

**Δημοκρίτειο Πανεπιστήμιο Θράκης Πολυτεχνική Σχολή**

**Τμήμα Μηχανικών Παραγωγής και Διοίκησης**

Τομέας Συστημάτων Παραγωγής

Επιβλέπων: Αναπληρωτής Καθηγητής Κατσαβούνης Στέφανος

**Διαδικτυακή εφαρμογή παρακολούθησης & διαχείρισης έξυπνων συσκευών με σκοπό την μετατροπή ενός συμβατικού σπιτιού σε έξυπνο**

Διπλωματική Εργασία

**Μησσήν Στέφανος**

Αρ. Μητρώου: 91432



**Ξάνθη, Οκτώβριος 2018**



**Δημοκρίτειο Πανεπιστήμιο Θράκης Πολυτεχνική Σχολή**

**Τμήμα Μηχανικών Παραγωγής και Διοίκησης**

Τομέας Συστημάτων Παραγωγής

Επιβλέπων: Αναπληρωτής Καθηγητής Κατσαβούνης Στέφανος

**Διαδικτυακή εφαρμογή παρακολούθησης & διαχείρισης έξυπνων συσκευών με σκοπό την μετατροπή ενός συμβατικού σπιτιού σε έξυπνο**

Διπλωματική Εργασία

**Μησσήν Στέφανος**

Αρ. Μητρώου: 91432

Διπλωματική εργασία η οποία υποβλήθηκε τον Οκτώβριο 2018

για την απόκτηση του διπλώματος του Μηχανικού Παραγωγής και Διοίκησης

*Αφιερωμένο στους γονείς μου,*

***Κίμωνα*** *και* ***Μαρία***.

# Πρόλογος

Οι έννοιες των έξυπνων συσκευών (Smart Devices), των έξυπνων σπιτιών (Smart Homes) αλλά και των έξυπνων συμπλεγμάτων (Smart Clusters), συνδέονται με την έννοια των Πραγμάτων του Διαδικτύου (Internet of Things – IoT). Όλες οι παραπάνω έννοιες αποτελούν επίκεντρο της τεχνολογικής έρευνας. Έτσι, για την μετατροπή ενός συμβατικού σπιτιού σε έξυπνο, κρίνεται αναγκαία η δημιουργία ενός συστήματος επικοινωνίας, μεταξύ των έξυπνων συσκευών που είναι εγκατεστημένες στο σπίτι, αλλά και του ανθρώπου, έτσι ώστε ο δεύτερος, να διαχειρίζεται τις λειτουργίες που προσφέρουν οι έξυπνες συσκευές ανάλογα με τις απαιτήσεις του.

Η παρούσα διπλωματική εργασία μελετά την δημιουργία ενός συστήματος παρακολούθησης & διαχείρισης έξυπνων συσκευών με σκοπό την μετατροπή ενός συμβατικού σπιτιού σε έξυπνο. Πιο συγκεκριμένα, εξετάζονται οι εσωτερικές & εξωτερικές θερμοκρασίες του χώρου μέσω αισθητήριων οργάνων, γίνεται η αποθήκευση των μετρήσεων σε κατάλληλες βάσεις δεδομένων και έτσι ο χρήστης μπορεί να κρίνει πότε είναι αναγκαία η ενεργοποίηση ή η απενεργοποίηση των έξυπνων συσκευών εντός του χώρου του σπιτιού (στην παρούσα διπλωματική έχουν χρησιμοποιηθεί ως έξυπνες συσκευές, το κλιματιστικό και ο αφυγραντήρας). Η διαχείριση των έξυπνων συσκευών γίνεται μέσω από την διαδικτυακή εφαρμογή (Web Application) Smart Homie που δημιουργήθηκε για τις ανάγκες της παρούσας διπλωματικής εργασίας. Επιπλέον, εξετάζονται τα πρωτόκολλα αποστολής δεδομένων από τα αισθητήρια όργανα προς τις βάσεις δεδομένων, που είναι η ασύρματη μεταφορά δεδομένων (Wi-Fi) καθώς και η ενσύρματη μεταφορά δεδομένων (Ethernet), οι τρόποι αποφυγής απωλειών δεδομένων, καθώς και η εύρυθμη επικοινωνία των έξυπνων συσκευών με την διαδικτυακή εφαρμογή.

Το πρόβλημα προσεγγίστηκε από διαφορετικές διαστάσεις. Αρχικά διαπιστώθηκε το κατά πόσο είναι εφικτό, η δημιουργία μίας γέφυρας επικοινωνίας μεταξύ των έξυπνων συσκευών και του ανθρώπου. Έπειτα αναπτύχθηκε ο κατάλληλος αλγόριθμος για την συλλογή πρωτογενών δεδομένων θερμοκρασίας και υγρασίας εντός και εκτός του σπιτιού που χρησιμοποιήθηκε για την παρούσα εργασία, και τέλος αναπτύχθηκε μία διαδικτυακή εφαρμογή με σκοπό την διεπαφή του χρήστη με τις έξυπνες συσκευές αλλά και την διαχείριση τους.

Ακολούθησε μια εκτενής ανάλυση των αποτελεσμάτων. Διαπιστώθηκαν οι περιορισμοί και οι υποθέσεις της εργασίας που αφορούν το κατά πόσο εύχρηστη θα μπορεί να είναι μία τέτοιου είδους διαδικτυακή εφαρμογή για τον άνθρωπο, καθώς και οι τρόποι βελτίωσης της.

Αναλυτικότερα στο 1ο κεφάλαιο παρουσιάζεται το πρόβλημα, η σημασία του για την ανθρωπότητα, οι στόχοι της διπλωματικής εργασίας καθώς και η δομή αυτής.

Στο 2ο κεφάλαιο εξετάζεται το θεωρητικό υπόβαθρο, αναλύονται οι τεχνολογίες καθώς και ο εξοπλισμός που χρησιμοποιήθηκε για την υλοποίηση της διαδικτυακής εφαρμογής.

Έπειτα αναλύεται το υλικό μέρος, η συνδεσμολογία του συστήματος για την άντληση δεδομένων και τέλος, η σύνδεση αυτών με τον τοπικό διακομιστή και τις βάσεις δεδομένων στο 3ο κεφάλαιο.

Στο κεφάλαιο 4 επεξηγείται λεπτομερώς η σχεδίαση και η υλοποίηση της διαδικτυακής εφαρμογής. Πιο συγκεκριμένα, αναλύονται οι απαιτήσεις του συστήματος, γίνεται λόγος για την ασφάλεια των δεδομένων του χρήστη και πώς προστατεύονται τα παραπάνω. Ακόμη, αναλύονται οι βάσεις δεδομένων και οι λειτουργίες της εφαρμογής.

Στο 5ο κεφάλαιο συνοψίζονται τα συμπεράσματα και τα προβλήματα που αντιμετωπίστηκαν κατά την υλοποίηση της διπλωματικής εργασίας και αναλύονται περαιτέρω, τρόποι βελτίωσης της διαδικτυακής εφαρμογής

# Ευχαριστίες

Στα πλαίσια της παρούσας διπλωματικής εργασίας, αισθάνομαι την υποχρέωση να ευχαριστήσω ορισμένους ανθρώπους που γνώρισα, όπου χωρίς αυτούς θα ήταν πολύ δυσκολότερος και δύσβατος ο δρόμος που έχω διανύσει για να φτάσω ως εδώ. Συνέβαλλαν όλοι με τον τρόπο τους στην διαμόρφωση του χαρακτήρα και της νοοτροπίας μου. Με βοήθησαν να πετύχω πολλά από τα όνειρα μου αλλά και να δημιουργήσω νέα, βάζοντας έτσι στόχο, στο προσεχές μέλλον να κατορθώσω και να εκπληρώσω όλες τις φιλοδοξίες μου.

Αρχικά οφείλω να ευχαριστήσω τον κ. Κατσαβούνη Στέφανο, Αναπληρωτή Καθηγητή του Τμήματος Μηχανικών Παραγωγής και Διοίκησης της Πολυτεχνικής Σχολής του Δημοκριτείου Πανεπιστημίου Θράκης, επιβλέποντα της παρούσας διπλωματικής εργασίας μου. Η καθοδήγηση, οι προτάσεις για βελτίωση καθώς και η συνεχής διαθεσιμότητα του σε ότι χρειαζόμουν, με βοήθησε να ξεπεράσω τις δυσκολίες που αντιμετώπισα στην διπλωματική μου εργασία και να κάνω τις ιδέες μου πραγματικότητα. Από την πρώτη κιόλας μέρα της γνωριμίας μας, κατάφερε να μου κεντρίσει το ενδιαφέρον, μεταδίδοντας μου έτσι κομμάτι των γνώσεων & και των εμπειριών του όσο αφορά επιστημονικούς κλάδους, που ανταποκρίνονται στα ενδιαφέροντα μου.

Τους Καθηγητές του Τμήματος Μηχανικών Παραγωγής και Διοίκησης, που με την πολυετή πείρα τους κατάφεραν να μεταδώσουν με επιτυχία πολύτιμες γνώσεις και εξελίξεις της επιστήμης μας, χρήσιμες για την περαιτέρω επαγγελματική μου σταδιοδρομία.

Τους φίλους και συναδέλφους μου, για την υποστήριξη και τη συνέργεια μας τα τελευταία αυτά 5 χρόνια. Τους ευχαριστώ θερμά για τον ευχάριστο χρόνο, τις χαρές αλλά και τις λύπες που περάσαμε μαζί, και πάνω από όλα που καταφέραμε να είμαστε φίλοι και συνοδοιπόροι όλο αυτό τον καιρό χωρίς διαλλείματα και παρεξηγήσεις.

Τέλος, έχω αφήσει τους σημαντικότερους ανθρώπους όλης μου της ζωής, την οικογένεια μου, χωρίς τους οποίους δεν θα έγραφα αυτές τις σελίδες της παρούσας διπλωματικής εργασίας. Τους ευχαριστώ με όλη μου την καρδιά για την απεριόριστη υποστήριξη, δύναμη αλλά και επιμονή που μου ενέπνευσαν ώστε να φτάσω μέχρι εδώ.

Τον πατέρα μου και την μητέρα μου, Κίμωνα και Μαρία, που με στήριξαν σε κάθε ξεχωριστό βήμα της ζωής μου, που μου έδωσαν την οικονομική αλλά και ψυχολογική υποστήριξη να σπουδάσω και πάνω από όλα στην αγάπη που μου έχουν δείξει καθ’ όλη την διάρκεια της ζωής μου.

Έπειτα τον παππούδες μου, Νικόλαο & Ωριγένη καθώς και τις γιαγιάδες μου, Μαρία & Άννα, με τους οποίους αν και δεν έχω περάσει αρκετές στιγμές, έχουν χαράξει στην μνήμη μου ο καθένας ξεχωριστά την δική του εικόνα & προσωπικότητα .

# Πίνακας περιεχομένων

# Περιεχόμενα Πινάκων και Σχημάτων

Πίνακας 1. Πίνακας γειτνίασης κατευθυνόμενου γραφήματος 19

Πίνακας 2. Πίνακας γειτνίασης μη κατευθυνόμενου γραφήματος 20

Πίνακας 3. Περιγραφή των κύριων μεταβλητών του προβλήματος δρομολόγησης οχημάτων με την Χρήση Χρονικών Παραθύρων 26

Πίνακας 4. Χαρακτηριστικά Κόμβων - Πελατών προς Εξυπηρέτηση, σύμφωνα με το

πρόβλημα RC201 του Solomon 74

Πίνακας 5. Βέλτιστες λύσεις του προβλήματος RC201 76

Πίνακας 6. Στοιχεία Μικρού Συνόλου Δεδομένων 79

Πίνακας 7. Στοιχεία Μεσαίου Συνόλου Δεδομένων 80

Πίνακας 8. Στοιχεία Μεγάλου Συνόλου Δεδομένων 81

Πίνακας 9. Σύγκριση των 5 Προσεγγίσεων που χρησιμοποιήθηκαν στο Σενάριο 1. 89

Πίνακας 10. Σύγκριση Αποτελεσμάτων Σεναρίου 1. 90

Πίνακας 11. Σύγκριση των 5 Προσεγγίσεων που χρησιμοποιήθηκαν στο Σενάριο 2. 93

Πίνακας 12. Σύγκριση Αποτελεσμάτων Σεναρίου 2 94

Πίνακας 13. Σύγκριση των 5 Προσεγγίσεων που χρησιμοποιήθηκαν στο Σενάριο 3. 97

Πίνακας 14. Σύγκριση Αποτελεσμάτων Σεναρίου 3. 98

Πίνακας 15. Σύγκριση των 5 Προσεγγίσεων που χρησιμοποιήθηκαν στο Σενάριο 4. 101

Πίνακας 16. Σύγκριση Αποτελεσμάτων Σεναρίου 4. 102

Πίνακας 17. Σύγκριση των 5 Προσεγγίσεων που χρησιμοποιήθηκαν στο Σενάριο 5. 105

Πίνακας 18. Σύγκριση Αποτελεσμάτων Σεναρίου 5. 106

Πίνακας 19. Σύγκριση των 5 Προσεγγίσεων που χρησιμοποιήθηκαν στο Σενάριο 6. 109

Πίνακας 20. Σύγκριση Αποτελεσμάτων Σεναρίου 6. 110

Πίνακας 21. Σύγκριση των 5 Προσεγγίσεων που χρησιμοποιήθηκαν στο Σενάριο 7. 113

Πίνακας 22. Σύγκριση Αποτελεσμάτων Σεναρίου 7. 114

Πίνακας 23. Σύγκριση των 5 Προσεγγίσεων που χρησιμοποιήθηκαν στο Σενάριο 8. 117

Πίνακας 24. Σύγκριση Αποτελεσμάτων Σεναρίου 8. 118

Πίνακας 25. Σύγκριση των 5 Προσεγγίσεων που χρησιμοποιήθηκαν στο Σενάριο 9. 122

Πίνακας 26. Σύγκριση Αποτελεσμάτων Σεναρίου 9. 123

Πίνακας 27. Σύγκριση των 5 Προσεγγίσεων που χρησιμοποιήθηκαν στο Σενάριο 10. 127

Πίνακας 28. Σύγκριση Αποτελεσμάτων Σεναρίου 10. 128

Πίνακας 29. Σύγκριση των 5 Προσεγγίσεων που χρησιμοποιήθηκαν στο Σενάριο 11. 132

Πίνακας 30. Σύγκριση Αποτελεσμάτων Σεναρίου 11. 133

Πίνακας 31. Σύγκριση των 5 Προσεγγίσεων που χρησιμοποιήθηκαν στο Σενάριο 12. 137

Πίνακας 32. Σύγκριση Αποτελεσμάτων Σεναρίου 12. 138

Πίνακας 33. Χαρακτηριστικά Κόμβων - Κάδων προς Άδειασμα. 143

Πίνακας 34. Σύγκριση των 5 Προσεγγίσεων για την περίπτωση δρομολόγησης

Απορριμματοφόρων Οχημάτων 153

Πίνακας 35. Εξυπηρέτηση Μέγιστου Αριθμού Κόμβων 156

Πίνακας 36. Εξυπηρέτηση Μέγιστης Ζήτησης 158

Πίνακας 37. Καλύτερη Κατανομή του Φορτίου 159

Πίνακας 38. Ελαχιστοποίηση της Απόστασης που Διανύει το κάθε Όχημα 160

Πίνακας 39. Ελαχιστοποίηση Χρόνου Αναμονής 161

Σχήμα 2-1. Ένα στιγμιότυπο δρομολόγησης στόλου Οχημάτων. Υπάρχουν δύο διαδρομές και

χρησιμοποιήθηκαν δύο οχήματα για να εξυπηρετήσουν οκτώ πελάτες.[17] 12

Σχήμα 2-2. Παραδείγματα κατευθυνόμενων και μη κατευθυνόμενων γραφημάτων.[12] 16

Σχήμα 2-3. Γράφημα G με 10 κορυφές και 14 ακμές[12] 17

Σχήμα 2-4. Παράδειγμα αναπαράστασης κατευθυνόμενου γραφήματος.[12] 19

Σχήμα 3-1. Γραφική σύνοψη της κατηγοριοποίησης των προβλημάτων 32

Σχήμα 3-2. Τα πιθανά ενδεχόμενα όσον αφορά τη σχέση των κλάσεων P, NP, NP-Hard και NP- Complete. 33

Σχήμα 3-3. Διαφορετικοί τρόποι ομαδοποίησης του ίδιου συνόλου σημείων.[9] 43

Σχήμα 3-4.. Διαφορετικοί τύποι συστάδων όπως απεικονίζονται από σύνολα δισδιάστατων

σημείων [9] 47

Σχήμα 3-5. Δημιουργία 3 συστάδων με τη χρησιμοποίηση του αλγορίθμου k-means[9] 50

Σχήμα 4-1. Παράδειγμα Ομαδοποίησης Κόμβων[32] 53

Σχήμα 6-1. Οδικός χάρτης περιοχής Κουφαλίων. Με πράσινο χρώμα σημειώνονται οι κόμβοι με φορτίο 300 κιλών, ενώ με μαύρο αυτοί των 100 κιλών, 147

Σχήμα 6-2. Οι συστάδες μετά την χρήση του Αλγορίθμου Improved k-means. 148

Σχήμα 6-3. Τελικές συστάδες μετά την παρέμβαση. 150

# Περίληψη

Η μεταφορά των αγαθών και των επιβατών είναι πάντοτε ένας επίκαιρος και σημαντι- κός στόχος σε κάθε κοινωνία. Μεγάλα χρηματικά ποσά δαπανώνται καθημερινά σε καύσιμα, εξοπλισμό, κόστη συντήρησης και αμοιβές εργαζομένων. Η προφανής ανάγκη μείωσης του κόστους των υπηρεσιών μεταφοράς, παράλληλα με την απαίτηση για έ- γκαιρη και ποιοτική κάλυψη της ζήτησης, αποτελούν τα θεμελιώδη χαρακτηριστικά του γενικευμένου προβλήματος της δρομολόγησης οχημάτων.

Το πρόβλημα δρομολόγησης οχημάτων (VRP) συνίσταται στον προσδιορισμό της αλ- ληλουχίας επίσκεψης ενός συνόλου θέσεων από ένα προκαθορισμένο σύνολο οχημάτων με στόχο την ικανοποίηση των απαιτήσεων που προσδιορίζονται τόσο από το σύνολο των θέσεων, όσο και από τα χαρακτηριστικά του συνόλου των οχημάτων. Λόγω της πληθώρας των προβλημάτων, των περιορισμών, των αναγκών και των επιδιωκόμενων στόχων, έχουν δημιουργηθεί μοντέλα που επικεντρώνονται στην κάλυψη συγκεκριμέ- νων περιορισμών όπως η χωρητικότητα των οχημάτων, τα πολλαπλά χρονικά παράθυ- ρα, ο ετερογενής στόλος οχημάτων κ.α.

Στην παρούσα διπλωματική εργασία διατυπώνεται μια αλγοριθμική επίλυση μιας πα- ραλλαγής των Capacitated Clustering Problem και του Vehicle Routing Problem with Time Windows, για τη δρομολόγηση των οχημάτων και την ομαδοποίηση των σημείων προς εξυπηρέτηση με την ανάπτυξη μιας οικογένειας συναφών και συμπληρωματικών ευρετικών αλγορίθμων. Υλοποιούνται μια σειρά από προσεγγίσεις που αποσκοπούν στη διατήρηση του φορτίου όλων των οχημάτων στο ίδιο επίπεδο, στην ομοιόμορφη κατα-

νομή του χρόνου εργασίας σε όλα τα οχήματα και στην εξυπηρέτηση του μεγαλύτερου δυνατού αριθμού πελατών. Ανάλογα με τις τιμές κάθε συνόλου δεδομένων, κάποιες μέ- θοδοι κρίνονται πιο αποδοτικές από τις υπόλοιπες και προτείνονται για συγκεκριμένα είδη προβλημάτων. Η υλοποίηση πραγματοποιήθηκε σε γλώσσα προγραμματισμού C++ στο περιβάλλον του λογισμικού Microsoft Visual Studio Community 2015.

Ο προτεινόμενος αλγόριθμος θεωρεί γνωστό εξ αρχής τον αριθμό των σημείων που πρέπει να εξυπηρετηθούν καθώς και τον αριθμό των διαθέσιμων οχημάτων και εξετάζει μια σειρά από σενάρια ώστε να βρεθεί το καταλληλότερο σύμφωνα με τα χαρακτηρι- στικά κάθε συνόλου δεδομένων. Αναπτύσσονται πέντε προσεγγίσεις, κάθε μία εκ των οποίων εστιάζει σε διαφορετικά χαρακτηριστικά, όπως στη μείωση της απόστασης που διανύεται, στη μείωση των χρόνων αναμονής, στην τήρηση του επιπέδου του φορτίου στις ίδιες τιμές σε όλα τα οχήματα, στην εξυπηρέτηση σημείων με συγκεκριμένα χαρα- κτηριστικά κ.α.

Ο αλγόριθμος ελέγχθηκε για ένα ικανό πλήθος δεδομένων και για μια πληθώρα συν- δυασμών των παραμέτρων του. Πραγματοποιήθηκε παραμετροποίησή του στα δεδομέ- να ενός πραγματικού προβλήματος και τα αποτελέσματα κατέδειξαν την ευστάθειά και τη δυνατότητα περαιτέρω εφαρμογής σε πιο σύνθετα προβλήματα του πραγματικού κό- σμου.

**Λέξεις–Κλειδιά:** Πρόβλημα Δρομολόγησης Οχημάτων, Χρονικά Παράθυρα, Συσταδοποίηση, Ευρετικός Αλγόριθμος, Προσέγγιση Δύο Φάσεων, Βελτιστοποίηση.

# Abstract

## Vehicle Routing Problem with Time Windows Algorithms: Development, Implementation and Evaluation

Thesis submitted to the Department of Production and Management Engineering, School of Engineering, Democritus University of Thrace, Greece, on October 2017

for the degree

*Diploma in Production and Management Engineering (Dip. Eng.)*

Supervisor: Assoc. Prof. Katsavounis Stefanos

Transportation of goods and passengers is a topical and important timeless task in every society. Large amount of money is spent daily on fuel, equipment, maintenance costs and employee wages. The apparent need to reduce the cost of transportation services, achieving concurrently timely and qualitative coverage of demand, are fundamental characteristics of the generic Vehicle Routing Problem (VRP).

The VRP consists of determining the sequence of visits of a set of positions from a pre- determined set of vehicles, in order to meet the requirements specified by both the posi- tions, and the vehicles’ characteristics. Due to the variety of problems, constrains, needs and objectives, models have been developed that focus on specific constrains such as vehicle capacity, multiple time windows, heterogeneous fleet of vehicles, etc.

This thesis presents an algorithmic solution to a variation of the Capacitated Clustering Problem (CCP) and the Vehicle Routing Problem with Time Windows (VRPTW). The thesis studies the routing of the vehicles and the grouping of positions for service, by developing a group of related and complementary heuristic algorithms. A number of ap- proaches are being implemented to maintain the load of all vehicles at the same level, the even distribution of working time to all vehicles and the serving of the largest possi- ble number of customers. Depending on the values of each dataset, some methods are considered more efficient than the others, consequently suggested for specific types of problems. The implementation has been completed in C++ programming language, in Microsoft Visual Studio Community 2015 software environment.

The proposed algorithm takes into consideration a predetermined number of points to be

served and the number of available vehicles, checking a number of scenarios to find the most appropriate according to the characteristics of each dataset. Five approaches are being developed, each of which focuses on different features, such as the reduction of traveled distance and waiting time, keeping the level of loading at the same level among all vehicles, serving points with specific needs and features, etc.

The proposed algorithm was tested for a sufficient number of data and for multiple combinations of its parameters. It was customized into a real problem and the results demonstrated the stability and the possibility of further application to more complex re- al-world problems.

**Keywords:** Vehicle Routing Problem, Time Windows, Clustering, Heuristic Algorithm, Two-Phase Approach, Optimization.

Κεφάλαιο 1

**Εισαγωγή**

## ΤΟ ΠΡΟΒΛΗΜΑ ΚΑΙ Η ΣΗΜΑΣΙΑ ΤΟΥ

Το πρόβλημα της δρομολόγησης οχημάτων (Vehicle Routing Problem - VRP) είναι ένα από τα πιο σημαντικά προβλήματα στο χώρο της επιστήμης της πληροφορικής και της επιχειρησιακής έρευνας. Παρουσιάστηκε για πρώτη φορά το 1959 από τους G. B. Dantzig and R. H. Ramser. Από τότε το πρόβλημα έχει μελετηθεί συστηματικά και έχουν παρουσιαστεί αρκετοί αλγόριθμοι για την επίλυσή του. Όμως, καθώς παρουσιάζονται νέες παραλλαγές του προβλήματος που ανταποκρίνονται στις νέες σύγχρονες απαιτήσεις, παρουσιάζεται και μία νέα πρόκληση να βρεθούν αλγόριθμοι ικανοί να ανταπεξέλθουν στα νέα αυτά μοντέλα [1].

Πολλά προβλήματα μεταφοράς και διανομής προϊόντων μπορούν να διατυπωθούν ως προβλήματα δρομολόγησης οχημάτων, των οποίων ο στόχος είναι να καλυφθούν οι απαραίτητες ανάγκες των υποψήφιων πελατών, με την υλοποίηση σχεδίων δρομολογίων που θα ελαχιστοποιούν το κόστος περάτωσης. Η λύση του προβλήματος είναι να εξυπηρετούνται όλοι οι πελάτες χρησιμοποιώντας όσο το δυνατόν λιγότερα οχήματα, ελαχιστοποιώντας παράλληλα κάποιο κριτήριο που συνήθως είναι η συνολική απόσταση που διένυσαν τα οχήματα ή ο συνολικός χρόνος οδήγησης των οχημάτων. Τα συστήματα εφοδιαστικής αλυσίδας, στα οποία η συγκεκριμένη μέθοδος έχει μεγάλη εφαρμογή, είναι συνήθως μεγάλης κλίμακας και η πολυπλοκότητά τους μπορεί να προκύψει από πολλές διαφορετικές πηγές, όπως ο αριθμός των πελατών και των διαθέσιμων οχημάτων, το πλήθος των απαιτούμενων δρομολογίων προς διεκπεραίωση καθώς και η φυσική υποδομή των περιοχών που πρέπει να εξυπηρετηθούν, οι οποίες αλληλεπιδρούν μεταξύ τους με διάφορους τρόπους [8].

Επιπλέον, λόγω της συνεχώς αυξανόμενης συνεργασίας μεταξύ εταιρειών μεταφορών και logistics, παρατηρούνται περιπτώσεις που δημιουργούνται προβλήματα σχετιζόμενα με το περιθώριο κέρδους της εκάστοτε επιχείρησης. Τα προβλήματα αυτά αυξάνονται από άποψη μεγέθους και καθίστανται πιο περίπλοκα, όσον αφορά τους «νέους» περιορισμούς που τίθενται στα πλαίσια υλοποίησης των απαραίτητων δρομολογίων. Είναι συνηθισμένο για εφαρμογές δρομολόγησης οχημάτων στην καθημερινότητα (π.χ. συλλογή αποβλήτων, υπηρεσίες ταχυμεταφοράς δεμάτων και παραδόσεις εφημερίδων) να αλληλεπιδρούν δεκάδες εκατοντάδες ή και χιλιάδες πελάτες για την καθημερινή εξυπηρέτηση. Μια εκ των κύριων παραλλαγών του προβλήματος, εγγύτερα σε πραγματικές εφαρμογές είναι η δρομολόγηση στόλου οχημάτων με χρονικά παράθυρα (Vehicle Routing Problem with Time Windows - VRPTW). Στην παραλλαγή αυτή κάθε πελάτης εξυπηρετείται κατά τη διάρκεια ενός προκαθορισμένου χρονικού διαστήματος, που είναι εξ αρχής γνωστό.

Ένα επιπλέον και εξ ίσου σημαντικό σύγχρονο πρόβλημα συνδυαστικής βελτιστοποίησης με ευρεία εφαρμογή σε βιομηχανικά προβλήματα και προβλήματα εξυπηρέτησης είναι το πρόβλημα Χωροταξικής Συσσώρευσης (Capacitated Clustering Problem – CCP). Στην εν λόγω περίπτωση, όπως και τα περισσότερα προβλήματα VRP, το σύνολο του προβλήματος χωρίζεται σε διάφορες ομάδες μικρότερων επιμέρους προβλημάτων αποσκοπώντας στην ελαχιστοποίηση του αθροίσματος των αποστάσεων. Παράλληλα το άθροισμα των αντίστοιχων βαρών των κόμβων δεν πρέπει να υπερβαίνει μια σταθερή χωρητικότητα έτσι ώστε να επιτυγχάνεται μια ομοιόμορφη κατανομή (εξισορρόπηση) σε συγκεκριμένους παράγοντες του συνόλου των χαρακτηριστικών κάθε κόμβου. Η εξισορρόπηση αυτή κατά τη διαδικασία παραγωγής, παροχής υπηρεσιών ή εξυπηρέτησης πελατών μπορεί να οδηγήσει στη μείωση του κόστους προϊόντος ή υπηρεσίας. Δεδομένου ότι η χαμηλότερη απόδοση κόστους προϊόντων ή υπηρεσιών συμβάλλει στην ανταγωνιστική ωφέλεια της κάθε εταιρείας, τα κέρδη μπορούν να αυξηθούν αξιοσημείωτα.[20].

## ΓΙΑΤΙ ΜΑΣ ΑΠΑΣΧΟΛΕΙ ΤΟ ΠΡΟΒΛΗΜΑ ΔΡΟΜΟΛΟΓΗΣΗΣ ΟΧΗΜΑΤΩΝ (VEHICLE ROUTING PROBLEM);

Η μεταφορά των αγαθών και των επιβατών είναι πάντοτε ένας επίκαιρος και σημαντικός στόχος στην κοινωνία. Μεγάλα χρηματικά ποσά δαπανώνται καθημερινά σε καύσιμα, εξοπλισμό, κόστη συντήρησης του εξοπλισμού και αμοιβές εργαζομένων. Είναι επομένως προφανής η ανάγκη να γίνει κάποια προσπάθεια μείωσης του χρηματικού ποσού που δαπανάται στη μεταφορά, δεδομένου ότι ακόμη και οι μικρές βελτιώσεις μπορούν να οδηγήσουν σε μεγάλες διακυμάνσεις στο τελικό περιθώριο κέρδους για την

εκάστοτε επιχείρηση ή οργανισμό.

Το πρόβλημα δρομολόγησης οχημάτων (VRP) συνίσταται στον προσδιορισμό της αλληλουχίας επίσκεψης ενός συνόλου θέσεων από ένα προκαθορισμένο σύνολο οχημάτων με στόχο την ικανοποίηση των απαιτήσεων που προσδιορίζονται τόσο από το σύνολο των θέσεων, όσο και από τα χαρακτηριστικά του συνόλου των οχημάτων. Είναι ένα από τα βασικά προβλήματα που ελκύει την προσοχή πολλών ερευνητών κατά τη διάρκεια των τελευταίων δεκαετιών και συνδυάζεται με προβλήματα που χρειάζονται βελτιστοποίηση και έχουν αποτελέσει αντικείμενο πολλών μελετών. Είναι προφανές ότι στη γενική της έννοια, η δρομολόγηση οχημάτων αφορά ένα πλήθος διαφορετικών προορισμών, περιορισμών και παραδοχών που εξειδικεύονται στις ποικίλες παραλλαγές του προβλήματος. Στις περισσότερες περιπτώσεις, η δρομολόγηση αφορά προϊόντα (αγαθά) ή επιβάτες και η επίλυση προβλημάτων δρομολόγησης για περιπτώσεις μεγάλης κλίμακας έχουν εξαιρετικό ενδιαφέρον. Με εξαίρεση το πρόβλημα του περιοδεύοντος πωλητή (TSP), όπου περιπτώσεις με αρκετούς χιλιάδες κόμβους μπορούν να επιλυθούν με βέλτιστο τρόπο με σχετική ευκολία, οι περιπτώσεις VRP με περισσότερους από εκατό πελάτες μπορεί να είναι πολύ δύσκολο να επιλυθούν βέλτιστα. Περιορισμένο πλήθος μεθόδων επιτυγχάνουν να επιλύσουν το συγκεκριμένο πρόβλημα σε βαθμό κοντά στο βέλτιστο [22].

Οι περισσότερες περιπτώσεις επίλυσης περιορίζονται σε επίλυση μέρους του συνολικού προβλήματος. Για το λόγο αυτό, η σχετική βιβλιογραφία περιλαμβάνει πληθώρα μεταευρετικών (metaheurestic) αλγορίθμων, μεθόδων που βασίζονται σε πολλαπλή αναζήτηση αποτελεσμάτων καθώς και υβριδικών προσεγγίσεων. Οι υβριδικές προσεγγίσεις, κινούνται μεταξύ των ευρετικών μεθόδων και τεχνικών που επιδιώκουν τη βέλτιστη λύση και είναι ικανές να παράγουν υψηλής ποιότητας λύσεις με αντίστοιχα μεγάλο υπολογιστικό χρόνο για πρακτικές εφαρμογές. Παρόλα αυτά, και την πρόοδο που έχει σημειωθεί τα τελευταία χρόνια στο συγκεκριμένο κλάδο, πολλά προβλήματα εξακολουθούν να υπάρχουν και εμφανίζονται και νεότερα.

Η δρομολόγηση οχημάτων έχει πολλές πρακτικές εφαρμογές, όπως στους τομείς της έρευνας διαδικασιών, της διοικητικής μέριμνας, της διανομής και της διαχείρισης αλυσίδων εφοδιασμού. Γενικά, η δρομολόγηση οχημάτων περιλαμβάνει την εύρεση των αποδοτικών διαδρομών για τα οχήματα κατά μήκος των δικτύων μεταφοράς, προκειμένου να ελαχιστοποιηθούν το μήκος διαδρομών, το κόστος υπηρεσιών, ο χρόνος ταξιδιού, το πλήθος οχημάτων κλπ. Η ανάγκη εύρεσης μεθόδων επίλυσης τέτοιων προβλημάτων οφείλεται σε οικονομικούς κυρίως λόγους.

Η μεταφορά είναι από τις πιο σημαντικές λειτουργίες σε μια επιχείρηση και αποτελεί ένα από τα κύρια κόστη της. Δηλαδή, κάθε μεταφορά, όπως είναι κατανοητό, στοιχίζει σε χρόνο και χρήμα. Π.χ., για τη διανομή ενός προϊόντος σε μία αλυσίδα καταστημάτων πρέπει να ληφθούν υπόψη: η διαθεσιμότητα των απαραίτητων φορτηγών οχημάτων και των οδηγών τους, ο χρόνος ολοκλήρωσης, το σταθερό κόστος οχημάτων και το συνολικό κόστος ταξιδιού. Τίθεται συνεπώς το πρόβλημα δρομολόγησης οχημάτων, όπου για κάθε μεταφορά πρέπει να υπολογιστεί η περισσότερο συμφέρουσα διαδρομή ώστε να προκύψει μία βέλτιστη λύση που θα μειώνει το κόστος και / ή το χρόνο ολοκλήρωσης κάθε απαίτησης μεταφοράς [8].

Μία μεγάλη ποικιλία προβλημάτων δρομολόγησης οχημάτων έχει μελετηθεί τα τελευταία χρόνια. Διαφορετικά τέτοια προβλήματα αφορούν διαφορετικές περιπτώσεις και χαρακτηριστικά γνωρίσματα (αριθμός και τύπος των εμπορευμάτων και των οχημάτων, αριθμός τοποθεσιών παραλαβών και προορισμών, χρόνος εξυπηρέτησης πελατών κλπ.), εστιάζοντας πάντα στην αποτελεσματική χρήση των οχημάτων που πρέπει να παραλάβουν ή να παραδώσουν, ώστε να ανταπεξέλθουν στις απαιτήσεις των πελατών. Στόχος τους είναι να καλύψουν πλήρως τις ανάγκες των εταιρειών μεταφορών, ανεφοδιασμού και γενικότερα κάθε επιχείρησης που χρειάζεται ένα σύγχρονο, πλήρες πληροφοριακό σύστημα διοίκησης στόλου οχημάτων σε πραγματικό χρόνο.

Ο ανταγωνισμός στον τομέα αυτό αυξάνεται όλο και περισσότερο τα τελευταία χρόνια. Μια αποτελεσματική λύση στο πρόβλημα δρομολόγησης οχημάτων μπορεί να δώσει σε μια επιχείρηση μεταφορών ένα ανταγωνιστικό πλεονέκτημα έναντι των ανταγωνιστών της. Η μείωση των διανυόμενων χιλιομέτρων προφανώς μειώνει το κόστος, αλλά και περαιτέρω εξοικονόμηση μπορεί να πραγματοποιηθεί μέσω της μείωσης των φορτηγών οχημάτων που απαιτούνται. Προκύπτει επομένως ένα ακόμη πλεονέκτημα μέσω της χαμηλότερης δαπάνης για τον εν λόγω εξοπλισμό αλλά και του ανθρώπινου δυναμικού και του κόστους συντήρησής του. Ακόμα, οι πελάτες εκτιμούν τους μεταφορείς που επιτυγχάνουν ένα υψηλότερο ποσοστό έγκαιρης ικανοποίησης των αιτημάτων μεταφοράς, δεδομένου ότι μειώνεται το διάστημα αποθήκευσης των εμπορευμάτων τους. Όλα αυτά παρέχουν σε μια επιχείρηση μεταφορών ένα ανταγωνιστικό πλεονέκτημα μέσω της παροχής καλύτερης υπηρεσίας [8].

Επιπλέον, το ενδιαφέρον για το πρόβλημα δρομολόγησης οχημάτων προκύπτει εν μέρει λόγω της πρακτικής σημασίας του. Η ταχεία ανάπτυξη της βιομηχανίας μεταφορών δεμάτων, λόγω της αυξανόμενης ζήτησης ηλεκτρονικού εμπορίου, και η αύξηση της βιομηχανίας του συστήματος μεταφορών για αρκετές κατηγορίες του πληθυσμού, όπως ηλικιωμένοι, άτομα με κινητικά προβλήματα και άτομα με υψηλές απαιτήσεις

μετακίνησης, παίζουν σημαντικό ρόλο στη διάδοση και στην αναγκαιότητα επίλυσης αυτών των προβλημάτων. Φαίνεται πια ξεκάθαρα ότι το πρόβλημα δρομολόγησης οχημάτων αποτελεί καθημερινότητα όλο και περισσότερων επιχειρήσεων, όλο και περισσότερων ανθρώπων.

Η κατανόηση και η επίλυση ενός προβλήματος δρομολόγησης οχημάτων μπορεί να βοηθήσει σε μέγιστο βαθμό μία επιχείρηση, ακόμη και αν αυτή δεν διαθέτει μεγάλο αριθμό οχημάτων. Σε κάθε περίπτωση, τα οφέλη είναι πολυάριθμα και σε αυτά, ενδεικτικά, περιλαμβάνονται τα εξής:

* + - Αυτόματος εντοπισμός των οχημάτων και καλύτερος έλεγχός τους καθώς και του τρόπου με τον οποίο κινούνται σε πραγματικό χρόνο, μέσα στην πόλη, τη χώρα ή ακόμη και στο εξωτερικό.
    - Αποδοτικότερη δρομολόγηση των οχημάτων με παράλληλη μείωση του χρόνου απασχόλησης των εργαζομένων σε αυτά.
    - Βελτίωση των χρόνων παράδοσης και διεκπεραίωσης εργασιών.
    - Βελτίωση της ποιότητας των παρεχόμενων υπηρεσιών.
    - Αυξημένη πελατειακή πίστη (customer loyalty) μέσω της παροχής βελτιωμένων υπηρεσιών.
    - Εξασφάλιση των απαιτούμενων συνθηκών διακίνησης και παράδοσης (θερμοκρασία κλπ.), παράμετρος ιδιαίτερα σημαντική για ευπαθή προϊόντα που διακινούνται με οχήματα-ψυγεία.
    - Εντοπισμός των πελατών ή γενικότερα των σημείων που προκαλούν καθυστέρηση στα δρομολόγια των οχημάτων.
    - Αύξηση της απόδοσης του στόλου των οχημάτων.
    - Γεωγραφική απεικόνιση και κατηγοριοποίηση όλου του πελατολογίου.
    - Εξαγωγή στατιστικών δεδομένων σχετικά με τις κινήσεις των οχημάτων.
    - Καλύτερη διαχείριση ενός από τα σημαντικότερα περιουσιακά στοιχεία μίας επιχείρησης, των οχημάτων της.

## ΣΤΟΧΟΙ ΤΗΣ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Στόχοι της διπλωματικής εργασίας ήταν αφενός η πλήρης επισκόπηση του προβλήματος δρομολόγησης στόλου οχημάτων και των παραλλαγών του και αφετέρου η ανάπτυξη και η υλοποίηση μιας νέας αλγοριθμικής μεθόδου επίλυσης του προβλήματος δρομολόγησης

στόλου οχημάτων με παράθυρα χρόνου. Η μέθοδος που αναπτύχθηκε δημιουργεί συμπαγείς και εξισορροπημένες συστάδες με ομοιογενώς κατανεμημένο το φορτίο, σύμφωνα με την χωρητικότητα και τα χαρακτηριστικά των μέσων διανομής και της ζήτησης του εκάστοτε σημείου. Επειδή υπάρχουν πολλές παραλλαγές του προβλήματος, η παρούσα εργασία επικεντρώνεται στην υλοποίηση του σταδίου διαχωρισμού των κόμβων σε επιμέρους ομάδες με τρόπο όμοιο με αυτό του προβλήματος Χωροταξικής Συσταδοποίησης (Capacitated Clustering Problem – CPP) και τη δημιουργία περιοχών προς εξυπηρέτηση με ομοιόμορφα κατανεμημένα τα επίπεδα φορτίου σε κάθε όχημα. Στη συνέχεια υλοποιείται η διαδικασία δρομολόγησης του στόλου οχημάτων με χρονικά παράθυρα (Vehicle Routing Problem with Time Windows - VRPTW).

Στο μοντέλο που μελετάται στην παρούσα διπλωματική εργασία δημιουργούνται αρχικά ομάδες κόμβων σύμφωνα με κριτήρια προτεραιότητας τα οποία σχετίζονται με τη γεωγραφική τοποθεσία και τις απαιτήσεις σε ζήτηση κάθε κόμβου. Έπειτα κάθε πελάτης σε κάθε ομάδα πρέπει να εξυπηρετηθεί σε ένα συγκεκριμένο και προκαθορισμένο χρονικό διάστημα (π.χ. 08:00 – 10:00). Στόχος της διπλωματικής εργασίας είναι να αναπτυχθεί ένας ευρετικός αλγόριθμος που θα πραγματοποιεί τη συσταδοποίηση των πελατών σε ομάδες με κοινά χαρακτηριστικά, και θα μελετά πιθανές λύσεις για το πρόβλημα της δρομολόγησης στόλου οχημάτων για κάθε ομάδα με ξεχωριστά και με διαφορετικά κριτήρια. Οι συστάδες που δημιουργούνται θα είναι συμπαγείς και ισορροπημένες, αποσκοπώντας είτε στην υπερφόρτωση κάποιων οχημάτων σε σχέση με το σύνολο των φορτηγών που διατίθενται για την επίλυση του εκάστοτε προβλήματος, ή στην αποφυγή περιπτώσεων που η διανυόμενη απόσταση από κάποια φορτηγά είναι πολύ μεγαλύτερη σε σχέση με τα υπόλοιπα.

Η προσέγγιση που προτείνεται είναι αρκετά αποδοτική σε δυναμικά περιβάλλοντα, κάτι το οποίο δοκιμάζεται με την υλοποίηση του μοντέλου σε πραγματικά δεδομένα, και μπορεί να ανταποκριθεί χωρίς επιβάρυνση του υπολογιστικού χρόνου σε πιθανές αλλαγές των δεδομένων, όπως για παράδειγμα όταν συμβαίνει μια αλλαγή στις αρχικά υπολογισμένες διαδρομές, ή όταν αλλάζουν τα χρονικά παράθυρα των περιοχών που πρέπει να εξυπηρετηθούν. Με παρόμοια λογική, αν κάποιος πελάτης ακυρώσει την παραγγελία του, εντοπίζεται με ευκολία η συστάδα στην οποία ανήκει και αφαιρείται από τη διαδρομή. Συνοπτικά, η συνεισφορά της διπλωματικής εργασίας έγκειται στους παρακάτω τομείς :

* + - Κριτική επισκόπηση της περιοχής της δρομολόγησης στόλου Οχημάτων. Δόθηκε έμφαση στο πρόβλημα της δρομολόγησης στόλου οχημάτων με Χρονικά Παράθυρα.
    - Παρουσίαση πέντε παραλλαγών του προβλήματος της δρομολόγησης στόλου οχημάτων που εστιάζουν σε διαφορετικά χαρακτηριστικά των διαδρομών που δημιουργούνται.
    - Παρουσίαση των κύριων τεχνικών επίλυσης του προβλήματος της δρομολόγησης στόλου Οχημάτων.
    - Παρουσίαση νέας ευρετικής προσέγγισης δύο φάσεων για το πρόβλημα της δρομολόγησης στόλου οχημάτων με Παράθυρα Χρόνου.
    - Υλοποίηση και πειραματική αξιολόγηση της προτεινόμενης προσέγγισης. Η αξιολόγηση έγινε με πραγματικά δεδομένα επιτρέποντας έτσι στην προσέγγιση να δείξει τη συμπεριφορά της σε σενάρια του πραγματικού κόσμου.

## ΔΟΜΗ ΤΗΣ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Η δομή της διπλωματικής εργασίας είναι η εξής: στο Κεφάλαιο 2 παρουσιάζονται οι βασικές έννοιες και μέθοδοι, καθώς και το απαραίτητο μαθηματικό υπόβαθρο για την κατανόηση της εργασίας. Αναλύεται εκτενώς το γενικό πρόβλημα της δρομολόγησης στόλου οχημάτων καθώς και διάφορες παραλλαγές του.

Στο Κεφάλαιο 3 παρουσιάζονται διάφορες τεχνικές επίλυσης του προβλήματος τόσο για ακριβείς (exact) λύσεις όσο και για ευρετικές (heuristics) λύσεις και προσεγγίζεται η μέθοδος Cluster First, Route Second η οποία είναι και η μέθοδος που υλοποιείται στην παρούσα εργασία. Ακόμη αναλύεται η θεωρία και η πολυπλοκότητα των αλγορίθμων του προβλήματος.

Στο Κεφάλαιο 4 παρουσιάζεται διεξοδικά η προτεινόμενη προσέγγιση τριών φάσεων αναλύοντας κάθε φάση ξεχωριστά και παρουσιάζεται μια σειρά από αλγόριθμους, ο καθένας εκ των οποίων στοχεύει στη βελτίωση διαφορετικών χαρακτηριστικών κατά τη διαδικασία δρομολόγησης του κάθε οχήματος.

Στο Κεφάλαιο 5 περιγράφεται το περιβάλλον υλοποίησης, η πειραματική αξιολόγηση και τα αποτελέσματα των δοκιμών που προκύπτουν από αυτή. Πραγματοποιείται ανάλυση των δεικτών κάθε προσέγγισης ανάλογα με τα διαφορετικά χαρακτηριστικά και προτείνεται η βέλτιστη λύση σε κάθε περίπτωση και ανάλογα με τις πιθανές απαιτήσεις, όπως για παράδειγμα ελαχιστοποίηση χρόνου, ελαχιστοποίηση απόστασης ή διατήρηση επιπέδου κατανομής φορτίου σε ίδιο επίπεδο για όλα τα χρησιμοποιούμενα οχήματα.

Το Κεφάλαιο 6 περιέχει τα συμπεράσματα που προκύπτουν από την υπολογιστική διαδικασία και τις δυνατότητες εφαρμογής του μοντέλου σε δυναμικά σενάρια που περιέχουν δεδομένα του πραγματικού κόσμου. Σε αυτό το κεφάλαιο γίνεται καταγραφή

ενός πραγματικού προβλήματος που παρατηρήθηκε κατά τη διαδικασία συλλογής απορριμμάτων σε Δήμο του Νομού Θεσσαλονίκης. Μετά την καταγραφή των στοιχείων και την αποτύπωση του ακριβούς προβλήματος, έγινε προσαρμογή τους στη μορφή δεδομένων του μοντέλου που εξετάζεται στο Κεφάλαιο 5 ώστε να ελεγχθεί η αποτελεσματικότητα του σε ένα σενάριο του πραγματικού κόσμου. Εξάγονται και πάλι αποτελέσματα και συμπεράσματα, συμβάλλοντας στη διατύπωση προτάσεων που βοηθούν στην καλύτερη κατανόηση και πιθανή επίλυση του προβλήματος.

Τέλος, στο Κεφάλαιο 7 παρουσιάζονται τα συγκεντρωτικά συμπεράσματα της έρευνας από τα Κεφάλαια 5 και 6, προτείνοντας τις αποδοτικότερες μεθόδους και τα σενάρια τα οποία καλύπτουν, ενώ γίνονται προτάσεις για περαιτέρω έρευνα.

Κεφάλαιο 2

**Βασικές Έννοιες-Μέθοδοι, Μαθηματικό Yπόβαθρο**

Σε αυτό το κεφάλαιο, παρουσιάζονται οι βασικές έννοιες και το απαραίτητο μαθηματικό υπόβαθρο για το πρόβλημα της δρομολόγησης στόλου Οχημάτων. Παρουσιάζεται η βασική εκδοχή του προβλήματος της δρομολόγησης στόλου οχημάτων (VRP) με τη μορφή ενός προβλήματος βελτιστοποίησης. Επίσης, παρουσιάζονται διάφορες παραλλαγές του προβλήματος και δίνεται ιδιαίτερη έμφαση στο κύριο αντικείμενο της παρούσας διπλωματικής εργασίας, το πρόβλημα δρομολόγησης στόλου οχημάτων με χρονικά παράθυρα (VRPTW). Γίνεται αναλυτική μαθηματική διατύπωση του συγκεκριμένου προβλήματος και περιγράφονται οι παραδοχές και οι περιορισμοί που χρησιμοποιήθηκαν κατά την επίλυση του προβλήματος. Η διατύπωση του προβλήματος που αποτελεί το αντικείμενο αυτής της εργασίας γίνεται στα πλαίσια της θεωρίας γραφημάτων Για το λόγο αυτό, στο Κεφάλαιο 4, παρουσιάζονται βασικά στοιχεία από τη θεωρία γραφημάτων που είναι απαραίτητα για την παρουσίαση και την κατανόηση της νέας προσέγγισης.

## ΔΡΟΜΟΛΟΓΗΣΗ ΣΤΟΛΟΥ ΟΧΗΜΑΤΩΝ

Στο πρόβλημα της δρομολόγησης στόλου οχημάτων δίνεται ένας στόλος οχημάτων με 𝑚 οχήματα με μέγιστη χωρητικότητα 𝑄, ένα σύνολο από 𝑛 πελάτες και μία αποθήκη. Δίνεται επίσης και μία μήτρα N × N κόστους 𝐶 που περιλαμβάνει το κόστος μετακίνησης από έναν πελάτη 𝑖 σε έναν πελάτη j. Το κόστος είναι συνήθως η απόσταση ή ο χρόνος (𝑖, 𝑗) που απαιτείται μεταξύ των πελατών. Για κάθε πελάτη δίνεται ένα πλήθος πληροφοριών που τον αφορούν όπως: ένα μοναδικό αναγνωριστικό για κάθε πελάτη 𝑖𝑑𝑖, οι συντεταγμένες του (γεωγραφικό μήκος, γεωγραφικό πλάτος) που συμβολίζονται με

𝑙𝑜𝑛𝑔𝜄, 𝑙𝑎𝑡𝑖 αντίστοιχα και η ζήτηση που πρέπει να ικανοποιηθεί και συμβολίζεται με 𝑑𝑖. Στόχος είναι να δημιουργηθούν διαδρομές που ξεκινούν από την αποθήκη και

καταλήγουν στην αποθήκη και εξυπηρετούν όλους τους πελάτες, ελαχιστοποιώντας το συνολικό κόστος των διαδρομών. Κάθε πελάτης θα πρέπει να εξυπηρετηθεί από ένα όχημά και πρέπει επίσης να χρησιμοποιηθούν το πολύ 𝑚 οχήματα, όσα είναι διαθέσιμα [27].

Έστω ότι δίνεται ένα πλήρες μη κατευθυνόμενο γράφημα 𝐺 = (𝑉, 𝐸) στο οποίο με 𝑉 =

{0, . . . , 𝑛} συμβολίζεται το σύνολο των κόμβων - πελατών (με 0 συμβολίζεται η αποθήκη). Σε κάθε ακμή 𝑒 ∈ 𝐸 = {(𝑖, 𝑗) ∶ 𝑖, 𝑗 ∈ 𝑉, 𝑖 < 𝑗} αντιστοιχεί ένα κόστος 𝑐𝑒 ή

𝑐𝑖,𝑗. Για κάθε ακμή διατηρείται μια ακέραια μεταβλητή 𝑥𝑒 που μετρά πόσες φορές

διασχίστηκε η ακμή σε μια λύση. Έστω ότι το 𝑟(𝑆) δηλώνει το μικρότερο πλήθος οχημάτων που χρειάζεται για να εξυπηρετηθούν οι πελάτες ενός υποσυνόλου 𝑆 ⊂ 𝑉. Επίσης, έστω 𝛿(𝑆) = {(𝑖, 𝑗) ∶ 𝑖 ∈ 𝑆, 𝑗 ∉ 𝑆 ή 𝑖 ∉ 𝑆, 𝑗 ∈ 𝑆}. Αν 𝑆 = {𝑖}, τότε απλά ισχύει ο συμβολισμός 𝛿(𝑖) αντί για 𝛿({𝑖}). Η μαθηματική μορφοποίηση του προβλήματος από τον Laporte [18] είναι το εξής ακέραιο γραμμικό πρόγραμμα:

𝑚𝑖𝑛𝑖𝑚𝑖𝑧𝑒 ∑ 𝑐𝑒𝑥𝑒

𝑒∈𝐸

𝑠. 𝑡. ∑ 𝑥𝑒 = 2, 𝑖 ∈ 𝑉\{0},

𝑒∈𝛿(𝑖)

∑ 𝑥𝑒 = 2𝑚,

𝑒∈𝛿(0)

(2.1)

(2.2)

(2.3)

∑ 𝑥𝑒 ≥ 2𝑟(𝑆), 𝑆 ⊂ 𝑉{0}, 𝑆 ≠ ∅, (2.4)

𝑒∈𝛿(𝑆)

𝑥𝑒 ∈ {0, 1}, 𝑒 ∉ 𝛿(0), (2.5)

𝑥𝑒 ∈ {0, 1, 2}, 𝑒 ∈ 𝛿(0). (2.6)

Ο περιορισμός 2.2 δηλώνει ότι σε κάθε πελάτη η επίσκεψη γίνεται μία φορά ακριβώς. Ο περιορισμός 2.3 δηλώνει ότι θα δημιουργηθούν 𝑚 διαδρομές ενώ ο περιορισμός 2.4 δηλώνει ότι δεν θα ξεπεραστεί η μέγιστη χωρητικότητα κάθε οχήματος. Τέλος οι περιορισμοί 2.5 και 2.6 δηλώνουν ότι κάθε ακμή μεταξύ δύο πελατών θα χρησιμοποιηθεί το πολύ μία φορά και κάθε ακμή προσκείμενη στην αποθήκη θα χρησιμοποιηθεί το πολύ 2 φορές. Αυτό μπορεί να συμβεί αν ένα όχημα επισκεφτεί μόνο έναν πελάτη. Ακόμα και αυτό το βασικό μοντέλο που είναι σχετικά απλό οδηγεί σε ένα ακέραιο γραμμικό πρόβλημα.

Ένας άλλος ευρέως χρησιμοποιούμενος μαθηματικός συμβολισμός του προβλήματος παρουσιάστηκε από τους Balinski και Quandt [19] και βασίζεται στο διαχωρισμό συνόλων. Το μειονέκτημά του είναι ότι μπορεί να περιέχει έναν εκθετικό αριθμό από δυαδικές μεταβλητές. Έστω ότι το σύνολο 𝑅 = {𝑅1, . . . , 𝑅𝑠} συμβολίζει όλες τις εφικτές

διαδρομές δηλαδή όλες τις διαδρομές που δεν παραβιάζουν τους περιορισμούς με 𝑠 =

|𝑅|. Κάθε διαδρομή 𝑅𝑗 σχετίζεται με ένα κόστος 𝛾𝑗 και με 𝛼𝑖,𝑗 συμβολίζεται μία δυαδική

μεταβλητή που λαμβάνει την τιμή 1 αν και μόνο αν ο κόμβος 𝑖 επισκέφθηκε από την διαδρομή 𝑅𝑗. Η δυαδική μεταβλητή 𝑥𝑗 , 𝑗 = 1, . . . , 𝑠 είναι ίση με 1 αν και μόνο αν η διαδρομή 𝑅𝑗 επιλεγεί στην λύση. Το μοντέλο είναι το εξής :

𝑠

minimize ∑ 𝛾 𝑥

(2.7)

𝑗 𝑗

𝑗=1

𝑠

𝑠. 𝑡. ∑ 𝑎𝑖,𝑗𝑥𝑗 = 1, 𝑖 ∈ 𝑉\{0},

(2.8)

𝑗=1

𝑠

∑ 𝑥𝑗

= 𝑚,

(2.9)

𝑗=1

𝑥𝑒 ∈ {0, 1}, 𝑗 = 1, . . . , 𝑠. (2.10)

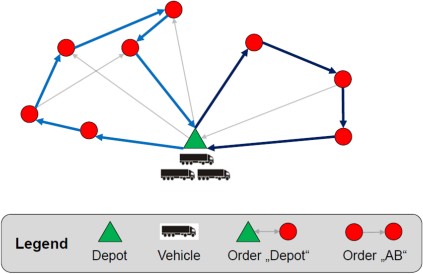
Ο περιορισμός 2.8 επιβάλλει κάθε πελάτης να εξυπηρετηθεί από ακριβώς μία διαδρομή και ο περιορισμός 2.9 ότι θα επιλεχθούν 𝑚 διαδρομές. Ο περιορισμός 2.10 δηλώνει ότι κάθε ακμή μεταξύ δύο πελατών θα χρησιμοποιηθεί το πολύ μία φορά, όπως και στο μοντέλο του Laporte. Το κύριο μειονέκτημα αυτού του μοντέλου είναι ο πολύ μεγάλος αριθμός των δυαδικών μεταβλητών. Ακόμα και για μεσαίου μεγέθους προβλήματα μπορεί να φτάσει τα δισεκατομμύρια μεταβλητές.

## 2.1.1 ΣΥΝΗΘΕΙΣ ΠΑΡΑΛΑΓΕΣ

Στο πρόβλημα δρομολόγησης οχημάτων, τα οχήματα μετακινούνται μεταξύ των πελατών της κάθε επιχείρησης, ξεκινώντας από έναν ή περισσότερους κεντρικούς σταθμούς μέσω ενός οδικού δικτύου, και επιστρέφοντας με τη λήξη του δρομολογίου στο σημείο εκκίνησης. Στα οχήματα αυτά αντιστοιχεί ένα συγκεκριμένο πλήρωμα και μπορούν να εξυπηρετήσουν ανάλογα με τη χωρητικότητά και τα χαρακτηριστικά τους, συγκεκριμένο αριθμό πελατών της επιχείρησης. Η δρομολόγηση των οχημάτων των επιχειρήσεων είναι ένα σύνθετο πρόβλημα μαθηματικού προγραμματισμού με αντικειμενική συνάρτηση το κόστος μεταφοράς και τους περιορισμούς εξυπηρέτησης που κάθε φορά προτάσσονται από τους πελάτες και τις ιδιαιτερότητες αυτών των περιορισμών. Με άλλα λόγια, στο πρόβλημα δρομολόγησης οχημάτων, πρέπει να κατασκευαστεί ένα σύνολο διαδρομών προκειμένου να ικανοποιηθούν όλα τα αιτήματα μεταφοράς για κάθε περίπτωση ξεχωριστά και ταυτόχρονα για το σύνολο των σημείων που πρόκειται να εξυπηρετηθούν. Κάθε αίτημα μεταφοράς προσδιορίζει μόνο μία τοποθεσία παραλαβής και μόνο έναν προορισμό και όλες οι τοποθεσίες τελικής παραλαβής ή παράδοσης βρίσκονται στην κεντρική αποθήκη (depot). Επιπλέον, τα συνολικά αιτήματα μεταφοράς των πελατών δεν

πρέπει να υπερβαίνουν την χωρητικότητα οχημάτων [2].

Σε αυτό το σημείο, πρέπει να αναφερθεί ότι μία διαδρομή (route) ορίζεται ως μια σειρά από τοποθεσίες πελατών, ξεκινώντας από την αποθήκη (depot) και πηγαίνοντας στη μία τοποθεσία μετά την άλλη μέχρι την ολοκλήρωση της εν λόγω διαδρομής. Μία εφικτή διαδρομή είναι η διαδρομή που ανταποκρίνεται στην χωρητικότητα του οχήματος, στην προτεραιότητα κάθε μεταφοράς και στη συμβατότητα μεταξύ φορτίου-φορτηγού [2].



*Σχήμα 2-1. Ένα στιγμιότυπο δρομολόγησης στόλου Οχημάτων. Υπάρχουν δύο διαδρομές και χρησιμοποιήθηκαν δύο οχήματα για να εξυπηρετήσουν οκτώ πελάτες.[17]*

Ένα στιγμιότυπο του προβλήματος δρομολόγησης στόλου οχημάτων φαίνεται στο *Σχήμα 2-1*. Εκτός από το βασικό μοντέλο, υπάρχει μια πληθώρα παραλλαγών για το πρόβλημα της δρομολόγησης στόλου οχημάτων. Τα μοντέλα αυτά προτάθηκαν στη βιβλιογραφία καθώς το βασικό μοντέλο δεν καλύπτει περιπτώσεις πραγματικών εφαρμογών στις οποίες υπάρχουν και επιπρόσθετοι περιορισμοί όπως: χωρητικότητα οχημάτων, (πολλαπλά) παράθυρα χρόνου για τους πελάτες, περισσότερες από μία αποθήκες, ετερογενής στόλος οχημάτων (τα οχήματα είναι διαφορετικού τύπου και έχουν διαφορετικές χωρητικότητες), τα οχήματα δεν είναι απαραίτητο να επιστρέψουν στην αποθήκη κ.ά. Παρακάτω αναφέρονται οι πιο συνηθισμένες παραλλαγές του προβλήματος δρομολόγησης στόλου οχημάτων και περιγράφονται οι διαφορές με το βασικό μοντέλο, όπως παρουσιάστηκε στην Ενότητα 2.1 [17].

* + - **Δρομολόγηση στόλου οχημάτων με Χωρητικότητες (Capacitated Vehicle Routing Problem - CVRP).** Είναι η άμεση επέκταση του βασικού μοντέλου. Σε αυτό το μοντέλο, τα οχήματα έχουν μία μέγιστη χωρητικότητα Q, την οποία δεν μπορούν να υπερβούν σε καμία περίπτωση.
    - **Δρομολόγηση στόλου οχημάτων με Παράδοση και Παραλαβή (Vehicle Routing Problem with Pickup and Delivery - VRPPD).** Σε αυτό το μοντέλο, τα οχήματα πρέπει να εξυπηρετήσουν (να παραδώσουν σε) κάποιους πελάτες και να παραλάβουν αγαθά από κάποιους άλλους. Υπάρχει περίπτωση ένας πελάτης να επιθυμεί ταυτόχρονα και παραλαβή και παράδοση κάποιων αγαθών.
    - **Δρομολόγηση στόλου οχημάτων με Παραλαβές (Vehicle Routing Problem with Backhauls - VRPB).** Αυτό το μοντέλο είναι παρόμοιο με το VRPPD αλλά η διαφορά έγκειται στο γεγονός ότι πρώτα πρέπει να γίνουν όλες οι παραδόσεις και στη συνέχεια να γίνουν οι παραλαβές από τους πελάτες (εάν υπάρχουν).
    - **Δρομολόγηση στόλου οχημάτων με Παράθυρα Χρόνου (Vehicle Routing Problem with Time Windows - VRPTW).** Σε αυτό το μοντέλο, κάθε πελάτης πρέπει να εξυπηρετηθεί σε ένα προκαθορισμένο χρονικό διάστημα. Τα παράθυρα χρόνου μπορεί να είναι είτε ελαστικά με την έννοια ότι αν η παράδοση γίνει εκτός ορίων υπάρχει μία ποινή, είτε αυστηρά που στην περίπτωση αυτή αν υπάρξει καθυστέρηση η παράδοση δεν μπορεί να γίνει καθόλου. Το μοντέλο αυτό είναι πιο κοντά σε πραγματικές εφαρμογές μιας και σχεδόν πάντα οι πελάτες αναμένουν την εξυπηρέτησή τους κατά τη διάρκεια ενός προκαθορισμένου χρονικού διαστήματος. Και εδώ τα οχήματα έχουν μια μέγιστη χωρητικότητα 𝑄 αγαθών που μπορούν να μεταφέρουν. Η παρούσα διπλωματική εργασία επικεντρώνεται σε αυτό το μοντέλο.
    - **Δρομολόγηση στόλου οχημάτων με Πολλαπλές διαδρομές (Vehicle Routing Problem with Multiple Trips - VRPMT).** Σε αυτό το μοντέλο, το ίδιο όχημα μπορεί να εκτελέσει παραπάνω από μία διαδρομές. Και εδώ τα οχήματα έχουν μία μέγιστη χωρητικότητα 𝑄, την οποία δεν μπορούν να υπερβούν σε καμία περίπτωση.
    - **Ανοιχτή δρομολόγηση στόλου οχημάτων (Opel Vehicle Routing Problem - OVRP).** Σε αυτό το μοντέλο, τα οχήματα δεν είναι υποχρεωμένα να επιστρέψουν στην αποθήκη. Αυτό το σενάριο μπορεί να συμβεί αν μισθωθεί ένας στόλος οχημάτων για να κάνουν την παράδοση των αγαθών στους πελάτες.
    - **Δρομολόγηση στόλου οχημάτων με Ξεχωριστές Παραδόσεις (Vehicle Routing Problem with Split Delivery - VRPSD).** Σε αυτό το μοντέλο, κάθε πελάτης μπορεί να εξυπηρετηθεί από παραπάνω από ένα οχήματα. Κάθε όχημα που εξυπηρετεί έναν πελάτη μπορεί να μεταφέρει ένα ποσοστό των αγαθών που πρέπει να παραδοθούν σε αυτόν. Στο τέλος κάθε πελάτης πρέπει να λάβει την ποσότητα των αγαθών που παρήγγειλε.
    - **Δυναμική δρομολόγηση στόλου οχημάτων (Dynamics Vehicle Routing Problem - DVRP).** Σε αυτό το μοντέλο, μερικές παραγγελίες είναι γνωστές εκ των προτέρων αλλά μπορεί να προκύψουν μερικές νέες παραγγελίες που πρέπει να ενσωματωθούν άμεσα στις παλιές. Επιπλέον, κάποιος πελάτης μπορεί να ακυρώσει ξαφνικά την προγραμματισμένη παραγγελία του, κάτι που θα πρέπει φυσικά να ληφθεί υπόψη. Στην βιβλιογραφία μπορεί να αναφέρεται και με τον όρο Online Vehicle Routing Problem - OVRP.

## Περιοδική δρομολόγηση στόλου οχημάτων (Period Vehicle Routing Problem

**- PVRP).** Σε αυτό το μοντέλο, πρέπει το πρόβλημα της δρομολόγησης στόλου οχημάτων να επιλυθεί για ένα χρονικό διάστημα. Για παράδειγμα, οι διαδρομές που θα δημιουργηθούν προς τους πελάτες θα πρέπει να διατηρούνται για μία εβδομάδα, για ένα μήνα κ.ο.κ.

* + - **Δρομολόγηση στόλου οχημάτων με Πολλαπλές Αποθήκες (Multi Depot Vehicle Routing Problem - MDVRP).** Σε αυτό το μοντέλο λαμβάνονται υπόψη περισσότερες από μία αποθήκες. Τα οχήματα μπορούν να ξεκινούν και να καταλήγουν στην ίδια αποθήκη ή μπορούν να ξεκινούν από μία αποθήκη 𝑎 και να καταλήγουν σε μία αποθήκη 𝑏.
    - **Δρομολόγηση στόλου οχημάτων με Στοχαστική Ζήτηση (Vehicle Routing Problem with Stohastic Demands - VRPSTD).** Σε αυτό το μοντέλο, η ζήτηση σε κάθε πελάτη δεν είναι γνωστή εκ των προτέρων αλλά υποθέτουμε ότι ακολουθεί κάποια κατανομή. Το σενάριο αυτό μπορεί να συμβεί για παράδειγμα μετά από κάποια φυσική καταστροφή (σεισμός, πλημμύρα). Η βοήθεια που πρέπει να φτάσει σε διάφορες περιοχές δεν είναι γνωστή εκ των προτέρων. Επομένως αν η ζήτηση είναι μεγάλη ένα όχημα θα πρέπει να επιστρέψει στην αποθήκη χωρίς να έχει εξυπηρετήσει όλους τους πελάτες που του είχαν ανατεθεί. Θα ανεφοδιαστεί και θα συνεχίσει την εξυπηρέτηση των υπόλοιπων πελατών.
    - **Χρονοεξαρτώµενη δρομολόγηση στόλου οχημάτων (Time Dependent Vehicle Routing Problem - TDVRP).** Σε αυτό το μοντέλο ο χρόνος ταξιδιού μεταξύ δύο πελατών (𝑖, 𝑗) δεν είναι σταθερός και εξαρτάται από την ώρα που θα ξεκινήσει το όχημα. Με αυτό τον τρόπο λαμβάνεται υπόψη η κίνηση του οδικού δικτύου και αυξάνεται η πολυπλοκότητα του προβλήματος.
    - **Δρομολόγηση Ετερογενούς στόλου οχημάτων (Heterogeneous Fleet Vehicle Routing Problem - HFVRP).** Σε αυτό το μοντέλο, υποθέτουμε ότι υπάρχει διαθέσιμος ένας στόλος οχημάτων με διαφορετικά χαρακτηριστικά. Η διαφορά μπορεί να βρίσκεται στη χωρητικότητα κάθε οχήματος, στο μέγεθος των

οχημάτων ή στο κόστος λειτουργίας τους (κατανάλωση καυσίμου, κόστος μίσθωσης οδηγού).

* + - **Δρομολόγηση στόλου οχημάτων με Εξισορρόπηση διαδρομών (Vehicle Routing Problem with Route Balancing - VRPRB).** Σε αυτό το μοντέλο, εκτός από την κύρια συνάρτηση ελαχιστοποίησης του συνολικού κόστους των διαδρομών, υπάρχει και μία δεύτερη συνάρτηση που πρέπει να ελαχιστοποιηθεί. Αυτή είναι η διαφορά μεταξύ της μεγαλύτερης σε μήκος διαδρομής μείον την μικρότερη σε μήκος διαδρομή. Αυτός ο επιπλέον περιορισμός είναι σημαντικός καθώς αφορά ζητήματα ίσης κατανομής φόρτου εργασίας μεταξύ των οδηγών. Για παράδειγμα, δεν μπορεί ένας οδηγός να τελειώνει τη βάρδια του σε μία ώρα (έχοντας μια μικρή σε μήκος διαδρομή να εξυπηρετήσει) και ένας άλλος σε τρεις ώρες (έχοντας να καλύψει μια μεγάλη σε μήκος διαδρομή).
    - **Εμπλουτισμένη δρομολόγηση στόλου οχημάτων (Rich Vehicle Routing Problem - RVRP).** Τα τελευταία χρόνια καθώς εμφανίζονται πολλές παραλλαγές που περιλαμβάνουν πολλαπλές αποθήκες, (πολλαπλά) παράθυρα χρόνου, ετερογενείς στόλους οχημάτων, πολλαπλές διαδρομές για κάθε όχημα, παράδοση και παραλαβή αγαθών στους πελάτες κ.ο.κ., έχει εμφανισθεί στη βιβλιογραφία το μοντέλο της Εμπλουτισμένης δρομολόγησης στόλου οχημάτων που περιλαμβάνει όλους αυτούς τους περιορισμούς.

## ΟΡΙΣΜΟΙ ΚΑΙ ΑΝΑΠΑΡΑΣΤΑΣΗ ΓΡΑΦΗΜΑΤΩΝ

Ένα γράφημα 𝐺 = (𝑉, 𝐸) ή απλούστερα, 𝐺(𝑉, 𝐸) είναι ένα διατεταγμένο ζεύγος το οποίο αποτελείται από το σύνολο των κορυφών (vertices) ή κόμβων (nodes) 𝑉 και από το σύνολο των ακμών (edges) 𝛦. Οι ακμές ενός γραφήματος μπορεί να είναι κατευθυνόμενες, οπότε το γράφημα ονομάζεται κατευθυνόμενο (directed graph, ή digraph) ή μη κατευθυνόμενες, οπότε το γράφημα ονομάζεται μη κατευθυνόμενο (undirected graph).

Το σύνολο 𝛦 των ακμών ενός κατευθυνόμενου γραφήματος είναι υποσύνολο του καρτεσιανού γινομένου 𝑉 × 𝑉(𝐸 ⊆ 𝑉 × 𝑉). Επομένως, αν 𝑣, 𝑤 είναι δύο κορυφές ενός κατευθυνόμενου γραφήματος, η ακμή (𝑣, 𝑤) είναι διαφορετική από την ακμή (𝑤, 𝑣). Οι ακμές ενός κατευθυνόμενου γραφήματος αναπαρίσταται με βέλη, με τη φορά του βέλους προς τη δεύτερη κορυφή του διατεταγμένου ζεύγους, *Σχήμα 2-2* (α) [12].

Το σύνολο 𝛦 των ακμών ενός μη κατευθυνόμενου γραφήματος αποτελείται από μη διατεταγμένα ζεύγη κορυφών. Στην περίπτωση ενός μη κατευθυνόμενου γραφήματος, η ακμή {𝑣, 𝑤} είναι ίδια με την ακμή {𝑤, 𝑣}, γεγονός που υποδηλώνεται συμβολίζοντας τις

ακμές με απλές γραμμές, *Σχήμα 2-2* (β). Μερικές φορές, καταχρηστικά συμβολίζουμε μία μη κατευθυνόμενη ακμή μεταξύ των κορυφών 𝑣 και 𝑤 με (𝑣, 𝑤).

Μία ακμή (𝑣, 𝑤) ενός κατευθυνόμενου γραφήματος λέμε ότι ξεκινά από την κορυφή 𝑣 και καταλήγει στην κορυφή 𝑤. Μία ακμή {𝑣, 𝑤} ενός κατευθυνόμενου γραφήματος λέμε ότι συντρέχει στις κορυφές 𝑣 και 𝑤.

Δύο κορυφές που συνδέονται από μία μη κατευθυνόμενη ακμή {𝑣, 𝑤} λέμε ότι είναι γειτονικές. Για δύο κορυφές που συνδέονται από μία κατευθυνόμενη ακμή (𝑣, 𝑤), λέμε ότι η κορυφή 𝑤 είναι γειτονική της κορυφής 𝑣. Στα μη κατευθυνόμενα γραφήματα, η σχέση γειτνίασης των κορυφών είναι συμμετρική, δηλαδή όταν η κορυφή 𝑤 είναι γειτονική της κορυφής 𝑣, τότε και η 𝑣 είναι γειτονική της 𝑤. Προφανώς, αυτό δεν ισχύει απαραίτητα στα κατευθυνόμενα γραφήματα.

Στις ακμές ενός γραφήματος 𝐺(𝑉, 𝐸) μπορούμε να αντιστοιχίσουμε βάρη μέσω μίας συνάρτησης 𝑤 ∶ 𝐸 → 𝛪𝑅. Σε αυτή την περίπτωση, το γράφημα ονομάζεται γράφημα με βάρη στις ακμές και, μερικές φορές, συμβολίζεται με 𝐺(𝑉, 𝐸, 𝑤).



**1**

**3**

**5**

**2**

**4**

**6**



**1**

**3**

**5**

**2**

**4**

**6**

**(α) (β)**



**1**

**3**

**2**

**4**



**5**

**6**

**(γ)**

*Σχήμα 2-2. Παραδείγματα κατευθυνόμενων και μη κατευθυνόμενων γραφημάτων.[12]*

Ο βαθμός μίας κορυφής 𝑣 ενός μη κατευθυνόμενου γραφήματος είναι ο αριθμός των ακμών που συντρέχουν στην 𝑣. Για παράδειγμα, στο *Σχήμα 2-2* (β) , η κορυφή 2 έχει βαθμό 2, ενώ η κορυφή 5 έχει βαθμό 1. Στα κατευθυνόμενα γραφήματα διακρίνουμε τον αριθμό των ακμών που ξεκινούν από μία κορυφή 𝑣, ο οποίος ονομάζεται βαθμός εξόδου της 𝑣, και τον αριθμό των ακμών που καταλήγουν σε μία κορυφή 𝑣, ο οποίος ονομάζεται βαθμός εισόδου της 𝑣. Για παράδειγμα, στο *Σχήμα 2-2* (α), ο βαθμός εξόδου της κορυφής 3 είναι 3, ενώ ο βαθμός εισόδου της ίδιας κορυφής είναι 1 [12].

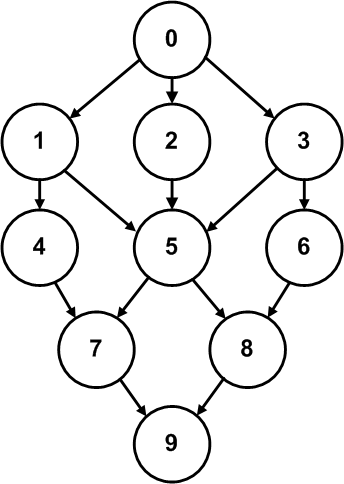
Μονοπάτι ή διαδρομή (path) ενός γραφήματος ονομάζεται μια ακολουθία ακμών (edges), όπου η τελική κορυφή (vertices) μιας ακμής είναι η αρχική κορυφή της επόμενης ακμής.

Ένα γράφημα ονομάζεται συνεκτικό (connected), αν υπάρχει τουλάχιστον ένα μονοπάτι που να συνδέει κάθε ζεύγος κορυφών του. Αν δεν συμβαίνει αυτό, τότε το γράφημα

ονομάζεται μη συνεκτικό (disconnected).

Σε ένα γράφημα, το μονοπάτι ορίζει κύκλο (cycle) αν και μόνο αν καμία ακμή δεν επαναλαμβάνεται. Ένα γράφημα του οποίου τα μονοπάτια δεν ικανοποιούν τον παραπάνω ορισμό, καλείται άκυκλο (acyclic).

Αν σε κάθε ακμή του γραφήματος αντιστοιχίσουμε ένα αριθμό, τότε ο αριθμός αυτός λέγεται βάρος (weight) και το γράφημα ονομάζεται γράφημα με βάρος στις ακμές (arc- weighted). Στην περίπτωση που σε κάθε κορυφή αντιστοιχίσουμε ένα βάρος, τότε το γράφημα που προκύπτει ονομάζεται γράφημα με βάρος στις κορυφές (vertex weighted). Αν ένα γράφημα είναι γράφημα με βάρος στις ακμές και στις κορυφές, τότε καλείται γράφημα με βάρος (weighted).

Έτσι, για παράδειγμα το γράφημα του παρακάτω *Σχήματος 2-3* είναι ένα προσανατολισμένο, συνεκτικό, άκυκλο γράφημα με βάρος, αποτελούμενο από δέκα κορυφές και δεκατέσσερις ακμές [12].

*Σχήμα 2-3. Γράφημα G με 10 κορυφές και 14 ακμές[12]*

Για κάθε κορυφή ενός μη προσανατολισμένου γραφήματος, ορίζεται το σύνολο των γειτονικών κορυφών (adjacent vertices) της κορυφής. Ο πληθικός αριθμός του συνόλου ονομάζεται βαθμός (degree) της κορυφής .

## ΑΝΑΠΑΡΑΣΤΑΣΗ ΓΡΑΦΗΜΑΤΟΣ ΣΤΗ ΜΝΗΜΗ H/Y

Η επεξεργασία ενός γραφήματος από έναν αλγόριθμο απαιτεί την αναπαράσταση του γραφήματος μέσω μιας κατάλληλης δομής δεδομένων. Οι πιο συνηθισμένοι τρόποι αναπαράστασης ενός γραφήματος είναι μέσω της λίστας γειτνίασης (adjacency list) και μέσω του πίνακα γειτνίασης (adjacency matrix).

Όταν δύο κορυφές συνδέονται μεταξύ τους με μια ακμή, τότε καλούνται γειτονικές

κορυφές (adjacent). Η κορυφή 𝑖 είναι γειτονική με την κορυφή 𝑗 αν (𝑖, 𝑗) ∈ 𝐸.

