ΑΝΑΠΤΥΞΗ ΚΑΙ ΥΛΟΠΟΙΗΣΗ ΑΛΓΟΡΙΘΜΩΝ ΕΞΩΤΕΡΙΚΩΝ ΣΗΜΕΙΩΝ ΤΥΠΟΥ SIMPLEX ΓΙΑ ΠΡΟΒΛΗΜΑΤΑ ΔΙΚΤΥΑΚΗΣ ΒΕΛΤΙΣΤΟΠΟΙΗΣΗΣ

ΑΓΓΕΛΟΣ ΣΙΦΑΛΕΡΑΣ

ΔΙΔΑΚΤΟΡΙΚΗ ΔΙΑΤΡΙΒΗ

Επιβλέπων Καθηγητής Κ. Παπαρρίζος

Μέλη Τριμελούς Επιτροπής Κ. Μαργαρίτης

Ι. Βλαχάβας

Τμήμα Εφαρμοσμένης Πληροφορικής

Πανεπιστήμιο Μακεδονίας

Θεσσαλονίκη

Δεκέμβριος 2006

ΥΠΟΥΡΓΕΙΟ ΕΘΝΙΚΗΣ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ ΕΠΕΑΕΚ
ΕΥΡΩΠΑΪΚΗ ΕΝΩΣΗ
ΣΥΓΧΡΗΜΑΤΟΔΟΤΗΣΗ
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
ΕΥΡΩΠΑΪΚΟ ΤΑΜΕΙΟ ΠΕΡΙΦΕΡΕΙΑΚΗΣ ΑΝΑΠΤΥΞΗΣ
Η ΠΑΙΔΕΙΑ ΣΤΗΝ ΚΟΡΥΦΗ
Επιχειρησιακό Πρόγραμμα
Εκπαίδευσης και Αρχικής
Επαγγελματικής Κατάρτισης

Το έργο «ΗΡΑΚΛΕΙΤΟΣ: Υποτροφίες Έρευνας στο Πανεπιστήμιο Μακεδονίας» - Υποέργο «**Ανάπτυξη και Υλοποίηση Αλγορίθμων Εξωτερικών Σημείων Τύπου Simplex για Προβλήματα Δικτυακής Βελτιστοποίησης**» υλοποιείται στα πλαίσια της Κατηγορίας Πράξεων 2.2.3.β. «Υποτροφίες Έρευνας με προτεραιότητα στη Βασική Έρευνα», Μέτρο 2.2 «Αναμόρφωση Προγραμμάτων Σπουδών - Διεύρυνση Τριτοβάθμιας Εκπαίδευσης», Ενέργεια 2.2.3 «Προγράμματα Μεταπτυχιακών Σπουδών - Έρευνα - Υποτροφίες», εκτελείται στα πλαίσια του Επιχειρησιακού Προγράμματος Εκπαίδευσης και Αρχικής Επαγγελματικής Κατάρτισης ΙΙ (ΕΠΕΑΕΚ ΙΙ) και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση [3ο Κοινοτικό Πλαίσιο Στήριξης κατά 75% Κοινοτική Συμμετοχή (ΕΚΤ) και 25% Εθνικοί Πόροι]

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

ΕΥΧΑΡΙΣΤΙΕΣ

Η προσπάθεια που κατέβαλα τα τελευταία χρόνια ώστε να πραγματοποιήσω βασική έρευνα και να υποστηρίξω τη παρούσα Διδακτορική Διατριβή, ενισχύθηκε οικονομικά από το ερευνητικό πρόγραμμα *ΗΡΑΚΛΕΙΤΟΣ: Υποτροφίες Έρευνας στο Πανεπιστήμιο Μακεδονίας* - Υποέργο *Ανάπτυξη και Υλοποίηση Αλγορίθμων Εξωτερικών Σημείων Τύπου Simplex για Προβλήματα Δικτυακής Βελτιστοποίησης* υλοποιείται στα πλαίσια της Κατηγορίας Πράξεων 2.2.3.β. *Υποτροφίες Έρευνας με προτεραιότητα στη Βασική Έρευνα*, Μέτρο 2.2 *Αναμόρφωση Προγραμμάτων Σπουδών - Διεύρυνση Τριτοβάθμιας Εκπαίδευσης*, Ενέργεια 2.2.3 *Προγράμματα Μεταπτυχιακών Σπουδών - Έρευνα - Υποτροφίες*, το οποίο εκτελείται στα πλαίσια του Επιχειρησιακού Προγράμματος Εκπαίδευσης και Αρχικής Επαγγελματικής Κατάρτισης ΙΙ (ΕΠΕΑΕΚ ΙΙ) και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση [3ο Κοινοτικό Πλαίσιο Στήριξης κατά 75% Κοινοτική Συμμετοχή (ΕΚΤ) και 25% Εθνικοί Πόροι]. Ως εκ τούτου οφείλω να εκφράσω τις αμέριστες ευχαριστίες μου σε όλους, όσους βοήθησαν στη λήψη αυτής της χρηματοδότησης.

Έπειτα, θα πρέπει να ευχαριστήσω θερμά τον Επιβλέποντα Καθηγητή μου κ. Παπαρρίζο Κωνσταντίνο για την επιστημονική καθοδήγηση που μου παρείχε όλα αυτά τα χρόνια των διδακτορικών μου σπουδών. Οι συμβουλές του, σε πληθώρα ακαδημαικών και ερευνητικών θεμάτων, πραγματικά ήταν ακριβέστατες και θα αποτελέσουν οδηγό για εμένα τα επόμενα χρόνια.

Θα πρέπει παράλληλα να ευχαριστήσω τον Καθηγητή κ. Μαργαρίτη Κωνσταντίνο και τον Καθηγητή κ. Βλαχάβα Ιωάννη για τις εύστοχες παρατηρήσεις τους καθώς και για τη καλή συνεργασία που αναπτύξαμε τα τελευταία χρόνια.

Ιδιαίτερη μνεία πρέπει να κάνω για όλους τους συνεργάτες μου στην ερευνητική ομάδα, με τους οποίους δουλέψαμε μαζί τόσο σε διάφορα ερευνητικά προγράμματα, όσο και σε ερευνητικές δημοσιεύσεις, αλλά και γενικότερα αναπτύξαμε δεσμούς φιλίας ανάμεσά μας. Επομένως, θέλω να απευθύνω ένα ευχαριστώ στο Λέκτορα κ. Σαμαρά Νικόλαο, τους Διδάκτορες κ. Δόσιο Κωνσταν-

τίνο, κ. Κώστογλου Κωνσταντίνο, κ. Λαζαρίδη Βασίλειο και κ. El-Said Badr καθώς και στους Υποψήφιους Διδάκτορες κ. Γεράνη Γεώργιο, κ. Καραγιάννη Παναγιώτη, κ. Μπαλούκα Αθανάσιο και κα. Ρώσσιου Ελένη.

Τέλος, οφείλω ευγνωμοσύνη στην οικογένειά μου για την αμέριστη και ποικιλότροπη στήριξη που μου παρείχε καθ' όλη τη διάρκεια της εκπόνησης της Διδακτορικής Διατριβής. Επιπλέον δε, πρέπει να ευχαριστήσω τη Φανή για τη συμπαράσταση και την κατανόηση που έδειξε, σε αυτή την κοπιαστική και απαιτητική για εμένα χρονική περίοδο.

Αφιερώνεται στην οικογένεια μου,

στην Τασία, στη Δέσποινα, στο Χρήστο, στην Έλενα, στην Κωνσταντίνα και

στη Φανή.

ΠΕΡΙΛΗΨΗ

Η παρούσα διατριβή αποτελείται απο τρεις άξονες. Πρώτον και κυριότερο παρουσιάζει αναλυτικά για πρώτη φορά έναν νέο πρωτεύοντα αλγόριθμο εξωτερικών σημείων για την επίλυση Προβλημάτων Ροής Ελαχίστου Κόστους. Πιό συγκεκριμένα, παρουσιάζεται ο ψευδοκώδικας, γραφικά παραδείγματα για βέλτιστα, ή και απεριόριστα, προβλήματα καθώς και οι μαθηματικές αποδείξεις ορθότητας. Δεύτερον, περιγράφει μια υλοποίηση σε γλώσσα Fortran 95/2003 και δείχνει μια ενδεικτική υπολογιστική συμπεριφορά του προτεινόμενου αλγορίθμου. Αναφέρονται εμπεριστατωμένα όλες οι δομές που χρησιμοποιήθηκαν και φυσικά το υπολογιστικό περιβάλλον υλοποίησης. Τέλος, παρουσιάζεται και ένα ολοκληρωμένο πακέτο επίλυσης προβλημάτων Δικτυακής Βελτιστοποίησης, κυρίως για εκπαιδευτικούς σκοπούς.

ABSTRACT

This PhD Thesis consists mainly of three parts. The first and primal part presents for the first time a new exterior point Simplex algorithm for the solution of the Minimum Cost Network Flow Problem. More precisely, we present the pseudocde, illustrative examples of an optimal or unbounded problem, as also the mathematical proofs of correctness. The second part describes an implementation in the Fortran 95/2003 programming language and also shows an indicative computational behaviour of the proposed algorithm. All the necessary data structures, which have been used, as also the software and hardware infastracture are analytically discussed. Finally, in the third part a new software for the solution of problems belonging to various categories of Network Optimization, mainly for educational purposes, is shown.

ΠΕΡΙΕΧΟΜΕΝΑ

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

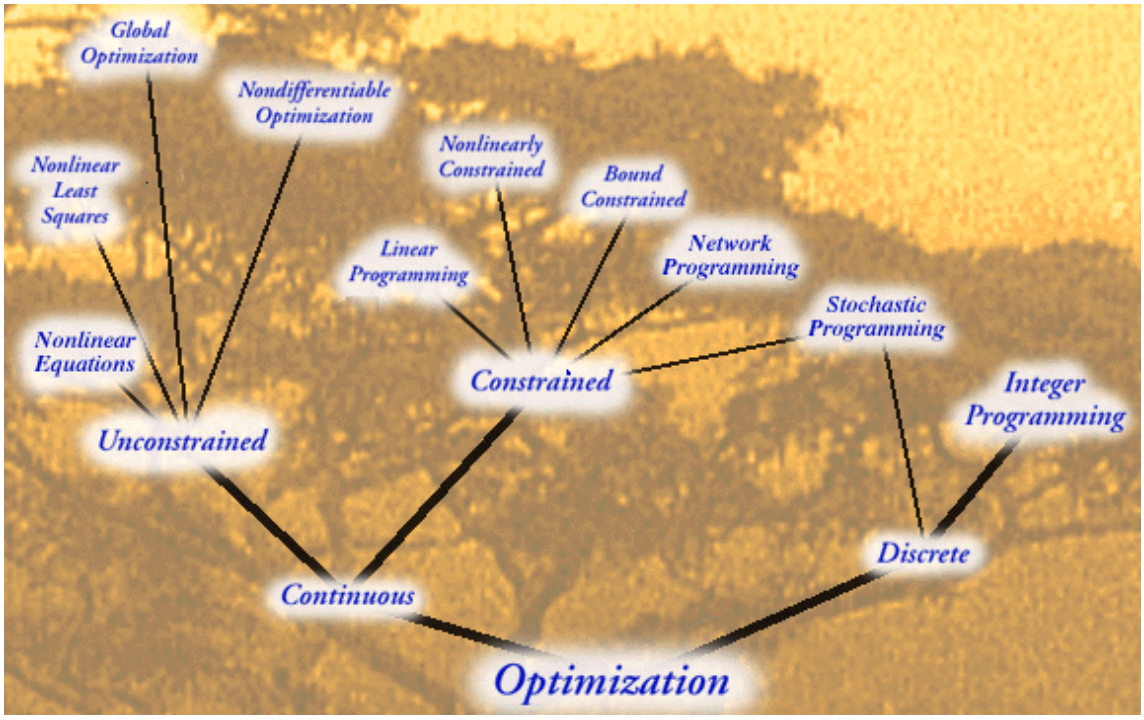
ΚΕΦΑΛΑΙΟ 1

Εισαγωγή

1.1 Ερευνητικό πεδίο της Διατριβής

Ο Μαθηματικός Προγραμματισμός είναι ένα πολύ μεγάλο ερευνητικό πεδίο, στο οποίο κύριο στόχο αποτελεί η ανάπτυξη αποτελεσματικών υπολογιστικών τεχνικών. Ένα ιδιαίτερο υποσύνολο του Μαθηματικού Προγραμματισμού είναι ο Γραμμικός Προγραμματισμός, ο οποίος αναφέρεται συνήθως και ως Γραμμική Βελτιστοποίηση, στον οποίο ασχολούμαστε αποκλειστικά με γραμμικά προβλήματα. Ακολουθώντας την προηγούμενη κατηγοριοποίηση, η Γραμμική Δικτυακή Βελτιστοποίηση, ή απλά *Δικτυακή Βελτιστοποίηση*, βλέπε Σχήμα 1.1 (www-fp.mcs.anl.gov/otc/Guide/OptWeb), είναι μια ερευνητική περιοχή η οποία με τη σειρά της αποτελεί υποσύνολο της Γραμμικής Βελτιστοποίησης. Η Δικτυακή Βελτιστοποίηση επικεντρώνεται κυρίως σε εκείνα τα γραμμικά προβλήματα, τα οποία είναι δυνατό να μοντελοποιηθούν χρησιμοποιώντας ειδικές δομές δεδομένων, όπως τα γραφήματα και τα δίκτυα.

Ίσως η σημαντικότερη κατηγορία προβλημάτων της Επιστημονικής περιοχής της Δικτυακής Βελτιστοποίησης είναι το Πρόβλημα Ροής Ελαχίστου Κόστους, το οποίο θα ονομάζουμε ΠΡΕΚ στη συνέχεια για λόγους συντομίας της παρούσης (Minimum Cost Network Flow Problem ή απλά MCNFP). Το ΠΡΕΚ περιγράφεται αναλυτικά σε πληθώρα βιβλίων όπως για παράδειγμα:



Σχήμα 1.1 Δένδρο Βελτιστοποίησης

(Paparrizos, Samaras, and Sifaleras 2005), (Ahuja, Magnanti, and Orlin. 1993) και (Bertsekas 1991).

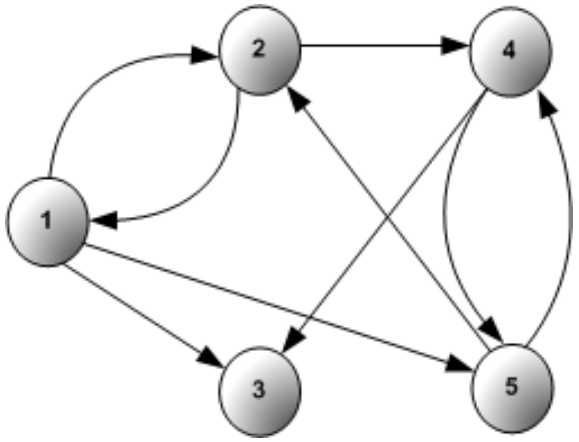
Πολλά σημαντικά προβλήματα του κλάδου της Δικτυακής Βελτιστοποίησης όπως το πρόβλημα της εύρεσης ελαχίστων δρόμων (Dijkstra 1959), το πρόβλημα της εύρεσης μεγίστης ροής (Ford and Fulkerson 1957), καθώς και το πρόβλημα της μεταφοράς (Dantzig 1951), ή αντιστοίχισης (Balinski 1985), είναι ειδικές περιπτώσεις του ΠΡΕΚ. Το ΠΡΕΚ αποτελεί μια ειδική περίπτωση - υποπρόβλημα του γενικού γραμμικού προβλήματος και για το λόγο αυτό είναι επιλύσιμο σε πολυωνυμικό χρόνο. Λόγω της ξεχωριστής συνδυαστικής δομής, έχουν βρεθεί αρκετοί αποτελεσματικοί αλγόριθμοι για το ΠΡΕΚ.

Παρ΄ ότι το ΠΡΕΚ μπορεί να λυθεί κάνοντας χρήση αλγορίθμων του γραμμικού προγραμματισμού, όλοι σχεδόν οι αλγόριθμοι για το ΠΡΕΚ αξιοποιούν την ιδιαίτερη συνδυαστική δομή του προβλήματος ούτως ώστε να γίνονται πιο αποτελεσματικοί. Σήμερα, υπάρχει μια πληθώρα άλλων

αλγορίθμων, οι οποίοι είναι αρκετα αποτελεσματικοί.

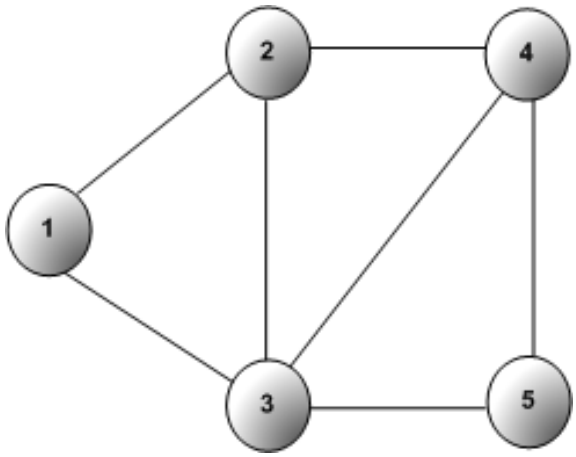
1.2 Ορισμοί και ορολογία

Σκόπιμο είναι, στο σημείο αυτό μια αναλυτική περιγραφή των ορισμών και της ορολογίας που χρησιμοποιούνται αμέσως στα επόμενα κεφάλαια. Δύο σύνολα N και A, τα οποία συμβολίζονται με $G = (N, A)$ αποτελούν ένα *γράφημα* ή αλλιώς *γράφο* (graph), G. Κάθε στοιχείο του συνόλου N ονομάζεται *κόμβος* (node), ή *σημείο* (point), ή *κορυφή* (vertice). Το σύνολο A αποτελείται από ζευγάρια κόμβων - στοιχείων του συνόλου N. Τα στοιχεία του συνόλου A παριστάνονται με (i, j) και από εδώ και έπειτα θα τα ονομάζουμε απλά ώς *τόξα* (arcs). Κάθε τόξο (i, j) διαφοροποιείται από το αντίστοιχο (j, i) τόξο. Τα γραφήματα τέτοιου είδους ονομάζονται *προσανατολισμένα* (directed). Αν τα ζεύγη κόμβων του συνόλου A δεν είναι προσανατολισμένα, οπότε ισχύει $(i, j) = (j, i)$, τότε το γράφημα ονομάζεται *μη προσανατολισμένο* (non directed) και τα ζεύγη κόμβων του A αποτελούν *ακμές* (edges). Οι ακμές συνήθως συμβολίζονται με παρόμοιο τρόπο όπως τα τόξα, δηλαδή η ακμή μεταξύ των κόμβων i και j συμβολίζεται (i, j).



Σχήμα 1.2 Προσανατολισμένο γράφημα

Για καλύτερη - εποπτική εικόνα των γράφων, οι γράφοι θα απεικονίζονται με γραφικό τρόπο

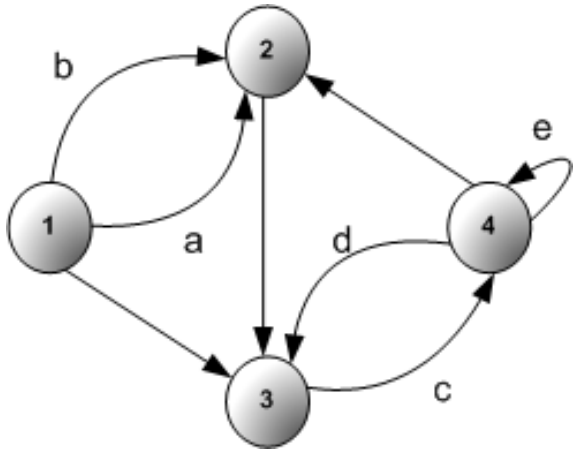


Σχήμα 1.3 Μη προσανατολισμένο γράφημα

ως εξής. Με κύκλο θα παριστάνεται ο κάθε κόμβος, στο κέντρο του οποίου θα αναγράφεται ένας αριθμός, ο οποίος θα δηλώνει το όνομα του συγκεκριμένου κόμβου. Ο πληθικός αριθμός των κόμβων συμβολίζεται με n, $(n = |N|)$. Με αυτόν τον τρόπο, οι κόμβοι θα παριστάνονται ως 1, 2, ..., n. Αντίστοιχα, με βέλος θα συμβολίζεται το κάθε τόξο. Για την ακρίβεια, το τόξο (i, j) απεικονίζεται με ένα βέλος, που εξέρχεται από τον κόμβο i και εισέρχεται τον κόμβο j. Επιπλέον, τον κόμβο i θα τον αναφέρουμε και ως κόμβο *ουρά* (tail), ενώ τον κόμβο j θα τον ονομάζουμε και ως κόμβο *κεφαλή* (head) αντίστοιχα. Ο αριθμός των τόξων συμβολίζεται με m, $(m = |A|)$. Τέλος, οι κόμβοι i και j οι οποίοι αποτελούν τόξο (i, j), ονομάζονται επίσης και *άκρα* (ends) του τόξου. Στο Σχήμα 1.2 απεικονίζεται ένα προσανατολισμένο γράφημα για το οποίο ισχύει $N = \{1, 2, 3, 4, 5\}$ και $A = (1, 2), (1, 3), (1, 5), (2, 1), (2, 4), (4, 3), (4, 5), (5, 2), (5, 4)$. Συνεπώς έχουμε $n = 5$ και $m = 9$. Οι ακμές (i, j) συμβολίζονται με γραμμές (όχι με βέλη) που ενώνουν τους κόμβους i και j. Στο Σχήμα 1.3 εμφανίζεται ένας μη προσανατολισμένος γράφος με σύνολα κόμβων και ακμών να είναι $N = \{1, 2, 3, 4, 5\}$ και $A = \{(1, 2), (1, 3), (2, 3), (2, 4), (3, 4), (3, 5), (4, 5)\}$ αντίστοιχα.

Παράλληλα, (parallel) ονομάζονται δύο ή παραπάνω τόξα που έχουν την ίδια αρχή i και ίδιο τέλος j, $(i \neq j)$. Επιπρόσθετα δε, ένα τόξο με αρχή και τέλος τον ίδιο κόμβο, $(i = j)$, ονομάζεται *βρόχος* (loop). Τα γραφήματα που περιέχουν παράλληλα τόξα ή / και βρόχους καλούνται *υπεργραφήματα*

(hypergraphs). Στη συνέχεια της Διατριβής, όταν θα υπάρχει ο όρος γράφημα θα εννοείται ένα προσανατολισμένο γράφημα δίχως παράλληλα τόξα, ούτε βρόχους. Στο Σχήμα 1.4 απεικονίζεται ένα υπεργράφημα. Τα τόξα a και b αφού έχουν την ίδια ουρά, τον κόμβο 1 και την ίδια κεφαλή, τον κόμβο 2 είναι παράλληλα. Απο την άλλη μεριά, τα τόξα c και d (έχουν τα ίδια άκρα τους κόμβους 3 και 4, αλλά δεν έχουν ίδια ουρά και ίδια κεφαλή) και ως εκ τούτου δεν είναι παράλληλα. Τέλος, τό τόξο e αποτελεί βρόχο, ενώ ένας γράφος που περιέχει ταυτόχρονα τόξα και ακμές ονομάζεται *μικτό*.

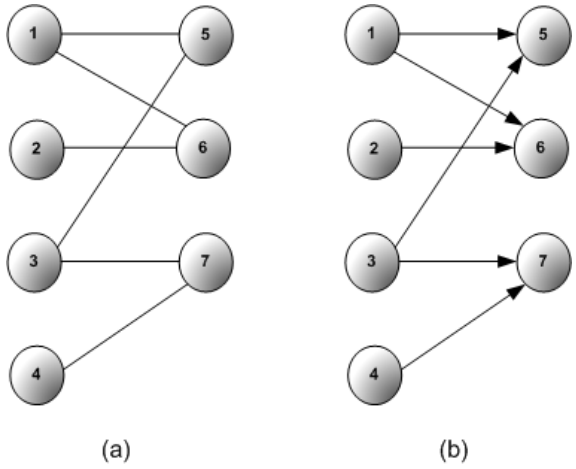


Σχήμα 1.4 Ένα υπεργράφημα

Τα μη προσανατολισμένα γραφήματα αποτελούν ειδικές περιπτώσεις των προσανατολισμένων γραφημάτων. Ένα μη προσανατολισμένο γράφημα μπορεί να μετατραπεί σε προσανατολισμένο με την αντικατάσταση κάθε ακμής (i, j) με άλλα δύο τόξα, (i, j) και (j, i).

Ένα γράφημα, με προσαρτημένους αριθμούς στους κόμβους ή και στα τόξα του ονομάζεται *δίκτυο* (network). Άν το σύνολο των κόμβων N ενός γραφήματος είναι διαμερισμένο, σε δύο υποσύνολα $N1$ και $N2$ με τέτοιο τρόπο ώστε όλα τα τόξα να έχουν ένα άκρο στο $N1$ και ένα άκρο στο $N2$, τότε το γράφημα ονομάζεται *διμερές* (bipartite graph). Ένα διμερές γράφημα μπορεί να είναι είτε προσανατολισμένο, είτε μη προσανατολισμένο ή ακόμη και μικτό. Στο Σχήμα 1.5 (a) απεικονίζεται ένα μη προσανατολισμένο διμερές γράφημα, ενώ στο Σχήμα 1.5 (b) φαίνεται ένα

προσανατολισμένο διμερές γράφημα.

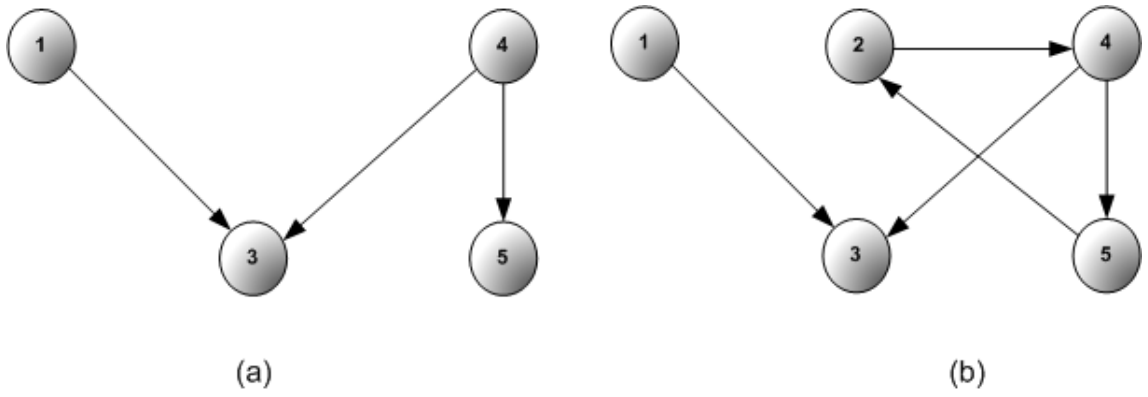


Σχήμα 1.5 Ένας διμερής (a) και ένας διμερής προσανατολισμένος γράφος (b)

Οι κόμβοι i και j προσπίπτουν στο τόξο (i, j), ενώ το τόξο (i, j) προσπίπτει στους κόμβους i και j. Ακόμη, όπου γίνεται λόγος ότι από ένα γράφο αφαιρείται ο κόμβος i, θα εξυπακούεται ότι ταυτόχρονα διαγράφονται και όλα τα τόξα που προσπίπτουν σ' αυτόν. Έστω $F = (N', A')$ ένα γράφημα που προκύπτει από το γράφημα $G = (N, A)$ διαγράφοντας κόμβους ή και τόξα. Τότε το γράφημα F θα αποτελεί *υπογράφημα* (subgraph) του G. Ισχύει πάντοτε ότι $N' \subseteq N$ και $A' \subseteq A$.

Το πλήθος των ακμών ή τόξων που προσπίπτουν στον κόμβο i το ονομάζουμε *βαθμό* (degree) του κόμβου i. Πιο συγκεκριμένα, ο αριθμός των τόξων με αρχή τον κόμβο i ονομάζεται *έξω βαθμός* (out degree), ενώ ο αριθμός των τόξων με τέλος τον κόμβο i ονομάζεται *έσω βαθμός* (in degree). Όπως φαίνεται στο Σχήμα 1.2, ο βαθμός του κόμβου 1 στο γράφημα του είναι 4, ο έξω βαθμός του είναι 3 και ο έσω βαθμός του είναι 1. *Απομονωμένος* (isolated) ονομάζεται ένας κόμβος με μηδενικό βαθμό, ενώ *φύλλο* (leaf) ονομάζεται ένας κόμβος με βαθμό 1. Στο γράφημα του Σχήματος 1.6 (a) οι κόμβοι 1 και 5 αποτελούν φύλλα. Αν (i, j) είναι τόξο του γραφήματος, τότε ο κόμβος j ονομάζεται *έξω γειτονικός* κόμβος του i, ενώ ο i ονομάζεται *έσω γειτονικός* κόμβος του j. Σε ένα τόξο (i, j) του γραφου, ο κόμβος j ονομάζεται *διπλανός* (adjacent) του κόμβου i και ο i διπλανός

του j. Το σύνολο των έξω γειτονικών κόμβων ονομάζεται *έξω γειτονιά* (out neighborhood) ενώ το σύνολο των έσω γειτονικών κόμβων ονομάζεται *έσω γειτονιά* (in neighborhood). Με τον όρο δε γειτονιά (neighborhood) ορίζουμε την ένωση της έξω και έσω γειτονιάς. Η έξω γειτονιά του κόμβου 2 στο γράφημα του Σχήματος 1.2 είναι το σύνολο των κόμβων $\{1, 4\}$, η έσω γειτονιά του είναι το σύνολο $\{1, 5\}$, ενώ η γειτονιά του το σύνολο $\{1, 5, 4\}$.



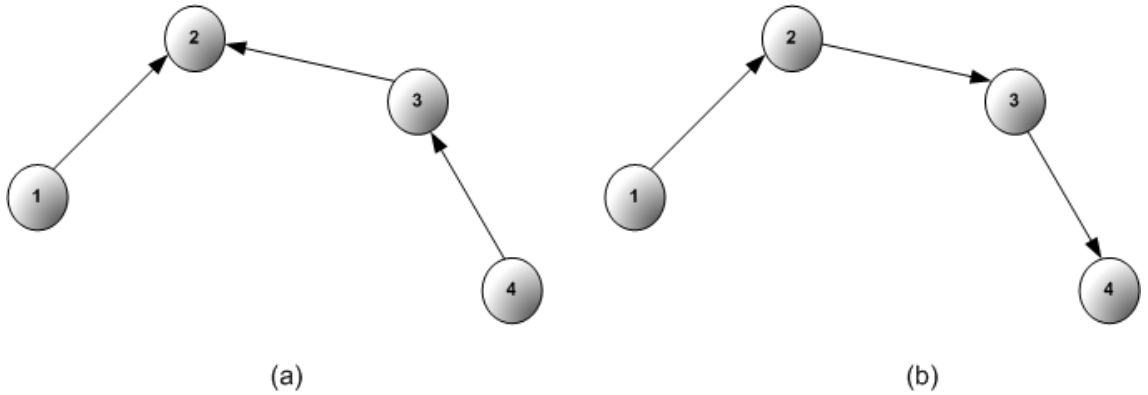
Σχήμα 1.6 Παράδειγμα αλυσίδων

Σε ένα γράφημα $G = (N, A)$ *αλυσίδα* (chain) ονομάζουμε μια αλληλουχία από κόμβους i_k και τόξα a_k, $i_1 - a_1 - i_2 - a_2 - \ldots - i_{r-1} - a_{r-1} - i_r$, η οποία αρχίζει και τελειώνει με κόμβο έτσι ώστε για κάθε $1 \leq k \leq r - 1$ να είναι $a_k = (i_k, i_{k+1}) \in A$ ή $a_k = (i_{k+1}, i_k) \in A$. Στο Σχήμα 1.6 απεικονίζονται δύο αλυσίδες του γράφου του Σχήματος 1.2. Πιο συγκεκριμένα, η αλυσίδα του Σχήματος 1.2 (a) είναι 1 - (1, 3) - 3 - (4, 3) - 4 - (4, 5) - 5, ενώ η αλυσίδα του Σχήματος 1.2 (b) είναι 1 - (1, 3) - 3 - (4, 3) - 4 - (4, 5) - 5 - (5, 2) - 2 - (2, 4) - 4. Σε μια αλυσίδα επιτρέπεται να υπάρχει ένας κόμβος ή ένα τόξο περισσότερες από μία φορές. Μία αλυσίδα είναι ένα γράφημα που σχηματίζεται από ένα άλλο γράφημα $G = (N, A)$ με κατάλληλο τρόπο. Επιλέγουμε ένα κόμβο i_1 και ένα τόξο που έχει ένα άκρο τον κόμβο i_1. Έστω i_2 το άλλο άκρο του τόξου. Επιλέγουμε τώρα ένα τόξο που έχει ένα άκρο τον κόμβο i_2 και άλλο άκρο ένα κόμβο i_3 κλπ. Εάν ισχύει $i_1 = i_r$ τότε η αλυσίδα ονομάζεται *κλειστή* (closed), ενώ αν ισχύει $i_1 \neq i_r$ τότε η αλυσίδα ονομάζεται *ανοιχτή*

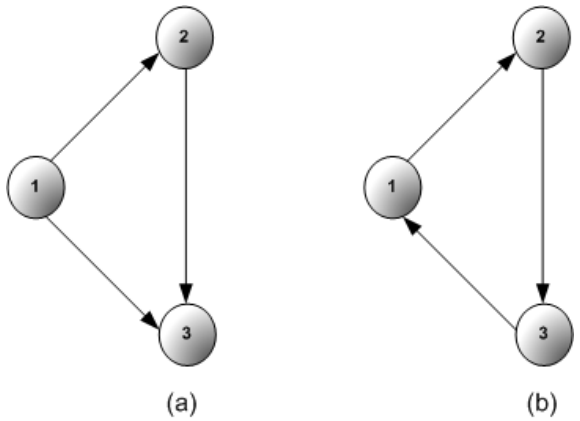
(open). Μια αλυσίδα ονομάζεται *προσανατολισμένη* (directed) αν για κάθε $1 \leq k \leq r-1$ ισχύει ότι $a_k = (i_k, i_{k+1})$, δηλαδή, αν για κάθε ζευγάρι διαδοχικών κόμβων i_k και i_{k+1} της αλυσίδας, το τόξο a_k έχει προσανατολισμό από τον κόμβο i_k στον κόμβο i_{k+1}. Οι αλυσίδες δεν προσδιορίζονται μονοσήμαντα με τη γραφική τους αναπαράσταση. Επομένως, πρέπει οπωσδήποτε να αναφέρεται η αλληλουχία των κόμβων και τόξων. Η αλυσίδα 1 - (1, 3) - 3 - (4, 3) - 4 - (2, 4) - 2 - (5, 2) - 5 παράδειγματος χάρην, όπως και η προηγούμενη, παριστάνεται ξανά με το γράφο του Σχήματος 1.6 (b).

Δρόμος (path) καλείται μία ανοιχτή αλυσίδα, όταν κάθε κόμβος (και κάθε τόξο) εμφανίζεται μονάχα μια φορά. Σε ένα δρόμο, αρκεί να καταγραφούν τα τόξα του. Στο σχήμα 1.6 (a) η αλυσίδα αποτελεί δρόμο, ενώ στο Σχήμα 1.6 (b) δεν αποτελεί, αφού ο κόμβος 4 εμφανίζεται δύο φορές. *Εμπροσθοδρομικό* (forward) ονομάζεται το τόξο (i, j) ενός δρόμου, όταν κατά την πορεία από τον πρώτο κόμβο i_1 προς τον τελευταίο i_r συναντάμε πρώτα τον κόμβο i και μετά τον κόμβο j. Σε αντίθετη περίπτωση το τόξο ονομάζεται *οπισθοδρομικό* (*backward*). *Προσανατολισμένο* δρόμο (directed path) ονομάζουμε εκείνο το δρόμο, του οποίου είτε όλα τα τόξα είναι εμπροσθοδρομικά είτε όλα είναι οπισθοδρομικά. Στο Σχήμα 1.7 (a) ο δρόμος είναι μη προσανατολισμένος ενώ στο Σχήμα 1.7 (b) είναι προσανατολισμένος. Για ένα προσανατολισμένο δρόμο αρκεί να γράφονται οι κόμβοι του με τη σειρά.

Άν σε ένα δρόμο ο πρώτος και ο τελευταίος κόμβος είναι ίδιος ($i_1 = i_r$), τότε ονομάζεται κλειστός δρόμος ή *κύκλος* (cycle). Άν ο δρόμος, από τον οποίο προκύπτει ο κύκλος, είναι προσανατολισμένος, τότε ο κύκλος καλείται *προσανατολισμένος* (directed). Σε αντίθετη περίπτωση ονομάζεται *μη προσανατολισμένος* (not directed). Στη συνέχεια της Διατριβής, όταν αναφερόμαστε σε δρόμο ή κύκλο, θα εννοούμε ένα μη προσανατολισμένο δρόμο ή κύκλο. Στο σχήμα 1.8 (a) ο κύκλος είναι μη προσανατολισμένος ενώ στο Σχήμα 1.8 (b) είναι προσανατολισμένος. Κύκλωμα (circuit) καλούμε ένα προσανατολισμένο κύκλο, ενώ *άκυκλο* γράφημα ονομάζουμε ένα γράφημα



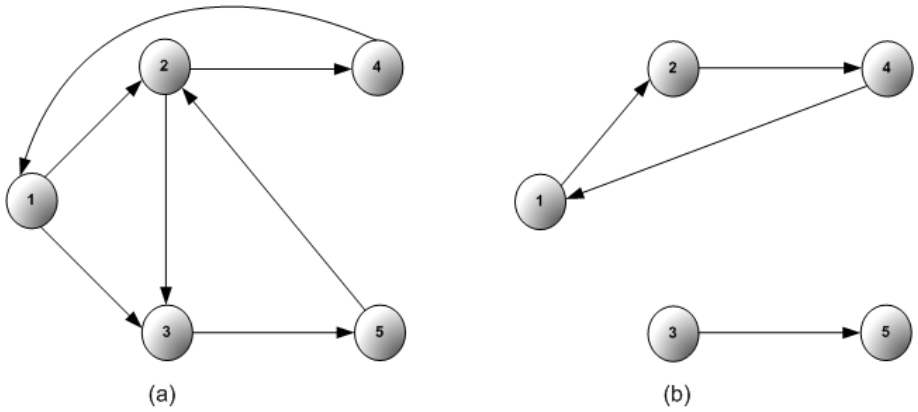
Σχήμα 1.7 Παράδειγμα μη προσανατολισμένου (a) και προσανατολισμένου (b) δρόμου δίχως προσανατολισμένους κύκλους.



Σχήμα 1.8 Παράδειγμα μη προσανατολισμένου (a) και προσανατολισμένου (b) κύκλου

Συνδεμένοι (connected) ονομάζονται δύο κόμβοι i και j ενός γράφου $G = (N, A)$, εάν υπάρχει τουλάχιστον ένας δρόμος (προσανατολισμένος ή μη) με άκρα τους κόμβους i και j. Ο γράφος G καλείται *συνεκτικός* (connected), αν κάθε ζευγάρι δύο κόμβων είναι συνδεμένο. Ειδάλλως, το γράφημα ονομάζεται μη συνεκτικό (disconnected). Συνιστώσες (components) ονομάζονται οι μέγιστοι συνεκτικοί υπογράφοι ενός μη συνεκτικού γράφου. Στο Σχήμα 1.9 (a) ο γράφος είναι συνεκτικός, ενώ αντιθέτως στο Σχήμα 1.9 (b) δεν είναι (μή συνεκτικός) και αποτελείται από δυο συνιστώσες. Αν για ένα οποιοδήποτε ζευγάρι κόμβων i και j υπάρχει τουλάχιστον ένας προσανατολισμένος δρόμος

από τον κόμβο i στον κόμβο j και τουλάχιστον ένας προσανατολισμένος δρόμος από τον κόμβο j στον κόμβο i, τότε ο γράφος ονομάζεται *ισχυρά* συνεκτικός (strongly connected). Στο Σχήμα 1.9 (a) το γράφημα είναι ισχυρά συνεκτικό.



Σχήμα 1.9 Παράδειγμα συνεκτικού (a) και μη συνεκτικού (b) γράφου

Μια διαμέριση του συνόλου των κόμβων του σε δυο υποσύνολα S και S' καλείται *τομή* (cut) ενός γραφου $G = (N, A)$ και συμβολίζεται με $[S, S']$. Ως τομή ακόμη θα καλούμε και το σύνολο των τόξων του A που έχουν το ένα άκρο στο S και το άλλο στο S'. Αν s και t είναι δύο διαφορετικοί κόμβοι του N, μια $s-t$ τομή $(s-tcut)$ αποτελεί μια τομή $[S, S']$ ούτως ώστε $s \in S$ και $t \in S'$. Μια 1-5 τομή $[S, S']$ του γραφήματος του Σχήματος 1.9 (a) είναι $S = \{1, 2, 3\}$ και $S' = \{4, 5\}$. Τα τόξα αυτής της τομής είναι $[S, S'] = \{(2, 4), (4, 1), (3, 5), (5, 2)\}$.

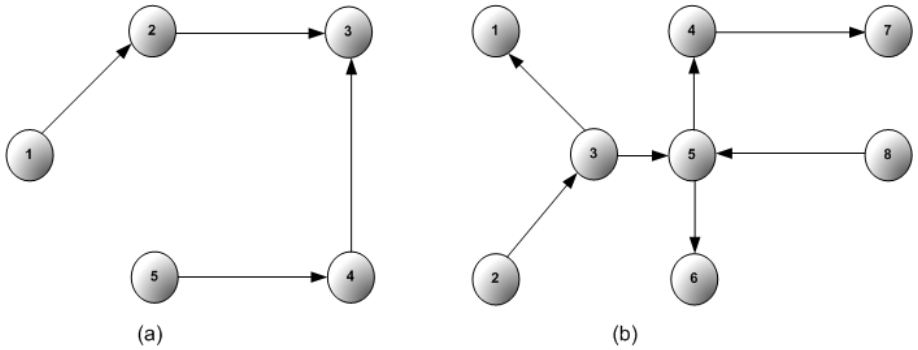
Τα δένδρα (trees) πιθανώς αποτελούν τα πιο σημαντικά γραφήματα. Βρίσκονται σχεδόν σε όλους τους αλγόριθμους Δικτυακής Βελτιστοποίησης και συνήθως η πλειοψηφία των αλγορίθμων περιγράφεται με όρους δέντρων. Ένα συνεκτικό γράφημα δίχως κύκλους θα καλείται *δένδρο*, (για τα δένδρα γενικώς υπάρχουν πολλοί ισοδύναμοι ορισμοί). Στο Σχήμα 1.10 εμφανίζονται δύο δέντρα. Τα δένδρα έχουν μια πληθώρα ενδιαφέροντων ιδιοτήτων. Μερικές απ' αυτές είναι :

- Ένα δέντρο με n κόμβους έχει n - 1 τόξα
- Ένα δέντρο έχει τουλάχιστον δυο φύλλα

- Κάθε δυο κόμβοι συνδέονται με ακριβώς ένα δρόμο

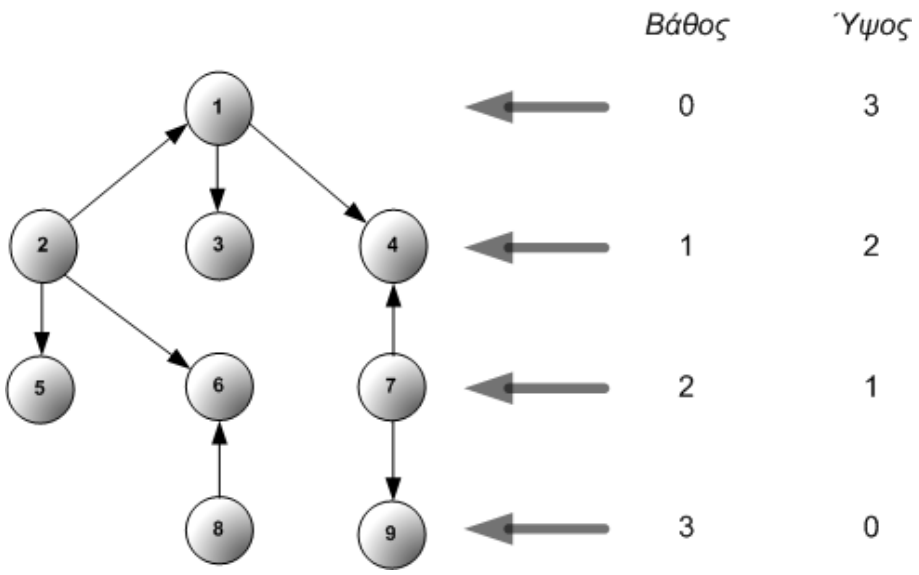
Δάσος (forest) ονομάζεται ένα γράφημα που αποτελεί συλλογή από δέντρα. Υπάρχει πληθώρα αλγορίθμων στη διεθνή βιβλιογραφία, οι οποίοι κάνουν χρήση δασών. Για παράδειγμα ο αλγόριθμος Παπαρρίζου για το πρόβλημα της μεταφοράς με δάσος ξεκινήματος το δάσος Α.Κ.Π. (Achatz, Kleinschmidt, and Paparrizos 1991) ή ο αλγόριθμος εξωτερικών σημείων με χρήση δάσους για το πρόβλημα της αντιστοίχισης (Achatz, Paparrizos, Samaras, and Tsiplidis 2002). Σε αυτούς τους αλγορίθμους, η αρχικοποίηση δεν γίνεται με τη βοήθεια ενός ριζωμένου δένδρου (βασική λύση) αλλά αντιθέτως με μικρά ριζωμένα δένδρα που αποτελούν το δάσος.

Τα *ριζωμένα* δέντρα (rooted trees) αποτελούν μια σημαντική κλάση δέντρων, όπου κάποιος κόμβος, η ρίζα (root), είναι διακεκριμένος, ενώ οι όλοι υπόλοιποι κόμβοι βρίσκονται σε διαδοχικά αριθμημένα επίπεδα 1, 2, 3,... . Ο κόμβος - ρίζα βρίσκεται στο μηδενικό επίπεδο. Ένα παράδειγμα ριζωμένου δένδρου απεικονίζεται στο Σχήμα 1.11. Ο κόμβος 1 αποτελεί τη ρίζα του δένδρου. Οι κόμβοι 2, 3 και 4, βρίσκονται στο πρώτο επίπεδο, οι κόμβοι 5, 6 και 7 βρίσκονται στο δεύτερο, ενώ το τρίτο επίπεδο περιλαμβάνει δυο κόμβους, τους κόμβους 8 και 9.



