ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ

ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΥΛΟΠΟΙΗΣΗ ΤΟΥ ΑΝΑΘΕΩΡΗΜΕΝΟΥ ΑΛΓΟΡΙΘΜΟΥ SIMPLEX (REVISED SIMPLEX ALGORITHM) ΣΕ PYTHON

ΤΣΕΡΠΙΣΝΟΣ ΓΕΩΡΓΙΟΣ

Επιβλέπων καθηγητής: ΣΑΜΑΡΑΣ ΝΙΚΟΛΑΟΣ

Επισκόπηση της Παρουσίασης

- Θέμα και στόχοι
- Θεωρητικό υπόβαθρο
- *Σχετική Έρευνα*
- *Αλγόριθμοι τύπου Simplex*
- *Υλοποίηση αλγορίθμων* σε γλώσσα Python
- *Υπολογιστική μελέτη*
- *Συμπεράσματα/επεκτάσεις*

Θέμα και στόχοι

- Η διπλωματική εργασία εξετάζει τον τρόπο λειτουργίας κάποιων βασικών αλγορίθμων τύπου Simplex για την επίλυση του γραμμικού προβλήματος και την ανάπτυξη προγραμμάτων σε γλώσσα Python για την υλοποίηση των αλγορίθμων αυτών.
- Περιγραφή των βημάτων από τα οποία αποτελούνται οι διάφορες παραλλαγές των αλγορίθμων τύπου Simplex και των προβλημάτων που μπορούν να προκύψουν κατά την επίλυση ενός γραμμικού προβλήματος.
- Παρουσίαση των τρόπων εύρεσης μίας αρχικής εφικτής λύσης, καθώς και των τεχνικών κλιμάκωσης που μπορούν να χρησιμοποιηθούν κατά την επίλυση των προβλημάτων.
- Σχεδίαση και υλοποίηση αλγορίθμων τύπου Simplex με τη γλώσσα προγραμματισμού Python, δίνοντας έμφαση στον αναθεωρημένο αλγόριθμο Simplex.

Θέμα και στόχοι

- Υλοποίηση συναρτήσεων για την ανάγνωση προβλημάτων που είναι αποθηκευμένα σε αρχεία μορφής mps και συναρτήσεων για τη δημιουργία τέτοιων αρχείων.
- Συλλογή αποτελεσμάτων και υπολογιστική μελέτη των αλγορίθμων που αναπτύχθηκαν.
- Παρουσίαση των αποτελεσμάτων με γραφικό τρόπο.
- Συμπεράσματα και προτάσεις για επέκταση και βελτίωση του λογισμικού που αναπτύχθηκε.

Θεωρητικό υπόβαθρο

- **Κανονική** ή ανισοτική μορφή του γραμμικού προβλήματος:

$$\begin{aligned} &\min\ (\text{ή } \max)\ z = c^T x \\ &\text{μ.π.} \quad Ax \otimes b \\ &x \geq 0 \end{aligned}$$

όπου

- x: πίνακας-στήλη διάστασης nx1 με τις μεταβλητές απόφασης.
- A: πίνακας διάστασης mxn με τους συντελεστές των μεταβλητών στους περιορισμούς.
- c: πίνακας-στήλη διάστασης nx1 με τους συντελεστές των μεταβλητών στην αντικειμενική συνάρτηση.
- b: πίνακας-στήλη διάστασης mx1 με τους σταθερούς όρους στους περιορισμούς.
- m: το πλήθος των περιορισμών.
- n: το πλήθος των μεταβλητών.
- z: η τιμή της αντικειμενικής συνάρτησης.
- το σύμβολο $\otimes$ δηλώνει κάποια από τις σχέσεις: $\geq$ ή $\leq$.

Θεωρητικό υπόβαθρο

- **Τυποποιημένη** ή ισοτική μορφή του γραμμικού προβλήματος:

$$\begin{aligned} &\min z = c^T x \\ &\text{μ.π.} \quad Ax = b \\ &x \geq 0 \end{aligned}$$

- Μετατροπή από πρόβλημα μεγιστοποίησης σε πρόβλημα ελαχιστοποίησης:

$$\max \sum_{i=1}^{n} c_i x_i = -\min \sum_{i=1}^{n} -c_i x_i$$

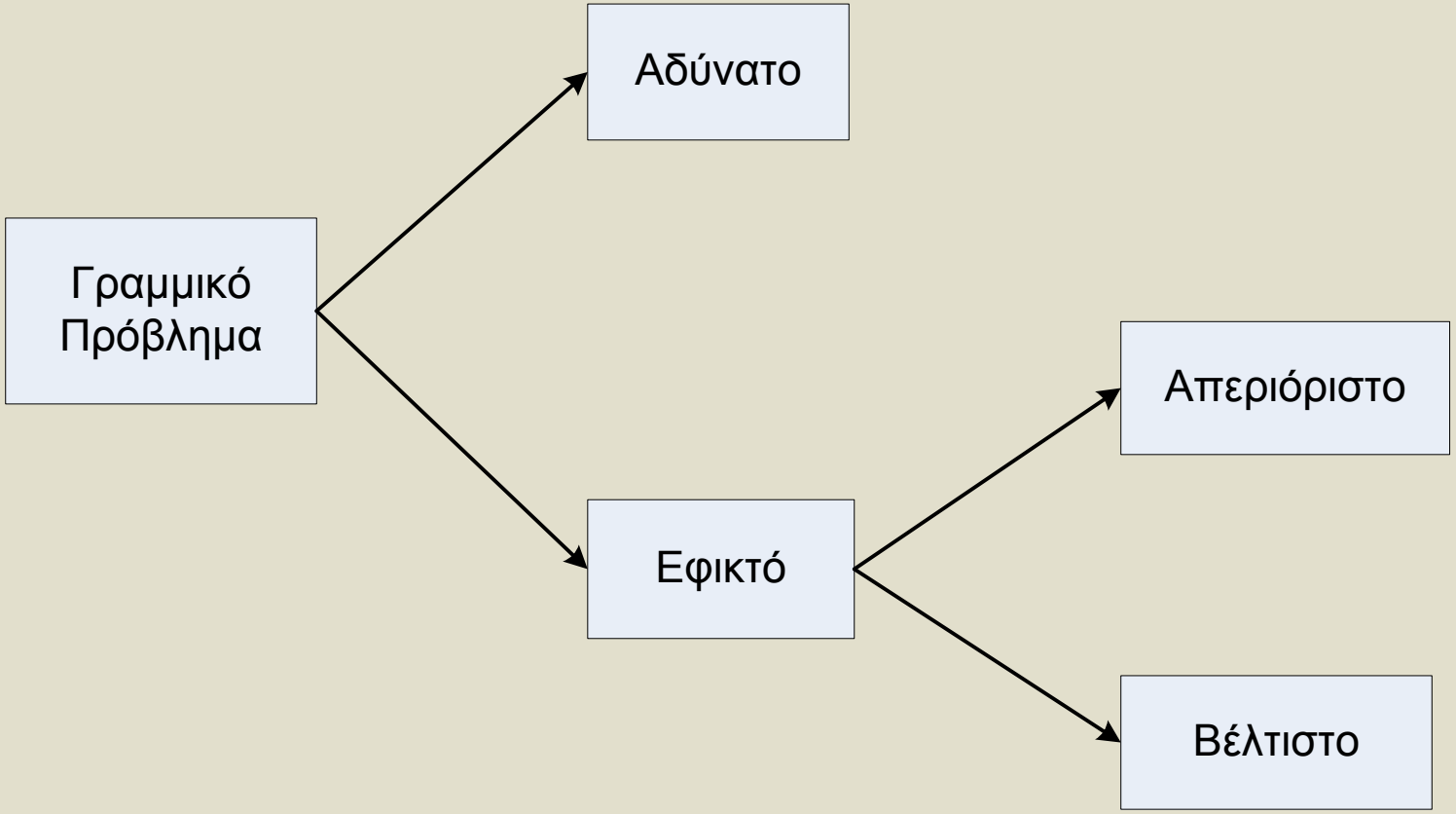
Θεωρητικό υπόβαθρο

- Μετατροπή από κανονική μορφή σε τυποποιημένη μορφή με χρήση **χαλαρών μεταβλητών $s_i \geq 0$:**

$$\sum_{j=1}^{m} \alpha_{ij} x_j \leq b_i \quad \Rightarrow \quad \sum_{j=1}^{m} \alpha_{ij} x_j + s_i = b_i$$

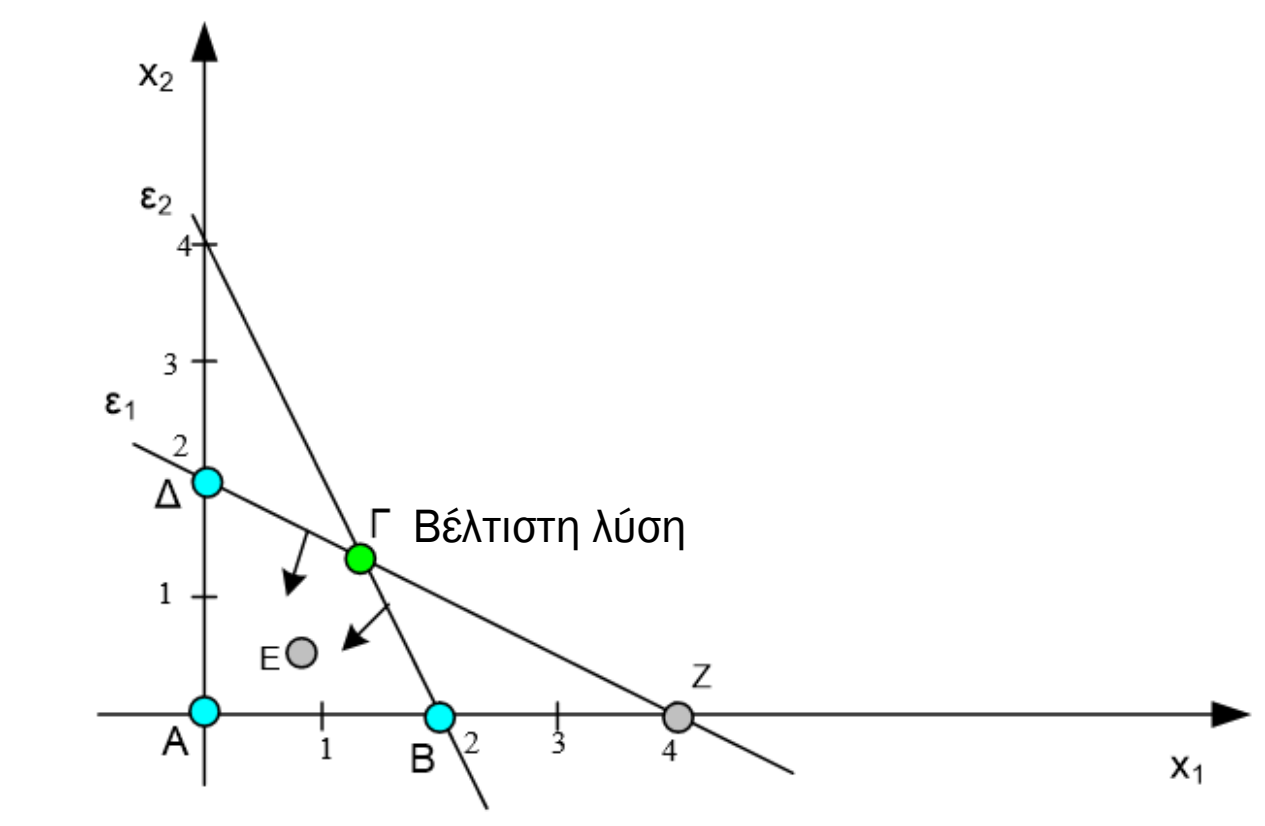
$$\sum_{j=1}^{m} \alpha_{ij} x_j \geq b_i \quad \Rightarrow \quad \sum_{j=1}^{m} \alpha_{ij} x_j - s_i = b_i$$

Θεωρητικό υπόβαθρο



Θεωρητικό υπόβαθρο

Κατηγορίες αλγορίθμων



- Αλγόριθμοι τύπου Simplex
- Αλγόριθμοι εξωτερικών σημείων
- Αλγόριθμοι εσωτερικών σημείων

Θεωρητικό υπόβαθρο

Μετατροπή στο Δυικό πρόβλημα

Ελαχιστοποίηση	↔	Μεγιστοποίηση
min $c^T x$	↔	max $b^T w$
Περιορισμός =	↔	Ελεύθερη μεταβλητή
Περιορισμός ≥	↔	Μεταβλητή ≥ 0
Περιορισμός ≤	↔	Μεταβλητή ≤ 0
Ελεύθερη μεταβλητή	↔	Περιορισμός =
Μεταβλητή ≥ 0	↔	Περιορισμός ≤
Μεταβλητή ≤ 0	↔	Περιορισμός ≥

$c^T x \geq b^T w$ Ασθενές δυικό θεώρημα (Chvatal 1983)

Θεωρητικό υπόβαθρο

Σχέση Δυικού και Πρωτεύοντος προβλήματος

		Δυικό πρόβλημα		
		Βέλτιστο	Αδύνατο	Απεριόριστο
Πρωτεύον πρόβλημα	Βέλτιστο	√		
	Αδύνατο		√	√
	Απεριόριστο		√	

