Πανεπιστήμιο Δυτικής Μακεδονίας

Τμήμα Ηλεκτρολόγων Μηχανικών & Μηχανικών Υπολογιστών

Συγκριτική ανάλυση αλγορίθμων για το πρόβλημα δρομολόγησης οχημάτων

Νικόλαος Θεοχάρης

Επιβλέπων Καθηγητής - Νικόλαος Πλόσκας

Κοζάνη 2019

Περίληψη

Σε αυτή την εργασία θα ασχοληθούμε εκτεταμένα με το Πρόβλημα Δρομολόγησης Οχημάτων με Περιορισμούς Χωρητικότητας (Capacitated Vehicle Routing Problem). Πρόκειται για ένα πρόβλημα που ανήκει στη κατηγορία προβλημάτων NP – Hard, τα οποία και θεωρούνται ως τα πιο δύσκολα προβλήματα Μη Πολυωνυμικού Χρόνου.

Το Πρόβλημα Δρομολόγησης Οχημάτων, είναι ένα κοινότυπο πρόβλημα στον τομέα των μεταφορών όλων των ειδών. Αντιμετωπίζεται από όλες τις μεταφορικές εταιρείες και παρουσιάζει εξαιρετική ποικιλομορφία. Είναι εξίσου πολλοί οι παράγοντες που μπορούν να οδηγήσουν σε αυτή τη ποικιλομορφία, όπως ο χρόνος, ο χώρος, οι πελάτες και πολλά άλλα. Στη συνέχεια της εργασίας θα αναλύσουμε πως οι παράγοντες που αναφέραμε επηρεάζουν τα διάφορα είδη του προβλήματος καθώς επίσης και ποια είναι αυτά τα είδη.

Στα πλαίσια της εργασίας θα αναλύσουμε το Πρόβλημα Δρομολόγησης Οχημάτων. Θα παρουσιάσουμε αναλυτικά τα δύο μοντέλα ακέραιου γραμμικού προγραμματισμού τα οποία υλοποιήσαμε σε γλώσσα C. Ως εκ τούτου, για την επίτευξη των καλύτερων αποτελεσμάτων χρησιμοποιήσαμε τον λύτη CPLEX. Τέλος, θα κάνουμε δοκιμές στα δύο μοντέλα προκειμένου να τα συγκρίνουμε μεταξύ τους, όσον αφορά τους χρόνους εκτέλεσης και τις απαιτήσεις του κάθε ενός μοντέλου, με σκοπό να καταλήξουμε στα γενικά συμπεράσματα μας.

Λέξεις κλειδιά: Πρόβλημα Δρομολόγησης Οχημάτων, Ακέραιος Γραμμικός Προγραμματισμός, CPLEX, Διάγραμμα Αποφάσεων

Abstract

In this work we will study the Capacitated Vehicle Routing Problem. This is a problem belonging to the category NP - Hard problems, which are considered as the most difficult problems of Non-Polynomial Time.

The Vehicle Routing Problem is a common transport problem of all kinds. It is handled by all shipping companies and exhibits exceptional diversity. There are just as many factors that can lead to this diversity, such as time, space, customers and much more. In the following section, we will analyze how the factors we mentioned affect the different types of problem as well as what these types are.

As part of this work we will analyze the Vehicle Routing Problem. We will present in detail the two models of integer programming that we implemented in C. Therefore, we used the CPLEX solver to achieve the best results. Finally, we will test the two models in order to compare each other in terms of execution times and requirements of each model, in order to arrive at our general conclusions.

Keywords: Vehicle Routing Problem, Integer Programming, CPLEX, Decision Diagram

Δήλωση Πνευματικών Δικαιωμάτων

Υπογραφή Φοιτητή

Περιεχόμενα

Κατάλογος σχημάτων

Κατάλογος πινάκων

Κεφάλαιο 1

Εισαγωγή

1.1 Τί πραγματεύεται η εργασία

Σε αυτή την εργασία θα ασχοληθούμε με τη σύγκριση δύο μοντέλων προκειμένου να βρούμε τη βέλτιστη λύση για ένα από τα μεγαλύτερα προβλήματα στον τομέα της μεταφοράς προϊόντων, το Πρόβλημα Δρομολόγησης Οχημάτων (Vehicle Routing Problem). Κύρια προβλήματα στην μεταφορά προϊόντων με οχήματα είναι ο χρόνος, η απόσταση και η κάλυψη των απαιτήσεων με αρκετά οχήματα. Ο χρόνος είναι αποτέλεσμα της συνολικής απόστασης που θα καλύψουν συνολικά όλα τα οχήματα. Προφανώς, όμως, όταν μία εταιρία έχει στον στόλο της περισσότερα οχήματα μπορεί να ελαχιστοποιεί το χρόνο ολοκλήρωσης των απαιτήσεων.