Σχήμα 1.10 Παράδειγμα δένδρων

Σε ένα δέντρο T, ο κόμβος j ονομάζεται *παιδί* (child) ή αμέσως επόμενος (immediate successor) του κόμβου i, ενώ ο κόμβος i ονομάζεται *πατέρας* (father) ή αμέσως προηγούμενος (immediate predecessor) του κόμβου j, όταν συμβαίνει το εξής. Ο i κόμβος ανήκει στο επίπεδο k και ο j κόμβος

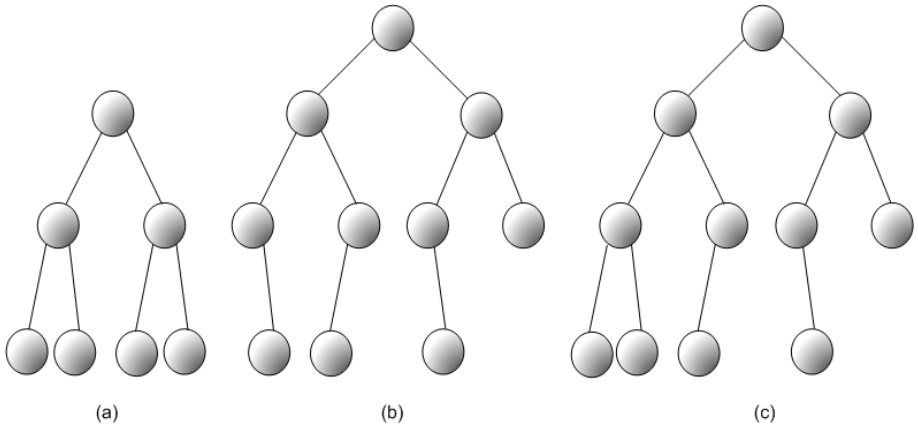


Σχήμα 1.11 Ένα ριζωμένο δένδρο με τα βάθη και ύψη των κόμβων του

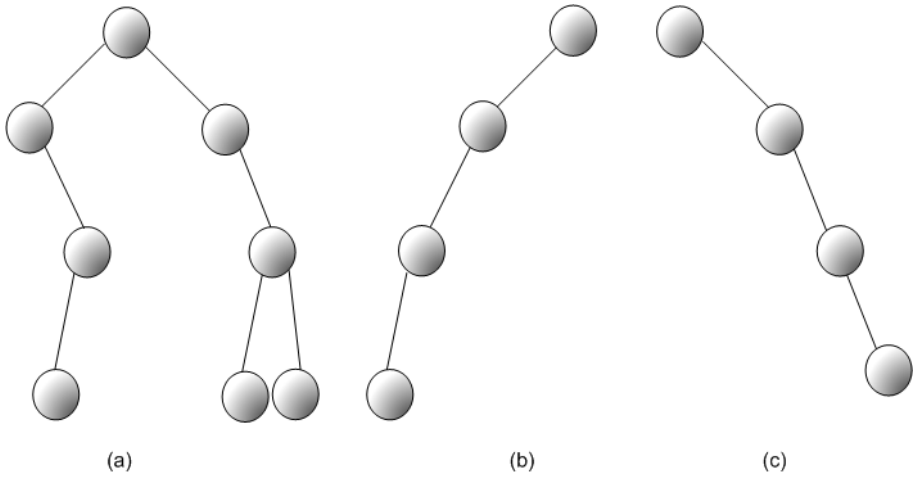
στο επίπεδο $k+1$ έτσι ώστε $(i,j) \in T$ ή $(j,i) \in T$. Το μοναδικό δρόμο στο δένδρο T που συνδέει τον κόμβο j με τη ρίζα, θα το συμβολίζουμε με $P[T,j]$. *Πρόγονος* (predecessor) του κόμβου j ονομάζεται ένας κόμβος $i \in P[T,j]$ με $i \neq j$, ενώ ο κόμβος j ονομάζεται απόγονος (successor) του κόμβου i. Για παράδειγμα, στο Σχήμα 1.11 ο κόμβος 9 αποτελεί απόγονο του κόμβου 4, ενώ ο κόμβος 2 είναι πρόγονος του κόμβου 8. Τα φύλλα έχουν μόνο προγόνους, ενώ απεναντίας η ρίζα έχει μόνο απογόνους.

Βάθος (depth) του δένδρου θα ονομάζεται το μέγιστο επίπεδο h. Παρόμοια, βάθος (depth) του κόμβου ονομάζεται το επίπεδο k, στο οποίο βρίσκεται κάποιος κόμβος . Ένας ισοδύναμος ορισμός του βάθους ενός δένδρου είναι το μέγιστο βάθος των κόμβων. Ύψος (height) ενός κόμβου βάθους i, ο οποίος ανήκει σε ένα δένδρο βάθους h, ονομάζεται ο αριθμός $h-i$. Με άλλα λόγια, το ύψος ενός κόμβου αποτελεί την απόστασή του από το χαμηλότερο επίπεδο του δέντρου. Η ρίζα πάντοτε βρίσκεται σε ύψος $h-0=h$, ενώ στο επίπεδο 0 περιέχονται οι κόμβοι βάθους h. Πάλι, ένας ισοδύναμος ορισμός του ύψος ενός δέντρου είναι ο αριθμός h. Για παράδειγμα, στο Σχήμα 1.11 φαίνονται τα ύψη των κόμβων. Πρέπει εδώ να γίνει λόγος και για τα δυαδικά δέντρα

(binary trees), όπου ο κάθε κόμβος έχει το πολύ δυο παιδιά. Κάποια παραδείγματα δυαδικών δένδρων παρουσιάζονται στα Σχήματα 1.12 και 1.13. Στο σχήμα 1.12 τα δυαδικά δέντρα είναι πλήρη (complete). Τα πλήρη δυαδικά δέντρα σε κάθε επίπεδο βάθους $j, 1 \leq j \leq h-1$ περιέχουν το μέγιστο δυνατό πλήθος κόμβων, δηλαδή $2j$. Τέλος, *ισχυρώς πλήρη* δένδρα (strongly complete trees) ονομάζονται τα πλήρη δυαδικά δένδρα, τα οποία έχουν τα φύλα τους στο τελευταίο επίπεδο βάθους h, συγκεντρωμένα όσο γίνεται αριστερά. Στο Σχήμα 1.12 απεικονίζονται τα ισχυρώς πλήρη δένδρα (a) και (c).



Σχήμα 1.12 Μερικά πλήρη δυαδικά δέντρα



Σχήμα 1.13 Μερικά δυαδικά (μη πλήρη) δέντρα

Για την οπτικοποίηση δένδρων με τη βοήθεια ηλεκτρονικού υπολογιστή έχει κατασκευαστεί πληθώρα λογισμικού. Ένα εκπαιδευτικό λογισμικό οπτικοποίησης, όπου υλοποιούνται οι δομές που περιγράφονται σε αυτήν την διατριβή, παρουσιάστηκε από τον Σιφαλέρα Α. στο 1ο Διεθνές Συνέδριο με θέμα *New Learning Paradigms and New Learning Tools* το 2004, με την εργασία (Paparrizos, Samaras, and Sifaleras 2004). Το συγκεκριμένο λογισμικό, γραμμένο σε γλώσσα Matlab, απαιτεί από τον χρήστη τρία διανύσματα εισόδου, που περιγράφουν ένα δένδρο. Το διάνυσμα πατέρων, το διάνυσμα της προδιάταξης και τέλος ένα διάνυσμα που περιγράφει τη φορά των τόξων. Ο χρήστης είναι σε θέση να οπτικοποιήσει ένα γενικευμένο ριζωμένο δένδρο, με κατευθυνόμενα ή μή τόξα.

Αξίζει να αναφερθεί ότι η εγγενής συνάρτηση treeplot του Matlab, μέχρι και την έκδοση R2006b, δεν επιτυγχάνει καλή οπτικοποίηση του δένδρου, αφου απεικονίζει λανθασμένα όλους τους κόμβους - φύλλα στο μέγιστο επίπεδο. Επίσης, η συνάρτηση treeplot του Matlab δεν απεικονίζει καθόλου κατευθυνόμενα τόξα, παρά μονάχα ακμές.

1.3 Μαθηματική μορφοποίηση του Προβλήματος Ροής Ελαχίστου Κόστους (ΠΡΕΚ)

Έστω ένα δίκτυο $G = (N, A)$, όπου N είναι ένα σύνολο n κόμβων και A είναι ένα σύνολο m τόξων. Με (i, j) συμβολίζεται το τόξο από τον κόμβο i στον κόμβο j. Για κάθε κόμβο $i \in N$, b_i είναι η ποσότητα προϊόντος που διαθέτει ο κάθε κόμβος. Εάν $b_i > 0$, ο κόμβος i, διαθέτει περρίσευμα ποσότητας και ονομάζεται *κόμβος προσφοράς*, εαν $b_i < 0$ ο κόμβος i, θα έχει έλλειμα σε ποσότητα και ονομάζεται *κόμβος ζήτησης*, ενώ εάν τέλος $b_i = 0$ ο κόμβος i, ούτε θα προσφέρει ούτε θα ζητάει κάποια ποσότητα, οπότε και θα ονομάζεται *κόμβος μεταφόρτωσης*. Για κάθε $(i, j) \in A$, έστω c_{ij} θα δηλώνει το κόστος της ροής 1 μονάδας προϊόντος, δια μέσου του (i, j).

Το ΠΡΕΚ ασχολείται με την εύρεση ροής ελαχίστου κόστους σε δίκτυο. Τη ροή ενός δικτύου

G θα τη συμβολίζουμε με ένα διάνυσμα $x = [x_{ij}, x_{gh}, x_{kl}, ...]$, όπου το κάθε στοιχείο π.χ. x_{ij} θα αντιστοιχεί στη ροή του βασικού τόξου (i, j). Σύμφωνα με το Γραμμικό Προγραμματισμό, η μαθηματική του μορφή περιγράφεται από τις σχέσεις (NP.1) που ακολουθούν :

$$
\begin{aligned}
&minimize \sum_{\text{(i,j)}\in\text{A}} \text{c}_{\text{ij}} x_{ij} \\
&subject\ to \sum_{\text{i:(k,i)}\in\text{A}} x_{ki} - \sum_{\text{j:(j,k)}\in\text{A}} x_{jk} = b_k, k \in N \qquad \text{(NP.1)} \\
&\qquad 0 \leq x_{ij} \leq +\infty, \forall (i,j) \in A
\end{aligned}
$$

Θα θεωρήσουμε ότι έχουμε δίκτυο χωρίς άνω χωρητικότητες, ενώ η κάτω χωρητικότητα $\forall (i, j) \in A$ ισούται με 0. Σύμφωνα με την παραπάνω μορφοποίηση, περιορισμοί της μορφής $\sum x_{ki} - \sum x_{jk} = b_k$. είναι γνωστοί ώς περιορισμοί συντήρησης της ροής, ενώ περιορισμοί της μορφής $0 \leq x_{ij} \leq +\infty$ ονομάζονται περιορισμοί της χωρητικότητας της ροής.

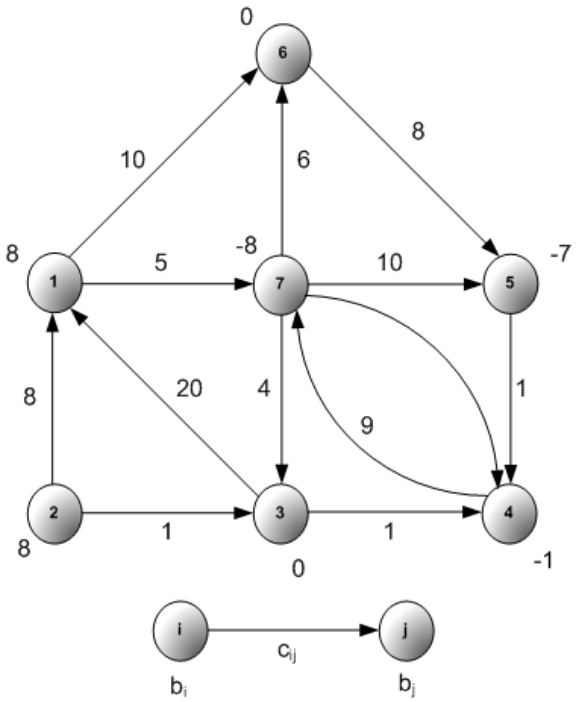
Σε μορφή μητρών το ΠΡΕΚ μπορεί να εκφραστεί ως ένα γραμμικό πρόβλημα με την εξής μορφή.

$$
\begin{aligned}
&minimize\ c^T x \\
&subject\ to\ Ax = b \qquad (NP.2) \\
&\qquad x \geq 0
\end{aligned}
$$

Όπου $A \in \Re^{nxm},\ c,\ x,\ \in \Re^m,\ b\ \in \Re^n$.

Θα δοθεί ένα παράδειγμα της μαθηματικής μορφοποίησης του ΠΡΕΚ. Έστω, ότι υπάρχει το δίκτυο του Σχήματος 1.14 και ζητείται να υπολογιστί μια ροή ελαχίστου κόστους. Όπως φαίνεται και στο Σχήμα, οι αριθμοί δίπλα στα τόξα υποδηλώνουν την τιμή κόστους του εκάστοτε τόξου. Αντίστοιχα, οι αριθμοί δίπλα στους κόμβους υποδηλώνουν την προσφορά ή τη ζήτηση του κάθε κόμβου.

Αμέσως παρακάτω, περιγράφονται αναλυτικά οι μήτρες οι οποίες αναφέρονται στις Σχέσεις (NP.2).



Σχήμα 1.14 Μαθηματική μορφοποίηση του ΠΡΕΚ

$$x^T = \begin{bmatrix} x_{16} & x_{17} & x_{21} & x_{23} & x_{31} & x_{34} & x_{47} & x_{54} & x_{65} & x_{73} & x_{74} & x_{75} & x_{76} \end{bmatrix}$$

$$A = \begin{bmatrix} 1 & 1 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 1 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 1 & -1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & -1 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & -1 \\ 0 & -1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}, b = \begin{bmatrix} 8 \\ 8 \\ 0 \\ -1 \\ -7 \\ 0 \\ -8 \end{bmatrix}$$

$$c^T = \begin{bmatrix} 10 & 5 & 8 & 1 & 20 & 1 & 9 & 1 & 8 & 4 & 9 & 10 & 6 \end{bmatrix}$$

Σε προβλήματα ροών δικτύων, η μήτρα Α των περιορισμών του προβλήματος, ονομάζεται *μήτρα κόμβων - τόξων*, (node-arc incidence matrix), του δικτύου G. Το ΠΡΕΚ, διαφέρει από την

κανονική μορφή, (canonical form), του γραμμικού προβλήματος σε δύο σημεία. Πρώτον, είναι πρόβλημα ελαχιστοποίησης, ενώ στο γραμμικό πρόβλημα μπορούμε να έχουμε είτε ελαχιστοποίηση, είτε μεγιστοποίηση. Κατά δεύτερον, οι περιορισμοί του προβλήματος είναι ισότητες αντί για ανισότητες.

Η ποσότητα του προϊόντος, που ρέει δια μέσω του δικτύου, δεν έχει άλλο τρόπο είτε να εισαχθεί είτε να εξαχθεί από το σύστημα, πέρα από τους κόμβους προσφοράς και ζήτησης. Ως εκ τούτου, συνεπάγεται ότι πρέπει η συνολική προσφορά να ισούται με τη συνολική ζήτηση, έτσι ώστε να υπάρχει μια εφικτή λύση. Συνεπώς, στη συνέχεια της παρούσης Διατριβής θα θεωρούμε ότι το ΠΡΕΚ είναι *ισοζυγισμένο*, δηλαδή ισχύει:

$$\sum_{i=1}^{n} b(i) = 0 \tag{1}$$

Απαραίτητο για τη συνέχεια της παρούσης διατριβής, είναι να εξηγηθεί πως εφαρμόζεται η δυϊκή θεωρία στο ΠΡΕΚ. Το δυϊκό του Προβλήματος (NP.1), μετά και την εισαγωγή χαλαρών μεταβλητών, περιγράφεται παρακάτω:

$$\begin{aligned} &maximize \sum_{i \in N} b_i w_i \\ &subject\ to\ s_{ij} = c_{ij} - w_i + w_j \\ &\qquad\qquad s_{ij} \geq 0 \end{aligned} \tag{NP.3}$$

Ένα διάνυσμα δυϊκών μεταβλητών του δικτύου G είναι ένα διάνυσμα w με n στοιχεία, όπου το κάθε στοιχείο w_i αντιστοιχεί σε έναν κόμβο i. Για κάθε διάνυσμα w, το ελλειματικό κόστος s_{ij} υπολογίζεται ως $s_{ij} = c_{ij} - w_i + w_j$, όπως φαίνεται και στις Σχέσεις (NP.3).

Επίσης, όλα τα δεδομένα θα θεωρούνται ακέραιοι αριθμοί. Πρέπει να αναφερθεί και το παρακάτω σημαντικό Θεώρημα 1, γνωστό και ώς Θεώρημα της ακεραιότητας, (Integrality Theorem).

Θεώρημα 1 *Σε προβλήματα ροών δικτύων με ακέραιες τιμές, κάθε βασική εφικτή λύση και πιο*

συγκεκριμένα κάθε βασική βέλτιστη λύση αντιστοιχεί μια ακέραιη τιμή ροής σε κάθε τόξο.

Αυτό το Θεώρημα είναι πολύ σημαντικό, διότι πολλές πρακτικές εφαρμογές προβλημάτων ροής ελαχίστου κόστους, έχουν ακέραιες τιμές προσφοράς και ζήτησης και απαιτούν οι λύσεις τους να έχουν ακέραιες τιμές επίσης. Αυτός ο περιορισμός τυγχάνει, όταν κάποιος πρέπει να μεταφέρει κάποιες αδιαίρετες μονάδες δια μέσω ενός δικτύου. Για παράδειγμα, δεν είναι λογικό να μεταφέρει κανείς από ένα εργοστάσιο αυτοκινήτων το $\frac{1}{4}$ ενός αυτοκινήτου, σε κάποια εμπορική αντιπροσωπεία, ενώ τα άλλα $\frac{3}{4}$ του ίδιου αυτοκινήτου, να διοχετευθούν σε έναν άλλο συνεργαζόμενο εμπορικό αντιπρόσωπο.

1.4 Πρακτικές εφαρμογές του ΠΡΕΚ

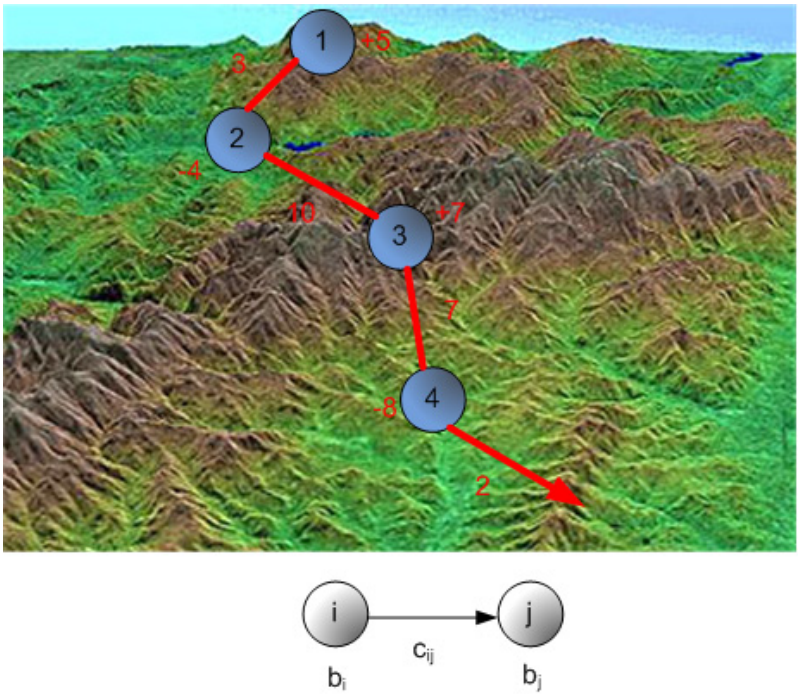
Ο ερευνητικός χώρος της Δικτυακής Βελτιστοποίησης έχει επεκταθεί ευρέως τις τελευταίες δεκαετίες και συμπεριλαμβάνει ένα αξιοσημείωτο εύρος εφαρμογών στο κλάδο της Πληροφορικής, των κατασκευών, των μεταφορών, των τηλεπικοινωνιών κ.λ.π., βλέπε πιο αναλυτικά (Glover and Klingman 1975), (Glover, Klingman, and Phillips. 1992) και (Ahuja, Magnanti, Orlin, and Reddy 1995). Χαρακτηριστικά σε αυτήν την ενότητα, θα περιγραφούν δύο πρακτικές εφαρμογές εύρεσης ροών ελαχίστου κόστους.

1.4.1 Ισοπέδωση ορεινού εδάφους, (*Levelling mountainous terrain*)

Η εφαρμογή αυτή (Farley 1980) ανήκει στο χώρο των κατασκευών. Πιο συγκεκριμένα, προέρχεται από ένα πρόβλημα που αντιμετωπίζουν οι πολιτικοί μηχανικοί στην κατασκευή οδικών δικτύων, δια μέσου λόφων και γενικότερα ορεινών εκτάσεων. Το πρόβλημα που τους απασχολεί είναι, η διανομή χωμάτων από υψηλά υψομετρικά σημεία στα χαμηλά υψομετρικά σημεία. Απώτερος στόχος είναι, να κατασκευαστεί ένας δρόμος, ο οποίος θα είναι ισοϋψής στο μεγαλύτερο μέρος (ή σε όλη) της διαδρομής. Ο μηχανικός θα πρέπει να σχεδιάσει για την ισοπέδωση της διαδρομής, ένα πλάνο

μεταφοράς των φορτωμένων φορτηγών σε χώμα, μεταξύ διαφόρων προορισμών στο προτεινόμενο οδικό δίκτυο.

Για να λυθεί το συγκεκριμένο πρόβλημα χρησιμοποιώντας κάποιον αλγόριθμο για το ΠΡΕΚ, πρέπει πρώτα να μοντελοποιηθεί σύμφωνα με το ΠΡΕΚ και έπειτα να γίνει η μαθηματική του μορφοποίηση. Οπότε, αρχικά κατασκευάζεται ένα γράφημα, το οποίο να απεικονίζει τη γεωγραφική περιοχή. Αυτό το γράφημα είναι μη κατευθυνόμενο και οι κόμβοι του παριστάνουν τοποθεσίες της διαδρομής με ζήτηση σε γή (χαμηλά υψομετρικά σημεία) ή τοποθεσίες με προσφορά σε γή (υψηλά υψομετρικά σημεία). Οι ακμές αυτού του γραφήματος παριστάνουν τα πιθανά δρομολόγια διανομής του χώματος - γής. Σε κάθε ακμή, αντιστοιχίζεται και μια τιμή κόστους, η οποία συμβολίζει το κόστος της μεταφοράς ενός φορτωμένου με χώμα φορτηγού, μεταξύ δύο τοποθεσιών - κόμβων. Ως βασική μονάδα διανομής της γής - χώματος, σε αυτήν την εφαρμογή θεωρείται το φόρτωμα ενός φορτηγού (truckload).



Σχήμα 1.15 Παράδειγμα ΠΡΕΚ με κατασκευή οδικού δικτύου σε ορεινό έδαφος

Στο Σχήμα 1.15 μπορεί να παρατηρήσει κανείς τις δυσκολίες κατασκευής ορεινού οδικού δικτύου (η αρχική εικόνα, δίχως την αλυσίδα κόμβων και τόξων είναι δανεισμένη από τον ηλεκτρονικό ιστοχώρο http://rst.gsfc.nasa.gov). Ένα πλάνο ισοπέδωσης του ανάγλυφου μιας γεωγραφικής περιοχής, στην ουσία αποτελεί μια ροή (το σύνολο των φορτωμάτων των φορτηγών), η οποία ικανοποιεί τους περιορισμούς ζήτησης των κόμβων (χαμηλών υψομετρικών σημείων) σε γή - χώμα, με την διαθέσιμη προσφορά των κόμβων (υψηλών υψομετρικών σημείων). Η ροή αυτή σαφώς πρέπει να έχει το ελάχιστο κόστος, για όλες τις κινήσεις των φορτωμένων φορτηγών.

1.4.2 Βέλτιστη φόρτωση περιοδεύοντος αεροπλάνου, (*Optimal loading of a hopping airplane*)

Το πρόβλημα αυτό, (Gupta 1985), βρίσκει εφαρμογή σε εταιρίες αερογραμμών. Έστω για παράδειγμα, ότι μια μικρή αεροπορική εταιρία χρησιμοποιεί ένα αεροπλάνο με χωρητικότητα τέτοια ώστε να χωράει το πολύ p επιβάτες σε μια πτήση - περιοδεία (*hopping flight*). Η περιοδεύουσα πτήση έστω ότι επισκέπτεται τις πόλεις 1, 2, ..., n, σε μία συγκεκριμένη σειρά. Το αεροπλάνο μπορεί να παίρνει επιβάτες σε κάθε κόμβο, αλλά και να τους αποβιβάζει σε οποιοδήποτε άλλον κόμβο. Έστω, ότι η ποσότητα b_{ij} δηλώνει το πλήθος των διαθέσιμων επιβατών στον κόμβο i, οι οποίοι επιθυμούν να πάνε στον κόμβο j. Επίσης, έστω ότι με f_{ij} δηλώνεται το αντίτιμο μεταφοράς ενός επιβάτη από τον κόμβο i στον κόμβο j. Η αεροπορική εταιρία επιθυμεί να καθορίσει τον αριθμό των επιβατών, ο οποίος πρέπει να μεταφερθεί μεταξύ των διαφόρων προορισμών. Το ζητούμενο πλήθος των ατόμων πρέπει να μεγιστοποιεί τα συνολικά έσοδα από τα αντίτιμα ανά αεροπορικό ταξίδι, ενώ θα πρέπει να τηρείται ο περιορισμός της πεπερασμένης χωρητικότητας του αεροπλάνου. Τέλος, κάθε φόρτωμα αεροπλάνου αντιστοιχεί σε μια εφικτή ροή του δικτύου.

1.5 Αλγόριθμοι τύπου Simplex για το ΠΡΕΚ

Στο Γραμμικό Προγραμματισμό ο αλγόριθμος Simplex υπολογίζει εφικτές βασικές λύσεις. Όταν λέμε ότι η λύση είναι εφικτή, εννοείται ότι είναι εφικτή για το πρωτεύων πρόβλημα, δηλαδή ισχύει $x \geq 0$. Το σύνολο των εφικτών λύσεων για το γραμμικό πρόβλημα αποτελεί ένα πολύεδρο. Η βασική λύση αντιστοιχεί σε κορυφή αυτού του πολυέδρου της εφικτής περιοχής. Οι αλγόριθμοι τύπου Simplex βρίσκουν βασικές λύσεις x και (w, s), οι οποίες είναι συμπληρωματικές.

Αντίστοιχα, η βασική λύση για το ΠΡΕΚ, με όρους δικτύων, αναπαριστάνεται χρησιμοποιώντας ένα δένδρο. Αξίζει να αναφερθεί, ότι για πρώτη φορά ο συσχετισμός μεταξύ βάσης (Γραμμική Βελτιστοποίηση) και δένδρου (Δικτυακή Βελτιστοποίηση), περιγράφτηκε στην εργασία (Edmonds 1965). Κάθε βασικό δένδρο θα δηλώνεται με T. Κάθε δένδρο - λύση ενός ΠΡΕΚ, είναι ένα διάνυσμα ροής x, το οποίο αποτελεί μοναδική λύση των παραπάνω ισοτικών περιορισμών του προβλήματος (NP.1). Στο διάνυσμα x, το κάθε στοιχείο x_{ij} αποτελεί την ροή του τόξου (i, j), ικανοποιώντας τους περιορισμούς προσφοράς ή ζήτησης ποσότητας για κάθε κόμβο. Εάν το διάνυσμα ροής x ενός δένδρου T είναι μη αρνητικό, δηλαδή για όλα τα στοιχεία (ροές του κάθε τόξου) ισχύει ότι $x_{ij} \geq 0$, τότε η ροή x θα ονομάζεται εφικτή. Κάθε δένδρο, το οποίο καθορίζει εφικτή ροή, θα το ονομάζουμε εφικτό δένδρο. Από εδώ και πέρα αυτό το δένδρο θα το συμβολίζουμε ως T και τη ροή του ως $x(T)$.

Δεδομένου ενός στιγμιότυπου ΠΡΕΚ και ενός εφικτού δένδρου T για αυτό, ένα δένδρο T' θα ονομάζεται διαδοχικό του T, αν το T' προκύπτει απο το T με την προσθήκη ενός τόξου $(g, h) \notin T$ και την διαγραφή ενός ήδη υπάρχοντος $(k, l) \in T$. Κάθε τέτοια επανάληψη ή αλλιώς διαδοχή δένδρων, την ονομάζουμε *περιστροφή* (pivot). Μια ακολουθία Simplex είναι μια ακολουθία δένδρων $T^1, T^2, ..., T^n$, τέτοια ώστε $\forall i$, το δένδρο T^{i+1} να είναι διαδοχικό του T^i και επιπλέον όλα τα διαδοχικά δένδρα να είναι εφικτά. Ο πιο γνωστός αλγόριθμος για το ΠΡΕΚ στη διεθνή βιβλιογραφία είναι ο πρωτεύων αλγόριθμος Simplex για δίκτυα και περιγράφεται για πρώτη φορά στην εργασία

(Dantzig 1951).

Ένα κοινό γνώρισμα όλων των αλγορίθμων τύπου Simplex είναι ότι μπορούν να περιγραφούν ως μια μέθοδο που ακολουθεί μονοπάτια τύπου Simplex, τα οποία οδηγούν στη βέλτιστη λύση. Στη συνέχεια, θα δοθεί μια σύντομη περιγραφή δύο αλγορίθμων τύπου Simplex για το ΠΡΕΚ. Στην Υποενότητα 1.5.1 παρατίθεται ο πρωτεύων αλγόριθμος τύπου Simplex για το ΠΡΕΚ, ενώ στην Υποενότητα 1.5.2 παρατίθεται ο δυϊκός αλγόριθμος τύπου Simplex για το ΠΡΕΚ. Πρέπει εδώ να αναφερθεί, ότι επιλέχθηκε η περιγραφή αυτών των δύο αλγορίθμων, διότι είναι οι περισσότερο γνωστοί αλγόριθμοι τύπου Simplex για το ΠΡΕΚ, αλλά και γιατί ο προτεινόμενος αλγόριθμος NEPSA συνδυάζει κάποια στοιχεία και των δύο αυτών αλγορίθμων.

1.5.1 Ο πρωτεύων αλγόριθμος τύπου Simplex για το ΠΡΕΚ

Ο πρωτεύων αλγόριθμος τύπου Simplex για το ΠΡΕΚ, ή (Network Primal Simplex Algorithm), από εδώ και πέρα θα ονομάζεται χάριν συντομίας NPSA. Ο NPSA είναι επαναληπτικός αλγόριθμος και ξεκινάει απο ένα εφικτό δένδρο. Στη συνέχεια κατασκευάζει μια ροή βασική και πάντοτε εφικτή.

Στο *Βήμα 0* κατά την αρχικοποίηση, κατασκευάζεται ένα εφικτό δένδρο με τη μέθοδο του μεγάλου M. Αυτή η μέθοδος παρουσιάζεται αναλυτικά στην Ενότητα 2.2. Το σύνολο τόξων A διαμερίζεται σε δύο υποσύνολα. Το πρώτο υποσύνολο απαρτίζεται από τα βασικά τόξα που ανήκουν στο T, ενώ το δεύτερο υποσύνολο αντιστοιχεί στα μη βασικά τόξα. Για λόγους διευκόλυνσης, θα συμβολίζουμε με το ίδιο γράμμα T το σύνολο των τόξων που απαρτίζουν το δένδρο T. Σε κάθε επανάληψη πρώτα επιλέγεται το εισερχόμενο και έπειτα το εξερχόμενο τόξο.

Ορισμός: Το εισερχόμενο τόξο θα το συμβολίζουμε απο εδώ και πέρα ως (g, h).

Ως εισερχόμενο τόξο επιλέγεται ένα μη βασικό τόξο (g, h), το οποίο είναι δυϊκά μη εφικτό. Στην περίπτωση, όπου υπάρχουν πολλά τόξα δυϊκά μη εφικτά, η επιλογή μπορεί να γίνει με διάφορους

κανόνες. Ο πιο γνωστός κανόνας, είναι ο παρακάτω κανόνας του Dantzig. Το εισερχόμενο τόξο (g, h), επιλέγεται έτσι ώστε:

$$(g,h) : s_{gh} = min\{s_{ij} : (i,j) \notin T \ \& \ s_{ij} < 0\} \tag{2}$$

Με την είσοδο ενός μη βασικού τόξου στο δένδρο, δημιουργείται ένας μοναδικός κύκλος C. Θα ονομάζουμε C^k τον κύκλο που σχηματίζεται, κατά την k επανάληψη, όταν εισάγεται το τόξο (g, h).

Ως γνωστό, κάθε μη βασικό τόξο, π.χ. (g, h), μπορεί να αναπαρασταθεί με τη βοήθεια των βασικών τόξων (i, j). Αυτή η αναπαράσταση γίνεται με τη βοήθεια ενός διανύσματος - στήλης h_{gh}. Για κάθε μη βασικό τόξο υπάρχει και μια αντίστοιχη στήλη h_{gh}. Το κάθε στοιχείο αυτής της στήλης αντιστοιχεί σε κάποιο βασικό τόξο. Τα στοιχεία αυτής της στήλης συμπληρώνονται ως εξής:

$h_{gh}(i, j) = -1$, ανν το τόξο (i, j) έχει τον ίδιο προσανατολισμό με το (g, h) στον κύκλο C^t

$$h_{gh}(i,j) = 0, (i,j) \notin C^{(t)}. \tag{3}$$

$h_{gh}(i, j) = 1$ ανν το τόξο (i, j) έχει τον αντίθετο προσανατολισμό με το (g, h) στον κύκλο C^t

Οπότε, όσα βασικά τόξα δεν συμμετέχουν στον κύκλο που δημιουργείται με την είσοδο του συγκεκριμένου τόξου, θα έχουν το αντίστοιχο στοιχείο ίσο με 0. Όσα βασικά τόξα (i, j) συμμετέχουν στον κύκλο και έχουν τον ίδιο προσανατολισμό με το εισερχόμενο, θα έχουν το αντίστοιχο στοιχείο της στήλης $h_{gh}(i, j) = -1$. Τέλος, όσα βασικά τόξα (i, j) συμμετέχουν στον κύκλο και έχουν αντίθετο προσανατολισμό με το εισερχόμενο, θα έχουν το αντίστοιχο στοιχείο της στήλης $h_{gh}(i, j) = 1$. Πιο αναλυτικά, αυτή η μέθοδος υπολογισμού περιγράφεται στο βιβλίο (Bazaraa, Jarvis, and Sherali 2005).

Τα διανύσματα h στη γεωμετρία του γραμμικού προβλήματος αποτελούν κατευθύνσεις προς τις

οποίες θα κινηθούμε, αν επιλέξουμε ως εισερχόμενη την εκάστοτε μη βασική μεταβλητή. Αντίστοιχα στο ΠΡΕΚ τα διανύσματα h, αποτελούν μία ένδειξη του πως θα μεταβληθεί η ροή σε κάποιο κύκλο του δικτύου, εάν επιλεχθεί ως εισερχόμενο το εκάστοτε τόξο. Κάθε μη βασικό τόξο εκφράζεται με τη βοήθεια των τόξων του βασικού δένδρου T, σύμφωνα με τη στήλη h. Εάν για κάποια μη βασικά τόξα, έστω (i, j), ισχύει ότι $\sum h_{ij} = 0 \Rightarrow$, τότε αυτά τα τόξα είναι γραμμικώς εξαρτημένα και επομένως σχηματίζουν κύκλο.

Στη συνέχεια παρουσιάζεται ο ψευδοκώδικας του αλγορίθμου NPSA.

Algorithm 1 Αλγόριθμος NPSA

Require: $G = (N, A), b, c, T$

procedure NPSA(G, T)

Βήμα 0 (Αρχικοποίηση)

  Compute x, w and s

  Compute vectors h_{ij}, using Relation 3

Βήμα 1 (Έλεγχος βελτιστότητας)

  while $\exists (i, j) \notin T : \ s_{ij} < 0$ **do**

Βήμα 2 (Επιλογή εισερχομένου τόξου)

    Choose the entering arc (g, h), using Relation 2

    if $C^- = \emptyset$ **then**

      STOP. The problem (NP.1) is unbounded.

    else

Βήμα 3 (επιλογή εξερχομένου τόξου)

      Choose the leaving arc (k, l), using Relation 6

    end if

Βήμα 4 (Περιστροφή)

    Set $T = T \sim (k, l) \cup (g, h)$

    Update x, w and s

  end while

  STOP. The problem (NP.1) is optimal.

end procedure

Ορισμός: Θα συμβολίζουμε με C^+ και C^- τα παρακάτω σύνολα:

$$C^+ = \{(i,j) \in C : h_{gh}(i,j) = -1\} \tag{4}$$

$$C^- = \{(i,j) \in C : h_{gh}(i,j) = 1\} \tag{5}$$

Ορισμός: Το εξερχόμενο τόξο θα το συμβολίζουμε απο εδώ και πέρα ως (k,l).

Ο αλγόριθμος NPSA επιλέγει ως εξερχόμενο τόξο (k,l) ένα βασικό τόξο, το οποίο πληρεί δύο προϋποθέσεις. Πρώτον, το τόξο (k,l) ανήκει στο σύνολο κύκλο C^- και δεύτερον, μεταξύ όλων των υποψηφίων εξερχομένων βασικών τόξων, επιλέγεται αυτό με τη μικρότερη ροή. Δηλαδή, το εξερχόμενο είναι τέτοιο ώστε:

$$(k,l) : x_{kl} = min\{x_{ij} : (i,j) \in C^-\} \tag{6}$$

Εάν ισχύει $s_{ij} \geq 0$, $\forall (i,j) \notin T$, τότε το τρέχων δένδρο αποτελεί βέλτιστη λύση και ο αλγόριθμος NPSA σταματά.

1.5.2 Ο δυϊκός αλγόριθμος τύπου Simplex για το ΠΡΕΚ

Ο δυϊκός αλγόριθμος τύπου Simplex για το ΠΡΕΚ, ή (Network Dual Simplex Algorithm), από εδώ και πέρα θα ονομάζεται χάριν συντομίας NDSA. Ο NDSA είναι επαναληπτικός αλγόριθμος και ξεκινάει απο ένα δυϊκά εφικτό δένδρο. Στη συνέχεια κατασκευάζει μια ροή βασική και πάντοτε δυϊκά εφικτή.

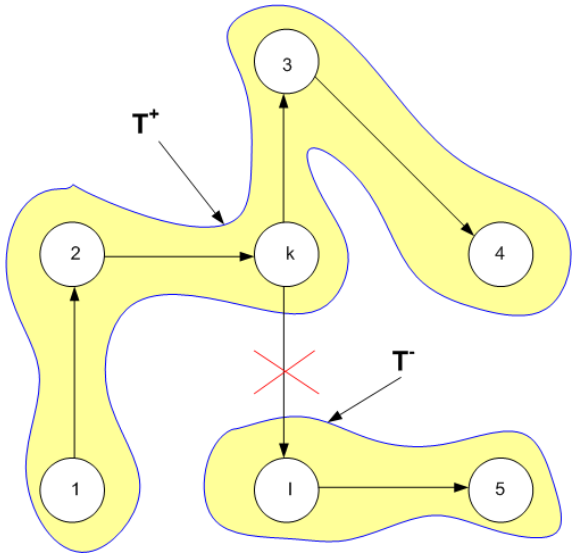
Στο *Βήμα 0* κατά την αρχικοποίηση, κατασκευάζεται ένα δυϊκά εφικτό δένδρο. Το σύνολο τόξων A διαμερίζεται σε δύο υποσύνολα. Το πρώτο υποσύνολο απαρτίζεται από τα βασικά τόξα

που ανήκουν στο T, ενώ το δεύτερο υποσύνολο αντιστοιχεί στα μη βασικά τόξα. Ομοίως, για λόγους διευκόλυνσης, θα συμβολίζουμε με το ίδιο γράμμα T το σύνολο των τόξων που απαρτίζουν το δένδρο T. Στον αλγόριθμο NDSA, σε κάθε επανάληψη πρώτα επιλέγεται το εξερχόμενο και έπειτα το εισερχόμενο τόξο.

Ο αλγόριθμος NDSA, σε αντίθεση με τον αλγόριθμο NPSA, επιλέγει ως εξερχόμενο τόξο ένα βασικό τόξο (i, j), το οποίο δεν είναι πρωτευόντως εφικτό, δηλαδή $x_{ij} < 0$. Συγκεκριμένα, το εξερχόμενο τόξο είναι τέτοιο ώστε:

$$(k, l) : x_{kl} = min\{x_{ij} : (i, j) \in T \ \& \ x_{ij} < 0\} \tag{7}$$

Μετά την απομάκρυνση του εξερχόμενου τόξου, ουσιαστικά το T θα κοπεί σε δύο υποδένδρα, όπως για παράδειγμα το δίκτυο του Σχήματος 1.16:



Σχήμα 1.16 Δημιουργία δύο υποδένδρων, μετά την αφαίρεση του εξερχομένου τόξου

Ορισμός: Το υποδένδρο που περιέχει τον κόμβο κεφαλή k του εξερχόμενου τόξου, θα το ονομάζουμε T^+. Το άλλο υποδένδρο που θα περιέχει τον κόμβο l θα το ονομάζουμε T^-.

Προφανώς, όταν εξέλθει το (k, l) απο το T, θα σταματήσει να ρέει ροή προς το T^-. Αυτό

σημαίνει ότι το T^+, θα αποκτήσει περίσσευμα ποσότητας, ίση με $\mid x_{kl} \mid$. Η περισσευούμενη ποσότητα *μένει* στον κόμβο k. Τότε, όλα τα μη βασικά τόξα (i, j) είτε δεν θα ενώνουν αυτά τα δύο υποδένδρα, είτε θα τα ενώνουν με δύο τρόπους, δηλαδή πρώτον ο κόμβος $i \in T^+$ ενώ ο κόμβος $j \in T^-$ και δεύτερον αντίστροφα.

Στη συνέχεια παρουσιάζεται ο ψευδοκώδικας του αλγορίθμου NDSA.

Algorithm 2 Αλγόριθμος NDSA

Require: $G = (N, A), b, c, T$

procedure NDSA(G, T)

Βήμα 0 (Αρχικοποίηση)

  Compute x, w and s

  Compute vectors h_{ij} using Relation 3

Βήμα 1 (Έλεγχος βελτιστότητας)

  while $\exists (i, j) \in T : \ x_{ij} < 0$ **do**

Βήμα 2 (Επιλογή εξερχομένου τόξου)

    Choose the leaving arc (k, l), using Relation 7

Βήμα 3 (επιλογή εισερχομένου τόξου)

    Choose the entering arc (g, h), using Relation 8

Βήμα 4 (Περιστροφή)

    Set $T = T \sim (k, l) \cup (g, h)$

    Update x, w and s

  end while

  STOP. The problem (NP.1) is optimal.

end procedure

Εάν ονομάζαμε C^k τον κύκλο που θα σχηματίζονταν, κατά την k επανάληψη, εάν προτού εξέλθει το τόξο (k, l), εισάγονταν το τόξο (g, h), τότε στο κύκλο C που θα είχε δημιουργηθεί, το τόξο (k, l) θα έχει τον ίδιο προσανατολισμό με το εισερχόμενο τόξο (g, h), δηλαδή θα ισχύει $h_{gh}(k, l) = -1$.

Ως εισερχόμενο τόξο επιλέγεται ένα μη βασικό τόξο (g, h), το οποίο πληρεί δύο προϋποθέσεις. Πρώτον, θα ενώνει τα δύο υποδένδρα με τον αντίθετο τρόπο από ότι το εξερχόμενο τόξο. Με άλλα

λόγια, θα ισχύει ότι $g \in T^-$ ενώ ο κόμβος $h \in T^+$. Δεύτερον, μεταξύ όλων των υποψηφίων εισερχομένων μη βασικών τόξων, επιλέγεται αυτό με το μικρότερο ελλειματικό κόστος. Συγκεκριμένα, το εισερχόμενο τόξο είναι τέτοιο ώστε:

$$(g,h) : s_{gh} = min\{s_{ij} : (i,j) \notin T \ \& \ h_{gh}(k,l) = -1\} \tag{8}$$

Εάν ισχύει $x_{ij} \geq 0$, $\forall (i,j) \in T$, τότε το τρέχων δένδρο αποτελεί βέλτιστη λύση και ο αλγόριθμος NDSA σταματά.

1.6 Πολυωνυμικοί αλγόριθμοι για το ΠΡΕΚ

Ο πρώτος πολυωνυμικός αλγόριθμος, (weakly polynomial-time), με χρήση τεχνικών κλιμάκωσης ή αλλιώς scaling περιγράφεται στην εργασία (Edmonds and Karp 1972). Το ανοιχτό ερευνητικό πρόβλημα εκείνης της εποχής, για την ύπαρξη ή όχι ενός ισχυρά πολυωνυμικού αλγορίθμου για το ΠΡΕΚ απαντήθηκε το έτος 1985. Ο πρώτος ισχυρά πολυωνυμικός αλγόριθμος περιγράφεται στην εργασία (Tardos 1985). Από τότε έχουν ανακαλυφθεί και άλλοι ισχυρά πολυωνυμικοί αλγόριθμοι όπως παράδειγμα οι (Galil and Tardos 1988) και (Ervolina and McCormick 1993). Ο δε πρώτος πολυωνυμικός πρωτεύων αλγόριθμος τύπου Simplex περιγράφεται στην εργασία (Orlin 1997). Ο αριθμός των περιστροφών που απαιτούνται γι΄ αυτόν τον αλγόριθμο είναι $O(min\{nmlognC, nm^2logn\})$, όπου C δηλώνει τη μέγιστη, κατά απόλυτη τιμή, τιμή κόστους ενός τόξου, εάν όλες οι τιμές κόστους είναι ακέραιοι αριθμοί και ∞ σε διαφορετική περίπτωση.

Έχουν ανακαλυφθεί από τότε και άλλοι δυϊκοί πολυωνυμικοί αλγόριθμοι όπως για παράδειγμα οι (Orlin, Plotkin, and Tardos 1993) και (Armstrong and Jin 1997). Για την επίλυση του ΠΡΕΚ υπάρχουν και άλλες προσεγγίσεις, όπως για παράδειγμα ο cycle cancelling αλγόριθμος (Sokkalingam, Ahuja, and Orlin 2000), ο Out-of-kilter αλγόριθμος (Fulkerson 1961) ή και μέθοδοι Simplex

που κάνουν χρήση της πιο κατηφορικής ακμής (steepest edge method) στις εργασίες (Curet 1997) και (Curet 1998). Τύπου Simplex ονομάζονται οι αλγόριθμοι που εκτελούν περιστροφές και αλλάζουν βάση ανά επανάληψη. Τέλος, υπάρχουν και άλλοι πρωτεύοντες, μη μονοτονικοί αλγόριθμοι Simplex όπως οι (Goldfarb and Hao 1992), (Tarjan 1991). Για την ακρίβεια, αυτού του είδους οι αλγόριθμοι δεν ελαττώνουν απαραίτητα την τιμή της αντικειμενικής συνάρτησης ανά επανάληψη, και ως εκ τούτου δεν μπορούν να χαρακτηριστούν πρωτεύοντες αλγόριθμοι τύπου Simplex με την ευρεία έννοια.

Σήμερα, οι ερευνητές αναγνωρίζουν ότι η χρήση τεχνικών κλιμάκωσης έχουν μεγάλη, τόσο θεωρητική αξία, όσο και πρακτική σημασία. Άλλοι πολυωνυμικοί αλγόριθμοι, που κάνουν χρήση τεχνικών κλιμάκωσης είναι οι παρακάτω. Ένας αλγόριθμος, ο οποίος κάνει χρήση τεχνικών κλιμάκωσης κόστους, (cost scaling technique) περιγράφτηκε για πρώτη φορά στην εργασία (Rock 1980). Ακόμη, υπάρχει ο αλγόριθμος που περιγράφεται στην εργασία (Orlin 1993). Ο συγκεκριμένος αλγόριθμος κάνει χρήση τεχνικών κλιμάκωσης χωρητικότητας, (capacity scaling technique). Επιπλέον, έχουν αναπτυχθεί και αλγόριθμοι οι οποίοι κάνουν χρήση τεχνικών διπλής κλιμάκωσης, (double scaling technique). Οι αλγόριθμοι αυτοί, κάνουν χρήση ταυτόχρονα τεχνικών κλιμάκωσης τόσο του κόστους, όσο και της χωρητικότητας. Ένας τέτοιος αλγόριθμος, περιγράφεται στην εργασία (Ahuja, Goldberg, Orlin, and Tarjan 1992).

Άλλες προσεγγίσεις, είναι ο αλγόριθμος (minimum mean cycle - cancelling), ο οποίος περιγράφεται στην εργασία (Goldberg and Tarjan 1989). Προσπάθειες των ερευνητών, να βελτιώσουν την πολυπλοκότητα της χειρότερης περίπτωσης, οδήγησαν στην ανάπτυξη του αλγορίθμου επαναλαμβανόμενης κλιμάκωσης της χωρητικότητας, (repeated capacity scaling technique), ο οποίος περιγράφεται στην εργασία (Goldberg, Tardos, and Tarjan 1989). Περαιτέρω έρευνα προς αυτές τις ιδέες, οδήγησε λίγο αργότερα στην ανάπτυξη αλγορίθμων που κάνουν χρήση των τεχνικών (enchanced capacity scaling technique). Ένας τέτοιος αλγόριθμος, περιγράφεται στην εργασία

(Plotkin and Tardos 1990). Τέλος, έχουν αναπτυχθεί και άλλοι πολυωνυμικοί αλγόριθμοι που κάνουν χρήση τριπλής κλιμάκωσης, (triple scaling technique), όπως για παράδειγμα ο αλγόριθμος (Gabow and Tarjan 1989).

