458	37.021	43.910	45.149	44.071	52.284
scagr7-2r-108	95.403	107.92	104.28	105.56	120.92	129.86	128.21	147.81
scfxm1-2b-4	4.3258	4.6702	4.7369	4.5480	5.0939	5.9246	5.4913	6.4219
scfxm1-2b-16	48.472	55.234	54.922	54.291	61.819	67.022	64.183	76.530
scfxm1-2c-	4.1084	4.5475	4.7520	4.3112	4.8192	5.6012	5.3279	6.2784

Πρόβλημα	Arithmetic mean-entropy	de Buchet p=1	de Buchet p=2	equilibration	de Buchet p=∞ geometricmean-LP norm p=∞	LP norm p=1	LP norm p=2	IBMX
4								
scfxm1-2r-4	3.9778	4.4211	4.6984	4.2978	5.5543	5.4537	5.1524	6.1809
scfxm1-2r-8	13.390	14.955	14.992	14.665	19.016	18.294	17.426	20.541
scfxm1-2r-16	49.082	55.665	54.829	54.183	71.411	66.872	65.059	77.087
scfxm1-2r-27	136.42	154.51	154.23	151.24	192.22	185.38	177.50	215.78
scfxm1-2r-32	196.29	221.49	216.57	212.66	277.99	261.41	250.07	302.99
scrs8-2b-4	0.7162	0.7129	0.6349	0.6768	0.9035	1.2237	1.0490	0.8820
scrs8-2b-16	1.7193	1.8149	1.9932	1.8116	1.7828	2.1337	2.0543	2.6177
scrs8-2b-64	21.528	23.958	23.730	23.809	27.453	29.540	28.351	34.428
scrs8-2c-4	0.2348	0.2620	0.2688	0.2877	0.3040	0.3645	0.3278	0.3356
scrs8-2c-16	1.6655	1.8451	1.9933	1.7659	2.0885	2.1782	2.0560	2.6815
scrs8-2c-32	5.7651	6.2095	6.6492	6.1266	6.5541	7.6943	7.3113	9.2161
scrs8-2c-64	21.638	24.024	23.777	23.817	29.122	29.579	28.114	34.286
scrs8-2r-4	0.2425	0.2640	0.2835	0.2472	0.3275	0.3573	0.3016	0.3292
scrs8-2r-8	0.5230	0.5824	0.5888	0.5858	0.8022	0.6442	0.6494	0.8010
scrs8-2r-16	1.7022	1.7725	1.9265	1.7703	2.3472	2.1490	2.0301	2.6220
scrs8-2r-27	4.2428	4.7369	4.8270	4.4592	6.0748	5.6432	5.5321	6.6530
scrs8-2r-32	5.7527	6.3775	6.3996	6.1097	8.5156	7.7274	7.3566	9.2325
scrs8-2r-64	21.521	24.745	23.699	23.963	30.108	29.292	28.515	34.402
scrs8-2r-64b	21.584	23.759	23.877	23.888	33.329	29.435	28.262	34.158
scrs8-2r-128	86.650	97.483	93.687	93.658	142.45	117.23	113.18	139.63
scrs8-2r-256	369.95	414.68	399.73	407.96	617.23	500.93	475.77	610.14

