ΥΛΟΠΟΙΗΣΗ ΕΝΟΣ ΠΡΩΤΕΥΟΝΤΟΣ - ΔΥΙΚΟΥ ΑΛΓΟΡΙΘΜΟΥ ΓΙΑ ΤΟ ΓΡΑΜΜΙΚΟ ΠΡΟΒΛΗΜΑ ΚΑΙ ΣΥΓΚΡΙΤΙΚΗ ΥΠΟΛΟΓΙΣΤΙΚΗ ΜΕΛΕΤΗ

ΧΑΡΑΛΑΜΠΟΣ ΠΑΝΑΓΙΩΤΟΥ ΤΡΙΑΝΤΑΦΥΛΛΙΔΗΣ

ΜΕΤΑΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Επιβλέπων Καθηγητής Νικόλαος Σαμαράς, Λέκτορας Εφ.Πληροφορικής

Εξεταστές Κωνσταντίνος Παπαρρίζος, Καθηγητής Εφ.Πληροφορικής

Τμήμα Εφαρμοσμένης Πληροφορικής

Πανεπιστήμιο Μακεδονίας

Θεσσαλονίκη

Ιούλιος 2007

Η έγκριση της μεταπτυχιακής εργασίας από το Τμήμα Εφαρμοσμένης Πληροφορικής του Πανεπιστημίου Μακεδονίας δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα εκ μέρους του Τμήματος.

ΠΕΡΙΛΗΨΗ

Ο Γραμμικός Προγραμματισμός εξετάζει ένα από τα πιο εφαρμοσμένα προβλήματα των μαθηματικών, και την καρδιά της Επιχειρησιακής Έρευνας, το *Γραμμικό Πρόβλημα*. Από την ανάπτυξη του αλγόριθμου Simplex έως και σήμερα έχουν αναπτυχθεί αρκετές οικογένειες αλγορίθμων ανάμεσα στις οποίες οι αλγόριθμοι εσωτερικών σημείων, αλγόριθμοι εξωτερικών σημείων αλλά και πρωτεύοντες - δυικοί αλγόριθμοι. Επίσης ιδιαίτερη έμφαση έχει δωθεί τα τελευταία χρόνια ως προς τις τεχνικές υλοποίησης που χρησιμοποιούνται, είτε μαθηματικές, είτε λογικές - αλγοριθμικές, όπως για παράδειγμα η γρήγορη εκκίνηση των αλγορίθμων (*warm - starting*) ως προς την κατασκευή ενός πρώτου εφικτού σημείου.

Στην παρούσα εργασία παρουσιάζεται ένας τέτοιος πρωτεύων δυικός περιστροφικός αλγόριθμος δύο δρόμων, ο $PDTPSA$[11] και συγκρίνεται ως προς την πρακτική του αποτελεσματικότητα με τον πρωτεύων αλγόριθμο Simplex, καθώς και με την υλοποίηση του αλγόριθμου εσωτερικών σημείων του πακέτου MATLAB σε τυχαία αραιά γραμμικά προβλήματα καθώς και σε ένα πλήθος βέλτιστων μετροπροβλημάτων. Επίσης γίνεται μια σύντομη περιγραφή ενός αλγόριθμου εξωτερικών σημείων στο τέταρτο κεφάλαιο.

Πιο συγκεκριμένα ο αλγόριθμος PDTPSA βρέθηκε να είναι καλύτερος σε όλες μας τις μετρήσεις ως προς τον αλγόριθμο Simplex, με *χειρότερη επίδοση* να είναι 4,39 φορές καλύτερος σε χρόνο επίλυσης και 3,66 φορές σε αριθμό επαναλήψεων σε προβλήματα 750*750, 2.5% πυκνότητας και *καλύτερη* 14,9 φορές σε χρόνο επίλυσης και 13,5 σε αριθμό επαναλήψεων σε προβλήματα διαστάσεων 2000*2000 και 20% πυκνότητας. Επίσης παρουσίασε ιδιαίτερη σταθερότητα και καλύτερη συμπεριφορά ως προς τον αλγόριθμο εσωτερικών σημείων του MATLAB όσο η πυκνότητα των προβλημάτων αυξανόταν.

Επιβλέπων: Σαμαράς Νικόλαος

Θέση: Λέκτορας Εφαρμοσμένης Πληροφορικής

ΕΥΧΑΡΙΣΤΙΕΣ

Ο κ.Σαμαράς Ν. , ο επιβλέπων των εργασιών μου, έχει δείξει ένα απίστευτα μεγάλο όγκο εμπιστοσύνης στο πρόσωπό μου. Έχω διδαχθεί από εκείνον τόσα πολλά ώστε να είναι αδύνατον να τα ξεπληρώσω. Μου είναι επίσης αδύνατον να συνοψίσω σε μερικές μόνο γραμμές την απροβλημάτιστη συνεργασία μας για τα τελευταία 4 χρόνια, εφόσον υπήρξε επιβλέπων και της πτυχιακής μου εργασίας. Έχει αποτελέσει για μένα μια αστείρευτη πηγή γνώσεων και βοήθειας από όλες τις απόψεις. Σκοπεύω να διατηρήσω το υψηλό του επίπεδο, ακαδημαϊκά και ανθρώπινα όπου και να βρίσκομαι μελλοντικά.

Θα ήθελα επίσης να ευχαριστήσω τον εξεταστή της μεταπτυχιακής μου εργασίας και Καθηγητή κ. Παπαρρίζο Κ., καθώς η καθοδήγηση που μου χάρισε υπήρξε κριτικής σημασίας για εμένα και τις σπουδές μου. Η εργασία αυτή βρίσκεται στα χέρια σας χάριν στην συνεχή του παρουσία.

Ο κ. Ιωάννης Ν. Τσιτσικλής, ένας διακεκριμένος επιστήμονας και Καθηγητής στο M.I.T, μέσα από τις ηλεκτρονικές μας συνομιλίες, υπήρξε ιδιαίτερα βοηθητικός με τις συμβουλές του σε όλες μου τις απορίες. Νιώθω την ανάγκη να τον ευχαριστήσω ξανά και από τις γραμμές αυτές.

Τέλος θα ήθελα να σταθώ σε έναν καλλιτέχνη, φίλο, και μεγάλη συμπαράσταση στις σπουδές μου, τον Διδάκτωρ του Πανεπιστημίου του Λονδίνου Goldsmiths, κ. Ματθαίο Τσαχουρίδη. Η καλλιτεχνική του εκτέλεση, το αδιαμφισβήτητο ταλέντο του αλλά και η επιμονή στην τελειότητα σε ότι κάνει, μου δίνει δύναμη να συνεχίζω. Έχει γεμίσει αμέτρητες ώρες έρευνάς μου, με τον πιο όμορφο τρόπο.

ΠΕΡΙΕΧΟΜΕΝΑ

1 Εισαγωγή στο Γραμμικό Πρόβλημα **9**

1.1 Μορφές Γραμμικών Προβλημάτων 10

1.2 Γραμμικότητα 11

1.3 Αποθήκευση γραμμικών προβλημάτων σε ένα υπολογιστικό σύστημα 12

1.4 Μετασχηματισμοί 12

2 Δυική Θεωρία **14**

2.1 Κατασκευή του Δυικού προβλήματος 14

2.2 Χαλαρή και Δυική Θεωρία 16

3 Ο αναθεωρημένος αλγόριθμος Simplex **18**

3.1 Χαρακτηριστικά αλγορίθμων τύπου - Simplex - Περιγραφή Ψευδοκώδικα 19

3.2 Ένα παράδειγμα εκτέλεσης του κώδικα 20

4 Ένας αλγόριθμος Εξωτερικών σημείων τύπου Simplex **22**

4.1 Κατασκευή αρχικά εφικτής διαμέρισης 23

4.2 Βήματα του αλγορίθμου 24

4.3 Ένα παράδειγμα εκτέλεσης του κώδικα 25

5 Ο πρωτεύων δυικός αλγόριθμος τύπου Simplex δύο δρόμων PDTPSA **27**

5.1 Κατασκευή αρχικά εφικτής διαμέρισης 28

5.2 Κατασκευή δυικά εφικτής διαμέρισης 29

5.3 Βήματα του αλγορίθμου 31

5.4 Εφαρμογή του αλγορίθμου 32

5.5 Ένα παράδειγμα εκτέλεσης του κώδικα 35

6 Θέματα Υλοποίησης **37**

6.1 Δεσμοί 37

6.2 Προλυτικές διαδικασίες 37

6.3 Τεχνικές κλιμάκωσης 38

6.4 Ανοχές 39

6.5 Re-inversion 39

6.6 Μετατροπή από την μορφή .MPS - Mathematical Programming Standard . . . 40

7 Συγκριτική Υπολογιστική Μελέτη **46**

7.1 Γραφικές Αναπαραστάσεις των μετήσεων 46

8 Συμπεράσματα **55**

"A cynic is a man who knows the price of everything but the value of nothing."

-Oscar Wilde

ΚΕΦΑΛΑΙΟ 1

Εισαγωγή στο Γραμμικό Πρόβλημα

Η Επιχειρησιακή Έρευνα, καρδιά της οποίας είναι ο *Γραμμικός Προγραμματισμός*, είναι γέννημα αλλά όχι θρέμμα του Β΄ Παγκοσμίου Πολέμου. Ήταν τότε που για πρώτη φορά εφαρμόστηκαν επιστημονικές μέθοδοι με στόχο να επιτευχθεί καλύτερη αναδιανομή των υπαρχόντων πόρων και προμηθειών.

Μετά το τέλος του Πολέμου, ο George B. Dantzig (*National Medal of Science, 1976*) κατασκεύασε το 1947 τον πρώτο αλγόριθμο, με το όνομα *Simplex*, μάλλον το πιο σημαντικό επίτευγμα της Επιχειρησιακής Έρευνας. Αργότερα όμως, οι Klee και Minty το 1972 έπληξαν την πρακτική αποτελεσματικότητα του αλγορίθμου αυτού, αποδεικνύοντας πως η χειρότερη περίπτωσή του είναι εκθετική και όχι πολυωνυμική όπως θα ήταν επιθυμητό.

Ο πρώτος θεωρητικά πολυωνυμικός αλγόριθμος αλλά στην πράξη πολύ χειρότερος από τον Simplex ήταν ο γνωστός *Ρώσικος -Ελλειψοειδής αλγόριθμος* το 1979 από τους L.Khachian και N.Z.Shor. Αργότερα, ένας ερευνητής των εργαστηρίων Bell ο Karmarkar, το 1983 στο συνέδριο της IEEE ανακοινώνει πως ο αλγόριθμος των *Εσωτερικών Σημείων* που κατασκεύασε, είναι στην πράξη μέχρι και 50 φορές καλύτερος (αν και ωστόσο αυτό αποδείχθηκε αργότερα υπερβολικό) από την μέθοδο Simplex. Από τότε οι μέθοδοι εσωτερικών σημείων έχουν επιδεχθεί πολλών βελτιώσεων και αποτελούν την ευρέως χρησιμοποιούμενη οικογένεια αλγορίθμων για προβλήματα γραμμικής βελτιστοποίησης.

Το κεφάλαιο αυτό αποτελεί μια σύντομη εισαγωγή στον Γραμμικό Προγραμματισμό. Αναφέρονται οι σημαντικότερες ιδιότητες ώστε να είναι ομαλή στα επόμενα κεφάλαια η μετάβαση στην ανάλυση των αλγορίθμων που χρησιμοποιούνται ως λύτες.

1.1 Μορφές Γραμμικών Προβλημάτων

Το πιο απλό γραμμικό πρόβλημα αποτελείται από δύο ουσιαστικά μέρη: α) την αντικειμενική συνάρτηση και β) τους περιορισμούς. Η αντικειμενική συνάρτηση είναι μια γραμμική συνάρτηση με τόσες ανεξάρτητες μεταβλητές όσες και το πρόβλημά μας. Έτσι, εάν αναφερόμαστε στον χώρο των δύο διαστάσεων, τότε η γραμμική αυτή συνάρτηση έχει 2 *μεταβλητές απόφασης*. Η αριστοποίηση (είτε ελαχιστοποίηση είτε μεγιστοποίηση) της συνάρτησης αυτής λαμβάνοντας υπόψην μας όλους τους περιορισμούς είναι ο στόχος ενός αλγόριθμου - λύτη γραμμικών προβλημάτων.

Τα γραμμικά προβλήματα συναντώνται συνήθως με δύο μορφές. Την γενική και την τυπική. Η διαφορά τους έγκειται στο ότι στην τυπική μορφή όλοι οι περιορισμοί έχουν μετατραπεί σε αυστηρές ισότητες.

Το γενικό γραμμικό πρόβλημα επομένως διατυπώνεται μαθηματικά ως εξής:

$$\begin{aligned} &\min c^T x \\ &\text{subject to } Ax \oplus b \\ &x \geq 0. \end{aligned}$$

Το σύμβολο $\oplus$ χρησιμοποιείται εδώ για να περιγράψει το είδος του περιορισμού από τις τρεις πιθανές καταστάσεις $\leq, =, \geq$. Παρατηρούμε επίσης τους περιορισμούς *μη αρνητικότητας* για την μεταβλητή απόφασης x. Αυτό συμβαίνει γιατί τα γραμμικά προβλήματα είναι συνήθως αληθινά καθημερινά προβλήματα που οι μεταβλητές απόφασης αναπαριστούν θετικές ποσότητες. Για το λόγο αυτό και οι περιορισμοί αυτοί είναι γνωστοί και ως *φυσικοί περιορισμοί*. Φυσικά το πρόβλημά μας θα μπορούσε να είναι πρόβλημα μεγιστοποίησης αλλά λόγω της εύκολα ισοδύναμης μετατροπής του (αντιστρέφουμε τα πρόσημα στην αντικειμενική συνάρτηση) θα θεωρούμε στο εξής όλα τα γραμμικά μας προβλήματα, σαν προβλήματα ελαχιστοποίησης. Ας προχωρήσουμε όμως σε μερικές δημοφιλείς έννοιες του Γραμμικού Προγραμματισμού.

Το σύνολο όλων των σημείων που ικανοποιούν όλους τους περιορισμούς αποτελεί το σύνολο των *εφικτών σημείων ή εφικτών λύσεων* σε αντίθεση με τα *μη εφικτά* σημεία. Αναπόφευκτα εάν το σύνολο αυτό είναι κενό, το πρόβλημά μας είναι *αδύνατο*. Διαφορετικά είναι *εφικτό*. Το σύνολο αυτό είναι γνωστό και ως η *εφικτή περιοχή* του προβλήματός μας.

Τα εφικτά προβλήματα διαχωρίζονται επίσης σε 2 υπο-κατηγορίες. Τα *απεριόριστα* και τα *βέλτιστα* προβλήματα. Ένα απεριόριστο πρόβλημα έχει την εφικτή του περιοχή να επεκτείνεται

προς το άπειρο σε ανοικτή μορφή ως προς μια κατεύθυνση και δεν χαρακτηρίζεται από μία βέλτιστη λύση καθώς αυτή συνεχώς βελτιώνεται όσο εμείς μετακινούμαστε προς το άπειρο. Ένα βέλτιστο πρόβλημα έχει κλειστή εφικτή περιοχή και χαρακτηρίζεται από συνήθως μία βέλτιστη λύση. Σίγουρα όμως το σύνολο των βέλτιστων λύσεων είναι πεπερασμένο. Μπορούμε να συνοψίσουμε τα παραπάνω στο ακόλουθο Θεώρημα:

Θεώρημα 1.1.1 (Θεμελιώδες Θεώρημα του Γραμμικού Προγραμματισμού) *Ένα γραμμικό πρόβλημα είναι είτε εφικτό είτε αδύνατο. Αν είναι εφικτό τότε είναι είτε απεριόριστο είτε βέλτιστο.*

Η επίλυση ενός γραμμικού προβλήματος εμπλέκει ουσιαστικά την κατηγοριοποίηση του προβλήματός μας σε μία από τις προαναφερθείσες κατηγορίες. Στην πραγματικότητα αυτό δεν είναι πάντα αρκετό. Αν το πρόβλημά μας είναι βέλτιστο θα χρειαστούμε τουλάχιστον μία βέλτιστη λύση. Αυτός είναι και ο λόγος που ο *Γραμμικός Προγραμματισμός* ασχολείται με κατασκευαστικές μεθόδους επίλυσης, δηλαδή *αλγόριθμους*.