Η λίστα γειτνίασης για ένα γράφημα 𝐺(𝑉, 𝐸) αποτελείται από έναν πίνακα 𝛢, ο οποίος περιέχει |𝑉| λίστες, μία για κάθε κορυφή του γραφήματος 𝐺. Για κάθε κορυφή 𝑣 ∈ 𝑉, η λίστα 𝛢[𝑣] αποτελείται από όλες τις κορυφές που είναι γειτονικές της κορυφής 𝑣. Αν το γράφημα 𝐺 είναι με βάρη στις ακμές, μαζί με κάθε γειτονική κορυφή της 𝑣 αποθηκεύεται και το βάρος της αντίστοιχης ακμής. Προφανώς, το μέγεθος της λίστας 𝐴[𝑣] είναι ίσο με το βαθμό της 𝑣 (το βαθμό εξόδου για κατευθυνόμενα γραφήματα), και ο αριθμός των θέσεων μνήμης που απαιτούνται για την αναπαράσταση του γραφήματος 𝐺 είναι |𝑉| +

|𝐸|.

Για την αναπαράσταση ενός γραφήματος 𝐺(𝑉, 𝐸) με τον πίνακα γειτνίασης, υποθέτουμε ότι στις κορυφές του συνόλου 𝑉 έχουν αντιστοιχηθεί οι αριθμοί 1, 2, … , |𝑉|. Ο πίνακας γειτνίασης 𝛢 έχει μέγεθος |𝑉| × |𝑉| και το στοιχείο 𝐴[𝑖, 𝑗] είναι 1, αν (𝑖, 𝑗) ∈ 𝐸, διαφορετικά είναι 0. Στην περίπτωση που το γράφημα 𝐺 είναι με βάρη στις ακμές, το στοιχείο 𝐴[𝑖, 𝑗] είναι ίσο με το βάρος 𝑤(𝑖, 𝑗) της ακμής (𝑖, 𝑗) ή έχει μία ειδική τιμή, όταν η ακμή (𝑖, 𝑗) δεν υπάρχει. Προφανώς, ο αριθμός των θέσεων μνήμης που απαιτεί η αναπαράσταση με πίνακα γειτνίασης είναι |𝑉|2. Όταν το γράφημα 𝐺 είναι μη κατευθυνόμενο, μπορούμε να μειώσουμε τον αριθμό των θέσεων μνήμης στο μισό, αποθηκεύοντας μόνο τα στοιχεία 𝛢[𝑖, 𝑗], όπου 1 ≤ 𝑖 < 𝑗 ≤ |𝑉|, αφού η σχέση γειτνίασης είναι συμμετρική για τα μη κατευθυνόμενα γραφήματα.

Από τον ορισμό που δόθηκε για το γράφημα, φαίνεται ότι το γράφημα είναι ένα σύνολο κόμβων {𝑣1, 𝑣2, … , 𝑣𝑛} στο επίπεδο και ένα σύνολο ακμών 𝑒1 ,𝑒2, … , 𝑒𝑞, οι οποίες συνδέουν όλα τα σημεία ή ένα μέρος αυτών μεταξύ τους. Με βάση αυτήν την παραδοχή, κάθε γράφημα μπορούμε να το παραστήσουμε γραφικά στο επίπεδο σχεδιάζοντας κάθε

κορυφή 𝑣𝑖 ∈ 𝑉 ως ένα κόμβο και την κάθε ακμή (𝑣𝑖,𝑣𝑗) ∈ 𝐸 με μια γραμμή που ενώνει

τη 𝑣𝑖 με την 𝑣𝑗.

Στην περίπτωση του προσανατολισμένου γραφήματος ένα τόξο κατευθύνεται από την κορυφή 𝑣𝑖 προς την κορυφή 𝑣𝑗 για κάθε διατεταγμένο ζεύγος (𝑣𝑖, 𝑣𝑗) ∈ 𝐸.

Η γραφική παράσταση των γραφημάτων έχει περιορισμένη πρακτική σημασία γιατί τα γραφήματα που παρουσιάζονται σε πρακτικές εφαρμογές, αποτελούνται από μεγάλο αριθμό κορυφών ή ακμών κι έτσι η γραφική τους παράσταση είναι πρακτικά αδύνατη.

Η παράσταση ενός γραφήματος στη μνήμη ενός H/Y είναι ανεξάρτητη από το μέγεθος του γραφήματος. Η επεξεργασία του γραφήματος με H/Y είναι εκείνη που καθορίζει την επιλογή του τρόπου αποθήκευσης στη μνήμη του H/Y.

Δεν υπάρχει ένας τρόπος παράστασης των γραφημάτων στη μνήμη ενός H/Y που να είναι

το ίδιο αποτελεσματικός για την εκτέλεση ενός αλγορίθμου της θεωρίας γραφημάτων. Αυτό σημαίνει πως για κάθε αλγόριθμο υπάρχει ένας τρόπος παράστασης στη μνήμη του H/Y που μπορεί να θεωρηθεί πιο κατάλληλος από τους άλλους.

Σε αυτό το σημείο θα παρουσιαστούν σχηματικά οι δύο τρόποι παράστασης γραφημάτων στη μνήμη του H/Y:

* + - Με πίνακα γειτνίασης (adjacency matrix), που αποτελεί και το συνηθέστερο τρόπο παράστασης ενός γραφήματος στη μνήμη ενός H/Y.
    - Με λίστα γειτνίασης (adjacency list)

Ένα γράφημα 𝐺 = (𝑉, 𝐸), με 𝑛 κορυφές μπορεί να αναπαρασταθεί ως ένας πίνακας γειτνίασης 𝑛 × 𝑛 που συμβολίζεται ως 𝐴 = [𝑎𝑖,𝑗] και ορίζεται ως εξής:

* + - 𝑎𝑖,𝑗 = 1 , αν και μόνο αν (𝑣𝑖, 𝑣𝑗) ∈ 𝐸.
    - 𝑎𝑖,𝑗 = 0 , σε άλλη περίπτωση.

**1**

**5**

**6**

**6**

**4**

**2**

**4**

**2**

**4**

**1**

**3**

**2**



**1**

**3**

**5**

**2**

**4**

**6**

**3**

**4**

**5**

**6**

*Σχήμα 2-4. Παράδειγμα αναπαράστασης κατευθυνόμενου γραφήματος.[12]*

*Πίνακας 1. Πίνακας γειτνίασης κατευθυνόμενου γραφήματος*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | V1 | V2 | V3 | V4 | V5 | V6 |
| V1 | 0 | 0 | 1 | 0 | 0 | 0 |
| V2 | 1 | 0 | 1 | 1 | 0 | 0 |
| V3 | 0 | 1 | 0 | 1 | 1 | 0 |
| V4 | 0 | 1 | 0 | 0 | 0 | 1 |
| V5 | 0 | 0 | 0 | 0 | 0 | 1 |
| V6 | 0 | 0 | 0 | 0 | 0 | 1 |

Στο *Σχήμα 2-4* απεικονίζεται ένα παράδειγμα αναπαράστασης ενός κατευθυνόμενου

γραφήματος με τον *Πίνακα 1* γειτνίασης, ενώ στο *Σχήμα 2-6* απεικονίζεται ένα παράδειγμα αναπαράστασης ενός μη κατευθυνόμενου γραφήματος και ο *Πίνακας 2* γειτνίασης.

**1**

**4**

**3**

**3**

**1**

**1**

**2**

|  |  |  |
| --- | --- | --- |
| **3** |  |  |
|  |  | |
| **3** |  |  |
|  |  | |
| **2** |  |  |
|  |
|  |  | |
| **6** |  |  |

**2**



**1**

**3**

**5**

**2**

**4**

**6**

**3**

**5**

**4**

**4**

**5**

**6**

*Σχήμα 2-5. Παράδειγμα Αναπαράστασης μη κατευθυνόμενου γραφήματος[12]*

*Πίνακας 2. Πίνακας γειτνίασης μη κατευθυνόμενου γραφήματος.*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | V1 | V2 | V3 | V4 | V5 | V6 |
| V1 | 0 | 1 | 1 | 0 | 0 | 0 |
| V2 | 1 | 0 | 1 | 0 | 0 | 0 |
| V3 | 1 | 1 | 0 | 1 | 1 | 0 |
| V4 | 0 | 0 | 1 | 0 | 0 | 1 |
| V5 | 0 | 0 | 1 | 0 | 0 | 0 |
| V6 | 0 | 0 | 0 | 1 | 0 | 0 |

Όταν το γράφημα 𝐺 είναι αραιό (sparse), δηλαδή έχει αριθμό ακμών ασυμπτωτικά μικρότερο από|𝑉|2, |𝐸| ≪ (|𝑉2|), η αναπαράσταση με λίστα γειτνίασης χρησιμοποιεί ασυμπτωτικά λιγότερες θέσεις μνήμης από αυτήν με πίνακα γειτνίασης. Αντίθετα, όταν το γράφημα 𝐺 είναι πυκνό (dense), δηλαδή όταν |𝐸| ≃ (|𝑉|2), ο αριθμός των θέσεων μνήμης που χρησιμοποιείται από τις δύο αναπαραστάσεις δεν διαφέρει σημαντικά. Από την άλλη πλευρά, η αναπαράσταση με πίνακα γειτνίασης πλεονεκτεί γιατί, δεδομένων δύο κορυφών (𝑣, 𝑤) ∈ 𝑉, η ύπαρξη της ακμής (𝑣, 𝑤) μπορεί να διαπιστωθεί σε σταθερό χρόνο εξετάζοντας το αντίστοιχο στοιχείο του πίνακα, ενώ στην περίπτωση της λίστας γειτνίασης πρέπει να εξετάσουμε όλα τα στοιχεία της λίστας 𝐴[𝑣], τα οποία μπορεί να είναι μέχρι και |𝑉| .

## Γενικά χαρακτηριστικά του πίνακα γειτνίασης 𝒏 × 𝒏:

1. 𝑛2 στοιχεία
2. Συμμετρικός για μη κατευθυνόμενα γραφήματα
3. Όλες οι διαγώνιες είναι μηδενικές
4. Σε κάθε γράφημα αντιστοιχούν 𝑛! διαφορετικοί πίνακες γειτνίασης
5. Εξετάζουμε αν δύο κορυφές 𝜈𝑖, 𝜈𝑗 γειτνιάζουν μεταξύ τους, βλέποντας την τιμή του στοιχείου 𝑎𝑖,𝑗

## Παρατηρήσεις:

1. Σε γράφημα με βάρη, προσθέτουμε ένα επιπλέον πεδίο για το βάρος της πλευράς
2. Σε κατευθυνόμενο γράφημα έχουμε δύο λίστες για κάθε κορυφή, μία για τις επόμενες και μια για τις προηγούμενές της κορυφές
3. Απαιτούμενη μνήμη: 𝑂(𝑛 + 𝑚)
4. Στην περίπτωση μη κατευθυνόμενου γραφήματος ισχύει ότι: 𝑎𝑖𝑗 = 𝑎𝑗𝑖.
5. Σε απλό γράφημα ισχύει: 𝑎𝑖,𝑖 = 0.

Όσον αφορά στο ζήτημα της επιλογής αναπαράστασης ενός γραφήματος με πίνακα ή λίστα γειτνίασης, αυτό εξαρτάται από το είδος του γραφήματος, καθώς: για τους πυκνούς γράφους ενδείκνυνται οι πίνακες γειτνίασης, ενώ για τους αραιούς γράφους οι λίστες γειτνίασης. Και αυτό γιατί με τις τελευταίες επιτυγχάνεται εξοικονόμηση χώρου [7].

## ΠΡΟΒΛΗΜΑ ΔΡΟΜΟΛΟΓΗΣΗΣ ΟΧΗΜΑΤΩΝ ΜΕ ΧΡΟΝΙΚΑ ΠΑΡΑΘΥΡΑ (VEHICLE ROUTING PROBLEM WITH TIME WINDOWS - VRPTW)

Στην κατηγορία των Προβλημάτων δρομολόγησης οχημάτων ανήκουν τα Προβλήματα με χρονικά παράθυρα (Time Windows), στα οποία κάποιοι πελάτες απαιτούν η εξυπηρέτησή τους από κάποιο όχημα να ξεκινά και να τελειώνει μέσα σε ένα χρονικό διάστημα, το οποίο και ονομάζεται χρονικό παράθυρο. Όλα τα αιτήματα πρέπει να ικανοποιούνται χωρίς να παραβιάζονται οι περιορισμοί που σχετίζονται με τη χωρητικότητα των οχημάτων και την προτεραιότητα επίσκεψης των σημείων σύμφωνα με τα χρονικά διαστήματα, που καθορίζονται είτε από τη φύση του φορτίου είτε από τις απαιτήσεις των πελατών.

Το VRPTW είναι σίγουρα μία μέθοδος με μεγάλο αντικείμενο προς μελέτη και αποτελεί παραλλαγή του κλασσικού VRP. Από κάποιους θεωρείται ως το πρωτότυπο των λεγόμενων «εμπλουτισμένων»-rich VRP, με την έννοια ότι οι χρονικοί περιορισμοί απαιτούν περίπλοκες τεχνικές και συνεχείς ελέγχους σχετικά με την τελική τους περάτωση. Αυτή η παραλλαγή εμφανίζεται σε πολλά συστήματα εφοδιαστικής αλυσίδας

και μεταφορών, και χρησιμοποιείται για να μοντελοποιήσει πολυάριθμες εφαρμογές στην καθημερινότητά μας. Εφαρμογές χρήσης VRPTW σε μεγάλη κλίμακα συναντώνται στη συλλογή απορριμμάτων, στη διανομή φαγητού, στη δρομολόγηση σχολικών λεωφορείων καθώς και σε διανομές [7].

Στο πρόβλημα αυτό καθορίζονται οι διαδρομές που πρέπει να ακολουθήσουν τα φορτηγά οχήματα προκειμένου να ικανοποιήσουν ένα σύνολο πελατών, αλλά και ένα σύνολο περιορισμών με συνολικό ελάχιστο κόστος. Το VRPTW μοιάζει με το κλασσικό VRP, μόνο που εδώ για κάθε πελάτη πέρα από την ποσότητα η οποία προσδιορίζεται, υπάρχει και ένα χρονικό πλαίσιο [𝑒𝑣, 𝑙𝑣] , μέσα στη διάρκεια του οποίου ο πελάτης πρέπει να εξυπηρετηθεί. Το διάστημα [𝑒0, 𝑙0] στον κόμβο ανεφοδιασμού (depot) λέγεται ορίζοντας προγραμματισμού. Κάθε περιοχή πρέπει να ικανοποιηθεί από ένα μόνο όχημα και όλα τα οχήματα ξεκινούν και επιστρέφουν από και προς την κεντρική αποθήκη (depot) [7]. Στις στατικές περιπτώσεις, όλες οι απαιτήσεις είναι γνωστές εκ των προτέρων, πριν από τον προγραμματισμό των δρομολογίων.

Το VRPTW έχει συνήθως πολλαπλούς στόχους. Ο κύριος αντικειμενικός σκοπός του VRPTW είναι η ελαχιστοποίηση του αριθμού των οχημάτων, του συνολικού χρόνου μετάβασης και του χρόνου αναμονής (waiting time). Αυτό έχει ως αποτέλεσμα να καθοριστεί ο ελάχιστος αριθμός διαδρομών και η βέλτιστη ακολουθία πελατών που επισκέπτεται κάθε όχημα, έτσι ώστε να εξυπηρετούνται όλοι οι πελάτες και να ικανοποιούνται όλοι οι περιορισμοί που επιβάλλονται από τα χαρακτηριστικά του κάθε οχήματος, τους χρόνους εξυπηρέτησης και τα χρονικά παράθυρα. Ο. Το VRPTW, σε σχέση με το κλασσικό VRP, χαρακτηρίζεται από τους επιπλέον περιορισμούς:

1. Ένα μόνον όχημα επισκέπτεται κάθε πελάτης μόνο μία φορά, εντός ενός δεδομένου χρονικού διαστήματος το οποίο διαμορφώνει το νωρίτερο και τον αργότερο χρόνο προσέλευσης κατά τη διάρκεια της ημέρας από το σημείο προς εξυπηρέτηση.
2. Το όχημα παραμένει στη θέση του πελάτη κατά τη διάρκεια της εν λόγω εξυπηρέτησης και σε περίπτωση που το όχημα φτάσει πριν ο πελάτης είναι έτοιμος να ξεκινήσει την διαδικασία εξυπηρέτησης, περιμένει. Η άφιξη ενός οχήματος πριν το κάτω φράγμα του χρονοπαραθύρου σε κάποιο πελάτη συντελεί στην αύξηση της καθυστέρησης της διαδρομής.
3. Μια λύση είναι αδύνατη αν κάποιος πελάτης εξυπηρετείται μετά το άνω φράγμα του χρονοπαραθύρου του.
4. Κάθε εφικτή διαδρομή πρέπει να ξεκινά και να τερματίζει εντός του χρονοπαραθύρου της κεντρικής αποθήκης (depot).
5. Οι αθροιστικές ποσότητες που παραδίδουν/λαμβάνουν σε οποιονδήποτε πελάτη μιας διαδρομής, δεν πρέπει να υπερβαίνει τη χωρητικότητα του οχήματος.
6. Στην περίπτωση χαλαρών χρονοπαραθύρων (να επιτρέπεται δηλαδή μια μικρή καθυστέρηση) μια καθυστερημένη άφιξη δεν επηρεάζει την δυνατότητα μιας λύσης να είναι βέλτιστη, αλλά οπωσδήποτε εκχωρεί μια ποσότητα ποινής στην αντικειμενική συνάρτηση.

Το ενδιαφέρον εστιάζεται στην αντιμετώπιση του προβλήματος δρομολόγησης μέσων μεταφοράς και διανομής τόσο σε επίπεδο επιχειρησιακό όσο και σε επίπεδο στρατηγικού σχεδιασμού. Δηλαδή, στις ανάγκες που προκύπτουν σε δεδομένη χρονική στιγμή και πρέπει να αντιμετωπιστούν, αλλά και στο σχεδιασμό του στόλου των οχημάτων σε μακροπρόθεσμη βάση για τη συνολική αντιμετώπιση του σχεδιασμού των χρονοδιαγραμμάτων του προσωπικού και του βαθμού ετοιμότητας των οχημάτων. Ωστόσο, οι επιχειρήσεις που πραγματοποιούν μεταφορές στον πραγματικό κόσμο αντιμετωπίζουν περιορισμούς πόρων όπως είναι για παράδειγμα ο σταθερός στόλος. Το ερώτημα που πρέπει να τεθεί είναι το εάν ένα πρόβλημα είναι υπερβολικά περιορισμένο με την έννοια των ανεπαρκών οχημάτων ή κάποιας άλλης παραμέτρου, ποια θα θεωρούταν μια καλή λύση και πώς μπορεί να βρεθεί αυτή τη λύση; Είναι επιθυμητό να υπάρχει ένας αλγόριθμος ο οποίος όχι μόνο να εκτελεί καλά ένα τυποποιημένο πρόβλημα δρομολόγησης VRPTW, αλλά και να χειρίζεται προβλήματα με πολλαπλούς περιορισμούς αποσκοπώντας σε καλύτερα επίπεδα:

1. **Βελτιστοποίησης:** Επιστρέφει λύσεις που εξυπηρετούν όσο το δυνατόν περισσότερους πελάτες ως πρωταρχικό στόχο, ενώ βελτιστοποιεί τα τυποποιημένα κριτήρια όπως ο αριθμός των οχημάτων και η απόσταση που διανύθηκε.
2. **Σταθερότητας:** Βαθμιαία οι περιορισμοί υποβαθμίζονται, δηλαδή όταν μειώνεται ο αριθμός των οχημάτων, η πυκνότητα των πελατών ή ο μέσος αριθμός πελατών ανά όχημα σε λειτουργία, η σταθερότητα πρέπει να αυξάνεται ανάλογα, παρόλο που ο συνολικός αριθμός πελατών που θα εξυπηρετηθούν είναι μικρότερος.

Από τα τέλη της δεκαετίας του 1980, οι περισσότερες προσεγγίσεις επίλυσης του VRPTW αξιολογήθηκαν με βάση κάποια σύνολα δεδομένων μεσαίας κλίμακας του Solomon (1987), ενώ πρόσφατα το επίκεντρο των περισσότερων ερευνητών έχει

μετατοπιστεί σε σύνολα δεδομένων μεγαλύτερης κλίμακας, χρησιμοποιώντας τα σύνολα δεδομένων των Gehring και Homberger (1999) [24]. Αν και έχουν πραγματοποιηθεί σημαντικές έρευνες σχετικά με τα σύνολα δεδομένων του Solomon, καμία μέθοδος δεν μπορεί να δώσει με ακρίβεια τις καλύτερες γνωστές λύσεις και λίγοι μπορούν να βρουν το βέλτιστο μέγεθος στόλου [23]. Από την άλλη πλευρά, τα καλύτερα αποτελέσματα για τις περιπτώσεις προβλημάτων μεγάλης κλίμακας, ενημερώνονται συνεχώς τα τελευταία χρόνια. Ωστόσο, εξακολουθεί να υπάρχει περιθώριο βελτίωσης, ιδίως όσον αφορά την αποτελεσματικότητα της κάθε μεθόδου. Πολλές προσεγγίσεις λύσεων αποτυγχάνουν να δώσουν έναν καλό συμβιβασμό μεταξύ ποιότητας αποτελεσμάτων και ταχύτητας επίλυσης του προβλήματος, ενώ λίγες θεωρείται πως ικανοποιούν άλλες πτυχές, όπως η απλότητα και η ευελιξία.

## ΜΑΘΗΜΑΤΙΚΗ ΔΙΑΤΥΠΩΣΗ ΤΟΥ ΠΡΟΒΛΗΜΑΤΟΣ

Θεωρούμε ένα δίκτυο 𝑁 = (𝑉, 𝐸, 𝑑), όπου 𝐺 = (𝑉, 𝐸) ένα γράφημα του δικτύου 𝑁. Το σύνολο 𝑉 αποτελείται από ένα σύνολο κορυφών (vertices) ή κόμβων (nodes)

𝑉 = {𝑉0, 𝑉1, 𝑉2, . . . , 𝑉𝑛}, και από ένα σύνολο από διατεταγμένα ζεύγη κορυφών που ονομάζονται πλευρές (edges) ή ακμές (arcs) 𝐸 = { (𝑉𝑖, 𝑉𝑗): 𝑉𝑖, 𝑉𝑗 ∈ 𝑉, 𝑖 ≠ 𝑗}. Κάθε πλευρά (𝑖, 𝑗) ∈ 𝐸 συνδέει δυο κορυφές 𝑖 και 𝑗 του 𝑉, και αντιστοιχεί στον περιορισμό προτεραιότητας μεταξύ των κορυφών 𝑖 και 𝑗, όπου η επίσκεψη της κορυφής 𝑖 πρέπει να προηγηθεί της κορυφής 𝑗 από το ίδιο όχημα.

Κάθε ακμή (𝑖, 𝑗) ∈ 𝐸 αντιστοιχεί στον περιορισμό προτεραιότητας μεταξύ των κορυφών

𝑖 και 𝑗, όπου η επίσκεψη της κορυφής 𝑖 προηγείται της κορυφής 𝑗 από το ίδιο όχημα..Ο κόμβος 𝑉0 παριστάνει την αποθήκη στην οποία βρίσκονται 𝑘 οχήματα, και οι υπόλοιποι κόμβοι { 𝑉1, 𝑉2, … , 𝑉𝑛} αναπαριστούν πελάτες που πρέπει να εξυπηρετηθούν.

Το γράφημα 𝐺 έχει συνολικά 𝑛 + 1 κορυφές (0, 1, … , 𝑛) αριθμημένες τυχαία και τοποθετημένες σύμφωνα με την τοπολογική διάταξη που προκύπτει από τις συντεταγμένες (𝑥, 𝑦) κάθε σημείου. Η κορυφή 0 αντιπροσωπεύει την αποθήκη, ενώ οι υπόλοιποι 𝑛 κόμβοι αντιπροσωπεύουν τις θέσεις των αιτημάτων των πελατών για τη μεταφορά φορτίων, ορίζοντας έτσι το σύνολο 𝑁 = {1, … , 𝑛}. Με κάθε κορυφή 𝑖 ∈ 𝑁, δυο μη αρνητικοί ακέραιοι αριθμοί 𝑤𝑖, 𝑠𝑖 αντιπροσωπεύουν αντίστοιχα το φορτίο που πρέπει να μεταφερθεί στον κόμβο 𝑖 (demand) και τη διάρκεια (service time) εκφόρτωσης/εξυπηρέτησης του εν λόγω κόμβου Τα αντίστοιχα σύνολα είναι

𝑊 = {𝑤1, 𝑤2, … , 𝑤𝑛} και 𝐷 = {𝑑1, 𝑑2, … , 𝑑𝑛}. Ένα αίτημα 𝑤𝑖 αποτελείται από την παραλαβή φορτίου από την θέση 𝑉0 (depot) και από την παράδοση στην τελική θέση 𝑖. Το παραπάνω αίτημα εξυπηρετείται μόνον από ένα όχημα.

Επίσης, ισχύει: 𝑤0 = 0, 𝑑0 = 0.

Ορίζουμε μία θετική συνάρτηση 𝑑 σε κάθε άκρο (𝑖, 𝑗) ∈ Ε, 𝑑: (𝑖, 𝑗) ∈ Ε → 𝑑𝑖,𝑗 ∈ 𝑅+, η οποία αντιπροσωπεύει την απόσταση 𝑑𝑖,𝑗 από το άκρο (𝑖, 𝑗) ∈ 𝐸. Επίσης, θεωρούμε ότι

𝑑𝑖,𝑗 = 𝑑𝑖,𝑗∀(𝑖, 𝑗) ∈ 𝐸. Κάθε κορυφή 𝑁 είναι και ένα σημείο ζήτησης. Η κορυφή 0

θεωρείται σημείο έναρξης και σημείο τερματισμού για κάθε διαδρομή.

Ακόμα, το γράφημα 𝐺 περιλαμβάνει όλους τους χρόνους μεταφοράς μεταξύ όλων των ζευγών των κορυφών (𝑖, 𝑗) ∈ 𝑉, 𝑖 ≠ 𝑗. Κάθε ακμή (𝑖, 𝑗) ∈ 𝐸 έχει ένα μη αρνητικό μήκος

𝑡𝑖,𝑗 ∈ 𝑇 που είναι ίσο με τον χρόνο μεταφοράς μεταξύ των κορυφών 𝑖 και 𝑗, υποθέτοντας

ότι 𝑡𝑖,𝑗 = 𝑡𝑗,𝑖 ∀ (𝑖, 𝑗) ∈ 𝐸. Χωρίς απώλεια της γενικότητας, υποθέτουμε ότι ο χρόνος ταξιδιού απευθείας από μία τοποθεσία 𝑖 σε μία άλλη τοποθεσία 𝑗 είναι ίδιος για όλα τα φορτηγά οχήματα.

Ο σταθερός στόλος των πανομοιότυπων φορτηγών οχημάτων, που εκφράζεται από το σύνολο 𝐾 = {𝑘1, … , 𝑘𝑚}, είναι έτοιμος να εξυπηρετήσει τα αιτήματα μεταφορών. Η μόνη ιδιότητα που λαμβάνεται υπόψιν είναι η χωρητικότητα 𝑄𝑘𝑗 , 𝑘𝑗 ∈ 𝐾. Οι χωρητικότητες ορίζουν το σύνολο 𝑄 = {𝑄𝑘1 , … , 𝑄𝑘𝑚 }. Σύμφωνα με τον τρίτο περιορισμό, κάθε φορτηγό όχημα 𝑘𝑗 ∈ 𝐾 μπορεί να μεταφέρει μόνο ένα εφικτό υποσύνολο 𝐶𝑘𝑗 από τα συμβατά φορτία, γεγονός που εξαρτάται από τα χαρακτηριστικά του οχήματος (τεχνικές προδιαγραφές, δυνατότητες των οδηγών κλπ.) και από τη φύση των φορτίων. Τα φορτηγά οχήματα ξεκινούν και τερματίζουν στις κεντρικές αποθήκες 0 και 𝑛 + 1, αντίστοιχα.

Ένα έγκυρο όχημα ορίζεται ως ένα συμβατό όχημα που έχει ικανοποιητική ελεύθερη χωρητικότητα έτσι ώστε να μεταφέρει τουλάχιστον ένα φορτίο. Παρακάτω ακολουθεί η περιγραφή των κύριων μεταβλητών του προβλήματος.

*Πίνακας 3. Περιγραφή των κύριων μεταβλητών του προβλήματος δρομολόγησης οχημάτων με την Χρήση Χρονικών Παραθύρων.*

|  |  |
| --- | --- |
| 𝐶𝑖,𝑗 | Το κόστος που συνδέεται με την ακμή (𝑖, 𝑗) και το οποίο μπορεί να είναι εκφρασμένο είτε ως χιλιομετρική απόσταση, είτε ως χρόνος εκτέλεσης, ή ως κόστος ταξιδιού αθροιστικά. |
| 𝑒𝑖 | Ο νωρίτερα επιτρεπόμενος χρόνος άφιξης στον πελάτη 𝑖 |
| 𝑙𝑖 | Ο αργότερα επιτρεπόμενος χρόνος άφιξης στον πελάτη 𝑖 |
| 𝑠𝑖 | Ο χρόνος εξυπηρέτησης του πελάτη 𝑖 |
| 𝑑𝑖 | Η ζήτηση του πελάτη 𝑖 |
| 𝑄 = {𝑄𝐾1 , … , 𝑄𝐾𝑚 } | Η μέγιστη χωρητικότητα των μέσων διανομής |
| 𝑂 | Ο μέγιστος λειτουργικός χρόνος ανά μέσο |
| 𝐸 | Σύνολο των ακμών του γράφου 𝐺 με  𝐸 = { (𝑉𝑖, 𝑉𝑗): 𝑉𝑖, 𝑉𝑗 ∈ 𝑉, 𝑖 ≠ 𝑗} |
| 𝑡𝑖,𝑗 | Ο απαιτούμενος χρόνος μεταφοράς από τον κόμβο 𝑖 στον 𝑗 |
| 𝑉0 | Η κεντρική αποθήκη (depot) |
| 𝑉 = {𝑉0, 𝑉1, … , 𝑉𝑛} | Κεντρική αποθήκη και πελάτες |
| 𝐾 = {𝛫1, … , 𝛫𝑚} | Πλήθος πανομοιότυπων οχημάτων |
|  | Όταν το όχημα 𝛫𝑖 επισκέπτεται τον πελάτη 𝑖 |
| 𝛢𝛫𝑖,𝑗 | Ο χρόνος άφιξης του οχήματος 𝛫𝑖 στον πελάτη 𝑗 |
| 𝑊𝛫𝑖,𝑗 | Ο χρόνος αναμονής του οχήματος 𝛫𝑖 στον πελάτη 𝑗 |
| 𝐿𝛫𝑖,𝑗 | Ο χρόνος αποχώρησης του οχήματος 𝛫𝑖 στον πελάτη 𝑗 |
| 𝛸𝛫𝜄 = { 1  𝑖,𝑗 0 | 1, Εάν το όχημα 𝑘 επισκέπτεται τον πελάτη 𝑗 μετά τον 𝑖  0, Εναλλακτικά |
| 𝐿 = {𝐿𝑘1 , … , 𝐿𝑘𝑚 } | Το σύνολο των συνόλων των κορυφών που έχουν ήδη διατεθεί σε οποιοδήποτε φορτηγό. Κάθε σύνολο 𝐿𝐾𝑗 ορίζει το υπό-γράφημα  𝐺𝑗 = (𝐿𝐾𝑗 , 𝐸𝑗). |
| 𝑅𝐾𝑖 = {𝑅𝐾𝑖,1 , … , 𝑅𝐾𝑖,𝑗 } | Το σύνολο όλων των εφικτών διαδρομών του φορτηγού οχήματος  𝑘𝑖 |
| 𝑆 | Το σύνολο των κορυφών που έχουν ήδη διατεθεί σε κάποιο όχημα |
| 𝐵𝑖 = {𝐾𝑎, … , 𝐾𝑖} | Όλα τα συμβατά οχήματα για τον πελάτη 𝑖 |
| 𝐵 = {𝐵1, … , 𝐵𝑛} | Το σύνολο των συνόλων {𝐵1, … , 𝐵𝑛} |

## ΠΑΡΑΔΟΧΕΣ ΚΑΙ ΠΕΡΙΟΡΙΣΜΟΙ

Η ζήτηση 𝑑𝑖 της κορυφής 𝑖 πρέπει να είναι μικρότερη ή ίση της χωρητικότητας τουλάχιστον ενός φορτηγού 𝐾𝑗, δηλαδή:

𝑤𝑖 ≤ 𝑄𝐾𝑗 ∀ 𝑖 ∈ 𝑁, 𝐾𝑗 ∈ 𝐾 (2.11)

Κάθε αίτημα πρέπει να ικανοποιηθεί από ένα και μόνο όχημα, δηλαδή

𝑛

∑ 𝑥𝑖 ≤ 𝑛, 𝑥𝑖 ∈ {0,1}, 𝑥𝑖 ∈ 𝑋, 𝑖 ∈ 𝑁

𝑖=1

(2.12)

Οι δυαδικές μεταβλητές 𝑥𝑖 αντιστοιχούν στα φορτία, λαμβάνοντας την τιμή 1 όταν το αίτημα φόρτωσης/εκφόρτωσης 𝑖 διατίθεται σε ένα φορτηγό, ενώ διαφορετικά λαμβάνεται η τιμή 0.

Το συνολικό φορτίο σε μία εφικτή διαδρομή 𝑅𝐾𝑝,𝑗 δεν μπορεί να υπερβαίνει τη χωρητικότητα του οχήματος, δηλαδή

|𝑅𝐾𝑝,𝑗|

∑ 𝑑𝑖 ≤ 𝑄𝐾𝑝 , 𝑖 ∈ 𝑅𝐾𝑝,𝑗 , 𝐾𝑝 ∈ 𝐾

𝑖=1

(2.13)

Η έγκυρη ανισότητα:

|𝑅𝐾𝑝,𝑗|

(2.14)

∑ 𝑥𝑖𝑑𝑖 ≤ 𝑄𝐾𝑝 , 𝑖 ∈ 𝑅𝐾𝑝,𝑗 , 𝐾𝑝 ∈ 𝐾

𝑖=1

ικανοποιεί τους περιορισμούς χωρητικότητας και συμβατότητας και εφαρμόζεται σε κάθε φορτηγό όχημα (ή διαδρομή) 𝐾𝑝 ∈ 𝐾. Υπάρχουν συνολικά 𝑚 τέτοιες έγκυρες ανισότητες, μία για κάθε όχημα. 𝑄𝑘𝑝 είναι η τρέχουσα ελεύθερη χωρητικότητα ενός φορτηγού 𝐾𝑝. Κάθε φορά που μία κορυφή 𝑗 με φορτίο 𝑤𝑗 διατίθεται σε ένα φορτηγό 𝐾𝑝, η τιμή της χωρητικότητας 𝑄𝑘𝑝 μειώνεται κατά το ίδιο ποσό.

Το πρόβλημα έχει, πιθανόν, μία εφικτή λύση, αν η συνολική χωρητικότητα του στόλου των οχημάτων είναι μεγαλύτερη ή ίση με το σύνολο των αιτημάτων φόρτωσης/εκφόρτωσης, δηλαδή όταν η παρακάτω ανισότητα είναι αληθής:

𝑛 𝑚

∑ 𝑤𝑖 ≤ ∑ 𝑄𝑘𝑗

, 𝑖 ∈ 𝑁, 𝐾𝑗 ∈ 𝐾 (2.15)

𝑖=1 𝑗=1

Επίσης, το πρόβλημα έχει μία εφικτή λύση όταν κάθε φορτίο μπορεί να διατεθεί σε ένα φορτηγό όχημα, δηλαδή όταν η ακόλουθη Boolean ανισότητα είναι αληθής:

𝑛

∑ 𝑥𝑖 ≡ 𝑛, 𝑖 ∈ 𝑁 (2.16)

𝑖=1

Κεφάλαιο 3

**Αλγόριθμοι, Πολυπλοκότητα και Τεχνικές Επίλυσης του Προβλήματος**

Για ένα δεδομένο πρόβλημα υπάρχουν ποικίλοι αλγόριθμοι επίλυσης, οι οποίοι έχουν και διαφορετική αποδοτικότητα. Στο κεφάλαιο αυτό, παρουσιάζονται στοιχεία της θεωρίας των αλγορίθμων καθώς και οι ορισμοί και οι συμβολισμοί που χρησιμοποιούνται μετέπειτα, ενώ γίνεται αναφορά και στη θεωρία της πολυπλοκότητας. Καθώς το πρόβλημα της δρομολόγησης στόλου οχημάτων έχει μελετηθεί εκτενώς, παρουσιάζονται οι πιο βασικές τεχνικές επίλυσής του. Το ενδιαφέρον στοιχείο έγκειται στο γεγονός ότι δεν υπάρχει μία μόνο τεχνική που να εφαρμόζεται σε όλες τις παραλλαγές του προβλήματος δρομολόγησης στόλου οχημάτων και να δίνει τα καλύτερα αποτελέσματα. Διαφορετικές τεχνικές αποδίδουν καλά σε διαφορετικές παραλλαγές του εκάστοτε προβλήματος. Τέλος γίνεται εκτενής αναφορά στη μέθοδο που χρησιμοποιήθηκε στην παρούσα διπλωματική εργασία, γνωστή στην βιβλιογραφία με την ονομασία *Cluster First, Route Second*. Μέρος της ανάλυσης αποτελεί και το βήμα δημιουργίας συνόλων σύμφωνα με διαφορετικά κριτήρια κατηγοριοποίησης. Ο ακριβής όρος των συνόλων αυτών που δημιουργούνται είναι «clusters», και στη βιβλιογραφία αναφέρονται ως συστάδες. Στη συγκεκριμένη εργασία θα αναφέρονται στη συνέχεια ως *ομάδες* ή *συστάδες*

## ΣΤΟΙΧΕΙΑ ΘΕΩΡΙΑΣ ΑΛΓΟΡΙΘΜΩΝ

Με τον όρο αλγόριθμος εννοείται μία ακολουθία ξεκάθαρων, ρητών εντολών για την επίλυση ενός προβλήματος, δηλαδή για την παραγωγή της απαιτούμενης εξόδου για κάθε αποδεκτή είσοδο, σε πεπερασμένο χρόνο.

Πέρα από ένα πεπερασμένο σύνολο κανόνων που δίνουν μία ακολουθία πράξεων για την επίλυση ενός συγκεκριμένου τύπου προβλήματος, ένας αλγόριθμος έχει τα εξής πέντε

βασικά χαρακτηριστικά:

* + - Περατότητα (τελειώνει, αλλιώς είναι μία υπολογιστική μέθοδος).
    - Καθορισμός (δεν υπάρχει ασάφεια ως προς την εκτέλεση κάποιου βήματος).
    - Είσοδος (υπάρχουν μηδέν ή περισσότερα στοιχεία).
    - Έξοδος (υπάρχουν ένα ή περισσότερα στοιχεία).
    - Αποτελεσματικότητα (οι πράξεις είναι τόσο απλές ώστε να είναι εκτελέσιμες σε πεπερασμένο χρόνο με μολύβι και χαρτί) [13].

## ΘΕΩΡΙΑ ΚΑΙ ΚΛΑΣΕΙΣ ΠΟΛΥΠΛΟΚΟΤΗΤΑΣ

Η Θεωρία Πολυπλοκότητας είναι το μέρος εκείνο της θεωρίας υπολογισμού, το οποίο ασχολείται με την κοστολόγηση των πόρων που απαιτούνται για την αλγοριθμική επίλυση ενός προβλήματος. Επομένως η θεωρία πολυπλοκότητας αποτελεί βασικό δομικό λίθο της ανάλυσης αλγορίθμων και κεντρικό γνωστικό πεδίο της επιστήμης υπολογιστών.

Οι συνηθέστεροι πόροι για τους οποίους ενδιαφερόμαστε είναι ο χρόνος, οπότε μιλάμε για τη χρονική πολυπλοκότητα του αλγορίθμου, δηλαδή πόσα «βήματα» χρειάζεται να εκτελέσει ο αλγόριθμος συναρτήσει της εισόδου του, και ο χώρος, οπότε μιλάμε για τη χωρική πολυπλοκότητα, δηλαδή πόσο χώρο (μνήμη) χρειάζεται ο αλγόριθμος συναρτήσει της εισόδου του. Δηλαδή, η θεωρία πολυπλοκότητας έχει ως στόχο τη μελέτη των έμφυτων δυσκολιών των διαφόρων προβλημάτων.

Στην υπολογιστική θεωρία της πολυπλοκότητας, υπάρχει μια οικογένεια προβλημάτων, έστω 𝑁, για τα οποία έχουν αναπτυχθεί μόνον αλγόριθμοι εκθετικού χρόνου και έχουν την παρακάτω ιδιότητα:

Αν για κάποιο πρόβλημα 𝑃1 ∈ 𝑁 αναπτυχθεί αλγόριθμος πολυωνυμικού χρόνου τότε όλα τα υπόλοιπα προβλήματα της οικογένειας 𝑁 μπορούν να επιλυθούν με αλγόριθμο πολυωνυμικού χρόνου βάση του γεγονότος ότι ένα πρόβλημα 𝑃1 ∈ 𝑁 μπορεί να μετασχηματιστεί σε ισοδύναμο πρόβλημα 𝑃1 ∈ 𝑁 με αλγόριθμο πολυωνυμικού χρόνου. Η οικογένεια 𝑁 ανήκει στην κατηγορία των προβλημάτων που έχει ονομαστεί προβλήματα 𝑁𝑃 − 𝐶𝑜𝑚𝑝𝑙𝑒𝑡𝑒. Τα αντίστοιχα προβλήματα βελτιστοποίησης των προβλημάτων 𝑁𝑃 − 𝐶𝑜𝑚𝑝𝑙𝑒𝑡𝑒 ανήκουν στην κατηγορία 𝑁𝑃 − 𝐻𝑎𝑟𝑑.

## Προβλήματα 𝑷 κλάσης

Η πρώτη κατηγορία είναι τα λεγόμενα 𝑃 προβλήματα. Το 𝑃 συμβολίζει τη φράση

«Polynomial-time computable». Αυτά είναι προβλήματα τα οποία μπορούν να λυθούν σε ένα φυσιολογικό υπολογιστή και σε χρόνο της τάξης 𝑛, ή 𝑛2, ή 𝑛3, …, όπου 𝑛 είναι το μέγεθος της εισόδου στο πρόβλημα. Έτσι, σε αυτά τα προβλήματα, ο χρόνος εκτέλεσης είναι φραγμένος από πάνω από ένα πολυώνυμο - συνάρτηση της εισόδου.

Δηλαδή, ένα πρόβλημα ανήκει στην κλάση 𝑃 εάν υπάρχει ένας αλγόριθμος επίλυσης του συγκεκριμένου προβλήματος, ο οποίος εκτελείται σε συνάρτηση πολυωνυμικού χρόνου σε σχέση με το μέγεθος εισόδου. Με άλλα λόγια, ο αλγόριθμος σταματά πάντα σε αριθμό βημάτων πολυωνυμικό σε σχέση με το μέγεθος εισόδου [13].

**Προβλήματα** 𝑵𝑷 **κλάσης**

Ιστορικά είναι σωστό να πούμε, ότι μια από τις πιο σημαντικές ρίζες της υπολογιστικής θεωρίας πολυπλοκότητας είναι η έννοια της κλάσης 𝑁𝑃*.* Αυτά είναι προβλήματα τα οποία μπορούν να λυθούν μη αιτιοκρατικά σε πολυωνυμικό χρόνο. Η κλάση αυτή των προβλημάτων είναι αρκετά γνωστή στους κύκλους της Πληροφορικής. Το 𝑁𝑃 συμβολίζει τη φράση «Nondeterministic Polynomial-time computable» και αντιπροσωπεύει την κατηγορία εκείνη των προβλημάτων η οποία μπορεί μεν να λυθεί σε πολυωνυμικό χρόνο, χρειάζεται δε έναν «ειδικό» υπολογιστή, ικανό να εκτελεί «μη αιτιοκρατικούς» (non-deterministic) αλγορίθμους. Παρακάτω αναλύεται εκτενέστερα ο όρος «μη αιτιοκρατικός» αλγόριθμος. Επιπρόσθετα, στη συγκεκριμένη κατηγορία προβλημάτων περιλαμβάνονται πολύ γνωστά προβλήματα, όπως εκείνο του περιοδεύοντος πωλητή (Traveling Salesman Problem).

Οι «μη αιτιοκρατικοί» αλγόριθμοι είναι παρόμοιοι στη λειτουργία τους με τους κανονικούς αλγορίθμους, με μόνη διαφορά ότι επιτρέπεται ένα πρόσθετο είδος εντολής, με το οποίο ο υπολογιστής καλείται να «μαντέψει». Ο υπολογιστής ο οποίος εκτελεί αυτήν την εντολή, υποθέτουμε ότι έχει την ειδική ικανότητα να πραγματοποιεί πάντα μία σωστή πρόβλεψη, εάν μια τέτοια υπάρχει. Αυτό ακριβώς είναι και το χαρακτηριστικό της μη αιτιοκρατικότητας, το οποίο είναι δύσκολο να υλοποιηθεί στην πράξη.

Τα προβλήματα που ταξινομούνται στην 𝑃 και στην 𝑁𝑃 είναι όλα προβλήματα “απόφασης”, δηλαδή προβλήματα που μπορούν να επιλυθούν με ένα Ναι ή Όχι.

𝑵𝑷 − 𝑪𝒐𝒎𝒑𝒍𝒆𝒕𝒆 **προβλήματα**

Τα 𝑁𝑃 προβλήματα είναι τα προβλήματα (έστω A ένα τέτοιο πρόβλημα) τα οποία έχουν την ιδιότητα ότι, για κάθε άλλο πρόβλημα Β που ανήκει στην κατηγορία, μπορεί να βρεθεί ένα αιτιοκρατικό πρόγραμμα που εκτελείται σε πολυωνυμικό χρόνο, και το οποίο

«μετατρέπει» την είσοδο 𝑥 του προβλήματος Β, σε μία είσοδο 𝑓(𝑥) για το πρόβλημα Α,

έτσι ώστε η λύση του με είσοδο 𝑓(𝑥) και η λύση του B με είσοδο 𝑥 να είναι ίδιες. Αν υπάρχει, λοιπόν μία τέτοια συνάρτηση, η οποία να εκτελεί τη μετατροπή 𝑥 → 𝑓(𝑥) σε πολυωνυμικό χρόνο, λέμε ότι το πρόβλημα Β «ανάγεται» στο Α, δηλαδή το A θα περιέχει το πρόβλημα Β ως μία ειδική περίπτωση, και το πρόβλημα θα ονομάζεται 𝑁𝑃 −

𝐶𝑜𝑚𝑝𝑙𝑒𝑡𝑒, αν κάθε άλλο πρόβλημα 𝑁𝑃 ανάγεται σε αυτό.

Έτσι, για να δείξουμε τη 𝑁𝑃-πληρότητα (completeness) ενός δεδομένου προβλήματος A, το μόνο που πρέπει να κάνουμε είναι να δείξουμε ότι το πρόβλημα ανήκει στη 𝑁𝑃 και να ανάγουμε ένα πρόβλημα που είναι γνωστό ότι είναι 𝑁𝑃 − 𝐶𝑜𝑚𝑝𝑙𝑒𝑡𝑒 στο πρόβλημα A.

Η δυσκολία των 𝑁𝑃 − 𝐶𝑜𝑚𝑝𝑙𝑒𝑡𝑒 (NPC) προβλημάτων προέρχεται από τα ακόλουθα χαρακτηριστικά:

* + - έχουν πολλά διαφορετικά τοπικά βέλτιστα (έτσι τοπικές τεχνικές αναζήτησης δεν θα αποδώσουν)
    - δεν μπορούν να αναλυθούν σε επιμέρους προβλήματα που να λύνονται ξεχωριστά
    - εμφανίζονται τυχαία στατιστικά (έτσι οι στατιστικές μέθοδοι δεν θα λειτουργήσουν)

Επιλύσιμα Προβλήματα Μη Επιλύσιμα Προβλήματα

NP Προβλήματα

?

Πολυωνυμικά Προβλήματα

Μη Πολυωνυμικά Προβλήματα

*Σχήμα 3-1. Γραφική σύνοψη της κατηγοριοποίησης των προβλημάτων.*

Ένα πρόβλημα 𝑃𝐼 είναι 𝑁𝑃 − 𝐻𝑎𝑟𝑑*,* δεδομένου οποιουδήποτε προβλήματος 𝑃′ στο 𝑁𝑃*,* εάν υπάρχει πολυωνυμικός μετασχηματισμός του κάθε στιγμιότυπου του 𝑃′ σε ένα στιγμιότυπο του 𝑃𝐼 . Αυτό σημαίνει, ότι εάν μπορούμε να λύσουμε το 𝑃𝐼 σε πολυωνυμικό χρόνο, τότε μπορούμε να λύσουμε το 𝑃′ πολυωνυμικά, επίσης. Εάν, επιπροσθέτως, το

𝐼

𝐼

𝐼

πρόβλημα 𝑃𝐼 ανήκει στη κλάση 𝑁𝑃. λέμε ότι το 𝑃𝐼 είναι 𝑁𝑃 − 𝐶𝑜𝑚𝑝𝑙𝑒𝑡𝑒*.* Είναι σαν να λέμε ότι είναι το «δυσκολότερο» πρόβλημα στο 𝑁𝑃 [13].

Επομένως, ένα πρόβλημα 𝑃𝐼 θεωρείται 𝑁𝑃 − 𝐻𝑎𝑟𝑑 αν και μόνο αν υπάρχει ένα πρόβλημα 𝑃′ , (NP − Complete)*,* το οποίο μπορεί να πετύχει μείωση του χρόνου υπολογισμού του, ώστε να εντάσσεται στα χρονικά όρια του προβλήματος 𝑃𝐼 [15].

𝐼

Για πολλά από τα προβλήματα που εξετάζονται, το σύνολο εφικτών λύσεων είναι τόσο μεγάλο που ακόμα κι αν είχαμε έναν υπολογιστή που με έναν συστηματικό τρόπο θα μπορούσε να κατασκευάσει και να αξιολογήσει το κόστος τρισεκατομμυρίων λύσεων ανά δευτερόλεπτο, και είχε ξεκινήσει 14 δισεκατομμύρια έτη πριν, ακόμα δεν θα είχε αξιολογήσει όλες τις εφικτές λύσεις σήμερα. Συνεπώς, πρέπει να χρησιμοποιήσουμε άλλες μεθόδους εκτός της απλής απαρίθμησης [11].

**NP-Hard**

**ΠΡΟΒΛΗΜΑΤΑ**

**NP-Complete**

**P**

**NP**

*Σχήμα 3-2. Τα πιθανά ενδεχόμενα όσον αφορά τη σχέση των κλάσεων P, NP, NP-Hard και NP-Complete.*

Έτσι, η μέθοδος λύσης που υιοθετείται χαρακτηριστικά για να λύσει αυτούς τους τύπους προβλημάτων (προβλήματα 𝑁𝑃 − 𝐻𝑎𝑟𝑑) είναι ευρετική. Ευρετικές είναι οι μέθοδοι λύσης που, σχετικά γρήγορα, μπορούν να βρουν μια εφικτή λύση με λογική ποιότητα, παρόλο που δεν υπάρχει καμία εγγύηση για την ποιότητα της λύσης και εξετάζονται εμπειρικά. Ευρετικές μέθοδοι χρησιμοποιούνται, χαρακτηριστικά, για την επίλυση των πραγματικών προβλημάτων λόγω της ταχύτητάς τους και της δυνατότητάς τους να χειριστούν μεγάλο φάσμα περιπτώσεων [11].

Το πρόβλημα των χρονικών παραθύρων ως περίπτωση προβλήματος δρομολόγησης οχημάτων είναι επίσης 𝑁𝑃 − 𝐻𝑎𝑟𝑑. Σχετικοί ευρετικοί αλγόριθμοι αναπτύσσουν μεθόδους:

* + - καθορίζοντας τη σειρά με την οποία ένα όχημα πρέπει να επισκεφθεί τις τοποθεσίες και
    - προσδιορίζοντας το χρόνο στον οποίο το φορτηγό πρέπει να επισκεφθεί κάθε τοποθεσία.

Σκοπός είναι μία σειρά από διαδρομές ελάχιστου κόστους να ικανοποιεί όλες τις απαιτήσεις με ελάχιστο συνολικό κόστος (ή ελάχιστο χρόνο ολοκλήρωσης ή οποιαδήποτε άλλη παρόμοια αντικειμενική λειτουργία).

## ΚΑΤΗΓΟΡΙΕΣ ΑΛΓΟΡΙΘΜΩΝ

Για ένα δεδομένο πρόβλημα υπάρχουν ποικίλοι αλγόριθμοι επίλυσης, οι οποίοι έχουν διαφορετική αποδοτικότητα. Ο χρόνος και ο χώρος αποτελούν βασικούς παράγοντες μέτρησης της αποδοτικότητας ενός αλγορίθμου.

Η χρονική πολυπλοκότητα ενός αλγορίθμου είναι μια συνάρτηση που υπολογίζει τον όγκο των εισερχόμενων δεδομένων, το πλήθος των διαδικασιών που απαιτούνται για την εύρεση λύσης και κατά συνέπεια το χρόνο που χρειάζεται για την εύρεση λύσης του.

Η χωρική πολυπλοκότητα, αποτελεί μια συνάρτηση που υπολογίζει τον όγκο της μνήμης που απαιτείται για την εκτέλεση του αλγορίθμου [16].

## Αλγόριθμοι πολυωνυμικού χρόνου

Ένας αλγόριθμος θεωρείται αποδοτικός, αν η χρονική πολυπλοκότητα του βρίσκεται εντός πολυωνυμικών ορίων. Αυτό σημαίνει ότι για την επίλυση του συγκεκριμένου αλγορίθμου απαιτείται πλήθος διαδικασιών ίσο με αυτό που θα χρειαζόταν για την επίλυση ενός πολυωνύμου με ίδιο όγκο εισερχόμενων δεδομένων.

Ένας αλγόριθμος που λύνει ένα πρόβλημα, θεωρείται πολυωνυμικού χρόνου εάν, υπάρχει ένα πολυώνυμο 𝑞*,* τέτοιο ώστε για οποιοδήποτε παράδειγμα 𝐼 του προβλήματος, ο αριθμός των διαδικασιών που απαιτούνται από τον αλγόριθμο δεν μπορούν να ξεπεράσουν σε πολυπλοκότητα τον αριθμό 𝑂 (𝑞 (|𝐼|𝑏)) όπου |𝐼|𝑏 είναι το μήκος των εισερχόμενων δεδομένων για το πολυώνυμο q [16].

## Εκθετικοί αλγόριθμοι

Ένας αλγόριθμος που δεν είναι πολυωνυμικού χρόνου καλείται εκθετικός αλγόριθμος. Η χρονική πολυπλοκότητα ενός εκθετικού αλγορίθμου είναι είτε μια εκθετική συνάρτηση, είτε αυξάνει με το ρυθμό που αυξάνει μια εκθετική συνάρτηση. Δεδομένου ότι οι εκθετικές συναρτήσεις αυξάνουν γρηγορότερα από τις πολυωνυμικές, οι αλγόριθμοι

εκθετικού χρόνου δεν θεωρούνται αποδοτικοί [16].

## Ακριβείς αλγόριθμοι

Οι λύσεις ενός αλγορίθμου μπορεί να είναι είτε ακριβείς είτε όχι. Για παράδειγμα εάν σε ένα πρόβλημα παραλαβής και παράδοσης υπάρχουν διαφορετικές διαδρομές για ένα σύνολο αιτημάτων μεταφοράς, η διαδρομή που ικανοποιεί όλους τους περιορισμούς και ταυτόχρονα επιτυγχάνει την καλύτερη τιμή της αντικειμενικής συνάρτησης θεωρείται ακριβής ή βέλτιστη, εφόσον μπορεί να αποδειχθεί ότι δεν μπορεί να βρεθεί καλύτερη λύση. Ωστόσο, επειδή τα προβλήματα παραλαβής και παράδοσης που έχουν αξιοσημείωτη πρακτική εφαρμογή και χρησιμοποιούν ρεαλιστικούς περιορισμούς ανήκουν στην κατηγορία 𝑁𝑃 − 𝐻𝑎𝑟𝑑ς δεν μπορούν να βρεθούν βέλτιστες λύσεις.

Αντίθετα, η επίλυση ενός προβλήματος παραλαβής και παράδοσης που ικανοποιεί τους περιορισμούς αλλά επιφέρει μεγαλύτερο κόστος από το βέλτιστο δυνατό καλείται εφικτή. Για τα προβλήματα που δεν μπορούν να λυθούν εντός πολυωνυμικού χρόνου, υπάρχουν ευρετικοί αλγόριθμοι που παράγουν σχεδόν βέλτιστες λύσεις (near optimal solutions) [16].

## Ευρετικοί αλγόριθμοι

Ευρετικός αλγόριθμος είναι μια υπολογιστική τεχνική η οποία, δίχως να εξασφαλίζει την εύρεση της βέλτιστης λύσης, εντοπίζει ικανοποιητικές λύσεις σε αποδεκτό υπολογιστικό χρόνο. Κατά κανόνα μάλιστα δεν μπορεί να εκτιμηθεί πόσο κοντά στη βέλτιστη είναι μια τέτοια λύση. Δεδομένου ότι για το ίδιο πρόβλημα προτείνονται διάφοροι ευρετικοί αλγόριθμοι, τίθεται θέμα αξιολόγησής τους τόσο ως προς την ποιότητα των λύσεων που παρέχουν, όσο και ως προς τον απαιτούμενο υπολογιστικό χρόνο για τον εντοπισμό τους.

Δηλαδή, ένας αλγόριθμος καλείται ευρετικός, εάν δεν είναι γνωστή η χειρότερη περίπτωση αναλογίας του. Για παράδειγμα, όταν δεν μπορεί να προβλεφθεί εξ αρχής η αντίδραση του αλγορίθμου για όλες τις περιπτώσεις ενός δεδομένου προβλήματος. Η αποδοτικότητα ενός ευρετικού αλγορίθμου μπορεί να υπολογισθεί με τη βοήθεια μιας υπολογιστικής δοκιμής. Στη δοκιμή αυτή οι λύσεις του ευρετικού αλγορίθμου συγκρίνονται με εκείνες που εξάγει ένας ακριβής αλγόριθμος, με σκοπό την αξιολόγηση της αποδοτικότητάς τους [16].

Η αξιολόγηση μιας ευρετικής μεθόδου πρέπει να απαντά σε τρία βασικά ερωτήματα:

* + - Αν θέτει ρεαλιστικές παραδοχές.
    - Αν είναι αρκετά σύνθετη ώστε να λαμβάνει υπόψη την αρχιτεκτονική του συστήματος.
    - Αν η πολυπλοκότητά της επιτρέπει τη χρήση της σε μεγάλα γραφήματα διεργασιών.

## Άπληστοι (greedy) αλγόριθμοι

Οι άπληστοι αλγόριθμοι ανήκουν στην κατηγορία ευρετικών αλγορίθμων. Άπληστος αλγόριθμος καλείται ο αλγόριθμος που εκτελεί επαναληπτικά μια διαδικασία και επιλέγοντας περιοδικά μια τοπικά βέλτιστη λύση, οδηγείται τελικά στη λύση του προβλήματος. Σε ορισμένες περιπτώσεις, η παραπάνω μεθοδολογία οδηγεί στην εύρεση ικανοποιητικών προσεγγιστικών λύσεων. Ωστόσο, οι άπληστοι αλγόριθμοι δεν καταλήγουν πάντα σε ικανοποιητικές λύσεις [16].

Ο όρος τοπική αναζήτηση (local search), αναφέρεται σε μια γενική προσέγγιση που στοχεύει στην εύρεση μιας βέλτιστης λύσης ενός προβλήματος. Η βασική ιδέα είναι η εξής:

Η τοπική αναζήτηση ξεκινά από μια μερική λύση 𝑠0 και ψάχνει στην κοντινή περιοχή N (𝑠0) για κάποια καλύτερη. Βασικά, θεωρείται ότι όλο το σύνολο 𝑁 περιλαμβάνει μόνο εφικτές και ολοκληρωμένες λύσεις, οι οποίες πλησιάζουν αρκετά τη δομή της λύσης 𝑠0 [16].

Ένας άπληστος αλγόριθμος (greedy algorithm) διαθέτει γενικά μία απλή δομή που αποτελείται από τα εξής στοιχεία [13].

* + - Ένα σύνολο υποψηφίων επιλογών (π.χ. οι κορυφές ενός γραφήματος).
    - Ένα σύνολο επιλογών που έχουν ήδη χρησιμοποιηθεί.
    - Μία συνάρτηση ελέγχου, που απαντά στο ερώτημα αν ένα συγκεκριμένο σύνολο υποψηφίων αποδίδει μία λύση (όχι απαραίτητα τη βέλτιστη για τη στιγμή που εξετάζεται).
    - Μία συνάρτηση που ελέγχει αν ένα σύνολο υποψηφίων επιλογών είναι εφικτό, με την έννοια ότι μπορεί αυτό να συμπληρωθεί με τέτοιο τρόπο, ώστε να δώσει μία λύση στο πρόβλημα.
    - Μία συνάρτηση επιλογής, που ανά πάσα στιγμή δείχνει ποια επιλογή έχει την καλύτερη προοπτική για να είναι μέρος της λύσης του προβλήματος.
    - Μία αντικειμενική συνάρτηση, που δίνει την τιμή της λύσης: είναι αυτή που επιθυμούμε να βελτιστοποιήσουμε.

Οι άπληστοι αλγόριθμοι εφαρμόζονται συνήθως για την επίλυση προβλημάτων βελτιστοποίησης, όπως π.χ. η εύρεση του συντομότερου μονοπατιού μεταξύ δύο κορυφών ενός γραφήματος ή η εύρεση της βέλτιστης σειράς για την εκτέλεση από έναν υπολογιστή ενός συνόλου έργων.

Τα προβλήματα βελτιστοποίησης λύνονται με μια σειρά επιλογών που είναι: εφικτές, τοπικά βέλτιστες, αμετάκλητες. Αξίζει να σημειωθεί όμως, ότι δεν λύνονται όλα τα προβλήματα βελτιστοποίησης με αυτόν τον τρόπο.

Έτσι, ένας άπληστος αλγόριθμος προχωρά στο επόμενο βήμα με την απόφαση που εκείνη τη στιγμή φαίνεται να είναι η καλύτερη για την επίλυση του προβλήματος. Αυτό, όμως, δεν σημαίνει ότι αποδίδει πάντα τη βέλτιστη λύση.

## ΤΕΧΝΙΚΕΣ ΕΠΙΛΥΣΗΣ

Το πρόβλημα της δρομολόγησης στόλου οχημάτων αποτελεί τη γενίκευση του προβλήματος του Περιοδεύοντος Πωλητή (Traveling Salesman Problem - TSP) το οποίο είναι γνωστό ότι ανήκει στην κατηγορία των NP-Ηard προβλημάτων. Αυτό σημαίνει ότι η ακριβής λύση του προβλήματος μπορεί να υπολογιστεί για μικρές τιμές του 𝑛 (30-40 πελάτες). Για αυτό το λόγο, στη βιβλιογραφία έχουν προταθεί πολλές ευρετικές (heuristics) και μεταευρετικές (metaheuristics) προσεγγίσεις που στην πράξη δίνουν πολύ καλά αποτελέσματα. Από την άλλη πλευρά οι προσεγγίσεις αυτές δεν δίνουν κανένα άνω φράγμα για την πολυπλοκότητα χειρότερης περίπτωσής τους. Οι προσεγγίσεις αυτές μπορούν να λύσουν στιγμιότυπα με 1000 πελάτες σε μερικά δευτερόλεπτα σε ένα σύγχρονο υπολογιστικό σύστημα.

Οι ακριβείς λύσεις υπολογίζονται με τις τεχνικές της Διακλάδωσης και Φραγής (Branch and Bound) και της Διακλάδωσης και Αποκοπής (Branch and Cut). Οι δύο αυτές τεχνικές δημιουργούν ένα δέντρο αναζήτησης. Η Διακλάδωση και Φραγή φτιάχνει ένα μέρος του δέντρου αναζήτησης και το εξερευνά. Η Διακλάδωση και Αποκοπή κόβει κλαδιά του δέντρου με κάποιο κριτήριο για να ελαττώσει το χώρο αναζήτησης. Μία από τις πιο επιτυχείς ακριβείς προσεγγίσεις για τη δρομολόγηση στόλου οχημάτων με χωρητικότητες είναι η μέθοδος k−trees [31]. Χρησιμοποιεί την τεχνική της Διακλάδωσης και Φραγής χωρίζοντας πρώτα τους πελάτες σε ομάδες και μπορεί να επεκταθεί ώστε να λαμβάνει υπόψη της παράθυρα χρόνου και ετερογενείς στόλους οχημάτων.

Στη βιβλιογραφία, υπάρχει μια μεγάλη συλλογή από προσεγγιστικές λύσεις για το πρόβλημα της δρομολόγησης στόλου οχημάτων. Σχεδόν όλες ανήκουν στην κατηγορία των ευρετικών ή μεταευρετικών λύσεων εξαιτίας της πολυπλοκότητας του προβλήματος. Μπορούν να κατηγοριοποιηθούν σε κατασκευαστικές προσεγγίσεις και σε βελτιωτικές

προσεγγίσεις. Όπως μπορεί να γίνει εύκολα αντιληπτό από το όνομα τους, οι κατασκευαστικές προσεγγίσεις κατασκευάζουν αρχικές διαδρομές ή υποψήφιες διαδρομές. Στη συνέχεια οι διαδρομές αυτές βελτιώνονται περαιτέρω από τις βελτιωτικές προσεγγίσεις. Οι κυριότερες τεχνικές που έχουν προταθεί στην βιβλιογραφία ανήκουν σε μία από τις παρακάτω κατηγορίες [36]:

* + - **Βελτιστοποίηση με Αποικία Μυρμηγκιών (Ant Colony Optimization - ACO).** Είναι μια πιθανοτική τεχνική εμπνευσμένη από τη συμπεριφορά των μυρμηγκιών προς αναζήτηση τροφής. Αρχικά, τα μυρμήγκια περιπλανώνται τυχαία προς αναζήτηση τροφής. Όταν τη βρουν, επιστρέφουν στη φωλιά τους αφήνοντας ίχνη κατά μήκος της διαδρομής. Αν κάποιο άλλο μυρμήγκι βρει την ίδια διαδρομή την ενισχύει και αυτό, αφήνοντας το δικό του ίχνος. Τελικά, κατασκευάζεται μία διαδρομή από τη φωλιά προς την τροφή. Στη δρομολόγηση στόλου οχημάτων ακολουθείται παρόμοια λογική ενισχύοντας καλές διαδρομές έναντι των αρχικών τυχαίων.
    - **Γενετικοί Αλγόριθμοι (Genetic Algorithms).** Η ιδέα των γενετικών αλγορίθμων βασίζεται στη φυσική επιλογή. Ένας γενετικός αλγόριθμος εξελίσσει έναν πληθυσμό δημιουργώντας νέες γενιές απογόνων μέσα από μια επαναληπτική διαδικασία μέχρι να ικανοποιηθούν κάποια κριτήρια σύγκλισης. Τέτοια κριτήρια μπορεί να είναι για παράδειγμα ο μέγιστος αριθμός γενεών, μία βέλτιστη λύση ή η δημιουργία ενός ομογενούς πληθυσμού. Υπάρχουν τρεις φάσεις που εκτελούνται σε κάθε γενετικό αλγόριθμο. Αρχικά, εκτελείται η φάση επιλογής στην οποία επιλέγονται (αρχικά τυχαία) οι γονείς που θα δημιουργήσουν τους απογόνους. Στη συνέχεια, εκτελείται η φάση της αναπαραγωγής στην οποία επιλέγονται τα γονίδια των γονέων για να παραχθούν οι απόγονοι που θα αποτελέσουν την επόμενη γενιά. Στην τρίτη φάση γίνεται η μετάλλαξη κάποιων γονιδίων προκειμένου να επιτευχθεί γενετική ποικιλομορφία και να εξερευνηθεί περαιτέρω ο χώρος αναζήτησης του προβλήματος.
    - **Αναζήτηση Ταμπού (Tabu Search).** Η Αναζήτηση Ταμπού εξερευνά το χώρο αναζήτησης κινούμενη σε κάθε επανάληψη από μια λύση 𝑠 στην καλύτερη λύση σε ένα υποσύνολο της γειτονιάς της 𝑁(𝑠). Σε αντίθεση με άλλες μεθόδους αναζήτησης, η τρέχουσα λύση μπορεί να χειροτερεύει από μια επανάληψη στην άλλη. Για να αποφευχθεί η επιλογή λύσεων που οδηγούν σε κυκλική συμπεριφορά, οι λύσεις που έχουν κάποια χαρακτηριστικά πρόσφατα εξερευνημένων λύσεων κηρύσσονται προσωρινά απαγορευμένες (ταμπού). Η διάρκεια κατά την οποία μερικές λύσεις παραμένουν ταμπού ποικίλει. Η

κατάσταση ταμπού μπορεί να ξεπεραστεί αν συμβαίνει για παράδειγμα κάποια από τις ταμπού λύσεις να είναι καλύτερη από όλες τις προηγούμενες λύσεις [36].

* + - **Προγραμματισμός με Περιορισμούς (Constraint Programming).** Στην τεχνική αυτή θέλουμε να ελαχιστοποιήσουμε μία συνάρτηση κόστους λαμβάνοντας υπόψη κάποιους περιορισμούς. Ελαχιστοποιούμε για παράδειγμα τη συνολική απόσταση που διένυσαν τα οχήματα, υπό τους περιορισμούς να μην ξεπεραστεί η μέγιστη χωρητικότητα κάθε οχήματος, να εξυπηρετηθούν όλοι οι πελάτες και τα οχήματα να επιστρέψουν στην αποθήκη. Πολλές φορές οι περιορισμοί που επιβάλλουμε μπορεί να είναι μη γραμμικές συναρτήσεις, αυξάνοντας έτσι την πολυπλοκότητα του προβλήματος. Όταν υπάρχουν μόνο γραμμικοί περιορισμοί και η συνάρτηση ελαχιστοποίησης είναι επίσης γραμμική, έχουμε ένα πρόβλημα Γραμμικού Προγραμματισμού.
    - **Προσομοιωμένη Ανόπτηση (Simulated Annealing).** Πρόκειται για μια στοχαστική τεχνική χαλάρωσης, που χρησιμοποιείται στη στατιστική μηχανική. Βασίζεται στην αναλογία της αναδιάταξης των στερεών σωμάτων. Στην αρχή, ένα στερεό σώμα θερμαίνεται σε υψηλή θερμοκρασία και σταδιακά ψύχεται για να κρυσταλλοποιηθεί. Στην επιστήμη της πληροφορικής χρησιμοποιείται μια μέθοδος τοπικής αναζήτησης και τροποποιείται για να μπορεί να αποδράσεις από τοπικά βέλτιστα. Μία λύση 𝑆′ αποτελεί μια νέα τρέχουσα λύση αν 𝛥 ≤ 0, όπου

𝛥 = 𝑓(𝑥) − 𝑓(𝑥𝑖). Για αποφύγει η αναζήτηση τοπικά βέλτιστα, γίνονται κινήσεις που αυξάνουν τη συνάρτηση κόστους με πιθανότητα 𝑒−𝛥/𝛵, όπου 𝑇 είναι μία παράμετρος που ονομάζεται θερμοκρασία. Η τιμή της παραμέτρου 𝑇 μεταβάλλεται από μια αρχική υψηλή τιμή σε μία τιμή κοντά στο 0. Η μεταβολή ελέγχεται από ένα πρόγραμμα ψύξης που καθορίζει την αρχική τιμή της T και τη μεταβολή της σε κάθε βήμα. Παρόμοια λογική ακολουθεί και η μέθοδος της **Αιτιοκρατικής Ανόπτησης (Deterministic Annealing)** μόνο που σε αυτή τη τεχνική η επιλογή κάθε κίνησης γίνεται με αιτιοκρατικό και όχι με πιθανοτικό τρόπο [36].

Εκτός από τις ευρετικές και μεταευρετικές λύσεις στη βιβλιογραφία έχουν προταθεί και αλγόριθμοι που αποτελούνται από δύο φάσεις (2-Phase Approaches). Η γενική ιδέα των αλγορίθμων αυτών είναι πρώτον η συσταδοποίηση των κορυφών σε ομάδες και δεύτερον η κατασκευή των διαδρομών σε κάθε ομάδα - συστάδα. Η ερευνητική εργασία των Fisher και Jaikumar [2] αποτελεί ένα παράδειγμα τέτοιας τεχνικής. Οι αλγόριθμοι χωρίζονται σε δύο κατηγορίες :

* + - **Συσταδοποίησης Πρώτα, δρομολόγησης στη Συνέχεια (Cluster-First, Route- Second).** Σε αυτήν την τεχνική, πρώτα δημιουργούνται συστάδες πελατών με κοινά χαρακτηριστικά και στη συνέχεια υπολογίζονται οι διαδρομές που θα τους εξυπηρετήσουν. Σε έρευνες που έχουν γίνει, οι συγγραφείς τείνουν να δημιουργούν συστάδες με τον ίδιο περίπου αριθμό πελατών χρησιμοποιώντας τον αλγόριθμο k−means και στη συνέχεια να πραγματοποιούν τη δρομολόγηση [3].
    - **Δρομολόγησης Πρώτα, Συσταδοποίησης στη Συνέχεια (Route First Cluster Second).** Σε αυτήν την τεχνική πρώτα επιλύεται το πρόβλημα του Περιοδεύοντος Πωλητή δημιουργώντας μία γιγαντιαία διαδρομή. Στη συνέχεια, η γιγαντιαία διαδρομή κόβεται σε μικρότερες που ικανοποιούν τους περιορισμούς χωρητικότητας των οχημάτων.

## ΜΕΘΟΔΟΣ CLUSTER FIRST, ROUTE SECOND

Για την επίλυση του προβλήματος δρομολόγησης οχημάτων στη βιβλιογραφία έχουν προταθεί πολλές ευρετικές (heuristics) μέθοδοι, οι οποίες αναπτύχθηκαν κυρίως μεταξύ του 1960 και του 1990. Ενώ οι μεταευρετικές (metaheuristics) προσεγγίσεις, που δίνουν πολύ καλά αποτελέσματα στην πράξη, έχουν αναλυθεί εκτενέστερα μετά το 1990. Μία ευρέως γνωστή τεχνική είναι η μέθοδος αποταμίευσης (savings method) των Clarke και Wright η οποία εφαρμόστηκε σε προβλήματα όπου ο αριθμός των οχημάτων δεν ήταν σταθερός [33]. Ο κύριος στόχος της μεθόδου Cluster First, Route Second είναι να επιμεριστεί το πρόβλημα σε μικρότερα, για το λόγο αυτό χωρίζεται το αρχικό σύνολο που περιλαμβάνει όλους τους κόμβους-πελάτες στο επίπεδο των συστάδων. Αυτό σημαίνει πως για κάθε συστάδα μεμονωμένοι πελάτες του συνολικού δικτύου μπορεί να παραλείπονται, ενώ άλλοι να κατανέμονται προς εξυπηρέτηση σε κάποιο διαθέσιμο όχημα [37].

Έχουν προταθεί διάφορες βελτιώσεις στον αλγόριθμο των Clarke και Wright,, οι οποίες αποσκοπούσαν στη μείωση του υπολογιστικού χρόνου καθώς και των απαιτήσεων της μνήμης. H ευρετική προσέγγιση με εισαγωγές (insertion heuristics) είναι μια ευρέως γνωστή μέθοδος. Οι Mole και Jameson πρότειναν μια μέθοδο διεύρυνσης μιας διαδρομής κάθε φορά, και ο Christofides εφάρμοσε με τη σειρά του, μία διαδοχική και παράλληλη διαδικασία κατασκευής διαδρομών. Εισήγαγε έναν αλγόριθμο Ομαδοποίησης Πρώτα και δρομολόγησης στη Συνέχεια (Cluster-First, Route-Second) όπου αρχικά σχηματίστηκαν εφικτές συστάδες περιστρεφόμενες ακτινικά γύρω από μία κεντροποιημένη αποθήκη και στη συνέχεια καθορίστηκε η διαδρομή για κάθε όχημα επιλύοντας το πρόβλημα του περιοδεύοντος πωλητή (TSP) [34].

Οι Fisher και Jaikumar προσπάθησαν επίσης να επιλύσουν το VRP με έναν αλγόριθμο

Cluster-First, Route-Second. Διέταξαν ένα πρόβλημα γενικευμένης ανάθεσης (Generalized Assignment Problem), αντί να χρησιμοποιήσουν μια γεωμετρική μέθοδο για να σχηματίσουν τις ομάδες (clusters). Ακόμη οι Bramel και Simchi-Levi περιέγραψαν μια ευρετική μέθοδο δύο φάσεων, όπου οι διαδρομές καθορίστηκαν με την επίλυση προβλημάτων χωρητικότητας. Οι υπόλοιπες κορυφές που έμειναν εκτός από την αρχική διαδικασία συμπεριλήφθηκαν σταδιακά στη διαδρομή σε ένα δεύτερο στάδιο [34].

## ΑΝΑΛΥΣΗ ΤΩΝ ΣΥΣΤΑΔΩΝ (CLUSTERS)

Η ανάλυση των συμπλεγμάτων χωρίζει τα δεδομένα σε ομάδες ανάλογα με τη χρησιμότητα και τη σημαντικότητά τους. Εάν η ανάλυση αποσκοπεί στη σημαντικότητα των ομάδων δεδομένων τότε δίνεται έμφαση στην καταγραφή της φυσικής δομής των δεδομένων. Σε ορισμένες περιπτώσεις ωστόσο, η ανάλυση των συμπλεγμάτων αποτελεί ένα χρήσιμο σημείο εκκίνησης για άλλους υπολογισμούς, όπως για παράδειγμα η σύνοψη δεδομένων. Είτε πρόκειται για την κατανόηση είτε για τη χρησιμότητα των δεδομένων, η εν λόγω ανάλυση έχει παίξει πολύ σημαντικό ρόλο σε ένα ευρύ φάσμα τομέων, όπως η ψυχολογία και άλλες κοινωνικές επιστήμες, και στη βιολογία, τη στατιστική, την αναγνώριση προτύπων, την ανάκτηση πληροφοριών, τη μηχανική μάθηση (machine learning) και την εξόρυξη δεδομένων (data mining).

Έχουν υπάρξει πολλές εφαρμογές ανάλυσης συστάδων σε πρακτικά προβλήματα. Παρατίθενται δύο συγκεκριμένα παραδείγματα εφαρμογής της ανάλυσης των συστάδων αποσκοπώντας στην κατανόηση και τη χρησιμότητα των δεδομένων [9].

**Ομαδοποίηση για την κατανόηση κλάσεων**, ή εννοιολογικά σημαντικών ομάδων αντικειμένων που έχουν κοινά χαρακτηριστικά παίζουν σημαντικό ρόλο στον τρόπο που οι άνθρωποι αναλύουν και περιγράφουν τον κόσμο. Πράγματι, ο άνθρωπος είναι εξειδικευμένος στη διαίρεση αντικειμένων σε ομάδες και στην ανάθεση συγκεκριμένων αντικειμένων σε αυτές τις ομάδες (classification - ταξινόμηση). Για παράδειγμα, ακόμα και σχετικά μικρά παιδιά μπορούν γρήγορα να επισημάνουν τα αντικείμενα σε μια φωτογραφία όπως τα κτίρια, τα οχήματα, οι άνθρωποι, τα ζώα, τα φυτά κλπ. Στο πλαίσιο της κατανόησης των δεδομένων, οι συστάδες είναι πιθανές τάξεις και η ανάλυσή τους είναι η μελέτη των τεχνικών αυτόματης εύρεσης τάξεων. Για παράδειγμα οι επιχειρήσεις συλλέγουν μεγάλες ποσότητες πληροφοριών σχετικά με τους τρέχοντες και τους δυνητικούς πελάτες. Η ομαδοποίηση μπορεί να χρησιμοποιηθεί για να χωρίσει τους πελάτες σε μικρότερο αριθμό ομάδων για πρόσθετες δραστηριότητες ανάλυσης και μάρκετινγκ.

**Η ομαδοποίηση σχετικά με την ανάλυση βάσει της χρησιμότητας**, παρέχει την αφαίρεση μεμονωμένων στοιχείων των δεδομένων, στις συστάδες στις οποίες βρίσκονται

τα αντικείμενα αυτών των δεδομένων. Επιπρόσθετα, ορισμένες τεχνικές ομαδοποίησης χαρακτηρίζουν κάθε ομάδα που δημιουργείται από την άποψη μιας πρωτότυπης συστάδας, δηλαδή αν ένα αντικείμενο εκ των δεδομένων είναι αντιπροσωπευτικό των άλλων αντικειμένων στη συγκεκριμένη ομάδα δεδομένων. Αυτές οι πρωτότυπες συστάδες μπορούν να χρησιμοποιηθούν ως βάση για μια σειρά τεχνικών ανάλυσης ή επεξεργασίας δεδομένων. Επομένως, στο πλαίσιο της χρησιμότητας, η ανάλυση μιας συστάδας είναι η μελέτη τεχνικών για την εύρεση των πιο αντιπροσωπευτικών πρωτοτύπων σε αυτό το αντικειμένων. Ένα χαρακτηριστικό παράδειγμα αποτελεί η εύρεση των πλησιέστερων γειτόνων [9].