Πρόβλημα	Arithmetic mean-entropy	de Buchet p=1	de Buchet p=2	equilibration	de Buchet p=∞ geometricmean-LP norm p=∞	LP norm p=1	LP norm p=2	IBMX
scsd8-2b-4	1.6011	1.7824	1.5649	1.8161	2.0741	3.7140	2.5513	2.1462
scsd8-2b-16	6.7342	8.0005	8.0981	8.1151	9.2996	9.8817	9.0746	9.8068
scsd8-2c-4	0.6253	0.6519	0.6997	0.6901	0.7498	0.8041	0.7798	0.8296
scsd8-2c-16	6.7955	8.1125	8.1294	8.1906	9.2620	10.219	9.3501	9.8731
scsd8-2r-4	0.5887	0.6374	0.7013	0.6902	0.7195	0.9013	0.7511	0.8598
scsd8-2r-8	1.9225	2.1855	2.2530	2.2549	2.5991	2.7758	2.5741	2.7433
scsd8-2r-8b	1.9027	2.2305	2.2466	2.3115	2.5713	2.6554	2.5284	2.7285
scsd8-2r-16	6.8931	8.2715	8.0386	8.1564	9.2819	9.9199	9.4398	9.7029
scsd8-2r-27	19.036	22.856	22.413	22.974	25.824	27.139	25.738	27.478
scsd8-2r-32	26.742	32.052	30.992	32.012	36.431	37.935	35.863	38.056
scsd8-2r-54	78.907	91.985	88.842	90.937	122.00	107.46	102.11	108.54
scsd8-2r-64	115.03	129.97	122.67	128.47	199.56	150.77	143.67	152.11
scsd8-2r-108	360.08	384.67	370.37	379.74	594.96	452.44	433.84	449.79
sctap1-2b-4	1.2930	1.3693	1.4146	1.4090	1.4615	1.8668	1.6176	1.6717
sctap1-2b-16	9.6084	10.890	10.837	10.537	12.170	13.008	12.459	14.305
sctap1-2c-4	0.8757	0.9289	1.0010	0.9252	0.9675	1.0323	1.0175	1.2511
sctap1-2c-16	9.5093	10.809	10.882	10.596	14.678	12.880	12.265	14.223
sctap1-2c-64	115.14	130.41	127.48	125.04	203.04	155.29	148.35	170.26
sctap1-2r-4	1.1274	1.1818	1.1787	1.1768	1.7260	1.5453	1.4082	1.5526
sctap1-2r-8	2.7701	3.0209	3.1101	2.9495	4.7133	3.4776	3.3498	4.0001
sctap1-2r-8b	10.998	2.9672	9.6802	9.7843	8.3125	41.570	22.888	13.192
sctap1-2r-16	9.4931	10.891	10.997	10.549	14.206	13.081	12.461	14.437
sctap1-2r-	26.588	30.317	29.738	29.378	43.818	36.408	34.922	40.048

Πρόβλημα	Arithmetic mean-entropy	de Buchet p=1	de Buchet p=2	equilibration	de Buchet p=∞ geometricmean-LP norm p=∞	LP norm p=1	LP norm p=2	IBMX
27								
sctap1-2r-32	36.876	42.333	41.865	41.190	60.105	49.935	48.668	55.515
sctap1-2r-54	107.71	122.14	117.94	117.57	203.26	142.24	137.63	159.85
sctap1-2r-64	152.56	172.66	165.40	162.85	348.74	207.75	194.16	220.85
sctap1-2r-108	453.79	504.67	492.96	486.93	793.85	884.40	567.90	672.42
sctap1-2r-216	2262.0	2590.5	2628.1	2451.0	3042.9	4973.6	2761.7	3401.8
zed	7.8268	8.9216	11.311	7.6529	9.5160	64.678	14.086	9.8043

5 Επίλογος

5.1 Σύνοψη και συμπεράσματα

Η κλιμάκωση είναι μια μέθοδος που εφαρμόζεται στα μεγάλα προβλήματα γραμμικού προγραμματισμού για να φέρει το γραμμικό πρόβλημα στην καλύτερη δυνατή κατάσταση για να μπορεί το πρόγραμμα να επιλυθεί πιο εύκολα από έναν lp-solver.Η κλιμάκωση εφαρμόζεται σε γραμμικά προβλήματα για 3 λόγους:i) μια συμπαγής αναπαράσταση των ορίων των μεταβλητών, ii) μείωση του αριθμού των επαναλήψεων που απαιτούνται για την επίλυση του γραμμικού προβλήματος, iii)βελτίωση της αριθμητικής συμπεριφοράς των αλγορίθμων. Στην εργασία αυτή παρουσιάστηκαν 11 τεχνικές κλιμάκωσης και αναλύθηκαν με βάση την αριθμητική τους μορφή, ένα αναλυτικό αριθμητικό παράδειγμα και την εισαγωγή και υλοποίηση τους με την γλώσσα προγραμματισμού Python. Έπειτα, πραγματοποιήθηκε μια υπολογιστική μελέτη ανάμεσα σε 136 γνωστά γραμμικά προβλήματα για να εξακριβωθεί αν είναι δυνατή η υλοποίηση των τεχνικών κλιμάκωσης με την Python αλλά και για να γίνει σύγκριση της υλοποίησης των τεχνικών μεταξύ τους. Οι τεχνικές που παρουσιάστηκαν