1.2 Γραμμικότητα

Θα πρέπει εδώ να ξεκαθαρίσουμε την εύκολα συγχέουσα έννοια της γραμμικότητας. Αυτή έχει 3 κύρια χαρακτηριστικά:

- *Αναλογικότητα* : Αυτό σημαίνει πως κάθε μεταβλητή απόφασης μπορεί να επηρεάσει την τιμή της αντικειμενικής συνάρτησης μόνο με πολλαπλασιασμό του εαυτού της με μια σταθερή τιμή.

- *Προσθετικότητα* : Το σύνολο των συνεισφορών δύο ή περισσότερων μεταβλητών απόφασης είναι αποτέλεσμα της πρόσθεσης των αντίστοιχων προσφορών κάθε μιας χωριστά.

- *Διαιρετότητα* : Μια μεταβλητή απόφασης μπορεί να πάρει οποιαδήποτε τιμή, ακέραια ή κλασματική. Ένα γραμμικό πρόβλημα με δεδομένα μόνο ακέραιους αριθμούς ονομάζεται *ακέραιο γραμμικό πρόβλημα* διαφορετικά το πρόβλημα είναι *μεικτό ακέραιο*.

1.3 Αποθήκευση γραμμικών προβλημάτων σε ένα υπολογιστικό σύστημα

Ένα γραμμικό πρόβλημα μπορεί να αναπαρασταθεί με μορφή μητρών όπως είδαμε προηγουμένως. Έτσι, η μήτρα A κρατά τους συντελεστές των περιορισμών, τα διανύσματα c,b αντίστοιχα το διάνυσμα κόστους και το δεξιό μέρος. Για τον τύπο του κάθε περιορισμού χρησιμοποιούμε ένα νέο διάνυσμα στήλη, με το όνομα Eqin, το οποίο κρατά τους τύπους των περιορισμών με βάση το ακόλουθο σχήμα:

$$Eqin = \{-1 :\leq |0 := |1 :\geq\}$$

Τέλος (η μη αρνητικότητα των μεταβλητών απόφασης θεωρείται δεδομένη και δεν αναφέρεται) αποθηκεύουμε τον τύπο της βελτιστοποίησης (μεγιστοποίηση ή ελαχιστοποίηση) χρησιμοποιώντας μία σημαιοφόρο μεταβλητή MinMaxLP. Μπορεί να λάβει τις ακόλουθες τιμές:

$$MinMaxLP = \{-1 : \min|1 : \max\}$$

1.4 Μετασχηματισμοί

Ένα γραμμικό πρόβλημα χρειάζεται να μετατραπεί για να μπορεί να επιλυθεί με έναν αλγόριθμο λύτη σε μια συγκεκριμένη μορφή αποδεκτή και κατανοητή από τον ίδιο τον αλγόριθμο. Οι περισσότεροι αλγόριθμοι χρησιμοποιούν την μορφή μητρών που περιγράφηκε προηγουμένως. Επειδή το υπολογιστικό περιβάλλον που χρησιμοποιήσαμε ήταν το MATLAB, αυτή η μορφή ήταν επιθυμητή και για εμάς.

Η μετατροπή ενός γραμμικού προβλήματος από την γενική στην τυπική μορφή γίνεται προσθέτοντας *χαλαρές μεταβλητές*. Ας δούμε όμως αναλυτικά πως συμβαίνει αυτό.

Ένας ανισοτικός περιορισμός της μορφής

$$a_1x_1 + a_2x_2 + a_3x_3 + ... + a_nx_n \leq b$$

μπορεί να μετατραπεί σε ισοτικό με την προσθήκη μιας νέας μεταβλητής x_{n+1} στο αριστερό της μέλος. Τότε η προηγούμενη ανισότητα είναι ισοδύναμη με το σύστημα των περιορισμών

$$a_1x_1 + a_2x_2 + a_3x_3 + ... + a_nx_n + x_{n+1} = b \text{ και } x_{n+1} \geq 0.$$

Η νέα αυτή μη αρνητική μεταβλητή ονομάζεται *ελλειματική* (deflict). Αντίστοιχα στην περίπτωση που ο περιορισμός μας είναι της μορφής

$$a_1x_1 + a_2x_2 + a_3x_3 + ... + a_nx_n \geq b$$

η διαδικασία επαναλαμβάνεται αφαιρώντας τώρα μια όμοια μεταβλητή x_{n+1}, η οποία και ονομάζεται *πλεονοσματική* (surplus). Μετά την κατάλληλη αυτή μετατροπή το πρόβλημά μας παίρνει την εξής μορφή:

$$\begin{gathered} \min c^T x \\ \text{subject to } Ax + s = b \\ x, s \geq 0. \end{gathered}$$

όπου με s συμβολίζουμε τις *χαλαρές μεταβλητές* που προσθέσαμε.

ΚΕΦΑΛΑΙΟ 2

Δυική Θεωρία

Η Δυική Θεωρία είναι εξέχουσας σημασίας για να μπορέσουμε να κατανοήσουμε σε βάθος αλγόριμους γραμμικής βελτιστοποίησης καθώς βάση αυτής ορίστηκαν τα κριτήρια σταματήματος ενός αλγόριθμου - λύτη. Κάθε γραμμικό πρόβλημα συνοδεύεται από το δυικό του, το οποίο μπορεί να μας παρέχει ιδαίτερα σημαντικές πληροφορίες για το πρωτεύον μας πρόβλημα. Το δυικό πρόβλημα είναι αποτέλεσμα έρευνας καλών κάτω - ορίων για τα γραμμικά προβλήματα. Στο κεφάλαιο αυτό θα παρουσιάσουμε τις δύο πιο χαρακτηριστικές θεωρίες, την *χαλαρή* και την *ισχυρή* δυικότητα.

2.1 Κατασκευή του Δυικού προβλήματος

Ας υποθέσουμε τώρα ότι έχουμε το ακόλουθο γραμμικό πρόβλημα:

$$\begin{aligned} \min\ & z = 12x_1 + 10x_2 \\ \text{s.t}\ & x_1 + 2x_2 \geq 1 \\ & 3x_1 + x_2 \geq 2 \\ & x_1 + x_2 \geq 3 \\ & x_i \geq 0, (i = 1, 2) \end{aligned}$$

Πολλαπλασιάζουμε την πρώτη ανισότητα με 3, τη δεύτερη με 2 και προσθέτουμε οπότε παίρνουμε

$$9x_1 + 8x_2 \geq 7$$

Ο αριθμός 7 είναι ένα κάτω όριο της βέλτιστης αντικειμενικής τιμής. Πράγματι λόγω των περιορισμών μη αρνητικότητας των μεταβλητών απόφασης είναι :

$$12x_1 \geq 9x_1 \text{ και } 10x_2 \geq 8x_2$$

από τις οποίες με πρόσθεση προκύπτει ότι

$$z = 12x_1 + 10x_2 \geq 9x_1 + 8x_2 \geq 7$$

Βλέπουμε ότι πολλαπλασιάζοντας τις ανισότητες του προβλήματος με κατάλληλους μη αρνητικούς αριθμούς και προσθέτοντας τις ανισότητες που προκύπτουν υπολογίζουμε κάτω όρια της αντικειμενικής συνάρτησης. Για να βρούμε το βέλτιστο κάτω όριο (μέγιστο) θα πρέπει να κατασκευάσουμε και να λύσουμε ένα νέο γραμμικό πρόβλημα, το *δυικό* του αρχικού μας προβλήματος. Ας δούμε πως μπορούμε να κατασκευάσουμε το πρόβλημα αυτό. Πολλαπλασιάζουμε την πρώτη, δεύτερη και τρίτη ανισότητα του αρχικού μας προβλήματος αντίστοιχα με $w_1, w_2, w_3 \geq 0$. Μετά από τις κατάλληλες πράξεις παίρνουμε :

$$(w_1 + 3w_2 + w_3)x_1 + (2w_1 + w_2 + w3)x_2 \geq w_1 + 2w_2 + 3w_3$$

Αν τα w_1, w_2, w_3 επιλεγούν έτσι ώστε να ισχύουν οι ανισότητες :

$$12 \geq w_1 + 3w_2 + w_3 \text{ και } 10 \geq 2w_1 + w_2 + w3$$

τότε η παράσταση $w_1 + 2w_2 + 3w_3$ είναι ένα κάτω όριο της αντικειμενικής τιμής z. Θέλουμε να βρούμε το μέγιστο κάτω όριο επομένως πρέπει να λύσουμε το ακόλουθο γραμμικό πρόβλημα :

$$\begin{gathered}\max\ w_1 + 2w_2 + 3w_3 \\ \text{s.t } w_1 + 3w_2 + w_3 \leq 12 \\ 2w_1 + w_2 + w3 \leq 10 \\ w_i \geq 0, (i = 1, 2, 3)\end{gathered}$$

Γενικότερα εάν το αρχικό μας γραμμικό πρόβλημα είναι το ακόλουθο

$$\begin{gathered}\min\ c^T x \\ \text{subject to } Ax \oplus b \\ x \geq 0.\end{gathered}$$

το δυικό αυτού είναι αντίστοιχα το εξής :

$$\begin{gathered}\min\ y^T b \\ \text{subject to } yA \oplus c^T \\ y \geq 0.\end{gathered}$$

2.2 Χαλαρή και Δυική Θεωρία

Η στενή σχέση μεταξύ του ζεύγους πρωτεύοντος - δυικού προβλήματος περιγράφεται κυρίως από τα δύο Θεωρήματα της χαλαρής και ισχυρής δυικότητας.

Θεώρημα 2.2.1 (Χαλαρή δυικότητα) *Ας υποθέσουμε ότι το x είναι μια εφικτή λύση του αρχικού προβλήματός μας. Αν ισχύει $c^T x = y^T b$ τότε τα x,y είναι οι βέλτιστες λύσεις του πρωτεύοντος και δυικού προβλήματος αντιστοίχως.*

Αυτό σημαίνει πώς η λύση για την οποία και τα δύο προβλήματα είναι εφικτά, και μάλιστα η τιμή της αντικειμενικής τους συνάρτησης είναι ίδια, είναι και η βέλτιση για το ζεύγος αυτό των προβλημάτων. Γενικά, κάποιοι αλγόριθμοι αναφέρονται στο *δυικό χάσμα* για να δείξουν κατά τον χρόνο εκτέλεσής τους την πρόοδο σύγκλισης που παρουσιάζουν. Αυτό γιατί όταν το δυικό χάσμα γίνει ίσο με μηδέν σύμφωνα με το προηγούμενο Θεώρημα, έχει επιτευχθεί η βελτιστότητα.

Πρωτεύοντα και δυικά προβλήματα, όπως ήδη είδαμε είναι στενά συνδεδεμένα μεταξύ τους. Η κατάσταση της λύσης του ενός χαρακτηρίζει και τον τύπο της λύσης του άλλου και αντίστροφα. Μπορούμε να φανταστούμε το ζεύγος αυτό σαν ένα ζεύγος δυνάμεων ζήτησης και προσφοράς όπως μας είναι γνωστό ότι συμβαίνει στην ελεύθερη αγορά. Το σημείο ισορροπίας των δύο αυτών αντικρουώμενων δυνάμεων είναι και το βέλτιστο, που ικανοποιεί και τις δύο πλευρές.

Πολλοί αλγόριθμοι ονομάζονται επίσης δυικοί. Αυτό σημαίνει πως απαιτούν μια δυικά εφικτή λύση κατα την διαδικασία αρχικοποίησής τους και πως επιλύουν το δυικό πρόβλημα. Αυτό είναι βολικό όταν μια τέτοια λύση είναι εύκολα διαθέσιμη ενώ μια αρχικώς εφικτή όχι. Υπάρχουν επίσης αλγόριθμοι που κατασκευάζουν ταυτόχρονα δύο ακολουθίες σημείων, η μία εφικτή στο πρωτεύον και η άλλη στο δυικό αυτού. Ένας τέτοιος πρωτεύων - δυικός αλγόριθμος περιγράφεται από αυτή την εργασία. Γενικότερα η φύση των σημείων με τα οποία δουλεύουν οι διάφοροι αλγόριθμοι τους κατατάσσει και στις αντίστοιχες κατηγορίες. Οι αλγόριθμοι εσωτερικών σημείων ασχολούνται για παράδειγμα με σημεία που ικανοποιούν τους τεχνολογικούς περιορισμούς σαν αυστηρές ανισότητες. Τα μη εφικτά σημεία ονομάζονται και *εξωτερικά*, τα οποία και χρησιμοποιούν οι αλγόριθμοι Εξωτερικών Σημείων.

Το *ισχυρό* δυικό Θεώρημα είναι το ακόλουθο:

Θεώρημα 2.2.1 (Ισχυρή Συμπληρωματική Θεωρία) *Οι ακόλουθες σχέσεις ισχύουν για ένα ζεύγος πρωτεύοντος (P) και δυικού προβλήματος (D):*

1.Αν το (P) είναι βέλτιστο τότε και το (D) είναι βέλτιστο, και αντίστροφα.

2.Αν το (P) είναι απεριόριστο τότε το (D) είναι αδύνατο. Αν το (D) είναι απεριόριστο τότε το (P) είναι αδύνατο.

3.Αν το (P) είναι αδύνατο τότε το (D)μπορεί να είναι είτε αδύνατο είτε απεριόριστο και αντιστρόφως.

ΚΕΦΑΛΑΙΟ 3

Ο αναθεωρημένος αλγόριθμος Simplex

Πριν περάσουμε στην ανάλυση των αλγορίθμων γραμμικής βελτιστοποίησης, θα σταθούμε πρώτα σε μερικούς ουσιώδεις και σημαντικούς ορισμούς. Ας υποθέσουμε τώρα ότι έχουμε το ακολουθο γραμμικό πρόβλημα:

$$\begin{gathered}\min c^T x \\ \text{subject to } Ax + s = b \\ x, s \geq 0.\end{gathered}$$

όπου $c, x \in \Re^n$, $b, s \in \Re^m$ και τέλος $A \in \Re^{mxn}$. Επίσης θεωρούμε πως η μήτρα Α είναι πλήρους βαθμού για να μην είναι το πρόβλημα τετριμμένο, ότι δηλαδή δεν υπάρχουν γραμμικώς εξαρτημένοι περιορισμοί, καθώς και ότι $1 \leq m < n$.

Η *Βασική λίστα* περιέχει τους δείκτες των μεταβλητών που χρησιμοποιούνται για να κατασκευαστεί το τρέχον σημείο. Ουσιαστικά με αυτόν τον τρόπο αναφερόμαστε στις αντίστοιχες στήλες της μήτρας Α. Όλες οι υπόλοιπες μεταβλητές αποθηκεύονται στην *μη Βασική λίστα*. Έτσι έχουμε τώρα:

$$B = A._{BasicList}, N = A._{NonBasicList}$$

Με τον ίδιο τρόπο, χωρίζουμε και τα υπόλοιπα διανύσματα και το πρόβλημά μας παίρνει την μορφή:

$$\begin{gathered}\min c_B^T x_B + c_N^T x_N \\ \text{subject to } Bx_B + Nx_N + s = b \\ x_B, x_N, s \geq 0.\end{gathered}$$

Όταν είναι $x_B \geq 0$ το τρέχον σημείο είναι εφικτό. Το σύνολο των δεικτών που ονομάσαμε πριν *Βασική λίστα*, σχηματίζει μία *Βάση*, αν η τρέχουσα τετραγωνική μήτρα είναι αντιστρέψιμη. Τότε το σύνολο αυτό μας παρέχει ένα *Βασικό σημείο*. Τα βασικά σημεία μπορούν να είναι μόνο κορυφές της πολυεδρικής επιφάνειας που σχηματίζει η εφικτή περιοχή. Ένα βασικό σημείο δεν είναι απαραίτητα εφικτό. Η λύση του δυικού δίνεται από:

$$s = c - A^T w$$

όπου w είναι οι γνωστοί *πολλαπλασιαστές Simplex* και υπολογίζονται από:

$$w^T = (c_B)^T B^{-1}$$

Όπως ήδη γνωρίζουμε, η τρέχουσα βάση είναι δυικά εφικτή αν ισχύει επίσης $s \geq 0$.

