Χαροκόπειο Πανεπιστήμιο
Τμήμα Πληροφορικής και Τηλεματικής

Διπλωματική εργασία

«Υλοποίηση αλγορίθμων βελτιστοποίησης ευσταθούς κατανομής»

Ευαγγελία Γιαννούση, itp13103

Επιβλέπων καθηγητής: Δρ. Παύλος Ειρηνάκης

Ευχαριστίες

Αρχικά θα ήθελα να εκφράσω τις ιδιαίτερες ευχαριστίες μου στον επιβλέποντα καθηγητή της διπλωματικής μου εργασίας, Παύλο Ειρηνάκη, για την συνεχή καθοδήγηση, τις πολύτιμες συμβουλές και την υπομονή που έδειξε στο πρόσωπο μου, κατά τη διάρκεια εκπόνησης της διπλωματικής μου εργασίας.

Επίσης, θα ήθελα να ευχαριστήσω την οικογένεια μου, για την κατανόηση, την υπομονή και την ψυχολογική υποστήριξη κατά τη διάρκεια εκπόνησης της διπλωματικής μου εργασίας αλλά και την συνολική συμπαράσταση και υποστήριξη κατά τη διάρκεια των σπουδών μου.

Τέλος, δεν θα μπορούσα να παραλείψω να ευχαριστήσω τον σύντροφό μου Γεώργιο Κλάδη, για την στήριξη και την ψυχολογική υποστήριξη κατά τη διάρκεια εκπόνησης της διπλωματικής μου εργασίας.

Περιεχόμενα

Λίστα σχημάτων

Λίστα πινάκων

Αλγόριθμοι

Περίληψη

Η διατριβή αυτή πραγματεύεται το πρόβλημα ακεραίας ευσταθούς κατανομής (Integer Stable Allocation), εξετάζοντας την εφαρμογή του στον τομέα της μεταφοράς φορτίων από ένα στόλο φορτηγών και οχημάτων. Πιο συγκεκριμένα, εξετάζουμε την περίπτωση όπου έχουμε έναν αριθμό παλετών, που ανήκουν σε μία ή περισσότερες εταιρίες, οι οποίες πρέπει να μεταφερθούν στους τελικούς τους προορισμούς. Η μεταφορά αυτή γίνεται από έναν αριθμό φορτηγών που έχουν στη διάθεσή τους οι εταιρίες, τα οποία μπορεί να είναι ιδιόκτητα ή να ανήκουν σε παρόχους μεταφορικών υπηρεσιών (πχ. Μεταφορικές εταιρίες ή εταιρίες 3rd Party Logistics - 3PL). Κάθε ένα από τα φορτηγά χωράει ή έχει υπόλοιπο συγκεκριμένο αριθμό παλετών. Οι εταιρίες έχουν συγκεκριμένες προτιμήσεις για τα φορτία τους στα διαθέσιμα φορτηγά και τα φορτηγά έχουν προτιμήσεις στα φορτία. Οι προτιμήσεις αυτές εκφράζονται με τη χρήση λίστας προτίμησης. Το πρόβλημα ακεραίας ευσταθούς κατανομής ζητά την εύρεση μιας κατανομής των φορτίων στα φορτηγά η οποία να είναι ευσταθής, δηλαδή να μην υπάρχει κάποιο φορτίο και κάποιο φορτηγό τα οποία έχουν αντιστοιχηθεί με άλλους συμμετέχοντες (φορτηγό και φορτία αντίστοιχα) από την εν λόγω κατανομή αλλά έχουν κοινό συμφέρον να αντιστοιχηθούν μεταξύ τους.

Στα πλαίσια αυτά παρουσιάζεται το πρόβλημα της ευσταθούς κατανομής καθώς και της ευσταθούς ροής και μελετούνται διαφορετικές προσεγγίσεις επίλυσής τους. Επίσης, διερευνούνται και πιο ειδικές περιπτώσεις, και πιο συγκεκριμένα διαφορετικές παραλλαγές του προβλήματος ευσταθούς ταιριάσματος (Stable Matching). Βασιζόμενοι στις σχετικές αλγοριθμικές προσεγγίσεις που υπάρχουν στη βιβλιογραφία, μοντελοποιούμε το πρόβλημα μεταφοράς φορτίων ως πρόβλημα ακεραίας ευσταθούς κατανομής και υλοποιούμε έναν αλγόριθμο βελτιστοποίησης, ο οποίος γενικεύει τον αλγόριθμο των Gale-Shapley για το πρόβλημα ευσταθούς ταιριάσματος. Στη συνέχεια υλοποιούμε έναν μηχανισμό γέννησης σχετικών προβλημάτων (problem generator), και ελέγχουμε τη λειτουργία και την αποτελεσματικότητα του υλοποιημένου αλγορίθμου.

Κεφάλαιο 1: Εισαγωγή

Καθημερινά, οι άνθρωποι έρχονται αντιμέτωποι με προβλήματα αντιστοίχισης που περιλαμβάνουν την έννοια των προτιμήσεων. Για παράδειγμα, κάθε χρόνο, δεκάδες χιλιάδες μαθητές γράφουν εξετάσεις για την εισαγωγή τους στο πανεπιστήμιο και δίνοντας μια λίστα προτιμήσεων για τις σχολές στις οποίες θέλουν να εισαχθούν, περιμένουν να αντιστοιχηθούν σε μία από αυτές, σύμφωνα με τη βαθμολογία την οποία συγκέντρωσαν, καθώς και με τις απαιτήσεις της κάθε σχολής. Αντίστοιχα, η κάθε σχολή έχει προτιμήσεις στους αιτούντες φοιτητές, ανάλογα με τη βαθμολογία τους. Το τελικό αποτέλεσμα είναι μια αντιστοίχιση των υποψηφίων φοιτητών στις σχολές που λαμβάνει υπόψη τις προτιμήσεις και των δύο πλευρών. Παρόμοια, όταν ένας δημόσιος υπάλληλος ζητάει να πάρει μετάθεση, δηλώνει με σειρά προτεραιότητας τις περιοχές και τις θέσεις στις οποίες θα επιθυμούσε να μετατεθεί, σε μια λίστα προτίμησης. Από την άλλη, ελέγχονται οι απαιτήσεις της περιοχής, καθώς και τα μόρια τα οποία διαθέτει ο εν λόγω υπάλληλος, δημιουργώντας για κάθε θέση μία σειρά προτίμησης στους υποψηφίους για την κατάληψή της. Στη συνέχεια, εφόσον γίνει ο κατάλληλος έλεγχος, αντιστοιχείται ο υπάλληλος με κάποια από τις θέσεις που έχει στη λίστα του.

Ο καθένας λοιπόν, που συμμετέχει σε μια τέτοιου είδους διαδικασία έχοντας μια λίστα προτιμήσεων, επιθυμεί να αντιστοιχηθεί με την βέλτιστη για αυτόν επιλογή. Επιστήμονες από διάφορους κλάδους έχουν ασχοληθεί με αυτού του είδους τα προβλήματα τις τελευταίες δεκαετίες προσφέροντας μια σειρά από διαφορετικές προσεγγίσεις ανάλογα με τον τύπο του προβλήματος που παρουσιάζεται.

Στη πιο γενική τους μορφή, τέτοιου είδους προβλήματα ανάγονται στο πεδίο προβλημάτων της ευσταθούς κατανομής (Stable Allocation), μια παραλλαγή των προβλημάτων αντιστοίχισης παρουσία προτιμήσεων. Μια εξειδικευμένη μορφή της ευσταθούς κατανομής είναι το πρόβλημα του ευσταθούς ταιριάσματος (Stable Matching) ή ευσταθούς γάμου (Stable Marriage), όπως επίσης αποκαλείται. Το πρόβλημα αυτό ζητά την εύρεση ενός ευσταθούς ταιριάσματος ανάμεσα σε δύο διακριτά σύνολα στοιχείων, για κάθε ένα από τα οποία δίνεται μια διάταξη των προτιμήσεων του για τα στοιχεία του απέναντι συνόλου. Ένα ταίριασμα είναι μια ένα-προς-ένα αντιστοίχιση των στοιχείων ενός συνόλου με τα στοιχεία του άλλου. Ένα ταίριασμα είναι ευσταθές όταν δεν υπάρχει κανένα ζευγάρι στοιχείων που ενώ

δεν είναι αντιστοιχισμένα, προτιμούν το ένα το άλλο περισσότερο από τα υπάρχοντά τους ταίρια.

Το πρόβλημα ευσταθούς γάμου είναι ένα ένα-προς-ένα (one-to-one) πρόβλημα [1], [2], όπου ο κάθε άντρας μπορεί να παντρευτεί μια γυναίκα και το αντίθετο. Πιο συγκεκριμένα, στο άρθρο [1], οι Gale και Shapley εισάγουν την έννοια του ευσταθούς ταιριάσματος και το επιλύνουν, καθώς και το πρόβλημα επιλογής φοιτητών από τα πανεπιστήμια. Στο άρθρο [2], οι Irving κ.α. επιλύνουν το πρόβλημα του ευσταθούς ταιριάσματος και βρίσκουν ένα βέλτιστο ταίριασμα, χρησιμοποιώντας ισότιμα κριτήρια και για τις δυο πλευρές των συμμετεχόντων. Γενικότερα, η επιστημονική κοινότητα ασχολήθηκε αρκετά με παρόμοια προβλήματα με το ευσταθές ταίριασμα, καθώς και παραλλαγές του συγκεκριμένου, ώστε να μπορεί να λύσει και πιο πολύπλοκα προβλήματα, όπως στο άρθρο [3], όπου οι Bansal κ.α., προτείνουν έναν αλγόριθμο για βέλτιστο ευσταθές ταίριασμα, όπου ο κάθε άντρας και η κάθε γυναίκα, μπορούν να έχουν πολλαπλούς συντρόφους. Στο παράδειγμα του Σχήματος 1.1, παρουσιάζεται ένα πρόβλημα ευσταθούς γάμου, δηλαδή μία ένα-προς-ένα αντιστοίχιση με προτιμήσεις, όπου ο κάθε άντρας μπορεί να είναι μόνο με μια γυναίκα και αντίστροφα. Από την άλλη, στην περίπτωση του προβλήματος επιλογής πανεπιστημίου, ο κάθε φοιτητής θέλει να αντιστοιχηθεί μόνο με ένα πανεπιστήμιο, αλλά το κάθε πανεπιστήμιο, μπορεί να αντιστοιχηθεί σε περισσότερους από έναν φοιτητές (one-to-many). Τέλος, όταν αναφερόμαστε σε περιπτώσεις όπως π.χ. επιχειρήσεις με εξωτερικούς συνεργάτες [4], τότε η κάθε επιχείρηση μπορεί να αντιστοιχηθεί με πάνω από έναν συνεργάτες και ο κάθε ένας εξωτερικός συνεργάτης μπορεί να αντιστοιχηθεί σε πάνω από μια επιχειρήσεις (many-to-many) [3], [5]. Σε μια από τις πλέον νέες εκδοχές του προβλήματος, το άρθρο [6] γενικεύει τα προβλήματα ευσταθούς ταιριάσματος σε κατανομές πραγματικών χρόνων και ποσοτήτων, δηλαδή στο πρόβλημα ευσταθούς κατανομής.

	Α	Β	Γ	Δ
α	1,3	2,3	3,2	4,3
β	1,4	4,1	3,3	2,2
γ	2,2	1,4	3,4	4,1
δ	4,1	2,2	3,1	1,4

Σχήμα 1.1: Παράδειγμα ευσταθούς ταιριάσματος (stable matching), όπου α, β, γ, δ είναι οι άνδρες και Α, Β, Γ, Δ οι γυναίκες

1.1. Προβλήματα ευσταθούς ταιριάσματος

Οι πρώτοι που μελέτησαν αυτού του είδους τα προβλήματα είναι οι Gale και Shapley το 1962 [1]. Από τότε, τα προβλήματα ευσταθούς ταιριάσματος μελετήθηκαν εκτενώς, για παράδειγμα στα άρθρα [2], [7]. Στο άρθρο [7], οι Balinski και Ratier αναπτύσσουν κάποια γνωστά και κάποια νέα αποτελέσματα στο ευσταθές ταίριασμα, χρησιμοποιώντας κατευθυνόμενους γράφους. Στην κλασσική εκδοχή του προβλήματος ευσταθούς ταιριάσματος, έχουμε ίδιο πλήθος ανδρών και γυναικών. Έστω n άνδρες και n γυναίκες, κάθε ένας από τους οποίους, έχει και μια λίστα προτιμήσεων ως προς το αντίθετο φύλο. Ένα ταίριασμα M ανδρών και γυναικών, είναι ευσταθές, αν δεν υπάρχει κανένα ζευγάρι (i, j) που να εμποδίζει την διαδικασία (blocking pair), δηλαδή κανένα ζευγάρι που η γυναίκα i και ο άνδρας j να μην είναι ταιριασμένοι αλλά να ήταν και οι δύο πιο ευτυχισμένοι αν ήταν μαζί, παρά με το τωρινό τους ζευγάρι.

Η πλέον κλασική αλγοριθμική προσέγγιση στο πρόβλημα δόθηκε από τους Gale και Shapley [1] (ο αλγόριθμός τους είναι γνωστός ως Gale-Shapley (GS) αλγόριθμος). Στο Σχήμα 1.1, βλέπουμε έναν πίνακα κατάταξης, όπου α, β, γ, δ είναι οι άνδρες και Α, Β, Γ, Δ οι γυναίκες. Ο πρώτος αριθμός σε κάθε ζεύγος, είναι η κατάταξη της γυναίκας ως προς τις προτιμήσεις του άντρα, ενώ ο δεύτερος αριθμός είναι η κατάταξη του άντρα ως προς τις προτιμήσεις της γυναίκας. Παρατηρούμε ότι, ο άντρας α, προτιμά πρώτα την γυναίκα Α, μετά την Β, μετά την Γ και τέλος την Δ. Από την άλλη η γυναίκα Α προτιμά πρώτα τον άντρα δ. Η λογική που χρησιμοποιούν οι Gale και Shapley για να καταλήξουν στο ευσταθές ταίριασμα είναι η χρήση ενός αλγορίθμου “πρότασης και απόρριψης” (propose and reject).

Πιο συγκεκριμένα, όπως βλέπουμε στο Αλγόριθμος 1, ο κάθε άντρας, προτείνει πρώτα στην γυναίκα που είναι η πρώτη του επιλογή. Αν η γυναίκα αυτή είναι ελεύθερη, τον αποδέχεται προσωρινά. Αν είναι δεσμευμένη, τον συγκρίνει με τον άντρα με τον οποίο είναι προσωρινά ταιριασμένη και απορρίπτει τον λιγότερο προτιμητέο από τους δύο. Αφού λοιπόν, προτείνουν όλοι οι άνδρες, με σειρά προτίμησης στις γυναίκες, υπάρχουν κάποιοι οι οποίοι έχουν απορριφθεί. Οπότε προτείνουν εκ νέου στις επόμενες γυναίκες στη λίστα τους και αυτό επαναλαμβάνεται, ώσπου όλοι να έχουν από ένα ταίρι. Μάλιστα, η σειρά με την οποία προτείνουν οι άνδρες μπορεί να αλλάξει χωρίς να αλλάξει το τελικό αποτέλεσμα.

Ψευδοκώδικας GS

1) do

2) {

3) for i 0 to men

4) for j 0 to women

5) Ορίζεται η τιμή pl (δηλαδή η γυναίκα που είναι στη θέση j στην preference list του άντρα i)

6) Ο άντρας m[i] προτείνει στη γυναίκα w[pl]

7) if (η w[pl] είναι ελεύθερη)

8) Η γυναίκα w[pl] αποδέχεται προσωρινά τον m[i]

9) Αποθηκεύει αυτόν που αποδέχεται προσωρινά ως temp

10) else if (η w[pl] είναι δεσμευμένη)

11) Συγκρίνει τον m[i] με τον temp

12) if (η w[pl] προτιμάει τον m[i])

13) Αποθηκεύει τον m[i] προσωρινά ως temp

14) else Απορρίπτει τον m[i] και παραμένει δεσμευμένη με τον temp

15) Αποθηκεύεται στον πίνακα table το ζευγάρι (αν υπάρχει κάποιο νέο) που προκύπτει στο τέλος της επανάληψης

16) }while(όλοι οι άντρες να αποκτήσουν ταίρι ή όσοι είναι ελεύθεροι να μην έχουν άλλη επιθυμητή γυναίκα στη λίστα τους (να έχουν προτείνει σε όλες τις γυναίκες στη λίστα τους και να έχουν απορριφθεί από όλες).)

17) Αποτελέσματα table (πίνακας αντιστοίχισης αντρών i και γυναικών j)

Αλγόριθμος 1:Ο ψευδοκώδικας του GS

Το πρόβλημα του Σχήματος 1.1, καταλήγει σε ένα ευσταθές ταίριασμα που φαίνεται από τα κυκλωμένα ζευγάρια, όπου μπορούμε να παρατηρήσουμε ότι κανένας από τους άνδρες και τις γυναίκες δεν είναι με την πρώτη του επιλογή. Είναι όμως ευσταθές, εφόσον αυτός που προτιμά περισσότερο κάποια γυναίκα δεν την προτιμά περισσότερο από το υπάρχον ταίρι του (μιας και δεν της έκανε πρόταση) και αυτή που προτιμά περισσότερο κάποιος άντρας τον έχει ήδη απορρίψει (μιας και της πρότεινε κάποιος πιο προτιμητέος άντρας).

Η παραπάνω διαδικασία αναβαλλόμενης αποδοχής (deferred acceptance) αποτελεί τη βασική μέθοδο επίλυσης προβλημάτων ευσταθούς ταιριάσματος, και ονομάζεται, όπως αναφέρθηκε και παραπάνω, αλγόριθμος Gale-Shapley (GS). Το αποτέλεσμα του ταιριάσματος είναι ευσταθές, και μάλιστα είναι το βέλτιστο ταίριασμα για τους άνδρες και το χείριστο για τις γυναίκες. Αντίθετα, το ταίριασμα θα ήταν βέλτιστο για τις γυναίκες, και χείριστο για τους άνδρες αν πρότειναν αυτές και απαντούσαν οι άνδρες. Πέραν του κλασσικού αλγορίθμου GS, αναπτύχθηκε και μια διευρυμένη εκδοχή αυτού, ο οποίος είναι γνωστός ως ο Extended Gale-Shapley

(EGS) [8] αλγόριθμος, όπως φαίνεται στο Αλγόριθμος 2, βάση του οποίου μπορούν να αναπτυχθούν νέοι αλγόριθμοι.

Ψευδοκώδικας EGS
1) do
2) {
3) for i 0 to men
4) for j 0 to women
5) Ορίζεται η τιμή pl (δηλαδή η γυναίκα που είναι στη θέση j στην preference list του άντρα i)
6) Ο άντρας m[i] προτείνει στη γυναίκα w[pl]
7) if (η w[pl] είναι ελεύθερη)
8) Η w[pl] αποδέχεται προσωρινά τον m[i]
9) Διαγράφει όποιον άντρα υπάρχει μετά από αυτόν στη λίστα προτίμησής της.
10) Αποθηκεύει αυτόν που αποδέχεται προσωρινά ως temp
11) else if (η w[pl] είναι δεσμευμένη)
12) Συγκρίνει τον m[i] με τον temp
13) if (η w[pl] προτιμάει τον m[i])
14) Αποθηκεύει τον m[i] προσωρινά ως temp
15) Διαγράφει όποιον άντρα υπάρχει μετά από αυτόν στη λίστα προτίμησής της.
16) else Απορρίπτει τον m[i] και παραμένει δεσμευμένη με τον temp
17) Αποθηκεύεται στον πίνακα table το ζευγάρι (αν υπάρχει κάποιο νέο) που προκύπτει στο τέλος της επανάληψης
18) }while(μέχρι όλοι οι άντρες να αποκτήσουν ταίρι ή όσοι είναι ελεύθεροι να μην έχουν άλλη επιθυμητή γυναίκα στη λίστα τους (η λίστα τους να είναι κενή))
19) Αποτελέσματα table (πίνακας αντιστοίχισης αντρών i και γυναικών j)

Αλγόριθμος 2:Ο ψευδοκώδικας του EGS

Μια εναλλακτική προσέγγιση στο πρόβλημα ευσταθούς ταιριάσματος δόθηκε από τους Balinski και Ratier [7] οι οποίοι χρησιμοποίησαν ένα γράφημα-γραμμής (line-graph), το οποίο ονόμασαν γράφημα γάμου (marriage graph), για να αναπαραστήσουν το πρόβλημα. Στο γράφημα γάμου οι κόμβοι αντιστοιχούν σε ζευγάρια αντρών-γυναικών, ενώ οι κατευθυνόμενοι κλάδοι δείχνουν τις προτιμήσεις των συμμετεχόντων. Οι Balinski και Ratier έδωσαν και έναν αλγόριθμο που βρίσκει τη βέλτιστη λύση για τους άντρες ή τις γυναίκες και στην ουσία αποτελεί επέκταση του αλγορίθμου GS στο γράφημα γάμου.

Θα πρέπει να καταστεί σαφές ότι τόσο ο αλγόριθμος GS όσο και ο EGS παράγουν μία "άδικη" λύση, υπό την έννοια ότι η λύση αυτή είναι βέλτιστη για την

μία πλευρά (πχ. για τους άνδρες) και χείριστη για την άλλη (πχ για τις γυναίκες). Λόγω του άδικου ταιριάσματος για την μια πλευρά που δίνει ο GS αλγόριθμος, στο άρθρο [2], αναπτύχθηκε ένας αλγόριθμος πολυωνυμικού χρόνου, για την εύρεση του βέλτιστου ευσταθούς ταιριάσματος, ο οποίος χρησιμοποιεί περιστροφές (rotations). Σε αυτή την περίπτωση, το κάθε ζευγάρι (άντρα, γυναίκα) έχει και ένα κόστος. Ο αλγόριθμος ταιριάζει τα ζευγάρια με τέτοιο τρόπο ώστε να δημιουργεί την αντιστοίχιση με το ελάχιστο συνολικό κόστος. Για να το πετύχει αυτό, ο αλγόριθμος μετατρέπει το πρόβλημα ευσταθούς ταιριάσματος σε ένα πρόβλημα εύρεσης μέγιστης ροής-ελάχιστης τομής σε ένα γράφημα που παράγεται από το γράφημα μερικής διάταξης των περιστροφών. Να σημειωθεί ότι ενώ ο αλγόριθμος λειτουργεί για οποιεσδήποτε τιμές κόστους, για την ειδική περίπτωση εύρεσης ενός δίκαιου (egalitarian) ευσταθούς ταιριάσματος, το κόστος αυτό είναι ίσο με τη θέση που έχει το κάθε ζευγάρι στη λίστα προτίμησης του κάθε συμμετέχοντα.

Γενικότερα, η επιστημονική κοινότητα ασχολήθηκε αρκετά σε παρόμοια προβλήματα με το ευσταθές ταίριασμα, καθώς και παραλλαγές του συγκεκριμένου, ώστε να μπορεί να λύσει και πιο πολύπλοκα προβλήματα. Μια χαρακτηριστική περίπτωση παραλλαγής είναι ο αριθμός των μελών της άλλης ομάδας με τα οποία μπορεί να αντιστοιχηθεί ένας συμμετέχοντας. Πιο συγκεκριμένα, όταν αναφερόμαστε στο πρόβλημα του ευσταθή γάμου, ο κάθε άντρας μπορεί να είναι μόνο με μια γυναίκα και αντίστροφα (one-to-one).

Σε άλλες περιπτώσεις όμως, όταν για παράδειγμα αναφερόμαστε στην επιλογή πανεπιστημίου, ο κάθε φοιτητής θέλει να αντιστοιχηθεί μόνο με ένα πανεπιστήμιο, αλλά από την άλλη το κάθε πανεπιστήμιο, μπορεί να αντιστοιχηθεί με περισσότερους από έναν φοιτητές (ένα-προς-πολλά). Τέλος, αναφερόμενοι σε επιχειρήσεις με εξωτερικούς συνεργάτες [4], βλέπουμε την πολλά-προς-πολλά εκδοχή του προβλήματος όπου η κάθε επιχείρηση μπορεί να αντιστοιχηθεί με πάνω από έναν συνεργάτες και ο κάθε ένας εξωτερικός συνεργάτης μπορεί να αντιστοιχηθεί με πάνω από μια επιχειρήσεις με τις οποίες μπορεί να συνεργαστεί [3], [5]. Στο άρθρο [3], οι Bansal κ.α., παρουσιάζουν το πρόβλημα του πολλά-προς-πολλά ευσταθούς ταιριάσματος, δίνουν την επέκταση του αλγορίθμου GS και εισάγουν την έννοια των περιστροφών στο πρόβλημα αυτό. Στο άρθρο [5], οι Eirinakis κ.α., προτείνουν έναν αλγόριθμο βέλτιστου χρόνου, ο οποίος λαμβάνοντας υπόψη τις ατομικές προτιμήσεις και τα κριτήρια μέγιστου και ελάχιστου, προσδιορίζει όλα τα ευσταθή ζεύγη "εργαζόμενος – εταιρία", δηλαδή όλα αυτά τα ζεύγη που συμμετέχουν σε

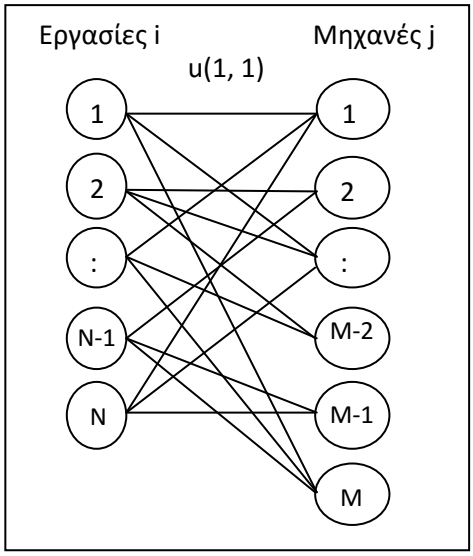
τουλάχιστον ένα ευσταθές ταίριασμα. Επίσης, δίνουν έναν αλγόριθμο για τη δημιουργία του γραφήματος μερικής διάταξης των περιστροφών και για την εύρεση του ευσταθούς ταιριάσματος ελαχίστου κόστους. Σε μια από τις πλέον νέες εκδοχές του προβλήματος, το άρθρο [6] γενικεύει τα προβλήματα ευσταθούς ταιριάσματος σε κατανομές πραγματικών χρόνων και ποσοτήτων, δηλαδή στο πρόβλημα ευσταθούς κατανομής. Αυτού του είδους τα προβλήματα αποτελούν το επίκεντρο της εργασίας αυτής και παρουσιάζονται αναλυτικότερα στο κεφάλαιο που ακολουθεί.