Η εύρεση πλησιέστερων γειτόνων μπορεί να απαιτεί υπολογισμό της απόστασης ανά ζεύγος μεταξύ όλων των σημείων. Συχνά οι συστάδες και τα πρωτότυπα αντικείμενά τους, μπορούν να βρεθούν πολύ πιο αποτελεσματικά. Εάν τα αντικείμενα είναι σχετικά κοντά στην πρωτότυπη συστάδα τους, τότε μπορούμε να χρησιμοποιήσουμε τα πρωτότυπα για να μειώσουμε τον αριθμό των υπολογισμών απόστασης που είναι απαραίτητοι για να βρούμε τους πλησιέστερους γείτονες ενός αντικειμένου. Εάν δύο πρωτότυπες συστάδες είναι πολύ απομακρυσμένες, τότε τα αντικείμενα στα αντίστοιχα σύνολα δεν μπορούν να είναι πλησιέστεροι γείτονες μεταξύ τους. Συνεπώς, για να βρούμε τους πλησιέστερους γείτονες ενός αντικειμένου, είναι απαραίτητο να υπολογίσουμε μόνο την απόσταση από αντικείμενα σε κοντινά σύνολα, όπου η απόσταση των δύο συστάδων μετράται από την απόσταση μεταξύ των πρωτοτύπων αντικειμένων τους.

Ακολουθεί μια αναφορά στο απαραίτητο θεωρητικό υπόβαθρο, προσδιορίζοντας περαιτέρω στοιχεία σχετικά με την ανάλυση μιας συστάδας. Συγκρίνεται η μέθοδος της ομαδοποίησης (clustering) με άλλες τεχνικές ομαδοποίησης δεδομένων και πραγματοποιείται διερεύνηση δύο σημαντικών θεμάτων. Αρχικά διερευνώνται οι διαφορετικοί τρόποι ομαδοποίησης σε ένα σύνολο αντικειμένων και πραγματοποιείται ανάλυση σχετικά με τους διαφορετικούς τύπους συστάδων που συναντώνται [9].

## ΤΙ ΕΙΝΑΙ ΜΙΑ ΣΥΣΤΑΔΑ;

Η ανάλυση μιας συστάδας ομαδοποιεί αντικείμενα δεδομένων που βασίζονται μόνο σε πληροφορίες που βρίσκονται στα δεδομένα που περιγράφουν τα αντικείμενα και τις σχέσεις τους. Ο στόχος είναι τα αντικείμενα μιας ομάδας να είναι παρόμοια (ή να σχετίζονται) μεταξύ τους και διαφορετικά από αυτά (ή να μην σχετίζονται) τα αντικείμενα σε άλλες ομάδες. Όσο μεγαλύτερη είναι η ομοιότητα (ή ομοιογένεια) μέσα σε μια ομάδα τόσο μεγαλύτερη είναι και η διαφορά μεταξύ των ομάδων, τόσο καλύτερη ή πιο ξεχωριστή είναι η ομαδοποίηση.

Σε πολλές εφαρμογές, η έννοια της συστάδας δεν είναι καλά καθορισμένη. Αυτό γίνεται πιο κατανοητό αν εξετάσουμε το *Σχήμα 3-3* το οποίο δείχνει είκοσι σημεία και τρεις διαφορετικούς τρόπους διαίρεσής τους σε ομάδες. Τα σχήματα των δεικτών υποδηλώνουν την ένταξή τους σε διαφορετική συστάδα. Τα στοιχεία στο (β) και (δ) χωρίζουν τα δεδομένα σε δύο και έξι μέρη, αντίστοιχα. Ωστόσο, η φαινομενική κατανομή των δύο μεγαλύτερων ομάδων σε τρεις υποκλάσεις μπορεί απλά να είναι ένα τεχνούργημα του ανθρώπινου οπτικού συστήματος. Επίσης, μπορεί να μην είναι παράλογο να πούμε ότι τα σημεία σχηματίζουν τέσσερις ομάδες, όπως φαίνεται στο (γ). Αυτό δείχνει ότι ο ορισμός ενός συμπλέγματος είναι ασαφής και ότι ο καλύτερος ορισμός εξαρτάται από τη φύση των δεδομένων και τα επιθυμητά αποτελέσματα [9].



(α) Αρχικά Σημεία (β) Δύο Ομάδες



(γ) Τέσσερις Ομάδες (δ) Έξι Ομάδες

*Σχήμα 3-3. Διαφορετικοί τρόποι ομαδοποίησης του ίδιου συνόλου σημείων.[9]*

Η ανάλυση μιας συστάδας σχετίζεται και με άλλες τεχνικές που χρησιμοποιούνται για τη διαίρεση αντικειμένων δεδομένων σε ομάδες. Για παράδειγμα, η ομαδοποίηση μπορεί να θεωρηθεί ως μια μορφή ταξινόμησης των αντικειμένων ως προς το ότι δημιουργεί μια επισήμανση των αντικειμένων με ετικέτες ανάλογα με την κάθε ομάδα στην οποία κατατάσσονται. Ωστόσο, αυτές οι ετικέτες προέρχονται μόνο από τα δεδομένα. Για τον λόγο αυτό, η ανάλυση των συστάδων αναφέρεται μερικές φορές ως ταξινόμηση χωρίς την απαιτούμενη κατηγοριοποίηση σε περιπτώσεις που δεν παρέχονται σαφή δεδομένα ή η ομαδοποίηση βασίζεται σε δεδομένα η φύση των οποίων είτε αμφισβητείται είτε δεν μας παρέχει τα επιθυμητά αποτελέσματα. Όταν όμως ο όρος ταξινόμηση χρησιμοποιείται χωρίς κάποιον εξειδικευμένο όρο σε ένα καλά ορισμένο και σαφές σύνολο, όπως για παράδειγμα στο πλαίσιο της εξόρυξης δεδομένων (data mining), εκεί τυπικά αναφέρεται ως εποπτευόμενη ταξινόμηση.

Ακόμη οι όροι **segmentation** (χωρισμός σε τμήματα) και **partitioning** (διαμέριση) χρησιμοποιούνται μερικές φορές ως συνώνυμα για την ομαδοποίηση. Αυτοί οι όροι

χρησιμοποιούνται πιο συχνά για προσεγγίσεις εκτός των παραδοσιακών ορίων μιας ανάλυσης συστάδων. Για παράδειγμα ο όρος *partitioning* χρησιμοποιείται πιο συχνά σε τεχνικές οι οποίες σχετίζονται με διαίρεση γραφημάτων σε υπογραφήματα και δεν συνδέεται άμεσα με την ομαδοποίηση. Ο όρος *segmentation* συχνά αναφέρεται στην κατανομή των δεδομένων σε ομάδες χρησιμοποιώντας όμως απλούστερες τεχνικές, όπως για παράδειγμα μια εικόνα μπορεί να χωριστεί σε τμήματα που βασίζονται μόνο στην ένταση και το χρώμα των pixel ή οι άνθρωποι μπορούν να χωριστούν σε ομάδες με βάση το εισόδημά τους. Παρόλα αυτά, υπάρχουν και κάποιοι οι οποίοι στοχεύουν, με αυτές τις απλούστερες μεθόδους, να επιτύχουν ομαδοποιήσεις αντίστοιχες της ανάλυσης μιας συστάδας [9].

## ΔΙΑΦΟΡΕΤΙΚΟΙ ΤΥΠΟΙ CLUSTER

Η ομαδοποίηση σκοπεύει να βρει χρήσιμες ομάδες αντικειμένων, όπου η χρησιμότητα ορίζεται από τους στόχους της ανάλυσης δεδομένων. Δεν αποτελεί έκπληξη το γεγονός ότι υπάρχουν πολλές διαφορετικές έννοιες για μια ομάδα, που αποδεικνύονται όλες χρήσιμες στην πράξη. Προκειμένου να απεικονιστούν οπτικά οι διαφορές μεταξύ αυτών των τύπων ομάδων, χρησιμοποιούμε σημεία δύο διαστάσεων (όπως φαίνεται και στο *Σχήμα 3-4*) ως αντικείμενα δεδομένων μας. Τονίζουμε, ωστόσο, ότι οι τύποι των συστάδων που περιγράφονται παρακάτω ισχύουν εξίσου και για άλλα είδη δεδομένων [9]. Πιο αναλυτικά οι διαφορετικοί τύποι συστάδων διακρίνονται σε:

* + - **Καλά Διαχωρισμένες (Well Separated).** Μία συστάδα είναι ένα σύνολο αντικειμένων στα οποία κάθε αντικείμενο είναι πλησιέστερο (ή πανομοιότυπο) με κάθε άλλο αντικείμενο της ομάδας παρά με οποιοδήποτε αντικείμενο που δεν ανήκει σε αυτήν. Μερικές φορές χρησιμοποιείται ένα όριο για να καθοριστεί ότι όλα τα αντικείμενα της ομάδας πρέπει να είναι αρκετά κοντά (ή παρόμοια) μεταξύ τους. Αυτός ο ορισμός ικανοποιείται μόνο όταν τα δεδομένα περιέχουν φυσικά σύνολα που απέχουν πολύ το ένα από το άλλο. Το *Σχήμα 3-4* (α) δίνει ένα παράδειγμα καλά διαχωρισμένων συστάδων που αποτελείται από δύο ομάδες σημείων σε ένα δισδιάστατο χώρο. Η απόσταση μεταξύ οποιωνδήποτε δύο σημείων σε διαφορετικές ομάδες είναι μεγαλύτερη από την απόσταση μεταξύ οποιωνδήποτε δύο σημείων μέσα σε μια ομάδα. Οι καλά διαχωρισμένες συστάδες δεν χρειάζεται να είναι σφαιροειδή, αλλά μπορούν να έχουν οποιοδήποτε σχήμα.
    - **Βασισμένες σε Πρωτότυπα (Prototype Based)**. Μία συστάδα είναι ένα σύνολο αντικειμένων στα οποία κάθε αντικείμενο είναι πιο κοντά (πιο παρόμοιο) με το πρωτότυπο που ορίζει την εν λόγω συστάδα από το πρωτότυπο αντικείμενο οποιασδήποτε άλλης συστάδας. Για δεδομένα με συνεχή χαρακτηριστικά, ως

πρωτότυπο αντικείμενο ενός συμπλέγματος συχνά ορίζεται ένα σημείο με μεγαλύτερο κέντρο βάρους (centroid), π.χ. το μέσο ή κάποιο σημείο κοντά στο μέσο όλων των σημείων της συστάδας. Για πολλούς τύπους δεδομένων, το πρωτότυπο μπορεί να θεωρηθεί ως το πιο κεντρικό σημείο. Σε τέτοιες περιπτώσεις συνήθως αναφερόμαστε σε συστάδες που τείνουν να έχουν σφαιρικό σχήμα γύρω από τα σημεία με μεγάλο κέντρο βάρους. Το *Σχήμα 3-4* (β) δείχνει ένα παράδειγμα από τέτοιες ομάδες.

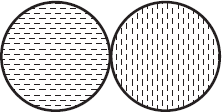
* + - **Βασισμένες σε Γραφήματα (Graph Based).** Αν τα δεδομένα αναπαρίστανται ως γράφημα, όπου οι κόμβοι είναι αντικείμενα και οι ακμές αντιπροσωπεύουν τη σύνδεση μεταξύ των αντικειμένων, τότε μια συστάδα μπορεί να οριστεί ως το σύνολο αυτών των συνδεδεμένων στοιχείων, δηλαδή μια ομάδα αντικειμένων που είναι συνδεδεμένα μεταξύ τους, αλλά που δεν έχουν καμία σχέση με αντικείμενα εκτός της ομάδας. Ένα παράδειγμα από συστάδες που βασίζονται σε γράφημα είναι αυτά που ταξινομούνται με βάση τη γειτνίασή τους, όπου δύο αντικείμενα συνδέονται μεταξύ τους όταν βρίσκονται εντός μιας καθορισμένης απόστασης μεταξύ τους. Αυτό υποδηλώνει πως κάθε αντικείμενο εντός της συγκεκριμένης ομάδας είναι μονίμως πιο κοντά σε κάποιο σημείο που ανήκει σε αυτή παρά σε οποιοδήποτε άλλο σημείο. Το *Σχήμα 3-4* (γ) απεικονίζει ένα τέτοιο παράδειγμα συστάδων. Ο ορισμός μιας τέτοιας συστάδας είναι χρήσιμος όταν οι ομάδες στοιχείων είναι ακανόνιστες ή αλληλένδετες, αλλά μπορεί να παρουσιαστεί πρόβλημα με την εμφάνιση θορύβου (όπως φαίνεται από τις δύο σφαιρικές ομάδες που σχηματίζονται στο *Σχήμα 3-4* (γ), μια μικρή «γέφυρα» μεταξύ των στοιχείων μπορεί να συγχωνεύσει δύο διαφορετικές συστάδες.

Και άλλοι τύποι συστάδων με γραφήματα εμφανίζονται σύμφωνα με μία προσέγγιση με την ονομασία *cliques*. Δηλαδή ένα σύνολο κόμβων σε ένα γράφημα που είναι πλήρως συνδεδεμένοι μεταξύ τους. Πιο συγκεκριμένα, αν προσθέσουμε συνδέσεις μεταξύ των αντικειμένων με τη σειρά και ανάλογα με την απόσταση του ενός από το άλλο, τελικά σχηματίζεται μία συστάδα η οποία έχει τη μορφή σφήνας (clique). Όπως και οι συστάδες οι οποίες βασίζονται σε πρωτότυπα, έτσι και αυτές τα τείνουν να είναι σφαιρικές.

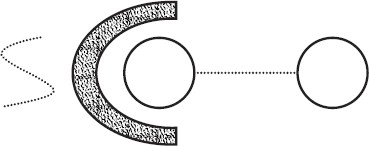
* + - **Βασισμένες στην Πυκνότητα (Density Based).** Μια συστάδα είναι μια πυκνή περιοχή η οποία περιβάλλεται από μια άλλη περιοχή χαμηλότερης πυκνότητας. Το *Σχήμα 3-4* (δ) δείχνει ορισμένες ομάδες με βάση την πυκνότητα οι οποίες δημιουργήθηκαν από την προσθήκη θορύβου στα δεδομένα. Οι δύο κυκλικές ομάδες δεν συγχωνεύονται όπως στο *Σχήμα 3-4* (γ) επειδή η μεταξύ τους

«γέφυρα» εξασθενεί λόγω του θορύβου. Ομοίως και η καμπύλη που υπάρχει στο ίδιο σχήμα εξασθενεί από το θόρυβο. Μια συστάδα με βάση την πυκνότητα χρησιμοποιείται συχνά όταν τα στοιχεία είναι ακανόνιστα ή αλληλένδετα, και όταν υπάρχει θόρυβος και ακραίες τιμές. Στην αντίθετη περίπτωση ύπαρξης συσχέτισης μεταξύ των σημείων, δεν θα λειτουργούσε καλά για τα συγκεκριμένα δεδομένα καθώς ο θόρυβος πιθανότατα θα δημιουργούσε γέφυρες μεταξύ των ομάδων.

* + - **Βασισμένες σε κοινές ιδιότητες (Shared Property – Conceptual Clusters).** Πιο γενικά μπορεί να οριστεί μια συστάδα ως ένα σύνολο αντικειμένων τα οποία μοιράζονται κάποια κοινή ιδιότητα. Ο συγκεκριμένος ορισμός περιλαμβάνει όλους τους προηγούμενους ορισμούς, για παράδειγμα αντικείμενα που ανήκουν σε συγκεκριμένη ομάδα μοιραζόμενα την ιδιότητα ότι είναι πιο κοντά στο ίδιο κέντρο (centroid). Ωστόσο ο συγκεκριμένος τύπος περιλαμβάνει επίσης νέους τύπους ομάδων. Το *Σχήμα 3-4* (ε) απεικονίζει μια τριγωνική περιοχή δίπλα σε μια ορθογώνια και υπάρχουν επίσης δύο συνδεδεμένοι κύκλοι. Και στις δύο περιπτώσεις χρειάζεται ένας αλγόριθμος ομαδοποίησης με μια πολύ συγκεκριμένη ιδέα για να επιτευχθεί η ορθή ανίχνευση αυτών των ομάδων. Η διαδικασία εύρεσης τέτοιων ομάδων ονομάζεται conceptual clustering (ομαδοποίηση με βάση κάποια εννοιολογική προσέγγιση) [9].

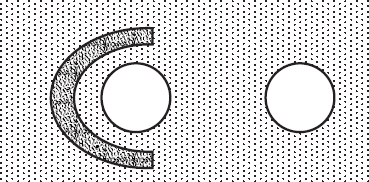
 

(α) Καλά διαχωρισμένες. Κάθε σημείο είναι πιο κοντά σε όλα τα σημεία του ενός cluster από οποιοδήποτε σημείο του άλλου.

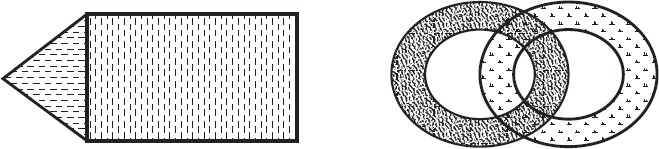


(γ) Βασισμένες στη συνοχή. Κάθε σημείο είναι πιο κοντά σε τουλάχιστον ένα σημείο στο cluster του, παρά σε οποιοδήποτε σημείο ενός άλλου cluster.

(β) Συγκεντρωμένες στο κέντρο. Κάθε σημείο είναι πιο κοντά στο κέντρο του cluster στο οποίο ανήκει απ’ ότι στο κέντρο οποιουδήποτε άλλου cluster.



(δ) Βασισμένες στην πυκνότητα. Οι συστάδες είναι περιοχές με υψηλή πυκνότητα, διαχωρισμένες από περιοχές με χαμηλότερη πυκνότητα.



(ε) Βασισμένες σε κοινές ιδιότητες. Τα σημεία σε ένα cluster έχουν μια κοινή ιδιότητα που προέρχεται από το σύνολο των σημείων του. (Τα σημεία στην τομή των κύκλων ανήκουν και στις δύο ομάδες).

*Σχήμα 3-4.. Διαφορετικοί τύποι συστάδων όπως απεικονίζονται από σύνολα δισδιάστατων σημείων [9].*

Τα παραπάνω ήταν μια εισαγωγική ανάλυση στις συστάδες. Η διαδικασία αρχίζει με μία ανάλυση, η οποία περιέχει διάφορες προσεγγίσεις σχετικά με τη διαίρεση των ομάδων, ανάλογα με τους διαφορετικούς τύπους αντικειμένων και των συνόλων που πρόκειται να δημιουργηθούν. Χρησιμοποιούνται οι ακόλουθες τρεις απλές αλλά σημαντικές τεχνικές ομαδοποίησης για να εισαχθούν πολλές από τις έννοιες που εμπλέκονται στην ανάλυση μιας συστάδας.

* + - **K-means.** Πρόκειται για μια πρωτότυπη τεχνική ομαδοποίησης που προσπαθεί να βρει έναν καθορισμένο από τον χρήστη αριθμό ομάδων (k), που αντιπροσωπεύονται από τα κέντρα τους.
    - **Αθροιστική Ιεραρχική Ομαδοποίηση (Agglomerative Hierarchical Clustering).** Αυτή η προσέγγιση ομαδοποίησης αναφέρεται σε μια συλλογή στενά συνδεδεμένων τεχνικών ομαδοποίησης που παράγουν μια ιεραρχική ομαδοποίηση ξεκινώντας από κάθε σημείο ως μία μοναδική ομάδα. Κατόπιν συγχωνεύονται επανειλημμένα οι δύο πλησιέστερες ομάδες μέχρις ότου παραμείνει μία ενιαία και συμπαγής συστάδα. Ορισμένες από αυτές τις τεχνικές έχουν μια φυσική ερμηνεία όσον αφορά την ομαδοποίηση βάσει γραφημάτων, ενώ άλλες ερμηνεύονται βάσει της προσέγγισης που βασίζεται σε πρωτότυπα.
    - **DBSCAN.** Αυτός είναι ένας αλγόριθμος ομαδοποίησης ο οποίος βασίζεται στην πυκνότητα που παράγεται από ένα μέρος του συνόλου των αντικειμένων, και ο αριθμός των συστάδων καθορίζεται αυτόματα από τον αλγόριθμο. Η ομαδοποίηση με βάση την πυκνότητα εντοπίζει περιοχές υψηλής πυκνότητας που χωρίζονται μεταξύ τους από περιοχές χαμηλής πυκνότητας. Τα σημεία σε περιοχές χαμηλής πυκνότητας ταξινομούνται ως θόρυβος και παραλείπονται. Επομένως, το DBSCAN δεν παράγει πλήρη ομαδοποίηση [9].

## Η ΜΕΘΟΔΟΣ ΟΜΑΔΟΠΟΙΗΣΗΣ K-MEANS

Οι τεχνικές ομαδοποίησης βάσει πρωτότυπων δημιουργούν έναν διαχωρισμό ενός επιπέδου των δεδομένων σύμφωνα με συγκεκριμένες ιδιότητες μέρους των αντικειμένων του συνόλου. Η μέθοδος k-means ορίζει ένα πρωτότυπο, ορίζοντας ένα κεντρικό σημείο, το οποίο συνήθως είναι ο μέσος όρος μιας ομάδας σημείων και τυπικά εφαρμόζεται σε αντικείμενα σε n-διάστατο χώρο. Ένα κεντρικό σημείο (centroid) σχεδόν ποτέ δεν αντιστοιχεί σε ένα πραγματικό σημείο δεδομένων. Σε μια παραλλαγή του k-means που καλείται k-medoid, το κέντρο πρέπει από τον ορισμό του να είναι ένα σημείο πραγματικών δεδομένων το οποίο πρέπει να είναι το πιο αντιπροσωπευτικό σημείο από μια ομάδα σημείων στον χώρο. Σε αυτή την ενότητα θα επικεντρωθούμε αποκλειστικά στα μέσα k, τα οποία είναι ένας από τους παλαιότερους και πιο ευρέως χρησιμοποιούμενους αλγόριθμους ομαδοποίησης [9].

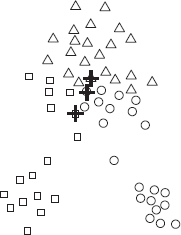
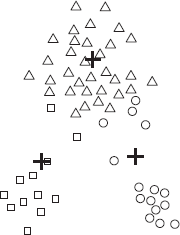
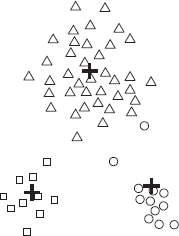
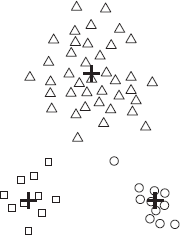
Η τεχνική ομαδοποίησης k-means είναι απλή. Αρχικά επιλέγονται k αρχικά σημεία με μεγαλύτερο κέντρο βάρους (centroids), όπου το k είναι μια συγκεκριμένη παράμετρος, δηλαδή ο αριθμός των επιθυμητών ομάδων προς δημιουργία. Κάθε σημείο στη συνέχεια αντιστοιχίζεται στο πλησιέστερο κέντρο και κάθε σύνολο σημείων που έχουν αντιστοιχισθεί σε ένα κεντροειδές αποτελούν μια συστάδα. Έπειτα το κέντρο της κάθε συστάδας ενημερώνεται βάσει των σημείων που έχουν αντιστοιχισθεί σε αυτήν την συστάδα. Επαναλαμβάνονται τα βήματα εκχώρησης και ενημέρωσης έως ότου κανένα σημείο δεν αλλάξει ομάδα, ή ισοδύναμα, έως ότου τα κεντρικά σημεία παραμείνουν ίδια.

Η λειτουργία του k-means απεικονίζεται στο *Σχήμα 3-6*, το οποίο δείχνει πώς, ξεκινώντας από τρία κεντροειδή, τα τελικά συμπλέγματα δημιουργούνται μετά από τέσσερις επαναλήψεις της διαδικασίας. Σε αυτές τις νέες μορφές που εμφανίζονται παρατηρείται πως σε κάθε βήμα, κάθε υποσύνολο έχει διαφορετικά κεντρικά σημεία στην αρχή της επανάληψης και έπειτα εκχωρούνται τα σημεία σε αυτά. Τα κεντρικά αυτά σημεία σημειώνονται με το σύμβολο "+". Όλα τα σημεία που ανήκουν στην ίδια ομάδα έχουν το ίδιο σχήμα δείκτη [9].

**Αλγόριθμος 1.**Βασικός Αλγόριθμος k-means.

1. Επιλογή K σημείων ως αρχικά κεντροειδή.
2. **Έναρξη Επανάληψης**
3. Σχηματισμός k συστάδων αναθέτοντας κάθε σημείο στο κοντινότερό κεντρικό σημείο.
4. Υπολογισμός εκ νέου του κεντρικού σημείου για κάθε συστάδα.
5. **Μέχρι**: Τα κεντρικά σημεία να μην αλλάξουν τιμή.

Στο πρώτο βήμα όπως φαίνεται και στο *Σχήμα 3-5* (α) τα σημεία αντιστοιχίζονται στα αρχικά κεντροειδή, εκ των οποίων τα περισσότερα είναι στη μεγαλύτερη ομάδα σημείων. Για το παράδειγμα αυτό, χρησιμοποιείται το μέσο ως το κέντρο. Μετά την εκχώρηση των σημείων σε ένα κέντρο, στη συνέχεια το κέντρο επικαιροποιείται σύμφωνα με τα σημεία που είχαν αντιστοιχισθεί σε αυτό το κέντρο. Σε κάθε βήμα φαίνεται το κεντροειδές κατά την έναρξη της διαδικασίας και στη συνέχεια γίνεται η εκχώρηση των σημείων σε αυτά τα κεντροειδή. Στο δεύτερο βήμα, τα σημεία αντιστοιχίζονται στα ενημερωμένα κεντροειδή και τα κεντροειδή ενημερώνονται εκ νέου από τις νέες τιμές. Στα βήματα 2, 3 και 4, όπως φαίνονται στο *Σχήμα 3-5* (β), (γ) και (δ) αντίστοιχα, δύο από τα κεντροειδή μετακινούνται στις δύο μικρές ομάδες σημείων στο κάτω μέρος των σχημάτων. O *Αλγόριθμος k-means* τερματίζεται στο (δ), επειδή δεν πραγματοποιούνται περαιτέρω αλλαγές.

(α) Επανάληψη 1. (β) Επανάληψη 2. (γ) Επανάληψη 3. (δ) Επανάληψη 4.

*Σχήμα 3-5. Δημιουργία 3 συστάδων με τη χρησιμοποίηση του αλγορίθμου k-means[9].*

Για ορισμένους συνδυασμούς λειτουργιών, τύπους δεδομένων και τύπους κεντρικών σημείων ο k-means πάντοτε συγκλίνει σε μία λύση. Δηλαδή φτάνει σε μια κατάσταση στην οποία κανένα σημείο δεν μετατοπίζεται από τη μια ομάδα σε κάποια άλλη και επομένως τα κεντροειδή δεν αλλάζουν. Ωστόσο, επειδή το μεγαλύτερο μέρος της σύγκλισης συμβαίνει στα αρχικά βήματα, η συνθήκη στη γραμμή 5 του *Αλγορίθμου 1* συχνά αντικαθίσταται από μια συνθήκη που τερματίζεται σχετικά πιο εύκολα, όπως για παράδειγμα το να επαναλαμβάνεται η διαδικασία μέχρις ότου μόνο το 1% των σημείων να μεταβάλλει τις συστάδες

Κεφάλαιο 4

**Νέα Ευρετική Προσέγγιση Δύο Φάσεων**

Σε αυτό το κεφάλαιο παρουσιάζεται η προτεινόμενη προσέγγιση δύο φάσεων για το πρόβλημα της δρομολόγησης στόλου οχημάτων με Χρονικά Παράθυρα. Στην πρώτη φάση πραγματοποιείται η δημιουργία ομάδων κόμβων προς εξυπηρέτηση μέσω ενός αλγορίθμου που στοχεύει στην αντιστοίχιση του συνόλου των κόμβων του δικτύου σε κάθε επιμέρους ομάδα ώστε να κατανέμεται ομοιόμορφα το φορτίο σε κάθε περίπτωση. Λεπτομέρειες σχετικά με τη συγκεκριμένη φάση παρουσιάζονται στην Ενότητα 4.1, ενώ στην Ενότητα 4.2 παρουσιάζεται η δεύτερη φάση επίλυσης του προβλήματος που σχετίζεται με τη δρομολόγηση των οχημάτων με χρονικά παράθυρα. Σε αυτή τη φάση προσεγγίζονται πέντε διαφορετικοί τρόποι επίλυσης του προβλήματος δρομολόγησης οχημάτων με χρονικά παράθυρα, ενώ η κάθε περίπτωση επικεντρώνεται σε διαφορετικά χαρακτηριστικά (ελάχιστος χρόνος, ελάχιστη απόσταση, ελάχιστη αναμονή, κλπ.). Έτσι εξάγονται δεδομένα και δείκτες που ανάλογα με τα δεδομένα κάθε προβλήματος, παράγουν μια σειρά από προτάσεις προς εφαρμογή για την επίλυση ή βελτίωση του κάθε υποπροβλήματος.

## ΦΑΣΗ I: ΣΥΣΤΑΔΟΠΟΙΗΣΗ ΒΑΣΙΣΜΕΝΗ ΣΤΗ ΓΕΩΓΡΑΦΙΚΗ ΕΓΓΥΤΗΤΑ ΤΩΝ ΚΟΜΒΩΝ

Η πρώτη φάση είναι υπεύθυνη για τη γεωγραφική διαμέριση της περιοχής εντός του γραφήματος που περιλαμβάνει όλους τους κόμβους σε επιμέρους περιοχές. Χρησιμοποιείται μία τεχνική η οποία προσπαθεί να διαμερίσει μια γεωγραφική περιοχή σε μικρότερα μέρη επιτυγχάνοντας παράλληλα και την ομοιόμορφη κατανομή των φορτίων κάθε περιοχής. Η συγκεκριμένη προσέγγιση, αναφέρεται στη βιβλιογραφία ως πρόβλημα Χωροταξικής Συσταδοποίησης (Capacitated Clustering Problem). Το CCP έχει αποδειχθεί ότι είναι πρόβλημα NP-Hard. Αυτό σημαίνει ότι ο χρόνος λύσης του CCP

αυξάνεται εκθετικά με την αύξηση των σημείων διανομής. Σε περιπτώσεις μικρού μεγέθους δεδομένων αλγόριθμοι που εστιάζουν στη μορφή και τα χαρακτηριστικά κάθε προβλήματος χαρακτηρίζονται πιο κατάλληλοι για την επίτευξη βέλτιστης λύσης σε σύντομο χρονικό διάστημα, όμως όταν το μέγεθος του προβλήματος αυξάνεται, οι ακριβείς μέθοδοι δεν μπορούν να το λύσουν επαρκώς, έτσι χρησιμοποιούνται ευρετικοί και μεταευρετικοί αλγόριθμοι για την επίλυση του προβλήματος. Ως εκ τούτου, όταν η εύρεση βέλτιστης λύσης σε εύλογο χρονικό διάστημα δεν είναι δυνατή, για περιπτώσεις μεγάλου μεγέθους δεδομένων, χρησιμοποιούνται τέτοιες προσεγγίσεις [34].

Για την επίλυση λοιπόν του προβλήματος της παρούσας διπλωματικής ακολουθήθηκε μία βελτιωμένη προσέγγιση του κλασσικού αλγορίθμου k-means. Η εν λόγω βελτίωση χωρίζει ένα σύνολο από 𝑛 στοιχεία (π.χ. παραγγελίες πελατών) σε 𝛫 διακεκριμένες ομάδες με γνωστή χωρητικότητα. Κατά τη διάρκεια της συσταδοποίησης τα αντικείμενα με τις μικρότερες διαδρομές από τα κεντροειδή ομαδοποιούνται μαζί. Το άθροισμα των ομαδοποιημένων αντικειμένων δεν πρέπει να υπερβαίνει την συνολική χωρητικότητα του κάθε Cluster (π.χ. χωρητικότητα φορτηγού). Όλες οι ομάδες που δημιουργούνται έχουν ομοιόμορφη χωρητικότητα. Το CCP είναι ένα πρόβλημα 𝑁𝑃 𝐶𝑜𝑚𝑝𝑙𝑒𝑡𝑒 και αποτελεί μία περίπτωση συνδυαστικού προβλήματος βελτιστοποίησης. Το συνδυαστικό πρόβλημα βελτιστοποίησης μπορεί να θεωρηθεί ότι αναζητά το καλύτερο στοιχείο σε ένα σύνολο διακριτών αντικειμένων, τα οποία μπορούν να λυθούν είτε χρησιμοποιώντας έναν αλγόριθμο αναζήτησης είτε έναν μεταευρετικό αλγόριθμο. Ωστόσο, οι γενικοί αλγόριθμοι αναζήτησης δεν εγγυώνται την εύρεση βέλτιστης λύσης. Πολλοί ευρετικοί αλγόριθμοι διαμορφώνονται για να λύσουν το CCP.

Αυτή η εργασία περιλαμβάνει τη χρήση του πιο γνωστού αλγορίθμου ομαδοποίησης k- means, τροποποιημένο κατάλληλα, ώστε να περιλαμβάνει ένα μέτρο προτεραιότητας των αιτούμενων κόμβων προς εισαγωγή στο κάθε σύμπλεγμα. Το κάθε σημείο ανατίθεται στο πλησιέστερο σύμπλεγμα με βάση τη μέγιστη ζήτηση και την ελάχιστη απόσταση, έτσι ώστε τα σημεία με μεγαλύτερη ζήτηση να αντιστοιχίζονται πρώτα στο σύμπλεγμα και αυτά με μικρότερη ζήτηση να μπορεί εύκολα να ανατεθούν και σε άλλες ομάδες, αν περιορίζονται χωρικά. Εάν τα σημεία ανατίθενται μόνο με βάση την απόσταση, ο αριθμός των ομάδων που επρόκειτο να σχηματιστούν μπορεί να μην είναι ο βέλτιστος, δεδομένου ότι οι κόμβοι με μικρότερη ζήτηση μπορούν να ανατεθούν στο εκάστοτε σύμπλεγμα πριν από κάποιους με μεγαλύτερη ζήτηση. Γεγονός που μπορεί να οδηγήσει στο σχηματισμό πρόσθετης ομάδας για την ικανοποίηση όλων των κόμβων [32].

## 4.1.1 Ο ΟΡΙΣΜΟΣ ΤΟΥ ΠΡΟΒΛΗΜΑΤΟΣ

Το πρόβλημα Χωροταξικής Συσταδοποίησης (Capacitated Clustering Problem)

θεωρείται πως έχει 𝑛 αιτούμενους κόμβους προς εξυπηρέτηση, των οποίων οι απαιτήσεις είναι γνωστές και είναι κατανεμημένοι σε συντεταγμένες (𝑥, 𝑦). Οι 𝑛 αιτούμενοι κόμβοι ομαδοποιούνται έτσι ώστε να σχηματίσουν 𝑘 συστάδες. Κάθε συστάδα έχει

𝑛1, 𝑛2, . . . , 𝑛𝑘 αριθμό αιτούμενων κόμβων με την προϋπόθεση ότι:

𝑘

∑ 𝑛𝑗 = 𝑛

𝑗=1

όπου 𝑛 είναι ο συνολικός αριθμός των αιτούμενων κόμβων. Το πρόβλημα δίνεται με ένα σύνολο στοιχείων τα οποία είναι:

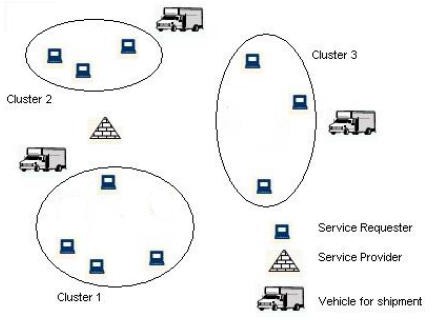
* + - Αιτούμενοι κόμβοι προς εξυπηρέτηση: 𝑟1, 𝑟2, 𝑟3, … , 𝑟𝑛
    - Συντεταγμένες των κόμβων: (𝑥1, 𝑦1), (𝑥2, 𝑦2), (𝑥3, 𝑦3), … , (𝑥𝑛, 𝑦𝑛)
    - Ζήτηση: 𝑑1, 𝑑2, 𝑑3, … , 𝑑𝑛
    - Χωρητικότητα: 𝐶

όπου 𝑟𝑖 ∈ 𝑅 είναι τα σύνολα των κόμβων που πρέπει να εξυπηρετηθούν και αναπαρίστανται στο Ευκλείδειο επίπεδο (𝑥𝑖, 𝑦𝑖), με ζήτηση (𝑑𝑖) και χωρητικότητα 𝐶 του κάθε συμπλέγματος, με όλους τους προηγούμενους αριθμούς να έχουν θετικές τιμές. Μια ιδιότητα του Ευκλείδειου επιπέδου είναι πως οι αποστάσεις είναι συμμετρικές.

Έστω 𝑋 ο δυαδικός πίνακας, για τον οποίο ισχύει:

𝑥𝑖𝑗

= {1, 𝛼𝜈 𝜊 𝜅ό𝜇𝛽𝜊𝜍 𝑖 έ𝜒𝜀𝜄 𝛼𝜈𝛼𝜏𝜀𝜃𝜀ί 𝜎𝜏𝜂𝜈 𝜊𝜇ά𝛿𝛼 𝑗 0, 𝜀𝜈𝛼𝜆𝜆𝛼𝜅𝜏𝜄𝜅ά

Για παράδειγμα ένα δίκτυο με 10 πελάτες που πρέπει να εξυπηρετηθούν από 3 οχήματα βάσει του περιορισμού χωρητικότητας ότι η ζήτηση του πελάτη πρέπει να είναι μικρότερη ή ίση με τη χωρητικότητα του οχήματος.

*Σχήμα 4-1. Παράδειγμα Ομαδοποίησης Κόμβων[32]*

Έστω 𝑐1, 𝑐2, 𝑐3, … , 𝑐10 οι πελάτες στους οποίους γίνονται οι διανομές στο επίπεδο (𝑥, 𝑦). Ο δυαδικός πίνακας είναι μεγέθους 10 × 3, όπου οι γραμμές αντιπροσωπεύουν τους πελάτες και η κάθε στήλη αντιπροσωπεύει το κάθε όχημα ή αλλιώς το κάθε cluster. Ο πίνακας 𝑥𝑖,𝑗 αναπαρίσταται ως: 001, 010, 100, 001, 100, 010, 001, 100, 010, 100 αναδεικνύοντας τις τρεις ομάδες όπως φαίνεται και στο παραπάνω σχήμα [32].

Ο στόχος είναι να βρεθεί το 𝑋 για το οποίο ελαχιστοποιείται η συνάρτηση:

𝑘 𝑛

(4.1)

∑ ∑ 𝑐𝑜𝑠𝑡𝑖,𝑗 ∗ 𝑥𝑖𝑗

𝑗=1 𝑖=1

Η οποία έχει τους παρακάτω περιορισμούς:

𝑘

∑ 𝑥𝑖,𝑗 = 1, 𝑖 = 1, 2, … , 𝑛

𝑗=1

𝑛

∑ 𝑑𝑖𝑥𝑖,𝑗 ≤ 𝐶, 𝑗 = 1, 2, … , 𝑘

𝑖=1

(4.2)

(4.3)

Όπου 𝑐𝑜𝑠𝑡𝑖,𝑗 αντιπροσωπεύει το κόστος του πλησιέστερου κόμβου προς εξυπηρέτηση στο σύμπλεγμα 𝑗 που γίνεται η αντιστοίχιση (δηλαδή είναι το κόστος ή ο χρόνος μετακίνησης από το σημείο 𝑖 στο 𝑗). Η αντικειμενική συνάρτηση (4.1) προσπαθεί να ελαχιστοποιήσει το συνολικό κόστος εκχώρησης όλων των αιτούμενων κόμβων προς εξυπηρέτηση σε κάποιο σύμπλεγμα. Ο περιορισμός (4.2) εξασφαλίζει ότι κάθε αιτούμενος κόμβος προς εξυπηρέτηση 𝑖 θα εκχωρηθεί σε μία μόνο ομάδα 𝑗. Και τέλος ο περιορισμός (4.3) στοχεύει ώστε η συνολική ζήτηση του κάθε αιτούμενου κόμβου 𝑖 να μην υπερβαίνει την χωρητικότητα 𝐶 των συμπλεγμάτων [32].

Το CPP επιλύεται με τη χρήση του improved k-means που περιλαμβάνει τη χωρητικότητα ως έναν από τους περιορισμούς για την ομαδοποίηση των στοιχείων σε συνδυασμό με τον έλεγχο των πλησιέστερων σημείων σε κάθε σύμπλεγμα, υπολογίζοντας σε κάθε περίπτωση τις αποστάσεις στο Ευκλείδειο επίπεδο [32].

## Ο ΑΛΓΟΡΙΘΜΟΣ ΤΟΥ ΠΡΟΒΛΗΜΑΤΟΣ

Σύμφωνα με τους Geetha, Poonthalir και Vanathi [32], πριν το σχηματισμό των ομάδων υπολογίζονται οι αποστάσεις μεταξύ όλων των αιτούμενων προς εισαγωγή σε cluster κόμβων 𝑖 και όλων των κεντρικών σημείων (centroids). Ο υπολογισμός γίνεται με την Ευκλείδεια απόσταση 𝑐𝑜𝑠𝑡𝑖,𝑗.

𝑐𝑜𝑠𝑡𝑖,𝑗 = √(𝑥𝑖 − 𝑥𝑗)2 + (𝑦𝑖 − 𝑦𝑗)2 (4.4)

Όπου: 𝑖 = 1, 2, … , 𝑛 και 𝑗 = 1, 2, … , 𝑘. Τα 𝑥 και 𝑦 αναφέρονται στο Ευκλείδειο επίπεδο, το 𝑖 παραπέμπει στον εκάστοτε αιτούμενο κόμβο και το 𝑗 στο εκάστοτε κεντρικό σημείο του cluster.

Στη συνέχεια περιγράφονται τα κύρια βήματα που ακολουθούνται για την υλοποίηση του αλγορίθμου.

**Βήμα 1:** Υπολογισμός του πλήθους των ομάδων (clusters)

Υπολογίζεται βάσει της ζήτησης 𝑑𝑖. του κάθε κόμβου και της συνολικής χωρητικότητας του κάθε cluster.

𝑛

𝑘 = ∑ 𝑑𝑖/𝑐

𝑖=1

**Βήμα 2:** Επιλογή των αρχικών κεντρικών σημείων (centroids)

(4.5)

Τα αρχικά 𝑘 centroids επιλέγονται κατηγοριοποιώντας τους κόμβους που πρέπει να εξυπηρετηθούν κατά φθίνουσα σειρά 𝑑1 > 𝑑2 > 𝑑3 > … > 𝑑𝑛. Έτσι δημιουργείται μια λίστα (έστω 𝐷). Στη συνέχεια τα πρώτα 𝑘 σημεία της λίστας τίθενται ως τα centroids για τα οποία θα πραγματοποιηθεί η υπολογιστική διαδικασία τουλάχιστον μία φορά.

**Βήμα 3:** Ανάθεση των κόμβων

Εφόσον έχει υπολογιστεί η Ευκλείδεια απόσταση μεταξύ όλων των κόμβων όπως αναφέρεται στη συνάρτηση (4.4), στη συνέχεια ομαδοποιούνται όλοι οι αιτούμενοι κόμβοι 𝑖 στο κοντινότερο σημείο 𝑗. Στη συνέχεια υπολογίζεται μια τιμή προτεραιότητας

𝑃𝑟𝑖𝑜𝑟𝑖𝑡𝑦: 𝑃𝑖 = 𝑐𝑜𝑠𝑡𝑖𝑗./ 𝑑𝑖. (4.6)

σύμφωνα με την οποία καθορίζεται ο κόμβος ο οποίος έχει μεγαλύτερη προτεραιότητα για να ενταχθεί σε ένα cluster ανάλογα με την απόσταση και τη ζήτησή του. Το επιλεγμένο 𝑟𝑖 εκχωρείται με βάση τον περιορισμό (4.3). Αν ο περιορισμός (4.3) δεν ικανοποιείται, τότε θα αντιστοιχιστεί στο επόμενο πλησιέστερο κέντρο με βάση τους περιορισμούς (4.6) και (4.3)

**Βήμα 4:** Υπολογισμός Νέων Centroid

Το νέο κεντρικό σημείο (𝑋𝑗, 𝑌𝑗) για κάθε cluster υπολογίζεται με βάση τα μέλη του. Έστω

(𝑥1, 𝑦1), (𝑥2, 𝑦2), … , (𝑥𝑗, 𝑦𝑗) είναι οι συντεταγμένες των κόμβων που αντιστοιχίσθηκαν

στο εν λόγω cluster.

𝑗

𝑥𝑗 = ∑ 𝑥𝑚/𝑛𝑗

𝑚=1

(4.7)

𝑗

𝑦𝑗 = ∑ 𝑦𝑚/𝑛𝑗

𝑚=1

(4.8)

𝑐𝑗 = (𝑥𝑗 , 𝑦𝑗 ) (4.9)

Όπου 𝑐𝑗 αναπαριστά το 𝑗-οστό κεντρικό σημείο και 𝑛𝑗 αναπαριστά τον αριθμό των αιτούμενων κόμβων που περιέχονται στο cluster 𝑗 [32].

**Βήμα 5:** Έλεγχος κριτηρίων σύγκλισης

Η διαδικασία επαναλαμβάνεται μέχρι να μην υπάρχει κάποια αλλαγή στις ομάδες που σχηματίζονται, ή μέχρι να ικανοποιηθεί κάποιο κριτήριο τερματισμού (π.χ. μέχρις ότου μόνο το 1% των σημείων να μεταβάλλει τις συστάδες) [32].

## Αλγόριθμος 2. Improved k-means

**Είσοδος:**

Συντεταγμένες: (𝒙𝒊, 𝒚𝒊)

Ζήτηση: 𝒅𝒊

Κόμβοι προς εξυπηρέτηση: 𝒓𝒊

**Έξοδος:**

𝑘 ομάδες – clusters

**Διαδικασία:**

* + - 1. Υπολογισμός του 𝑘 χρησιμοποιώντας τον περιορισμό (4.5)
      2. Επιλογή των πρώτων 𝑘 κεντροειδών από την λίστα 𝐷
      3. Αρχικοποίηση του δυαδικού πίνακα 𝛸 με μηδενικά
      4. **Όσο** δεν πραγματοποιείται σύγκλιση **επανάληψη**
      5. **Για κάθε** κόμβο προς εξυπηρέτηση 𝑟𝑖 ∈ 𝑅
      6. **Όσο** το 𝑟𝑖 δεν έχει αντιστοιχιστεί **επανάληψη**
      7. Υπολογισμός του μέτρου της Ευκλείδειας απόστασης χρησιμοποιώντας τον περιορισμό (4.4) για κάθε μία από τις ομάδες, και ταξινόμηση με τη σειρά.
      8. Ανάθεση του πλησιέστερου κεντροειδούς του 𝑟𝑖 ως 𝑚
      9. Ομαδοποίηση όλων σημείων που δεν έχουν επιλεχθεί ως 𝐺

με το 𝑚 ως το πλησιέστερο τους κέντρο.

* + - 1. Υπολογισμός της τιμής προτεραιότητας για το 𝑟𝑖 ∈ 𝐺

χρησιμοποιώντας τον περιορισμό (4.6)

* + - 1. Ανάθεση του 𝑟𝑖 ∈ 𝐺 στο πλησιέστερο κεντροειδές με βάση την τιμή προτεραιότητας και χωρίς να παραβιαστεί ο περιορισμός (4.3)
      2. Ενημέρωση του 𝑥𝑖𝑗
      3. **Αν** το 𝑟𝑖 δεν έχει αντιστοιχισθεί **τότε**
      4. Επιλογή του αμέσως επόμενου κεντροειδούς
      5. **Τέλος Αν**
      6. **Τέλος Επανάληψης**
      7. **Τέλος Για**
      8. Υπολογισμός του νέου κέντρου από τις ομάδες που σχηματίστηκαν
      9. **Τέλος Επανάληψης**

## ΦΑΣΗ II: ΔΡΟΜΟΛΟΓΙΣΗ ΟΧΗΜΑΤΩΝ ΜΕ ΧΡΟΝΙΚΑ ΠΑΡΑΘΥΡΑ

Στην κατηγορία των Προβλημάτων δρομολόγησης οχημάτων ανήκουν τα Προβλήματα με χρονικά παράθυρα (Time Windows), στα οποία κάποιοι πελάτες απαιτούν η εξυπηρέτησή τους από κάποιο όχημα να ξεκινά και να τελειώνει μέσα σε ένα χρονικό διάστημα, το οποίο και ονομάζεται χρονικό παράθυρο. Όλα τα αιτήματα πρέπει να ικανοποιούνται χωρίς να παραβιάζονται οι περιορισμοί που σχετίζονται με τη χωρητικότητα των οχημάτων και την προτεραιότητα επίσκεψης των σημείων σύμφωνα με τα χρονικά διαστήματα που καθορίζονται είτε από τη φύση του φορτίου είτε από τις απαιτήσεις των πελατών.

Για την εφαρμογή των παραπάνω, καθώς και συνυπολογίζοντας την ανάγκη δημιουργίας διαδρομών με ομοιόμορφα κατανεμημένα τα φορτία τους σε όλα τα οχήματα υλοποιήθηκε ο *Αλγόριθμος 3*, στον οποίο εμφωλευμένα γίνεται η χρήση 5 διαφορετικών αλγορίθμων καταλήγοντας σε λύσεις που εστιάζουν στην βελτιστοποίηση της δρομολόγησης των οχημάτων η κάθε μία με διαφορετικά κριτήρια. Οι λειτουργίες και η χρηστικότητα του κάθε αλγορίθμου αναλύεται στη συνέχεια.

**Αλγόριθμος 3.** Δρομολόγηση Οχημάτων με χρήση Χρονικών Παραθύρων

**Είσοδος:**

Κόμβοι: **Nodes**

Τα στοιχεία κάθε κόμβου διαβάζονται από αρχείο κατά τη διάρκεια εκτέλεσης του αλγορίθμου

Αριθμός Οχημάτων: **Vehicles**

Χωρητικότητα: **Capacity**

Μέγιστος Χρόνος Λειτουργίας: **OppTime**

Συστάδες: **Clusters Έξοδος:**

Δρομολογήσεις ανά cluster

Πιθανοί κόμβοι που δεν εξυπηρετήθηκαν Χρόνος αναμονής ανά δρομολόγιο

Χρόνος – Απόσταση που διανύθηκε ανά δρομολόγιο Ζήτηση που καλύφθηκε από κάθε όχημα

**Διαδικασία:**

1. Αρχικοποίηση (μηδενισμός) πινάκων
2. Ανάγνωση δεδομένων των κόμβων προς εξυπηρέτηση
3. Έλεγχος τιμών, αν η ζήτηση σε κάθε κόμβο αλλά και η αθροιστική πληροί τους περιορισμούς χωρητικότητας των οχημάτων.
4. **Αν** δεν πληρούνται τα κριτήρια **τότε**
5. Ενημέρωση σφάλματος και τερματισμός αλγορίθμου
6. **Τέλος Αν**
7. Εισαγωγή δεδομένων των κόμβων σε vector
8. Κόμβος προς εξυπηρέτηση: **in\_node**
9. Συντεταγμένες X: **in\_x**
10. Συντεταγμένες Y: **in\_y**
11. Ζήτηση: **in\_demand**
12. Νωρίτερος Χρόνος Επίσκεψης: **in\_early**
13. Αργότερος Χρόνος Επίσκεψης: **in\_late**
14. Χρόνος Εξυπηρέτησης: **in\_service**
15. Υπολογισμός Ευκλείδειων αποστάσεων μεταξύ των κόμβων D[][]
16. Ανάγνωση δεδομένων από τον ***Αλγόριθμο Συσταδοποίησης***
17. **Επιλογή Σεναρίου** Επίλυσης του προβλήματος
18. Σενάριο 1: Ελάχιστη Απόσταση (Dijkstra)
19. Σενάριο 2: Αντίστροφη Διαδρομή Ελάχιστης Απόστασης (Inverse Dijkstra)
20. Σενάριο 3: Ελάχιστος Χρόνος Αναμονής (Min Wait Time)
21. Σενάριο 4: Εξυπηρέτησης Πελατών με Μικρή Ζήτηση (Min Demand)
22. Σενάριο 5: Εξυπηρέτησης Πελατών με Μεγάλη Ζήτηση (Max Demand)
23. **Τέλος Επιλογής**
24. Αρχικοποίηση με μηδενικές τιμές του vector **(RoutesNotSatisfied)** των κόμβων που πιθανώς να μην μπορέσουν να εξυπηρετηθούν
25. **Για κάθε** cluster από τον *Αλγόριθμο Συσταδοποίησης* **επανέλαβε**
26. Αρχικοποίηση πρώτου σημείου σε κάθε διαδρομή τον κόμβο 0 (depot)
27. Αρχικοποίηση αριθμού πελατών που έχουν εξυπηρετηθεί για κάθε cluster: **NumberOfNodesOnRoute** = 1 (ο κόμβος 0 – depot)
28. Αρχικοποίηση της μεταβλητής μου αναφέρεται στον τελευταίο κόμβο που επισκέφθηκε το όχημα με την τιμή αυτού του κόμβου, δηλαδή 0 (depot), αφού δεν έχει πραγματοποιηθεί κάποια δρομολόγηση **(LastNode = 0)**
29. Αρχικοποίηση της μεταβλητής που αναφέρεται στην απόσταση και τον χρόνο που έχει δαπανηθεί στο συγκεκριμένο δρομολόγιο. Δεν

έχει πραγματοποιηθεί κάποια δρομολόγηση, οπότε **time\_distance = 0,0**

1. **Αν** Επιλέχθηκε το *Σενάριο 1* **τότε**
2. ***Αλγόριθμος Ελάχιστης Απόστασης***
3. **Αλλιώς αν** Επιλέχθηκε το *Σενάριο 2* **τότε**
4. ***Αντίστροφος Αλγόριθμος Ελάχιστης Απόστασης***
5. **Αλλιώς αν** Επιλέχθηκε το *Σενάριο 3* **τότε**
6. ***Αλγόριθμος Ελάχιστου Χρόνου Αναμονής***
7. **Αλλιώς αν** Επιλέχθηκε το *Σενάριο 4* **τότε**
8. ***Αλγόριθμος Εξυπηρέτησης Πελατών με Μικρή Ζήτηση***
9. **Αλλιώς αν** Επιλέχθηκε το *Σενάριο 5* **τότε**
10. ***Αλγόριθμος Εξυπηρέτησης Πελατών με Μεγάλη Ζήτηση***
11. **Τέλος Αν**
12. Εμφάνιση κόμβων που πιθανώς δεν έχουν εξυπηρετηθεί
13. **Τέλος επανάληψης**
14. **Επιλογής Σεναρίου:** Έλεγχος δημιουργίας επιπλέον δρομολόγησης με τους κόμβους που δεν έχουν εξυπηρετηθεί.
15. Σενάριο 1: Ναι
16. Σενάριο 2: Όχι
17. **Τέλος Επιλογής**
18. **Αν** Επιλέχθηκε το *Σενάριο 1* **τότε**
19. **Αν** Δεν έχουν εξυπηρετηθεί όλοι οι κόμβοι **τότε**
20. Επανάληψη βημάτων από το 17 και έπειτα
21. **Αλλιώς**
22. Τερματισμός Αλγορίθμου
23. **Τέλος Αν**
24. **Αλλιώς Αν** Επιλέχθηκε το Σενάριο 2 **τότε**
25. Τερματισμός Αλγορίθμου
26. **Τέλος Αν**

Ο *Αλγόριθμος 4*, αποτελεί την κλασσική περίπτωση ελαχιστοποίησης της απόστασης κάθε δρομολογούμενου οχήματος. Στόχος του συγκεκριμένου αλγορίθμου είναι η δρομολόγηση των οχημάτων με τρόπο ο οποίος επιτυγχάνει την εξυπηρέτηση όσο το δυνατόν περισσότερων πελατών, ακολουθώντας τη συντομότερη διαδρομή ώστε να παραμείνει σε χαμηλά επίπεδα το πεδίο της τιμής της συνολικής απόστασης που καλύπτεται από το κάθε όχημα.

**Αλγόριθμος 4**. Αλγόριθμος Ελάχιστης Απόστασης

**Είσοδος:**

Δεδομένα *Αλγορίθμου 3 – δρομολόγησης οχημάτων με Χρονικά Παράθυρα*

Δεδομένα *Αλγορίθμου 2 - Συσταδοποίησης*

**flag**=false **NumberOfNodesOnRoute**=1 **LastNode=0 time\_distance=0,0**

**Έξοδος:**

Δρομολογήσεις ανά cluster

Πιθανοί κόμβοι που δεν εξυπηρετήθηκαν Χρόνος αναμονής δρομολογίου

Χρόνος – Απόσταση δρομολογίου

Ζήτηση που καλύφθηκε το όχημα που εξυπηρετεί το δρομολόγιο

**Διαδικασία:**

1. Αρχικοποίηση της μεταβλητής που αναφέρεται στην ελάχιστη απόσταση (**min\_local**) με μία πολύ μεγάλη τιμή που θα αλλάζει με την πρώτη επανάληψη.
2. Αρχικοποίηση θέσης της παραπάνω μεταβλητής (**min\_local\_place**) με τιμή που θα αλλάζει με την πρώτη επανάληψη.
3. Αρχικοποίηση μεταβλητής που θα «βλέπει» στον κόμβο που αντιστοιχεί η παραπάνω μεταβλητή (**min\_node**) με τιμή που θα αλλάζει με την πρώτη επανάληψη.
4. **Για κάθε** κόμβο του cluster που δεν έχει εξυπηρετηθεί **επανέλαβε**
5. **Αν** η απόσταση του σημείου από το τελευταίο σημείο που επισκέφθηκε το όχημα (*LastNode*) είναι μικρότερη της ελάχιστης απόστασης (*min\_local*) **τότε**
6. Αντικατάσταση της τιμής του *min\_local* με την τιμή της απόσταση του κόμβου από το τελευταίο σημείο που

το όχημα και ικανοποιεί την συνθήκη του ***βήματος 5***

1. Αντικατάσταση της τιμής του *min\_node* με την τιμή του εν λόγω κόμβου που ικανοποιεί την συνθήκη του ***βήματος 5***
2. Αντικατάσταση της τιμής του *min\_local\_place* με την τιμή που ικανοποιεί την συνθήκη του ***βήματος 5***
3. **Τέλος Αν**
4. **Τέλος Επανάληψης**
5. Αρχικοποίηση τιμής **Late\_TW**, αναφερόμενη στο αργότερο χρονικό περιθώριο που μπορεί να εξυπηρετηθεί ένας κόμβος, με το αντίστοιχο χρονικό περιθώριο του κόμβου *min\_node* από το βήμα 7.
6. Αρχικοποίηση τιμής **Early\_TW**, αναφερόμενη στο νωρίτερο χρονικό περιθώριο που μπορεί να εξυπηρετηθεί ένας κόμβος, με το αντίστοιχο χρονικό περιθώριο του κόμβου *min\_node* από το βήμα 7.
7. Αρχικοποίηση τιμής **Service\_Time**, αναφερόμενη στο χρονικό διάστημα που χρειάζεται να εξυπηρετηθεί ένας κόμβος, με το αντίστοιχο χρονικό περιθώριο του κόμβου *min\_node* από το βήμα 7.
8. Αρχικοποίηση της τιμής **Wait\_Time = 0.0**. Γίνεται μετέπειτα ο υπολογισμός του χρόνου αναμονής σε κάθε δρομολόγιο.
9. **Αν** εξυπηρετηθεί ο κόμβος *min\_node* και ικανοποιείται η συνθήκη του αργότερου χρονικού παραθύρου *Late\_TW* **τότε**
10. **Αν** εξυπηρετηθεί ο κόμβος *min\_node* και ικανοποιείται η συνθήκη του αργότερου χρονικού παραθύρου *Early\_TW* **τότε**
11. Υπολογισμός χρόνου αναμονής (*Wait\_Time*) του κόμβου

*min\_node*

1. **Τέλος Αν**
2. **Αλλιώς**
3. flag=true
4. Εισαγωγή του *min\_node* στο vector των κόμβων που δεν έχουν εξυπηρετηθεί (**NotSelectedNodes**)
5. **Τέλος Αν**
6. **Αν** Εξυπηρετηθεί ο κόμβος *min\_node* και το όχημα προλαβαίνει να επιστρέψει στον κόμβο 0 (depot) δηλαδή αν καλύπτει και το χρονικό παράθυρο της αποθήκης **τότε**
7. Ανάθεση κόμβου *min\_node* ως τον επόμενο προς επίσκεψη κόμβο
8. Αλλαγή τιμής *LastNode* με αυτή του νέου κόμβου *min\_node*
9. Αλλαγή τιμής *time\_distance* με την επαυξημένη κατά το σύνολο

𝑚𝑖𝑛\_𝑙𝑜𝑐𝑎𝑙 + 𝑆𝑒𝑟𝑣𝑖𝑐𝑒\_𝑇𝑖𝑚𝑒 + 𝑊𝑎𝑖𝑡\_𝑇𝑖𝑚𝑒 από την εξυπηρέτηση του κόμβου *min\_node*

1. **Αλλιώς**
2. **Αν** (flag=false) **τότε**
3. Εισαγωγή του *min\_node* στο vector των κόμβων που δεν έχουν εξυπηρετηθεί (**NotSelectedNodes**)
4. **Τέλος Αν**
5. **Τέλος Αν**
6. **Αν** υπάρχουν κόμβοι του cluster για τους οποίους δεν έχει πραγματοποιηθεί έλεγχος ως προς την εξυπηρέτησή τους **τότε**
7. Επανάληψη των βημάτων από το 1 και έπειτα
8. **Αλλιώς**
9. Ανάθεση τελευταίου σταθμού του δρομολογίου τον κόμβο 0 (depot)
10. **Τερματισμός αλγορίθμου**
11. **Τέλος Αν**

Ο *Αλγόριθμος 5* αποτελεί μια παραλλαγή του *Αλγορίθμου 4 - Ελάχιστης Απόστασης*. Η κύρια διαφορά τους είναι πως ανάλογα και με τα χρονικά παράθυρα επιδιώκεται η εξυπηρέτηση σε πρώτη φάση ενός εκ των κόμβων που βρίσκονται στο πιο μακρινό σημείο του δικτύου, και στη συνέχεια εξυπηρετώντας τον πλησιέστερο κόμβο σε κάθε επόμενο βήμα, το όχημα επιστρέφει στην αποθήκη επιδιώκοντας παράλληλα την εξυπηρέτηση όσο το δυνατόν περισσότερων αιτημάτων σε συνδυασμό με την κάλυψη της μικρότερης εφικτής απόσταση;.

**Αλγόριθμος 5**. Αντίστροφος Αλγόριθμος Ελάχιστης Απόστασης

**Είσοδος:**

Δεδομένα *Αλγορίθμου 3 – δρομολόγησης οχημάτων με Χρονικά Παράθυρα*

Δεδομένα *Αλγορίθμου 2 - Συσταδοποίησης*

**flag**=false **NumberOfNodesOnRoute**=1 **LastNode=0 time\_distance=0,0**

**Έξοδος:**

Δρομολογήσεις ανά cluster

Πιθανοί κόμβοι που δεν εξυπηρετήθηκαν Χρόνος αναμονής δρομολογίου

Χρόνος – Απόσταση δρομολογίου

Ζήτηση που καλύφθηκε το όχημα που εξυπηρετεί το δρομολόγιο

**Διαδικασία:**

1. Αρχικοποίηση μεταβλητής που αναφέρεται στην μέγιστη απόσταση (**max\_local**) με μία αρνητική τιμή που θα αλλάζει με την πρώτη επανάληψη.
2. Αρχικοποίηση θέσης της παραπάνω μεταβλητής (**max\_local\_place**) με τιμή που θα αλλάζει με την πρώτη επανάληψη.
3. Αρχικοποίηση μεταβλητής που θα «βλέπει» στον κόμβο που αντιστοιχεί η παραπάνω μεταβλητή (**max\_node**) με τιμή που θα αλλάζει με την πρώτη επανάληψη.
4. **Για κάθε** κόμβο του cluster που δεν έχει εξυπηρετηθεί **επανέλαβε**
5. **Αν** η απόσταση του σημείου από το τελευταίο σημείο που επισκέφθηκε το όχημα (*LastNode*) είναι μεγαλύτερη της ελάχιστης απόστασης (*max\_local*) **τότε**
6. Αντικατάσταση της τιμής του *max\_local* με την τιμή της απόσταση του κόμβου από το τελευταίο σημείο που

το όχημα και ικανοποιεί την συνθήκη του ***βήματος 5***

1. Αντικατάσταση της τιμής του *max\_node* με την τιμή του εν λόγω κόμβου που ικανοποιεί την συνθήκη του ***βήματος 5***
2. Αντικατάσταση της τιμής του *max\_local\_place* με την τιμή που ικανοποιεί την συνθήκη του ***βήματος 5***
3. **Τέλος Αν**
4. **Τέλος Επανάληψης**
5. Αρχικοποίηση τιμής **Late\_TW**, αναφερόμενη στο αργότερο χρονικό περιθώριο που μπορεί να εξυπηρετηθεί ένας κόμβος, με το αντίστοιχο χρονικό περιθώριο του κόμβου *max\_node* από το βήμα 7.
6. Αρχικοποίηση τιμής **Early\_TW**, αναφερόμενη στο νωρίτερο χρονικό περιθώριο που μπορεί να εξυπηρετηθεί ένας κόμβος, με το αντίστοιχο χρονικό περιθώριο του κόμβου *max\_node* από το βήμα 7.
7. Αρχικοποίηση τιμής **Service\_Time**, αναφερόμενη στο χρονικό διάστημα που χρειάζεται να εξυπηρετηθεί ένας κόμβος, με το αντίστοιχο χρονικό περιθώριο του κόμβου *max\_node* από το βήμα 7.
8. Αρχικοποίηση της τιμής **Wait\_Time=0.0**. Γίνεται μετέπειτα ο υπολογισμός του χρόνου αναμονής σε κάθε δρομολόγιο.
9. **Αν** εξυπηρετηθεί ο κόμβος *max\_node* και ικανοποιείται η συνθήκη του αργότερου χρονικού παραθύρου *Late\_TW* **τότε**
10. **Αν** εξυπηρετηθεί ο κόμβος *max\_node* και ικανοποιείται η συνθήκη του αργότερου χρονικού παραθύρου *Early\_TW* **τότε**
11. Υπολογισμός χρόνου αναμονής(*Wait\_Time*) του κόμβου

*max\_node*

1. **Τέλος Αν**
2. **Αλλιώς**
3. flag=true
4. Εισαγωγή του *max\_node* στο vector των κόμβων που δεν έχουν εξυπηρετηθεί (**NotSelectedNodes**)
5. **Τέλος Αν**
6. **Αν** Εξυπηρετηθεί ο κόμβος *max\_node* και το όχημα προλαβαίνει να επιστρέψει στον κόμβο 0 (depot) δηλαδή αν καλύπτει και το χρονικό παράθυρο της αποθήκης **τότε**
7. Ανάθεση κόμβου *max\_node* ως τον επόμενο προς επίσκεψη κόμβο
8. Αλλαγή τιμής *LastNode* με αυτή του νέου κόμβου *max\_node*
9. Αλλαγή τιμής *time\_distance* με την επαυξημένη κατά

𝑚𝑎𝑥\_𝑙𝑜𝑐𝑎𝑙 + 𝑆𝑒𝑟𝑣𝑖𝑐𝑒\_𝑇𝑖𝑚𝑒 + 𝑊𝑎𝑖𝑡\_𝑇𝑖𝑚𝑒 από την εξυπηρέτηση του κόμβου *min\_node*

1. **Αλλιώς**
2. **Αν** (flag=false) **τότε**
3. Εισαγωγή του *max\_node* στο vector των κόμβων που δεν έχουν εξυπηρετηθεί (**NotSelectedNodes**)
4. **Τέλος Αν**
5. **Τέλος Αν**
6. **Αν** υπάρχουν κόμβοι του cluster για τους οποίους δεν έχει πραγματοποιηθεί έλεγχος ως προς την εξυπηρέτησή τους **τότε**
7. Επανάληψη των βημάτων από το 1 και έπειτα του ***Αλγορίθμου4. Αλγόριθμος Ελάχιστης Απόστασης*** με μοναδική τροποποίηση την τιμή NumberOfNodesOnRoute που έχει επαυξηθεί κατά 1, καθώς έχει εξυπηρετηθεί πλέον ο πρώτος κόμβος.
8. **Αλλιώς**
9. Ανάθεση τελευταίου σταθμού του δρομολογίου τον κόμβο 0 (depot)
10. **Τερματισμός αλγορίθμου**
11. **Τέλος Αν**

Ο *Αλγόριθμος 6* αποτελεί μια περίπτωση δρομολόγησης των οχημάτων αποσκοπώντας στην ελαχιστοποίηση των χρόνων αναμονής για την εξυπηρέτηση κάποιου πελάτη. Σε κάθε περίπτωση δίνεται προτεραιότητα στην εξυπηρέτηση του σημείου ή των σημείων από τα οποία ο συνολικός χρόνος της διαδρομής θα επιβαρυνθεί με τον λιγότερο δυνατό τρόπο λόγω της αναμονής πριν την εξυπηρέτηση. Επιλέγοντας τους κόμβους με τον μικρότερο χρόνο αναμονής θεωρητικά απομένει μεγαλύτερο χρονικό διάστημα για την εξυπηρέτηση των υπολοίπων κόμβων στη συνέχεια. Σε κάθε βήμα εξετάζονται τα χρονικά παράθυρα καθώς και η δυνατότητα επιστροφής στην αποθήκη.

**Αλγόριθμος 6**. Αλγόριθμος Ελάχιστου Χρόνου Αναμονής

**Είσοδος:**

Δεδομένα *Αλγορίθμου 3 – δρομολόγησης οχημάτων με Χρονικά Παράθυρα*

Δεδομένα *Αλγορίθμου 2 - Συσταδοποίησης*

**flag**=false **NumberOfNodesOnRoute**=1 **LastNode=0 time\_distance=0,0**

**Έξοδος:**

Δρομολογήσεις ανά cluster

Πιθανοί κόμβοι που δεν εξυπηρετήθηκαν Χρόνος αναμονής δρομολογίου

Χρόνος – Απόσταση δρομολογίου

Ζήτηση που καλύφθηκε το όχημα που εξυπηρετεί το δρομολόγιο

**Διαδικασία:**

1. Αρχικοποίηση μεταβλητής που αναφέρεται στον ελάχιστο χρόνο αναμονής (**min\_wait\_local**) με μία μεγάλη τιμή που θα αλλάζει με την πρώτη επανάληψη.
2. Αρχικοποίηση θέσης της παραπάνω μεταβλητής (**min\_wait\_local\_place**) με τιμή που θα αλλάζει με την πρώτη επανάληψη.
3. Αρχικοποίηση μεταβλητής που θα «βλέπει» στον κόμβο που αντιστοιχεί η παραπάνω μεταβλητή (**min\_wait \_node**) με τιμή που θα αλλάζει με την πρώτη επανάληψη.
4. Αρχικοποίηση της τιμής **Wait\_Time = 0.0**.
5. **Για κάθε** κόμβο του cluster που δεν έχει εξυπηρετηθεί **επανέλαβε**
6. Εισαγωγή της μεταβλητής Early\_TW με την τιμή του νωρίτερου χρόνου εξυπηρέτησης κάθε κόμβου
7. Υπολογισμός του χρόνου αναμονής *(Wait\_Time)* σε κάθε κόμβο σύμφωνα με την απόσταση και τον χρόνο από τον τελευταίο κόμβο που εξυπηρετήθηκε *(LastNode)*.
8. **Αν** ο χρόνος αναμονής του σημείου από το τελευταίο σημείο που επισκέφθηκε το όχημα (*LastNode*) είναι μικρότερος του ελάχιστου μέχρι στιγμής χρόνου (*min\_wait\_local*) **τότε**
9. **Αν** ο χρόνος άφιξης στον κόμβο είναι πριν τον χρόνο Early\_TW **τότε**
10. Αντικατάσταση της τιμής του *min\_wait \_local* με τον

χρόνο

1. **Αλλιώς**

αναμονής του κόμβου που ικανοποιεί την συνθήκη του

***βήματος 8***

1. Αντικατάσταση της τιμής του *min\_wait \_local* με την τιμή μηδέν (0), αφού μόλις φτάσει το όχημα αμέσως

θα εξυπηρετηθεί ο κόμβος που ικανοποιεί την συνθήκη του ***βήματος 8*** χωρίς καθυστέρηση.

1. **Τέλος Αν**
2. Αντικατάσταση της τιμής του *min\_wait \_node* με την τιμή του εν λόγω κόμβου που ικανοποιεί την συνθήκη του ***βήματος 8***
3. Αντικατάσταση της τιμής του *min\_wait \_local\_place* με την τιμή που ικανοποιεί την συνθήκη του ***βήματος 8***
4. **Τέλος Αν**
5. **Τέλος Επανάληψης**
6. Αρχικοποίηση τιμής **Late\_TW**, αναφερόμενη στο αργότερο χρονικό περιθώριο που μπορεί να εξυπηρετηθεί ένας κόμβος, με το αντίστοιχο χρονικό περιθώριο του κόμβου *max\_node* από το βήμα 7.
7. Αρχικοποίηση τιμής **Service\_Time**, αναφερόμενη στο χρονικό διάστημα που χρειάζεται να εξυπηρετηθεί ένας κόμβος, με το αντίστοιχο χρονικό περιθώριο του κόμβου *max\_node* από το βήμα 7.
8. Ενημέρωση της τιμής Early\_TW για τον κόμβο *min\_wait \_node.*
9. **Αν** εξυπηρετηθεί ο κόμβος *min\_wait \_node* και ικανοποιείται η συνθήκη του αργότερου χρονικού παραθύρου *Late\_TW* **τότε**
10. **Αν** εξυπηρετηθεί ο κόμβος *min\_wait\_node* και ικανοποιείται η συνθήκη του αργότερου χρονικού παραθύρου *Early\_TW* **τότε**
11. Υπολογισμός χρόνου αναμονής (*Wait\_Time*) του κόμβου

*min\_wait \_node*

1. **Τέλος Αν**
2. **Αλλιώς**
3. flag=true
4. Εισαγωγή του *min\_wait\_node* στο vector των κόμβων που δεν έχουν εξυπηρετηθεί (**NotSelectedNodes**)
5. **Τέλος Αν**
6. **Αν** Εξυπηρετηθεί ο κόμβος *min\_wait\_node* και το όχημα προλαβαίνει να επιστρέψει στον κόμβο 0 (depot) δηλαδή αν καλύπτει και το χρονικό παράθυρο της αποθήκης **τότε**
7. Ανάθεση κόμβου *min\_wait\_node* ως τον επόμενο προς επίσκεψη κόμβο
8. Αλλαγή τιμής *LastNode* με αυτή του νέου κόμβου *min\_wait\_node*
9. Αλλαγή τιμής *time\_distance* με την επαυξημένη κατά

την Ευκλείδεια απόσταση μεταξύ του *min\_wait*\_node και του

𝐿𝑎𝑠𝑡𝑁𝑜𝑑𝑒 + 𝑆𝑒𝑟𝑣𝑖𝑐𝑒\_𝑇𝑖𝑚𝑒 + 𝑊𝑎𝑖𝑡\_𝑇𝑖𝑚𝑒 από την εξυπηρέτηση του κόμβου *min\_node*

1. **Αλλιώς**
2. **Αν** (flag=false) **τότε**
3. Εισαγωγή του *min\_wait\_node* στο vector των κόμβων που δεν έχουν εξυπηρετηθεί (**NotSelectedNodes**)
4. **Τέλος Αν**
5. **Τέλος Αν**
6. **Αν** υπάρχουν κόμβοι του cluster για τους οποίους δεν έχει πραγματοποιηθεί έλεγχος ως προς την εξυπηρέτησή τους **τότε**
7. Επανάληψη των βημάτων από το 1 και έπειτα.
8. **Αλλιώς**
9. Ανάθεση τελευταίου σταθμού του δρομολογίου τον κόμβο 0 (depot)
10. **Τερματισμός αλγορίθμου**
11. **Τέλος Αν**

Στην περίπτωση του *Αλγορίθμου 7*, εξετάζεται μια μέθοδος η οποία εστιάζει στη δρομολόγηση των οχημάτων επιδιώκοντας την εξυπηρέτηση των διαθέσιμων κόμβων με μικρότερη ποσότητα φορτίου και στη συνέχεια αυτών με υψηλότερη. Έτσι επιδιώκεται η υλοποίηση μίας λύσης η οποία θα εξυπηρετεί μεγαλύτερο αριθμό κόμβων, χωρίς να δεσμεύεται μεγάλο ποσοστό της χωρητικότητας κάθε οχήματος για την ικανοποίηση πελατών με μεγάλη ζήτηση. Στοχεύοντας στην ικανοποίηση των κόμβων με μικρή ποσότητα φορτίου, είναι διαθέσιμο μεγαλύτερο ποσοστό χωρητικότητας για να εξυπηρετήσει περισσότερους κόμβους ίδιου ή πανομοιότυπου επιπέδου ζήτησης. Στη συνέχεια εξετάζεται η ικανοποίηση των κόμβων με μεγαλύτερη ζήτηση μέχρι τη λήξη των χρονικών περιθωρίων ή την υπέρβαση του ορίου χωρητικότητας του οχήματος.