3.1 Χαρακτηριστικά αλγορίθμων τύπου - Simplex - Περιγραφή Ψευδοκώδικα

Κάθε τέτοια μέθοδος κατασκευάζει ένα σύνολο από εφικτές λύσεις - διαμερίσεις. Παραλλαγές αλγορίθμων, μπορούν επίσης να κατασκευάζουν μη βασικά (εσωτερικά) ή μη εφικτά σημεία από επανάληψη σε επανάληψη (όλοι οι αλγόριθμοι γραμμικής βελτιστοποίσης είναι επαναληπτικοί), για να αποφύγουν κορυφές και να φτάσουν έτσι στην βέλτιστη λύση πιο γρήγορα από το να ακολουθήσουν αυστηρά γειτονικές κορυφές - μονοπάτια. Ας περάσουμε τώρα στην μαθηματική περιγραφή των βημάτων του πρωτεύοντος αλγορίθμου Simplex:

Βήμα 0 (Αρχικοποίηση) : Ξεκίνα με μια βασική εφικτή διαμέριση (B,N). Υπολόγισε την μήτρα B^{-1} και τα διανύσματα x_B, s_N, w^T από τις σχέσεις :

$$x_B = B^{-1}b$$
$$w = c_B B^{-1}$$
$$s_N = c_N - wA_N$$

Βήμα 1 (Έλεγχος βελτιστότητας) : Άν $s_N \geq 0$ STOP. Το πρόβλημα είναι βέλτιστο.

Βήμα 2 (Επιλογή εισερχόμενης/εξερχόμενης μεταβλητής) :

α) Επέλεξε την εισερχόμενη μεταβλητή από την σχέση:

Κανόνας Dantzig :

$$s_l = min\{s_j : s_j < 0 \wedge j \in N\}$$

Η μεταβλητή x_l είναι εισερχόμενη.

β) Υπολόγισε την στήλη περιστροφής από την σχέση :

$$h_l = \overline{B}^{-1} A_{.l}$$

Αν ισχύει $h_{il} \leq 0$ STOP. Το πρόβλημα είναι απεριόριστο. Διαφορετικά επέλεξε την εξερχόμενη μεταβλητή x_k από την σχέση :

$$x_k = x_{B[r]} = \frac{(B^{-1}b)_r}{h_{rl}} = min\left\{\frac{(B^{-1}b)_i}{h_{rl}} : h_{rl} > 0 \wedge i = 1, 2, ...m\right\}$$

Βήμα 3 (Περιστροφή) : Ανανέωσε τα σύνολα δεικτών B και N σύμφωνα με :

$$B \longleftarrow B|\{k\} \cup \{l\}$$
$$N \longleftarrow N|\{l\} \cup \{k\}$$

Υπολόγισε την νέα αντίστροφη της βάσης και ανανέωσε τα διανύσματα x_B, s_N. Πήγαινε στο Βήμα 1.

3.2 Ένα παράδειγμα εκτέλεσης του κώδικα

Ας υποθέσουμε ότι επιθυμούμε να λύσουμε το μετροπρόβλημα adlittle. Στο command window του MATLAB έχουμε:

revised-simplex

Please give the name of the linear problem mat file:adlittle

Loading from: matlab.mat

Do you want scaling (1(YES)/2(NO)):2

Define the number of re-inversion calculations:-1

Current default tolerance is: 1.0000e-008

Change values?(y—n):n

Starting Revised Primal Simplex algorithm:

Slack variables have been added.

Elapsed time in seconds:

0

Selecting initial basis:

Selection done.

Elapsed time in seconds:

0.1092

ENTERING PHASE 1

Elapsed time in seconds:

0.0936

ENTERING PHASE 2

Iter:100 2.254950E+005 feas

Optimization terminated successfully.

Elapsed time is 0.262347 seconds.

Βλέπουμε ότι ο αλγόριθμος χρειάστηκε 100 επαναλήψεις για να καταλήξει στην σωστή βέλτιστη αντικειμενική τιμή του προβλήματος Z=2.254950E+005.

ΚΕΦΑΛΑΙΟ 4

Ένας αλγόριθμος Εξωτερικών σημείων τύπου Simplex

Ένας τρόπος να βελτιώσουμε την υπολογιστική συμπεριφορά ενός αλγορίθμου γραμμικής βελτιστοποίησης είναι να μετακινούμαστε σε *μη γειτονικές* κορυφές. Αυτό μπορεί να γίνει αν κινηθούμε εκτός των συνόρων του πολυέδρου $P = \{x|Ax \leq b, x \geq 0\}$, κατασκευάζοντας έτσι μη εφικτά μονοπάτια. Τέτοιοι αλγόριθμοι ονομάζονται αλγόριθμοι *Εξωτερικών Σημείων* ακριβώς γιατί χειρίζονται εξωτερικά μή εφικτά σημεία.

Ένας τέτοιος αλγόριθμος (εν συντομία EPSA) κατασκευάστηκε για πρώτη φορά από τον *Paparrizos* το 1991 για το πρόβλημα αντιστοίχησης. Αργότερα ο ίδιος ερευνητής γενίκευσε τον αλγόριθμό του για γραμμικά προβλήματα (1993).

Σε κάθε επανάληψη του πρωτεύοντος αλγορίθμου Simplex, πραγματοποιείται μια αλλαγή μιας στήλης της βασικής λίστας (ουσιαστικά ένα στοιχείο που αποτελεί δείκτη στήλης για την μήτρα A) με μια της μη βασικής. Η διαδικασία αυτή είναι γνωστή και ως *περιστροφή*. Η μεταβλητή που εισέρχεται στην βασική λίστα ονομάζεται *εισερχόμενη* και αντίστοιχα αυτή που εισέρχεται στην μη βασική λίστα ονομάζεται *εξερχόμενη*. Γεωμετρικά αυτό σημαίνει ότι ο αλγόριθμος μετακινείται κατά μήκος των κορυφών του πολυέδρου $P = \{x|Ax \leq b, x \geq 0\}$. Αυτό το μονοπάτι είναι γνωστό και ως *Μονοπάτι Simplex*.

Οι αλγόριθμοι εξωτερικών σημείων χρησιμοποιούν δύο μονοπάτια για να φτάσουν στην βέλτιστη λύση. Ένα εφικτό και ένα μη εφικτό. Για τον λόγο αυτό δεν ακολουθούν κατανάγκην γειτονικές κορυφές εξετάζοντάς τες μία -προς -μία αλλά μπορούν να τις αποφύγουν φτάνοντας έτσι πιο γρήγορα στην βέλτιστη λύση. Σε κάθε περίπτωση δεν πρέπει ποτέ να χαθεί η επαφή με την εφικτή περιοχή. Αυτό σημαίνει ότι σε κάθε επανάληψη η επιλεγμένη βελτιώνουσα κατεύθυνση πρέπει να διασχίζει την πολυεδρική επιφάνεια της εφικτής περιοχής. Διαφορετικά το να

βρεθεί η *βέλτιστη κορυφή* μπορεί να αποδειχθεί πολύ δύσκολο ή και αδύνατο. Την επαφή αυτή την διατηρεί ο αλγόριθμος χάριν στο εφικτό του μονοπάτι, το οποίο όμως δεν είναι *Μονοπάτι Simplex*.

Οι αλγόριθμοι Εξωτερικών σημείων κατασκευάζουν μια ακολουθία από βασικά σημεία. Ας υποθέσουμε ότι η B είναι η τρέχουσα βάση. Γίνεται το τεστ βελτιστότητας. Αν το τρέχον σημείο δεν είναι βέλτιστο γίνεται μια προσπάθεια να επιλεγεί η εξερχόμενη και εισερχόμενη μεταβλητή. Αν δεν μπορεί να επιλεγεί μια εξερχόμενη μεταβλητή, το πρόβλημα είναι απεριόριστο. Η επιλογή τόσο της εξερχόμενης όσο και της εισερχόμενης μεταβλητής γίνεται μέσω του επιλεγμένου *κανόνα περιστροφής* που υλοποιεί ο εκάστοτε αλγόριθμος. Ο κανόνας αυτός είναι συνήθως κριτικής σημασίας για την υπολογιστική συμπεριφορά του αλγόριθμου.

4.1 Κατασκευή αρχικά εφικτής διαμέρισης

Ο αλγόριθμος των εξωτερικών σημείων που περιγράφουμε εδώ απαιτεί μια αρχικά εφικτή διαμέριση για ξεκινήσει, όπως και ο αλγόριθμος Simplex. Ο τρόπος με τον οποίο κατασκευάζεται το τροποποιημένο πρόβλημα της Φάσης I περιγράφεται αναλυτικά στην Ενότητα 5.1. Εδώ όμως θα πρέπει να σημειώσουμε τα εξής. Ενώ στον αλγόριθμο Simplex η κατασκευή συνεχώς βασικών εφικτών διαμερίσεων είναι εγγυημένη από τον τρόπο που λειτουργεί ο αλγόριθμος εκ φύσεως του, στον αλγόριθμο των εξωτερικών σημείων η διατήρηση της εφικτότητας από επανάληψη σε επανάληψη δεν είναι εγγυημένη καθώς όπως είπαμε επισκέπτεται και εξωτερικά μη εφικτά σημεία κατά την περιστροφικότητά του.

Για τον λόγο αυτό όταν η τεχνητή μεταβλητή εξέλθη της βάσεως, δεν σημαίνει απαραίτητα ότι μπορούμε να μεταβούμε στην Φάση II απευθείας. Θα πρέπει πρώτα λοιπόν να γίνει ο έλεγχος εφικτότητας του τρέχοντος σημείου. Αν το τρέχον σημείο είναι εφικτό τότε η Φάση I μπορεί να τερματιστεί άμεσα. Αν όμως δεν είναι μπορούμε να ακολουθήσουμε δύο διαφορετικές τεχνικές για τον τερματισμό της. Η πρώτη τερματίζει την Φάση I μόνο όταν το πρόβλημα της Φάσης I έχει βελτιστοποιηθεί, όταν αυτό είναι δυνατό φυσικά. Έτσι η εφικτότητα του σημείου με το οποίο περνάμε στην Φάση II είναι δεδομένη όπως γνωρίζουμε. Η δεύτερη τεχνική, αυτή που υλοποιείται και προγραμματιστικά εδώ, συμπεριφέρεται όπως η πρώτη μόνο ως προς την χειρότερη περίπτωσή της. Έτσι από την στιγμή που η τεχνητή μεταβλητή γίνει εξερχόμενη μέχρι και το τέλος της Φάσης I, πραγματοποιείται ανα επανάληψη έλεγχος εφικτότητας έτσι ώστε να μπορέσουμε να τερματίσουμε την Φάση I πριν φτάσουμε αναγκαστικά και στην επίλυση

του προβλήματος που περιέχει η φάση αυτή.

4.2 Βήματα του αλγορίθμου

Βήμα 0 (Αρχικοποίηση): Ξεκίνα με μια βασική εφικτή διαμέριση (B,N). Υπολόγισε την B^{-1} καθώς και τα διανύσματα x_B, w^T, s_N. Βρες τα σύνολα δεικτών P,Q χρησιμοποιώντας την ακόλουθη σχέση :

$$P = \{j \in N : s_j < 0\} \text{ ανδ } Q = \{j \in N : s_j \geq 0\}$$

Επέλεξε ένα αφηρημένο διάνυσμα $\lambda = (\lambda_1, \lambda_2, ..., \lambda_{|P|}) > 0$ και υπολόγισε το s_0 με χρήση του

$$s_0 = \sum \lambda_j s_j, j \in P$$

Επίσης υπολόγισε την κατεύθυνση d_B από

$$d_B = -\sum h_j, j \in P \text{ όπου } h_j = B^{-1}A_{.j}.$$

Βήμα 1 (Βήμα τερματισμού):

ι.(Τεστ Βελτιστότητας): Αν $P = \oslash .STOP$. Το πρόβλημα είναι βέλτιστο.
ιι.(Επιλογή εξερχόμενης μεταβλητής): Άν $d_B \geq 0, STOP$. Αν επίσης είναι $s_0 = 0$, το πρόβλημα είναι βέλτιστο. Αν $s_0 = 0$, το πρόβλημα είναι απεριόριστο. Διαφορετικά επέλεξε την εξερχόμενη μεταβλητή $x_{B(r)} = x_k$ με χρήση του:

$$\alpha = \frac{x_{B(r)}}{-d_{B(r)}} = \left\{ \frac{x_{B(i)}}{-d_{B(i)}} : d_{B(i)} < 0 \right\}$$

Βήμα 2 (Επιλογή εισερχόμενης μεταβλητής): Υπολόγισε τα διανύσματα

$$H_{rP} = B_{r.}^{-1}A_{.p} \text{ και } H_{rQ} = B_{r.}^{-1}A_{.q}$$

Βρες τα θ_1 και θ_2 χρησιμοποιώντας τις παρακάτω σχέσεις:

$$\theta_1 = -\frac{-s_Q}{H_{rQ}} = min \left\{ \frac{-s_j}{H_{rj}} : H_{rj} > 0, j \in P \right\}$$
$$\theta_2 = -\frac{-s_Q}{H_{rQ}} = min \left\{ \frac{-s_j}{H_{rj}} : H_{rj} < 0, j \in Q \right\}$$

Εντόπισε τους δείκτες t_1, t_2 έτσι ώστε $P(t_1) = p$ και $Q(t_2) = q$. Αν $\theta_1 \leq \theta_2$ θέσε $l = p$. Διαφορετικά $l = q$. Η μη βασική μεταβλητή x_l εισέρχεται στην βάση.

Βήμα 3 (Περιστροφή): Θέσε $B[r] = l$. Άν $\theta_1 \leq \theta_2$ θέσε $P \leftarrow P \ \{l\}$ και $Q \leftarrow Q \cup \{k\}$. Διαφορετικά $Q[t_2] = k$. Κάνοντας χρήση της νέας πλέον διαμέρισης (B,N), όπου $N = \{P, Q\}$ υπολόγισε την νέα B^{-1} και ανανέωσε αντίστοιχα τα διανύσματα x_B, w^T, s_N. Επίσης ανανέωσε την κατεύθυνση d_B από :

$$d_B = E^{-1} d_B$$

Άν $l \in P$ τότε $d_{B(r)} \leftarrow d_{B(r)} + l_l$. Πήγαινε στο Βήμα 1.

4.3 Ένα παράδειγμα εκτέλεσης του κώδικα

Για την επίλυση του μετροπροβλήματος adlittle έχουμε:

epsa2-b

Please give the name of the linear problem mat file:adlittle

Loading from: matlab.mat

Starting Exterior Point Simplex algorithm:

Slack variables have been added.

Elapsed time in seconds:

0

Selecting initial basis:

Selection done.

Elapsed time in seconds:

0.0156

flag = 0

ENTERING PHASE 1

-1.0198e-011

-1.0198e-011

Artificial variable left the basis and current direction crosses the feasible region.Exiting

Phase I...

Elapsed time in seconds:

0.2184

ENTERING PHASE 2

-Inf

Current direction crosses feasible region.

T H E L P I S O P T I M A L

O b j e c t i v e:

Z = 2.254949631623801e+005

Iterations =

130

Elapsed time in seconds:

0.1404

Total time in seconds:

0.5773

ΚΕΦΑΛΑΙΟ 5

Ο πρωτεύων δυικός αλγόριθμος τύπου **Simplex** δύο δρόμων **PDTPSA**

Σε αυτό το κεφάλαιο παρουσιάζουμε έναν νέο αλγόριθμο για το γραμμικό πρόβλημα. Αυτός ο αλγόριθμος μπορεί να θεωρηθεί και ως μια παραλλαγή ενός απλού δυικού αλγορίθμου. Τα πλεονεκτήματα της νέας αυτής μεθόδου είναι τα ακόλουθα:

- ευκολία στην υλοποίηση
- μικρό υπολογιστικό κόστος
- πολύ καλή πρακτική συμπεριφορά
- δυνατότητα να συνδυαστεί με μεθόδους εσωτερικών σημείων για την απόκτηση ενός αρχικού σημείου ξεκινήματος - μικρός αριθμός ενεργών επαναλήψεων
- σταθερή συμπεριφορά, χωρίς προβλήματα ακρίβειας
- ευκολία στην απόκτηση ενός δυικά εφικτού σημείου

Ένα από τα δύο μειονεκτήματα του αλγόριθμου εξωτερικών σημείων είναι η δυσκολία στην κατασκευή μιας καλής βελτιώνουσας κατεύθυνσης. Χρησιμοποιούμε τον όρο *καλή* εδώ με χαλαρότητα. Το δεύτερο είναι ότι δεν υπάρχει γνωστή μέχρι τώρα μέθοδος για να μπορεί να περάσει ο αλγόριθμος αυτός στο εσωτερικό της εφικτής περιοχής έτσι ώστε να παρέχεται ευελιξία την επιλογή της κίνησης. Με την μετατροπή του αλγορίθμου αυτού σε έναν δυικά εφικτό αλγόριθμο τα προβλήματα αυτά αφαιρούνται.