1.2. Πρόβλημα ευσταθούς κατανομής

Το πρόβλημα ευσταθούς κατανομής εισήχθη πρώτη φορά από τους Baïou και Balinski [6] το 2002. Στο παράδειγμα που εισήγαγαν, θεωρούν ότι υπάρχει μια αγορά από μηχανές και εργασίες, όπου κάθε εργασία μπορεί να εκτελεστεί εν μέρη ή συνολικά σε κάποια από τις μηχανές. Αναπαριστώντας το πρόβλημα ως διμερές γράφημα, όπως φαίνεται στο Σχήμα 1.2, με τις εργασίες από τη μία πλευρά και τις μηχανές από την άλλη, η αντιστοίχιση μιας εργασίας με μία μηχανή δίνεται μέσω της ακμής από τον κόμβο της εργασίας προς το κόμβο της μηχανής. Οι ακμές αυτές έχουν μια χωρητικότητα και οι συμμετέχοντες μπορούν να πάρουν ένα μέρος από τις ακμές με τις οποίες έχουν ταιριαστεί. Το μέρος αυτό, αντιπροσωπεύει τον χρόνο που χρειάζεται μια εργασία για να ολοκληρωθεί, καθώς και το πόσο μπορεί να δουλέψει η κάθε μηχανή στο σύνολό της. Οι ακμές αυτές, χρησιμοποιούνται ώστε να ανατεθούν οι εργασίες στις μηχανές, με τέτοιο τρόπο ώστε να μην ξοδεύει περισσότερο χρόνο σε μια δουλειά από αυτή που επιτρέπεται από την χωρητικότητα της ακμής. Επιπροσθέτως χρησιμοποιούνται εικονικές (dummy) μηχανές και εργασίες, οι οποίες όμως δεν μπορούν να αποτελούν μέρος της λύσης, απλά χρησιμοποιούνται στην επίλυση του προβλήματος. Επίσης δίνει συγκεκριμένη συνδεσμολογία μεταξύ μηχανών και εργασιών, χρησιμοποιεί λίστες προτιμήσεων όπως στο άρθρο [1] και παρέχει έναν αλγόριθμο, ο οποίος παράγει μία κατανομή που δεν περιέχει ζευγάρια εμπόδισης, και καταλήγει σε ευσταθή κατανομή.

Για παράδειγμα, έστω ότι έχουμε i = 1,…, N αριθμό εργασιών και j = 1, …, M αριθμό μηχανών, όπως φαίνεται στο Σχήμα 1.2. Οι εργασίες έχουν χρόνο επεξεργασίας p(i) και οι μηχανές χωρητικότητα c(j) και θέλουμε να υλοποιηθούν οι εργασίες με το μικρότερο δυνατό κόστος. Η κάθε εργασία μπορεί να υλοποιηθεί σε περισσότερες από μια μηχανές και η κάθε μηχανή μπορεί να εκτελέσει παραπάνω

από μια εργασίες (πολλά-προς-πολλά). Η κάθε ακμή, έχει ένα μέγιστο χρόνο u(i, j), ο οποίος είναι γνωστός και δεν μπορεί να ξεπεραστεί. Ο μέγιστος χρόνος αναφέρεται στο πόσο χρόνο μπορεί να διαθέσει η μηχανή j για να εκτελεστεί η εργασία i.



Σχήμα 1.2: Παράδειγμα αντιστοίχισης πολλών μηχανών με πολλές εργασίες (many-to-many)

1.3. Πρόβλημα ευσταθούς ροής

Το πρόβλημα της ευσταθούς κατανομής συνδέεται με αυτό της ευσταθούς ροής (stable flow) που παρουσιάστηκε πρώτα από τον Fleiner [9], ο οποίος έδωσε επίσης μια απόδειξη για την ύπαρξη μιας ευσταθής ροής σε κάθε δίκτυο λύνοντας ένα σταθερό πρόβλημα κατανομής σε ένα τροποποιημένο δίκτυο. Όπως βλέπουμε, κάθε πρόβλημα ευσταθούς ροής μπορεί να αναχθεί σε πρόβλημα ευσταθούς κατανομής και αντίστροφα.

Ένα δίκτυο ροής είναι ένα κατευθυνόμενο γράφημα G(V, E), κόμβων V ϵ R^n του οποίου οι ακμές E ϵ R^m έχουν κάποια συγκεκριμένη χωρητικότητα και δέχονται κάποια ροή, όπου n ο μέγιστος αριθμός κόμβων και m ο μέγιστος αριθμός ακμών. Η ποσότητα της ροής αυτής δεν μπορεί να ξεπεράσει την χωρητικότητα της ακμής. Για κάθε ροή που στέλνεται στο δίκτυο, πρέπει να ικανοποιείται ο περιορισμός ότι η εισερχόμενη ροή σε κάθε κόμβο του γραφήματος πρέπει να ισούται με την εξερχόμενη. Από τον περιορισμό αυτό εξαιρούνται μόνο οι κόμβοι-πηγές και οι τερματικοί κόμβοι. Το πρόβλημα της ευσταθούς ροής γενικεύει το πρόβλημα της ροής σε δίκτυα δίνοντας τη δυνατότητα ορισμού προτιμήσεων σε κάθε κόμβο του

γραφήματος G ως προς τις εισερχόμενες και τις εξερχόμενες ακμές του. Πέραν των περιορισμών διατήρησης της ροής, στο πρόβλημα ευσταθούς ροής ζητείται να ικανοποιούνται και περιορισμοί ευστάθειας που προκύπτουν όταν δεν υπάρχουν μονοπάτια εμπόδισης (blocking paths) [10]. Ένα μονοπάτι εμπόδισης με ροή f είναι ένα μονοπάτι του G στο οποίο αποστέλλεται ροή f με την παρακάτω ιδιότητα: ο αρχικός κόμβος του έχει τη δυνατότητα και προτιμά να αποστείλει μέρος της ροής f σε κάποιον άλλο κόμβο εκτός του μονοπατιού εμπόδισης καθώς επίσης ο τελικός κόμβος έχει τη δυνατότητα και προτιμά να παραλάβει μέρος της ροής f από κάποιο άλλο κόμβο εκτός του μονοπατιού εμπόδισης. Θέτοντας συγκεκριμένο κόστος σε κάθε ακμή, προκύπτει το πρόβλημα της βελτιστοποίησης της ευσταθούς ροής που μπορεί να αποσταλεί στο γράφο G. Το πρόβλημα βελτιστοποίησης ευσταθούς ροής βρίσκει εφαρμογή σε πολλούς τομείς, όπως η διαχείριση οδικού δικτύου, η αποστολή πληροφορίας σε τηλεπικοινωνιακά δίκτυα και άλλα.

1.4. Εφαρμογές σε πραγματικά προβλήματα

Μπορούμε να βρούμε αμέτρητα παραδείγματα ευσταθούς ταιριάσματος, όπου προκύπτουν από την αληθινή ζωή και σε όλες τις διαφοροποιήσεις τις οποίες αναφέραμε παραπάνω (ένα-προς-ένα, ένα-προς-πολλά, πολλά-προς-πολλά). Για τον λόγο αυτό, έχουν ασχοληθεί και έχουν συνεργαστεί επιστήμονες από διάφορους τομείς (όπως τα Οικονομικά, την Επιχειρησιακή Έρευνα, την Επιστήμη Υπολογιστών, την Θεωρία Παιγνίων κ.α.), ανάλογα με το πρόβλημα που αντιμετωπίζεται. Αξίζει να σημειωθεί πως η σημαντικότητα των προβλημάτων αυτών τόσο σε θεωρητικό όσο και σε πρακτικό επίπεδο αναγνωρίστηκε στα πρόσωπα των Roth και Shapley, οι οποίοι το 2012 βραβεύτηκαν για την προσφορά τους με το Βραβείο Νόμπελ στα Οικονομικά “για τη θεωρία των ευσταθών κατανομών και την πρακτική εφαρμογή του σχεδιασμού της αγοράς” [11].

Τη βάση για να δημιουργηθεί ένα πλαίσιο για μηχανισμούς ταιριάσματος την έβαλαν οι Gale και Shapley [1], οι οποίοι πρώτοι εισήγαγαν αυτή την έννοια στην επιστημονική κοινότητα, παρουσιάζοντας μια μελέτη για μια ορισμένη κατηγορία προβλημάτων κατανομής, ευσταθή ταιριάσματα. Σε αυτή, μελέτησαν τους μηχανισμούς με τους οποίους εισάγονται οι μαθητές σε πανεπιστήμια, λαμβάνοντας φυσικά υπόψη τους τις επιλογές και των δυο πλευρών, με την βοήθεια λιστών προτίμησης. Έχοντας σαν στόχο την επίτευξη της ευστάθειας, προσπαθούν να

ταιριάξουν τους μαθητές με τα πανεπιστήμια, με τέτοιο τρόπο ώστε κανένα ζευγάρι (μαθητής, πανεπιστήμιο) να μην ήταν πιο ικανοποιημένο αν είχε συνδυαστεί διαφορετικά.

Μια από τις πιο γνωστές εφαρμογές, είναι το Εθνικό Πρόγραμμα Ταιριάσματος των Ειδικευόμενων (National Resident Matching Program (NRMP)), το οποίο χρησιμοποιείται στις ΗΠΑ. Το πρόγραμμα αυτό, αντιστοιχεί απόφοιτους ιατρικών σχολών σε νοσοκομεία για να πάρουν ειδίκευση. Το περίεργο είναι ότι αυτός ο αλγόριθμος υπήρχε από το 1951. Δηλαδή, 11 χρόνια πριν την εισαγωγή του πλαισίου των Gale και Shapley, κάτι το οποίο πρώτος παρατήρησε ο Roth το 1984 [12], καθώς και πως χρησιμοποιούσαν ένα παρόμοιο αλγόριθμο αυτών.

Η εφαρμογή αυτή χρησιμοποιήθηκε αρκετό καιρό, αλλά παρά την επιτυχία που είχε, στην πορεία προέκυψαν προβλήματα ως προς την αποτελεσματικότητά της. Το κύριο πρόβλημα, ήταν ο συνεχώς αυξανόμενος αριθμός γιατρών, οι οποίοι είχαν παντρευτεί μεταξύ τους και ζητούσαν να πάρουν ειδικότητα στην ίδια περιοχή. Το NRMP, δεν μπορούσε να συμβιβάσει όλες αυτές τις αιτήσεις. Οπότε, πολλοί υποψήφιοι επέλεγαν να μην χρησιμοποιούν τον μηχανισμό αυτό. Το γεγονός αυτό, οδηγούσε ήδη σε ασταθή αποτελέσματα. Επίσης, πολλοί επέκριναν τον μηχανισμό NRMP, ότι ευνοούσε συστηματικά τα νοσοκομεία, έναντι των φοιτητών, κάτι το οποίο ήταν αληθές. Γιατί, όπως έχουμε αναφέρει, ο αλγόριθμος GS, ευνοεί πάντα την πλευρά που προτείνει, δηλαδή τα νοσοκομεία στην περίπτωση αυτή, και δίνει το βέλτιστο ευσταθές ταίριασμα αυτής. Το 1995, ζητήθηκε από τους Roth και Peranson να δημιουργήσουν έναν βελτιωμένο αλγόριθμο, ώστε να εξαλειφθούν τα προβλήματα που είχαν δημιουργηθεί. Έτσι, δημιούργησαν ένα νέο αλγόριθμο, ο οποίος είχε βασιστεί στις ανάγκες των υποψηφίων, καθώς και στην ανάγκη να διευκολύνονται τα παντρεμένα ζευγάρια. Ο αλγόριθμος αυτός χρησιμοποιήθηκε από το NRMP, από το 1997. Όμως, άρχισαν να προκύπτουν διάφορα προβλήματα μετά από λίγο διάστημα, καθώς οι αιτούντες, είχαν βρει τρόπο να χειρίζονται το σύστημα, βάση των προτιμήσεών τους. Βασιζόμενοι σε αυτό, ο αναθεωρημένος αλγόριθμος NRMP των Roth και Peranson [13] σχεδιάστηκε για να μην επηρεάζεται από ψευδείς δηλώσεις των φοιτητών.

Άλλη μια σημαντική εφαρμογή αφορά μηχανισμούς που έχουν σχεδιαστεί για την αντιστοίχιση μαθητών σε σχολεία. Τέτοιοι μηχανισμοί έχουν σχεδιαστεί και χρησιμοποιηθεί για την πόλη της Νέας Υόρκης και της Βοστώνης [14], [15]. Σε αυτές τις περιπτώσεις, ο αυξανόμενος αριθμός του πληθυσμού των πόλεων αυτών

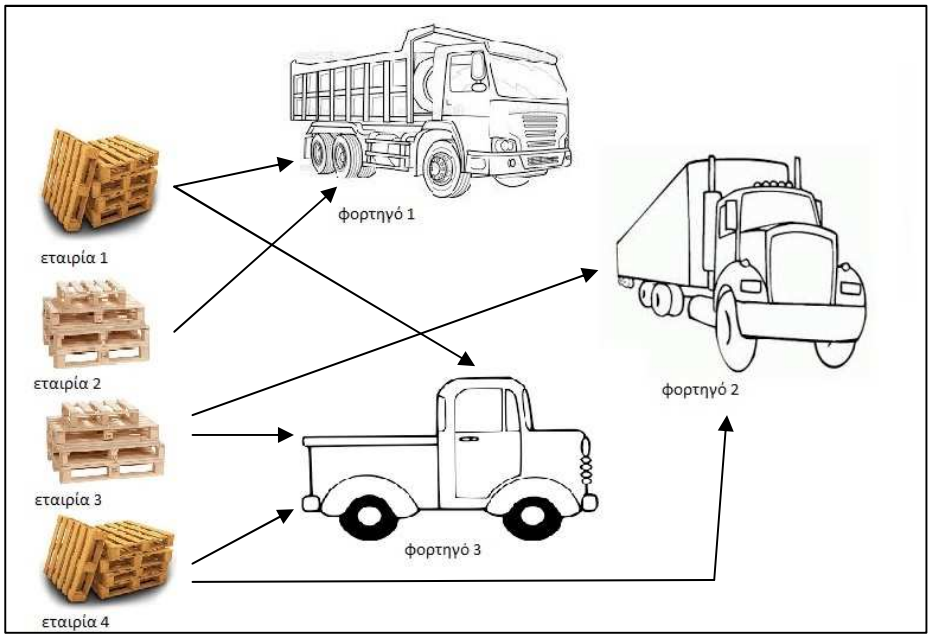
δημιουργεί προβλήματα στην κατανομή των μαθητών στα σχολεία. Για να αντιμετωπιστούν αυτά τα προβλήματα, υιοθετείται μία παραλλαγή του αλγορίθμου των Gale και Shapley. Σημαντική εφαρμογή επίσης, έχει γίνει για την αντιστοίχιση των νεφρών και άλλων ανθρώπινων οργάνων σε ασθενείς που έχουν ανάγκη από ένα μόσχευμα. Σε αυτή την περίπτωση βέβαια, μόνο η μια πλευρά των συμμετεχόντων έχει λίστα προτιμήσεων, ενώ η άλλη πλευρά είναι εντελώς παθητική. Ο προτεινόμενος αλγόριθμος Top Trading Cycle, είναι ουσιαστικά πολύ απλός και βασίζεται σε μια αρχική κατανομή και στις μετέπειτα εναλλαγές. Ένας βασικός περιορισμός βέβαια σε αυτή την περίπτωση, είναι ότι ορισμένα ζευγάρια ασθενή-οργάνου, μπορεί να μην είναι συμβατά, κάτι το οποίο μπορεί να δημιουργήσει πολλές χρονοβόρες διαδικασίες για την αλλαγή των ζευγών.

Το πλαίσιο της ευσταθούς κατανομής, μπορεί να χρησιμοποιηθεί στη ρύθμιση αγορών μεταφοράς [6], επεκτείνοντας το κλασσικό πρόβλημα της μεταφοράς. Επιτρέπει να καθορίζονται οι όροι της κατανομής από την ποιότητα και την ευστάθεια και όχι από το απόλυτο αριθμητικό κόστος του αποτελέσματος. Σε αυτή την περίπτωση υπάρχουν δύο διακριτά σύνολα παραγόντων, όπου ο κάθε ένας που συμμετέχει στο ένα σύνολο, έχει μια λίστα προτιμήσεων από το άλλο σύνολο. Το πρόβλημα αυτό αναζητεί ένα σχέδιο μεταφοράς, το οποίο να παρέχει και ευστάθεια. Δηλαδή, όπως έχουμε αναφέρει παραπάνω, όταν δεν υπάρχει κανένα ζευγάρι στοιχείων που ενώ δεν είναι αντιστοιχισμένα, προτιμούν το ένα το άλλο περισσότερο από τα υπάρχοντα ταίρια.

1.5. Εφαρμογή στο χώρο των μεταφορών

Τα προβλήματα που αναφέρθηκαν στα προηγούμενα κεφάλαια μοιάζουν πολύ με το πρόβλημα που πραγματεύεται η παρούσα εργασία. Πιο συγκεκριμένα, θα προσπαθήσουμε να επιλύσουμε το πρόβλημα ενός συνόλου εταιριών που θέλουν να μεταφέρουν συγκεκριμένο αριθμό παλετών με την βοήθεια φορτηγών. Πιο συγκεκριμένα, αναφερόμενοι στο Σχήμα 1.3, η αγορά την οποία θέλουμε να μοντελοποιήσουμε απαρτίζεται από πολλές εταιρίες, όπου η κάθε μια θα έχει έναν αριθμό παλετών που πρέπει να μεταφερθούν. Από την άλλη πλευρά, υπάρχει ένας αριθμός φορτηγών (που μπορεί να ανήκουν πχ. στις εταιρίες ή σε παρόχους υπηρεσιών μεταφοράς), κάθε ένα από τα οποία, μπορεί να χωρέσει συγκεκριμένο αριθμό παλετών. Οι εταιρίες έχουν συγκεκριμένες προτιμήσεις για τα φορτία τους

στα διαθέσιμα φορτηγά βασιζόμενες στα χαρακτηριστικά του καθενός (πχ. τιμή, ποιότητα υπηρεσιών, κλπ) και τα φορτηγά έχουν προτιμήσεις στα φορτία (πχ. με βάση τον προορισμό τους, το φορτίο, κλπ). Οι προτιμήσεις αυτές εκφράζονται με τη χρήση λιστών προτίμησης, οι οποίες μοντελοποιούν ένα διαισθητικό κριτήριο για τις προτιμήσεις του κάθε συμμετέχοντα. Το πρόβλημα λοιπόν που πρέπει να επιλυθεί, είναι πως θα μεταφερθούν οι παλέτες που έχουμε με ευσταθή τρόπο. Για την επίλυση ενός τέτοιου προβλήματος μπορεί να χρησιμοποιηθεί ο GS αλγόριθμος, για την περίπτωση που η κάθε εταιρία, μπορεί να συνεργαστεί μόνο με ένα φορτηγό και αντίστροφα. Παρόλα αυτά, όταν η κάθε εταιρία μπορεί να συνεργαστεί με περισσότερα από ένα φορτηγά και κάθε φορτηγό μπορεί να εξυπηρετήσει περισσότερες από μια εταιρίες, πρέπει να διαφοροποιήσουμε τον GS αλγόριθμο.



Σχήμα 1.3. Σχηματική αναπαράσταση της εφαρμογής μας

Στην παρούσα εργασία προσεγγίζουμε το πρόβλημα αλγοριθμικά με τη χρήση πινάκων και λιστών. Χρησιμοποιούμε τις λίστες έναντι των γράφων, καθώς στους γράφους πρέπει να δημιουργηθούν συσχετίσεις με πίνακες γειτνίασης, το οποίο έχει ως αποτέλεσμα, στα μεγάλα προβλήματα, να προκύπτουν τεράστιοι πίνακες, οι οποίοι θα πρέπει να ελεγχθούν ένα-ένα στοιχείο στις επαναλήψεις του προβλήματος, ώστε να εκτελεστεί ο αλγόριθμος ευσταθούς κατανομής. Σε αντίθετη περίπτωση, στις λίστες δηλώνουμε μόνο τις εταιρίες και τα φορτηγά που χρησιμοποιούνται για την επίλυση του προβλήματος, ενώ με τη χρήση μιας επέκτασης του αλγορίθμου GS

χρειάζεται μόνο να ελέγχουμε τα πρώτα στοιχεία της κάθε λίστας, δηλαδή αυτά στα οποία προτείνει ο κάθε συμμετέχοντας κάθε φορά.

1.6. Σύντομη περιγραφή των κεφαλαίων της πτυχιακής

Το κεφάλαιο 1 αποτελεί την εισαγωγή. Αναφέρονται έρευνες και τεχνικές οι οποίες χρησιμοποιήθηκαν για την ολοκλήρωση της παρούσας εργασίας.

Στο κεφάλαιο 2, αναφέρονται αναλυτικότερα η ευσταθή κατανομή και οι ευσταθείς ροές, όπως παρουσιάζεται σε ερευνητικές εργασίες, καθώς και πως ανάγεται το ευσταθές ταίριασμα στην ευσταθή ροή και αντίστροφα.

Στη συνέχεια, στο κεφάλαιο 3, παρουσιάζουμε το πρόβλημα το οποίο μας απασχόλησε στην παρούσα εργασία. Αναλύεται και αναφέρεται η δική μας λύση και ο αλγόριθμος που υλοποιήθηκε.

Στο κεφάλαιο 4 δίνονται τα αποτελέσματα της εκτέλεσης του αλγόριθμου που παρουσιάσαμε στο κεφάλαιο 3. Υπάρχει ανάλυση ως προς τον χρόνο τον οποίο χρειάζεται για να παραχθούν τα αποτελέσματα, καθώς και για την αποτελεσματικότητα της λύσης.

Στο κεφάλαιο 5 δίνουμε κάποια αποτελέσματα, που προκύπτουν όταν οι εταιρίες που λαμβάνουν μέρος σε ένα πρόβλημα, δεν προσπαθούν να εξυπηρετηθούν χρησιμοποιώντας οποιαδήποτε φορτηγά, αλλά μόνο κάποια από τα πρώτα στη λίστα προτίμησής τους.

Στο κεφάλαιο 6 βρίσκονται συμπεράσματα και συζήτηση σχετικά με τη προσέγγιση που προτείνεται. Επίσης θα αναφέρουμε μελλοντική εργασία η οποία μπορεί να γίνει, επεκτείνοντας την παρούσα εργασία.

Τέλος, στο Παράρτημα μπορούμε να δούμε τον αλγόριθμο που δημιουργήσαμε σε γλώσσα C#. Επίσης στην ίδια γλώσσα προγραμματισμού, μπορούμε να δούμε και τον μηχανισμό γέννησης προβλημάτων.

Κεφάλαιο 2: Ευσταθής κατανομή – ευσταθής ροή

Σε αυτό το κεφάλαιο, θα αναφερθούμε αναλυτικότερα στην ευσταθή κατανομή καθώς και στην ευσταθή ροή. Επίσης, θα αναφερθούμε στον τρόπο που ανάγεται ένα πρόβλημα ευσταθούς κατανομής σε ευσταθή ροή και αντίστροφα. Όπως αναφέραμε στο κεφάλαιο 1.2, στο πρόβλημα της ευσταθούς κατανομής, δημιουργείται το πρόβλημα των δίκαιων λύσεων για τους συμμετέχοντες. Αυτό μπορεί να αντιμετωπιστεί αποτελεσματικά με την εισαγωγή περιστροφών, όπου συμπεριλαμβάνεται σε αυτό το κεφάλαιο.

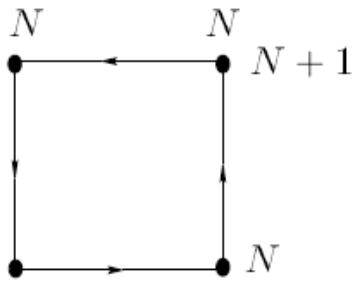
2.1. Ευσταθής κατανομή

Το πρόβλημα της ευσταθούς κατανομής, πρώτοι εισήγαγαν οι Baïou και Balinski [6] το 2002. Πιο πρόσφατη έρευνα για τέτοιου είδους προβλήματα έχουμε από τους Dean και Munshi [10]. Το πρόβλημα αυτό γενικεύει το κλασσικό πρόβλημα ευσταθούς ταιριάσματος συνδυάζοντας το με το πρόβλημα της μεταφοράς (transpotation).

Γενικά, ένα πρόβλημα ευσταθούς κατανομής (Γ, s, d, π), μπορεί να παρασταθεί από ένα κατευθυνόμενο γράφημα Γ με μορφή πλέγματος και μια ακολουθία πραγματικών αριθμών s, d > 0 και π ≥ 0. Υπάρχουν δύο διακριτά, πεπερασμένα σύνολα συντελεστών, έστω I={1, …, N} και J = {1, …,M}, και ο κάθε ένας συντελεστής έχει μια αυστηρή σειρά προτίμησης του άλλου συνόλου. Έστω ότι το σύνολο I, αποτελείται από εργαζόμενους και το σύνολο J από εργοδότες. Σε αυτή την περίπτωση ο κάθε εργαζόμενος $i \epsilon I$, έχει διαθέσιμες s(i) ώρες εργασίας τις οποίες μπορεί να προσφέρει. Από την άλλη, ο κάθε εργοδότης $j \epsilon J$, ζητάει d(j) ώρες εργασίας για να ολοκληρωθεί η δουλειά του. Επιπρόσθετα, υπάρχει η τιμή π(i, j), η οποία είναι ο μέγιστος χρόνος εργασίας που μπορεί να προσφέρει ο κάθε εργαζόμενος $i \epsilon I$ σε κάθε εργοδότη $j \epsilon J$. Στη συνέχεια τα δεδομένα αναπαριστάνονται σε ένα γράφο, όπως φαίνεται στο Σχήμα 2.1 με κόμβους και ακμές, ώστε να φαίνονται οι προτιμήσεις του καθένα.

Το πρόβλημα του ευσταθούς ταιριάσματος μπορεί να θεωρηθεί ως πρόβλημα ευσταθούς κατανομής με s(i) = p(j) =1 καθώς και π(i, j) = 1. Με τα ορίσματα αυτά, ο κάθε ένας από τους συμμετέχοντες, μπορεί να αντιστοιχηθεί αποκλειστικά με ένα και μόνο συμμετέχοντα από το απέναντι σύνολο, ικανοποιώντας τους περιορισμούς του

προβλήματος ένα-προς-ένα ευσταθούς ταιριάσματος. Αντίθετα όταν μιλάμε για χρόνο, όπως παραπάνω, δεν χρειάζεται να είναι ακέραιος ο αριθμός προσφοράς ή ζήτησης. Αναφερόμενοι λοιπόν στην ευσταθή κατανομή πολλών προς πολλά, τότε τα s(i) και d(j) θα είναι θετικοί ακέραιοι.