Παρακάτω στον Πίνακα **1.1**, παρατίθενται οι χρόνοι εκτέλεσης των μερικών πολυωνυμικών αλγορίθμων για το ΠΡΕΚ. Η μεταβλητή n δηλώνει τον αριθμό των κόμβων, ενώ το πλήθος των τόξων δηλώνεται με m. Επιπλέον, με C δηλώνεται η μεγαλύτερη τιμή κόστους του ΠΡΕΚ και με U δηλώνεται είτε η μεγαλύτερη τιμή προσφοράς ή ζήτησης, είτε η μέγιστη τιμώ άνω χωρητικότητας για κάποιο τόξο του ΠΡΕΚ.

Αλγόριθμος	**Χρόνος εκτέλεσης (running time)**
Capacity scaling algorithm	$O((m\ log\ U)(m\ +\ n\ log\ n))$
Cost scaling algorithm	$O(n^3\ log(nC))$
Double scaling algorithm	$O(nm\ log\ U\ log\ U\ log(nC))$
Minimum mean cycle-cancelling algorithm	$O(n^2m^3\ log\ n)$
Repeated capacity scaling algorithm	$O((m^2\ log\ n)(m\ +\ n\ log\ n))$
Enchanced capacity scaling algorithm	$O((m\ log\ n)(m\ +\ n\ log\ n))$

Πίνακας **1.1** Χρόνοι εκτέλεσης πολυωνυμικών αλγορίθμων για το ΠΡΕΚ

Μερικές απο τις πιο αποτελεσματικές προγραμματιστικές υλοποιήσεις μέχρι τώρα είναι ο κώδικας RELAX (Bertsekas and Tseng 1988), ο οποίος έχει φτάσει στην έκδοση RELAX-IV (Bertsekas and Tseng 1994), ο λύτης NETOPT της εμπορικής εφαρμογής CPLEX (ILOG), ο κώδικας NETFLO (Kennington and Helgason 1980), ο κώδικας "cs2" (Goldberg 1997) που κάνει χρήση τεχνικών κλιμάκωσης, καθώς και ο κώδικας RNET (Grigoriadis 1984). Τις τελευταίες δεκαετίες, έχει διεξαχθεί μια πληθώρα από συγκριτικές υπολογιστικές μελέτες μεταξύ διαφόρων υλοποιήσεων, όπως για παράδειγμα η μελέτη η οποία περιγράφεται στην ερευνητική εργασία (Bland, Cheriyan,

Jensen, and Ladanyi 1993).

Λόγω της στενής σχέσης των ερευνητικών πεδίων της Γραμμικής Βελτιστοποίησης με τη Δικτυακή Βελτιστοποίηση, πολλές αλγοριθμικές ιδέες από το χώρο του Γραμμικού Προγραμματισμού έχουν εφαρμοσθεί με μεγάλη επιτυχία και στο χώρο της Δικτυακής Βελτιστοποίησης. Παράδειγματος χάρην, ο πολυωνυμικός αλγόριθμος εσωτερικών σημείων, ο οποίος παρουσιάστηκε για πρώτη φορά στην εργασία (Karmarkar 1984), έχει εμπνεύσει πληθώρα ερευνητών, ώστε να αναπτύξουν με μεγάλη επιτυχία αλγόριθμους εσωτερικών σημείων για προβλήματα δικτύων. Χαρακτηριστικά αναφέρονται οι εργασίες (Resende and Veiga 1993), (Portugal, Resende, Veiga, and Judice 2000) και (Resende and Veiga 2003). Η ονομασία *εσωτερικών σημείων* προέρχεται από τη γεωμετρία του Γραμμικού Προβλήματος, διότι οι αλγόριθμοι εσωτερικών σημείων ακολουθούν μονοπάτια στο εσωτερικό της εφικτής περιοχής και όχι στο σύνορο, της όπως ο κλασικός αλγόριθμος Simplex.

1.7 Στόχοι, καινοτόμα στοιχεία και δομή της Διατριβής

Η παρούσα Διατριβή υπάγεται στον επιστημονικό χώρο της Δικτυακής Βελτιστοποίησης. Το κύριο ερευνητικό αποτέλεσμα στο οποίο στηρίζεται η πρωτοτυπία αυτής της διατριβής είναι η ανάπτυξη ενός νέου αλγορίθμου για την επίλυση Προβλημάτων Ροής Ελαχίστου Κόστους. Ο προτεινόμενος αλγόριθμος είναι τύπου Simplex εξωτερικών σημείων. Τον αλγόριθμο που θα περιγραφεί στη συνέχεια, από εδώ και πέρα θα τον ονομάζουμε για λόγους συντομίας NEPSA, (Network Exterior Point Simplex Algorithm) και σε αυτήν την περίπτωση, η ονομασία *εξωτερικών σημείων* προέρχεται από το χώρο του Γραμμικού Προβλήματος, διότι οι αλγόριθμοι εξωτερικών σημείων ακολουθούν δύο μονοπάτια για τον εντοπισμό της βέλτιστης λύσης. Το ένα μονοπάτι είναι εφικτό, ενώ το άλλο όχι. Δηλαδή, το μη εφικτό μονοπάτι αποτελείται από κορυφές εξωτερικές της εφικτής περιοχής στο εξωτερικό της εφικτής περιοχής.

Υλοποίηση αλγορίθμου εξωτερικών σημείων σε δίκτυα μέχρι σήμερα, υπάρχει μόνο για το πρό-

βλημα της ανάθεσης (Paparrizos 1991) και το πρόβλημα της μεταφοράς (Paparrizos 1996b). Η απόδειξη ορθότητας του πρώτου αλγορίθμου περιγράφεται στην εργασία (Paparrizos 1996a). Υπολογιστικές μελέτες με αλγορίθμους εξωτερικών σημείων για τα προβλήματα της ανάθεσης και της μεταφοράς, περιγράφονται στις εργασίες (Papamanthou, Paparrizos, Samaras, Sifaleras, and Stergiou 2006) και (Papamanthou, Paparrizos, and Samaras 2004) αντιστοίχως. Όσον αφορά το γενικό γραμμικό πρόβλημα, προηγούμενες υπολογιστικές μελέτες της ίδιας κατηγορίας αλγορίθμων σε τυχαία, τόσο αραιά όσο και πυκνά, γραμμικά προβλήματα, έχουν δείξει ότι αυτή η κατηγορία αλγορίθμων είναι μέχρι και δέκα φορές πιο γρήγορη (Paparrizos, Samaras, and Stephanides 2003), απ΄ ότι ο αναθεωρημένος πρωτεύων αλγόριθμος Simplex, όσον αφορά το χρόνο της CPU. Αυτή η υπολογιστική βελτίωση οφείλεται στο γεγονός της σημαντικής ελάττωσης των επαναλήψεων.

Αποτελεί ανοιχτό πρόβλημα η γενίκευση αυτής της κατηγορίας των αλγόριθμων και σε άλλου είδους προβλήματα όσο δε και η μελέτη της υπολογιστικής αποτελεσματικότητας τους. Η μεταφορά λοιπόν αυτού του αλγορίθμου στο ΠΡΕΚ, έτσι ώστε να περιγράφεται αποκλειστικά με έννοιες δικτύων, αλλά και να αξιοποιούνται κατάλληλες δομές δεδομένων παρουσιάζει σημαντικό ενδιαφέρον. Σε αυτή τη διατριβή παρουσιάζεται για πρώτη φορά ένας πρωτεύων αλγόριθμος τύπου Simplex εξωτερικών σημείων για το ΠΡΕΚ. Πρέπει πάντως να σημειωθεί, ότι συνεχίζεται η έρευνα επάνω σε αλγορίθους εξωτερικών σημείων για προβλήματα ροής ελαχίστου κόστους. Ήδη καταβάλλεται προσπάθεια για την ανάπτυξη ενός δυϊκού αλγορίθμου τύπου Simplex, εξωτερικών σημείων για το ΠΡΕΚ, (Geranis 2007).

Η δομή της παρούσας Διατριβής έχει ως εξής. Το κεφάλαιο 1 αποτελεί ουσιαστικά μια εισαγωγή. Περιγράφονται σε αυτό κάποιοι βασικοί ορισμοί, η ορολογία που θα χρησιμοποιηθεί εκτενώς στα επόμενα κεφάλαια, καθώς και αναλύονται οι αλγόριθμοι του τύπου Simplex. Επιπλέον, αναφέρονται κάποιοι πολυωνυμικοί αλγόριθμοι, όχι απαραίτητα τύπου Simplex και δίνεται η πολυπλοκότητα χειρότερης περίπτωσης τους. Γίνεται επίσης αναφορά και στην κατηγορία των προβλημάτων στα οποία

εφαρμόζεται ο προτεινόμενος αλγόριθμος. Στο κεφάλαιο 2 ακολουθεί η βηματική περιγραφή του αλγορίθμου NEPSA συνοδευόμενη από κατάλληλα παραδείγματα, τόσο βέλτιστου προβλήματος, όσο δε και απεριόριστου προβλήματος. Στο κεφάλαιο 3 παρουσιάζονται οι μαθηματικές αποδείξεις των θεωρημάτων που διασφαλίζουν την ορθότητα του αλγορίθμου. Στο κεφάλαιο 4 περιγράφεται η υλοποίηση του NEPSA στη γλώσσα προγραμματισμού Fortran 95/2003 και αναφέρονται αναλυτικά οι δομές που χρησιμοποιήθηκαν. Ακόμη, γίνεται αναφορά και σε γεννήτριες προβλημάτων Δικτυακής Βελτιστοποίησης, οι οποίες κατασκευάζουν τυχαία, αραιά ή και πυκνά, ΠΡΕΚ και γενικώς είναι απαραίτητες στη διεξαγωγή υπολογιστικών μελετών. Στο ίδιο κεφάλαιο γίνεται αναφορά και στη χρήση μιας μεθόδου, για την αντιμετώπιση εκφυλισμένων προβλημάτων.

Έπειτα, στο κεφάλαιο 5 παρουσιάζεται ένα ολοκληρωμένο λογισμικό επίλυσης προβλημάτων Δικτυακής Βελτιστοποίησης, διαθέσιμο για εξ΄ αποστάσεως χρήση μέσω Διαδικτύου, που αναπτύχθηκε κυρίως για εκπαιδευτικούς σκοπούς. Περιγράφεται η επίλυση προβλημάτων με τη χρήση του προτεινόμενου λογισμικού, με τη βοήθεια αναλυτικών παραδειγμάτων. Ολοκληρώνοντας, στο κεφάλαιο 6 παρουσιάζονται τα αποτελέσματα της παρούσας έρευνας και αναφέρονται κατευθύνσεις για μελλοντική έρευνα.

ΚΕΦΑΛΑΙΟ 2

Ο αλγόριθμος εξωτερικών σημείων NEPSA για το ΠΡΕΚ

2.1 Περιγραφή του αλγορίθμου

Ο NEPSA παρουσιάστηκε για πρώτη φορά από τον Σιφαλέρα Α. στο 7ο Βαλκανικό Συνέδριο Επιχειρησιακής Έρευνας στη Ρουμανία το 2005, με την εργασία (Karagiannis, Paparrizos, Samaras, and Sifaleras 2005). Ο αλγόριθμος NEPSA έχει διαφορετικούς κανόνες περιστροφής σε σχέση με τον κλασικό αλγορίθμο Simplex για δίκτυα. Στο Κεφάλαιο 3 παρουσιάζονται αναλυτικά οι αποδείξεις ορθότητας του NEPSA. Ο NEPSA είναι μεν επαναληπτικός αλγόριθμος και ξεκινάει απο ένα εφικτό δένδρο, όπως ο NPSA, αλλά διαφοροποιείται στο γεγονός ότι κατασκευάζει δύο ροές. Η μία ροή είναι βασική, αλλά όχι πάντοτε εφικτή, ενώ η άλλη είναι εφικτή, αλλά όχι πάντοτε βασική.

Στο *Βήμα 0* κατά την αρχικοποίηση, ένα εφικτό δένδρο κατασκευάζεται με τη μέθοδο του μεγάλου M. Το σύνολο τόξων A διαμερίζεται σε δύο υποσύνολα. Το πρώτο υποσύνολο απαρτίζεται από τα βασικά τόξα που ανήκουν στο T, ενώ το δεύτερο υποσύνολο αντιστοιχεί στα μη βασικά τόξα. Το τελευταίο υποσύνολο, διαμερίζεται επιπλέον σε άλλα δύο υποσύνολα P και Q ως εξής:

$$P = \{(i,j) : s_{ij} < 0\}, \forall\ (i,j)\ \notin T \tag{1}$$

$$Q = \{(i,j) : s_{ij} \geq 0\}, \forall\ (i,j)\ \notin T \tag{2}$$

Για λόγους διευκόλυνσης, θα συμβολίζουμε με το ίδιο γράμμα T το σύνολο των τόξων που απαρτίζουν το δένδρο T. Με αυτόν τον τρόπο, το αρχικό σύνολο των τόξων A τελικά θα διαμερίζεται ως: $A = [T\ \ P\ \ Q]$.

Λόγω της φύσεως του συγκεκριμένου αλγορίθμου, (αλγόριθμος εξωτερικών σημείων για το γραμμικό πρόβλημα), πρέπει οπωσδήποτε να υπάρχει επαφή του εκάστοτε δένδρου με αρνητική ροή, με την εφικτή περιοχή - σύνολο των εφικτών δένδρων. Διαφορετικά, επάνοδος σε εφικτή ροή και επομένως στη βέλτιστη λύση του προβλήματος είναι πολύ δύσκολη, αν όχι ανέφικτη. Αυτή η επαφή διευκολύνεται με μια εφικτή ροή. Αυτή η εφικτή ροή, (δρόμος κατά το γραμμικό προγραμματισμό) όμως δεν είναι τύπου Simplex. Αποτελείται από τόξα με μη αρνητική ροή τα οποία δεν σχηματίζουν δένδρο απαραίτητα, αλλά δίκτυο γενικότερα. Από εδώ και πέρα το διάνυσμα αυτής της εφικτής ροής θα το συμβολίζουμε ως y. Το ρόλο της *κατεύθυνσης* στο γραμμικό πρόβλημα, στην περίπτωση του αλγορίθμου για το ΠΡΕΚ, τον επιτελεί ένα διάνυσμα αθροίσματος συγκεκριμένων ροών μη βασικών τόξων, (από εδώ και πέρα θα αναφερόμαστε σε αυτό ως $d(T)$. Στον προγραμματισμό του αλγορίθμου δεν είναι αναγκαίο να διατηρείται η εφικτή ροή y, παρά μόνο η τρέχουσα λύση $x(T)$, καθώς φυσικά να ανανεώνεται και η *κατεύθυνση - ροή*. Παρ΄ όλα αυτά, οι αποδείξεις θα στηριχθούν στην ύπαρξη και των δύο δρόμων. Ο αλγόριθμος NEPSA είναι δύο δρόμων, υπολογίζει με άλλα λόγια αυτές τις δύο ροές. Στη βιβλιογραφία συναντώνται και άλλοι αλγόριθμοι ιδίου τύπου, δηλαδή δύο δρόμων, όπως για παράδειγμα στις εργασίες (Dosios, Paparrizos, Samaras, and Sifaleras 2003a) και (Badr, Paparrizos, Samaras, and Sifaleras 2005), για το γενικό γραμμικό πρόβλημα

του Μαθηματικού Προγραμματισμού.

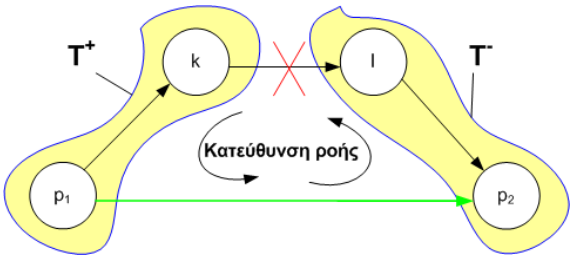
Σε αντίθεση με τον αλγόριθμο NPSA και παρόμοια με τον αλγόριθμο NDSA, στον αλγόριθμο NEPSA σε κάθε επανάληψη πρώτα επιλέγεται το εξερχόμενο τόξο (k, l) και έπειτα το εισερχόμενο τόξο (g, h).

Το εξερχόμενο τόξο (k, l) επιλέγεται σύμφωνα με έναν έλεγχο ελαχίστου λόγου, βλέπε Σχέση 6, ο οποίος θα περιγραφεί στη συνέχεια αναλυτικά. Τότε ουσιαστικά το T θα κοπεί σε δύο υποδένδρα (παρόμοια με τον αλγόριθμο NDSA, όπως στο δίκτυο του Σχήματος 1.16:

Όταν έχουμε περιστροφή τύπου Α, το εισερχόμενο τόξο θα το συμβολίζουμε και ως (p_1, p_2). Ενώ, αν γνωρίζουμε ότι έχουμε περιστροφή τύπου Β, το εισερχόμενο τόξο, θα το συμβολίζουμε και ως (q_1, q_2)

Τώρα πια, θα ονομάζουμε C^k τον κύκλο που θα σχηματίζονταν, κατά την k επανάληψη, εάν προτού εξέλθει το τόξο (k, l), εισάγονταν το τόξο (g, h). Επιπλέον δε, η αναπαράσταση κάθε μη βασικού τόξου, π.χ. (g, h), με τη βοήθεια των βασικών τόξων γίνεται με τη βοήθεια του διανύσματος - στήλης h_{gh}, όπως και στην περίπτωση του αλγορίθμου NPSA.

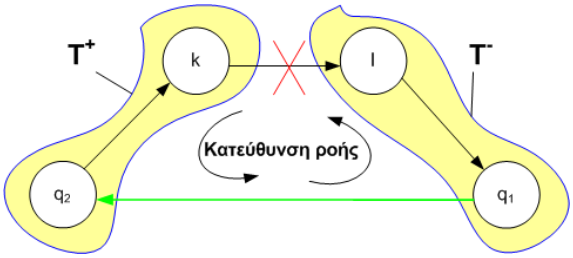
Περιστροφή τύπου Α: Απο τον κόμβο k η *εγκλωβισμένη* ποσότητα x_{kl} επιστρέφει προς τα πίσω για να διοχετευτεί μέσω του (p_1, p_2) από το T^+ στο T^-, βλέπε Σχήμα 2.1. Στον κύκλο που δημιουργείται το τόξο (p_1, p_2) έχει αντίθετο προσανατολισμό απο το (k, l). Άρα $h_{p_1p_2}(k, l) = 1 > 0$.



Σχήμα 2.1 Ομόρροπο εισερχόμενο τόξο

Περιστροφή τύπου Β: Απο τον κόμβο k η *εγκλωβισμένη* ποσότητα x_{kl} επιστρέφει προς τα

πίσω για να διοχετευτεί μέσω του (q_1, q_2) από το T^+ στο T^-, βλέπε Σχήμα 2.2. Στον κύκλο που δημιουργείται τό (q_1, q_2) έχει ίδιο προσανατολισμό με το (k, l). Άρα $h_{q_1 q_2}(k, l) = -1 < 0$. Προφανώς δεν μπορεί να διοχετευθεί ροή αφού το τόξο έχει αντίθετο προσανατολισμό, άρα κυλάει *αρνητική* ροή, δηλαδή κάποια ποσότητα προιόντων μένει στον κόμβο q_2.



Σχήμα 2.2 Αντίρροπο εισερχόμενο τόξο

Όταν μια μονάδα ρέει δια μέσου του κύκλου, η αντικειμενική τιμή μεταβάλλεται κατά την ποσότητα

$$\sum_{(i,j)\in C} t_{ij} c_{ij} \tag{3}$$

όπου

$$t_{ij} = 1, \text{ αν } (i,j) \in C^+$$

$$t_{ij} = -1, \text{ αν } (i,j \in C^-$$

Αντικαθιστώντας τα c_{ij}, όπου $(i,j) \neq (g,h)$, στο άθροισμα 3 βρίσκουμε ότι είναι:

$$\sum_{(i,j)\in C} t_{ij} c_{ij} = c_{gh} - w_g + w_h = s_{gh}$$

Αν είναι $P = \emptyset$, το πρώτο δένδρο αποτελεί τη βέλτιστη λύση και ο αλγόριθμος σταματά. Διαφορετικά υπολογίζεται η ροή d και η ποσότητα s_0, θέτοντας

$$s_0 = \sum_{(i,j)\in P} s_{ij} \tag{4}$$

και

$$d(T) = -\sum_{(i,j)\in P} h_{ij}, \quad d(P) = (1, 1, ..., 1), \quad d(Q) = 0 \tag{5}$$

Το διάνυσμα $d(T)$ δεν περιγράφει ακριβώς ροή, αλλά είναι ένα άθροισμα με τους σχετικούς προσανατολισμούς των μη βασικών τόξων σε σχέση με τα βασικά τόξα. Οι συνιστώσες του διανύσματος, αντιστοιχούν μονοσήμαντα σε κάθε βασικό τόξο. Θα αναφερόμαστε απο εδώ και πέρα στην εκάστοτε συνιστώσα, (π.χ. για το $(i, j) \in T$) με τον συμβολισμό $d_{ij}(T)$. Το διάνυσμα $d(T)$ στο γραμμικό πρόβλημα προκύπτει ως άθροισμα διανυσμάτων - κατευθύνσεων.

Το διάνυσμα *ροής* $d(T)$, χρησιμοποιείται για τον προσδιορισμό του εξερχόμενου τόξου (k, l) από το T. Σύμφωνα με τον έλεγχο ελαχίστου λόγου, Εξίσωση 6, υπολογίζουμε το λόγο a και επιλέγουμε το εξερχόμενο τόξο (k, l) :

$$a = \frac{x_{kl}}{-d_{kl}(T)} = min\{\frac{x_{ij}}{-d_{ij}(T)} : d_{ij}(T) < 0\} \tag{6}$$

Αν δεν υπάρχει εξερχόμενο τόξο, δηλαδή αν είναι $d(T) \geq 0$, το πρόβλημα είναι βέλτιστο ή απεριόριστο, ανάλογα με το αν είναι $P = \emptyset$ ή $P \neq \emptyset$ αντίστοιχα. Διαφορετικά, υπολογίζεται η βοηθητική ροή y, η οποία περιγράφεται απο το διάνυσμα ροής $y = x + ad$ και προχωρούμε στον προσδιορισμό του εισερχόμενου τόξου (g, h) για το T. Η επιλογή του γίνεται με ένα σύνθετο έλεγχο ελαχίστου λόγου, ο οποίος μπορεί να θεωρηθεί σαν γενίκευση του ελαχίστου λόγου του δυϊκού αλγορίθμου *Simplex* για ΠΡΕΚ. Πιο συγκεκριμένα, τίθεται

$$\theta_1 = -s_{p_1p_2} = min\{-s_{ij} : h_{ij}(k, l) = 1, (i, j) \in P\} \tag{7}$$

$$\theta_2 = s_{q_1q_2} = min\{s_{ij} : h_{ij}(k, l)) = -1, (i, j) \in Q\} \tag{8}$$

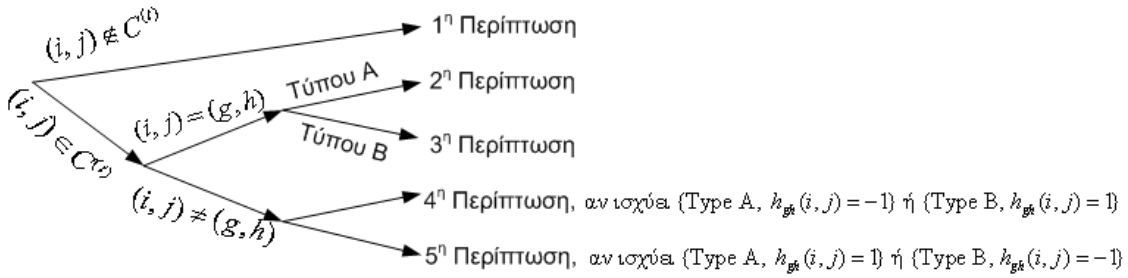
και εντοπίζονται τα τόξα (p_1, p_2) και (q_1, q_2). Αν είναι $\theta_1 \leq \theta_2$, τίθεται $(g, h) = (p_1, p_2)$. Διαφορετικά τίθεται $(g, h) = (q_1, q_2)$. Αν σε κάποια επανάληψη είναι $\theta_1 \leq \theta_2$, τότε η επανάληψη αυτή είναι τύπου Α, ειδάλλως είναι επανάληψη τύπου Β.

Όταν επιλεγούν το εξερχόμενο και εισερχόμενο τόξο, ανανεώνεται το δένδρο T και τα σύνολα P, Q. Η ανανέωσή τους γίνεται ως εξής. Τίθεται $T = T \sim (k, l) \cup (g, h)$ και $Q = Q \cup (k, l)$. Αν είναι $\theta_1 \leq \theta_2$, τίθεται $P = P \sim (g, h)$, διαφορετικά, τίθεται $Q = Q \sim (g, h)$.

Παρατηρούμε, ότι όταν έχουμε περιστροφή τύπου Α, τότε $h_{q_1 q_2}(k, l) = 1 > 0$, οπότε αφού επιλέχθηκε με τον έλεγχο ελαχίστου λόγου, $\Rightarrow (p_1, p_2) \in P$. Αντίστοιχα, όταν έχουμε περιστροφή τύπου Β, τότε $h_{q_1 q_2}(k, l) = -1 < 0$, οπότε αφού επιλέχθηκε με τον έλεγχο ελαχίστου λόγου, $\Rightarrow (q_1, q_2) \in Q$.

Από εδώ και πέρα θα γίνεται εκτενής αναφορά στις διάφορες επαναλήψεις του αλγορίθμου. Κάθε στοιχείο ή σύνολο στοιχείων κατά την i επανάληψη, θα συμβολίζεται με έναν (i) εκθέτη εντός παρενθέσεων. Για παράδειγμα το διάνυσμα ροής του δένδρου T κατά την 9 επανάληψη, θα γράφεται ως $x^{(9)}(T)$, κ.ο.κ. Στη συνέχεια, περιγράφεται αναλυτικά ο τρόπος ανανεώσης για τις ποσότητες $x(T)$, w_i και $s(T)$.

Για κάθε βασικό τόξο (i, j), οι αντίστοιχες μεταβλητές x_{ij}, d_{ij}, s_{ij}, ανανεώνονται ανάλογα με το συνδυασμό δυο παραμέτρων. Η πρώτη παράμετρος είναι ο τύπος της επανάληψης, ενώ η δεύτερη είναι ο προσανατολισμός του τόξου (i, j) σε σχέση με τον προσανατολισμό του εξερχόμενου τόξου (k, l). Κάποιοι απο αυτούς τους συνδυασμούς μπορούν να ομαδοποιηθούν, όπως στο Σχήμα 2.3, διότι οι αντίστοιχες μεταβλητές των τόξων, οι οποίες ανήκουν στις ίδια περίπτωση, ανανεώνονται παρόμοια. Για το λόγο αυτό, οι ακόλουθες περιπτώσεις - συνδυασμοί θα χρησιμοποιηθούν ώστε να περιγραφούν οι ανανεώσεις όλων των μεταβλητών που αντιστοιχούν σε κάθε τόξο. Η διερεύνηση των ίδιων περιπτώσεων θα χρησιμοποιηθεί επίσης και στην Ενότητα 3. με τις αποδείξεις ορθότητας του αλγορίθμου NEPSA.



Σχήμα 2.3 Πιθανοί συνδυασμοί για κάθε βασικό τόξο (i, j)

Αξίζει να αναφερθεί εδώ, ότι δεν είναι απαραίτητη η ανανέωση των στηλών h_{ij}, αλλά ούτε φυσικά ο εκ νέου υπολογισμός τους για όλα τα μη βασικά τόξα. Πιο συγκεκριμένα, σε κάθε επανάληψη αφού επιλέγεται το εξερχόμενο τόξο (k, l) θα υπολογίζεται το διάνυσμα γραμμή $h_{kl}(i, j), \forall (i, j) \notin T$, το οποίο αποτελείται από το προσανατολισμό όλων των μη βασικών τόξων σε σχέση με το εξερχόμενο. Στην ουσία, πρόκειται για τη γραμμή περιστροφής του κλασικού Simplex tableau για δίκτυα. Έπειτα, θα υπολογίζεται το διάνυσμα στήλη $h_{ij}(g, h), (i, j) \in T$, το οποίο αντιστοιχεί στο εισερχόμενο τόξο (g, h).

Παρακάτω, στον Πίνακα **2.1**, μπορεί κάποιος να δεί όλες τις ανανεώσεις των μεταβλητών, οι οποίες αντιστοιχούν στα βασικά τόξα, χρησιμοποιώντας την προηγούμενη κατηγοριοποίηση. Για τις στήλες h_{ij}, ας υποτεθεί ότι $(n_1, n_2) \notin T$ και $(i, j) \in T$. Για παράδειγμα, έστω ότι το τόξο $(i, j) \in T$ έχει βασική ροή ίση με $x_{ij}^{(t)}$. Εάν η t^{th} επανάληψη, είναι τύπου A, και το τόξο (i, j) έχει τον ίδιο προσανατολισμό με το εισερχόμενο τόξο (g, h) στον $C^{(t)}$. Τότε σύμφωνα με το Σχήμα 2.3 αυτό αντιστοιχεί στην 4η περίπτωση. Σε αυτή την περίπτωση λοιπόν, η ανανεωμένη βασική ροή του τόξου (i, j) στην $t+1$ επανάληψη, θα προκύψει σύμφωνα με την 4η γραμμή του Πίνακα **2.1**. Οπότε παίρνουμε ότι ισχύει $x_{ij}^{(t+1)} = x_{ij}^{(t)} + x_{kl}^{(t)}$.

Τέλος, τα ελλειματικά κόστη μπορούν να ανανεωθούν, χρησιμοποιώντας δύο διαφορετικούς τρόπους. Πρώτον ανανεώνοντας απευθείας τα s_{ij} και δεύτερον ανανεώνοντας τα w_i και έπειτα υπολογίζοντας τα s_{ij}. Ο πρώτος τρόπος φαίνεται παρακάτω:

Διερεύνηση περιπτώσεων	$x_{ij}^{(t+1)} =$	$d_{ij}^{(t+1)} =$
1η Περίπτωση	$x_{ij}^{(t)}$	$d_{ij}^{(t)}$
2η Περίπτωση	$x_{kl}^{(t)}$	$d_{kl}^{(t)} + 1$
3η Περίπτωση	$-x_{kl}^{(t)}$	$-d_{kl}^{(t)}$
4η Περίπτωση	$x_{ij}^{(t)} + x_{kl}^{(t)}$	$d_{ij}^{(t)} + d_{kl}^{(t)}$
5η Περίπτωση	$x_{ij}^{(t)} - x_{kl}^{(t)}$	$d_{ij}^{(t)} - d_{kl}^{(t)}$

Πίνακας **2.1** Ανανεώσεις τιμών βασικών τόξων

$$s_{ij}^{(t+1)} = \begin{cases} s_{ij}^{(t)} - s_{gh}^{(t)}, \ (i,g \in T^+ \wedge j,h \in T^-) \vee (i,g \in T^- \wedge j,h \in T^+) \\ s_{ij}^{(t)} + s_{gh}^{(t)}, \ (i,h \in T^+ \wedge j,g \in T^-) \vee (i,h \in T^- \wedge j,g \in T^+) \\ s_{ij}^{(t)}, \ (i,j \in T^+) \vee (i,j \in T^-) \end{cases}$$

2.2 Αρχικοποίηση του NEPSA

Ο NEPSA για να ξεκινήσει χρειάζεται ένα αρχικό εφικτό δένδρο. Για την κατασκευή αυτού του αρχικού εφικτού δένδρου για τον NEPSA ή την διαπίστωση ότι το πρόβλημα μας είναι αδύνατο, θα χρησιμοποιήσουμε ενα ειδικά κατασκευασμένο ΠΡΕΚ, και όχι το πραγματικό ΠΡΕΚ που θέλουμε να λύσουμε. Το νέο ΠΡΕΚ δημιουργείται με την εισαγωγή ενός τεχνητού κόμβου (artificial node) και μερικών τεχνητών τόξων (artificial arcs), τα οποία όμως έχουν μία πολύ μεγάλη τιμή κόστους M. Λόγω του γεγονότος ότι η τιμή του κόστους M είναι πολύ μεγάλη, το πρόβλημα ονομάζεται πρόβλημα του μεγάλου M (big M problem) ή πιό απλά τεχνητό πρόβλημα M (artificial problem). Τα συμπεράσματα για το αρχικό ΠΡΕΚ προκύπτουν από την επίλυση του προβλήματος του μεγάλου M. Ένα πλεονέκτημα της μεθόδου του μεγάλου M είναι το εύκολα υπολογιζόμενο εφικτό δέντρο, το οποίο και χρησιμοποιείται έπειτα ως δένδρο ξεκινήματος του NEPSA.

Πρέπει να αναφερθεί, ότι αυτή η μέθοδος μπορεί να εφαρμοσθεί και στον κλασικό αλγόριθμο NPSA, αφού και οι δύο ξεκινούν από ένα αρχικό πρωτευόντως εφικτό δένδρο. Αντιθέτως, ο δυϊκός αλγόριθμος Simplex για δίκτυα, ή πιο απλά NDSA (Network Dual Simplex Algorithm) χρειάζεται ένα αρχικό δυικά εφικτό δένδρο ξεκινήματος. Αντίστοιχα για τον NDSA έχουν αναπτυχθεί άλλες μέθοδοι κατασκευής αρχικού δυϊκά εφικτού δένδρου, όπως για παράδειγμα η μέθοδος που περιγράφεται στις εργασίες (Glover, Klingman, and Napier 1972a), (Glover, Klingman, and Napier 1972b) και (Hultz, Klingman, and Russell 1976).

Πρέπει να σημειωθεί ότι, η επιλογή του αλγορίθμου που θα χρησιμοποιηθεί έχει να κάνει με το αρχικό δένδρο ξεκινήματος. Για παράδειγμα, για να επιλυθεί ένα ΠΡΕΚ με τη βοήθεια του δυϊκού αλγορίθμου NDSA δεν πρέπει να μετατραπεί το πρωτεύων ΠΡΕΚ σε δυϊκό. Είναι προτιμότερο, ο δυϊκός αλγορίθμος NDSA να εφαρμόζεται μόνο όταν υπάρχει ένα αρχικά δυϊκά εφικτό ΠΡΕΚ.

Το πρόβλημα του μεγάλου M κατασκευάζεται από το αρχικό ΠΡΕΚ με τον παρακάτω τρόπο. Στην αρχή προστίθεται ένας τεχνητός κόμβος με ετικέτα $n+1$ και $b(n+1) = 0$. Σε όλους τους κόμβους προσφοράς i του αρχικού δικτύου προσαρτάται ένα τεχνητό τόξο $(i, n+1)$ το οποίο έχει μοναδιαίο κόστος $d_{i,n+1} = M$. Για κάθε έναν από τους υπόλοιπους κόμβους j του αρχικού δικτύου $G = (N, A)$ προστίθεται ένα τεχνητό τόξο $(n+1, j)$, ξανά με μοναδιαίο κόστος $d_{n+1}, j = M$. Όλα τα υπόλοιπα τόξα, δηλαδή τα τόξα του αρχικού ΠΡΕΚ, συνεχίζουν να έχουν κόστος $d_{ij} = c_{ij}$. Το προσαυξημένο δίκτυο του προβλήματος του μεγάλου M περιέχει τώρα πιά όλα τα τεχνητά τόξα, τον τεχνητό κόμβο μεταφόρτωσης $n+1$, καθώς και όλα τα τόξα του αρχικού δικτύου.

Ένα κρίσιμο ερώτημα που προκύπτει, έχει να κάνει με το μέγεθος του μεγάλου αριθμού M. Είναι δυνατόν να μη δοθεί συγκεκριμένη τιμή στο M, αλλά να θεωρηθεί ότι είναι τόσο μεγάλος έτσι ώστε η βέλτιστη τιμή της αντικειμενικής συνάρτησης του ΠΡΕΚ, (στην περίπτωση που αυτό είναι βέλτιστο), να είναι μικρότερη από κάθε άλλη αντικειμενική τιμή του προβλήματος του μεγάλου M, (στις εφικτές λύσεις στις οποίες τουλάχιστον μια τεχνητή μεταβλητή είναι θετική). Σε περίπτωση

που είναι επιθυμητό να δοθεί κάποια συγκεκριμένη τιμή στο M, τότε αυτή πρέπει να είναι μεγαλύτερη του αριθμού $(n-1)CU+1$, όπου:

$$C = max|c_{ij}| : (i,j) \in A \tag{9}$$

$$U = \sum b(i) : b(i) > 0 \tag{10}$$

Πρέπει να σημειωθεί ότι ο αριθμός $(n-1)CU$ αποτελεί ένα πάνω όριο των αντικειμενικών τιμών στα εφικτά δέντρα του ΠΡΕΚ. Αυτό ισχύει διότι, το δέντρο έχει $n-1$ τόξα, κάθε τόξο έχει ροή το πολύ U και τα τόξα του δέντρου έχουν κόστος το πολύ C. Παρ΄ όλα αυτά, το μειονέκτημα της μεθόδου του μεγάλου M εστιάζεται στο γεγονός ότι είναι δυνατόν να προκύψουν αριθμητικά λάθη. Αυτό συμβαίνει κάποιες φορές μετά από προσθέσεις ή πολλαπλασιασμούς δυϊκών ή χαλαρών μεταβλητών π.χ. w_i ή s_{ij}, που εκφράζονται ως πολλαπλάσια του μεγάλου M. Σε αυτές τις περιπτώσεις είναι δυνατόν να ξεπεραστεί το εύρος του συγκεκριμένου τύπου ακεραίων αριθμών που είναι δηλωμένες οι μεταβλητές w_i και s_{ij}, με αποτέλεσμα να προκύψει αριθμητικό λάθος.

Για την αποφυγή αυτού του δυσάρεστου γεγονότος, στην υλοποίηση του NEPSA αποφασίστηκε να χωριστούν οι συντελεστές του M στο διάνυσμα κόστους, από τους πραγματικούς συντελεστές του αρχικού διανύσματος κόστους. Με αυτόν τον τρόπο, όποτε έπρεπε να γίνουν συγκρίσεις μεταξύ τιμών κόστους ή δυϊκών μεταβλητών ή χαλαρών μεταβλητών, γινόταν σύγκριση πρώτα σε επίπεδο συντελεστών του M και έπειτα σε επίπεδο πραγματικών τιμών.

Στην αμέσως επόμενη ενότητα, παρουσιάζεται ο ψευδοκώδικας του αλγορίθμου NEPSA.

2.3 Ψευδοκώδικας

Algorithm 3 Αλγόριθμος NEPSA

Require: $G = (N, A), b, c, T$

procedure NEPSA(G, T)
Βήμα 0 (Αρχικοποίηση)
    Compute x, w and s
    Compute vectors h_{ij}, and d, using Relation 3, (Chapter 1), and 5, (Chapter 2)
Βήμα 1 (Έλεγχος βελτιστότητας)
    while $P \neq \emptyset$ **do**
        if $d(T) \geq 0$ **then**
            STOP. The problem (NP.1) is unbounded.
        else
Βήμα 2 (Επιλογή εξερχομένου τόξου)
            Compute a, using Relation 6.
            Choose the leaving arc (k, l) and compute y
Βήμα 3 (επιλογή εισερχομένου τόξου)
            Compute θ_1, θ_2 using Relations 7 and 8.
            Choose the entering arc (g, h)
Βήμα 4 (Περιστροφή)
            if $\theta_1 \leq \theta_2$ **then**
                Set the entering arc $(g, h) = (p_1, p_2)$
                Set $P = P \sim (g, h)$.
            else
                Set the entering arc $(g, h) = (q_1, q_2)$
                Set $Q = Q \sim (g, h)$.
            end if
            $Q = Q \cup (k, l)$
            Set $T = T \sim (k, l) \cup (g, h)$
            Update x, s, h_{ij} and d
        end if
    end while
    STOP. The problem (NP.1) is optimal.
end procedure

Ο αλγόριθμος NEPSA αρχικοποιείται κατασκευάζοντας ένα εφικτό δένδρο με τη μέθοδο του μεγάλου M. Κατά την αρχικοποίηση ο υπολογισμός της βασικής ροής x απαιτεί n επαναλήψεις, όσα είναι και τα αρχικά βασικά - τεχνητά τόξα. Ακόμη, ο υπολογισμός των δυϊκών μεταβλητών w απαιτεί συνολικά $n + 1$ επαναλήψεις, όσοι είναι και συνολικά οι κόμβοι. Τέλος, ο υπολογισμός των ελλειματικών κόστων s απαιτεί m επαναλήψεις, όσα είναι και τα μη βασικά - πραγματικά τόξα. Άρα,

για την κατασκευή του αρχικού δένδρου με τη μέθοδο του μεγάλου M, απαιτούνται $(n)+(n+1)+(m)=2n+m+1$ επαναλήψεις. Άρα, η πολυπλοκότητα της φάσης αρχικοποίησης είναι $\Theta(n+m)$.

2.4 Γραφικά παραδείγματα του αλγορίθμου NEPSA

Σε αυτήν την Ενότητα θα δοθούν δύο γραφικά παραδείγματα, έτσι ώστε να γίνει πιο κατανοητός ο τρόπος λειτουργίας του αλγορίθμου NEPSA. Το πρώτο παράδειγμα, παρουσιάζεται στην Υποενότητα 2.4.1 και περιγράφει ένα ΠΡΕΚ μέ βέλτιστη λύση. Το δεύτερο παράδειγμα, παρουσιάζεται στην Υποενότητα 2.4.2 και περιγράφει ένα ΠΡΕΚ μέ απεριοριστία.

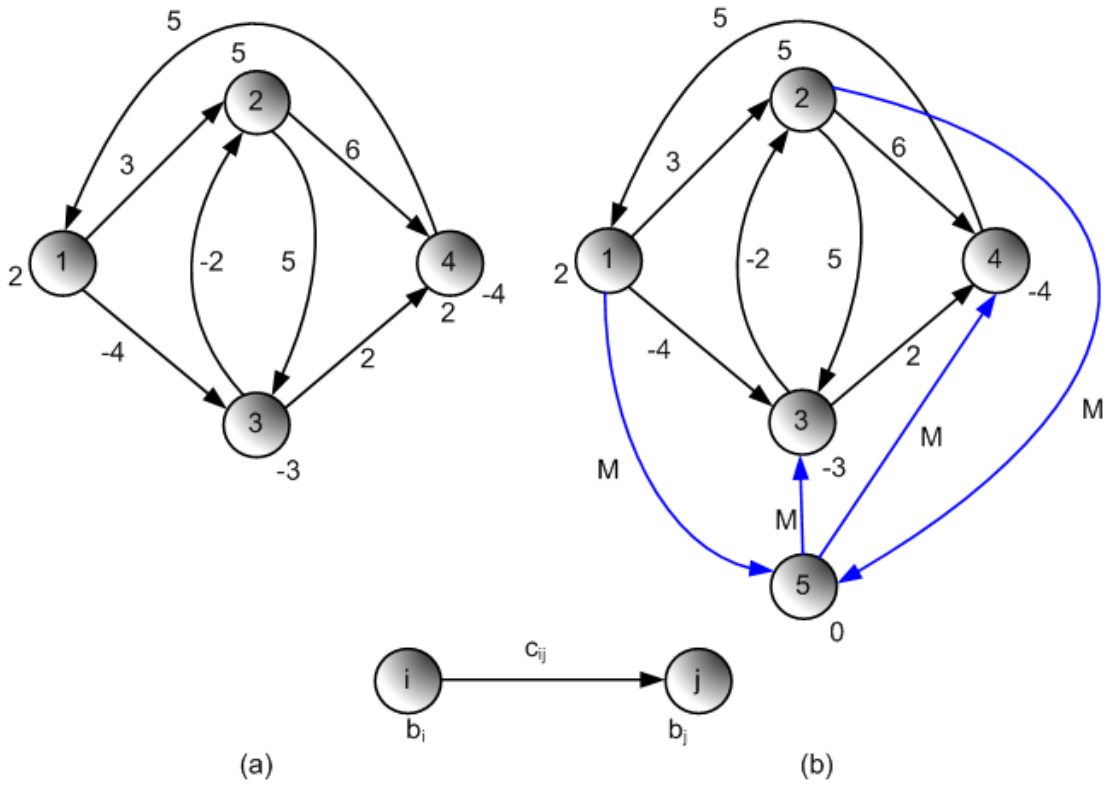
2.4.1 Ένα πρόβλημα με βέλτιστη λύση

Σε αυτήν την Υποενότητα θα γίνει μια παρουσίαση των δύο διαφορετικών τύπων επαναλήψεων, με τη βοήθεια του επομένου παραδείγματος. Θα αρκεστούμε στη περιγραφή μόνο δύο επαναλήψεων και όχι μέχρι την εύρεση της τελικής λύσης, για λόγους συντομίας. Το δίκτυο που παριστάνει το ΠΡΕΚ που συζητάμε είναι αυτό που φαίνεται στο Σχήμα 2.4 (a). Στο *Βήμα 0*, η μέθοδος του Μεγάλου M εφαρμόζεται στο αρχικό δίκτυο. Μετά την εισαγωγή του τεχνητού κόμβου και των αντιστοίχων τεχνητών τόξων, το προσαυξημένο δίκτυο φαίνεται στο Σχήμα 2.4 (b).

Στο *Βήμα 0*, ο αλγόριθμος NEPSA ξεκινάει με το αρχικό εφικτό δένδρο, το οποίο απεικονίζεται με μπλέ χρώμα στο Σχήμα 2.4 (b). Το διάνυσμα βασικής ροής είναι $x^{(1)}=(x_{15}^{(1)},x_{25}^{(1)},x_{53}^{(1)},x_{54}^{(1)})$ $\Rightarrow x^{(1)}=(2,5,3,4)$. Το διάνυσμα δυϊκών μεταβλητών είναι $w^{(1)}=(w_1^{(1)},w_2^{(1)},w_3^{(1)},w_4^{(1)},w_5^{(1)})\Rightarrow$ $w^{(1)}=(127,127,-127,-127,0)$, τα ελλειματικά κόστη είναι $s^{(1)}=(s_{13}^{(1)},s_{23}^{(1)},s_{24}^{(1)},s_{12}^{(1)},s_{32}^{(1)},s_{34}^{(1)})$ $\Rightarrow s^{(1)}=(-258,-249,-248,3,252,2)$.

Οπότε έχουμε $P^{(1)}=\{(1,3),(2,3),(2,4)\}$ και $Q^{(1)}=\{(1,2),(3,2),(3,4),(4,1)\}$. Το συνολικό κόστος που αντιστοιχεί στο διάνυσμα βασικής ροής $x^{(1)}$, είναι $z^{(1)}=\sum_{(i,j)\in T^{(1)}} c_{ij}x_{ij}^{(1)}\Rightarrow z^{(1)}=1778$.

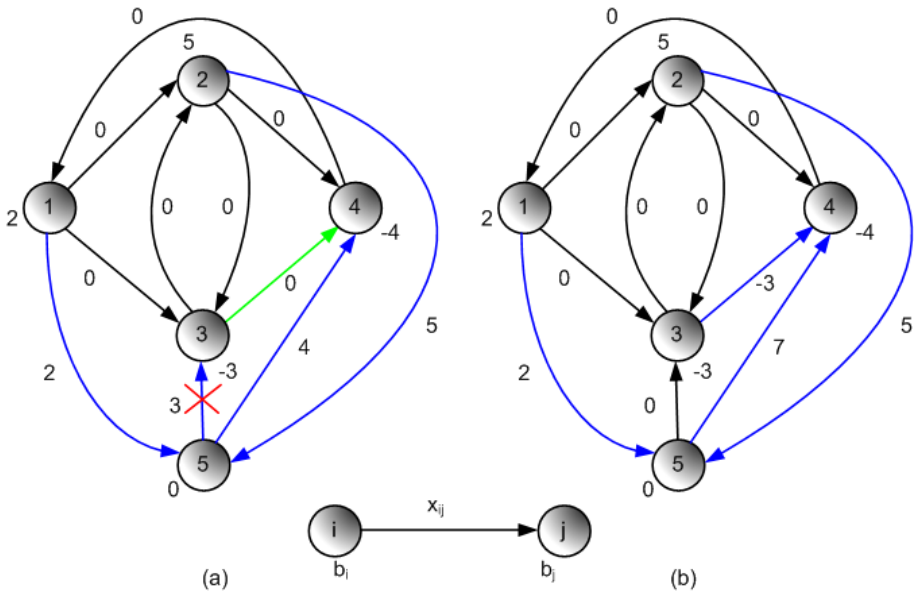
1η Επανάληψη



Σχήμα 2.4 Μέθοδος του Μεγάλου M: Το αρχικό δίκτυο (a) και το προσαυξημένο δίκτυο (b).