**Αλγόριθμος 7**. Αλγόριθμος Εξυπηρέτησης Πελατών με Μικρή Ζήτηση

**Είσοδος:**

Δεδομένα *Αλγορίθμου 3 – δρομολόγησης οχημάτων με Χρονικά Παράθυρα*

Δεδομένα *Αλγορίθμου 2 - Συσταδοποίησης*

**flag**=false **NumberOfNodesOnRoute**=1 **LastNode=0 time\_distance=0,0**

**Έξοδος:**

Δρομολογήσεις ανά cluster

Πιθανοί κόμβοι που δεν εξυπηρετήθηκαν Χρόνος αναμονής δρομολογίου

Χρόνος – Απόσταση δρομολογίου

Ζήτηση που καλύφθηκε το όχημα που εξυπηρετεί το δρομολόγιο

**Διαδικασία:**

1. Αρχικοποίηση της μεταβλητής που αναφέρεται στην ελάχιστη ζήτηση (**min\_demand\_local**) με μία πολύ μεγάλη τιμή που θα αλλάζει με την πρώτη επανάληψη.
2. Αρχικοποίηση θέσης της παραπάνω μεταβλητής (**min\_demand \_local\_place**) με τιμή που θα αλλάζει με την πρώτη επανάληψη.
3. Αρχικοποίηση μεταβλητής που θα «βλέπει» στον κόμβο που αντιστοιχεί η παραπάνω μεταβλητή (**min\_demand \_node**) με τιμή που θα αλλάζει με την πρώτη επανάληψη.
4. **Για κάθε** κόμβο του cluster **επανέλαβε**
5. **Αν** η ζήτηση του σημείου είναι μικρότερη της ελάχιστης ζήτησης (*min\_demand\_local*) **τότε**
6. Αντικατάσταση της τιμής του *min\_demand \_local* με την τιμή της ζήτησης του κόμβου από το τελευταίο σημείο που

το όχημα και ικανοποιεί την συνθήκη του ***βήματος 5***

1. Αντικατάσταση της τιμής του *min\_demand \_ node* με την τιμή του κόμβου που ικανοποιεί την συνθήκη του ***βήματος 5***
2. Αντικατάσταση της τιμής του *min\_demand \_local\_place* με την τιμή που ικανοποιεί την συνθήκη του ***βήματος 5***
3. **Τέλος Αν**
4. **Τέλος Επανάληψης**
5. Αρχικοποίηση τιμής **Late\_TW**, αναφερόμενη στο αργότερο χρονικό περιθώριο που μπορεί να εξυπηρετηθεί ένας κόμβος, με το αντίστοιχο χρονικό περιθώριο του κόμβου *min\_demand \_node* από το βήμα 7.
6. Αρχικοποίηση τιμής **Early\_TW**, αναφερόμενη στο νωρίτερο χρονικό περιθώριο που μπορεί να εξυπηρετηθεί ένας κόμβος, με το αντίστοιχο χρονικό περιθώριο του κόμβου *min\_demand \_node* από το βήμα 7.
7. Αρχικοποίηση τιμής **Service\_Time**, αναφερόμενη στο χρονικό διάστημα που χρειάζεται να εξυπηρετηθεί ένας κόμβος, με το αντίστοιχο χρονικό περιθώριο του κόμβου *min\_demand \_node* από το βήμα 7.
8. Αρχικοποίηση της τιμής **Wait\_Time = 0.0**. Γίνεται μετέπειτα ο υπολογισμός του χρόνου αναμονής σε κάθε δρομολόγιο.
9. **Αν** εξυπηρετηθεί ο κόμβος *min\_demand \_node* και ικανοποιείται η συνθήκη του αργότερου χρονικού παραθύρου *Late\_TW* **τότε**
10. **Αν** εξυπηρετηθεί ο κόμβος *min\_demand \_node* και ικανοποιείται η συνθήκη του αργότερου χρονικού παραθύρου *Early\_TW* **τότε**
11. Υπολογισμός χρόνου αναμονής (*Wait\_Time*) του κόμβου

*min\_demand \_node*

1. **Τέλος Αν**
2. **Αλλιώς**
3. flag = true
4. Εισαγωγή του *min\_demand\_node* στο vector των κόμβων που δεν έχουν εξυπηρετηθεί (**NotSelectedNodes**)
5. **Τέλος Αν**
6. **Αν** Εξυπηρετηθεί ο κόμβος *min\_demand \_node* και το όχημα προλαβαίνει να επιστρέψει στον κόμβο 0 (depot) δηλαδή αν καλύπτει και το χρονικό παράθυρο της αποθήκης **τότε**
7. Ανάθεση κόμβου *min\_demand \_node* ως τον επόμενο προς επίσκεψη κόμβο
8. Αλλαγή τιμής *LastNode* με αυτή του νέου κόμβου *min\_demand \_node*
9. Αλλαγή τιμής *time\_distance* με την επαυξημένη κατά την Ευκλείδεια απόσταση μεταξύ του *min\_demand*\_node και του

𝐿𝑎𝑠𝑡𝑁𝑜𝑑𝑒 + 𝑆𝑒𝑟𝑣𝑖𝑐𝑒\_𝑇𝑖𝑚𝑒 + 𝑊𝑎𝑖𝑡\_𝑇𝑖𝑚𝑒 από την εξυπηρέτηση του κόμβου *min\_demand\_node.*

1. **Αλλιώς**
2. **Αν** (flag=false) **τότε**
3. Εισαγωγή του *min\_demand\_node* στο vector των κόμβων που δεν έχουν εξυπηρετηθεί (**NotSelectedNodes**)
4. **Τέλος Αν**
5. **Τέλος Αν**
6. **Αν** υπάρχουν κόμβοι του cluster για τους οποίους δεν έχει πραγματοποιηθεί έλεγχος ως προς την εξυπηρέτησή τους **τότε**
7. Επανάληψη των βημάτων από το 1 και έπειτα
8. **Αλλιώς**
9. Ανάθεση τελευταίου σταθμού του δρομολογίου τον κόμβο 0 (depot)
10. **Τερματισμός αλγορίθμου**
11. **Τέλος Αν**

Στην περίπτωση του *Αλγορίθμου 8*, εξετάζεται μια μέθοδος η οποία εστιάζει στη δρομολόγηση του κάθε οχήματος δίνοντας βάση κατά κύριο λόγο στην εξυπηρέτηση των διαθέσιμων κόμβων με μικρότερη ποσότητα φορτίου και στη συνέχεια αυτών με υψηλότερη. Έτσι επιδιώκεται η υλοποίηση μίας λύσης η οποία θα εξυπηρετεί σε μεγαλύτερο ποσοστό κόμβους με υψηλά επίπεδα ζήτησης, χωρίς να δεσμεύεται μεγάλο ποσοστό της χωρητικότητας κάθε οχήματος για την ικανοποίηση πελατών με μικρή ζήτηση. Στοχεύοντας στην ικανοποίηση των κόμβων με μεγάλη ποσότητα φορτίου σε πρώτη φάση, υπάρχει η δυνατότητα να εξυπηρετηθούν μετέπειτα οι κόμβοι χαμηλότερου επιπέδου ζήτησης χωρίς όμως να τίθεται κίνδυνος μη εξυπηρέτησης κάποιου «μεγάλου» κόμβου λόγω έλλειψης χωρητικότητας σε κάποιο δρομολόγιο. Στο σύνολο των περιπτώσεων εξετάζεται η ικανοποίηση των κόμβων μέχρι τη λήξη των χρονικών περιθωρίων ή την υπέρβαση του ορίου χωρητικότητας του οχήματος.

**Αλγόριθμος 8**. Αλγόριθμος Εξυπηρέτησης Πελατών με Μεγάλη Ζήτηση

**Είσοδος:**

Δεδομένα *Αλγορίθμου 3 – δρομολόγησης οχημάτων με Χρονικά Παράθυρα*

Δεδομένα *Αλγορίθμου 2 - Συσταδοποίησης*

**flag**=false **NumberOfNodesOnRoute**=1 **LastNode=0 time\_distance=0,0**

**Έξοδος:**

Δρομολογήσεις ανά cluster

Πιθανοί κόμβοι που δεν εξυπηρετήθηκαν Χρόνος αναμονής δρομολογίου

Χρόνος – Απόσταση δρομολογίου

Ζήτηση που καλύφθηκε το όχημα που εξυπηρετεί το δρομολόγιο

**Διαδικασία:**

1. Αρχικοποίηση της μεταβλητής που αναφέρεται στην ελάχιστη ζήτηση (**max\_demand\_local**) με μία πολύ μεγάλη τιμή που θα αλλάζει με την πρώτη επανάληψη.
2. Αρχικοποίηση θέσης της παραπάνω μεταβλητής (**max\_demand \_local\_place**) με τιμή που θα αλλάζει με την πρώτη επανάληψη.
3. Αρχικοποίηση μεταβλητής που θα «βλέπει» στον κόμβο που αντιστοιχεί η παραπάνω μεταβλητή (**max\_demand \_node**) με τιμή που θα αλλάζει με την πρώτη επανάληψη.
4. **Για κάθε** κόμβο του cluster **επανέλαβε**
5. **Αν** η ζήτηση του σημείου είναι μικρότερη της ελάχιστης ζήτησης (*max\_demand\_local*) **τότε**
6. Αντικατάσταση της τιμής του *max\_demand \_local* με την τιμή της ζήτησης του κόμβου από το τελευταίο σημείο που

το όχημα και ικανοποιεί την συνθήκη του ***βήματος 5***

1. Αντικατάσταση της τιμής *max\_demand \_ node* με την τιμή του κόμβου που ικανοποιεί την συνθήκη του ***βήματος 5***
2. Αντικατάσταση της τιμής του *max\_demand \_local\_place* με την

τιμή που ικανοποιεί την συνθήκη του ***βήματος 5***

1. **Τέλος Αν**
2. **Τέλος Επανάληψης**
3. Αρχικοποίηση τιμής **Late\_TW**, αναφερόμενη στο αργότερο χρονικό περιθώριο που μπορεί να εξυπηρετηθεί ένας κόμβος, με το αντίστοιχο χρονικό περιθώριο του κόμβου *max\_demand \_node* από το βήμα 7.
4. Αρχικοποίηση τιμής **Early\_TW**, αναφερόμενη στο νωρίτερο χρονικό περιθώριο που μπορεί να εξυπηρετηθεί ένας κόμβος, με το αντίστοιχο χρονικό περιθώριο του κόμβου *max\_demand \_node* από το βήμα 7.
5. Αρχικοποίηση τιμής **Service\_Time**, αναφερόμενη στο χρονικό διάστημα που χρειάζεται να εξυπηρετηθεί ένας κόμβος, με το αντίστοιχο χρονικό περιθώριο του κόμβου *max\_demand \_node* από το βήμα 7.
6. Αρχικοποίηση της τιμής **Wait\_Time = 0.0**. Γίνεται μετέπειτα ο υπολογισμός του χρόνου αναμονής σε κάθε δρομολόγιο.
7. **Αν** εξυπηρετηθεί ο κόμβος *max\_demand \_node* και ικανοποιείται η συνθήκη του αργότερου χρονικού παραθύρου *Late\_TW* **τότε**
8. **Αν** εξυπηρετηθεί ο κόμβος *max\_demand \_node* και ικανοποιείται η συνθήκη του αργότερου χρονικού παραθύρου *Early\_TW* **τότε**
9. Υπολογισμός χρόνου αναμονής (*Wait\_Time*) του κόμβο

*max\_demand \_node*

1. **Τέλος Αν**
2. **Αλλιώς**
3. flag = true
4. Εισαγωγή του *max\_demand\_node* στο vector των κόμβων που δεν έχουν εξυπηρετηθεί (**NotSelectedNodes**)
5. **Τέλος Αν**
6. **Αν** Εξυπηρετηθεί ο κόμβος *max\_demand \_node* και το όχημα προλαβαίνει να επιστρέψει στον κόμβο 0 (depot) δηλαδή αν καλύπτει και το χρονικό παράθυρο της αποθήκης **τότε**
7. Ανάθεση του *max\_demand \_node* ως τον επόμενο προς επίσκεψη κόμβο
8. Αλλαγή τιμής *LastNode* με αυτή του νέου κόμβου

*max\_demand \_node*

1. Αλλαγή τιμής *time\_distance* με την επαυξημένη κατά την Ευκλείδεια απόσταση μεταξύ του *max\_demand*\_*node* και του

𝐿𝑎𝑠𝑡𝑁𝑜𝑑𝑒 + 𝑆𝑒𝑟𝑣𝑖𝑐𝑒\_𝑇𝑖𝑚𝑒 + 𝑊𝑎𝑖𝑡\_𝑇𝑖𝑚𝑒 από την εξυπηρέτηση του κόμβου *max\_demand\_node.*

1. **Αλλιώς**
2. **Αν** (flag=false) **τότε**
3. Εισαγωγή του *max\_demand\_node* στο vector των κόμβων που δεν έχουν εξυπηρετηθεί (**NotSelectedNodes**)
4. **Τέλος Αν**
5. **Τέλος Αν**
6. **Αν** υπάρχουν κόμβοι του cluster για τους οποίους δεν έχει πραγματοποιηθεί έλεγχος ως προς την εξυπηρέτησή τους **τότε**
7. Επανάληψη των βημάτων από το 1 και έπειτα
8. **Αλλιώς**
9. Ανάθεση τελευταίου σταθμού του δρομολογίου τον κόμβο 0 (depot)
10. **Τερματισμός αλγορίθμου**
11. **Τέλος Αν**

Κεφάλαιο 5

**Υπολογιστική Εμπειρία – Πειραματικά Αποτελέσματα**

Σε αυτό το κεφάλαιο, παρουσιάζεται η δοκιμαστική μελέτη και αξιολόγηση της υλοποίησης του προβλήματος δρομολόγησης στόλου οχημάτων με τη χρήση χρονικών παραθύρων, για τις περιπτώσεις που παρουσιάστηκαν στο προηγούμενο κεφάλαιο. Η υλοποίηση έγινε στη γλώσσα προγραμματισμού C++ και κωδικοποιήθηκε στο περιβάλλον του Microsoft Visual Studio Community 2015, με τον compiler Visual C++ 2015 και λειτουργικό σύστημα Microsoft Windows 10 Home. Τα χαρακτηριστικά του υπολογιστή που έτρεξε τις δοκιμές είναι: επεξεργαστής: Intel® Core™ i7-5500U CPU @ 2,40GHz με χρήση turbo 3,00GHz, 2 πυρήνες, 4 λογικοί επεξεργαστές, Μνήμη Cache 4MB, κύρια μνήμη 8GB RAM και σκληρούς δίσκους SSD 256GB και HDD 1TB.

Η δοκιμαστική αξιολόγηση έγινε τόσο με συνθετικά σύνολα δεδομένων (Solomon) όσο και με πραγματικά σύνολα δεδομένων. Το πρώτο βήμα ήταν η προσέγγιση με συνθετικά δεδομένα προκειμένου να αξιολογηθεί το κάθε τμήμα της υλοποίησης και να πραγματοποιηθούν παρατηρήσεις ανάλογα με τα χαρακτηριστικά κάθε περίπτωσης. Το δεύτερο βήμα ήταν η αξιολόγηση μιας υλοποίησης η οποία συνδυάζει περισσότερες από μία μεθόδους, αποσκοπώντας στη βέλτιστη επίλυση του κάθε προβλήματος και την προσέγγιση λύσεων με διαφορετικά χαρακτηριστικά. Τέλος το τρίτο βήμα ήταν η εφαρμογή αυτών των μοντέλων σε ένα σύνολο πραγματικών δεδομένων ώστε να διαπιστωθεί η συμπεριφορά του εν λόγω τρόπου επίλυσης του προβλήματος δρομολόγησης οχημάτων στην πράξη.

Τα σύνολα πραγματικών δεδομένων αντλήθηκαν κατά τη διάρκεια πρακτικής άσκησης στην εταιρεία EDIL Hellas με έδρα στην Θεσσαλονίκη. Το σύνολο των δεδομένων προέρχεται από το Τμήμα Ενεργειακών και Περιβαλλοντικών Έργων και πιο συγκεκριμένα από το Τμήμα Αποκομιδής και παρήχθησαν από την εταιρεία για

ερευνητικούς σκοπούς, αλλά και λόγω της ανάγκης επίλυσης προβλήματος που σχετιζόταν με τις δρομολογήσεις των οχημάτων της σε μία συγκεκριμένη περιοχή. Η ανάλυση του συγκεκριμένου σεναρίου, καθώς και στοιχεία σχετικά με τη δομή του προβλήματος, τις παραδοχές και αποτελέσματα της εφαρμογής του μοντέλου των δύο φάσεων που ερευνάται στην παρούσα διπλωματική εργασία παρουσιάζονται αναλυτικά στο Κεφάλαιο 6.

Και στις δύο περιπτώσεις, συνθετικών και πραγματικών, το σύνολο των δεδομένων είναι της ίδιας μορφής αλλά ο σκοπός των δοκιμών απευθύνεται σε δύο διαφορετικούς κόσμους. Τα συνθετικά δεδομένα χρησιμοποιούνται κυρίως για να εξετάσουν τη λειτουργία των μεθόδων και συναρτήσεων της υλοποίησης καθώς και την ποιότητα της κάθε λύσης. Εν αντιθέσει, τα πραγματικά δεδομένα περιγράφουν πολύπλοκα προβλήματα του πραγματικού κόσμου. Σε πραγματικά σενάρια, το πρόβλημα μπορεί να μην είναι 100% σαφές ή να μην έχει οριστεί επαρκώς. Μια ακριβής κατηγοριοποίηση δεν είναι εύκολη, καθώς πολλά προβλήματα του πραγματικού κόσμου περιλαμβάνουν χαρακτηριστικά από παραπάνω από ένα μοντέλα. Ως εκ τούτου, η προσφορά λύσεων για τις ανάγκες προβλημάτων του πραγματικού κόσμου αποτελεί μια πραγματική πρόκληση.

## ΔΡΟΜΟΛΟΓΗΣΗ ΣΤΟΛΟΥ ΟΧΗΜΑΤΩΝ ΜΕ ΣΥΝΘΕΤΙΚΑ ΔΕΔΟΜΕΝΑ

Οι δοκιμές με συνθετικά δεδομένα χρησιμεύουν κυρίως για την απόδειξη της λειτουργίας της προσέγγισης. Γι’ αυτόν το λόγο χρησιμοποιούνται απλά δεδομένα δημιουργημένα με τυχαίο τρόπο. Τέτοια σύνολα δεδομένων, ευρέως γνωστά στην κατηγορία των Προβλημάτων δρομολόγησης οχημάτων με χρονικά παράθυρα, αποτελούν τα δεδομένα του Solomon, τα οποία έχουν σχεδιαστεί για την επίλυση έξι συνόλων προβλημάτων, το καθένα εστιάζοντας σε διαφορετικούς παράγοντες που επηρεάζουν τη συμπεριφορά των αλγορίθμων δρομολόγησης και προγραμματισμού. Πρόκειται για: γεωγραφικά δεδομένα, το πλήθος των πελατών που εξυπηρετούνται από ένα όχημα, το ποσοστό των χρονικά περιορισμένων πελατών, καθώς και τη φύση των χρονικών παραθύρων, αν είναι δηλαδή ευέλικτα ή πολύ αυστηρά [25].

Τα γεωγραφικά δεδομένα παράγονται τυχαία στα σύνολα προβλημάτων R1 και R2, συγκεντρωμένα στα σύνολα προβλημάτων C1 και C2, και αποτελούν μίγμα τυχαίων και συγκεντρωμένων τιμών στα σύνολα προβλημάτων RC1 και RC2. Τα σύνολα προβλημάτων R1, C1 και RC1 έχουν έναν σύντομο ορίζοντα προγραμματισμού και επιτρέπουν μόνο λίγους πελάτες ανά διαδρομή (περίπου 5 έως 10). Αντίθετα, τα σύνολα R2, C2 και RC2 έχουν έναν μακρύ ορίζοντα προγραμματισμού που επιτρέπει σε πολλούς πελάτες (πάνω από 30) να εξυπηρετούνται από το ίδιο όχημα.

Οι συντεταγμένες του πελάτη είναι ίδιες για όλα τα προβλήματα ενός τύπου (δηλ. R, C και RC). Τα προβλήματα διαφέρουν σε σχέση με το εύρος των χρονικών παραθύρων. Ορισμένοι έχουν πολύ σφιχτά χρονικά παράθυρα, ενώ άλλοι έχουν χρονικά παράθυρα τα οποία δύσκολα περιορίζονται. Όσον αφορά την πυκνότητα παραθύρου χρόνου, δηλαδή το ποσοστό των πελατών με χρονικά παράθυρα, έχουν δημιουργηθεί προβλήματα με παράθυρα 25, 50, 75 και 100% του συνολικού χρόνου [25].

Τα μεγαλύτερα είναι τα προβλήματα 100 ευκλείδειων πελατών, όπου οι χρόνοι ταξιδιού είναι ίσοι με τις αντίστοιχες αποστάσεις. Για κάθε τέτοιο πρόβλημα, έχουν δημιουργηθεί μικρότερα προβλήματα εξετάζοντας μόνο τους πρώτους 25 ή 50 πελάτες.

Για να εξεταστεί η λειτουργία των διαφορετικών προσεγγίσεων καθώς και η ποιότητα των λύσεων χρησιμοποιήθηκε από το σύνολο των προβλημάτων το πρόβλημα RC201, καθώς το μίγμα τυχαίων αλλά και συγκεντρωμένων τιμών είναι πιο κοντά σε ένα πραγματικό πρόβλημα, αλλά θα είχε και καλύτερη εφαρμογή στην πρώτη φάση της επίλυσης, τη συσταδοποίηση. Η μερική συγκέντρωση τιμών σε κάποια σημεία θα βοηθούσε αρχικά στη δημιουργία ομάδων και οι τυχαίες τιμές θα μπορούσαν να παρομοιαστούν με πιθανούς νέους πελάτες προς εξυπηρέτηση εκτός του ήδη υπάρχοντα κύκλου δρομολογίων οι οποίοι επηρεάζουν θετικά ή αρνητικά την τελική εξυπηρέτηση του συνόλου των πελατών ανάλογα με τα χαρακτηριστικά τους. Όπως αναφέρθηκε και σε προηγούμενο κεφάλαιο για την απόσταση μεταξύ των πελατών χρησιμοποιήθηκε η Ευκλείδεια απόσταση. Στόχος των δοκιμών με συνθετικά δεδομένα ήταν η επιβεβαίωση της ορθής λειτουργίας της προσέγγισης σε ένα ελεγχόμενο περιβάλλον χωρίς επιρροές από άλλους παράγοντες [25].

## 5.1.1 ΑΝΑΛΥΣΗ ΣΥΝΘΕΤΙΚΩΝ ΔΕΔΟΜΕΝΩΝ

Για τις ανάγκες της εργασίας χρησιμοποιήθηκε ένα εκ των συνόλων δεδομένων του Solomon, το πρόβλημα RC201, το οποίο περιλαμβάνει στοιχεία που αφορούν τα χαρακτηριστικά των οχημάτων καθώς και των σημείων προς εξυπηρέτηση. Σχετικά με τα χαρακτηριστικά των οχημάτων δηλώνεται ότι ο αριθμός των οχημάτων που πρόκειται να χρησιμοποιηθούν δεν πρέπει να υπερβαίνει τα 25. Ακόμη όλα τα φορτηγά χαρακτηρίζονται ως πανομοιότυπα, με μέγιστη χωρητικότητα τις 1000 μονάδες προϊόντων. Σχετικά με τα χαρακτηριστικά των κόμβων – πελατών προς εξυπηρέτηση, τα στοιχεία που δίνονται παρουσιάζονται αναλυτικά στον παρακάτω πίνακα [25].

*Πίνακας 4. Χαρακτηριστικά Κόμβων - Πελατών προς Εξυπηρέτηση, σύμφωνα με το πρόβλημα RC201 του Solomon*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Customer Number | X  Coord. | Y  Coord. | Demand | Ready Time (Early\_TW) | Due Time (Late\_TW) | Service Time |
| 0 | 40 | 50 | 0 | 0 | 960 | 0 |
| 1 | 25 | 85 | 20 | 673 | 793 | 10 |
| 2 | 22 | 75 | 30 | 152 | 272 | 10 |
| 3 | 22 | 85 | 10 | 471 | 591 | 10 |
| 4 | 20 | 80 | 40 | 644 | 764 | 10 |
| 5 | 20 | 85 | 20 | 73 | 193 | 10 |
| 6 | 18 | 75 | 20 | 388 | 508 | 10 |
| 7 | 15 | 75 | 20 | 300 | 420 | 10 |
| 8 | 15 | 80 | 10 | 367 | 487 | 10 |
| 9 | 10 | 35 | 20 | 371 | 491 | 10 |
| 10 | 10 | 40 | 30 | 519 | 639 | 10 |
| 11 | 8 | 40 | 40 | 195 | 315 | 10 |
| 12 | 8 | 45 | 20 | 223 | 343 | 10 |
| 13 | 5 | 35 | 10 | 653 | 773 | 10 |
| 14 | 5 | 45 | 10 | 35 | 155 | 10 |
| 15 | 2 | 40 | 20 | 174 | 294 | 10 |
| 16 | 0 | 40 | 20 | 255 | 375 | 10 |
| 17 | 0 | 45 | 20 | 703 | 823 | 10 |
| 18 | 44 | 5 | 20 | 335 | 455 | 10 |
| 19 | 42 | 10 | 40 | 254 | 374 | 10 |
| 20 | 42 | 15 | 10 | 537 | 657 | 10 |
| 21 | 40 | 5 | 10 | 215 | 335 | 10 |
| 22 | 40 | 15 | 40 | 375 | 495 | 10 |
| 23 | 38 | 5 | 30 | 201 | 321 | 10 |
| 24 | 38 | 15 | 10 | 681 | 801 | 10 |
| 25 | 35 | 5 | 20 | 784 | 904 | 10 |
| 26 | 95 | 30 | 30 | 529 | 649 | 10 |
| 27 | 95 | 35 | 20 | 146 | 266 | 10 |
| 28 | 92 | 30 | 10 | 149 | 269 | 10 |
| 29 | 90 | 35 | 10 | 194 | 314 | 10 |
| 30 | 88 | 30 | 10 | 246 | 366 | 10 |
| 31 | 88 | 35 | 20 | 165 | 285 | 10 |
| 32 | 87 | 30 | 10 | 621 | 741 | 10 |
| 33 | 85 | 25 | 10 | 80 | 200 | 10 |
| 34 | 85 | 35 | 30 | 487 | 607 | 10 |
| 35 | 67 | 85 | 20 | 657 | 777 | 10 |
| 36 | 65 | 85 | 40 | 43 | 163 | 10 |
| 37 | 65 | 82 | 10 | 557 | 677 | 10 |
| 38 | 62 | 80 | 30 | 278 | 398 | 10 |
| 39 | 60 | 80 | 10 | 64 | 184 | 10 |
| 40 | 60 | 85 | 30 | 329 | 449 | 10 |
| 41 | 58 | 75 | 20 | 376 | 496 | 10 |
| 42 | 55 | 80 | 10 | 33 | 153 | 10 |
| 43 | 55 | 85 | 20 | 574 | 694 | 10 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 44 | 55 | 82 | 10 | 217 | 337 | 10 |
| 45 | 20 | 82 | 10 | 37 | 157 | 10 |
| 46 | 18 | 80 | 10 | 489 | 609 | 10 |
| 47 | 2 | 45 | 10 | 105 | 225 | 10 |
| 48 | 42 | 5 | 10 | 732 | 852 | 10 |
| 49 | 42 | 12 | 10 | 440 | 5560 | 10 |
| 50 | 72 | 35 | 30 | 507 | 627 | 10 |
| 51 | 55 | 20 | 19 | 326 | 446 | 10 |
| 52 | 25 | 30 | 3 | 175 | 295 | 10 |
| 53 | 20 | 50 | 5 | 375 | 495 | 10 |
| 54 | 55 | 60 | 16 | 601 | 721 | 10 |
| 55 | 30 | 60 | 16 | 599 | 719 | 10 |
| 56 | 50 | 35 | 19 | 557 | 677 | 10 |
| 57 | 30 | 25 | 23 | 397 | 517 | 10 |
| 58 | 15 | 10 | 20 | 782 | 902 | 10 |
| 59 | 10 | 20 | 19 | 42 | 162 | 10 |
| 60 | 15 | 60 | 17 | 694 | 814 | 10 |
| 61 | 45 | 65 | 9 | 258 | 378 | 10 |
| 62 | 65 | 35 | 3 | 167 | 287 | 10 |
| 63 | 65 | 20 | 6 | 39 | 159 | 10 |
| 64 | 45 | 30 | 17 | 191 | 311 | 10 |
| 65 | 35 | 40 | 16 | 11 | 131 | 10 |
| 66 | 41 | 37 | 16 | 566 | 686 | 10 |
| 67 | 64 | 42 | 9 | 268 | 388 | 10 |
| 68 | 40 | 60 | 21 | 612 | 732 | 10 |
| 69 | 31 | 52 | 27 | 157 | 277 | 10 |
| 70 | 35 | 69 | 23 | 810 | 930 | 10 |
| 71 | 65 | 55 | 14 | 241 | 361 | 10 |
| 72 | 63 | 65 | 8 | 60 | 180 | 10 |
| 73 | 2 | 60 | 5 | 286 | 406 | 10 |
| 74 | 20 | 20 | 8 | 645 | 765 | 10 |
| 75 | 5 | 5 | 16 | 232 | 352 | 10 |
| 76 | 60 | 12 | 31 | 268 | 388 | 10 |
| 77 | 23 | 3 | 7 | 764 | 884 | 10 |
| 78 | 8 | 56 | 27 | 265 | 485 | 10 |
| 79 | 6 | 68 | 30 | 352 | 472 | 10 |
| 80 | 47 | 47 | 13 | 822 | 942 | 10 |
| 81 | 49 | 58 | 10 | 355 | 475 | 10 |
| 82 | 27 | 43 | 9 | 152 | 272 | 10 |
| 83 | 37 | 31 | 14 | 105 | 225 | 10 |
| 84 | 57 | 29 | 18 | 395 | 515 | 10 |
| 85 | 63 | 23 | 2 | 344 | 464 | 10 |
| 86 | 21 | 24 | 28 | 349 | 469 | 10 |
| 87 | 12 | 24 | 13 | 359 | 479 | 10 |
| 88 | 24 | 58 | 19 | 260 | 380 | 10 |
| 89 | 67 | 5 | 25 | 713 | 833 | 10 |
| 90 | 37 | 47 | 6 | 359 | 479 | 10 |
| 91 | 49 | 42 | 13 | 719 | 839 | 10 |
| 92 | 53 | 43 | 14 | 14 | 134 | 10 |
| 93 | 61 | 52 | 3 | 808 | 928 | 10 |
| 94 | 57 | 48 | 23 | 392 | 512 | 10 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 95 | 56 | 37 | 6 | 100 | 220 | 10 |
| 96 | 55 | 54 | 26 | 562 | 682 | 10 |
| 97 | 4 | 18 | 35 | 549 | 667 | 10 |
| 98 | 26 | 52 | 9 | 192 | 292 | 10 |
| 99 | 26 | 35 | 15 | 308 | 428 | 10 |
| 100 | 31 | 67 | 3 | 810 | 930 | 10 |

Στον Πίνακα 5 παρουσιάζονται οι βέλτιστες λύσεις για το παραπάνω σύνολο δεδομένων. Εδώ πρέπει να επισημανθεί πως οι λύσεις αυτές σχετίζονται με την επίλυση του προβλήματος δρομολόγησης οχημάτων με χρονικά παράθυρα, ωστόσο εφόσον δεν λαμβάνεται υπόψη ο περιορισμός για ομοιόμορφη κατανομή των φορτίων σε όλα τα φορτηγά, υπάρχει διαφοροποίηση στα αποτελέσματα των παρακάτω μεθόδων σε σχέση με αυτά των αλγορίθμων της παρούσας διπλωματικής εργασίας [10].

*Πίνακας 5. Βέλτιστες λύσεις του προβλήματος RC201*

|  |  |  |  |
| --- | --- | --- | --- |
| Βέλτιστες Λύσεις του προβλήματος RC201 | | | |
| Αριθμός Πελατών | Αριθμός Οχημάτων | Απόσταση | Συγγραφείς |
| 25 | 3 | 360,2 | CR + L |
| 50 | 5 | 684,8 | L + KLM |
| 100 | 9 | 1261,8 | KLM |

Οι συγγραφείς που αναφέρονται στα 3 σύνολα λύσεων του Πίνακα 5 είναι οι:

* + - CR - W. Cook and J. L. Rich [4].
    - KLM - B. Kallehauge, J. Larsen, and O.B.G. Madsen [5].
    - L - J. Larsen [6].

## ΑΞΙΟΛΟΓΗΣΗ ΠΡΟΣΕΓΓΙΣΕΩΝ

Για να εξεταστεί η ορθή λειτουργία των διαφορετικών προσεγγίσεων καθώς και η ποιότητα των λύσεων εξετάσθηκαν οι αλγόριθμοι σε τριών ειδών προβλήματα με διαφορετικά σύνολα δεδομένων:

* + - Δοκιμή με **Μικρό** σύνολο δεδομένων: 25 κόμβοι και 1 αποθήκη
    - Δοκιμή με **Μεσαίο** σύνολο δεδομένων: 50 κόμβοι και 1 αποθήκη
    - Δοκιμή με **Μεγάλο** σύνολο δεδομένων 100 κόμβοι και 1 αποθήκη

Αξίζει να σημειωθεί πως κατά την υλοποίηση των παραπάνω προβλημάτων έγιναν κάποιες αλλαγές στα δεδομένα του Solomon σχετικά με τα χαρακτηριστικά των οχημάτων. Οι εν λόγω αλλαγές πραγματοποιήθηκαν έτσι ώστε να ικανοποιείται ο παράγοντας που σχετίζεται με την ομοιόμορφη κατανομή των φορτίων σε όλα τα χρησιμοποιούμενα οχήματα. Η υλοποίηση της ομαδοποίησης των κόμβων σε μια σειρά από συστάδες ανάλογα με την χωρητικότητα και το διαθέσιμο πλήθος οχημάτων γίνεται μέσω του *Αλγορίθμου 2* και τα αποτελέσματα που εξάγονται προσαρμόζονται στις ανάγκες του *Αλγορίθμου 3* έτσι ώστε στη συνέχεια να πραγματοποιηθεί η δρομολόγηση των οχημάτων. Για κάθε σύνολο δεδομένων τα χαρακτηριστικά αναλύονται στις ενότητες 5.2.1, 5.2.2, και 5.2.3, ενώ τα αποτελέσματα και οι συστάδες που δημιουργούνται για κάθε σενάριο που μελετάται παρουσιάζονται εκτενώς στην ενότητα 5.4.

Σο σύνολο δεδομένων του Solomon, προτείνεται η χωρητικότητα όλων των φορτηγών να έχει την τιμή των 1000 μονάδων προϊόντων. Κατά την πειραματική υλοποίηση των αλγορίθμων η τιμή αυτή προσαρμόστηκε ανάλογα με τα δεδομένα του κάθε σεναρίου, δηλαδή ανάλογα με τη συνολική ζήτηση των κόμβων που πρέπει να εξυπηρετηθούν, τον αρχικό αριθμό οχημάτων, το πλήθος των συστάδων που δημιουργούνται καθώς και τον αριθμό των επιπλέον οχημάτων που ενδέχεται να χρησιμοποιηθούν αν δεν καλυφθεί η ζήτηση. Πιο συγκεκριμένα η χωρητικότητα των φορτηγών προκύπτει από τις σχέσεις.

max 𝑑𝑖 , 𝑖 = 1, 2, … , 𝑛 (5.1)

min 𝑑𝑖 , 𝑖 = 1, 2, … , 𝑛 (5.2)

∑𝑛

𝑑𝑖

max 𝐶𝑎𝑝𝑎𝑐𝑖𝑡𝑦 : 𝐶′ = [ 𝑖=1 ] + (max 𝑑

𝛫𝑚 𝑖

− min 𝑑𝑖) (5.3)

Όπου max 𝑑𝑖 και min 𝑑𝑖 αναφέρεται στον μέγιστο και ελάχιστο αντίστοιχα αριθμό μονάδων ζήτησης του δείγματος και 𝛫𝑚 ο αρχικός αριθμός των οχημάτων που προβλέπεται να χρησιμοποιηθούν ώστε να εξυπηρετηθούν οι υποψήφιοι πελάτες.

## ΜΙΚΡΟ ΣΥΝΟΛΟ ΔΕΔΟΜΕΝΩΝ

Η αρχική εκτέλεση του κώδικα έγινε με τις εξής τιμές των παραμέτρων:

* + - Κόμβοι εξυπηρέτησης: 25
    - Κεντρική αποθήκη: 1
    - Μέγιστος αριθμός οχημάτων: 5

Για το παραπάνω σύνολο δεδομένων εξετάστηκαν 4 σενάρια με διαφοροποίηση στον αριθμό ομάδων που επρόκειτο να δημιουργηθούν και τη χωρητικότητα κάθε οχήματος. Πιο αναλυτικά τα 4 σενάρια:

Σενάριο 1

* + - Συστάδες κόμβων προς εξυπηρέτηση: 1
    - Μέγιστη χωρητικότητα κάθε οχήματος: 540 (αριθμός ίσος με τη συνολική ζήτηση των κόμβων προς εξυπηρέτηση, εφόσον οι τιμές ανατίθενται σε μόνο μία συστάδα)
    - Αρχικά οχήματα: 1
    - Επιπλέον οχήματα: 4 Σενάριο 2
    - Συστάδες κόμβων προς εξυπηρέτηση: 2
    - Μέγιστη χωρητικότητα κάθε οχήματος: 300
    - Αρχικά οχήματα: 2
    - Επιπλέον οχήματα: 3 Σενάριο 3
    - Συστάδες κόμβων προς εξυπηρέτηση: 3
    - Μέγιστη χωρητικότητα κάθε οχήματος: 210
    - Αρχικά οχήματα: 3
    - Επιπλέον οχήματα: 2 Σενάριο 4
    - Συστάδες κόμβων προς εξυπηρέτηση: 4
    - Μέγιστη χωρητικότητα κάθε οχήματος: 160
    - Αρχικά οχήματα: 4
    - Επιπλέον οχήματα: 1

Συνοπτικά παρατίθεται ο *Πίνακας 6* με τα στοιχεία των 4 σεναρίων.

*Πίνακας 6. Στοιχεία Μικρού Συνόλου Δεδομένων.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Μικρό Σύνολο Δεδομένων | | | | |
| Παρατηρήσεις | Σενάριο 1 | Σενάριο 2 | Σενάριο 3 | Σενάριο 4 |
| Κόμβοι | 25 | 25 | 25 | 25 |
| Αρχικές Συστάδες | 1 | 2 | 3 | 4 |
| Μέγιστη Χωρητικότητα Οχήματος | 540 | 300 | 210 | 160 |
| Αρχικά Οχήματα | 1 | 2 | 3 | 4 |
| Επιπλέον Οχήματα | 4 | 3 | 2 | 1 |

## ΜΕΣΑΙΟΥ ΣΥΝΟΛΟ ΔΕΔΟΜΕΝΩΝ

Η αρχική εκτέλεση του κώδικα έγινε με τις εξής τιμές των παραμέτρων:

* + - Κόμβοι εξυπηρέτησης: 50
    - Κεντρική αποθήκη: 1
    - Μέγιστος αριθμός οχημάτων: 7

Για το παραπάνω σύνολο δεδομένων εξετάστηκαν 4 σενάρια με διαφοροποίηση στον αριθμό ομάδων που επρόκειτο να δημιουργηθούν και τη χωρητικότητα κάθε οχήματος. Πιο αναλυτικά τα 4 σενάρια:

Σενάριο 5

* + - Συστάδες κόμβων προς εξυπηρέτηση: 3
    - Μέγιστη χωρητικότητα κάθε οχήματος: 340
    - Αρχικά οχήματα: 3
    - Επιπλέον οχήματα: 4 Σενάριο 6
    - Συστάδες κόμβων προς εξυπηρέτηση: 4
    - Μέγιστη χωρητικότητα κάθε οχήματος: 260
    - Αρχικά οχήματα: 4
    - Επιπλέον οχήματα: 3 Σενάριο 7
    - Συστάδες κόμβων προς εξυπηρέτηση: 5
    - Μέγιστη χωρητικότητα κάθε οχήματος: 220
    - Αρχικά οχήματα: 5
    - Επιπλέον οχήματα: 2

Σενάριο 8

* + - Συστάδες κόμβων προς εξυπηρέτηση: 6
    - Μέγιστη χωρητικότητα κάθε οχήματος: 190
    - Αρχικά οχήματα: 6
    - Επιπλέον οχήματα: 1

Συνοπτικά παρατίθεται ο *Πίνακας 7* με τα στοιχεία των 4 σεναρίων.

*Πίνακας 7. Στοιχεία Μεσαίου Συνόλου Δεδομένων*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Μεσαίο Σύνολο Δεδομένων | | | | |
| Παρατηρήσεις | Σενάριο 5 | Σενάριο 6 | Σενάριο 7 | Σενάριο 8 |
| Κόμβοι | 50 | 50 | 50 | 50 |
| Αρχικές Συστάδες | 3 | 4 | 5 | 6 |
| Μέγιστη Χωρητικότητα Οχήματος | 340 | 260 | 220 | 190 |
| Αρχικά Οχήματα | 3 | 4 | 5 | 6 |
| Επιπλέον Οχήματα | 4 | 3 | 2 | 1 |

## ΜΕΓΑΛΟ ΣΥΝΟΛΟ ΔΕΔΟΜΕΝΩΝ

Η αρχική εκτέλεση του κώδικα έγινε με τις εξής τιμές των παραμέτρων:

* + - Κόμβοι εξυπηρέτησης: 100
    - Κεντρική αποθήκη: 1
    - Μέγιστος αριθμός οχημάτων: 11

Για το παραπάνω σύνολο δεδομένων εξετάστηκαν 4 σενάρια με διαφοροποίηση στον αριθμό ομάδων που επρόκειτο να δημιουργηθούν και τη χωρητικότητα κάθε οχήματος. Πιο αναλυτικά τα 4 σενάρια:

Σενάριο 9

* + - Συστάδες κόμβων προς εξυπηρέτηση: 7
    - Μέγιστη χωρητικότητα κάθε οχήματος: 280
    - Αρχικά οχήματα: 7
    - Επιπλέον οχήματα: 4 Σενάριο 10
    - Συστάδες κόμβων προς εξυπηρέτηση: 8
    - Μέγιστη χωρητικότητα κάθε οχήματος: 240
    - Αρχικά οχήματα: 8
    - Επιπλέον οχήματα: 3 Σενάριο 11
    - Συστάδες κόμβων προς εξυπηρέτηση: 9
    - Μέγιστη χωρητικότητα κάθε οχήματος: 220
    - Αρχικά οχήματα: 9
    - Επιπλέον οχήματα: 2 Σενάριο 12
    - Συστάδες κόμβων προς εξυπηρέτηση: 10
    - Μέγιστη χωρητικότητα κάθε οχήματος: 200
    - Αρχικά οχήματα: 10
    - Επιπλέον οχήματα: 1

Συνοπτικά παρατίθεται ο *Πίνακας 8* με τα στοιχεία των 4 σεναρίων.

*Πίνακας 8. Στοιχεία Μεγάλου Συνόλου Δεδομένων*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Μεγάλο Σύνολο Δεδομένων | | | | |
| Παρατηρήσεις | Σενάριο 5 | Σενάριο 6 | Σενάριο 7 | Σενάριο 8 |
| Κόμβοι | 100 | 100 | 100 | 100 |
| Αρχικές Συστάδες | 7 | 8 | 9 | 10 |
| Μέγιστη Χωρητικότητα Οχήματος | 280 | 240 | 220 | 200 |
| Αρχικά Οχήματα | 7 | 8 | 9 | 10 |
| Επιπλέον Οχήματα | 4 | 3 | 2 | 1 |

Στην επόμενη ενότητα παρουσιάζονται τα αποτελέσματα δοκιμής των αλγορίθμων για τα παραπάνω σύνολα δεδομένων.

## ΠΑΡΑΤΗΡΗΣΕΙΣ, ΠΑΡΑΔΟΧΕΣ ΚΑΙ ΤΡΟΠΟΣ ΠΑΡΟΥΣΙΑΣΗΣ ΑΠΟΤΕΛΕΣΜΑΤΩΝ

Σε αυτήν την ενότητα παρουσιάζονται οι πέντε προσεγγίσεις με τις οποίες εξετάστηκε η αποτελεσματικότητα των αλγορίθμων για τα διαφορετικά σύνολα δεδομένων, καθώς και μέρος των παραδοχών που λήφθηκαν υπόψη. Για την εξαγωγή των κατάλληλων συμπερασμάτων σχετικά με την προσέγγιση που προτιμάται σε κάθε σενάριο με διαφορετικά χαρακτηριστικά, παρουσιάζεται μια σειρά αποτελεσμάτων με μορφή πινάκων που αφορούν τα δεδομένα που εξήχθησαν κατά την εφαρμογή της κάθε προσέγγισης.

Πιο αναλυτικά, σε κάθε σενάριο παρουσιάζονται μια σειρά από πίνακες που σχετίζονται με:

1. Την ομαδοποίηση και τις συστάδες που παράγονται από την χρήση του

*Αλγορίθμου 2* σε κάθε σύνολο δεδομένων.

1. Τα στοιχεία των αρχικών και των επιπρόσθετων οχημάτων που εξυπηρετούν την κάθε συστάδα.
2. Δείκτες σχετικοί με τα χαρακτηριστικά της φύσης του προβλήματος (χρόνος, απόσταση, φορτίο, κλπ.) καθώς και έλεγχος την ομοιογένειας κάθε συνόλου εξυπηρετούμενων κόμβων.
3. Συγκεντρωτικά αποτελέσματα σύγκρισης των 5 προσεγγίσεων και καταγραφή της βέλτιστης, ανάλογα με τα διαφορετικά χαρακτηριστικά του κάθε συνόλου δεδομένων. Ακόμη υπολογίζεται το επίπεδο της περαιτέρω επιβάρυνσης κάθε κατηγορίας σε περίπτωση επιλογής άλλης μεθόδου.

## ΑΝΑΛΥΣΗ ΤΡΟΠΩΝ ΥΛΟΠΟΙΗΣΗΣ ΤΩΝ ΔΟΚΙΜΩΝ

Για την υλοποίηση των δοκιμών χρησιμοποιήθηκαν τρία σύνολα δεδομένων (μικρό, μεσαίο και μεγάλο) όπως αναλύθηκε στις 3 προηγούμενες ενότητες και ανάλογα με τα χαρακτηριστικά των κόμβων και των οχημάτων που είναι διαθέσιμα να χρησιμοποιηθούν σε κάθε περίπτωση, δημιουργήθηκαν 12 σενάρια προς δοκιμή. Σε κάθε ένα από αυτά τα σενάρια γίνεται έλεγχος των δεδομένων και εξετάζονται τρόποι επίλυσής τους με πέντε διαφορετικές προσεγγίσεις. Πιο αναλυτικά αυτές οι προσεγγίσεις:

**Προσέγγιση 1**: Ο χρόνος ολοκλήρωσης των διαδρομών ισούται με το χρόνο της διαδρομής που θα εξυπηρετήσει τον μεγαλύτερο αριθμό πελατών χρησιμοποιώντας ως κύριο κριτήριο αυτό της συντομότερης διαδρομής. Η εφαρμογή του κριτήριου προτεραιότητας στην εξυπηρέτηση των εγγύτερων κόμβων και στη συνέχεια των υπολοίπων υλοποιείται με τη χρήση του *Αλγορίθμου 4* εντός μιας σειράς επαναλήψεων του *Αλγορίθμου 3* ο οποίος είναι ο κύριος Αλγόριθμος δρομολόγησης Οχημάτων.

**Προσέγγιση 2**: Ο χρόνος ολοκλήρωσης των διαδρομών ισούται με το χρόνο της διαδρομής που θα εξυπηρετήσει το μεγαλύτερο αριθμό πελατών χρησιμοποιώντας ως κύριο κριτήριο αυτό της συντομότερης διαδρομής όπως και στην *Προσέγγιση 1*, περιέχοντας όμως και μία μικρή παραλλαγή. Σε αυτήν την προσέγγιση η διαφοροποίηση σχετίζεται με τη σειρά εξυπηρέτησης των κόμβων. Το κάθε όχημα καλείται να εξυπηρετήσει σε πρώτη φάση τους μακρινότερους κόμβους της κάθε συστάδας κάνοντας χρήση του *Αλγορίθμου 5*, και στη συνέχεια ακολουθεί την όμοια διαδικασία με αυτήν της *Προσέγγισης 1*, όπου κύρια προτεραιότητα προς εξυπηρέτηση δίνεται σε κάθε βήμα στους πλησιέστερους κόμβους.

**Προσέγγιση 3**: Ο χρόνος ολοκλήρωσης της συγκεκριμένης προσέγγισης ισούται με το χρόνο της διαδρομής που θα εξυπηρετήσει το μεγαλύτερο αριθμό πελατών χρησιμοποιώντας ως κύριο κριτήριο αυτό της ελαχιστοποίησης του χρόνου αναμονής στο σύνολο των πελατών που θα επιλεχθούν για κάθε διαδρομή. Η εφαρμογή του συγκεκριμένου κριτηρίου υλοποιείται με τη χρήση του *Αλγορίθμου 6* σε μια σειρά επαναλήψεων εντός του *Αλγορίθμου 3*.

**Προσέγγιση 4**: Ο χρόνος ολοκλήρωσης της παρούσας περίπτωσης δρομολόγησης ισούται με το χρόνο της διαδρομής που θα εξυπηρετήσει τον μεγαλύτερο αριθμό πελατών με κύριο χαρακτηριστικό την ποσότητα του φορτίου τους. Στην *Προσέγγιση 4* δίνεται προτεραιότητα στην εξυπηρέτηση πελατών οι οποίοι έχουν φορτίο χαμηλής ποσότητας. Η παρούσα προσέγγιση θα μπορούσε να παρομοιαστεί με περιπτώσεις εταιρειών μεταφορών που έχουν να εξυπηρετήσουν μεγάλο αριθμό παραγγελιών. Έτσι φορτώνουν τα φορτηγά τους σε πρώτη φάση με δέματα μικρού όγκου, ώστε να εξυπηρετήσουν όσο το δυνατόν περισσότερους πελάτες, και στη συνέχεια, σε περίπτωση ύπαρξης υπολειπόμενου χώρου και χρόνου παράδοσης, εισάγεται μέρος των παραγγελιών με μεγάλο όγκο. Όπως και στις προηγούμενες 3 προσεγγίσεις η υλοποίηση γίνεται μέσω μιας επαναληπτικής διαδικασίας εντός του *Αλγορίθμου 3*, στην οποία καλείται ο *Αλγόριθμος 7* για τη δρομολόγηση του κάθε οχήματος με τα παραπάνω αναφερόμενα κριτήρια. Και τέλος,

**Προσέγγιση 5**: Η παρούσα προσέγγιση αποτελεί την ακριβώς αντίθετη περίπτωση από αυτήν που αναλύθηκε στην προηγούμενη παράγραφο. Όπως και πριν, έτσι και εδώ ο χρόνος ολοκλήρωσης της εκάστοτε διαδρομής ισούται με τον χρόνο της διαδρομής που θα εξυπηρετήσει το μεγαλύτερο αριθμό πελατών με κύριο χαρακτηριστικό την ποσότητα του φορτίου τους. Σε αυτή την περίπτωση όμως δίνεται προτεραιότητα στους κόμβους που διαθέτουν μεγάλο φορτίο. Αντίστοιχα αν έπρεπε να παρομοιαστεί με κάποιο πραγματικό σενάριο και πάλι θα μπορούσε να αναφερθεί η περίπτωση μεταφορών παραγγελιών προϊόντων ίδιου τύπου, όπου γεμίζει σε πρώτη φάση το δρομολόγιο με

παραγγελίες μεγάλου όγκου, ώστε να αποφευχθεί αργότερα η πιθανότητα να μην μπορέσει να εξυπηρετηθεί κάποιος πελάτης μεγάλης ζήτησης λόγω περιορισμένου χώρου στα οχήματα. Ο χρόνος ολοκλήρωσης της *Προσέγγισης 5* υπολογίζεται με τη χρήση του *Αλγορίθμου 8*.

Σύμφωνα με τις πέντε παραπάνω προσεγγίσεις, στις επόμενες ενότητες παρουσιάζονται τα πειραματικά αποτελέσματα των αλγορίθμων που αναπτύχθηκαν στα πλαίσια της υπολογιστικής εμπειρίας και εφαρμόστηκαν στο σύνολο δεδομένων RC201 για να ελεγχθεί η ορθότητα και η αποτελεσματικότητά τους, η διάρκεια εκτέλεσης και η γενική τους συμπεριφορά σε σύνθετα γραφήματα. Κυρίως όμως, για να εξαχθούν δεδομένα και δείκτες που θα συμβάλουν στην εξαγωγή χρήσιμων συμπερασμάτων που σχετίζονται με τις παραμέτρους που καθορίζουν την κάθε προσέγγιση. Τα αποτελέσματα αυτά αφορούν:

* + - 1. Την ικανοποίηση του κριτηρίου ομοιόμορφης κατανομής του φορτίου σε όλα τα φορτηγά, αλλά και σύγκριση των προσεγγίσεων μεταξύ τους.
      2. Τον έλεγχο του επιπέδου εξυπηρέτησης των κόμβων, καθώς και την επιλογή της καταλληλότερης μεθόδου σε κάθε περίπτωση.
      3. Τον έλεγχο της προσέγγισης με τη βέλτιστη απόσταση δρομολογίου, και παράλληλα την επίτευξη ομοιόμορφης κατανομής στο σύνολο των αποστάσεων που διανύουν όλα τα οχήματα της κάθε προσέγγισης, ώστε να μην επιβαρύνονται συγκεκριμένα οχήματα και προσωπικό περισσότερο από τα υπόλοιπα.
      4. Τον έλεγχο του χρόνου αναμονής ανάλογα με την κάθε προσέγγιση, καθώς και σύγκριση μεταξύ τους.
      5. Την ικανοποίηση του κριτηρίου ικανοποίηση της μέγιστης ζήτησης στα επιπρόσθετα οχήματα, εφόσον έχουν παραμείνει κόμβοι που δεν έχουν εξυπηρετηθεί από τον αρχικό στόλο οχημάτων. Ακόμη εξετάζεται η καταλληλότερη προσέγγιση για την κάλυψη μεγαλύτερων αναγκών.
      6. Και τέλος, τον πραγματικό χρόνο εκτέλεσης του αλγορίθμου.

## ΑΝΑΛΥΣΗ ΑΠΟΤΕΛΕΣΜΑΤΩΝ

Σε κάθε μια από τις προσεγγίσεις ο χρόνος ολοκλήρωσης επηρεάζεται από την απόσταση μεταξύ των σημείων, το χρόνο αναμονής για εξυπηρέτηση των κόμβων κάθε διαδρομής, το χρόνο εξυπηρέτησης καθώς και το χρόνο επιστροφής κάθε οχήματος στην αποθήκη πριν την λήξη του ωραρίου το οποίο ορίζεται ως ο αργότερος χρόνος εξυπηρέτησης του σημείου 0 (depot). Ακόμη έγιναν κάποιες παραδοχές λόγω του μεγάλου φάσματος των τρόπων επίλυσης, ώστε να περιοριστεί το πρόβλημα και να παραχθούν μια σειρά από

αποτελέσματα ικανά να συγκριθούν μεταξύ τους και να επιφέρουν μια σειρά από συμπεράσματα. Πιο αναλυτικά:

* + - 1. Σε κάθε σύνολο δεδομένων γίνεται αξιολόγηση της λύσης με κάθε μια από τις 5 προσεγγίσεις που αναφέρθηκαν παραπάνω. Κάθε προσέγγιση δρομολόγησης των οχημάτων για εξυπηρέτηση των πελατών, χρησιμοποιεί σε πρώτη φάση το σύνολο των αρχικών οχημάτων που ορίζονται σε κάθε σενάριο, και στη συνέχεια σε περίπτωση που δεν καλυφθούν όλοι οι κόμβοι γίνεται χρήση των επιπλέον οχημάτων.
      2. Στην περίπτωση των επιπλέον οχημάτων δίνεται η δυνατότητα φόρτωσης μεγαλύτερων ποσοτήτων (ενδεικτικά δοκιμάστηκε στην περίπτωσή των προσεγγίσεων της παρούσας διπλωματικής εργασίας αύξηση της τάξης του 20%) από αυτές που έχουν οριστεί κατά την ομαδοποίηση σε κάθε συνόλου δεδομένων του προβλήματος, αποσκοπώντας στην εξυπηρέτηση όσο το δυνατόν περισσότερων από τους υπολειπόμενους πελάτες. Γι’ αυτόν το λόγο και στον πίνακα αποτελεσμάτων κάθε σεναρίου, εξετάζεται η ομοιογένεια της κατανομής του φορτίου: (α) στο σύνολο των οχημάτων που χρησιμοποιούνται, (β) στα αρχικά οχήματα ξεχωριστά και (γ) στα επιπλέον οχήματα ξεχωριστά.
      3. Σε καθένα εκ των τριών συνόλων δεδομένων, 25, 50 και 100 κόμβοι, εξετάζονται από 4 σενάρια με συνολικό αριθμό οχημάτων 5, 7 και 11 αντίστοιχα. Οι παραπάνω αριθμοί προέκυψαν έπειτα από κάποιες δοκιμές στο σύνολο των δεδομένων όπου και παρατηρήθηκε η ανάγκη επιπλέον οχήματος ή οχημάτων σε όλες τις περιπτώσεις. Από τον *Πίνακα 5* με τα στοιχεία των βέλτιστων λύσεων σε παρεμφερές πρόβλημα είχαν χρησιμοποιηθεί 3, 5 και 9 οχήματα στις αντίστοιχες περιπτώσεις, οπότε για να υπάρχει κάποιο πιθανό μέτρο σύγκρισης ο αριθμός των οχημάτων που χρησιμοποιήθηκε στις δοκιμές κρίθηκε πως πρέπει να είναι κοντά σε αυτές τις 3 τιμές για κάθε σύνολο δεδομένων. Έτσι σε κάθε περίπτωση ορίστηκε ο μέγιστος αριθμός του συνόλου των οχημάτων που θα χρησιμοποιηθούν να μην υπερβαίνει αυτό των βέλτιστων λύσεων κατά 2 μονάδες.

Για την καλύτερη παρουσίαση των αποτελεσμάτων έγινε χρήση του προγράμματος Microsoft Excel. Για την εξέταση της ομοιογένειας κατά την κατανομή των τιμών σε κατηγορίες όπως το φορτίο κάθε οχήματος, το χρόνο αναμονής κάθε οχήματος, την απόσταση κάθε διαδρομής κ.α., εξετάσθηκαν τα δείγματα ως προς την διακύμανση και την τυπική απόκλιση των τιμών τους με τις συναρτήσεις VAR και STDEV. Με την βοήθεια των συναρτήσεων αλλά και τη χρήση μιας σειράς μορφοποιήσεων του προγράμματος με τη βοήθεια χρωματικών διαχωρισμών (Conditional Formatting >>

Color Scales) πραγματοποιήθηκε η σύγκριση των πέντε προσεγγίσεων σε κάθε σενάριο ξεχωριστά, ώστε να παραχθούν μια σειρά από αποτελέσματα τα οποία και παρουσιάζονται στις ενότητες 5.4.1, 5.4.2 και 5.4.3 αναλυτικά, όπως και στο Κεφάλαιο 7 με τα Συμπεράσματα της παρούσας διπλωματικής εργασίας. Σχετικά με τη χρήση χρωμάτων στα αποτελέσματα του κάθε σεναρίου, παρατίθενται οι παρακάτω διευκρινιστικές λεπτομέρειες:

* + - * + Γίνεται η χρήση μιας κλίμακας χρωμάτων από πράσινο (καλή - επιθυμητή λύση) έως κόκκινο (κακή – μη επιθυμητή λύση) με τις ενδιάμεσες τιμές να έχουν έντονο το κίτρινο στοιχείο. Λύσεις με χρωματολογία κίτρινο προς πράσινο θεωρούνται μέτριες προς καλές και αντίστοιχα κίτρινο προς κόκκινο (ή πορτοκαλί σε πολλές περιπτώσεις) θεωρούνται μέτριες προς κακές οι οποίες είναι και πάλι μη επιθυμητές. Τέλος σε ορισμένες περιπτώσεις γίνεται η χρήση του μπλε χρώματος, κάτι το οποίο θα αναφερθεί παρακάτω.
        + Σχετικά με το κριτήριο που λαμβάνεται υπόψη κατά το σχεδιασμό των συστάδων, δηλαδή το φορτίο κάθε οχήματος και την ανάγκη να είναι κατανεμημένο ομοιόμορφα, στα δεδομένα που εξάγονται κατά τις δοκιμές στα 12 σενάρια έγινε σε 2 σημεία η χρήση χρωμάτων. Αρχικά σε κάθε προσέγγιση στο σύνολο των φορτίων που μεταφέρεται από κάθε όχημα και στη συνέχεια, έπειτα από τον υπολογισμό των διακυμάνσεων και των τυπικών αποκλίσεων κάθε προσέγγισης, για την επιλογή της καλύτερης προσέγγισης. Από την πειραματική αυτή διαδικασία στο αρχικό στάδιο της μελέτης κάθε προσέγγισης, παρατηρείται οπτικά τουλάχιστον σε πρώτη φάση, στο σύνολο των διαδρομών παρεμφερής χρωματολογία, προσδιορίζοντας πως όλα τα δρομολόγια έχουν τιμές που διαφέρουν λίγο και ικανοποιούν το κριτήριο. Στη συνέχεια γίνεται ανάλυση των αποτελεσμάτων κάθε μεθόδου για να αξιολογηθεί εκτενέστερα η ομοιογένεια στην κατανομή, καθώς και η σύγκριση της κάθε προσέγγισης με τις υπόλοιπες για το κάθε σενάριο και σύνολο δεδομένων.
        + Σε συγκεκριμένες λύσεις εμφανίζονται κάποια από τα επιπλέον οχήματα με φορτίο μπλε χρώματος. Ο μπλε χρωματισμός προσδιορίζει πως το συγκεκριμένο όχημα συμπεριλαμβάνεται στην περίπτωση των επιπρόσθετων οχημάτων που το φορτίο τους υπερβαίνει κατά 20% (όπως αναφέρθηκε παραπάνω) το αρχικά υπολογισμένο φορτίο για το αντίστοιχο σύνολο δρομολογίων σε κάθε περίπτωση.

## 5.4 ΠΙΝΑΚΕΣ ΑΠΟΤΕΛΕΣΜΑΤΩΝ

Σε κάθε σενάριο παρουσιάζονται αρχικά οι πίνακες με την ομαδοποίηση των δεδομένων που προέκυψαν από τον *Αλγόριθμο 2* - ομαδοποίησης με τη μέθοδο Improved k-means. Στη συνέχεια παρουσιάζονται οι τιμές που προέκυψαν από την εφαρμογή των πέντε προσεγγίσεων για κάθε ένα από τα σύνολα δεδομένων. Έπειτα από την εξαγωγή των αποτελεσμάτων γίνεται σχετική σύγκριση στο σύνολο των τιμών που σχετίζονται με το φορτίο του κάθε οχήματος ώστε να παρατηρηθεί αν και κατά πόσο έχει κατανεμηθεί ομοιογενώς. Η χρωματολογία σε όλες τις ενδείξεις που προκύπτει από το Conditional Formatting του Excel συμβάλλει στην ευκολότερη κατανόηση μέρους των αποτελεσμάτων. Σε κάθε σενάριο, έπειτα από τους πίνακες αποτελεσμάτων, παρατίθενται μια σειρά από συγκρίσεις μεταξύ των προσεγγίσεων για κάθε σύνολο δεδομένων αποσκοπώντας στην επιλογή του ή των καλύτερων για κάθε περίπτωση.

## 5.4.1 ΑΠΟΤΕΛΕΣΜΑΤΑ ΜΙΚΡΟΥ ΣΥΝΟΛΟΥ ΔΕΔΟΜΕΝΩΝ Σενάριο 1

Η ομαδοποίηση των κόμβων σύμφωνα με το **Σενάριο 1**:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Σενάριο 1 | | | | | | | |
| Συστάδες | Κόμβοι | | | | | | |
|  | 0 | 21 | 3 | 8 | 13 | 14 | 20 |
| 1 | 24  1 | 25  7 | 18  9 | 16  6 | 17  12 | 5  23 | 15  10 |
|  | 2 | 19 | 4 | 22 | 11 |  |  |

Οι πίνακες αποτελεσμάτων των 5 αλγορίθμων:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Σενάριο 1 | Προσέγγιση 1: Min Distance | | | | | Προσέγγιση 2: Inv. Dijkstra | | | | |
| Time + Dist. | Wait Time | Demand | Nodes Served | Travel Dist. | Time + Dist. | Wait Time | Demand | Nodes Served | Travel Dist. |
| Cluster 1 | 830,38 568,12 | | 200,00 | 10,00 262,26 | | 760,77 552,34 | | 170,00 | 9,00 208,43 | |
| Sum | 830,38 568,12 200,00 10,00 262,26 | | | | | 760,77 552,34 170,00 9,00 208,43 | | | | |
| Extra Cluster 1 | 547,00 369,33  481,00 370,93  345,00 251,57 | | 200,00 | 8,00 177,67  3,00 110,08  4,00 93,44 | | 683,00 523,47  501,77 418,69  427,06 255,95  162,00 72,16 | | 170,00 | 6,00 159,53  3,00 83,08  5,00 171,11  2,00 89,84 | |
| Extra Cluster 2 | 40,00 | 50,00 |
| Extra Cluster 3 | 100,00 | 110,00 |
| Extra Cluster 4 | \_ \_ 0,00 \_ \_ | | | | | 40,00 |
| General Sum | 2203,38 1559,94 540,00 25,00 643,43 | | | | | 2534,59 1822,60 540,00 25,00 711,99 | | | | |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Σενάριο 1 | Προσέγγιση 3: Min Wait Time | | | | | Προσέγγιση 4: Min Served First | | | | |
| Time  + Dist. | Wait Time | Demand | Nodes Served | Travel Dist. | Time  + Dist. | Wait Time | Demand | Nodes Served | Travel Dist. |
| Cluster 1 | 803,32 | 382,66 | 210 | 8 | 420,66 | 794 | 515,16 | 60 | 5 | 278,84 |
| Sum | 803,32 | 382,66 | 210 | 8 | 420,66 | 794 | 515,16 | 60 | 5 | 278,84 |
| Extra Cluster 1 | 711,58  579,60  493,10  55,17 | 385,10  328,91  273,04  0 | 70 | 5  6  5  1 | 326,48  250,7  220,05  55,177 | 770,17  654  385  205 | 498,3  275,03  191,14  116,5 | 70 | 5  8  5  2 | 271,87  378,97  193,86  88,502 |
| Extra Cluster 2 | 150 | 190 |
| Extra Cluster 3 | 90 | 150 |
| Extra Cluster 4 | 20 | 70 |
| General Sum | 2642,8 | 1369,73 | 540 | 25 | 1273,06 | 2808,2 | 1596,1 | 540 | 25 | 1212,05 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Σενάριο 1 | Προσέγγιση 5: Max Served First | | | | |
| Time + Dist. | Wait Time | Demand | Nodes Served | Travel Dist. |
| Cluster 1 | 794 | 574,08 | 100 | 3 | 219,91 |
| Sum | 794 | 574,08 | 100 | 3 | 219,91 |
| Extra | 770,17  711,59  547  118,15 | 485,29  349,10  163,83  0 | 180 | 6  7  7  2 | 284,88  362,49  383,16  118,15 |
| Cluster 1 |
| Extra Cluster 2 | 120 |
| Extra Cluster 3 | 120 |
| Extra | 20 |
| Cluster 4 |
| General Sum | 2940,91 | 1572,32 | 540 | 25 | 1368,59 |

*Πίνακας 9. Σύγκριση των 5 Προσεγγίσεων που χρησιμοποιήθηκαν στο Σενάριο 1.*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ΥΠΟΛΟΓΙΣΜΟΙ ΜΕΤΑ ΤΗΝ ΟΛΟΚΛΗΡΩΣΗ ΤΩΝ ΔΡΟΜΟΛΟΓΙΩΝ** | | | | | |
| Παρατηρήσεις | Προσέγγιση 1 | Προσέγγιση 2 | Προσέγγιση 3 | Προσέγγιση 4 | Προσέγγιση 5 |
| **ΑΡΙΘΜΟΣ ΟΧΗΜΑΤΩΝ** | | | | | |
| Αρχικά Οχήματα | 1 | 1 | 1 | 1 | 1 |
| Επιπλέον Οχήματα | 3 | 4 | 4 | 4 | 4 |
| **ΑΡΙΘΜΟΣ ΚΟΜΒΩΝ ΠΟΥ ΕΞΥΠΗΡΕΤΗΘΗΚΑΝ** | | | | | |
| Αρχικοί Κόμβοι | 10 | 9 | 8 | 5 | 3 |
| Κόμβοι Επιπλέον Διαδρομών | 15 | 16 | 17 | 20 | 22 |
| Συνολικοί Κόμβοι | 25 | 25 | 25 | 25 | 25 |
| **ΖΗΤΗΣΗ ΑΝΑ ΟΧΗΜΑ** | | | | | |
| Συνολική Ζήτηση | 540 | 540 | 540 | 540 | 540 |
| Μέση Τιμή | 135 | 108 | 108 | 108 | 108 |
| Διακύμανση | 6233,33 | 3920,00 | 5420,00 | 3420,00 | 3320,00 |
| Τυπική Απόκλιση | 78,95 | 62,61 | 73,62 | 58,48 | 57,62 |
| Τυπική Απόκλιση Αρχικών Δρομολ. | Δεν διατίθεται για μέγεθος πληθυσμού μικρότερο των 2 σημείων | | | | |
| Τυπική Απόκλιση Επιπλέον Δρομολ. | 80,83 | 60,21 | 53,77 | 60,00 | 66,33 |
| **ΑΠΟΣΤΑΣΗ ΑΝΑ ΟΧΗΜΑ** | | | | | |
| Συνολική Απόσταση | 643,43 | 711,99 | 1273,06 | 1212,05 | 1368,59 |
| Μέση Τιμή | 160,86 | 142,40 | 254,61 | 242,41 | 273,72 |
| Διακύμανση | 5896,25 | 2939,85 | 18429,83 | 11722,34 | 11769,64 |
| Τυπική Απόκλιση | 76,79 | 54,22 | 135,76 | 108,27 | 108,49 |
| **ΑΝΑΜΟΝΗ ΑΝΑ ΟΧΗΜΑ** | | | | | |
| Συνολικός Χρόνος Αναμονής | 1559,94 | 1822,60 | 1369,73 | 1596,12 | 1572,32 |
| Μέση Τιμή | 389,99 | 364,52 | 273,95 | 319,22 | 314,46 |
| Διακύμανση | 17227,70 | 6233,33 | 10,92 | 5896,25 | 24035,45 |
| Τυπική Απόκλιση | 131,25 | 200,46 | 159,88 | 180,22 | 234,18 |
| **ΣΥΓΚΡΙΣΕΙΣ ΕΠΙΠΛΕΟΝ ΔΕΙΚΤΩΝ** | | | | | |
| Απόσταση Ανά Κόμβο | 25,74 | 28,48 | 50,92 | 48,48 | 54,74 |
| Απόσταση Ανά  Ζήτηση | 1,19 | 1,32 | 2,36 | 2,24 | 2,53 |
| Χρόνος Αναμονής Ανά Κόμβο | 62,40 | 72,90 | 54,79 | 63,84 | 62,89 |
| Χρόνος Αναμονής  Ανά Ζήτηση | 2,89 | 3,38 | 2,54 | 2,96 | 2,91 |

*Πίνακας 10. Σύγκριση Αποτελεσμάτων Σεναρίου 1.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Προσέγγιση 1 | Προσέγγιση 2 | Προσέγγιση 3 | Προσέγγιση 4 | Προσέγγιση 5 |
| **ΣΥΝΟΛΟ ΧΡΗΣΙΜΟΠΟΙΟΥΜΕΝΩΝ ΟΧΗΜΑΤΩΝ** | | | | |
| 40% | 36% | 32% | 20% | 12% |
| **ΠΟΣΟΣΤΟ ΕΞΥΠΗΡΕΤΗΣΗΣ ΚΟΜΒΩΝ ΜΕ ΤΗ ΧΡΗΣΗ ΤΩΝ ΑΡΧΙΚΩΝ ΟΧΗΜΑΤΩΝ** | | | | |
| 100% | 100% | 100% | 100% | 100% |
| **ΠΟΣΟΣΤΟ ΕΞΥΠΗΡΕΤΗΣΗΣ ΚΟΜΒΩΝ ΜΕ ΤΗΝ ΟΛΟΚΛΗΡΩΣΗ ΟΛΩΝ ΤΩΝ ΔΡΟΜΟΛΟΓΙΩΝ** | | | | |
| 100% | 100% | 100% | 100% | 100% |
| **ΚΑΤΑΝΟΜΗ ΦΟΡΤΙΟΥ ΣΤΟ ΣΥΝΟΛΟ ΤΩΝ ΔΙΑΔΡΟΜΩΝ** | | | | |
| +37,02% | +8,66% | +27,77% | +1,49% | Μικρότερη τυπική απόκλιση της τάξης των 57,62 μονάδων |
| **ΚΑΤΑΝΟΜΗ ΦΟΡΤΙΟΥ ΣΤΑ ΕΠΙΠΡΟΣΘΕΤΑ ΟΧΗΜΑΤΑ** | | | | |
| +50,31% | +11,96% | Μικρότερη τυπική απόκλιση της τάξης των 53,77 μονάδων | +11,58% | +23,35% |
| **ΣΥΝΟΛΙΚΗ ΑΠΟΣΤΑΣΗ ΔΡΟΜΟΛΟΓΙΩΝ** | | | | |
| Βέλτιστη λύση 643,43 μονάδες | +11% | +98% | +88% | +113% |
| **ΚΑΤΑΝΟΜΗ ΑΠΟΣΤΑΣΗΣ ΔΡΟΜΟΛΟΓΙΩΝ ΣΤΟ ΣΥΝΟΛΟ ΤΩΝ ΔΙΑΔΡΟΜΩΝ** | | | | |
| +41,62% | Μικρότερη τυπική απόκλιση της τάξης των 54,22 μονάδων | +150,38% | +99,68% | +100,09% |
| **ΣΥΝΟΛΙΚΟΣ ΧΡΟΝΟΣ ΑΝΑΜΟΝΗΣ ΔΡΟΜΟΛΟΓΙΩΝ** | | | | |
| +13,89% | +33,06% | Βέλτιστη λύση με αναμονή 1369,73 μονάδες χρόνου | +16,53% | +14,79% |
| **ΚΑΤΑΝΟΜΗ ΧΡΟΝΟΥ ΑΝΑΜΟΝΗΣ ΣΤΟ ΣΥΝΟΛΟ ΤΩΝ ΔΙΑΔΡΟΜΩΝ** | | | | |
| Μικρότερη τυπική απόκλιση της τάξης των 131,25 μονάδων | +52,73% | +21,81% | +37,31% | +78,42% |
| **ΑΠΟΣΤΑΣΗ ΠΟΥ ΔΙΑΝΎΕΤΑΙ ΑΝΑ ΕΞΥΠΗΡΕΤΟΥΜΕΝΟ ΚΟΜΒΟ** | | | | |
| Βέλτιστη λύση 25,74 μονάδες | +10,66% | +97,85% | +88,37% | +112,70% |
| **ΧΡΟΝΟΣ ΑΝΑΜΟΝΗΣ ΑΝΑ ΕΞΥΠΗΡΕΤΟΥΜΕΝΟ ΚΟΜΒΟ** | | | | |
| +13,89% | +33,06% | Βέλτιστη λύση 54,79 μονάδες | +16,53% | +14,79% |

90

## Σενάριο 2

Η ομαδοποίηση των κόμβων σύμφωνα με το **Σενάριο 2**:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Σενάριο 2 | | | | | | | |
| Συστάδες | Κόμβοι | | | | | | |
| 1 | 0 | 21 | 20 | 18 | 25 | 24 | 2 |
| 23 | 11 | 19 | 22 |  |  |  |
|  | 0 | 3 | 13 | 8 | 14 | 4 | 1 |
| 2 | 5 | 9 | 16 | 15 | 10 | 17 | 6 |
|  | 7 | 12 |  |  |  |  |  |

Οι πίνακες αποτελεσμάτων των 5 αλγορίθμων:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Σενάριο 2 | Προσέγγιση | | | 1: Min Distance | | | Προσέγγιση 2: Inv. Dijkstra | | | | |
| Time + Dist. | Wait Time | Demand | | Nodes Served | Travel Dist. | Time  + Dist. | Wait Time | Demand | Nodes Served | Travel Dist. |
| Cluster 1 | 794  780,38 | 636,00  620,19 | 100 | | 4  6 | 157,99  160,19 | 691  798,03 | 552,34  634,38 | 140 | 7  5 | 138,66  163,65 |
| Cluster 2 | 140 | | 90 |
| Sum | 1574,38 | 1256,2 | | 240 | 10 | 318,18 | 1489 | 1186,7 | 230 | 12 | 302,31 |
| Extra Cluster 1 | 563,80  345  50,31 | 302,26  241,23  0 | 140  140 | | 9  5  1 | 261,53  103,77  50,311 | 654  529  427,06 | 576,3  406,05  292,26 | 70  80 | 3  4  5 | 77,696  122,95  134,8 |
| Extra Cluster 2 |
| Extra Cluster 3 | 20 | | 130 |
| General Sum | 2533,49 | 1799,7 | | 540 | 25 | 733,79 | 3099,1 | 2461,3 | 510 | 24 | 637,76 |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Σενάριο 2 | Προσέγγιση: Min Wait Time | | | | |  | Προσέγγιση | | 4: Min Served First | | |
| Time + Dist. | Wait Time | Demand | Nodes Served | Travel Dist. | Time  + Dist. | Wait Time | Demand | | Nodes Served | Travel Dist. |
| Cluster 1 | 794  770,17 | 627,36  464,37 | 140 | 5  7 | 166,64  305,8 | 794  784,02 | 684,36  550,83 | 50  60 | | 4  4 | 109,64  233,19 |
| Cluster 2 | 130 |
| Sum | 1564,17 | 1091,73 | 270 | 12 | 472,44 | 1578 | 1235,2 | | 110 | 8 | 342,83 |
| Extra Cluster 1 | 720,43  597,61  423,28 | 397,23  355,45  273,05 | 90 | 5  5  3 | 323,2  242,16  150,23 | 654  399,34  355,25 | 289,41  155,44  76,51 | 150 | | 7  5  5 | 364,59  243,89  278,74 |
| Extra Cluster 2 | 130 | 120 | |
| Extra Cluster 3 | 50 | 160 | |
| General Sum | 3305,49 | 2117,47 | 540 | 25 | 1188 | 2986,6 | 1756,6 | | 540 | 25 | 1230 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Σενάριο 2 | Π | ροσέγγιση | | 5: Max Served First | | |
| Time + Dist. | Wait Time | Demand | | Nodes Served | Travel Dist. |
| Cluster 1 | 794 | 639,98 | 160 | | 5 | 154,03 |
| Cluster 2 | 761,35 | 528,28 | 120 | | 5 | 233,07 |
| Sum | 1555,35 | 1168,25 | | 280 | 10 | 387,1 |
| Extra | 691  481  377 | 377,94  289,90  184,33 | 120 | | 6  5  3 | 313,06  191,1  192,67 |
| Cluster 1 |
| Extra | 90 | |
| Cluster 2 |
| Extra | 40 | |
| Cluster 3 |
| General Sum | 3104,35 | 2020,42 | | 530 | 24 | 1083,9 |

*Πίνακας 11. Σύγκριση των 5 Προσεγγίσεων που χρησιμοποιήθηκαν στο Σενάριο 2.*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ΥΠΟΛΟΓΙΣΜΟΙ ΜΕΤΑ ΤΗΝ ΟΛΟΚΛΗΡΩΣΗ ΤΩΝ ΔΡΟΜΟΛΟΓΙΩΝ** | | | | | |
| Παρατηρήσεις | Προσέγγιση 1 | Προσέγγιση 2 | Προσέγγιση 3 | Προσέγγιση 4 | Προσέγγιση 5 |
| **ΑΡΙΘΜΟΣ ΟΧΗΜΑΤΩΝ** | | | | | |
| Αρχικά Οχήματα | 2 | 2 | 2 | 2 | 2 |
| Επιπλέον Οχήματα | 3 | 3 | 3 | 3 | 3 |
| **ΑΡΙΘΜΟΣ ΚΟΜΒΩΝ ΠΟΥ ΕΞΥΠΗΡΕΤΗΘΗΚΑΝ** | | | | | |
| Αρχικοί Κόμβοι | 10 | 12 | 12 | 8 | 10 |
| Κόμβοι Επιπλέον Διαδρομών | 15 | 12 | 13 | 17 | 14 |
| Συνολικοί Κόμβοι | 25 | 24 | 25 | 25 | 24 |
| **ΖΗΤΗΣΗ ΑΝΑ ΟΧΗΜΑ** | | | | | |
| Συνολική Ζήτηση | 540 | 510 | 540 | 540 | 530 |
| Μέση Τιμή | 108 | 102 | 108 | 108 | 106 |
| Διακύμανση | 2720,00 | 970,00 | 6880,00 | 2570,00 | 1980,00 |
| Τυπική Απόκλιση | 52,15 | 31,14 | 37,68 | 50,70 | 44,50 |
| Τυπική Απόκλιση Αρχικών Δρομολ. | 28,28 | 35,36 | 7,07 | 7,07 | 28,28 |
| Τυπική Απόκλιση Επιπλέον Δρομολ. | 69,28 | 32,15 | 40,00 | 20,82 | 40,41 |
| **ΑΠΟΣΤΑΣΗ ΑΝΑ ΟΧΗΜΑ** | | | | | |
| Συνολική Απόσταση | 733,79 | 637,76 | 1188,02 | 1230,05 | 1083,93 |
| Μέση Τιμή | 146,76 | 127,55 | 237,60 | 246,01 | 216,79 |
| Διακύμανση | 6157,38 | 996,40 | 6167,15 | 8474,48 | 3678,67 |
| Τυπική Απόκλιση | 78,47 | 31,57 | 78,53 | 92,06 | 60,65 |
| **ΑΝΑΜΟΝΗ ΑΝΑ ΟΧΗΜΑ** | | | | | |
| Συνολικός Χρόνος Αναμονής | 1799,70 | 2461,33 | 2117,47 | 1756,56 | 2020,42 |
| Μέση Τιμή | 359,94 | 492,27 | 423,49 | 351,31 | 404,08 |
| Διακύμανση | 72730,24 | 19576,31 | 17796,52 | 67111,19 | 33270,48 |
| Τυπική Απόκλιση | 269,69 | 139,92 | 133,40 | 259,06 | 182,40 |
| **ΣΥΓΚΡΙΣΕΙΣ ΕΠΙΠΛΕΟΝ ΔΕΙΚΤΩΝ** | | | | | |
| Απόσταση Ανά Κόμβο | 29,35 | 26,57 | 47,52 | 49,20 | 45,16 |
| Απόσταση Ανά  Ζήτηση | 1,36 | 1,25 | 2,20 | 2,28 | 2,05 |
| Χρόνος Αναμονής Ανά Κόμβο | 71,99 | 102,56 | 84,70 | 70,26 | 84,18 |
| Χρόνος Αναμονής  Ανά Ζήτηση | 3,33 | 4,83 | 3,92 | 3,25 | 3,81 |