Σχήμα 2.1: Παράδειγμα ευσταθούς κατανομής 2X2

Όπως βλέπουμε και στο Σχήμα 2.1, δημιουργείται ένα πλέγμα, το οποίο έχει IxJ κόμβους. Οι κόμβοι οι οποίοι υπάρχουν σε αυτό το πλέγμα, αντιπροσωπεύουν τα ζευγάρια (i, j), όπου iϵI και jϵJ. Τέλος, παρατηρούμε ότι ανάμεσα στους κόμβους υπάρχουν κατευθυνόμενες ακμές. Ανάλογα με τις κατευθύνσεις των ακμών, φαίνονται οι προτιμήσεις που υπάρχουν. Η μια περίπτωση είναι όταν έχουμε οριζόντια ακμή. Δηλαδή όταν η ακμή κατευθύνεται από το ζευγάρι (i, j) προς το ζευγάρι (i, j΄). Αυτό σημαίνει ότι ο εργαζόμενος i, προτιμάει τον εργοδότη j΄ από τον εργοδότη j. Η άλλη περίπτωση είναι μια κατακόρυφη ακμή. Δηλαδή η ακμή που κατευθύνεται από το ζευγάρι (i, j) προς το ζευγάρι (i΄, j). Σε αυτή την περίπτωση, αντίστοιχα με προηγουμένως, ο εργοδότης j, προτιμάει τον εργαζόμενο i΄, σε σχέση με τον εργαζόμενο i. Να σημειωθεί ότι το γράφημα έχει την ιδιότητα της μεταβατικότητας, επομένως η ύπαρξη (οριζόντιου ή καθέτου) μονοπατιού από ένα κόμβο σε έναν άλλο υποδηλώνει και τις αντίστοιχες προτιμήσεις μεταξύ των κόμβων αυτών. Επίσης, αν το π(i, j) είναι 0, τότε ο σχετικός κόμβος παραλείπεται. Τέλος να αναφέρουμε ότι ένα ταίριασμα είναι ευσταθές όταν δεν υπάρχει κανένα ζευγάρι στοιχείων που ενώ δεν είναι αντιστοιχισμένα, προτιμούν το ένα το άλλο περισσότερο από τα υπάρχοντα ταίρια.

Οι Baïou και Balinski [6], έδειξαν ότι μια γενίκευση του Gale-Shapley αλγόριθμου, επιλύει το ακέραιο πρόβλημα ευσταθούς κατανομής ενός διμερούς γράφου, όπως ακριβώς είναι και το πρόβλημα που επιλύουμε στην παρούσα εργασία. Το πρόβλημα βέβαια σε αυτόν τον αλγόριθμο, είναι ότι ο αριθμός των κύκλων του

αλγορίθμου, θα πρέπει να είναι ίσος με το συνολικό άθροισμα των κόμβων, ακόμη και για έναν πολύ μικρό γράφο. Έτσι, ενώ ο γενικευμένος GS είναι πολύ γρήγορος στην πράξη, υπάρχουν ειδικές προβληματικές περιπτώσεις στις οποίες ο αλγόριθμος καθίσταται εκθετικός. Για να ξεπεράσουν το πρόβλημα αυτό, οι Baiou & Balinski πρότειναν ένα νέο αλγόριθμο, έναν επαγωγικό αλγόριθμο που λύνει το διμερές πρόβλημα ευσταθούς κατανομής σε πολυωνυμικό χρόνο. Τον αλγόριθμο αυτό στη συνέχεια βελτίωσαν οι Dean & Munshi.

Οι Dean και Munshi [10], στηριζόμενοι στον αλγόριθμο των Baïou και Balinski [6] δημιούργησαν έναν αλγόριθμο εύρεσης ευσταθούς κατανομής τον λεγόμενο αλγόριθμο DM. Ο αλγόριθμος ξεκινάει με μια υπόθεση, ότι όλες οι εργασίες εκτός από την εικονική εργασία είναι πλήρως εκχωρημένες (assigned) στην εικονική μηχανή και ότι όλες οι μηχανές, εκτός από την εικονική μηχανή, είναι πλήρως εκχωρημένες στην εικονική εργασία.

Για να αποθηκεύσει ο αλγόριθμος τα στοιχεία του γράφου, χρησιμοποιείται μια δομή δεδομένων δυναμικού δέντρου. Επίσης κρατείται μια λίστα L, η οποία κρατάει τα στοιχεία που δεν είναι πλήρως εκχωρημένα, ώστε σε κάθε επανάληψη, να επιλέγεται τυχαία ένα από τα στοιχεία της λίστας, για να βελτιώσει τη θέση του. Με κάθε στοιχείο που προσπαθεί να εκχωρηθεί πλήρως από τη λίστα L, αλλάζει η δομή του γράφου G(x), καθώς κάποιος κόμβος εξέρχεται από τον γράφο και εισέρχεται ένας νέος, ενώ ταυτόχρονα τροποποιείται η L. Όταν λέμε, ότι ένα στοιχείο είναι πλήρως εκχωρημένο, σημαίνει ότι το άθροισμα της χωρητικότητας των εξερχόμενων ακμών του, ισούται με τη χωρητικότητα του στοιχείου αυτού. Σε διαφορετική περίπτωση, το στοιχείο αυτό, βρίσκεται στη λίστα L. Ο αλγόριθμος τερματίζεται, όταν η λίστα L έχει αδειάσει. Δηλαδή, όταν όλα τα στοιχεία, είναι πλήρως εκχωρημένα.

Όταν αναφερόμαστε σε ένα στοιχείο που βελτιώνει τη θέση του, στην ουσία εννοούμε μια σειρά από ταυτόχρονες προτάσεις και απορρίψεις (“propose and reject”) ενός στοιχείου του δέντρου. Σε αυτό, η διαδικασία βελτίωσης ξεκινά από οποιαδήποτε εργασία i η οποία βρίσκεται στη λίστα L, οπότε δεν είναι πλήρως εκχωρημένη και ακολουθεί ένα μοναδικό μονοπάτι από το i ως την ρίζα. Κατά τη διάρκεια ενός κύκλου βελτιοποίησης, αυξάνεται η χωρητικότητα που δίνει μια εργασία i σε μια μηχανή j και μειώνεται αντίστοιχα η χωρητικότητα που δίνεται στην εικονική μηχανή, καθώς και η χωρητικότητα της αντίστοιχης μηχανής από την εικονική εργασία.

Κατά τον τερματισμό του αλγορίθμου, το αποτέλεσμα είναι η βέλτιστη για τις εργασίες ευσταθής κατανομή, όπως θα δούμε στο Σχήμα 2.4, όπου την ίδια στιγμή, είναι και η χείριστη, για τις μηχανές, ευσταθής κατανομή.

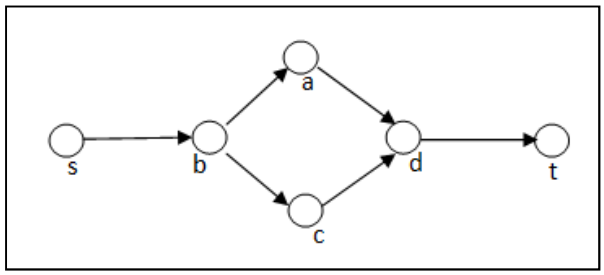
Οι Eirinakis κ.α. στο άρθρο [16], παρέχουν την πρώτη γραμμική αναπαράσταση δύο σημαντικών παραλλαγών της αντιστοίχισης δύο πλευρών υπό συνθήκες. Πιο συγκεκριμένα, παρέχεται ένα γραμμικό πρόβλημα για ακραίες ευσταθείς κατανομές και καθορίζεται η ελαχιστότητα της εν λόγω αναπαράστασης χρησιμοποιώντας αποτελέσματα γνωστών πολυεδρικών δομών για μερικώς διατεταγμένα σύνολα. Επίσης, παρέχεται ένα ακέραιο πρόβλημα για κατανομή κατοικίας (House Allocation) το οποίο έχει εκθετικά αυξανόμενο αριθμό περιορισμών, γι' αυτό και συνοδεύεται από ένα πολυωνυμικό αλγόριθμο διαχωρισμού.

2.2. Ευσταθής ροή

Λέγοντας δίκτυο ροής [9], θεωρούμε το σετ (D, s, t, c), όπου έχουμε τον γράφο D = (V, A), με V κόμβους και A ακμές. Τα s και t, είναι δύο διαφορετικοί κόμβοι του γράφου D. Τέλος το c: $A \rightarrow R_+$, είναι μια συνάρτηση, η οποία καθορίζει την χωρητικότητα c(a), της κάθε ακμής a του A. Η ροή του δικτύου (D, s, t, c), ορίζεται αρχικά από μια συνάρτηση f: $A \rightarrow R$. Η χωρητικότητα της f, για την κάθε ακμή a θα πρέπει να είναι τέτοια, ώστε να μην ξεπερνάει την τιμή της χωρητικότητας που επιτρέπεται να περνάει από μια ακμή a του A, καθώς και να μην είναι αρνητική ($0 \leq f(a) \leq c(a)$). Επίσης, όσον αφορά την ροή του δικτύου, ο κάθε κόμβος v του γράφου D, εκτός των κόμβων s και t, θα πρέπει να ικανοποιούν το νόμο του Kirchhoff:

$$\sum_{uv \in A} f(uv) = \sum_{vu \in A} f(vu) \quad (5)$$

Δηλαδή, αναφερόμενοι στην εξίσωση (5), η συνολική ποσότητα της ροής που εισέρχεται σε έναν κόμβο v, πρέπει να ισούται με την ποσότητα της ροής που εξέρχεται αυτής. Γενικά, οι κόμβοι s και t, δεν έχουν καμία διαφορά, είναι και οι δύο κοινοί κόμβοι, οι οποίοι απλά εξαιρούνται του νόμου του Kirchhoff. Αυτό το οποίο διαφέρει, είναι ο ρόλος που έχει αναλάβει καθένας από αυτούς τους δυο κόμβους μέσα στο δίκτυο. Ο κόμβος s έχει τον ρόλο του «source» και ο κόμβος t έχει τον ρόλο του «sink».



Σχήμα 2.2: Παράδειγμα δικτύου ευσταθούς ροής

Στο δίκτυο ροής, όπως το αναφέραμε παραπάνω, και φαίνεται στο Σχήμα 2.2, μπορεί να υπάρχει σειρά προτίμησης, που αναφέρεται σε κάθε κόμβο v ($\leq_v$), όπου το $\leq_v$, είναι μια γραμμική διάταξη των ακμών που καταλήγουν ή που φεύγουν από τον κόμβο v. Όμως, δεν χρειάζεται να συγκρίνεται μια εισερχόμενη με μια εξερχόμενη ακμή της ίδιας κορυφής. Έτσι μπορεί να υπάρχει μια σειρά προτιμήσεων για τις εισερχόμενες ακμές και μια άλλη σειρά προτιμήσεων για τις εξερχόμενες ακμές του v. Από την άλλη, η σειρά προτίμησης που αναφέρεται στους κόμβους s ($\leq_s$) και t ($\leq_t$), δεν παίζουν κανένα ρόλο στην ευστάθεια. Έστω ότι έχουμε ένα δίκτυο με προτιμήσεις, όπου οι κόμβοι του γράφου D είναι έμποροι, οι οποίοι συναλλάσσονται ένα συγκεκριμένο προϊόν. Σε αυτή την περίπτωση, κάθε ακμή uv, χαρακτηρίζεται από μια χωρητικότητα c(uv), η οποία δείχνει πόσες μονάδες του προϊόντος είναι το μέγιστο που μπορεί να δώσει ο έμπορος u στον έμπορος v. Από την άλλη το σύστημα, περιγράφεται από την ροή f του δικτύου, όπου f(uv), είναι οι οριστικές μονάδες προϊόντος που πουλάει τελικά ο έμπορος u στον έμπορο v. Γενικά, μέσα στο δίκτυο, όλοι προσπαθούν να μεγιστοποιήσουν τις μονάδες που ανταλλάσσουν. Οπότε, ο κάθε έμπορος, προσπαθεί να μεγιστοποιήσει τη ροή μέσω του κόμβου v. Αν όμως η f, επέτρεπε σε κάποιον κόμβο v να λάβει μεγαλύτερη ροή, τότε θα υπήρχε πρόβλημα στο σύστημά μας. Γιατί, αν ο κόμβος v, μπορούσε να λάβει μεγαλύτερη ροή, θα μπορούσε αντίστοιχα και να στείλει. Άρα, κάποιοι έμποροι, όσον αφορά το πρόβλημά μας, θα ήθελαν να στραφούν προς τον v, για να αγοράσουν περισσότερο. Οπότε όλη αυτή η διαδικασία, θα οδηγούσε σε μια μη ευσταθή κατάσταση του δικτύου ροής.

Γενικότερα, η παραπάνω περίπτωση, δεν είναι η μόνη, η οποία μπορεί να καθιστά το δίκτυο σε ασταθή κατάσταση. Αστάθεια, μπορεί να προκληθεί στην περίπτωση που ο έμπορος v, προτιμάει να πουλήσει στον έμπορο w, παρά στον έμπορο u ($vw \leq_v vu$) τη στιγμή που η ροή f, είναι τέτοια, ώστε ο έμπορος w, να προτιμά να αγοράσει περισσότερο προϊόν από τον έμπορο v. Σε αυτή την περίπτωση,

θα ισχύει f(vw) < c(vw), με τον w να έχει κι άλλες μονάδες προϊόντος προς πώληση. Επιπλέον, η ροή της ακμής vu, θα είναι θετική, δηλαδή ο v, πουλάει ένα ποσό προϊόντος στον u. Επομένως, ο v θα έστελνε ροή στον w αντί για τον u. Ένα ευσταθές σύστημα όμως, δεν επιτρέπει μια τέτοια κατάσταση.

Επομένως, η ροή ενός δικτύου με προτιμήσεις, μπορεί να χαρακτηριστεί ευσταθής, όταν δεν υπάρχουν μπλοκαρισμένα μονοπάτια στην f. Σε ένα πρόβλημα ευσταθούς ροής λοιπόν, δίνεται ένα δίκτυο με τις προτιμήσεις του και πρέπει να βρεθεί μια ευσταθής ροή, αν αυτή υπάρχει.

Στη συνέχεια, θα περιγραφεί συνοπτικά πως το πρόβλημα ευσταθούς κατανομής ανάγεται σε πρόβλημα ευσταθούς ροής.

2.3. Αναγωγή ευσταθούς κατανομής σε ευσταθή ροή και αντίστροφα

Σε ένα πρόβλημα ευσταθούς κατανομής [9], όπως αναφέραμε στο κεφάλαιο 2.1., υπάρχουν δύο διακριτά, πεπερασμένα σύνολα συντελεστών, έστω I και J, και ο κάθε ένας συντελεστής έχει μια αυστηρή σειρά προτίμησης του άλλου συνόλου. Έστω ότι το σύνολο I, αποτελείται από εργαζόμενους και το σύνολο J από εργοδότες. Επιπρόσθετα, ορίζεται από έναν χάρτη q: IUJ, ένα σετ ακμών E, μεταξύ των I και J, μαζί με έναν χάρτη p: E→R και για κάθε εργαζόμενο ή εργοδότη vϵ IUJ, μια σειρά $<_v$ για τις ακμές E που περιλαμβάνουν το v. Οι ακμές E, ουσιαστικά είναι τα ζεύγη (i, j), όπου iϵI και jϵJ. Το q(v) αναφέρεται στη μέγιστη τιμή ανάθεσης που μπορεί να προσφέρει ένας εργαζόμενος ή να δεχθεί μια εταιρία και η τιμή p(ij) της ακμής e=ij, αναφέρεται στο μέγιστο χρόνο εργασίας που μπορεί να προσφέρει ο κάθε εργαζόμενος iϵI σε κάθε εργοδότη jϵJ. Η κατανομή, είναι ένας θετικός χάρτης g: E→R, τέτοιος ώστε g(e) ≤ p(e), για κάθε eϵE και vϵ IUJ, όπου:

$$g(v) \coloneqq \sum_{ux \in E} g(vx) \leq q(v)\,, \quad (6)$$

είναι η συνολική τιμή ανάθεσης g(v) που μπορεί να δεχθεί ή να στείλει ο v, χωρίς να ξεπερνά την τιμή q(v). Αν η (6) ισχύει ως ισότητα, τότε μπορούμε να πούμε ότι ο v είναι κορεσμένος (g-saturated).

Μια τέτοια κατανομή είναι ευσταθής, αν ισχύει τουλάχιστον μια από τις παρακάτω ιδιότητες, για κάθε ακμή E:

- g(ij) = p(ij) (η συγκεκριμένη απασχόληση να πραγματοποιείται με πλήρη χωρητικότητα) (7)
- ο i εργαζόμενος να είναι κορεσμένος και να μην προτιμά τον j ως κάποιος από τους εργοδότες του (το ij κυριαρχεί (g-dominated) στο I) (8)
- η j εταιρία να είναι κορεσμένη και να μην προτιμά τον i ως κάποιον από τους υπαλλήλους της (το ij κυριαρχεί (g-dominated) στο J) (9)

Αν η g_1 και η g_2 είναι κατανομές και ο iϵI είναι ένας εργαζόμενος, τότε λέμε ότι η κατανομή g1 κυριαρχεί την κατανομή g2 για τον εργαζόμενο i ($g_1 \leq_i g_2$), αν ισχύει μια από τις παρακάτω ιδιότητες:

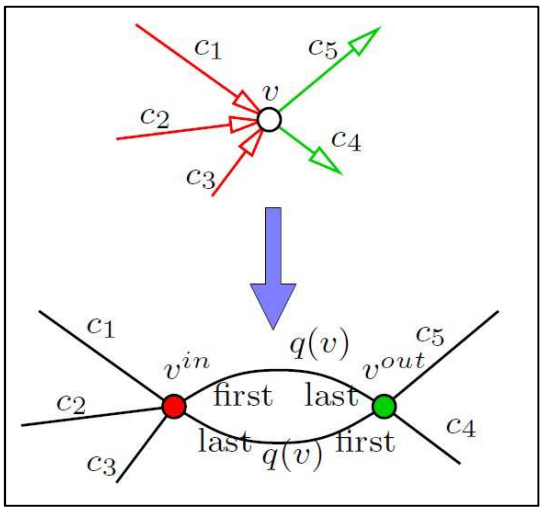
- $g_1(ij) = g_2(ij)$, για κάθε jϵJ (10)
- $\sum_{j' \in J} g_1(ij') = \sum_{j' \in J} g_2(ij') = q(i)$ και $g_1(ij) < g_2(ij)$ και $g_1(ij') > 0$, που συνεπάγεται ότι $ij' \leq_i ij$ (11)

Αυτό συμβαίνει, γιατί αν ο i μπορούσε να επιλέξει ελεύθερα από την $\max(g_1, g_2)$, τότε ο i θα επέλεγε την g_1, είτε γιατί οι g_1 και g_2 είναι πανομοιότυπες για τον i ή γιατί ο i είναι κορεσμένος και στις δυο κατανομές και το g_1 αντιπροσωπεύει την επιλογή του i από την $\max(g_1, g_2)$. Αλλάζοντας τους ρόλους των εργαζομένων και των εταιριών, μπορεί να οριστεί η κυρίαρχη σχέση $\leq_j$ και για κάθε εταιρία j.

Κάθε πρόβλημα ευσταθούς κατανομής μπορεί να μετατραπεί σε ένα δίκτυο ροής (D, s, t, c), τέτοιο ώστε: V(D) ={s, t}∪I∪J, A(D) = {si : i ϵ I}∪{jt : jϵ J}∪{ij : ij ϵ E} και c(si) = q(i), c(jt) = q(j) και c(ij) = p(ij) για κάθε εργαζόμενο i και για κάθε εταιρία j. Για αυτό, θεωρούμε το διμερές γράφημα, προσανατολίζοντας τις ακμές του από το I προς το J. Προσθέτουμε δύο νέους κόμβους s και t, με μια εξερχόμενη ακμή από τον κόμβο s προς κάποιον κόμβο-εργαζόμενο καθώς και μια εισερχόμενη ακμή στο t, από κάποιον κόμβο-εργοδότη, των οποίων η χωρητικότητα δίνεται από τη χωρητικότητα των αρχικών ακμών και των καινούριων ακμών που δημιουργήθηκαν. Η σειρά προτίμησης $\leq_v$ της ακμής που αναφέρεται στον κόμβο v, προκύπτει από τη σειρά προτίμησης των αντίστοιχων κόμβων v, ενώ στην περίπτωση που δεν υπάρχει, τότε είναι μια ασήμαντη γραμμική διάταξη.

Η αντίστροφη μετατροπή, δηλαδή από ένα πρόβλημα ευσταθούς ροής σε ένα πρόβλημα ευσταθούς κατανομής, είναι πιο περίπλοκη. Αρχικά, για την δημιουργία του γράφου G_D, όπως βλέπουμε στο Σχήμα 2.3, διασπάται ο κάθε κόμβος v του D σε

δύο διακριτές κορυφές v^{in} και v^{out} και για κάθε ακμή uv του D, προστίθεται μια ακμή $u^{out}v^{in}$ του G_D.



Σχήμα 2.3: Διάσπαση κόμβου για τη δημιουργία του G_D

Για κάθε μη τερματικό κόμβο v του D, προστίθενται δύο παράλληλες ακμές ανάμεσα στο v^{in} και το v^{out}. Για να τις ξεχωρίζουν μεταξύ τους, αναφέρονται ως $v^{in}v^{out}$ και $v^{out}v^{in}$. Έστω $p(v^{in}v^{out}) = (v^{in}v^{out})$ $p(v^{out}v^{in}) := q(v)$, $p(u^{out}v^{in}) := c(uv)$ και $q(v^{in}) = q(v^{out}) := q(v)$. Για να ολοκληρωθεί η κατασκευή του προβλήματος ευσταθούς κατανομής, πρέπει να καθοριστεί μια γραμμική επιθυμητή σειρά για κάθε κόμβο του GD. Για κάθε κόμβο v^{in} έστω $v^{in}v^{out}$ η επιλογή που προτιμάται και $v^{out}v^{in}$ η λιγότερο προτιμώμενη ακμή (αν οι ακμές συμπεριλαμβάνονται, δηλαδή το v είναι μη τερματικός κόμβος), και η σειρά των υπόλοιπων ακμών ακουμπάνε στο v^{in} και προέρχονται από τη σειρά του κόμβου v για τις αντίστοιχες ακμές. Για τον κόμβο v^{out} η πιο προτιμητέα ακμή είναι η $(v^{out}v^{in})$ και η λιγότερο προτιμώμενη η $(v^{out}v^{in})$ (σε περίπτωση που αυτές οι ακμές υπάρχουν) και οι άλλες προτιμήσεις προέρχονται από $<_v$.

Χρησιμοποιώντας τις κατασκευαστικές μεθόδους και τα γραφήματα που περιγράφονται παραπάνω, ο Fleiner [9] λοιπόν αποδεικνύει ότι το g είναι ευσταθής κατανομή αν και μόνο αν υπάρχει μια σταθερή ροή f,τέτοια ώστε g(e) = f(e) για κάθε ακμή, όπου το e είναι η ακμή που αντιστοιχεί στον κόμβο e. Το αποτέλεσμα αυτό δείχνει ουσιαστικά ότι οποιοδήποτε πρόβλημα ευσταθούς ροής μπορεί να μετατραπεί σε πρόβλημα ευσταθούς κατανομής (και το αντίθετο). Αυτό σημαίνει ότι μπορούμε να χρησιμοποιήσουμε τους αλγορίθμους που υπάρχουν για το πρόβλημα ευσταθούς

ροής για να αντιμετωπίσουμε προβλήματα που φαινομενικά παρουσιάζονται πιο περίπλοκα από αυτό της ευσταθούς κατανομής.

2.4. Περιστροφές

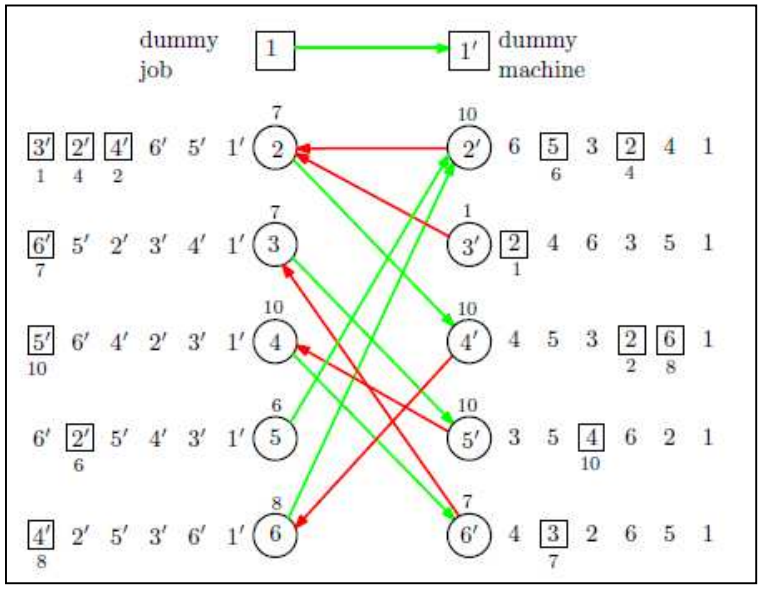
Ο αλγόριθμος των Dean και Munshi για το πρόβλημα της ευσταθούς κατανομής, όπως την είδαμε στο κεφάλαιο 2.1, μας δίνει την βέλτιστη λύση για την πλευρά που προτείνει και την χείριστη για την πλευρά που απορρίπτει. Με την βοήθεια των περιστροφών, μπορούμε να φτάσουμε σε μια πιο δίκαιη λύση και για τις δυο πλευρές των συμμετεχόντων.

Πιο συγκεκριμένα, κάθε πρόβλημα ευσταθούς κατανομής δεν έχει μόνο τις λύσεις που παράγει ο αλγόριθμος GS ή DM. Υπάρχουν και άλλες κατανομές, οι οποίες είναι ευσταθείς. Για το λόγο αυτό οι Dean και Munshi [10], χρησιμοποίησαν τις περιστροφές (rotation), ώστε να παρέχουν έναν αλγόριθμο που μπορεί να παράξει μία τέτοια λύση βασιζόμενος σε κόστη που μπορεί να ανατεθούν στα ζευγάρια εργασιών-μηχανών. Σε αυτή την περίπτωση, το κάθε ζευγάρι (εργασία, μηχανή) είχε και ένα κόστος το οποίο ο αλγόριθμος ελαχιστοποιεί, ταιριάζοντας τα ζευγάρια με το ελάχιστο συνολικό κόστος.