Σχετική έρευνα

Βελτιώσεις αλγορίθμων τύπου Simplex

Μεθοδολογία	Συγγραφέας	Χρονολογία
Tableau μορφή	G. Dantzig	1947
Αναθεωρημένη μορφή	G. Dantzig	
	W. Orchad-Hays	1953-54
	P. Wolfe	
Παραγοντοποίηση Βάσης	H. Markowitz	1954
	L. Beale	1971
	E. Hellerman	1971
	D. Rarick	1972
Τεχνικές ανανέωσης αραιών μητρών	H. Bartels, H. Golub	1969
	H. Forrest, A. Tomlin	1972
	K. Reid	1976
Τεχνικές περιστροφής	J. Harris	1973
	D. Goldfarb, K. Reid	1977
	H. Forrest, D. Goldfarb	1992
Προλυτικές διαδικασίες	L. Brearly, G. Mitra, P. Williams	1975

Σχετική έρευνα

Αλγόριθμοι εσωτερικών σημείων

Θεωρητική ανάλυση	Συγγραφέας	Χρονολογία
Αλγόριθμος λογαριθμικών φραγμών	K. Frisch	1955
Μέθοδος των κέντρων	P. Huard	1967
Αφινικός Κλιμακωτός Αλγόριθμος	I. Dikin	1974
Αλγόριθμος του Karmarkar	N. Karmarkar	1984
Ανάκτηση βέλτιστης λύσης	Y. Ye, M. Kojima	1987
Αλγόριθμος ιχνιλάτης	M. Kojima S. Mizuno A. Yoshise	1989
Ανέφικτοι Αλγόριθμοι	I. Lustig	1990
Αλγόριθμοι ελάττωσης δυναμικού	Y. Ye	1991
Θετικά ημιορισμένος προγραμματισμός	Y. Nesterov A. Nemirowski	1993

Αλγόριθμοι Τύπου Simplex - Γενική Μορφή

Βήμα-0: Αρχικοποίηση (Initialization)

Υπολογισμός όλων των τιμών των μεταβλητών που είναι απαραίτητες για να αρχίσει να εκτελείται ο αλγόριθμος.

Βήμα-1: Έλεγχος βελτιστότητας (Test of Optimality)

Ελέγχεται αν το τρέχον βασικό σημείο αποτελεί τη βέλτιστη λύση. Αν είναι βέλτιστο, τότε ο αλγόριθμος σταματάει.

Βήμα-2: Επιλογή εισερχόμενης και εξερχόμενης μεταβλητής

Με κάποιο κανόνα περιστροφής επιλέγεται η εισερχόμενη και η εξερχόμενη μεταβλητή. Αν δεν μπορεί να γίνει η επιλογή αυτή, τότε το πρόβλημα είναι απεριόριστο.

Βήμα-3: Περιστροφή (Pivoting)

Στο βήμα αυτό υπολογίζεται η νέα βάση, οι τιμές των βασικών μεταβλητών καθώς και οι δυϊκές μεταβλητές, όπως και οι δυϊκές χαλαρές μεταβλητές.

Η διαδικασία επαναλαμβάνεται από το βήμα-1.

Αλγόριθμοι τύπου Simplex

Πρωτεύων Αλγόριθμος Simplex

Βήμα-0: Αρχικοποίηση (Initialization)

(α) Ξεκινάμε με μία εφικτή βασική διαμέριση (B,N)

(β) Υπολογίζουμε τις μήτρες: c_B, c_N, A_B^{-1}, $x_B = A_B^{-1}b$, $w^T = c_B^T A_B^{-1}$, $s_N^T = c_N^T - w^T A_N$.

Βήμα-1: Έλεγχος βελτιστότητας (Test of Optimality)

Υπολογίζουμε το σύνολο $J = \{j: j \in N \text{ και } s_j < 0\}$

Αν είναι $J = \varnothing$ τότε έχει βρεθεί η βέλτιστη λύση και ο αλγόριθμος σταματά.

Βήμα-2: Επιλογή εισερχόμενης και εξερχόμενης μεταβλητής.

(α) **Επιλογή εισερχόμενης μεταβλητής.** Με κάποιο κανόνα περιστροφής επιλέγεται ένα στοιχείο του J, που αντιστοιχεί στη θέση t του συνόλου N και προσδιορίζει την εισερχόμενη μεταβλητή x_l. Δηλαδή $l = N(t)$.

(β) **Επιλογή εξερχόμενης μεταβλητής.** Υπολογίζουμε τον mx1 πίνακα $h_l = (A_B)^{-1} a_l$

όπου a_l η l στήλη του πίνακα A και το σύνολο $I_+ = \{i: 1 \le i \le m \text{ και } h_{il} > 0\}$

Αν είναι $I_+ = \varnothing$ τότε το πρόβλημα είναι απεριόριστο και ο αλγόριθμος σταματά.

Διαφορετικά επιλέγεται δείκτης r που αντιστοιχεί στον **ελάχιστο λόγο**:

$$\frac{(A_B^{-1}b)_r}{h_{ir}} = \min\left\{\frac{(A_B^{-1}b)_i}{h_{il}} : i \in I_+\right\}$$

Ο δείκτης r του ελάχιστου λόγου, προσδιορίζει τη θέση στο σύνολο B για την εξερχόμενη μεταβλητή x_k. Δηλαδή $k = B(r)$.

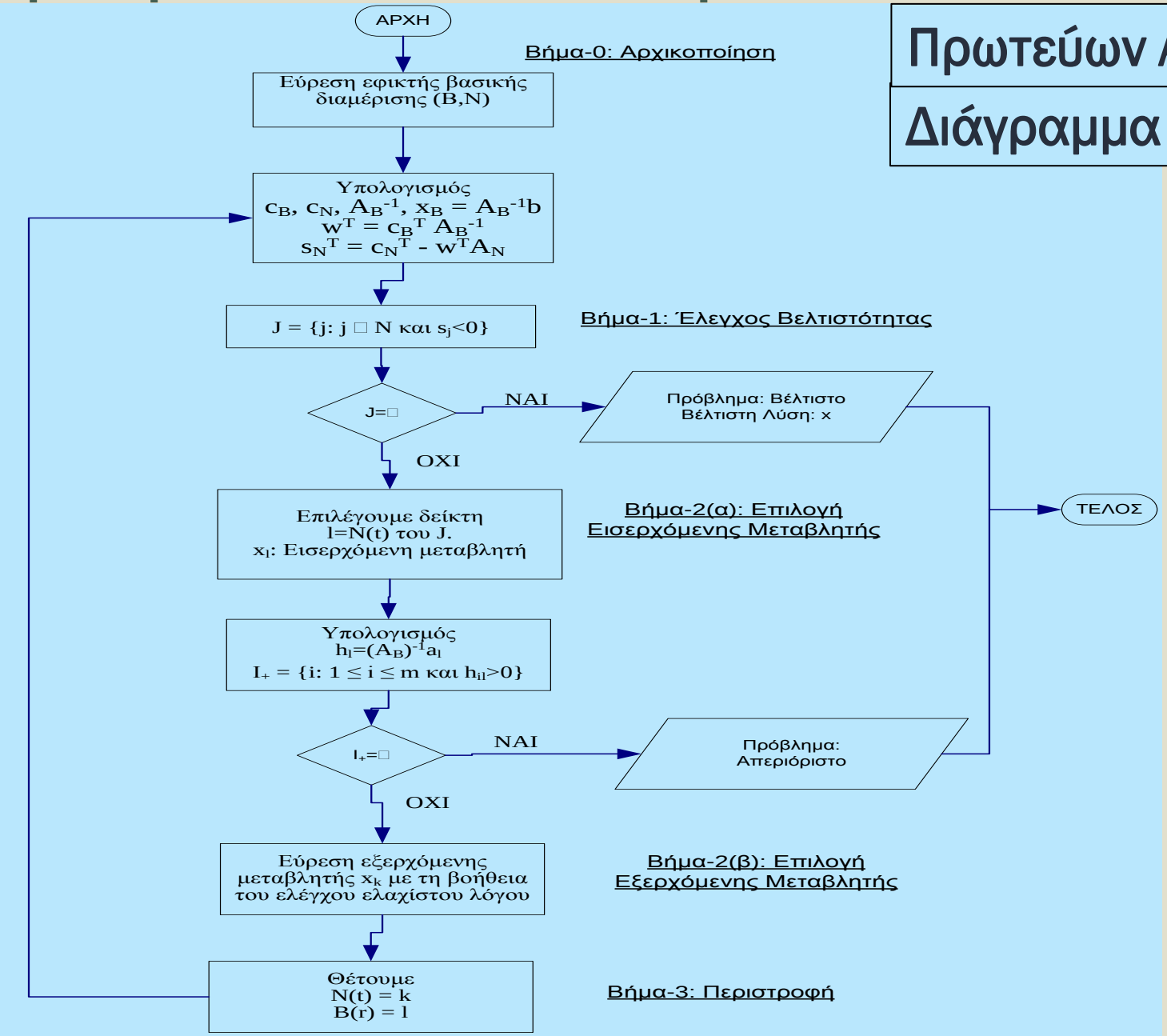
Βήμα-3: Περιστροφή (Pivoting).