Αυτός ο νέος αλγόριθμος μπορεί να θεωρηθεί επίσης και ως μια μέθοδος για να περάσουμε από ένα εσωτερικό σημείο στην βέλτιστη λύση. Δεν συντρέχει λόγος όμως να κατασκευάσουμε ένα αρχικό εσωτερικό σημείο κάνοντας χρήση κάποιας IPM μεθόδου. Χρειαζόμαστε απλά ένα

αρχικό εσωτερικό σημείο, ή ένα όχι απαραίτητα βασικό αλλά σίγουρα εφικτό, το οποίο θα χρησιμοποιήσουμε σαν σημείο ξεκινήματος. Το σημείο αυτό μας παρέχει μια κατεύθυνση που διασχίζει την εφικτή περιοχή. Το επόμενο βήμα είναι να εντοπίσουμε μια δυικά εφικτή διαμέριση. Αυτό μπορεί να γίνει με χρήση ενός τροποποιημένου προβλήματος. Τότε η μέθοδός μας είναι έτοιμη να εφαρμοστεί. Ας τα δούμε όμως όλα αυτά πιο αναλυτικά.

5.1 Κατασκευή αρχικά εφικτής διαμέρισης

Όπως είδαμε προηγουμένως ο αλγόριθμος αυτός απαιτεί την κατασκευή μιας αρχικά εφικτής διαμέρισης. Όταν λέμε αρχικά, εννοούμε πως αναφέρεται στο αρχικό μας πρόβλημα και όχι στο δυικό αυτού. Μια τέτοια απαίτηση είναι εύκολο να πραγματοποιηθεί με ένα τροποποιημένο πρόβλημα με μία τεχνητή μεταβλητή. Ένα τέτοιο πρόβλημα είναι γνωστό ότι κατασκευάζει μια εφικτή διαμέριση καθώς δίνεται προτεραιότητα στην έξοδο της τεχνητής μεταβλήτης κατά την περιστροφικότητα και το ίδιο το πρόβλημα εκ της κατασκευής του διαθέτει πάντα μια εφικτή λύση για να ξεκινήσει ένας αλγόριθμος να το επιλύσει. Έτσι εφαρμόζοντας τον πρωτεύων αλγόριθμο Simplex στο τροποποιημένο πρόβλημα, επειδή από την φύση του ο αλγόριθμος κατασκευάζει αλληλουχίες εφικτών διαμερίσεων, όταν η τεχνητή μεταβλητή εξέλθη της βάσεως, η εφικτότητα παραμένει στην επόμενη επανάληψη, και η ίδια βασική λίστα θα παρέχει επομένως και εφικτή διαμέριση στο αρχικό μας πρόβλημα. Βέβαια αν το πρόβλημα βελτιστοποιηθεί και η τεχνητή μεταβλητή παραμένει στην βάση, πραγματοποιούμε μια προσεκτική εναλλαγή δεικτών και απαλλαγόμαστε από αυτήν.

Το τροποποιημένο πρόβλημα της Φάσης Ι κατασκευάζεται ως εξής:
Εισάγουμε την τεχνητή μεταβλητή x_{n+1} στο αρχικό μας πρόβλημα ως μία επιπλέον στήλη στην μήτρα Α η οποία δίνεται από:

$$d = -A_B e$$

όπου e είναι ένα διάνυσμα στήλη με όλα του τα στοιχεία ίσα με την μονάδα και τόσες γραμμές όσες και η μήτρα Α. Το πρόβλημα που επιλύεται στην Φάση Ι έχει την παρακάτω μορφή:

$$\min x_{n+1}$$

$$\text{subject to } A_x + dx_{n+1} = b$$
$$x, x_{n+1} \geq 0.$$

Πρέπει τώρα να γίνει μια περιστροφή ώστε η τεχνητή μεταβλήτη να εισαχθεί στην βάση. Έτσι η εξερχόμενη μεταβλητή επιλέγεται με την παρακάτω σχέση:

$$x_k = x_{B[r]} = min\left\{x_{B[i]} : i = 1, 2, ..., m\right\}$$

και ανανεώνονται τα σύνολα δεικτών B,N ώς εξής :

$$B \longleftarrow B|\left\{k\right\} \cup \left\{n+1\right\}$$
$$N \longleftarrow N|\left\{n+1\right\} \cup \left\{k\right\}$$

Η στήλη περιστροφής που αντιστοιχεί στην τεχνητή μεταβλήτη υπολογίζεται από την ακόλουθη σχέση:

$$h_{n+1} = B^{-1}A_{.(n+1)}$$

Αυτή η επιλογή εισερχόμενης και εξερχόμενης μεταβλητής εγγυάται την κατασκευή εφικτής βάσης για να μπορέσουμε να εφαρμόσουμε τον αλγόριθμο Simplex στο τροποποιημένο μας πρόβλημα.

Με αυτή την διαδικασία κατασκευάζουμε ουσιαστικά το ζητούμενο εφικτό σημείο y για τον PDTPSA. Επίσης το αν το πρόβλημά μας είναι αδύνατο το αντιλαμβανόμαστε όταν ισχύει $x_{n+1} > 0$.

5.2 Κατασκευή δυικά εφικτής διαμέρισης

Η κατασκευή ενός δυικά εφικτού σημείου γίνεται με παρόμοιο τρόπο. Ας υποθέσουμε ότι το αρχικό γραμμικό πρόβλημα που επιθυμούμε να λύσουμε είναι το ακόλουθο:

$$\min c^T x$$
$$\text{subject to } Ax + s = b$$
$$x, s \geq 0.$$

Κατασκευάζουμε τώρα ένα δεύτερο τροποποιημένο πρόβλημα που διαφοροποιείται όπως φαίνεται παρακάτω, σε σχέση με το αρχικό:

$$\min c^T x$$
$$\text{subject to } Ax + s = 0$$
$$x, s \geq 0.$$

Παρατηρούμε ότι το δεξί μέρος είναι εξολοκλήρου ίσο με το μηδέν. Αυτό μας δίνει την δυνατότητα να εφαρμόσουμε εξαρχής σε αυτό τον πρωτεύων αλγόριθμο Simplex καθόσον η αρχική μας βάση θα είναι σίγουρα εφικτή αφού

$$x_B = B^{-1}b = 0 \ \geq 0$$

Το πρόβλημα αυτό έχει τον ίδιο τύπο λύσης με το αρχικό. Αν το αρχικό μας είναι βέλτιστο ή απεριόριστο τότε αντίστοιχα είναι και αυτό, διότι η ύπαρξη ή μη όπως γνωρίζουμε των ακροτάτων σε μια συνάρτηση δεν επηρεάζεται από την ύπαρξη σταθερών όρων. Το πρόβλημα αυτό δεν είναι αδύνατο καθότι υπάρχει σίγουρα η προφανής λύση $x = 0$. Ανεξάρτητα από αυτό, έχει νόημα να προχωρήσουμε στην κατασκευή του μόνο εάν η προηγούμενη Φάση έχει αποφανθεί ότι το αρχικό μας πρόβλημα δεν είναι αδύνατο. Άρα είναι είτε βέλτιστο είτε απεριόριστο. Αν τώρα είναι βέλτιστο, η βέλτιστη λύση του είναι δυικά εφικτή όπως γνωρίζουμε σε αυτό το πρόβλημα αλλά επίσης και στο αρχικό διότι το δεξί μέρος που μηδενίσαμε και είναι η μόνη διαφορά μεταξύ των δύο προβλημάτων δεν συμμετέχει στον υπολογισμό των δυικά χαλαρών μεταβλητών. Άρα έχουμε κατασκευάσει επιτυχώς μια δυικά εφικτή διαμέριση για το αρχικό μας πρόβλημα. Αν τώρα είναι απεριόριστο τότε είναι απεριόριστο και το αρχικό μας πρόβλημα, κάτι το οποίο αποδεικνύεται εύκολα με την χρήση των οριζουσών για το σύστημα των περιορισμών. Αυτό γιατί αν το σύστημα αυτό είναι αόριστο, τότε είναι αόριστο και για οποιοδήποτε δεξιό μέρος του.

5.3 Βήματα του αλγορίθμου

Είμαστε τώρα έτοιμοι να περάσουμε στην αυστηρή μαθηματική δομή του αλγορίθμου. Χρησιμοποιούμε το πρότυπο /*...*/ για να *αγκαλιάσουμε* τα σχόλια. Βήματα του αλγορίθμου:

Βήμα 0 (Αρχικοποίηση): Ξεκίνα με μια αρχικώς εφικτή λύση y και μία δυικά εφικτή διαμέριση *(B,N)*. Θέσε
$x_B = (A_B)^{-1}b$, $w^T = (c_B)^T(A_B)^{-1}$, $(s_N)^T = (c_N)^T - w^T A_N$, $d = y - x$.

Βήμα 1 (Γενικό βήμα):

όσο $(\exists \in \{1, 2, ..., m\} : x_{B(i)} < 0)$

$\lambda = \frac{x_{B(r)}}{-d_{B(r)}} = max\left\{\frac{x_{B(i)}}{-d_{B(i)}} : x_{B(i)} < 0\right\}$

/* Επιλογή της εξερχόμενης μεταβλητής $x_{B(r)} = x_k$ */

$y = x + \lambda d$

/* Υπολογισμός του επόμενου σημείου y */

$H_{rN} = (B_{r.}^{-1}A_N)$

/* Υπολογισμός της γραμμής περιστροφής */

$\mu = \frac{s_{N(t)}}{H_{rN(t)}} = min\left\{\frac{s_{N(j)}}{-H_{rN(j)}} : H_{rN(j)} < 0\right\}$

/* Επιλογή της εισερχόμενης μεταβλητής $x_{N(t)} = x_p$ */

$k = B(r), p = N(t), B(r) = p, N(t) = k$

/* Ανανέωση των σετ B, N*/

$x_B = (A_B)^{-1}b, w^T = (c_B)^T(A_B)^{-1}, (s_N)^T = (c_N)^T - w^T A_N$
$d = y - x$

/* Υπολογισμός της επόμενης κατεύθυνσης */

τέλος

5.4 Εφαρμογή του αλγορίθμου

Συνοψίζοντας μπορούμε να χωρίσουμε τον αλγόριθμό μας σε 3 φάσεις υλοποίησης :

α) Κατασκευή εφικτής διαμέρισης με χρήση Simplex σε τροποποιημένο πρόβλημα με μία τεχνητή μεταβλητή

β) Κατασκευή δυικά εφικτής διαμέρισης με χρήση Simplex σε τροποποιημένο πρόβλημα με δεξιό μέρος μηδέν

γ) Εφαρμογή του PDTPSA στο αρχικό μας πρόβλημα κάνοντας χρήση των δύο προηγούμενων διαμερίσεων.

Θα αναλύσουμε εδώ ένα παράδειγμα για να γίνει πιο κατανοητή η υλοποίηση του αλγορίθμου. Ας υποθέσουμε ότι το γραμμικό μας πρόβλημα μετά την προσθήκη των χαλαρών μεταβλητών είναι το ακόλουθο

$$\begin{gathered}\min x_1 + x_2 + x_3 + x_4 \\ \text{s.t } x_1 - x_2 + x_3 - x_5 = 16 \\ x_2 + x_3 - x_4 - x_6 = -19 \\ x_i \geq 0, (i = 1:6)\end{gathered}$$

Έστω ότι επιλέγουμε ως βάση ξεκινήματος την B=[1 2], οπότε θα είναι και N=[3 4 5 6]. Τότε τα δεδομένα του προβλήματος διαμορφώνονται ως εξής :

$$\mathbf{c_B} = \begin{bmatrix} 1 & 1 \end{bmatrix} \quad , \mathbf{c_N} = \begin{bmatrix} 1 & 1 & 0 & 0 \end{bmatrix} \quad , \mathbf{w^T} = \begin{bmatrix} 1 & 2 \end{bmatrix} \quad , \mathbf{s_N} = \begin{bmatrix} -2 & 3 & 1 & 2 \end{bmatrix}$$
$$\mathbf{x_B} = \begin{bmatrix} -3 & -19 \end{bmatrix} \quad , \mathbf{B^{-1}} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$$

Παρατηρούμε ότι αντιμετωπίζουμε την χειρότερη περίπτωση όπου η τρέχουσα βάση μας δεν είναι εφικτή ούτε στο αρχικό αλλά ούτε και στο δυικό πρόβλημα. Περνάμε επομένως στην Φάση Ι όπου και κατασκευάζουμε ένα τροποποιημένο πρόβλημα με μία τεχνητή μεταβλητή και το επιλύουμε με τον πρωτεύων αλγόριθμο Simplex.

Βήμα 1. (Κατασκευή αρχικά εφικτής διαμέρισης) Εισάγουμε την τεχνητή μεταβλητή x_7 στην βάση μας σύμφωνα πάντα με την μέθοδο που περιγράφηκε οπότε τα νέα δεδομένα του προβλήματος είναι τα ακόλουθα :

$$\mathbf{B} = \begin{bmatrix} 1 & 7 \end{bmatrix} \quad , \mathbf{N} = \begin{bmatrix} 3 & 4 & 5 & 6 & 2 \end{bmatrix} \quad , \mathbf{c_B} = \begin{bmatrix} 0 & 1 \end{bmatrix} \quad , \mathbf{c_N} = \begin{bmatrix} 0 & 0 & 0 & 0 \end{bmatrix} \quad ,$$
$$\mathbf{w^T} = \begin{bmatrix} 0 & -1 \end{bmatrix} \quad , \mathbf{s_N} = \begin{bmatrix} 1 & -1 & 0 & -1 & 1 \end{bmatrix} \quad \mathbf{x_B} = \begin{bmatrix} 16 & 19 \end{bmatrix} \quad ,$$

$$\mathbf{B^{-1}} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad , t = 2 \quad , l = N(t) = 4 \ , \quad \mathbf{h_l} = \begin{bmatrix} 0 & 1 \end{bmatrix} \quad , x_B./h_l = 19$$
$$r = 2, k = B(r) = 7 \quad \text{Pivot } B(r) = l, N(t) = k \Longrightarrow$$
$$, \mathbf{B} = \begin{bmatrix} 1 & 4 \end{bmatrix} \quad , \mathbf{N} = \begin{bmatrix} 3 & 7 & 5 & 6 & 2 \end{bmatrix}$$

Βλέπουμε φυσικά ότι η πρώτη βάση μας εδώ είναι αρχικά εφικτή κάτι που μας το εγγυάται όπως προαναφέραμε η μέθοδος κατασκευής του τροποποιημένου προβλήματος. Η περιστροφή που προηγήθηκε επέλεξε ως εξερχόμενη μεταβλητή την τεχνητή. Έτσι σε αυτή την επανάληψη οι αντίστοιχες μήτρες και διανύσματα διαμορφώνονται ως εξής :

$$\mathbf{c_B} = \begin{bmatrix} 0 & 0 \end{bmatrix} \quad , \mathbf{c_N} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \end{bmatrix} \quad , \mathbf{w^T} = \begin{bmatrix} 0 & -1 \end{bmatrix} \quad ,$$
$$\mathbf{s_N} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \end{bmatrix} \quad \mathbf{x_B} = \begin{bmatrix} 16 & 19 \end{bmatrix}$$

Η τρέχουσα βάση είναι όπως βλέπουμε εφικτή (και βέλτιστη ταυτόχρονα για το πρόβλημα της Φάσης Ι) στο αρχικό μας πρόβλημα. Κρατάμε λοιπόν την διαμέριση

$$\mathbf{B} = \begin{bmatrix} 1 & 4 \end{bmatrix} \quad , \mathbf{N} = \begin{bmatrix} 3 & 5 & 6 & 2 \end{bmatrix}$$

ως αρχικά εφικτή διαμέριση για την εφαρμογή του PDTPSA στην τελική τρίτη του Φάση.