Οι περιστροφές γενικότερα, είναι γνωστές από το πρόβλημα του ευσταθούς ταιριάσματος [17]. Γενικεύονται σχετικά άμεσα και στην περίπτωση της ευσταθούς κατανομής, και ουσιαστικά, μας παρέχουν ένα μηχανισμό, για να μπορούμε να μετακινηθούμε από μια ευσταθή λύση σε μια άλλη. Οι περιστροφές μπορούν να χρησιμοποιηθούν για τη δημιουργία ενός αλγόριθμου εύρεσης της ευσταθούς κατανομής ελαχίστου κόστους. Για να το πετύχει αυτό, ο σχετικός αλγόριθμος μετατρέπει το πρόβλημα ευσταθούς κατανομής σε ένα πρόβλημα εύρεσης μέγιστης ροής-ελάχιστης τομής σε ένα παραγόμενο γράφημα. Ενώ ο αλγόριθμος λειτουργεί για οποιεσδήποτε τιμές κόστους, για την ειδική περίπτωση εύρεσης μιας δίκαιης ευσταθούς κατανομής, το κόστος αυτό είναι ίσο με τη θέση που έχει το κάθε ζευγάρι στη λίστα προτίμησης του κάθε συμμετέχοντα, όπως θα δούμε παρακάτω.

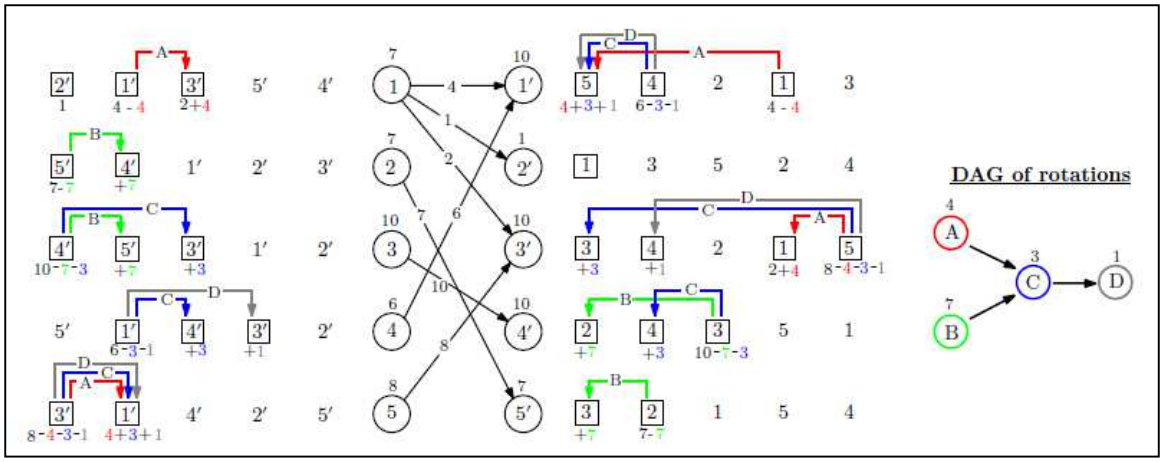
Σύμφωνα με τους Dean και Munshi [10], η περιστροφή ορίζεται ως κύκλος στον γράφο G(x) από τη στιγμή που έχει βρεθεί βέλτιστη λύση για την πλευρά που προτείνει και την χείριστη για την πλευρά που απορρίπτει και μετά. Στη συνέχεια λοιπόν ψάχνουμε για κύκλους στο G(x). Όταν τους βρούμε, μεταφέρουμε την μέγιστη ποσότητα που μπορούμε στον κάθε κύκλο. Αυτό αλλάζει το G(x) και συνεχίζουμε

μέχρι να μην υπάρχουν άλλοι κύκλοι. Τότε ξέρουμε ότι έχουμε βρει όλες τις περιστροφές. Κατά τη διάρκεια του αλγορίθμου αυτού, που αποκαλύπτει όλες τις περιστροφές, ο αλγόριθμος κρατά τις σχέσεις μεταξύ των περιστροφών. Αυτό επιτρέπει τη δημιουργία του γραφήματος διάταξης περιστροφών (rotation-poset graph). Αυτό το γράφημα έχει την ιδιότητα ότι τα κλειστά του υποσύνολα (υποσύνολα του γραφήματος με καμία εισερχόμενη ακμή) βρίσκονται σε μία ένα-προς-ένα σχέση με τις "ακέραιες" λύσεις του προβλήματος ευσταθούς κατανομής. Είναι αυτό το γράφημα που, με κατάλληλες μετατροπές, μπορεί να χρησιμοποιηθεί στη συνέχεια για την εύρεση της βέλτιστης λύσης, χρησιμοποιώντας έναν αλγόριθμο εύρεσης του μέγιστου κλειστού υποσυνόλου στο γράφημα διάταξης περιστροφών [18].



Σχήμα 2.4: Ο γράφος G(x), μόλις φτάνει ο αλγόριθμος DM στο βέλτιστο για τις εργασίες

Όπως αναφέραμε, στο σχήμα 2.4 [19], παρατηρούμε τον γράφο G(x), όπως αυτός αναφέρεται στον αλγόριθμο των Dean και Munshi στο κεφάλαιο 2.1, μόλις έχει φτάσει στη βέλτιστη για τις εργασίες κατανομή. Παρατηρούμε ότι χωρίζεται σε τρία ασύνδετα στοιχεία: ένα στοιχείο του δέντρου [1, 1’], ένα κυκλικό στοιχείο 1: [2, 2’, 3’, 4’, 5, 6] και ένα κυκλικό στοιχείο 2: [3, 4, 5’, 6’]. Καθένα από αυτά τα κυκλικά στοιχεία αντιστοιχεί σε μια περιστροφή. Επομένως, στο συγκεκριμένο παράδειγμα, ξεκινώντας από τη βέλτιστη για τις εργασίες λύση, έχουμε δύο περιστροφές, εφόσον έχουμε δύο ασύνδετα κυκλικά στοιχεία (κόμβοι A και B στο Σχήμα 2.5 – DAG of rotations).



Σχήμα 2.5: Ένα διμερές παράδειγμα και οι περιστροφές του

Αφού βρούμε μία περιστροφή, μεταφέρουμε την αντίστοιχη ποσότητα κατά μήκος του κύκλου (αποκαλύπτοντας την περιστροφή). Αυτό έχει σαν αποτέλεσμα την αλλαγή και του γραφήματος G(x). Η διαδικασία αυτή συνεχίζεται, αποκαλύπτοντας περιστροφές και ανανεώνοντας το γράφημα G(x) μέχρι να μην υπάρχουν πλέον άλλοι κύκλοι στο G(x). Τη στιγμή αυτή το γράφημα G(x) αναπαριστά την βέλτιστη λύση για τις μηχανές. Να σημειωθεί ότι κατά την εκτέλεση του αλγορίθμου αυτού, η εικονική μηχανή και η εικονική εργασία δεν χρησιμοποιούνται, όπως βλέπουμε και στο Σχήμα 2.5.

Στο Σχήμα 2.5, μπορούμε να δούμε, το διμερές παράδειγμα και δίπλα σε κάθε στοιχείο τη λίστα προτίμησής του, οι περιστροφές από A, B, C, D, καθώς και οι αντίστοιχοι πολλαπλασιαστές όπως φαίνονται στο DAG (Directed Acyclic Graph) στα δεξιά. Κάθε στοιχείο της λίστας προτίμησης που είναι μέσα σε πλαίσιο, παίρνει μέρος σε κάποια ευσταθή λύση. Κάτω από τις καταχωρήσεις των λιστών προτίμησης, φαίνεται πως η αρχική βέλτιστη για τις εργασίες καταχώρηση, αλλάζει εξαιτίας μιας περιστροφής. Αντίστοιχα, το ίδιο φαίνεται και με βέλη, πάνω από τις λίστες προτίμησης. Για παράδειγμα, όπως φαίνεται στο Σχήμα 2.5 η αρχική εκχώρηση της εργασίας 1 στέλνει 4 μονάδες στη μηχανή 1΄ και 2 μονάδες στη μηχανή 3΄. Η περιστροφή A, όταν ολοκληρωθεί, μειώνει κατά 4 μονάδες την εκχώρησης της εργασίας 1 στην μηχανή 1΄ και αυξάνει κατά 4 μονάδες την εκχώρηση της εργασίας 1 στη μηχανή 3΄. Το γράφημα διάταξης περιστροφών που δημιουργείται (DAG of rotations) αποτυπώνει τη μερική διάταξη μεταξύ των περιστροφών του προβλήματος. Με την κατάλληλη μορφοποίηση, το γράφημα αυτό μπορεί να χρησιμοποιηθεί για

την εύρεση του κλειστού του υποσυνόλου με μέγιστο κόστος και άρα και της αντίστοιχης βέλτιστης ευσταθούς κατανομής.

Είδαμε σε αυτό το κεφάλαιο, λίγο αναλυτικότερα για την ευσταθή κατανομή και την ευσταθή ροή. Επίσης αναφερθήκαμε στον τρόπο που ανάγεται ένα πρόβλημα ευσταθούς κατανομής σε ευσταθή ροή και αντίστροφα. Τέλος αναφερθήκαμε στο πως οι περιστροφές, βοηθούν στο πρόβλημα εύρεσης δίκαιων λύσεων. Στη συνέχεια, στο κεφάλαιο 3, θα δούμε τη δική μας λύση, στο πρόβλημα που αναφέραμε στο κεφάλαιο 1.

Κεφάλαιο 3: Το πρόβλημά μας

Όπως αναφέραμε στο κεφάλαιο 1.4, το πρόβλημα που θα προσπαθήσουμε να επιλύσουμε, είναι το πρόβλημα ενός συνόλου εταιριών που θέλει να μεταφέρει συγκεκριμένο αριθμό παλετών με την βοήθεια φορτηγών. Πιο συγκεκριμένα, θα απαρτίζεται από πολλές εταιρίες, όπου η κάθε μια θα έχει έναν αριθμό παλετών που πρέπει να μεταφερθούν. Από την άλλη πλευρά, υπάρχει ένας αριθμός φορτηγών, κάθε ένα από τα οποία μπορεί να χωρέσει συγκεκριμένο αριθμό παλετών. Οι εταιρίες έχουν συγκεκριμένες προτιμήσεις για τα φορτία τους στα διαθέσιμα φορτηγά βασιζόμενες στα χαρακτηριστικά του καθενός (πχ. τιμή, ποιότητα υπηρεσιών, κλπ) και τα φορτηγά έχουν προτιμήσεις στα φορτία (πχ. με βάση τον προορισμό τους, το φορτίο, κλπ). Οι προτιμήσεις αυτές εκφράζονται με τη χρήση λιστών προτίμησης, οι οποίες μοντελοποιούν ένα διαισθητικό κριτήριο για τις προτιμήσεις του κάθε συμμετέχοντα. Το πρόβλημα λοιπόν που πρέπει να επιλυθεί, είναι πως θα μεταφερθούν οι παλέτες που έχουμε με ευσταθή τρόπο.

Στη συνέχεια, θα δούμε τον τρόπο επίλυσης που προτείνουμε για την επίλυση του προβλήματός μας. Επίσης θα παρουσιάσουμε τον αλγόριθμο που υλοποιήσαμε, το διάγραμμα ροής καθώς και κάποια παραδείγματα του αλγορίθμου μας.

3.1. Τρόποι επίλυσης του προβλήματος

Για την επίλυση ενός τέτοιου προβλήματος, όπως έχουμε αναφέρει μπορεί να χρησιμοποιηθεί ο GS αλγόριθμος, για την περίπτωση που η κάθε εταιρία, μπορεί να συνεργαστεί μόνο με ένα φορτηγό και αντίστροφα. Παρόλα αυτά, όταν η κάθε εταιρία μπορεί να συνεργαστεί με περισσότερα από ένα φορτηγά και κάθε φορτηγό μπορεί να εξυπηρετήσει περισσότερες από μια εταιρίες, πρέπει να διαφοροποιήσουμε τον GS αλγόριθμο.

Ένα παρόμοιο πρόβλημα με το δικό μας, φαίνεται στο σχήμα 1.2, όπου μιλάμε για εργασίες, οι οποίες έχουν ένα συγκεκριμένο χρόνο επεξεργασίας και οι μηχανές έχουν συγκεκριμένη χωρητικότητα και θέλουμε να υλοποιηθούν οι εργασίες με το μικρότερο δυνατό κόστος. Η κάθε εργασία μπορεί να υλοποιηθεί σε περισσότερες από μια μηχανές και η κάθε μηχανή μπορεί να εκτελέσει παραπάνω από μια εργασίες (πολλά-προς-πολλά). Παρατηρούμε λοιπόν ότι το πρόβλημά μας, μπορεί να επιλυθεί προσεγγίζοντάς το σαν πρόβλημα ευσταθούς κατανομής.

Αναφερόμενοι στο πρόβλημα ευσταθούς κατανομής σε σχέση με το χρόνο επεξεργασίας γενικά, δεν υπάρχει περιορισμός στις τιμές. Δηλαδή, μπορούν να είναι ακέραιες, όπως και δεκαδικές. Στην περίπτωσή μας όμως, δεν μπορούμε να έχουμε δεκαδικούς αριθμούς. Γιατί, μια εταιρεία, δεν μπορεί να στείλει μισή παλέτα με το ένα φορτηγό και άλλη μισή παλέτα με ένα άλλο φορτηγό. Οι τιμές λοιπόν στις οποίες αναφερόμαστε, πρέπει να είναι ακέραιες. Άρα θα προσεγγίσουμε το πρόβλημά μας ως πρόβλημα ακεραίας ευσταθούς κατανομής [20].

3.2. Ακέραια ευσταθή κατανομή

Όπως έχουμε αναφέρει παραπάνω, το πρόβλημα της ευσταθούς κατανομής το εισήγαγαν οι Baïou και Balinski [6] για διμερής γράφους. Η ακέραια έκδοση αυτού, όπου η κατανομή x απαιτείται να είναι ακέραια σε κάθε κόμβο, είχε ονομαστεί ως το ευσταθές πρόβλημα χρονοδιαγράμματος (stable schedule problem) από τους Alkan και Gale [21]. Στην πραγματικότητα, θεώρησαν ένα πιο γενικό μοντέλο, με «υποκατάστατες» προτιμήσεις.

Για ένα πρόβλημα ευσταθούς κατανομής λοιπόν, όπως το αναφέραμε στην πρώτη παράγραφο του κεφαλαίου 2.4, αν έχουμε c(e) = 1 για κάθε ακμή e, τότε αυτή η ειδική περίπτωση, ονομάζεται πρόβλημα ευσταθούς b-ταιριάσματος (stable b-matching problem) [22]. Αν ο γράφος, περιέχει παράλληλες ακμές, τότε, αυτό το πρόβλημα είναι γνωστό και ως το ευσταθές πρόβλημα των πολλαπλών δραστηριοτήτων των Cechlárová και Fleiner [23]. Στην περίπτωση των απλών γράφων, το πρόβλημα αναφέρεται, ως ευσταθές πρόβλημα χαρακτηριστικών (stable fixtures problem), από τους Irving και Scott [24]. Αν b(v) = 1 για κάθε κόμβο, τότε έχει επιτευχθεί το ευσταθές πρόβλημα, και ονομάζεται ως το ευσταθές πρόβλημα συγκατοίκων για απλούς γράφους.

Αν το δεδομένο γράφημα είναι απλό και διμερές, τότε το πρόβλημα ευσταθούς b-ταιριάσματος, αποκαλείται και πρόβλημα ευσταθούς ταιριάσματος πολλά-προς-πολλά. Αν για κάθε κόμβο της μιας πλευράς του διμερούς γράφου ισχύει b(v) = 1, τότε το πρόβλημα αναφέρεται σε ένα πρόβλημα ευσταθούς αντιστοίχισης πολλά-προς-ένα, όπως το πρόβλημα εισαγωγής σε ένα πανεπιστήμιο για παράδειγμα, που αναφέραμε στο πρώτο κεφάλαιο. Τέλος, αν ισχύει b(v) = 1 για κάθε κόμβο του γράφου, τότε καταλήγουμε στο πρόβλημα του ευσταθούς γάμου. Αυτά τα προβλήματα έχουν μελετηθεί εκτενώς, μιας και η θεωρία του ευσταθούς

ταιριάσματος έχει γίνει ένα σημαντικό υποπεδίο τόσο της θεωρίας παιγνίων και της θεωρίας των αλγορίθμων [4], [25].

3.3 Δεδομένα προβλήματος

Καταλήξαμε λοιπόν, ότι για την μοντελοποίηση του προβλήματός μας, θα χρησιμοποιηθεί το υπόδειγμα της ακεραίας ευσταθούς κατανομής. Για να φτάσουμε στην επίλυση του προβλήματος όμως, χρειαζόμαστε κάποια δεδομένα και κάποιους περιορισμούς που πρέπει να λάβουμε υπόψη μας, πριν δημιουργήσουμε τον αλγόριθμό μας.

Αρχικά, όπως φαίνεται στο Σχήμα 1.5, έχουμε i εταιρίες και j φορτηγά, όπου i, j ϵ Z^{+}. Η κάθε εταιρία, έχει μια χωρητικότητα (company.capacity), η οποία αναφέρεται στον αριθμό των παλετών που χρειάζεται να μεταφέρει, καθώς και το κάθε φορτηγό έχει μια χωρητικότητα (truck.capacity), η οποία αντίστοιχα αναφέρεται στον αριθμό των παλετών που μπορεί να μεταφέρει το κάθε φορτηγό. Επιπλέον, έχουμε ως δεδομένη τη λίστα προτίμησης (prefs) της κάθε εταιρίας ως προς τα φορτηγά, καθώς και του κάθε φορτηγού ως προς τις εταιρίες. Τέλος, μας δίνεται και ένας πίνακας minvalues[i, j], ο οποίος αναφέρεται στη μέγιστη χωρητικότητα που μπορεί να μεταφέρει για την εταιρία i το φορτηγό j.

Έχοντας λοιπόν στη διάθεσή μας τα παραπάνω δεδομένα, μπορούμε να προχωρήσουμε και να δημιουργήσουμε τον αλγόριθμο, ο οποίος θα επιστρέφει το ποια φορτηγά θα εξυπηρετούν την κάθε εταιρία και πόσες παλέτες θα φορτωθούν σε αυτά (χωρητικότητα). Ο αλγόριθμος που θα υλοποιήσουμε αποτελεί στην ουσία μία γενίκευση του GS αλγορίθμου στο πρόβλημα αυτό.

3.4. Αλγόριθμος- διάγραμμα ροής

Στο κεφάλαιο αυτό, θα δούμε τον ψευδοκώδικα του αλγορίθμου που υλοποιήσαμε όπως φαίνεται στο Αλγόριθμος 3, καθώς και το διάγραμμα ροής αυτού όπως φαίνεται στο Σχήμα 3.1. Στη συνέχεια, θα αναλύσουμε τον ψευδοκώδικα και θα εξηγήσουμε τη λειτουργία του βήμα-βήμα.

Ψευδοκώδικας

```
1) do
2) {
3) flag = false
4) for i 0 to companies
5)  for j 0 to trucks
6)   Ορίζεται η τιμή tr
7)    if (minvalues[i, tr] < company.capacity[i] and  minvalues[i, tr] <
          truck.capacity[tr] and tableM[i, tr] < minvalues[i, tr])
8)   Υπολογίζεται η τιμή value
9)   Μειώνεται η απομένουσα χωρητικότητα της εταιρίας i  κατά value
10) Μειώνεται η απομένουσα χωρητικότητα του φορτηγού tr  κατά value
11) Αυξάνεται ανάθεση του tr ως προς την i κατά value
12) if (value > 0) flag = true
13) else if  (company.capacity[i] > 0 and tableM[i, tr] < minvalues[i, tr])
14) Υπολογίζεται η τιμή temp
15) Συγκρίνεται με την χωρητικότητα της i εταιρίας και του tr φορτηγού
16) Προκύπτει η τιμή value
17) Μειώνεται η απομένουσα χωρητικότητα της εταιρίας i  κατά value
18) Μειώνεται η απομένουσα χωρητικότητα του φορτηγού tr  κατά value
19) Αυξάνεται ανάθεση του tr ως προς την i κατά value
20) if (value > 0) flag = true
21) if (company.capacity[i] > 0 και το tr δεν βρίσκεται τελευταίο στη λίστα
      προτίμησης)
22)  Ορίζουμε ως pos τη θέση της company[i] στη λίστα προτίμησης του truck[tr]
23)  for b companies to pos b- -
24)   Ονομάζω com την εταιρία που βρίσκεται στη θέση b της λίστας προτίμησης
        του truck[tr]
25)   if (company.capacity[i] > 0 and tableM[i, tr] < minvalues[i,tr])
26)   Υπολογίζεται η τιμή temp1
27)   Συγκρίνεται με την χωρητικότητα της i εταιρίας και της χωρητικότητας που
        έχει εξυπηρετήσει το φορτηγό tr στην εταιρία com
28)    Προκύπτει η τιμή value
29)    Μειώνεται η απομένουσα χωρητικότητα της εταιρίας i  κατά value
30)    Αυξάνεται η χωρητικότητα της εταιρίας com  κατά value
31)    Αυξάνεται ανάθεση του tr ως προς την i κατά value
32)    Μειώνεται ανάθεση του tr ως προς την com κατά value
33)    if (value > 0) flag = true
```

34) Υπολογίζουμε την τιμή s

35) }while(flag = false or s=0)

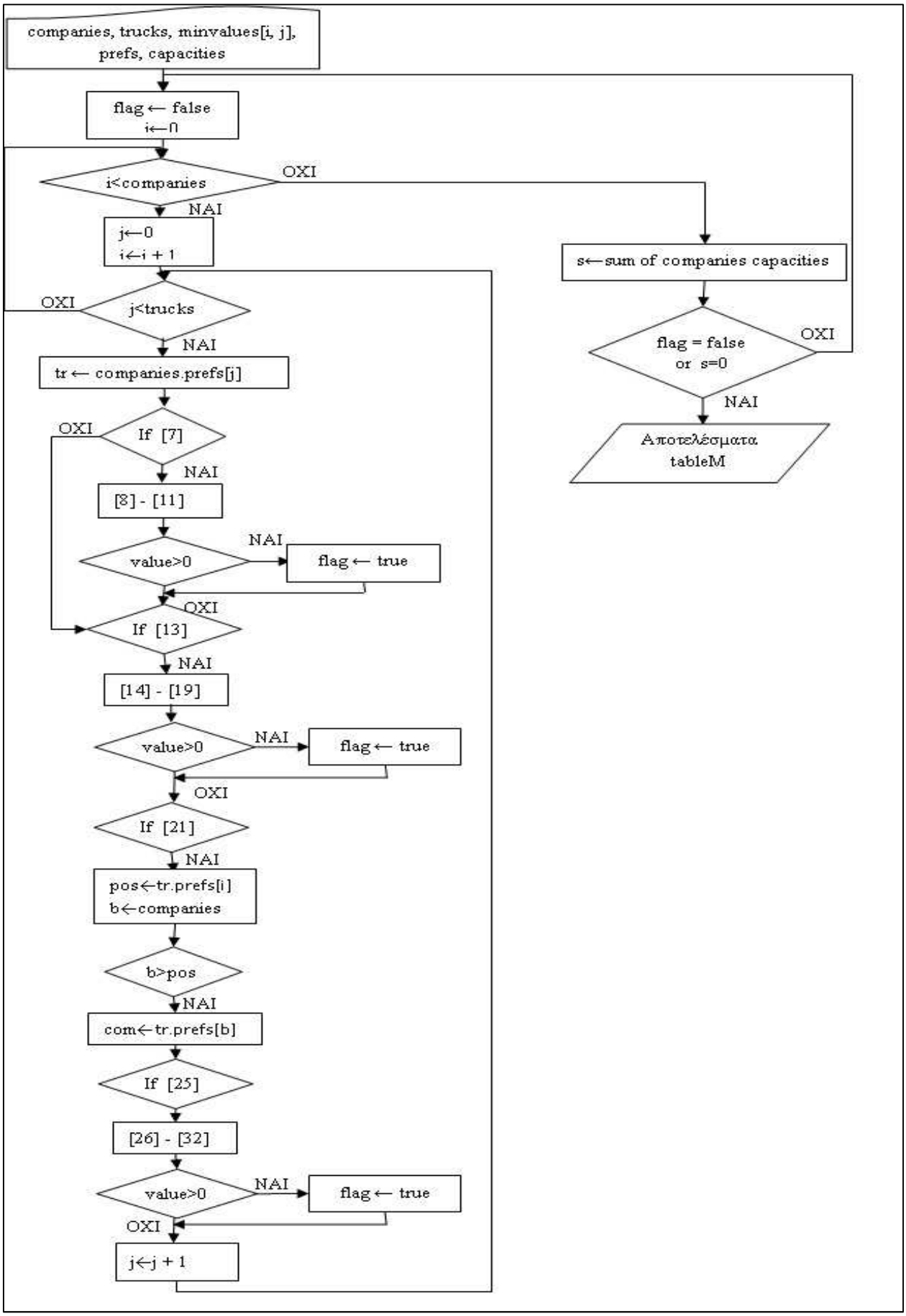
36) Αποτελέσματα tableM (πίνακας αντιστοίχισης χωρητικότητας που παρέχει το φορτηγό j στην εταιρία i)

Αλγόριθμος 3:Ο ψευδοκώδικας του αλγορίθμου μας

Πίνακας 3.1: Σημειογραφία που χρησιμοποιείται στον αλγόριθμο και περιγραφή κάθε παραμέτρου

flag	Λογική μεταβλητή, η οποία παίρνει την τιμή true, όταν έχει γίνει κάποια μεταβολή στον πίνακα tableM, κατά την εκτέλεση του αλγόριθμου
minvalues[i, j]	Πίνακας με τη μέγιστη τιμή που μπορεί να εξυπηρετήσει το φορτηγό j την εταιρία i
tableM	Τελικός πίνακας αποτελεσμάτων
s	άθροισμα από τις χωρητικότητες των εταιριών που δεν έχουν ικανοποιηθεί

Αρχικοποίηση δεδομένων (3): Αρχικά, για την επίλυση του αλγορίθμου, χρησιμοποιούμε μια λογική μεταβλητή, την flag, της οποίας της δίνουμε την τιμή false, όπως αναφέρουμε στον Πίνακα 3.1. Κάθε φορά που αλλάζει τιμή η flag και από false γίνεται true, σημαίνει ότι έχει γίνει κάποια αλλαγή στον πίνακα tableM, ο οποίος είναι ο πίνακας που θέλουμε να δημιουργήσουμε. Ο πίνακας αυτός, έχει διαστάσεις ixj, όπου i το πλήθος των εταιριών και j το πλήθος των φορτηγών και σε κάθε στοιχείο του πίνακα φαίνεται η χωρητικότητα που θα εξυπηρετήσει το φορτηγό j για την εταιρία i.