*Πίνακας 12. Σύγκριση Αποτελεσμάτων Σεναρίου 2*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Προσέγγιση 1 | Προσέγγιση 2 | Προσέγγιση 3 | Προσέγγιση 4 | Προσέγγιση 5 |
| **ΠΟΣΟΣΤΟ ΕΞΥΠΗΡΕΤΗΣΗΣ ΚΟΜΒΩΝ ΜΕ ΤΗ ΧΡΗΣΗ ΤΩΝ ΑΡΧΙΚΩΝ ΟΧΗΜΑΤΩΝ** | | | | |
| 40% | 48% | 48% | 32% | 40% |
| **ΠΟΣΟΣΤΟ ΕΞΥΠΗΡΕΤΗΣΗΣ ΚΟΜΒΩΝ ΜΕ ΤΗΝ ΟΛΟΚΛΗΡΩΣΗ ΟΛΩΝ ΤΩΝ ΔΡΟΜΟΛΟΓΙΩΝ** | | | | |
| 100% | 96% | 100% | 100% | 96% |
| **ΚΑΤΑΝΟΜΗ ΦΟΡΤΙΟΥ ΣΤΟ ΣΥΝΟΛΟ ΤΩΝ ΟΧΗΜΑΤΩΝ** | | | | |
| +67,46% | Μικρότερη τυπική απόκλιση της τάξης των 31,14 μονάδων | +20,99% | +62,77% | +42,87% |
| **ΚΑΤΑΝΟΜΗ ΦΟΡΤΙΟΥ ΣΤΑ ΑΡΧΙΚΑ ΟΧΗΜΑΤΑ** | | | | |
| +300,00% | +400,00% | Μικρότερη τυπική απόκλιση της τάξης των 7,07 μονάδων | Μικρότερη τυπική απόκλιση της τάξης των 7,07 μονάδων | +300,00% |
| **ΚΑΤΑΝΟΜΗ ΦΟΡΤΙΟΥ ΣΤΑ ΕΠΙΠΡΟΣΘΕΤΑ ΟΧΗΜΑΤΑ** | | | | |
| +232,82% | +54,42% | +92,15% | Μικρότερη τυπική απόκλιση της τάξης των 20,82 μονάδων | +94,15% |
| **ΣΥΝΟΛΙΚΗ ΑΠΟΣΤΑΣΗ ΔΡΟΜΟΛΟΓΙΩΝ** | | | | |
| +15,06% | Βέλτιστη λύση 637,76 μονάδες | +86,28% | +92,87% | +69,96% |
| **ΚΑΤΑΝΟΜΗ ΑΠΟΣΤΑΣΗΣ ΔΡΟΜΟΛΟΓΙΩΝ ΣΤΟ ΣΥΝΟΛΟ ΤΩΝ ΟΧΗΜΑΤΩΝ** | | | | |
| +148,59% | Μικρότερη τυπική απόκλιση της τάξης των 31,57 μονάδων | +148,79% | +191,63% | +92,14% |
| **ΣΥΝΟΛΙΚΟΣ ΧΡΟΝΟΣ ΑΝΑΜΟΝΗΣ ΟΧΗΜΑΤΩΝ** | | | | |
| +2,46% | +40,12% | +20,55% | Βέλτιστη λύση με αναμονή 1756,56 μονάδες χρόνου | +15,02% |
| **ΚΑΤΑΝΟΜΗ ΧΡΟΝΟΥ ΑΝΑΜΟΝΗΣ ΣΤΟ ΣΥΝΟΛΟ ΤΩΝ ΟΧΗΜΑΤΩΝ** | | | | |
| +102,16% | +4,88% | Μικρότερη τυπική απόκλιση της τάξης των 133,40 μονάδων | +94,19% | +36,73% |
| **ΑΠΟΣΤΑΣΗ ΠΟΥ ΔΙΑΝΎΕΤΑΙ ΑΝΑ ΕΞΥΠΗΡΕΤΟΥΜΕΝΟ ΚΟΜΒΟ** | | | | |
| +10,46% | Βέλτιστη λύση 26,57 μονάδες | +78,83% | +85,16% | +69,96% |
| **ΧΡΟΝΟΣ ΑΝΑΜΟΝΗΣ ΑΝΑ ΕΞΥΠΗΡΕΤΟΥΜΕΝΟ ΚΟΜΒΟ** | | | | |
| +2,46% | +45,96% | +20,55% | Βέλτιστη λύση 70,26 μονάδες | +19,81% |

94

## Σενάριο 3

Η ομαδοποίηση των κόμβων σύμφωνα με το **Σενάριο 3**:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Σενάριο 3 | | | | | | | |
| Συστάδες | Κόμβοι | | | | | | |
| 1 | 0  19 | 21  22 | 25 | 18 | 23 | 20 | 24 |
| 2 | 0  4 | 3  2 | 8 | 1 | 5 | 7 | 6 |
| 3 | 0  12 | 13  10 | 14  11 | 17 | 16 | 15 | 9 |

Οι πίνακες αποτελεσμάτων των 5 αλγορίθμων:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Σενάριο 3 | Προσέγγιση | | | 1: Min Distance | | | Προσέγγιση 2: Inv. Dijkstra | | | | |
| Time + Dist. | Wait Time | Demand | | Nodes Served | Travel Dist. | Time  + Dist. | Wait Time | Demand | Nodes Served | Travel Dist. |
| Cluster 1 | 794  683  734,18 | 702,56  568,12  599,57 | 80 | | 4  6  7 | 91,44  114,88  134,61 | 691  700,07  734 | 552,34  607,69  634,38 | 140 | 7  4  4 | 138,66  92,382  99,616 |
| Cluster 2 | 140 | | 90 |
| Cluster 3 | 160 | | 70 |
| Sum | 2211,18 | 1870,25 | | 380 | 17 | 340,93 | 2125,1 | 1794,4 | 300 | 15 | 330,66 |
| Extra Cluster 1 | 481  345 | 351,56  251,57 | 60 | | 4  4 | 129,44  93,435 | 529  427,06 | 406,05  292,26 | 80 | 4  5 | 122,95  134,8 |
| Extra Cluster 2 | 100 | | 130 |
| General Sum | 3037,18 | 2473,37 | | 540 | 25 | 563,81 | 3081,1 | 2492,7 | 510 | 24 | 588,41 |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Σενάριο 3 | Προσέγγιση 3: Min Wait Time | | | | |  | Προσέγγιση | | 4: Min Served First | | |
| Time + Dist. | Wait Time | Demand | Nodes Served | Travel Dist. | Time  + Dist. | Wait Time | Demand | | Nodes Served | Travel Dist. |
| Cluster 1 | 794  683  734,18 | 674,73  566,03  600,25 | 140 | 5  5  6 | 119,27  116,97  133,93 | 794  700,07  713 | 684,36  593,86  629,92 | 50 | | 4  4  3 | 109,64  106,21  83,079 |
| Cluster 2 | 130  130 | 80 | |
| Cluster 3 | 40 | |
| Sum | 2211,18 | 1841,01 | 400 | 16 | 370,17 | 2207,1 | 1908,1 | | 170 | 11 | 298,93 |
| Extra Cluster 1 | 563,80  435,41 | 224,29  213,88 | 90 | 6  3 | 339,51  221,54 | 529  385 | 280,33  212,17 | 130 | | 6  6 | 248,67  172,83 |
| Extra Cluster 2 | 50 | 170 | |
| General Sum | 3210,39 | 2279,18 | 540 | 25 | 931,22 | 3121,1 | 2400,6 | | 470 | 23 | 720,43 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Σενάριο 3 | Π | ροσέγγιση | | 5: Max Served First | | |
| Time + Dist. | Wait Time | Demand | | Nodes Served | Travel Dist. |
| Cluster 1 | 794 | 707,39 | 100 | | 3 | 86,615 |
| Cluster 2 | 683 | 601,12 | 90 | | 3 | 81,881 |
| Cluster 3 | 734,18 | 636,29 | 100 | | 4 | 97,886 |
| Sum | 2211,18 | 1944,8 | | 290 | 10 | 266,38 |
| Extra | 691  481 | 422,76  301,34 | 110 | | 6  4 | 268,25  179,66 |
| Cluster 1 |
| Extra | 70 | |
| Cluster 2 |
| General Sum | 3383,18 | 2668,89 | | 470 | 20 | 714,29 |

*Πίνακας 13. Σύγκριση των 5 Προσεγγίσεων που χρησιμοποιήθηκαν στο Σενάριο 3.*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ΥΠΟΛΟΓΙΣΜΟΙ ΜΕΤΑ ΤΗΝ ΟΛΟΚΛΗΡΩΣΗ ΤΩΝ ΔΡΟΜΟΛΟΓΙΩΝ** | | | | | |
| Παρατηρήσεις | Προσέγγιση 1 | Προσέγγιση 2 | Προσέγγιση 3 | Προσέγγιση 4 | Προσέγγιση 5 |
| **ΑΡΙΘΜΟΣ ΟΧΗΜΑΤΩΝ** | | | | | |
| Αρχικά Οχήματα | 3 | 3 | 3 | 3 | 3 |
| Επιπλέον Οχήματα | 2 | 2 | 2 | 2 | 2 |
| **ΑΡΙΘΜΟΣ ΚΟΜΒΩΝ ΠΟΥ ΕΞΥΠΗΡΕΤΗΘΗΚΑΝ** | | | | | |
| Αρχικοί Κόμβοι | 17 | 15 | 16 | 11 | 10 |
| Κόμβοι Επιπλέον Διαδρομών | 8 | 9 | 9 | 12 | 10 |
| Συνολικοί Κόμβοι | 25 | 24 | 25 | 23 | 20 |
| **ΖΗΤΗΣΗ ΑΝΑ ΟΧΗΜΑ** | | | | | |
| Συνολική Ζήτηση | 540 | 510 | 540 | 470 | 470 |
| Μέση Τιμή | 108 | 102 | 108 | 94 | 94 |
| Διακύμανση | 1720,00 | 970,00 | 1420,00 | 3030,00 | 230,00 |
| Τυπική Απόκλιση | 41,47 | 31,14 | 37,68 | 55,05 | 15,17 |
| Τυπική Απόκλιση Αρχικών Δρομολ. | 41,63 | 36,06 | 5,77 | 20,82 | 5,77 |
| Τυπική Απόκλιση Επιπλέον Δρομολ. | 28,28 | 35,36 | 28,28 | 28,28 | 28,28 |
| **ΑΠΟΣΤΑΣΗ ΑΝΑ ΟΧΗΜΑ** | | | | | |
| Συνολική Απόσταση | 563,81 | 588,41 | 931,22 | 720,43 | 714,29 |
| Μέση Τιμή | 112,76 | 117,68 | 186,24 | 144,09 | 142,86 |
| Διακύμανση | 397,07 | 431,84 | 9189,12 | 4526,87 | 6495,06 |
| Τυπική Απόκλιση | 19,93 | 20,78 | 95,86 | 67,28 | 80,59 |
| **ΑΝΑΜΟΝΗ ΑΝΑ ΟΧΗΜΑ** | | | | | |
| Συνολικός Χρόνος Αναμονής | 2473,37 | 2492,72 | 2279,18 | 2400,64 | 2668,89 |
| Μέση Τιμή | 494,67 | 498,54 | 455,84 | 480,13 | 533,78 |
| Διακύμανση | 34799,44 | 21092,08 | 48267,25 | 47201,49 | 27884,32 |
| Τυπική Απόκλιση | 186,55 | 145,23 | 219.70 | 217,26 | 166,99 |
| **ΣΥΓΚΡΙΣΕΙΣ ΕΠΙΠΛΕΟΝ ΔΕΙΚΤΩΝ** | | | | | |
| Απόσταση Ανά Κόμβο | 22,55 | 24,52 | 37,25 | 31,32 | 35,71 |
| Απόσταση Ανά  Ζήτηση | 1,04 | 1,15 | 1,72 | 1,53 | 1,52 |
| Χρόνος Αναμονής Ανά Κόμβο | 98,93 | 103,86 | 91,17 | 104,38 | 133,44 |
| Χρόνος Αναμονής Ανά Ζήτηση | 4,58 | 4,89 | 4,22 | 5,11 | 5,68 |

*Πίνακας 14. Σύγκριση Αποτελεσμάτων Σεναρίου 3.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Προσέγγιση 1 | Προσέγγιση 2 | Προσέγγιση 3 | Προσέγγιση 4 | Προσέγγιση 5 |
| **ΠΟΣΟΣΤΟ ΕΞΥΠΗΡΕΤΗΣΗΣ ΚΟΜΒΩΝ ΜΕ ΤΗ ΧΡΗΣΗ ΤΩΝ ΑΡΧΙΚΩΝ ΟΧΗΜΑΤΩΝ** | | | | |
| 68% | 60% | 64% | 44% | 40% |
| **ΠΟΣΟΣΤΟ ΕΞΥΠΗΡΕΤΗΣΗΣ ΚΟΜΒΩΝ ΜΕ ΤΗΝ ΟΛΟΚΛΗΡΩΣΗ ΟΛΩΝ ΤΩΝ ΔΡΟΜΟΛΟΓΙΩΝ** | | | | |
| 100% | 96% | 100% | 92% | 80% |
| **ΚΑΤΑΝΟΜΗ ΦΟΡΤΙΟΥ ΣΤΟ ΣΥΝΟΛΟ ΤΩΝ ΟΧΗΜΑΤΩΝ** | | | | |
| +173,46% | +105,36% | +148,47% | +262,96% | Μικρότερη τυπική απόκλιση της τάξης των 15,17 μονάδων |
| **ΚΑΤΑΝΟΜΗ ΦΟΡΤΙΟΥ ΣΤΑ ΑΡΧΙΚΑ ΟΧΗΜΑΤΑ** | | | | |
| +621,11% | +524,50% | Μικρότερη τυπική απόκλιση της τάξης των 5,77 μονάδων | +260,56% | Μικρότερη τυπική απόκλιση της τάξης των 5,77 μονάδων |
| **ΚΑΤΑΝΟΜΗ ΦΟΡΤΙΟΥ ΣΤΑ ΕΠΙΠΡΟΣΘΕΤΑ ΟΧΗΜΑΤΑ** | | | | |
| Μικρότερη τυπική απόκλιση της τάξης των 28,28 μονάδων | +25,00% | Μικρότερη τυπική απόκλιση της τάξης των 28,28 μονάδων | Μικρότερη τυπική απόκλιση της τάξης των 28,28 μονάδων | Μικρότερη τυπική απόκλιση της τάξης των 28,28 μονάδων |
| **ΣΥΝΟΛΙΚΗ ΑΠΟΣΤΑΣΗ ΔΡΟΜΟΛΟΓΙΩΝ** | | | | |
| Βέλτιστη λύση 563,81 μονάδες | +4,36% | +65,17% | +27,78% | +26,69% |
| **ΚΑΤΑΝΟΜΗ ΑΠΟΣΤΑΣΗΣ ΔΡΟΜΟΛΟΓΙΩΝ ΣΤΟ ΣΥΝΟΛΟ ΤΩΝ ΟΧΗΜΑΤΩΝ** | | | | |
| Μικρότερη τυπική απόκλιση της τάξης των 19,93 μονάδων | +4,29% | +381,06% | +237,65% | +304,44% |
| **ΣΥΝΟΛΙΚΟΣ ΧΡΟΝΟΣ ΑΝΑΜΟΝΗΣ ΟΧΗΜΑΤΩΝ** | | | | |
| +8,52% | +9,37% | Βέλτιστη λύση με αναμονή 2279,18 μονάδες χρόνου | +5,33% | +17,10% |
| **ΚΑΤΑΝΟΜΗ ΧΡΟΝΟΥ ΑΝΑΜΟΝΗΣ ΣΤΟ ΣΥΝΟΛΟ ΤΩΝ ΟΧΗΜΑΤΩΝ** | | | | |
| +28,45% | Μικρότερη τυπική απόκλιση της τάξης των 145,23 μονάδων | +51,27% | +49,60% | +14,98% |
| **ΑΠΟΣΤΑΣΗ ΠΟΥ ΔΙΑΝΎΕΤΑΙ ΑΝΑ ΕΞΥΠΗΡΕΤΟΥΜΕΝΟ ΚΟΜΒΟ** | | | | |
| Βέλτιστη λύση 22,55 μονάδες | +8,71% | +65,17% | +38,89% | +58,36% |
| **ΧΡΟΝΟΣ ΑΝΑΜΟΝΗΣ ΑΝΑ ΕΞΥΠΗΡΕΤΟΥΜΕΝΟ ΚΟΜΒΟ** | | | | |
| +8,52% | +13,93% | Βέλτιστη λύση 91,17 μονάδες | +14,49% | +46,37% |

98

## Σενάριο 4

Η ομαδοποίηση των κόμβων σύμφωνα με το **Σενάριο 4**:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Σενάριο 4 | | | | | | | |
| Συστάδες | Κόμβοι | | | | | | |
| 1 | 0  19 | 21 | 20 | 24 | 25 | 18 | 23 |
| 2 | 0  2 | 3 | 8 | 5 | 1 | 7 | 6 |
| 3 | 0 | 4 | 22 | 11 |  |  |  |
| 4 | 0  12 | 13  10 | 14 | 16 | 17 | 15 | 9 |

Οι πίνακες αποτελεσμάτων των 5 αλγορίθμων:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Σενάριο 4 | Προσέγγιση 1: Min Distance | | | | | Προσέγγιση 2: Inv. Dijkstra | | | | |
| Time + Dist. | Wait Time | Demand | Nodes Served | Travel Dist. | Time  + Dist. | Wait Time | Demand | Nodes Served | Travel Dist. |
| Cluster 1 | 794 | 714,50 | 40 | 3 | 79,497 | 691 | 552,34 | 140 | 7 | 138,66 |
| Cluster 2 | 683 | 568,59 | 110 | 6 | 114,41 | 683 | 607,69 | 50 | 3 | 75,311 |
| Cluster 3 | 654 | 481,86 | 120 | 3 | 172,14 | 385 | 236,58 | 120 | 3 | 148,43 |
| Cluster 4 | 713 | 620,19 | 80 | 4 | 92,803 | 734,18 | 634,38 | 70 | 4 | 99,796 |
| Sum | 2844 | 2385,15 | 350 | 16 | 458,85 | 2493,2 | 2031 | 380 | 17 | 462,19 |
| Extra Cluster 1 | 355,99 | 181,78 | 140 | 7 | 174,22 | 529 | 406,05 | 80 | 4 | 122,95 |
| General Sum | 3200 | 2566,93 | 490 | 23 | 633,07 | 3022,2 | 2437 | 460 | 21 | 585,15 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Σενάριο 4 | Προσέγγιση 3: Min Wait Time | | | | | Προσέγγιση 4: Min Served First | | | | |
| Time + Dist. | Wait Time | Demand | Nodes Served | Travel Dist. | Time  +  Dist. | Wait Time | Demand | Nodes Served | Travel Dist. |
| Cluster 1 | 794 | 685,71 | 100 | 4 | 108,29 | 794 | 684,36 | 50 | 4 | 109,64 |
| Cluster 2 | 683 | 561,35 | 110 | 5 | 121,65 | 683 | 593,86 | 40 | 3 | 89,14 |
| Cluster 3 | 654 | 481,86 | 120 | 3 | 172,14 | 385 | 236,58 | 120 | 3 | 148,43 |
| Cluster 4 | 734,18 | 589,66 | 100 | 6 | 144,52 | 713 | 629,92 | 40 | 3 | 83,079 |
| Sum | 2865,18 | 2318,58 | 430 | 18 | 546,6 | 2575 | 2144,7 | 250 | 13 | 430,28 |
| Extra Cluster 1 | 585,57 | 390,85 | 70 | 4 | 194,72 | 529 | 280,33 | 130 | 6 | 248,67 |
| General Sum | 3450,75 | 2709,43 | 500 | 22 | 741,32 | 3104 | 2425 | 380 | 19 | 678,95 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Σενάριο 4 | Προσέγγιση 5: Max Served First | | | | |
| Time + Dist. | Wait Time | Demand | Nodes Served | Travel Dist. |
| Cluster 1 | 794 | 714,55 | 90 | 3 | 79,453 |
| Cluster 2 | 683 | 621,75 | 50 | 2 | 61,246 |
| Cluster 3 | 385 | 236,58 | 120 | 3 | 148,43 |
| Cluster 4 | 734,18 | 650,19 | 60 | 3 | 83,983 |
| Sum | 2596,18 | 2223,07 | 320 | 11 | 373,11 |
| Extra Cluster 1 | 691 | 395,59 | 120 | 7 | 295,41 |
| General Sum | 3287,18 | 2618,67 | 440 | 18 | 668,52 |

*Πίνακας 15. Σύγκριση των 5 Προσεγγίσεων που χρησιμοποιήθηκαν στο Σενάριο 4.*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ΥΠΟΛΟΓΙΣΜΟΙ ΜΕΤΑ ΤΗΝ ΟΛΟΚΛΗΡΩΣΗ ΤΩΝ ΔΡΟΜΟΛΟΓΙΩΝ** | | | | | |
| Παρατηρήσεις | Προσέγγιση 1 | Προσέγγιση 2 | Προσέγγιση 3 | Προσέγγιση 4 | Προσέγγιση 5 |
| Αρχικά Οχήματα | 4 | 4 | 4 | 4 | 4 |
| Επιπλέον Οχήματα | 1 | 1 | 1 | 1 | 1 |
| **ΑΡΙΘΜΟΣ ΚΟΜΒΩΝ ΠΟΥ ΕΞΥΠΗΡΕΤΗΘΗΚΑΝ** | | | | | |
| Αρχικοί Κόμβοι | 16 | 17 | 18 | 13 | 11 |
| Κόμβοι Επιπλέον Διαδρομών | 7 | 4 | 4 | 6 | 7 |
| Συνολικοί Κόμβοι | 23 | 21 | 22 | 19 | 18 |
| **ΖΗΤΗΣΗ ΑΝΑ ΟΧΗΜΑ** | | | | | |
| Συνολική Ζήτηση | 490 | 460 | 500 | 380 | 440 |
| Μέση Τιμή | 98 | 92 | 100 | 76 | 88 |
| Διακύμανση | 1520,00 | 1370,00 | 350,00 | 2030,00 | 10950,00 |
| Τυπική Απόκλιση | 38,99 | 37,01 | 18,71 | 45,06 | 32,71 |
| Τυπική Απόκλιση Αρχικών Δρομολ, | 35,94 | 42,03 | 9,57 | 38,62 | 31,62 |
| Τυπική Απόκλιση Επιπλέον Δρομολ. | Δεν διατίθεται για μέγεθος πληθυσμού μικρότερο των 2 σημείων | | | | |
| **ΑΠΟΣΤΑΣΗ ΑΝΑ ΟΧΗΜΑ** | | | | | |
| Συνολική Απόσταση | 633,07 | 585,15 | 741,324 | 678,954 | 668,515 |
| Μέση Τιμή | 126,61 | 117,03 | 148,26 | 135,79 | 133,70 |
| Διακύμανση | 1962,76 | 881,54 | 1262,19 | 4635,15 | 9257,60 |
| Τυπική Απόκλιση | 44,30 | 29,69 | 35,53 | 68,08 | 96,22 |
| **ΑΝΑΜΟΝΗ ΑΝΑ ΟΧΗΜΑ** | | | | | |
| Συνολικός Χρόνος Αναμονής | 2566,93 | 2437,03 | 2709,43 | 2425,05 | 2618,67 |
| Μέση Τιμή | 513,39 | 487,41 | 541,89 | 485,01 | 523,73 |
| Διακύμανση | 41465,04 | 27455,58 | 12440,48 | 44050,85 | 40222,77 |
| Τυπική Απόκλιση | 203,63 | 165,70 | 111,54 | 209,88 | 200,56 |
| **ΣΥΓΚΡΙΣΕΙΣ ΕΠΙΠΛΕΟΝ ΔΕΙΚΤΩΝ** | | | | | |
| Απόσταση Ανά Κόμβο | 39,57 | 34,42 | 41,18 | 52,23 | 60,77 |
| Απόσταση Ανά Ζήτηση | 27,52 | 27,86 | 33,70 | 35,73 | 37,14 |
| Χρόνος Αναμονής Ανά Κόμβο | 160,43 | 143,35 | 150,52 | 186,54 | 238,06 |
| Χρόνος Αναμονής Ανά Ζήτηση | 111,61 | 116,05 | 123,16 | 127,63 | 145,48 |

*Πίνακας 16. Σύγκριση Αποτελεσμάτων Σεναρίου 4.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Προσέγγιση 1 | Προσέγγιση 2 | Προσέγγιση 3 | Προσέγγιση 4 | Προσέγγιση 5 |
| **ΠΟΣΟΣΤΟ ΕΞΥΠΗΡΕΤΗΣΗΣ ΚΟΜΒΩΝ ΜΕ ΤΗ ΧΡΗΣΗ ΤΩΝ ΑΡΧΙΚΩΝ ΟΧΗΜΑΤΩΝ** | | | | |
| 64% | 68% | 72% | 52% | 44% |
| **ΠΟΣΟΣΤΟ ΕΞΥΠΗΡΕΤΗΣΗΣ ΚΟΜΒΩΝ ΜΕ ΤΗΝ ΟΛΟΚΛΗΡΩΣΗ ΟΛΩΝ ΤΩΝ ΔΡΟΜΟΛΟΓΙΩΝ** | | | | |
| 92% | 84% | 88% | 76% | 72% |
| **ΚΑΤΑΝΟΜΗ ΦΟΡΤΙΟΥ ΣΤΟ ΣΥΝΟΛΟ ΤΩΝ ΟΧΗΜΑΤΩΝ** | | | | |
| +108,40% | +97,85% | Μικρότερη Τυπική απόκλιση της τάξης των 18,71 μονάδων | +140,83% | +74,85% |
| **ΚΑΤΑΝΟΜΗ ΦΟΡΤΙΟΥ ΣΤΑ ΑΡΧΙΚΑ ΟΧΗΜΑΤΑ** | | | | |
| +275,38% | +339,01% | Μικρότερη Τυπική απόκλιση της τάξης των 9,57 μονάδων | +303,39% | +230,29% |
| **ΣΥΝΟΛΙΚΗ ΑΠΟΣΤΑΣΗ ΔΡΟΜΟΛΟΓΙΩΝ** | | | | |
| +8,19% | Βέλτιστη Λύση 585,15 μονάδες | +26,69% | +16,03% | +14,25% |
| **ΚΑΤΑΝΟΜΗ ΑΠΟΣΤΑΣΗΣ ΔΡΟΜΟΛΟΓΙΩΝ ΣΤΟ ΣΥΝΟΛΟ ΤΩΝ ΟΧΗΜΑΤΩΝ** | | | | |
| +49,21% | Μικρότερη Τυπική απόκλιση της τάξης των 29,69 μονάδων | +19,66% | +129,30% | +224,06% |
| **ΣΥΝΟΛΙΚΟΣ ΧΡΟΝΟΣ ΑΝΑΜΟΝΗΣ ΟΧΗΜΑΤΩΝ** | | | | |
| +5,85% | +0,49% | +11,73% | Βέλτιστη Λύση με αναμονή 2425,05 μονάδες χρόνου | +7,98% |
| **ΚΑΤΑΝΟΜΗ ΧΡΟΝΟΥ ΑΝΑΜΟΝΗΣ ΣΤΟ ΣΥΝΟΛΟ ΤΩΝ ΟΧΗΜΑΤΩΝ** | | | | |
| +82,57% | +48,56% | Μικρότερη Τυπική απόκλιση της τάξης των 111,54 μονάδων | +88,17% | +79,81% |
| **ΑΠΟΣΤΑΣΗ ΠΟΥ ΔΙΑΝΎΕΤΑΙ ΑΝΑ ΕΞΥΠΗΡΕΤΟΥΜΕΝΟ ΚΟΜΒΟ** | | | | |
| +14,95% | Βέλτιστη Λύση 34,42 μονάδες | +19,65% | +51,73% | +76,56% |
| **ΧΡΟΝΟΣ ΑΝΑΜΟΝΗΣ ΑΝΑ ΕΞΥΠΗΡΕΤΟΥΜΕΝΟ ΚΟΜΒΟ** | | | | |
| +11,91% | Βέλτιστη Λύση 143,35 μονάδες | +5,00% | +30,13% | +66,06% |

102

## 5.4.2 ΑΠΟΤΕΛΕΣΜΑΤΑ ΜΕΣΑΙΟΥ ΣΥΝΟΛΟΥ ΔΕΔΟΜΕΝΩΝ Σενάριο 5

Η ομαδοποίηση των κόμβων σύμφωνα με το **Σενάριο 5**:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Σενάριο 5 | | | | | | | |
| Συστάδες | Κόμβοι | | | | | | |
|  | 0 | 28 | 33 | 29 | 30 | 32 | 27 |
| 1 | 26 | 31 | 10 | 2 | 11 | 4 | 34 |
|  | 50 |  |  |  |  |  |  |
|  | 0 | 37 | 39 | 8 | 46 | 44 | 45 |
| 2 | 42 | 3 | 40 | 35 | 36 | 7 | 41 |
|  | 6 | 43 | 5 | 38 | 1 |  |  |
|  | 0 | 48 | 21 | 47 | 49 | 20 | 14 |
| 3 | 24 | 13 | 18 | 17 | 25 | 16 | 15 |
|  | 12 | 19 | 22 | 23 | 9 |  |  |

Οι πίνακες αποτελεσμάτων των 5 αλγορίθμων:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Σενάριο 5 | Προσέγγιση | | | 1: Min Distance | | | Προσέγγιση 2: Inv. Dijkstra | | | | |
| Time + Dist. | Wait Time | Demand | | Nodes Served | Travel Dist. | Time  + Dist. | Wait Time | Demand | Nodes Served | Travel Dist. |
| Cluster 1 | 747,6  729  820,20 | 597,80  571,59  632,61 | 80 | | 3  7  8 | 149,79  157,4  187,6 | 724,6  683  821,19 | 524,48  543,36  649,72 | 100 | 5  5  6 | 200,12  139,64  171,46 |
| Cluster 2 | 130 | | 110 |
| Cluster 3 | 110 | | 100 |
| Sum | 2296,81 | 1802,02 | | 320 | 18 | 494,79 | 2228,8 | 1717,6 | 310 | 16 | 511,23 |
| Extra Cluster 1 | 641,33  605,83  430,58  335,41 | 419,93  414,69  267,62  22,26 | 210 | | 8  6  4  10 | 221,4  191,14  162,96  313,15 | 663  558,63  500,91  381,48 | 419,26  333,6  259,83  155,08 | 160 | 8  5  6  8 | 243,74  225,03  241,08  226,41 |
| Extra Cluster 2 | 100 | | 80 |
| Extra Cluster 3 | 70 | | 130 |
| Extra Cluster 4 | 190 | | 190 |
| General Sum | 4309,98 | 2926,55 | | 890 | 46 | 1383,4 | 4332,8 | 2885,3 | 870 | 43 | 1447,5 |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Σενάριο 5 | Προσέγγιση 3: Min Wait Time | | | | |  | Προσέγγιση | | 4: Min Served First | | |
| Time + Dist. | Wait Time | Demand | Nodes Served | Travel Dist. | Time  + Dist. | Wait Time | Demand | | Nodes Served | Travel Dist. |
| Cluster 1 | 724,6  719  873,15 | 306,94  440,81  632,16 | 150  150 | 7  8  4 | 417,67  278,19  241 | 749,14  729  823,92 | 499,12  461,6  579,68 | 120 | | 7  8  8 | 250,02  267,4  244,24 |
| Cluster 2 | 110 | |
| Cluster 3 | 60 | 100 | |
| Sum | 2316,75 | 1379,9 | 360 | 19 | 936,85 | 2302,1 | 1540,4 | | 330 | 23 | 761,66 |
| Extra Cluster 1 | 749,26  602,89  539  481 | 34,60  120,67  456,99  169,46 | 190 | 12  9  2  6 | 714,66  482,21  82,009  311,55 | 614,17  517  464,42  322,44 | 108,6  239,8  155,44  88,39 | 180 | | 10  5  7  4 | 505,57  277,2  308,98  234,05 |
| Extra Cluster 2 | 200 | 100 | |
| Extra Cluster 3 | 50 | 180 | |
| Extra Cluster 4 | 130 | 140 | |
| General Sum | 4688,9 | 2161,62 | 930 | 48 | 2527,3 | 4220,1 | 2132,6 | | 930 | 49 | 2087,5 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Σενάριο 5 | Π | ροσέγγιση | | 5: Max Served First | | |
| Time + Dist. | Wait Time | Demand | | Nodes Served | Travel Dist. |
| Cluster 1 | 747,6 | 558,71 | 90 | | 3 | 188,89 |
| Cluster 2 | 729 | 566,53 | 160 | | 6 | 162,47 |
| Cluster 3 | 811 | 575,18 | 150 | | 6 | 235,82 |
| Sum | 2287,6 | 1700,43 | | 400 | 15 | 587,18 |
| Extra | 757,481  663  631,01  460,16 | 364,89  411,03  93,30  191,94 | 110 | | 5  4  9  6 | 392,59  251,97  537,71  268,22 |
| Cluster 1 |
| Extra | 80 | |
| Cluster 2 |
| Extra | 170 | |
| Cluster 3 |
| Extra | 100 | |
| Cluster 4 |
| General Sum | 4799,26 | 2761,59 | | 860 | 39 | 2037,7 |

*Πίνακας 17. Σύγκριση των 5 Προσεγγίσεων που χρησιμοποιήθηκαν στο Σενάριο 5.*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ΥΠΟΛΟΓΙΣΜΟΙ ΜΕΤΑ ΤΗΝ ΟΛΟΚΛΗΡΩΣΗ ΤΩΝ ΔΡΟΜΟΛΟΓΙΩΝ** | | | | | |
| Παρατηρήσεις | Προσέγγιση 1 | Προσέγγιση 2 | Προσέγγιση 3 | Προσέγγιση 4 | Προσέγγιση 5 |
| **ΑΡΙΘΜΟΣ ΟΧΗΜΑΤΩΝ** | | | | | |
| Αρχικά Οχήματα | 3 | 3 | 3 | 3 | 3 |
| Επιπλέον Οχήματα | 4 | 4 | 4 | 4 | 4 |
| **ΑΡΙΘΜΟΣ ΚΟΜΒΩΝ ΠΟΥ ΕΞΥΠΗΡΕΤΗΘΗΚΑΝ** | | | | | |
| Αρχικοί Κόμβοι | 18 | 16 | 19 | 23 | 15 |
| Κόμβοι Επιπλέον Διαδρομών | 28 | 27 | 29 | 26 | 24 |
| Συνολικοί Κόμβοι | 46 | 43 | 48 | 49 | 39 |
| **ΖΗΤΗΣΗ ΑΝΑ ΟΧΗΜΑ** | | | | | |
| Συνολική Ζήτηση | 890 | 870 | 930 | 930 | 860 |
| Μέση Τιμή | 127,14 | 124,29 | 132,86 | 132,86 | 122,86 |
| Διακύμανση | 2890,48 | 1495,24 | 3423,81 | 1223,81 | 1323,81 |
| Τυπική Απόκλιση | 53,76 | 38,67 | 58,51 | 34,98 | 36,38 |
| Τυπική Απόκλιση Αρχικών Δρομολ. | 25,17 | 5,77 | 51,96 | 10,00 | 37,86 |
| Τυπική Απόκλιση Επιπλέον Δρομολ. | 68,01 | 46,90 | 68,98 | 38,30 | 38,73 |
| **ΑΠΟΣΤΑΣΗ ΑΝΑ ΟΧΗΜΑ** | | | | | |
| Συνολική Απόσταση | 1383,43 | 1447,49 | 2527,28 | 2087,46 | 2037,67 |
| Μέση Τιμή | 197,63 | 206,78 | 361,04 | 298,21 | 291,10 |
| Διακύμανση | 3193,58 | 1510,03 | 40752,85 | 8975,70 | 17203,58 |
| Τυπική Απόκλιση | 56,51 | 38,86 | 226,33 | 137,15 | 159,18 |
| **ΑΝΑΜΟΝΗ ΑΝΑ ΟΧΗΜΑ** | | | | | |
| Συνολικός Χρόνος Αναμονής | 2926,55 | 2885,32 | 2161,62 | 2132,63 | 2761,59 |
| Μέση Τιμή | 418,08 | 412,19 | 308,80 | 304,66 | 394,51 |
| Διακύμανση | 46869,20 | 30297,05 | 45657,15 | 41627,32 | 37017,74 |
| Τυπική Απόκλιση | 216,49 | 174,06 | 213,68 | 204,03 | 192,40 |
| **ΣΥΓΚΡΙΣΕΙΣ ΕΠΙΠΛΕΟΝ ΔΕΙΚΤΩΝ** | | | | | |
| Απόσταση Ανά Κόμβο | 30,07 | 33,66 | 52,65 | 42,60 | 52,25 |
| Απόσταση Ανά  Ζήτηση | 1,55 | 1,66 | 2,72 | 2,24 | 2,37 |
| Χρόνος Αναμονής Ανά Κόμβο | 63,62 | 67,10 | 45,03 | 43,52 | 70,81 |
| Χρόνος Αναμονής  Ανά Ζήτηση | 3,29 | 3,32 | 2,32 | 2,29 | 3,21 |

*Πίνακας 18. Σύγκριση Αποτελεσμάτων Σεναρίου 5.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Προσέγγιση 1 | Προσέγγιση 2 | Προσέγγιση 3 | Προσέγγιση 4 | Προσέγγιση 5 |
| **ΠΟΣΟΣΤΟ ΕΞΥΠΗΡΕΤΗΣΗΣ ΚΟΜΒΩΝ ΜΕ ΤΗ ΧΡΗΣΗ ΤΩΝ ΑΡΧΙΚΩΝ ΟΧΗΜΑΤΩΝ** | | | | |
| 36% | 32% | 38% | 46% | 30% |
| **ΠΟΣΟΣΤΟ ΕΞΥΠΗΡΕΤΗΣΗΣ ΚΟΜΒΩΝ ΜΕ ΤΗΝ ΟΛΟΚΛΗΡΩΣΗ ΟΛΩΝ ΤΩΝ ΔΡΟΜΟΛΟΓΙΩΝ** | | | | |
| 92% | 86% | 96% | 98% | 78% |
| **ΚΑΤΑΝΟΜΗ ΦΟΡΤΙΟΥ ΣΤΟ ΣΥΝΟΛΟ ΤΩΝ ΟΧΗΜΑΤΩΝ** | | | | |
| +53,68% | +10,53% | +67,26% | Μικρότερη τυπική απόκλιση της τάξης των 34,98 μονάδων | +4,01% |
| **ΚΑΤΑΝΟΜΗ ΦΟΡΤΙΟΥ ΣΤΑ ΑΡΧΙΚΑ ΟΧΗΜΑΤΑ** | | | | |
| +335,89% | Μικρότερη τυπική απόκλιση της τάξης των 5,77 μονάδων | +800,00% | +73,21% | +555,74% |
| **ΚΑΤΑΝΟΜΗ ΦΟΡΤΙΟΥ ΣΤΑ ΕΠΙΠΡΟΣΘΕΤΑ ΟΧΗΜΑΤΑ** | | | | |
| +77,58% | +22,47% | +80,12% | Μικρότερη τυπική απόκλιση της τάξης των 38,30 μονάδων | +1,13% |
| **ΣΥΝΟΛΙΚΗ ΑΠΟΣΤΑΣΗ ΔΡΟΜΟΛΟΓΙΩΝ** | | | | |
| Βέλτιστη λύση 1383,43 μονάδες | +4,63% | +82,68% | +50,89% | +47,29% |
| **ΚΑΤΑΝΟΜΗ ΑΠΟΣΤΑΣΗΣ ΔΡΟΜΟΛΟΓΙΩΝ ΣΤΟ ΣΥΝΟΛΟ ΤΩΝ ΟΧΗΜΑΤΩΝ** | | | | |
| +45,43% | Μικρότερη τυπική απόκλιση της τάξης των 38,86 μονάδων | +482,43% | +252,94% | +309,63% |
| **ΣΥΝΟΛΙΚΟΣ ΧΡΟΝΟΣ ΑΝΑΜΟΝΗΣ ΟΧΗΜΑΤΩΝ** | | | | |
| +37,23% | +35,29% | +1,36% | Βέλτιστη λύση με αναμονή 2132,63 μονάδες χρόνου | +29,49% |
| **ΚΑΤΑΝΟΜΗ ΧΡΟΝΟΥ ΑΝΑΜΟΝΗΣ ΣΤΟ ΣΥΝΟΛΟ ΤΩΝ ΟΧΗΜΑΤΩΝ** | | | | |
| +24,38% | Μικρότερη τυπική απόκλιση της τάξης των 174,06 μονάδων | +22,76% | +17,22% | +10,54% |
| **ΑΠΟΣΤΑΣΗ ΠΟΥ ΔΙΑΝΎΕΤΑΙ ΑΝΑ ΕΞΥΠΗΡΕΤΟΥΜΕΝΟ ΚΟΜΒΟ** | | | | |
| Βέλτιστη λύση 30,07 μονάδες | +11,93% | +75,07% | +41,65% | +73,73% |
| **ΧΡΟΝΟΣ ΑΝΑΜΟΝΗΣ ΑΝΑ ΕΞΥΠΗΡΕΤΟΥΜΕΝΟ ΚΟΜΒΟ** | | | | |
| +46,18% | +54,17% | +3,47% | Βέλτιστη λύση 43,52 μονάδες | +62,70% |

106

## Σενάριο 6

Η ομαδοποίηση των κόμβων σύμφωνα με το **Σενάριο 6**:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Σενάριο 6 | | | | | | | |
| Συστάδες | Κόμβοι | | | | | | |
| 1 | 0  20 | 48  49 | 21  11 | 25  19 | 18  22 | 24 | 23 |
|  | 0 | 37 | 8 | 43 | 46 | 39 | 45 |
| 2 | 3 | 44 | 42 | 40 | 35 | 7 | 38 |
|  | 6 | 5 | 41 |  |  |  |  |
| 3 | 0  26 | 33  31 | 28  36 | 30  34 | 32  50 | 29 | 27 |
| 4 | 0  2 | 13  15 | 4  14 | 47  17 | 1  10 | 9  12 | 16 |

Οι πίνακες αποτελεσμάτων των 5 αλγορίθμων:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Σενάριο 6 | Προσέγγιση | | | 1: Min Distance | | | Προσέγγιση 2: Inv. Dijkstra | | | | |
| Time + Dist. | Wait Time | Demand | | Nodes Served | Travel Dist. | Time  + Dist. | Wait Time | Demand | Nodes Served | Travel Dist. |
| Cluster 1 | 811 | 680,98 | 80 | | 4 | 130,02 | 762,77 | 649,72 | 80 | 5 | 113,05 |
| Cluster 2 | 689 | 571,59 | 110 | | 6 | 117,4 | 587,44 | 499,19 | 50 | 3 | 88,25 |
| Cluster 3 | 649 | 547,27 | 100 | | 4 | 101,73 | 631 | 524,48 | 60 | 4 | 106,52 |
| Cluster 4 | 761,35 | 609,74 | 120 | | 5 | 151,61 | 798,03 | 622,77 | 100 | 6 | 175,26 |
| Sum | 2910,35 | 2409,59 | | 410 | 19 | 500,76 | 2779,2 | 2296,2 | 290 | 18 | 483,08 |
| Extra | 560  515,40  394,28 | 432,32  327,53  267,63 | 110 | | 5  9  4 | 127,68  187,87  126,65 | 667,89  558,63  500,91 | 389,99  33,6  259 | 190 | 8  5  6 | 277,9  525,03  241,91 |
| Cluster 1 |
| Extra | 140 | | 80 |
| Cluster 2 |
| Extra | 60 | | 130 |
| Cluster 3 |
| General Sum | 4380,03 | 3437,08 | | 720 | 37 | 942,96 | 4506,7 | 2978,8 | 690 | 37 | 1527,9 |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Σενάριο 6 | Προσέγγιση 3: Min Wait Time | | | | |  | Προσέγγιση | | 4: Min Served First | | |
| Time + Dist. | Wait Time | Demand | Nodes Served | Travel Dist. | Time  + Dist. | Wait Time | Demand | | Nodes Served | Travel Dist. |
| Cluster 1 | 794  667  649  784,02 | 579,80  457,23  411,35  597,65 | 140 | 6  7  9  4 | 214,2  209,77  237,65  186,38 | 794  667  649  740,17 | 686,32  461,95  525,92  546,76 | 50 | | 4  6  5  5 | 107,68  205,05  123,08  193,41 |
| Cluster 2 | 100 | 80 | |
| Cluster 3 | 160 | 70  70 | |
| Cluster 4 | 60 |
| Sum | 2894,02 | 2046,03 | 460 | 26 | 847,99 | 2850,2 | 2220,9 | | 270 | 20 | 629,23 |
| Extra Cluster 1 | 765,57  578,29  535,69 | 0  0  391,39 | 280 | 13  3  3 | 765,57  578,29  144,3 | 686,18  564,6  443,08 | 391,94  85,895  216,15 | 140 | | 6  8  4 | 294,23  478,7  226,93 |
| Extra Cluster 2 | 50 | 130 | |
| Extra Cluster 3 | 60 | 80 | |
| General Sum | 4773,57 | 2437,42 | 850 | 45 | 2336,2 | 4544 | 2914,9 | | 620 | 38 | 1629,1 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Σενάριο 6 | Π | ροσέγγιση | | 5: Max Served First | | |
| Time + Dist. | Wait Time | Demand | | Nodes Served | Travel Dist. |
| Cluster 1 | 811 | 658,57 | 150 | | 5 | 152,44 |
| Cluster 2 | 680,61 | 560,70 | 110 | | 5 | 119,91 |
| Cluster 3 | 631 | 435,35 | 140 | | 5 | 195,65 |
| Cluster 4 | 761,35 | 517,13 | 150 | | 6 | 244,23 |
| Sum | 2883,96 | 2171,74 | | 550 | 21 | 712,22 |
| Extra | 691  560  515,40 | 412,04  227,59  254,68 | 100 | | 5  7  7 | 278,96  332,41  260,73 |
| Cluster 1 |
| Extra | 120 | |
| Cluster 2 |
| Extra | 100 | |
| Cluster 3 |
| General Sum | 4650,36 | 3066,04 | | 870 | 40 | 1584,3 |

*Πίνακας 19. Σύγκριση των 5 Προσεγγίσεων που χρησιμοποιήθηκαν στο Σενάριο 6.*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ΥΠΟΛΟΓΙΣΜΟΙ ΜΕΤΑ ΤΗΝ ΟΛΟΚΛΗΡΩΣΗ ΤΩΝ ΔΡΟΜΟΛΟΓΙΩΝ** | | | | | |
| Παρατηρήσεις | Προσέγγιση 1 | Προσέγγιση 2 | Προσέγγιση 3 | Προσέγγιση 4 | Προσέγγιση 5 |
| **ΑΡΙΘΜΟΣ ΟΧΗΜΑΤΩΝ** | | | | | |
| Αρχικά Οχήματα | 4 | 4 | 4 | 4 | 4 |
| Επιπλέον Οχήματα | 3 | 3 | 3 | 3 | 3 |
| **ΑΡΙΘΜΟΣ ΚΟΜΒΩΝ ΠΟΥ ΕΞΥΠΗΡΕΤΗΘΗΚΑΝ** | | | | | |
| Αρχικοί Κόμβοι | 19 | 18 | 26 | 20 | 21 |
| Κόμβοι Επιπλέον Διαδρομών | 18 | 19 | 19 | 18 | 19 |
| Συνολικοί Κόμβοι | 37 | 37 | 45 | 38 | 40 |
| **ΖΗΤΗΣΗ ΑΝΑ ΟΧΗΜΑ** | | | | | |
| Συνολική Ζήτηση | 720 | 690 | 850 | 620 | 870 |
| Μέση Τιμή | 102,86 | 98,57 | 121,43 | 88,57 | 124,29 |
| Διακύμανση | 690,48 | 2314,29 | 6680,95 | 1114,29 | 495,24 |
| Τυπική Απόκλιση | 26,28 | 48,11 | 81,74 | 33,38 | 22,25 |
| Τυπική Απόκλιση Αρχικών Δρομολ. | 17,08 | 22,17 | 44,35 | 12,58 | 18,93 |
| Τυπική Απόκλιση Επιπλέον Δρομολ. | 40,41 | 55,08 | 130,00 | 32,15 | 11,55 |
| **ΑΠΟΣΤΑΣΗ ΑΝΑ ΟΧΗΜΑ** | | | | | |
| Συνολική Απόσταση | 942,96 | 1527,93 | 2336,15 | 1629,09 | 1584,32 |
| Μέση Τιμή | 134,71 | 218,28 | 333,74 | 232,73 | 226,33 |
| Διακύμανση | 772,60 | 23421,91 | 57130,22 | 15715,04 | 5542,39 |
| Τυπική Απόκλιση | 27,80 | 153,04 | 239,02 | 125,36 | 74,45 |
| **ΑΝΑΜΟΝΗ ΑΝΑ ΟΧΗΜΑ** | | | | | |
| Συνολικός Χρόνος Αναμονής | 3437,08 | 2978,75 | 2437,42 | 2914,93 | 3066,04 |
| Μέση Τιμή | 491,01 | 425,54 | 348,20 | 416,42 | 438,01 |
| Διακύμανση | 23319,05 | 47831,07 | 62681,78 | 42308,48 | 24754,34 |
| Τυπική Απόκλιση | 152,71 | 218,70 | 250,36 | 205,69 | 157,34 |
| **ΣΥΓΚΡΙΣΕΙΣ ΕΠΙΠΛΕΟΝ ΔΕΙΚΤΩΝ** | | | | | |
| Απόσταση Ανά Κόμβο | 25,49 | 41,30 | 51,91 | 42,87 | 39,61 |
| Απόσταση Ανά  Ζήτηση | 1,31 | 2,21 | 2,75 | 2,63 | 1,82 |
| Χρόνος Αναμονής Ανά Κόμβο | 92,89 | 80,51 | 54,16 | 76,71 | 76,65 |
| Χρόνος Αναμονής  Ανά Ζήτηση | 4,77 | 4,32 | 2,87 | 4,70 | 3,52 |

*Πίνακας 20. Σύγκριση Αποτελεσμάτων Σεναρίου 6.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Προσέγγιση 1 | Προσέγγιση 2 | Προσέγγιση 3 | Προσέγγιση 4 | Προσέγγιση 5 |
| **ΠΟΣΟΣΤΟ ΕΞΥΠΗΡΕΤΗΣΗΣ ΚΟΜΒΩΝ ΜΕ ΤΗ ΧΡΗΣΗ ΤΩΝ ΑΡΧΙΚΩΝ ΟΧΗΜΑΤΩΝ** | | | | |
| 38% | 36% | 52% | 40% | 42% |
| **ΠΟΣΟΣΤΟ ΕΞΥΠΗΡΕΤΗΣΗΣ ΚΟΜΒΩΝ ΜΕ ΤΗΝ ΟΛΟΚΛΗΡΩΣΗ ΟΛΩΝ ΤΩΝ ΔΡΟΜΟΛΟΓΙΩΝ** | | | | |
| 74% | 74% | 90% | 76% | 80% |
| **ΚΑΤΑΝΟΜΗ ΦΟΡΤΙΟΥ ΣΤΟ ΣΥΝΟΛΟ ΤΩΝ ΟΧΗΜΑΤΩΝ** | | | | |
| +18,08% | +116,17% | +267,29% | +50,00% | Μικρότερη τυπική απόκλιση της τάξης των 22,25 μονάδων |
| **ΚΑΤΑΝΟΜΗ ΦΟΡΤΙΟΥ ΣΤΑ ΑΡΧΙΚΑ ΟΧΗΜΑΤΑ** | | | | |
| +35,72% | +76,22% | +252,44% | Μικρότερη τυπική απόκλιση της τάξης των 12,58 μονάδων | +50,44% |
| **ΚΑΤΑΝΟΜΗ ΦΟΡΤΙΟΥ ΣΤΑ ΕΠΙΠΡΟΣΘΕΤΑ ΟΧΗΜΑΤΑ** | | | | |
| +250,00% | +376,97% | +1025,83% | +178,39% | Μικρότερη τυπική απόκλιση της τάξης των 11,55 μονάδων |
| **ΣΥΝΟΛΙΚΗ ΑΠΟΣΤΑΣΗ ΔΡΟΜΟΛΟΓΙΩΝ** | | | | |
| Βέλτιστη λύση 942,96 μονάδες | +62,04% | +147,75% | +72,76% | +68,02% |
| **ΚΑΤΑΝΟΜΗ ΑΠΟΣΤΑΣΗΣ ΔΡΟΜΟΛΟΓΙΩΝ ΣΤΟ ΣΥΝΟΛΟ ΤΩΝ ΟΧΗΜΑΤΩΝ** | | | | |
| Μικρότερη τυπική απόκλιση της τάξης των 27,80 μονάδων | +450,60% | +759,91% | +351,00% | +167,84% |
| **ΣΥΝΟΛΙΚΟΣ ΧΡΟΝΟΣ ΑΝΑΜΟΝΗΣ ΟΧΗΜΑΤΩΝ** | | | | |
| +41,01% | +22,21% | Βέλτιστη λύση με αναμονή 2437,42 μονάδες χρόνου | +19,59% | +25,79% |
| **ΚΑΤΑΝΟΜΗ ΧΡΟΝΟΥ ΑΝΑΜΟΝΗΣ ΣΤΟ ΣΥΝΟΛΟ ΤΩΝ ΟΧΗΜΑΤΩΝ** | | | | |
| Μικρότερη τυπική απόκλιση της τάξης των 152,71 μονάδων | +43,22% | +63,95% | +34,70% | +3,03% |
| **ΑΠΟΣΤΑΣΗ ΠΟΥ ΔΙΑΝΎΕΤΑΙ ΑΝΑ ΕΞΥΠΗΡΕΤΟΥΜΕΝΟ ΚΟΜΒΟ** | | | | |
| Βέλτιστη λύση 25,49 μονάδες | +62,04% | +103,70% | +68,22% | +55,41% |
| **ΧΡΟΝΟΣ ΑΝΑΜΟΝΗΣ ΑΝΑ ΕΞΥΠΗΡΕΤΟΥΜΕΝΟ ΚΟΜΒΟ** | | | | |
| +71,50% | +48,63% | Βέλτιστη λύση 54,16 μονάδες | +41,62% | +41,51% |

110

## Σενάριο 7

Η ομαδοποίηση των κόμβων σύμφωνα με το **Σενάριο 7**:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Σενάριο 7 | | | | | | | |
| Συστάδες | Κόμβοι | | | | | | |
| 1 | 0 | 28 | 33 | 29 | 30 | 27 | 32 |
| 26 | 50 | 31 | 34 |  |  |  |
| 2 | 0 | 3 | 8 | 45 | 46 | 1 | 5 |
| 7 | 6 | 2 | 4 |  |  |  |
| 3 | 0 | 48 | 21 | 25 | 18 | 24 | 23 |
| 20 | 49 | 19 | 22 |  |  |  |
| 4 | 0 | 13 | 47 | 14 | 17 | 16 | 9 |
| 15 | 12 | 10 | 11 |  |  |  |
| 5 | 0 | 37 | 35 | 44 | 42 | 43 | 36 |
| 40 | 41 | 39 | 38 |  |  |  |

Οι πίνακες αποτελεσμάτων των 5 αλγορίθμων:

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Σενάριο 7 | Προσέγγιση | | | 1: Min Distance | | | Προσέγγιση 2: Inv. Dijkstra | | | | |
| Time + Dist. | Wait Time | Demand | | Nodes Served | Travel Dist. | Time  + Dist. | Wait Time | Demand | Nodes Served | Travel Dist. |
| Cluster 1 | 649 | 547,27 | 100 | | 4 | 101,73 | 631 | 524,48 | 60 | 4 | 106,52 |
| Cluster 2 | 683 | 558,12 | 150 | | 7 | 124,88 | 700,07 | 607,69 | 90 | 4 | 92,382 |
| Cluster 3 | 811 | 716,56 | 80 | | 4 | 94,44 | 762,77 | 649,72 | 80 | 5 | 113,05 |
| Cluster 4 | 734,18 | 599,56 | 160 | | 7 | 134,61 | 734,18 | 622,77 |  | 5 | 111,41 |
| Cluster 5 | 689 | 571,59 | 110 | | 6 | 117,4 | 587,44 | 487,8 | 90 | 4 | 99,644 |
| Sum | 3566,18 | 2993,12 | | 600 | 28 | 573,06 | 3415,5 | 2892,5 | 400 | 22 | 523,01 |
| Extra | 563,80  450 | 327,13  293,44 | 90 | | 7  4 | 236,67  156,56 | 616,66  542,23 | 389,99  333,6 | 150 | 7  4 | 226,67  208,63 |
| Cluster 1 |
| Extra | 60 | | 70 |
| Cluster 2 |
| General Sum | 4579,98 | 3613,7 | | 750 | 39 | 966,28 | 4574,4 | 3616 | 620 | 33 | 958,31 |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Σενάριο 7 | Προσέγγιση 3: Min Wait Time | | | | |  | Προσέγγιση | | 4: Min Served First | | |
| Time + Dist. | Wait Time | Demand | Nodes Served | Travel Dist. | Time  + Dist. | Wait Time | Demand | | Nodes Served | Travel Dist. |
| Cluster 1 | 649 | 466,90 | 120 | 8 | 182,1 | 649 | 499,12 | 80 | | 6 | 149,88 |
| Cluster 2 | 700,07 | 567,28 | 110 | 6 | 132,79 | 700,07 | 583,44 | 90 | | 5 | 116,63 |
| Cluster 3 | 794 | 701,95 | 70 | 4 | 92,044 | 794 | 686,32 | 50 | | 4 | 107,68 |
| Cluster 4 | 713 | 626,46 | 40 | 3 | 86,536 | 713 | 616,48 |  | | 4 | 96,519 |
| Cluster 5 | 667 | 519,98 | 110 | 6 | 147,02 | 689 | 574,02 | 60 | | 4 | 114,98 |
| Sum | 3523,07 | 2882,59 | 450 | 27 | 640,48 | 3545,1 | 2959,4 | | 330 | 23 | 585,69 |
| Extra | 691  517 | 26,756  210,32 | 230 | 12  6 | 664,24  306,68 | 607,8  6619 | 418,75  149,21 | 60 | | 4  9 | 189,06  6469,7 |
| Cluster 1 |
| Extra | 160 | 180 | |
| Cluster 2 |
| General Sum | 4731,07 | 3119,67 | 840 | 45 | 1611,4 | 10772 | 3527,3 | | 570 | 36 | 7244,5 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Σενάριο 7 | Π | ροσέγγιση | | 5: Max Served First | | |
| Time + Dist. | Wait Time | Demand | | Nodes Served | Travel Dist. |
| Cluster 1 | 631 | 490,55 | 100 | | 4 | 140,45 |
| Cluster 2 | 683 | 601,12 | 90 | | 3 | 81,881 |
| Cluster 3 | 811 | 707,39 | 110 | | 4 | 103,62 |
| Cluster 4 | 734,18 | 636,29 | 100 | | 4 | 97,886 |
| Cluster 5 | 689 | 566,53 | 140 | | 5 | 122,47 |
| Sum | 3548,18 | 3001,88 | | 540 | 20 | 546,3 |
| Extra | 691  640,898 | 231,68  201,17 | 160 | | 9  6 | 459,32  439,73 |
| Cluster 1 |
| Extra | 90 | |
| Cluster 2 |
| General Sum | 4880,08 | 3434,73 | | 790 | 35 | 1445,3 |

*Πίνακας 21. Σύγκριση των 5 Προσεγγίσεων που χρησιμοποιήθηκαν στο Σενάριο 7.*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ΥΠΟΛΟΓΙΣΜΟΙ ΜΕΤΑ ΤΗΝ ΟΛΟΚΛΗΡΩΣΗ ΤΩΝ ΔΡΟΜΟΛΟΓΙΩΝ** | | | | | |
| Παρατηρήσεις | Προσέγγιση 1 | Προσέγγιση 2 | Προσέγγιση 3 | Προσέγγιση 4 | Προσέγγιση 5 |
| **ΑΡΙΘΜΟΣ ΟΧΗΜΑΤΩΝ** | | | | | |
| Αρχικά Οχήματα | 5 | 5 | 5 | 5 | 5 |
| Επιπλέον Οχήματα | 2 | 2 | 2 | 2 | 2 |
| **ΑΡΙΘΜΟΣ ΚΟΜΒΩΝ ΠΟΥ ΕΞΥΠΗΡΕΤΗΘΗΚΑΝ** | | | | | |
| Αρχικοί Κόμβοι | 28 | 22 | 27 | 23 | 20 |
| Κόμβοι Επιπλέον Διαδρομών | 11 | 11 | 18 | 13 | 15 |
| Συνολικοί Κόμβοι | 39 | 33 | 45 | 36 | 35 |
| **ΖΗΤΗΣΗ ΑΝΑ ΟΧΗΜΑ** | | | | | |
| Συνολική Ζήτηση | 750 | 620 | 840 | 570 | 790 |
| Μέση Τιμή | 107 | 89 | 120 | 81 | 113 |
| Διακύμανση | 1323,81 | 847,62 | 3800,00 | 2114,29 | 723,81 |
| Τυπική Απόκλιση | 36,38 | 29,11 | 61,64 | 45,98 | 26,90 |
| Τυπική Απόκλιση Αρχικών Δρομολ. | 33,91 | 12,25 | 33,91 | 18,17 | 19,24 |
| Τυπική Απόκλιση Επιπλέον Δρομολ. | 21,21 | 56,57 | 49,50 | 84,85 | 49,50 |
| **ΑΠΟΣΤΑΣΗ ΑΝΑ ΟΧΗΜΑ** | | | | | |
| Συνολική Απόσταση | 966,28 | 958,31 | 1611,40 | 7244,50 | 1445,35 |
| Μέση Τιμή | 138,04 | 136,90 | 230,20 | 1034,93 | 206,48 |
| Διακύμανση | 2316,83 | 3119,20 | 42115,35 | 5744342,59 | 27941,97 |
| Τυπική Απόκλιση | 48,13 | 55,85 | 205,22 | 2396,74 | 167,16 |
| **ΑΝΑΜΟΝΗ ΑΝΑ ΟΧΗΜΑ** | | | | | |
| Συνολικός Χρόνος Αναμονής | 3613,70 | 3616,05 | 3119,67 | 3527,33 | 3434,73 |
| Μέση Τιμή | 516,24 | 516,58 | 445,67 | 503,90 | 490,68 |
| Διακύμανση | 23042,28 | 14616,89 | 58335,11 | 31712,20 | 39502,36 |
| Τυπική Απόκλιση | 151,80 | 120,90 | 241,53 | 178,08 | 198,75 |
| **ΣΥΓΚΡΙΣΕΙΣ ΕΠΙΠΛΕΟΝ ΔΕΙΚΤΩΝ** | | | | | |
| Απόσταση Ανά Κόμβο | 24,78 | 29,04 | 35,81 | 201,24 | 41,30 |
| Απόσταση Ανά Ζήτηση | 1,29 | 1,55 | 1,92 | 12,71 | 1,83 |
| Χρόνος Αναμονής Ανά Κόμβο | 92,66 | 109,58 | 69,33 | 97,98 | 98,14 |
| Χρόνος Αναμονής Ανά Ζήτηση | 4,82 | 5,83 | 3,71 | 6,19 | 4,35 |

*Πίνακας 22. Σύγκριση Αποτελεσμάτων Σεναρίου 7.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Προσέγγιση 1 | Προσέγγιση 2 | Προσέγγιση 3 | Προσέγγιση 4 | Προσέγγιση 5 |
| **ΠΟΣΟΣΤΟ ΕΞΥΠΗΡΕΤΗΣΗΣ ΚΟΜΒΩΝ ΜΕ ΤΗ ΧΡΗΣΗ ΤΩΝ ΑΡΧΙΚΩΝ ΟΧΗΜΑΤΩΝ** | | | | |
| 56% | 44% | 54% | 46% | 40% |
| **ΠΟΣΟΣΤΟ ΕΞΥΠΗΡΕΤΗΣΗΣ ΚΟΜΒΩΝ ΜΕ ΤΗΝ ΟΛΟΚΛΗΡΩΣΗ ΟΛΩΝ ΤΩΝ ΔΡΟΜΟΛΟΓΙΩΝ** | | | | |
| 78% | 66% | 90% | 72% | 70% |
| **ΚΑΤΑΝΟΜΗ ΦΟΡΤΙΟΥ ΣΤΟ ΣΥΝΟΛΟ ΤΩΝ ΟΧΗΜΑΤΩΝ** | | | | |
| +35,24% | +8,22% | +129,13% | +70,91% | Μικρότερη τυπική απόκλιση της τάξης των 26,90 μονάδων |
| **ΚΑΤΑΝΟΜΗ ΦΟΡΤΙΟΥ ΣΤΑ ΑΡΧΙΚΑ ΟΧΗΜΑΤΑ** | | | | |
| +176,89% | Μικρότερη τυπική απόκλιση της τάξης των 12,25 μονάδων | +176,89% | +48,32% | +57,06% |
| **ΚΑΤΑΝΟΜΗ ΦΟΡΤΙΟΥ ΣΤΑ ΕΠΙΠΡΟΣΘΕΤΑ ΟΧΗΜΑΤΑ** | | | | |
| Μικρότερη τυπική απόκλιση της τάξης των 21,21 μονάδων | +166,67% | +133,33% | +300,00% | +133,33% |
| **ΣΥΝΟΛΙΚΗ ΑΠΟΣΤΑΣΗ ΔΡΟΜΟΛΟΓΙΩΝ** | | | | |
| +0,83% | Βέλτιστη λύση 958,31 μονάδες | +68,15% | +655,97% | +50,82% |
| **ΚΑΤΑΝΟΜΗ ΑΠΟΣΤΑΣΗΣ ΔΡΟΜΟΛΟΓΙΩΝ ΣΤΟ ΣΥΝΟΛΟ ΤΩΝ ΟΧΗΜΑΤΩΝ** | | | | |
| Μικρότερη τυπική απόκλιση της τάξης των 48,13 μονάδων | +16,03% | +326,36% | +4879,35% | +247,28% |
| **ΣΥΝΟΛΙΚΟΣ ΧΡΟΝΟΣ ΑΝΑΜΟΝΗΣ ΟΧΗΜΑΤΩΝ** | | | | |
| +15,84% | +15,91% | Βέλτιστη λύση με αναμονή 3119,67 μονάδες χρόνου | +13,07% | +10,10% |
| **ΚΑΤΑΝΟΜΗ ΧΡΟΝΟΥ ΑΝΑΜΟΝΗΣ ΣΤΟ ΣΥΝΟΛΟ ΤΩΝ ΟΧΗΜΑΤΩΝ** | | | | |
| +25,56% | Μικρότερη τυπική απόκλιση της τάξης των 120,90 μονάδων | +99,77% | +47,29% | +64,39% |
| **ΑΠΟΣΤΑΣΗ ΠΟΥ ΔΙΑΝΎΕΤΑΙ ΑΝΑ ΕΞΥΠΗΡΕΤΟΥΜΕΝΟ ΚΟΜΒΟ** | | | | |
| Βέλτιστη λύση 24,78 μονάδες | +17,21% | +44,53% | +712,21% | +66,67% |
| **ΧΡΟΝΟΣ ΑΝΑΜΟΝΗΣ ΑΝΑ ΕΞΥΠΗΡΕΤΟΥΜΕΝΟ ΚΟΜΒΟ** | | | | |
| +33,66% | +58,06% | Βέλτιστη λύση 69,33 μονάδες | +41,33% | +41,56% |

114

## Σενάριο 8

Η ομαδοποίηση των κόμβων σύμφωνα με το **Σενάριο 8**:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Σενάριο 8 | | | | | | | |
| Συστάδες | Κόμβοι | | | | | | |
| 1 | 0  26 | 28  50 | 33  31 | 29  34 | 30 | 27 | 32 |
| 2 | 0  7 | 3  6 | 8  38 | 45  2 | 46 | 1 | 5 |
| 3 | 0 | 4 | 11 | 22 |  |  |  |
| 4 | 0  24 | 19  20 | 48  49 | 21 | 25 | 18 | 23 |
| 5 | 0  36 | 41  40 | 37  39 | 35 | 44 | 42 | 43 |
| 6 | 0  9 | 10  15 | 13  12 | 47 | 17 | 14 | 16 |

Οι πίνακες αποτελεσμάτων των 5 αλγορίθμων:

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Σενάριο 8 | Προσέγγιση 1: Min Distance | | | | | Προσέγγιση 2: Inv. Dijkstra | | | | |
| Time + Dist. | Wait Time | Demand | Nodes Served | Travel Dist. | Time  + Dist. | Wait Time | Demand | Nodes Served | Travel Dist. |
| Cluster 1 | 649 | 547,27 | 100 | 4 | 101,73 | 631 | 524,48 | 60 | 4 | 106,52 |
| Cluster 2 | 683 | 557,79 | 120 | 7 | 125,21 | 683 | 607,69 | 50 | 3 | 75,311 |
| Cluster 3 | 654 | 481,86 |  | 3 | 172,14 | 385 | 236,58 | 120 | 3 | 148,43 |
| Cluster 4 | 794 | 687,94 | 60 | 5 | 106,06 | 762,77 | 649,77 | 80 | 5 | 113 |
| Cluster 5 | 667 | 536,76 | 100 | 7 | 130,24 | 587,44 | 487,8 | 90 | 4 | 99,644 |
| Cluster 6 | 713 | 620,19 | 80 | 4 | 92,803 | 734,18 | 622,77 | 80 | 5 | 111,41 |
| Sum | 4160 | 3431,83 | 580 | 30 | 728,17 | 3783,4 | 3129,1 | 480 | 24 | 654,31 |
| Extra Cluster 1 | 447,583 | 169 | 180 | 9 | 278,58 | 616,66 389,99 | | 150 | 7 | 226,67 |
| General Sum | 4607,58 | 3600,83 | 760 | 39 | 1006,8 | 4400,1 | 3519,1 | 630 | 31 | 880,99 |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Σενάριο 8 | Προσέγγιση 3: Min Wait Time | | | | |  | Προσέγγιση | | 4: Min Served First | | |
| Time + Dist. | Wait Time | Demand | Nodes Served | Travel Dist. | Time  + Dist. | Wait Time | Demand | | Nodes Served | Travel Dist. |
| Cluster 1 | 649 | 466,90 | 120 | 8 | 182,1 | 649 | 499,12 | 80 | | 6 | 149,88 |
| Cluster 2 | 683 | 567,29 | 70 | 5 | 115,71 | 683 | 583,44 | 50 | | 4 | 99,562 |
| Cluster 3 | 385 | 236,57 | 120 | 3 | 148,43 | 385 | 236,58 | 120 | | 3 | 148,43 |
| Cluster 4 | 794 | 687,74 | 80 | 4 | 106,26 | 794 | 686,32 | 50 | | 4 | 107,68 |
| Cluster 5 | 667 | 483,95 | 160 | 8 | 183,05 | 689 | 555,32 | 80 | | 5 | 133,68 |
| 713 | 612,32 | 4 | 100,68 | 713 | 616,48 | 4 | 96,519 |
| Cluster 6 | 70 | 50 | |
| Sum | 3891 | 3054,78 | 620 | 32 | 836,22 | 3913 | 3177,3 | | 430 | 26 | 735,75 |
| Extra Cluster 1 | 568,74 0 | | 220 | 11 | 568,75 | 607,8 418,75 | | 60 | | 4 | 189,06 |
| General Sum | 4459,75 | 3054,78 | 840 | 43 | 1405 | 4520,8 | 3596 | | 490 | 30 | 924,8 |

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Σενάριο 8 | Προσέγγιση 5: Max Served First | | | | |
| Time + Dist. | Wait Time | Demand | Nodes Served | Travel Dist. |
| Cluster 1 | 631 | 490,55 | 100 | 4 | 140,45 |
| Cluster 2 | 683 | 565,05 | 80 | 3 | 117,95 |
| Cluster 3 | 385 | 236,57 | 120 | 3 | 148,43 |
| Cluster 4 | 811 | 714,54 | 100 | 4 | 96,453 |
| Cluster 5 | 689 | 555,33 | 130 | 5 | 133,66 |
| Cluster 6 | 734,18 | 650,19 | 60 | 3 | 83,983 |
| Sum | 3933,18 | 3212,26 | 590 | 22 | 720,92 |
| Extra Cluster 1 | 691 294,13 | | 130 | 8 | 396,87 |
| General Sum | 4624,18 | 3506,38 | 720 | 30 | 1117,8 |

*Πίνακας 23. Σύγκριση των 5 Προσεγγίσεων που χρησιμοποιήθηκαν στο Σενάριο 8.*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ΥΠΟΛΟΓΙΣΜΟΙ ΜΕΤΑ ΤΗΝ ΟΛΟΚΛΗΡΩΣΗ ΤΩΝ ΔΡΟΜΟΛΟΓΙΩΝ** | | | | | |
| Παρατηρήσεις | Προσέγγιση 1 | Προσέγγιση 2 | Προσέγγιση 3 | Προσέγγιση 4 | Προσέγγιση 5 |
| **ΑΡΙΘΜΟΣ ΟΧΗΜΑΤΩΝ** | | | | | |
| Αρχικά Οχήματα | 5 | 5 | 5 | 5 | 5 |
| Επιπλέον Οχήματα | 2 | 2 | 2 | 2 | 2 |
| **ΑΡΙΘΜΟΣ ΚΟΜΒΩΝ ΠΟΥ ΕΞΥΠΗΡΕΤΗΘΗΚΑΝ** | | | | | |
| Αρχικοί Κόμβοι | 30 | 24 | 32 | 26 | 22 |
| Κόμβοι Επιπλέον Διαδρομών | 9 | 7 | 11 | 4 | 8 |
| Συνολικοί Κόμβοι | 39 | 31 | 43 | 30 | 30 |
| **ΖΗΤΗΣΗ ΑΝΑ ΟΧΗΜΑ** | | | | | |
| Συνολική Ζήτηση | 760 | 630 | 840 | 490 | 720 |
| Μέση Τιμή | 108,57 | 90,00 | 120,00 | 70,00 | 102,86 |
| Διακύμανση | 1447,62 | 1200,00 | 3033,33 | 666,67 | 690,48 |
| Τυπική Απόκλιση | 38,05 | 34,64 | 55,08 | 25,82 | 26,28 |
| Τυπική Απόκλιση Αρχικών Δρομολ. | 23,38 | 24,49 | 36,15 | 27,87 | 25,63 |
| Τυπική Απόκλιση Επιπλέον Δρομολ. | Δεν διατίθεται για μέγεθος πληθυσμού μικρότερο των 2 σημείων | | | | |
| **ΑΠΟΣΤΑΣΗ ΑΝΑ ΟΧΗΜΑ** | | | | | |
| Συνολική Απόσταση | 1006,76 | 880,99 | 1404,97 | 924,80 | 1117,80 |
| Μέση Τιμή | 143,82 | 125,86 | 200,71 | 132,11 | 159,69 |
| Διακύμανση | 4215,78 | 2443,80 | 27498,87 | 1125,03 | 11483,72 |
| Τυπική Απόκλιση | 64,93 | 49,43 | 165,83 | 33,54 | 107,16 |
| **ΑΝΑΜΟΝΗ ΑΝΑ ΟΧΗΜΑ** | | | | | |
| Συνολικός Χρόνος Αναμονής | 3600,83 | 3519,07 | 3054,78 | 3596,00 | 3506,38 |
| Μέση Τιμή | 514,40 | 502,72 | 436,40 | 513,71 | 500,91 |
| Διακύμανση | 27522,31 | 21881,84 | 57469,61 | 22163,96 | 31290,36 |
| Τυπική Απόκλιση | 165,90 | 147,93 | 239,73 | 148,88 | 176,89 |
| **ΣΥΓΚΡΙΣΕΙΣ ΕΠΙΠΛΕΟΝ ΔΕΙΚΤΩΝ** | | | | | |
| Απόσταση Ανά Κόμβο | 25,81 | 28,42 | 32,67 | 30,83 | 37,26 |
| Απόσταση Ανά  Ζήτηση | 1,32 | 1,40 | 1,67 | 1,89 | 1,55 |
| Χρόνος Αναμονής Ανά Κόμβο | 92,33 | 113,52 | 71,04 | 119,87 | 116,88 |
| Χρόνος Αναμονής  Ανά Ζήτηση | 4,74 | 5,59 | 3,64 | 7,34 | 4,87 |

*Πίνακας 24. Σύγκριση Αποτελεσμάτων Σεναρίου 8.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Προσέγγιση 1 | Προσέγγιση 2 | Προσέγγιση 3 | Προσέγγιση 4 | Προσέγγιση 5 |
| **ΠΟΣΟΣΤΟ ΕΞΥΠΗΡΕΤΗΣΗΣ ΚΟΜΒΩΝ ΜΕ ΤΗ ΧΡΗΣΗ ΤΩΝ ΑΡΧΙΚΩΝ ΟΧΗΜΑΤΩΝ** | | | | |
| 60% | 48% | 64% | 52% | 44% |
| **ΠΟΣΟΣΤΟ ΕΞΥΠΗΡΕΤΗΣΗΣ ΚΟΜΒΩΝ ΜΕ ΤΗΝ ΟΛΟΚΛΗΡΩΣΗ ΟΛΩΝ ΤΩΝ ΔΡΟΜΟΛΟΓΙΩΝ** | | | | |
| 78% | 62% | 86% | 60% | 60% |
| **ΚΑΤΑΝΟΜΗ ΦΟΡΤΙΟΥ ΣΤΟ ΣΥΝΟΛΟ ΤΩΝ ΟΧΗΜΑΤΩΝ** | | | | |
| +47,36% | +34,16% | +113,31% | Μικρότερη τυπική απόκλιση της τάξης των 25,82 μονάδων | +1,77% |
| **ΚΑΤΑΝΟΜΗ ΦΟΡΤΙΟΥ ΣΤΑ ΑΡΧΙΚΑ ΟΧΗΜΑΤΑ** | | | | |
| Μικρότερη τυπική απόκλιση της τάξης των 23,38 μονάδων | +4,76% | +54,60% | +19,19% | +9,60% |
| **ΣΥΝΟΛΙΚΗ ΑΠΟΣΤΑΣΗ ΔΡΟΜΟΛΟΓΙΩΝ** | | | | |
| +14,28% | Βέλτιστη λύση 880,99 μονάδες | +59,48% | +4,97% | +26,88% |
| **ΚΑΤΑΝΟΜΗ ΑΠΟΣΤΑΣΗΣ ΔΡΟΜΟΛΟΓΙΩΝ ΣΤΟ ΣΥΝΟΛΟ ΤΩΝ ΟΧΗΜΑΤΩΝ** | | | | |
| +93,58% | +47,38% | +394,40% | Μικρότερη τυπική απόκλιση της τάξης των 33,54 μονάδων | +219,49% |
| **ΣΥΝΟΛΙΚΟΣ ΧΡΟΝΟΣ ΑΝΑΜΟΝΗΣ ΟΧΗΜΑΤΩΝ** | | | | |
| +17,88% | +15,20% | Βέλτιστη λύση με αναμονή 3054,78 μονάδες χρόνου | +17,72% | +14,78% |
| **ΚΑΤΑΝΟΜΗ ΧΡΟΝΟΥ ΑΝΑΜΟΝΗΣ ΣΤΟ ΣΥΝΟΛΟ ΤΩΝ ΟΧΗΜΑΤΩΝ** | | | | |
| +12,15% | Μικρότερη τυπική απόκλιση της τάξης των 147,93 μονάδων | +62,06% | +0,64% | +19,58% |
| **ΑΠΟΣΤΑΣΗ ΠΟΥ ΔΙΑΝΎΕΤΑΙ ΑΝΑ ΕΞΥΠΗΡΕΤΟΥΜΕΝΟ ΚΟΜΒΟ** | | | | |
| Βέλτιστη λύση 25,81 μονάδες | +10,09% | +26,57% | +19,42% | +44,34% |
| **ΧΡΟΝΟΣ ΑΝΑΜΟΝΗΣ ΑΝΑ ΕΞΥΠΗΡΕΤΟΥΜΕΝΟ ΚΟΜΒΟ** | | | | |
| +29,97% | +59,79% | Βέλτιστη λύση 71,04 μονάδες | +68,73% | +64,52% |