Εναλλάσσεται ο ρόλος των μεταβλητών x_l και x_k, θέτοντας $N(t) = k$ και $B(r)=l$. Πηγαίνουμε στο βήμα-0 (β).

Αλγόριθμοι τύπου Simplex

Πρωτεύων Αλγόριθμος Simplex

Διάγραμμα Ροής



Αλγόριθμοι τύπου Simplex

Αναθεωρημένος Αλγόριθμος Simplex

Βήμα-0: Αρχικοποίηση (Initialization)

(α) Ξεκινάμε από εφικτή βασική διαμέριση (B,N). Υπολογίζουμε A_B^{-1}

(β) Υπολογίζουμε τις μήτρες: c_B, c_N, $x_B = A_B^{-1}b$, $w^T = c_B^T A_B^{-1}$, $s_N^T = c_N^T - w^T A_N$.

Βήμα-1: Έλεγχος βελτιστότητας (Test of Optimality)

Υπολογίζουμε το σύνολο $J = \{j: j \in N \text{ και } s_j < 0\}$

Αν $J = \varnothing$ τότε έχει βρεθεί η βέλτιστη λύση. ΤΕΛΟΣ.

Βήμα-2: Επιλογή εισερχόμενης και εξερχόμενης μεταβλητής

(α) **Επιλογή εισερχόμενης μεταβλητής.** Με κάποιο κανόνα περιστροφής επιλέγεται η εισερχόμενη μεταβλητή x_l. Δηλαδή $l = N(t)$.

(β) **Επιλογή εξερχόμενης μεταβλητής.** Υπολογίζουμε τον mx1 πίνακα $h_l = (A_B)^{-1} a_l$

όπου a_l η l στήλη του πίνακα A και το σύνολο $I_+ = \{i: 1 \le i \le m \text{ και } h_{il} > 0\}$

Αν $I_+ = \varnothing$ τότε το πρόβλημα είναι απεριόριστο. ΤΕΛΟΣ.

Διαφορετικά επιλέγεται δείκτης r που αντιστοιχεί στον ελάχιστο λόγο:

$$\frac{(A_B^{-1}b)_r}{h_{ir}} = \min\left\{\frac{(A_B^{-1}b)_i}{h_{il}} : i \in I_+\right\}$$

Ο δείκτης r, δίνει τη θέση της εξερχόμενης μεταβλητής x_k. Δηλαδή $k = B(r)$.

Βήμα-3: Περιστροφή (Pivoting)

Εναλλάσσεται ο ρόλος των μεταβλητών x_l και x_k, θέτοντας $N(t) = k$ και $B(r)=l$.

Ανανέωση τιμής αντίστροφης μήτρας A_B^{-1}: $(A_B^{-1})_{new} = E^{-1} A_B^{-1}$. Πηγαίνουμε στο βήμα-0 (β).

Αλγόριθμοι τύπου Simplex

Αναθεωρημένος Αλγόριθμος Simplex

Για τον υπολογισμό της μήτρας A_B^{-1} χρησιμοποιείται ο τύπος:

$$(A_B^{-1})_{new} = E^{-1} A_B^{-1}$$

όπου

$$E^{-1} = \begin{bmatrix} 1 & 0 & \dots & -\frac{h_{1l}}{h_r} & \dots & 0 \\ 0 & 1 & \dots & -\frac{h_{2l}}{h_r} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & \frac{1}{h_r} & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & -\frac{h_{ml}}{h_r} & \dots & 1 \end{bmatrix}$$

Αλγόριθμοι τύπου Simplex

Δυϊκός Αλγόριθμος Simplex

Βήμα-0: Αρχικοποίηση (Initialization)

(α) Ξεκινάμε με μία δυϊκά εφικτή βασική διαμέριση (B,N)

(β) Υπολογίζουμε τις μήτρες: c_B, c_N, A_B^{-1}, $x_B = A_B^{-1}b$, $w^T = c_B^T A_B^{-1}$, $s_N^T = c_N^T - w^T A_N$.

Βήμα-1: Έλεγχος βελτιστότητας (Test of Optimality)

Αν είναι $x_B \geq 0$ τότε έχει βρεθεί η βέλτιστη λύση και ο αλγόριθμος σταματά.

Βήμα-2: Επιλογή εισερχόμενης και εξερχόμενης μεταβλητής

(α) **Επιλογή εξερχόμενης μεταβλητής.** Επιλέγεται θέση r του συνόλου B που αντιστοιχεί στη βασική μεταβλητή x_k με $x_k<0$ που προσδιορίζει την εξερχόμενη μεταβλητή x_k. Δηλ. k = B(r).

(β) **Επιλογή εισερχόμενης μεταβλητής.** Υπολογίζουμε τον mx1 πίνακα

$$H_r = (A_B)^{-1}{}_r A_N$$

όπου $=(A_B)^{-1}{}_r$ η r γραμμή του πίνακα $=(A_B)^{-1}$

- Αν είναι $H_{rN} \geq 0$ τότε το δυϊκό πρόβλημα είναι απεριόριστο, ενώ το πρωτεύον πρόβλημα είναι αδύνατο.
- Διαφορετικά επιλέγεται τη μη βασική μεταβλητή x_l για την οποία:

$$\frac{-s_l}{h_{rl}} = \min\{\frac{-s_j}{h_{rj}} : h_{rj} < 0\}$$

Η μεταβλητή αυτή είναι η εισερχόμενη και θεωρούμε l = N(t).