Σχήμα 3.1: Διάγραμμα ροής του αλγορίθμου

Υπολογισμός, προσωρινής ανάθεσης παλετών σε φορτηγά (4 - 12): Ξεκινά λοιπόν ο αλγόριθμος όπως φαίνεται στο Αλγόριθμος 3, να περνάει όλες τις εργασίες με τη σειρά. Για την κάθε εργασία ελέγχει τη λίστα προτίμησής της και με αυτή τη σειρά (με τη βοήθεια της μεταβλητής tr) περνάει όλα τα φορτηγά για να ελέγξει σε ποια μπορεί να δώσει την χωρητικότητα της εκάστοτε εταιρίας, μέχρι αυτή να γίνει 0, αν αυτό είναι δυνατό. Δηλαδή, το tr, μας δίνει με την σειρά που βρίσκονται στη λίστα προτίμησης, τα φορτηγά. Στη συνέχεια, ελέγχουμε αν η μέγιστη χωρητικότητα που μπορεί να εξυπηρετήσει το φορτηγό tr την εταιρία i, είναι μικρότερη από την χωρητικότητα που έχει απομείνει στην εργασία i και την χωρητικότητα που έχει απομείνει στο φορτηγό tr, καθώς και αν είναι μεγαλύτερη από την χωρητικότητα που έχει ήδη εξυπηρετήσει το φορτηγό tr την εταιρία i. Σε αυτή την περίπτωση, βρίσκουμε την διαφορά της μέγιστης χωρητικότητας, με τη χωρητικότητα που έχει ήδη ανατεθεί στο φορτηγό tr από την εταιρία i, και τη θέτουμε ως value. Μειώνουμε την απομένουσα χωρητικότητα της εταιρίας i κατά value, μειώνουμε την απομένουσα χωρητικότητα του φορτηγού tr κατά value και αυξάνουμε την ανάθεση του tr ως προς την i κατά value (tableM[i, tr]).

Ικανοποίηση άλλων συνθηκών, για προσωρινή ανάθεση παλετών σε φορτηγά (13 - 20): Αν δεν ισχύει η παραπάνω συνθήκη, τότε μπορούμε να ελέγξουμε την δεύτερη συνθήκη που έχουμε στον αλγόριθμο (else if). Ελέγχουμε λοιπόν, αν η χωρητικότητα της εταιρίας i είναι ακόμη μεγαλύτερη από το μηδέν και αν μπορεί το φορτηγό tr, να δώσει και άλλη χωρητικότητα στην εταιρία i. Εφόσον ισχύει αυτή η συνθήκη, βρίσκουμε την διαφορά της μέγιστης χωρητικότητας, με τη χωρητικότητα που έχει ήδη ανατεθεί στο φορτηγό tr από την εταιρία i, και τη θέτουμε ως temp. Στη συνέχεια συγκρίνουμε την temp με την χωρητικότητα που έχει απομείνει στην εταιρία i και με τη χωρητικότητα που έχει απομείνει στο φορτηγό tr. Όποια από τις τρεις τιμές είναι μικρότερη, την θέτουμε ως value. Τότε, όπως προηγουμένως, μειώνουμε την απομένουσα χωρητικότητα της εταιρίας i κατά value, μειώνουμε την απομένουσα χωρητικότητα του φορτηγού tr κατά value και αυξάνουμε την ανάθεση του tr ως προς την i κατά value (tableM[i, tr]). Για άλλη μια φορά, πρέπει να ελεγχθεί η χωρητικότητα της εταιρίας i, για να δούμε αν είναι ακόμη μεγαλύτερη από μηδέν.

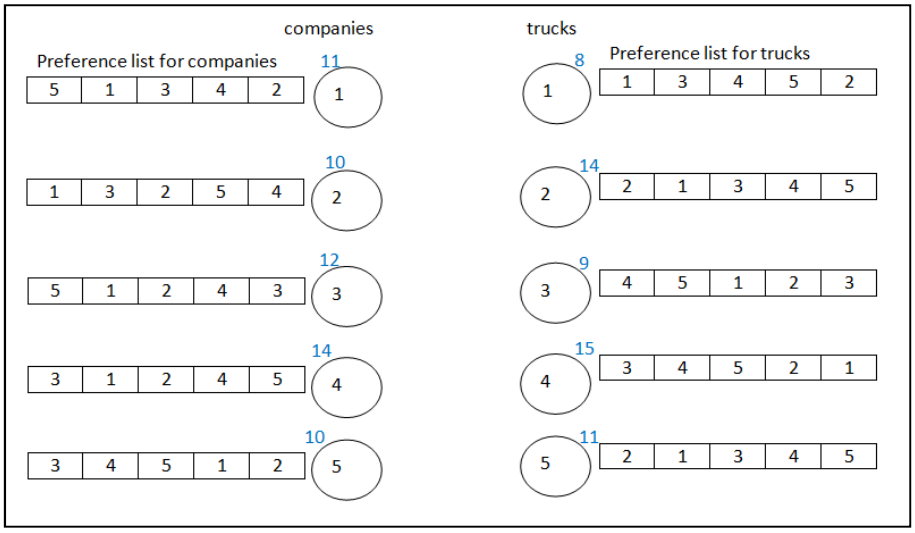
Έλεγχος προτίμησης στη λίστα και ανάθεση τιμών (21 - 33): Στην περίπτωση που δεν έχει καταφέρει να γίνει ακόμη μηδέν, ελέγχουμε σε ποια θέση της λίστας προτίμησης του φορτηγού tr, βρίσκεται η εταιρία i. Αν δεν είναι στην τελευταία, τότε ορίζουμε αυτή τη θέση ως pos και ξεκινάμε να ελέγχουμε τις εταιρίες

που βρίσκονται σε χειρότερη θέση από την pos, από την τελευταία, για να δούμε αν το φορτηγό tr, έχει δώσει κάποια χωρητικότητα σε κάποια από αυτές τις εταιρίες. Σε κάθε έλεγχο, η εταιρία που βρίσκεται σε αυτή την χειρότερη θέση από την pos είναι η com. Σε κάθε κύκλο, μέχρι να φτάσουμε στην pos, ελέγχουμε αν συνεχίζει η χωρητικότητα της εταιρίας να είναι μεγαλύτερη του μηδενός, καθώς και αν η χωρητικότητα που μπορεί να δώσει το φορτηγό tr στην εταιρία i, είναι μεγαλύτερη από αυτή που έχει ήδη δώσει. Τότε, βρίσκουμε τη διαφορά τους και την ορίζουμε ως temp1. Στη συνέχεια συγκρίνεται με την χωρητικότητα της i εταιρίας και της χωρητικότητας που έχει εξυπηρετήσει το φορτηγό tr στην εταιρία com. Η μικρότερη από τις τρεις αυτές τιμές ονομάζεται value. Οπότε, μειώνουμε την απομένουσα χωρητικότητα της εταιρίας i και την ανάθεση του tr ως προς την com κατά value και αυξάνουμε τη χωρητικότητα της εταιρίας com και την ανάθεση του tr ως προς την i κατά value.

Συνθήκες τερματισμού αλγορίθμου (34 - 35): Όταν έχουν εκτελεστεί τα παραπάνω, έχει εκτελεστεί ένας ολόκληρος κύκλος του αλγορίθμου μας. Δηλαδή, έχουν ελέγξει όλες οι εταιρίες, από μια φορά, τι χωρητικότητα θα μπορούσαν να πάρουν από το κάθε φορτηγό. Τότε, υπολογίζουμε την τιμή s, η οποία είναι το άθροισμα από τις χωρητικότητες των εταιριών που δεν έχουν εξυπηρετηθεί ακόμη. Δηλαδή, ελέγχουμε την s και την μεταβλητή flag. Αν η μεταβλητή s, είναι μηδέν, τότε σημαίνει ότι όλες οι εταιρίες έχουν εξυπηρετηθεί πλήρως, άρα ο αλγόριθμός μας τερματίζεται. Σε περίπτωση όπου $s \neq 0$, αν η μεταβλητή flag έχει την τιμή false, τότε σημαίνει ότι έχει εκτελεστεί ένας ολόκληρος κύκλος του αλγορίθμου μας, χωρίς να έχει προκύψει καμία απολύτως αλλαγή στις αναθέσεις των εταιριών στα φορτηγά, οπότε και πάλι ο αλγόριθμός μας τερματίζεται.

3.5. Παράδειγμα εκτέλεσης του αλγορίθμου

Για την καλύτερη κατανόηση του αλγόριθμου που περιγράψαμε παραπάνω, παρέχουμε ένα παράδειγμα, ώστε να μπορέσουμε να δούμε στην πράξη, πως εκτελείται ο αλγόριθμός μας. Πρώτα θα οριστούν τα δεδομένα που χρειάζονται.



Σχήμα 3.2: Παράδειγμα με πέντε εταιρίες και πέντε φορτηγά (5X5)

Ας υποθέσουμε λοιπόν, ότι έχουμε πέντε εταιρίες και πέντε φορτηγά. Στο Σχήμα 3.2, μπορούμε να δούμε τους πέντε κόμβους των εταιριών αριστερά και τους πέντε κόμβους των φορτηγών δεξιά. Ο κάθε κόμβος έχει ένα id = {1, …,5}, μια χωρητικότητα (capacity) (η οποία φαίνεται δίπλα σε κάθε κόμβο) και μια λίστα προτιμήσεων σε φορτηγά για τις εταιρίες και των προτιμήσεων σε εταιρίες για τα φορτηγά που φαίνονται με μορφή κελιών. Τέλος, μας δίνεται πίνακας minvalues, όπως φαίνεται στο Σχήμα 3.3, ο οποίος μας δηλώνει για τη μέγιστη χωρητικότητα που μπορεί να δεχτεί το φορτηγό j από την εταιρία i, ή η μέγιστη χωρητικότητα που μπορεί να δώσει η εταιρία i στο φορτηγό j.

trucks (j)

companies (i)	1	2	3	4	5
1	7	10	9	11	9
2	7	8	9	8	8
3	8	12	8	11	10
4	8	11	8	12	11
5	6	10	7	9	9

Σχήμα 3.3: Πίνακας minvalues

Έχουμε οπότε όλα μας τα δεδομένα και μπορούμε να αρχίσουμε την εκτέλεση του αλγορίθμου μας. Ξεκινάμε οπότε τον πρώτο κύκλο εκτέλεσης. Θα περάσουμε δηλαδή από όλες τις εταιρίες, όπου η κάθε εταιρία θα προτείνει σε όλα τα φορτηγά συνεργασία, σύμφωνα με τη λίστα προτίμησής της.

Αρχικά, η λογική μεταβλητή μας flag, αρχικοποιείται με την τιμή false. Όσο η flag έχει αυτή την τιμή, ο πίνακας tableM δεν μεταβάλλεται, ενώ μόλις πάρει την τιμή true, έχουμε μεταβολή στον πίνακα tableM. Στη συνέχεια η μεταβλητή i, γίνεται 1, το οποίο σημαίνει ότι η εταιρία 1 αρχίζει να προτείνει σε φορτηγά. Το j γίνεται 1 και πάμε στην λίστα προτιμήσεων της εταιρίας 1 όπου βλέπουμε ότι σε αυτή τη θέση βρίσκεται το φορτηγό 5 (tr). Άρα, η εταιρία 1, προτείνει στο φορτηγό 5. Συνεχίζοντας, παρατηρούμε ότι ισχύει η συνθήκη if (όπως φαίνεται στη γραμμή 7 του αλγόριθμου που περιγράψαμε προηγουμένως). Ο πίνακας tableM, δεν έχει δεχτεί ακόμη τιμές, οπότε όλα του τα στοιχεία είναι 0. Για να βρούμε την τιμή value θα κάνουμε την αφαίρεση minvalues[i, tr] – tableM[i, tr], η οποία έχει την τιμή 9. Άρα η εταιρία 1, θα στείλει 9 μονάδες χωρητικότητας στο φορτηγό 5 και η flag γίνεται true. Επίσης, κατά 9 μονάδες θα μειωθεί η χωρητικότητα της εταιρίας i, καθώς και η χωρητικότητα του φορτηγού tr. Αμέσως μετά, το j παίρνει την τιμή 2 και το tr, παίρνει την τιμή 1. Σε αυτή την επανάληψη, η συνθήκη if (γραμμή κώδικα αλγόριθμου 7) δεν ισχύει και ελέγχεται η συνθήκη else if (γραμμή κώδικα αλγόριθμου 13), η οποία ισχύει. Βρίσκουμε την temp, η οποία είναι η διαφορά minvalues[i, tr] – tableM[i, tr] και είναι 7. Στη συνέχεια συγκρίνουμε την temp, με την χωρητικότητα που έχει απομείνει στην εταιρία i, δηλαδή 2, και την χωρητικότητα που έχει απομείνει στο φορτηγό tr, που είναι 8. Η μικρότερη από τις τρεις τιμές, η 2, καταχωρείται στη μεταβλητή value. Οπότε η εταιρία 1, θα στείλει 2 μονάδες χωρητικότητας στο φορτηγό 1, δηλαδή το tableM[1, 2] θα γίνει 2. Επίσης, κατά 2 μονάδες θα μειωθεί η χωρητικότητα της εταιρίας i, καθώς και η χωρητικότητα του φορτηγού tr. Τότε, η χωρητικότητα της εταιρίας 1 έχει γίνει 0 και παρότι θα περάσει ο αλγόριθμος και την υπόλοιπη λίστα προτίμησής της, δεν θα κάνει καμιά άλλη αλλαγή για αυτήν την εταιρία. Τα αποτελέσματα της διαδικασίας φαίνονται στο Σχήμα 3.4 και στο Σχήμα 3.5.

Στη συνέχεια, το i γίνεται 2, οπότε αρχίζουμε να μιλάμε για την δεύτερη εταιρία. Αντίστοιχα, αρχίζουμε να αλλάζουμε το j και να περνάμε με τη σειρά τα φορτηγά από τη λίστα προτίμησης της εταιρίας 2. Πάμε λοιπόν πρώτα, να ελέγξουμε το φορτηγό 1, του οποίου του έχει απομείνει χωρητικότητα 6, αφού έδωσε 2 μονάδες πριν. Πάμε στο else if (γραμμή κώδικα αλγόριθμου 13) του αλγόριθμου, το οποίο ισχύει και δίνονται όπως παραπάνω 6 μονάδες στην εταιρία 2. Στη συνέχεια, εφόσον η χωρητικότητα της εταιρίας δεν είναι ακόμη 0, ελέγχεται η συνθήκη if (γραμμή κώδικα αλγόριθμου 21), αλλά η εταιρία 2, είναι τελευταία στη λίστα προτίμησης του

φορτηγού 1. Οπότε συνεχίζει ο αλγόριθμος, το j γίνεται 2. Εδώ όπως είδαμε και παραπάνω, για το φορτηγό 3, ισχύει η συνθήκη else if (γραμμή κώδικα αλγόριθμου 13) και όπως παραπάνω, δίνονται 4 μονάδες χωρητικότητας στην εταιρία 2 από το φορτηγό 3.

Στη συνέχεια, το i γίνεται 3. Αντίστοιχα, αρχίζουμε να αλλάζουμε το j και να ελέγχουμε με τη σειρά τα φορτηγά από τη λίστα προτίμησης της εταιρίας 3. Πάμε λοιπόν πρώτα, να ελέγξουμε το φορτηγό 5, του οποίου του έχει απομείνει χωρητικότητα 2, αφού έδωσε 9 μονάδες πριν, στην εταιρία 1. Πάμε στο else if (γραμμή κώδικα αλγόριθμου 13) του αλγόριθμου, το οποίο ισχύει και δίνονται 2 μονάδες στην εταιρία 3. Στη συνέχεια, εφόσον η χωρητικότητα της εταιρίας δεν είναι ακόμη 0, ελέγχεται η συνθήκη if (γραμμή κώδικα αλγόριθμου 21), η οποία ισχύει. Βρίσκεται οπότε η θέση pos, της εταιρίας 3 στη λίστα προτίμησης του φορτηγού 5, η οποία είναι επίσης 3, όπως φαίνεται και στο Σχήμα 3.2. Το πρόβλημα σε αυτή την περίπτωση, είναι ότι παρόλο που θα ξεκινήσει η επανάληψη (γραμμή κώδικα αλγόριθμου 23) και ισχύει και η συνθήκη if (γραμμή κώδικα αλγόριθμου 25), η τιμή value θα προκύπτει 0. Γιατί αρχικά, ο αλγόριθμος, υπολογίζει την τιμή temp1, η οποία είναι η διαφορά minvalues[i, tr] – tableM[i, tr], η οποία είναι 8. Στη συνέχεια, αυτή η τιμή συγκρίνεται με την χωρητικότητα που έχει απομείνει στην εταιρία 3 που είναι 10 και με τη χωρητικότητα που έχει εξυπηρετήσει το φορτηγό tr στην εταιρία com (όπου com είναι η εταιρία που βρίσκεται στη θέση pos της λίστας προτίμησης του φορτηγού 5) που και στις 2 επαναλήψεις που θα γίνουν θα είναι 0, καθώς δεν έχει δώσει καθόλου χωρητικότητα το φορτηγό 5, στις εταιρίες 4 και 5. Οπότε, εφόσον στη value μπαίνει η μικρότερη τιμή και στις δύο περιπτώσεις θα είναι 0 και δεν θα γίνει καμιά αλλαγή στον πίνακα tableM. Αντίθετα, όταν το j, γίνει 2, οπότε θα αναφερόμαστε στο φορτηγό 1, τα πράγματα θα είναι διαφορετικά. Σε αυτή την περίπτωση, ελέγχεται η συνθήκη else if (γραμμή κώδικα αλγόριθμου 13) του αλγόριθμου, η οποία ισχύει, αλλά το φορτηγό 1, δεν έχει διαθέσιμη χωρητικότητα να δώσει. Τότε, εκτελείται η συνθήκη if (γραμμή κώδικα αλγόριθμου 21) η οποία ισχύει, ξεκινάει η επανάληψη (γραμμή κώδικα αλγόριθμου 23) από την τελευταία θέση της λίστας προτίμησης του φορτηγού 1, όπου βρίσκεται η εταιρία 2 και ισχύει και η συνθήκη if (γραμμή κώδικα αλγόριθμου 25). Τότε η temp1, υπολογίζεται και θα πάρει την τιμή 8. Στη συνέχεια, αυτή η τιμή συγκρίνεται με την χωρητικότητα που έχει απομείνει στην εταιρία 3 που είναι 10 και με τη χωρητικότητα που έχει εξυπηρετήσει το φορτηγό 1 στην εταιρία 2 είναι 6. Οπότε η τιμή value γίνεται 6. Τότε

μειώνεται η απομένουσα χωρητικότητα της εταιρίας 3 κατά value και γίνεται 4, αυξάνεται η χωρητικότητα της εταιρίας 2 κατά value και γίνεται 6, αυξάνεται ανάθεση του φορτηγού 1 ως προς την εταιρία 3 κατά value και γίνεται 6 και τέλος μειώνεται η ανάθεση του φορτηγού 1 ως προς την εταιρία 2 κατά value και γίνεται 0. Άρα το tableM[2, 1] μηδενίζεται και το tableM[3, 1] γίνεται 6. Τα αποτελέσματα της διαδικασίας φαίνονται στο Σχήμα 3.4 και στο Σχήμα 3.5.

Με τον ίδιο τρόπο, συνεχίζει η κάθε εταιρία, να προσπαθεί να μηδενίσει την χωρητικότητά της. Εφόσον ολοκληρωθεί ο πρώτος κύκλος επαναλήψεων, έχει προκύψει ο πίνακας tableM, όπως φαίνεται στο Σχήμα 3.4.

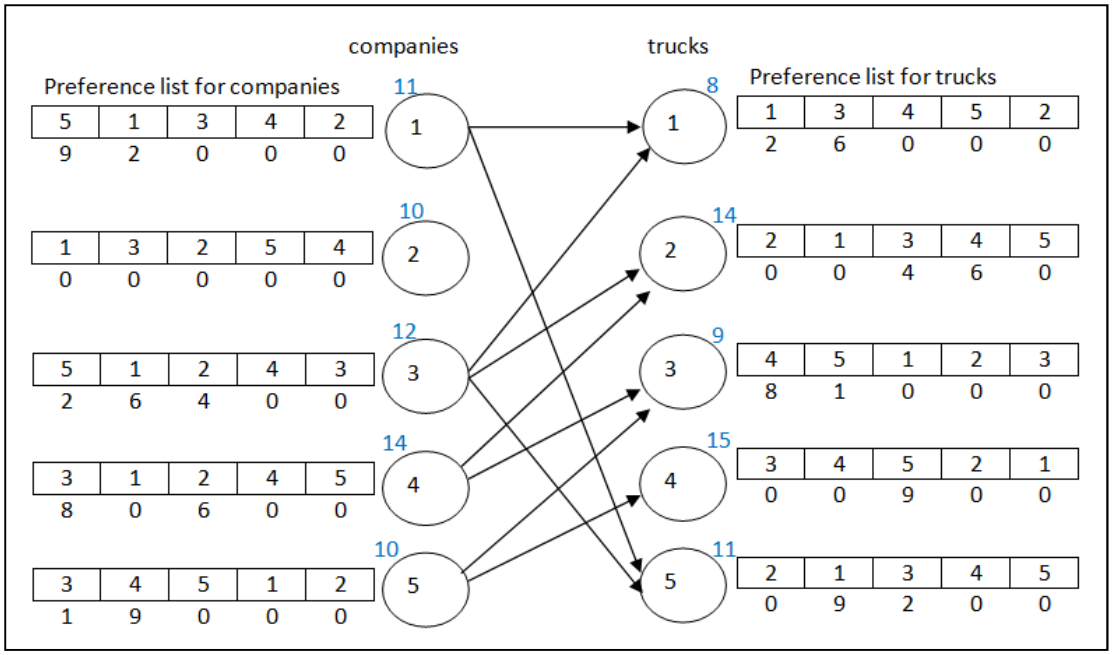
trucks (j)

companies (i)	1	2	3	4	5
1	2	0	0	0	9
2	0	0	0	0	0
3	6	4	0	0	2
4	0	6	8	0	0
5	0	0	1	9	0

Σχήμα 3.4: Πίνακας tableM μετά τον πρώτο κύκλο επαναλήψεων

Τα ίδια αποτελέσματα, τα βλέπουμε καλύτερα στο Σχήμα 3.5. Παρατηρούμε, ότι ενώ στην αρχή η εταιρία 2, είχε ικανοποιηθεί πλήρως, με το τέλος του πρώτου κύκλου, δεν έχει κανένα φορτηγό να την εξυπηρετεί.

Στη συνέχεια, υπολογίζεται το s, το οποίο προκύπτει 10. Επίσης η flag έχει γίνει ήδη true. Οπότε η συνθήκη δεν ικανοποιείται και πάμε για τον δεύτερο κύκλο εκτέλεσης.



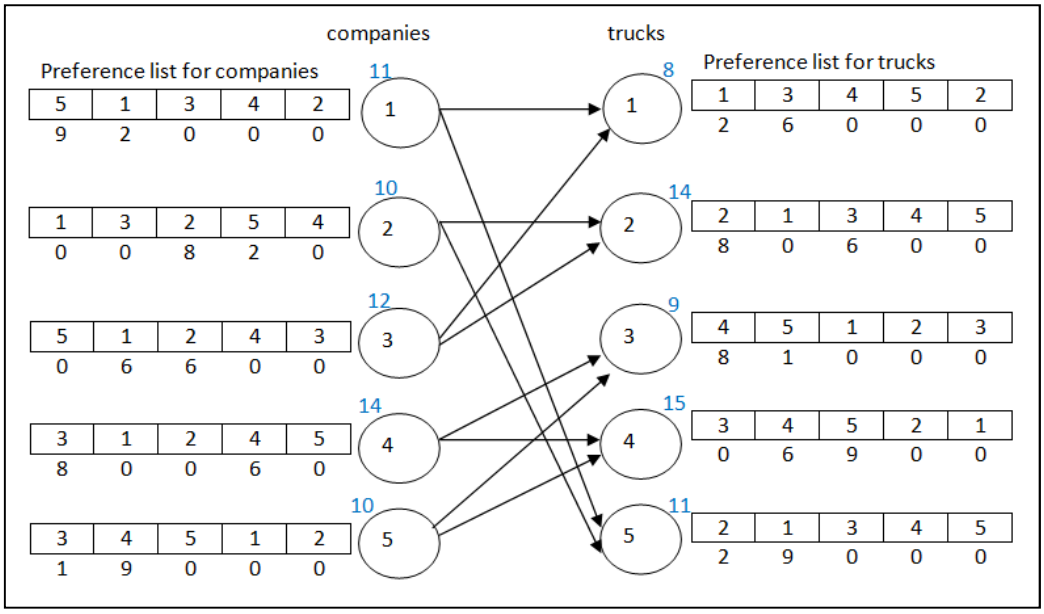
Σχήμα 3.5: Σχηματική αναπαράσταση μετά τον πρώτο κύκλο επαναλήψεων

Ο δεύτερος κύκλος του αλγόριθμου, εκτελείται με τον ίδιο τρόπο όπως παραπάνω. Σε αυτόν, όλες οι χωρητικότητες ικανοποιούνται, με αποτέλεσμα το s να προκύπτει 0. Η flag, θα είναι και πάλι true, αλλά εφόσον ικανοποιείται η μια συνθήκη από τις δύο (flag = false or s = 0, γραμμή κώδικα αλγορίθμου 35), ο αλγόριθμος τερματίζεται και το αποτέλεσμα φαίνεται στα σχήματα 3.6 και 3.7.

trucks (j)

companies (i)	1	2	3	4	5
1	2	0	0	0	9
2	0	8	0	0	2
3	6	6	0	0	0
4	0	0	8	6	0
5	0	0	1	9	0

Σχήμα 3.6: Τελικός πίνακας tableM



Σχήμα 3.7: Τελική σχηματική αναπαράσταση

Σε αυτό το κεφάλαιο, αναφέραμε τρόπους επίλυσης του προβλήματος που επιλύουμε στη παρούσα εργασία. Καταλήξαμε ότι για την επίλυση του προβλήματός μας, θα χρησιμοποιηθεί ακέραια ευσταθή κατανομή. Στη συνέχεια, ορίσαμε τα δεδομένα που χρειάζονται για την επίλυση του προβλήματός μας, παρείχαμε τον σχετικό αλγόριθμο και παρουσιάσαμε ένα παράδειγμα της λειτουργίας του.

Στο επόμενο κεφάλαιο παρουσιάζουμε αρχικά τον μηχανισμό δημιουργίας δεδομένων προβλήματος που δημιουργήσαμε. Με την βοήθειά του, δημιουργούμε δεδομένα και τα εισάγουμε στον αλγόριθμό μας. Αυτό μας επιτρέπει να παρουσιάσουμε και να αναλύσουμε τα αποτελέσματα που προκύπτουν.