Βήμα 2. (Κατασκευή δυικά εφικτής διαμέρισης)

Το τροποποιημένο πρόβλημα που επιλύουμε εδώ είναι το ακόλουθο :

$$\min x_1 + x_2 + x_3 + x_4$$
$$\text{s.t } x_1 - x_2 + x_3 - x_5 = 0$$
$$x_2 + x_3 - x_4 - x_6 = 0$$
$$x_i \geq 0, (i = 1 : 6)$$

Εδώ όπως είδαμε οποιαδήποτε αρχική επιλογή βασικής λίστας είναι εφικτή σε αυτό το πρόβλημα γιατί το δεξιό μέρος θα μηδενίζει πάντα την ποσότητα x_B. Ας συνεχίσουμε με την πρώτη διαμέριση που είχαμε επιλέξει και επομένως οι αντίστοιχες μήτρες και διανύσματα που προκύπτουν θα είναι :

$$\mathbf{B} = \begin{bmatrix} 1 & 2 \end{bmatrix} \quad , \mathbf{N} = \begin{bmatrix} 3 & 4 & 5 & 6 \end{bmatrix} \quad , \mathbf{c_B} = \begin{bmatrix} 1 & 1 \end{bmatrix} \quad , \mathbf{c_N} = \begin{bmatrix} 1 & 1 & 0 & 0 \end{bmatrix} \quad ,$$
$$\mathbf{w^T} = \begin{bmatrix} 1 & 2 \end{bmatrix} \quad , \mathbf{s_N} = \begin{bmatrix} -2 & 3 & 1 & 2 \end{bmatrix} \quad \mathbf{x_B} = \begin{bmatrix} 0 & 0 \end{bmatrix} \quad , \mathbf{B^{-1}} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$$

$$t = 1, l = 3, \quad \mathbf{h_l} = \begin{bmatrix} 2 & 1 \end{bmatrix} \quad \mathbf{x_B}./\mathbf{h_l} = \begin{bmatrix} 0 & 0 \end{bmatrix}$$
$$r = 1, k = 1, \text{Pivot } B(r) = l, N(t) = k \Longrightarrow \quad \mathbf{B} = \begin{bmatrix} 3 & 2 \end{bmatrix} \quad , \mathbf{N} = \begin{bmatrix} 1 & 4 & 5 & 6 \end{bmatrix} \quad ,$$
$$\mathbf{c_B} = \begin{bmatrix} 1 & 1 \end{bmatrix} \quad , \mathbf{c_N} = \begin{bmatrix} 1 & 1 & 0 & 0 \end{bmatrix} \quad , \mathbf{w^T} = \begin{bmatrix} 1 & 2 \end{bmatrix} \quad , \mathbf{s_N} = \begin{bmatrix} 1 & 2 & 0 & 1 \end{bmatrix}$$
$$\mathbf{x_B} = \begin{bmatrix} 0 & 0 \end{bmatrix} \quad , \mathbf{B^{-1}} = \begin{bmatrix} 1/2 & 1/2 \\ -1/2 & 1/2 \end{bmatrix}$$

Βλέπουμε ότι η τρέχουσα βάση είναι δυικά εφικτή σε αυτό το πρόβλημα της Φάσης II, και μάλιστα βέλτιστη. Είναι όμως δυικά εφικτή και στο αρχικό μας πρόβλημα καθώς με την ίδια διαμέριση παράγουμε και εκεί $s_N \geq 0$ αφού η μόνη διαφορά μεταξύ των δύο αυτών προβλημάτων είναι το μηδενικό δεξιό μέρος το οποίο όμως δεν συμμετέχει στον υπολογισμό των δυικά χαλαρών μεταβλητών. Επομένως την διαμέριση αυτή την κρατάμε ως δυικά εφικτή για το αρχικό μας πρόβλημα, για να την εφαρμόσουμε με τον αλγόριθμο PDTPSA στην τελική Φάση III.

Βήμα 3. (Εφαρμογή PDTPSA)

Η τελική τρίτη Φάση κάνει χρήση των δύο προηγούμενων διαμερίσεων που προέκυψαν από τα δύο αντίστοιχα τροποποιημένα προβλήματα των δύο πρώτων Φάσεων της υλοποίησης. Εδώ γίνεται η καθαρή εφαρμογή του αλγορίθμου PDTPSA. Έχουμε λοιπόν :

$$\underbrace{B = [1 \quad 4], N = [3 \quad 5 \quad 6 \quad 2]}_{PrimalFeasiblePartition} \quad , \underbrace{B = [3 \quad 2], N = [1 \quad 4 \quad 5 \quad 6]}_{DualFeasiblePartition}$$

Κάνοντας χρήση των δύο αυτών διαμερίσεων είμαστε έτοιμοι να ξεκινήσουμε την εφαρμογή του PDTPSA στο αρχικό μας πρόβλημα :

$$\mathbf{B} = \begin{bmatrix} 3 & 2 \end{bmatrix} \quad , \mathbf{N} = \begin{bmatrix} 1 & 4 & 5 & 6 \end{bmatrix} \quad , \mathbf{c_B} = \begin{bmatrix} 1 & 1 \end{bmatrix} \quad , \mathbf{c_N} = \begin{bmatrix} 1 & 1 & 0 & 0 \end{bmatrix} \quad ,$$
$$\mathbf{s_N} = \begin{bmatrix} 1 & 2 & 0 & 1 \end{bmatrix} \quad \mathbf{x_B} = \begin{bmatrix} -1.5 & -17.5 \end{bmatrix} \quad , \mathbf{B^{-1}} = \begin{bmatrix} 1/2 & 1/2 \\ -1/2 & 1/2 \end{bmatrix} \quad ,$$
$$\mathbf{d_B} = \begin{bmatrix} 17.5 & 36.5 \end{bmatrix} \quad , \mathbf{y} = \begin{bmatrix} 16 & 19 \end{bmatrix} \quad , \mathbf{LamdaRatio}(\mathbf{x_B}./\mathbf{d_B}) = \begin{bmatrix} 0.0857 & 0.4795 \end{bmatrix}$$
$$, p = 0.4795, r = 2, k = 2, \quad , \mathbf{H_r N} = \begin{bmatrix} -1/2 & -1/2 & 1/2 & -1/2 \end{bmatrix} \quad , t = 4, l = 6, \quad ,$$
$$\mathbf{h_l} = \begin{bmatrix} -1/2 & -1/2 \end{bmatrix}$$

Στην επόμενη περιστροφή έχουμε επομένως :

$$\mathbf{B} = \begin{bmatrix} 3 & 6 \end{bmatrix} \quad , \mathbf{N} = \begin{bmatrix} 1 & 4 & 5 & 2 \end{bmatrix} \quad , \mathbf{c_B} = \begin{bmatrix} 1 & 0 \end{bmatrix} \quad , \mathbf{c_N} = \begin{bmatrix} 1 & 1 & 0 & 1 \end{bmatrix} \quad ,$$
$$\mathbf{s_N} = \begin{bmatrix} 0 & 1 & 1 & 2 \end{bmatrix} \quad \mathbf{x_B} = \begin{bmatrix} 16 & 35 \end{bmatrix} \quad , \mathbf{B^{-1}} = \begin{bmatrix} 1 & 0 \\ 1 & -1 \end{bmatrix} \quad ,$$
$$\mathbf{d_B} = \begin{bmatrix} -9.1096 & -35.0000 \end{bmatrix} \quad , \mathbf{y} = \mathbf{x_B} + \mathbf{p} * \mathbf{d_B} = \begin{bmatrix} 6.8904 & 0 \end{bmatrix}$$

Παρατηρούμε ότι το πρόβλημα είναι βέλτιστο με βέλτιστη τιμή της αντικειμενικής συνάρτησης : $Z = c_B x_B = 16$.

5.5 Ένα παράδειγμα εκτέλεσης του κώδικα

Το ίδιο μετροπρόβλημα όπως και στα προηγούμενα παραδείγματα έχει ως εξής εδώ:

hybrid-f

Please give the name of the linear problem mat file:adlittle

Loading from: matlab.mat

Do you want scaling (1(YES)/2(NO)):2

Define the number of re-inversion calculations:-1

Current default tolerances are:

1.0000e-008

1.0000e-008

Change values?(y—n):n

Starting Primal - Dual Two Paths Pivoting Simplex Algorithm:

Slack variables have been added.

Elapsed time in seconds:

0.0312

Selecting initial basis:

Selection done.

Elapsed time in seconds:

0.0468

Starting to construct primal feasibility...

TEST 1.1 PASSED:Primal feasible partition found

Starting to construct dual feasibility...

TEST 1.2 PASSED:Dual feasible partition found

TEST 2 PASSED:Both Dual and Primal feasible partitions found:Ready to apply PDTPSA

Iter:107 2.254950E+005 feas

Optimization terminated successfully.

Elapsed time is 0.268150 seconds.

ΚΕΦΑΛΑΙΟ 6

Θέματα Υλοποίησης

Ο προγραμματισμός ενός αλγορίθμου γραμμικής βελτιστοποίησης διαφέρει πολύ από την απλότητα της περιγραφής του ψευδοκώδικά του. Γραμμικά προβλήματα μικρών διαστάσεων είναι συνήθως εύκολο να λυθούν και χωρίς να δίνεται ιδιαίτερη προσοχή στην υλοποίηση. Αυτό όμως δεν συμβαίνει όταν μιλάμε για υπολογισμούς μεγάλης κλίμακας. Εκεί θα αντιμετωπίσουμε σίγουρα προβλήματα διαχείρησης μνήμης, αριθμητικά σφάλματα, μαθηματικές μεθόδους μή ικανοποιητικές - επαρκείς πρακτικά πλέον, καθώς και άλλα λιγότερο σημαντικά.

6.1 Δεσμοί

Ένα κύριο χαρακτηριστικό των μετροπροβλημάτων είναι ότι όλα σχεδόν στο σύνολό τους είναι εκφυλισμένα. Αυτό δημιουργεί πρόβλημα στην περίπτωση που δεν λάβουμε τα κατάλληλα μέτρα για να σπάσουμε τους δεσμούς. Διαφορετικά η τιμή της αντικειμενικής συνάρτησης δεν θα βελτιώνεται από επανάληψη σε επανάληψη και θα έχουμε το φαινόμενο της *στασιμότητας* ή αλλιώς stalling. Στις υλοποιήσεις των αλγορίθμων δώσαμε προτεραιότητα στις μεταβλητές με την μικρότερη τιμή δείκτη στην βασική λίστα, εκτός από την περίπτωση της Φάσης I όπου λόγω του ότι θέλουμε να δώσουμε προτεραιότητα στην τεχνητή μεταβλητή εκεί επιλέγουμε την μέγιστη τιμή δείκτη, δεδομένου ότι η τεχνητή μεταβλητή έχει πάντα τον μέγιστο δείκτη αφού προστίθεται στο τέλος.

6.2 Προλυτικές διαδικασίες

Πριν από την επίλυση των βέλτιστων μετροπροβλημάτων που χρησιμοποιήθηκαν εδώ, εφαρμόστηκαν σε αυτά τεχνικές προλυτικών διαδικασιών. Οι διαδικασίες αυτές έχουν ως στόχο να

μειώσουν όσο το δυνατόν περισσότερο το μέγεθος των προβλημάτων πριν εφαρμοστεί σε αυτά ο αλγόριθμος - λύτης είτε αφαιρώντας μηδενικούς περιορισμούς, είτε γραμμικώς εξαρτημένους - περιττούς.

6.3 Τεχνικές κλιμάκωσης

Ένας από τους σημαντικότερους λόγους της αδυναμίας επίλυσης των γραμμικών προβλημάτων με ηλεκτρονικό υπολογιστή είναι η ύπαρξη στοιχείων περιστροφής με απόλυτη τιμή κοντά στο μηδέν. Η ύπαρξη τέτοιων στοιχείων οφείλεται στις μεγάλες διαστάσεις των προβλημάτων και στην διαφορά τάξης των αριθμητικών τιμών των δεδομένων. Μια διαίρεση με το μηδέν είναι λογικό ότι θα αποβεί απαγορευτική για την συνέχιση της εκτέλεσης του αλγορίθμου. Μια μήτρα Α που δεν δημιουργεί τέτοια προβλήματα ονομάζεται *καλά κλιμακωμένη*. Γραμμικά προβλήματα των οποίων οι μήτρες είναι καλά κλιμακωμένες μειώνουν την υπολογιστική προσπάθεια που απαιτείται για την επίλυσή τους καθώς αποφεύγονται τα αριθμητικά σφάλματα.

Εμείς κάναμε χρήση της τεχνικής της *ισσορόπησης* (equilibration technique) η οποία περιγράφεται ακολούθως :

Βήμα 1. Εύρεση του μεγαλύτερου κατά απόλυτη τιμή στοιχείου κάθε στήλης της μήτρας Α από την σχέση

$$n_j = max\left\{|a_{ij}| : i = 1, 2..., m\right\}, j = 1, 2, ..., n$$

και πολλαπλασιασμός της στήλης $A_{.j}$ με τον αριθμό $1/n(j)$ έτσι ώστε το μεγαλύτερο στοιχείο της να είναι +1 ή -1.

Βήμα 1. Εύρεση του μεγαλύτερου κατά απόλυτη τιμή στοιχείου κάθε γραμμής της μήτρας Α από την σχέση

$$m_j = max\left\{|a_{ij}| : j = 1, 2..., m\right\}, i = 1, 2, ..., n$$

και πολλαπλασιασμός της γραμμής $A_{i.}$ με τον αριθμό $1/m(i)$ έτσι ώστε το μεγαλύτερο στοιχείο της να είναι +1 ή -1.

Πρέπει να σημειωθεί ότι ενώ η εύρεση του μεγαλύτερου κατά απόλυτη τιμή στοιχείου κάθε στήλης και γραμμής γίνεται από την μήτρα A, ο πολλαπλασιασμός όμως με τον αριθμό $1/n(j)$ και $1/m(i)$ περιλαμβάνει και το διάνυσμα κόστους - δεξιού μέρους. Έτσι η βέλτιστη λύση που παράγεται με αυτόν τον τρόπο είναι και αυτή κλιμακωμένη, όχι όμως και η βέλτιστη τιμή που παραμένει ίδια στα κλιμακωμένο και αρχικό προβλήματα.

6.4 Ανοχές

Ένας αλγόριθμος ο οποίος συνήθως συγκλίνει μετά από μερικές χιλιάδες επαναλήψεις είναι δύσκολο να αποφύγει τα αριθμητικά σφάλματα όταν είναι υλοποιημένος σε ένα υπολογιστικό σύστημα. Για τον λόγο αυτό καθίσταται απαγορευτική η παράλειψη εφαρμογής ανοχών στον υπολογισμό όλων των ζωτικών μητρών και διανυσμάτων που υπολογίζονται και ανανεώνονται ανά επανάληψη του αλγορίθμου. Η συνήθης τιμή σε τέτοιες περιπτώσεις είναι η 1.0E-008. Η τιμή αυτή μπορεί να κυμαίνεται από 1.00E-010 έως 8.0E-005 αναλόγως τις απαιτήσεις της πολυπλοκότητας του προβλήματος που προσπαθούμε να επιλύσουμε.

6.5 Re-inversion

Ένα από τα μεγαλύτερα υπολογιστικά μειονεκτήματα σε αλγόριθμους γραμμικής βελτιστοποίησης είναι ο υπολογισμός της αντιστρόφου μήτρας της τρέχουσας ανά επανάληψη βάσεως. Αν φανταστούμε ότι η βάση αυτή συνήθως έχει μερικές χιλιάδες γραμμές και στήλες είναι εύκολο να αντιληφθούμε ότι ο υπολογισμός της με την μέθοδο των οριζουσών είναι απαγορευτικός για να πραγματοποιείται σε κάθε επανάληψη. Έτσι η τεχνική που χρησιμοποιούμε είναι η εξής. Αρχικά υπολογίζουμε κανονικά την αντίστροφη της βάσεως με την μέθοδο inv που είναι ήδη υλοποιημένη σαν built-in function του MATLAB. Σε κάθε επόμενη επανάληψη όμως και για ένα συγκεκριμένο αριθμό η ανανέωση γίνεται βάση της μεθόδου των *ήτα - μητρών* ή αλλιώς eta-matrices. Μόλις φτάσουμε τον αριθμό αυτό γίνεται ξανά χρήση της inv και ξανά η διαδικασία επαναλαμβάνεται. Μία καλή επιλογή πρακτικά είναι συνήθως η χρήση της inv να γίνεται ανά 60-80 επαναλήψεις.