118

## 5.4.3 ΑΠΟΤΕΛΕΣΜΑΤΑ ΜΕΓΑΛΟΥ ΣΥΝΟΛΟΥ ΔΕΔΟΜΕΝΩΝ Σενάριο 9

Η ομαδοποίηση των κόμβων σύμφωνα με το **Σενάριο 9**:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Σενάριο 9 | | | | | | | |
| Συστάδες | Κόμβοι | | | | | | |
| 1 | 0  68 | 53  55 | 80  88 | 90  41 | 82  11 | 65  69 | 98 |
| 2 | 0  26 | 28  50 | 33  31 | 29  34 | 30 | 27 | 32 |
|  | 0 | 52 | 9 | 47 | 87 | 99 | 59 |
| 3 | 14 | 10 | 78 | 17 | 13 | 97 | 16 |
|  | 15 | 12 |  |  |  |  |  |
|  | 0 | 73 | 100 | 3 | 60 | 45 | 8 |
| 4 | 46 | 70 | 1 | 5 | 79 | 7 | 4 |
|  | 6 | 2 |  |  |  |  |  |
|  | 0 | 77 | 75 | 74 | 48 | 21 | 20 |
| 5 | 49 | 58 | 18 | 24 | 86 | 22 | 57 |
|  | 25 | 23 |  |  |  |  |  |
|  | 0 | 93 | 81 | 61 | 38 | 37 | 71 |
| 6 | 44 | 72 | 39 | 42 | 35 | 43 | 54 |
|  | 96 | 40 | 36 |  |  |  |  |
|  | 0 | 85 | 62 | 63 | 67 | 83 | 89 |
| 7 | 95 | 91 | 66 | 92 | 94 | 76 | 51 |
|  | 64 | 56 | 19 | 84 |  |  |  |

Οι πίνακες αποτελεσμάτων των 5 αλγορίθμων:

119

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Σενάριο 9 | Προσέγγιση | | | 1: Min Distance | | | Προσέγγιση 2: Inv. Dijkstra | | | | | Προσέγγιση 3: Min Wait Time | | | | |
| Time + Dist. | Wait Time | Demand | | Nodes Served | Travel Dist. | Time  + Dist. | Wait Time | Demand | Nodes Served | Travel Dist. | Time  + Dist. | Wait Time | Demand | Nodes Served | Travel Dist. |
| Cluster 1 | 832 | 685,05 | 117 | | 8 | 146,95 | 832 | 693,95 | 95 | 5 | 138,05 | 832 | 662,31 | 60 | 5 | 169,69 |
| Cluster 2 | 649 | 547,27 | 100 | | 4 | 101,73 | 631 | 524,48 | 60 | 4 | 106,52 | 649 | 466,9 | 120 | 8 | 182,1 |
| Cluster 3 | 713 | 574,85 |  | | 6 | 138,15 | 713 | 549,61 | 147 | 7 | 163,4 | 734,18 | 315,79 | 237 | 13 | 418,39 |
| Cluster 4 | 820 | 776,29 | 26 | | 2 | 43,708 | 834,47 | 700,59 | 116 | 6 | 133,88 | 834,47 | 676,1 | 61 | 5 | 158,37 |
| 851,79 | 700,72 | 7 | 151,08 | 851,8 | 703,81 | 5 | 147,99 | 826,92 | 527,28 | 9 | 299,64 |
| Cluster 5 | 116 | | 73 | 131 |
| Cluster 6 | 818 | 747,31 | 55 | | 4 | 70,691 | 818 | 651,03 | 135 | 7 | 166,97 | 818 | 680,36 | 71 | 5 | 137,64 |
| 723 | 575,22 | 6 | 147,78 | 729 | 553,87 | 127 | 7 | 175,13 | 774,15 | 500,3 | 9 | 273,85 |
| Cluster 7 | 110 | | 151 |
| Sum | 5406,8 | 4606,71 | | 620 | 37 | 800,08 | 5409,3 | 4377,3 | 753 | 41 | 1031,9 | 5468,7 | 3829 | 831 | 54 | 1639,7 |
| Extra | 776,687  691  560  476,25 | 571,17  439,18  416,70  211,09 | 186 | | 9  9  7  9 | 205,51  251,82  143,3  265,16 | 769,99  593,98  492,04  451,69 | 473,9  248,62  272,03  222,67 | 171 | 10  10  8  9 | 296,09  345,36  220  229,02 | 815,38  711,18  663,02  547 | 149,84  0  317,59  333,68 | 251 | 13  17  9  5 | 665,54  711,18  345,43  213,32 |
| Cluster 1 |
| Extra | 161 | | 110 | 329 |
| Cluster 2 |
| Extra | 90 | | 215 | 203 |
| Cluster 3 |
| Extra | 122 | | 155 | 66 |
| Cluster 4 |
| General Sum | 7910,73 | 6244,86 | | 1179 | 71 | 1665,9 | 7717 | 5594,5 | 1404 | 78 | 2122,4 | 8205,3 | 4630,1 | 1680 | 98 | 3575,2 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Σενάριο 9 |  | Προσέγγιση | | 4: Min Served First | | | Π | ροσέγγιση | | 5: Max Served First | | |
| Time + Dist. | Wait Time | Demand | | Nodes Served | Travel Dist. | Time  + Dist. | Wait Time | Demand | | Nodes Served | Travel Dist. |
| Cluster 1 | 832 | 754,74 | 24 | | 3 | 77,263 | 832 | 679,09 | 117 | | 5 | 152,91 |
| Cluster 2 | 649 | 499,12 | 80 | | 6 | 149,88 | 631 | 490,55 | 100 | | 4 | 140,45 |
| Cluster 3 | 713 | 586,36 | 53 | | 5 | 126,64 | 734,18 | 600,85 | 95 | | 4 | 133,33 |
| Cluster 4 | 820 | 596,49 | 88 | | 7 | 223,51 | 834,47 | 706,49 |  | | 4 | 127,98 |
| Cluster 5 | 826,92 | 618,13 | 75 | | 6 | 208,79 | 868,72 | 641,04 | 178 | | 8 | 227,68 |
| Cluster 6 | 689 | 455,95 | 96 | | 8 | 233,05 | 818 | 572,13 | 185 | | 8 | 245,87 |
| 780,146 | 597,71 | 6 | 182,43 | 774,15 | 624,94 | 4 | 149,21 |
| Cluster 7 | 58 | | 109 | |
| Sum | 5310,07 | 4108,5 | | 474 | 41 | 1201,6 | 5492,5 | 4315,1 | | 880 | 37 | 1177,4 |
| Extra | 749,402  607,495  604,418  581,745 | 304,01  109,09  153,02  187,45 | 166  155 | | 11  9  8  10 | 445,4  498,4  451,4  394,3 | 780,46  655  593,07  521,7 | 467,44  287,17  188,15  182,39 | 147 | | 7  8  10  7 | 313,02  367,83  404,92  339,3 |
| Cluster 1 |
| Extra | 133 | |
| Cluster 2 |
| Extra | 139 | | 167 | |
| Cluster 3 |
| Extra | 208 | | 104 | |
| Cluster 4 |
| General Sum | 7853,13 | 4862,08 | | 1142 | 79 | 2991,1 | 8042,8 | 5440,3 | | 1431 | 69 | 2602,5 |

*Πίνακας 25. Σύγκριση των 5 Προσεγγίσεων που χρησιμοποιήθηκαν στο Σενάριο 9.*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ΥΠΟΛΟΓΙΣΜΟΙ ΜΕΤΑ ΤΗΝ ΟΛΟΚΛΗΡΩΣΗ ΤΩΝ ΔΡΟΜΟΛΟΓΙΩΝ** | | | | | |
| Παρατηρήσεις | Προσέγγιση 1 | Προσέγγιση 2 | Προσέγγιση 3 | Προσέγγιση 4 | Προσέγγιση 5 |
| **ΑΡΙΘΜΟΣ ΟΧΗΜΑΤΩΝ** | | | | | |
| Αρχικά Οχήματα | 7 | 7 | 7 | 7 | 7 |
| Επιπλέον Οχήματα | 4 | 4 | 4 | 4 | 4 |
| **ΑΡΙΘΜΟΣ ΚΟΜΒΩΝ ΠΟΥ ΕΞΥΠΗΡΕΤΗΘΗΚΑΝ** | | | | | |
| Αρχικοί Κόμβοι | 37 | 41 | 54 | 41 | 37 |
| Κόμβοι Επιπλέον Διαδρομών | 34 | 37 | 44 | 38 | 32 |
| Συνολικοί Κόμβοι | 71 | 78 | 98 | 79 | 69 |
| **ΖΗΤΗΣΗ ΑΝΑ ΟΧΗΜΑ** | | | | | |
| Συνολική Ζήτηση | 1179 | 1404 | 1680 | 1142 | 1431 |
| Μέση Τιμή | 107,18 | 127,64 | 152,73 | 103,82 | 130,09 |
| Διακύμανση | 1929,56 | 1976,25 | 8311,82 | 3133,96 | 1156,29 |
| Τυπική Απόκλιση | 43,93 | 44,46 | 91,17 | 55,98 | 34,00 |
| Τυπική Απόκλιση Αρχικών Δρομολ. | 34,76 | 32,59 | 63,56 | 24,62 | 38,93 |
| Τυπική Απόκλιση  Επιπλέον Δρομολ. | 42,35 | 43,36 | 110,46 | 29,50 | 26,47 |
| **ΑΠΟΣΤΑΣΗ ΑΝΑ ΟΧΗΜΑ** | | | | | |
| Συνολική Απόσταση | 1665,87 | 2122,41 | 3575,16 | 2991,05 | 2602,50 |
| Μέση Τιμή | 151,44 | 192,95 | 325,01 | 271,91 | 236,59 |
| Διακύμανση | 4680,61 | 5375,89 | 39738,69 | 21831,34 | 10845,53 |
| Τυπική Απόκλιση | 68,41 | 73,32 | 199,35 | 147,75 | 104,14 |
| **ΑΝΑΜΟΝΗ ΑΝΑ ΟΧΗΜΑ** | | | | | |
| Συνολικός Χρόνος Αναμονής | 6244,86 | 5594,55 | 4630,15 | 4862,08 | 5440,25 |
| Μέση Τιμή | 567,71 | 508,60 | 420,92 | 442,01 | 494,57 |
| Διακύμανση | 27425,63 | 34010,46 | 49042,00 | 47941,12 | 36982,82 |
| Τυπική Απόκλιση | 165,61 | 184,42 | 221,45 | 218,95 | 192,31 |
| **ΣΥΓΚΡΙΣΕΙΣ ΕΠΙΠΛΕΟΝ ΔΕΙΚΤΩΝ** | | | | | |
| Απόσταση Ανά Κόμβο | 23,46 | 27,21 | 36,48 | 37,86 | 37,72 |
| Απόσταση Ανά  Ζήτηση | 1,41 | 1,51 | 2,13 | 2,62 | 1,82 |
| Χρόνος Αναμονής Ανά Κόμβο | 87,96 | 71,72 | 47,25 | 61,55 | 78,84 |
| Χρόνος Αναμονής  Ανά Ζήτηση | 5,30 | 3,98 | 2,76 | 4,26 | 3,80 |

122

*Πίνακας 26. Σύγκριση Αποτελεσμάτων Σεναρίου 9.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Προσέγγιση 1 | Προσέγγιση 2 | Προσέγγιση 3 | Προσέγγιση 4 | Προσέγγιση 5 |
| **ΠΟΣΟΣΤΟ ΕΞΥΠΗΡΕΤΗΣΗΣ ΚΟΜΒΩΝ ΜΕ ΤΗ ΧΡΗΣΗ ΤΩΝ ΑΡΧΙΚΩΝ ΟΧΗΜΑΤΩΝ** | | | | |
| 37% | 41% | 54% | 41% | 37% |
| **ΠΟΣΟΣΤΟ ΕΞΥΠΗΡΕΤΗΣΗΣ ΚΟΜΒΩΝ ΜΕ ΤΗΝ ΟΛΟΚΛΗΡΩΣΗ ΟΛΩΝ ΤΩΝ ΔΡΟΜΟΛΟΓΙΩΝ** | | | | |
| 71% | 78% | 98% | 79% | 69% |
| **ΚΑΤΑΝΟΜΗ ΦΟΡΤΙΟΥ ΣΤΟ ΣΥΝΟΛΟ ΤΩΝ ΟΧΗΜΑΤΩΝ** | | | | |
| +29,18% | +30,73% | +168,11% | +64,63% | Μικρότερη τυπική απόκλιση της τάξης των 34 μονάδων |
| **ΚΑΤΑΝΟΜΗ ΦΟΡΤΙΟΥ ΣΤΑ ΑΡΧΙΚΑ ΟΧΗΜΑΤΑ** | | | | |
| +41,16% | +32,35% | +158,16% | Μικρότερη τυπική απόκλιση της τάξης των 24,62 μονάδων | +58,10% |
| **ΚΑΤΑΝΟΜΗ ΦΟΡΤΙΟΥ ΣΤΑ ΕΠΙΠΡΟΣΘΕΤΑ ΟΧΗΜΑΤΑ** | | | | |
| +59,97% | +63,79% | +317,24% | +11,41% | Μικρότερη τυπική απόκλιση της τάξης των 26,47 μονάδων |
| **ΣΥΝΟΛΙΚΗ ΑΠΟΣΤΑΣΗ ΔΡΟΜΟΛΟΓΙΩΝ** | | | | |
| Βέλτιστη λύση 1665,87 μονάδες | +27,41% | +114,61% | +79,55% | +56,22% |
| **ΚΑΤΑΝΟΜΗ ΑΠΟΣΤΑΣΗΣ ΔΡΟΜΟΛΟΓΙΩΝ ΣΤΟ ΣΥΝΟΛΟ ΤΩΝ ΟΧΗΜΑΤΩΝ** | | | | |
| Μικρότερη τυπική απόκλιση της τάξης των 68,41 μονάδων | +7,17% | +191,38% | +115,97% | +52,22% |
| **ΣΥΝΟΛΙΚΟΣ ΧΡΟΝΟΣ ΑΝΑΜΟΝΗΣ ΟΧΗΜΑΤΩΝ** | | | | |
| +34,87% | +20,83% | Βέλτιστη λύση με αναμονή 4630,15 μονάδες χρόνου | +5,01% | +17,50% |
| **ΚΑΤΑΝΟΜΗ ΧΡΟΝΟΥ ΑΝΑΜΟΝΗΣ ΣΤΟ ΣΥΝΟΛΟ ΤΩΝ ΟΧΗΜΑΤΩΝ** | | | | |
| Μικρότερη τυπική απόκλιση της τάξης των 165,61 μονάδων | +11,36% | +33,72% | +32,21% | +16,12% |
| **ΑΠΟΣΤΑΣΗ ΠΟΥ ΔΙΑΝΎΕΤΑΙ ΑΝΑ ΕΞΥΠΗΡΕΤΟΥΜΕΝΟ ΚΟΜΒΟ** | | | | |
| Βέλτιστη λύση 23,46 μονάδες | +15,97% | +55,48% | +61,37% | +60,75% |
| **ΧΡΟΝΟΣ ΑΝΑΜΟΝΗΣ ΑΝΑ ΕΞΥΠΗΡΕΤΟΥΜΕΝΟ ΚΟΜΒΟ** | | | | |
| +86,16% | +51,81% | Βέλτιστη λύση 47,25 μονάδες | +30,26% | +66,88% |

123

## Σενάριο 10

Η ομαδοποίηση των κόμβων σύμφωνα με το **Σενάριο 10**:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Σενάριο 10 | | | | | | | |
| Συστάδες | Κόμβοι | | | | | | |
| 1 | 0  69 | 93  94 | 90  68 | 71  81 | 61  96 | 80  4 | 54 |
| 2 | 0  26 | 28  50 | 33  31 | 29  34 | 30 | 27 | 32 |
| 3 | 0  57 | 52  9 | 75  59 | 13  97 | 74  87 | 99  11 | 58 |
|  | 0 | 73 | 82 | 98 | 53 | 47 | 14 |
| 4 | 88 | 16 | 60 | 15 | 17 | 10 | 78 |
|  | 12 |  |  |  |  |  |  |
| 5 | 0  22 | 77  25 | 89  20 | 21  18 | 48  23 | 24  49 | 76 |
| 6 | 0  39 | 72  40 | 37  36 | 44  41 | 35  38 | 42 | 43 |
|  | 0 | 85 | 62 | 63 | 67 | 65 | 95 |
| 7 | 83 | 92 | 51 | 91 | 66 | 84 | 64 |
|  | 86 | 56 | 19 |  |  |  |  |
| 8 | 0  46 | 100  70 | 3  79 | 2  1 | 55  5 | 8  7 | 45  6 |

Οι πίνακες αποτελεσμάτων των 5 αλγορίθμων:

124

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Σενάριο 10 | Προσέγγιση 1: Min Distance | | | | | Προσέγγιση 2: Inv. Dijkstra | | | | | Προσέγγιση 3: Min Wait Time | | | | |
| Time + Dist. | Wait Time | Demand | Nodes Served | Travel Dist. | Time  + Dist. | Wait Time | Demand | Nodes Served | Travel Dist. | Time  + Dist. | Wait Time | Demand | Nodes Served | Travel Dist. |
| Cluster 1 | 856,866 753,14  649 547,27  792 686,95  735,21 642,35  840,105 716,56  689 564,74  729 654,82  834,472 778,78 | | 70 | 5 103,73  4 101,73  5 105,05  4 92,86  5 123,55  6 124,25  4 74,179  3 55,685 | | 856,87 722,9  631 524,48  792 703,81  744,21 622,77  816,17 655,16  587,44 487,8  729 579,23  834,47 715,82 | | 77 | 4 133,97  4 106,52  2 88,189  5 121,44  6 161,01  4 99,644  6 149,77  5 118,65 | | 832 697,26  649 466,9  792 544,3  735,21 504,76  801,45 608,98  667 482,95  729 357,15  834,47 743,66 | | 55 | 5 134,75  8 182,1  7 247,7  8 230,45  5 192,47  7 184,05  11 371,85  3 90,809 | |
| Cluster 2 | 100 | 60 | 120  116  124 |
| Cluster 3 | 69 | 36 |
| Cluster 4 | 51 | 87 |
| Cluster 5 | 87 | 113 | 72 |
| Cluster 6 | 108 | 90 | 118 |
| Cluster 7 | 64 | 111 | 137 |
| Cluster 8 | 42 | 76 | 36 |
| Sum | 6125,66 5344,63 549 36 725,34 | | | | | 5991,2 5012 574 36 860,55 | | | | | 6040,1 4406 742 54 1543,4 | | | | |
| Extra Cluster 1 | 836,382 575,49  641,274 425,87  515,403 337,25 | | 147  142  150 | 7 260,89  8 215,4  9 178,15 | | 773,59 429,5  609,92 275,13  556,04 365,68 | | 212 | 12 344,1  11 334,8  6 190,37 | | 748,39 219,68  700,35 21,996  616,29 331,1 | | 273  292 | 15 528,72  14 678,36  8 285,19 | |
| Extra Cluster 2 | 147 |
| Extra Cluster 3 | 123 | 171 |
| General Sum | 8118,72 6683,26 988 60 1379,8 | | | | | 7930,7 6082,3 1056 65 1729,8 | | | | | 8105,2 4978,7 1478 91 3035,6 | | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Σενάριο 10 |  | Προσέγγιση | | 4: Min Served First | | | Π | ροσέγγιση | | 5: Max Served First | | |
| Time + Dist. | Wait Time | Demand | | Nodes Served | Travel Dist. | Time  + Dist. | Wait Time | Demand | | Nodes Served | Travel Dist. |
| Cluster 1 | 832 | 718,93 | 32 | | 4 | 113,07 | 856,87 | 635,36 | 146 | | 7 | 221,51 |
| Cluster 2 | 649 | 499,11 | 80 | | 6 | 149,88 | 631 | 490,55 | 100 | | 4 | 140,45 |
| Cluster 3 | 792 | 667,68 | 41 | | 4 | 124,32 | 792 | 692,51 | 95 | | 3 | 99,488 |
| Cluster 4 | 735,21 | 602,93 |  | | 4 | 132,28 | 744,21 | 602,36 | 114 | | 5 | 141,85 |
| Cluster 5 | 836 | 653,69 | 82 | | 6 | 182,31 | 840,11 | 604,69 | 163 | | 7 | 235,42 |
| Cluster 6 | 689 | 588,81 | 58 | | 4 | 100,18 | 689 | 566,53 | 140 | | 5 | 122,47 |
| 729 | 597,71 | 5 | 131,29 | 729 | 535,01 | 6 | 193,99 |
| Cluster 7 | 33 | | 135 | |
| 820 | 646,55 | 6 | 173,45 | 834,47 | 712,71 | 4 | 121,76 |
| Cluster 8 | 82 | | 86 | |
| Sum | 6082,21 | 4975,44 | | 373 | 39 | 933,32 | 6116,7 | 4839,7 | | 893 | 41 | 1155,2 |
| Extra | 751,08  607,49  658,58 | 355,37  220,46  93,10 | 156 | | 10  9  11 | 395,71  387,03  565,49 | 763,85  719,68  643,68 | 515,85  224,59  295,74 | 112 | | 6  9  8 | 248  495,09  347,94 |
| Cluster 1 |
| Extra | 136 | | 158 | |
| Cluster 2 |
| Extra | 205 | | 120 | |
| Cluster 3 |
| General Sum | 8099,38 | 5644,38 | | 870 | 69 | 2281,6 | 8243,9 | 5875,9 | | 1283 | 64 | 2246,2 |

*Πίνακας 27. Σύγκριση των 5 Προσεγγίσεων που χρησιμοποιήθηκαν στο Σενάριο 10.*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ΥΠΟΛΟΓΙΣΜΟΙ ΜΕΤΑ ΤΗΝ ΟΛΟΚΛΗΡΩΣΗ ΤΩΝ ΔΡΟΜΟΛΟΓΙΩΝ** | | | | | |
| Παρατηρήσεις | Προσέγγιση 1 | Προσέγγιση 2 | Προσέγγιση 3 | Προσέγγιση 4 | Προσέγγιση 5 |
| **ΑΡΙΘΜΟΣ ΟΧΗΜΑΤΩΝ** | | | | | |
| Αρχικά Οχήματα | 8 | 8 | 8 | 8 | 8 |
| Επιπλέον Οχήματα | 3 | 3 | 3 | 3 | 3 |
| **ΑΡΙΘΜΟΣ ΚΟΜΒΩΝ ΠΟΥ ΕΞΥΠΗΡΕΤΗΘΗΚΑΝ** | | | | | |
| Αρχικοί Κόμβοι | 36 | 36 | 54 | 39 | 41 |
| Κόμβοι Επιπλέον Διαδρομών | 24 | 29 | 37 | 30 | 23 |
| Συνολικοί Κόμβοι | 60 | 65 | 91 | 69 | 64 |
| **ΖΗΤΗΣΗ ΑΝΑ ΟΧΗΜΑ** | | | | | |
| Συνολική Ζήτηση | 988 | 1056 | 1478 | 870 | 1283 |
| Μέση Τιμή | 89,82 | 96,00 | 134,36 | 79,09 | 116,64 |
| Διακύμανση | 1518,25 | 2254,89 | 6608,25 | 3168,07 | 665,67 |
| Τυπική Απόκλιση | 38,96 | 47,49 | 81,29 | 56,29 | 25,80 |
| Τυπική Απόκλιση Αρχικών Δρομολ. | 22,97 | 25,49 | 37,36 | 21,83 | 27,57 |
| Τυπική Απόκλιση Επιπλέον Δρομολ. | 4,04 | 46,05 | 65,07 | 35,50 | 24,58 |
| **ΑΠΟΣΤΑΣΗ ΑΝΑ ΟΧΗΜΑ** | | | | | |
| Συνολική Απόσταση | 1379,77 | 1729,81 | 3035,62 | 2281,55 | 2246,19 |
| Μέση Τιμή | 125,43 | 157,26 | 275,97 | 207,41 | 204,20 |
| Διακύμανση | 3894,49 | 8029,50 | 31565,92 | 23703,90 | 13962,21 |
| Τυπική Απόκλιση | 62,41 | 89,61 | 177,67 | 153,96 | 118,16 |
| **ΑΝΑΜΟΝΗ ΑΝΑ ΟΧΗΜΑ** | | | | | |
| Συνολικός Χρόνος Αναμονής | 6683,26 | 6082,25 | 4978,73 | 5644,38 | 5875,92 |
| Μέση Τιμή | 607,57 | 552,93 | 452,61 | 513,13 | 534,17 |
| Διακύμανση | 18470,99 | 22673,38 | 44482,94 | 41190,66 | 23278,59 |
| Τυπική Απόκλιση | 135,91 | 150,58 | 210,91 | 202,95 | 152,57 |
| **ΣΥΓΚΡΙΣΕΙΣ ΕΠΙΠΛΕΟΝ ΔΕΙΚΤΩΝ** | | | | | |
| Απόσταση Ανά Κόμβο | 23,00 | 26,61 | 33,36 | 33,07 | 35,10 |
| Απόσταση Ανά  Ζήτηση | 1,40 | 1,64 | 2,05 | 2,62 | 1,75 |
| Χρόνος Αναμονής Ανά Κόμβο | 111,39 | 93,57 | 54,71 | 81,80 | 91,81 |
| Χρόνος Αναμονής  Ανά Ζήτηση | 6,76 | 5,76 | 3,37 | 6,49 | 4,58 |

127

*Πίνακας 28. Σύγκριση Αποτελεσμάτων Σεναρίου 10.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Προσέγγιση 1 | Προσέγγιση 2 | Προσέγγιση 3 | Προσέγγιση 4 | Προσέγγιση 5 |
| **ΠΟΣΟΣΤΟ ΕΞΥΠΗΡΕΤΗΣΗΣ ΚΟΜΒΩΝ ΜΕ ΤΗ ΧΡΗΣΗ ΤΩΝ ΑΡΧΙΚΩΝ ΟΧΗΜΑΤΩΝ** | | | | |
| 36% | 36% | 54% | 39% | 41% |
| **ΠΟΣΟΣΤΟ ΕΞΥΠΗΡΕΤΗΣΗΣ ΚΟΜΒΩΝ ΜΕ ΤΗΝ ΟΛΟΚΛΗΡΩΣΗ ΟΛΩΝ ΤΩΝ ΔΡΟΜΟΛΟΓΙΩΝ** | | | | |
| 60% | 65% | 91% | 69% | 64% |
| **ΚΑΤΑΝΟΜΗ ΦΟΡΤΙΟΥ ΣΤΟ ΣΥΝΟΛΟ ΤΩΝ ΟΧΗΜΑΤΩΝ** | | | | |
| +51,02% | +84,05% | +215,07% | +118,16% | Μικρότερη τυπική απόκλιση της τάξης των 25,80 μονάδων |
| **ΚΑΤΑΝΟΜΗ ΦΟΡΤΙΟΥ ΣΤΑ ΑΡΧΙΚΑ ΟΧΗΜΑΤΑ** | | | | |
| +5,23% | +16,74% | +71,11% | Μικρότερη τυπική απόκλιση της τάξης των 21,83 μονάδων | +26,29% |
| **ΚΑΤΑΝΟΜΗ ΦΟΡΤΙΟΥ ΣΤΑ ΕΠΙΠΡΟΣΘΕΤΑ ΟΧΗΜΑΤΑ** | | | | |
| Μικρότερη τυπική απόκλιση της τάξης των 4,04 μονάδων | +1039,37% | +1510,11% | +778,43% | +508,11% |
| **ΣΥΝΟΛΙΚΗ ΑΠΟΣΤΑΣΗ ΔΡΟΜΟΛΟΓΙΩΝ** | | | | |
| Βέλτιστη λύση 1379,77 μονάδες | +25,37% | +120,01% | +65,36% | +62,79% |
| **ΚΑΤΑΝΟΜΗ ΑΠΟΣΤΑΣΗΣ ΔΡΟΜΟΛΟΓΙΩΝ ΣΤΟ ΣΥΝΟΛΟ ΤΩΝ ΟΧΗΜΑΤΩΝ** | | | | |
| Μικρότερη τυπική απόκλιση της τάξης των 62,41 μονάδων | +43,59% | +184,70% | +146,71% | +89,34% |
| **ΣΥΝΟΛΙΚΟΣ ΧΡΟΝΟΣ ΑΝΑΜΟΝΗΣ ΟΧΗΜΑΤΩΝ** | | | | |
| +34,24% | +22,16% | Βέλτιστη λύση με αναμονή 4978,73 μονάδες χρόνου | +13,37% | +18,02% |
| **ΚΑΤΑΝΟΜΗ ΧΡΟΝΟΥ ΑΝΑΜΟΝΗΣ ΣΤΟ ΣΥΝΟΛΟ ΤΩΝ ΟΧΗΜΑΤΩΝ** | | | | |
| Μικρότερη τυπική απόκλιση της τάξης των 135,91 μονάδων | +10,79% | +55,19% | +49,33% | +12,26% |
| **ΑΠΟΣΤΑΣΗ ΠΟΥ ΔΙΑΝΎΕΤΑΙ ΑΝΑ ΕΞΥΠΗΡΕΤΟΥΜΕΝΟ ΚΟΜΒΟ** | | | | |
| Βέλτιστη λύση 23 μονάδες | +15,73% | +45,06% | +43,79% | +52,62% |
| **ΧΡΟΝΟΣ ΑΝΑΜΟΝΗΣ ΑΝΑ ΕΞΥΠΗΡΕΤΟΥΜΕΝΟ ΚΟΜΒΟ** | | | | |
| +103,59% | +71,03% | Βέλτιστη λύση 54,71 μονάδες | +49,52% | +67,81% |

128

## Σενάριο 11

Η ομαδοποίηση των κόμβων σύμφωνα με το **Σενάριο 11**:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Σενάριο 11 | | | | | | | |
| Συστάδες | Κόμβοι | | | | | | |
| 1 | 0 | 53 | 90 | 82 | 65 | 68 | 98 |
| 55 | 88 | 4 | 69 |  |  |  |
| 2 | 0  92 | 93  80 | 61  54 | 67  94 | 71  96 | 81 | 91 |
| 3 | 0  59 | 52  97 | 75  87 | 74  86 | 99  11 | 58 | 9 |
| 4 | 0  15 | 73  17 | 13  78 | 12  14 | 60  10 | 47 | 16 |
| 5 | 0  19 | 77  25 | 83  49 | 48  20 | 21  23 | 18  24 | 57 |
| 6 | 0  39 | 72  40 | 37  36 | 44  41 | 35  38 | 42 | 43 |
| 7 | 0  64 | 85  56 | 62  76 | 63  51 | 95  84 | 66  22 | 89 |
| 8 | 0  46 | 100  79 | 3  1 | 8  5 | 2  7 | 45  6 | 70 |
| 9 | 0 | 28 | 33 | 29 | 30 | 27 | 32 |
| 26 | 50 | 31 | 34 |  |  |  |

Οι πίνακες αποτελεσμάτων των 5 αλγορίθμων:

129

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Σενάριο 11 | Προσέγγιση | | | 1: Min Distance | | | Προσέγγιση 2: Inv. Dijkstra | | | | | Προσέγγιση 3: Min Wait Time | | | | |
| Time + Dist. | Wait Time | Demand | | Nodes Served | Travel Dist. | Time  + Dist. | Wait Time | Demand | Nodes Served | Travel Dist. | Time  + Dist. | Wait Time | Demand | Nodes Served | Travel Dist. |
| Cluster 1 | 667,28 | 506,82 | 144 | | 8 | 160,47 | 629 | 530,58 | 77 | 3 | 98,416 | 692,65 | 522,19 | 104 | 6 | 170,46 |
| Cluster 2 | 818 | 759,38 | 29 | | 3 | 58,621 | 842,87 | 767,51 | 30 | 3 | 75,361 | 842,87 | 771,19 | 30 | 3 | 71,672 |
| Cluster 3 | 792 | 693,86 | 74 | | 5 | 98,132 | 792 | 703,81 | 36 | 2 | 88,189 | 792 | 584,3 | 106 | 6 | 207,7 |
| Cluster 4 | 734,18 | 615,66 |  | | 5 | 118,51 | 771,11 | 622,77 | 97 | 6 | 148,34 | 735,21 | 534,25 |  | 7 | 200,96 |
| Cluster 5 | 840,11 | 702,29 | 84 | | 6 | 137,81 | 811 | 711,85 | 37 | 3 | 99,146 | 794 | 615,03 | 61 | 5 | 178,97 |
| 689 | 564,74 | 6 | 124,25 | 587,44 | 487,8 | 4 | 99,644 | 667 | 482,95 | 7 | 184,05 |
| Cluster 6 | 108 | | 90 | 118 |
| 723 | 620,34 | 4 | 102,65 | 586,22 | 408,94 | 7 | 177,28 | 723 | 479,03 | 8 | 243,98 |
| Cluster 7 | 77 | | 130 | 145 |
| 820 | 776,29 | 2 | 43,708 | 834,47 | 715,82 | 5 | 118,65 | 834,47 | 743,66 | 3 | 90,809 |
| Cluster 8 | 26 | | 76 | 36 |
| 649 | 547,27 | 4 | 101,73 | 631 | 524,48 | 4 | 106,52 | 649 | 466,9 | 8 | 182,1 |
| Cluster 9 | 100 | | 60 | 120 |
| Sum | 6732,57 | 5786,69 | | 595 | 43 | 800,45 | 6485,1 | 5473,6 | 497 | 37 | 786,37 | 6730,2 | 5199,5 | 673 | 53 | 1257,8 |
| Extra | 683  641,27 | 575,49  425,87 | 72 | | 4  8 | 107,51  215,4 | 768,16  674,15 | 564,95  425,29 | 133 | 7  8 | 203,21  248,86 | 767,57  668,55 | 231,56  21,996 | 222 | 11  15 | 536,01  646,55 |
| Cluster 1 |
| Extra | 142 | | 101 | 282 |
| Cluster 2 |
| General Sum | 8056,84 | 6788,06 | | 809 | 55 | 1123,4 | 7927,4 | 6463,8 | 731 | 52 | 1238,4 | 8166,3 | 5453,1 | 1177 | 79 | 2440,3 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Σενάριο 11 | Προσέγγιση 4: Min Served First | | | | | Προσέγγιση 5: Max Served First | | | | |
| Time + Dist. | Wait Time | Demand | Nodes Served | Travel Dist. | Time  + Dist. | Wait Time | Demand | Nodes Served | Travel Dist. |
| Cluster 1 | 667,284 526,97  832 657,14  792 714,64  735,21 582,60  794 663,70  689 588,82  723 547,44  820 696,76  649 499,12 | | 88 | 5 140,31  6 174,86  3 77,361  4 152,61  4 130,3  4 100,18  6 175,56  5 123,23  6 149,88 | | 642 513,82  856,87 723,63  792 692,51  781,14 589,82  840,11 642,59  689 566,53  723 627,88  834,47 712,71  631 490,55 | | 104 | 4 128,18  6 133,24  3 99,488  7 191,32  7 197,51  5 122,47  3 95,123  4 121,76  4 140,45 | |
| Cluster 2 | 57 | 94  95 |
| Cluster 3 | 31 |
| Cluster 4 | 52  47 | 144 |
| Cluster 5 | 150 |
| Cluster 6 | 58 | 140 |
| Cluster 7 | 71  66 | 96 |
| Cluster 8 | 86 |
| Cluster 9 | 80 | 100 |
| Sum | 6701,5 5477,21 404 43 951,18 | | | | | 6789,6 5560 823 43 967,33 | | | | |
| Extra Cluster 1 | 627 342,31  628,83 193,42 | | 96 | 8 284,69  8 435,41 | | 764,2 498,28  655 368,03 | | 98 | 5 265,92  8 286,97 | |
| Extra Cluster 2 | 131 | 133 |
| General Sum | 7957,33 6012,94 631 59 1671,3 | | | | | 8208,8 6426,4 1054 56 1520,2 | | | | |

*Πίνακας 29. Σύγκριση των 5 Προσεγγίσεων που χρησιμοποιήθηκαν στο Σενάριο 11.*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ΥΠΟΛΟΓΙΣΜΟΙ ΜΕΤΑ ΤΗΝ ΟΛΟΚΛΗΡΩΣΗ ΤΩΝ ΔΡΟΜΟΛΟΓΙΩΝ** | | | | | |
| Παρατηρήσεις | Προσέγγιση 1 | Προσέγγιση 2 | Προσέγγιση 3 | Προσέγγιση 4 | Προσέγγιση 5 |
| **ΑΡΙΘΜΟΣ ΟΧΗΜΑΤΩΝ** | | | | | |
| Αρχικά Οχήματα | 9 | 9 | 9 | 9 | 9 |
| Επιπλέον Οχήματα | 2 | 2 | 2 | 2 | 2 |
| **ΑΡΙΘΜΟΣ ΚΟΜΒΩΝ ΠΟΥ ΕΞΥΠΗΡΕΤΗΘΗΚΑΝ** | | | | | |
| Αρχικοί Κόμβοι | 43 | 37 | 53 | 43 | 43 |
| Κόμβοι Επιπλέον Διαδρομών | 12 | 15 | 26 | 16 | 13 |
| Συνολικοί Κόμβοι | 55 | 52 | 79 | 59 | 56 |
| **ΖΗΤΗΣΗ ΑΝΑ ΟΧΗΜΑ** | | | | | |
| Συνολική Ζήτηση | 809 | 731 | 1177 | 631 | 1054 |
| Μέση Τιμή | 73,55 | 66,45 | 107,00 | 57,36 | 95,82 |
| Διακύμανση | 1449,20 | 1283,36 | 5647,16 | 752,05 | 563,62 |
| Τυπική Απόκλιση | 38,07 | 35,82 | 75,15 | 27,42 | 23,74 |
| Τυπική Απόκλιση Αρχικών Δρομολ. | 36,84 | 33,09 | 40,06 | 17,38 | 25,01 |
| Τυπική Απόκλιση Επιπλέον Δρομολ. | 49,50 | 22,63 | 42,43 | 24,75 | 24,75 |
| **ΑΠΟΣΤΑΣΗ ΑΝΑ ΟΧΗΜΑ** | | | | | |
| Συνολική Απόσταση | 1123,35 | 1238,44 | 2440,34 | 1671,27 | 1520,22 |
| Μέση Τιμή | 102,12 | 112,59 | 221,85 | 151,93 | 138,20 |
| Διακύμανση | 2169,45 | 3024,03 | 32063,68 | 10195,44 | 4253,53 |
| Τυπική Απόκλιση | 46,58 | 54,99 | 179,06 | 100,97 | 65,22 |
| **ΑΝΑΜΟΝΗ ΑΝΑ ΟΧΗΜΑ** | | | | | |
| Συνολικός Χρόνος Αναμονής | 6788,06 | 6463,80 | 5453,06 | 6012,94 | 6426,36 |
| Μέση Τιμή | 617,10 | 587,62 | 495,73 | 546,63 | 584,21 |
| Διακύμανση | 11682,73 | 15496,05 | 45709,44 | 24890,66 | 12118,28 |
| Τυπική Απόκλιση | 108,09 | 124,48 | 213,80 | 157,77 | 110,08 |
| **ΣΥΓΚΡΙΣΕΙΣ ΕΠΙΠΛΕΟΝ ΔΕΙΚΤΩΝ** | | | | | |
| Απόσταση Ανά Κόμβο | 20,42 | 23,82 | 30,89 | 28,33 | 27,15 |
| Απόσταση Ανά  Ζήτηση | 1,39 | 1,69 | 2,07 | 2,65 | 1,44 |
| Χρόνος Αναμονής Ανά Κόμβο | 123,42 | 124,30 | 69,03 | 101,91 | 114,76 |
| Χρόνος Αναμονής  Ανά Ζήτηση | 8,39 | 8,84 | 4,63 | 9,53 | 6,10 |

132

*Πίνακας 30. Σύγκριση Αποτελεσμάτων Σεναρίου 11.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Προσέγγιση 1 | Προσέγγιση 2 | Προσέγγιση 3 | Προσέγγιση 4 | Προσέγγιση 5 |
| **ΠΟΣΟΣΤΟ ΕΞΥΠΗΡΕΤΗΣΗΣ ΚΟΜΒΩΝ ΜΕ ΤΗ ΧΡΗΣΗ ΤΩΝ ΑΡΧΙΚΩΝ ΟΧΗΜΑΤΩΝ** | | | | |
| 43% | 37% | 53% | 43% | 43% |
| **ΠΟΣΟΣΤΟ ΕΞΥΠΗΡΕΤΗΣΗΣ ΚΟΜΒΩΝ ΜΕ ΤΗΝ ΟΛΟΚΛΗΡΩΣΗ ΟΛΩΝ ΤΩΝ ΔΡΟΜΟΛΟΓΙΩΝ** | | | | |
| 55% | 52% | 79% | 59% | 56% |
| **ΚΑΤΑΝΟΜΗ ΦΟΡΤΙΟΥ ΣΤΟ ΣΥΝΟΛΟ ΤΩΝ ΟΧΗΜΑΤΩΝ** | | | | |
| +60,35% | +50,90% | +216,54% | +15,51% | Μικρότερη τυπική απόκλιση της τάξης των 23,74 μονάδων |
| **ΚΑΤΑΝΟΜΗ ΦΟΡΤΙΟΥ ΣΤΑ ΑΡΧΙΚΑ ΟΧΗΜΑΤΑ** | | | | |
| +111,97% | +90,36% | +130,48% | Μικρότερη τυπική απόκλιση της τάξης των 17,38 μονάδων | +43,90% |
| **ΚΑΤΑΝΟΜΗ ΦΟΡΤΙΟΥ ΣΤΑ ΕΠΙΠΡΟΣΘΕΤΑ ΟΧΗΜΑΤΑ** | | | | |
| +118,75% | Μικρότερη τυπική απόκλιση της τάξης των 22,63 μονάδων | +87,50% | +9,38% | +9,38% |
| **ΣΥΝΟΛΙΚΗ ΑΠΟΣΤΑΣΗ ΔΡΟΜΟΛΟΓΙΩΝ** | | | | |
| Βέλτιστη λύση1123,35 μονάδες | +10,25% | +117,24% | +48,78% | +35,33% |
| **ΚΑΤΑΝΟΜΗ ΑΠΟΣΤΑΣΗΣ ΔΡΟΜΟΛΟΓΙΩΝ ΣΤΟ ΣΥΝΟΛΟ ΤΩΝ ΟΧΗΜΑΤΩΝ** | | | | |
| Μικρότερη τυπική απόκλιση της τάξης των 46,58 μονάδων | +18,06% | +284,44% | +116,78% | +40,02% |
| **ΣΥΝΟΛΙΚΟΣ ΧΡΟΝΟΣ ΑΝΑΜΟΝΗΣ ΟΧΗΜΑΤΩΝ** | | | | |
| +24,48% | +18,54% | Βέλτιστη λύση με αναμονή 5453,06 μονάδες χρόνου | +10,27% | +17,85% |
| **ΚΑΤΑΝΟΜΗ ΧΡΟΝΟΥ ΑΝΑΜΟΝΗΣ ΣΤΟ ΣΥΝΟΛΟ ΤΩΝ ΟΧΗΜΑΤΩΝ** | | | | |
| Μικρότερη τυπική απόκλιση της τάξης των 108,09 μονάδων | +15,17% | +97,80% | +45,96% | +1,85% |
| **ΑΠΟΣΤΑΣΗ ΠΟΥ ΔΙΑΝΎΕΤΑΙ ΑΝΑ ΕΞΥΠΗΡΕΤΟΥΜΕΝΟ ΚΟΜΒΟ** | | | | |
| Βέλτιστη λύση 20,42 μονάδες | +16,61% | +51,24% | +38,69% | +32,91% |
| **ΧΡΟΝΟΣ ΑΝΑΜΟΝΗΣ ΑΝΑ ΕΞΥΠΗΡΕΤΟΥΜΕΝΟ ΚΟΜΒΟ** | | | | |
| +78,80% | +80,08% | Βέλτιστη λύση 69,03 μονάδες | +47,65% | +66,25% |

133

## Σενάριο 12

Η ομαδοποίηση των κόμβων σύμφωνα με το **Σενάριο 12**:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Σενάριο 12 | | | | | | | |
| Συστάδες | Κόμβοι | | | | | | |
| 1 | 0  51 | 85  19 | 63 | 89 | 64 | 76 | 84 |
| 2 | 0  38 | 100  6 | 61  2 | 88 | 55 | 68 | 70 |
| 3 | 0  97 | 52  87 | 75  11 | 58 | 74 | 57 | 59 |
| 4 | 0  9 | 73  16 | 53  17 | 12  15 | 13  78 | 47  14 | 10 |
| 5 | 0  25 | 77  49 | 48  24 | 21  23 | 20  86 | 18 | 22 |
| 6 | 0  43 | 72  40 | 41  36 | 37  39 | 44 | 35 | 42 |
| 7 | 0  65 | 90  69 | 98 | 83 | 66 | 99 | 82 |
| 8 | 0  46 | 3  79 | 60  5 | 45  7 | 8 | 1 | 4 |
| 9 | 0  26 | 28  50 | 33  31 | 29  34 | 30 | 27 | 32 |
| 10 | 0  54 | 62  56 | 93  80 | 95  91 | 81  92 | 67  96 | 71  94 |

Οι πίνακες αποτελεσμάτων των 5 αλγορίθμων:

134

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Σενάριο 12 | Προσέγγιση 1: Min Distance | | | | | Προσέγγιση 2: Inv. Dijkstra | | | | | Προσέγγιση 3: Min Wait Time | | | | |
| Time + Dist. | Wait Time | Demand | Nodes Served | Travel Dist. | Time  + Dist. | Wait Time | Demand | Nodes Served | Travel Dist. | Time  + Dist. | Wait Time | Demand | Nodes Served | Travel Dist. |
| Cluster 1 | 723 604,10  834,47 762,15  792 697,56  713 606,36  840,11 702,12  689 574,85  576 527,76  683 545,56  649 547  818 759,37 | | 81 | 5 118,9  4 72,313  4 94,432  5 106,64  5 137,98  5 114,14  3 48,231  7 137,44  4 102  3 58,621 | | 405 266,16  834,47 685,98  792 703,81  734,18 622,77  811 711,85  587,44 487,8  576 454,78  730,69 607,69  631 524,48  842,87 726,9 | | 95  102 | 5 138,84  6 148,49  2 88,189  5 111,41  3 99,146  4 99,644  6 121,22  5 123  4 106,52  5 115,97 | | 723 559,1  820 623,42  792 615,51  713 500,68  794 663,7  667 452,86  576 486,13  751,69 606,7  649 466,9  859,01 723,78 | | 101 | 6 163,9  7 196,58  5 176,49  8 212,32  4 130,3  9 214,14  4 89,872  5 144,99  8 182,1  6 135,22 | |
| Cluster 2 | 56  54 | 121 |
| Cluster 3 | 36 | 98 |
| Cluster 4 | 85 | 80 | 117 |
| Cluster 5 | 75  78 | 37 | 47 |
| Cluster 6 | 90  82 | 168 |
| Cluster 7 | 38 | 53 |
| Cluster 8 | 147 | 107 | 97 |
| Cluster 9 | 100 | 60 | 120 |
| Cluster 10 | 29 | 51 | 52 |
| Sum | 7317,58 6326,88 467 45 692,64 | | | | | 6944,6 5792,2 740 45 806,94 | | | | | 7344,7 5698,8 705 62 1183,6 | | | | |
| Extra Cluster 1 | 660 542,13 | | 113 | 6 117,87 | | 768,16 549,92 | | 152 | 8 218,24 | | 744,28 0 | | 259 | 14 744,28 | |
| General Sum | 7977,58 6869,01 580 51 810,51 | | | | | 7712,8 6342,1 892 53 1025,2 | | | | | 8089 5698,8 964 76 1927,9 | | | | |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Σενάριο 12 | Προσέγγιση 4: Min Served First | | | | | Προσέγγιση 5: Max Served First | | | | |
| Time + Dist. | Wait Time | Demand | Nodes Served | Travel Dist. | Time  + Dist. | Wait Time | Demand | Nodes Served | Travel Dist. |
| Cluster 1 | 723 561,09  820 700,51  792 714,63  713 580,72  794 663,70  588,82 689  576 477,42  757,99 582,42  649 499,11  842,86 751,38 | | 87 | 6 161,91  5 119,49  3 77,361  4 132,28  4 130,3  4 -  100,18  5 98,576  7 175,57  6 149,88  3 91,485 | | 723 624,94  834,47 698,17  792 692,51  734,18 579,91  840,11 634,47  689 555,34  576 528,75  719,93 573,01  631 490,55  873,01 704,63 | | 96 | 3 98,06  4 136,3  3 99,488  6 154,27  7 205,63  5 133,66  2 47,247  4 146,92  4 140,45  7 168,38 | |
| Cluster 2 | 72 | 86 |
| Cluster 3 | 31 | 95 |
| Cluster 4 | 40  47 | 127 |
| Cluster 5 | 155 |
| Cluster 6 | 58  55 | 130 |
| Cluster 7 | 43 |
| Cluster 8 | 117 | 107  100  113 |
| Cluster 9 | 80 |
| Cluster 10 | 19 |
| Sum | 7256,68 6220,03 507 47 619,72 | | | | | 7412,7 6082,3 732 45 874,66 | | | | |
| Extra Cluster 1 | 729 550,84 | | 58 | 6 178,16 | | 726,39 538,67 | | 78 | 5 187,71 | |
| General Sum | 7985,68 6770,87 565 53 797,88 | | | | | 8139,1 6621 810 50 1062,4 | | | | |

*Πίνακας 31. Σύγκριση των 5 Προσεγγίσεων που χρησιμοποιήθηκαν στο Σενάριο 12.*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ΥΠΟΛΟΓΙΣΜΟΙ ΜΕΤΑ ΤΗΝ ΟΛΟΚΛΗΡΩΣΗ ΤΩΝ ΔΡΟΜΟΛΟΓΙΩΝ** | | | | | |
| Παρατηρήσεις | Προσέγγιση 1 | Προσέγγιση 2 | Προσέγγιση 3 | Προσέγγιση 4 | Προσέγγιση 5 |
| **ΑΡΙΘΜΟΣ ΟΧΗΜΑΤΩΝ** | | | | | |
| Αρχικά Οχήματα | 10 | 10 | 10 | 10 | 10 |
| Επιπλέον Οχήματα | 1 | 1 | 1 | 1 | 1 |
| **ΑΡΙΘΜΟΣ ΚΟΜΒΩΝ ΠΟΥ ΕΞΥΠΗΡΕΤΗΘΗΚΑΝ** | | | | | |
| Αρχικοί Κόμβοι | 45 | 45 | 62 | 47 | 45 |
| Κόμβοι Επιπλέον Διαδρομών | 6 | 8 | 14 | 6 | 5 |
| Συνολικοί Κόμβοι | 51 | 53 | 76 | 53 | 50 |
| **ΖΗΤΗΣΗ ΑΝΑ ΟΧΗΜΑ** | | | | | |
| Συνολική Ζήτηση | 580 | 892 | 964 | 565 | 810 |
| Μέση Τιμή | 53 | 81 | 88 | 51 | 74 |
| Διακύμανση | 1159,76 | 1173,89 | 3676,29 | 764,45 | 876,02 |
| Τυπική Απόκλιση | 34,06 | 34,26 | 60,63 | 27,65 | 29,60 |
| Τυπική Απόκλιση Αρχικών Δρομολ. | 33,72 | 26,26 | 38,04 | 29,13 | 29,98 |
| Τυπική Απόκλιση Επιπλέον Δρομολ. | Δεν διατίθεται για μέγεθος πληθυσμού μικρότερο των 2 σημείων | | | | |
| **ΑΠΟΣΤΑΣΗ ΑΝΑ ΟΧΗΜΑ** | | | | | |
| Συνολική Απόσταση | 810,51 | 1025,17 | 1927,87 | 797,88 | 1062,37 |
| Μέση Τιμή | 73,68 | 93,20 | 175,26 | 72,53 | 96,58 |
| Διακύμανση | 895,21 | 1272,67 | 31969,85 | 5994,04 | 1965,45 |
| Τυπική Απόκλιση | 29,92 | 35,67 | 178,80 | 77,42 | 44,33 |
| **ΑΝΑΜΟΝΗ ΑΝΑ ΟΧΗΜΑ** | | | | | |
| Συνολικός Χρόνος Αναμονής | 6869,01 | 6342,14 | 5698,79 | 6770,87 | 6620,96 |
| Μέση Τιμή | 624,46 | 576,56 | 518,07 | 615,53 | 601,91 |
| Διακύμανση | 8009,52 | 19465,65 | 37024,93 | 8529,88 | 5485,51 |
| Τυπική Απόκλιση | 89,50 | 139,52 | 192,42 | 92,36 | 74,06 |
| **ΣΥΓΚΡΙΣΕΙΣ ΕΠΙΠΛΕΟΝ ΔΕΙΚΤΩΝ** | | | | | |
| Απόσταση Ανά Κόμβο | 15,89 | 19,34 | 25,37 | 15,05 | 21,25 |
| Απόσταση Ανά Ζήτηση | 1,40 | 1,15 | 2,00 | 1,41 | 1,31 |
| Χρόνος Αναμονής Ανά Κόμβο | 134,69 | 119,66 | 74,98 | 127,75 | 132,42 |
| Χρόνος Αναμονής Ανά Ζήτηση | 11,84 | 7,11 | 5,91 | 11,98 | 8,17 |

137

*Πίνακας 32. Σύγκριση Αποτελεσμάτων Σεναρίου 12.*

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Προσέγγιση 1 | Προσέγγιση 2 | Προσέγγιση 3 | Προσέγγιση 4 | Προσέγγιση 5 |
| **ΠΟΣΟΣΤΟ ΕΞΥΠΗΡΕΤΗΣΗΣ ΚΟΜΒΩΝ ΜΕ ΤΗ ΧΡΗΣΗ ΤΩΝ ΑΡΧΙΚΩΝ ΟΧΗΜΑΤΩΝ** | | | | |
| 45% | 45% | 62% | 47% | 45% |
| **ΠΟΣΟΣΤΟ ΕΞΥΠΗΡΕΤΗΣΗΣ ΚΟΜΒΩΝ ΜΕ ΤΗΝ ΟΛΟΚΛΗΡΩΣΗ ΟΛΩΝ ΤΩΝ ΔΡΟΜΟΛΟΓΙΩΝ** | | | | |
| 51% | 53% | 76% | 53% | 50% |
| **ΚΑΤΑΝΟΜΗ ΦΟΡΤΙΟΥ ΣΤΟ ΣΥΝΟΛΟ ΤΩΝ ΟΧΗΜΑΤΩΝ** | | | | |
| +23,17% | +23,92% | +119,30% | Μικρότερη τυπική απόκλιση της τάξης των 27,65 μονάδων | +7,05% |
| **ΚΑΤΑΝΟΜΗ ΦΟΡΤΙΟΥ ΣΤΑ ΑΡΧΙΚΑ ΟΧΗΜΑΤΑ** | | | | |
| +28,41% | Μικρότερη τυπική απόκλιση της τάξης των 26,26 μονάδων | +44,83% | +10,92% | +14,14% |
| **ΣΥΝΟΛΙΚΗ ΑΠΟΣΤΑΣΗ ΔΡΟΜΟΛΟΓΙΩΝ** | | | | |
| +1,58% | +28,49% | +141,62% | Βέλτιστη λύση 797,88 μονάδες | +33,15% |
| **ΚΑΤΑΝΟΜΗ ΑΠΟΣΤΑΣΗΣ ΔΡΟΜΟΛΟΓΙΩΝ ΣΤΟ ΣΥΝΟΛΟ ΤΩΝ ΟΧΗΜΑΤΩΝ** | | | | |
| Μικρότερη τυπική απόκλιση της τάξης των 29,92 μονάδων | +19,23% | +497,60% | +158,76% | +48,17% |
| **ΣΥΝΟΛΙΚΟΣ ΧΡΟΝΟΣ ΑΝΑΜΟΝΗΣ ΟΧΗΜΑΤΩΝ** | | | | |
| +20,53% | +11,29% | Βέλτιστη λύση με αναμονή 5698,79 μονάδες χρόνου | +18,81% | +16,18% |
| **ΚΑΤΑΝΟΜΗ ΧΡΟΝΟΥ ΑΝΑΜΟΝΗΣ ΣΤΟ ΣΥΝΟΛΟ ΤΩΝ ΟΧΗΜΑΤΩΝ** | | | | |
| +20,84% | +88,38% | +159,80% | +24,70% | Μικρότερη τυπική απόκλιση της τάξης των 74,06 μονάδων |
| **ΑΠΟΣΤΑΣΗ ΠΟΥ ΔΙΑΝΎΕΤΑΙ ΑΝΑ ΕΞΥΠΗΡΕΤΟΥΜΕΝΟ ΚΟΜΒΟ** | | | | |
| +5,57% | +28,49% | +68,50% | Βέλτιστη λύση 15,05 μονάδες | +41,14% |
| **ΧΡΟΝΟΣ ΑΝΑΜΟΝΗΣ ΑΝΑ ΕΞΥΠΗΡΕΤΟΥΜΕΝΟ ΚΟΜΒΟ** | | | | |
| +79,62% | +59,58% | Βέλτιστη λύση 74,98 μονάδες | +70,37% | +76,60% |

138

Κεφάλαιο 6

**Εφαρμογή στη Διαχείριση Απορριμμάτων**

Στο έκτο κεφάλαιο εξετάζεται η δυνατότητα εφαρμογής της προτεινόμενης μεθοδολογίας εύρεσης της αποδοτικότερης μεθόδου δρομολόγησης οχημάτων με ομοιόμορφη κατανομή των φορτίων και καλύπτοντας παράλληλα τα όποια χρονικά παράθυρα παρουσιάζονται. Η εν λόγω μέθοδος δοκιμάστηκε σε πραγματικές συνθήκες για την τροποποίηση των δρομολογίων απορριμματοφόρων οχημάτων που πραγματοποιούν διακομιδή απορριμμάτων στην περιοχή των Κουφαλίων του Δήμου Χαλκηδόνας, στην ευρύτερη περιοχή της Θεσσαλονίκης. Την υλοποίηση της διαδικασίας διακομιδής στη συγκεκριμένη περιοχή έχει αναλάβει ο όμιλος εταιρειών EDIL Hellas ΑΤΕΒΕ. Ο όμιλος ιδρύθηκε το 1985 και έχει έδρα τη Θεσσαλονίκη. Δραστηριοποιείται κυρίως στους τομείς των τεχνικών έργων και υπηρεσιών κοινής ωφέλειας προς οργανισμούς δημοσίων ή ιδιωτικών συμφερόντων, καθώς και φορέων της τοπικής αυτοδιοίκησης.

Κατά τη διάρκεια πρακτικής άσκησης και ενασχόλησης στο Τμήμα Ενεργειακών και Περιβαλλοντικών Έργων, παρατηρήθηκε πρόβλημα σχετικό με την ανομοιομορφία στα μέχρι τότε δρομολόγια των απορριμματοφόρων οχημάτων, το οποίο οδηγούσε στην περαιτέρω επιβάρυνση συγκεκριμένων οδηγών και οχημάτων για την κάλυψη των αναγκών των περιοχών που τους είχαν ανατεθεί, συγκριτικά με άλλα δρομολόγια τα οποία περατώνονταν με μεγαλύτερη ευκολία.

## ΑΝΑΛΥΣΗ ΣΕΝΑΡΙΟΥ ΔΡΟΜΟΛΟΓΗΣΗΣ ΑΠΟΡΡΙΜΜΑΤΟΦΟΡΩΝ ΟΧΗΜΑΤΩΝ

Πιο αναλυτικά όσον αφορά τη διαδικασία αποκομιδής στην εν λόγω περιοχή καθώς και τα γεωγραφικά κριτήρια:

* + - Υπάρχουν 4 περιοχές τις οποίες η εταιρεία καλείται να συλλέγει απορρίμματα 2 φορές κάθε εβδομάδα.
    - Το συνολικό επιτρεπόμενο βάρος ανά φορτηγό ανέρχεται στους 10 τόνους, Το μέγεθος αυτό μπορεί να αποκλίνει ανερχόμενο έως τους 12 τόνους, ανάλογα με τη συμπίεση που θα υποστούν τα απορρίμματα, καθώς και τη φύση τους.
    - Στην περιοχή υπάρχουν κάδοι χωρητικότητας 250 και 1.100 λίτρων. Αναγωγικά και σύμφωνα με δεδομένα της εταιρείας, κατά τους καλοκαιρινούς μήνες που διενεργήθηκε η παρούσα έρευνα, οι συγκεκριμένες τιμές αποκρίνονται περίπου σε 15 έως 20 κιλά απορριμμάτων στην περίπτωση των μικρών κάδων και 50 με 60 κιλών αντίστοιχα στην περίπτωση των κάδων χωρητικότητας 1.100 λίτρων.
    - Για την κάλυψη των παραπάνω αναγκών διατίθενται από την εταιρεία 6 δρομολόγια.

Η καταγραφή των γεωγραφικών στοιχείων της περιοχής έγινε αρχικά με τη χρήση των χαρτών της Google, και στη συνέχεια πραγματοποιήθηκαν επισκέψεις στην περιοχή όπου έγινε καταμέτρηση κάδων σε κάθε δρόμο, του τρόπου με τον οποίο πραγματοποιείται η διαδικασία της αποκομιδής, καθώς και της μορφολογίας του εδάφους που πιθανώς θα οδηγούσε σε καθυστερήσεις των δρομολογίων. Παρατηρήθηκαν τα παρακάτω:

* + - Όλα τα φορτηγά ξεκινούν άδεια από το αμαξοστάσιο, και πραγματοποιούν αποκομιδή μέχρι να καλύψουν τη ζήτηση των περιοχών που τους υποδεικνύονται.
    - Μετά την ολοκλήρωση της διαδρομής επιστρέφουν σε περιοχή κοντά στο αμαξοστάσιο όπου υπάρχει Σταθμός Μεταφόρτωσης Απορριμμάτων (ΣΜΑ), όπου και αδειάζουν τα απορρίμματα που συλλέχθηκαν και στη συνέχεια επιστρέφουν στο αμαξοστάσιο. Για ευκολότερη υλοποίηση προγραμματιστικά χρησιμοποιήθηκε η τοποθεσία του αμαξοστασίου (κόμβος 0) ως αρχικό και τελικό σημείο της διαδρομής, όπως ακριβώς δηλαδή και στην περίπτωση της αποθήκης των δοκιμαστικών σεναρίων στο Κεφάλαιο 5.
    - Σε κάθε δρόμο υπάρχουν κατά μέσο όρο:
      * 2 μεγάλοι κάδοι
      * 25 με 30 μικροί κάδοι
    - Οι περιοχές των οποίων πρέπει να καλυφθεί η ζήτηση είναι είτε ασφαλτοστρωμένες είτε η πρόσβαση γίνεται μέσω χωματόδρομου. Στην περίπτωση των χωματόδρομων χρησιμοποιήθηκε στη δοκιμαστική προσέγγιση

μεγαλύτερη τιμή όσον αφορά το χρόνο εξυπηρέτησης (Service Time), σε σχέση με τις περιοχές που είχαν άσφαλτο.

* + - Οι κεντρικές περιοχές πρέπει να καθαρίζονται νωρίς το πρωί, όπου τα απορριμματοφόρα κινούνται με μεγαλύτερη ευκολία καθώς δεν αντιμετωπίζουν καθυστερήσεις λόγω κίνησης.

Παρακάτω παρουσιάζεται χάρτης με την περιοχή, στον οποίο έχει προστεθεί αρίθμηση με την ακριβή τοποθεσία των κόμβων μέσω της εφαρμογής Easy ImageMap Generator [43], καθώς και ο Πίνακας με τα δεδομένα όλων των αναφερόμενων κόμβων.

Λόγω της διαφοροποίησης των παραγόμενων στοιχείων από την εφαρμογή σε σχέση με τα πραγματικά δεδομένα σχετικά με τις αποστάσεις μεταξύ των κόμβων έγιναν κάποιες παραδοχές ώστε το σενάριο για το οποίο επιδιώκεται η εύρεση μιας λύσης, να προσεγγίζει την πραγματική κατάσταση. Πιο αναλυτικά, η απόσταση μεταξύ των κόμβων υπολογίζεται με την Ευκλείδεια απόσταση. Υπολογίζοντας μέσω των παραγόμενων συντεταγμένων που προέκυψαν από τη χρήση της εφαρμογής την απόσταση μεταξύ των κόμβων, προέκυψε μία διαφορά της τάξης του 40 έως 50% σε σχέση με την πραγματικότητα. Για τον λόγο αυτό προσαρμόστηκαν και όλα τα υπόλοιπα δεδομένα του *Πίνακα 33* σε τιμές μικρότερες κατά το ίδιο ποσοστό ώστε όλα τα δεδομένα να προσεγγίζουν το πρόβλημα στην ίδια κλίμακα. Πιο αναλυτικά έγιναν οι παρακάτω παραδοχές:

* + - Δημιουργούνται 3 συστάδες, οι οποίες και εξυπηρετούνται 2 φορές την εβδομάδα (Δευτέρα – Τετάρτη, Πέμπτη – Σάββατο) ώστε να καλύψουν τις ανάγκες των τεσσάρων περιοχών που αναφέρθηκαν παραπάνω.
    - Για την ευκολότερη σύσταση ενός δείγματος με παρεμφερή στοιχεία με αυτά των δοκιμών που διεξήχθησαν στο Κεφάλαιο 5, τοποθετήθηκαν οι κόμβοι προς εξυπηρέτηση σε κάθε διασταύρωση των οδών που έπρεπε να συλλεχθούν απορρίμματα, και έγινε κατανομή της ζήτησης (demand) του κάθε δρόμου αναλογικά στους κόμβους των γύρω περιοχών. Λόγω της ύπαρξης λιγότερων κόμβων (διασταυρώσεων) σε σχέση με τον ολικό αριθμό κάδων σε κάθε οδό, υπολογίστηκε η μέση τιμή ζήτησης ανά οδό και χρησιμοποιήθηκε ως σταθερά σε κάθε κόμβο η τιμή 100 για να υποδειχθεί η ζήτηση σε μικρούς κάδους και αντίστοιχα η τιμή 300 για τους μεγάλους.
    - Ως χωρητικότητα του κάθε φορτηγού χρησιμοποιήθηκε η τιμή 10.000.
    - Ως χρόνος εξυπηρέτησης για τους μεγάλους κάδους ορίστηκε η τιμή 15 και για τους μικρούς η τιμή 16. Με την επαύξηση κατά 50% των τιμών αυτών ώστε να

προσεγγίζεται το πρόβλημα στην ίδια κλίμακα, γίνεται η αναγωγή στις τιμές 15 και 24, οι οποίες αναφερόμενες σε δευτερόλεπτα θεωρούνται εύλογες για να αντιπροσωπεύουν την διάρκεια εξυπηρέτησης κάθε κόμβου/αδειάσματος ενός κάδου. Για τις περιοχές σε χωματόδρομο χρησιμοποιήθηκαν οι τιμές 16 για τους μικρούς κάδους και αντίστοιχα 24 για τους μεγάλους οι οποίες αναγόμενες στην κλίμακα τιμών του προβλήματος έγιναν 24 και 36 δευτερόλεπτα αντίστοιχα.

* + - Ως προς τα χρονικά παράθυρα οι κόμβοι 1 έως και 13 που καλύπτουν την κεντρική οδό (Εθνικής Αντιστάσεως), είναι αυτοί που πρέπει να καλυφθούν νωρίτερα από τους υπόλοιπους, καθώς και οι δρόμοι βόρεια και νότια του συγκεκριμένου, συμβολιζόμενοι με τους αριθμούς 14 έως 25 και 84 έως 95.
    - Στα χρονικά παράθυρα υπάρχει περιορισμός μόνο ως προς τον αργότερο χρόνο τον οποίο κατά τον οποίο θα πρέπει να ικανοποιηθεί κάποιος κόμβος
    - Τέλος, ως προς τα χρονικά παράθυρα, χρησιμοποιώντας όπως αναφέρθηκε και νωρίτερα το δευτερόλεπτο ως σημείο αναφοράς για τις διαδρομές, και λαμβάνοντας υπόψη πως το ωράριο εργασίας, δηλαδή τις ώρες λειτουργίας κάθε οχήματος οι οποίες ανέρχονται στις 6,5 ώρες, προέκυψαν τα παρακάτω χρονικά διαστήματα εξυπηρέτησης:
      * Αποθήκη: Έναρξη 0 – Λήξη 60 × 60 × 6,5 = 23400 𝑠𝑒𝑐
      * Κεντρικοί κόμβοι: Έναρξη 0 – Λήξη 60 × 60 × 1,5 = 5400 𝑠𝑒𝑐
      * Δρόμοι βόρεια και νότια των κεντρικών: Έναρξη 0 – Λήξη 60 × 60 × 2,5 = 9000 𝑠𝑒𝑐
      * Λοιποί κόμβοι: Έναρξη 0 – Λήξη 60 × 60 × 6 = 21600𝑠𝑒𝑐

Ώρα εκκίνησης των δρομολογίων θεωρείται η χρονική στιγμή 0. Θεωρώντας πως οι κεντρικοί κόμβοι πρέπει να εξυπηρετηθούν εντός της πρώτης 1,5 ώρας του δρομολογίου, οι 2 δρόμοι βόρεια και νότια της «Εθνικής Αντιστάσεως» εντός των πρώτων 2,5 ωρών και οι υπόλοιποι κόμβοι μέσα στο 6ωρο ώστε να υπάρχει η δυνατότητα κάθε όχημα να επιστρέψει στο ΣΜΑ και στο αμαξοστάσιο εντός του ωραρίου της βάρδιας. Όπως και στην περίπτωση του χρόνου εξυπηρέτησης κάθε σημείου, έγινε αναγωγή των μονάδων σε μικρότερες τιμές κατά 40%. Πιο συγκεκριμένα παρουσιάζονται παρακάτω οι αλλαγές:

* + - * Αποθήκη: Έναρξη 0 – Λήξη 14040 𝑠𝑒𝑐
      * Κεντρικοί κόμβοι: Έναρξη 0 – Λήξη 3240 𝑠𝑒𝑐
      * Δρόμοι βόρεια και νότια των κεντρικών: Έναρξη 0 – Λήξη 5400 𝑠𝑒𝑐
      * Λοιποί κόμβοι: Έναρξη 0 – Λήξη 12960 𝑠𝑒𝑐

Στις επόμενες σελίδες παρουσιάζονται τα στοιχεία του εν λόγω πίνακα.