Κεφάλαιο 4: Εκτέλεση αλγορίθμου και καταγραφή αποτελεσμάτων

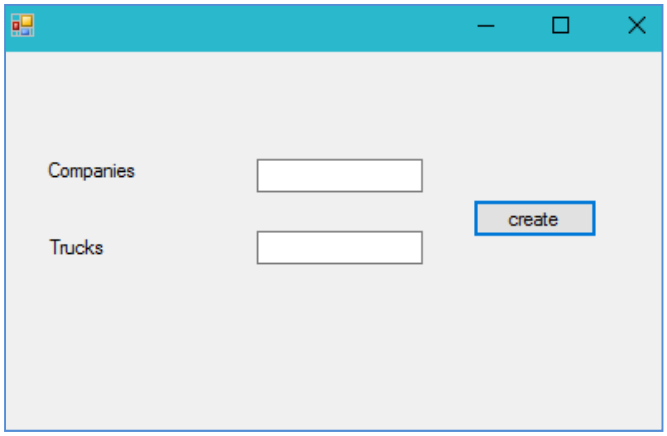
Στο προηγούμενο κεφάλαιο, έγινε περιγραφή του αλγόριθμου που λύνει την ακέραια ευσταθή κατανομή και σε αυτό το κεφάλαιο θα εξετάσουμε την αποτελεσματικότητα του αλγορίθμου παρουσιάζοντας αποτελέσματα εκτέλεσής του υπό διαφορετικά σενάρια. Για τον λόγο αυτό, φτιάξαμε μια γεννήτρια τυχαίων προβλημάτων ευσταθούς κατανομής, ένα μηχανισμό δηλαδή, ο οποίος δημιουργεί τυχαίες τιμές δεδομένων (για παράδειγμα χωρητικότητα, λίστα προτιμήσεων κ.τ.λ.) από ένα δεδομένο εύρος τιμών που έχουμε ορίσει. Βέβαια σε ένα ρεαλιστικό πρόβλημα, αυτά τα δεδομένα είναι γνωστά και πολλές φορές μπορεί να εμφανιστούν περιστάσεις οι οποίες δεν μπορούν εύκολα να προβλεφθούν. Άρα, το να θεωρήσουμε κάποια τυχαία δεδομένα για να εξετάσουμε την αποτελεσματικότητα του αλγορίθμου μας ίσως να είναι μια ρεαλιστική επιλογή.

Όπως αναφέραμε στο κεφάλαιο 1, θα προσπαθήσουμε να επιλύσουμε το πρόβλημα ενός συνόλου εταιριών που θέλει να μεταφέρει συγκεκριμένο αριθμό παλετών με την βοήθεια φορτηγών. Πιο συγκεκριμένα, θα απαρτίζεται από πολλές εταιρίες, όπου η κάθε μια θα έχει έναν αριθμό παλετών που πρέπει να μεταφερθούν. Από την άλλη πλευρά, υπάρχει ένας αριθμός φορτηγών, κάθε ένα από τα οποία, μπορεί να χωρέσει συγκεκριμένο αριθμό παλετών. Οι εταιρίες έχουν συγκεκριμένες προτιμήσεις για τα φορτία τους στα διαθέσιμα φορτηγά βασιζόμενες στα χαρακτηριστικά του καθενός και τα φορτηγά έχουν προτιμήσεις στα φορτία. Οι προτιμήσεις αυτές εκφράζονται με τη χρήση λιστών προτίμησης, οι οποίες μοντελοποιούν ένα διαισθητικό κριτήριο για τις προτιμήσεις του κάθε συμμετέχοντα. Το πρόβλημα λοιπόν που πρέπει να επιλυθεί, είναι πως θα μεταφερθούν οι παλέτες που έχουμε με ευσταθή τρόπο.

4.1: Μηχανισμός γέννησης προβλημάτων και υλοποίηση του αλγορίθμου

Στον μηχανισμό γέννησης προβλημάτων, αρχικά, απαιτείται από τον χρήστη να εισάγεται το πλήθος των εταιριών καθώς και το πλήθος των φορτηγών. Για να γίνει αυτό, εμφανίζεται ένα παράθυρο, όπως φαίνεται στο Σχήμα 4.1. Μόλις

πατήσουμε το κουμπί “create”, ξεκινάει η διαδικασία δημιουργίας των δεδομένων μας και αμέσως μετά, εκτελείται ο αλγόριθμος που δημιουργήσαμε (κεφάλαιο 3.4.).



Σχήμα 4.1: Εισαγωγή πλήθους εταιριών και φορτηγών

Με την εισαγωγή του αριθμού των εταιριών και των φορτηγών, το πρώτο που θα κάνει ο μηχανισμός μας, είναι να δημιουργήσει τις εταιρίες και τα φορτηγά. Θα δοθεί ένα id στο καθένα από αυτά καθώς και μια χωρητικότητα, η οποία είναι μια τυχαία τιμή από το 10 έως το 20. Αναφερόμενοι στον όρο χωρητικότητα, εννοούμε αριθμό παλετών. Από την πλευρά των εταιριών, έχουμε τις παλέτες τις οποίες θέλει να μεταφέρει η κάθε εταιρία και από την πλευρά των φορτηγών, έχουμε τις παλέτες τις οποίες μπορεί να μεταφέρει το κάθε φορτηγό. Αυτό που πρέπει να επισημάνουμε, είναι ότι οι χωρητικότητες είναι τυχαίοι αριθμοί, όπως θα μπορούσε εν μέρει να ήταν και σε ένα αληθινό πρόβλημα. Επίσης πρέπει να σημειωθεί, ότι στο τέλος όλης της διαδικασίας, μπορεί να υπάρξουν εταιρίες οι οποίες να μην εξυπηρετηθούν πλήρως ή να υπάρχει περισσευούμενη χωρητικότητα από την πλευρά των φορτηγών.

Στη συνέχεια για την κάθε εταιρία, καθώς και για το κάθε φορτηγό, δημιουργείται μια λίστα προτιμήσεων. Φτιάχνεται δηλαδή μια λίστα με τα id του απέναντι συνόλου και στη συνέχεια, γίνεται ένα ανακάτεμα (shuffle) αυτών. Το ανακάτεμα, γίνεται με αντιμετάθεση στα στοιχεία της λίστας ανά δύο, όπου οι τιμές των στοιχείων που θα αλλάξουν θέση μεταξύ τους, η μία επιλέγεται τυχαία και η άλλη περνάει με τη σειρά τις θέσεις της λίστας. Τέλος, δημιουργείται ο πίνακας minvalues[i, j]. Σε αυτόν τον πίνακα, φαίνεται η μέγιστη χωρητικότητα που μπορεί να δοθεί από την εταιρία i στο φορτηγό j. Για την δημιουργία του κάθε στοιχείου του πίνακα αυτού, συγκρίνεται η ολική χωρητικότητα της εταιρίας i, με την ολική

χωρητικότητα του φορτηγού j, κρατείται η μικρότερη από τις δύο (temp) και στον πίνακα εισάγεται μια τυχαία τιμή ανάμεσα στην temp και στην temp - 5.

Μετά τη δημιουργία όλων των δεδομένων ορίζουμε έναν μετρητή, ο οποίος ξεκινάει μαζί με τον αλγόριθμό μας, ώστε να μπορούμε να γνωρίζουμε τον χρόνο εκτέλεσης αυτού. Ένα παράδειγμα εκτέλεσης όλης της διαδικασίας αυτής, μπορούμε να δούμε στο Σχήμα 4.2, για ένα πολύ μικρό παράδειγμα, όπου συμμετέχουν δύο εταιρίες και δύο φορτηγά.

```
Company with id= 1  and capacity=  11
 has preference truck= 2
 has preference truck= 1
Company with id= 2  and capacity=  11
 has preference truck= 2
 has preference truck= 1
Truck with id= 1 and capacity=  16
 has preference company= 2
 has preference company= 1
Truck with id= 2 and capacity=  19
 has preference company= 2
 has preference company= 1
minvalues:
8 6
11 10
Loop 1:
tableM[1, 2 ] = 6 tableM[1, 1 ] = 5
Loop 2:
tableM[2, 2 ] = 10 tableM[2, 1 ] = 1
table M final:
5 6
1 10
Total duration (sec): 0,0067946
```

Σχήμα 4.2: Παράδειγμα εκτέλεσης αλγορίθμου με δύο εταιρίες και δύο φορτηγά (2X2)

Βλέπουμε λοιπόν, από το Σχήμα 4.2, δίνεται το id, η χωρητικότητα (capacity) και μια λίστα προτίμησης σε κάθε συμμετέχοντα (για παράδειγμα η εταιρία 1 που έχει capacity 11 μονάδες με λίστα προτίμησης τα [2, 1] φορτηγά). Αμέσως μετά εμφανίζεται ο πίνακας minvalues. Τέλος, δίνονται τα αποτελέσματα του κάθε κύκλου εκτέλεσης του αλγορίθμου μας, ώσπου να προκύψει ο πίνακας tableM, καθώς και ο χρόνος που χρειάστηκε για να δημιουργηθεί σε δευτερόλεπτα (sec). Ο tableM, δηλώνει το τελικό αποτέλεσμα του αλγορίθμου μας. Η πρώτη γραμμή του tableM, δηλώνει τον αριθμό παλετών (χωρητικότητα), που έδωσε η εταιρία 1 στα φορτηγά (5 στο φορτηγό 1 και 6 στο φορτηγό 2). Η δεύτερη γραμμή του tableM, δηλώνει τον αριθμό παλετών (χωρητικότητα), που έδωσε η εταιρία 2 στα φορτηγά (1 στο φορτηγό 1 και 10 στο φορτηγό 2). Αντίστοιχα, η πρώτη στήλη, δηλώνει τον αριθμό παλετών που μετέφερε το φορτηγό 1 για τις εταιρίες (5 παλέτες για την εταιρία 1 και 1 παλέτα

για την εταιρία 2) και η δεύτερη στήλη, δηλώνει τον αριθμό παλετών που μετέφερε το φορτηγό 2 για τις εταιρίες (6 παλέτες για την εταιρία 1 και 10 παλέτες για την εταιρία 2).

Επίσης, παρατηρούμε, ότι το άθροισμα της χωρητικότητας των εταιριών, είναι μικρότερο από το άθροισμα της χωρητικότητας των φορτηγών. Οπότε περιμέναμε το γεγονός ότι εξυπηρετήθηκαν πλήρως οι εταιρίες από τα φορτηγά. Επίσης, καθώς και οι δύο εταιρίες, προτιμούν το φορτηγό 2 από το φορτηγό 1, δίνουν σε αυτό όσο περισσότερη χωρητικότητα μπορούν και την υπόλοιπη την δίνουν στο φορτηγό 1, εφόσον αυτό βρίσκεται πιο χαμηλά στις λίστες προτίμησης.

4.2: Προδιαγραφές συστήματος

Για να μπορέσουμε να αξιολογήσουμε τα αποτελέσματα τα οποία παίρνουμε από τον αλγόριθμο που δημιουργήσαμε, θα πρέπει να ενημερώσουμε για τις προδιαγραφές του συστήματος, στο οποίο εκτελέστηκε. Έτσι ώστε, να μπορούμε να καταλάβουμε πως θα λειτουργούσε σε ένα διαφορετικού τύπου σύστημα.

Αρχικά, θα πρέπει να αναφέρουμε ότι το πρόγραμμα είναι γραμμένο σε γλώσσα C# (sharp). Για τη δημιουργία της εφαρμογής χρησιμοποιήθηκε το Microsoft Visual Studio 2013.

Ο υπολογιστής που χρησιμοποιήθηκε για την εκτέλεση έχει λειτουργικό σύστημα Windows 10, 64 bit. Έχει επεξεργαστή Intel(R) Core i5(TM) – 4200U CPU @ 1.60 GHz 2.30 GHz, μνήμη RAM 4,00 GB.

4.3: Εξέταση αποτελεσμάτων σε σχέση με τον αριθμό των κόμβων

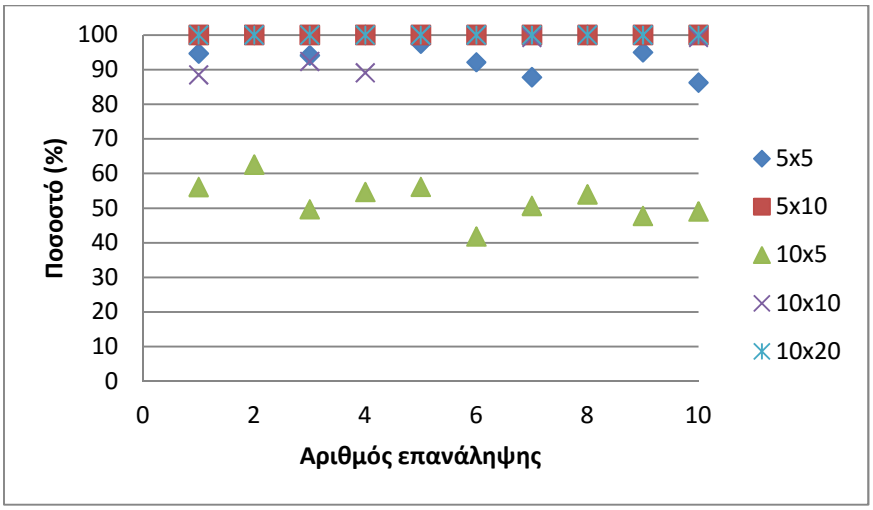
Ο αλγόριθμος εκτελέστηκε αρκετές φορές υπό διαφορετικά σενάρια, με διαφορετικό αριθμό κόμβων. Με αυτό τον τρόπο, μπορούμε να δούμε, για τις διάφορες τιμές εταιριών και φορτηγών, κατά πόσο έχουμε τα επιθυμητά αποτελέσματα. Δηλαδή, αν όλες οι εταιρίες εξυπηρετούνται πλήρως από τα διαθέσιμα φορτηγά. Αυτό φυσικά, δεν είναι πάντα εφικτό, καθώς για να επιτευχθεί, θα πρέπει η χωρητικότητα που διαθέτουν όλα τα φορτηγά συνολικά, να είναι ίση ή μεγαλύτερη από τη συνολική χωρητικότητα που θέλουν να δώσουν οι εταιρίες.

Για κάθε ζεύγος τιμών που χρησιμοποιήσαμε, εκτελέσαμε δέκα παραδείγματα, ώστε να έχουμε μια πιο πλήρη εικόνα της λειτουργίας του

αλγορίθμου, όπως θα δούμε στη συνέχεια. Αρχίσαμε με πέντε εταιρίες και πέντε φορτηγά (5x5) και συνεχίσαμε με πέντε εταιρίες και δέκα φορτηγά (5x10), με δέκα εταιρίες και πέντε φορτηγά (10x5), με δέκα εταιρίες και δέκα φορτηγά (10x10) και με δέκα εταιρίες και είκοσι φορτηγά (10x20). Στον Πίνακα 4.1, μπορούμε να δούμε τα αποτελέσματα που βγάλαμε από την εκτέλεση των παραδειγμάτων μας, καθώς και τη σχηματική αναπαράσταση αυτού στο Σχήμα 4.3.

εταιρίες x φορτηγά	αριθμός επανάληψης και μέσος όρος										
	1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**M.O.**
5x5	94,67	100	94,12	100	97,47	92,13	87,8	100	95	86,25	94,74
5x10	100	100	100	100	100	100	100	100	100	100	100
10x5	56,12	62,59	49,69	54,68	56,16	41,83	50,67	54,01	47,77	49,06	52,26
10x10	88,48	100	92,36	89,1	100	100	99,3	100	100	99,3	96,85
10x20	100	100	100	100	100	100	100	100	100	100	100

Πίνακας 4.1: Ποσοστό (%) χωρητικότητας εταιριών που εξυπηρετήθηκε από τα φορτηγά



Σχήμα 4.3: Σχηματική αναπαράσταση, ποσοστού (%) χωρητικότητας εταιριών που εξυπηρετήθηκε από τα φορτηγά

Μπορούμε να παρατηρήσουμε από τον Πίνακα 4.1, ότι στις περιπτώσεις 5x10 και 10x20, εξυπηρετείται κάθε φορά, το 100% της χωρητικότητας που διαθέτουν οι εταιρίες. Επομένως, μπορούμε να διαπιστώσουμε, ότι όταν τα φορτηγά είναι περισσότερα από τις εταιρίες, είναι πολύ πιο εύκολο να εκτελεστεί ολόκληρη η εργασία που έχει ανατεθεί στα φορτηγά από τις εταιρίες, όπως και είναι φυσικό. Βέβαια, το πρόβλημα για τα φορτηγά, είναι ότι και στις δύο περιπτώσεις που

αναφέρουμε, δεν χρησιμοποιούνται κάποια από αυτά (3 με 4 φορτηγά από συνολικά 10 και 20 αντίστοιχα) για την μεταφορά των παλετών. Αυτό συμβαίνει, γιατί τα φορτηγά είναι πολύ περισσότερα από τις εταιρίες, οπότε το άθροισμα της χωρητικότητας που προσφέρεται από τα φορτηγά στις εταιρίες, είναι κατά πολύ μεγαλύτερο από το άθροισμα της χωρητικότητας που χρειάζονται οι εταιρίες να εξυπηρετήσουν.

Στις άλλες δύο περιπτώσεις, όπου οι εταιρίες έχουν ίδιο πλήθος με τα φορτηγά, δηλαδή τις περιπτώσεις 5x5 και 10x10, βλέπουμε μικρές αποκλίσεις στα αποτελέσματα. Παρατηρούμε τόσο στον Πίνακα 4.1, όσο και στο γράφημα στο Σχήμα 4.3, ότι και στις δύο περιπτώσεις, οι εταιρίες εξυπηρετούνται σε πολύ μεγάλο ποσοστό από τα φορτηγά. Σε αρκετές περιπτώσεις μάλιστα, το ποσοστό αυτό φτάνει στο 100%, ενώ σε καμία από τις περιπτώσεις αυτές δεν πέφτει κάτω από 85%. Το αποτέλεσμα που παίρνουμε την κάθε φορά, εξαρτάται εξολοκλήρου, στις διαθέσιμες χωρητικότητες των φορτηγών.

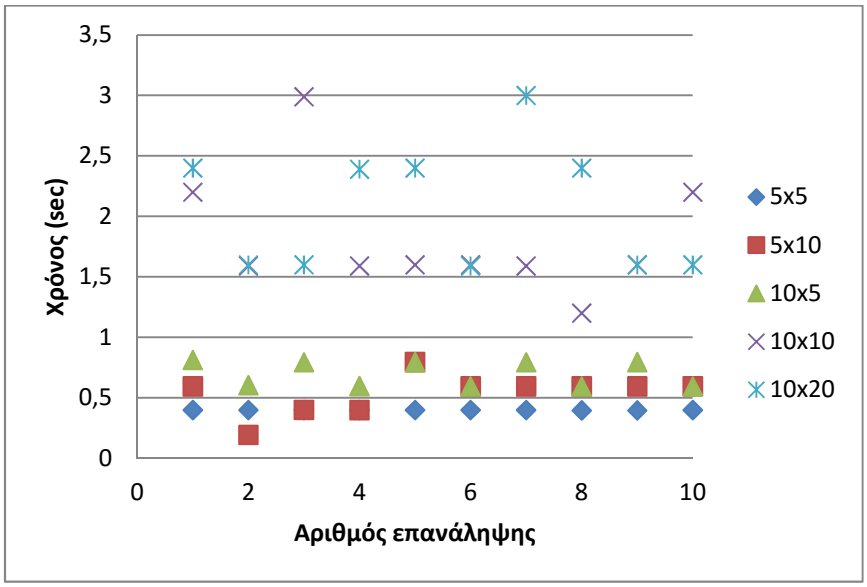
Τέλος, έχουμε την περίπτωση 10x5, όπου τα φορτηγά, είναι πολύ λιγότερα από τις εταιρίες. Σε αυτή την περίπτωση, είναι αρκετές οι εταιρίες που δεν θα εξυπηρετηθούν. Θα εξυπηρετηθούν αυτές που βρίσκονται πιο πάνω στη λίστα προτίμησης των φορτηγών. Ενώ, αυτές που βρίσκονται σε χαμηλότερη θέση, δεν θα ικανοποιήσουν καμία μονάδα χωρητικότητας. Επίσης, μπορούμε να παρατηρήσουμε, ότι σε αυτή την περίπτωση, υπάρχουν αρκετά μεγάλες αποκλίσεις στα ποσοστά της χωρητικότητας που εξυπηρετήθηκε, σε αντίθεση με τις προηγούμενες περιπτώσεις. Γεγονός που είναι λογικό εφόσον έχουμε λιγότερα φορτηγά από εταιρίες, με αρκετά μικρή χωρητικότητα για να τις εξυπηρετήσουν πλήρως.

4.4:Αποτελεσματικότητα αλγορίθμου σε σχέση με τον χρόνο εκτέλεσης

Στη συνέχεια, για τα ίδια παραδείγματα που είδαμε στην κεφάλαιο 4.3, επιστρέφουμε τον χρόνο εκτέλεσης του αλγορίθμου, όπως βλέπουμε στον Πίνακα 4.2 και στο Σχήμα 4.4.

αριθμός επανάληψης και μέσος όρος											
πλήθος χρηστών x εξυπηρετητές	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Μ.Ο.**
5x5	0,399	0,398	0,399	0,398	0,398	0,399	0,398	0,395	0,395	0,398	0,398
5x10	0,594	0,193	0,400	0,398	0,795	0,595	0,595	0,594	0,594	0,596	0,476
10x5	0,813	0,604	0,794	0,597	0,795	0,595	0,794	0,594	0,795	0,595	0,698
10x10	2,2	1,59	2,99	1,59	1,6	1,6	1,59	1,2	1,6	2,2	1,82
10x20	2,4	1,599	1,6	2,39	2,4	1,59	3	2,4	1,6	1,6	2,058

Πίνακας 4.2: Χρόνος (σε sec) που χρειάστηκε για την εκτέλεση του αλγορίθμου



Σχήμα 4.4: Σχηματική αναπαράσταση χρόνου (σε sec) που χρειάστηκε για την εκτέλεση του αλγορίθμου

Όσον αφορά τον χρόνο εκτέλεσης του αλγόριθμου, τα αποτελέσματα διαφέρουν σε σχέση με τα προηγούμενα. Σε αυτή την περίπτωση, δεν παίζει μεγάλο ρόλο αν είναι περισσότερα τα φορτηγά ή οι εταιρίες, αλλά όπως μπορούμε να δούμε από το Σχήμα 4.4, μεγαλύτερο ρόλο παίζει το άθροισμα των συμμετεχόντων (άθροισμα φορτηγών και εταιριών) γενικά.

Αρχικά, στην πρώτη περίπτωση, όπου έχουμε πέντε εταιρίες και πέντε φορτηγά, δηλαδή συνολικά δέκα συμμετέχοντες, ο χρόνος εκτέλεσης παραμένει πολύ χαμηλός, αλλά και σταθερός. Σε όλες τις εκτελέσεις, είναι μικρότερος από 0,4 δευτερόλεπτα. Συγκριτικά, μπορούμε να αναφερθούμε στο Σχήμα 4.2, όπου είχαμε μόλις δύο εταιρίες και δύο φορτηγά, δηλαδή τέσσερις συμμετέχοντες και ο χρόνος εκτέλεσης φαίνεται να είναι 0,0068 δευτερόλεπτα.

Στην περίπτωση των 15 συμμετεχόντων, έχουμε δύο περιπτώσεις, την 5x10 και την 10x5. Όπως πολύ καθαρά φαίνεται στο Σχήμα 4.4, οι χρόνοι των περιπτώσεων αυτών, κυμαίνονται στις ίδιες τιμές, καθώς και ταυτίζονται σε κάποιες

περιπτώσεις. Παρόλα αυτά, από τον Πίνακα 4.2, φαίνεται ότι γενικότερα, κατά μέσο όρο, η περίπτωση 10x5, έχει λίγο μεγαλύτερο χρόνο εκτέλεσης. Αυτό, μπορεί να οφείλεται στο ότι σε αυτή την περίπτωση, επειδή έχουμε λιγότερα φορτηγά, μπορεί να γίνονται περισσότεροι κύκλοι εκτέλεσης, επειδή η κάθε εταιρία, θα προσπαθεί να πάρει λίγη χωρητικότητα από κάποια άλλη, με βάση τη λίστα προτίμησης των φορτηγών. Εφόσον δεν θα μηδενίζεται η συνολική χωρητικότητα των εταιριών, θα καταλήξει ο αλγόριθμος να κάνει έναν επιπλέον κύκλο, όπου δεν θα γίνει καμία αλλαγή και θα οδηγήσει στον τερματισμό του.

Τέλος, οι περιπτώσεις 10x10 και 10x20, κυμαίνονται στον ίδιο χρόνο εκτέλεσης, με λίγο αυξημένο κατά μέσο όρο την περίπτωση 10x20, καθώς έχει και μεγαλύτερο πλήθος συμμετεχόντων. Αυτό που παρατηρείται όμως σε αυτές τις δύο περιπτώσεις, είναι ότι υπάρχουν μεγάλες χρονικές αποκλίσεις από τη μια εκτέλεση στην άλλη. Αυτό, μπορεί να συμβαίνει, λόγω του ότι για να ολοκληρωθεί ο αλγόριθμος με τις συγκεκριμένες τιμές δεδομένων, χρειάζεται να εκτελεστεί επιπλέον κύκλος επαναλήψεων στον αλγόριθμο.

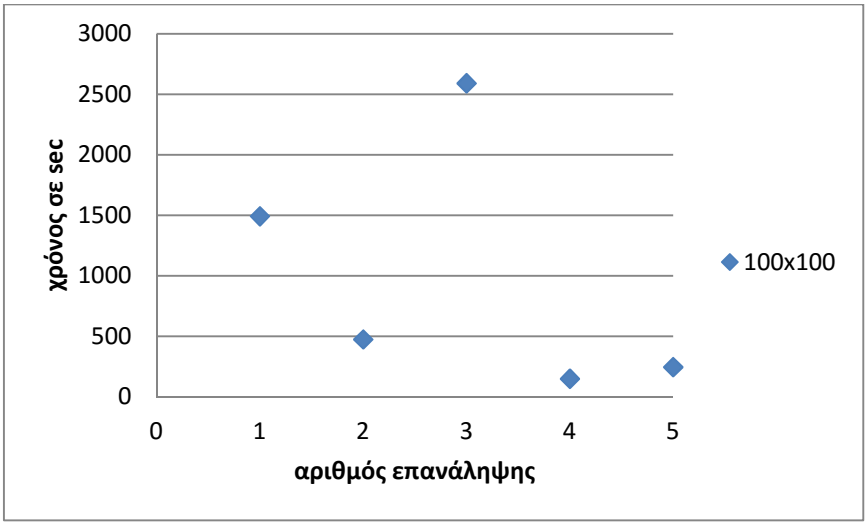
4.5: Εκτέλεση του αλγόριθμου για μεγάλο πλήθος εταιριών και φορτηγών

Για να παρατηρήσουμε την απόδοση του αλγορίθμου επιλέξαμε μεγάλο αριθμό συμμετεχόντων.