Με λίγα λόγια, στην εργασία θα ασχοληθούμε με την λήψη αποφάσεων για την κατάλληλη διαδρομή που θα πρέπει να ακολουθήσουν τα διαθέσιμα οχήματα μίας μεταφορικής, έτσι ώστε να καλυφθεί η ζήτηση όλων των πόλεων/πελατών στο μικρότερο δυνατό χρόνο. Κάτι τέτοιο μπορεί να επιτευχθεί μόνο όταν οι διαδρομές είναι οι μικρότερες δυνατές και δεν χρειάζεται ο ανεφοδιασμός των οχημάτων, αρκεί πάντα να έχουμε αρκετά οχήματα στη διάθεσή μας για να επιτύχουμε κάτι τέτοιο. Επιπλέον, θα αναλύσουμε εκτενώς τα εργαλεία που θα χρειαστούμε για την υλοποίηση αυτού του εγχειρήματος, καθώς επίσης θα δούμε αναλυτικά τα αποτελέσματα της σύγκρισης δύο διαφορετικών μοντέλων ακέραιου γραμμικού προγραμματισμού.

Θα χρησιμοποιήσουμε, λοιπόν, δύο διαφορετικά μοντέλα ακέραιου γραμμικού προγραμματισμού. Το πρώτο, είναι ένα κλασσικό μοντέλο που χρησιμοποιείται για τέτοιου είδους προβλήματα. Το δεύτερο είναι βασισμένο σε διαγράμματα αποφάσεων, μια διαφορετική προσέγγιση για να διαπιστώσουμε αν μπορούμε με αυτή τη μέθοδο να βγάλουμε καλύτερα αποτελέσματα. Τα δύο μοντέλα είναι παρόμοια στο τομέα του υπολογισμού της τιμής των μεταβλητών, αλλά παρουσιάζουν διαφορές στην δημιουργία των διαθέσιμων διαδρομών και στην αναπαράσταση του γραφήματος. Τέλος, προκειμένου να ληφθούν οι κατάλληλες αποφάσεις θα επιλύσουμε τα μοντέλα με έναν λύτη προβλημάτων ακέραιου γραμμικού προγραμματισμού.

1.2 Ποιος είναι ο σκοπός της εργασίας

Στην εργασία αυτή αποσκοπούμε στην σύγκριση δύο μοντέλων για την εύρεση της βέλτιστης λύσης για το Πρόβλημα Δρομολόγησης Οχημάτων. Στόχος είναι να ανακαλύψουμε εάν η δεύτερη προσέγγιση την οποία δημιουργήσαμε από την αρχή θα είναι ανταγωνιστική από άποψη χρόνου εύρεσης λύσης. Τα συμπεράσματα που θα βγάλουμε θα βοηθήσουν να καταλάβουμε εάν το δικό μας μοντέλο είναι ανταγωνιστικό σε σύγκριση με το πρώτο στο χρόνο εύρεσης βέλτιστης λύσης αλλά και στους απαιτούμενους πόρους για την μοντελοποίηση του προβλήματος.

Κάθε μοντέλο έχει τις δικές του ιδιαιτερότητες. Συνήθως, ο αριθμός των περιορισμών καθορίζει το πόσος χρόνος απαιτείται για την επεξεργασία όλων των στοιχείων και την εύρεση της λύσης. Στόχος μας είναι να δημιουργήσουμε ένα μοντέλο που θα μπορεί να ανταπεξέλθει στο μέγεθος των πειραμάτων που θα τρέξουμε, χωρίς να ξεφεύγει σε απαιτήσεις από ένα απλό μοντέλο. Επιπλέον, η δέσμευση πόρων μίας υπολογιστικής μονάδας είναι χαρακτηριστικό μίας καλής μοντελοποίησης. Ο βασικός παράγοντας που επηρεάζει το μέγεθος των πόρων που καταναλώνει ένα μοντέλο, είναι ο αριθμός των περιορισμών που πρέπει να λάβουμε υπόψιν προκειμένου να έχουμε το επιθυμητό αποτέλεσμα.