Βήμα-3: Περιστροφή (Pivoting)

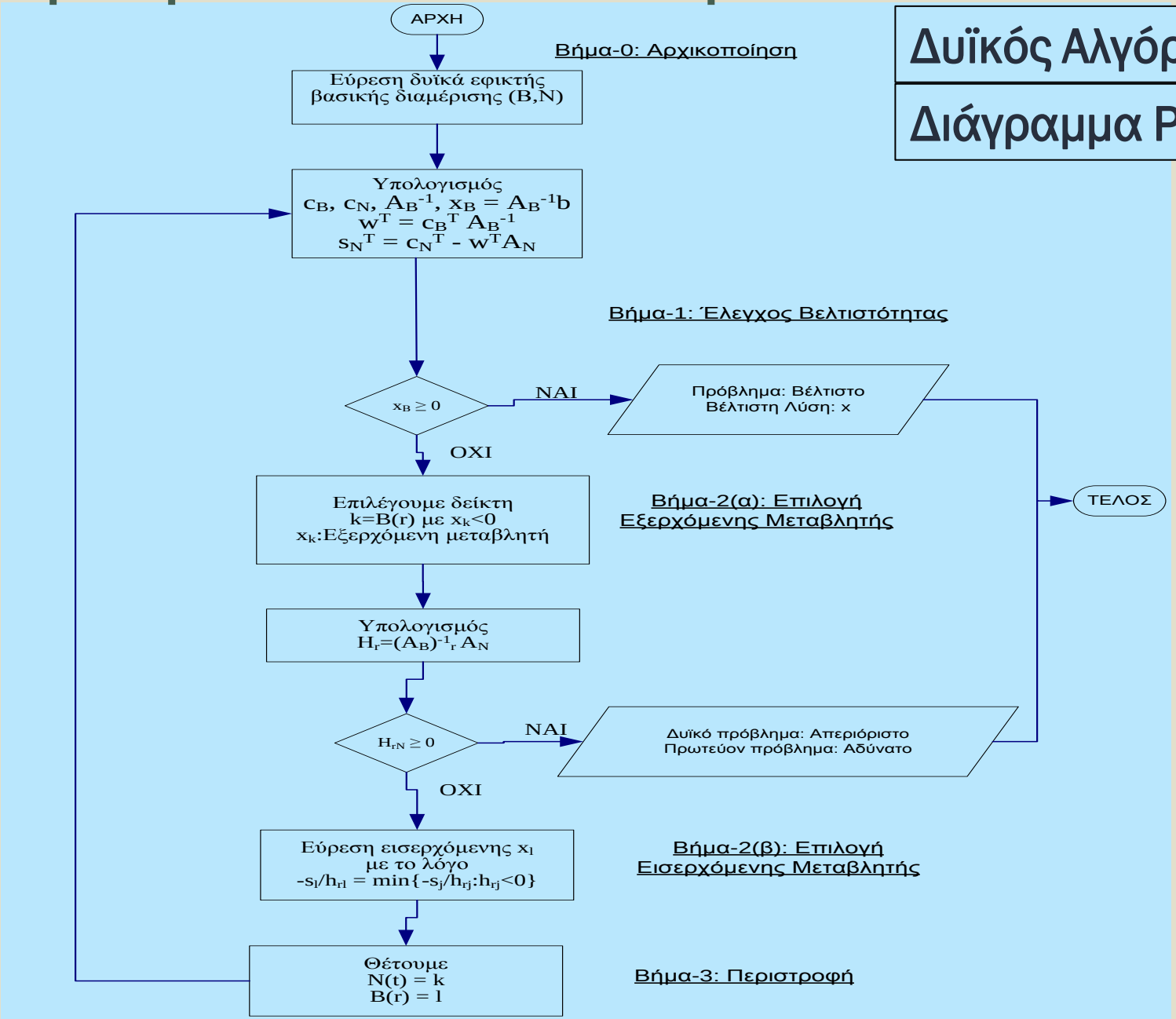
Εναλλάσσεται ο ρόλος των μεταβλητών x_l και x_k, θέτοντας N(t) = k και B(r)=l.

Ο αλγόριθμος συνεχίζεται από το βήμα-0 (β).

Αλγόριθμοι τύπου Simplex

Δυϊκός Αλγόριθμος Simplex

Διάγραμμα Ροής



Αλγόριθμοι Τύπου Simplex

Κανόνες Περιστροφής

- **Κανόνας του Dantzig** ή κανόνας του μέγιστου συντελεστή.

  Ως εισερχόμενη μεταβλητή x_l ορίζεται η μεταβλητή που αντιστοιχεί στην ελάχιστη τιμή για την πλεονασματική μεταβλητή s, δηλαδή $s_l = \min\{s_j \mid s_j<0\}$ (Klee & Minty 1972).

- **Κανόνας του Bland** ή κανόνας του ελαχίστου δείκτη.

  Τόσο για κατά την επιλογή της εισερχόμενης μεταβλητής, όσο και για την επιλογή της εξερχόμενης μεταβλητής, επιλέγεται αυτή που έχει τον μικρότερο δείκτη (Bland 1977).

  ✓ Βοηθάει στην αποφυγή του φαινομένου της κύκλωσης.

- **Κανόνας της Στοίβας.**

  Διατηρείται μία στοίβα για τις βασικές μεταβλητές και μία άλλη για τις μη βασικές μεταβλητές. Η επιλογή της εισερχόμενης ή εξερχόμενης μεταβλητής γίνεται με βάση τη μεταβλητή που βρίσκεται στο υψηλότερο σημείο της στοίβας. Χρησιμοποιείται δηλαδή η αρχή LIFO.

- **Κανόνας της Ουράς.**

  Παρόμοιος με τον κανόνα της Στοίβας που περιγράφτηκε πιο πάνω, με τη διαφορά ότι χρησιμοποιείται η αρχή FIFO .

Αλγόριθμοι Τύπου Simplex

Κανόνες Περιστροφής

- **Μέθοδος της λιγότερο πρόσφατης θεώρησης**

  Σε κάθε επανάληψη, επιλέγεται ως εισερχόμενη μεταβλητή x_l, αυτή η οποία έχει το μικρότερο δείκτη που είναι μεγαλύτερος από τον δείκτη που επιλέχτηκε κατά την προηγούμενη επανάληψη (Gartner 1995).

- **Κανόνας τη μέγιστης βελτίωσης**

  Επιλέγεται ως εισερχόμενη μεταβλητή αυτή η οποία επιφέρει τη μεγαλύτερη βελτίωση στην τιμή της αντικειμενικής συνάρτησης (Gartner 1995).