6.6 Μετατροπή από την μορφή .MPS - Mathematical Programming Standard

Τα benchmarks είναι ένα σύνολο γνωστών πραγματικών γραμμικών προβλημάτων που υπάρχουν στο διαδίκτυο για να ελέγχει κανείς την αποδοτικότητα και την αξιοπιστία του αλγορίθμου που αναπτύσσει. Η μορφή στην οποία μπορεί κανείς να τα αποκτήσει κατεβάζοντας τα από το διαδίκτυο είναι η ευρέως διαδεδομένη MPS (Mathematical Programming Standard). Για παράδειγμα η δομή ενός γραμμικού προβλήματος στην μορφή αυτή είναι η εξής:

NAME Γ.Π1 (MIN)
ROWS
N OBJ
E R1
E R2
COLUMNS
X1 R1 1
X1 R2 1
X1 OBJ 2
X2 R1 2
X2 OBJ 4
X3 R2 3
X3 OBJ 9
X4 R1 2
X4 R2 -4
X4 OBJ 4
X5 R1 1
X5 R2 -1
X5 OBJ 1
RHS
RHS1 R1 2
RHS1 R2 -3

ENDATA

Το αρχείο MPS ξεκινά με την ονομασία του γραμμικού προβλήματος καθώς και τον τύπο αυτού, υπό την έννοια αν πρόκειται για πρόβλημα ελαχιστοποίησης ή μεγιστοποίησης. Στη συνέχεια με την λέξη ROWS μας γνωστοποιείται πως ακολουθούν οι τύποι των περιορισμών του, τα ονόματά τους καθώς και το όνομα της αντικειμενικής συνάρτησης. Τα τρία είδη των περιορισμών αντιπροσωπεύονται αντίστοιχα από:

$$\{L :\leq | E := | G :\geq\}$$

Επίσης η αναφορά στην αντικειμενική συνάρτηση γίνεται με το γράμμα N. Έτσι έχουμε 2 περιορισμούς εδώ, με τα ονόματα R1 και R2 που είναι και οι δύο ισοτικοί και την αντικειμενική συνάρτηση που ονομάζεται OBJ. Ακολουθεί το πεδίο COLUMNS που περιέχει την μήτρα συντελεστών A του γραμμικού προβλήματος, δηλαδή τους συντελεστές των μεταβλητών στους αντίστοιχους περιορισμούς. Έτσι από την πρώτη γραμμή του πεδίου αυτού βλέπουμε πως η μεταβλητή X1 στον πρώτο περιορισμό έχει τιμή ίση με την μονάδα. Στην τρίτη γραμμή του πεδίου αυτού βλέπουμε επίσης πώς η ίδια μεταβλητή στην αντικειμενική συνάρτηση έχει τιμή 2.

Είδαμε ως τώρα τα 3 πρώτα υποχρεωτικά πεδία (NAME, ROWS, COLUMNS) της αυστηρής μορφής MPS. Το τέταρτο υποχρεωτικό πεδίο RHS περιέχει τους σταθερούς όρους των περιορισμών. Υποχρεωτική είναι κάθε φορά η αναφορά του ονόματος RHS1 πριν από το όνομα κάθε περιορισμού και τον αντίστοιχο σταθερό του όρο. Εδώ βλέπουμε πως ο περιορισμός R1 έχει σταθερό όρο 2 και ο R2 ,-3. Η δεσμευμένη λέξη ENDATA δηλώνει το τέλος της περιγραφής του γραμμικού προβλήματος. Υπάρχουν ακόμα 2 προαιρετικά πεδία μετά από τους σταθερούς όρους, RANGES και BOUNDS, που εμφανίζονται μόνο όταν το γραμμικό πρόβλημα έχει αντίστοιχα όρια και περιοχές. Τα πεδία αυτά δεν θα αναλυθούν διότι τα προβλήματα που θα αναλύσουμε στη μελέτη αυτή δεν περιέχουν όρια ή περιοχές.

Μια διαφορετική εμφάνιση του αρχείου μπορεί να είναι η ακόλουθη

NAME Γ.Π 1 (MIN)
ROWS
N OBJ
E R1

E R2

COLUMNS

X1 R1 1 R2 1

X1 OBJ 2

X2 R1 2 OBJ 4

X3 R2 3 OBJ 9

X4 R1 2 R2 -4

X4 OBJ 4

X5 R1 1 R2 -1

X5 OBJ 1

RHS

RHS1 R1 2

RHS1 R2 -3

ENDATA

δηλαδή να γίνεται δίστηλη καταχώρηση, όπου αυτό είναι δυνατό στο πεδίο COLUMNS. Όλα αυτά βέβαια κάτω από αυστηρούς περιορισμούς σχετικά με το πόσα γράμματα μπορεί να έχει το όνομα του προβλήματος, σε ποιες στήλες γίνεται η αναφορά του, σε ποιο διάστημα στηλών καταχωρούνται οι αριθμοί, τα ονόματα των μεταβλητών και των περιορισμών κ.α.

Ο προγραμματιστής που επιθυμεί να μελετήσει τα benchmarks και να τα λύσει με τον δικό του αλγόριθμο θα πρέπει πρώτα να μετατρέψει τα προβλήματα αυτά από την μορφή αυτή σε μια άλλη, κατανοητή από το προγραμματιστικό περιβάλλον ανάπτυξης που χρησιμοποιεί. Εμείς χρησιμοποιώντας το MATLAB ως γλώσσα προγραμματισμού έπρεπε να μετατρέψουμε τα αρχεία mps σε μορφή μητρών (.mat), αφού είναι και η μορφή που χρησιμοποιεί το MATLAB. Γράφτηκαν επομένως δύο συναρτήσεις μετατροπής, mps2mat και mat2mps. Η πρώτη μετατρέπει οποιοδήποτε γ.π από την μορφή mps σε μορφή μητρών και η δεύτερη το αντίστροφο. Ειδικότερα η mat2mps ήταν απαραίτητη για την μετατροπή ενός τυχαίου γραμμικού προβλήματος που δημιουργούσαμε στο MATLAB σε μορφή mps για να είναι δυνατή η επίλυση του και από άλλα πακέτα (λύτες), όπως το LINDO που παρέχει υποστήριξη σε αρχεία mps. Θα αναλύσουμε τώρα τον τρόπο με τον οποίο η συνάρτηση mps2mat μετατρέπει ένα γραμμικό πρόβλημα της μορφής mps σε αρχείο .mat για χρήση του στο MATLAB.

Αρχικά πρέπει να σημειωθεί ότι δώθηκε μέγιστη βαρύτητα στην όσο το δυνατόν πιο μεγάλη

μείωση της υπολογιστικής πολυπλοκότητας χρόνου του αλγορίθμου που υλοποιεί η συνάρτηση mps2mat, διότι στην μετατροπή μεγάλων benchmarks ή γενικά μεγάλων αρχείων mps - ο χρόνος μετατροπής μεταβάλλεται υπερβολικά πολύ και ξεφεύγει από τα λογικά και πεπερασμένα χρονικά διαστήματα. Αξίζει να αναφερθεί ότι η πρώτη υλοποίηση της συνάρτησης για ένα συγκεκριμένο μετροπρόβλημα απαιτούσε περίπου 10 λεπτά, ενώ μετά την βελτιστοποίηση που επιδέχθηκε ο κώδικας για το ίδιο benchmark απαιτούσε 7 δεύτερα!

Η συνάρτηση δουλεύει χρησιμοποιώντας ενσωματωμένες συναρτήσεις του MATLAB για συγκεκριμένη ανάγνωση από αρχεία κειμένου (.txt, όπως δηλαδή τα mps), ανοίγοντας αρχικά το mps για σειριακή ανάγνωση. Στόχος της συνάρτησης είναι να αποθηκεύσει σε ένα αρχείο .mat με το ίδιο όνομα του αρχείου mps ουσιαστικά 4 μήτρες: A, c, b, Eqin, όπως εξηγήθηκε στο Κεφάλαιο 1. Αφού αποθηκευτεί το όνομα του αρχείου mps αρχίζει η ανάγνωση του πεδίου ROWS. Εδώ κρατάμε προσωρινά σε ένα κελί (cell στο Matlab), πίνακα στήλη, τα ονόματα των περιορισμών με την σειρά που διαβάζονται και στο τέλος προστίθεται και το όνομα της αντικειμενικής συνάρτησης.

Στο γραμμικό πρόβλημα που χρησιμοποιήσαμε εδώ για παράδειγμα το κελί αυτό κρατά:

$$X = \{R1, R2, OBJ\}$$

και ταυτόχρονα δημιουργείται ο πίνακας Eqin=[0 0] ως πίνακας στήλη. Εδώ το κελί Q ουσιαστικά μας δίνει την αρίθμηση για κάθε περιορισμό (ποιος είναι πρώτος, ποιος δεύτερος), ανάλογα με την γραμμή στην οποία βρίσκεται, ανεξαρτήτως ονόματος, δηλαδή πρώτος περιορισμός ο R1 και δεύτερος ο R2. Πρέπει να γίνει κατανοητό πως ενώ εδώ τα ονόματα των περιορισμών μας προδιαθέτουν για την αρίθμηση τους (από τον αριθμό που περιέχουν), αλλά αυτό δεν συμβαίνει πάντα. Τι γίνεται στην περίπτωση που το αρχείο mps έχει την παρακάτω μορφή

NAME 25FV47
ROWS
N R0000
E 2SF145
E 2SF089
E 2SF129

E 30M00

E 30M94

E 30M91

E 30ATK

E 30PGK

E 30G30

E 30G17

E 30G44

E 30G22

E 30D22

E 30D30

E 30121

E 30F52

E 30F48

E 30128

E RB017

●●●●

Εμείς απλά κρατάμε την σειρά με την οποία διαβάζεται κάθε γραμμή. Ουσιαστικά με αυτόν τον τρόπο κατοχυρώνουμε την αρίθμηση των περιορισμών. Ο τρόπος αυτός δεν είναι υποχρεωτικός, απλά πριν περάσουμε στην ανάγνωση του πεδίου COLUMNS πρέπει να έχουμε αποφασίσει ποιος περιορισμός είναι πρώτος, ποιος δεύτερος κ.ο.κ. Αυτό θα γίνει κατανοητό στην συνέχεια.

Προχωρώντας στο πεδίο COLUMNS τώρα, από όπου γίνεται ο σχηματισμός της μήτρας A, η ανάγνωση γίνεται ως εξής: Αφού κρατηθεί το όνομα της μεταβλητής σαν πρώτη μεταβλητή, ορίζοντας έτσι την στήλη στην οποία θα γίνει η καταχώρηση του συντελεστή, διαβάζεται το όνομα του περιορισμού στο οποίο γίνεται η αναφορά, και ταυτόχρονα ξεκινά μια αναζήτηση στο κελί X για να δούμε σε ποιόν περιορισμό αναφέρεται και επομένως σε ποια γραμμή του πίνακα A θα γίνει η καταχώρηση. Αφού βρεθούν η γραμμή και η στήλη καταχωρείται στην συγκεκριμένη θέση η τιμή που έχει διαβαστεί. Όταν όμως το όνομα του περιορισμού αντιστοιχεί στην αντικειμενική συνάρτηση τότε η καταχώρηση γίνεται στην αντίστοιχη στήλη του διανύσματος c τώρα και όχι στον A, Εδώ δεν χρειαζόμαστε αριθμό γραμμής γιατί το c είναι διάνυσμα.

Όταν κάτι τέτοιο συμβεί μια μεταβλητή σημαία κρατά το γεγονός ότι για την τρέχουσα μεταβλητή βρέθηκε αναφορά της στην αντικειμενική συνάρτηση, για να μην υπάρξει πρόβλημα όταν η τρέχουσα μεταβλητή αλλάξει. Αυτό γιατί αν η τρέχουσα μεταβλητή αλλάξει στην αντίστοιχη στήλη του c θα πρέπει υποχρεωτικά να γίνει καταχώρηση, είτε συμμετέχει η μεταβλητή αυτή στην αντικειμενική συνάρτηση, είτε όχι και τότε καταχωρείται η τιμή της ίση με μηδέν. Αν πάλι αλλάξει η τρέχουσα μεταβλητή και μετά επανέλθουμε σε αυτήν θα πρέπει να γνωρίζουμε αν προηγουμένως είχε βρεθεί η τιμή του συντελεστή της στην αντικειμενική συνάρτηση.

Τεχνικές που χρησιμοποιήθηκαν στον σχηματισμό της μήτρας A ήταν το preallocation για το μέγεθος που θα έχει η μήτρα A, ενδεικτικά με το μέγεθος του αρχείου του mps, για να εξασφαλίσουμε ελεύθερη μνήμη, καθώς οι μήτρες στο MATLAB όταν είναι δυναμικές μεταχειρίζονται πολύ αργά, και καταλαμβάνοντας προληπτικά μεγάλα ποσά μνήμης. Επίσης κατά την αναζήτηση του ονόματος του περιορισμού στον οποίον γίνεται η αναφορά για την τιμή του συντελεστή της αντίστοιχης μεταβλητής υλοποιήθηκε μια αναζήτηση τεμαχισμού του κελιού X, υπό την έννοια ότι αν ένα όνομα περιορισμού βρέθηκε για την τρέχουσα μεταβλητή, πιο πιθανόν είναι το όνομα του περιορισμού για την ίδια μεταβλητή στην επόμενη αναφορά, να βρίσκεται από την προηγούμενη θέση και κάτω. Αν πάλι δεν βρεθεί εκεί η αναζήτηση παίρνει την μορφή της χειρότερης περίπτωσης, δηλαδή αναζητά στο υπόλοιπο του κελιού X.

Τέλος η συνάρτηση διαβάζει με παρόμοιο τρόπο τους σταθερούς όρους από το πεδίο RHS και αποθηκεύει τις τιμές στον πίνακα στήλη b. Η συνάρτηση επίσης έχει την δυνατότητα να διαβάζει BOUNDS και RANGES, όταν αυτά υπάρχουν. Από την άλλη η συνάρτηση mat2mps μετατρέπει ένα γ.π από μορφή μητρών σε μορφή mps για χρήση του προβλήματος σε άλλα πακέτα γραμμικού προγραμματισμού. Η συνάρτηση δέχεται για είσοδο το όνομα του .mat αρχείου και τον τύπο του προβλήματος (1 για max,-1 για min), και αποθηκεύει στον ίδιο κατάλογο, όπως και mps2mat, το ανάλογο .mps αρχείο.

ΚΕΦΑΛΑΙΟ 7

Συγκριτική Υπολογιστική Μελέτη

Ένα αποτελεσματικό εργαλείο για την σύγκριση της πρακτικής αποτελεσματικότητας δύο ή περισσοτέρων αλγορίθμων είναι οι υπολογιστικές μελέτες. Αυτές οι μελέτες παρέχουν ένα πολύ καλό κριτήριο όχι μόνο για την πολυπλοκότητα των αλγορίθμων αλλά και για την αξιοπιστία τους.

Σε αυτό το κεφάλαιο παρουσιάζουμε μια συγκριτική υπολογιστική μελέτη μεταξύ του πρωτεύοντος αλγορίθμου Simplex, του PDTPSA, σε αναθεωρημένες μορφές, καθώς και του αλγορίθμου εσωτερικών σημείων που είναι ήδη υλοποιημένος στο MATLAB σε τυχαία γραμμικά προβλήματα διαστάσεων από 750x750 έως και 2000x2000 με πυκνότητες 2.5%, 5%, 10% και 20%. Επίσης μεταξύ των δύο πρώτων αλγορίθμων πραγματοποιήθηκε και υπολογιστική μελέτη στα γνωστά βέλτιστα μετροπροβλήματα του Γραμμικού Προγραμματισμού.

Το σύστημα που χρησιμοποιήσαμε είχε Intel Core 2 Duo 1.73 Ghz cpu, και 1024 Mb DDR2 RAM. Χρησιμοποιήσαμε τα MS-Windows Vista Business 64-bit edition. Τέλος το υπολογιστικό μας περιβάλλον ήταν το MATLAB R2007a x64. Κατα την γνώμη μας το MATLAB είναι ένα ισχυρό εργαλείο για την ανάπτυξη τέτοιου είδους αλγορίθμων.