ήταν: 1)arithmetic mean, 2)de Buchet για την περίπτωση p=1, 3)de Buchet για την περίπτωση p=2, 6)de Buchet για την περίπτωση p=∞,5) entropy 6) equilibration , 7) geometric mean, 8) IBM-MPSX, 9) Lp-Norm για την περίπτωση p=1, 10)Lp-Norm για την περίπτωση p=2, 11) Lp-Norm για την περίπτωση p=∞. Τα συμπεράσματα που προκύπτουν για τις τεχνικές κλιμάκωσης είναι οτι σε μικρά γραμμικά προβλήματα με λίγες μεταβλητές και περιορισμούς, οι χρόνοι υλοποίησης των τεχνικών δεν παρουσιάζουν μεγάλη διαφορά μεταξύ τους. Όσο, όμως μεγαλώνει το μέγεθος του προβλήματος και μεγαλώνει ο αριθμός των μεταβλητών και των περιορισμών παρατηρείται ότι οι τεχνικές arithmetic mean και entropy υλοποιούνται πιο σύντομα , ενώ παρόμοιους αλλά πιο αργούς χρόνους παρατηρούμε και στην τεχνική equilibration.

5.2 Μελλοντικές Επεκτάσεις

Η Python είναι μια γλώσσα προγραμματισμού με πολλές εφαρμογές σε αρκετά επιστημονικά πεδία. Οι χρόνοι υλοποίησης που παρουσιάστηκαν κατά αυτή εδώ την εργασία είναι μια ενθαρυντική αρχή για την αποτελεσματική υλοποίηση των τεχνικών κλιμάκωσης σε μεγάλα προβλήματα γραμμικού προγραμματισμού μέσω αυτής. Ως μελλοντική επέκταση της εργασιας αυτης προτείνεται να γίνει η αντίστοιχη υπολογιστική μελέτη σε έναν υπολογιστή με μεγάλες υπολογιστικές δυνατότητες, για να είναι δυνατή η σύγκριση των χρόνων υλοποίησης που θα προκύψουν με αντίστοιχους χρόνους από άλλα γνωστά λογισμικά.Επιπλέον, ως επέκταση μπορεί να θεωρηθεί και η υλοποίηση των τεχνικών κλιμάκωσης σε GPU και η διεξαγωγή αντίστοιχης υπολογιστικής μελέτης για να παρατηρηθεί ποια τεχνική κλιμάκωσης είναι πιο αποτελεσματική καθώς και για να γίνει σύγκριση, εκτίμηση και αναπαράσταση των χρόνων υλοποίησης όλων των τεχνικών μεταξύ τους.

6 Βιβλιογραφία

Ελληνική Βιβλιογραφία

Δαρζέντας Ε.(1999). Επιχειρησιακή Έρευνα, Πανεπιστήμιο Αιγαίου Τμήμα Μαθηματικών

Κώστογλου Β. (2015). Επιχειρησιακή Έρευνα. Θεσσαλονίκη: Εκδόσεις Τζιόλα.

Λεμεσιανός Α.(2014). Γραμμικός Προγραμματισμός [pdf], available at :

https://apothetirio.teiep.gr/xmlui/bitstream/handle/123456789/1047/fin_20070236.pdf?sequence=1

Ε.Μ.Π. (2006) Εισαγωγη Στην Επιχειρησιακη Ερευνα Γραμμικός Προγραμματισμος,

Σχολή Εφαρμοσμένων Μαθηματικών και Φυσικών Επιστημών [pdf].

Available at: https://www.math.ntua.gr/~coletsos/Documents/linearprogramming.pdf

Ξένη/Ξενόγλωσση Βιβλιογραφία

Ploskas N., Samaras N. (2007) Linear Programming using Matlab®, Springer International Publishing

Ploskas N., Samaras N. (2013) The impact of scaling on simplex type algorithms,ACM International Conference Proceeding Series.

The Editors of Encyclopedia Britannica (1999), Revised: Erik Gregesen (2017), Linear Programming, Britannica,
available at: https://www.britannica.com/science/linear-programming-mathematics.

Tomlin J.A. Mathematical Programming Study 4 (1975) p.146-166, North-Holland Publishing Company