Επομένως, σε αυτή την εργασία επιχειρούμε να μελετήσουμε το Πρόβλημα Δρομολόγησης Οχημάτων, να το ερμηνεύσουμε ως πρόβλημα ακεραίου γραμμικού προγραμματισμού και να δημιουργήσουμε ένα μοντέλο βασισμένο στα διαγράμματα αποφάσεων.

1.3 Τι θα δούμε στην εργασία

Στο Κεφάλαιο 2, θα αποκτήσουμε μία γενική εικόνα για το τί είναι το Πρόβλημα Δρομολόγησης Οχημάτων. Πρόκειται να δούμε τις κατηγορίες στις οποίες χωρίζεται το VRP και τα χαρακτηριστικά κάθε κατηγορίας. Στη συνέχεια, θα αναλύσουμε την κατηγορία του VRP με την οποία ασχολούμαστε σε αυτή την εργασία, το CVRP (Capacitated Vehicle Routing Problem) ή αλλιώς το Πρόβλημα Δρομολόγησης Οχημάτων με περιορισμούς χωρητικότητας. Έπειτα, θα παρουσιάσουμε την δουλειά άλλων ερευνητών που ασχολήθηκαν με άλλες μεθόδους επίλυσης προβλημάτων CVRP στο παρελθόν και τέλος θα αναλύσουμε τις εφαρμογές του Προβλήματος Δρομολόγησης Οχημάτων στην καθημερινότητά μας.

Στο Κεφάλαιο 3, θα παρουσιάσουμε αναλυτικά τα δύο μοντέλα. Θα πραγματοποιήσουμε μια γενική περιγραφή των προβλημάτων και θα εξηγήσουμε διεξοδικά τις μεταβλητές και τους περιορισμούς τους. Θα κάνουμε σύγκριση των δύο μοντέλων ως προς τις διαφορές που παρουσιάζουν μεταξύ τους και θα δείξουμε δύο απλά παραδείγματα για να δούμε τη συμπεριφορά κάθε μοντέλου.

Στο 4ο Κεφάλαιο, θα παρουσιάσουμε τα αποτελέσματα των πειραμάτων που τρέξαμε στα πλαίσια των δοκιμών των δύο μοντέλων. Θα δούμε, ποιες είναι οι απαιτήσεις κάθε μοντέλου, δηλαδή πόσους περιορισμούς και μεταβλητές δημιουργεί με βάση το πόσες πόλεις πρέπει να λάβουμε υπόψιν και πόσα οχήματα θα εκτελέσουν τις διαδρομές. Στη συνέχεια, θα επιδείξουμε τα εργαλεία που χρησιμοποιήσαμε και την υπολογιστική δύναμη που είχαμε στη διάθεσή μας για την εκτέλεση των πειραμάτων και τέλος θα αναλύσουμε το πρόγραμμα CPLEX εκτεταμένα βλέποντας τις κύριες συναρτήσεις και λειτουργίες του.

Στο κεφάλαιο 5, θα οδηγηθούμε στα γενικά συμπεράσματα για την έρευνα που εκτελέσαμε και θα παραθέσουμε τις σκέψεις μας για μελλοντικές επεκτάσεις πάνω στο Πρόβλημα Δρομολόγησης Οχημάτων.