7.1 Γραφικές Αναπαραστάσεις των μετήσεων

Παρακάτω παραθέτουμε τα αντίστοιχα γραφήματα και πίνακες των λόγων ως προς το σύνολο των επαναλήψεων και ως προς τον συνολικά απαιτούμενο χρόνο επίλυσης.

density 2.5%			density 5%		
sizes	cpu time	niter	sizes	cpu time	niter
750*750	4,39	3,66	750*750	5,88	4,22
1000*1000	5,2	4,36	1000*1000	7,36	5,4
1250*1250	6,59	5,14	1250*1250	6,59	5,14
1500*1500	8,4	6,12	1500*1500	11,54	7,79
1750*1750	10	6,98	1750*1750	13,25	9,15
2000*2000	12,01	7,96	2000*2000	14,45	9,74

Πίνακας 7.1.1. Λόγοι Revised PSA / PDTPSA για τυχαία αραιά γραμμικά προβλήματα πυκνότητας 2,5%,5%.

density 10%			density 20%		
sizes	cpu time	niter	sizes	cpu time	niter
750*750	7,15	5,39	750*750	6,5	5,88
1000*1000	7,9	6,56	1000*1000	7,25	9,72
1250*1250	10,8	8,6	1250*1250	8,16	7,86
1500*1500	11,25	8,89	1500*1500	10,01	9,45
1750*1750	13,23	10,01	1750*1750	12	10,8
2000*2000	14,7	10,97	2000*2000	14,9	13,5

Πίνακας 7.1.2. Λόγοι Revised PSA / PDTPSA για τυχαία αραιά γραμμικά προβλήματα πυκνότητας 10%,20%.

	density 2.5%		density 5%	
sizes	cpu time IPM	cpu time PDTPSA	cpu time IPM	cpu time PDTPSA
750*750	16,8	28,2	28,2	23
1000*1000	50	69	55,49	60
1250*1250	94	148	120,6	148
1500*1500	213	243	206,87	200
1750*1750	284	411	745	338
2000*2000	674	615	798	510

Πίνακας 7.1.3. Πραγματικοί χρόνοι των IPM / PDTPSA για τυχαία αραιά γραμμικά προβλήματα πυκνότητας 2.5%,5%.

	density 10%		density 20%	
sizes	cpu time IPM	cpu time PDTPSA	cpu time IPM	cpu time PDTPSA
750*750	73,6	19	467,8	18
1000*1000	130	46,8	1432,43	54
1250*1250	2511	99	166	99
1500*1500	963	179	228,44	165,7
1750*1750	1378,21	294	307	271
2000*2000	2227	437	320	345

Πίνακας 7.1.4. Πραγματικοί χρόνοι των IPM / PDTPSA για τυχαία αραιά γραμμικά προβλήματα πυκνότητας 10%,20%.

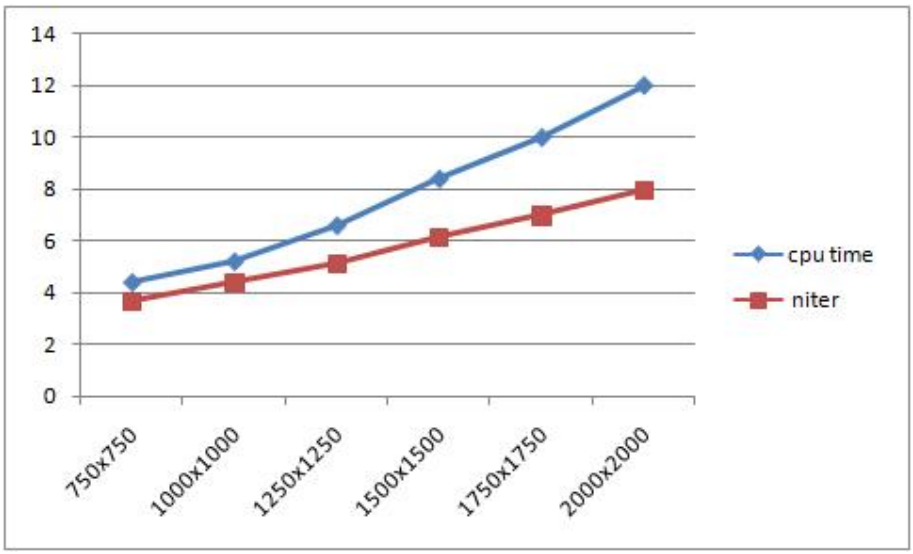
IPM/PDTPSA		densities		
sizes	2,50%	5%	10%	20%
750*750	0,59	1,23	3,87	25,98
1000*1000	0,72	0,92	2,77	26,52
1250*1250	0,63	0,81	25,36	1,67
1500*1500	0,87	1,03	5,37	1,37
1750*1750	0,69	2,2	4,68	1,13
2000*2000	1,09	1,56	5,09	0,92

Πίνακας 7.1.5. Λόγοι IPM / PDTPSA για τυχαία αραιά γραμμικά προβλήματα πυκνότητας 2,5%,5%,10%,20%.

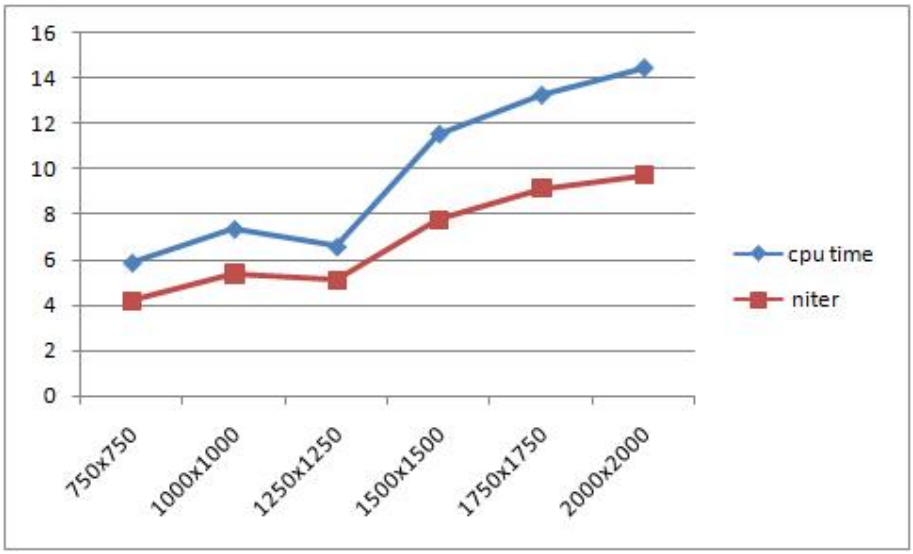
Στον πίνακα μετρήσεων μεταξύ rPSA και PDTPSA για τα μετροπροβλήματα η στήλη config αναφέρεται στις τιμές που παίρνουν τα ορίσματα για την κλιμάκωση, το re-inversion καθώς και την τιμή των ανοχών (n σημαίνει πως χρησιμοποιείται η κατεπιλογήν τιμή 1.0E-008), κατα τον χρόνο εκτέλεσης των αλγορίθμων.

Η αναπαραγωγή των τυχαίων γραμμικών προβλημάτων έγινε βάση γεννήτριας με αριθμό seed number = 3. Όλα τα προβλήματα που κατασκευάστηκαν ήταν βέλτιστα. Τα διαστήματα τιμών που δώσαμε για τις αντίστοιχες μήτρες και διανύσματα ήταν τα ακόλουθα :

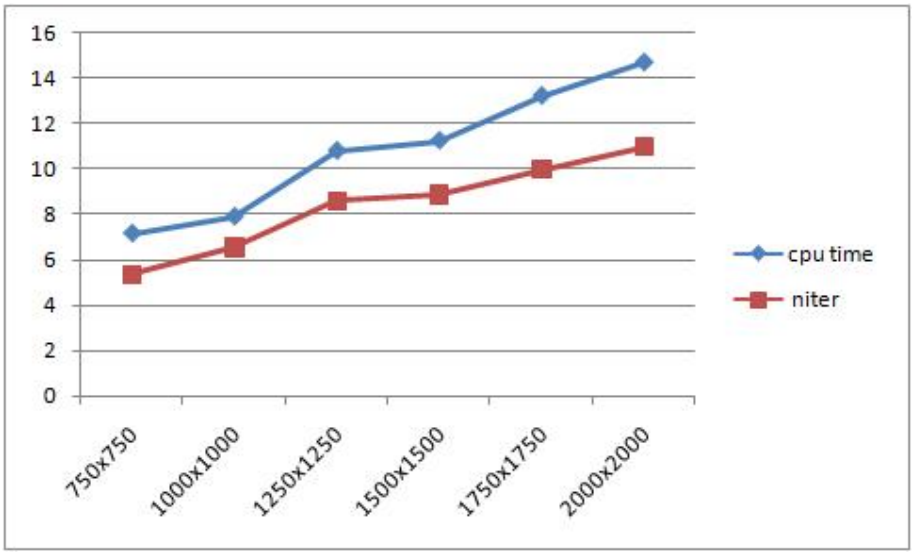
$$A = [100\ 10.000] \quad c = [-1.000\ 2.500] \quad b = [50\ 5.000] \quad Eqin = [-1|1]$$



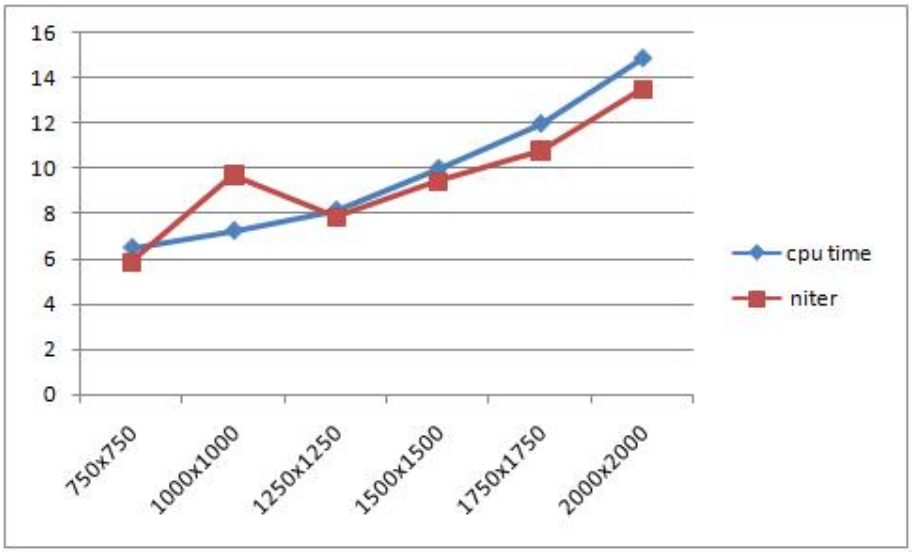
Σχήμα **7.1.1**: Τυχαία Γραμμικά προβλήματα - Πυκνότητα 2.5% - Λόγοι rPSA/PDTPSA ως προς χρόνο επίλυσης cputime και αριθμό επαναλήψεων niter



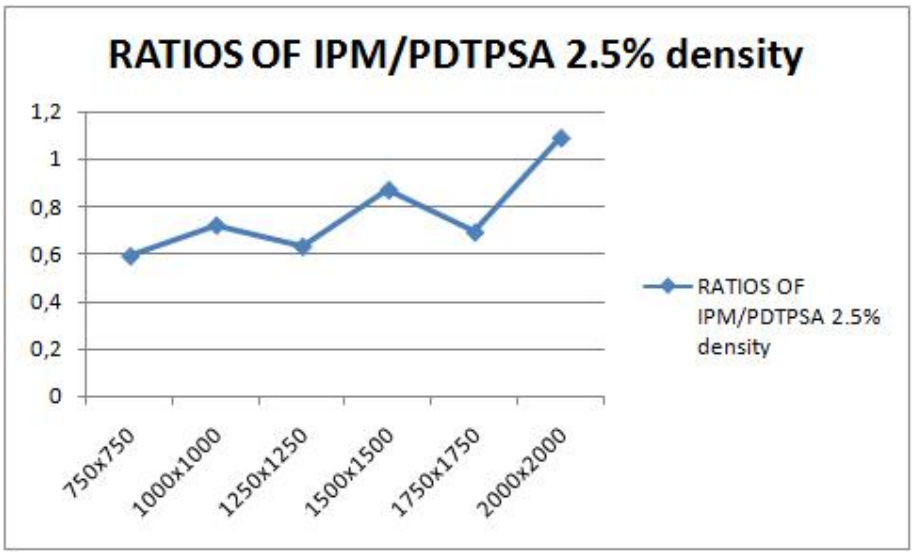
Σχήμα **7.1.2**: Τυχαία Γραμμικά προβλήματα - Πυκνότητα 5% - Λόγοι rPSA/PDTPSA ως προς χρόνο επίλυσης cputime και αριθμό επαναλήψεων niter



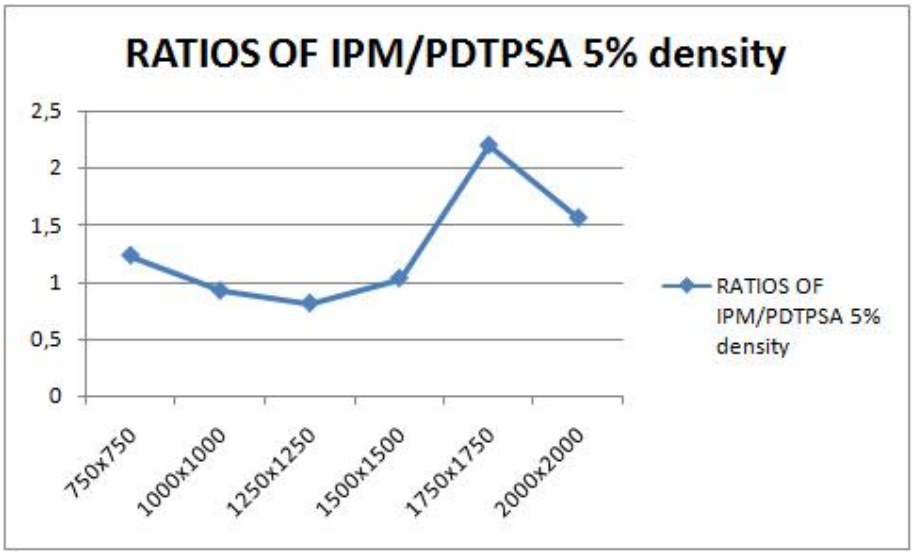
Σχήμα **7.1.3**: Τυχαία Γραμμικά προβλήματα - Πυκνότητα 10% - Λόγοι rPSA/PDTPSA ως προς χρόνο επίλυσης cputime και αριθμό επαναλήψεων niter



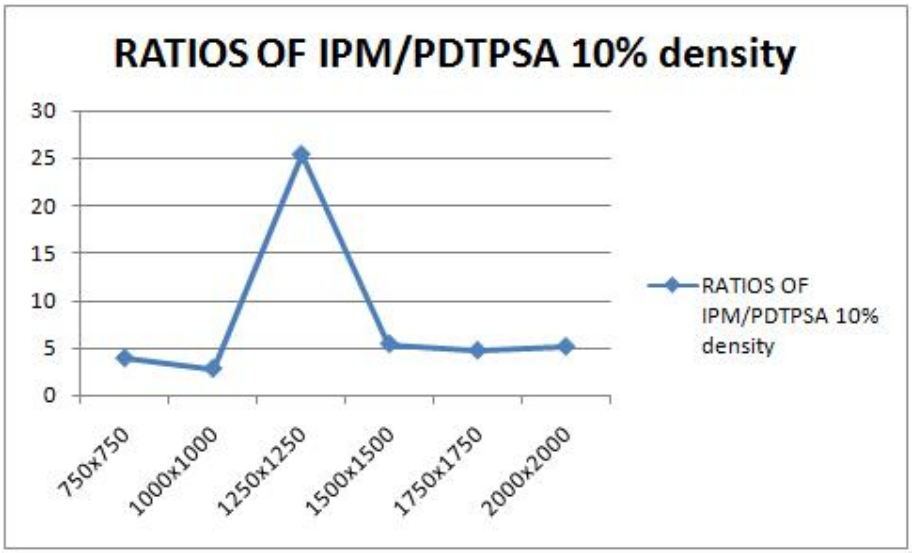
Σχήμα **7.1.4**: Τυχαία Γραμμικά προβλήματα - Πυκνότητα 20% - Λόγοι rPSA/PDTPSA ως προς χρόνο επίλυσης cputime και αριθμό επαναλήψεων niter