- **Κανόνας της μερικής αποτίμησης**

  Για την επιλογή της εισερχόμενης μεταβλητής, οι μη βασικές μεταβλητές ομαδοποιούνται σε σύνολα, τα οποία εξετάζονται διαδοχικά κατά τις επαναλήψεις του αλγορίθμου (Benichou et al 1977).

- **Κανόνας της περισσότερο απότομης μεταβολής**

  Ως εισερχόμενη επιλέγεται η μεταβλητή που επιφέρει την πιο απότομη μεταβολή στην τιμή της αντικειμενική συνάρτησης ανά μονάδα απόστασης, με βάση τη γεωμετρία του προβλήματος (Swietanowski 1998, Vanderbei 2001).

Αλγόριθμοι Τύπου Simplex

Εύρεση αρχικής εφικτής λύσης

- **Μέθοδος των 2 φάσεων**
  - **Φάση Ι**

    Προστίθενται τεχνητές μεταβλητές και η αντικειμενική συνάρτηση είναι το άθροισμα των μεταβλητών αυτών.

    Επιλύεται το τροποποιημένο πρόβλημα.

    Αν η ελάχιστη τιμή είναι 0, πάμε στη φάση ΙΙ. Αλλιώς το πρόβλημα αδύνατο.
  - **Φάση ΙΙ**

    Χρησιμοποιείται η λύση της φάσης Ι, ως αρχική λύση.
- **Μέθοδος του μεγάλου Μ**

  Επιλύει ένα μόνο τροποποιημένο γραμμικό πρόβλημα κάνοντας χρήση ενός πολύ μεγάλου αριθμού Μ, τροποποιώντας την αντικειμενική συνάρτηση.
  - Αν το νέο πρόβλημα είναι βέλτιστο, τότε αν η τιμή των τεχνητών μεταβλητών είναι 0, το αρχικό πρόβλημα είναι επίσης βέλτιστο και έχει βρεθεί μία βέλτιστη λύση.
  - Αν στη βέλτιστη λύση υπάρχουν τεχνητές μεταβλητές με μη μηδενικές τιμές, τότε το αρχικό πρόβλημα είναι αδύνατο.
  - Αν το τροποποιημένο πρόβλημα είναι απεριόριστο, τότε το αρχικό πρόβλημα θα είναι είτε απεριόριστο, είτε αδύνατο.

Αλγόριθμοι Τύπου Simplex

Τεχνικές κλιμάκωσης

- Περιπτώσεις που έχουμε πολύ μεγάλους πίνακες, με μεγάλη απόκλιση ανάμεσα στις απόλυτες τιμές των αριθμών, αυτό μπορεί να προκαλέσει προβλήματα ως προς την ακρίβεια των λύσεων.

- **Τεχνική της Ισορρόπησης**
  - Πολλαπλασιάζει τα στοιχεία κάθε στήλης του A και του c με

$$colmax(j) = \max\left\{\frac{1}{|\alpha_{ij}|} : i = 1,2,\dots,m\right\}, j = 1,2,\dots,n$$

  - Στη συνέχεια πολλαπλασιάζει τα στοιχεία κάθε γραμμής του A και του c με

$$rowmax(i) = \max\left\{\frac{1}{|\alpha_{ij}|} : j = 1,2,\dots,n\right\}, i = 1,2,\dots,m$$

  ώστε να προκύψουν τιμές ανάμεσα στο -1 και το +1.

  - Επιλύει το τροποποιημένο πρόβλημα γραμμικού προγραμματισμού.
  - Εκτελώντας τις αντίστροφες πράξεις υπολογίζεται η λύση του αρχικού προβλήματος.

Αλγόριθμοι Τύπου Simplex

Τεχνικές κλιμάκωσης

- Περιπτώσεις που έχουμε πολύ μεγάλους πίνακες, με μεγάλη απόκλιση ανάμεσα στις απόλυτες τιμές των αριθμών, αυτό μπορεί να προκαλέσει προβλήματα ως προς την ακρίβεια των λύσεων.

- **Τεχνική του Γεωμετρικού Μέσου**
  - Πολλαπλασιάζει τα στοιχεία κάθε στήλης j του A και του c με το 1/mc(j)

$$colmax(j) = \max\{|\alpha_{ij}| : i = 1,2,\dots,m\}, j = 1,2,\dots,n$$

$$colmin(j) = \min\{|\alpha_{ij}| : i = 1,2,\dots,m\}, j = 1,2,\dots,n$$

$$\mathrm{mc(j)} = \sqrt{colmax(j)\ colmin(j)}$$

  - Στη συνέχεια πολλαπλασιάζει τα στοιχεία κάθε γραμμής i του A και του c με το 1/mr(i)

$$rowmax(j) = \max\{|\alpha_{ij}| : j = 1,2,\dots,n\}, i = 1,2,\dots,m$$

$$rowmin(j) = \min\{|\alpha_{ij}| : j = 1,2,\dots,n\}, i = 1,2,\dots,m$$

$$\mathrm{mr(i)} = \sqrt{rowmax(i)\ rowmin(i)}$$

  - Επιλύει το τροποποιημένο πρόβλημα γραμμικού προγραμματισμού.
  - Εκτελώντας τις αντίστροφες πράξεις υπολογίζεται η λύση του αρχικού προβλήματος.

Υλοποίηση Αλγορίθμων σε γλώσσα Python

- Χρήση γλώσσας προγραμματισμού Python – έκδοση 3.6
- Βιβλιοθήκες Anaconda
- Χρήση μαθηματικών βιβλιοθηκών NumPy
- Περιβάλλον προγραμματισμού Spyder
- Λειτουργικό σύστημα: Windows 10

Υλοποίηση Αλγορίθμων σε γλώσσα Python

Αλγόριθμοι-τεχνικές που υλοποιήθηκαν σε γλώσσα Python

- Αναθεωρημένος αλγόριθμος Simplex
- Δυϊκός αλγόριθμος Simplex
- Μέθοδος του μεγάλου M
- Τεχνικές κλιμάκωσης
  - *Κλιμάκωση ισορρόπησης*
  - *Κλιμάκωση γεωμετρικού μέσου*

Υλοποίηση Αλγορίθμων σε γλώσσα Python

Αλγόριθμοι-τεχνικές που υλοποιήθηκαν σε γλώσσα Python

- Συναρτήσεις για ανάγνωση αρχείων αποθηκευμένων σε μορφή **mps**
- Συναρτήσεις για αποθήκευση προβλημάτων σε μορφή **mps**
- Συναρτήσεις για δημιουργία τυχαίων προβλημάτων ανάλογα με τις παραμέτρους που δίνει ο χρήστης