Κεφάλαιο 2

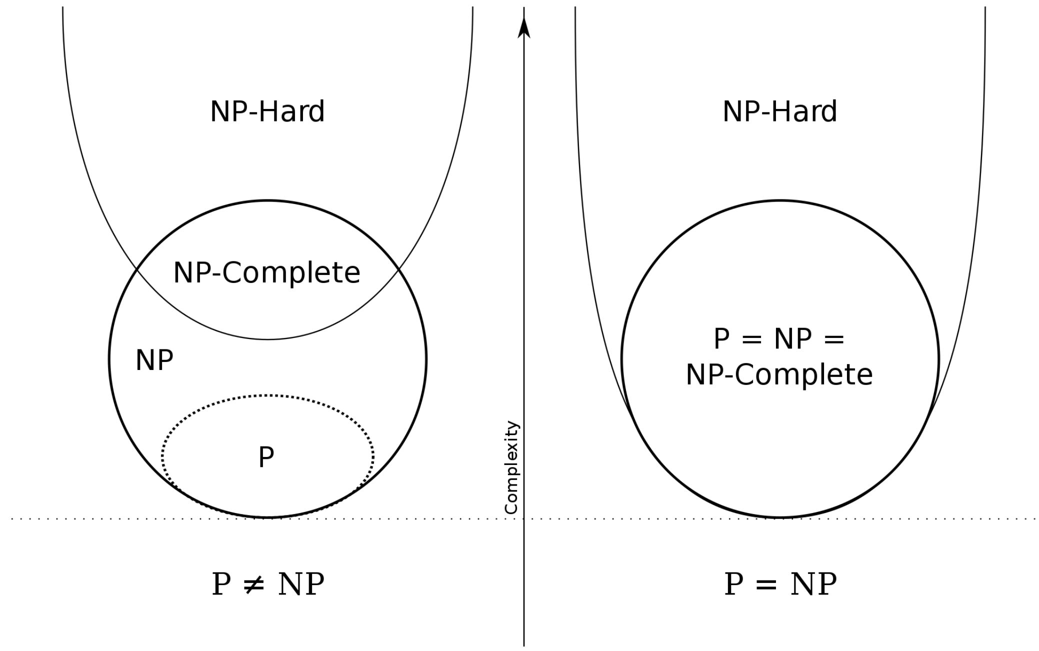
Βιβλιογραφική Ανασκόπηση

Το πρόβλημα δρομολόγησης οχημάτων ανήκει στη κατηγορία προβλημάτων NP – Complete (Non Polynomial). Τα NP – Complete προβλήματα αποτελούν τα δυσκολότερα προβλήματα της κατηγορίας NP. Υπάρχουν τρεις κατηγορίες αλγορίθμων που μπορούν να λύσουν αυτά τα προβλήματα. Οι ακριβείς αλγόριθμοι (exact algorithms) που βρίσκουν την βέλτιστη λύση, οι προσεγγιστικοί αλγόριθμοι (approximation algorithms) που βρίσκουν μία προσεγγιστική λύση η οποία έρχεται με εγγυήσεις από το πόσο απέχει από την βέλτιστη λύση, και τέλος οι ευρετικές διαδικασίες (heuristics) που βρίσκουν μία λύση χωρίς εγγυήσεις από το πόσο απέχει από την βέλτιστη λύση.

Πιο συγκεκριμένα, κάθε είσοδος στο πρόβλημα πρέπει να συσχετίζεται με ένα σύνολο λύσεων πολυωνυμικού μήκους, των οποίων η εγκυρότητα μπορεί να δοκιμαστεί σε πολυωνυμικό χρόνο, έτσι ώστε η έξοδος για κάθε είσοδο να είναι ”ναι” αν το σύνολο των λύσεων δεν είναι άδειο και ”όχι” αν είναι άδειο. Η τάξη πολυπλοκότητας των προβλημάτων αυτής της μορφής ονομάζεται NP, δηλαδή non polynomial(μη ντετερμινιστικού πολυωνυμικού χρόνου).Ένα πρόβλημα μπορεί να κατονομαστεί ως NP – Hard αν όλα μέσα στο NP πρόβλημα μπορούν να υπολογιστούν σε πολυωνυμικό χρόνο. Τα προβλήματα που είναι ταυτοχρόνως NP και NP – Hard, ονομάζονται NP – Complete. Αν καταφέρουμε να λύσουμε έστω ένα από τα NP – Complete προβλήματα εντός πολυωνυμικού χρόνου, τότε όλα τα προβλήματα στην κλάση NP θα λύνονται σε αυτόν τον χρόνο.

Στο σχήμα 2.1 βλέπουμε τον συσχετισμό μεταξύ των προβλημάτων NP – Complete και NP – Hard. Το σχήμα αναπαριστά ένα διάγραμμα Euler για P (polynomial), NP, NP – Complete και NP – Hard προβλήματα. Στην αριστερή μεριά του σχήματος η πραγμάτευση είναι έγκυρη εφόσον ισχύει $P \neq NP$, ενώ η δεξιά μεριά του άξονα πολυπλοκότητας ισχύει μόνο εάν $P = NP$.