Αρχικά, εκτελέσαμε τον αλγόριθμο για εκατό εταιρίες με εκατό φορτηγά (100x100). Και αυτή την περίπτωση, την εκτελέσαμε πέντε φορές. Παρατηρούμε με τη βοήθεια του Πίνακα 4.3, ότι οι αποκλίσεις σε αυτή την περίπτωση είναι πολύ μεγάλες από τη μια εκτέλεση στην επόμενη. Αυτό συμβαίνει, γιατί όταν έχει μείνει μια εταιρία, η οποία μπορεί να μην εξυπηρετήθηκε πλήρως στον πρώτο κύκλο, ακόμη και πολύ μικρή διεργασία να έχει να κάνει, θα πρέπει να περάσει από όλες τις εταιρίες ξανά. Οπότε όταν χρειαστεί να γίνει επιπλέον κύκλος εκτέλεσης, τότε υπάρχει μεγάλη καθυστέρηση.

	αριθμός επανάληψης και μέσος όρος						
εταιρίες x φορτηγά		1	2	3	4	5	Μ.Ο.
	100x100	1491,997	474,277	2590,648	148,992	245,458	990,418

Πίνακας 4.3: Χρόνος (σε sec) που χρειάστηκε για την εκτέλεση του αλγορίθμου σε 100x100



Σχήμα 4.5: Σχηματική αναπαράσταση, χρόνου που χρειάστηκε για την εκτέλεση του αλγορίθμου σε 100x100

Βλέποντας το Σχήμα 4.4 από το προηγούμενο κεφάλαιο όπου είχαμε μικρό πλήθος κόμβων και το Σχήμα 4.5, μπορούμε να διαπιστώσουμε ότι ο χρόνος αυξάνεται απότομα σε σχέση με το πλήθος των κόμβων. Για να το διακρίνουμε καλύτερα αυτό, κάναμε διάφορα παραδείγματα, αρχικά για να δούμε πως αυξάνεται ο χρόνος εκτέλεσης από τη μια περίπτωση στην άλλη, καθώς επίσης και για να βρούμε τα όρια του αλγόριθμου.

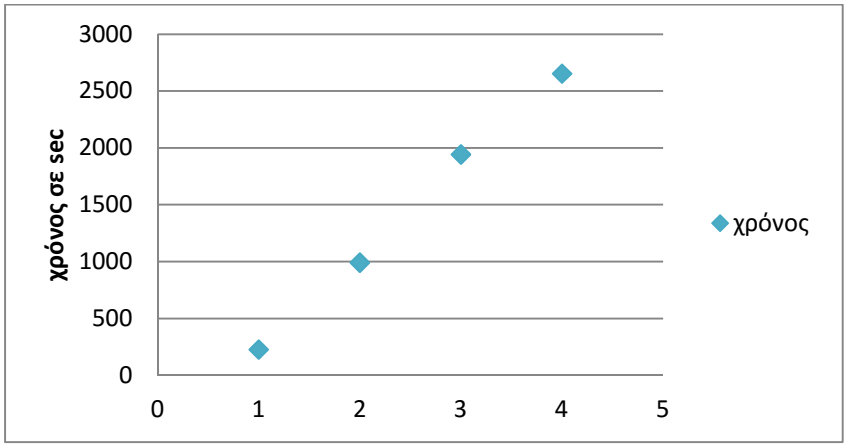
Ο αλγόριθμός μας λοιπόν, εκτελέστηκε μέχρι και το παράδειγμα 200x200, ενώ σε μια προσπάθεια εκτέλεσης 250x250, καθώς και σε 300x300, δεν κατάφερνε να τερματίσει, χωρίς ωστόσο να βγάζει κάποιο σφάλμα. Πιθανώς να χρειαζόταν πολύ περισσότερο χρόνο από αυτόν που του δόθηκε, ο οποίος βέβαια ήταν αρκετές ώρες (περίπου 6 με 8 ώρες στην κάθε περίπτωση). Παρόλα αυτά, σε ένα ρεαλιστικό πρόβλημα ίσως να μην χρειάζεται να εκτελεστεί κάποιο παράδειγμα τόσο μεγάλου πλήθους εταιριών και φορτηγών. Για παράδειγμα, σαν ένα ρεαλιστικό πρόβλημα, θα μπορούσαμε να αναφερθούμε σε μια αλυσίδα καταστημάτων, όπως είναι το ΙΚΕΑ, όπου για να διανείμει τις παραγγελίες της στους πελάτες της χρειάζεται φορτηγά. Το κάθε ένα κατάστημα της αλυσίδας αυτής, έχει συγκεκριμένο αριθμό παραγγελιών, ή παλετών για το πρόβλημά μας και από την άλλη έχει φορτηγά με συγκεκριμένη χωρητικότητα. Και παρότι η αλυσίδα εταιριών αυτή είναι αρκετά μεγάλη, δεν έχουν τεράστιο αριθμό καταστημάτων και φορτηγών που εξυπηρετούν τις παραγγελίες.

Στη συνέχεια, εκτελέσαμε τον αλγόριθμο για τα μεγέθη 50x50, 100x100, 150x150 και για 200x200 όπου πήραμε τον μέσο όρο, για τις διάφορες εκτελέσεις,

όπως φαίνεται στον Πίνακα 4.4, όπου μπορούμε να δούμε τα αποτελέσματα, καθώς και τη γραφική απεικόνιση αυτού στο Σχήμα 4.6.

εταιρίες x φορτηγά				
	50x50	100x100	150x150	200x200
χρόνος σε sec	226,17	990,418	1941,95	2652,6

Πίνακας 4.4: Χρόνος (σε sec) που χρειάστηκε για την εκτέλεση του αλγορίθμου



Σχήμα 4.6: Σχηματική αναπαράσταση χρόνου

Στο Σχήμα 4.6, παρατηρούμε, ότι ο χρόνος, αυξάνεται πολύ απότομα, όσο ανεβαίνει το πλήθος των συμμετεχόντων. Πολύ μεγάλη αύξηση βέβαια, παρατηρείται από το παράδειγμα 50x50, στο 100x100. Γενικότερα, όσο μένουμε σε χαμηλό πλήθος εταιριών και φορτηγών, οι αλλαγές στο χρόνο δεν είναι πολύ μεγάλες, σε σύγκριση με τα αποτελέσματα στο Σχήμα 4.4. Όταν όμως το πλήθος το συμμετεχόντων αυξάνεται, ο χρόνος που απαιτείται, μεγαλώνει απότομα. Γεγονός που είναι λογικό, εφόσον ο αριθμός επαναλήψεων που απαιτείται σε ένα παράδειγμα 100x100, είναι πολύ μεγαλύτερος από τον αριθμό επαναλήψεων που απαιτείται στο παράδειγμα 50x50. Οπότε καταλαβαίνουμε ότι ο χρόνος που θα χρειαστεί ο υπολογιστής για να ολοκληρώσει τον αλγόριθμο στην κάθε περίπτωση θα είναι ανάλογος.

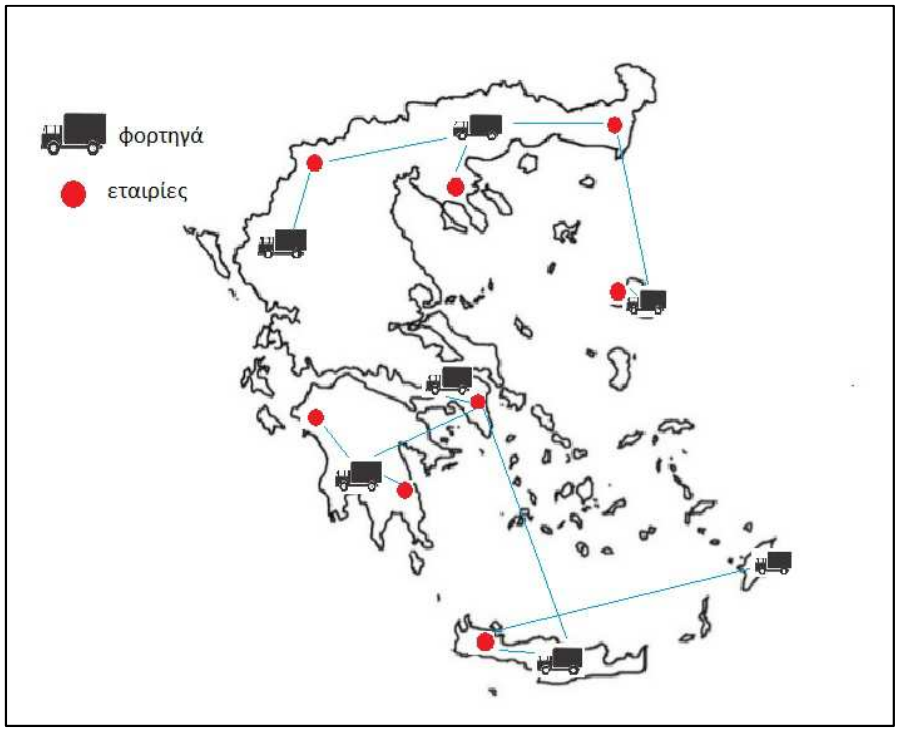
Γενικά, εκτός από τις περιπτώσεις που έχουμε πολύ λιγότερα φορτηγά από εταιρίες, παρατηρούμε ότι η χωρητικότητα των εταιριών εξυπηρετείται σε μεγάλο βαθμό και πολλές φορές πλήρως (100%). Από την άλλη, έχουμε και τον χρόνο που απαιτείται, ο οποίος παρατηρούμε ότι όσο έχουμε μικρά παραδείγματα είναι από δέκατα του δευτερολέπτου έως και 3 δευτερόλεπτα. Όταν προχωρήσαμε σε μεγαλύτερα παραδείγματα (50x50, 100x100, 150x150, 200x200), ο χρόνος αυτός

ανέβηκε και από δευτερόλεπτα έγινε αρκετά λεπτά. Αλλά αυτό δεν μας προβληματίζει ιδιαίτερα, καθώς δεν απαιτούνται συχνά τόσο μεγάλα παραδείγματα. Από την άλλη βέβαια, σε ένα πιο γρήγορο υπολογιστικό σύστημα, θα απαιτούσε λιγότερο χρόνο.

Στη συνέχεια, στο κεφάλαιο 5, θα μεταβάλλουμε λίγο το υπόδειγμα των προβλημάτων στα οποία τεστάρουμε τον αλγόριθμό μας, μεταβάλλοντας αντίστοιχα τον μηχανισμό γέννησης προβλημάτων. Πιο συγκεκριμένα, θα ελέγξουμε την αποτελεσματικότητα και τον χρόνο που κάνει ο αλγόριθμος για να βρει το βέλτιστο πάντρεμα εταιριών-φορτηγών, σε περίπτωση που έχουμε σταθερό αριθμό κόμβων (εταιριών και φορτηγών), αλλά ο αριθμός των ακμών είναι μεταβαλλόμενος. Δηλαδή, θα εξετάσουμε τον τρόπο με τον οποίο η “αραιότητα” του γράφου επηρεάζει το χρόνο εκτέλεσης του αλγόριθμου και τα αποτελέσματά του. Αυτό θα το πετύχουμε, αναγκάζοντας την κάθε εταιρία, να επιλέξει ανάμεσα σε συγκεκριμένο αριθμό φορτηγών (τα οποία όμως θα βρίσκονται στην κορυφή της λίστας προτίμησής της), και όχι σε όλα όσα είναι διαθέσιμα.

Κεφάλαιο 5: Αποτελέσματα και σύγκρισή τους σχετικά με αραιότητα του γράφου

Στο προηγούμενο κεφάλαιο είδαμε την αποτελεσματικότητα και τον χρόνο εκτέλεσης του αλγορίθμου, μεταβάλλοντας τον αριθμό των κόμβων (τον αριθμό εταιριών και φορτηγών). Σε αυτό το κεφάλαιο θα ελέγξουμε την αποτελεσματικότητα και τον χρόνο που κάνει ο αλγόριθμος για να βρει το βέλτιστο πάντρεμα εταιριών-φορτηγών. Δηλαδή, αυτή η σειρά των παραδειγμάτων θα συμπεριλαμβάνει σταθερό αριθμό κόμβων (εταιριών και φορτηγών), ενώ ο αριθμός των ακμών θα είναι μεταβαλλόμενος. Εν ολίγοις, θα εξετάσουμε τον τρόπο με τον οποίο η "αραιότητα" του γράφου επηρεάζει το χρόνο εκτέλεσης του αλγόριθμου και τα αποτελέσματά του. Για παράδειγμα, έστω ότι έχουμε 5 εταιρίες οι οποίες θα μπορούσαν να εξυπηρετηθούν από 5 φορτηγά και οι εταιρίες είχαν λίστα προτίμησης μικρότερη από το πλήθος των φορτηγών. Ένα τέτοιο σενάριο, μπορεί να είναι το απλό πρόβλημα ότι κάποιο πλήθος φορτηγών, βρίσκονται πάρα πολύ μακριά από τις εταιρίες, όπως μπορούμε να δούμε και στο Σχήμα 5.1, με αποτέλεσμα οι εταιρίες αν δεν τα προτιμούν (χαμηλά στη λίστα προτίμησης) να τα αποκλείουν τελείως σαν επιλογή, είτε λόγω απόστασης είτε λόγω κόστους. Άρα η συνδεσιμότητα του γράφου είναι πιο αραιή σε σχέση με την χειρότερη περίπτωση, όπου όλα τα φορτηγά είναι συνδεδεμένα με όλες τις εταιρίες.



Σχήμα 5.1: Παράδειγμα δικτύου εταιριών και φορτηγών στην Ελλάδα

5.1: Εκτέλεση του αλγόριθμου σε σχέση με τον αριθμό των ακμών και αποτελέσματα

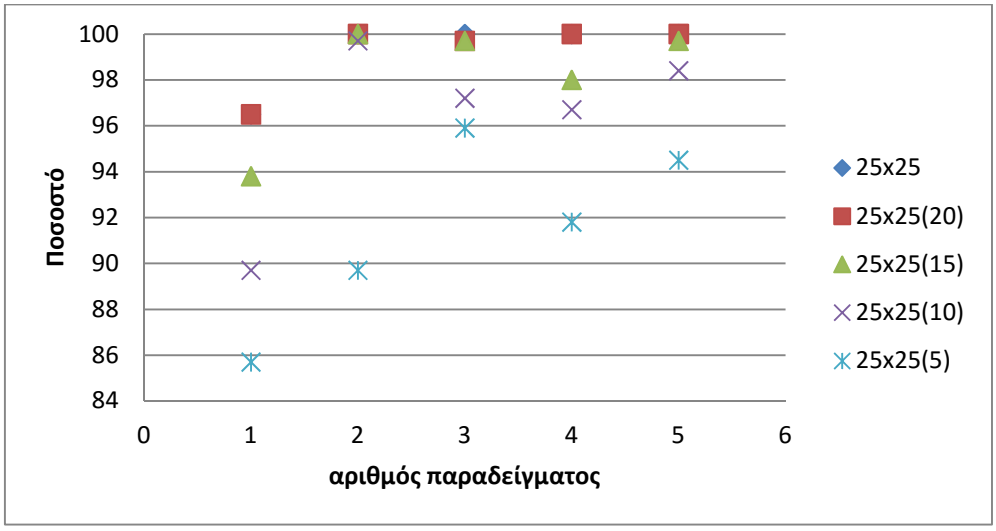
Στα παραδείγματα που θα περιγράψουμε εδώ, θα εκτελέσουμε τον αλγόριθμο για πέντε παραδείγματα 25x25 (25 εταιρίες x 25 φορτηγά). Στην αρχή, όλες οι εταιρίες, θα μπορούν να χρησιμοποιήσουν οποιαδήποτε από τα 25 φορτηγά. Στη συνέχεια, για τα ίδια παραδείγματα, θα μειώνονται οι επιλογές των εταιριών ως προς τα φορτηγά. Δηλαδή, θα μπορεί να χρησιμοποιεί η κάθε εταιρία κάποια από τα πέντε πρώτα φορτηγά στη λίστα προτίμησής της, τα δέκα πρώτα, τα δεκαπέντε και τα είκοσι πρώτα. Πρέπει να σημειωθεί ότι για κάθε παράδειγμα, η χωρητικότητα της κάθε εταιρίας καθώς και του κάθε φορτηγού παραμένει ίδια. Αυτή, μπορεί να δημιουργηθεί για το κάθε πιθανό ζεύγος σε μορφή λιστών αρχικά, και στη συνέχεια, για κάθε παράδειγμα, επιλέγονται οι συγκεκριμένες χωρητικότητες. Με αυτή τη σειρά εκτέλεσης του αλγόριθμου, θα δούμε αρχικά, πως λειτουργεί ο αλγόριθμος, με λιγότερες επιλογές, οι οποίες όμως είναι και αυτές που προτιμά η κάθε εταιρία, καθώς βρίσκονται στις πρώτες θέσεις της λίστας προτίμησής της. Στη συνέχεια, θα εξετάσουμε παρόμοια αποτελέσματα, σε σχέση με τον χρόνο εκτέλεσης του αλγορίθμου.

εταιρίες x φορτηγά	αριθμός παραδείγματος και μέσος όρος					
	1	2	3	4	5	M.O.
25x25	96,5	100	100	100	100	99,3
25x25(20)	96,5	100	99,7	100	100	99,24
25x25(15)	93,8	100	99,7	98	99,7	98,24
25x25(10)	89,7	99,7	97,2	96,7	98,4	96,34
25x25(5)	85,7	89,7	95,9	91,8	94,5	91,52

Πίνακας 5.1: Ποσοστό (%) χωρητικότητας εταιριών που εξυπηρετήθηκε από τα φορτηγά, ανάλογα με τα στοιχεία που χρησιμοποιήθηκαν από την λίστα προτίμησης

Στον Πίνακα 5.1, βλέπουμε τα αποτελέσματα από την εκτέλεση του αλγορίθμου, όπως την περιγράψαμε παραπάνω. Στην πρώτη περίπτωση, έχουμε 25 εταιρίες, οι οποίες μπορούν να εξυπηρετηθούν από οποιοδήποτε από τα 25 φορτηγά. Στην δεύτερη περίπτωση, έχουμε τις ίδιες εταιρίες, οι οποίες μπορούν να εξυπηρετηθούν μόνο από τα 20 πρώτα φορτηγά στη λίστα προτίμησής τους. Αντίστοιχα στην τρίτη περίπτωση από τα 15 πρώτα φορτηγά στη λίστα προτίμησης και ούτω κάθε εξής.

Παρατηρούμε, ότι τόσο από την πρώτη περίπτωση στη δεύτερη, υπάρχουν πολύ μικρές έως μηδενικές διαφορές. Όσο μειώνουμε τα διαθέσιμα φορτηγά για την κάθε εταιρία, το ποσοστό εξυπηρέτησης των εταιριών μειώνεται. Παρόλα αυτά, δεν μειώνεται αισθητά. Αυτό, μπορούμε καλύτερα να το δούμε στο Σχήμα 5.2. Η περίπτωση που φαίνεται να αποκλίνει, είναι αυτή κατά την οποία υπάρχουν μόνο 5 διαθέσιμα φορτηγά για την κάθε εταιρία, κάτι το οποίο φαίνεται να είναι λογικό. Γιατί η εκτέλεση του αλγόριθμου γίνεται με βάση τις προτιμήσεις των εταιριών. Έχουμε όμως προτιμήσεις και στα φορτηγά και αυτά μπορεί να μην προτιμούν τις εταιρίες από τις οποίες προτιμούνται.



Σχήμα 5.2: Σχηματική αναπαράσταση, ποσοστού (%) χωρητικότητας εταιριών που εξυπηρετήθηκε από τα φορτηγά, ανάλογα με τα στοιχεία που χρησιμοποιήθηκαν από την λίστα προτίμησης

Έχοντας λοιπόν τα αποτελέσματα που προκύπτουν ως προς τα ποσοστά εξυπηρέτησης των εταιριών από τα μηχανήματα, μένει να δούμε και τον χρόνο εκτέλεσης του αλγορίθμου για τα συγκεκριμένα παραδείγματα. Με αυτό τον τρόπο θα έχουμε μια πιο πλήρη εικόνα, για να μπορούμε να βγάλουμε τα συμπεράσματά μας.

5.2: Αποτελεσματικότητα αλγορίθμου σε σχέση με τον χρόνο εκτέλεσης

Στον Πίνακα 5.2, μπορούμε να δούμε τον χρόνο που χρειάστηκε σε δευτερόλεπτα, για την εκτέλεση του αλγορίθμου, ανάλογα με τα στοιχεία που χρησιμοποιήθηκαν από την λίστα προτίμησης. Αντίθετα με τα αποτελέσματα που

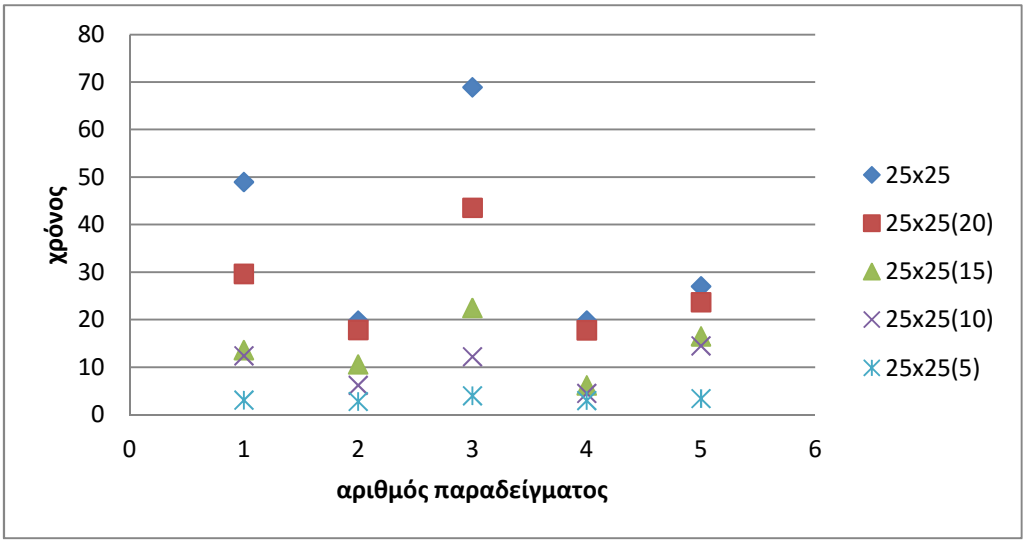
πήραμε όσον αφορά το ποσοστό εξυπηρέτησης των εταιριών από τα φορτηγά, εδώ, βλέπουμε πολύ μεγάλες αποκλίσεις από τη μια περίπτωση στην άλλη.

	αριθμός παραδείγματος και μέσος όρος					
	1	2	3	4	5	M.O.
25x25	48,99	19,85	68,94	19,86	27,04	36,94
25x25(20)	29,68	17,86	43,58	17,8	23,71	25,53
25x25(15)	13,64	10,63	22,5	6,21	16,54	13,9
25x25(10)	12,44	6,22	12,24	4,52	14,53	9,99
25x25(5)	3,1	2,82	4,02	3,01	3,42	3,27

φορτηγά x εταιρίες

Πίνακας 5.2: Χρόνος (σε sec) που χρειάστηκε για την εκτέλεση του αλγορίθμου, ανάλογα με τα στοιχεία που χρησιμοποιήθηκαν από την λίστα προτίμησης

Στο Σχήμα 5.3, μπορούμε να δούμε ακόμη καλύτερα τα αποτελέσματα του χρόνου εκτέλεσης. Βλέπουμε ότι σε κάποιες περιπτώσεις, ο χρόνος μπορεί να είναι ακόμη και ο μισός από την μια περίπτωση στην άλλη. Κι ενώ περιμέναμε να είναι μικρότερος ο χρόνος εκτέλεσης, εφόσον έχουμε λιγότερα φορτηγά να διαχειριστούμε για την κάθε εταιρία, δεν περιμέναμε αυτή η απόκλιση να είναι τόσο μεγάλη.



Σχήμα 5.3: Σχηματική αναπαράσταση, χρόνου (σε sec) που χρειάστηκε για την εκτέλεση του αλγορίθμου, ανάλογα με τα στοιχεία που χρησιμοποιήθηκαν από την λίστα προτίμησης

5.3: Συμπεράσματα

Παρατηρώντας τα παραπάνω αποτελέσματα, μπορούμε να βγάλουμε κάποια συμπεράσματα. Αρχικά, καταλαβαίνουμε, ότι στην περίπτωση που έχουμε πολύ μεγάλο αριθμό εταιριών και φορτηγών, για να ελαττώσουμε το χρόνο εκτέλεσης του αλγορίθμου, μπορούμε να επιτρέψουμε στους συμμετέχοντες να εισάγουν

συγκεκριμένο αριθμό φορτηγών στη λίστα προτίμησης. Κάτι τέτοιο είναι κυρίως χρήσιμο σε περιπτώσεις που έχουμε πολύ μεγάλο πλήθος εταιριών και φορτηγών, όπως στο κεφάλαιο 4.5. Σε τέτοιες περιπτώσεις θα μπορούσαμε να περιορίζουμε τον αριθμό φορτηγών από τη λίστα προτίμησης που θα μπορούσε να χρησιμοποιήσει η κάθε εταιρία, ώστε να χρειαζόμαστε πολύ μικρότερο χρόνο εκτέλεσης.

Κεφάλαιο 6: Επίλογος

Σε αυτή την εργασία παρουσιάσαμε μια οικογένεια προβλημάτων που πηγάζουν από τη θεωρία του ευσταθούς ταιριάσματος. Πιο συγκεκριμένα, αναφερθήκαμε στο πρόβλημα ευσταθούς κατανομής καθώς και ευσταθούς ροής, αλλά και στα προβλήματα ευσταθούς ταιριάσματος ένα-προς-ένα, ένα-προς-πολλά και πολλά-προς-πολλά.

Για να επιλύσουμε το πρόβλημα που προτείναμε, βασιστήκαμε στη θεωρία του προβλήματος ακεραίας ευσταθούς κατανομής. Στα πλαίσια αυτά προτείναμε και υλοποιήσαμε έναν αλγόριθμο βελτιστοποίησης, ο οποίος γενικεύει τον GS αλγόριθμο.

6.1: Η δική μας πρόταση

Η διατριβή αυτή πραγματεύτηκε το πρόβλημα ακεραίας ευσταθούς κατανομής, εξετάζοντας την εφαρμογή του στον τομέα της μεταφοράς φορτίων από ένα στόλο φορτηγών και οχημάτων. Πιο συγκεκριμένα, το πρόβλημα περιλαμβάνει ένα σύνολο από εταιρίες που κάθε μία έχει έναν αριθμό παλετών που πρέπει να μεταφερθούν. Από την άλλη πλευρά, υπάρχει ένας αριθμός φορτηγών, κάθε ένα από τα οποία, μπορεί να χωρέσει συγκεκριμένο αριθμό παλετών. Οι εταιρίες έχουν συγκεκριμένες προτιμήσεις για τα φορτία τους στα διαθέσιμα φορτηγά βασιζόμενες στα χαρακτηριστικά του καθενός και τα φορτηγά έχουν προτιμήσεις στα φορτία. Οι προτιμήσεις αυτές εκφράζονται με τη χρήση λιστών προτίμησης, οι οποίες μοντελοποιούν ένα διαισθητικό κριτήριο για τις προτιμήσεις του κάθε συμμετέχοντα. Το πρόβλημα λοιπόν που επιλύσαμε, είναι πως θα μεταφερθούν οι παλέτες που έχουμε με ευσταθή τρόπο.