*Πίνακας 33. Χαρακτηριστικά Κόμβων - Κάδων προς Άδειασμα.*

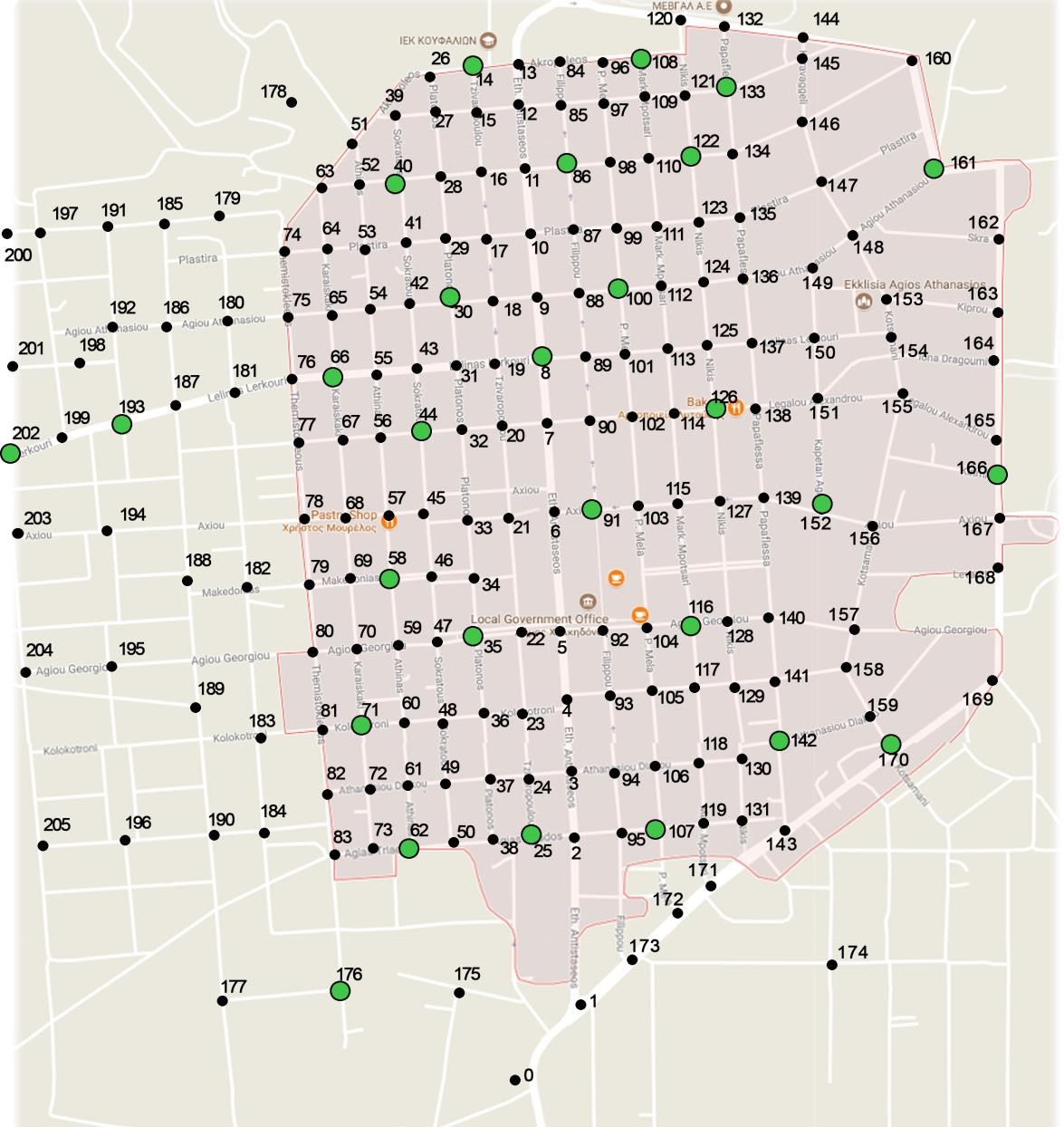
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Bin Number | X  Coord. | Y  Coord. | Demand | Ready Time (Early\_TW) | Due Time (Late\_TW) | Service Time |
| 0 | 515 | 542 | 0 | 0 | 14040 | 0 |
| 1 | 535 | 500 | 100 | 0 | 3240 | 10 |
| 2 | 534 | 418 | 100 | 0 | 3240 | 10 |
| 3 | 532 | 387 | 100 | 0 | 3240 | 10 |
| 4 | 529 | 348 | 100 | 0 | 3240 | 16 |
| 5 | 526 | 317 | 100 | 0 | 3240 | 10 |
| 6 | 523 | 262 | 100 | 0 | 3240 | 10 |
| 7 | 520 | 217 | 100 | 0 | 3240 | 10 |
| 8 | 518 | 187 | 300 | 0 | 3240 | 15 |
| 9 | 515 | 157 | 100 | 0 | 3240 | 10 |
| 10 | 512 | 127 | 100 | 0 | 3240 | 10 |
| 11 | 510 | 96 | 100 | 0 | 3240 | 10 |
| 12 | 507 | 65 | 100 | 0 | 3240 | 10 |
| 13 | 506 | 45 | 100 | 0 | 3240 | 10 |
| 14 | 485 | 48 | 300 | 0 | 5400 | 15 |
| 15 | 486 | 67 | 100 | 0 | 5400 | 10 |
| 16 | 489 | 98 | 100 | 0 | 5400 | 10 |
| 17 | 492 | 128 | 100 | 0 | 5400 | 10 |
| 18 | 494 | 159 | 100 | 0 | 5400 | 10 |
| 19 | 497 | 189 | 100 | 0 | 5400 | 10 |
| 20 | 499 | 219 | 300 | 0 | 5400 | 15 |
| 21 | 503 | 263 | 100 | 0 | 5400 | 10 |
| 22 | 507 | 319 | 100 | 0 | 5400 | 10 |
| 23 | 510 | 358 | 100 | 0 | 5400 | 10 |
| 24 | 512 | 388 | 100 | 0 | 5400 | 10 |
| 25 | 514 | 421 | 300 | 0 | 5400 | 15 |
| 26 | 466 | 50 | 100 | 0 | 12960 | 10 |
| 27 | 467 | 69 | 100 | 0 | 12960 | 10 |
| 28 | 470 | 100 | 100 | 0 | 12960 | 10 |
| 29 | 472 | 130 | 100 | 0 | 12960 | 10 |
| 30 | 475 | 162 | 300 | 0 | 12960 | 15 |
| 31 | 477 | 190 | 100 | 0 | 12960 | 10 |
| 32 | 480 | 221 | 100 | 0 | 12960 | 10 |
| 33 | 483 | 266 | 100 | 0 | 12960 | 10 |
| 34 | 484 | 295 | 100 | 0 | 12960 | 10 |
| 35 | 487 | 322 | 300 | 0 | 12960 | 15 |
| 36 | 490 | 361 | 100 | 0 | 12960 | 10 |
| 37 | 492 | 390 | 100 | 0 | 12960 | 10 |
| 38 | 496 | 420 | 100 | 0 | 12960 | 10 |
| 39 | 448 | 69 | 100 | 0 | 12960 | 10 |
| 40 | 451 | 101 | 300 | 0 | 12960 | 15 |
| 41 | 453 | 131 | 100 | 0 | 12960 | 10 |
| 42 | 456 | 163 | 100 | 0 | 12960 | 10 |
| 43 | 458 | 191 | 100 | 0 | 12960 | 10 |
| 44 | 460 | 222 | 300 | 0 | 12960 | 15 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 45 | 463 | 260 | 100 | 0 | 12960 | 10 |
| 46 | 466 | 296 | 100 | 0 | 12960 | 10 |
| 47 | 468 | 323 | 100 | 0 | 12960 | 10 |
| 48 | 470 | 361 | 100 | 0 | 12960 | 10 |
| 49 | 473 | 392 | 100 | 0 | 12960 | 10 |
| 50 | 475 | 423 | 100 | 0 | 12960 | 10 |
| 51 | 430 | 79 | 100 | 0 | 12960 | 10 |
| 52 | 431 | 103 | 100 | 0 | 12960 | 10 |
| 53 | 434 | 132 | 300 | 0 | 12960 | 15 |
| 54 | 436 | 163 | 100 | 0 | 12960 | 10 |
| 55 | 438 | 139 | 100 | 0 | 12960 | 10 |
| 56 | 441 | 224 | 100 | 0 | 12960 | 10 |
| 57 | 444 | 263 | 100 | 0 | 12960 | 10 |
| 58 | 446 | 294 | 300 | 0 | 12960 | 15 |
| 59 | 449 | 326 | 100 | 0 | 12960 | 10 |
| 60 | 452 | 363 | 100 | 0 | 12960 | 10 |
| 61 | 454 | 394 | 100 | 0 | 12960 | 10 |
| 62 | 457 | 424 | 300 | 0 | 12960 | 15 |
| 63 | 412 | 104 | 100 | 0 | 12960 | 10 |
| 64 | 414 | 134 | 100 | 0 | 12960 | 10 |
| 65 | 417 | 165 | 100 | 0 | 12960 | 10 |
| 66 | 419 | 196 | 300 | 0 | 12960 | 15 |
| 67 | 422 | 227 | 100 | 0 | 12960 | 10 |
| 68 | 425 | 263 | 100 | 0 | 12960 | 10 |
| 69 | 427 | 294 | 100 | 0 | 12960 | 10 |
| 70 | 430 | 327 | 100 | 0 | 12960 | 10 |
| 71 | 432 | 364 | 300 | 0 | 12960 | 15 |
| 72 | 435 | 394 | 100 | 0 | 12960 | 10 |
| 73 | 437 | 425 | 100 | 0 | 12960 | 10 |
| 74 | 396 | 135 | 300 | 0 | 12960 | 15 |
| 75 | 397 | 167 | 100 | 0 | 12960 | 10 |
| 76 | 399 | 197 | 100 | 0 | 12960 | 10 |
| 77 | 401 | 228 | 100 | 0 | 12960 | 10 |
| 78 | 406 | 267 | 300 | 0 | 12960 | 15 |
| 79 | 406 | 295 | 100 | 0 | 12960 | 10 |
| 80 | 410 | 329 | 100 | 0 | 12960 | 10 |
| 81 | 414 | 366 | 100 | 0 | 12960 | 10 |
| 82 | 416 | 395 | 100 | 0 | 12960 | 10 |
| 83 | 419 | 427 | 100 | 0 | 12960 | 10 |
| 84 | 526 | 45 | 100 | 0 | 5400 | 10 |
| 85 | 528 | 64 | 100 | 0 | 5400 | 10 |
| 86 | 530 | 94 | 300 | 0 | 5400 | 15 |
| 87 | 534 | 126 | 100 | 0 | 5400 | 10 |
| 88 | 535 | 156 | 100 | 0 | 5400 | 10 |
| 89 | 538 | 185 | 100 | 0 | 5400 | 10 |
| 90 | 541 | 216 | 100 | 0 | 5400 | 10 |
| 91 | 543 | 263 | 100 | 0 | 5400 | 15 |
| 92 | 547 | 315 | 300 | 0 | 5400 | 10 |
| 93 | 551 | 347 | 100 | 0 | 5400 | 10 |
| 94 | 553 | 387 | 100 | 0 | 5400 | 10 |
| 95 | 556 | 416 | 100 | 0 | 5400 | 10 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 96 | 546 | 43 | 100 | 0 | 12960 | 10 |
| 97 | 547 | 62 | 100 | 0 | 12960 | 10 |
| 98 | 550 | 92 | 100 | 0 | 12960 | 10 |
| 99 | 553 | 125 | 100 | 0 | 12960 | 10 |
| 100 | 555 | 155 | 300 | 0 | 12960 | 15 |
| 101 | 557 | 185 | 100 | 0 | 12960 | 10 |
| 102 | 560 | 215 | 100 | 0 | 12960 | 10 |
| 103 | 564 | 260 | 100 | 0 | 12960 | 10 |
| 104 | 568 | 317 | 100 | 0 | 12960 | 10 |
| 105 | 571 | 346 | 100 | 0 | 12960 | 10 |
| 106 | 573 | 384 | 100 | 0 | 12960 | 10 |
| 107 | 576 | 414 | 300 | 0 | 12960 | 15 |
| 108 | 565 | 42 | 300 | 0 | 12960 | 15 |
| 109 | 566 | 61 | 100 | 0 | 12960 | 10 |
| 110 | 569 | 90 | 100 | 0 | 12960 | 10 |
| 111 | 572 | 123 | 100 | 0 | 12960 | 10 |
| 112 | 575 | 153 | 100 | 0 | 12960 | 10 |
| 113 | 577 | 184 | 100 | 0 | 12960 | 10 |
| 114 | 580 | 213 | 100 | 0 | 12960 | 10 |
| 115 | 583 | 257 | 100 | 0 | 12960 | 10 |
| 116 | 588 | 314 | 300 | 0 | 12960 | 15 |
| 117 | 591 | 344 | 100 | 0 | 12960 | 10 |
| 118 | 594 | 382 | 100 | 0 | 12960 | 10 |
| 119 | 596 | 413 | 100 | 0 | 12960 | 10 |
| 120 | 587 | 26 | 100 | 0 | 12960 | 10 |
| 121 | 586 | 59 | 100 | 0 | 12960 | 10 |
| 122 | 588 | 88 | 300 | 0 | 12960 | 15 |
| 123 | 591 | 120 | 100 | 0 | 12960 | 10 |
| 124 | 594 | 150 | 100 | 0 | 12960 | 10 |
| 125 | 596 | 181 | 100 | 0 | 12960 | 10 |
| 126 | 599 | 212 | 300 | 0 | 12960 | 15 |
| 127 | 602 | 256 | 100 | 0 | 12960 | 10 |
| 128 | 606 | 314 | 100 | 0 | 12960 | 10 |
| 129 | 609 | 344 | 100 | 0 | 12960 | 10 |
| 130 | 614 | 382 | 100 | 0 | 12960 | 10 |
| 131 | 614 | 411 | 100 | 0 | 12960 | 10 |
| 132 | 604 | 29 | 100 | 0 | 12960 | 10 |
| 133 | 606 | 56 | 300 | 0 | 12960 | 15 |
| 134 | 609 | 88 | 100 | 0 | 12960 | 10 |
| 135 | 612 | 118 | 100 | 0 | 12960 | 10 |
| 136 | 614 | 147 | 100 | 0 | 12960 | 10 |
| 137 | 617 | 180 | 100 | 0 | 12960 | 10 |
| 138 | 619 | 209 | 100 | 0 | 12960 | 10 |
| 139 | 622 | 255 | 100 | 0 | 12960 | 10 |
| 140 | 626 | 312 | 100 | 0 | 12960 | 10 |
| 141 | 628 | 340 | 100 | 0 | 12960 | 10 |
| 142 | 631 | 372 | 300 | 0 | 12960 | 15 |
| 143 | 635 | 410 | 100 | 0 | 12960 | 10 |
| 144 | 642 | 33 | 100 | 0 | 12960 | 10 |
| 145 | 642 | 42 | 100 | 0 | 12960 | 10 |
| 146 | 642 | 72 | 100 | 0 | 12960 | 10 |

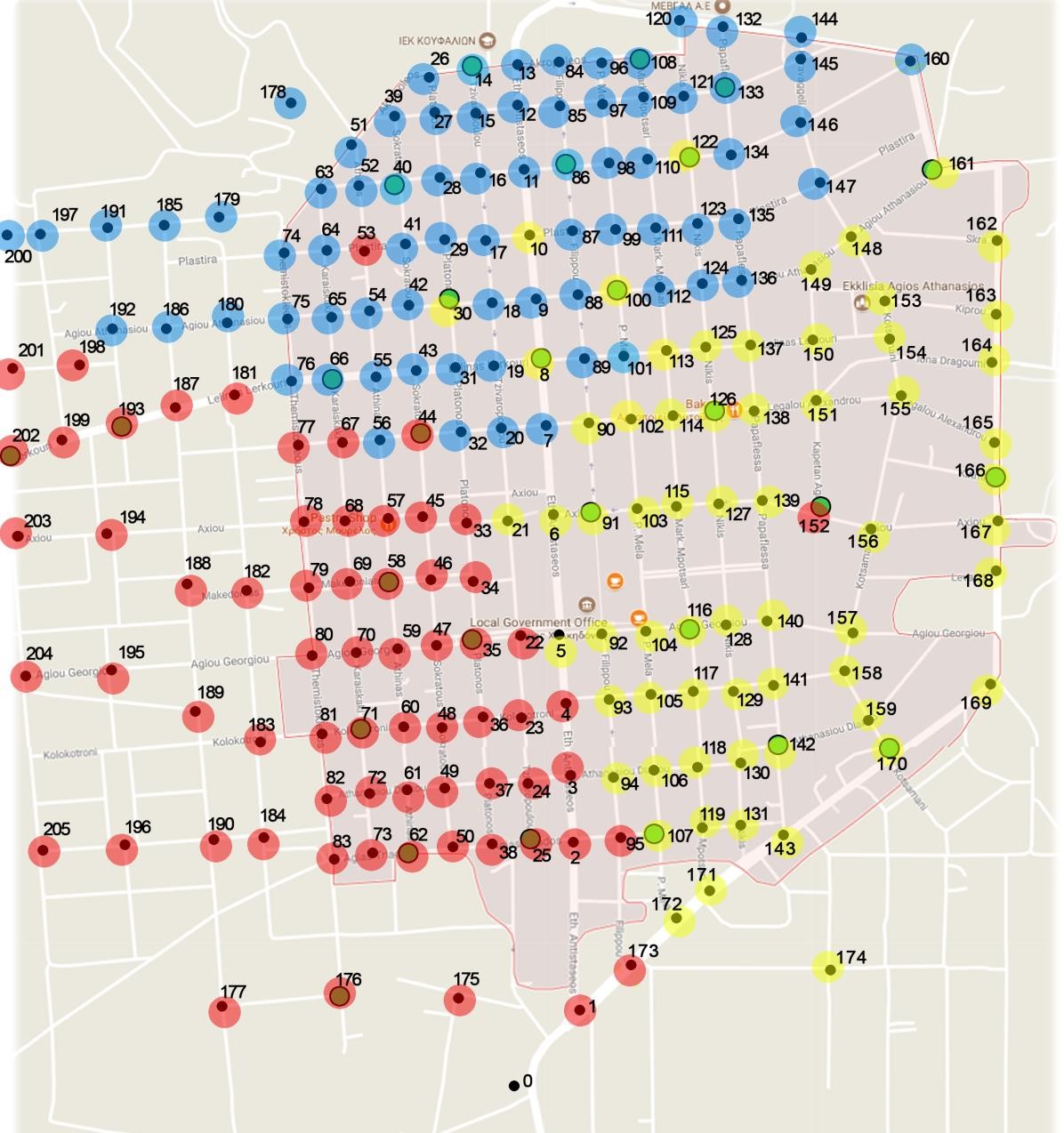
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 147 | 650 | 103 | 100 | 0 | 12960 | 10 |
| 148 | 665 | 129 | 100 | 0 | 12960 | 10 |
| 149 | 646 | 146 | 100 | 0 | 12960 | 10 |
| 150 | 648 | 177 | 100 | 0 | 12960 | 10 |
| 151 | 649 | 208 | 100 | 0 | 12960 | 10 |
| 152 | 650 | 262 | 300 | 0 | 12960 | 15 |
| 153 | 682 | 158 | 100 | 0 | 12960 | 10 |
| 154 | 684 | 175 | 100 | 0 | 12960 | 10 |
| 155 | 688 | 204 | 100 | 0 | 12960 | 10 |
| 156 | 676 | 269 | 100 | 0 | 12960 | 10 |
| 157 | 664 | 318 | 100 | 0 | 12960 | 10 |
| 158 | 660 | 336 | 100 | 0 | 12960 | 10 |
| 159 | 675 | 360 | 100 | 0 | 12960 | 10 |
| 160 | 694 | 43 | 100 | 0 | 12960 | 10 |
| 161 | 705 | 96 | 300 | 0 | 12960 | 15 |
| 162 | 734 | 131 | 100 | 0 | 12960 | 10 |
| 163 | 735 | 164 | 100 | 0 | 12960 | 10 |
| 164 | 735 | 190 | 100 | 0 | 12960 | 10 |
| 165 | 734 | 227 | 100 | 0 | 12960 | 10 |
| 166 | 735 | 245 | 300 | 0 | 12960 | 15 |
| 167 | 733 | 265 | 100 | 0 | 12960 | 10 |
| 168 | 734 | 289 | 100 | 0 | 12960 | 10 |
| 169 | 732 | 340 | 100 | 0 | 12960 | 10 |
| 170 | 685 | 378 | 300 | 0 | 12960 | 15 |
| 171 | 598 | 444 | 100 | 0 | 12960 | 10 |
| 172 | 586 | 454 | 100 | 0 | 12960 | 10 |
| 173 | 560 | 479 | 100 | 0 | 12960 | 10 |
| 174 | 656 | 479 | 100 | 0 | 12960 | 10 |
| 175 | 479 | 493 | 100 | 0 | 12960 | 16 |
| 176 | 423 | 492 | 100 | 0 | 12960 | 24 |
| 177 | 367 | 498 | 100 | 0 | 12960 | 16 |
| 178 | 400 | 79 | 100 | 0 | 12960 | 16 |
| 179 | 365 | 119 | 100 | 0 | 12960 | 16 |
| 180 | 368 | 168 | 100 | 0 | 12960 | 16 |
| 181 | 371 | 205 | 100 | 0 | 12960 | 16 |
| 182 | 378 | 299 | 100 | 0 | 12960 | 16 |
| 183 | 382 | 369 | 100 | 0 | 12960 | 16 |
| 184 | 385 | 415 | 100 | 0 | 12960 | 16 |
| 185 | 338 | 121 | 100 | 0 | 12960 | 16 |
| 186 | 341 | 170 | 100 | 0 | 12960 | 16 |
| 187 | 345 | 211 | 100 | 0 | 12960 | 16 |
| 188 | 350 | 301 | 100 | 0 | 12960 | 16 |
| 189 | 354 | 360 | 100 | 0 | 12960 | 16 |
| 190 | 359 | 417 | 100 | 0 | 12960 | 16 |
| 191 | 312 | 123 | 100 | 0 | 12960 | 16 |
| 192 | 314 | 172 | 100 | 0 | 12960 | 16 |
| 193 | 319 | 218 | 300 | 0 | 12960 | 24 |
| 194 | 315 | 271 | 100 | 0 | 12960 | 16 |
| 195 | 313 | 350 | 100 | 0 | 12960 | 16 |
| 196 | 319 | 420 | 100 | 0 | 12960 | 16 |
| 197 | 279 | 125 | 100 | 0 | 12960 | 16 |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 198 | 287 | 190 | 100 | 0 | 12960 | 16 |
| 199 | 293 | 225 | 100 | 0 | 12960 | 16 |
| 200 | 255 | 126 | 100 | 0 | 12960 | 16 |
| 201 | 260 | 192 | 100 | 0 | 12960 | 16 |
| 202 | 264 | 236 | 100 | 0 | 12960 | 24 |
| 203 | 266 | 272 | 100 | 0 | 12960 | 16 |
| 204 | 272 | 346 | 100 | 0 | 12960 | 16 |
| 205 | 279 | 423 | 100 | 0 | 12960 | 16 |



*Σχήμα 6-1. Οδικός χάρτης περιοχής Κουφαλίων. Με πράσινο χρώμα σημειώνονται οι κόμβοι με φορτίο 300 κιλών, ενώ με μαύρο αυτοί των 100 κιλών,*

## ΟΜΑΔΟΠΟΙΗΣΗ ΚΟΜΒΩΝ

Στην πρώτη φάση της επίλυσης του προβλήματος, έγινε εισαγωγή των νέων δεδομένων στον *Αλγόριθμο Improved k-means* για τη δημιουργία των απαιτούμενων ομάδων με κόμβους που πρέπει να εξυπηρετηθούν. Έπειτα από την εκτέλεσή του προέκυψαν οι παρακάτω ομάδες όπως φαίνονται σχηματισμένες και στο χάρτη της περιοχής.

*Σχήμα 6-2. Οι συστάδες μετά την χρήση του Αλγορίθμου Improved k-means.*

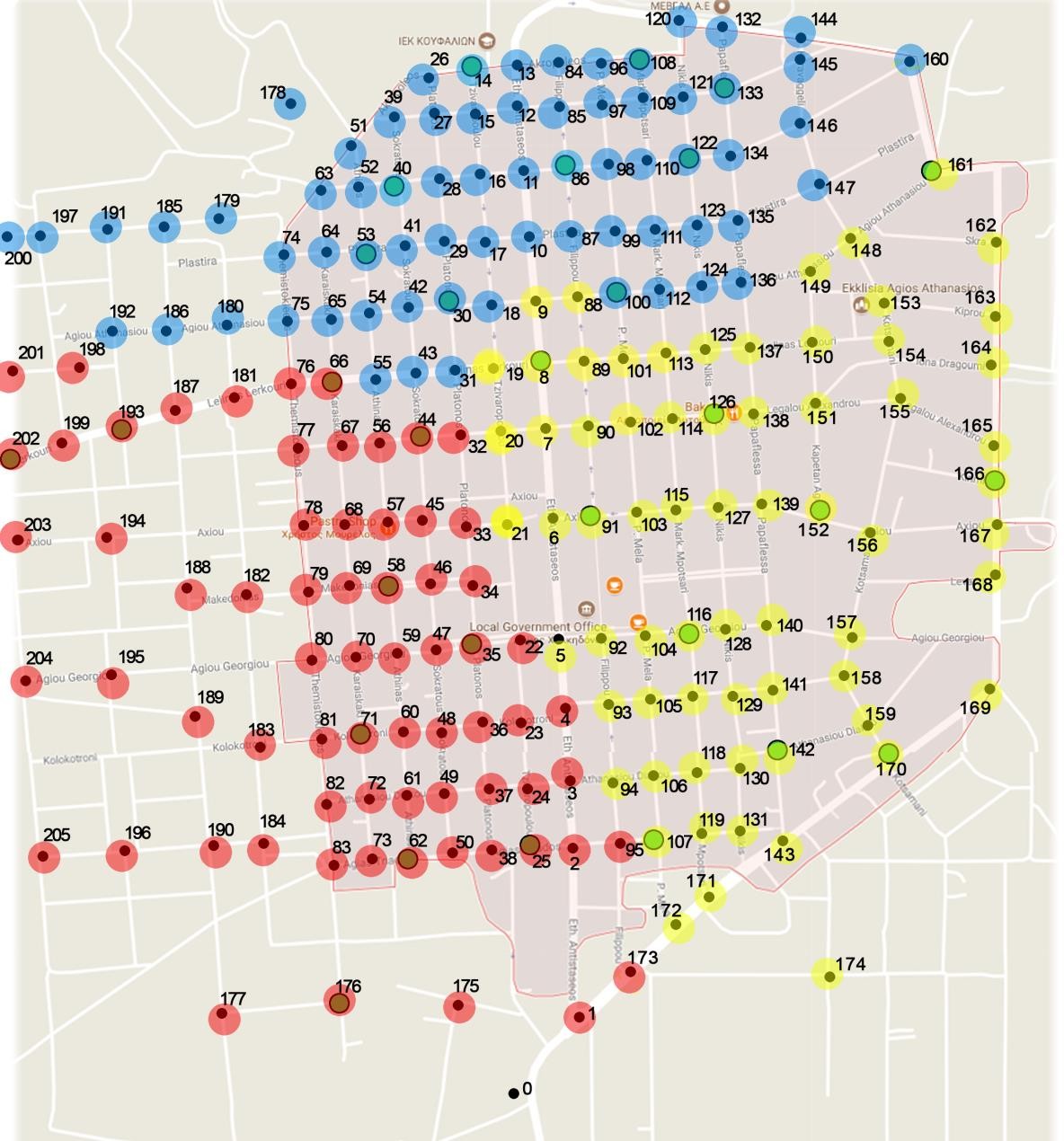
Όπως παρατηρείται, οι ομάδες που σχηματίστηκαν διαχωρίζονται με 3 χρώματα, κόκκινο, κίτρινο και μπλε όπως φαίνεται και στο *Σχήμα 6-2*. Παρατηρώντας το σχήμα, οι κόμβοι 8, 14, 30, 40, 53, 86, 122 και 152, δεν αντιστοιχισθεί στις ομάδες τους με τον καλύτερο δυνατό τρόπο. Ανατρέχοντας στο σύνολο δεδομένων του *Πίνακα 33*,

παρατηρείται πως πρόκειται για κόμβους με κάδους μεγάλης χωρητικότητας (300 κιλά). Αυτό συμβαίνει λόγω της απόστασής τους από το κέντρο κάθε ομάδας και της μεγάλης ζήτησης που καλύπτουν. Όταν υπολογίζεται η προτεραιότητα εξυπηρέτησης οι συγκεκριμένοι κόμβοι ανέρχονται σε υψηλές θέσεις της συγκεκριμένης λίστας και αντιστοιχίζονται πρώτοι σε ένα εκ των δρομολογίων έτσι ώστε να επιτευχθεί ο περιορισμός της ομοιόμορφης κατανομής του φορτίου σε όλα τα οχήματα.

Με τις παρούσες ομάδες τα δρομολόγια έχουν τα παρακάτω φορτία:

* + - Δρομολόγιο 1 (Κίτρινο): 8.800 κιλά
    - Δρομολόγιο 2 (Κόκκινο): 9.100 κιλά
    - Δρομολόγιο 3 (Μπλε): 9.000 κιλά

Στη συνέχεια γίνεται παρέμβαση στο σύνολο των αποτελεσμάτων, έτσι ώστε οι κίτρινοι και κόκκινοι κόμβοι που έχουν αντιστοιχισθεί με λανθασμένο τρόπο να αλλάξουν με κόμβους των άλλων ομάδων μη επηρεάζοντας την ομοιομορφία της κατανομής των φορτίων. Έτσι σχηματίζονται με μικρές αλλαγές νέες ομάδες, χωρίς να μεταβάλλεται σημαντικά το κέντρο τους και κυρίως χωρίς να μεταβάλλεται σημαντικά το φορτίο των οχημάτων. Τα αποτελέσματα της διαδικασίας φαίνονται στο επόμενο σχήμα.



*Σχήμα 6-3. Τελικές συστάδες μετά την παρέμβαση.*

## ΑΠΟΤΕΛΕΣΜΑΤΑ Σενάριο δρομολόγησης Απορριμματοφόρων οχημάτων

Στον επόμενο πίνακα παρουσιάζεται η ομαδοποίηση των κόμβων σύμφωνα με τις

αλλαγές που αναφέρθηκαν στην προηγούμενη παράγραφο.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Σενάριο Διαχείρισης Απορριμμάτων | | | | | | | |
| Συστάδες | Κόμβοι | | | | | | |
|  | 0 | 5 | 6 | 7 | 8 | 9 | 19 |
|  | 20 | 21 | 88 | 89 | 90 | 91 | 92 |
|  | 93 | 94 | 174 | 101 | 102 | 103 | 104 |
|  | 105 | 106 | 107 | 113 | 114 | 115 | 116 |
| 1 | 117  129 | 118  130 | 119  131 | 125  137 | 126  138 | 127  139 | 128  140 |
|  | 141 | 142 | 143 | 148 | 149 | 150 | 151 |
|  | 152 | 153 | 154 | 155 | 156 | 157 | 158 |
|  | 159 | 161 | 162 | 163 | 164 | 165 | 166 |
|  | 167 | 168 | 169 | 170 | 171 | 172 |  |
|  | 0 | 1 | 2 | 3 | 4 | 22 | 23 |
|  | 24 | 25 | 32 | 33 | 34 | 35 | 36 |
|  | 37 | 38 | 44 | 45 | 46 | 47 | 48 |
|  | 49 | 50 | 56 | 57 | 58 | 59 | 60 |
| 2 | 61  71 | 62  72 | 66  73 | 67  76 | 68  77 | 69  78 | 70  79 |
|  | 80 | 81 | 82 | 83 | 95 | 173 | 175 |
|  | 176 | 177 | 181 | 182 | 183 | 184 | 187 |
|  | 188 | 189 | 190 | 193 | 194 | 195 | 196 |
|  | 198 | 199 | 201 | 202 | 203 | 204 | 205 |
|  | 0 | 200 | 197 | 191 | 192 | 185 | 186 |
|  | 179 | 180 | 178 | 74 | 75 | 63 | 64 |
|  | 65 | 51 | 52 | 53 | 54 | 55 | 39 |
|  | 40 | 41 | 42 | 43 | 26 | 27 | 28 |
| 3 | 29  18 | 30  13 | 31  12 | 14  11 | 15  10 | 16  84 | 17  85 |
|  | 86 | 87 | 96 | 97 | 98 | 99 | 108 |
|  | 109 | 110 | 111 | 112 | 121 | 122 | 123 |
|  | 124 | 120 | 132 | 133 | 134 | 135 | 136 |
|  | 144 | 145 | 146 | 147 | 160 | 100 |  |

Οι πίνακες αποτελεσμάτων των 5 αλγορίθμων:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| EDIL  Case | Προσέγγιση 1 | | | | | Προσέγγιση | | | | 2 | | Προσέγγιση 3 | | | | |
| Time + Dist. | Wait Time | Demand | Nodes Served | Travel Dist. | Time + Dist. | Wait Time | Demand | Nodes Served | | Travel Dist. | Time + Dist. | Wait Time | Demand | Nodes Served | Travel Dist. |
| Cluster 1 | 4225,80  4597,13  4289,63 | 0  0  0 | 8800 | 66,00  68,00  67,00 | 4225,80  4597,13  4289,63 | 4402,29  4761,04  4644,95 | 0  0  0 | 8500 | 65,00 | | 4402,29 | 5398,40 | 0 | 7500 | 57,00 | 5398,40 |
| Cluster  2 | 9000 | 8800 | 66,00 | | 4761,04 | 6536,60 | 0 | 8200 | 62,00 | 6536,60 |
| Cluster  3 | 8700 | 8800 | 68,00 | | 4644,95 | 5588,71 | 0 | 8800 | 68,00 | 5588,71 |
| Sum | 13112,56 | 0 | 26500 | 201,00 | 13112,56 | 13808,28 | 0 | 26100 | | 199,00 | 13808,28 | 17523,71 | 0 | 24500 | 187,00 | 17523,71 |

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| EDIL  Case | Προσέγγιση 4 | | | | | Προσέγγιση 5 | | | | |
| Time + Dist. | Wait Time | Demand | Nodes Served | Travel Dist. | Time + Dist. | Wait Time | Demand | Nodes Served | Travel Dist. |
| Cluster | 6926,39  8131,83  6136,45 | 0  0  0 | 8100 | 65,00  67,00  66,00 | 6926,39  8131,83  6136,45 | 7582,91  8299,11  5989,27 | 0  0  0 | 9000 | 68  67  64 | 7582,91  8299,11  5989,27 |
| 1 |
| Cluster | 8700 | 8900 |
| 2 |
| Cluster | 8200 | 8400 |
| 3 |
| Sum | 21194,67 | 0 | 25000 | 198 | 21194,67 | 21871,29 | 0 | 26300 | 199 | 21871,29 |

152

*Πίνακας 34. Σύγκριση των 5 Προσεγγίσεων για την περίπτωση δρομολόγησης Απορριμματοφόρων Οχημάτων.*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **ΥΠΟΛΟΓΙΣΜΟΙ ΜΕΤΑ ΤΗΝ ΟΛΟΚΛΗΡΩΣΗ ΤΩΝ ΔΡΟΜΟΛΟΓΙΩΝ** | | | | | |
| Παρατηρήσεις | Προσέγγιση 1 | Προσέγγιση 2 | Προσέγγιση 3 | Προσέγγιση 4 | Προσέγγιση 5 |
| **ΑΡΙΘΜΟΣ ΟΧΗΜΑΤΩΝ** | | | | | |
| Οχήματα | 3 | 3 | 3 | 3 | 3 |
| **ΑΡΙΘΜΟΣ ΚΟΜΒΩΝ ΠΟΥ ΕΞΥΠΗΡΕΤΗΘΗΚΑΝ** | | | | | |
| Εξυπηρετούμενοι Κόμβοι | 201 | 199 | 187 | 198 | 199 |
| Κόμβοι Μη  Εξυπηρετούμενοι | 4 | 6 | 18 | 7 | 6 |
| **ΖΗΤΗΣΗ ΑΝΑ ΦΟΡΤΗΓΟ** | | | | | |
| Συνολική Ζήτηση | 26500 | 26100 | 24500 | 25000 | 26300 |
| Μέση Τιμή | 8833,33 | 8700,00 | 8166,67 | 8333,33 | 8766,67 |
| Διακύμανση | 23333,33 | 30000,00 | 423333,33 | 103333,33 | 103333,33 |
| Τυπική Απόκλιση | 152,75 | 173,21 | 650,64 | 321,46 | 321,46 |
| **ΑΠΟΣΤΑΣΗ ΑΝΑ ΦΟΡΤΗΓΟ** | | | | | |
| Συνολική Απόσταση | 13112,56 | 13808,28 | 17523,71 | 21194,67 | 21871,29 |
| Μέση Τιμή | 4370,85 | 4602,76 | 5841,24 | 7064,89 | 7290,43 |
| Διακύμανση | 39419,41 | 33510,39 | 371702,10 | 1009772,02 | 1397998,62 |
| Τυπική Απόκλιση | 198,54 | 183,06 | 609,67 | 1004,87 | 1182,37 |
| **ΣΥΓΚΡΙΣΕΙΣ ΕΠΙΠΛΕΟΝ ΔΕΙΚΤΩΝ** | | | | | |
| Απόσταση Ανά Κόμβο | 65,24 | 69,39 | 93,71 | 107,04 | 109,91 |
| Απόσταση Ανά Ζήτηση | 0,49 | 0,53 | 0,72 | 0,85 | 0,83 |

Συγκρίνοντας τα παραπάνω αποτελέσματα και τους δείκτες του *Πίνακα 34*, προκύπτει εύκολα στο συμπέρασμα πως η λύση που ενδείκνυται για την συγκεκριμένη περίπτωση σχετίζεται με την κάλυψη της ζήτησης σε κάθε περίπτωση των πλησιέστερων σημείων κάθε κόμβου. Είτε χρησιμοποιώντας την *Προσέγγιση 1* είτε την *Προσέγγιση 2*, παρατηρήθηκαν πως τα επίπεδα ομοιόμορφης κατανομής του φορτίου, καθώς και των αποστάσεων που διανύονται είναι σε πολύ καλύτερα επίπεδα από τις υπολειπόμενες 3 προσεγγίσεις.

Παρά την υλοποίηση του σεναρίου για το συγκεκριμένο σύνολο δεδομένων που ανταποκρίνεται σε μεγάλο βαθμό σε μία κατάσταση παρόμοια με την πραγματική, τα αποτελέσματα δεν μπορούν να θεωρηθούν πλήρως αξιόπιστα για τους παρακάτω λόγους:

* + - Ο υπολογισμός των αποστάσεων μεταξύ των κόμβων έγινε με τη χρήση του τύπου της Ευκλείδειας απόστασης. Έτσι οι τιμές που χρησιμοποιήθηκαν για τον υπολογισμό των αποστάσεων κατά τη διαδικασία της δρομολόγησης του εκάστοτε οχήματος απέχουν από τις πραγματικές.
    - Σε αντίθεση με το σενάριο που εξετάσθηκε παραπάνω, στην πραγματικότητα υπάρχουν μονόδρομοι που δεν επιτρέπουν την κυκλοφορία οχημάτων προς συγκεκριμένη κατεύθυνση, κάτι που δεν λαμβάνεται υπόψη στην επίλυση του προβλήματος και θα μπορούσε εν μέρει να αλλάξει τη σειρά επίσκεψης κάποιων κόμβων.
    - Λόγω της προσέγγισης της λύσης με αναγωγή σε μικρότερες ή μεγαλύτερες τιμές (στις περιπτώσεις των χρονικών παραθύρων και των φορτίων κάθε κόμβου) τα αποτελέσματα σε θέματα χρόνου και αποστάσεων διαφέρουν από τις πραγματικές συνθήκες.

Όμως, πάρα το γεγονός πως το δείγμα δεδομένων που χρησιμοποιήθηκε δεν ήταν πλήρως αντιπροσωπευτικό, το αποτέλεσμα του συγκεκριμένου μοντέλου θεωρείται αξιόπιστο και η συγκεκριμένη μέθοδος υπολογισμού του τρόπου διαχωρισμού των περιοχών θα συνέβαλλε θετικά στη διαδικασία διαχείρισης των δρομολογίων συλλογής απορριμμάτων. Μελλοντικά ως μέρος κάποιας μελέτης είτε από την εταιρεία, είτε από την τοπική κοινότητα, θα μπορούσαν να προστεθούν περισσότερα στοιχεία ώστε να πραγματοποιηθεί μια ακριβέστερη μελέτη όσον αφορά τον διαχωρισμό των περιοχών ανάλογα με τη ζήτησή τους, αλλά και με την υλοποίηση καλύτερων δρομολογίων κατά τη συλλογή απορριμμάτων. Με την προσθήκη εξειδικευμένων στοιχείων θα μπορούσαν ακόμη να γίνουν προβλέψεις για περιπτώσεις αυξημένης ζήτησης, όπως τα καλοκαίρια όπου το επίπεδο απορριμμάτων είναι μεγαλύτερο ή για την εξυπηρέτηση με τον καλύτερο τρόπο των αναγκών της περιοχής σε περιπτώσεις μεγαλύτερων αναγκών, όπως για παράδειγμα συγκεντρώσεων, εκδηλώσεων, λαϊκών αγορών κ.α..

Κεφάλαιο 7

**Συμπεράσματα και Μελλοντική Έρευνα**

Στην παρούσα διπλωματική εργασία μελετήθηκε μια σειρά περιπτώσεων που συμβάλλουν στην επίλυση προβλημάτων δρομολόγησης στόλου Οχημάτων, και πιο συγκεκριμένα με το πρόβλημα των Χρονικών Παραθύρων. Παρουσιάστηκε μια ευρετική προσέγγιση που έδωσε πολύ ικανοποιητικά αποτελέσματα, με αμελητέο χρόνο εκτέλεσης του κώδικα. Η αύξηση του μεγέθους του προβλήματος μπορεί να αντιμετωπιστεί εξίσου αποτελεσματικά από τους προτεινόμενους αλγορίθμους χωρίς ουσιαστικές αλλαγές στον κώδικα και με χρήση τεχνικών προγραμματισμού που έχουν ήδη χρησιμοποιηθεί.

Παρουσιάστηκε μία προσέγγιση που αποτελείται από δύο φάσεις. Στην Φάση Ι δημιουργούνται ισορροπημένες και συμπαγείς ομάδες κόμβων που εξυπηρετούνται ικανοποιώντας το κριτήριο της ομοιόμορφης κατανομής των φορτίων σε όλα τα οχήματα. Η Φάση ΙΙ αποτελεί τη φάση δημιουργίας των διαδρομών που πρέπει να ακολουθήσουν τα εν λόγω οχήματα σε κάθε περίπτωση. Υλοποιείται μια προσέγγιση που εξετάζει 5 διαφορετικές μεθόδους επίλυσης του προβλήματος, καθώς και δοκιμαστικός έλεγχος σε συνθετικά αλλά και πραγματικά σύνολα δεδομένων για κάθε περίπτωση. Οι προσεγγίσεις επίλυσης εστιάζουν σε διαφορετικά κριτήρια η καθεμία όπως για παράδειγμα ελαχιστοποίηση της απόστασης, ελαχιστοποίηση του χρόνου αναμονής σε κάθε επισκεπτόμενο σημείο και έμφαση σε εξυπηρέτηση συγκεκριμένων ομάδων πελατών του συνόλου.

Για τον έλεγχο της αποτελεσματικότητας των παραπάνω κριτηρίων, εξετάστηκαν 12 διαφορετικά σενάρια, εξήχθησαν τα ανάλογα αποτελέσματα για κάθε σύνολο δεδομένων, και ανάλογα με τις τιμές των δεικτών που περιέχονται στα αποτελέσματα κρίθηκε η καταλληλόλητα της χρήσης ή μη της κάθε μεθόδου, ανάλογα με τις απαιτήσεις του εκάστοτε συνόλου δεδομένων.

## 7.1 ΣΥΜΠΕΡΑΣΜΑΤΑ

Κατά τον έλεγχο της αποτελεσματικότητας της κάθε προσέγγισης προέκυψαν τα παρακάτω αποτελέσματα που σχετίζονται με τη χρήση της κάθε μεθόδου επίλυσης για την προσέγγιση βέλτιστης επίλυσης προβλημάτων με συγκεκριμένα χαρακτηριστικά. Στη συνέχεια ακολουθεί αναλυτικότερη περιγραφή του κάθε προβλήματος. Αξίζει να επισημανθεί πως τα παρακάτω συμπεράσματα σχετίζονται με την υλοποίηση της δεύτερης φάσης της προσέγγισης που εξετάζεται στην παρούσα εργασία, και αφού έχει ήδη πραγματοποιηθεί ομοιόμορφη κατανομή της ζήτησης σε όλες τις εξυπηρετούμενες συστάδες από την Φάση Ι. Η ύπαρξη διπλής γραμμής στους πίνακες δηλώνει την εναλλαγή μεταξύ των κατηγοριών των προβλημάτων, μικρό – μεσαίο – μεγάλο σύνολο δεδομένων.

## Εξυπηρέτηση Μέγιστου Αριθμού Κόμβων

Στον *Πίνακα 35* συγκρίνονται από τα αποτελέσματα του *Κεφαλαίου 5* οι βέλτιστες λύσεις, επικεντρώνοντας το ενδιαφέρον στην εξυπηρέτηση σε κάθε περίπτωση όσο το δυνατόν περισσότερων κόμβων-πελατών. Όπως παρατηρείται για μικρά σύνολα δεδομένων προτιμάται η *Προσέγγιση 1* ενώ για μεσαία και μεγάλα η *Προσέγγιση 3,* η οποία παρέχει εξυπηρέτηση από 76% έως και 100% των συνολικών κόμβων ανάλογα με τα διαθέσιμα οχήματα και τα χαρακτηριστικά των δεδομένων κάθε συνόλου.

*Πίνακας 35. Εξυπηρέτηση Μέγιστου Αριθμού Κόμβων.*

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Πελάτες | Αρχικά Οχήματα | Επιπλέον Οχήματα | Βέλτιστη Προσέγγιση | Κόμβοι Αρχικών Οχημάτων | %  Αρχικοί Κόμβοι | Κόμβοι Τελικών Οχημάτων | %  Τελικοί Κόμβοι | Σύνολο Κόμβων | %  Συνολικών Κόμβων |
| 25 | 1 | 3 | 1 | 10 | 40% | 15 | 60% | 25 | 100% |
| 25 | 2 | 3 | 3 | 12 | 48% | 13 | 52% | 25 | 100% |
| 25 | 3 | 2 | 1 | 17 | 68% | 8 | 32% | 25 | 100% |
| 25 | 4 | 1 | 1 | 16 | 64% | 7 | 28% | 23 | 92% |
| 50 | 3 | 4 | 4 | 23 | 46% | 26 | 52% | 49 | 98% |
| 50 | 4 | 3 | 3 | 26 | 52% | 19 | 38% | 45 | 90% |
| 50 | 5 | 2 | 3 | 27 | 54% | 18 | 36% | 45 | 90% |
| 50 | 6 | 1 | 3 | 32 | 64% | 11 | 22% | 43 | 86% |
| 100 | 7 | 4 | 3 | 54 | 54% | 44 | 44% | 98 | 98% |
| 100 | 8 | 3 | 3 | 54 | 54% | 37 | 37% | 91 | 91% |
| 100 | 9 | 2 | 3 | 53 | 53% | 26 | 26% | 79 | 79% |
| 100 | 10 | 1 | 3 | 62 | 62% | 14 | 14% | 76 | 76% |

## Εξυπηρέτηση Μέγιστης Ζήτησης

Στον *Πίνακα 36* συγκρίνονται οι βέλτιστες λύσεις από τα αποτελέσματα του *Κεφαλαίου 5*, επικεντρώνοντας το ενδιαφέρον στην εξυπηρέτηση σε κάθε περίπτωση των κόμβων που θα αποφέρουν τα μεγαλύτερα επίπεδα ικανοποίησης της ζήτησης (demand). Όπως παρατηρείται τόσο για τα μικρά όσο και για μεσαία και μεγάλα σύνολα δεδομένων με την *Προσέγγιση 3* επιτυγχάνονται τα καλύτερα αποτελέσματα. Παρέχεται εξυπηρέτηση από 56% έως και 100% της συνολικής ζήτησης ανάλογα με τα διαθέσιμα οχήματα και τα χαρακτηριστικά των δεδομένων κάθε συνόλου, με τις περιπτώσεις ύπαρξης περισσότερων αρχικών οχημάτων (1 ή 2 μόνο επιπρόσθετα) να σημειώνουν κατά μέσο όρο εξυπηρέτηση της τάξης του 94%.

## Καλύτερη Κατανομή του Φορτίου

Στον *Πίνακα 37* συγκρίνονται οι βέλτιστες λύσεις των αποτελεσμάτων του *Κεφαλαίου 5*, επικεντρώνοντας το ενδιαφέρον σε κάθε περίπτωση στην καλύτερη κατανομή φορτίου στα οχήματα του κάθε δρομολογίου. Όπως παρατηρείται τόσο για τα μικρά όσο και για μεσαία και μεγάλα σύνολα δεδομένων οι προσεγγίσεις των οποίων η λύση εστιάζει στην εξυπηρέτηση κόμβων ανάλογα με τη ζήτησή τους (4 και 5) είναι αυτές που αποφέρουν τα καλύτερα αποτελέσματα, με την *Προσέγγιση 5* να προτιμάται στο 67% του συνόλου των περιπτώσεων. Με ποσοστό 100% θα χαρακτηριζόταν η κατανομή ίδιου ακριβώς φορτίου σε όλα τα οχήματα. Με την *Προσέγγιση 5* παρατηρούνται επίπεδα κατανομής με ποσοστό κατά μέσο 74%, και οι μέγιστες τιμές του δείγματος να φτάνουν μέχρι και 84%. Ενώ αντίστοιχα οι τιμές κατανομής στα αρχικά οχήματα που δρομολογούνται με βάση την συσταδοποίηση της Φάσης Ι, παρατηρείται επίπεδο ομοιόμορφης κατανομής της τάξης του 94%, με μέσο όρο στο σύνολο των οχημάτων 82%.

*Πίνακας 36. Εξυπηρέτηση Μέγιστης Ζήτησης.*

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Πελάτες | Αρχικά Οχήματα | Επιπλέον Οχήματα | Συνολική Ζήτηση | Βέλτιστη Προσέγγιση | Κάλυψη Ζήτησης Αρχικά  Οχήματα | %  Ζήτηση Αρχικά  Οχήματα | Κάλυψη Ζήτησης Επιπλέον  Οχήματα | %  Ζήτηση Επιπλέον  Οχήματα | Τελική Κάλυψη Ζήτηση | %  Τελική Ζήτηση |
| 25 | 1 | 3 | 540 | 5 | 210 | 39% | 330 | 61% | 540 | 100% |
| 25 | 2 | 3 | 540 | 3 | 270 | 50% | 270 | 50% | 540 | 100% |
| 25 | 3 | 2 | 540 | 3 | 400 | 74% | 140 | 26% | 540 | 100% |
| 25 | 4 | 1 | 540 | 3 | 430 | 80% | 70 | 13% | 500 | 93% |
| 50 | 3 | 4 | 970 | 4 | 360 | 37% | 570 | 59% | 930 | 96% |
| 50 | 4 | 3 | 970 | 5 | 460 | 47% | 390 | 40% | 850 | 88% |
| 50 | 5 | 2 | 970 | 3 | 450 | 46% | 390 | 40% | 840 | 87% |
| 50 | 6 | 1 | 970 | 3 | 620 | 64% | 220 | 23% | 840 | 87% |
| 100 | 7 | 4 | 1724 | 3 | 831 | 48% | 849 | 49% | 1680 | 97% |
| 100 | 8 | 3 | 1724 | 3 | 742 | 43% | 736 | 43% | 1478 | 86% |
| 100 | 9 | 2 | 1724 | 3 | 673 | 39% | 504 | 29% | 1177 | 68% |
| 100 | 10 | 1 | 1724 | 3 | 705 | 41% | 259 | 15% | 964 | 56% |

*Πίνακας 37. Καλύτερη Κατανομή του Φορτίου.*

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Πελάτες | Αρχικά Οχήματα | Επιπλέον Οχήματα | Συνολική Ζήτηση | Βέλτιστη Προσέγγιση | Ζήτηση Βέλτιστης Προσέγγισης | %  Συνολικής ζήτησης | % Επίπεδο Κατανομής Αρχικά  Οχήματα | %Επίπεδο Κατανομής Τελικά  Οχήματα | % Επίπεδο Συνολικής Κατανομής |
| 25 | 1 | 3 | 540 | 5 | 540 | 100,00% | - | 38,58% | 46,65% |
| 25 | 2 | 3 | 540 | 2 | 510 | 94,44% | 65,34% | 68,48% | 69,47% |
| 25 | 3 | 2 | 540 | 5 | 470 | 87,04% | 93,86% | 69,91% | 83,87% |
| 25 | 4 | 1 | 540 | 3 | 500 | 92,59% | 90,43% | - | 81,29% |
| 50 | 3 | 4 | 970 | 4 | 930 | 95,88% | 92,47% | 71,17% | 73,67% |
| 50 | 4 | 3 | 970 | 5 | 870 | 89,69% | 84,09% | 90,71% | 82,09% |
| 50 | 5 | 2 | 970 | 5 | 790 | 81,44% | 82,96% | 56,14% | 76,16% |
| 50 | 6 | 1 | 970 | 5 | 720 | 74,23% | 75,09% | - | 74,45% |
| 100 | 7 | 4 | 1724 | 5 | 1431 | 83,00% | 70,08% | 79,65% | 73,86% |
| 100 | 8 | 3 | 1724 | 5 | 1283 | 74,42% | 76,36% | 78,93% | 77,88% |
| 100 | 9 | 2 | 1724 | 5 | 1054 | 61,14% | 73,90% | 74,17% | 75,22% |
| 100 | 10 | 1 | 1724 | 4 | 565 | 32,77% | 43,17% | - | 46,17% |

## Ελαχιστοποίηση της Απόστασης που Διανύει το Κάθε Όχημα

Στον *Πίνακα 38* συγκρίνονται οι βέλτιστες λύσεις από τα αποτελέσματα του *Κεφαλαίου 5*, με το ενδιαφέρον επικεντρωμένο στην ελαχιστοποίηση της συνολικής απόστασης που διανύεται από τα οχήματα μέχρι την ολοκλήρωση όλων των δρομολογίων. Παρατηρείται πως στο σύνολο των περιπτώσεων βέλτιστες είναι οι *Προσεγγίσεις 1 και 2*, για τα μικρά, τα μεσαία και τα μεγάλα σύνολα δεδομένων. Αναμενόμενα τα συγκεκριμένα αποτελέσματα καθώς η λύση των εν λόγω προσεγγίσεων βασίζεται σε αλγορίθμους ελαχιστοποίησης της απόστασης με διαφορετικά χαρακτηριστικά η καθεμία. Η *Προσέγγιση 1* προτιμάται στο 67% του συνόλου των περιπτώσεων ενώ η *Προσέγγιση 2* στο υπολειπόμενο 33%. Με ποσοστό 100% θα χαρακτηριζόταν η κατανομή ίδιας ακριβώς διανυόμενης απόστασης σε όλα τα οχήματα. Ακόμη, παρατηρείται ποσοστό κατανομής των αποστάσεων στο σύνολο των χρησιμοποιούμενων οχημάτων κοντά στο 65% στις περιπτώσεις της *Προσέγγισης 1* και κοντά στο 70% στις περιπτώσεις που προτιμάται λύση της *Προσέγγισης 2*.

*Πίνακας 38. Ελαχιστοποίηση της Απόστασης που Διανύει το κάθε Όχημα*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Πελάτες | Αρχικά Οχήματα | Επιπλέον Οχήματα | Βέλτιστη Προσέγγιση | Ελάχιστη Συνολική  Απόσταση | Μέσος Όρος  Απόστασης | % Επίπεδο Κατανομής  Απόστασης |
| 25 | 1 | 3 | 1 | 643,43 | 160,86 | 52,26% |
| 25 | 2 | 3 | 2 | 637,76 | 127,55 | 75,25% |
| 25 | 3 | 2 | 1 | 563,81 | 112,76 | 82,33% |
| 25 | 4 | 1 | 2 | 585,15 | 117,03 | 74,63% |
| 50 | 3 | 4 | 1 | 1383,43 | 197,63 | 71,41% |
| 50 | 4 | 3 | 1 | 942,96 | 134,71 | 79,36% |
| 50 | 5 | 2 | 2 | 958,31 | 136,90 | 59,20% |
| 50 | 6 | 1 | 2 | 880,99 | 125,86 | 60,72% |
| 100 | 7 | 4 | 1 | 1665,87 | 151,44 | 54,83% |
| 100 | 8 | 3 | 1 | 1379,77 | 125,43 | 50,24% |
| 100 | 9 | 2 | 1 | 1123,35 | 102,12 | 54,39% |
| 100 | 10 | 1 | 1 | 810,51 | 73,68 | 59,39% |

**Ελαχιστοποίηση Χρόνου Αναμονής**

Στον *Πίνακα 39* συγκρίνονται οι βέλτιστες λύσεις από των αποτελεσμάτων του *Κεφαλαίου 5* που επικεντρώνονται στην ελαχιστοποίηση του συνολικού χρόνου αναμονής που παρατηρείται κατά την εξυπηρέτηση των κόμβων από το σύνολο των

οχημάτων και μέχρι την ολοκλήρωση όλων των δρομολογίων. Παρατηρείται πως οι βέλτιστες λύσεις παράγονται από τις *Προσεγγίσεις 3 και 4*, με την 4 να υπερτερεί στα μικρά σύνολα δεδομένων, ενώ η 3 να υπερισχύει στα μεσαία και μεγάλα σύνολα. Αναμενόμενα και τα συγκεκριμένα αποτελέσματα σχετικά με την *Προσέγγιση 3* καθώς η υλοποίησή της βασίζεται στον *Αλγόριθμο 6,* ελαχιστοποίησης των χρόνων αναμονής. Η *Προσέγγιση 3* προτιμάται στο 75% του συνόλου των περιπτώσεων και το 88% των περιπτώσεων μεσαίου και μεγάλου πλήθους δεδομένων, τα χαρακτηριστικά των οποίων είναι παρόμοια με αυτά των προβλημάτων που παρουσιάζονται σε πραγματικές καταστάσεις. Τέλος, παρατηρείται πως η κατανομή των νεκρών χρόνων στο σύνολο των οχημάτων δεν είναι σε πολύ υψηλά επίπεδα, γεγονός που θα μπορούσε να εξεταστεί σε μελλοντικές έρευνες ώστε η δρομολόγηση των οχημάτων να γίνεται με τέτοιο τρόπο ώστε κάθε όχημα και το προσωπικό που απασχολεί να εργάζονται κατά μέσο όρο το ίδιο χρονικό διάστημα.

*Πίνακας 39. Ελαχιστοποίηση Χρόνου Αναμονής*

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Πελάτες | Αρχικά Οχήματα | Επιπλέον Οχήματα | Βέλτιστη Προσέγγιση | Ελάχιστη Συνολικός Χρόνος  Αναμονής | Μέσος Όρος Χρόνου  Αναμονής | % Επίπεδο Κατανομής Απόστασης |
| 25 | 1 | 4 | 3 | 1369,73 | 273,95 | 41,64% |
| 25 | 2 | 3 | 4 | 1756,56 | 351,31 | 26,26% |
| 25 | 3 | 2 | 3 | 2279,18 | 455,84 | 51,80% |
| 25 | 4 | 1 | 4 | 2425,05 | 485,01 | 56,73% |
| 50 | 3 | 4 | 4 | 2132,63 | 304,66 | 33,03% |
| 50 | 4 | 3 | 3 | 2437,42 | 348,20 | 28,10% |
| 50 | 5 | 2 | 3 | 3119,67 | 445,67 | 45,80% |
| 50 | 6 | 1 | 3 | 3054,78 | 436,40 | 45,07% |
| 100 | 7 | 4 | 3 | 4630,15 | 420,92 | 47,39% |
| 100 | 8 | 3 | 3 | 4978,73 | 452,61 | 53,40% |
| 100 | 9 | 2 | 3 | 5453,06 | 495,73 | 56,87% |
| 100 | 10 | 1 | 3 | 5698,79 | 518,07 | 62,86% |

## Χρόνος Εκτέλεσης του Αλγορίθμου

Σε κάθε σύνολο δεδομένων και ανάλογα με το μέγεθός του, σημειώθηκαν παρόμοιοι χρόνοι εκτέλεσης του αλγορίθμου. Αρχικά στο σύνολο δεδομένων με τις 25 κορυφές, όλες οι διαφορετικές λύσεις που προήλθαν από τη μεταβολή των χωρητικοτήτων των οχημάτων, τον ακριβή αριθμό αρχικών και επιπρόσθετων οχημάτων και την προσέγγιση που χρησιμοποιούνταν, έδωσαν χρόνο εκτέλεσης κατά μέσο όρο 7 με 8 δευτερολέπτων.

Στην περίπτωση των 50 κορυφών όπως και σε αυτήν των 100, σημειώθηκαν χρόνοι 9 με 10 δευτερολέπτων στην πρώτη περίπτωση και 14 με 15 δευτερόλεπτα αντίστοιχα στην δεύτερη. Έτσι προκύπτει το συμπέρασμα πως σε γραφήματα με σχετικά μικρό αριθμό κορυφών ο χρόνος εκτέλεσης του αλγορίθμου είναι αμελητέος και ανάλογα με την περίπτωση του προβλήματος κινείται μεταξύ των χρονικών περιθωρίων που αναφέρθηκαν.

Αξίζει να αναφερθεί πως οι συγκεκριμένοι χρόνοι αφορούν την εκτέλεση μόνο της Φάσης ΙΙ και μόνο μίας εκ των προσεγγίσεων. Για την ολοκλήρωση των ελέγχων σε κάθε σύνολο δεδομένων πρέπει να εκτελεστεί ο αλγόριθμος 5 φορές (μία για κάθε προσέγγιση), οπότε για να μπορέσουν να εξαχθούν τα δεδομένα κάθε σεναρίου χρειάζεται χρόνος από 1 έως 1,5 λεπτό. Αυτό το γεγονός οδηγεί στο συμπέρασμα ότι ο αλγόριθμος μπορεί να λειτουργήσει σε προβλήματα πραγματικού χρόνου, γεγονός το οποίο αργότερα επαληθεύτηκε και με την εφαρμογή του στην επίλυση του προβλήματος δρομολόγησης απορριμματοφόρων οχημάτων.

## Μεταβολή Πλήθους Φορτηγών (Αρχικά – Επιπρόσθετα)

Όσον αφορά τα αποτελέσματα που προέκυψαν από τη μεταβολή του πλήθους των οχημάτων θα παρουσιαστούν μια σειρά αποτελέσματα για τις συγκρίσεις που γίνονται στους *Πίνακες 35 έως 39*. Στην περίπτωση εξυπηρέτησης του μέγιστου αριθμού κόμβων- πελατών, στην περίπτωση εξυπηρέτησης της μέγιστης ζήτησης και στην καλύτερη κατανομή του φορτίου κάθε οχήματος παρατηρείται πως οι αλγόριθμοι των προσεγγίσεων έχουν μεγαλύτερη αποτελεσματικότητα όταν έχουμε έναν προκαθορισμένο αριθμό αρχικών οχημάτων. Όσο μειώνεται ο αριθμός των αρχικών οχημάτων και αυξάνονται τα επιπρόσθετα, τα οποία δεν έχουν ταξινομηθεί σύμφωνα με τα αποτελέσματα του *Αλγορίθμου 2*, Improved k-means, παρατηρείται σταδιακά μικρή μείωση της αποτελεσματικότητάς του. Αντίθετα για τις περιπτώσεις ελαχιστοποίησης της απόστασης και του χρόνου αναμονής φαίνεται πως αποφέρουν καλύτερα αποτελέσματα με τη χρήση λιγότερων αρχικών οχημάτων και περισσότερων επιπρόσθετων σε

«ελεύθερο ρόλο» καλύπτοντας μέρος του γενικού προβλήματος της επίτευξης μεγαλύτερων επιπέδων εξυπηρέτησης.

Στις επόμενες 3 ενότητες ακολουθούν συγκρίσεις σχετικά με τη φύση του κάθε προβλήματος βάσει του μεγέθους του, μικρό – μεσαίο – μεγάλο, όπου πραγματοποιείται αναλυτικότερη περιγραφή των όσων αναφέρθηκαν παραπάνω εστιάζοντας στο κάθε σύνολο δεδομένων.

## ΣΥΜΠΕΡΑΣΜΑΤΑ ΜΙΚΡΟΥ ΣΥΝΟΛΟΥ ΔΕΔΟΜΕΝΩΝ

Στο σύνολο των περιπτώσεων που αποτελούν το μικρό σύνολο δεδομένων παρατηρήθηκε πως η *Προσέγγιση 3* ήταν αυτή με τα μεγαλύτερα επίπεδα εξυπηρέτησης. Λόγω του μικρού μεγέθους του δείγματος, με τη χρήση των επιπρόσθετων οχημάτων τα επίπεδα εξυπηρέτησης σε όλες τις περιπτώσεις ήταν υψηλά, με το ποσοστό εξυπηρέτησης του συνόλου των κόμβων να αγγίζει το 94%, με την κατώτερη τιμή στο σύνολο των περιπτώσεων να είναι το 72%, και αντίστοιχα τη ανώτερη, δηλαδή η εξυπηρέτηση όλων των κόμβων (100%), εντοπίστηκε σε 10 δοκιμές. Τα αντίστοιχα ποσοστά εξυπηρέτησης μόνο με τη χρήση των αρχικών οχημάτων είναι: κατώτερο 12% του συνόλου των κόμβων, όταν διατίθεται μόνο ένα όχημα, ανώτερο 72% με τη χρήση τεσσάρων οχημάτων, και μέσο ποσοστού εξυπηρέτησης στο σύνολο των περιπτώσεων το 47%.

Ακόμη παρατηρήθηκαν τα καλύτερα επίπεδα ομοιόμορφης κατανομής του φορτίου κάθε οχήματος στις *Προσεγγίσεις 3 και 5*, ανάλογα με το σύνολο των αρχικά χρησιμοποιούμενων οχημάτων. Ενώ, εξετάζοντας ξεχωριστά τα αρχικά και τα πρόσθετα οχήματα, η *Προσέγγιση 3* υπερτερούσε όλων των προσεγγίσεων στην ομοιόμορφη κατανομή του φορτίου. Στην περίπτωση των επιπρόσθετων οχημάτων, καλύτερη κατανομή του φορτίου παρατηρήθηκε στις *Προσεγγίσεις 3 και 4*. Λόγω της φύσης των *Προσεγγίσεων 4 και 5*, ήταν και αναμενόμενα τα συγκεκριμένα συμπεράσματα αφού οι αυτές οι προσεγγίσεις κατηγοριοποιούν την προτεραιότητα εξυπηρέτησης στοχεύοντας στην εξυπηρέτηση κατά κύριο λόγω μικρών και αντίστοιχα μεγάλων φορτίων.

Όσον αφορά το κριτήριο της απόστασης, και πάλι όπως ήταν αναμενόμενο λόγω της φύσης των αλγορίθμων τους, οι *Προσεγγίσεις 1 και 2* ήταν αυτές που προσέφεραν αποτελέσματα στα οποία η απόσταση κυμαινόταν στα χαμηλότερα επίπεδα, συγκρίνοντάς τα με τις υπόλοιπες περιπτώσεις. Ακόμη, σε αυτές τις περιπτώσεις παρατηρήθηκαν και τα καλύτερα επίπεδα κατανομής της απόστασης σε όλα τα διαθέσιμα οχήματα.

Τέλος, όσον αφορά το χρόνο αναμονής μέχρι την έναρξη της εξυπηρέτησης σε κάποιο κόμβο, προτιμότερη προσέγγιση είναι η 3. Στην περίπτωση εφαρμογής της εν λόγω μεθόδου σε προβλήματα δρομολόγησης με κύριο μέλημα τη μείωση των νεκρών χρόνων, τα αποτελέσματα που προσφέρονται πέρα από την μεγάλη διαφορά που έχουν σε σχέση με τις υπόλοιπες προσεγγίσεις είναι και πολύ καλά κατανεμημένα, με τις άλλες μεθόδους να έχουν τιμές ακόμη και 88% μεγαλύτερες.

## ΣΥΜΠΕΡΑΣΜΑΤΑ ΜΕΣΑΙΟΥ ΣΥΝΟΛΟΥ ΔΕΔΟΜΕΝΩΝ

Στο μεσαίο σύνολο δεδομένων, όπως και στο μικρό, η *Προσέγγιση 3* ήταν αυτή με τα μεγαλύτερα επίπεδα εξυπηρέτησης, όμως παρατηρήθηκε άνοδος στα επίπεδα εξυπηρέτησης και της *Προσέγγισης 4* όπου τα ποσοστά της έφτασαν μέχρι και το 98%. Και πάλι έγινε χρήση επιπρόσθετων οχημάτων ώστε τα επίπεδα εξυπηρέτησης έπειτα από τη χρήση των αρχικών οχημάτων να αυξηθούν ακόμη περισσότερο και να εξυπηρετηθούν περισσότεροι κόμβοι. Κατά μέσο όρο μετά τη λήξη των δρομολογίων έχει δρομολογηθεί επίσκεψη σε ποσοστό κοντά στο 80%, κατώτερο επίπεδο εξυπηρέτησης 60% παρατηρήθηκε στην *Προσέγγιση 5*, ενώ το ανώτατο ποσοστό που σημειώθηκε ήταν της τάξης του 98% από την *Προσέγγιση 4*. Τα ποσοστά εξυπηρέτησης μόνο με τη χρήση των αρχικών οχημάτων είναι: κατώτερο 30% και πάλι από την *Προσέγγιση 5* όταν διατίθεται μόνο ένα όχημα, και ανώτερο 64% με τη χρήση των τεσσάρων αρχικών οχημάτων. Ο μέσος όρος του ποσοστού εξυπηρέτησης αυτών των περιπτώσεων ήταν 45%.

Τα επίπεδα ομοιόμορφης κατανομής του φορτίου κάθε οχήματος είχαν τις υψηλότερες τιμές στις *Προσεγγίσεις 4 και 5*, όπου η επιλογή των σημείων που πρόκειται να εξυπηρετηθούν γίνεται δίνοντας έμφαση στη ζήτηση κάθε κόμβου. Αναμένεται όσο αυξάνεται ο αριθμός των κόμβων που πρέπει να εξυπηρετηθούν, οι συγκεκριμένες προσεγγίσεις να βρίσκονται σε υψηλότερα επίπεδα προτίμησης σε σχέση με τις υπόλοιπες 3. Εξετάζοντας ξεχωριστά τα αρχικά και τα πρόσθετα οχήματα, ανάλογα με τα χαρακτηριστικά του κάθε προβλήματος, παρατηρήθηκε η προτίμηση των *Προσεγγίσεων 1 και 4* στο 57% των περιπτώσεων.

Όσον αφορά το κριτήριο της απόστασης, και πάλι όπως και στο μικρό σύνολο δεδομένων οι *Προσεγγίσεις 1 και 2* ήταν αυτές που προσέφεραν τα καλύτερα αποτελέσματα. Λόγω των αλγορίθμων ελαχιστοποίησης της απόστασης που υλοποιούνται στις συγκεκριμένες περιπτώσεις, η επιλογή των συγκεκριμένων μεθόδων κρίθηκε η πιο συμφέρουσα, ενώ με αυτές τις προσεγγίσεις παρατηρήθηκαν και τα υψηλότερα επίπεδα ομοιόμορφης κατανομής της απόστασης σε όλα τα οχήματα.

Τέλος, όσον αφορά το χρόνο αναμονής μέχρι την έναρξη της εξυπηρέτησης στον κάθε κόμβο, όπως και στο μικρό σύνολο δεδομένων, προτιμότερη είναι η *Προσέγγιση 3*.

## ΣΥΜΠΕΡΑΣΜΑΤΑ ΜΕΓΑΛΟΥ ΣΥΝΟΛΟΥ ΔΕΔΟΜΕΝΩΝ

Τελευταίο αλλά με τη μεγαλύτερη σημαντικότητα, καθώς λόγω της πληθώρας των σημείων που πρέπει να εξυπηρετηθούν το συγκεκριμένο σύνολο δεδομένων μπορεί να παρομοιαστεί με ένα πραγματικό πρόβλημα δρομολόγησης. Όπως και στα προηγούμενα σύνολα, η *Προσέγγιση 3* ήταν αυτή με τα μεγαλύτερα επίπεδα εξυπηρέτησης, και οι υπόλοιπες προσεγγίσεις στο σύνολό τους κινούνταν σε επίπεδα περίπου 20% χαμηλότερα. Και πάλι έγινε χρήση επιπρόσθετων οχημάτων ώστε τα επίπεδα εξυπηρέτησης να αυξηθούν ακόμη περισσότερο και να εξυπηρετηθούν περισσότεροι κόμβοι. Κατά μέσο όρο μετά τη λήξη των δρομολογίων έχει επιτευχθεί κάλυψη κοντά στο 70% στο σύνολο των προσεγγίσεων, με κατώτερο επίπεδο εξυπηρέτησης 50% που παρατηρήθηκε στην *Προσέγγιση 5*, και ανώτατο το ποσοστό της τάξης του 98% από την *Προσέγγιση 3*. Τα ποσοστά εξυπηρέτησης μόνο με την χρήση των αρχικών οχημάτων είναι: κατώτερο 36% από την *Προσέγγιση 2*, και ανώτερο 62% με την *Προσέγγιση 3* και τη χρήση των τεσσάρων αρχικών οχημάτων. Ο μέσος όρος του ποσοστού εξυπηρέτησης αυτών των περιπτώσεων είναι το 45%.

Τα επίπεδα ομοιόμορφης κατανομής του φορτίου κάθε οχήματος είχαν τις υψηλότερες τιμές στις *Προσεγγίσεις 4 και 5*, όπου η επιλογή των σημείων που πρόκειται να εξυπηρετηθούν γίνεται δίνοντας έμφαση στη ζήτηση κάθε κόμβου στο 73% των περιπτώσεων. Όπως ήταν αναμενόμενο, όσο αυξάνεται ο αριθμός των κόμβων που πρέπει να εξυπηρετηθούν, οι συγκεκριμένες προσεγγίσεις προτιμώνται σε περισσότερες περιπτώσεις συγκριτικά με τις υπόλοιπες. Εξετάζοντας ξεχωριστά τα αρχικά και τα επιπλέον οχήματα κάθε σεναρίου, η *Προσέγγιση 4* προτιμάται στο 75% των περιπτώσεων των αρχικών κόμβων, ενώ για τα επιπρόσθετα οχήματα δεν παρατηρείται κάποια προσέγγιση η οποία να ξεχωρίζει συγκριτικά με τις υπόλοιπες.

Σχετικά με τη συνολική απόσταση των δρομολογίων, η *Προσέγγιση 1* προσφέρει τα καλύτερα αποτελέσματα σε όλα τα πιθανά σενάρια, ενώ στην ίδια μέθοδο παρατηρήθηκαν και πάλι τα υψηλότερα επίπεδα ομοιόμορφης κατανομής της διανυόμενης απόστασης σε όλα τα οχήματα. Εύκολα προκύπτει το συμπέρασμα πως σε δίκτυα με μεγάλο αριθμό κόμβων, προσεγγίσεις των οποίων η δομή και οι αλγόριθμοι πλησιάζουν αυτή των Προσεγγίσεων 1 και 2, πρόκειται να παρέχουν καλύτερα αποτελέσματα σε περιπτώσεις ελαχιστοποίησης των διανυόμενων αποστάσεων.

Κλείνοντας, όσον αφορά την επίτευξη του μικρότερου χρόνου αναμονής σε κάθε κόμβο η Προσέγγιση 3 προτιμάται στο σύνολο των περιπτώσεων, όμως αξίζει να αναφερθεί πως

στα αποτελέσματα της *Προσέγγισης 1* παρατηρήθηκε η καλύτερη κατανομή του χρόνου αναμονής στα χρησιμοποιούμενα οχήματα.

## ΠΡΟΒΛΗΜΑΤΑ ΣΤΗΝ ΥΛΟΠΟΙΗΣΗ ΚΑΙ ΤΑ ΑΠΟΤΕΛΕΣΜΑΤΑ

Λόγω των πολλαπλών σεναρίων και προσεγγίσεων που υλοποιήθηκαν στην παρούσα διπλωματική εργασία, ένα από τα προβλήματα που αντιμετωπίστηκαν σχετιζόταν με την αξιολόγηση των αποτελεσμάτων, καθώς σε κάθε περίπτωση πρέπει να εξεταστούν παράλληλα διαφορετικές παραλλαγές του προβλήματος που η καθεμία εστιάζει σε διαφορετικά χαρακτηριστικά. Λόγω του συγκεκριμένου προβλήματος είναι δύσκολο να χαρακτηριστεί κάποια προσέγγιση ως βέλτιστη ή καλύτερη, γιατί πρέπει σε κάθε περίπτωση να καθοριστεί το πρόβλημα στο οποίο πρόκειται να εφαρμοστεί καθώς και τα αποτελέσματα τα οποία αναμένονται σε κάθε περίπτωση.

Ακόμη, λόγω των διαφορετικών προσεγγίσεων της λύσης στην παρούσα εργασία σε σχέση με τις υπάρχουσες βέλτιστες που παρουσιάζονται στον Πίνακα 5 της ενότητας 5.1, τα αποτελέσματα διαφορετικών μεθόδων δεν μπορούν να αποτελέσουν αξιόπιστο μέτρο σύγκρισης. Ο παράγοντας της ομοιομορφίας κατά την κατανομή του φορτίου σε όλες τις συστάδες, που πραγματοποιείται στην πρώτη φάση της επίλυσης επηρεάζει σε μεγάλο βαθμό τα τελικά αποτελέσματα και συμβάλλει στην προσέγγιση ένας φάσματος λύσεων τελείως διαφορετικού προβλήματος συγκριτικά με το κλασσικό πρόβλημα δρομολόγησης οχημάτων με χρονικά παράθυρα.

Τέλος αξίζει να γίνει αναφορά σε 2 προβλήματα που παρατηρήθηκαν κατά την υλοποίηση των 12 σεναρίων, τα οποία θα μπορούσαν να επηρεάσουν τον τρόπο προσέγγισης συγκεκριμένων προσεγγίσεων, ανάλογα με τη μορφή των δεδομένων που δίνονται στην παρούσα διπλωματική εργασία. Πιο αναλυτικά:

* Κατά την υλοποίηση του πρώτου αλγορίθμου ομαδοποίησης των κόμβων σε ομάδες, στα σύνολα δεδομένων των 50 και 100 κόμβων, παρατηρήθηκαν σε συγκεκριμένες περιπτώσεις συστάδες οι οποίες δεν συνέκλιναν σε συγκεκριμένες τιμές με αποτέλεσμα να γίνονται σε κάποιες περιπτώσεις επαναλήψεις και εναλλαγές των κέντρων μεταξύ 2 ίδιων τιμών. Οι συγκεκριμένες αλλαγές οδηγούσαν στην επαναλαμβανόμενη αλλαγή θέσης για συγκεκριμένους κόμβους μεταξύ δύο ομάδων. Γι’ αυτόν το λόγο στις συγκεκριμένες περιπτώσεις τέθηκε τερματική συνθήκη η οποία σχετιζόταν με την λήξη των υπολογισμών των στοιχείων κάθε συστάδας έπειτα από έναν συγκεκριμένο αριθμό επαναλήψεων.
* Ως προς την ποιότητα των λύσεων, λόγω της αυστηρότητας των χρονικών παραθύρων του δείγματος του Solomon, παρατηρήθηκε μεγάλη καθυστέρηση στην εξυπηρέτηση συγκεκριμένων κόμβων που βρισκόταν στην κρίσιμη διαδρομή. Αυτό είχε ως αποτέλεσμα το όχημα στο οποίο είχε ανατεθεί η εξυπηρέτηση του εν λόγω κόμβου να έχει μεγάλο διάστημα αναμονής, αναλογικά πάντα με το συνολικό χρόνο του δρομολογίου. Τέτοιου είδους καθυστέρηση μπορεί να οδηγήσει στη μη ικανοποίηση μέρους της ζήτησης ή των συνθηκών των υπολοίπων αλγορίθμων που χρησιμοποιούνται σε κάθε σενάριο.

## ΠΕΡΑΙΤΕΡΩ ΕΡΕΥΝΑ

Στη συνέχεια παρατίθενται ορισμένα σημεία που μπορούν να αποτελέσουν τις προϋποθέσεις για περαιτέρω έρευνα, ως συνέχιση της προτεινόμενης μεθοδολογίας:

## Παραλλαγές της υπάρχουσας Μεθόδου:

* Κατά την κατηγοριοποίηση των συστάδων στην Φάση Ι, να ενταχθεί και παράγοντας που θα ελέγχει και τα χρονικά παράθυρα ώστε οι τελικές συστάδες να είναι συμπαγείς και ως προς αυτό τον παράγοντα, και όχι μόνο ως προς την κατανομή του φορτίου και την τοποθεσία κάθε σημείου.
* Στην περίπτωση που απομείνουν κόμβοι που δεν έχουν εξυπηρετηθεί μετά τη χρήση των αρχικών οχημάτων, να πραγματοποιείται εκ νέου η δημιουργία συστάδων για τα επιπλέον οχήματα.
* Οι δύο φάσεις, δημιουργίας των συστάδων και η δρομολόγηση, να πραγματοποιούνται ενιαία σε ένα πρόγραμμα. Μετά τον καθορισμό της πρώτης συστάδας πραγματοποιείται η δρομολόγηση του πρώτου μόνο οχήματος. Σε περίπτωση ύπαρξης κόμβων που είχαν αντιστοιχισθεί στη συστάδα αλλά δεν μπόρεσαν να εξυπηρετηθούν τίθενται εκ νέου ως κόμβοι προς εξυπηρέτηση στη δεύτερη συστάδα. Ομοίως συνεχίζει η διαδικασία μέχρι να συμπληρωθεί ο αριθμός των διαθέσιμων οχημάτων.

## Χρήση Διαφορετικής Μεθόδου

* Παρατηρώντας την ανταπόκριση των προσεγγίσεων της παρούσας εργασίας γίνεται αντιληπτό πως στις περιπτώσεις χρήσης μεγαλύτερου αριθμού κόμβων προς εξυπηρέτηση παρατηρείται και αύξηση των εξυπηρετούμενων σημείων. Πιο συγκεκριμένα, μετά τη χρήση των επιπρόσθετων οχημάτων παρατηρήθηκε

εξυπηρέτηση περισσότερων κόμβων καθώς σε κάθε δρομολόγιο δεν υπήρχε ο περιορισμός εξυπηρέτησης μόνο σημείων εντός της κάθε συστάδας.

Κάποια παραλλαγή του συγκεκριμένου τρόπου δρομολόγησης που θα εφαρμοζόταν από την έναρξη της επίλυσης του προβλήματος θα μπορούσε να παρομοιαστεί με την μέθοδο Route First, Cluster Second. Όπως αναφέρθηκε και σε προηγούμενη ενότητα, με αυτή την τεχνική πρώτα επιλύεται το πρόβλημα του Περιοδεύοντος Πωλητή δημιουργώντας μία γιγαντιαία διαδρομή και στη συνέχεια, η γιγαντιαία διαδρομή κόβεται σε μικρότερες που ικανοποιούν τους περιορισμούς χωρητικότητας των οχημάτων. Η συγκεκριμένη μέθοδος θα έπρεπε να διερευνηθεί διεξοδικά και να συγκριθεί η αποδοτικότητά της, καθώς και τα αποτελέσματά της με αυτά της παρούσας εργασίας.

## Περιπτώσεις Δυναμικής Ζήτησης

Σε αυτό το σημείο, πρέπει να αναφερθεί ότι ένα πολύ σημαντικό χαρακτηριστικό των προβλημάτων δρομολόγησης οχημάτων είναι ο τρόπος με τον οποίο τα αιτήματα μεταφοράς γίνονται διαθέσιμα. Σε μία στατική κατάσταση, όλα τα αιτήματα γίνονται γνωστά την ώρα που οι διαδρομές πρέπει να κατασκευαστούν. Σε μια δυναμική κατάσταση, μερικά από τα αιτήματα γίνονται γνωστά την ώρα που οι διαδρομές πρέπει να κατασκευαστούν και τα υπόλοιπα αιτήματα διατίθενται σε πραγματικό χρόνο, δηλαδή κατά τη διάρκεια της εκτέλεσης των διαδρομών. Συνεπώς, σε μια δυναμική κατάσταση, όταν ένα νέο αίτημα μεταφορών γίνεται διαθέσιμο τουλάχιστον μια διαδρομή πρέπει να αλλάξει προκειμένου να εξυπηρετηθεί αυτό το νέο αίτημα.

Έτσι, ο αλγόριθμος μπορεί να ανταπεξέλθει σε μία δυναμική κατάσταση, αφενός φυσικά όταν το αίτημα μεταφοράς γνωστοποιηθεί εγκαίρως, όταν δηλαδή τα φορτηγά έχουν μόλις ξεκινήσει από την αρχική αποθήκη, και αφετέρου όταν η συνολική χωρητικότητα των φορτηγών οχημάτων που είναι καθοδόν είναι μεγαλύτερη ή ίση από το σύνολο των φορτίων που πρέπει να φορτωθούν / ξεφορτωθούν. Αν δεν πληρούνται οι δύο αυτές προϋποθέσεις, τότε ο υπεύθυνος δρομολόγησης των οχημάτων μπορεί να επιλέξει να στείλει ένα μόνο όχημα για το συγκεκριμένο αίτημα μεταφοράς με επιπλέον επιβάρυνση του πελάτη.

* Κατά τη διάρκεια της κάθε διαδρομής ανάλογα με την τοποθεσία και το χρονικό σημείο, εξετάζεται το ενδεχόμενο προσθήκης ενός ή περισσότερων σημείων προς εξυπηρέτηση.
* Κάθε σημείο του πρέπει να εξυπηρετηθεί απέχει κάθε στιγμή μια γνωστή απόσταση από το «στίγμα» όλων των φορτηγών τη δεδομένη χρονική στιγμή. Ορίζεται η εξυπηρέτηση των πρώτων σημείων σύμφωνα με διαδικασία αντίστοιχη της κατηγοριοποίησης του Αλγορίθμου 2 – Improved k-means και η εξυπηρέτηση κάθε επόμενου σημείου (καταχωρημένου ή νέου σημείου) θα επιλέγεται το κατάλληλο φορτηγό με κριτήρια:
  + Συμβατότητα χρονικού παραθύρου
  + Ελάχιστη Απόσταση
  + Διατήρηση ομογενούς κατανομής του φορτίου του

# Βιβλιογραφία

## Βιβλία

[26] Caric, T. και Gold, H. (2008). *Vehicle Routing Problem*. I-Tech Education and Publishing KG, Vienna, Austria

[16] Gawiejnowicz, S. (2008). *Time-Dependent Scheduling*. Springer.

[9] Tan, P., Steinbach, M. και Kumar, V. (2006) *Introduction to Data Mining*. Addison-Wesley Companion Book Site

[27] Toth, P. και Vigo, D. (2001). *The Vehicle Routing Problem*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001.

## Άρθρα

[20] Ahmadi, V., Eskandari, Z. και Yousefikhoshbakht, M. (2013). *Solving the Capacitated Clustering Problem by a Combined Meta-Heuristic Algorithm*. Journal of Advances in Computer Research, Vol. 4, No 1, 2013, 89-100

[22] Ahuja, R., Ergun, O., Orlin, J. και Punnen, A. (2002). *A survey of very large-scale neighborhood search techniques*. Discrete Applied Mathematics 123 (2002) 75 – 102

[39] Alexiou, D. και Katsavounis, S. (2012). Determining the Minimum Number of Warehouses and their Space-Size for Storing Compatible Items.

[19] Balinski, M. και Quandt, R. (1964). *On an Integer Program for a Delivery Problem*.

Opperations Research 12(2): 300-304

[41] Byung-In, K., Seongbae, K. και Surya, S. (2005). *Waste collection vehicle routing problem with time windows*. Computers & Operations Research 33 (2006) 3624–3642.

[21] Cakir, F., Street, N. και Thomas, B. (2015). *Revisiting Cluster First, Route Second for the Vehicle Routing Problem*. Department of Management Sciences, Trippie College of Business,

University of Iowa, Iowa City, Iowa, USA 52242

1. Cao, B. και Glover, F. (2010). *Creating Balanced and Connected Clusters to Improve Service Delivery Routes in Logistics Planning*. Systems Engineering Society of China & Springer- Verlag Berlin Heidelberg
2. Cook, W. και Rich, J. (1999). *A parallel cutting plane algorithm for the vehicle routing problem with time windows*. Working Paper, Computational and Applied Mathematics, Rice University, Houston, TX

[1] Dantzig, G. και Ramser, J. (2008). *The Truck Dispatching Problem*. Management Science

[37] Defryn, C. και Sorensen, K. (2016). *A fast two-level Variable Neighborhood Search for the Clustered Vehicle Routing Problem*. Computers and Operations Research

[36] El-Sherbeny, N. (2010). *Vehicle routing with time windows: An overview of exact, heuristic and metaheuristic methods*. Journal of King Saud University (Science) 22, 123–131

[31] Fisher, M. (1994) *Optimal Solution of Vehicle Routing Problems Using Minimum K-Trees*.

Operations Research 42(4):626-642

[2] Fisherm, M. και Jaikumar, R. (1981). *A Generalized Assignment Heuristtic for Vehicle Routing*. John Willey & Sons, Inc.

[32] Geetha, S., Poonthalir, G. και Vanathi, P. (2009). *Improved K-Means Algorithm for Capacitated Clustering Problem*. PSG College of Technology, Tamil Nadu, India

[7] Gendreau, M. και Tarantilis, C. (2010). *Solving Large-Scale Vehicle Routing Problems with Time Windows: The State-of-the-Art*. CIRRELT Internuniversity Research Centre on Enterprise Networks, Logistics and Transportation

[35] Haksever, C., Render, B., Russell, R. and Murdick, R. (2000). *Online Tutorial 5: Vehicle Routing and Scheduling.* Service Management and Operations, 2nd ed. Prentice Hall: Upper Saddle River, NJ, 476–497.

[5] Kallehauge, B., Larsen, J. και Madsen, O. (2000).*Lagrangean duality and non-differentiable optimization applied on routing with time windows - experimental results*. Department of Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark.