Σχήμα 2.1: Διάγραμμα Euler για P, NP, NP-Complete και NP-Hard προβλήματα[1]



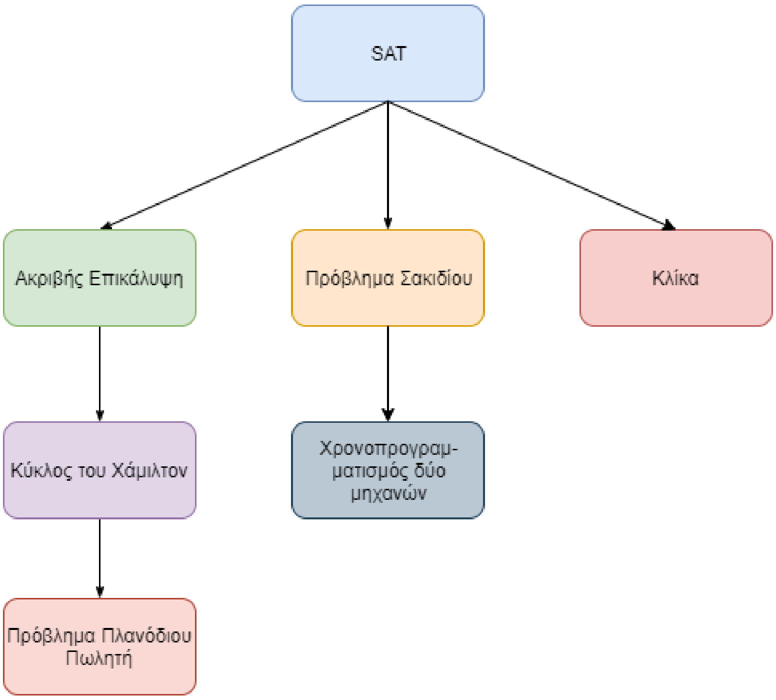
Παρά το γεγονός ότι, αυτά τα NP – Complete προβλήματα δεν καταφέρνουν να μας δώσουν μία απόδειξη ότι $P \neq NP$, καταλαμβάνουν σημαντικό χώρο στη μελέτη της Πολυπλοκότητας. Για παράδειγμα, αν υποθέσουμε ότι $P \neq NP$ – μια εικασία που είναι ευρέως αποδεκτή, αν και ακόμα κανείς δεν κατάφερε να το αποδείξει – τότε όλα τα NP – Complete προβλήματα δεν ανήκουν στη κατηγορία προβλημάτων P. Αυτή η κατά κάποιο τρόπο έμμεση ένδειξη δυσκολίας είναι το περισσότερο που μπορούμε να περιμένουμε για ένα πρόβλημα στο NP, ανεπαρκές για την απόδειξη ότι $P \neq NP$.

Υπάρχουν πολλά προβλήματα που ανήκουν στην κατηγορία NP – Complete. Μερικά από αυτά είναι το πρόβλημα Ικανοποιησιμότητας (SAT), το πρόβλημα Ακριβούς Επικάλυψης (Exact Cover Problem), ο κύκλος Χάμιλτον (Hamilton's Circle), το πρόβλημα του Πλανόδιου Πωλητή (Traveling Salesman Problem), το πρόβλημα το Σακιδίου (Knapsack Problem), ο χρονοπρογραμματισμός δύο μηχανών (Two-Machine Scheduling) και το πρόβλημα της κλίκας (Clique Problem). Στο σχήμα 2.2 , απεικονίζονται τα προβλήματα που προαναφέρθηκαν και αναπαρίσταται ο μεταξύ τους συσχετισμός. Υπάρχουν ακόμη πολλά περισσότερα προβλήματα τα

[1]https://en.wikipedia.org/wiki/NP-completeness

οποία ανήκουν στη κατηγορία NP – Complete, αλλά αυτά θεωρούμε ότι είναι τα πιο άξια αναφοράς.

Σχήμα 2.2: Συσχέτιση Μεταξύ NP-Complete Προβλημάτων[2]



2.1 Το Πρόβλημα Δρομολόγησης Οχημάτων

Το γενικό Πρόβλημα Δρομολόγησης Οχημάτων (Golden, Raghavan, & Wasil, 2008) (Toth & Vigo, 2002) είναι το κύριο πρόβλημα στον τομέα των μεταφορών παντός τύπου. Από την μεταφορά προϊόντων, με ένα ή περισσότερα οχήματα, προς διάφορους προορισμούς έως την παραλαβή ασθενών με ένα ασθενοφόρο και τη μεταφορά τους στο πλησιέστερο νοσοκομείο.