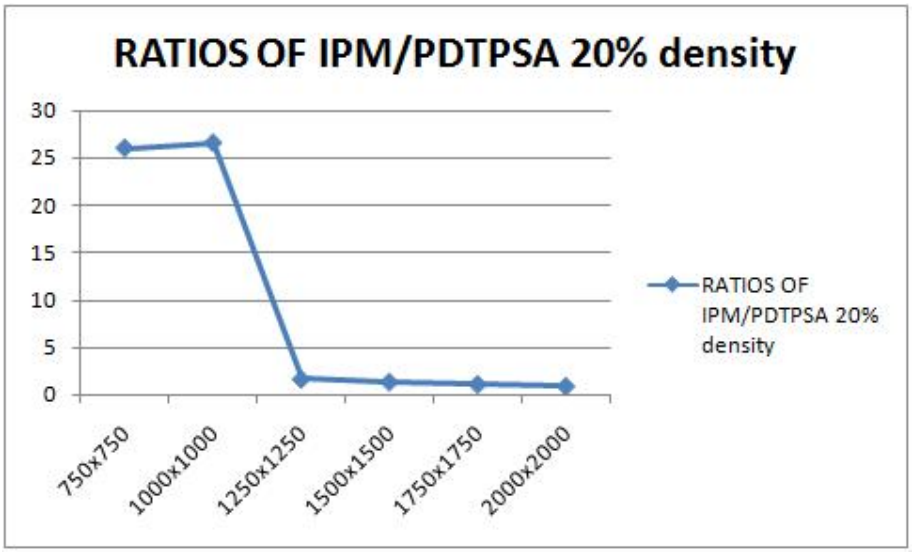
Σχήμα **7.1.5**: Τυχαία Γραμμικά προβλήματα - Πυκνότητα 2.5% - Λόγοι IPM/PDTPSA ως προς χρόνο επίλυσης cputime niter



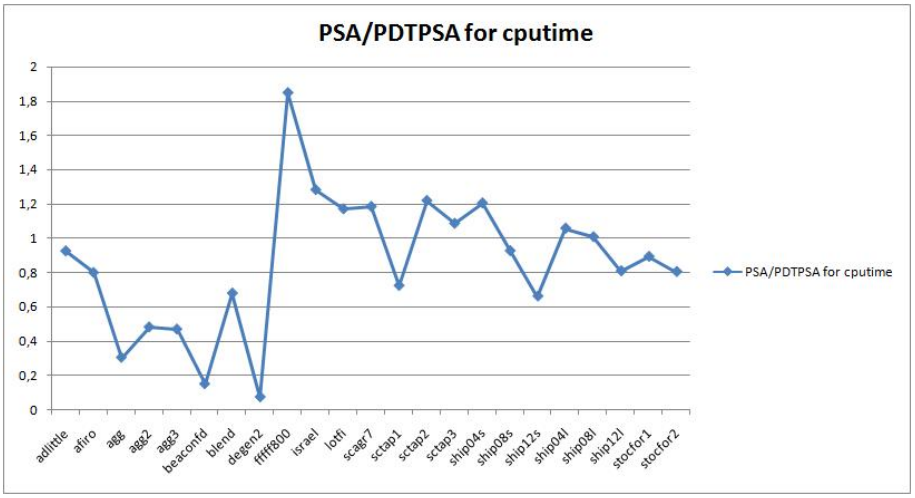
Σχήμα **7.1.6**: Τυχαία Γραμμικά προβλήματα - Πυκνότητα 5% - Λόγοι IPM/PDTPSA ως προς χρόνο επίλυσης cputime niter



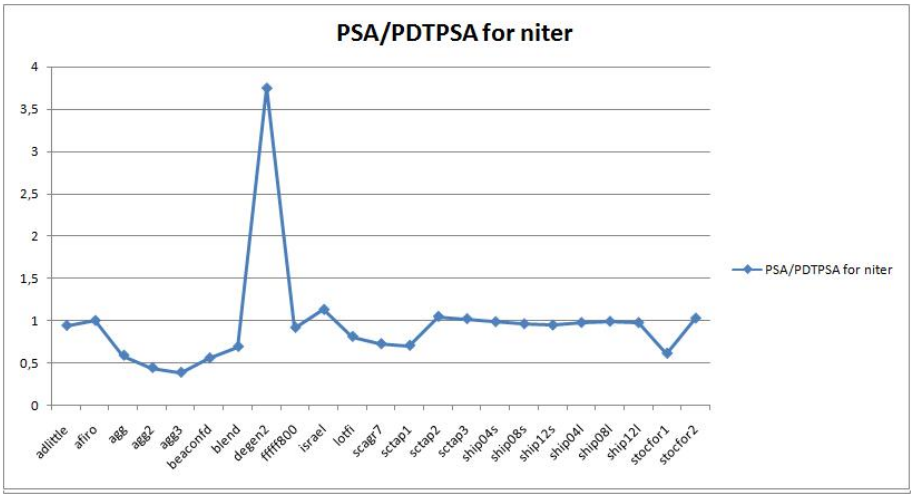
Σχήμα **7.1.7**: Τυχαία Γραμμικά προβλήματα - Πυκνότητα 10% - Λόγοι IPM/PDTPSA ως προς χρόνο επίλυσης cputime niter



Σχήμα **7.1.8**: Τυχαία Γραμμικά προβλήματα - Πυκνότητα 20% - Λόγοι IPM/PDTPSA ως προς χρόνο επίλυσης cputime niter



Σχήμα **7.1.9** Μετροπροβλήματα - Λόγοι rPSA/PDTPSA ως προς χρόνο επίλυσης



Σχήμα **7.1.10** Μετροπροβλήματα - Λόγοι rPSA/PDTPSA ως προς σύνολο επαναλήψεων

	Revised PSA			PDTPSA		
Benchmark	cputime	niter	config	cputime	niter	config
adlittle	0,16	100	2 80 n	0,17	108	2 80 n
afiro	0,02	12	2 80 n	0,02	15	2 80 n
agg	0,4	72	2 80 n	0,68	236	2 80 n
agg2	0,49	139	2 80 n	1,11	288	2 80 n
agg3	0,59	158	2 80 n	1,53	336	2 60 n
beaconfd	0,1	22	2 80 n	0,18	146	2 80 n
blend	0,09	70	2 80 n	0,13	103	2 80 n
degen2	81	6050	1 60 n	21,58	1082	2 80 n
fffff800	2,22	429	2 80 n	2,42	232	1 80 n
israel	0,52	358	2 80 n	0,46	279	2 80 n
lotfi	0,3	205	2 80 n	0,37	175	2 80 n
scagr7	0,21	96	2 80 n	0,29	81	2 80 n
sctap1	0,67	247	2 80 n	0,95	341	2 80 n
sctap2	4,39	593	2 80 n	4,19	486	2 80 n
sctap3	11,24	801	2 80 n	11,01	737	2 80 n
ship04s	2,04	199	2 80 n	2,07	165	2 80 n
ship08s	3,62	243	2 80 n	3,77	262	2 80 n
ship12s	6,78	381	2 80 n	7,17	575	2 80 n
ship04l	4,88	235	2 80 n	5	222	2 80 n
ship08l	20,78	466	2 80 n	21	462	2 80 n
ship12l	48,98	737	2 80 n	50,28	910	2 80 n
stocfor1	0,11	69	2 80 n	0,18	66	2 80 n
stocfor2	156,19	1036	2 80 n	152	1286	2 100 n

Πίνακας μετρήσεων μετροπροβλημάτων για τους αλγόριθμους Revised PSA και PDTPSA.

ΚΕΦΑΛΑΙΟ 8

Συμπεράσματα

Η εργασία αυτή παρουσίασε μια συγκριτική υπολογιστική μελέτη μεταξύ ενός πρωτεύοντος δυικού αλγορίθμου PDTPSA και των αλγορίθμων revised Primal Simplex και LIPSOL που αποτελεί την υλοποίηση του αλγορίθμου εσωτερικών σημείων του πακέτου MATLAB σε ένα πλήθος τυχαίων αραιών γραμμικών προβλημάτων. Επίσης οι δύο πρώτοι αλγόριθμοι συγκρίθηκαν και σε ένα πλήθος βέλτιστων μετροπροβλημάτων από την συλλογή netlib. Όλες μας οι υπολογιστικές μετρήσεις καθώς και η προγραμματιστική ανάπτυξη των αλγορίθμων, πραγματοποιήθηκαν στο υπολογιστικό περιβάλλον του MATLAB.

Όπως δείχνουν τα αποτελέσματα των μετρήσεων ο αλγόριθμος PDTPSA υπερέχει ξεκάθαρα έναντι του αλγορίθμου revised Primal Simplex ανεξαρτήτως διαστάσεων και πυκνότητας των τυχαίων γραμμικών προβλημάτων. Θα πρέπει επίσης εδώ να σημειώσουμε ότι οι διαφορές αυτές θα ήταν πολύ μεγαλύτερες αν δεν χρησιμοποιούσαμε τεχνικές κλιμάκωσης για τον αλγόριθμο Simplex την στιγμή που ο αλγόριθμος PDTPSA παρουσίαζε την ίδια συμπεριφορά και χωρίς την χρήση αυτών. Στα μετροπροβλήματα οι δύο αλγόριθμοι φαίνεται να παρουσιάζουν παρόμοια συμπεριφορά, όμως ο αλγόριθμος PDTPSA φαίνεται να δείχνει την αποτελεσματικότητά του όσο το μέγεθος των προβλημάτων αυξάνει. Εξάλλου η επίλυση και μόνο αυτών των προβλημάτων επιδεικνύει και την σταθερότητα του αλγορίθμου PDTPSA.

Σε σύγκριση με τον αλγόριθμο εσωτερικών σημείων τώρα θα πρέπει να σημειώσουμε πως οι μετρήσεις μας δεν περιελάμβαναν αριθμό επαναλήψεων καθώς καθίσταται σαφές ότι κάτι τέτοιο δεν μπορεί να αποτελέσει κριτήριο σύγκρισης γιατί οι αλγόριθμοι εσωτερικών σημείων είναι γνωστό πως καλύπτουν ελάχιστες πάντα επαναλήψεις αλλά με τεράστιο υπολογιστικό κόστος. Έτσι οι μετρήσεις μας περιορίστηκαν στο συνολικά απαιτούμενο χρόνο επίλυσης που είναι πάντα και το πιο ασφαλές κριτήριο. Έτσι ενώ στις κατηγορίες πυκνοτήτων 2.5% και 5%

ο αλγόριθμος εσωτερικών σημείων δείχνει να υπερισχύει, στις υπόλοιπες δύο κατηγορίες των 10% και 20% αντίστοιχα ο αλγόριθμος αυτός αποτυγχάνει να επιλύσει όλα τα προβλήματα όλων των διαστάσεων αποφαινόμενος ότι το πρόβλημα είναι είτε αδύνατο είτε απεριόριστο. Επίσης από άποψη χρόνου επίλυσης υστερεί αρκετά σε σχέση με τον αλγόριθμο PDTPSA σε αυτές τις κατηγορίες προβλημάτων, αν και αυτό δεν δείχνει τόσο σημαντικό όσο η αδυναμία επίλυσής τους.

Αναφορές

[1] Benjamin Van Roy and Kahn Mason: *Introduction to optimization: Formulation and Analysis of Linear Programs*, Stanford University, Course Notes (2006)

[2] Dimitris Bertsimas and John N. Tsitsiklis: *Introduction to Linear Optimization*, Athena Scientific, (1997)

[3] Samaras, N.: *Computational improvements and efficient implementation of two path pivoting algorithms*, Ph.D Thesis, Dept. of Applied Informatics, University of Macedonia, (2001)

[4] Stephanides, G., Samaras, N.: *Computational methods with MATLAB*, ZYGOS Publications, Thessaloniki. (In Greek), (1999)

[5] Meszaros Cs.: *The Efficient Implementation of Interior Point Methods for Linear Programming and their Applications*, PhD Thesis, School of Operations Research, Applied Mathematics and Statistics, Eotvos Lorand University of Sciences, Budapest, (1996)

[6] Paparrizos, K.: *Linear programming, algorithms and applications*, ZYGOS Publications, Thessaloniki. (In Greek), (1999)

[7] Paparrizos, K.: *An exterior point simplex algorithm for (general) linear programming problems*, Annals of Operations Research, Vol. 47, pp. 497-508, (1993)

[8] Paparrizos, K.: *An infeasible (Exterior Point) simplex algorithm for assignment problems*, Mathematical Programming, Vol. 51, pp. 45-54, (1991)

[9] Paparrizos, K.: *A fine improvement algorithm for the linear complementarity problem*, Ph.D Thesis, Department of Operational Research, CWRU, Cleveland, OHIO, (1983)

[10] Paparrizos K, Samaras N, Stephanides G.: *An efficient simplex type algorithm for sparse and dense linear programs*, European Journal of Operational Research,Vol. 148(2), pp. 323-334, (2003)

[11] Paparrizos K, Samaras N, Stephanides G.: *A new efficient primal dual simplex algorithm*, Computers and Operations Research, Vol. 30, pp.1383-1399, (2003)

[12] Paparrizos, K.: *An infeasible exterior point simplex a9lgorithm for assignment problems*, Mathematical Programming, Vol 51, pp. 45-54 (1991)

[13] Paparrizos, K.: *A new primal and dual pivoting rule for the simplex algorithm*, Proceedings of SYMOPIS '96,pp. 445-53 (1996)

[14] Paparrizos, K.: *Pivoting algorithms generating two paths*, Presented in ISMP '97, Lausanne, EPFL Switzerland, (1997)

[15] Paparrizos, K.: *Linear Programming*, ZYGOS Publications, Thessaloniki. (In Greek), (2007)

[16] Robert J. Vanderbei: *Linear Programming:Foundations and Extensions*, Second Edition (2001)

[17] Bland,G.R.: *New finite pivoting rules for the simplex method", Mathematics of Operations Research*, Mathematics of Operations Research, Vol. 2, pp. 103-107, (1977)

[18] Charnes,A.: *Optimality and degeneracy in linear programming*, Econometrica, Vol. 20(2), pp. 160-170, (1952)

[19] MathWorks.: *Pro-MATLAB: User's guide*, MathWorks, Inc., South Natick, (1990)

[20] Jacek, Gondzio: *Warm Start of the Primal-Dual Method Applied in the Cutting-Plane Scheme*, Logilab Technical Report 96.3, (1997)

[21] Erling D. Andersen and Yinyu Ye: *Combining Interior-Point and Pivoting Algorithms for Linear Programming*, Management Science, Vol. 42, No. 12, pp. 1719-1731, (1996)

[22] Elizabeth John, E. Alper Yildirim: *Implementation of Warm-Start Strategies in Interior Point Methods for Linear Programming in Fixed Dimension*, Optimization Online, (2006)

[23] Triantafyllidis, Ch.: *Comparative computational study on Exterior Point Algorithms*, B.S Thesis, Department of Applied Informatics, University of Macedonia, (2005)

[24] Borgwardt HK: *Some distribution-independent results about the assymptotic order of the average number of pivot steps of the simplex algorithm*, Mathematics of Operations Research, Vol 26, No. 5, pp. (1982)

[25] Borgwardt HK: *The average number of pivot steps required by the simplex method is polynomial*, Zeitschrift fur Operational Research, Series A: Theory , Vol 26, No 5, pp. 157-77 (1982)

[26] Klee V, Minty GJ: *How good is the simplex algorithm?*, Shisha O, editor. Inequalities III, New York Academic Press, pp. 158-72

[27] Gondzio J.: *Multiple centrality corrections in a primal-dual method for linear programming*, Computational Optimization and Applications, Vol 6, pp. 137-56 (1996)

[28] Gondzio J.: *Splitting dense columns of constraint matrix in interior point methods for large-scale linear programming*, Optimization, Vol 24, pp. 285-97 (1992)

[29] Karmarkar NK.: *A polynomial time algorithm for linear programming*, Combinatorica, Vol 4, pp. 373-95 (1984)

[30] Lustig JI, Marstenand RE, Shanno DF: *On implementing Mehrotra's predictor-corrector interior point method for linear programming*, SIAM Journal on Optimization, Vol 2, No. 3, pp. 435-49 (1992)

[31] Sifaleras A.: *Development and Implementation of Exterior Point Simplex Type Algorithms for Network Optimization Problems, (in Greek)*, Ph.D Thesis, Department of Applied Informatics, University of Macedonia, Greece, (2006)