Υλοποίηση Αλγορίθμων σε γλώσσα Python

Αναθεωρημένος αλγόριθμος Simplex

Εφαρμογή των βημάτων του αλγορίθμου μέσω επαναχρησιμοποιήσιμων συναρτήσεων Python:

```
def simpex_revised(problemname, A, b, c, m, n, doScaling, type, M):
    # Καλούμε τη μέθοδο του μεγάλου M για τον προσδιορισμό της αρχικής βάσης
    (art_vars, A, c, n, B, N) = bigM_method(M, A, c, m, n)
    # Κάνουμε κλιμάκωση αν χρειάζεται
    if doScaling:
        (A,b,c,colmulti,rowmulti)=scaling(A, b, c, m, n, type)
    # Αρχικοποίηση (βήμα 0)
    initialization(True, A, b, c, m, n, B, N)
    # Μετρητής επαναλήψεων
    iterations = 1
    while True:
        J = find_J()
        if J==[]:
            (is_optimal,values)=check_optimal(art_vars,m,n,B, N)
            # Αν έχουμε κλιμάκωση υπολογίζουμε τις τιμές των x
            if doScaling:
                for i in range(len(xB)):
                    xB[i] = xB[i] / colmulti[B[i]-1]
                    cB[i] = cB[i] * colmulti[B[i]-1]
            break
```

Υλοποίηση Αλγορίθμων σε γλώσσα Python

Αναθεωρημένος αλγόριθμος Simplex (συνέχεια)

```
else:
        # Επιλογή εισερχόμενης μεταβλητής
        I=choose_incoming(J)
        # Υπολογισμός συνόλου I+
        Iplus = find_Iplus(A, N, I, b)
        if Iplus==[]:
            state = "unbounded"
            z = None
            break
        else:
            # Υπολογισμός εξερχόμενης μεταβλητής
            (r, pos) = choose_outcoming(B, Iplus)
            # Εφαρμογή pivot
            pivot(B, N, I, r)
            iterations += 1
            initialization(False,  A, b, c, m, n, B, N)
    return (iterations, state, z)
```

Υλοποίηση Αλγορίθμων σε γλώσσα Python

Αναθεωρημένος αλγόριθμος Simplex

Παράμετροι

- **problemname**, το όνομα του προβλήματος
- **A**, ο πίνακας διάστασης mxn, των συντελεστών των περιορισμών
- **b**, ο πίνακας διάστασης mx1 που περιέχει τους σταθερούς όρους στους περιορισμούς.
- **c**, ο πίνακας διάστασης nx1 που περιέχει τους συντελεστές των μεταβλητών στην αντικειμενική συνάρτηση.
- **m**, το πλήθος των περιορισμών.
- **n**, το πλήθος των μεταβλητών.
- **doScaling**, μεταβλητή λογικού τύπου (boolean), η οποία όταν είναι ίση με True, κάνει τον αλγόριθμο να χρησιμοποιεί τεχνικές κλιμάκωσης.
- **type**, ο τύπος της τεχνικής κλιμάκωσης που χρησιμοποιείται, για την περίπτωση που το όρισμα doScaling έχει την τιμή True.
  - *Για type = 1, χρησιμοποιείται η τεχνική της ισορρόπησης,*
  - *Για type = 2, τότε χρησιμοποιείται γεωμετρική κλιμάκωση.*
- **M,** η μεγάλη τιμή που θα πρέπει να χρησιμοποιηθεί κατά την εφαρμογή της μεθόδου του μεγάλου M, για την εύρεση μίας αρχικής βασικής εφικτής λύσης.

Υλοποίηση Αλγορίθμων σε γλώσσα Python

Αναθεωρημένος αλγόριθμος Simplex

Έξοδοι

- **iterations**, που είναι ο αριθμός των επαναλήψεων, μέχρι να τερματιστεί ο αλγόριθμος.
- **state**, το οποίο καθορίζει τον τύπο του προβλήματος, εάν δηλαδή είναι βέλτιστο (optimal) ή αν είναι απεριόριστο (unbounded).
- **z**, η τιμή της αντικειμενικής συνάρτησης που αντιστοιχεί στη βέλτιστη λύση που υπολογίστηκε από τον αλγόριθμο, εάν το πρόβλημα είναι βέλτιστο.

Υλοποίηση Αλγορίθμων σε γλώσσα Python

Δυικός αλγόριθμος Simplex

Εφαρμογή των βημάτων του αλγορίθμου μέσω επαναχρησιμοποιήσιμων συναρτήσεων Python:

```
def dual_simplex(problemname, A, b, c, m, n):
    B=np.array([5, 6, 7, 8])
    N=np.array([1, 2, 3, 4])
    linear_problem.show_problem(problemname, A, b, c, B, N)
    # Αρχικοποίηση (βήμα 0)
    initialization(True, A, b, c, m, n, B, N)
    iterations = 1
    while True:
        # Βήμα 1 (Έλεγχος βελτιστότητας)
        (is_optimal, r, values) = check_optimal_dual(m, n, B, N)
        if is_optimal == True:
            show_opt_values(is_optimal, values, B, N, n, cB, xB)
            break
        # Βήμα 2α
        k = B[r]
         (t,l) = find_incoming_dual(A, B, r, N)
       # Περιστροφή
        pivot(B, N, t, r)
        iterations += 1
        initialization(True,  A, b, c, m, n, B, N)
        show_iteration(iterations)
```

Υλοποίηση Αλγορίθμων σε γλώσσα Python

Επαναχρησιμοποιήσιμες συναρτήσεις

- **initialization**, για την αρχικοποίηση των παραμέτρων του προβλήματος και την εκτέλεση των πράξεων κατά την εκκίνηση της εκτέλεσης κάθε επανάληψης.
- **bigM_method**, υλοποίηση της μεθόδου του μεγάλου M και την εύρεση μίας αρχικής βασικής εφικτής λύσης.
- **scaling**, υλοποίηση των τεχνικών κλιμάκωσης, ανάλογα με την τιμή της παραμέτρου doScaling και της παραμέτρου type.
- **check_optimal**, έλεγχος της βελτιστότητας της τρέχουσας λύσης. Αν η συνάρτηση αυτή επιστρέψει την τιμή True, τότε αυτό σημαίνει ότι η τρέχουσα λύση είναι βέλτιστη και ο αλγόριθμος σταματά.
- **find_Iplus**, για την εύρεση του συνόλου I_+ που χρησιμοποιεί ο αναθεωρημένος αλγόριθμος Simplex. Αν είναι κενό, τότε το πρόβλημα είναι απεριόριστο.

Υλοποίηση Αλγορίθμων σε γλώσσα Python

Επαναχρησιμοποιήσιμες συναρτήσεις (συνέχεια)

- **choose_incoming**, για την εύρεση της εισερχόμενης μεταβλητής στο σύνολο B των βασικών μεταβλητών.
- **choose_outcoming**, για την εύρεση της εξερχόμενης μεταβλητής από το σύνολο B των βασικών μεταβλητών.
- **pivot**, για την εφαρμογή της περιστροφής με βάση την εισερχόμενη και την εξερχόμενη μεταβλητή που επιλέχτηκε από τις προηγούμενες συναρτήσεις.
- **read_file**, συνάρτηση για την ανάγνωση ενός γραμμικού προβλήματος, το οποίο βρίσκεται αποθηκευμένο σε αρχείο σε **mps** μορφή.
- **store_problem**, συνάρτηση για την αποθήκευση ενός γραμμικού προβλήματος σε αρχείο σε μορφή **mps**.
- **generate_random_problem**, συνάρτηση για τη δημιουργία τυχαίων γραμμικών προβλημάτων, ανάλογα με τις παραμέτρους που της δίνονται.