Το VRP είναι ένα πρόβλημα που υπάρχει σαν ιδέα εδώ και 60 χρόνια. Οι Dantzig και Ramser (Dantzig & Ramser, 1959) παρουσίασαν το πρόβλημα το 1959, προτείνοντας το πρώτο μαθηματικό μοντέλο και την αλγοριθμική προσέγγιση για την επίλυση του. Στήριξαν την σημαντικότητα του VRP, χρησιμοποιώντας ως παράδειγμα την μεταφορά βενζίνης στα διάφορα πρατήρια. Έπειτα από 5 χρόνια, οι Clarke και Wright (Clarke & Wright, 1964) πρότειναν ένα πιο αποτελεσματικό άπληστο ευριστικό αλγόριθμο, ο οποίος αποδείχτηκε καλύτερος από αυτόν των Dantzig και Ramser.

[2] https://en.wikipedia.org/wiki/NP-completeness

Πιο συγκεκριμένα, το VRP είναι το γενικό πρόβλημα που ασχολείται με την βελτίωση των μεταφορών όσον αφορά το χρόνο που χρειάζονται τα οχήματα για να ολοκληρώσουν τις παραδόσεις. Στόχος του είναι να βρεθεί η κατάλληλη διαδρομή, έτσι ώστε να μειωθεί στο ελάχιστο ο χρόνος κατά τον οποίο τα οχήματα επισκέπτονται όλους τους σταθμούς και επιστρέφουν στον αρχικό, ενώ ταυτόχρονα πρέπει να αποφευχθούν περιπτώσεις όπου δύο οχήματα επισκέπτονται την ίδια πόλη. Βέβαια, ο αποκλεισμός επαναεπισκέψεων είναι απαραίτητος σε ορισμένες μόνο περιπτώσεις του VRP, τις οποίες θα δούμε στη συνέχεια.

Δύο από τα πιο σημαντικά προβλήματα που αποτελούν μίξη του VRP είναι το πρόβλημα του πλανόδιου πωλητή (TSP) και το πρόβλημα του Bin Packing(BPP)(Korf, 2002).

- **Πρόβλημα του πλανόδιου πωλητή (TSP)**: Το πρόβλημα του πλανόδιου πωλητή είναι το πιο απλό και εύκολα κατανοητό πρόβλημα δρομολόγησης, καθώς δεν διαθέτει ιδιαίτερους περιορισμούς. Πιο αναλυτικά, στο πρόβλημα του πλανόδιου πωλητή, όπως υποδηλώνει το όνομα, έχουμε μόνο ένα όχημα/πωλητή ο οποίος πρέπει να μεταφέρει όλα τα προϊόντα σε όλες τις πόλεις/πελάτες. Αυτό που κάνει το πρόβλημα ακόμη πιο απλό, είναι η ελευθερία που μας δίνεται στη ποσότητα προϊόντων που μπορεί να διαθέσει ο πωλητής, αφού στο TSP ο πωλητής μπορεί να μεταφέρει άπειρο αριθμό προϊόντων, με αποτέλεσμα να μην τίθεται θέμα ανεφοδιασμού.

- **Πρόβλημα Bin Packing (BPP)**: Στο πρόβλημα Bin Packing, βρισκόμαστε αντιμέτωποι με την δυσκολία τοποθέτησης ενός ορισμένου αριθμού αντικειμένων ποικίλου μεγέθους, σε έναν εξίσου ορισμένο αριθμό κάδων(bins), οι οποίοι όμως έχουν όλοι ίδια χωρητικότητα. Ένα απλό παράδειγμα, είναι να υποθέσουμε ότι μία μεταφορική έχει συγκεκριμένο μέγεθος στα προϊόντα που μεταφέρει και δύο οχήματα στη διάθεσή της, επομένως πρέπει να τα μοιράσει στα δύο, προκειμένου να μπορέσει να τα μεταφέρει όλα. Το BPP, μπορεί να αλλάξει σημαντικά το μέγεθος της δυσκολίας ενός VRP προβλήματος, εάν ληφθεί υπόψιν. Συνήθως όμως, στα περισσότερα προβλήματα ο παράγοντας αυτός δεν συνυπολογίζεται.

2.2 Περιπτώσεις Προβλήματος Δρομολόγησης Οχημάτων

Το Πρόβλημα Δρομολόγησης Οχημάτων χωρίζεται σε εξαιρετικά πολλές και διαφορετικές εκδοχές. Κάθε περιορισμός που μπορεί να αφαιρεθεί ή να προστεθεί, μπορεί να δημιουργήσει ένα νέο είδος VRP. Στη συνέχεια της εργασίας, θα αναλύσουμε ορισμένες περιπτώσεις του VRP στις οποίες οι ερευνητές έχουν αφιερώσει περισσότερο χρόνο ερευνώντας τες. Για ορισμένες από αυτές τις περιπτώσεις, έχουν προταθεί και εξετασθεί πολλές μικρές αλλαγές που θα μπορούσαν να κάνουν τη διαφορά, παρόλα αυτά τα ονόματα των περιπτώσεων παραμένουν τα ίδια.