Στο *Βήμα 1*, ισχύει ότι $P^{(1)} \neq \emptyset$, οπότε ο αλγόριθμος δεν τερματίζει. Αφού, $d^{(1)}(T) = \left\{d_{15}^{(1)}, d_{25}^{(1)}, d_{53}^{(1)}, d_{54}^{(1)}\right\} \Rightarrow d^{(1)} = \{-1, -2, -2, -1\}$, ο αλγόριθμος NEPSA προχωράει στο *Βήμα 2*. Ο έλεγχος ελαχίστου λόγου είναι $a^{(1)} = 1{,}5$ και το τόξο $(5, 3)$ εξέρχεται από το $T^{(1)}$. Το εξερχόμενο τόξο χρωματίζεται όπως φαίνεται στο Σχήμα 2.5 (a) χρησιμοποιώντας κόκκινες διακεκομμένες γραμμές.

Στο *Βήμα 3*, οι λόγοι $\vartheta_1^{(1)}$ και $\vartheta_1^{(2)}$ είναι 249 και 2 αντιστοίχως, οπότε ισχύει ότι $\theta_1^{(1)} > \theta_2^{(1)}$ και το τόξο (3,4) θα εισέλθει στο δένδρο $T^{(1)}$. Επομένως, αυτή είναι μια επανάληψη τύπου B. Στη 2η επανάληψη, το διάνυσμα βασικής ροής δεν πρόκειται να είναι εφικτό. Αυτό το γεγονός σε όρους του Γραμμικού Προγραμματισμού ισοδυναμεί με το να ακολουθηθεί ένα εξωτερικό μονοπάτι, (μονοπάτι στο εξωτερικό της εφικτής περιοχής). Το εισερχόμενο τόξο απεικονίζεται στο Σχήμα 2.5 (a) χρησιμοποιώντας πράσινο χρώμα. Στο *Βήμα 4*, το νέο βασικό δένδρο $T^{(2)}$ απεικονίζεται στο Σχήμα 2.5 (b).



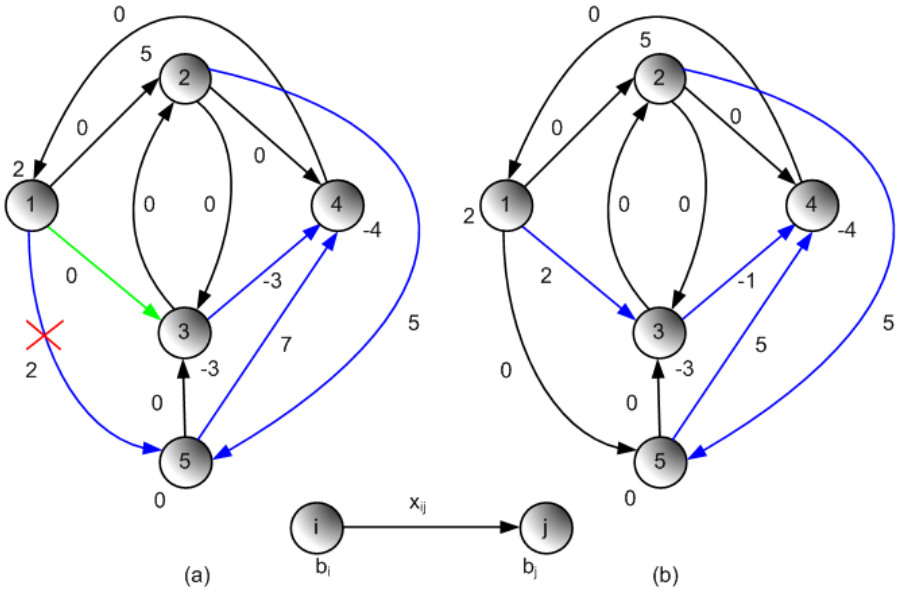
Σχήμα 2.5: Επανάληψη Τύπου B: Το εξερχόμενο & εισερχομένο τόξο (a) και το ανανεωμένο βασικό δένδρο $T^{(2)}$ (b).

Η ανανεωμένη βασική ροή γίνεται $x^{(2)} = (x_{15}^{(2)}, x_{25}^{(2)}, x_{34}^{(2)}, x_{54}^{(2)}) \Rightarrow x^{(2)} = (2, 5, -3, 7)$. Επιπλέον

δε, τα ανανεωμένα ελλειματικά κόστη τώρα πια γίνονται $s^{(2)} = (s_{13}^{(2)}, s_{23}^{(2)}, s_{24}^{(2)}, s_{12}^{(2)}, s_{32}^{(2)}, s_{41}^{(2)}) \Rightarrow$ $s^{(2)} = (-256, -247, -248, 3, 250, 259)$.

Οπότε έχουμε $P^{(2)} = \{(1,3), (2,3), (2,4)\}$ και $Q^{(2)} = \{(1,2), (3,2), (4,1)\}$, (το τόξο (5,3) ήταν τεχνητό, οπότε αφαιρείται από οποιοδήποτε μελλοντικούς υπολογισμούς). Η τιμή κόστους της αντικειμενικής συνάρτησης που αντιστοιχεί στη βασική ροή $x^{(2)}$, είναι $z^{(2)} = \sum_{(i,j)\in T} c_{ij} x_{ij}^{(2)} \Rightarrow$ $z^{(2)} = 1772$

2η Επανάληψη



Σχήμα 2.6: Επανάληψη Τύπου Α: Το εξερχόμενο & εισερχομένο τόξο (a) και το ανανεωμένο βασικό δένδρο $T^{(3)}$ (b).

Στο *Βήμα 1*, ισχύει ότι $P^{(2)} \neq \emptyset$ και ως εκ τούτου ο αλγόριθμος δεν ολοκληρώνεται. Αφού, $d^{(2)}(T) = \left\{d_{15}^{(2)}, d_{25}^{(2)}, d_{34}^{(2)}, d_{54}^{(2)}\right\} \Rightarrow d^{(2)} = \{-1, -2, 2, -3\}$, ο NEPSA προχωράει στο *Βήμα 2*. Ο έλεγχος ελαχίστου λόγου είναι $a^{(2)} = 2$ και το τόξο (1,5) εξέρχεται από το $T^{(2)}$. Το εξερχόμενο τόξο απεικονίζεται στο Σχήμα 2.6 (a) χρησιμοποιώντας πάλι κόκκινες διακεκομμένες γραμμές. Στο *Βήμα 2*, οι λόγοι $\vartheta_1^{(1)}$ και $\vartheta_1^{(2)}$ είναι 256 και 259 αντιστοίχως, οπότε ισχύει ότι $\theta_1^{(1)} < \theta_2^{(1)}$ και το

τόξο (1,3) θα εισέλθει στο βασικό δένδρο $T^{(2)}$. Συνεπώς, αυτή είναι μια επανάληψη τύπου A. Το εισερχόμενο τόξο απεικονίζεται πάλι στο Σχήμα 2.6 (a) χρησιμοποιώντας πάλι πράσινο χρώμα. Στο *Βήμα 4*, το νέο βασικό δένδρο $T^{(3)}$ εμφανίζεται στο Σχήμα 2.6 (b).

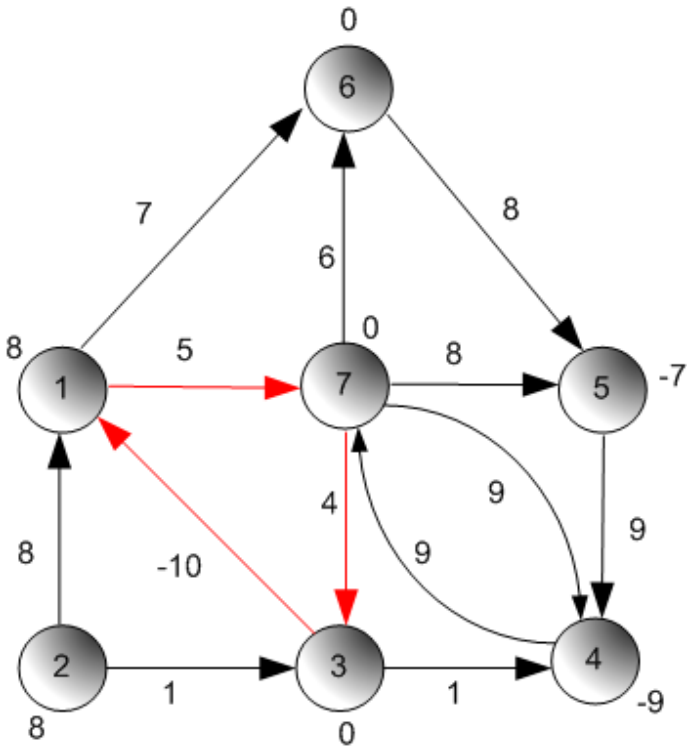
Αυτή ήταν μια γραφική αναπαράσταση των δυο χαρακτηριστικών τύπων επανάληψης του αλγόριθμου NEPSA. Οι υπόλοιποι υπολογισμοί είναι παρόμοιοι, οπότε το παράδειγμα θα σταματήσει εδώ και δεν θα ολοκληρωθεί.

2.4.2 Ένα πρόβλημα με απεριοριστία

Έστω ότι έχουμε το δίκτυο του Σχήματος 2.7. Θα εφαρμοστεί ο αλγόριθμος ώστε να βρεθεί η Ροή Ελαχίστου Κόστους. Στο συγκεκριμένο ΠΡΕΚ υπάρχουν 7 κόμβοι και 13 τόξα. Οι κόμβοι 1 και 2 είναι κόμβοι προσφοράς, οι κόμβοι 4 και 5 είναι κόμβοι ζήτησης και τέλος οι κόμβοι 3, 6 και 7 είναι κόμβοι μεταφόρτωσης. Οι αριθμοί, δίπλα στα τόξα συμβολίζουν τα αντίστοιχα κόστη. Θεωρούμε ότι δεν υπάρχει άνω χωρητικότητα, ενώ η κάτω χωρητικότητα των τόξων ισούται με 0. Παρατηρούμε ότι το δίκτυο περιέχει έναν αρνητικά προσανατολισμένο κύκλο. Τα τόξα που αποτελούν τον συγκεκριμένο κύκλο ζωγραφίζονται με κόκκινο χρώμα.

Όταν υπάρχει ένας αρνητικά προσανατολισμένος κύκλος, το πρόβλημα εξελίσσεται σε απεριόριστο. Αυτό γίνεται διότι μπορεί να σταλεί ροή κατά μήκος του κύκλου και το συνολικό κόστος της ροής συνεχώς να μειώνεται, όπως περιγράφεται και στο βιβλίο (Bazaraa, Jarvis, and Sherali 2005). Αυτό φυσικά συμβαίνει μόνο σε ΠΡΕΚ δίχως άνω χωρητικότητες, όπου μπορεί να διοχετευθεί οποιαδήποτε ποσότητα ροής. Εάν στο ΠΡΕΚ υπάρχουν και άνω χωρητικότητες, τότε η τιμή της αντικειμενικής συνάρτησης δεν θα ελαττωθεί από κάποιο σημείο και μετά, αφού δεν είναι δυνατόν να διοχετευθεί ποσότητα ροής, η οποία να ξεπερνάει την άνω χωρητικότητα των τόξων εκείνου του κύκλου.

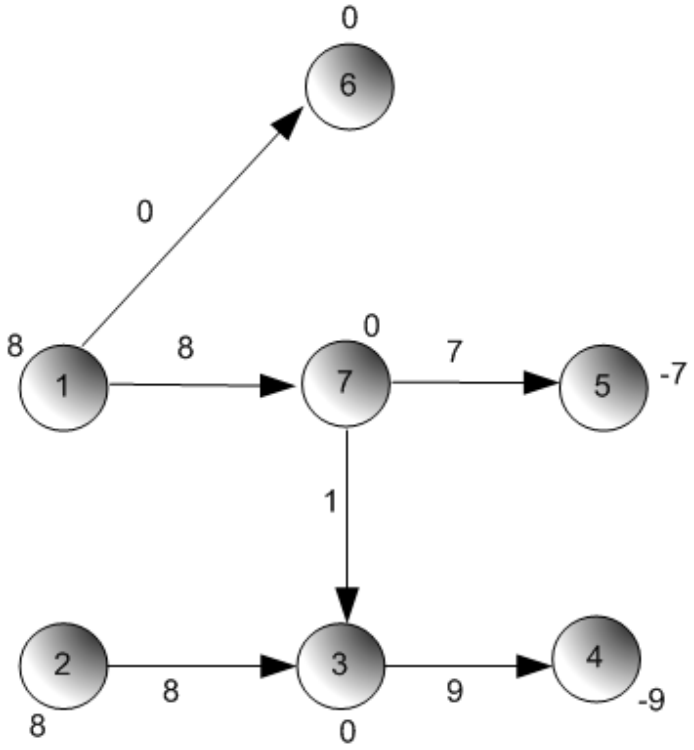
1η επανάληψη



Σχήμα 2.7 Παράδειγμα απεριοριστίας: Δίκτυο του προβλήματος

Έστω ότι ο αλγόριθμος θα ξεκινήσει με το πρωτευόντως εφικτό δένδρο του Σχήματος 2.8, άρα είναι

$T^{(1)} = \{(1,6),(1,7),(2,3),(3,4),(7,3),(7,5)\}$



Σχήμα 2.8 Παράδειγμα απεριοριστίας: Αρχικό εφικτό δένδρο $T^{(1)}$

Άρα το δέντρο $T^{(1)}$ είναι πρωτευόντως εφικτό

Τίθεται $w_7^{(1)} = 0$

Υπολογισμός $s_{ij}^{(1)}, \forall (i,j) \notin T^{(1)}$

$s_{21}^{(1)} = c_{21} - w_2^{(1)} + w_1^{(1)} = 8 - (-3) + 5 = 16 > 0$

$s_{31}^{(1)} = c_{31} - w_3^{(1)} + w_1^{(1)} = 1 - (-4) + 5 = -1 < 0$

$s_{47}^{(1)} = c_{47} - w_4^{(1)} + w_7^{(1)} = 9 - (-5) + 0 = 14 > 0$

$s_{54}^{(1)} = c_{54} - w_5^{(1)} + w_4^{(1)} = 9 - (-8) - 5 = 12 > 0$

$s_{65}^{(1)} = c_{65} - w_6^{(1)} + w_5^{(1)} = 8 - (-2) - 8 = 2 > 0$

$s_{74}^{(1)} = c_{74} - w_7^{(1)} + w_4^{(1)} = 9 - 0 - 5 = 4 > 0$

Κόμβος	Εξίσωση κόμβου	Τιμή αγνώστου
1	$x_{16}^{(1)} + x_{17}^{(1)} = 8$	$x_{17}^{(1)} = 8$
2	$x_{23}^{(1)} = 8$	$x_{23}^{(1)} = 8$
3	$x_{34}^{(1)} - x_{23} - x_{73}^{(1)} = 0$	$x_{73}^{(1)} = 1$
4	$-x_{34}^{(1)} = -9$	$x_{34}^{(1)} = 9$
5	$-x_{75}^{(1)} = -7$	$x_{75}^{(1)} = 7$
6	$-x_{16}^{(1)} = 0$	$x_{16}^{(1)} = 0$
7	$x_{73}^{(1)} + x_{75}^{(1)} - x_{17}^{(1)} = 0$	Ισχύει

Πίνακας **2.2** Υπολογισμός $x^{(1)}(T)$

Τόξο	Εξίσωση	Τιμή αγνώστου
(1, 6)	$w_1^{(1)} - w_6^{(1)} = 7$	$w_6^{(1)} = -2$
(1, 7)	$w_1^{(1)} - w_7^{(1)} = 5$	$w_1^{(1)} = 5$
(2, 3)	$w_2^{(1)} - w_3^{(1)} = 1$	$w_2^{(1)} = -3$
(3, 4)	$w_3^{(1)} - w_4^{(1)} = 1$	$w_4^{(1)} = -5$
(7, 3)	$w_7^{(1)} - w_3^{(1)} = 4$	$w_3^{(1)} = -4$
(7, 5)	$w_7^{(1)} - w_5^{(1)} = 8$	$w_5^{(1)} = -8$

Πίνακας **2.3** Υπολογισμός $w_i^{(1)}(T)$

$s_{76}^{(1)} = c_{76} - w_7^{(1)} + w_6^{(1)} = 6 - 0 - 2 = 4 > 0$

Άρα το δέντρο $T^{(1)}$ δεν είναι δυϊκά εφικτό

Είναι $P^{(1)} = \{(3,1)\}$ και $Q^{(1)} = \{(2,1),(4,7),(5,4),(6,5),(7,4),(7,6)\}$

	(3,1)	(2,1)	(4,7)	(5,4)	(6,5)	(7,4)	(7,6)
(1,6)	0	0	0	0	-1	0	1
(1,7)	-1	-1	0	0	1	0	-1
(2,3)	0	1	0	0	0	0	0
(3,4)	0	0	-1	1	0	1	0
(7,3)	-1	-1	-1	1	0	1	0
(7,5)	0	0	0	-1	1	0	0

Πίνακας **2.4** Υπολογισμός στηλών $h_{ij}^{(1)}$

Αφού μόνο το $(3,1) \in P$, άρα ισχύει ότι $d^{(1)}(T) = -h_{31}^{(1)} = (0,1,0,0,1,0)'$

Βήμα 1. Έλεγχος βελτιστότητας

Ισχύει ότι $P \neq \emptyset$, άρα ο αλγόριθμος δεν έχει φτάσει στη βέλτιστη λύση. Όμως, ισχύει ότι $h_{31}^{(1)} \leq 0$, άρα έχει ανιχνευθεί απεριοριστία. Οπότε ο αλγόριθμος τερματίζει.

Στο αμέσως επόμενο κεφάλαιο, δίνονται οι αποδείξεις ορθότητας του αλγορίθμου.

ΚΕΦΑΛΑΙΟ 3

Μαθηματικές αποδείξεις ορθότητας

3.1 Θεωρήματα

Οι αποδείξεις ορθότητας του NEPSA παρουσιάζονται στην εργασία (Paparrizos, Samaras, and Sifaleras 2006a) και στηρίζονται στην απόδειξη των θεωρημάτων και λημμάτων που θα περιγραφούν στη συνέχεια. Οι αποδείξεις στηρίζονται στην υπόθεση ότι τα προβλήματα στα οποία εφαρμόζεται ο αλγόριθμος δεν είναι εκφυλισμένα. Συνεπώς, ισχύει $x_{kl} \neq 0$ και $s_{gh} \neq 0$.

Παρ΄ όλα αυτά όμως, στην Ενότητα 4.2 παρουσιάζεται η χρήση μιας μεθόδου που υλοποιήθηκε για την αντιμετώπιση των εκφυλισμένων προβλημάτων στον αλγόριθμο NEPSA. Αυτή η προτεινόμενη μεθοδος, φαίνεται εμπειρικά να έχει επιτύχια. Όμως, σίγουρα χρειάζεται να αναπτυχθεί μια μέθοδο για την αποφυγή του προβλήματος της κύκλωσης και της στασιμότητας, η οποία να αποδεικνύεται και με μαθηματικό τρόπο.

Συνολικά για την απόδειξη της ορθότητας του NEPSA χρειάστηκε να αποδειχθούν εννιά θεωρήματα. Το Θεώρημα 2, διασφαλίζει ότι υπό οποιοδήποτε τύπο επανάληψης, η τιμή της αντικειμενικής συνάρτησης θα μειώνεται συνεχώς.

Θεώρημα 2 *Εάν το πρόβλημα δεν είναι εκφυλισμένο, τότε ελαττώνεται αυστηρά η τιμή της αντικειμενικής συνάρτησης, από επανάληψη σε επανάληψη.*

Απόδειξη . Έστω z^t η τιμή της αντικειμενικής συνάρτησης κατά την t επανάληψη. Αν η επανάληψη είναι τύπου Α τότε, η διαφορά ανάμεσα στην τρέχουσα και στην επόμενη αντικειμενική τιμή $\Delta z = z^{t+1} - z^t$ οφείλεται στη ροή των x_{kl}^t μονάδων που ρέει δια μέσου του εισερχόμενου τόξου (g,h) στον κύκλο C^t. Αρκεί αυτή η μεταβολή ροής να συνοδεύεται και με μείωση του συνολικού κόστους. Όταν μια μονάδα ($x_{kl} = 1$) ρέει δια μέσου του κύκλου, η αντικειμενική τιμή μεταβάλλεται κατά $\Delta z = 1 \cdot \sum_{(i,j)\in C} t_{ij}c_{ij}$, όπου $t_{ij} = 1$, εαν $h_{gh}(i,j) = -1$ και $t_{ij} = -1$, εαν $h_{gh}(i,j) = 1$. Ως γνωστό ισχύει ότι $\sum_{(i,j)\in C} t_{ij}c_{ij} = s_{gh} \Rightarrow \Delta z < 0$, (αφού $s_{gh}^t < 0$ σε επανάληψη τύπου Α).

Αν η επανάληψη είναι τύπου Β, τότε η διαφορά ανάμεσα στην τρέχουσα και στην επόμενη αντικειμενική τιμή Δz οφείλεται στη ροή των $-x_{kl}^t$ μονάδων, που ρέει δια μέσου του εισερχόμενου τόξου στον κύκλο C^t. Όπως και προηγουμένως, αρκεί αυτή η μεταβολή ροής να συνοδεύεται και με μείωση του συνολικού κόστους. Όταν μια αρνητική μονάδα ρέει δια μέσου του κύκλου, η αντικειμενική τιμή μεταβάλλεται κατά $\Delta z = 1 \cdot \sum_{(i,j)\in C} t'_{ij}c_{ij}$, όπου τώρα $t'_{ij} = -1$, εαν $h_{gh}(i,j) = -1$ και $t'_{ij} = 1$, εαν $h_{gh}(i,j) = 1$. Αντίστοιχα, $\sum_{(i,j)\in C} t'_{ij}c_{ij} = -\sum_{(i,j)\in C} t_{ij}c_{ij} \Rightarrow \Delta z = -s_{gh} \Rightarrow \Delta z < 0$, (αφού $s_{gh}^t > 0$ σε επανάληψη τύπου Β).

Άρα σε κάθε είδους επανάληψη, υπάρχει μείωση του συνολικού κόστους. Αυτό ολοκληρώνει την απόδειξη.♣

Το Θεώρημα 3, διασφαλίζει ότι ο προτεινόμενος αλγόριθμος θα ολοκληρωθεί μετά από πεπερασμένο αριθμό επαναλήψεων.

Θεώρημα 3 *Εάν το πρόβλημα δεν είναι εκφυλισμένο, ο αλγόριθμος θα εκτελέσει πεπερασμένου πλήθους επαναλήψεις.*

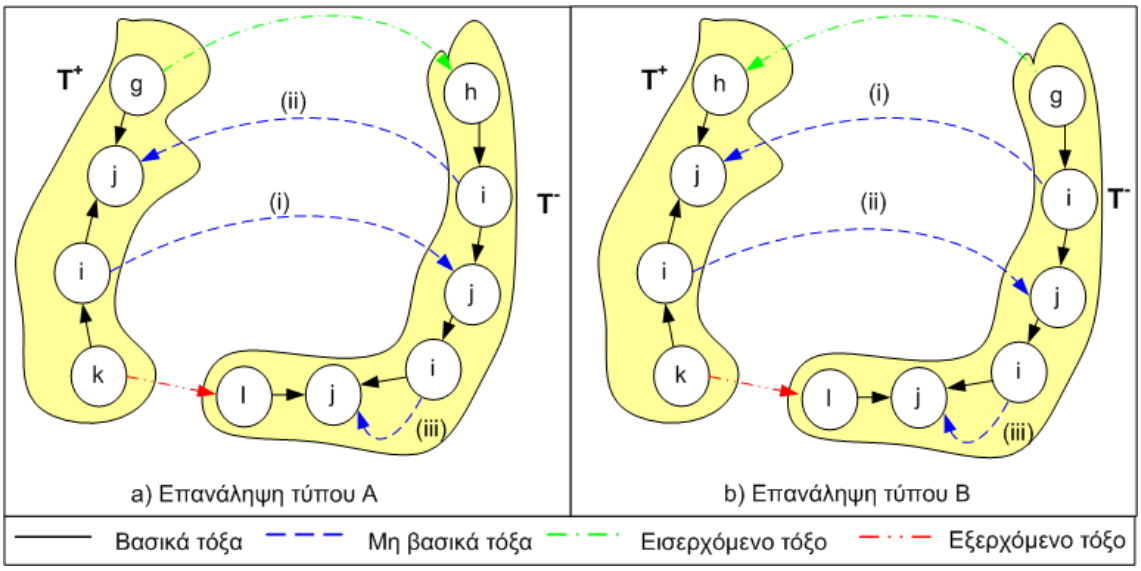
Σύμφωνα με το Θεώρημα 2 η αντικειμενική τιμή ελαττώνεται αυστηρά από επανάληψη σε επανάληψη. Άρα κανένα δέντρο δεν θα κατασκευαστεί δυο φορές. Τα δέντρα του δικτύου G κατασκευάζονται με πεπερασμένους συνδυασμούς των n κόμβων και των m τόξων. Επειδή το πλήθος

των κόμβων και των τόξων είναι πεπερασμένο, το πλήθος των δέντρων είναι επίσης πεπερασμένο. Επομένως, αλγόριθμος θα σταματήσει μετά από πεπερασμένο αριθμό επαναλήψεων. ♣

Το Θεώρημα 4, διασφαλίζει ότι σε όλη τη διάρκεια του αλγορίθμου NEPSA και μετά από κάθε επανάληψη τύπου A, οι τιμές των ελλειματικών κόστων θα συνεχίζουν να είναι τέτοιες ώστε να ισχύουν οι Σχέσεις 1 και 2, του Κεφαλάιου 2.

Θεώρημα 4 *Σε κάθε επανάληψη τύπου A του αλγορίθμου ισχύει ότι αν* $(i,j) \in P$ *τότε* $s_{ij} < 0$, *και αν* $(i,j) \in Q$ *τότε* $s_{ij} \geq 0$.

Απόδειξη: Η απόδειξη του θεωρήματος 4 γίνεται επαγωγικά. Στην 1η επανάληψη ορίσαμε ότι αν $(i,j) \in P$ τότε $s_{ij}^1 < 0$. Έστω ότι στην t επανάληψη ισχύει ότι αν $(i,j) \in P$, τότε $s_{ij}^{(t)} < 0$. Θα δειχθεί ότι και στην $(t+1)$ επανάληψη θα εξακολουθεί να ισχύει. Αφού η t επανάληψη ήταν τύπου A, τότε $s_{gh}^{(t)} < 0$. Στη συνέχεια παρατίθεται διερεύνηση των παρακάτω τριών περιπτώσεων για το τόξο (i,j):



Σχήμα 3.1 Πιθανοί συνδυασμοί για οποιοδήποτε μη βασικό τόξο (i,j)

- 1η Περίπτωση : Άν το $(i,j) \in P$ συνδέει τα υποδένδρα T^+ και T^- με τον ίδιο προσανατολισμό

όπως το (g,h), τότε ήταν επιλέξιμο ως εισερχόμενο. Σε αυτή την περίπτωση έχουμε $-s_{ij}^{(t)} > \theta_1^{(t)}$, ή $-s_{ij}^{(t)} > -s_{gh}^{(t)} \Rightarrow s_{ij}^{(t)} - s_{gh}^{(t)} < 0 \Rightarrow s_{ij}^{(t+1)} < 0$, (απεικονίζεται στην περίπτωση (i) του Σχήματος 3.1 a).

- 2η Περίπτωση : Άν το $(i,j) \in P$ συνδέει τα υποδένδρα T^+ και T^- με τον αντίθετο προσανατολισμό σε σχέση με το (g,h), τότε δεν ήταν επιλέξιμο ως εισερχόμενο. Σε αυτή την περίπτωση έχουμε $s_{ij}^{(t+1)} = s_{ij}^{(t)} + s_{gh}^{(t)} \Rightarrow s_{ij}^{(t+1)} < 0$, (απεικονίζεται στην περίπτωση (ii) του Σχήματος 3.1 a).

- 3η Περίπτωση : Άν το $(i,j) \in P$ δεν συνδέει τα υποδένδρα T^+ και T^-, τότε $s_{ij}^{(t+1)} = s_{ij}^{(t)} \Rightarrow s_{ij}^{(t+1)} < 0$ (απεικονίζεται στην περίπτωση (iii) του Σχήματος 3.1 a).

Επομένως σε κάθε επανάληψη τύπου A του αλγορίθμου ισχύει ότι αν $(i,j) \in P : s_{ij} < 0$.

Στην 1η επανάληψη ορίσαμε ότι αν $(i,j) \in Q : s_{ij}^1 \geq 0$. Έστω ότι στην t επανάληψη ισχύει ότι αν $(i,j) \in Q$, τότε $s_{ij}^{(t)} > 0$. Θα δειχθεί ότι και στην $(t+1)$ επανάληψη θα εξακολουθεί να ισχύει. Αφού η t επανάληψη ήταν τύπου A, τότε $s_{gh}^{(t)} < 0$. Θα γίνει διερεύνηση των παρακάτω τριών περιπτώσεων για το τόξο (i,j):

- 1η Περίπτωση : Άν το $(i,j) \in Q$ συνδέει τα υποδένδρα T^+ και T^- με τον ίδιο προσανατολισμό όπως το (g,h), τότε δεν ήταν επιλέξιμο ως εισερχόμενο. Σε αυτή την περίπτωση έχουμε $s_{ij}^{(t+1)} = s_{ij}^{(t)} - s_{gh}^{(t)} \Rightarrow s_{ij}^{(t+1)} > 0$, (απεικονίζεται στην περίπτωση (i) του Σχήματος 3.1 a).

- 2η Περίπτωση : Άν το $(i,j) \in Q$ συνδέει τα υποδένδρα T^+ και T^- με τον αντίθετο προσανατολισμό όπως το (g,h), τότε ήταν επιλέξιμο ως εισερχόμενο. Σε αυτή την περίπτωση έχουμε $s_{ij}^{(t)} > \theta_2^{(t)} \Rightarrow s_{ij}^{(t)} > \theta_1^{(t)}$, ή $s_{ij}^{(t)} > -s_{gh}^{(t)} \Rightarrow s_{ij}^{(t)} + s_{gh}^{(t)} > 0 \Rightarrow s_{ij}^{(t+1)} > 0$, (απεικονίζεται στην περίπτωση (ii) του Σχήματος 3.1 a).

- 3η Περίπτωση : Άν το $(i,j) \in P$ δεν συνδέει τα υποδένδρα T^+ και T^-, τότε $s_{ij}^{(t+1)} = s_{ij}^{(t)} \Rightarrow$

$s_{ij}^{(t+1)} > 0$ (απεικονίζεται στην περίπτωση (iii) του Σχήματος 3.1 a).

Επομένως, σε κάθε επανάληψη τύπου A του αλγορίθμου ισχύει ότι αν $(i,j) \in Q : s_{ij} \geq 0$. ♣

Το Θεώρημα 5, διασφαλίζει ότι σε όλη τη διάρκεια του αλγορίθμου NEPSA και μετά από κάθε επανάληψη τύπου B, οι τιμές των ελλειματικών κόστων θα συνεχίζουν να είναι τέτοιες ώστε να ισχύουν οι Σχέσεις 1 και 2, του Κεφαλάιου 2.

Θεώρημα 5 *Σε κάθε επανάληψη τύπου B του αλγορίθμου ισχύει ότι αν $(i,j) \in P$ τότε $s_{ij} < 0$ και αν $(i,j) \in Q$ τότε $s_{ij} \geq 0$.*

Απόδειξη: Στην 1η επανάληψη ορίσαμε ότι αν $(i,j) \in P$ τότε $s_{ij}^1 < 0$. Έστω ότι στην t επανάληψη ισχύει ότι αν $(i,j) \in P$, τότε $s_{ij}^{(t)} < 0$. Θα δειχθεί ότι και στην $(t+1)$ επανάληψη θα εξακολουθεί να ισχύει. Αφού η t επανάληψη ήταν τύπου B, τότε $s_{gh}^{(t)} > 0$. Στη συνέχεια παρατίθεται διερεύνηση των παρακάτω τριών περιπτώσεων για το τόξο (i,j):

- 1η Περίπτωση : Άν το $(i,j) \in P$ συνδέει τα υποδένδρα T^+ και T^- με τον ίδιο προσανατολισμό όπως το (g,h), τότε δεν ήταν επιλέξιμο ως εισερχόμενο. Σε αυτή την περίπτωση έχουμε $s_{ij}^{(t+1)} = s_{ij}^{(t)} - s_{gh}^{(t)} \Rightarrow s_{ij}^{(t+1)} < 0$, (απεικονίζεται στην περίπτωση (i) του Σχήματος 3.1 b).

- 2η Περίπτωση : Άν το $(i,j) \in P$ συνδέει τα υποδένδρα T^+ και T^- με τον αντίθετο προσανατολισμό όπως το (g,h), τότε ήταν επιλέξιμο ως εισερχόμενο. Σε αυτή την περίπτωση έχουμε $-s_{ij}^{(t)} > \theta_1^{(t)} \Rightarrow -s_{ij}^{(t)} > \theta_2^{(t)}$, ή $-s_{ij}^{(t)} > s_{gh}^{(t)} \Rightarrow s_{ij}^{(t)} + s_{gh}^{(t)} > 0 \Rightarrow s_{ij}^{(t+1)} < 0$, (απεικονίζεται στην περίπτωση (ii) του Σχήματος 3.1 b).

- 3η Περίπτωση : Άν το $(i,j) \in P$ δεν συνδέει τα υποδένδρα T^+ και T^-, τότε $s_{ij}^{(t+1)} = s_{ij}^{(t)} \Rightarrow s_{ij}^{(t+1)} < 0$, (απεικονίζεται στην περίπτωση (iii) του Σχήματος 3.1 b).

Επομένως, σε κάθε επανάληψη τύπου B του αλγορίθμου ισχύει ότι αν $(i,j) \in P$ τότε $s_{ij} < 0$.

Στην 1η επανάληψη ορίσαμε ότι αν $(i,j) \in Q$ τότε $s_{ij}^1 \geq 0$. Έστω ότι στην t επανάληψη ισχύει ότι αν $(i,j) \in Q$, τότε $s_{ij}^{(t)} > 0$. Θα δειχθεί ότι και στην $(t+1)$ επανάληψη θα εξακολουθεί να ισχύει. Αφού η t επανάληψη ήταν τύπου B, τότε $s_{gh}^{(t)} > 0$. Θα γίνει διερεύνηση των παρακάτω τριών περιπτώσεων για το τόξο (i,j):

- 1η Περίπτωση : Άν το $(i,j) \in Q$ συνδέει τα υποδένδρα T^+ και T^- με τον ίδιο προσανατολισμό όπως το (g,h), τότε ήταν επιλέξιμο ως εισερχόμενο. Σε αυτή την περίπτωση έχουμε $s_{ij}^{(t)} > \theta_2^{(t)}$, ή $s_{ij}^{(t)} > s_{gh}^{(t)} \Rightarrow s_{ij}^{(t)} - s_{gh}^{(t)} > 0 \Rightarrow s_{ij}^{(t+1)} > 0$, (απεικονίζεται στην περίπτωση (i) του Σχήματος 3.1 b).

- 2η Περίπτωση : Άν το $(i,j) \in Q$ συνδέει τα υποδένδρα T^+ και T^- με τον αντίθετο προσανατολισμό όπως το (g,h), τότε δεν ήταν επιλέξιμο ως εισερχόμενο. Σε αυτή την περίπτωση έχουμε $s_{ij}^{(t+1)} = s_{ij}^{(t)} + s_{gh}^{(t)} \Rightarrow s_{ij}^{(t+1)} > 0$, (απεικονίζεται στην περίπτωση (ii) του Σχήματος 3.1 b).

- 3η Περίπτωση : Άν το $(i,j) \in P$ δεν συνδέει τα υποδένδρα T^+ και T^+, τότε $s_{ij}^{(t+1)} = s_{ij}^{(t)} \Rightarrow s_{ij}^{(t+1)} > 0$, (απεικονίζεται στην περίπτωση (iii) του Σχήματος 3.1 b).

Επομένως σε κάθε επανάληψη τύπου B, του αλγορίθμου ισχύει ότι αν $(i,j) \in Q$ τότε $s_{ij} \geq 0$

♣

Τα επόμενα δύο Θεωρήματα 6 και 7 παίζουν πολύ σημαντικό ρόλο στην κατασκευή των αποδείξεων για την απεριοριστία, όπως επίσης και για την εφικτότητα της ροής y. Σε όρους Γραμμικής Βελτιστοποίησης θα μπορούσε να ειπωθεί ότι τα επόμενα δύο Θεωρήματα διασφαλίζουν ότι ο αλγόριθμος NEPSA, θα υπολογίσει ένα μονοπάτι - κατεύθυνση, η οποία να δισχίζει το εσωτερικό της εφικτής περιοχής του προβλήματος.

Αφού είναι απαραίτητο, για τους αλγορίθμους εξωτερικών σημείων πάντοτε να κρατούν επαφή με την εφικτή περιοχή, ως εκ τούτου τα επόμενα δύο Θεωρήματα είναι πολύ σημαντικά.

Θεώρημα 6 *Αν* $x_{ij} < 0$ *τότε* $\exists\ d_{ij} > 0$ *τέτοιο ώστε* $\frac{x_{ij}}{-d_{ij}} < \alpha$

Απόδειξη: Η μέθοδος της επαγωγής θα χρησιμοποιηθεί στην απόδειξη. Στην 1η επανάληψη, $\nexists (i,j) : x_{ij}^{(1)} < 0$. Έστω ότι για το τόξο (i,j) στην t επανάληψη, αν $x_{ij}^{(t)} < 0 \Rightarrow \exists d_{ij}^{(t)} > 0 : \frac{x_{ij}^{(t)}}{-d_{ij}^{(t)}} < \alpha^t$. Θα δειχθεί ότι το (i,j) αν και στην $t+1$ επανάληψη διατηρεί $x_{ij}^{(t+1)} < 0 \Rightarrow \exists d_{ij}^{(t+1)} > 0 : \frac{x_{ij}^{t+1}}{-d_{ij}^{t+1}} < \alpha^{(t+1)}$.

Θα δειχθεί πρώτα ότι $\forall\ (i,j) : x_{ij}^{(t+1)} < 0,\ \exists\ d_{ij}^{(t+1)} > 0$ και έπειτα ότι είναι τέτοιο ώστε $\frac{x_{ij}^{(t+1)}}{-d_{ij}^{(t+1)}} \leq \alpha^{(t+1)}$. Για να εξακριβώσουμε την καταυθυντήρια ροή $d_{ij}^{(t+1)}$, εξετάζουμε όλους τους πιθανούς συνδυασμούς, που αντιστοιχούν σε οποιοδήποτε τύπο επανάληψης. Η κατηγοριοποίηση των ανανεώσεων των βασικών τόξων είναι του Σχήματος 2.3, θα χρησιμοποιηθεί σε αυτό το Θεώρημα.

Διερεύνηση περιπτώσεων για το (i,j)*, από τον Πίνακα* **2.1**

- *(1η Περίπτωση).* Τότε $d_{ij}^{(t+1)} = d_{ij}^{(t)} > 0$.

- *(2η και 3η Περίπτωση).* Σε αυτές τις περιπτώσεις αφού $(i,j) \notin T^{(t)}$ (είναι το αμέσως επόμενο εισερχόμενο τόξο) ισχύει $x_{ij}^{(t)} = 0$. Όμως εμείς υποθέσαμε ότι $x_{ij}^{(t)} < 0$. Ως εκ τούτου αυτές οι περιπτώσεις έρχονται σε αντίφαση με την υπόθεση μας και δεν θα εξεταστούν.

- *(4η Περίπτωση).* Τότε

$$x_{ij}^{(t+1)} < 0 \Rightarrow x_{ij}^{(t)} + x_{kl}^{(t)} < 0 \Rightarrow x_{kl}^{(t)} < -x_{ij}^{(t)} \tag{1}$$

Όμως σύμφωνα με την επαγωγική υπόθεση,

$$\frac{x_{ij}^{(t)}}{-d_{ij}^{(t)}} < \frac{x_{kl}^{(t)}}{-d_{kl}^{(t)}} \overset{\cdot(-d_{ij}^{(t)} d_{kl}^{(t)})>0}{\rightarrow} \underbrace{d_{kl}^{(t)} x_{ij}^{(t)}}_{>0} < \underbrace{x_{kl}^{(t)} d_{ij}^{(t)}}_{>0} \tag{2}$$

Πολλαπλασιάζοντας κατά μέλη τις Εξισώσεις 1 και 2, παίρνουμε

$$x_{kl}^{(t)} d_{kl}^{(t)} x_{ij}^{(t)} < -x_{ij}^{(t)} x_{kl}^{(t)} d_{ij}^{(t)} \overset{\div(-x_{kl}^{(t)} x_{ij}^{(t)}>0)}{\rightarrow} -d_{kl}^{(t)} < d_{ij}^{(t)} \Rightarrow d_{ij}^{(t)} + d_{kl}^{(t)} > 0 \Rightarrow d_{ij}^{(t+1)} > 0.$$

- *(5η Περίπτωση).* Σύμφωνα με την υπόθεση ισχύει:

$$x_{ij}^{(t)} < 0 \Rightarrow d_{ij}^{(t)} > 0 \tag{3}$$

Για το (k,l) στην t επανάληψη, ισχύει:

$$d_{kl}^{(t)} < 0 \Rightarrow -d_{kl}^{(t)} > 0 \tag{4}$$

Προσθέτοντας κατά μέλη τις Εξισώσεις 3 και 4, παίρνουμε $d_{ij}^{(t)} - d_{kl}^{(t)} > 0 \Rightarrow d_{ij}^{(t+1)} > 0$.

Άρα σε κάθε περίπτωση $\exists d_{ij}^{(t+1)} > 0$. Αρκεί να δειχτεί και ότι το $d_{ij}^{(t+1)}$, είναι τέτοιο ώστε $\frac{x_{ij}^{(t+1)}}{-d_{ij}^{(t+1)}} < \frac{x_{kl}^{(t+1)}}{-d_{kl}^{(t+1)}}$. Αυτό θα γίνει σε δύο βήματα. Πρώτα, θα δειχθεί ότι $\frac{x_{ij}^{(t+1)}}{-d_{ij}^{(t+1)}} < \frac{x_{kl}^{(t)}}{-d_{kl}^{(t)}}$.

Διερεύνηση περιπτώσεων για το (i,j), από τον Πίνακα **2.1**

- *(1η Περίπτωση).* Τότε, σύμφωνα με την επαγωγική υπόθεση, ισχύει: $\frac{x_{ij}^{(t+1)}}{-d_{ij}^{(t+1)}} = \frac{x_{ij}^{(t)}}{-d_{ij}^{(t)}} < \frac{x_{kl}^{(t)}}{-d_{kl}^{(t)}} = \alpha^{(t)}$.

- *(2η και 3η Περίπτωση).* Σε αυτές τις περιπτώσεις αφού $(i,j) \notin T^{(t)}$ (είναι το αμέσως επόμενο εισερχόμενο τόξο) ισχύει $x_{ij}^{(t)} = 0$. Όμως εμείς υποθέσαμε ότι $x_{ij}^{(t)} < 0$. Ως εκ τούτου αυτές οι περιπτώσεις έρχονται σε αντίθεση με την υπόθεση μας και δεν θα εξεταστούν.

- *(4η και 5η Περίπτωση).* Σύμφωνα με την επαγωγική υπόθεση, ισχύει:

$$\frac{x_{ij}^{(t)}}{-d_{ij}^{(t)}} < \frac{x_{kl}^{(t)}}{-d_{kl}^{(t)}} \overset{\cdot(-d_{kl}^{(t)} d_{ij}^{(t)})>0}{\rightarrow} \underbrace{d_{kl}^{(t)} x_{ij}^{(t)}}_{>0} < \underbrace{d_{ij}^{(t)} x_{kl}^{(t)}}_{>0} \overset{\div(-x_{kl}^{(t)})>0}{\rightarrow} d_{ij}^{(t)} > d_{kl}^{(t)} \frac{x_{ij}^{(t)}}{x_{kl}^{(t)}} \tag{5}$$

Επίσης, ισχύει:

$$\frac{x_{ij}^{(t+1)}}{-d_{ij}^{(t+1)}} = \frac{\overbrace{-(x_{ij}^{(t)} \pm x_{kl}^{(t)})}^{>0}}{\underbrace{d_{ij}^{(t)} \pm d_{kl}^{(t)}}_{>0}} \tag{6}$$

Οπότε αντικαθιστώντας την ποσότητα $d_{ij}^{(t)}$, σύμφωνα με την Σχέση 5, στη Σχέση 6 προκύπτει ότι:

$\frac{x_{ij}^{(t+1)}}{-d_{ij}^{(t+1)}} < \frac{-(x_{ij}^{(t)} \pm x_{kl}^{(t)})}{d_{kl}^{(t)} \frac{x_{ij}^{(t)}}{x_{kl}^{(t)}} \pm d_{kl}^{(t)}} \Rightarrow \frac{x_{ij}^{(t+1)}}{-d_{ij}^{(t+1)}} < \frac{-(x_{ij}^{(t)} \pm x_{kl}^{(t)})}{\frac{d_{kl}^{(t)} x_{ij}^{(t)} \pm d_{kl}^{(t)} x_{kl}^{(t)}}{x_{kl}^{(t)}}} \Rightarrow \frac{x_{ij}^{(t+1)}}{-d_{ij}^{(t+1)}} < \frac{\frac{-(x_{ij}^{(t)} \pm x_{kl}^{(t)})}{1}}{\frac{d_{kl}^{(t)}(x_{ij}^{(t)} \pm x_{kl}^{(t)})}{x_{kl}^{(t)}}} \Rightarrow \frac{x_{ij}^{(t+1)}}{-d_{ij}^{(t+1)}} < \frac{-x_{kl}^{(t)}(x_{ij}^{(t)} \pm x_{kl}^{(t)})}{d_{kl}^{(t)}(x_{ij}^{(t)} \pm x_{kl}^{(t)})} \overset{\div (x_{ij}^{(t)} \pm x_{kl}^{(t)}) \neq 0}{\rightarrow} \frac{x_{ij}^{(t+1)}}{-d_{ij}^{(t+1)}} < \frac{x_{kl}^{(t)}}{-d_{kl}^{(t)}} \Rightarrow \frac{x_{ij}^{(t+1)}}{-d_{ij}^{(t+1)}} < \alpha^t$. Ισχύει ότι $x_{ij}^{(t)} < 0$ και $x_{kl}^{(t)} > 0$. Οπότε, $x_{ij}^{(t)} \neq x_{kl}^{(t)} \Rightarrow x_{ij}^{(t)} - x_{kl}^{(t)} \neq 0$. Επιπλέον όμως δεχόμαστε ότι πρέπει να ισχύει και $x_{ij}^{(t)} \neq -x_{kl}^{(t)} \Rightarrow x_{ij}^{(t)} + x_{kl}^{(t)} \neq 0$, ειδάλλως για την 4η περίπτωση όπου $x_{ij}^{(t+1)} = x_{ij}^{(t)} + x_{kl}^{(t)}$, θα ισχύει ότι $x_{ij}^{(t+1)} = 0$, άρα δεν ικανοποιεί την υπόθεση της t επανάληψης ότι δηλαδή διατηρεί $x_{ij}^{(t+1)} < 0$.

Για να ολοκληρωθεί η απόδειξη, μένει να δειχθεί ότι $\alpha^t < \alpha^{t+1}$. Στην t επανάληψη, για οποιοδήποτε $(i,j):\ d_{ij}^{(k)} < 0$ έχουμε ότι $x_{ij}^{(t)} > 0$, σύμφωνα με την εναντίωση της υπόθεσης μας.

Διερεύνηση περιπτώσεων για το (i,j), από τον Πίνακα **2.1**

- *(1η Περίπτωση).* Σύμφωνα με την επαγωγική υπόθεση, ισχύει: $\frac{x_{ij}^{(t+1)}}{-d_{ij}^{(t+1)}} = \frac{x_{ij}^{(t)}}{-d_{ij}^{(t)}} \Rightarrow \frac{x_{ij}^{(t+1)}}{-d_{ij}^{(t+1)}} > \alpha^{(t)}$.