[28] Katsavounis, S., Chatzimichailidou, M., Makipaa, M. και Moyseos, M. (2013). *A Set Covering Heuristics for Logistics Centers*. Proceedings of International Conference for Entrepreneurship, Innovation and Regional Development

[34] Kwangeheol, S. και Sangyong, H. (2011). A Centroid-Based Heuristic Algorithm For The Capacitated Vehicle Routing Problem. Computing and Informatics, Vol. 30, 2011, 721-732

[18] Laporte, G., Nobert, Y. και Desrochers, M. (1985). Optimal Routing under Capacity and Distance Restrictions. Journal Operations Research, Vol. 33 Issue 5, 1985, 1050-1073

[6] Larsen, J. (1999). *Parallelization of the vehicle routing problem with time windows*. Department of Mathematical Modelling, Technical University of Denmark, Lyngby, Denmark

[30] Lau, H., Sim, M. και Teo, K. (2003). *Vehicle routing problem with time windows and a limited number of vehicles*. European Journal of Operational Research 148 (2003) 559–569

[42] Mirabi, M., Shokri, N. και Sadeghieh, A. (2016). *Modeling and Solving the Multi-Depot Vehicle Routing Problem with Time Window by Considering the Flexible End Depot in Each Route*. International Journal of Supply and Operations Management, Vol 3, Issue 3, 1373-1390

[8] Ochoa-Ortiz, A. και Ornels-Zapata, F. (2015). *Capacitated vehicle routing problem for PSS uses based on ubiquitous computing*. An emerging markets approach. DYNA, Vol. 82, Nο 191, p. 20-26

[29] Tan, K., Lee, L., Zhu, Q. και Ou, K. (2001). *Heuristic methods for vehicle routing problem with time windows*. Artificial Intelligence in Engineering 15, 2001, 281-295

## Internet

[23] Gehring & Homberger benchmark (2008). *Gehring & Homberger's extended VRPTW benchmark.* <https://www.sintef.no/projectweb/top/vrptw/homberger-benchmark/>(10/5/2017)

[40] Huntley, A. (2012) *C++ Tutorials and common material found in computer science curricula around the world.* [http://www.programminghelp.org/lib/c++/](http://www.programminghelp.org/lib/c%2B%2B/) (25/2/2017)

[43] Muller, D. (2017) Easy Imagemap Generator for html image mapping. [http://imagemap- generator.dariodomi.de/](http://imagemap-generator.dariodomi.de/) (6/7/2017)

[13] Simonson, S. (2013). *Introduction to Algorithms.* Cormen, Rivest, Leiserson. <http://aduni.org/courses/algorithms/>(30/8/2017)

1. Solomon benchmark (2008). *Solomon's VRPTW benchmark problems from 1987*. <https://www.sintef.no/projectweb/top/vrptw/solomon-benchmark/>(10/5/2017)
2. Solomon, M. (2005). *VRPTW Benchmark Problems.*

[http://w.cba.neu.edu/~msolomon/problems.htm](http://w.cba.neu.edu/%7Emsolomon/problems.htm) (20/4/2017)

[10] Solomon, M. (2005). *Optimal Solutions for RC1 and RC2 problems*. [http://w.cba.neu.edu/~msolomon/rc12solu.htm](http://w.cba.neu.edu/%7Emsolomon/rc12solu.htm) (20/4/2017)

[15] Wikipedia (2017). NP-hardness. <https://en.wikipedia.org/wiki/NP-hardness>(30/08/2017)

## Εργασίες

[14] Barman, S. (2014). Modeling and solving vehicle routing problems with many available vehicle types. Master’s Thesis, University of Gothenburg, Chalmers University of Technology, Division of Mathematics, Department of Mathematical Sciences.

[41] Gintaras, V. (2014) *Genetic Algorithm for Vehicle Routing Problem*. Doctoral Dissertation,

Vilnius University, Technological Sciences, Informatics Engineering

[17] Γκορτσίλας, Δ. (2014). *Νέες Ευρετικές Προσεγγίσεις για Δρομολόγηση Στόλου Οχημάτων*. Μεταπτυχιακή Διπλωματική Εργασία, Πανεπιστήμιο Πατρών, Πολυτεχνική Σχολή, Τμήμα Μηχανικών Ηλεκτρονικών Υπολογιστών και Πληροφορικής.

[12] Καραλιά, Α. (2014) *Χωροθέτηση κέντρων εξυπηρέτησης και Συσταδοποίηση κέντρων εφοδιασμού σε εφοδιαστικές αλυσίδες*. Διπλωματική Εργασία, Δημοκρίτειο Πανεπιστήμιο Θράκης, Πολυτεχνική Σχολή, Τμήμα Μηχανικών Παραγωγής και Διοίκησης.

[11] Καρτσοβίτη, Σ. (2009). *Αλγοριθμική Προσέγγιση του Προβλήματος Δρομολόγησης Οχημάτων με Περιορισμούς Συμβατότητας & Προτεραιότητας*. Διπλωματική Εργασία, Δημοκρίτειο Πανεπιστήμιο Θράκης, Πολυτεχνική Σχολή, Τμήμα Μηχανικών Παραγωγής και Διοίκησης.

[38] Τσούρος, Α. (2010) *Βελτιστοποίηση του Χρόνου Εξυπηρέτησης Σημείων Ζήτησης και Φορτίου Οχημάτων Διανομής/Συλλογής σε Αστικό Δίκτυο*. Διδακτορική Διατριβή, Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης, Πολυτεχνική Σχολή, Τμήμα Πολιτικών Μηχανικών.

# Παράρτημα

**// PHASE ONE - CLUSTERING**

#include <iostream> #include <fstream> #include <string> #include <math.h>

#include <stdio.h> /\* printf, scanf, puts, NULL \*/ #include <stdlib.h> /\* srand, rand \*/ #include <time.h> /\* time \*/

#include "NodesInfo.h" #include "VehicleInfo.h" #include <vector>

using namespace std;

static int node;

#define N 26 // number of nodes

#define Q 150 // capacity of each vehicle

#define k 4 // number of vehicles

#define O 5000 // maximum operation time per vehicle

#define C 50 // number of Clusters

int FIL[N][7]; // file array

static double D[N][N]; // distance between node i & j static double ShortDEMAND[N][2];

static double Centers[3][k]; static double New\_Centers[3][k]; static int validation = 0; static int NumberOfNodesOnRoute; static int LastNode;

static int non\_zero\_elements;

static double time\_distance; static int iteration = 2;

int i, j;

bool flag\_centers = false;

void fillVectorFile(vector<NodesInfo>&);

// Fills vector with node's information void printVector(const vector<NodesInfo>&);

// Prints the information of all nodes

void ValidationCheckOfDemand(const vector<NodesInfo>&);

// Checks if demand of each node, and overal demand exceeds our vehicles' capacity and prints messege to continue or not. If not terminates program

void CalculateAllNodesDistance(const vector<NodesInfo>&);

// Fills up D[N][N] array with the distances between nodes void fillVectorVehicle(vector<VehicleInfo>& newNodesVector); void printVectorVehicles(const vector<VehicleInfo>);

void ShortBasedOnDemand(vector<NodesInfo>);

void FirstClusteringIteration(vector<VehicleInfo>); void NewCenters(vector<VehicleInfo>);

void ShortBasedOnPriority(vector<NodesInfo>); void GetNewCluster(vector<VehicleInfo>);

vector <NodesInfo> MyNodesInformation; vector <VehicleInfo> MyVehicleInformation; static int No\_Vehicle;

int main()

{

fillVectorFile(MyNodesInformation);

// fills in vector "MyNodesInformation" from file R201a.txt printVector(MyNodesInformation);

// prints list of each node's information ValidationCheckOfDemand(MyNodesInformation);

// Validation check ;

if (validation > 0) // Terminating program

{

endl;

}

cout << endl << "Terminating program. Check inputs.." <<

int terminationintl; cin >> terminationintl; return 0;

CalculateAllNodesDistance(MyNodesInformation);

// fills in static array D[i][j] with distance between nodes fillVectorVehicle(MyVehicleInformation);

// Fills each Vehicle's Vector with information about, capacity and opperation time

printVectorVehicles(MyVehicleInformation);

// Prints list of each vehicle's information

ShortBasedOnDemand(MyNodesInformation);

// Shorts Nodes by Demand Rate (High to Low) and print results FirstClusteringIteration(MyVehicleInformation);

// Calculates and Prints Clusters after 1st iteration procedure

- setting by demand value NewCenters(MyVehicleInformation);

// Calculates possible changes in each vehicle's centers after clustering

bool flag11 = true;

for (int kkk = 0; kkk < C; kkk++)

{

No\_Vehicle = 0; do

{

flag\_centers = false;

if (Centers[1][No\_Vehicle] != New\_Centers[1][No\_Vehicle])

{

flag\_centers = true; No\_Vehicle++; break;

}

if (Centers[2][No\_Vehicle] != New\_Centers[2][No\_Vehicle])

{

flag\_centers = true; No\_Vehicle++; break;

}

No\_Vehicle++;

} while (flag\_centers == false);

// Checks if any change occurs on centers of each vehicle if (flag\_centers == true)

{

cout << endl;

for (int i = 0; i < k; i++)

{

Centers[1][i] = New\_Centers[1][i]; Centers[2][i] = New\_Centers[2][i];

}

int Overal\_Capacity\_Counter = 0; for (int i = 0; i < k; i++)

{

Overal\_Capacity\_Counter = Overal\_Capacity\_Counter + Q - MyVehicleInformation[i].get\_Capacity();

cout << endl << "Vehicle: " << i << " - Capacity: " << Q - MyVehicleInformation[i].get\_Capacity();

}

cout << endl << endl << "Overal Capacity: " << Overal\_Capacity\_Counter << endl << endl;

GetNewCluster(MyVehicleInformation);

}

}

int Overal\_Capacity\_Counter = 0; for (int i = 0; i < k; i++)

{

Overal\_Capacity\_Counter = Overal\_Capacity\_Counter + Q - MyVehicleInformation[i].get\_Capacity();

cout << endl << "Vehicle: " << i << " - Capacity: " << Q - MyVehicleInformation[i].get\_Capacity();

}

cout << endl << endl << "Overal Capacity: " << Overal\_Capacity\_Counter << endl << endl;

ofstream outFile; outFile.open("Test\_Text\_File.txt"); for (int i = 0; i < k; i++)

{

for (int j = 0; j < MyVehicleInformation[i].NodesOfCluster.size(); j++)

{

int test = MyVehicleInformation[i].NodesOfCluster.at(j);

outFile << test << " ";

}

outFile << endl;

}

outFile.close();

int a;

cout << "ENDING PROGRAM...";

cin >> a;

}

//-------------- E N D O F M A I N ---------------//

void GetNewCluster(vector<VehicleInfo>)

{

cout << endl << "Prioritizing Nodes(High to Low)..."; ShortBasedOnPriority(MyNodesInformation);

for (int i = 0; i < k; i++)

{

int size = MyVehicleInformation[i].NodesOfCluster.size(); for (int j = 0; j < size-1; j++)

{

MyVehicleInformation[i].EraseData();

}

}

for (int i = 0; i < k; i++)

{

MyVehicleInformation[i].AddNode(0);

// Sets starting point of each Vehicle Depot (0) MyVehicleInformation[i].set\_Capacity(Q);

}

bool flag;

double x1, x2, y1, y2;

for (int i = 0; i < N-1; i++)

{

flag = false;

double NextNode = ShortDEMAND[i][1];

x2 = MyNodesInformation[NextNode].get\_X(); y2 = MyNodesInformation[NextNode].get\_Y(); double Min\_Distance = 9999999.99;

int Min\_Node\_Vehicle = -1;

for (int j = 0; j < k; j++)

{

if (MyVehicleInformation[j].get\_Capacity() - MyNodesInformation[NextNode].get\_Demand() >= 0)

{

flag = true;

x1 = Centers[1][j]; y1 = Centers[2][j];

double dx = (x1 - x2)\*(x1 - x2);

// distance from center point j - X angle

double dy = (y1 - y2)\*(y1 - y2); // distance from center point j - Y angle

double Calculated\_Distance = sqrt(abs(dx +

dy));

if (Calculated\_Distance < Min\_Distance)

{

Min\_Distance = Calculated\_Distance; Min\_Node\_Vehicle = j;

} // end if

} // end if

} // end for

if (flag = true)

{

MyVehicleInformation[Min\_Node\_Vehicle].AddNode(NextNode);

MyVehicleInformation[Min\_Node\_Vehicle].set\_Capacity(MyVehicleInf ormation[Min\_Node\_Vehicle].get\_Capacity() - MyNodesInformation[NextNode].get\_Demand());

} // end if else

{

cout << "All Vehicles are full, Node - " << NextNode

<< " - will be served next day.";

} // end else-if

} // end for

cout << endl << "Iteration: " << iteration << " - New Vehicle Clusters:" << endl << "--------------------" << endl ;

for (int i = 0; i < k; i++)

{

cout << "Vehicle " << i << ": "; MyVehicleInformation[i].DisplayVectorContents(); cout << endl;

}

NewCenters(MyVehicleInformation); do

{

int No\_Vehicle = 0;

if (Centers[1][No\_Vehicle] != New\_Centers[1][No\_Vehicle])

{

flag\_centers = true;

}

if (Centers[2][No\_Vehicle] != New\_Centers[2][No\_Vehicle])

{

flag\_centers = true;

}

No\_Vehicle++;

} while (flag\_centers == false); // Checks if any change occur on vehicle centers

if (flag\_centers == true)

{

iteration++; cout << endl;

for (int i = 0; i < k; i++)

{

Centers[1][i] = New\_Centers[1][i]; Centers[2][i] = New\_Centers[2][i];

}

}

}

void ShortBasedOnPriority(vector<NodesInfo>)

{

double Priority[N]; Priority[0] = 0;

for (int i = 0; i < k; i++)

{

double x = New\_Centers[1][i]; double y = New\_Centers[2][i];

for (int j = 1; j < MyVehicleInformation[i].NodesOfCluster.size(); j++)

{

int node = MyVehicleInformation[i].NodesOfCluster.at(j);

double Priority\_x = MyNodesInformation[node].get\_X();

double Priority\_y = MyNodesInformation[node].get\_Y();

double distancex = (x - Priority\_x)\*(x -

Priority\_x); Priority\_y); distancey));

double distancey = (y - Priority\_y)\*(y - double Priority\_Distance = sqrt(abs(distancex + Priority[node] = Priority\_Distance /

MyNodesInformation[node].get\_Demand();

}

}

for (int i = 0; i < N; i++)

{

ShortDEMAND[i][1] = i; ShortDEMAND[i][2] = Priority[i];

}

// Shorting D array by desceding order for (int i = 0; i < N; i++)

{

for (int j = i + 1; j < N; j++)

{

if (ShortDEMAND[i][2] < ShortDEMAND[j][2])

{

double temp1 = ShortDEMAND[i][2]; ShortDEMAND[i][2] = ShortDEMAND[j][2];

ShortDEMAND[j][2] = temp1;

double temp2 = ShortDEMAND[i][1]; ShortDEMAND[i][1] = ShortDEMAND[j][1];

ShortDEMAND[j][1] = temp2;

}

}

}

}

void NewCenters(vector<VehicleInfo>)

{

cout << endl << "New Centers" << endl << "-----------" << endl; for (int i = 0; i < k; i++)

{

double Sum\_X = 0.0; double Sum\_Y = 0.0;

int size = MyVehicleInformation[i].NodesOfCluster.size(); for (int j = 0; j < size; j++)

{

int node = MyVehicleInformation[i].NodesOfCluster.at(j);

Sum\_X = Sum\_X + MyNodesInformation[node].get\_X(); Sum\_Y = Sum\_Y + MyNodesInformation[node].get\_Y();

}

New\_Centers[1][i] = Sum\_X / size; New\_Centers[2][i] = Sum\_Y / size;

cout << "Center: " << i << " | (x,y)=(" << New\_Centers[1][i] << "," << New\_Centers[2][i] << ")" << endl;

}

}

void FirstClusteringIteration(vector<VehicleInfo>) { for (int i = 0; i < k; i++)

{

MyVehicleInformation[i].AddNode(0);

// Sets starting point of each Vehicle Depot (0)

}

for (int i = 0; i < k; i++)

{

MyVehicleInformation[i].AddNode(ShortDEMAND[i][1]);

// Adds the first node to each vehicle, based on shorted list "ShortDEMAND"

}

for (int i = 0; i < k; i++)

{

node = MyVehicleInformation[i].NodesOfCluster.at(1); Centers[1][i] =

MyNodesInformation[MyVehicleInformation[i].NodesOfCluster.at(1)].get\_X ();

Centers[2][i] = MyNodesInformation[MyVehicleInformation[i].NodesOfCluster.at(1)].get\_Y ();

cout << endl << "Center: " << i << " | NODE: " << node << " | (x,y)=(" << Centers[1][i] << "," << Centers[2][i] << ")"

<< endl;

}

bool flag;

double x1, x2, y1, y2; cout << endl;

for (int i = 0; i < N-1; i++)

{

flag = false;

double NextNode = ShortDEMAND[i][1];

x2 = MyNodesInformation[NextNode].get\_X(); y2 = MyNodesInformation[NextNode].get\_Y(); double Min\_Distance = 9999999.99;

int Min\_Node\_Vehicle = -1; for (int j = 0; j < k; j++)

{

if (MyVehicleInformation[j].get\_Capacity() - MyNodesInformation[NextNode].get\_Demand() >= 0)

{

flag = true;

x1 = Centers[1][j]; y1 = Centers[2][j];

dy));

double dx = (x1 - x2)\*(x1 - x2);

// distance from center point j - X angle

double dy = (y1 - y2)\*(y1 - y2);

// distance from center point j - Y angle

double Calculated\_Distance = sqrt(abs(dx +

if (Calculated\_Distance < Min\_Distance)

{

Min\_Distance = Calculated\_Distance; Min\_Node\_Vehicle = j;

} // end if

} // end if

} // end for

if (flag = true)

{

MyVehicleInformation[Min\_Node\_Vehicle].AddNode(NextNode);

MyVehicleInformation[Min\_Node\_Vehicle].set\_Capacity(MyVehicleInf ormation[Min\_Node\_Vehicle].get\_Capacity() - MyNodesInformation[NextNode].get\_Demand());

cout << endl << Min\_Node\_Vehicle << " - " << NextNode << " - " << MyNodesInformation[NextNode].get\_Demand() << " - " << MyVehicleInformation[j].get\_Capacity() << endl;

} // end if else

{

cout << "All Vehicles are full, Node - " << NextNode

<< " - will be served next day.";

} // end else-if

} // end for

cout << endl << endl << "Clusters after 1st iteration:" << endl

<< "------------------------------------" << endl;

for (int i = 0; i < k; i++)

{

cout << "Vehicle " << i << ": "; MyVehicleInformation[i].DisplayVectorContents(); cout << endl;

}

}

void ShortBasedOnDemand(vector<NodesInfo>)

{

double Priority[N]; Priority[0] = 0;

for (int i = 1; i < N; i++)

{

Priority[i] = D[0][i] / MyNodesInformation[i].get\_Demand();

cout << "Node : " << i << " - Priority :" << Priority[i]

<< endl;

}

for (int i = 0; i < N; i++)

{

ShortDEMAND[i][1] = i; ShortDEMAND[i][2] = Priority[i];

}

// Shorting D array with decreasing order for (int i = 0; i < N; i++)

{

for (int j = i + 1; j < N; j++)

{

if (ShortDEMAND[i][2] < ShortDEMAND[j][2])

{

double temp1 = ShortDEMAND[i][2]; ShortDEMAND[i][2] = ShortDEMAND[j][2];

ShortDEMAND[j][2] = temp1;

double temp2 = ShortDEMAND[i][1]; ShortDEMAND[i][1] = ShortDEMAND[j][1];

ShortDEMAND[j][1] = temp2;

}

}

}

cout << endl << "Shorted Nodes by Priority Rate (High to Low): "

<< endl;

for (i = 0; i < N; i++)

{

cout << "#" << i + 1 << " | Node:" << ShortDEMAND[i][1] << " | Priority:" << ShortDEMAND[i][2] << endl;

// ShortDEMAND[i][1] <-- Node | ShortDEMAND[i][2] <--

Distance

}

}

//ok

void fillVectorFile(vector<NodesInfo>& newNodesVector)

{

int in\_node, in\_x, in\_y, in\_demand, in\_early, in\_late, in\_service;

cout << "display RC201.100.txt contents" << endl; // Open file

ifstream infile; infile.open("RC201.100.txt");

if (infile.fail()) //check for error

{

cerr << "Error Opening File" << endl; exit(1);

}

int count = 0;

for (i = 0; i < N; i++) //read file, every 7 elements changes line, which means that node changes too

{

infile >> in\_node >> in\_x >> in\_y >> in\_demand >> in\_early

>> in\_late >> in\_service;

NodesInfo Node1(in\_node, in\_x, in\_y, in\_demand, in\_early, in\_late, in\_service);

count += 1; // counter for node number newNodesVector.push\_back(Node1);

}

cout << "Access on file achieved." << endl << count << " Nodes found in file." << endl;

}

void printVector(const vector<NodesInfo>& newNodesVector) {

unsigned int size = newNodesVector.size(); for (unsigned int i = 0; i < size; i++)

{

<< endl;

}

cout << "Node: " << newNodesVector[i].get\_Node() << " X: " << newNodesVector[i].get\_X() <<

* Y: " << newNodesVector[i].get\_Y() <<
* Demand: " << newNodesVector[i].get\_Demand() << " Early: " << newNodesVector[i].get\_Early() << " Late: " << newNodesVector[i].get\_Late() <<
* Service: " << newNodesVector[i].get\_Service()

}

void ValidationCheckOfDemand(const vector<NodesInfo>& newNodesVector)

{

unsigned int count = 0;

unsigned int size = newNodesVector.size(); for (unsigned int i = 0; i < size; i++)

{

if (newNodesVector[i].get\_Demand() > Q)

{

cout << i << " node's demand, exceeds vehicle's capacity." << endl;

validation++;

}

count = count + newNodesVector[i].get\_Demand();

}

if (count > Q\*k)

{

cout << endl << "Overal demand exceeds the demand that can be coverd with " << k << " vehicles." << endl;

validation++;

}

else

{

cout << endl << "Demand values approved, continue.";

}

}

void CalculateAllNodesDistance(const vector<NodesInfo>& newNodesVector)

{

double x1, x2, y1, y2;

//cout << endl; double max = -1; double min = 100000;

for (unsigned int i = 0; i < N; i++)

{

x1 = newNodesVector[i].get\_X(); y1 = newNodesVector[i].get\_Y();

for (unsigned int j = 0; j < N; j++)

{

x2 = newNodesVector[j].get\_X(); y2 = newNodesVector[j].get\_Y();

double distancex = (x1 - x2)\*(x1 - x2); double distancey = (y1 - y2)\*(y1 - y2);

double calcdist = sqrt(abs(distancex + distancey)); D[i][j] = calcdist;

}

}

}

void fillVectorVehicle(vector<VehicleInfo>& newNodesVector)

{

for (i = 0; i < k; i++)

{

int in\_number =i; double in\_opp\_time = O; double in\_capacity = Q;

VehicleInfo Node1(in\_number, in\_opp\_time, in\_capacity); newNodesVector.push\_back(Node1);

}

}

void printVectorVehicles(const vector<VehicleInfo>) { unsigned int size = MyVehicleInformation.size(); cout << endl;

for (unsigned int i = 0; i < size; i++)

{

cout << "Vehicle: " << MyVehicleInformation[i].get\_Number() <<

* + Opperation Time: " << MyVehicleInformation[i].get\_Opperation\_Time() <<
  + Capacity: " << MyVehicleInformation[i].get\_Capacity()

<< endl;

}

}

**// HEADERS – PHASE ONE - CLUSTERING**

#ifndef \_NODEINFO\_H #define \_NODEINFO\_H #define N 26 #include <math.h> class NodesInfo { private:

// Member Variables int newNode;

int newX; int newY;

int newDemand; int newEarly; int newLate; int newService;

public:

// Default Constructor NodesInfo();

// Overload Constructor

NodesInfo(int, int, int, int, int, int, int);

// Destructor

~NodesInfo();

// Accessor Functions

int get\_Node() const; // returns Node number

int get\_X() const; // returns Node X coords

int get\_Y() const; // returns Node Y coords

int get\_Demand() const; // returns Node Demands

int get\_Early() const; // returns Node Early TW

int get\_Late() const; // returns Node Late TW

int get\_Service() const; // returns Node Service time double get\_DistanceTo(NodesInfo OtherNode) {

double tempX = (newX - OtherNode.get\_X()); double tempY = (newY - OtherNode.get\_Y()); return sqrt((tempX\*tempX) + (tempY\*tempY));

}; // returns distance to another node

//Mutator Functions void set\_Node(int);

void set\_X(int); void set\_Y(int);

void set\_Demand(int); void set\_Early(int); void set\_Late(int); void set\_Service(int);

};

#endif //\_NODEINFO\_H

#ifndef \_VEHICLEINFO\_H #define \_VEHICLEINFO\_H #define N 26

#include <math.h> #include <vector> #include "NodesInfo.h" using namespace std;

class VehicleInfo { private:

// Member Variables int newNumber; double newOppTime; double newCapacity;

public:

std::vector <int> NodesOfCluster; void AddNode (int node\_number)

{

NodesOfCluster.push\_back(node\_number);

}

void EraseData()

{

NodesOfCluster.clear();

}

void DisplayVectorContents()

{

for (unsigned int i = 0; i < NodesOfCluster.size(); i++)

{

cout << NodesOfCluster[i] << " ";

}

}

// Default Constructor VehicleInfo();

// Overload Constructor VehicleInfo(int, double, double);

// Destructor

~VehicleInfo();

// Accessor Functions int get\_Number() const;

int get\_Opperation\_Time() const; int get\_Capacity() const;

int get\_Number\_of\_Nodes();

//Mutator Functions void set\_Number(int);

void set\_Opperation\_Time(double); void set\_Capacity(double);

void set\_Nodes\_Of\_Cluster(int);

};

#endif //\_VEHICLEINFO\_H

#include "NodesInfo.h" #define N 26

// Default Constructor NodesInfo::NodesInfo() {

newNode = 0;

newX = 0;

newY = 0;

newDemand = 0;

newEarly = 0;

newLate = 0;

newService = 0;

}

// Overload Constructor

NodesInfo::NodesInfo(int in\_node, int in\_x, int in\_y, int in\_demand, int in\_early, int in\_late, int in\_service)

{

newNode = in\_node; newX = in\_x;

newY = in\_y;

newDemand = in\_demand; newEarly = in\_early; newLate = in\_late; newService = in\_service;

}

// Destructor NodesInfo::~NodesInfo() { }

// Accessor Functions

int NodesInfo::get\_Node() const { return newNode; } int NodesInfo::get\_X() const { return newX; }

int NodesInfo::get\_Y() const { return newY; }

int NodesInfo::get\_Demand() const { return newDemand; } int NodesInfo::get\_Early() const { return newEarly; } int NodesInfo::get\_Late() const { return newLate; }

int NodesInfo::get\_Service() const { return newService; }

// Mutator Functions

void NodesInfo::set\_Node(int in\_node) { newNode = in\_node; } void NodesInfo::set\_X(int in\_x) { newX = in\_x; }

void NodesInfo::set\_Y(int in\_y) { newY = in\_y; }

void NodesInfo::set\_Demand(int in\_demand) { newDemand = in\_demand; } void NodesInfo::set\_Early(int in\_early) { newEarly = in\_early; } void NodesInfo::set\_Late(int in\_late) { newLate = in\_late; }

void NodesInfo::set\_Service(int in\_service) { newService = in\_service;

}

#include "VehicleInfo.h" #define N 101

// Default Constructor VehicleInfo::VehicleInfo() {

newNumber = 0;

newOppTime = 0.0;

newCapacity = 0.0;

}

// Overload Constructor

VehicleInfo::VehicleInfo(int in\_number, double in\_opp\_time, double in\_capacity)

{

newNumber = in\_number; newOppTime = in\_opp\_time; newCapacity = in\_capacity;

}

// Destructor VehicleInfo::~VehicleInfo() { }

// Accessor Functions

int VehicleInfo::get\_Number() const { return newNumber; }

int VehicleInfo::get\_Opperation\_Time() const { return newOppTime; } int VehicleInfo::get\_Capacity() const { return newCapacity; }

int VehicleInfo::get\_Number\_of\_Nodes() { return NodesOfCluster.size();

}

// Mutator Functions

void VehicleInfo::set\_Number(int in\_number) { newNumber = in\_number; } void VehicleInfo::set\_Opperation\_Time(double in\_opp\_time) { newOppTime

= in\_opp\_time; }

void VehicleInfo::set\_Capacity(double in\_capacity) { newCapacity = in\_capacity; }

**// PHASE TWO – VEHICLE ROUTING**

#include <iostream> #include <fstream> #include <string> #include <math.h>

#include <stdio.h> /\* printf, scanf, puts, NULL \*/ #include <stdlib.h> /\* srand, rand \*/ #include <time.h> /\* time \*/

#include "NodesInfo.h" #include <vector> using namespace std;

#define Nodes 101 // number of nodes

#define Capacity 100 // capacity of each vehicle #define Vehicles 20 // number of vehicles

#define O 5000 // maximum operation time per vehicle

#define Clusters 20 // number of Clusters

int N = Nodes; int C = Clusters; int k = Vehicles; int Q = Capacity;

static double D[Nodes][Nodes]; // distance between node i & j static int DataSet;

static int validation = 0; static int NumberOfNodesOnRoute; static int LastNode;

static int non\_zero\_elements; static double time\_distance; static int File\_i = Clusters; static int File\_j = Clusters;

static int R[Clusters][Nodes - Clusters]; static int User\_Selection;

static double Total\_Wait\_Time; static int Counter\_i;

static int Counter\_j; static int countersize; int i, j;

void DataSetsCall(); void ReadClusterFile(); void LoadDataSet();

void fillVectorFile(vector<NodesInfo>&);

// fillVector - Fill in vector with each node's information void printVector(const vector<NodesInfo>&);

// printVector - prints the information of all nodes void ValidationCheckOfDemand(const vector<NodesInfo>&);

// ValidationCheckOfDemand - Checks if demand of each node, and

overal demand exceeds our vehicles' capacity and prints messege to continue or not. If not terminates program

void CalculateAllNodesDistance(const vector<NodesInfo>&);

// CalculateAllNodesDistance - Fills up D[N][N] array with the distances between nodes

void fillVectorRouteNull(vector<NodesInfo>&);

void printVectorRouteNull(const vector<NodesInfo>&);

// CASE 1

void GetNextNode(vector<NodesInfo>);

// CASE 2

void InverseDijkstra(vector<NodesInfo>); void GetNextNode\_Inverse(vector<NodesInfo>);

// CASE 3

void MinWaitTimeServedFirst(vector<NodesInfo>);

// CASE 4

void MinDemandFirstServed(vector<NodesInfo>);

// CASE 5

void MaxDemandFirstServed(vector<NodesInfo>);

// EXTRA ITERATION ON REMAINING NODES

void Part2(); void Part3();

void extracluster();

vector <NodesInfo> MyNodesInformation; vector <NodesInfo> RoutesNotSatisfied; vector <NodesInfo> NotSelectedNodes; static int Not\_Selected\_Nodes\_Counter;

vector <NodesInfo> DijkstraRoute; // CASE 1 - Output Vector vector <NodesInfo> DijkstraInverse; // CASE 2 - Output Vector vector <NodesInfo> MinWait; // CASE 3 - Output Vector vector <NodesInfo> MinDemand; // CASE 3 - Output Vector vector <NodesInfo> MaxDemand; // CASE 3 - Output Vector

int main()

{

fillVectorRouteNull(RoutesNotSatisfied); DataSetsCall(); fillVectorFile(MyNodesInformation);

// fills in vector "MyNodesInformation" from file R201a.txt printVector(MyNodesInformation);

// prints list of each node's information ValidationCheckOfDemand(MyNodesInformation);

// Validation check ;

if (validation > 0) // Terminating program

{

endl;

}

cout << endl << "Terminating program. Check inputs.." << int terminationintl;

cin >> terminationintl; return 0;

CalculateAllNodesDistance(MyNodesInformation);

// fills in static array D[i][j] with distance between nodes LoadDataSet(); // Allazei se kathe periptwsi ta N-k-C

//ReadING Cluster File

for (i = 0; i < File\_i; i++)

{

for (j = 0; j <= File\_j; j++)

{

cout << R[i][j] <<" ";

}

cout << endl;

}

cout << "Select Solution Type:" << endl

<< "1 - Dijkstra" << endl

<< "2 - Dijkstra Inversed" << endl

<< "3 - Minimum Wait Time" << endl

<< "4 - Minimum Demand" << endl

<< "5 - Maximum Demand" << endl; cin >> User\_Selection;

while ((User\_Selection != 1) && (User\_Selection != 2) && (User\_Selection != 3) && (User\_Selection != 4) && (User\_Selection != 5))

{

cout << "Invalid Value. Try again..."; cin >> User\_Selection;

}

Part2(); string input; do

{

cout << endl << endl << endl<< "Wanna try again? (Y/N)"; cin >> input;

while ((input != "Y") && (input != "N"))

{

cout << "Insert correct value (Y/N)."; cin >> input;

}

if (input == "Y")

{

cout << "Select Solution Type:" << endl

<< "1 - Dijkstra" << endl

<< "2 - Dijkstra Inversed" << endl

<< "3 - Minimum Wait Time" << endl

<< "4 - Minimum Demand" << endl

<< "5 - Maximum Demand" << endl; cin >> User\_Selection;

while ((User\_Selection != 1) && (User\_Selection !=

2) && (User\_Selection != 3) && (User\_Selection != 4) && (User\_Selection != 5))

{

cout << "Invalid Value. Try again..."; cin >> User\_Selection;

}

NotSelectedNodes.clear(); Part2();

}

} while (input == "Y"); do

{

cout << endl << "Do you need to create an extra route? (Y/N)" << endl;

cin >> input;

while ((input != "Y") && (input != "N"))

{

cout << "Insert correct value (Y/N)."; cin >> input;

}

int non\_zero = 0;

for (int j = 0; j < NotSelectedNodes.size(); j++)

{

if (NotSelectedNodes[j].get\_Node() > 0)

{

non\_zero++;

}

}

if (non\_zero > 0)

{

if (input == "Y")

{

}

}

else

{

extracluster();

<< endl;

}

input = "N";

cout << endl << "All nodes are assigned to clusters"

} while (input == "Y");

cout << endl << endl << "ENDING PROGRAM..."; int a;

cin >> a;

}

//--------------E N D O F M A I N---------------//

void Part3()

{

int size = RoutesNotSatisfied.size(); non\_zero\_elements = 0;

cout << endl << "--------------------" << endl << "Nodes to be

routed: ";

for (j = 0; j < size; j++)

{

if (RoutesNotSatisfied[j].get\_Node() > 0)

{

non\_zero\_elements += 1;

cout << RoutesNotSatisfied[j].get\_Node() << " ";

}

}

cout << endl << "Number of nodes to be routed: " << non\_zero\_elements << endl;

// CALL FUNCTIONS - PRINT RESULTS

NumberOfNodesOnRoute = 1; // Number of Nodes visited on the specific route which is selected (depot included)

LastNode = 0;

// Last visited node on the specific route which is selected time\_distance = 0.0;

// CASE 1 - Dijkstra - Shortest Node if (User\_Selection == 1)

{

cout << endl << "---EXTRA CLUSTER CASE 1---" << endl;

DijkstraRoute.clear(); fillVectorRouteNull(DijkstraRoute);

// Setting NULLS to all values of vector "DijkstraRoute" DijkstraRoute[0].set\_Node(0);

// Setting first point of each route the depot point GetNextNode(RoutesNotSatisfied);

for (i = 0; i < DijkstraRoute.size(); i++)

{

if (DijkstraRoute[i].get\_Node() != 0)

{

cout << DijkstraRoute[i].get\_Node() << " ";

}

}

int Demand\_Counter = 0; countersize = 0;

for (int i = 1; i < DijkstraRoute.size(); i++)

{

Demand\_Counter += MyNodesInformation[DijkstraRoute[i].get\_Node()].get\_Demand();

if (DijkstraRoute[i].get\_Node() > 0)

{

countersize = countersize + 1;

}

}

cout << endl << "Extra Route Demand: " << Demand\_Counter

<< " Number of Nodes: " << countersize;

}

else if (User\_Selection == 2)

{

cout << endl << "---EXTRA CLUSTER CASE 2---" << endl;

DijkstraInverse.clear(); fillVectorRouteNull(DijkstraInverse);

// Setting NULLS to all values of vector "DijkstraRoute" DijkstraInverse[0].set\_Node(0);

// Setting first point of each route the depot point InverseDijkstra(RoutesNotSatisfied);

for (i = 0; i < DijkstraInverse.size(); i++)

{

if (DijkstraInverse[i].get\_Node() != 0)

{

cout << DijkstraInverse[i].get\_Node() << " ";

}

}

int Demand\_Counter = 0; countersize = 0;

for (int i = 1; i < DijkstraInverse.size(); i++)

{

Demand\_Counter += MyNodesInformation[DijkstraInverse[i].get\_Node()].get\_Demand();

if (DijkstraInverse[i].get\_Node() > 0)

{

countersize = countersize + 1;

}

}

cout << endl << "Extra Route Demand: " << Demand\_Counter

<< " Number of Nodes: " << countersize ;

}

else if (User\_Selection == 3)

{

cout << endl << "---EXTRA CLUSTER CASE 3---" << endl;

MinWait.clear();

fillVectorRouteNull(MinWait); // Setting NULLS to all values of vector "DijkstraRoute"

MinWait[0].set\_Node(0); //

Setting first point of each route the depot point MinWaitTimeServedFirst(RoutesNotSatisfied); for (i = 0; i < MinWait.size(); i++)

{

if (MinWait[i].get\_Node() != 0)

{

cout << MinWait[i].get\_Node() << " ";

}

}

int Demand\_Counter = 0; countersize = 0;

for (int i = 1; i < MinWait.size(); i++)

Demand\_Counter += MyNodesInformation[MinWait[i].get\_Node()].get\_Demand();

if (MinWait[i].get\_Node() > 0)

{

countersize = countersize + 1;

}

}

cout << endl << "Extra Route Demand: " << Demand\_Counter

<< " Number of Nodes: " << countersize;

}

else if (User\_Selection == 4)

{

cout << endl << "---EXTRA CLUSTER CASE 4---" << endl;

MinDemand.clear(); fillVectorRouteNull(MinDemand);

// Setting NULLS to all values of vector "DijkstraRoute" MinDemand[0].set\_Node(0);

// Setting first point of each route the depot point MinDemandFirstServed(RoutesNotSatisfied);

for (i = 0; i < MinDemand.size(); i++)

{

if (MinDemand[i].get\_Node() != 0)

{

cout << MinDemand[i].get\_Node() << " ";

}

}

int Demand\_Counter = 0; countersize = 0;

for (int i = 1; i < MinDemand.size(); i++)

{

Demand\_Counter += MyNodesInformation[MinDemand[i].get\_Node()].get\_Demand();

if (MinDemand[i].get\_Node() > 0)

{

countersize = countersize + 1;

}

}

cout << endl << "Extra Route Demand: " << Demand\_Counter

<< " Number of Nodes: " << countersize;

}

else

{

cout << endl << "---EXTRA CLUSTER CASE 5---" << endl;

MaxDemand.clear(); fillVectorRouteNull(MaxDemand);

// Setting NULLS to all values of vector "DijkstraRoute" MaxDemand[0].set\_Node(0);

// Setting first point of each route the depot point MaxDemandFirstServed(RoutesNotSatisfied);

for (i = 0; i < MaxDemand.size(); i++)

if (MaxDemand[i].get\_Node() != 0)

{

cout << MaxDemand[i].get\_Node() << " ";

}

}

int Demand\_Counter = 0; countersize = 0;

for (int i = 1; i < MaxDemand.size(); i++)

{

Demand\_Counter += MyNodesInformation[MaxDemand[i].get\_Node()].get\_Demand();

if (MaxDemand[i].get\_Node() > 0)

{

countersize = countersize + 1;

}

}

cout << endl << "Extra Route Demand: " << Demand\_Counter

<< " Number of Nodes: " << countersize;

}

// PRINT VECTOR WITH NOT SELECTED NODES

cout << endl << "Not Selected Nodes: ";

for (i = 0; i < NotSelectedNodes.size(); i++)

{

if (NotSelectedNodes[i].get\_Node() != 0)

{

cout << NotSelectedNodes[i].get\_Node() << " ";

}

}

}

void extracluster()

{

Not\_Selected\_Nodes\_Counter = 0; int in\_node;

int counter = 0; RoutesNotSatisfied.clear(); fillVectorRouteNull(RoutesNotSatisfied); Total\_Wait\_Time = 0.0;

for (int i = 0; i < NotSelectedNodes.size(); i++)

{ // for each route, sets on RoutesNotSatisfied vector a different spot, and the rest places are nulls as set at "fillVectorRouteNull"

if (NotSelectedNodes[i].get\_Node() != 0)

{

in\_node = NotSelectedNodes[i].get\_Node(); RoutesNotSatisfied[counter].set\_Node(in\_node); counter++;

}

}

cout << endl << endl; NotSelectedNodes.clear();

fillVectorRouteNull(NotSelectedNodes);

// Efoson perasa ta nodes ston NotSelected, midenizw to vector me ayta pou den exoun ikanopoihthei

int size = RoutesNotSatisfied.size(); non\_zero\_elements = 0;

cout << endl << "----------" << endl << "Nodes to be routed: "; cout << "Select Solution Type:" << endl

<< "1 - Dijkstra" << endl

<< "2 - Dijkstra Inversed" << endl

<< "3 - Minimum Wait Time" << endl

<< "4 - Minimum Demand" << endl

<< "5 - Maximum Demand" << endl; cin >> User\_Selection;

while ((User\_Selection != 1) && (User\_Selection != 2) && (User\_Selection != 3) && (User\_Selection != 4) && (User\_Selection != 5))

{

cout << "Invalid Value. Try again..."; cin >> User\_Selection;

}

for (j = 0; j < size; j++)

{

if (RoutesNotSatisfied[j].get\_Node() > 0)

{

non\_zero\_elements += 1;

cout << RoutesNotSatisfied[j].get\_Node() << " ";

}

}

NumberOfNodesOnRoute = 1; // Number of Nodes visited on the specific route which is selected (depot included)

LastNode = 0;

// Last visited node on the specific route which is selected time\_distance = 0.0;

cout << endl << "non zero elements: " << non\_zero\_elements <<

endl;

}

Part3();

void Part2()

{

fillVectorRouteNull(RoutesNotSatisfied);

// Setting NULLS to all values of vector "RoutesNotSatisfied" fillVectorRouteNull(NotSelectedNodes);

// Setting NULLS to all values of vector "NotSelectedNodes" Not\_Selected\_Nodes\_Counter = 0;

for (int ii = 0; ii < File\_i; ii++)

{

int in\_node; int counter = 0;

for (j = 0; j < File\_j; j++)

{ // for each route, sets on RoutesNotSatisfied vector a different spot, and the rest places are nulls as set at "fillVectorRouteNull"

in\_node = R[ii][j]; RoutesNotSatisfied[counter].set\_Node(in\_node); counter++;

routed: ";

}

int size = RoutesNotSatisfied.size(); non\_zero\_elements = 0;

cout << endl << "-------------" << endl << "Nodes to be

for (j = 0; j < size; j++)

{

if (RoutesNotSatisfied[j].get\_Node() > 0)

{

non\_zero\_elements += 1;

cout << RoutesNotSatisfied[j].get\_Node() << "

";

}

}

<< endl;

endl;

cout << endl << "non zero elements: " << non\_zero\_elements

// CALL FUNCTIONS - PRINT RESULTS

cout << endl << "Cluster " << ii + 1 << " elements: " <<

cout << endl << ii + 1 << " Route Nodes: "; NumberOfNodesOnRoute = 1; // Number

of Nodes visited on the specific route which is selected (depot included)

LastNode = 0; //

Last visited node on the specific route which is selected time\_distance = 0.0;

// CASE 1 - Dijksta - Shortest Node if (User\_Selection == 1)

{

cout << endl << "---C A S E 1---" << endl; DijkstraRoute.clear(); fillVectorRouteNull(DijkstraRoute);

// Setting NULLS to all values of vector "DijkstraRoute" DijkstraRoute[0].set\_Node(0);

// Setting first point of each route the depot point Total\_Wait\_Time = 0.0; GetNextNode(RoutesNotSatisfied);

for (i = 0; i < DijkstraRoute.size(); i++)

{

if (DijkstraRoute[i].get\_Node() != 0)

{

cout << DijkstraRoute[i].get\_Node() << " ";

}

}

int Demand\_Counter = 0;

countersize = 0;

for (int i = 1; i < DijkstraRoute.size(); i++)

{

Demand\_Counter += MyNodesInformation[DijkstraRoute[i].get\_Node()].get\_Demand();

if (DijkstraRoute[i].get\_Node() > 0)

{

countersize = countersize + 1;

}

}

cout << endl << "Route " << ii << " demand : " << Demand\_Counter << " Number on Nodes: " << countersize;

}

else if ( User\_Selection == 2)

{

"DijkstraRoute"

cout << endl << "---C A S E 2---" << endl; Total\_Wait\_Time = 0.0; DijkstraInverse.clear(); fillVectorRouteNull(DijkstraInverse);

// Setting NULLS to all values of vector

DijkstraInverse[0].set\_Node(0);

// Setting first point of each route the depot point InverseDijkstra(RoutesNotSatisfied);

for (i = 0; i < DijkstraInverse.size(); i++)

{

if (DijkstraInverse[i].get\_Node() != 0)

{

cout << DijkstraInverse[i].get\_Node() << " ";

}

}

int Demand\_Counter = 0; countersize = 0;

for (int i = 1; i < DijkstraInverse.size(); i++)

{

Demand\_Counter += MyNodesInformation[DijkstraInverse[i].get\_Node()].get\_Demand();

if (DijkstraInverse[i].get\_Node() > 0)

{

countersize = countersize + 1;

}

}

cout << endl << "Route " << ii << " demand : " << Demand\_Counter << " Number on Nodes: " << countersize ;

}

else if ( User\_Selection ==3 )

{

cout << endl << "---C A S E 3---" << endl; Total\_Wait\_Time = 0.0;

MinWait.clear();

fillVectorRouteNull(MinWait);

// Setting NULLS to all values of vector "DijkstraRoute" MinWait[0].set\_Node(0);

// Setting first point of each route the depot point MinWaitTimeServedFirst(RoutesNotSatisfied); for (i = 0; i < MinWait.size(); i++)

{

if (MinWait[i].get\_Node() != 0)

{

cout << MinWait[i].get\_Node() << " ";

}

}

int Demand\_Counter = 0; countersize = 0;

for (int i = 1; i < MinWait.size(); i++)

{

Demand\_Counter += MyNodesInformation[MinWait[i].get\_Node()].get\_Demand();

if (MinWait[i].get\_Node() > 0)

{

countersize = countersize + 1;

}

}

cout << endl << "Route " << ii << " demand : " << Demand\_Counter << " Number on Nodes: " << countersize;

}

else if (User\_Selection == 4)

{

cout << endl << "---C A S E 4---" << endl; Total\_Wait\_Time = 0.0;

MinDemand.clear(); fillVectorRouteNull(MinDemand);

// Setting NULLS to all values of vector "DijkstraRoute" MinDemand[0].set\_Node(0);

// Setting first point of each route the depot point MinDemandFirstServed(RoutesNotSatisfied);

for (i = 0; i < MinDemand.size(); i++)

{

if (MinDemand[i].get\_Node() != 0)

{

cout << MinDemand[i].get\_Node() << " ";

}

}

int Demand\_Counter = 0; countersize = 0;

for (int i = 1; i < MinDemand.size(); i++)

{

Demand\_Counter += MyNodesInformation[MinDemand[i].get\_Node()].get\_Demand();

if (MinDemand[i].get\_Node() > 0)

{

countersize = countersize + 1;

}

}

cout << endl << "Route " << ii << " demand : " << Demand\_Counter << " Number on Nodes: " << countersize;

}

else

{

cout << endl << "---C A S E 5---" << endl; Total\_Wait\_Time = 0.0;

MaxDemand.clear(); fillVectorRouteNull(MaxDemand);

// Setting NULLS to all values of vector "DijkstraRoute" MaxDemand[0].set\_Node(0);

// Setting first point of each route the depot point MaxDemandFirstServed(RoutesNotSatisfied);

for (i = 0; i < MaxDemand.size(); i++)

{

if (MaxDemand[i].get\_Node() != 0)

{

cout << MaxDemand[i].get\_Node() << " ";

}

}

int Demand\_Counter = 0; countersize = 0;

for (int i = 1; i < MaxDemand.size(); i++)

{

Demand\_Counter += MyNodesInformation[MaxDemand[i].get\_Node()].get\_Demand();

if (MaxDemand[i].get\_Node() > 0)

{

countersize = countersize + 1;

}

}

cout << endl << "Route " << ii << " demand : " << Demand\_Counter << " Number on Nodes: " << countersize;

}

// PRINT VECTOR WITH NOT SELECTED NODES

cout << endl << "Not Selected Nodes: ";

for (i = 0; i < NotSelectedNodes.size(); i++)

{

if (NotSelectedNodes[i].get\_Node() != 0)

{

cout << NotSelectedNodes[i].get\_Node() << " ";

}

}

}

}

// CASE 1 - Dijkstra - Shortest Node void GetNextNode(vector<NodesInfo>)

{

bool flag=false;

double min\_local = 999999999.99; int min\_local\_place = -1;

int min\_node = -1;

for (i = 0; i <= non\_zero\_elements; i++)

{

if (RoutesNotSatisfied[i].get\_Node() != 0)

{

if (D[LastNode][RoutesNotSatisfied[i].get\_Node()] < min\_local)

// check mellontika an prepei na einai k to 0 (depot) mesa

{

// pernwntas apo to depot gia kapoia apostash mporei na einai mikrotero to D[][]

min\_local =

D[LastNode][RoutesNotSatisfied[i].get\_Node()]; // minimum distance

min\_node = RoutesNotSatisfied[i].get\_Node();

min\_local\_place = i;

// node's place on vector "RoutesNotSatisfied" with the minimum distance

}

}

}

cout <<NumberOfNodesOnRoute << ") min distance: " << min\_local << " node: " << RoutesNotSatisfied[min\_local\_place].get\_Node() ;

double Late\_TW = MyNodesInformation[min\_node].get\_Late();

// Late Time Window of closest node

double Service\_Time = MyNodesInformation[min\_node].get\_Service();

double Early\_TW = MyNodesInformation[min\_node].get\_Early();

double Wait\_Time = 0.0;

if (min\_local + time\_distance <= Late\_TW) // min\_distance

<= Late TW of closest node

{

if (time\_distance < Early\_TW)

{

time\_distance;

Wait\_Time = Early\_TW - D[LastNode][min\_node] -

if (Wait\_Time < 0)

{

Wait\_Time = 0.0;

}

if (NumberOfNodesOnRoute ==1)

{

Wait\_Time = 0.0;

}

}

Total\_Wait\_Time = Total\_Wait\_Time + Wait\_Time;

cout << " Wait Time: " << Wait\_Time << " " << " Total Wait Time: " << Total\_Wait\_Time ;

}

else

{

NotSelectedNodes[Not\_Selected\_Nodes\_Counter].set\_Node(min\_node); Not\_Selected\_Nodes\_Counter++;

cout << endl << " Node Status: Not Selected

||";

}

flag = true;

if ((time\_distance + min\_local + Service\_Time + Wait\_Time

+ D[min\_node][0] <= MyNodesInformation[0].get\_Late()) &&

(time\_distance + min\_local /\*+ Service\_Time + Wait\_Time \*/<= MyNodesInformation[min\_node].get\_Late()))

{

DijkstraRoute[NumberOfNodesOnRoute].set\_Node(min\_node); cout << endl << "Node Status: Selected

||";

NumberOfNodesOnRoute += 1; LastNode = min\_node;

time\_distance = time\_distance + min\_local +

Service\_Time + Wait\_Time;

}

else

{

if (flag == false)

{

NotSelectedNodes[Not\_Selected\_Nodes\_Counter].set\_Node(min\_node); Not\_Selected\_Nodes\_Counter++;

cout << endl << " Node Status: Not Selected

||";

}

}

RoutesNotSatisfied.erase(RoutesNotSatisfied.begin() + min\_local\_place);

non\_zero\_elements -= 1;

cout << " Current Time: " << time\_distance << endl; if (non\_zero\_elements > 0)

{

}

else

{

}

GetNextNode(RoutesNotSatisfied);

DijkstraRoute[NumberOfNodesOnRoute+1].set\_Node(0);

}// end void GetNextNode

// CASE 2 - Invers Dijkstra - First Node is the furtherst and then closest till the depot node - Fuel Economy

void InverseDijkstra(vector<NodesInfo>)

{

bool flag = false;

double max\_local = - 1.00; int max\_local\_place = -1; int max\_node = -1;

for (i = 0; i <= non\_zero\_elements; i++)

{

if (RoutesNotSatisfied[i].get\_Node() != 0)

{

if (D[LastNode][RoutesNotSatisfied[i].get\_Node()] > max\_local) // check mellontika an prepei na einai k to 0 (depot) mesa

{ // pernwntas apo to depot gia kapoia apostash mporei na einai mikrotero to D[][]

max\_local = D[LastNode][RoutesNotSatisfied[i].get\_Node()]; // minimum distance

max\_node = RoutesNotSatisfied[i].get\_Node(); max\_local\_place = i;

// node's place on vector "RoutesNotSatisfied" with the minimum distance

}

}

}

cout << NumberOfNodesOnRoute << ") max distance: " << max\_local

<< " node: " << RoutesNotSatisfied[max\_local\_place].get\_Node(); double Late\_TW = MyNodesInformation[max\_node].get\_Late(); //

Late Time Window of closest node double Service\_Time =

MyNodesInformation[max\_node].get\_Service();

double Early\_TW = MyNodesInformation[max\_node].get\_Early(); double Wait\_Time = 0.0;

if (max\_local <= Late\_TW) // min\_distance <= Late TW of closest

node

{

if (time\_distance < Early\_TW)

{

time\_distance;

Wait\_Time = Early\_TW - D[LastNode][max\_node] -

if (Wait\_Time < 0)

{

Wait\_Time = 0.0;

}

if (NumberOfNodesOnRoute == 1)

{

Wait\_Time = 0.0;

}

}

Total\_Wait\_Time = Total\_Wait\_Time + Wait\_Time;

cout << " Wait Time: " << Wait\_Time << " " << " Total Wait Time: " << Total\_Wait\_Time;

}

else

{

flag == true;

NotSelectedNodes[Not\_Selected\_Nodes\_Counter].set\_Node(max\_node); Not\_Selected\_Nodes\_Counter++;

cout << endl << " Node Status: Not Selected ||";

}

if ((time\_distance + max\_local + Service\_Time + Wait\_Time +

D[max\_node][0] <= MyNodesInformation[0].get\_Late()) && (time\_distance + max\_local <=

MyNodesInformation[max\_node].get\_Late()))

{

Wait\_Time;

}

else

{

DijkstraInverse[NumberOfNodesOnRoute].set\_Node(max\_node); cout << endl << "Node Status: Selected ||"; NumberOfNodesOnRoute += 1;

LastNode = max\_node;

time\_distance = time\_distance + max\_local + Service\_Time +

if (flag == false)

{

NotSelectedNodes[Not\_Selected\_Nodes\_Counter].set\_Node(max\_node); Not\_Selected\_Nodes\_Counter++;

cout << endl << " Node Status: Not Selected

||";

}

}

RoutesNotSatisfied.erase(RoutesNotSatisfied.begin() + max\_local\_place);

non\_zero\_elements -= 1;

cout << " Current Time: " << time\_distance << endl; if (non\_zero\_elements > 0)

{

}

else

{

}

}

GetNextNode\_Inverse(RoutesNotSatisfied);

DijkstraInverse[NumberOfNodesOnRoute + 1].set\_Node(0);

void GetNextNode\_Inverse(vector<NodesInfo>)

{

bool flag = false;

double min\_local = 999999999.99; int min\_local\_place = -1;

int min\_node = -1;

for (i = 0; i <= non\_zero\_elements; i++)

{

if (RoutesNotSatisfied[i].get\_Node() != 0)

{

if (D[LastNode][RoutesNotSatisfied[i].get\_Node()] < min\_local) // check mellontika an prepei na einai k to 0 (depot) mesa

{ // pernwntas apo to depot gia kapoia apostash mporei na einai mikrotero to D[][]

min\_local = D[LastNode][RoutesNotSatisfied[i].get\_Node()]; // minimum distance

min\_node = RoutesNotSatisfied[i].get\_Node(); min\_local\_place = i;

// node's place on vector "RoutesNotSatisfied" with the minimum distance

}

}

}

cout << NumberOfNodesOnRoute << ") min distance: " << min\_local

<< " node: " << RoutesNotSatisfied[min\_local\_place].get\_Node(); double Late\_TW = MyNodesInformation[min\_node].get\_Late();

// Late Time Window of closest node double Service\_Time =

MyNodesInformation[min\_node].get\_Service();

double Early\_TW = MyNodesInformation[min\_node].get\_Early(); double Wait\_Time = 0.0;

if (min\_local <= Late\_TW) // min\_distance <= Late TW of closest

node

{

if (time\_distance < Early\_TW)

{

time\_distance;

Wait\_Time = Early\_TW - D[LastNode][min\_node] -

if (Wait\_Time < 0)

{

Wait\_Time = 0.0;

}

if (NumberOfNodesOnRoute == 1)

{

Wait\_Time = 0.0;

}

}

Total\_Wait\_Time = Total\_Wait\_Time + Wait\_Time;

cout << " Wait Time: " << Wait\_Time << " " << " Total Wait Time: " << Total\_Wait\_Time;

}

else

{

flag == true;

NotSelectedNodes[Not\_Selected\_Nodes\_Counter].set\_Node(min\_node); Not\_Selected\_Nodes\_Counter++;

cout << endl << " Node Status: Not Selected ||";

}

if ((time\_distance + min\_local + Service\_Time + Wait\_Time +

D[min\_node][0] <= MyNodesInformation[0].get\_Late()) && (time\_distance + min\_local<=

MyNodesInformation[min\_node].get\_Late()))

{

Wait\_Time;

}

else

{

DijkstraInverse[NumberOfNodesOnRoute].set\_Node(min\_node); cout << endl << "Node Status: Selected ||"; NumberOfNodesOnRoute += 1;

LastNode = min\_node;

time\_distance = time\_distance + min\_local + Service\_Time +

if (flag == false)

{

NotSelectedNodes[Not\_Selected\_Nodes\_Counter].set\_Node(min\_node); Not\_Selected\_Nodes\_Counter++;

cout << endl << " Node Status: Not Selected

||";

}

}

RoutesNotSatisfied.erase(RoutesNotSatisfied.begin() + min\_local\_place);

non\_zero\_elements -= 1;

cout << " Current Time: " << time\_distance << endl; if (non\_zero\_elements > 0)

{

}

else

{

}

}

GetNextNode\_Inverse(RoutesNotSatisfied);

DijkstraInverse[NumberOfNodesOnRoute + 1].set\_Node(0);

// CASE 3 - Min Wait Time

void MinWaitTimeServedFirst (vector<NodesInfo>)

{

bool flag = false; double Wait\_Time = 0.0;

double min\_wait\_local = 9999.99; int min\_wait\_local\_place = -1; int min\_wait\_node = -1;

double Early\_TW;

for (i = 0; i <= non\_zero\_elements; i++)

{

if (RoutesNotSatisfied[i].get\_Node() != 0)

{

Early\_TW = MyNodesInformation[RoutesNotSatisfied[i].get\_Node()].get\_Early();

Wait\_Time = Early\_TW - D[LastNode][RoutesNotSatisfied[i].get\_Node()] - time\_distance;

if (Wait\_Time < min\_wait\_local)

{ // CHECK HERE TIME WINDOW IN CASE < 0, SETTING ZERO (0) OR NEGATIVE VALUE

if (Early\_TW - D[LastNode][RoutesNotSatisfied[i].get\_Node()] < 0)

{

distance

}

else

{

min\_wait\_local = 0; // minimum

min\_wait\_local = 0; // Early\_TW -

D[LastNode][RoutesNotSatisfied[i].get\_Node()]; // minimum distance

}

min\_wait\_node = RoutesNotSatisfied[i].get\_Node();

min\_wait\_local\_place = i;

// node's place on vector "RoutesNotSatisfied" with the minimum distance

}

}

}

cout << NumberOfNodesOnRoute << ") distance: " << D[LastNode][min\_wait\_node] << " node: " << RoutesNotSatisfied[min\_wait\_local\_place].get\_Node();

double Late\_TW = MyNodesInformation[min\_wait\_node].get\_Late();

// Late Time Window of closest node double Service\_Time =

MyNodesInformation[min\_wait\_node].get\_Service();

Early\_TW = MyNodesInformation[min\_wait\_node].get\_Early(); Wait\_Time = 0.0;

if (D[LastNode][min\_wait\_node] <= Late\_TW)

// min\_distance <= Late TW of closest node

{

if (time\_distance < Early\_TW)

{

time\_distance;

Wait\_Time = Early\_TW - D[LastNode][min\_wait\_node] -

if (Wait\_Time < 0)

{

Wait\_Time = 0.0;

}

if (NumberOfNodesOnRoute == 1)

{

Wait\_Time = 0.0;

}

}

Total\_Wait\_Time = Total\_Wait\_Time + Wait\_Time;

cout << " Wait Time: " << Wait\_Time << " " << " Total Wait Time: " << Total\_Wait\_Time;

}

else

{

flag == true;

ode);

NotSelectedNodes[Not\_Selected\_Nodes\_Counter].set\_Node(min\_wait\_n

Not\_Selected\_Nodes\_Counter++;

cout << endl << " Node Status: Not Selected ||";

}

if ((time\_distance + D[LastNode][min\_wait\_node] + Service\_Time + Wait\_Time + D[min\_wait\_node][0] <= MyNodesInformation[0].get\_Late()) &&

(time\_distance + D[LastNode][min\_wait\_node] <= MyNodesInformation[min\_wait\_node].get\_Late()))

{

MinWait[NumberOfNodesOnRoute].set\_Node(min\_wait\_node); cout << endl << "Node Status: Selected ||"; NumberOfNodesOnRoute += 1;

time\_distance = time\_distance + D[LastNode][min\_wait\_node]

+ Service\_Time + Wait\_Time;

LastNode = min\_wait\_node;

}

else

{

if (flag == false)

{

ode);

NotSelectedNodes[Not\_Selected\_Nodes\_Counter].set\_Node(min\_wait\_n

Not\_Selected\_Nodes\_Counter++;

cout << endl << " Node Status: Not Selected

||";

}

}

RoutesNotSatisfied.erase(RoutesNotSatisfied.begin() + min\_wait\_local\_place);

non\_zero\_elements -= 1;

cout << " Current Time: " << time\_distance << endl; if (non\_zero\_elements > 0)

{

MinWaitTimeServedFirst(RoutesNotSatisfied);

}

else

{

}

}

MinWait[NumberOfNodesOnRoute + 1].set\_Node(0);

// CASE 4 - Min Demand Nodes First void MinDemandFirstServed(vector<NodesInfo>)

{

bool flag = false;

double min\_demand\_local = 99999.99; int min\_demand\_local\_place = -1; int min\_demand\_node = -1;

for (i = 0; i <= non\_zero\_elements; i++)

{

if (RoutesNotSatisfied[i].get\_Node() != 0)

{

int Demand = MyNodesInformation[RoutesNotSatisfied[i].get\_Node()].get\_Demand();

if (Demand < min\_demand\_local)

// check mellontika an prepei na einai k to 0 (depot) mesa

{ // pernwntas apo to depot gia kapoia apostash mporei na einai mikrotero to D[][]

min\_demand\_local = Demand;

// minimum distance

min\_demand\_node = RoutesNotSatisfied[i].get\_Node();

min\_demand\_local\_place = i;

// node's place on vector "RoutesNotSatisfied" with the minimum distance

}

}

}

cout << NumberOfNodesOnRoute << ") min demand: " << min\_demand\_local << " node: " << RoutesNotSatisfied[min\_demand\_local\_place].get\_Node();

double Late\_TW = MyNodesInformation[min\_demand\_node].get\_Late();

// Late Time Window of closest node double Service\_Time =

MyNodesInformation[min\_demand\_node].get\_Service(); double Early\_TW =

MyNodesInformation[min\_demand\_node].get\_Early(); double Wait\_Time = 0.0;

if (D[LastNode][min\_demand\_node] <= Late\_TW) // min\_distance <= Late TW of closest node

{

if (time\_distance < Early\_TW)

{

- time\_distance;

Wait\_Time = Early\_TW - D[LastNode][min\_demand\_node]

if (Wait\_Time < 0)

{

Wait\_Time = 0.0;

}

if (NumberOfNodesOnRoute == 1)

{

Wait\_Time = 0.0;

}

}

Total\_Wait\_Time = Total\_Wait\_Time + Wait\_Time;

cout << " Wait Time: " << Wait\_Time << " " << " Total Wait Time: " << Total\_Wait\_Time;

}

else

{

flag == true;

NotSelectedNodes[Not\_Selected\_Nodes\_Counter].set\_Node(min\_demand

\_node);

Not\_Selected\_Nodes\_Counter++;

cout << endl << " Node Status: Not Selected ||";

}

if ((time\_distance + D[LastNode][min\_demand\_node] + Service\_Time

+ Wait\_Time + D[min\_demand\_node][0] <=

MyNodesInformation[0].get\_Late()) &&

(time\_distance + D[LastNode][min\_demand\_node] <= MyNodesInformation[min\_demand\_node].get\_Late()))

{

MinDemand[NumberOfNodesOnRoute].set\_Node(min\_demand\_node); cout << endl << "Node Status: Selected ||"; NumberOfNodesOnRoute += 1;

time\_distance = time\_distance + D[LastNode][min\_demand\_node] + Service\_Time + Wait\_Time;

LastNode = min\_demand\_node;

}

else

{

if (flag == false)

{

NotSelectedNodes[Not\_Selected\_Nodes\_Counter].set\_Node(min\_demand

\_node);

||";

}

}

Not\_Selected\_Nodes\_Counter++;

cout << endl << " Node Status: Not Selected

RoutesNotSatisfied.erase(RoutesNotSatisfied.begin() + min\_demand\_local\_place);

non\_zero\_elements -= 1;

cout << " Current Time: " << time\_distance << endl;

if (non\_zero\_elements > 0)

{

}

else

{

}

}

MinDemandFirstServed(RoutesNotSatisfied);

MinDemand[NumberOfNodesOnRoute + 1].set\_Node(0);

// CASE 5 - Max Demand Nodes First void MaxDemandFirstServed(vector<NodesInfo>)

{

bool flag = false;

double max\_demand\_local = -99999.99; int max\_demand\_local\_place = -1;

int max\_demand\_node = -1;

for (i = 0; i <= non\_zero\_elements; i++)

{

if (RoutesNotSatisfied[i].get\_Node() != 0)

{

int Demand = MyNodesInformation[RoutesNotSatisfied[i].get\_Node()].get\_Demand();

if (Demand > max\_demand\_local)

{

max\_demand\_local = Demand; max\_demand\_node =

RoutesNotSatisfied[i].get\_Node();

max\_demand\_local\_place = i;

}

}

}

cout << NumberOfNodesOnRoute << ") max demand: " << max\_demand\_local << " node: " << RoutesNotSatisfied[max\_demand\_local\_place].get\_Node();

double Late\_TW = MyNodesInformation[max\_demand\_node].get\_Late();

// Late Time Window of closest node double Service\_Time =

MyNodesInformation[max\_demand\_node].get\_Service(); double Early\_TW =

MyNodesInformation[max\_demand\_node].get\_Early(); double Wait\_Time = 0.0;

if (D[LastNode][max\_demand\_node] <= Late\_TW) // min\_distance <= Late TW of closest node

{

if (time\_distance < Early\_TW)

{

- time\_distance;

Wait\_Time = Early\_TW - D[LastNode][max\_demand\_node]

if (Wait\_Time < 0)

{

Wait\_Time = 0.0;

}

if (NumberOfNodesOnRoute == 1)

{

Wait\_Time = 0.0;

}

}

Total\_Wait\_Time = Total\_Wait\_Time + Wait\_Time;

cout << " Wait Time: " << Wait\_Time << " " << " Total Wait Time: " << Total\_Wait\_Time;

}

else

{

flag == true;

NotSelectedNodes[Not\_Selected\_Nodes\_Counter].set\_Node(max\_demand

\_node);

Not\_Selected\_Nodes\_Counter++;

cout << endl << " Node Status: Not Selected ||";

}

if ((time\_distance + D[LastNode][max\_demand\_node] + Service\_Time

+ Wait\_Time + D[max\_demand\_node][0] <=

MyNodesInformation[0].get\_Late()) &&

(time\_distance + D[LastNode][max\_demand\_node] <= MyNodesInformation[max\_demand\_node].get\_Late()))

{

MaxDemand[NumberOfNodesOnRoute].set\_Node(max\_demand\_node); cout << endl << "Node Status: Selected ||"; NumberOfNodesOnRoute += 1;

time\_distance = time\_distance + D[LastNode][max\_demand\_node] + Service\_Time + Wait\_Time;

LastNode = max\_demand\_node;

}

else

{

if (flag == false)

{

NotSelectedNodes[Not\_Selected\_Nodes\_Counter].set\_Node(max\_demand

\_node);

||";

}

}

Not\_Selected\_Nodes\_Counter++;

cout << endl << " Node Status: Not Selected

RoutesNotSatisfied.erase(RoutesNotSatisfied.begin() + max\_demand\_local\_place);

non\_zero\_elements -= 1;

cout << " Current Time: " << time\_distance << endl; if (non\_zero\_elements > 0)

{

}

else

{

}

}

MaxDemandFirstServed(RoutesNotSatisfied);

MaxDemand[NumberOfNodesOnRoute + 1].set\_Node(0);

void fillVectorRouteNull(vector<NodesInfo>& newNodesVector)

{

int in\_node;

cout << endl << "Setting zeroes" << endl; // Open file ifstream infile;

infile.open("NULL\_Route.txt");

if (infile.fail()) //check for error

{

cerr << "Error Opening File" << endl; exit(1);

}

for (i = 0; i < Nodes ; i++)

{

infile >> in\_node; NodesInfo Node; Node.set\_Node(in\_node);

newNodesVector.push\_back(Node);

}

}// end void fillVectorRouteNull

void printVectorRouteNull(const vector<NodesInfo>& newNodesVector) {

unsigned int size = newNodesVector.size(); cout << endl;

for (unsigned int i = 0; i < size; i++)

{

cout << newNodesVector[i].get\_Node() << " " ;

}

cout << endl << endl;

} // end void printVectorRouteNull

void fillVectorFile(vector<NodesInfo>& newNodesVector)

{

int in\_node, in\_x, in\_y, in\_demand, in\_early, in\_late, in\_service;

cout << "display RC201.100.txt contents" << endl; // Open file ifstream infile;

infile.open("RC201.100.txt");

if (infile.fail()) //check for error

{

cerr << "Error Opening File" << endl; exit(1);

}

int count = 0;

for (i = 0; i < N; i++) //read file, every 7 elements changes line, which means that node changes too

{

infile >> in\_node >> in\_x >> in\_y >> in\_demand >> in\_early

>> in\_late >> in\_service;

NodesInfo Node1(in\_node, in\_x, in\_y, in\_demand, in\_early, in\_late, in\_service);

count += 1; // counter for node number newNodesVector.push\_back(Node1);

}

cout << "Access on file achieved." << endl<< count << " Nodes found in file." << endl;

}

void printVector(const vector<NodesInfo>& newNodesVector) { unsigned int size = N;

for (unsigned int i = 0; i < size; i++)

{

cout << "Node: " << newNodesVector[i].get\_Node() << " "<<

* X: " << newNodesVector[i].get\_X()

<<

* Y: " << newNodesVector[i].get\_Y()

<<

* Demand: " <<

newNodesVector[i].get\_Demand() <<

* Early: " << newNodesVector[i].get\_Early() <<
* Late: " << newNodesVector[i].get\_Late() <<
* Service: " << newNodesVector[i].get\_Service() << endl;

}

}

void ValidationCheckOfDemand(const vector<NodesInfo>& newNodesVector)

{

unsigned int count = 0; unsigned int size = N;

cout << endl << endl << endl << size << endl << endl; for (unsigned int i = 0; i < size; i++)

{

if (newNodesVector[i].get\_Demand() > Q) {

cout << i << " node's demand, exceeds vehicle's capacity." << endl;

validation++;

}

count = count + newNodesVector[i].get\_Demand();

}

if (count > Q\*k)

{

cout << endl <<"Overal demand exceeds the demand that can be coverd with " << k << " vehicles." << endl;

validation++;

}

else

{

}

}

cout << endl << "Demand values approved, continue.";

void CalculateAllNodesDistance(const vector<NodesInfo>& newNodesVector)

{

double x1, x2, y1, y2; double max = -1; double min = 100000;

for(unsigned int i = 0; i < N; i++)

{

x1 = newNodesVector[i].get\_X(); y1 = newNodesVector[i].get\_Y();

for (unsigned int j = 0; j < N; j++)

{

x2 = newNodesVector[j].get\_X(); y2 = newNodesVector[j].get\_Y();

double distancex = (x1 - x2)\*(x1 - x2); double distancey = (y1 - y2)\*(y1 - y2);

double calcdist = sqrt(abs(distancex + distancey)); D[i][j] = calcdist;

}

}

}

void ReadClusterFile()

{

file

cout << "display sample.txt contents" << endl; // Open

ifstream infile; infile.open("sample.txt");

if (infile.fail()) //check for error

{

cerr << "Error Opening File" << endl; exit(1);

}

int item; infile >> item; R[0][0] = item; Counter\_i = 1;

Counter\_j = 0;

while (!infile.eof())

{

infile >> item; if (item > 0)

{

R[Counter\_j][Counter\_i] = item; Counter\_i++;

}

else

{

}

}

}

Counter\_j++; Counter\_i = 1;

R[Counter\_j][Counter\_i] = item;

void LoadDataSet()

{

ifstream infile; if (DataSet == 1)

{

cout << endl << endl << "display Case\_25.540.1.txt contents" << endl; // Open file

infile.open("Case\_25.540.1.txt");

}

else if (DataSet ==2)

{

cout << endl << endl << "display Case\_25.280.2.txt contents" << endl; // Open file

infile.open("Case\_25.280.2.txt");

}

else if (DataSet == 3)

{

cout << endl << endl << "display Case\_25.200.3.txt contents" << endl; // Open file

infile.open("Case\_25.200.3.txt");

}

else if (DataSet == 4)

{

cout << endl << endl << "display Case\_25.160.4.txt contents" << endl; // Open file

infile.open("Case\_25.160.4.txt");

}

else if (DataSet == 5)

{

cout << endl << endl << "display Case\_50.3.340.txt contents" << endl; // Open file

infile.open("Case\_50.3.340.txt");

}

else if (DataSet == 6)

{

cout << endl << endl << "display Case\_50.4.260.txt contents" << endl; // Open file

infile.open("Case\_50.4.260.txt");

}

else if (DataSet == 7)

{

cout << endl << endl << "display Case\_50.5.220.txt contents" << endl; // Open file

infile.open("Case\_50.5.220.txt");

}

else if (DataSet == 8)

{

cout << endl << endl << "display Case\_50.6.190.txt contents" << endl; // Open file

infile.open("Case\_50.6.190.txt");

}

else if (DataSet == 9)

{

cout << endl << endl << "display Case\_100.7.280.txt contents" << endl; // Open file

infile.open("Case\_100.7.280.txt");

}

else if (DataSet == 10)

{

cout << endl << endl << "display Case\_100.8.240.txt contents" << endl; // Open file

infile.open("Case\_100.8.240.txt");

}

else if (DataSet == 11)

{

cout << endl << endl << "display Case\_100.9.220.txt contents" << endl; // Open file

infile.open("Case\_100.9.220.txt");

}

else

{

cout << endl << endl << "display Case\_100.10.200.txt

contents" << endl; // Open file infile.open("Case\_100.10.200.txt");

}

if (infile.fail()) //check for error

{

cerr << "Error Opening File" << endl; exit(1);

}

// read items int item; infile >> item; R[0][0] = item; int max\_i = 0; int max\_j = 0; Counter\_i = 1;

Counter\_j = 0;

while (!infile.eof())

{

infile >> item; if (item > 0)

{

R[Counter\_j][Counter\_i] = item;

Counter\_i++;

if (Counter\_i > max\_i)

{

max\_i = Counter\_i;

}

}

else

{

Counter\_j++; Counter\_i = 1;

R[Counter\_j][Counter\_i] = item;

}

}

File\_j = max\_i; File\_i = Counter\_j+1;

}

void DataSetsCall()

{

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| cout << "Select Data Set:" << endl | | | | | | | | | |
| 1 | " << | endl | << | "1 | - Nodes: | 25 | - Capacity: | 540 | - Clusters: |
| 2 | " << | endl | << | "2 | - Nodes: | 25 | - Capacity: | 280 | - Clusters: |
|  |  |  | << | "3 | - Nodes: | 25 | - Capacity: | 200 | - Clusters: |
| 3 | " << | endl | << | "4 | - Nodes: | 25 | - Capacity: | 160 | - Clusters: |
| 4 | " << | endl | << | "5 | - Nodes: | 50 | - Capacity: | 340 | - Clusters: |
| 3 | " << | endl |  |  |  |  |  |  |  |
| 4 | " << | endl | << | "6 | - Nodes: | 50 | - Capacity: | 260 | - Clusters: |
| 5 | " << | endl | << | "7 | - Nodes: | 50 | - Capacity: | 220 | - Clusters: |
| 6 | " << | endl | << | "8 | - Nodes: | 50 | - Capacity: | 190 | - Clusters: |
|  | << "9  Clusters: 7 | | | | - Nodes: 100 - Capacity:  " << endl | | | | 280 - |

Clusters: 10 " << endl << endl; cin >> DataSet;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| << "10 - Nodes:  Clusters: 8 " << endl | 100 | - Capacity: | 240 | - |
| << "11 - Nodes: | 100 | - Capacity: | 220 | - |
| Clusters: 9 " << endl |  |  |  |  |
| << "12 - Nodes: | 100 | - Capacity: | 200 | - |

while ((DataSet != 1) && (DataSet != 2) && (DataSet != 3) && (DataSet != 4) && (DataSet != 5) && (DataSet != 6)

&& (DataSet != 7) && (DataSet != 8) && (DataSet != 9) && (DataSet != 10) && (DataSet != 11) && (DataSet != 12) )

{

cout << "Invalid Value. Try again...";

cin >> DataSet;

}

220;}

200;}

}

if (DataSet == 1)

{

N = 26; k = 1; C = 1; Q = 540;}

else if (DataSet == 2)

{

N = 26; k = 2; C = 2; Q = 280;}

else if (DataSet == 3)

{

N = 26; k = 3; C = 3; Q = 200;}

else if (DataSet == 4)

{

N = 26; k = 4; C = 4; Q = 160;}

else if (DataSet == 5)

{

N = 51; k = 3; C = 3; Q = 340;}

else if (DataSet == 6)

{

N = 51; k = 4; C = 4; Q = 260;}

else if (DataSet == 7)

{

N = 51; k = 5; C = 5; Q = 220;}

else if (DataSet == 8)

{

N = 51; k = 6; C = 6; Q = 190;}

else if (DataSet == 9)

{

N = 101; k = 7; C = 7; Q = 280;}

else if (DataSet == 10)

{

N = 101; k = 8; C = 8; Q = 240;}

else if (DataSet == 11)

{ N = 101; k = 9; C = 9; Q =

else

{ N = 101; k = 10; C = 10; Q =

**// HEADERS – PHASE TWO – VEHICLE ROUTING**

#ifndef \_NODEINFO\_H #define \_NODEINFO\_H #define Nodes 101 #include <math.h> class NodesInfo { private:

// Member Variables int newNode;

int newX; int newY;

int newDemand; int newEarly; int newLate; int newService;

public:

// Default Constructor NodesInfo();

// Overload Constructor

NodesInfo(int, int, int, int, int, int, int);

// Destructor

~NodesInfo();

// Accessor Functions

int get\_Node() const; // returns Node number int get\_X() const; // returns Node X coords

int get\_Y() const; // returns Node Y coords int get\_Demand() const; // returns Node Demands int get\_Early() const; // returns Node Early TW int get\_Late() const; // returns Node Late TW

int get\_Service() const; // returns Node Service time double get\_DistanceTo(NodesInfo OtherNode) {

double tempX = (newX - OtherNode.get\_X()); double tempY = (newY - OtherNode.get\_Y()); return sqrt((tempX\*tempX) + (tempY\*tempY));

}; // returns distance to another node

//Mutator Functions void set\_Node(int); void set\_X(int); void set\_Y(int);

void set\_Demand(int); void set\_Early(int); void set\_Late(int); void set\_Service(int);

};

#endif //\_NODEINFO\_H

#include "NodesInfo.h" #define Nodes 101

// Default Constructor NodesInfo::NodesInfo() {

newNode = 0;

newX = 0;

newY = 0;

newDemand = 0;

newEarly = 0;

newLate = 0;

newService = 0;

}

// Overload Constructor

NodesInfo::NodesInfo(int in\_node, int in\_x, int in\_y, int in\_demand, int in\_early, int in\_late, int in\_service)

{

newNode = in\_node; newX = in\_x;

newY = in\_y;

newDemand = in\_demand; newEarly = in\_early; newLate = in\_late; newService = in\_service;

}

// Destructor NodesInfo::~NodesInfo() { }

// Accessor Functions

int NodesInfo::get\_Node() const { return newNode;} int NodesInfo::get\_X() const { return newX;}

int NodesInfo::get\_Y() const { return newY;}

int NodesInfo::get\_Demand() const { return newDemand;} int NodesInfo::get\_Early() const {return newEarly;} int NodesInfo::get\_Late() const {return newLate;}

int NodesInfo::get\_Service() const {return newService;}

// Mutator Functions

void NodesInfo::set\_Node(int in\_node) { newNode = in\_node; } void NodesInfo::set\_X(int in\_x) { newX = in\_x; }

void NodesInfo::set\_Y(int in\_y) { newY = in\_y; }

void NodesInfo::set\_Demand(int in\_demand) { newDemand = in\_demand; }

void NodesInfo::set\_Early(int in\_early) { newEarly = in\_early; } void NodesInfo::set\_Late(int in\_late) { newLate = in\_late; } void NodesInfo::set\_Service(int in\_service) { newService =

in\_service; }