Προκειμένου να δημιουργηθούν οι διάφορες εκδοχές του VRP, θα πρέπει να λάβουμε υπόψιν τις διαφοροποιήσεις που μπορεί να υπάρχουν στις απαιτήσεις των πελατών, στις δυνατότητες των οχημάτων καθώς επίσης και στο γενικό στόχο της λύσης. Πιο αναλυτικά, τα κύρια χαρακτηριστικά των πελατών είναι:

- Το μέρος που βρίσκεται ο πελάτης στο χάρτη,
- Οι απαιτήσεις του πελάτη σε ποσότητα, ή και σε είδος προϊόντων, τα οποία το όχημα θα παραδώσει ή θα παραλάβει,
- Περίοδοι μέσα στην ημέρα κατά τις οποίες κάθε πελάτης είναι διαθέσιμος για επίσκεψη,
- Ο χρόνος που χρειάζεται για την φόρτωση και εκφόρτωση των προϊόντων από κάθε σταθμό.

Αυτοί είναι οι περιορισμοί που πρέπει να λάβει υπόψιν ο αλγόριθμος, όσον αφορά τις απαιτήσεις των πελατών. Ένα άλλος παράγοντας που μπορεί να επηρεάσει τις διάφορες περιπτώσεις του VRP, είναι τα χαρακτηριστικά των οχημάτων. Τα οχήματα και τα διάφορα χαρακτηριστικά τους, δημιουργούν νέους περιορισμούς στο σχεδιασμό ενός μοντέλου. Πιο συγκεκριμένα, οι παράγοντες που επηρεάζονται από τα οχήματα είναι:

- Ο σταθμός εφοδιασμού, όπως επίσης και ο τερματικός σταθμός στις περιπτώσεις που αυτοί οι δύο δεν αποτελούν το ίδιο μέρος,

- Η χωρητικότητα του οχήματος, η οποία μετράται σε μέγιστο όγκο, βάρος ή και αριθμό παλετών που μπορεί να μεταφέρει ένα όχημα,
- Η δυνατότητα μεταφοράς όλων των ειδών προϊόντων από ίδια οχήματα (δηλαδή προϊόντα που χρειάζονται ψυγείο και άλλα που δεν χρειάζονται),
- Ο διαθέσιμος εξοπλισμός κάθε οχήματος που θα επιταχύνει ή θα επιβραδύνει τη διαδικασία φόρτωσης και εκφόρτωσης,
- Η δυνατότητα ενός οχήματος να ακολουθήσει μία διαδρομή βάση του μεγέθους του και,
- Το κόστος που απαιτεί το όχημα για να ολοκληρώσει ένα δρομολόγιο (δηλαδή το κόστος κατανάλωσης καυσίμου).

Όλα τα παραπάνω, επιβαρύνουν με περιορισμούς τον αλγόριθμο που θα λάβει την απόφαση για την δημιουργία των βέλτιστων δρομολογίων. Ταυτόχρονα, συμβάλλουν στη δημιουργία ακόμη περισσότερων περιπτώσεων στις οποίες χωρίζεται το VRP. Τέλος, άλλος ένας παράγοντας που επηρεάζει αυτή τη ποικιλία, είναι οι επιθυμητοί στόχοι που θέτει κάθε μοντέλο προς επίτευξη. Αναλυτικότερα, οι στόχοι αυτοί είναι:

- Η ελαχιστοποίηση του συνολικού κόστους των μεταφορών, ανάλογα με την απόσταση που θα διανύσουν τα οχήματα, όπως επίσης και σε άλλα κόστη που μπορεί να σχετίζονται με τα οχήματα,
- Η ελαχιστοποίηση του αριθμού οχημάτων που θα χρειαστούν για την ολοκλήρωση της παράδοσης όλων των προϊόντων,
- Η ισορροπία μεταξύ των διαδρομών που θα ακολουθήσουν τα οχήματα καθώς επίσης και του φορτίου που καλούνται να μεταφέρουν και
- Η ελαχιστοποίηση των ποινών που μπορεί να συνδεθούν με την μερική εξυπηρέτηση που μπορεί να προσφέρουν οι πελάτες.