Υπολογιστική μελέτη

Μορφή mps για την αποθήκευση γραμμικών προβλημάτων

```
NAME          PROBLEM3
ROWS
 N  OBJ
 L  CONSTR1
 L  CONSTR2
 L  CONSTR3
COLUMNS
    X1        OBJ                 1.   CONSTR1            1.
    X1        CONSTR2             1.   CONSTR3           -1.
    X2        OBJ                 1.   CONSTR1            1.
    X2        CONSTR2             1.   CONSTR3            1.
    X3        OBJ                -4.   CONSTR1            2.
    X3        CONSTR2            -1.   CONSTR3            1.
RHS
    RHS1      CONSTR1             9.   CONSTR2            2.
    RHS1      CONSTR3             4.
ENDATA
```

Υπολογιστική μελέτη

Δημιουργία τυχαίων γραμμικών προβλημάτων

- **Διαστάσεις προβλημάτων:** 50x50, 100x100, 150x150, ..., 1000x1000
- 10 διαφορετικά προβλήματα για την κάθε κατηγορία
- Για κάθε κατηγορία προβλημάτων υπολογίστηκε ο μέσος χρόνος εκτέλεσης και το μέσο πλήθος επαναλήψεων.
- Οι συντελεστές των προβλημάτων επιλέχτηκαν με τυχαίο τρόπο και κυμάνθηκαν στα πιο κάτω όρια:

	Ελάχιστη τιμή	Μέγιστη τιμή
Συντελεστής πίνακα A	-100	1000
Συντελεστής πίνακα b	10	10000
Συντελεστής πίνακα c	-300	900

Υπολογιστική μελέτη

Δημιουργία τυχαίων προβλημάτων

```
minA = -100.0 ; maxA = 1000.0; minb = 10.0        # Τιμές των παραμέτρων
maxb = 10000.0; minc = -300.0; maxc = 900.0
iterations = 10  # Αριθμός προβλημάτων
file = open("results.txt", "w")
for k in range(50,1001, 50):                      # Δημιουργία των προβλημάτων
    sum_iterations = 0;     sum_time = 0.0
    for i in range(iterations):
        (problemname, A, b, c, m, n)=generate_random_problem(k,k,minA, maxA, minb, maxb, minc,
maxc,1)
        time1 = time.time()
        (niter, state, z) = Simplex_method.simpex_revised(problemname, A, b, c, m, n, True,1, 10**6)
        time2 = time.time()
        time_diff = time2-time1
        sum_iterations += niter
        sum_time += time_diff
    avg_iterations = sum_iterations/iterations
    avg_time = sum_time/iterations
    file.write(str(m)+"\t"+str(m)+"\t"+str(avg_iterations)+"\t"+str(avg_time)+"\n")
file.close()
```

Υπολογιστική μελέτη

Στατιστικά ανά κατηγορία προβλήματος

Υπολογιστική μελέτη

Στατιστικά ανά κατηγορία προβλήματος

Υπολογιστική μελέτη

Δημιουργία τυχαίων προβλημάτων διαφορετικών πυκνοτήτων

```
minA = -100; maxA = 1000; minb = 10
maxb = 10000; minc = -300; maxc = 900
m=100; n=100; iterations = 20
file = open("results_dens.txt", "w")
for k in range(10,101,10):
    sum_iterations = 0;     sum_time = 0.0
    for i in range(iterations):
        (problemname,A,b,c,m,n)=generate_random_problem(m,n,minA,maxA,minb,maxb,minc, maxc,k/100)
        time1 = time.time()
         (niter, state, z)=Simplex_method.simpex_revised(problemname, A, b, c, m, n, False, 2, 10**6)
         time2 = time.time()
         time_diff = time2-time1
         sum_iterations += niter
         sum_time += time_diff
    avg_iterations = sum_iterations/iterations
    avg_time = sum_time/iterations
    file.write(str(m)+"\t"+str(m)+"\t"+str(avg_iterations)+"\t"+str(avg_time)+"\n")
```

Υπολογιστική μελέτη

Στατιστικά ανά κατηγορία πυκνότητας για προβλήματα 100x100

Υπολογιστική μελέτη

Στατιστικά ανά κατηγορία πυκνότητας για προβλήματα 100x100

Συμπεράσματα

- Η Python αποτελεί πολύ ισχυρό εργαλείο που μπορεί να χρησιμοποιηθεί για την ανάπτυξη προγραμμάτων και βιβλιοθηκών για την επίλυση γραμμικών προβλημάτων και την υλοποίηση αλγορίθμων τύπου Simplex.
- Η Python προσφέρει πολλές έτοιμες βιβλιοθήκες, οι οποίες μπορούν να βοηθήσουν στην ανάπτυξη προγραμμάτων που αφορούν αλγορίθμους τύπου Simplex και επιπλέον, επιτρέπει τη δημιουργία επαναχρησιμοποιήσιμων συναρτήσεων.
- Ο αριθμός των επαναλήψεων που χρειάζονται για την επίλυση ενός γραμμικού προβλήματος, αυξάνεται όσο μεγαλώνει η διάσταση του προβλήματος.
- Όσο αυξάνεται η διάσταση ενός γραμμικού προβλήματος, τόσο αυξάνεται και ο απαιτούμενος χρόνος που απαιτείται για την επίλυσή του και μάλιστα, η αύξηση αυτή έχει μεγαλύτερη επίδραση σε σχέση με την αύξηση του αριθμού επαναλήψεων.
- Όσο αυξάνεται η πυκνότητα ενός προβλήματος, δηλαδή όσο μειώνεται το ποσοστό των μηδενικών που υπάρχουν στον πίνακα των συντελεστών του προβλήματος, τόσο απαιτείται μεγαλύτερος χρόνος για την επίλυση του προβλήματος αυτού (περισσότερο χρονοβόρες πράξεις).

Επεκτάσεις

- Υλοποίηση επιπλέον αλγορίθμων τύπου Simplex με τη βοήθεια της γλώσσας προγραμματισμού Python.
- Εκτεταμένη σύγκριση των διαφορετικών αλγορίθμων μεταξύ τους, ως προς τον αριθμό των επαναλήψεων και τον χρόνο που απαιτείται, για την επίλυση προβλημάτων διαφόρων διαστάσεων.
- Ανάπτυξη γραφικού περιβάλλοντος για την δημιουργία και την επίλυση γραμμικών προβλημάτων με φιλικό τρόπο προς τον τελικό χρήστη (τεχνολογίας Django).
- Εκτενέστερη υπολογιστική μελέτη για περισσότερες κατηγορίες προβλημάτων.

Ερωτήσεις

Ευχαριστώ για την προσοχή σας!