- *(2η και 3η Περίπτωση).* Σε αυτές τις περιπτώσεις αφού $(i,j) \notin T^{(t)}$ (είναι το αμέσως επόμενο εισερχόμενο τόξο) ισχύει $x_{ij}^{(t)} = 0$. Όμως εμείς υποθέσαμε ότι $x_{ij}^{(t)} < 0$. Ως εκ τούτου αυτές οι περιπτώσεις έρχονται σε αντίφαση με την υπόθεση μας και δεν θα εξεταστούν.

- *(4η και 5η Περίπτωση).* Βάσει του ορισμού του α ισχύει ότι:

$$\frac{x_{ij}^{(k)}}{-d_{ij}^{(k)}} > \alpha^k \tag{7}$$

Η Σχέση 7 μπορεί να γίνει:

$$\frac{x_{ij}^{(t)}}{-d_{ij}^{(t)}} > \frac{x_{kl}^{(t)}}{-d_{kl}^{(t)}} \overset{\cdot(-d_{kl}^{(t)} d_{ij}^{(t)}) < 0}{\rightarrow} d_{kl}^{(t)} x_{ij}^{(t)} < d_{ij}^{(t)} x_{kl}^{(t)} \overset{\div d_{kl}^{(t)} < 0}{\rightarrow} \underbrace{x_{ij}^{(t)}}_{>0} > \underbrace{x_{kl}^{(t)} \frac{d_{ij}^{(t)}}{d_{kl}^{(t)}}}_{>0} \tag{8}$$

Επίσης ισχύει:

$$\frac{x_{ij}^{(t+1)}}{-d_{ij}^{(t+1)}} = \frac{x_{ij}^{(t)} \pm x_{kl}^{(t)}}{-(d_{ij}^{(t)} \pm d_{kl}^{(t)})} \tag{9}$$

Οπότε με αντικατάσταση της ποσότητας $x_{ij}^{(t)}$ από τη Σχέση 8 στη Σχέση 9, έχουμε ότι: $\frac{x_{ij}^{(t+1)}}{-d_{ij}^{(t+1)}} = \frac{x_{ij}^{(t)} \pm x_{kl}^{(t)}}{-(d_{ij}^{(t)} \pm d_{kl}^{(t)})} \Rightarrow \frac{x_{ij}^{(t+1)}}{-d_{ij}^{(t+1)}} > \frac{x_{kl}^{(t)} \frac{d_{ij}^{(t)}}{d_{kl}^{(t)}} \pm x_{kl}^{(t)}}{-(d_{ij}^{(t)} \pm d_{kl}^{(t)})} \Rightarrow \frac{x_{ij}^{(t+1)}}{-d_{ij}^{(t+1)}} > \frac{\frac{x_{kl}^{(t)} d_{ij}^{(t)} \pm x_{kl}^{(t)} d_{kl}^{(t)}}{d_{kl}^{(t)}}}{-(d_{ij}^{(t)} \pm d_{kl}^{(t)})} \Rightarrow \frac{x_{ij}^{(t+1)}}{-d_{ij}^{(t+1)}} > \frac{\frac{x_{kl}^{(t)}(d_{ij}^{(t)} \pm d_{kl}^{(t)})}{d_{kl}^{(t)}}}{\frac{-(d_{ij}^{(t)} \pm d_{kl}^{(t)})}{1}} \Rightarrow \frac{x_{ij}^{(t+1)}}{-d_{ij}^{(t+1)}} > \frac{x_{kl}^{(t)}(d_{ij}^{(t)} \pm d_{kl}^{(t)})}{-d_{kl}^{(t)}(d_{ij}^{(t)} \pm d_{kl}^{(t)})} \overset{\div(d_{ij}^{(t)} \pm d_{kl}^{(t)}) \neq 0}{\rightarrow} \frac{x_{ij}^{(t+1)}}{-d_{ij}^{(t+1)}} > \frac{x_{kl}^{(t)}}{d_{kl}^{(t)}}$ ή $\frac{x_{ij}^{(t+1)}}{-d_{ij}^{(t+1)}} > \alpha^t$. Είναι $d_{ij}^{(t)},\ d_{kl}^{(t)} < 0$, άρα $d_{ij}^{(t)} + d_{kl}^{(t)} < 0$. Επιπλέον ισχύει και ότι $d_{ij}^{(t)} \neq d_{kl}^{(t)} \Rightarrow d_{ij}^{(t)} - d_{kl}^{(t)} \neq 0$. Σε αντίθετη περίπτωση, στην 5η περίπτωση, θα ίσχυε ότι $d_{ij}^{(t+1)} = d_{ij}^{(t)} - d_{kl}^{(t+1)} \Rightarrow d_{ij}^{(t+1)} = 0$, οπότε έρχεται σε αντίθεση με την υπόθεση μας, ότι και στην $t+1$ επανάληψη το τόξο (i,j) διατηρεί $d_{kl}^{(t+1)} < 0$.

Αφού $\forall (i,j)$ που διατηρεί και στην $t+1$ επανάληψη $d_{kl}^{(t+1)} < 0$, ισχύει ότι $\frac{x_{ij}^{(t+1)}}{-d_{ij}^{(t+1)}} > \alpha^t$, θα ισχύει και για το ελάχιστο αυτών των λόγων. Άρα σύμφωνα με τον ορισμό του α θα ισχύει ότι $\alpha^{t+1} > \alpha^t$. Συνδυάζοντας ότι $\frac{x_{ij}^{(t+1)}}{-d_{ij}^{(t+1)}} < \alpha^t$ και $\alpha^t < \alpha^{t+1}$, προκύπτει τελικά ότι $\frac{x_{ij}^{(t+1)}}{-d_{ij}^{(t+1)}} < \alpha^{t+1}$ και αυτό ολοκληρώνει την απόδειξη. ♣

Θεώρημα 7 *Ισχύει ότι* $\beta = max\left\{\frac{x_{ij}}{-d_{ij}} : x_{ij} < 0\right\} < \alpha$.

Αφού το Θεώρημα 6 ισχύει $\forall (i,j) : x_{ij} < 0$, θα ισχύει επίσης για το $max\left\{\frac{x_{ij}^{(t+1)}}{-d_{ij}^{(t+1)}}\right\} \Rightarrow \beta < \alpha$. ♣

Το επόμενο Θεώρημα 8, είναι απαραίτητο για την ανίχνευση της απεριοριστίας ενός προβλήματος από τον αλγόριθμο NEPSA. Για την ακρίβεια, όταν δεν είναι δυνατόν να υπολογιστεί μια βελτιώνουσα κατεύθυνση και το πρόβλημα μας δεν έχει φτάσει στη βέλτιστη λύση, τότε αυτό σημαίνει ότι το πρόβλημα είναι απεριόριστο.

Θεώρημα 8 *Άν* $d(T) \geq 0$ *και* $P \neq \emptyset$, *τότε το πρόβλημα είναι απεριόριστο.*

Απόδειξη Στην περίπτωση που ισχύει $d(T) = 0 \Rightarrow \sum_{(i,j)\in P} h_{ij} = 0$. Οπότε τα μη βασικά τόξα, που εκφράζονται βάσει αυτών των στηλών, είναι γραμμικώς εξαρτημένα. Συνεπώς, αυτά τα τόξα σχηματίζουν έναν κατευθυνόμενο κύκλο. Ως γνωστόν ισχύει $\sum_{(i,j)\in C} t_{ij}c_{ij} = \sum_{(i,j)\in C} t_{ij}s_{ij}$. Αφού ο C αποτελεί κατευθυνόμενο κύκλο, $t_{ij} = 1, \forall (i,j) \in C$. Οπότε, $\sum_{(i,j)\in C} c_{ij} = \sum_{(i,j)\in C} s_{ij} \Rightarrow \sum_{(i,j)\in C} c_{ij} = S_o \Rightarrow \sum_{(i,j)\in C} c_{ij} < 0$. Επομένως, ο κύκλος C έχει αναγνωριστεί ως αρνητικού κόστους κατευθυνόμενος και το πρόβλημα μας είναι απεριόριστο.

Στην περίπτωση που ισχύει $d(T) \geq 0$, ας υποθέσουμε ότι $\exists (t_1, t_2) \in T : d_{t_1t_2} = k > 0$ και $d_{ij} = 0, \forall (i,j) \in T - \{(t_1, t_2)\}$. Δίχως χάσιμο της γενικότητας θα εξετάσουμε την περίπτωση στην οποία υπάρχει μόνο ένα στοιχείο που έχει θετική τιμή d. Επίσης, για ένα διάνυσμα c, θα συμβολίζουμε με c^T το ανάστροφο διάνυσμα. Τότε:

$$d(T) = [0, 0, ..., k, ..., 0, 0]^T \Leftrightarrow \sum_{(i,j)\in P} h_{ij} = [0, 0, ..., -k, ..., 0, 0]^T \ k > 0$$

Σε αυτή την περίπτωση ένας κατευθυνόμενος κύκλος έχει δημιουργηθεί αποτελούμενος όχι μόνο από τόξα του συνόλου P. Για την ακρίβεια έχουν δημιουργηθεί k κύκλοι, διαφορετικοί μεταξύ τους. Στους κύκλους αυτούς συμμετέχουν και βασικά τόξα (t_1, t_2). Αυτοί οι κύκλοι αποτελούν ένα μεγαλύτερο διευρυμένο κύκλο. Το τόξο (t_1, t_2) συμμετέχει σε κάθε έναν από αυτούς τους k κύκλους.

Κάθε βασικό τόξο εκφράζεται σε σχέση με τα βασικά τόξα, χρησιμοποιώντας μόνο τον εαυτό του. Ας υποθέσουμε ότι $(t_1, t_2) \in T$, τότε η στήλη $h_{t_1t_2}$ θα είναι $h_{t_1t_2} = [0, 0, ..., 1, ..., 0, 0]^T$. Οπότε,

$$\sum_{(i,j)\in P} h_{ij} + kh_{t_1t_2} = [0, 0, ..., -k, ..., 0, 0]^T + k \cdot [0, 0, ..., 1, ..., 0, 0]^T = [0, 0, ..., -k + k, ..., 0, 0]^T =$$
$$[0, 0, ..., 0, ..., 0, 0]^T$$

Συνεπώς, τα τόξα $(i,j) \in P$ με την προσθήκη k φορές το τόξο (t_1, t_2) είναι γραμμικώς εξαρτημένα και ως εκ τούτου σχηματίζουν k κατευθυνόμενους κύκλους. Ισχύει επίσης ότι $(t_1, t_2) \in T \Rightarrow s_{t_1t_2} = 0$. Οπότε, $\sum_{(i,j)\in P} s_{ij} + ks_{t_1t_2} = \sum_{(i,j)\in P} s_{ij} = S_o < 0$. Το προηγούμενο άθροισμα ισούται με το άθροισμα των κόστων των τόξων, που σχηματίζουν τους κύκλους.

Ολοκληρώνοντας, όλοι οι διαφορετικοί συνδυασμοί των τόξων, τα οποία σχηματίζουν ένα κύκλο, είναι αρνητικά προσανατολισμένοι. Με αυτόν τον τρόπο, το πρόβλημα μας είναι απεριόριστο. Παρ΄ όλα αυτά, για το συγκεκριμένο αλγόριθμο μένει να αποδειχθεί ότι εάν $\exists(t_1,t_2) \in T : x_{t_1t_2} < 0 \Rightarrow (t_1,t_2) \in C$. Αυτό πρέπει να αποδειχθεί, διότι σε αντίθετη περίπτωση, δεν έχει σημασία πόση ροή θα στείλουμε δια μέσου του κύκλου ώστε να ελαχιστοποιηθεί η τιμή της αντικειμενικής συνάρτησης, αφού η λύση θα ήταν αδύνατη, διότι θα ίσχυε ότι $x_{ij} < 0$. Τότε το πρόβλημα θα ήταν αδύνατο και όχι απεριόριστο.

Άν $\exists(t_1,t_2) \in T : x_{t_1t_2} < 0$, τότε σύμφωνα με το Θεώρημα 6 θα είναι $d_{t_1t_2} > 0$. Οπότε, το τόξο (t_1,t_2) συμμετέχει στους προαναφερθέντες αρνητικού κόστους προσανατολισμένους κύκλους. ♣

Στην τελευταία επανάληψη οι δύο ροές προκύπτει ότι ταυτίζονται. Δηλαδή, η βασική ροή x ισούται με τη ροή y. Ως εκ τούτου, είναι σημαντικό να αποδειχθεί ότι η ροή x παραμένει εφικτή σε όλη τη διάρκεια της λειτουργίας του αλγορίθμου. Ακόμη, το επόμενο Θεώρημα 9, είναι απαραίτητο γιατι διασφαλίζει ότι ο αλγόριθμος NEPSA διατηρεί επαφή με το εσωτερικό της εφικτής περιοχής.

Θεώρημα 9 *Η ροή y είναι πάντοτε εφικτή*

Απόδειξη: Πρέπει να αποδειχθεί ότι $y \geq 0 \Leftrightarrow x + ad \geq 0 \Leftrightarrow x \geq -ad$. Οπότε αυτή η σχέση πρέπει να δειχτεί ότι ισχύει για κάθε τόξο, ή αλλιώς:

$$\forall(i,j) : x_{ij} \geq -ad_{ij} \tag{10}$$

Διερεύνηση περιπτώσεων για το (i,j), από τον Πίνακα **2.1**

- *(1η Περίπτωση).* Άν $d_{ij} < 0$, τότε σύμφωνα με το Θεώρημα 6 ισχύει $x_{ij} \geq 0$. Οπότε η Σχέση 10 γίνεται $x_{ij} \geq -ad_{ij} \overset{\div(-d_{ij})>0}{\rightarrow} \frac{x_{ij}}{-d_{ij}} \geq a$ το οποίο ισχύει σύμφωνα με τον ορισμό του a.

- *(2η Περίπτωση).* Άν $d_{ij} > 0$, τότε υπάρχουν οι εξής υποπεριπτώσεις:

  - Άν $x_{ij} \geq 0$, τότε αφού $a,\ d_{ij} > 0$, ισχύει ότι $y \geq 0$

- Άν $x_{ij} < 0$, τότε η Σχέση 10 γίνεται $x_{ij} \geq -ad_{ij} \overset{\div(-d_{ij})<0}{\rightarrow} \frac{x_{ij}}{-d_{ij}} \leq a$. Ισχύει σύμφωνα με το Θεώρημα 6.

- *(3η Περίπτωση).* Άν $d_{ij} = 0$, σύμφωνα με το Θεώρημα 6 $x_{ij} \geq 0$. Οπότε, $y = x + ad \overset{d=0}{\rightarrow} y = x \Rightarrow y \geq 0$

Τελευταίο, αλλά όχι λιγότερης σημασίας, παρουσιάζεται το επόμενο Θεώρημα 10. Συγκεκριμένα, το Θεώρημα 10, είναι αρκετά σημαντικό γιατι μας δίνει μια ικανή και αναγκαία συνθήκη τερματισμού για τον αλγορίθμο NEPSA.

Πιο συγκεκριμένα, όταν όλα τα μη βασικά τόξα είναι δυϊκώς εφικτά, (δηλαδή ισχύει ότι $s_{ij} \geq 0$, $\forall (i,j) \notin T$ και επομένως το σύνολο P έχει γίνει κενό), τότε αυτό συνεπάγεται ότι ο αλγόριθμος NEPSA έχει υπολογίσει το βέλτιστο δένδρο - ροή.

Θεώρημα 10 *Άν $P = \emptyset$, τότε ο αλγόριθμος NEPSA έχει υπολογίσει τη βέλτιστη ροή.*

Απόδειξη: Σε κάθε επανάληψη ισχύει ότι:

$$d = \{d(T), d(P), d(Q)\} = \left\{ d(T), \overbrace{1, 1, \ldots, 1}^{d(P)}, \overbrace{0, 0, \ldots, 0}^{d(Q)} \right\} \tag{11}$$

Σύμφωνα με τον ορισμό του $d(T)$, $\forall (i,j) \in T$, θα ισχύει:

$$-|P| \leq d_{ij} \leq |P| \tag{12}$$

Όταν θα γίνει $P = \oslash \Rightarrow |P| = 0$, η Σχέση 12 γίνεται:

$$d_{ij} = 0, \forall (i,j) \in T \Rightarrow d(T) = 0 \tag{13}$$

Η Σχέση 11 με τη βοήθεια της Σχέσης 13, γίνεται $d = \{0, 0, ..., 0\}$. Οπότε, αφού είναι $y = x + ad \overset{d=0}{\rightarrow} y = x$. Όμως σύμφωνα με το Θεώρημα 9, η ροή y είναι πάντοτε εφικτή, άρα θα ισχύει ότι $x \geq 0$.

Αφού λοιπόν η ροή $x(T)$ είναι πρωτευόντως εφικτή, αν συνδυάσει κανείς το γεγονός ότι ισχύει και $P = \emptyset$, προκύπτει ότι το βασικό δένδρο T είναι και δυϊκά εφικτό. Με αυτόν τον τρόπο, καταλήγουμε ότι το δένδρο T αποτελεί τη βέλτιστη λύση.

ΚΕΦΑΛΑΙΟ 4

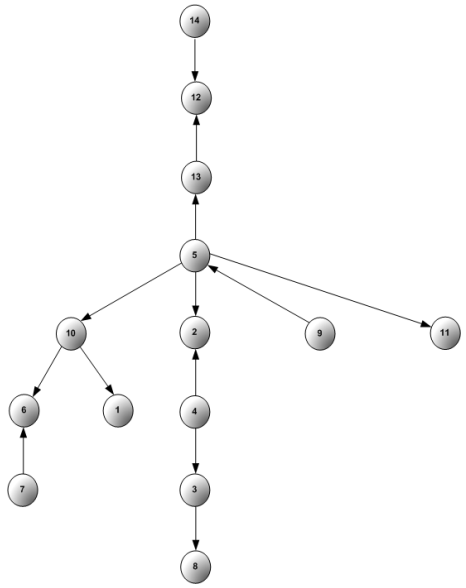
Προγραμματιστική Υλοποίηση και ενδεικτική υπολογιστική συμπεριφορά του NEPSA

4.1 Υλοποίηση του NEPSA & Δομές δεδομένων

Υπάρχουν πολλοί τρόποι αποθήκευσης των γράφων και δικτύων σε Η/Υ. Κανένας όμως τρόπος δεν υπερτερεί των υπολοίπων σε όλες τις περιπτώσεις. Η επιλογή του τρόπου αποθήκευσης εξαρτάται από τον αλγόριθμο που χρησιμοποιείται. Επειδή, αυτό μας ενδιέφερει είναι η γρήγορη και εύκολη ανανέωση του βασικού δένδρου διάσχισης, επιλέχθηκε η δομή δεδομένων που κάνει χρήση διανυσμάτων βάθους, κόμβων πατέρων, αλλά και της προδιάταξης, γνωστής και ως Augmented Thread Index, (ATI method), όπως περιγράφεται στην εργασία (Glover, Karney, and Klingman 1972). Η δομή αυτή επιλέχθηκε, διότι επιτρέπει τη γρήγορη ανανέωση του βασικού δένδρου, αλλά και την εύκολη εύρεση του κύκλου που θα δημιουργούνταν ανά επανάληψη με την είσοδο του εισερχομένου τόξου. Η μέθοδος αυτή αναπτύχθηκε περαιτέρω στις εργασίες (Glover, Klingman, and Stutz 1973), (Glover, Klingman, and Stutz 1974) και (Barr, Glover, and Klingman 1979), όπου η τελευταία μέθοδος αναφέρεται και ως Extended Thread Index, (XTI method). Η αποτελεσματικότητα της αναφέρεται σε διάφορες υπολογιστικές μελέτες, όπως για παράδειγμα στην εργασία (Glover, Karney, Klingman, and Napier 1974).

Θα υποθέσουμε ότι ο αλγόριθμος NEPSA ανανεώνει σε κάθε επανάληψη ένα δέντρο T. Στη συνέχεια αυτού του κεφαλαίου θα γίνει μια περιγραφή όλων των δομών δεδομένων που χρησιμοποιήθηκαν για την αναπαράσταση του βασικού δένδρου. Επιπλέον θα γίνει και μια λεπτομερή ανάλυση της ανανέωσης των.

Για την αποθήκευση ενός ριζωμένου δένδρου, μπορούμε να χρησιμοποιήσουμε ένα διάνυσμα p με n στοιχεία, ένα για κάθε κόμβο. Το στοιχείο $p(i)$ είναι ο πατέρας του κόμβου i. Αν r είναι η ρίζα του δέντρου, θέτουμε $p(r) = -1$. Για παράδειγμα, το διάνυσμα p του δέντρου του Σχήματος 4.1 είναι ίσο με $p =$ [10 5 4 2 13 10 6 3 5 5 5 14 12 -1].



Σχήμα 4.1 Περιγραφή της δομής υλοποίησης ATI

Αν το δέντρο περιέχει τόξα, τότε μόνο το διάνυσμα p δεν επαρκεί για την πλήρη περιγραφή του δέντρου. Πρέπει με ένα τρόπο να προσδιορίσουμε για κάθε κόμβο i εκτός της ρίζας αν το δέντρο περιέχει το τόξο $(p(i), i)$ ή το τόξο $(i, p(i))$. Θα χρησιμοποιήσουμε επομένως ένα επιπλέον διάνυσμα t, πάλι με n στοιχεία ένα για κάθε κόμβο, το οποίο θα προσδιορίζει την κατεύθυνση του κάθε τόξου. Δηλαδή αν το $(i, p(i))$ ανήκει στο δέντρο ή αν το $(p(i), i)$ ανήκει στο δέντρο. Αν r είναι η ρίζα του δέντρου, θέτουμε $t(r) = -1$. Για την ακρίβεια οι τιμές που θα έχει το διάνυσμα t

θα είναι ως εξής:

$$t(i) = \begin{cases} 1, \text{ αν } (i, p(i)) \text{ ανήκει στο δέντρο} \\ \\ 0, \text{ αν } (p(i), i) \text{ ανήκει στο δέντρο} \end{cases}$$

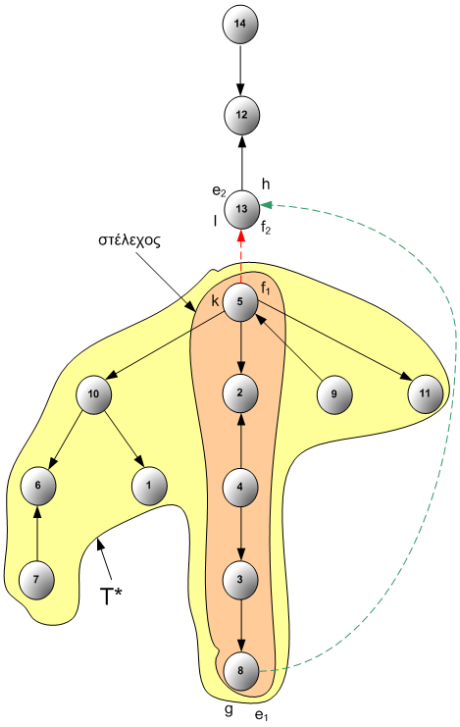
Για παράδειγμα, το διάνυσμα t του δέντρου του Σχήματος 4.1 είναι ίσο με t = [0 0 0 1 1 0 1 0 1 0 0 0 1 -1]. Παρόλα αυτά, μπορεί να γίνει και πιο αποτελεσματικός προγραμματισμός και να μην χρειαζόμαστε το διάνυσμα t. Προς το σκοπό αυτό, μπορούμε να τροποποιήσουμε το διάνυσμα p με την εξής αλλαγή. Αν r είναι η ρίζα του δέντρου, θα θέσουμε $p(r) = r$ (εννοείται ότι $\forall i \in N,\ p(i) \neq i$), ενώ για κάθε κόμβο i εκτός της ρίζας με $p(i) = j$, θέτουμε:

$$p(i) = \begin{cases} j, \text{ αν } (i, p(i)) \text{ ανήκει στο δέντρο} \\ \\ -j, \text{ αν } (p(i), i) \text{ ανήκει στο δέντρο} \end{cases}$$

Επιπλέον, θα χρησιμοποιήσουμε και την προδιάταξη (*preorder*) του ριζωμένου δένδρου με n στοιχεία, ένα για κάθε κόμβο. Για παράδειγμα, στο δέντρο του Σχήματος 4.1 είναι $preorder$ = [14 12 13 5 10 6 7 1 2 4 3 8 9 11]. Τέλος, θα κάνουμε χρήση ενός ακόμη διανύσματος με n στοιχεία ένα για κάθε κόμβο, το οποίο θα αποτελείται από τις τιμές βάθους *depth* καθενός κόμβου. Για παράδειγμα, στο δέντρο του Σχήματος 4.1 είναι $depth$ = [5 4 6 5 3 5 6 7 4 4 4 1 2 0]. Πριν αναφερθούμε στην ανανέωση των προαναφερθέντων διανυσμάτων, είναι απαραίτητο να επεξηγήσουμε ακόμη κάποιους όρους. Θα συμβολίζουμε T^*, εκείνο το υποδένδρο το οποίο μετά τη διαγραφή του εξερχόμενου τόξου δεν θα περιέχει τον κόμβο ρίζα. Το κυρίως υπολογιστικό φορτίο για την ανανέωση ενός βασικού δένδρου μπορεί να περιοριστεί μονάχα στο υποδένδρο T^*.

Έστω στο Σχήμα 4.2, ότι από το δένδρο T επιλέγεται ως εξερχόμενο το τόξο (k, l) = (5,13) και ως εισερχόμενο το τόξο (g, h) = (8,13). Τότε, το υποδένδρο, με ρίζα τον κόμβο 5, που αποκόπτεται από το δένδρο T, αποτελεί το T^*. Επίσης, με f_1 θα συμβολίζουμε όποιον από τους δύο κόμβους k

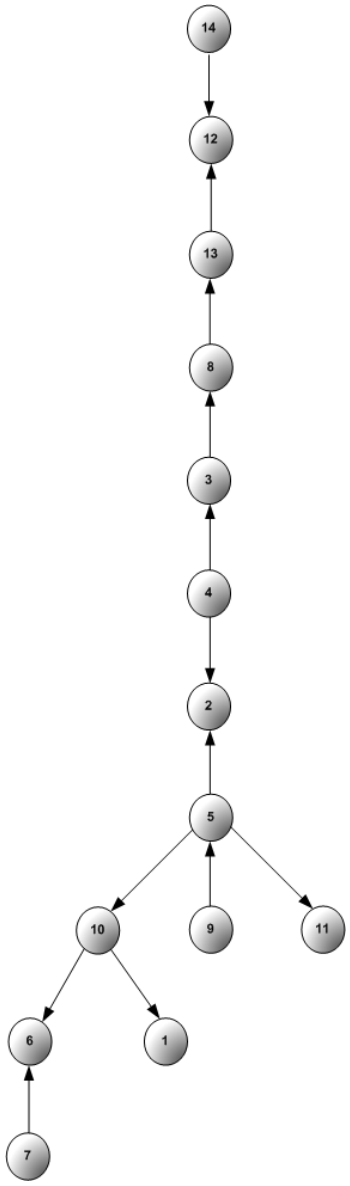
ή l, ανήκει στο T^* ενώ τον άλλο εκ αυτών των δύο, θα τον συμβολίζουμε με f_2. Ομοίως δε, με e_1 θα συμβολίζουμε όποιον από τους δύο κόμβους g ή h, ανήκει στο T^* ενώ τον άλλο εκ αυτών των δύο, θα τον συμβολίζουμε με e_2. Στο παράδειγμα μας, έχουμε ότι $f_1 = 5$, $f_2 = 13$, $e_1 = 8$, $e_2 = 13$, (είναι απλά θέμα σύμπτωσης ότι $f_2 = e_2 = 13$).



Σχήμα 4.2 Το υποδένδρο T^* και το στέλεχος

Επιπρόσθετα, θα ονομάζουμε *στέλεχος*, εκείνο το μονοπάτι του δένδρου T, που συνδέει τον κόμβο e_1 με τον κόμβο f_1. Έστω ότι u σε πλήθος κόμβοι βρίσκονται σε αυτό το μονοπάτι. Τότε με τη σειρά που τους συναντάμε, θα αποθηκεύονται σε ένα διάνυσμα z. Με αυτόν τον τρόπο, πάντοτε θα ισχύει $z_1 = e_1$ και $z_u = f_1$. Εν προκειμένοις, στο Σχήμα 4.2 έχουμε $u = 4$ και $z = [8\ 3\ 4\ 2\ 5]$. Μετά την περιστροφή, προκύπτει το βασικό δένδρο του Σχήματος 4.3. Τώρα πια είμαστε σε θέση να αναπτύξουμε τον τρόπο ανανέωσης όλων αυτών των απαιτούμενων δομών.

Εξετάζοντας προσεκτικά τα δένδρα T και T', παρατηρεί κανείς ότι αλλάζουν πατέρα μονάχα οι κόμβοι i, $i \in z$. Για την ακρίβεια, ο πρώτος κόμβος του στελέχους $z(1) = e_1$, αλλάζει πατέρα σε



Σχήμα 4.3 Το δένδρο T' μετά την περιστροφή

$p^{'}(z(1)) = e_1$. Οι υπόλοιποι κόμβοι $z(i),\ i > 1$ του στελέχους αλλάζουν πατέρα σε $p^{'}(z(i)) = z(i-1)$, όπου $p^{'}(i)$ είναι ο πατέρας του κόμβου i στο δέντρο $T^{'}$. Το ανανεωμένο διάνυσμα πατέρων συνεπώς είναι $p^{'}$ = [10 4 8 3 2 10 6 13 5 5 5 14 12 -1]. Η πολυπλοκότητα της ανανέωσης του διανύσματος πατέρων των βασικών τόξων p, είναι $\Theta(u)$, όπου u είναι το πλήθος των κόμβων που ανήκουν στο στέλεχος.

Ομοίως, μονάχα οι κόμβοι $i,\ i \in z$ αλλάζουν προσανατολισμό t. Πιο συγκεκριμένα, για τον πρώτο κόμβο του στελέχους $z(1) = e_1$, υπάρχουν δύο περιπτώσεις. Εάν $e_1 = h$, τότε το εισερχόμενο τόξο έχει προσανατολισμό προς τα κάτω και θα ισχύει ότι $t^{'}(z(1)) = 0$. Εάν πάλι ισχύει ότι $e_1 = g$, τότε το εισερχόμενο τόξο έχει προσανατολισμό προς τα πάνω οπότε τώρα θα έχουμε ότι $t^{'}(z(1)) = 1$. Για όλους τους υπόλοιπους κόμβους $i,\ i \in z$, με $i \neq 1$, η ανανέωση γίνεται ως εξής: $t^{'}(z(i)) = 0$, αν $t(z(i-1)) = 1$ και $t^{'}(z(i)) = 1$, αν $t(z(i-1)) = 0$. Το ανανεωμένο διάνυσμα προσανατολισμού των βασικών τόξων συνεπώς γίνεται $t^{'}$ = [0 0 1 1 1 0 1 1 1 0 0 0 1 -1]. Η πολυπλοκότητα της ανανέωσης του διανύσματος προσανατολισμού των βασικών τόξων t, είναι $\Theta(u)$, όπου u είναι το πλήθος των κόμβων που ανήκουν στο στέλεχος.

Για τις ανάγκες της παρουσίασης την υλοποίησης του διανύσματος $d(T)$, θα γίνει μια αλλαγή στο συμβολισμό. Πιο συγκεκριμένα αντί για το συμβολισμό d_{ij}, που εισήχθηκε στο κεφάλαιο 2 και αντιστοιχούσε σε ένα βασικό τόξο, θα χρησιμοποιήσουμε το συμβολισμό $d(i)$, το οποίο αντιστοιχεί στον κόμβο $i,\ i \in N$. Με αυτόν τον τρόπο όμως, η ποσότητα $d(i)$ αντιστοιχεί πραγματικά στο τόξο $(i, p(i))$ ή $(p(i), i)$, (αναλόγως ποιό τόξο ανήκει στο δένδρο). Το διάνυσμα $d(T)$ θα αποτελείται από n στοιχεία, ένα για κάθε κόμβο.

Ομοίως, θα προχωρήσουμε σε μια επιπλέον αλλαγή στο συμβολισμό όσον αφορά το διάνυσμα στήλη $h_{gh}(i, j)$, πάλι σε σχέση με το κεφάλαιο 2. Μέχρι τώρα το προηγούμενο διάνυσμα συμβόλιζε τον προσανατολισμό του τόξου (g, h), σε σχέση με τα βασικά τόξα (i, j). Το προηγούμενο διάνυσμα υλοποιείται (και τώρα πια έτσι θα συμβολίζεται) ως ένα διάνυσμα h_{gh}, όπου το i στοιχείο του $h_{gh}(i)$

θα δηλώνει τον προσανατολισμό του τόξου (g, h) σε σχέση με το τόξο $(i, p(i))$ ή $(p(i), i)$, (αναλόγως ποιό τόξο ανήκει στο δένδρο).

Για την ανανέωση λοιπόν του διανύσματος $d(T)$ εξετάζουμε δύο περιπτώσεις αναλόγως του τύπου της επανάληψης, A ή B. Έστω ότι για το δένδρο T έχουμε $d(T)$ = [-4 24 16 -20 -11 -17 8 11 -6 -28 7 0 -7]. Ας εξηγήσουμε όμως, με ποιόν τρόπο προκύπτει η τιμή d_{ij} κάθε τόξου (i, j) με ένα παράδειγμα για το στοιχείο $d(5)$ = -11. Ο πέμπτος κόμβος έχει $p(5) = 13$, άρα έχει πατέρα τον κόμβο 13. Επίσης, έχει $t(5) = 1$, επομένως έχει προσανατολισμό προς τα επάνω. Ως εκ τούτου λοιπόν αναφερόμαστε στο τόξο (5,13) το οποίο τυγχάνει να είναι το εξερχόμενο τόξο με d_{kl} = -11. Σε περίπτωση, όπου υπάρχει επανάληψη τύπου A, τότε εάν ο κόμβος i δεν ανήκει στο στέλεχος, υπάρχουν τρείς υποπεριπτώσεις.

- Άν $h_{gh}(i) = 1$, τότε $d(i) = d(i)$ - d_{kl}
- Άν $h_{gh}(i)$ = -1, τότε $d(i) = d(i) + d_{kl}$
- Άν $h_{gh}(i) = 0$, τότε $d(i) = d(i)$

Εάν πάλι, ο κόμβος i ανήκει στο στέλεχος, υπάρχουν ξανά τρείς υποπεριπτώσεις.

- Άν $h_{gh}(z(i-1)) = 1$, τότε $d(z(i)) = d(z(i-1))$ - d_{kl}
- Άν $h_{gh}(z(i-1))$ = -1, τότε $d(z(i)) = d(z(i-1)) + d_{kl}$
- Άν $h_{gh}(z(i-1)) = 0$, τότε $d(z(i)) = d(z(i))$

Επιπλέον θα έχουμε για το εισερχόμενο τόξο ότι:

$d(e_1) = d_{kl} + 1$

Σε περίπτωση όπου έχουμε επανάληψη τύπου B, τότε εάν ο κόμβος i δεν ανήκει στο στέλεχος, υπάρχουν τρείς υποπεριπτώσεις.

- Άν $h_{gh}(i) = 1$, τότε $d(i) = d(i) + d_{kl}$
- Άν $h_{gh}(i) =$ -1, τότε $d(i) = d(i)$ - d_{kl}
- Άν $h_{gh}(i) = 0$, τότε $d(i) = d(i)$

Εάν πάλι, ο κόμβος i ανήκει στο στέλεχος, υπάρχουν ξανά τρείς υποπεριπτώσεις.

- Άν $h_{gh}(z(i-1)) = 1$, τότε $d(z(i)) = d(z(i-1)) + d_{kl}$
- Άν $h_{gh}(z(i-1)) =$ -1, τότε $d(z(i)) = d(z(i-1))$ - d_{kl}
- Άν $h_{gh}(z(i-1)) = 0$, τότε $d(z(i)) = d(z(i))$

Επιπλέον, θα έχουμε για το εισερχόμενο τόξο ότι:

$d(e_1) =$ - d_{kl}

Όπως διαπιστώνεται με απλό τρόπο, η πολυπλοκότητα της ανανέωσης του κατευθυντήριου διανύσματος ροής d, είναι $\Theta(n)$. Για παράδειγμα, στο δέντρο του Σχήματος 4.1 είναι $d =$ [-4 24 16 -20 -11 -17 8 11 -6 -28 7 0 -7]. Η επανάληψη αυτή είναι τύπου Α και ισχύει ότι $d_{kl} =$ -11. Άρα το ανανεωμένο διάνυσμα κατεύθυντήριας ροής d, θα είναι πλέον $d' =$ [-4 -9 0 5 13 -17 8 -10 -6 -28 7 0 -7].

Το διάνυσμα ροής υλοποιείται (και τώρα πια έτσι θα συμβολίζεται) ως ένα διάνυσμα x, με συνιστώσες $x(i)$ το οποίο αντιστοιχεί στον κόμβο i, $i \in N$. Με αυτόν τον τρόπο όμως, η ποσότητα $x(i)$ αντιστοιχεί πραγματικά στο τόξο $(i, p(i))$ ή $(p(i), i)$, (αναλόγως ποιό τόξο ανήκει στο δένδρο). Το διάνυσμα $x(T)$ θα αποτελείται από n στοιχεία, ένα για κάθε κόμβο. Για την ανανέωση του διανύσματος $x(T)$, θα χρησιμοποιήσουμε λοιπόν αυτόν τον συμβολισμό και όχι το συμβολισμό x_{ij} που εισήχθηκε στην ενότητα 1.3 και αντιστοιχούσε σε ένα βασικό τόξο (i, j).

Για την ανανέωση του διανύσματος κάθε στοιχείου $x(i)$ του διανύσματος ροής x εξετάζουμε δύο περιπτώσεις αναλόγως εάν ο κόμβος i ανήκει ή όχι στο στέλεχος. Ακόμη, αναλόγως του τύπου της

περιστροφής έχουμε διοχέτευση ροής ίση με x_{kl} (τύπου Α) ή - x_{kl} (τύπου Β), στον κύκλο C όπως περιγράφεται στο κεφάλαιο 2. Πριν περιγράψουμε την ανανέωση του διανύσματος x, θα κάνουμε χρήση του παρακάτω συμβολισμού.

$$\epsilon = \begin{cases} x_{kl,}, \tau\upsilon\pi ου\ A \\ -x_{kl}, \tau\upsilon\pi ου\ B \end{cases}$$

Οπότε, εάν ο κόμβος i ανήκει στο στέλεχος έχουμε

- Άν $h_{gh}(z(i-1))$ = -1, τότε $x(z(i)) = x(z(i-1)) + \epsilon$
- Άν $h_{gh}(z(i-1)) = 1$, τότε $x(z(i)) = x(z(i-1))$ - ϵ

καθώς επίσης και $x(e_1) = \epsilon$

Ενώ, εάν ο κόμβος i δεν ανήκει στο στέλεχος έχουμε

- Άν $h_{gh}(i)$ = -1, τότε $x(i) = x(i) + \epsilon$
- Άν $h_{gh}(i) = 1$, τότε $x(i) = x(i)$ - ϵ
- Άν $h_{gh}(i) = 0$, τότε $x(i) = x(i)$

Όπως φαίνεται εύκολα η πολυπλοκότητα της ανανέωσης της βασικής ροής είναι $\Theta(n)$. Για παράδειγμα, στο δέντρο του Σχήματος 4.1 είναι x = [253 -584 -237 544 282 465 0 -226 177 718 -20 0 282]. Η επανάληψη αυτή είναι τύπου Α και επομένως ϵ = 282. Άρα το ανανεωμένο διάνυσμα βασικής ροής θα είναι πλέον $x^{'}$ = [253 262 56 45 -302 465 0 282 177 718 -20 0 282].

Παρατηρήθηκε στην πράξη, κατά την εκτέλεση διαφόρων στιγμιοτύπων ΠΡΕΚ, ότι είναι προτιμότερο να ανανεώνουμε τις δυϊκές μεταβλητές w και έπειτα να υπολογίζουμε τα ελλειματικά κόστη s. Η άλλη μέθοδος απευθείας ανανέωσης του διανύσματος s, η οποία περιγράφτηκε στην ενότητα 2 δεν είναι υπολογιστικά συμφέρουσα. Αυτό συμβαίνει διότι, με τη τελευταία μέθοδο πρέπει m φορές

(για κάθε ένα τόξο) να γίνεται έλεγχος εάν η ουρά ή η κεφαλή κάθε μη βασικού τόξου ανήκει στο υποδένδρο T^*. Άρα η πολυπλοκότητα αυτής της μεθόδου είναι $\Theta(m * u)$, όπου u είναι το πλήθος των κόμβων που ανήκουν στο στέλεχος.

Αυτή μέθοδος είναι αποδοτικότερη, διότι ανανεώνουμε μονάχα τις τιμές w_i για τους κόμβους που ανήκουν στο υποδένδρο T^*, δηλαδή n φορές, ενώ εκτελούμε και m φορές τον υπολογισμό των s_{ij}. Άρα η πολυπλοκότητα αυτής της μεθόδου είναι $\Theta(n + m)$. Επειδή, όμως είναι σε κάθε περίπτωση $m > n$, ενώ επίσης $u \geq 1$, αποδεικνύεται ότι $\Theta(n + m) < \Theta(m * u)$.

Σύμφωνα με το Θεώρημα 1 της ακεραιότητας, αν όλα τα αρχικά δεδομένα του ΠΡΕΚ αποτελούν ακέραιους αριθμούς, τότε ο NEPSA, όπως άλλωστε και ο NPSA, δεν χρειάζεται να χρησιμοποιεί πραγματικούς αριθμούς (κινητής υποδιαστολής) κατά τους αριθμητικούς υπολογισμούς. Μπορεί, λοιπόν κάλλιστα, να δουλέψει με ακέραιους αριθμούς και συνεπώς να γλυτώσει απο κάποια αριθμητικά σφάλματα. Επιπλέον, οι πράξεις με ακεραίους αριθμούς εκτελούνται γρηγορότερα σε έναν Η/Υ από ότι οι πράξεις με αριθμούς κινητής υποδιαστολής.

Παρ΄ όλα αυτά υπάρχουν κάποιες υλοποιήσεις, οι οποίες επιτρέπουν τη βασική ροή x να παίρνει πραγματικές τιμές, (Grigoriadis 1984), δίχως μεγάλη καθυστέρηση στο χρόνο εκτέλεσης. Αυτό ισχύει διότι η βασική ροή x παίρνει μέρος σε λίγες μόνο πράξεις στο *Βήμα 0* (το οποίο εκτελείται μια φορά αρχικά) και στα *Βήματα* 2 και 4. Εξάλλου, σύμφωνα με διάφορες υπολογιστικές μελέτες π.χ. (Mulvey 1978), το κυρίως υπολογιστικό φορτίο εκτελείται στο *Βήμα 3*, όπου η ροή x δεν παίρνει καθόλου μέρος στους υπολογισμούς. Για την ακρίβεια, στο *Βήμα 3* πρέπει η μήτρα κόμβων - τόξων να σαρωθεί για την εύρεση κατάλληλου εισερχόμενου τόξου, ή την ανίχνευση ότι τέτοιο δεν υπάρχει. Αυτό απαιτεί για κάθε μη βασικό τόξο (i, j), τον υπολογισμό του ελλειματικού του κόστους s_{ij} αλλά και της αντίστοιχης στήλης $h(i, j)$.

4.2 Μία μέθοδος για την αντιμετώπιση των εκφυλισμένων προβλημάτων στον αλγόριθμο NEPSA

Παρ΄ ότι υπάρχει μια πληθώρα αλγορίθμων για την επίλυση του ΠΡΕΚ, παρ΄ όλα αυτά ο πιό γνωστός αλγόριθμος γι΄ αυτήν την οικογένεια προβλημάτων είναι ο κλασικός αλγόριθμος NPSA. Ο NPSA σε κάποιες περιπτώσεις - στιγμιότυπα προβλημάτων δεν τερματίζεται επιτυχώς, εξαιτίας ενός μεγάλου αριθμού εκφυλισμένων επαναλήψεων. Ένα βασικό δένδρο - λύση ονομάζεται πρωτευόντως *εκφυλισμένο*, εάν η ροή ενός τουλάχιστον τόξου ισούται με μηδέν. Ένα στιγμιότυπο ενός ΠΡΕΚ θα καλείται εκφυλισμένο, εάν έχει ένα τουλάχιστον εκφυλισμένο βασικό δένδρο. Μια εκφυλισμένη περιστροφή συμβαίνει όταν μετά από μια περιστροφή η τιμή της αντικειμενικής συνάρτησης δεν βελτιώνεται, αλλά παραμένει σταθερή. Όταν το πρόβλημα είναι εκφυλισμένο, η τιμή της αντικειμενικής συνάρτησης παραμένει ίδια κατά τη διάρκεια διαδοχικών επαναλήψεων. Αυτό το φαινόμενο μπορεί να οδηγήσει τον NPSA σε *κύκλωση*, (cycling).

Η κύκλωση συμβαίνει όταν ο αλγόριθμος ξεκινάει με ένα αρχικό βασικό δένδρο και έπειτα από κάποιες επαναλήψεις ο αλγόριθμος υπολογίζει πάλι το ίδιο βασικό δένδρο - λύση. Αυτή η διαδικασία μπορεί να επαναλαμβάνεται συνεχώς, με άμεσο επακόλουθο ο αλγόριθμος να αποτυγχάνει να τερματίσει. Σύντομα μετά την ανάπτυξη του αλγορίθμου Simplex, η πιθανότητα της κύκλωσης αναγνωρίστηκε από πολλούς ερευνητές, (Hoffman 1953) και (Beale 1955). Το φαινόμενο της κύκλωσης σε αλγόριθμο τύπου Simplex για δίκτυα, παρατηρήθηκε για πρώτη φορά στην εργασία (Gassner 1964) για το πρόβλημα της μεταφοράς. Από τότε έχει παρατηρηθεί και σε άλλου είδους προβλήματα, όπως για παράδειγμα στο πρόβλημα της ανάθεσης (Gavish, Schweitzer, and Shlifer 1977), αλλά και γενικότερα σε προβλήματα ροών δικτύων (Cunningham and Klincewicz 1983). Το πρόβλημα της κύκλωσης παραμένει πολύ σημαντικό μέχρι και σήμερα, αφού σύγχρονα άρθρα αφιερώνονται αποκλειστικά στο πρόβλημα αυτό, όπως για παράδειγμα οι εργασίες (Gassa and Vin-

jamurib 2004) και (Hall and McKinnon 2004). Επιπλέον δε, ακόμη και στην περίπτωση όπου δεν εμφανιστεί κύκλωση, μπορεί να συμβούν εκθετικά μεγάλες ακολουθίες εκφυλισμένων περιστροφών (το φαινόμενο της *στασιμότητας* ή stalling) με τον αλγόριθμο NPSA όπως αποδείχθηκε από άλλους ερευνητές (Cunningham 1979).

Για την αποφυγή του φαινομένου της κύκλωσης έχουν αναπτυχθεί διάφορες τεχνικές. Μια μέθοδος είναι η σωστή επιλογή του εξερχόμενου τόξου, όταν υπάρχει εκφυλισμένο δένδρο, όπως αναφέρεται στις εργασίες (Cunningham 1976) όπου εισάγεται για πρώτη φορά ο όρος του ισχυρά εφικτού δένδρου, ή άλλες μέθοδοι όπως στην εργασία (Barr, Glover, and Klingman 1977) . Παρ΄ όλα αυτά, υπάρχουν και άλλες μέθοδοι οι οποίες απαιτούν κατάλληλη επιλογή τόσο του εισερχόμενου όσο και του εξερχόμενου τόξου, (Bland 1977). Τέλος, έχουν ανακαλυφθεί ακόμη *παθολογικά* προβλήματα, (Zadeh 1973a) και (Zadeh 1973b), στα οποία ο κλασικός αλγόριθμος NPSA υποπίπτει σε κύκλωση. Μέχρι σήμερα, δεν έχει αποδειχθεί αν οι αλγόριθμοι εξωτερικών σημείων για το ΠΡΕΚ υποπίπτουν στο φαινόμενο της κύκλωσης. Σε αυτή τη Διατριβή, παρουσιάζουμε για πρώτη φορά, κάποια προκαταρκτικά αποτελέσματα βασισμένα στην υπολογιστική συμπεριφορά του NEPSA.