Για την επίλυση του προβλήματός μας χρησιμοποιήσαμε τον GS (Gale-Shapley) αλγόριθμο, τον οποίο τον επεκτείναμε και τον διαφοροποιήσαμε ώστε να ανταπεξέρχεται στις απαιτήσεις μας. Παράλληλα, υλοποιήσαμε και έναν μηχανισμό γέννησης προβλημάτων, ο οποίος, δημιουργεί τυχαίες τιμές δεδομένων, από ένα δεδομένο εύρος τιμών που έχουμε ορίσει, δίνοντάς μας έτσι τη δυνατότητα να δημιουργήσουμε μια γκάμα από διαφορετικά σενάρια εκτέλεσης για τον αλγόριθμό μας.

Για να ελέγξουμε λοιπόν τον αλγόριθμο που δημιουργήσαμε, τον δοκιμάσαμε σε διαφορετικά σενάρια, διαφοροποιώντας τον αριθμό των συμμετεχόντων εταιριών και φορτηγών αλλά και τον αριθμό των αντίστοιχων τέτοιων ζευγαριών. Με αυτό τον τρόπο είδαμε, για τις διάφορες τιμές εταιριών και φορτηγών, κατά πόσο μπορούμε να έχουμε τα επιθυμητά αποτελέσματα. Δηλαδή, αν όλες οι εταιρίες εξυπηρετούνται πλήρως από τα διαθέσιμα φορτηγά. Αυτό φυσικά, δεν είναι πάντα εφικτό, καθώς για να επιτευχθεί, θα πρέπει η χωρητικότητα που διαθέτουν όλα τα φορτηγά συνολικά, να είναι ίση ή μεγαλύτερη από τη συνολική χωρητικότητα που θέλουν να δώσουν οι εταιρίες.

Κάτι τέτοιο, το παρατηρήσαμε στο κεφάλαιο 4.3, στο παράδειγμα που χρησιμοποιήσαμε δέκα εταιρίες με πέντε φορτηγά (10x5). Σε αυτή την περίπτωση, ο αριθμός των φορτηγών, είναι ο μισός από τον αριθμό των εταιριών. Οπότε ο αριθμός της χωρητικότητας που θα έχουν διαθέσιμη, θα είναι μικρός για να εξυπηρετήσουν τις διπλάσιες εταιρίες. Οπότε, όπως ήταν φυσικό, έμεινε μεγάλος αριθμός από την χωρητικότητα των εταιριών που δεν εξυπηρετήθηκε. Αντίθετα, σε περιπτώσεις που ο αριθμός των φορτηγών ήταν ίσος ή και μεγαλύτερος από τον αριθμό των εταιριών, η χωρητικότητα των εταιριών εξυπηρετήθηκε πλήρως ή σε πολύ μεγάλο βαθμό.

Όσον αφορά τον χρόνο εκτέλεσης του αλγορίθμου όμως, όπως βλέπουμε στην κεφάλαιο 4.4, δεν έχει σημασία αν έχουμε περισσότερα φορτηγά ή εταιρίες. Περισσότερο ρόλο σε αυτή την περίπτωση παίζει το πλήθος των συμμετεχόντων. Δηλαδή το άθροισμα του πλήθους των φορτηγών με το πλήθος των εταιριών.

6.2: Γενικά συμπεράσματα

Με τα αποτελέσματα που προέκυψαν από την εκτέλεση του αλγορίθμου μας, βγάλαμε συμπεράσματα σχετικά με τον χρόνο που απαιτείται για την εκτέλεσή του, καθώς και για την αποτελεσματικότητά του.

Όπως παρατηρήσαμε από τον Πίνακα 4.1 του κεφαλαίου 4.3 όπου παίρνουμε τα πρώτα αποτελέσματα, ο αλγόριθμος είναι αποτελεσματικός και οι εργασίες εκτελούνται πλήρως ή σε πολύ μεγάλο ποσοστό. Φυσικά όταν τα φορτηγά είναι περισσότερα από τις εταιρίες, είναι πολύ πιο εύκολο να εκτελεστεί ολόκληρη η εργασία που έχει ανατεθεί στα φορτηγά από τις εταιρίες. Στην περίπτωση, όπου οι εταιρίες έχουν ίδιο πλήθος με τα φορτηγά, παίζει μεγάλο ρόλο η χωρητικότητα που έχει δοθεί στα φορτηγά. Παρατηρούμε όμως, ότι και στις δύο περιπτώσεις, οι εταιρίες

εξυπηρετούνται πλήρως ή σε πολύ μεγάλο ποσοστό από τα φορτηγά. Το αποτέλεσμα που παίρνουμε την κάθε φορά, εξαρτάται εξολοκλήρου, στις διαθέσιμες χωρητικότητες των φορτηγών. Από την άλλη, έχουμε την περίπτωση όπου τα φορτηγά, είναι πολύ λιγότερα από τις εταιρίες. Σε αυτή, είναι αρκετές οι εταιρίες που δεν θα εξυπηρετηθούν, ενώ θα εξυπηρετηθούν αυτές που βρίσκονται πιο πάνω στη λίστα προτίμησης των φορτηγών. Αυτές που βρίσκονται σε χαμηλότερη θέση, είναι πιθανό να μην ικανοποιήσουν καμία μονάδα χωρητικότητας. Επίσης, μπορούμε να παρατηρήσουμε, ότι σε αυτή την περίπτωση, υπάρχουν αρκετά μεγάλες αποκλίσεις στα ποσοστά της χωρητικότητας που εξυπηρετήθηκε, σε αντίθεση με τις προηγούμενες περιπτώσεις. Γεγονός που είναι λογικό εφόσον έχουμε λιγότερα φορτηγά από εταιρίες, με αρκετά μικρή χωρητικότητα για να τις εξυπηρετήσουν πλήρως.

Όσον αφορά τον χρόνο εκτέλεσης του αλγόριθμου, όπως βλέπουμε στο κεφάλαιο 4.4, τα αποτελέσματα διαφέρουν σε σχέση με τα παραπάνω. Σε αυτή την περίπτωση, δεν παίζει μεγάλο ρόλο αν είναι περισσότερα τα φορτηγά ή οι εταιρίες, αλλά όπως μπορούμε να δούμε από το Σχήμα 4.4, μεγαλύτερο ρόλο παίζει το άθροισμα των συμμετεχόντων (άθροισμα φορτηγών και εταιριών) γενικά. Σύμφωνα με τα αποτελέσματα που πήραμε οπότε, συμπεραίνουμε ότι ο αλγόριθμος που υλοποιήσαμε έχει πολύ μικρό χρόνο εκτέλεσης σε μικρά προβλήματα. Όταν όμως το πλήθος των συμμετεχόντων έγινε πολύ μεγάλο, αυξήθηκε και ο χρόνος κατά πολύ. Ειδικά σε περιπτώσεις του τύπου 200x200. Αλλά αυτό δεν μας προβληματίζει ιδιαίτερα, καθώς σε πραγματικές συνθήκες δεν απαιτούνται τόσο μεγάλες τιμές για αντίστοιχα παραδείγματα.

Επίσης, αυτό που παρατηρήσαμε παίρνοντας τα πρώτα αποτελέσματα του αλγορίθμου μας, είναι ότι μπορεί να απεικονίσει εύκολα τα δεδομένα, ώστε να μπορεί να το χειριστεί κάποιος εύκολα σε μια εταιρία.

Στη συνέχεια, μεταβάλλαμε λίγο τον αλγόριθμό μας και ελέγξαμε την αποτελεσματικότητα και τον χρόνο που κάνει ο αλγόριθμος για να βρει το βέλτιστο πάντρεμα εταιριών-φορτηγών, σε περίπτωση που έχουμε σταθερό αριθμό κόμβων (εταιριών και φορτηγών), αλλά ο αριθμός των ακμών είναι μεταβαλλόμενος. Δηλαδή, εξετάσαμε τον τρόπο με τον οποίο η “αραιότητα” του γράφου επηρεάζει το χρόνο εκτέλεσης του αλγόριθμου και τα αποτελέσματά του. Αυτό το πετύχαμε, αναγκάζοντας την κάθε εταιρία, να επιλέξει ανάμεσα σε συγκεκριμένο αριθμό

φορτηγών (τα οποία όμως βρίσκονται στην κορυφή της λίστας προτίμησής της), και όχι σε όλα όσα είναι διαθέσιμα.

Όπως παραπάνω, τα συμπεράσματά μας είναι αντίστοιχα. Δηλαδή και πάλι, ο χρόνος που απαιτείται σε μικρά παραδείγματα είναι πολύ μικρός και μεγαλώνει όσο ανεβαίνει ο αριθμός των συμμετεχόντων. Στην προκειμένη περίπτωση βέβαια, στους συμμετέχοντες δεν παίρνουμε όλο τον αριθμό των φορτηγών, αλλά τον αριθμό των φορτηγών που επιτρέπουμε σε κάθε εταιρία να επιλέξει. Επίσης σε αυτή την περίπτωση, η αποτελεσματικότητα του αλγορίθμου μας, εξαρτάται από την χωρητικότητα που διαθέτει το κάθε φορτηγό, καθώς και το πόσες εταιρίες προτιμούν το κάθε φορτηγό. Θα πρέπει να σημειωθεί ότι ο περιορισμός του αριθμού των επιλογών που μπορούν να εισαχθούν στη λίστα προτίμησης του κάθε συμμετέχοντα είναι μία μέθοδος που χρησιμοποιείται σε υλοποιημένους πραγματικούς μηχανισμούς αντιστοίχισης ώστε να βελτιώνει το χρόνο εκτέλεσης του σχετικού αλγορίθμου. Ένα τέτοιο παράδειγμα αποτελεί το πρόβλημα της εισαγωγής φοιτητών στα πανεπιστήμια, όπου οι φοιτητές έχουν ένα περιορισμένο αριθμό σχολών (πχ. 80 σχολές) τις οποίες μπορούν να εισάγουν στο μηχανογραφικό τους.

6.3: Μελλοντική εργασία

Αρχικά, σε μελλοντική εργασία θα θέλαμε να συγκρίνουμε τον αλγόριθμο που υλοποιήσαμε με αυτόν των Dean & Munshi [10] που χρησιμοποιεί το γράφο G(x), ώστε να δούμε ποιος είναι πιο γρήγορος στην επίλυση του προβλήματος αυτού.

Επίσης, σε επόμενη εργασία, θα μπορούσαμε να εμπλουτίσουμε τον υπάρχοντα αλγόριθμο. Να προσθέσουμε επιπλέον παραμέτρους, όπως καθυστερήσεις φορτηγών λόγω κίνησης ή απόστασης από την εταιρία, σφάλματα και άλλα, ώστε να μπορεί να επιλύνει πιο σύνθετες μορφές του προβλήματος. Με αυτόν τον τρόπο, το σενάριο που θα εκτελεστεί θα μας δίνει πιο ρεαλιστικά αποτελέσματα. Μια άλλη πιθανή επέκταση είναι να προσθέσουμε τη δυνατότητα αποκάλυψης περιστροφών, επεκτείνοντας τον αλγόριθμο που ήδη έχουμε δημιουργήσει. Με αυτόν τον τρόπο, θα μπορέσουμε να δημιουργήσουμε έναν αλγόριθμο που παρέχει μια λύση που να είναι πιο δίκαιη. Άλλη μια μεταβολή που θα μπορούσαμε να κάνουμε σε αυτό τον αλγόριθμο, είναι για να μειώσουμε τον χρόνο εκτέλεσής του. Αυτό θα μπορούσαμε να το κάνουμε τεμαχίζοντας τις λίστες προτίμησης. Να τις χωρίσουμε δηλαδή ανάλογα με την περιοχή στην οποία βρίσκονται. Με αυτό τον τρόπο δεν θα

χρειάζεται να προσπελαστούν από την κάθε εταιρία όλα τα φορτηγά, αλλά μόνο αυτά που βρίσκονται σε περιοχές που προτιμούνται και να προχωράμε σε άλλες περιοχές μόνο στην περίπτωση που απαιτείται. Έτσι, θα μπορούσε να μειωθεί πολύ ο χρόνος εκτέλεσης, κυρίως σε παραδείγματα με πολλούς συμμετέχοντες.

Επιπλέον, παρατηρούμε στον αλγόριθμο που υλοποιήσαμε ότι χρησιμοποιούμε κυρίως τις λίστες προτίμησης, χωρίς να έχουμε στοιχεία για το που βρίσκεται το κάθε φορτηγό (χιλιομετρική απόσταση από την εταιρία), ώστε να μπορούμε να προβλέψουμε καθυστερήσεις. Μια τέτοια εργασία θα μπορούσε να γίνει μελλοντικά, ώστε να ανταποκρίνεται περισσότερο το αποτέλεσμά μας σε πραγματικές συνθήκες.

Ένα από τα βασικά πλεονεκτήματα αυτής της εφαρμογής είναι ότι θα μπορούσαμε να δημιουργήσουμε ένα εντελώς νέο σενάριο και να χρησιμοποιήσουμε τον αλγόριθμο που δημιουργήσαμε στην παρούσα εργασία, για την επίλυση ενός εντελώς διαφορετικού προβλήματος. Στα πλαίσια αυτά θα πρέπει να σημειωθεί ότι η υλοποίηση έχει γίνει με τέτοιο τρόπο ώστε να μπορεί να χρησιμοποιηθεί για πολλές διαφορετικές περιπτώσεις ακέραιης ευσταθούς κατανομής, με εντελώς διαφορετικά δεδομένα. Για παράδειγμα, στη θέση των φορτηγών, θα μπορούσαν να είναι εργαζόμενοι, οι οποίοι προσφέρουν ώρες εργασίας σε εταιρίες, ως εξωτερικοί συνεργάτες. Άρα, η εφαρμογή των αποτελεσμάτων αυτής της εργασίας σε κάποιο άλλο χώρο, πέραν αυτού της μεταφοράς φορτίων, θα μπορούσε να είναι άλλος ένας τομέας μελλοντικής εργασίας.

Ένα ακόμη πλεονέκτημα της προσέγγισής μας προκύπτει από το γεγονός ότι οποιοδήποτε πρόβλημα ευσταθούς ροής μπορεί να μετατραπεί σε πρόβλημα ευσταθούς κατανομής (κεφάλαιο 2.3). Αυτό σημαίνει ότι μπορούμε να χρησιμοποιήσουμε τους αλγορίθμους που αναπτύξαμε ή θα αναπτύξουμε για το πρόβλημα ευσταθούς ροής για να αντιμετωπίσουμε προβλήματα που φαινομενικά παρουσιάζονται πιο περίπλοκα. Πιο συγκεκριμένα, μπορούμε να μοντελοποιήσουμε προβλήματα που δεν έχουν απαραίτητα τη μορφή της διμερούς αγοράς (όπως είναι αυτή που μελετήσαμε στην διατριβή αυτή, δηλαδή φορτία - οχήματα) αλλά περιλαμβάνουν πιο ιδιαίτερες μορφές δικτύου, όπως πχ. ενδιάμεσους μεταφορείς, μεταγωγικούς ή μεταφορτωτικούς κόμβους, κλπ. Επομένως, πιθανή μελλοντική εργασία αποτελεί και η υλοποίηση του σχετικού αλγορίθμου μετατροπής που παρέχεται από τον Fleiner [9] και η ενσωμάτωσή του στον υπάρχοντα αλγόριθμο, που

θα μας δώσει τη δυνατότητα να αντιμετωπίζουμε και να λύνουμε μια ακόμη πιο ευρεία γκάμα προβλημάτων.

Βιβλιογραφία

[1] D. Gale, L. S.Shapley. College Admissions and the Stability of Marriage. *The American Mathematical Monthly.* January 1962, Vol. 69, 1, pp. 9-15.

[2] R.W. Irving, P.Leather, and D.Gusfield. An efficient algorithm for the “optimal” stable marriage. *Journal of the ACM.* 1987, Vol. 34, 3, pp. 532-534.

[3] V. Bansal, A. Agrawal, and V. S. Malhotra, "Polynomial time algorithm for an optimal stable assignment with multiple partners.". *Theoretical Computer Science.* 2007, Vol. 379, 3, pp. 317-328.

[4] A. Roth, and M. O. Sotomayor. Two-sided matching: A study in game-theoretic modeling and analysis. *Cambridge University Press.* 1992, 18.

[5] P. Eirinakis, D. Magos, I. Mourtos, and P. Miliotis. Finding all stable pairs and solutions to the many-to-many stable matching problem. *INFORMS Journal on Computing.* 2012, Vol. 2, 24, pp. 245-259.

[6] M. Baïou, M. Balinski. The Stable Allocation (or Ordinal Transportation) Problem. *Mathematics of Operations Research.* August 2002, Vol. 27, 3, pp. 485–503.

[7] M. Balinski, G. Ratier. Of stable marriages and graphs, and strategy and polytopes. *SIAM Review.* 1997, 39, pp. 575–604.

[8] R. W. Irving. Stable marriage and indifference. *Discrete Applied Mathematics.* 1994, Vol. 3, 48, pp. 261-272.

[9] T. Fleiner, "On stable matchings and flows," in *In Proceedings of the 36th International Workshop on Graph-Theoretic Concepts in Computer Science*, Berlin, 2010.

[10] B.C. Dean, S. Munshi. Faster Algorithms For Stable Allocation Problems. *Algorithmica.* September 2010, Vol. 58, 1, pp. 59-81.

[11] Nobel Foundation, «http://www.nobelprize.org/nobel_prizes/economic-sciences/laureates/2012/,» 2012. [Ηλεκτρονικό].

[12] A.E. Roth. "The evolution of the labor market for medical interns and residents: a case study in game theory.". *The Journal of Political Economy.* 1984, pp. 991-1016.

[13] A.E. Roth, and E. Peranson. The redesign of the matching market for American physicians: Some engineering aspects of economic design. *National bureau of economic research.* 1999, w6963.

[14] A. Abdulkadiroğlu, P.A. Pathak, and A. E. Roth. "The new york city high school match.".

American Economic Review. 2005, pp. 364-367.

[15] A. Abdulkadiroğlu, P.A. Pathak, A.E. Roth, and T. Sönmez. The Boston public school match. *American Economic Review.* 2005, pp. 368-371.

[16] P. Eirinakis, D. Magos, I. Mourtos. Linear and Integer Programming formulations for Stable Allocations and House Allocations.

[17] R. W. Irving, P. Leather, and D. Gusfield. An efficient algorithm for the "optimal" stable marriage. *Journal of the ACM (JACM).* 1987, Vol. 3, 34, pp. 532-543.

[18] J.P. Picard. "Maximal closure of a graph and applications to combinatorial problems.". *Management Science.* 1976, pp. 1268-1272.

[19] S. Munshi. Faster algorithms for stable allocation problems. *PhD Thesis. Clemson University* 2007.

[20] P. Biró, and T. Fleiner. The integral stable allocation problem on graphs. *Discrete Optimization.* 2010, Vol. 1, 7, pp. 64-73.

[21] A. Alkan, and D. Gale. Stable schedule matching under revealed preference. *Journal of Economic Theory.* 2003, Vol. 2, 112, pp. 289-306.

[22] T. Fleiner. On the stable b-matching polytope. *Math. Social Sci.* 2003, Vol. 2, 46, pp. 149–158.

[23] K. Cechlárová, and T. Fleiner. On a generalization of the stable roommates problem. *ACM Transactions on Algorithms (TALG).* 2005, Vol. 1, 1, pp. 143-156.

[24] R.W. Irving, S. Scott. The stable fixtures problem. *Discrete Applied Mathematics.* 2007, 155, pp. 2118–2129.

[25] D. Gusfield, R. W. Irving. The stable marriage problem: structure and algorithms. *Foundations of Computing Series.* MIT Press, Cambridge, MA, 1989.

[26] B.C. Dean, N. Swar. The Generalized Stable Allocation Problem. *WALCOM '09: the 3rd International Workshop o-n Algorithms and Computation, Lecture Notes in Computer Science.* 2009, Vol. 5431 (Springer), pp. 238–249.

Παράρτημα

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace WindowsFormsApplication3
{
    public partial class Form3 : Form
    {
        private int countCompanies, countTrucks;
        private Random rnd = new Random();
        private List<Create_Agent> Companies = null;
        private List<Create_Agent> Trucks = null;
        public Form3()
        {
            InitializeComponent();
        }

        private void button2_Click(object sender, EventArgs e)
        {
            String val1, val2;
            val1 = txtJobs.Text;
            val2 = txtMachines.Text;
            Int32.TryParse(val1, out countCompanies);
            Int32.TryParse(val2, out countTrucks);

            if (countCompanies > 0 && countTrucks > 0)
            {
                Trucks = new List<Create_Agent>();
                Companies = new List<Create_Agent>();

                Random rnd = new Random();
                Create_Agent compObj = null, truckObj = null;

     // Step 1 Create trucks and companies from user input
                for (int i = 0; i < countTrucks; i++)
                {
                    truckObj = new Create_Agent(i + 1);
                    Trucks.Add(truckObj);
                }
                for (int k = 0; k < countCompanies; k++)
                {
                    compObj = new Create_Agent(k + 1);
                    Companies.Add(compObj);
                }

    //Step 2 Assign trucks to companies and the opposite and set Capacity (int)
                foreach (Create_Agent compItem in Companies)
                {
                    for (int i = 0; i < Trucks.Count; i++)
                    {

                        compItem.Prefs.Add(Trucks[i]);
                    }
```

```
                    compItem.Capacity = rnd.Next(10, 21);
                }

                foreach (Create_Agent truckItem in Trucks)
                {
                    for (int j = 0; j < Companies.Count; j++)
                    {
                        truckItem.Prefs.Add(Companies[j]);
                    }
                    truckItem.Capacity = rnd.Next(10, 21);
                }

  //Step 3 Randomize assigns preferencies
                foreach (Create_Agent compItem in Companies)
                {
                    compItem.Prefs.Shuffle();
                    Console.WriteLine("Company with id= " + compItem.id + "
and capacity=  " + compItem.Capacity);
                    foreach (Create_Agent pref in compItem.Prefs)

                        Console.WriteLine(" has preference truck= " + pref.id);
                }
                foreach (Create_Agent truckItem in Trucks)
                {
                    truckItem.Prefs.Shuffle();

                    Console.WriteLine("Truck with id= " + truckItem.id + " and
capacity=  " + truckItem.Capacity);
                    foreach (Create_Agent pref in truckItem.Prefs)
                        Console.WriteLine(" has preference company= " +
pref.id);
                }
                int[,] minValues = CreateTable();

                CreateTableM(minValues);
            }
        }

        private void CreateTableM(int[,] minValues)
        {
            DateTime start = DateTime.Now;
            DateTime stop;
            int[,] tableM = new int[countCompanies, countTrucks];
            int s;
            bool flag;
            do
            {
                flag = false;
                for (int i = 0; i < countCompanies; i++)
                {
                    Console.WriteLine(String.Format("Loop {0}: ", i + 1));
                    for (int j = 0; j < countTrucks; j++)
                    {
                        int tr = Companies[i].Prefs[j].id - 1;
                        if (minValues[i, tr] < Companies[i].Capacity &&
minValues[i, tr] < Trucks[tr].Capacity && tableM[i, tr] < minValues[i, tr])
                        {
                            int value = minValues[i, tr] - tableM[i, tr];
                            Companies[i].Capacity -= value;
                            Trucks[tr].Capacity -= value;
                            tableM[i, tr] += value;
                            if (value != 0) flag = true;
```

```
                        }
                        else if (Companies[i].Capacity > 0 && tableM[i, tr] <
minValues[i, tr])
                        {
                            int temp1 = minValues[i, tr] - tableM[i, tr];
                            int temp = Math.Min(temp1, Companies[i].Capacity);
                            int value = Math.Min(temp, Trucks[tr].Capacity);
                            Companies[i].Capacity -= value;
                            Trucks[tr].Capacity -= value;
                            tableM[i, tr] += value;
                            if (value != 0) flag = true;
                            if (Companies[i].Capacity > 0 &&
Trucks[tr].Prefs.IndexOf(Companies[i]) + 1 < countCompanies)
                            {
                                int pos =
Trucks[tr].Prefs.IndexOf(Companies[i]) + 1;
                                for (int b = countCompanies - 1; b >= pos; b--)
                                {
                                    int com = Trucks[tr].Prefs[b].id - 1;
                                    if (Companies[i].Capacity > 0 && tableM[i,
tr] < minValues[i, tr])
                                    {
                                        int fincap, tempcap;
                                        int temp2 = minValues[i, tr] -
tableM[i, tr];
                                        tempcap = Math.Min(temp2, tableM[com,
tr]);
                                        fincap = Math.Min(tempcap,
Companies[i].Capacity);
                                        tableM[i, tr] = tableM[i, tr] + fincap;
                                        tableM[com, tr] = tableM[com, tr] -
fincap;
                                        Companies[i].Capacity -= fincap;
                                        Companies[com].Capacity += fincap;
                                        if (fincap != 0) flag = true;
                                    }
                                }
                            }
                        } Console.Write("tableM[" + (i+1)  + ", " + (tr+1)  + "
] = "  + tableM[i, tr] + " ");
                    } Console.WriteLine();

                }
                s = 0;
                for (int i = 0; i < Companies.Count; i++)
                {
                    s += Companies[i].Capacity;
                }
            } while (flag == true && s > 0);

            Console.Write("table M final:");
            Console.WriteLine();
            for (int i = 0; i < countCompanies; i++)
            {
                for (int j = 0; j < countTrucks; j++)
                {
                    Console.Write(tableM[i, j] + " ");
                } Console.WriteLine();
            }
            stop = DateTime.Now;
```

```
            Console.WriteLine(String.Format("Total duration (sec): {0}", (stop
- start).TotalSeconds));
        }

        private int[,] CreateTable()
        {
            int[,] tableMinValues = new int[countCompanies, countTrucks];
             Console.Write("minvalues: ");
             Console.WriteLine();

            for (int i = 0; i < countCompanies; i++)
            {
                for (int j = 0; j < countTrucks; j++)
                {
                    if (Trucks[j].Capacity < Companies[i].Capacity)
                    {
                        tableMinValues[i, j] = rnd.Next(Trucks[j].Capacity - 5,
Trucks[j].Capacity + 1);
                        if (tableMinValues[i, j] < 0) { tableMinValues[i, j] =
0; }
                    }
                    else
                    {
                        tableMinValues[i, j] = rnd.Next(Companies[i].Capacity -
5, Companies[i].Capacity + 1);
                    }

                    Console.Write(tableMinValues[i, j] + " ");
                }
                Console.WriteLine();

            }
            return tableMinValues;
        }

        private void label2_Click(object sender, EventArgs e)
        {

        }

        private void Form3_Load(object sender, EventArgs e)
        {

        }
    }
}
```