Υπάρχει περίπτωση ακόμη και με την επιλογή εξερχόμενου σύμφωνα με τη Σχέση 6 ή και εισερχόμενου τόξου σύμφωνα με τις Σχέσεις 7 και 8, ο αλγόριθμος να έχει να επιλέξει μεταξύ δύο ή και περισσοτέρων ισοδύναμων τόξων. Αυτό το φαινόμενο το ονομάζουμε *δεσμό* (tie). Ο αλγόριθμος NEPSA προγραμματίστηκε με κατάλληλο τρόπο ώστε να *σπαέι* τους δεσμούς με τον εξής τρόπο. Όλα τα τόξα αριθμήθηκαν κατά μια διάταξη και πάντοτε επιλέγονταν, μεταξύ ισοδύναμων εξερχομένων ή εισερχομένων, εκείνο το τόξο με τη μικρότερη διάταξη. Η μέθοδος αυτή είναι αντίστοιχη με τον κανόνα του Bland (Bland 1977) για το γενικό γραμμικό πρόβλημα. Πάντως, πρέπει να σημειωθεί ότι είναι δυνατόν να κατασκευαστούν γραμμικά προβλήματα που οδηγούν σε κύκλωση, ακόμη και με τη χρήση διαφόρων κανόνων περιστροφής (Zörnig 2006).

Με τη χρήση της προαναφερθείσας μεθόδου ο NEPSA έλυσε όλα τα προβλήματα και τερματίστη-

κε επιτυχώς. Επιπλέον δε, εξετάζοντας τις ενδιάμεσες λύσεις των προβλημάτων, που υπολογίζονταν σε κάθε επανάληψη, προκύπτει ότι η τιμή της αντικειμενικής συνάρτησης δεν παρέμεινε αμετάβλητη για πολλές διαδοχικές επαναλήψεις (*στασιμότητα*), ούτε από την άλλη μεριά παρατηρήθηκε κάποια μακρά ακολουθία κάποιων τόξων να εξέρχονται και να εισέρχονται στο βασικό δένδρο επανειλλημένα. Από προκαταρκτικά ερευνητικά αποτελέσματα πάνω σε τυχαία αραιά προβλήματα (Paparrizos, Samaras, and Sifaleras 2006c), ο αλγόριθμος NEPSA φαίνεται να μην υποπίπτει στο φαινόμενο της κύκλωσης. Παρ΄ όλα αυτά είναι σίγουρο πως τα πειράματα πρέπει να συνεχιστούν ούτως ώστε να εξαχθεί με ασφάλεια αυτό το συμπέρασμα.

Το φαινόμενο της κύκλωσης μπορεί να αποτραπεί, κατασκευάζοντας έναν κανόνα με τη βοήθεια του οποίου θα γίνεται η επιλογή για το πως θα *σπάνε* οι δεσμοί κατά τον έλεγχο του ελαχίστου λόγου για την επιλογή του εξερχομένου καθώς και του εισερχομένου τόξου. Τα τελευταία χρόνια έχουν εμφανιστεί διάφορες μέθοδοι στη διεθνή βιβλιογραφία για την εξάλειψη των εκφυλισμένων περιστροφών, ούτως ώστε να αποτραπεί το φαινόμενο της κύκλωσης και της στασιμότητας. Μεταξύ αυτών είναι και η εργασία (Maros 1993). Παρ΄ όλα αυτά η κύκλωση αποτελεί σπάνιο φαινόμενο στην πράξη ακόμη και σε ισχυρά εκφυλισμένα προβλήματα. Ως εκ τούτου, οι επιπρόσθετοι υπολογισμοί που απαιτούνται για την αντιμετώπιση της κύκλωσης πολλές φορές είναι μη αποτελεσματικοί.

4.3 Γεννήτριες προβλημάτων Δικτυακής Βελτιστοποίησης

Όλες οι υπολογιστικές μελέτες διεξάγονται με μεγάλης διαστάσεως προβλήματα. Απαραίτητο για το σκοπό αυτό είναι και η ύπαρξη γεννητριών τέτοιων προβλημάτων, όπως για παράδειγμα η ευρέως χρησιμοποιούμενη γεννήτρια NETGEN (Klingman, Napier, and Stutz 1974). Η NETGEN κατασκευάζει απλά προβλήματα δικτύων (pure network problems), με ή χωρίς, άνω χωρητικότητες, καθώς επίσης και προβλήματα μεταφοράς ή αντιστοίχισης. Είναι γραμμένη τόσο σε γλώσσα Fortran 77 όσο και σε C και όπως και όλες οι γεννήτριες, δέχονται κάποια λίστα παραμέτρων. Ο χρήστης

μεταξύ άλλων παραμέτρων, (για λόγους συντομίας δεν τις αναφέρουμε όλες σε αυτό το σημείο) δηλώνει το πλήθος των κόμβων συνολικά NODES, των κόμβων προσφοράς NTSORC, των κόμβων ζήτησης NSINK, των κόμβων μεταφόρτωσης από το σύνολο των κόμβων προσφοράς NTSORC, των κόμβων μεταφόρτωσης από το σύνολο των κόμβων ζήτησης NTSINK και τη συνολίκη ποσότητα προϊόντων που ρέουν ITSUP. Ο χρήστης κατασκευάζει στιγμιότυπα ΠΡΕΚ, εκτός αν ισχύει ότι NSORC + NSINK = NODES και NTSORC = NTSINK = 0 οπότε κατασκευάζεται στιγμιότυπο προβλήματος μεταφοράς. Τέλος, η NETGEN κατασκευάζει στιγμιότυπο προβλήματος ανάθεσης, όταν ισχύουν οι δύο προηγούμενες συνθήκες αλλά και επιπλέον NSORC = NSINK και ITSUP = NSORC. Αυτός ο τρόπος λειτουργίας της ανωτέρω γεννήτριας, εξηγείται διότι το πρόβλημα της μεταφοράς αποτελεί εξειδίκευση του ΠΡΕΚ, ενώ το πρόβλημα της ανάθεσης αποτελεί περαιτέρω εξειδίκευση του πρόβληματος της μεταφοράς.

Επιπλέον, υπάρχει η γεννήτρια GNETGEN η οποία αποτελεί τροποποίηση της NETGEN και αναπτύχθηκε από τον ερευνητή Fred Glover για την κατασκευή γενικευμένων προβλημάτων ροής, (generalized network problems). Αυτή η γεννήτρια χρησιμοποιεί αριθμητική διπλής ακρίβειας για περισσότερο μεταφέρσιμο (portable) κώδικα, ενώ μία υπολογιστική μελέτη με προβλήματα δημιουργημένα με τη συγκεκριμένη γεννήτρια παρουσιάζεται στην εργασία (Clark, Kennington, Meyer, and Ramamurti 1992). Για την ακρίβεια στο γενικευμένο ΠΡΕΚ, ή αλλιώς και (the flow with gains model), επιπρόσθετα του απλού ΠΡΕΚ, σε κάθε τόξο αντιστοιχεί και ένας πολλαπλασιαστής της ροής. Στο απλό ΠΡΕΚ, η τιμή αυτού του πολλαπλασιαστή ισούται με μονάδα.

Ακόμη υπάρχει η γεννήτρια MESH, γραμμένη σε γλώσσα C, η οποία παράγει minimum - cost circulation προβλήματα και η γεννήτρια Grid-On-TOrus (GOTO) (Goldberg 1991), επίσης γραμμένη σε γλώσσα C, η οποία παράγει στιγμιότυπα προβλημάτων μεταφοράς με χωρητικότητες.

Επιπλέον υπάρχει η γεννήτρια GRIDGRAPH (Resende and Veiga 1993), γραμμένη σε γλώσσα Fortran 77, η οποία παράγει προβλήματα μεγίστης ροής όσο και προβλήματα ροής ελαχίστου

κόστους. Η συγκεκριμένη γεννήτρια κάνει χρήση επιπλέον δυο υπορουτίνων Fortran 77. Η η πρώτη υπορουτίνα περιγράφεται στην εργασία (Goldfarb and Grigoriadis 1988) και χρησιμοποιείται για τον υπολογισμό μεγίστης ροής, ενώ η δεύτερη περιγράφεται στην εργασία (Schrage 1979) και αποτελεί γεννήτρια τυχαίων αριθμών.

Κάθε γεννήτρια δέχεται ώς δεδομένα είσοδου μια λίστα τιμών διαφόρων παραμέτρων και εξάγει ένα κατευθυνόμενο γράφημα μαζί με τα κόστη και τις χωρητικότητες για κάθε τόξο καθώς και την προσφορά ή ζήτηση προϊόντος για κάθε κόμβο. Αυτές όλες οι γεννήτριες καθώς και κάποιοι κώδικες από λύτες διατίθενται δωρεάν για ακαδημαϊκή - μη εμπορική χρήση με ανώνυμο ftp από την ηλεκτρονική διεύθυνση ftp://dimacs.rutgers.edu/pub/netflow. Πρέπει όμως, για λόγους ασφαλείας του ερευνητικού κέντρου DIMACS, κάποιος να ανακτήσει τα αρχεία, αφού πρώτα δηλώσει την συγκεκριμένη IP διεύθυνση του, ώστε να αποκτήσει δικαίωμα εισόδου. (Morris 2004).

Τα στιγμιότυπα ΠΡΕΚ, τα οποία δημιουργηθήκαν και χρησιμοποιήθηκαν σε αυτή τη Διατριβή, κατασκευάστηκαν με τη γεννήτρια GRIDGEN (Lee and Orlin 1991), η οποία προγραμματίστηκε στη γλώσσα C. Στη συνέχεια, θα περιγραφεί ο τρόπος χρήσης της και θα εξηγηθεί αναλυτικά η δημιουργία των στιγμιοτύπων που χρησιμοποιήθηκαν στην υπολογιστική μελέτη. Ο χρήστης αφού δημιουργήσει ένα εκτελέσιμο αρχείο της GRIDGEN, το καλεί έπειτα μέσω της γραμμής εντολών ως εξής:

gridgen batchfile

Το αρχείο με όνομα batchfile ουσιαστικά καθορίζει το πλήθος, τα ονόματα καθώς και τα χαρακτηριστικά όλων των δεδομένων εξόδου / στιγμιοτύπων ΠΡΕΚ. Η λίστα των παραμέτρων που μπορεί να δεχθεί είναι η παρακάτω:

1. *number of jobs*, (Είναι δυνατόν να κατασκευαστούν ταυτοχρόνως περισσότερα από ένα στιγμιότυπα ΠΡΕΚ).

2. *output file name*, (Όνομα για το αρχείο εξόδου που θα περιέχει τα δεδομένα του στιγμιοτύπου).

3. *no two-way arcs*, (Εισάγουμε 1 εάν επιθυμούμε τόξα με ταυτόχρονα δύο κατευθύνσεις - ακμή, αλλιώς 0 για κατευθυνόμενα τόξα).

4. *random number seed*, (8 ψήφιος θετικός αριθμός για την αρχικοποίηση της γεννήτριας τυχαίων αριθμών, πρέπει να περιλαμβάνει τουλάχιστον ένα μη μηδενικό αριθμό),

5. *number of nodes*, (Συνολικός αριθμός κόμβων).

6. *grid width*, (Πλάτος πλέγματος).

7. *number of source nodes*, (Συνολικός αριθμός κόμβων προσφοράς).

8. *number of sink nodes*, (Συνολικός αριθμός κόμβων ζήτησης).

9. *average degree*, (Μέσος βαθμός κάθε κόμβου).

10. *total flow*, (Συνολική ποσότητα προσφοράς).

11. *distribution of arc costs, 1 for UNIFORM*, (Κατανομή τιμών κόστους των τόξων, 1 για ομοιόμορφη).

12. *1st parameter for arc cost (float): lower bound*, (Ελάχιστη τιμή κόστους για κάθε τόξο).

13. *2nd parameter for arc cost (float): upper bound*, (Μέγιστη τιμή κόστους για κάθε τόξο).

14. *distribution of arc capacities, 1 for UNIFORM*, (Κατανομή τιμών χωρητικότητας των τόξων, 1 για ομοιόμορφη).

15. *1st parameter for arc capacity (float): lower bound*, (Ελάχιστη τιμή κάτω χωρητικότητας).

16. *2nd parameter for arc capacity (float): upper bound*, (Μέγιστη τιμή άνω χωρητικότητας).

Αυτή η γεννήτρια δημιουργεί ένα δίκτυο σε μορφή *πλέγματος*, με επιπλέον έναν *υπέρ* κόμβο. Επιπρόσθετα των τόξων που συνδέουν τους κόμβους στο πλέγμα, υπάρχει ένα τόξο από κάθε κόμβο προσφοράς στον υπέρ κόμβο και από τον υπέρ κόμβο προς κάθε κόμβο ζήτησης, ούτως ώστε να εγγυάται η εφικτότητα. Πάντως, αυτά τα τελευταία τόξα έχουν πολύ μεγάλη τιμή κόστους και πολύ μεγάλη τιμή χωρητικότητας. Ο τρόπος λειτουργίας αυτής της γεννήτριας έχει ως εξής, πρώτον κατασκευάζεται ένα πλέγμα διαστάσεων n1 * n2. Για παράδειγμα, 5 * 3. Οι κόμβοι αριθμούνται από το 1 ως το 15, ενώ ο υπερ κόμβος αριθμείται ως n1*n2+1. Έπειτα, δημιουργούνται τα τόξα που ενώνουν τους γειτονικούς κόμβους. Γι΄ αυτά τα τόξα, ο χρήστης επιτρέπεται να καθορίσει εάν θα έχουν φορά και προς τις δυο κατευθύνσεις ή μόνο προς μία. Εάν καθοριστεί να δημιουργηθούν τόξα με φορά και προς τις δύο κατευθύνσεις τότε, δύο τόξα, ένα προς κάθε κατεύθυνση, θα δημιουργηθούν μεταξύ κάθε ζεύγους γειτονικών κόμβων. Ειδάλλως, μονάχα ένα τόξο θα κατασκευαστεί.

Στη συνέχεια θα προστεθούν στο δίκτυο τα τόξα που ενώνουν τον υπέρ κόμβο με τους κόμβους προσφοράς / ζήτησης όπως αναφέρθηκε προηγουμένως. Σε περίπτωση όπου πάλι ο αριθμός των τόξων δεν είναι ο απαιτούμενος, τότε προστίθενται τόξα με ομοιόμορφη επιλογή τυχαίων ζευγαριών κόμβων. Οι κόμβοι προσφοράς και ζήτησης επιλέγονται ομοιόμορφα από το δίκτυο και η προσφορά / ζήτηση σε κάθε κόμβο ανατίθενται και πάλι με ομοιόμορφη κατανομή. Ο χρήστης ορίζει την πυκνότητα του δικτύου, με τον εξής τρόπο. Δηλώνει το μέσο βαθμό v κάθε κόμβου $i \in N$, $i = \{1, 2, ..., n\}$, οπότε το σύνολο των τόξων που θα κατασκευαστούν είναι ίσο με περίπου nv.

4.4 Μορφή προβλημάτων της υπολογιστικής μελέτης

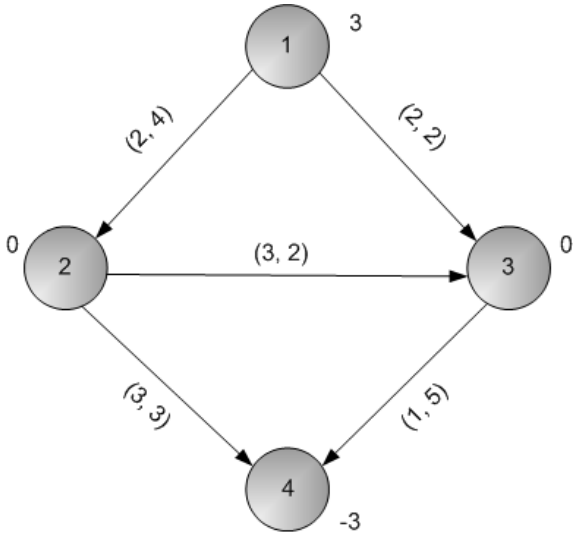
Όλες οι γεννήτριες παράγουν κάποια στιγμιότυπα ΠΡΕΚ σε αρχείο κειμένου, με διάφορες καταλήξεις αναλόγως την υλοποίηση. Υπάρχουν όμως τρία εδραιωμένα πρότυπα για τη μορφή του

παραγόμενου αρχείου. Το πρώτο είναι η μορφή DIMACS, (πήρε την ονομασία απο workshop που διοργανώθηκε στο ερευνητικό κέντρο DIMACS), το δεύτερο είναι το πρότυπο RELAX4, (πήρε την ονομασία απο το γνωστό αλγόριθμο RELAX4) και τρίτο είναι το πρότυπο NETFLO, (πήρε την ονομασία απο το γνωστό αλγόριθμο NETFLO). Η κάθε γεννήτρια συνήθως παράγει αρχείο κάποιας μόνο απο τις τρεις προαναφερθέντες μορφές. Ο αλγόριθμος NEPSA προγραμματίστηκε έτσι ώστε να δέχεται αρχείο με τα δεδομένα εισόδου σε μορφή RELAX4. Η επιλογή αυτή έγινε, ώστε σε κάποια μελλοντική βελτίωση του NEPSA να μπορεί να συγκριθεί με εύκολο τρόπο με τον αλγόριθμο RELAX4, (αφού και οι δύο είναι προγραμματισμένοι σε γλώσσα Fortran και χρησιμοποιούν ίδιου τύπου αρχεία εισόδου).

Αξίζει να αναφερθεί ότι η γεννήτρια GRIDGEN εξάγει αρχεία της μορφής DIMACS και όχι της μορφής RELAX4. Υπάρχουν βέβαια προγράμματα που αναλαμβάνουν την μετατροπή ενός αρχείου μιας μορφής σε αρχείο κάποιας άλλης μορφής, ένα - ένα κάθε φορά. Παρ΄ όλα αυτά προτιμήθηκε να τροποποιηθεί από εμάς ο κώδικας της GRIDGEN, ούτως ώστε να εξάγονται ταυτόχρονα πολλαπλά στιγμιότυπα ΠΡΕΚ, όλα της μορφής RELAX4.

Σε αυτό το σημείο, θα δοθεί ένα παράδειγμα αποθήκευσης δικτύου σε μορφή RELAX4. Έστω ότι είχαμε το δίκτυο του Σχήματος 4.4, το οποίο αποτελείται από τέσσερις κόμβους και πέντε τόξα. Δίπλα από κάθε τόξο φαίνεται ένα ζευγάρι αριθμών, εντός παρενθέσων. Ο πρώτος αριθμός συμβολίζει το κόστος, ενώ ο δεύτερος συμβολίζει την άνω χωρητικότητα του εκάστοτε τόξου.

Τότε ένα τυπικό αρχείο της μορφής RELAX4, που περιγράφει το δίκτυο του Σχήματος 4.4, θα έχει την παρακάτω δομή:



Σχήμα 4.4 Παράδειγμα αποθήκευσης δικτύου σε αρχείο κειμένου τύπου RELAX4

```
4   5
1   2   3   4
1   3   2   2
2   3   1   2
2   4   3   3
3   4   1   5
3
0
0
-3
```

Η πρώτη γραμμή του αρχείου εισόδου σε μορφή RELAX4 πρέπει να περιέχει τον αριθμό των κόμβων n καθώς και τον αριθμό των τόξων m. Οι επόμενες m γραμμές πρέπει να περιέχουν τα τόξα του δικτύου περιγεγραμμένα στην εξής σειρά. Πρώτα, γράφεται ο κόμβος - ουρά i του τόξου, έπειτα ο κόμβος - κεφαλή j του τόξου,, στη συνέχεια το κόστος c_{ij} που αντιστοιχεί στο τόξο, την

άνω χωρητικότητα u_{ij} του τόξου, (η κάτω χωρητικότητα l_{ij} κάθε τόξου θεωρείται ότι είναι μηδέν). Οι εναπομείναντες n γραμμές περιέχουν την προσφορά του καθενός κόμβου (μια αρνητική τιμή παριστάνει ζήτηση). Τέλος, η αρίθμηση των κόμβων θεωρείται ότι είναι από 1 έως n και επιπλέον τα δεδομένα θεωρούνται ότι αποτελούνται από ακέραιες τιμές.

Αξίζει να αναφερθεί ότι το ίδιο δίκτυο σε μορφή DIMACS θα είχε την παρακάτω δομή:

```
c
c  Ένα απλό παράδειγμα παρουσίασης της μορφής DIMACS
c  DIMACS μορφή αρχείου εισόδου για ΠΡΕΚ.
c
c  γραμμή προβλήματος :
p  min  4  5
c
c  γραμμές περιγραφής των κόμβων :
n  1   4
n  3   -3
c
c  γραμμές περιγραφής των τόξων :
a  1  2  0  4  3
a  1  3  0  2  2
a  2  3  0  2  1
a  2  4  0  3  3
a  3  4  0  5  1
```

Στη μορφή αυτή ο χρήστης μπορεί να εισάγει σχόλια σε γραμμές που αρχίζουν με το χαρακτήρα c. Αυτές οι γραμμές αγνοούνται από το κυρίως πρόγραμμα. Σχόλια είναι δυνατόν να τοποθετηθούν σε οποιοδήποτε σημείο του αρχείου. Σε κάθε αρχείο μορφής DIMACS υπάρχει μια γραμμή με την περιγραφή του προβλήματος. Αυτή η γραμμή εμφανίζεται πριν από οποιαδήποτε περιγραφή κόμβου ή τόξου. Όσον αφορά ένα στιγμιότυπο ενός ΠΡΕΚ η γραμμή ξεκινάει με το χαρακτήρα p και περιέχει τη λέξη min, αφού το ΠΡΕΚ είναι πρόβλημα ελαχιστοποίησης της τιμής της αντικειμενικής συνάρτησης. Έπειτα στη γραμμή αυτή, βρίσκει κανείς το συνολικό αριθμό (ακέραιες μεταβλητές) των κόμβων και έπειτα των τόξων. Στη συνέχεια, ακολουθούν οι γραμμές περιγραφής των κόμβων. Αυτές οι γραμμές ξεκινούν με το χαρακτήρα n, μετά υπάρχει η ετικέτα του κάθε κόμβου και τέλος η προσφορά ή η ζήτηση του συγκεκριμένου κόμβου. Πρέπει να σημειωθεί, ότι οι κόμβοι μεταφόρτωσης δεν περιλαμβάνονται σε αυτές τις γραμμές περιγραφής των κόμβων. Δηλαδή, θεωρείται ότι ισχύει $b_i = 0,\ \forall$ για κάθε κόμβο i που δεν υπάρχει περιγραφή για αυτόν. Τέλος, ακολουθούν οι γραμμές περιγραφής των τόξων. Αυτές οι γραμμές ξεκινούν με το χαρακτήρα a, ενώ μετά για κάθε τόξο αναγράφεται ο κόμβος ουρά, ο κόμβος κεφαλή, η κάτω χωρητικότητα, η άνω χωρητικότητα και τέλος η αντίστοιχη τιμή του κόστους του.

4.5 Υπολογιστική συμπεριφορά του NEPSA

Όλες οι κλάσεις στιγμιοτύπων ΠΡΕΚ εκτελέστηκαν χρησιμοποιώντας ένα προσωπικό υπολογιστή (PC) ο οποίος έχει τα ακόλουθα κύρια χαρακτηριστικά. Αποτελείται από επεξεργαστή Intel P4, Prescott 560 series, 3.6 GHz, μνήμη RAM 1 GB τεχνολογίας DDR 2, με λειτουργικό Windows XP Professional, με εγκατεστημένο το Service Pack 2. Επίσης, χρησιμοποιήθηκε ο compiler για γλώσσα Fortran 95/2003 της εταιρίας Intel, της εκδόσεως 9.1.033.

Επιπλέον, το τελικό εκτελέσιμο αρχείο της υπολογιστικής μελέτης, κατασκευάστηκε με τις ακόλουθες ρυθμίσεις στον Intel Visual Fortran compiler:

- *Optimization*: Maximize Speed plus Higher Level optimizations
- *Global Optimization*: Yes
- *Favor Size or Speed*: Favor Fast Code
- *Optimize for Intel(R) Processors*: Pentium(R) 4 and Additional Intel Processors
- *Use Intel(R) Processor Extensions*: Intel Pentium(R) 4 Processor with Streaming (SIMD) Extensions 3 (SSE3)
- *Parallelization*: Yes

Για τις ανάγκες της παρούσας υπολογιστικής μελέτης, δημιουργήθηκαν τέσσερις κλάσεις προβλημάτων. Μία με πυκνότητα 2,5%, μια με 5,0%, μια με 10,0% και τέλος μια με 0,1%. Κάθε μια κλάση από τις πρώτες τρείς κλάσεις περιλαμβάνει 6 κατηγορίες προβλημάτων, με διαφορετικές διαστάσεις. Οι κόμβοι και στις τρείς κλάσεις, αρχίζουν από τους 500 και φτάνουν μέχρι και τους 1000, με βήμα 100, (έτσι δημιουργούνται οι έξι προαναφερθείσες κατηγορίες). Όσον αφορά όμως το πλήθος των τόξων, αυτό προκύπτει ανάλογα με την κλάση όπου ανήκει το στιγμιότυπο. Επιπλέον δε, σε κάθε μια κατηγορία έχουν δημιουργηθεί 10 στιγμιότυπα, ώστε να εξάγεται ο μέσος όρος των επαναλήψεων, καθώς και του συνολικού χρόνου, από πιο αντιπροσωπευτικό δείγμα.

Η τέταρτη κλάση προβλημάτων αποτελείται από τρείς κατηγορίες προβλημάτων, με διαφορετικές διαστάσεις. Ο αριθμός των κόμβων στην πρώτη κατηγορία είναι 5.000, στη δεύτερη είναι 7.500 και τέλος η τρίτη κατηγορία περιέχει 10.000 κόμβους. Ομοίως με τις προηγούμενες τρείς κλάσεις, ο αριθμός των τόξων εξαρτάται από την πυκνότητα της κλάσης. Επιπρόσθετα δε, σε κάθε μία από αυτές τις τρείς κατηγορίες έχουν κατασκευαστεί από πέντε προβλήματα, έτσι ώστε να υπολογιστεί ο μέσος αριθμός των επαναλήψεων αλλά και ο συνολικός χρόνος. Συνοψίζοντας, συνολικά έχουν κατασκευαστεί και επιλυθεί $3 * 6 * 10 + 1 * 3 * 5 = 195$ στιγμιότυπα ΠΡΕΚ.

Στις επόμενες Ενότητες 4.5.1, 4.5.2, 4.5.3 και 4.5.4 παρουσιάζονται συγκεντρωμένα τα υπολογιστικά αποτελέσματα. Συγκεκριμένα, στην Ενότητα 4.5.1 παρουσιάζονται τα αποτελέσματα για προβλήματα πυκνότητας 2,5 %, στην Ενότητα 4.5.2 παρουσιάζονται τα αποτελέσματα για προβλήματα πυκνότητας 5 %, στην Ενότητα 4.5.3 παρουσιάζονται τα αποτελέσματα για προβλήματα πυκνότητας 10 % και τέλος στην Ενότητα 4.5.4 παρουσιάζονται τα αποτελέσματα για προβλήματα πυκνότητας 0,1 %.

4.5.1 Προβλήματα πυκνότητας 2,5 %

Στον Πίνακα **4.1**, φαίνονται όλοι οι παράμετροι που χρησιμοποιήθηκαν για την κατασκευή των στιγμιοτύπων ΠΡΕΚ που πήραν μέρος στην συγκεκριμένη κλάση προβλημάτων.

Πυκνό--τητα	ακμές (two-way arcs)	κόμβοι	πλάτος πλέγματος	κόμβοι προσφοράς	κόμβοι ζήτησης	βαθμός κόμβου	συνολική ποσότητα	κατανομή κόστους, 1 για ομοιόμορφη	κάτω όριο κόστους	άνω όριο κόστους
	1	500	100	250	150	12	100.000	1	1	100
	1	600	100	300	200	15	200.000	1	1	100
2,5%	1	700	100	350	250	17	300.000	1	1	100
	1	800	100	400	300	20	400.000	1	1	100
	1	900	100	400	300	22	400.000	1	1	100
	1	1.000	100	500	400	25	400.000	1	1	100

Πίνακας **4.1** Παράμετροι της γεννήτριας GRIDGEN για προβλήματα πυκνότητας 2,5 %

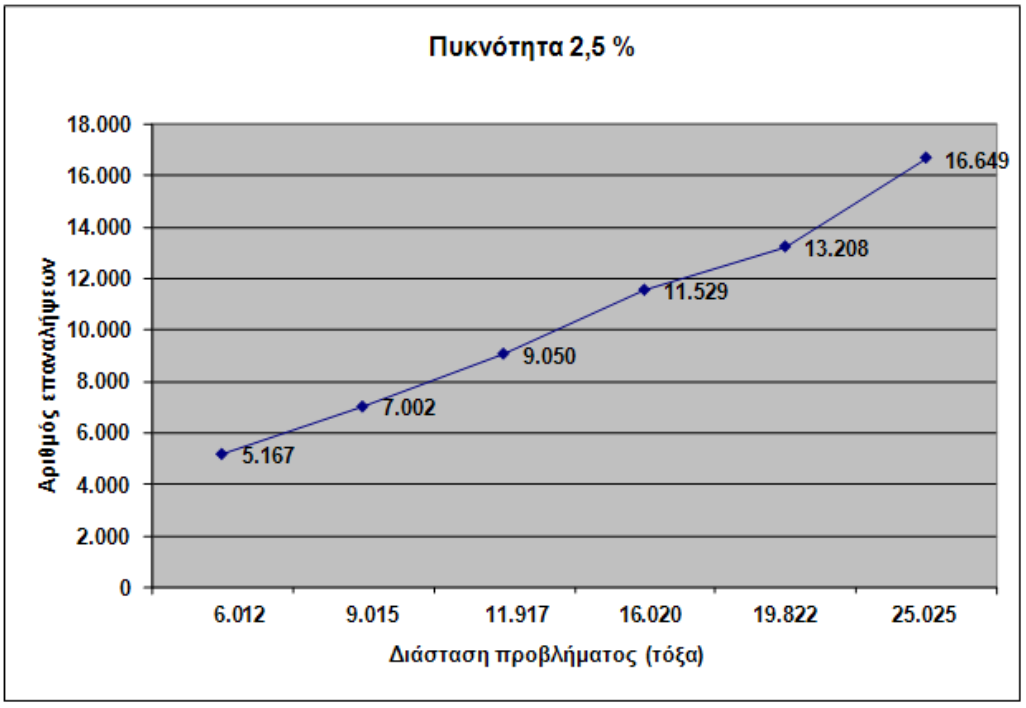
Στον Πίνακα **4.2**, μετά την επιτυχή λύση των προαναφερθέντων στιγμιοτύπων, αναφέρονται τα υπολογιστικά αποτελέσματα τα οποία μας παρουσιάζουν μια ενδεικτική υπολογιστική συμπεριφορά του αλγορίθμου NEPSA, όσον αφορά τον αριθμό των επαναλήψεων καθώς και το χρόνο που δαπανήθηκε από τον επεξεργαστή (CPU time) για τη συγκεκριμένη κλάση προβλημάτων.

Στο παρακάτω Σχήμα 4.5, φαίνεται πως μεταβάλλεται ο μέσος αριθμός των επαναλήψεων, ο οποίος προκύπτει από τα διάφορα στιγμιότυπα ΠΡΕΚ που λύθηκαν στη συγκεκριμένη κατηγορία.

Στο παρακάτω Σχήμα 4.6, φαίνεται πως μεταβάλλεται ο μέσος χρόνος που δαπανήθηκε από τον

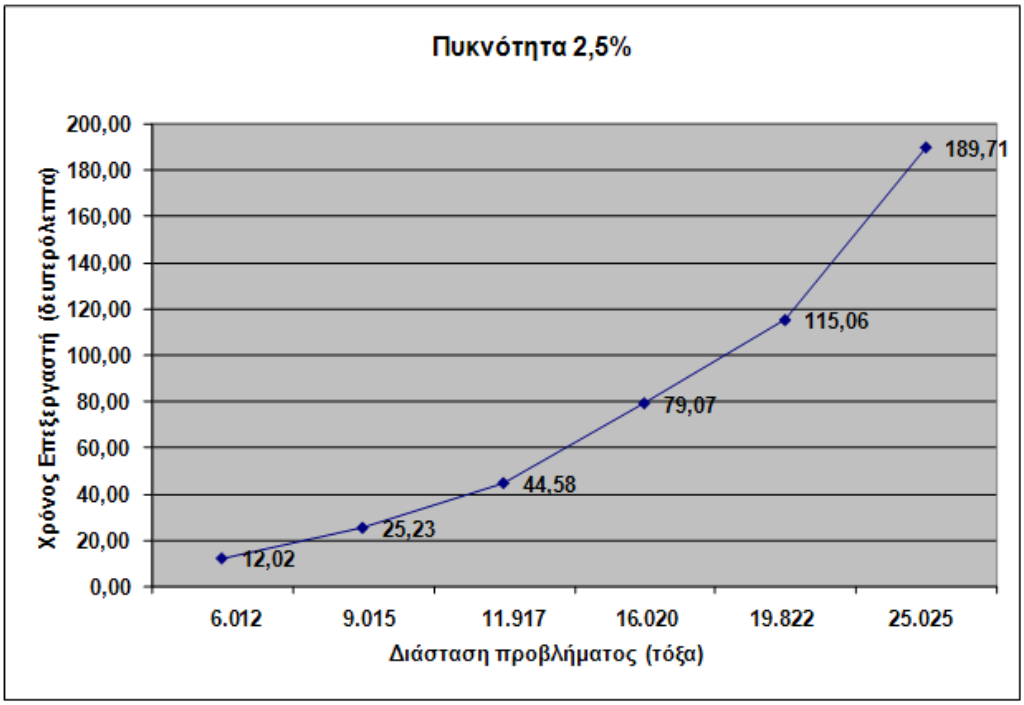
Πυκνότητα	Αριθμός κόμβων	Αριθμός τόξων	Χρόνος επεξεργαστή (sec.)	Αριθμός επαναλήψεων
	501	6.012	12,02	5.167
	601	9.015	25,23	7.002
2,5%	701	11.917	44,58	9.050
	801	16.020	79,07	11.529
	901	19.822	115,06	13.208
	1001	25.025	189,71	16.649

Πίνακας **4.2** Υπολογιστικά αποτελέσματα για προβλήματα πυκνότητας 2,5%



Σχήμα 4.5 Μεταβολή του μέσου αριθμού των επαναλήψεων για προβλήματα πυκνότητας 2,5%

επεξεργαστή, ο οποίος προκύπτει από τα διάφορα στιγμιότυπα ΠΡΕΚ που λύθηκαν σε αυτή την κλάση των προβλημάτων.



Σχήμα 4.6 Μεταβολή του μέσου χρόνου του επεξεργαστή για προβλήματα πυκνότητας 2,5%

4.5.2 Προβλήματα πυκνότητας 5 %

Στον Πίνακα **4.3**, φαίνονται όλοι οι παράμετροι που χρησιμοποιήθηκαν για την κατασκευή των στιγμιοτύπων ΠΡΕΚ που πήραν μέρος στην συγκεκριμένη κλάση προβλημάτων.

Πυκνό-τητα	ακμές (two-way arcs)	κόμβοι	πλάτος πλέγματος	κόμβοι προσφοράς	κόμβοι ζήτησης	βαθμός κόμβου	συνολική ποσότητα	κατανομή κόστους, 1 για ομοιόμορφη	κάτω όριο κόστους	άνω όριο κόστους
	1	500	100	250	150	25	100.000	1	1	100
	1	600	100	300	200	30	200.000	1	1	100
5%	1	700	100	350	250	35	300.000	1	1	100
	1	800	100	400	300	40	400.000	1	1	100
	1	900	100	400	300	45	400.000	1	1	100
	1	1.000	100	500	400	50	400.000	1	1	100

Πίνακας **4.3** Παράμετροι της γεννήτριας GRIDGEN για προβλήματα πυκνότητας 0,5 %

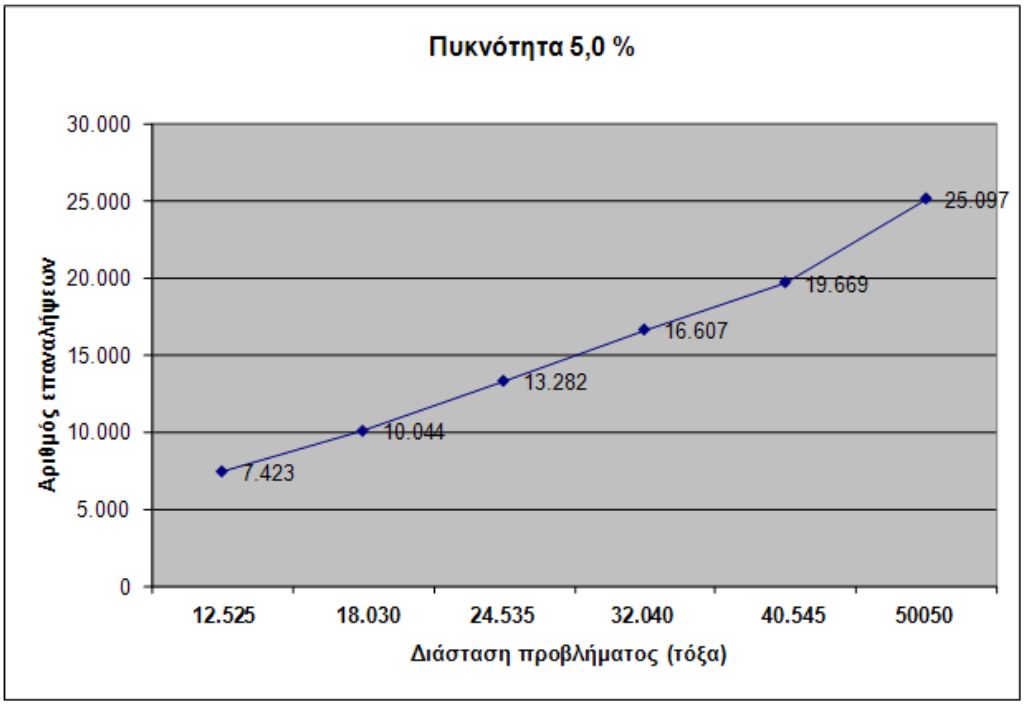
Στον Πίνακα **4.4**, μετά την επιτυχή λύση των προαναφερθέντων στιγμιοτύπων, αναφέρονται τα υπολογιστικά αποτελέσματα τα οποία μας παρουσιάζουν μια ενδεικτική υπολογιστική συμπεριφορά του αλγορίθμου NEPSA, όσον αφορά τον αριθμό των επαναλήψεων καθώς και το χρόνο που δαπανήθηκε από τον επεξεργαστή (CPU time) για τη συγκεκριμένη κλάση προβλημάτων.

Πυκνότητα	Αριθμός κόμβων	Αριθμός τόξων	Χρόνος επεξεργαστή (sec.)	Αριθμός επαναλήψεων
	501	12.525	36,61	7.423
	601	18.030	72,48	10.044
5%	701	24.535	134,04	13.282
	801	32.040	229,11	16.607
	901	40.545	352,70	19.669
	1001	50.050	582,80	25.097

Πίνακας **4.4** Υπολογιστικά αποτελέσματα για προβλήματα πυκνότητας 5%

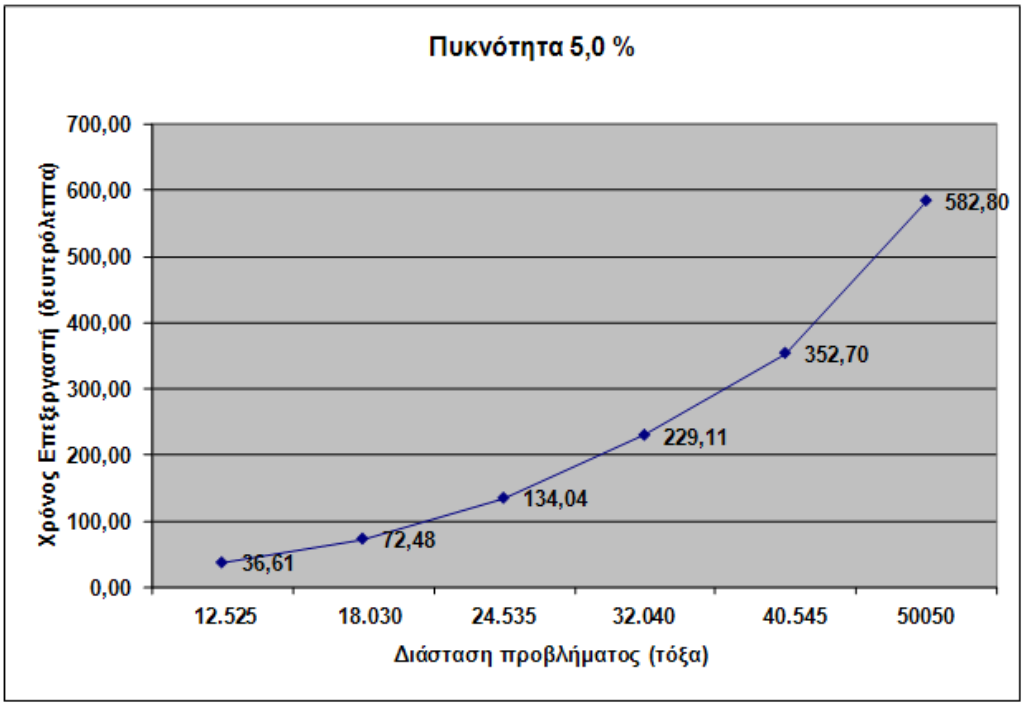
Στο παρακάτω Σχήμα 4.7, φαίνεται πως μεταβάλλεται ο μέσος αριθμός των επαναλήψεων, ο

οποίος προκύπτει από τα διάφορα στιγμιότυπα ΠΡΕΚ που λύθηκαν στη συγκεκριμένη κατηγορία.



Σχήμα 4.7 Μεταβολή του μέσου αριθμού των επαναλήψεων για προβλήματα πυκνότητας 5,0%

Στο παρακάτω Σχήμα 4.8, φαίνεται πως μεταβάλλεται ο μέσος χρόνος που δαπανήθηκε από τον επεξεργαστή, ο οποίος προκύπτει από τα διάφορα στιγμιότυπα ΠΡΕΚ που λύθηκαν σε αυτή την κλάση των προβλημάτων.



Σχήμα 4.8 Μεταβολή του μέσου χρόνου του επεξεργαστή για προβλήματα πυκνότητας 5,0%

4.5.3 Προβλήματα πυκνότητας 10 %

Στον Πίνακα **4.5**, φαίνονται όλοι οι παράμετροι που χρησιμοποιήθηκαν για την κατασκευή των στιγμιοτύπων ΠΡΕΚ που πήραν μέρος στην συγκεκριμένη κλάση προβλημάτων.

Πυκνό--τητα	ακμές (two-way arcs)	κόμβοι	πλάτος πλέγματος	κόμβοι προσφοράς	κόμβοι ζήτησης	βαθμός κόμβου	συνολική ποσότητα	κατανομή κόστους, 1 για ομοιόμορφη	κάτω όριο κόστους	άνω όριο κόστους
	1	500	100	250	150	50	100.000	1	1	100
	1	600	100	300	200	60	200.000	1	1	100
10%	1	700	100	350	250	70	300.000	1	1	100
	1	800	100	400	300	80	400.000	1	1	100
	1	900	100	400	300	90	400.000	1	1	100
	1	1.000	100	500	400	100	400.000	1	1	100

Πίνακας **4.5** Παράμετροι της γεννήτριας GRIDGEN για προβλήματα πυκνότητας 10 %

Στον Πίνακα **4.6**, μετά την επιτυχή λύση των προαναφερθέντων στιγμιοτύπων, αναφέρονται τα υπολογιστικά αποτελέσματα τα οποία μας παρουσιάζουν μια ενδεικτική υπολογιστική συμπεριφο-

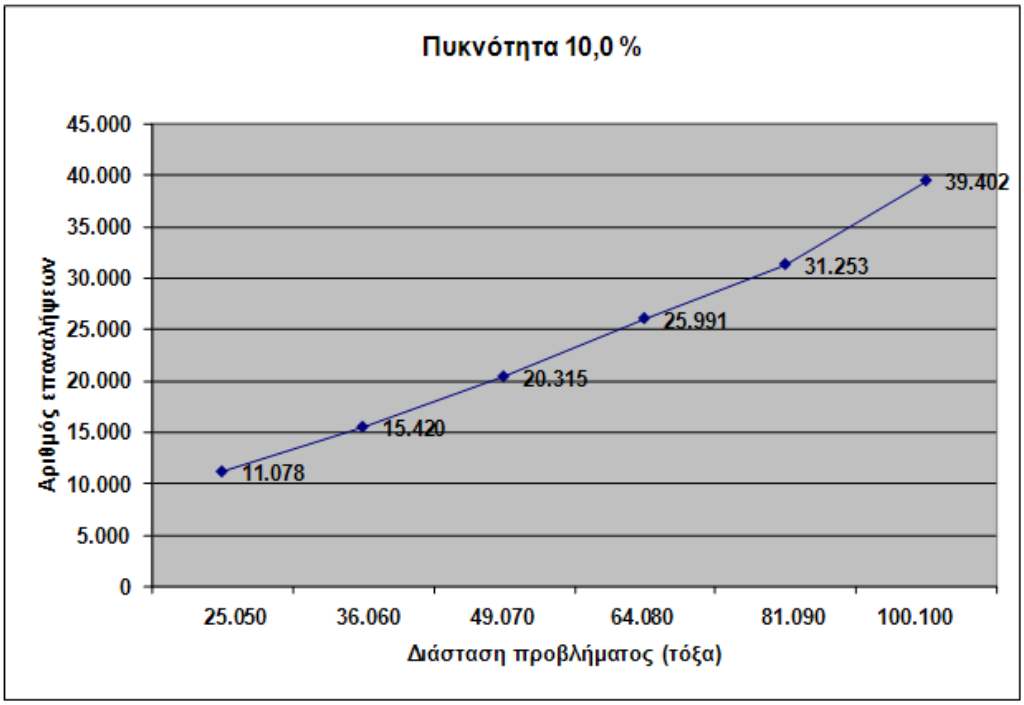
ρά του αλγορίθμου NEPSA, όσον αφορά τον αριθμό των επαναλήψεων καθώς και το χρόνο που δαπανήθηκε από τον επεξεργαστή (CPU time) για τη συγκεκριμένη κλάση προβλημάτων.

Πυκνότητα	Αριθμός κόμβων	Αριθμός τόξων	Χρόνος επεξεργαστή (sec.)	Αριθμός επαναλήψεων
10%	501	25.050	106,35	11.078
	601	36.060	226,04	15.420
	701	49.070	428,04	20.315
	801	64.080	750,30	25.991
	901	81.090	1.155,10	31.253
	1001	100.100	1.896,36	39.402

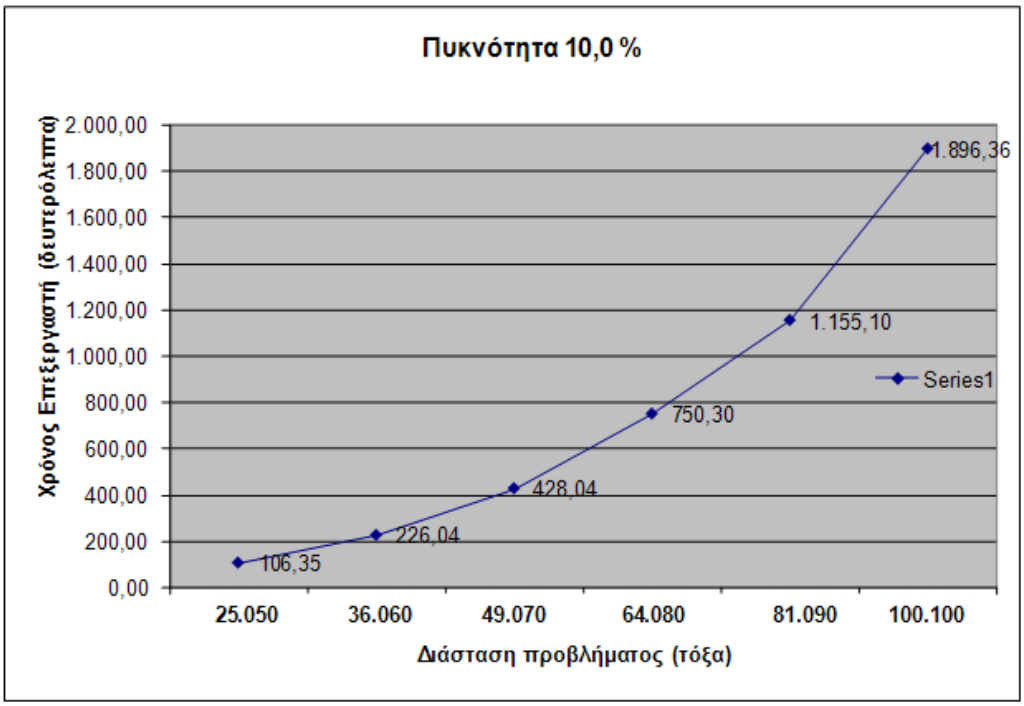
Πίνακας **4.6** Υπολογιστικά αποτελέσματα για προβλήματα πυκνότητας 10%

Στο παρακάτω Σχήμα 4.9, φαίνεται πως μεταβάλλεται ο μέσος αριθμός των επαναλήψεων, ο οποίος προκύπτει από τα διάφορα στιγμιότυπα ΠΡΕΚ που λύθηκαν στη συγκεκριμένη κατηγορία.

Στο παρακάτω Σχήμα 4.10, φαίνεται πως μεταβάλλεται ο μέσος χρόνος που δαπανήθηκε από τον επεξεργαστή, ο οποίος προκύπτει από τα διάφορα στιγμιότυπα ΠΡΕΚ που λύθηκαν σε αυτή την κλάση των προβλημάτων.



Σχήμα 4.9 Μεταβολή του μέσου αριθμού των επαναλήψεων για προβλήματα πυκνότητας 10,0%



Σχήμα 4.10 Μεταβολή του μέσου χρόνου του επεξεργαστή για προβλήματα πυκνότητας 10,0%

4.5.4 Προβλήματα πυκνότητας 0,1 %

Στον Πίνακα **4.7**, φαίνονται όλοι οι παράμετροι που χρησιμοποιήθηκαν για την κατασκευή των στιγμιοτύπων ΠΡΕΚ που πήραν μέρος στην συγκεκριμένη κλάση προβλημάτων.

Πυκνό--τητα	ακμές (two-way arcs)	κόμβοι	πλάτος πλέγματος	κόμβοι προσφοράς	κόμβοι ζήτησης	βαθμός κόμβου	συνολική ποσότητα	κατανομή κόστους, 1 για ομοιόμορφη	κάτω όριο κόστους	άνω όριο κόστους
	1	5.000	70	2.500	1.500	5	100.000	1	1	100
0,1%	1	7.500	90	3.000	3.000	7	100.000	1	1	100
	1	10.000	100	5.000	4.000	10	100.000	1	1	100

Πίνακας **4.7** Παράμετροι της γεννήτριας GRIDGEN για προβλήματα πυκνότητας 0,1 %

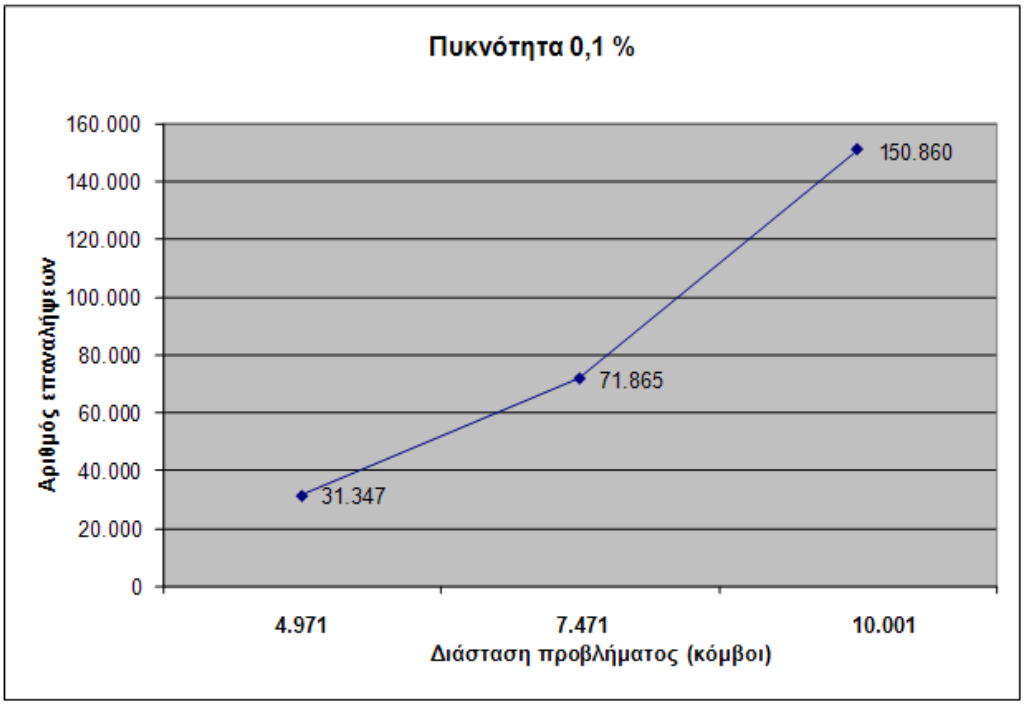
Στον Πίνακα **4.8**, μετά την επιτυχή λύση των προαναφερθέντων στιγμιοτύπων, αναφέρονται τα υπολογιστικά αποτελέσματα τα οποία μας παρουσιάζουν μια ενδεικτική υπολογιστική συμπεριφορά του αλγορίθμου NEPSA, όσον αφορά τον αριθμό των επαναλήψεων καθώς και το χρόνο που δαπανήθηκε από τον επεξεργαστή (CPU time) για τη συγκεκριμένη κλάση προβλημάτων.

Πυκνότητα	Αριθμός κόμβων	Αριθμός τόξων	Χρόνος επεξεργαστή (sec.)	Αριθμός επαναλήψεων
	4.971	24.855	371,69	31.347
0,1%	7.471	52.297	3.200,25	71.865
	10.001	100.010	16.440,15	150.860

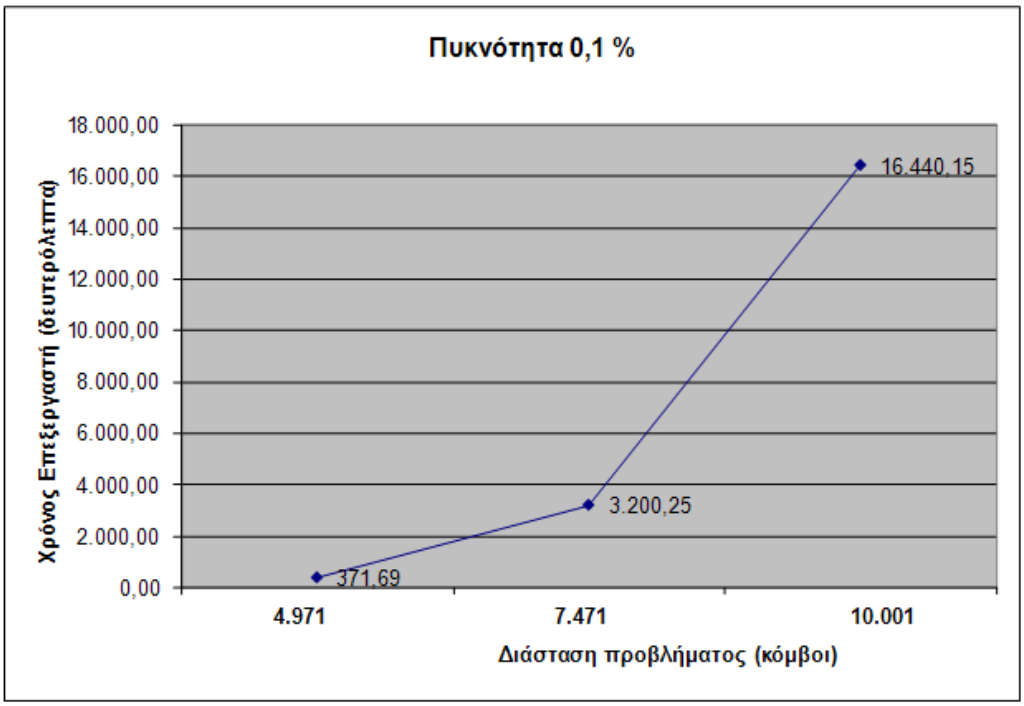
Πίνακας **4.8** Υπολογιστικά αποτελέσματα για προβλήματα πυκνότητας 0,1 %

Στο παρακάτω Σχήμα 4.11, φαίνεται πως μεταβάλλεται ο μέσος αριθμός των επαναλήψεων, ο οποίος προκύπτει από τα διάφορα στιγμιότυπα ΠΡΕΚ που λύθηκαν σε κάθε κατηγορία.

Στο παρακάτω Σχήμα 4.12, φαίνεται πως μεταβάλλεται ο μέσος χρόνος που δαπανήθηκε από τον επεξεργαστή, ο οποίος προκύπτει από τα διάφορα στιγμιότυπα ΠΡΕΚ που λύθηκαν σε αυτή την κλάση των προβλημάτων.



Σχήμα 4.11 Μεταβολή του μέσου αριθμού των επαναλήψεων για προβλήματα πυκνότητας 0,1%



Σχήμα 4.12 Μεταβολή του μέσου χρόνου του επεξεργαστή για προβλήματα πυκνότητας 0,1%

Τα αποτελέσματα, που προέκυψαν από αυτή την υπολογιστική μελέτη, παρουσιάστηκαν για πρώτη φορά από τον Σιφαλέρα Α. στο *Διεθνές Συνέδριο Βελτιστοποίησης: Προχωρημένοι Αλγόριθμοι*, στο Veszprém της Ουγγαρίας το 2006, με την εργασία (Paparrizos, Samaras, and Sifaleras 2006b).

ΚΕΦΑΛΑΙΟ 5

Το Εκπαιδευτικό λογισμικό **WebNetPro**, για την επίλυση προβλημάτων δικτυακής βελτιστοποίησης

5.1 Εκπαιδευτικό λογισμικό δικτυακής βελτιστοποίησης

Είναι κοινά αποδεκτό ότι η κατανόηση των θεμάτων αλγορίθμων δικτυακής βελτιστοποίησης απαιτεί αρκετή προσπάθεια από τους φοιτητές. Για αυτό το λόγο έχουν δημιουργηθεί αρκετά λογισμικά οπτικοποίησης αλγορίθμων και επίλυσης αντίστοιχων προβλημάτων. Έχουν διεξαχθεί πολλές μελέτες στη διεθνή βιβλιογραφία, οι οποίες καταδεικνύουν τα πλεονεκτήματα της χρήσης εκπαιδευτικού λογισμικού, όπως για παράδειγμα στις εργασίες (Hundhausen, Douglas, and Stasko 2002), (Lawrence, Badre, and Stasko 1994). Χαρακτηριστικά αναφέρονται οι εξής διδακτικές εφαρμογές δικτυακής βελτιστοποίησης, (Andreou, Paparrizos, Samaras, and Sifaleras 2005), (Andreou, Paparrizos, Samaras, and Sifaleras 2006), (Baloukas, Paparrizos, and Sifaleras 2005) και (Khuri and Holzapfel 2001).

Μία από τις καλύτερες δωρεάν διαθέσιμες εφαρμογές είναι το Netflo (Kennington and Helgason 1980). Η εφαρμογή Netflo μπορεί να επιλύσει προβλήματα ροής ελαχίστου κόστους. ο χρήστης υποβάλλει το πρόβλημα σε μορφή αρχείου του τύπου DIMACS ή τύπου Netflo. Τα προβλήματα μπορούν να υποβληθούν είτε μέσω του διαδικτύου (http ή ftp URL's), είτε μέσω ηλεκτρονικής

αλληλογραφίας. Παρ΄ όλα αυτά, η εφαρμογή Netflo ασχολείται κυρίως με βελτιστοποίηση μεγάλης κλίμακας και δεν έχει τόσο πολύ εκπαιδευτικό χαρακτήρα. Επιπρόσθετα, δεν επιλύει άλλες κατηγορίες προβλημάτων πέρα από το πρόβλημα ροής ελαχίστου κόστους.

Μεταξύ διαφόρων ιστοσελίδων, που περιέχουν εκπαιδευτικά εργαλεία, είναι και οι (Vanderbei 1997b) και (Vanderbei 1997a). Οι δυο προαναφερθέντες ιστοσελίδες αποτελούν αποτελούν δικτυακές εφαρμογές (Java applets). Η πρώτη παρουσιάζει ένα διδακτικό εργαλείο οπτικοποίησης, με το οποίο μπορεί ένας φοιτητής να λύσει ένα πρόβλημα ροής ελαχίστου κόστους, χρησιμοποιώντας τον κλασικό πρωτεύοντα αλγόριθμο Simplex. Η δεύτερη ιστοσελίδα περιέχει πάλι ένα αντίστοιχο (Java applet) για εύρεση ελαχίστων δρόμων, χρησιμοποιώντας τον αλγόριθμο του Dijkstra. Στις δύο προηγούμενες εφαρμογές, ο χρήστης με εύκολο τρόπο καθορίζει τον αριθμό των κόμβων και τόξων κάνοντας άμεση διαχείριση αυτών των δεδομένων. Παρ΄ όλα αυτά, ο χρήστης δεν μπορεί να εξετάσει, ή πόσο μάλλον να επιλέξει, κάποιον τρόπο αποθήκευσης γραφημάτων, ούτε έχει την επιλογή να επιλέξει μεταξύ διαφορετικών αλγορίθμων για κάποια άλλη κατηγορία προβλημάτων. Μία άλλη ιστοσελίδα με χρήσιμο περιεχόμενο είναι η (Greenberg 1999). Παρ΄ όλα αυτά, αυτή η ιστοσελίδα παρέχει την οπτικοποιημένη λύση τους προβλήματος εύερεσης ελαχίστων δρόμων βάσει του αλγορίθμου Dijkstra. Η συγκεκριμένη εφαρμογή είναι υλοποιημένη σε Java, όπου ο χρήστης καθορίζει τους κόμβους και τα τόξα του δικτύου κάνοντας μερικά κλικ εντός του παραθύρου της εφαρμογής.

Ένα πάρα πολύ χρήσιμο εκπαιδευτικό λογισμικό είναι το DIDAGRAPH, (Dagdilelis and Satratzemi 1998). Το DIDAGRAPH υλοποιεί ένα ευρύ σύνολο αλγορίθμων και είναι ικανό να επιλύσει παραμετροποιημένα προβλήματα. Παρ΄ όλα αυτά, δεν είναι προσβάσιμο μέσω διαδικτύου και δεν δίδεται η δυνατότητα στο χρήστη να δοκιμάσει πληθώρα διαφορετικών δομών δεδομένων. Ολοκληρώνοντας, στην εργασία (Lazaridis, Paparrizos, Samaras, and Sifaleras 2007) περιγράφεται μια διαδικτυακή εφαρμογή, η οποία κάνει χρήση τεχνικών οπτικοποίησης για επίλυση πλήρως παραμε-

τροποιημένων προβλημάτων. Όμως, ο κύριος σκοπός του είναι η εκμάθηση του αναθεωρημένου αλγορίθμου Simplex για το γραμμικό πρόβλημα του Μαθηματικού Προγραμματισμού και όχι οι αλγόριθμοι για προβλήματα βελτιστοποίησης ροών σε δίκτυα. Επίσης, δεν υπάρχει επιλογή μεταξύ διαφορετικών δομών για το πρόβλημα προς επίλυση.

5.2 Εισαγωγή στο WebNetPro

Η ανάπτυξη μιας εφαρμογής, προσβάσιμης μέσω διαδικτύου, ικανή να παρουσιάσει μια οπτική υλοποίηση διαφόρων αλγορίθμων γραφημάτων, αποτελεί ένα χρήσιμο εκπαιδευτικό εργαλείο. Οπότε, η υλοποίηση ενός τέτοιου διδακτικού εργαλείου υπήρξε ένα σοβαρό κίνητρο. Μέχρις στιγμής και μετά από προσεκτική έρευνα, είναι η πρώτη φορά όπου ένα τέτοιο μεγάλο σύνολο υλοποιημένων αλγορίθμων, για τελείως παραμετροποιήσιμα προβλήματα, ενσωματώνονται σε ένα ολοκληρωμένο γραφικό περιβάλλον μέσω του διαδικτύου.

Υπάρχει μια περιορισμένη διαθεσιμότητα σε εκπαιδευτικούς λύτες προβλημάτων, οι οποίοι να είναι ελεύθερα διαθέσιμοι, προσβάσιμοι μέσω διαδικτύου, και να επιλύουν πλήρως παραμετροποιήσιμα προβλήματα. Ακόμη δυσκολότερο είναι να βρεθούν λύτες, που να είναι σε θέση να αξιοποιούν μια πληθώρα διαφορετικών δομών δεδομένων όπως η μήτρα κόμβων - κόμβων, η μήτρα κόμβων - τόξων, λίστα τόξων και αστεροειδής αποθήκευση (εξερχομένων, εισερχομένων ή ταυτόχρονη). Ο αριθμός των αλγορίθμων γραφημάτων που συμπεριλαμβάνονται στο WebNetPro (Karagiannis, Markelis, Paparrizos, Samaras, and Sifaleras 2006), ξεπερνάει τον αριθμό των αλγορίθμων που συμπεριλαμβάνονται σε τέτοιους λύτες. Επιπλέον δε, οι περισσότερες εκπαιδευτικές εφαρμογές ασχολούνται με έτοιμα λυμένα παραδείγματα από κάποιου είδους προβλήματα.

Αυτά είναι τα σημεία, που η εφαρμογή WebNetPro διαφέρει πραγματικά. Με τη χρήση του προτεινόμενου λογισμικού ο εισηγητής μπορεί να συνεπικουρηθεί ούτως ώστε οι φοιτητές να κατανοήσουν καλύτερα τις διάφορες δομές δεδομένων των γραφημάτων και των αλγορίθμων τους. Η

εφαρμογή WebNetPro αποτελεί εξέλιξη του πακέτου NetPro, το οποίο παρουσιάστηκε για πρώτη φορά από τον Σιφαλέρα Α. στο 16ο Πανελλήνιο Συνέδριο της Ελληνικής Εταιρίας Επιχειρησιακών Ερευνών (HELORS) το 2003, με την εργασία (Dosios, Paparrizos, Samaras, and Sifaleras 2003b). Το πακέτο λογισμικού NetPro ήταν μια εφαρμογή βασισμένη αποκλειστικά στο Matlab. Το κύριο μειονέκτημα του NetPro ήταν ότι, για να μπορέσει να χρησιμοποιηθεί από κάποιον, θα έπρεπε προηγουμένως να είναι εγκατεστημένο τόσο το ίδιο, όσο φυσικά και το ίδιο το Matlab στον τοπικό υπολογιστή του χρήστη. Ακόμη, το σύνολο των υλοποιημένων αλγορίθμων, όσο και των δομών δεδομένων έχει επεκταθεί σημαντικά.

Μια άλλη πολύ σημαντική αναβάθμιση στο WebNetPro είναι ότι ο χρήστης πλέον έχει τη δυνατότητα να αποθηκεύει τις παραμέτρους των προβλημάτων που εισάγει, σε αρχείο κατάλληλης μορφής. Αυτά τα αρχεία μπορούν αργότερα να μεταφορτωθούν από τους χρήστες / φοιτητές στο WebNetPro. Αυτή η λειτουργία περιγράφεται αναλυτικά στην Ενότητα 5.6. Είναι κοινά αποδεκτό, ότι τέτοιου είδους εκπαιδευτικά εργαλεία και μεθόδοι μπορούν να βοηθήσουν τους φοιτητές, οι οποίοι για διάφορους λόγους δεν μπορούν να παρακολουθήσουν με φυσική παρουσία κάποιο μάθημα.

Το λογισμικό WebNetPro το οποίο στην τελευταία του έκδοση είναι διαθέσιμο μέσω Διαδικτύου. Το *WebNetPro* είναι ουσιαστικά μια πλατφόρμα επίλυσης προβλημάτων δικτυακής βελτιστοποίησης. Αρχικά είχε κατασκευαστεί ως μια εργαλειοθήκη (toolbox) η οποία ενσωματώνονταν στο Matlab, αλλά βρίσκεται διαρκώς υπό ανάπτυξη. Όλοι οι αλγόριθμοι έχουν υλοποιηθεί στο προγραμματιστικό περιβάλλον του Matlab εκδόσεως R14, με εγκατεστημένο το Service Pack 3.

Το διαδραστικό λογισμικό που περιγράφεται σε αυτό το κεφάλαιο περιέχει πολλούς υλοποιημένους αλγορίθμους σε Matlab, για την επίλυση προβλημάτων Δικτυακής Βελτιστοποίησης. Συγκεκριμένα ο χρήστης μπορεί να επιλύσει προβλήματα εύρεσης ελαχίστων δρόμων σε άκυκλα δίκτυα, σε δίκτυα με μη αρνητικού κόστους προσανατολισμένους κύκλους καθώς και σε δίκτυα με μη αρνητικά τόξα (Dijkstra). Επιπλέον, ο χρήστης μπορεί να επιλύσει προβλήματα εύρεσης ελαχίστων δέντρων

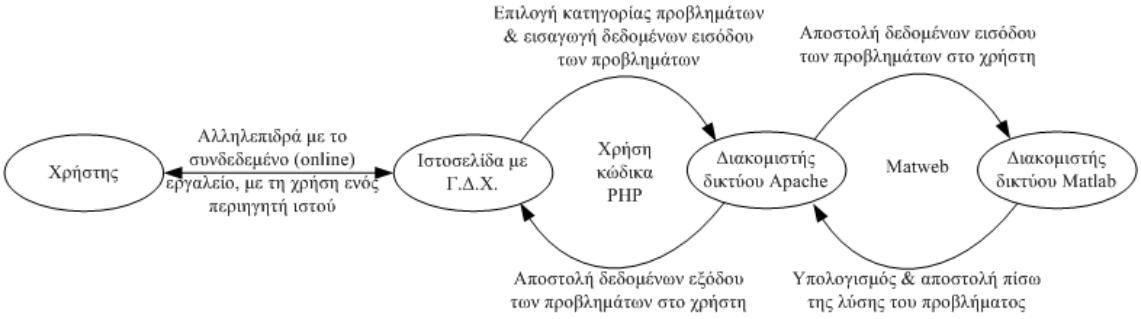
καλυμμάτων είτε με τον αλγόριθμο του Prim, είτε με τον αλγόριθμο του Kruscal. Υπάρχουν ακόμη αλγόριθμοι για την εύρεση έσω ή έξω δέντρου διάσχισης σε προσανατολισμένους γράφου, αλγόριθμοι οι οποίοι ανιχνεύουν απλή αλλά και ισχυρή συνεκτικότητα σε γράφους, αλλά και αλγόριθμοι υπολογισμού τοπολογικής διάταξης. Ακόμη υπάρχει δυνατότητα επίλυσης προβλημάτων μεγίστης ροής, σύμφωνα με τον αλγόριθμο υπολογισμού αυξανόντων δρόμων (augmenting path algorithm). Ο χρήστης μπορεί να λύσει κάποιο από τα προαναφερθέντα προβλήματα επιλέγοντας μεταξύ διαφορετικών δομών δεδομένων, όπως η μήτρα γειτονιά ή κόμβων - κόμβων, μήτρα κόμβων - τόξων, η λίστα τόξων, ή αστεροειδή αποθήκευση εισερχομένων ή εξερχομένων (ή και των δύο) τόξων. Τέλος, παρέχεται στον χρήστη η δυνατότητα να σχεδιάσει ριζωμένα δέντρα, είτε κατευθυνόμενα (με τόξα) είτε όχι (με ακμές) με εντελώς παραμετροποιήσιμο τρόπο.

5.3 Υλοποίηση της εφαρμογής WebNetPro

Τα τελευταία χρόνια, το Matlab χρησιμοποιείται εκτενώς από πολλούς επιστήμονες σε όλο τον κόσμο και διδάσκεται σε όλο και περισσότερα ακαδημαϊκά ιδρύματα, βλέπε για παράδειγμα στις εργασίες (Colgan 2000) και (Dunn and Harman 2002). Παρ'ολα αυτά οι χρήστες του λογισμικού WebNetPro δεν απαιτείται να είναι εξοικειωμένοι με τη γλώσσα προγραμματισμού Matlab, ούτε καν χρειάζεται το Matlab να είναι εγκατεστημένο στους δικούς τους προσωπικούς υπολογιστές (ή από όπου έχουν πρόσβαση στο διαδίκτυο). Με αυτό τον τρόπο το WebNetPro αξιοποιεί όλα τα προγραμματιστικά πλεονεκτήματα του Matlab, όπως ένα τεράστιο σύνολο από μαθηματικές εγγενείς ρουτίνες κ.α, βλέπε για παράδειγμα (Paparrizos 2004). Το Matlab συνδυάζει γραφικά και οπτικοποίηση (Marchand and Holland 2002) και παρέχει στο χρήστη τη δυνατότητα να κατασκευάζει Γραφική Διεπιφάνεια Χρήστη (Γ.Δ.Χ. ή G.U.I.) με έναν απλό τρόπο, ενώ δεν αφήνει καμμία δυσκολία στον τελικό χρήστη του λογισμικού.

Όλη η διαδικασία της επίλυσης ενος παραμετροποιημένου, από το φοιτητή, προβλήματος, περι-

γράφεται στο Σχήμα 5.1. Ο φοιτητής - χρήστης πρώτα επισκέπτεται μέσω ενός περιηγητή ιστού την on-line εφαρμογή. Έπειτα, δια μέσου ενός πτυσσόμενου μενού, επιλέγει κάποιον αλγόριθμο. Η Γραφική Διεπιφάνεια Χρήστη (Γ.Δ.Χ.) ή (G.U.I.) θα περιγραφθεί αναλυτικά στην επόμενη ενότητα. Για την ακρίβεια, η εφαρμογή έχει διαδραστικό χαρακτήρα αφού, ο χρήστης είναι σε θέση να εισάγει τα απαιτούμενα δεδομένα εισόδου του προβλήματος, ή να τα τροποποιήσει αργότερα και να παρατηρήσει με ποιό τρόπο μεταβάλλεται η λύση. Κατόπιν, ο κώδικας PHP μεταβιβάζει τα δεδομένα εισόδου από την html φόρμα στον διακομιστή δικτύου Apache Web Server. Ο εξυπηρετητής δικτύου Apache χρησιμοποιεί ένα ειδικό αρχείο το οποίο ονομάζεται matweb, ούτως ώστε να επικοινωνήσει με τον διακομιστή δικτύου Matlab Web server. Σε αυτό το σημείο, η λύση υπολογίζεται εξ΄ ολοκλήρου από τον Matlab Web Server και όχι από τον υπολογιστή *πελάτη* client computer. Στη συνέχεια, τα δεδομένα εξόδου επιστρέφονται στον Apache Web server και μετά πίσω στην ιστοσελίδα της εκπαιδευτικής εφαρμογής. Με αυτόν τον τρόπο, ο χρήστης παραλαμβάνει τα δεδομένα εξόδου. Επιπρόσθετα, ο χρήστης μπορεί να τροποποιήσει τα δεδομένα εισόδου και να δεί πως επηρρεάζεται η υπολογισμένη λύση (δίχως να εισάγει από την αρχή όλα τα δεδομένα).



Σχήμα 5.1 Διαδικασία επίλυσης προβλημάτων

Αυτή τη στιγμή έχει χρησιμοποιηθεί ο ανοιχτού κώδικα εξυπηρετητής δικτύου Apache, έκδοσης 2.0.55. Υπήρχαν επίσης ανάγκες για διαδραστικότητα μεταξύ εφαρμογής και χρήστη. Συγκεκριμένα, ο χρήστης πρέπει να δίνει κάποια δεδομένα εισόδου, τα οποία θα προωθούνται στον Matlab Web

server και αντίστροφα. Αυτή η ανάγκη καλύφθηκε, καθώς επίσης και κάποια θέματα ασφαλείας, με τη χρήση της ανοιχτού κώδικα γλώσσας προγραμματισμού PHP, η έκδοση 5.1.1 χρησιμοποιήθηκε στην προτεινόμενη εφαρμογή. Τέλος, όλα τα πτυσσόμενα μενού στη γραφική διεπιφάνεια υλοποιήθηκαν με τη βοήθεια της γλώσσας προγραμματισμού JavaScript.

Επιλέον, το WebNetPro είναι ανεξάρτητο πλατφόρμας και μπορεί να χρησιμοποιείται επιτυχώς για διδασκαλία - εκμάθηση αλγορίθμων συνδυαστικής βελτιστοποίησης εξ΄ αποστάσεως. Το WebNetPro σε αυτήν την έκδοση στηρίζεται στον Matlab Web server, έκδοση 1.2.4 και εκτελείται εξ΄ ολοκλήρου στον *Matlab Web Server* και όχι στους υπολογιστές που συνδέονται σε αυτόν (client). Μπορεί οποιοσδήποτε να το χρησιμοποιήσει, οποιαδήποτε χρονική στιγμή, από οποιοδήποτε προσωπικό υπολογιστή με πρόσβαση στο Διαδίκτυο. Η πρόσβαση γίνεται με τη βοήθεια ενός περιηγητή ιστού όπως Mozilla Firefox, Opera ή τέλος τον Microsoft Internet Explorer. Η εφαρμογή WebNetPro μπορεί να ξεκινήσει μέσω του ηλεκτρονικού ιστοχώρου:

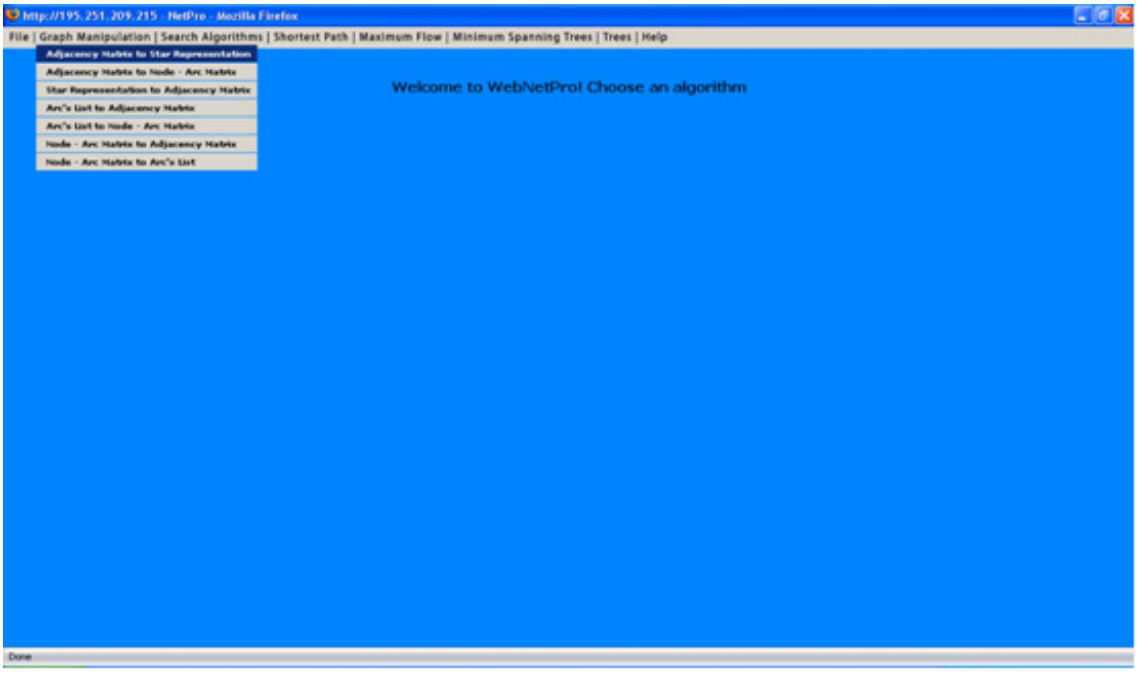
http://eos.uom.gr/∼sifalera/WebNetPro

5.4 Η γραφική διεπιφάνεια χρήστη (Γ.Δ.Χ.) του WebNetPro

Το λογισμικό WebNetPro αποτελεί μια φιλική εφαρμογή τόσο για τον Μαθηματικό - Επιχειρησιακό ερευνητή όσο και για κάποιον άπειρο χρήστη - φοιτητή. Εκτελεί γρήγορα και αποτελεσματικά τα προβλήματα που εισάγονται από τους χρήστες, ενώ από την άλλη μεριά το γραφικό του περιβάλλον χρήσης μαθαίνεται εύκολα και γρήγορα.

Το Matlab στις τελευταίες εκδόσεις του παρέχει πολλά χρήσιμα εργαλεία κατασκευής γραφικών διεπιφανειών στον χρήστη, ο αναγνώστης μπορεί να βρεί αναλυτικές πληροφορίες στο βιβλίο (Paparrizos 2004) και ειδικότερα στο βιβλίο (Marchand and Holland 2002). Η γραφική διεπιφάνεια

του WebNetPro αποτελείται από μια κεντρική μπάρα επιλογών η οποία με τη σειρά της αποτελείται από αρκετές πτυσσόμενες λίστες, όπως φαίνεται και στο ακόλουθο Σχήμα 5.2.

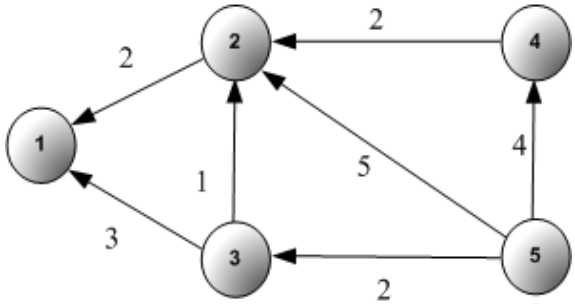


Σχήμα 5.2 Κεντρική μπάρα επιλογών

Το προτεινόμενο εργαλείο έχει αναπτυχθεί κυρίως για διδακτικούς σκοπούς, παρά για σκοπούς βελτιστοποίησης μεγάλης κλίμακας, βλέπε χαρακτηριστικά το λογισμικό (Kennington and Helgason 1980). Λόγω λοιπόν του εκπαιδευτικού χαρακτήρα της εφαρμογής, ο χρήστης είναι περιορισμένος να εισάγει πρόβλημα με μέγιστη διάσταση 10 κόμβους και 15 τόξα. Αυτό γίνεται έτσι ώστε τα αποτελέσματα να απεικονίζονται σωστά στην οθόνη του χρήστη. Τέλος, πρέπει να αναφερθεί ότι κάποια από τα παραδείγματα αυτού του κεφαλαίου, είναι διαθέσιμα και στον ηλεκτρονικό ιστοχώρο του WebNetPro με τη μορφή βοηθητικής ιστοσελίδας (help file in HTML format), μεταφρασμένα στην αγγλική γλώσσα.

5.5 Επίλυση προβλημάτων με χρήση της Γ.Δ.Χ. του Web-NetPro

Έστω ότι δίνεται στο φοιτητή - χρήστη το παρακάτω κατευθυνόμενο γράφημα του Σχήματος 5.3.



Σχήμα 5.3 Δίκτυο παραδείγματος

Το ζητούμενο σε αυτή την ενότητα είναι να γίνει μετατροπή του τρόπου αποθήκευσης μεταξύ δυο δομών δεδομένων. Συγκεκριμένα μια μήτρα γειτονιάς θα μετατραπεί σε αστεροειδή αποθήκευση εξερχομένων τόξων. Σε αυτό το παράδειγμα η μήτρα γειτονιάς A είναι η παρακάτω:

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \end{pmatrix}$$

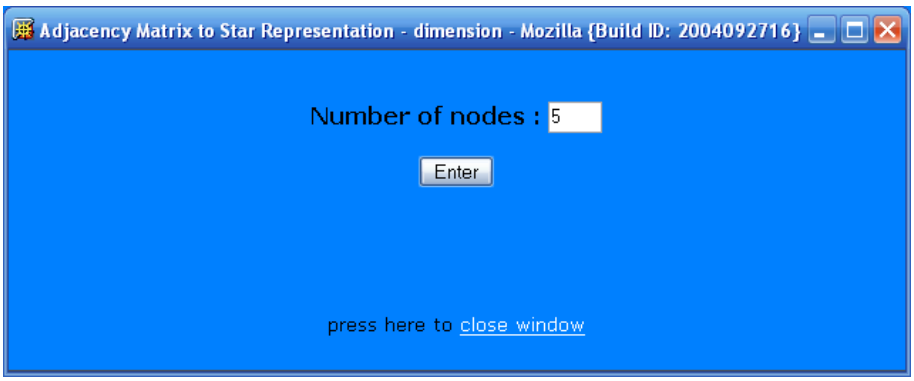
Ο παρακάτω πίνακας W είναι ο πίνακας με τα βάρη - κόστη των τόξων.

$$\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 \\ 3 & 1 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 5 & 2 & 4 & 0 \end{pmatrix}$$

Η αστεροειδής αποθήκευση εξερχομένων τόξων είναι παρόμοια με την αποθήκευση σε μορφή λίστας τόξων. Η ομοιότητα στηρίζεται στο γεγονός ότι η αστεροειδής αποθήκευση εξερχομένων τόξων αποθηκεύει τη λίστα γειτονικών κόμβων, κάθε κόμβου. Αυτού του είδους η αποθήκευση αποθηκεύει αυτές τις λίστες ως διανύσματα. Για να μπορέσει να αποθηκευθεί το δίκτυο, με τη μορφή αστεροειδής αποθήκευση εξερχομένων τόξων, κάθε τόξο αντιστοιχεί σε μια μοναδική ακολουθία αριθμών. Με αυτόν τον τρόπο, δημιουργείται μια διάταξη της λίστας τόξων. Έπειτα, όλα τα τόξα είναι αριθμημένα σε μια συγκεκριμένη σειρά, πρώτον όσα εξέρχονται από τον πρώτο κόμβο μετά αυτά που εξέρχονται από το δεύτερο κόμβο, κ.ο.κ. Τα τόξα που εξέρχονται από τον ίδιο κόμβο είναι αριθμημένα με αυθαίρετη σειρά και όλες οι πληροφορίες που αφορούν το κάθε τόξο αποθηκεύονται σειριακά στη λίστα τόξων. Για παράδειγμα, εάν το τόξο (g, h) είναι το ένατο τόξο, η ουρά, η κεφαλή, το κόστος κ.λ.π. αποθηκεύονται στις θέσεις των διανυσμάτων ουρά(9), κεφαλή(9) και κόστοσ(9). Επιπλέον δε, ένας δείκτης είναι συχετισμένος με κάθε κόμβο i και θα συμβολίζεται με $K(i)$. Ο $K(i)$ δείκτης υποδεικνύει το ελάχιστο αριθμημένο τόξο στη λίστα τόξων, το οποίο εξέρχεται από τον κόμβο i κόμβο. Οι δείκτες $K(i)$ θα αποθηκεύονται σε ένα διάνυσμα αποτελούμενο από $n + 1$ στοιχεία. Στην περίπτωση όπου ο i κόμβος δεν έχει κανένα εξερχόμενο τόξο, τότε θέτουμε $K(i) = K(i + 1)$. Επίσης, η αστεροειδής αποθήκευση εξερχομένων τόξων αποθηκεύει τα εξερχόμενα τόξα από τον i κόμβο στις θέσεις $K(i)$ έως $(K(i + 1) - 1)$ στη λίστα τόξων. Θέτουμε $K(1) = 1$, ενώ στην περίπτωση που ισχύει $K(i) > (K(i + 1) - 1)$, τότε κανένα τόξο δεν εξέρχεται

από τον κόμβο i. Για να μπορούμε να ελέγξουμε αν δεν ξεκινούν τόξα από τον κόμβο n θέτουμε $K(n+1) = m+1$. Περισσότερες αναλυτικές πληροφορίες για τα πλεονεκτήματα ή μειονεκτήματα σχετικά με τους διαφορετικούς τρόπους αποθήκευσης μπορούν να βρεθούν στο βίβλίο (Ahuja, Magnanti, and Orlin. 1993).

Ο χρήστης πρέπει δια μέσου της κεντρικής μπάρας επιλογών να επιλέξει: Graph Manipulation και έπειτα Adjacency Matrix to Star representation. Σε αυτό το σημείο το παράθυρο του Σχήματος 5.4, θα απαιτεί από το χρήστη να εισάγει τη διάσταση της μήτρας γειτονιάς.

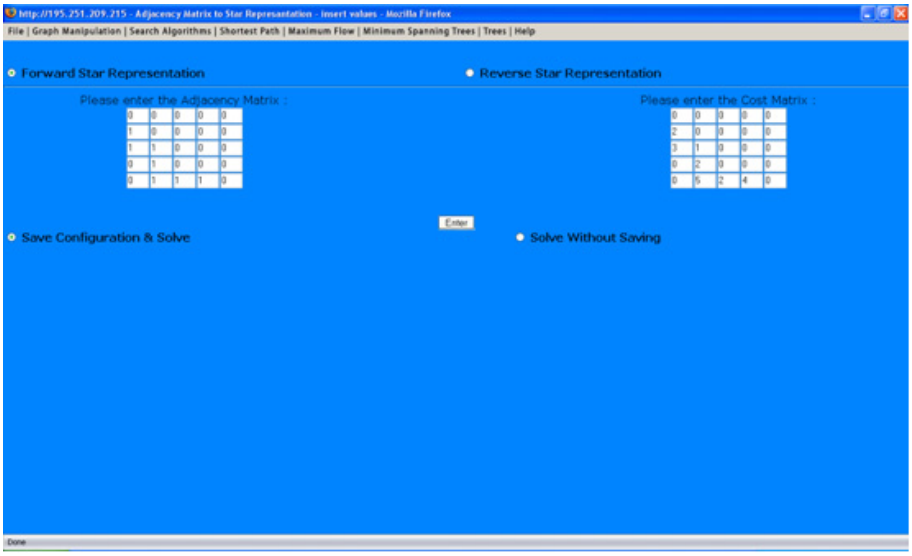


Σχήμα 5.4 Ο χρήστης επιλέγει τη διάσταση της μήτρας γειτονιάς

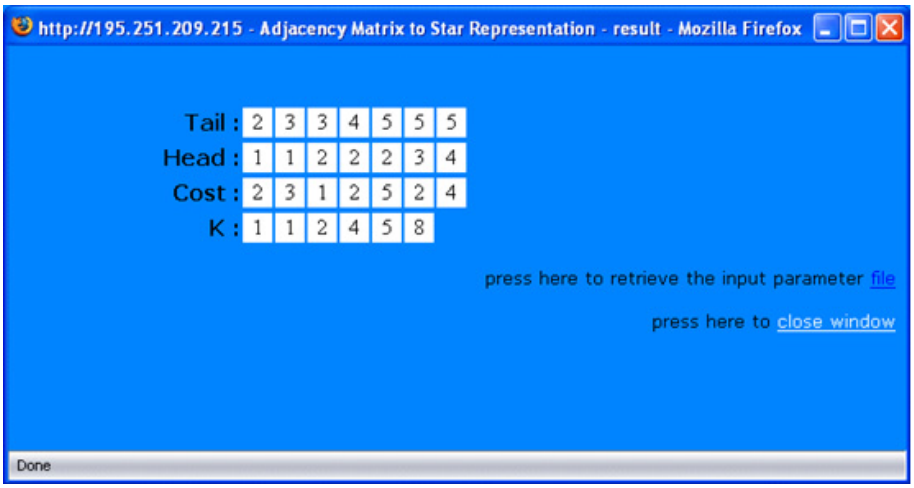
Αφού εισάγει τη τιμή 5 ως πλήθος των κόμβων για το παράδειγμα μας, αμέσως μετά σε ένα νέο παράθυρο, στο Σχήμα 5.5, θα απαιτεί από το χρήστη να καταχωρήσει τα στοιχεία των δύο μητρών A, W, που προαναφέρθηκαν καθώς επίσης και να ορίσει το είδος της αστεροειδούς αποθηκεύσεως.

Ομοίως με προηγουμένως η λύση θα παρουσιασθεί σε ένα νέο παράθυρο, βλέπε Σχήμα 5.6, όπου φαίνονται τα δεδομένα εξόδου με τη μορφή διανυσμάτων. Για την ακρίβεια υπάρχουν τα διανύσματα *ουρά* ή Tail, *κεφαλή* ή Head, *Κόστος* ή Cost και το διάνυσμα K το οποίο δείχνει τη θέση οποιουδήποτε κόμβου στο χρήστη.

Έστω ότι, ο χρήστης επιθυμεί να εξετάσει την ύπαρξη προσανατολισμένων κύκλων στο προηγούμενο δίκτυο. Τότε, ο χρήστης δια μέσω της κεντρικής γραμμής επιλογών θα πρέπει να επιλέξει Search Algorithms και έπειτα Identification of a Network's Directed Cycle. Με την ίδια διαδικασία

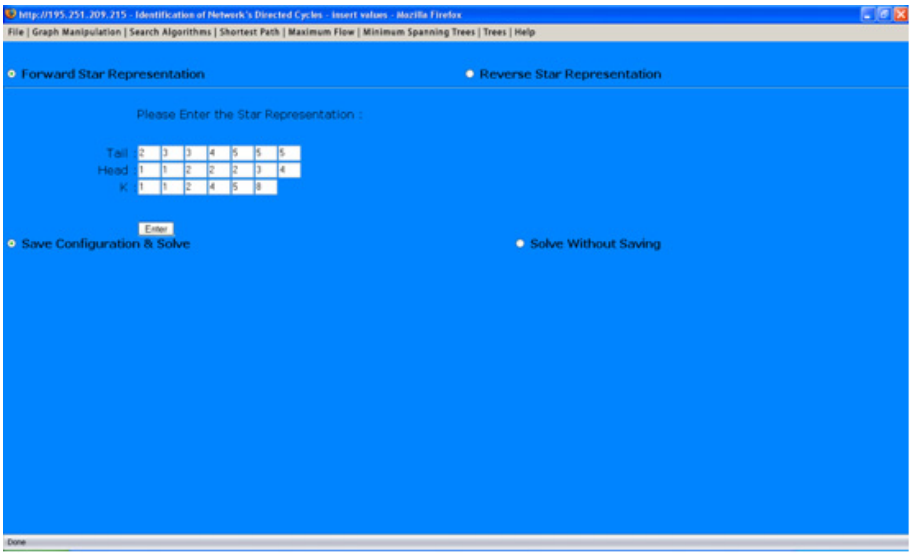


Σχήμα 5.5 Ο χρήστης συμπληρώνει τα αντίστοιχα πεδία



Σχήμα 5.6 Η αστεροειδή αποθήκευση των εξερχομένων τόξων

όπως και πριν επιλέγει τον τύπο δεδομένων (Αστεροειδή αποθήκευση), τη διάσταση του προβλήματος και το είδος της αστεροειδούς αποθήκευσης (εξερχομένων ή εισερχομένων τόξων). Κατόπιν, εισάγει τα δεδομένα εξόδου που είχε πάρει προηγουμένως από το δεύτερο παράδειγμα και τα εισάγει ως δεδομένα εισόδου, βλέπε Σχήμα 5.7, (μόνο που τώρα δεν χρειάζονται τα κόστη των τόξων, ώστε να ανιχνευθούν προσανατολισμένοι κύκλοι).

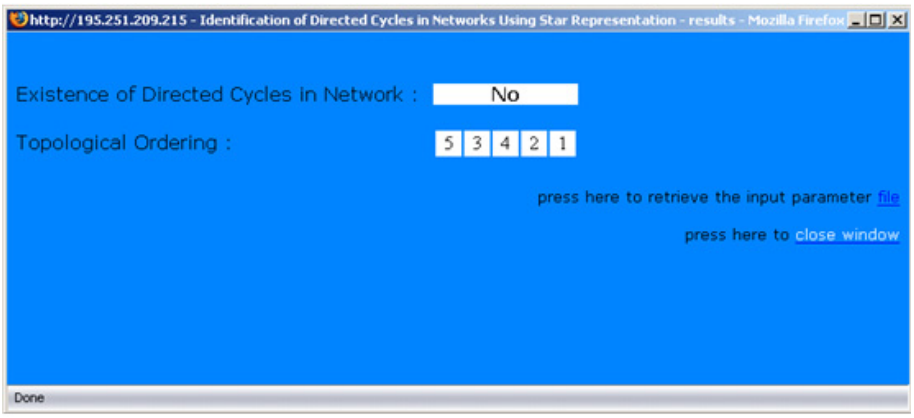


Σχήμα 5.7 Τα προηγούμενα δεδομένα εξόδου χρησιμοποιούνται τώρα ως δεδομένα εισόδου

Όντως μπορεί να επαληθευθεί και οπτικά από το Σχήμα 5.3, ότι δεν υπάρχουν προσανατολισμένοι κύκλοι στο δίκτυο. Συνεπώς, υπάρχει τοπολογική διάταξη για το δίκτυο μας και ο χρήστης είναι σε θέση να τη δει στο παράθυρο που προκύπτει στο Σχήμα 5.8.

Θυμίζεται, ότι μια διάταξη, order, κόμβων, (order(i), i = 1, 2, ..., n), καλείται τοπολογική διάταξη topological order, εάν ισχύει ότι διάταξη$(i) \prec$ διάταξη(j), $\forall (i,j) \in A$. Το δίκτυο που περιγράφηκε στο παράδειγμα είναι άκυκλο, αφού για κάθε τόξο (i,j) ισχύει ότι διάταξη$(i) \prec$ διάταξη(j). Για παράδειγμα, ο 5ος κόμβος έχει πάρει ετικέτα πρώτος και έχει ως εκ τούτου διάταξη$(5) = 1$, ο 3ος κόμβος έχει διάταξη$(3) = 2$, ο 4ος κόμβος έχει διάταξη$(4) = 3$, κ. ο. κ.

Το προτεινόμενο εργαλείο έχει αναπτυχθεί κυρίως για διδακτικούς σκοπούς, παρά για σκοπούς



Σχήμα 5.8: Το παράθυρο με τα δεδομένα εξόδου που πληροφορεί το χρήστη για τη πιθανή ύπαρξη τοπολογικής διάταξης.

βελτιστοποίησης μεγάλης κλίμακας, βλέπε χαρακτηριστικά το λογισμικό (Kennington and Helgason 1980). Λόγω λοιπόν του εκπαιδευτικού χαρακτήρα της εφαρμογής, ο χρήστης είναι περιορισμένος να εισάγει πρόβλημα με μέγιστη διάσταση 10 κόμβους και 15 τόξα. Αυτό γίνεται έτσι ώστε τα αποτελέσματα να απεικονίζονται σωστά στην οθόνη του χρήστη. Οι περιορισμοί που επιβάλλονται υλοποιούνται με δύο τρόπους. Πρώτον, με τη βοήθεια κώδικα JavaScript (client side) και δεύτερον βάσει ελέγχων στο πηγαίο κώδικα του καθενός αλγορίθμου (server side).

Επιπλέον, επιτελούνται και άλλοι προληπτικοί έλεγχοι στο πηγαίο κώδικα του καθενός αλγορίθμου. Για την ακρίβεια, οι τιμές των διαφόρων μητρών, οι οποίες εισάγονται απο το χρήστη, όπως για παράδειγμα τα στοιχεία της μήτρας γειτονιάς ή τα στοιχεία μιας λίστας τόξων, όπως το διάνυσμα ουρά, κεφαλή ή τα διανύσματα K / PK ελέγχονται επίσης. Γίνονται έλεγχοι εάν οι εισαχθέντες τιμές μίας μήτρας κόμβων - κόμβων είναι 0 ή 1. Ακόμη, ελέγχονται για παράδειγμα οι τιμές των κόστων των τόξων στον αλγόριθμο το Dijksta είναι μη αρνητικά. Επιπλέον, ελέγχονται εάν οι τιμές των στοιχείων των διανυσμάτων ουρά, κεφαλή είναι μεγαλύτερες από 0 και μικρότερες από n.

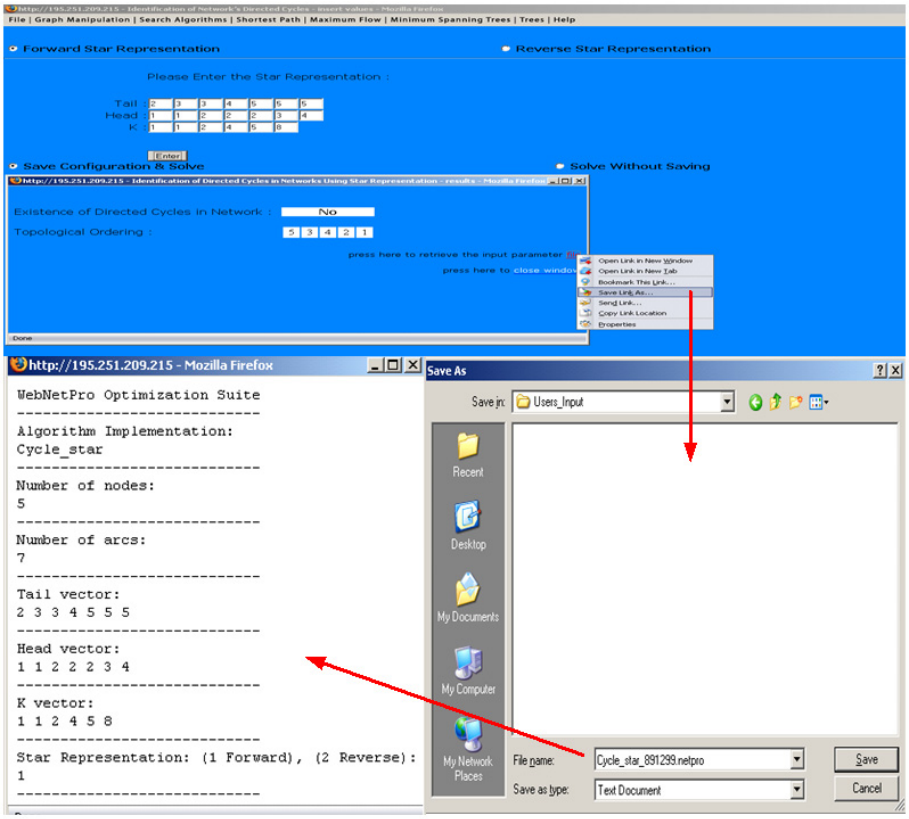
Παρ΄ όλα αυτά, δεν είναι δυνατόν να προβλεφθούν όλα τα πιθανά σενάρια. Δηλαδή, έστω η υπόθεση κατά την οποία κάποιος χρησιμοποιεί έναν αλγόριθμο εύρεσης ελαχίστων δρόμων σε

άκυκλα δίκτυα, ενώ οι εισαχθέντες τιμές είναι τέτοιες που αντιστοιχούν σε δίκτυο με κύκλο. Σε αυτήν την περίπτωση θα συμβούν λάθη κατά την εκτέλεση του κώδικα run time τα οποία είναι αναπόφευκτα. Συμπληρωματικά, έστω ότι ένας χρήστης επιθυμεί να χρησιμοποιήσει μια αστεροειδή αποθήκευση και συμπληρώνει λανθασμένα τις τιμές του διανύσματος K. Αυτό επίσης μπορεί να οδηγήσει σε παρόμοια λάθη.

Τέλος, πρέπει να αναφερθεί ότι κάποια από τα παραδείγματα αυτού του κεφαλαίου, είναι διαθέσιμα και στον ηλεκτρονικό ιστοχώρο του WebNetPro με τη μορφή βοηθητικής ιστοσελίδας (help file in HTML format), μεταφρασμένα στην αγγλική γλώσσα.

5.6 Αποθήκευση προβλημάτων σε αρχείο και μεταφόρτωση του αρχείου στο WebNetPro

Η εφαρμογή WebNetPro δίνει τη δυνατότητα στο χρήστη να αποθηκεύσει τα δικά του πλήρως παραμετροποιημένα προβλήματα. Με αυτόν τον τρόπο ο χρήστης είναι σε θέση να επαναϋποβάλλει οποιοδήποτε πρόβλημα δημιούργησε προηγουμένως, δίχως να απαιτείται να συμπληρώσει ξανά όλες τις τιμές στα αντίστοιχα πλαίσια κειμένου (text boxes). Για παράδειγμα, ας υποτεθεί ότι ο χρήστης στο Σχήμα 5.7, είχε επιλέξει στο κουμπί επιλογής radio button, "Solve and save the configuration". Τότε, στο επόμενο Σχήμα 5.8, ο χρήστης μπορεί πλέον να ανακτήσει το αρχείο με τις παραμέτρους του προβλήματος κάνοντας δεξί κλικ στον υπερσύνδεσμο "file" στο κάτω δεξιό μέρος του παραθύρου των αποτελεσμάτων. Επιλέγοντας "Save link as", (ή "Save as" στην περίπτωση που ο περιηγητής ιστού του χρήστη είναι ο Microsoft Internet Explorer) μπορεί να αποξηκεύσει το δικό του παραμετροποιημένο πρόβλημα στον προσωπικό του υπολογιστή. Ολόκληρη η διαδικασία περιγράφεται διαγραμματικά στο Σχήμα 5.9. Πρόσθετα στο Σχήμα 5.9, μπορεί να δει κάποιος την εσωτερική δομή του αρχείου.

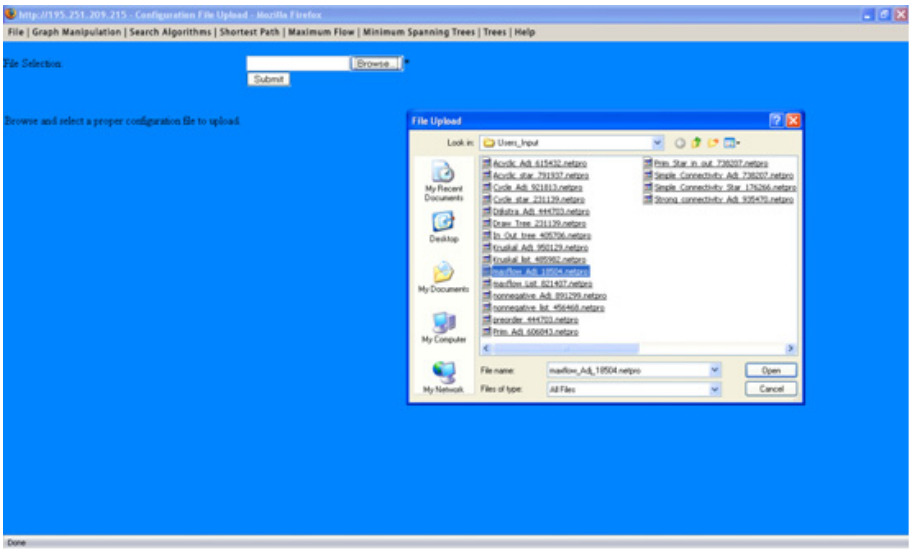


Σχήμα 5.9 Περιγραφή της διαδικασίας μεταφόρτωσης αρχείου

Η εφαρμογή WebNetPro σχεδιάστηκε ούτως ώστε να διαβάζει το αρχείο προβλήματος του χρήστη και να εξακριβώνει με τη χρήση ποιού αλγορίθμου δημιουργήθηκε το συγκεκριμένο αρχείο. Ας υποτεθεί για παράδειγμα ότι ο χρήστης είχε δημιουργήσει ένα αρχείο από ένα πρόβλημα εύρεσης τοπολογικής διάταξης. Μετά τη μεταφόρτωση του αρχείου, η φόρμα html, που αντιστοιχεί στον αλγόριθμο εύρεσης τοπολογικής διάταξης, θα επαναφορτώσει και θα περιέχει τις τιμές που υπάρχουν στο αρχείο. Αυτό είναι σημαντικό, διότι προλαμβάνονται με αυτόν τον τρόπο πιθανά λάθη του χρήστη. Πιο συγκεκριμένα, εάν ο χρήστης αποθηκεύσει μια μήτρα γειτονιάς αποτελούμενη από 5 γραμμές και στήλες και έναν κόμβο εκκίνησης για τον αλγόριθμο του Dijkstra για παράδειγμα, τότε ο χρήστης δεν μπορεί να μεταφορτώσει το συγκεκριμένο αρχείο σε μια html φόρμα η οποία έστω ότι αντιστοιχεί σε μια φόρμα εισαγωγής στοιχείων του αλγόριθμου εύρεσης μεγίστης ροής με υλοποίηση λίστας τόξων η ακόμη και σε μια μήτρα γειτονιάς με διαφορετική διάσταση.

Η ίδια προσέγγιση (user - centred) επίσης εφαρμόστηκε και στην ονοματολογία του καθενός αρχείου. Για την ακρίβεια, κάθε όνομα αρχείου έχει πρόθεμα αποτελούμενο απο κατάλληλες λέξεις - κλειδιά για να υποβοηθήσει το χρήστη στο να θυμηθεί τις παραμέτρους του προβλήματος. Το επίθεμα δε, κάθε όνομα αρχείου είναι ένας εξαψήφιος αριθμός που δημιουργείται από γεννήτρια τυχαίων αριθμών, με σκοπό να αποτρέψει την ύπαρξη αρχείων με πανομοιότυπο όνομα. Ολοκληρώνοντας, όλα τα αρχεία που δημιουργούνται έχουν κατάληξη *.netpro και έτσι μπορούν να ξεχωρίζουν τα αρχεία της εφαρμογής WebNetPro από λοιπά προσωπικά αρχεία του χρήστη. Η μεταφόρτωση ενός αρχείου αποτελεί μια αρκετά απλή διαδικασία. ο χρήστης επιλέγει από τη κεντρική μπάρα την εντολή Load file από το υπομενού File. Μετά ο χρήστης πρέπει να επιλέξει το απαραίτητο αρχείο προβλήματος αναζητώντας το τοπικά στον υπολογιστή του χρησιμοποιώντας το κουμπί Browse, όπως στο Σχήμα 5.10. Μόλις βρεθεί το επιθυμητό αρχείο, ο χρήστης πιέζει το κουμπί Submit.

Σε αυτή τη φάση, το αρχείο προβλήματος έχει μεταφορτωθεί στον εξυπηρετητή και μένει μονάχα να διαβαστεί και να συμπληρωθούν οι παράμετροι του προβλήματος στις αντίστοιχες φόρμες html



Σχήμα 5.10 Αναζήτηση του αρχείου προς μεταφόρτωση

της εφαρμογής. Αυτό επιτυγχάνεται, μόλις ο χρήστης πιέσει το κουμπί Configure. Πρέπει να παρατηρηθεί ότι το προηγούμενο κουμπί Submit έχει πια αντικατασταθεί με το κουμπί Configure. Σε αυτό το σημείο το αρχείο, (που αντιστοιχούσε σε πρόβλημα που είχε αποθηκευθεί παλαιότερα), έχει πια φορτωθεί επιτυχώς και ο χρήστης μπορεί είτε να το επιλύσει ξανά είτε να το τροποποιήσει εάν το επιθυμεί.

Η εφαρμογή WebNetPro δεν εμφανίζει τα ενδιάμεσα βήματα επίλυσης ή τις τρέχουσες ανά επανάληψη λύσεις ενός αλγορίθμου. Βοηθάει ένα φοιτητή - χρήστη να επαληθεύσει τη δική του/της λύση που υπολόγισε ή να μετατρέψει έναν τρόπο αποθήκευσης δικτύου σε κάποιο διαφορετικό. Μελλοντικές αναβαθμίσεις ή νεότερες εκδόσεις του λογισμικού δεν θα είναι παρατηρήσιμες στο φοιτητή που κάνει χρήση της εφαρμογής WebNetPro. Αφού είναι το ίδιο σαν να επισκέπτεται ο φοιτητής μια συνηθισμένη ιστοσελίδα του δικτύου. Κατά τη γνώμη μου, το προτεινόμενο λογισμικό θα πρέπει να χρησιμοποιείται από τους φοιτητές σε συνεργασία κατά προτίμηση με τον εισηγητή / καθηγητή τους. Είναι καλύτερο να συνδυάζεται η διδασκαλία της θεωρητικής πλευράς οποιουδήποτε αλγορίθμου με κάποιο συνοδευτικό λογισμικό ώστε να μπορεί να επαληθευθεί η κατανόηση ενός

αλγόριθμου. Παρ΄ όλα αυτά, είναι πολύ ευκολότερο να βρεί οποιοσδήποτε κάποιο βιβλίο, το οποίο να περιγράφει τις εσωτερικές πράξεις κατά την εκτέλεση ενός αλγορίθμου γραφημάτων, ή τις διαφορές της πληθώρας των διαφορετικών τρόπων αποθήκευσης ενός δικτύου, από ότι να βρεί ένα ελεύθερο, εύχρηστο, διαδικτυακό συνοδευτικό εργαλείο το οποίο να είναι σε θέση να βοηθήσει κάποιο φοιτητή.

ΚΕΦΑΛΑΙΟ 6

Συμπεράσματα και μελλοντική έρευνα

6.1 Σύνοψη αποτελεσμάτων

Σε αυτή τη Διατριβή παρουσιάστηκε κυρίως ένας νέος αλγόριθμος δύο δρόμων για το Πρόβλημα της Ροής Ελαχίστου Κόστους. Ο προτεινόμενος αλγόριθμος περιγράφτηκε αποκλειστικά με όρους δικτύων και αξιοποίησε κατάλληλες δομές δεδομένων για το σκοπό αυτό. Επιπλέον, αποδείχτηκε μαθηματικά η περατότητα του αλγορίθμου για μη εκφυλισμένα προβλήματα και προγραμματίστηκε, όσο το δυνατό, με περισσότερο αποτελεσματικό τρόπο. Έγινε εκτενής χρήση τεχικών διανυσματοποίησης vectorizing, (στον κώδικα έχουν περιοριστεί σημαντικά οι βρόχοι του DO - END DO) αλλά και χρήση δυναμικών δομών (για παράδειγμα allocatable arrays) και σε καμμία περίπτωση στατικών.

Επιπλέον, διεξήχθηκε υπολογιστική μελέτη του NEPSA με προβλήματα μεγάλης διάστασης. Τα αποτελέσματα της μελέτης απεικονίζουν την εμπειρική συμπεριφορά του προτεινόμενου αλγορίθμου και ιδιαίτερα τον τρόπο μεταβολής του μέσου χρόνου του επεξεργαστή CPU time, αλλά και του μέσου αριθμού των επαναλήψεων.

Τέλος, ως παράπλευρο αποτέλεσμα παρουσιάστηκε ακόμη και ένα ολοκληρωμένο λογισμικό επίλυσης προβλημάτων Δικτυακής Βελτιστοποίησης, διαθέσιμο είτε για εγκατάσταση σε τοπικό

υπολογιστή, είτε για εξ΄ αποστάσεως χρήση μέσω Διαδικτύου. Το λογισμικό αυτό αναπτύχθηκε παράλληλα μη την ανάπτυξη του NEPSA, κυρίως λόγω της συνεπικούρησης διδασκαλίας του γράφοντος στο μάθημα *Δικτυακός Προγραμματισμός*.

6.2 Υποδείξεις για μελλοντική έρευνα

Στη διεθνή βιβλιογραφία συναντά κανείς πολλές άλλες αποδοτικές δομές δεδομένων. Κάποιες απο αυτές είναι τα *δυναμικά δένδρα* όπως περιγράφονται στην εργασία (Goldberg, Grigoriadis, and Tarjan 1991), ή οι σωροί Fibonacci όπως περιγράφονται στην εργασία (Fredman and Tarjan 1987). Η θεωρητική πολυπλοκότητα ενός αλγορίθμου καθορίζεται σε σημαντικό βαθμό από τις δομές δεδομένων που χρησιμοποιούνται για την ανανέωση όλων των απαραιτήτων μεταβλητών και διανυσμάτων / πινάκων. Οπότε, παρουσιάζει εξαιρετικό ενδιαφέρον να χρησιμοποιηθούν και αυτές οι δομές δεδομένων στην υλοποίηση του αλγορίθμου NEPSA.

Εάν μπορέσουν να γίνουν κάποιες βελτιώσεις σχετικά με αποτελεσματικές μεθόδους προγραμματισμού όπως είναι οι τεχνικές κλιμάκωσης, (scaling techniques), τότε θα είναι δυνατό να συγκριθεί ο αλγορίθμος NEPSA με κάποιους αλγόριθμους που θεωρούνται ως (state-of-the-art) οι καλύτερες υλοποιήσεις όπως για παράδειγμα, πέρα αυτών που προαναφέρθηκαν, ο αλγόριθμος (Goldberg and Tarjan 1987). Άλλωστε έχουν διεξαχθεί υπολογιστικές μελέτες οι οποίες καταδεικνύουν ότι η χρήση αυτών των τεχνικών, είναι ιδιαίτερα αποτελεσματική (Bland and Jensen 1992).

Παρ΄ όλα αυτά θα απαιτούσε πρώτα να αναπτυχθούν κατάλληλοι κανόνες αντί-κύκλωσης και να ελεχθεί η συμπεριφορά του αλγορίθμου NEPSA σε εκφυλισμένα προβλήματα. Περαιτέρω υπολογιστικές μελέτες θα δείξουν εάν ο NEPSA είναι πράγματι αποτελεσματικός ή όχι στην πράξη. Επιπλέον, είναι ενδιαφέρον να αναπτυχθούν ιδίου τύπου αλγόριθμοι για άλλα γνωστά προβλήματα δικτύων, όπως για παράδειγμα το πρόβλημα εύρεσης ελαχίστων δρόμων.

Πολύ ενδιαφέρον σίγουρα θα παρουσιάσει μια παραλληλοποίηση του προτεινόμενου αλγόριθ-

μου NEPSA και διεξαγωγή εκτενούς υπολογιστικής μελέτης με άλλες παράλληλες υλοποιήσεις αλγορίθμων για το ΠΡΕΚ. Ήδη, κάποια αρχικά αποτελέσματα επάνω σε παράλληλες υλοποιήσεις αλγορίθμων εξωτερικών σημείων έχουν αρχίσει να εμφανίζονται με την εργασία (Badr, Moussa, Paparrizos, Samaras, and Sifaleras 2006). Ειδικότερα δε, τα τελευταία χρόνια με την αλματώδη εξέλιξη της τεχνολογίας GRID, ένας ερευνητής έχει πια πρόσβαση σε σημαντικούς υπολογιστικούς πόρους, που υπό άλλες συνθήκες παλαιότερα θα έπρεπε να ανήκει μονάχα σε κάποιο πανίσχυρο ερευνητικό κέντρο, (π.χ. CERN). Πιο συγκεκριμένα, η Ελλάδα συμμετέχει δια μέσου του Εθνικού Δικτύου Έρευνας και Τεχνολογίας ΕΔΕΤ (GRNET), ο οποίος φορέας είναι συντονιστής της Ομάδας Εργασίας Hellasgrid, σε διάφορα προγράμματα προώθησης της τεχνολογίας GRID.

Αξίζει να αναφερθεί ότι αποτελεί νέο ερευνητικό πεδίο ο συγκερασμός μεθόδων Επιχειρησιακής Έρευνας και Τεχνητής Νοημοσύνης, όπως φαίνεται για παράδειγμα στις εργασίες (Seow and How 2002) αλλά και (Rashidi and Tsang 2005). Η εφαρμογή αλγορίθμων εξωτερικών σημείων για το πρόβλημα της ανάθεσης σε πολυπρακτορικά συστήματα, π.χ. (Karagiannis, Paparrizos, Samaras, and Sifaleras 2004a), αποτελεί επιπλέον ανοιχτό πρόβλημα. Προς την κατεύθυνση αυτή κινείται και το ερευνητικό πρόγραμμα '*ΠΥΘΑΓΟΡΑΣ Ι: Ενίσχυση Ερευνητικών Ομάδω στα Πανεπιστήμια, (Υπολογιστικές Βελτιώσεις αλγορίθμων Συνδυαστικής Βελτιστοποίησης)*', το οποίο εκτελείται από κοινού από την ερευνητική ομάδα '*Υπολογιστικής Επιχειρησιακής Έρευνας* (Computational Operations Research)' του Τμήματος Εφαρμοσμένης Πληροφορικής, του Πανεπιστημίου Μακεδονίας, και της ομάδας *Ευφυών Συνεργατικών Συστημάτων, (Intelligent Cooperative Systems)* του Τμήματος Μηχανικών Πληροφοριακών & Επικοινωνιακών Συστημάτων, του Πανεπιστημίου Αιγαίου. Ήδη η προσπάθεια αυτή έχει αποδώσει κάποια προκαταρκτικά ερευνητικά αποτελέσματα, (Karagiannis, Paparrizos, Samaras, and Sifaleras 2004b).

Επιπλέον, για να μπορέσει να γίνει μια εκτίμηση της εκπαιδευτικής αξίας της εφαρμογής WebNetPro, ένα ερωτηματολόγιο, το οποίο θα είναι διαδικτυακά διαθέσιμο, μπορεί να ενσωματωθεί

στην ιστοσελίδα της εφαρμογής. Στην κατεύθυνση αυτή, η εφαρμογή WebNetPro είναι προγραμματισμένο να αξιολογηθεί προσεκτικά κατά τη διάρκεια των διαλέξεων εντός του εργαστηρίου του προπτυχιακού μαθήματος ¨Δικτυακή Βελτιστοποίηση¨, του Τμήματος Εφαρμοσμένης πληροφορικής του Πανεπιστημίου Μακεδονίας. Επειδή κατά κύριο λόγο φοιτητές είναι όσοι εργάζονται με την εφαρμογή WebNetPro, τά σχόλια και οι υποδείξεις τους είναι ανεκτίμητη βοήθεια στην προσπάθεια για περαιτέρω ανάπτυξη και βελτίωση της εφαρμογής.

Είναι πιο σημαντικό να εξακριβωθεί με ποιό τρόπο οι απομακρυσμένοι χρήστες / φοιτητές, κάνουν χρήση του συγκεκριμένου λογισμικού, παρά απλά να εξηγείται στους χρήστες ποιά θα είναι τα πιθανά ωφέλη που θα έχουν, όπως περιγράφεται από τους (Hundhausen, Douglas, and Stasko 2002) και (Lawrence, Badre, and Stasko 1994). Άλλοι ερευνητές δείχνουν τη σωστή ενσωμάτωση και χρήση εφαρμογών οπτικοποίησης αλγορίθμων σε διαδακτικά περιβάλλοντα, όπως οι (Byrne, Catrambone, and Stasko 1999). Τέτοιου είδους εκπαιδευτικές εφαρμογές δείχνουν πολύ ενθαρρυντικές για την εξ΄ αποστάσεως εκπαίδευση. Παρ΄ όλα αυτά, θα πρέπει να σημειωθεί ότι άλλες έρευνες αναφέρουν ότι παιδαγωγικά οφέλη μπορούν μόνο μερικώς να αποδοθούν σε τεχνολογίες οπτικοποίησης αλγορίθμων, όπως περιγράφεται από τους (Byrne, Catrambone, and Stasko 1999).

Τέλος, ο αλγορίθμος NEPSA θα ενσωματωθεί στο λογισμικό Δικτυακής Βελτιστοποίησης WebNetPro. Με αυτόν τον τρόπο, το WebNetPro θα επεκτείνει τις δυνατότητες του με νέους αλγόριθμους για το ΠΡΕΚ.

Μια μελλοντική δουλειά θα είναι επίσης η ανάπτυξη της εφαρμογής LinPro σε διαδικτυακή μορφή και ενοποίηση με το WebNetPro. Η εφαρμογή LinPro παρουσιάστηκε από τον Σιφαλέρα Α. στο 15ο Πανελλήνιο Συνέδριο της Ελληνικής Εταιρίας Επιχειρησιακών Ερευνών (HELORS) το 2002, με την εργασία (Dosios, Paparrizos, Papatzikos, and Sifaleras 2002). Το εκπαιδευτικό πληροφοριακό σύστημα LinPro είναι υλοποιημένο σε Matlab και ασχολείται με την επίλυση προβλημάτων Γραμμικής Βελτιστοποίησης. Η συνένωση και των δύο εφαρμογών μπορεί να αποτελέσει ένα χρήσιμο

πακέτο εφαρμογών Μαθηματικού Προγραμματισμού, κυρίως για διδακτικούς σκοπούς.

ΑΝΑΦΟΡΕΣ

Achatz, A., K. Paparrizos, N. Samaras, and K. Tsiplidis (2002). A forest exterior point algorithm for assignment problems. In M. Pardalos, A. Midgalas, and R. Buckard (Eds.), *Combinatorial and Global Optimization*, Combinatorial and Global Optimization, pp. 1–10. Word Scientific Publishing Co.

Achatz, H., P. Kleinschmidt, and K. Paparrizos (1991). A dual forest algorithm for the assignment problem. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science 4*, 1–11.

Ahuja, R., T. L. Magnanti, J. B. Orlin, and M. R. Reddy (1995). Applications of network optimization. In M. O. Ball, T. L. M. C. L. Monma, and G. L. Nemhauser (Eds.), *Network Models*, Volume 7 of *Handbooks of Operations Research and Management Science*, pp. 1–83. North-Holland, Amsterdam: Elsevier.

Ahuja, R. K., A. V. Goldberg, J. B. Orlin, and R. E. Tarjan (1992). Finding minimum-cost flows by double scaling. *Mathematical Programming 53*(3), 243–266.

Ahuja, R. K., T. L. Magnanti, and J. B. Orlin. (1993). *Network Flows: Theory, Algorithms and Applications.* Englewood Cliffs, NJ: Prentice Hall.

Andreou, D., K. Paparrizos, N. Samaras, and A. Sifaleras (2005). Application of a new network-enabled solver for the assignment problem in computer-aided education. *Journal*

of Computer Science 1(1), 19–23.

Andreou, D., K. Paparrizos, N. Samaras, and A. Sifaleras (2006). A visualization of the network exterior primal simplex algorithm for the minimum cost network flow problem. *Submitted for possible publication in International Journal for Technology in Mathematics Education (IJTME)*.

Armstrong, R. D. and Z. Jin (1997). A new strongly polynomial dual network simplex algorithm. *Mathematical Programming 78*(2), 131–148.

Badr, E. S., M. Moussa, K. Paparrizos, N. Samaras, and A. Sifaleras (2006, 8-10 December). Some computational results on mpi parallel implementation of dense simplex method. *Enformatika Journal (Transactions on Engineering, Computing and Technology) 17*, 228–231. presented in the Seventeenth International Conference on Computer & Information Science and Engineering, (CISE 2006), Cairo, Egypt.

Badr, E. S., K. Paparrizos, N. Samaras, and A. Sifaleras (2005, 16-18 June). On the basis inverse of the exterior point simplex algorithm. In *Proceedings of the 17th National Conference of Hellenic Operations Research Society, HELORS*, Rio, Greece, pp. 677–687.

Balinski, M. L. (1985). Signature methods for the assignment problem. *Operations Research 33*(3), 527–537.

Baloukas, T., K. Paparrizos, and A. Sifaleras (2005, 25-28 May). Promoting operations research education using a new web - accessible didactic tool. In *Proceedings of the 7th Balkan Conference on Operational Research, (BACOR 05)*, Constanta, Romania.

Barr, R., F. Glover, and D. Klingman (1979). Enhancements of spanning tree labelling procedures for network optimization. *INFOR 17*(1), 16–34.

Barr, R. S., F. Glover, and D. Klingman (1977). The alternating basis algorithm for assign-

ment problems. *Mathematical Programming 13*(1), 1–13.

Bazaraa, M. S., J. J. Jarvis, and H. D. Sherali (2005). *LINEAR PROGRAMMING AND NETWORK FLOWS* (3rd ed.). John Wiley and Sons, Inc.

Beale, E. M. L. (1955). Cycling in the dual simplex algorithm. *Naval Research Logistics Quarterly 2*(4), 269–275.

Bertsekas, D. P. (1991). *Linear Network Optimization: Algorithms and Codes.* Cambridge, MA: MIT Press.

Bertsekas, D. P. and P. Tseng (1988). Relaxation methods for minimum cost ordinary and generalized network flow problems. *Operations Research 36*(1), 93–114.

Bertsekas, D. P. and P. Tseng (1994). Relax-iv : a faster version of the relax code for solving minimum cost flow problems. Technical report, Massachusetts Institute of Technology, Laboratory for Information and Decision Systems.

Bland, R. G. (1977). New finite pivoting rules for the simplex method. *Mathematics of Operations Research 2*(2), 103–107.

Bland, R. G., J. Cheriyan, D. L. Jensen, and L. Ladanyi (1993). An empirical study of min cost flow algorithms for the minimum-cost flow problem. *DIMACS Series in Discrete Mathematics and Theoretical Computer Science 12*, 119–157.

Bland, R. G. and D. L. Jensen (1992). On the computational behavior of a polynomial-time network flow algorithm. *Mathematical Programming 54*(1-3), 1 – 39.

Byrne, M. D., R. Catrambone, and J. T. Stasko (1999). Evaluating animations as students aids in learning computer algorithms. *Computers and Education 33*(4), 253 – 278.

Clark, R. H., J. L. Kennington, R. R. Meyer, and M. Ramamurti (1992). Generalized networks: Parallel algorithms and an empirical analysis. *Journal on Computing 4*(2), 132–145.

Colgan, L. (2000). Matlab in first year engineering mathematics. *International Journal of Mathematical Education in Sciences and Technology 31*(1), 15–25.

Cunningham, W. and J. Klincewicz (1983). On cycling in the network simplex method. *Mathematical Programming 26*, 182–189.

Cunningham, W. H. (1976). A network simplex method. *Mathematical Programming 11*, 105–116.

Cunningham, W. H. (1979). Theoretical properties of the network simplex method. *Mathematics of Operations Research 4*(2), 196–208.

Curet, N. D. (1997). Applying steepest - edge techniques to a network primal - dual algorithm. *Computers & Operations Research 24*(7), 601–609.

Curet, N. D. (1998). Implementation of a steepest-edge primal - dual simplex method for network linear programs. *Annals of Operations Research 81*(0), 251 – 270.

Dagdilelis, V. and M. Satratzemi (1998). Didagraph: A software for teaching graph theory algorithms. *SIGSE Bulletin, ACM Press 30*(3), 64–68.

Dantzig, G. B. (1951). Applications of the simplex method to a transportation problem. In T. C. Koopmans (Ed.), *Activity Analysis of Production and Allocation*. New York: Wiley.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik 1*(1), 269–271.

Dosios, K., K. Paparrizos, N. Papatzikos, and A. Sifaleras (2002). Linpro, an educational informational system for linear programming. In *Proceedings of the 15th National Conference of HELORS*, Tripoli, Greece.

Dosios, K., K. Paparrizos, N. Samaras, and A. Sifaleras (2003a). An efficient modification of the primal-dual two paths simplex algorithm. In *Proceedings of the International Confere-*

nce on Computational Methods in Sciences and Engineering, Volume 1, Kastoria, Greece, pp. 144–147. World Scientific Publishing Co., Inc.

Dosios, K., K. Paparrizos, N. Samaras, and A. Sifaleras (2003b). Netpro, an educational platform for network optimization. In *Proceedings of the 16th National Conference of HELORS*, Volume 1, Larissa, Greece, pp. 287–295.

Dunn, P. K. and C. Harman (2002). Calculus demonstrations using matlab. *International Journal of Mathematical Education in Sciences and Technology 33*(4), 584–596.

Edmonds, J. (1965). Paths, trees, and flowers. *Canadian Journal of Mathematics 17*, 449–467.

Edmonds, J. and R. M. Karp (1972). Theoretical improvements in algorithmic efficiency for network flow problems. *Journal of the ACM 19*(2), 248–264.

Ervolina, T. R. and S. T. McCormick (1993). Two strongly polynomial cut cancelling algorithms for minimum cost network flow. *Discrete Applied Mathematics 46*(2), 133–165.

Farley, A. R. (1980). Levelling terrain trees: A transhipment problem. *Information Processing Letters 10*(4/5), 189–192.

Ford, L. and D. Fulkerson (1957). A simple algorithm for finding maximal network flows and an application to the hitchcock problem. *Canadian Journal of Mathematics 9*, 210–218.

Fredman, M. C. and R. E. Tarjan (1987, July). Fibonacci heaps and their uses in improved network optimization algorithms. *Journal of the ACM 34*(3), 596–615.

Fulkerson, D. R. (1961). An out-of-kilter method for minimal cost flow problems. *J. SIAM 9*(1), 18–27.

Gabow, H. N. and R. E. Tarjan (1989). Faster scaling algorithms for network problems. *SIAM Journal on Computing 18*(5), 1013–1036.

Galil, Z. and E. Tardos (1988). A $o(n^2(m + nlogn)logn)$ min-cost flow algorithm. *Journal of the ACM 35*(2), 374–386.

Gassa, S. I. and S. Vinjamurib (2004). Cycling in linear programming problems. *Computers & Operations Research 31*(2), 303–311.

Gassner, B. J. (1964). Cycling in the transportation problem. *Naval Research Logistics 11*(1), 43–58.

Gavish, B., P. Schweitzer, and E. Shlifer (1977). The zero pivot phenomenon in transportation and assignment problems and its computational implications. *Mathematical Programming 12*(1), 226 – 240.

Geranis, G. (2007). *Data Trasmission Algorithms over Networks (work in progress)*. Ph. D. thesis, Department of Applied Informatics, University of Macedonia.

Glover, F., D. Karney, and D. Klingman (1972). The augmented predecessor index method for locating stepping stone paths and assigning dual prices in distribution problems. *Transportation Science 6*(2), 171–179.

Glover, F., D. Karney, D. Klingman, and A. Napier (1974). A computation study on start procedures, basis change criteria, and solution algorithms for transportation problems. *Management Science 20*(5), 793–813.

Glover, F. and D. Klingman (1975). Real world applications of network related problems and breakthroughs in solving them efficiently. *ACM Transactions on Mathematical Software (TOMS) 1*(1), 47–55.

Glover, F., D. Klingman, and A. Napier (1972a). Basic dual feasible solutions for a class of generalized networks. *Operations Research 20*(1), 126–136.

Glover, F., D. Klingman, and A. Napier (1972b). An efficient dual approach to network

problems. *Opsearch 9*(1), 1–18.

Glover, F., D. Klingman, and N. Phillips. (1992). *Network Models in Optimization and Their Applications in Practice* (1st ed.). New York: Wiley.

Glover, F., D. Klingman, and J. Stutz (1973). Extensions of the augmented predecessor index method to generalized transportation problems. *Transportation Science 7*(4), 377–384.

Glover, F., D. Klingman, and J. Stutz (1974). Augmented threaded index method for network optimization. *INFOR 12*(3), 293–298.

Goldberg, A. and R. Tarjan (1987). Solving minimum-cost flow problems by successive approximation. In *Proceedings of the 19th annual ACM conference on Theory of computing*, New York, pp. 7–18. ACM Press.

Goldberg, A. and R. Tarjan (1989). Finding minimum-cost circulations by canceling negative cycles. *Journal of the ACM 36*(4), 873–886.

Goldberg, A. V. (1991). The grid-on-torus (goto) generator. In *Proceedings of the First DIMACS International Algorithm Implementation Challenge*.

Goldberg, A. V. (1997). An efficient implementation of a scaling minimum-cost flow algorithm. *Journal of Algorithms 22*(1), 1–29.

Goldberg, A. V., M. D. Grigoriadis, and R. E. Tarjan (1991). Use of dynamic trees in a network simplex algorithm for the maximum flow problem. *Mathematical Programming 50*(3), 277–290.

Goldberg, A. V., E. Tardos, and R. E. Tarjan (1989). Network flow algorithms. Technical report, Department of Computer Science, Stanford University.

Goldfarb, D. and M. D. Grigoriadis (1988). A computational comparison of the dinic and

network simplex methods for maximum flow. *Annals of Operations Research 13*(1), 83–123.

Goldfarb, D. and J. Hao (1992). Polynomial-time primal simplex algorithms for the minimum cost network flow problem. *Algorithmica 8*(1), 145–160.

Greenberg, J. (1999). Java applet for the teaching of dijkstra algorithm. Personal web page, (University of Colorado).

Grigoriadis, M. (1984). An efficient implementation of the network simplex method. *Mathematical Programming Study 26*, 83–111.

Gupta, S. K. (1985). *Linear Programming and Network Models*. New Delhi: Affiliated East-West Press Private Limited.

Hall, J. and K. McKinnon (2004). The simplest examples where the simplex method cycles and conditions where expand fails to prevent cycling. *Mathematical Programming 100*(1), 133–150.

Hoffman, A. J. (1953). Cycling in the simplex method. Technical Report 2974, National Bureau of Standards, Washington D. C., USA.

Hultz, J., D. Klingman, and R. Russell (1976). An advanced dual basic feasible solution for a class of capacitated generalized networks. *Operations Research 24*(2), 301–313.

Hundhausen, C. D., S. A. Douglas, and J. T. Stasko (2002, Jun). A meta-study of algorithm visualization effectiveness. *Journal of Visual Languages and Computing 13*(3), 259–290.

ILOG, I. *CPLEX Network Optimizer*. 1080 Linda Vista Ave.

Karagiannis, P., I. Markelis, K. Paparrizos, N. Samaras, and A. Sifaleras (2006). E - learning technologies: employing matlab web server to facilitate the education of mathematical

programming. *International Journal of Mathematical Education in Science and Technology 37*(7), 765–782.

Karagiannis, P., K. Paparrizos, N. Samaras, and A. Sifaleras (2004a, 19 - 23 April). Employing mobile agents to secure local networks. In *Proceedings of the 1st IEEE International Conference on Information & Communication Technologies: From Theory to Applications, (ICTTA 2004)*, Damascus, Syria, pp. 501–502.

Karagiannis, P., K. Paparrizos, N. Samaras, and A. Sifaleras (2004b). A short bibliography on the assingment problem and its applications. Technical report, Department of Applied Informatics, University of Macedonia. Deliverable in Work Package 1, from the Research Project (EPEAEK II) PYTHAGORAS I.

Karagiannis, P., K. Paparrizos, N. Samaras, and A. Sifaleras (2005, 25-28 May). A new simplex type algorithm for the minimum cost network flow problem. In *Proceedings of the 7th Balkan Conference on Operational Research, (BACOR 05)*, Constanta, Romania.

Karmarkar, N. (1984). A new polynomial time algorithm for linear programming. *Combinatorica 4*(4), 373–395.

Kennington, J. L. and R. V. Helgason (1980). *Algorithms for Network Programming*. John Wiley and Sons, New-York.

Khuri, S. and K. Holzapfel (2001, SEP). Evega: An educational visualization environment for graph algorithms. *ACM SIGCSE Bulletin 33*(3), 101–104.

Klingman, D., A. Napier, and J. Stutz (1974). Netgen: A program for generating large scale capacitated assignment, transportation, and minimum cost flow networks. *Management Science 20*(5), 814–821.

Lawrence, A. W., A. N. Badre, and J. T. Stasko (1994). Empirically evaluating the use

of animations to teach algorithms. In *Proceedings of the IEEE Symposium on Visual Languages*, St. Louis, MO.

Lazaridis, V., K. Paparrizos, N. Samaras, and A. Sifaleras (2007). Visual linprog: A web-based educational software for linear programming. *Computer Applications in Engineering Education 15*(1), 1–14.

Lee, Y. and J. B. Orlin (1991). Computational testing of a network simplex algorithm. In *Proceedings of the First DIMACS International Algorithm Implementation Challenge*.

Marchand, P. and O. T. Holland (2002). *Graphics and GUIs with MATLAB* (3rd ed.). CHAPMAN AND HALL/CRC.

Maros, I. (1993). A practical anti-degeneracy row selection technique in network linear programming. *Annals of operations Research 47*(2), 431 – 442.

Morris, W. (2004). Personal communication with dimacs unit computing manager. CoRE 427 732-445-4592, wmorris@dimacs.rutgers.edu.

Mulvey, J. M. (1978). Pivot strategies for primal-simplex network codes. *Journal of the ACM 25*(2), 266–270.

Orlin, J. B. (1993). A faster strongly polynomial minimum cost flow algorithm. *Operations Research 41*(2), 338–350.

Orlin, J. B. (1997). A polynomial time primal network simplex algorithm for minimum cost flows. *Mathematical Programming 78*(2), 109–129.

Orlin, J. B., S. Plotkin, and E. Tardos (1993). Polynomial dual network simplex algorithms. *Mathematical Programming 60*(3), 255–276.

Papamanthou, C., K. Paparrizos, and N. Samaras (2004). Computational experience with exterior point algorithms for the transportation problem. *Journal of Applied Mathematics*

and Computation 158(2), 459–475.

Papamanthou, C., K. Paparrizos, N. Samaras, A. Sifaleras, and K. Stergiou (2006). On the initialization methods of simplex type algorithm for the assignment problem. *Submitted for possible publication in Computational Optimization and Applications*.

Paparrizos, K. (1991). An infeasible (exterior point) simplex algorithm for assignment problems. *Mathematical Programming 51*(1), 45–54.

Paparrizos, K. (1996a, October). Exterior point simplex algorithms: simple and short proofs of correctness. In *Proceedings of the XXIII SYMOPIS*, Zlatibor, pp. 13–18.

Paparrizos, K. (1996b). A non improving simplex algorithm for transportation problems. *RAIRO Operations Research 30*(1), 1–15.

Paparrizos, K. (2004). *MATLAB 6.5*. Thessaloniki: ZYGOS Publications.

Paparrizos, K., N. Samaras, and A. Sifaleras (2004, 10 -12 May). A learning tool for the visualization of general directed or undirected rooted trees. In K. Morgan and J. M. Spector (Eds.), *WIT Transactions on Information and Communication Technologies*, Volume 30, Chapter The Internet Society: Advances in Learning, Commerce and Security, pp. 205–213. Skiathos, Greece: WIT Press. presented in the 1st International Conference on New Learning Paradigms and New Learning Tools, (New Learning 2004).

Paparrizos, K., N. Samaras, and A. Sifaleras (2005). *Network Optimization*. University of Macedonia. University Notes.

Paparrizos, K., N. Samaras, and A. Sifaleras (2006a). A new exterior simplex type algorithm for the minimum cost network flow problem. *Submitted for possible publication in Computers & Operations Research*.

Paparrizos, K., N. Samaras, and A. Sifaleras (2006b, 13-15 December). On the empirical

behaviour of a new network exterior point simplex algorithm for the minimum cost network flow problem. In *Proceedings of the Veszprém Optimization Conference: Advanced Algorithms (VOCAL 2006)*, Veszprém, Hungary, pp. 66.

Paparrizos, K., N. Samaras, and A. Sifaleras (2006c, 3-5 August). Some preliminary results on the cycling problem for the network exterior point simplex algorithm. In *Proceedings of the Seventh International Conference on Operations and Quantitative Management, (ICOQM-VII)*, Jaipur, India, pp. 536–540.

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

Plotkin, S. A. and E. Tardos (1990). Improved dual network simplex. In *Proceedings of the 1st Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 367–376.

Portugal, L., M. Resende, G. Veiga, and J. Judice (2000). A truncated primal-infeasible dual-feasible network interior point method. *Networks 35*(2), 91–108.

Rashidi, H. and E. P. K. Tsang (2005). Applying the extended network simplex algorithm to dynamic automated guided vehicles scheduling. In *Proceedings of the 2nd Multidisciplinary International Conference on Scheduling : Theory and Applications (MISTA 2005)*, pp. 677–692.

Resende, M. and G. Veiga (1993). An efficient implementation of a network interior point method. In D. Johnson and C. McGeoch (Eds.), *NETWORK FLOWS AND MATCHING*, Providence, RI, pp. 299–349. First DIMACS Implementation Challenge: American Mathematical Society.

Resende, M. and G. Veiga (2003). An annotated bibliography of network interior point me-

thods. *Networks 42*(2), 114–121.

Rock, H. (1980). *Scaling techniques for minimal cost network flows*, pp. 181–191. Munich: C. Hansen.

Schrage, L. (1979). A more portable fortran random number generator. *ACM Transactions on Mathematical Software (TOMS) 5*(2), 132–138.

Seow, K. T. and K. Y. How (2002). Collaborative assignment: a multiagent negotiation approach using bdi concepts. In *Proceedings of the first international joint conference on Autonomous agents and multiagent systems, (AAMAS 2002)*, pp. 256–263. Bologna, Italy: ACM.

Sokkalingam, P. T., R. K. Ahuja, and J. B. Orlin (2000). New polynomial-time cycle-canceling algorithms for minimum cost flows. *Networks 36*(1), 53–63.

Tardos, E. (1985). A strongly polynomial minimum cost circulation algorithm. *Combinatorica 5*(3), 247–255.

Tarjan, R. E. (1991). Efficiency of the primal network simplex algorithm for the minimum-cost circulation problem. *Mathematics of Operations Research 16*(2), 272–291.

Vanderbei, R. (1997a). Java applet for the teaching of shortest paths in a network. Personal web page, (Princeton University).

Vanderbei, R. (1997b). Java applet for the teaching of the network simplex method. Personal web page, (Princeton University).

Zadeh, N. (1973a). A bad network problem for the simplex method and other minimum cost flow algorithms. *Mathematical Programming 5*(1), 255–266.

Zadeh, N. (1973b). More pathological examples for network flow problems. *Mathematical Programming 5*(1), 217–224.

Zörnig, P. (2006). Systematic construction of examples for cycling in the simplex method. *Computers & Operations Research 33*(8), 2247–2262.