Αναλογιζόμενοι όλα τα παραπάνω, είμαστε σε θέση να κατανοήσουμε πλήρως ότι, ο αριθμός των περιπτώσεων που οδηγούν σε αυτή την ποικιλότητα του VRP δικαιολογείται, αφού θα ήταν εξαιρετικά δύσκολο να ληφθούν υπόψιν όλοι αυτοί οι παράγοντες σε ένα γενικό πρόβλημα. Παρακάτω θα αναλύσουμε τις πιο αξιοσημείωτες και ευρέως διαδεδομένες περιπτώσεις του VRP:

- **VRP με χρονικά περιθώρια (VRPTW)**: Σε αυτή την περίπτωση του Προβλήματος Δρομολόγησης Οχημάτων, ο χρόνος είναι αυτός που επηρεάζει τις αποφάσεις του αλγόριθμου για το σύνολο των δρομολογίων που θα ακολουθήσουν τα οχήματα. Πιο αναλυτικά, στο VRPTW (Kallehauge, Larsen, Madsen, & Solomon, 2005), οι πελάτες ορίζουν στα οχήματα ένα χρονικό διάστημα εντός του οποίου θα πρέπει να έχει ολοκληρωθεί η παράδοση των προϊόντων σε αυτούς. Αυτό έχει ως αποτέλεσμα, να δημιουργούνται δυσκολίες στην εύρεση βέλτιστης λύσης από τον αλγόριθμο, αφού κύριο μέλημα των οχημάτων είναι να καλυφθεί η ζήτηση μέσα στα χρονικά περιθώρια που έχουν τεθεί από την αρχή.

- **VRP με ανεφοδιασμούς (VRPB)**: Στη περίπτωση αυτή έχουμε δύο κατηγορίες σταθμών. Από την μία μεριά υπάρχουν οι σταθμοί, όπου τα διαθέσιμα οχήματα παραδίδουν προϊόντα ίσα σε αριθμό με τη ζήτηση του κάθε πελάτη. Από την άλλη πλευρά, υπάρχουν και οι σταθμοί από τους οποίους τα οχήματα παραλαμβάνουν επιπλέον προϊόντα τα οποία θα παραδώσουν εκ νέου στους πελάτες. Δημιουργείται έτσι μία αλυσίδα μεταφορών, που δίνει τη δυνατότητα στον αλγόριθμο να χρησιμοποιήσει μικρότερο αριθμό οχημάτων αφού, είναι σίγουρο ότι ποτέ δεν θα υπάρξει πρόβλημα με την διάθεση προϊόντων.

- **VRP με περιορισμό αποστάσεων (DCVRP)**: Αυτή είναι μια ιδιαίτερη εκδοχή του VRP. Σε αυτή τη περίπτωση, τα οχήματα έχουν σαν κύριο περιορισμό την απόσταση που μπορούν να διανύσουν μεταξύ δύο πόλεων/πελατών. Ως αποτέλεσμα, οι διαδρομές καθορίζονται κυρίως από το αν ένα όχημα έχει τη δυνατότητα να καλύψει την απόσταση μίας διαδρομής βάση των περιορισμών που ισχύουν. Ανάλογα με αυτούς τους περιορισμούς, ορίζονται και τα δρομολόγια όλων των οχημάτων που έχει στη διάθεσή της μία εταιρεία.

- **VRP πολλαπλών αφετηριών (MDVRP)**: Αποτελεί μία αρκετά ρεαλιστική προσέγγιση του VRP. Με λίγα λόγια, σε αντίθεση με τις άλλες περιπτώσεις όπου όλα τα οχήματα ξεκινούσαν από το ίδιο μέρος, σ' αυτή τη περίπτωση υπάρχουν διάφοροι σταθμοί ανεφοδιασμού από τους οποίους αναχωρούν διαφορετικά οχήματα. Έτσι, ο στόχος του VRP παραμένει ίδιος, δηλαδή πρέπει να επισκεφθούν τα οχήματα όλους του πελάτες ελαχιστοποιώντας τον χρόνο και να επιστρέφουν στον ίδιο σταθμό από τον οποίο ξεκίνησαν.

- **VRP με στοιχεία παραλαβής και μεταφοράς (VRPPD)**: Το VRPPD (Desaulniers & Groupe d'études et de recherche en analyse des décisions (Montréal, 2000) είναι το γενικό πρόβλημα όλων των μεταφορικών που ασχολούνται με μετακομίσεις. Αναλυτικότερα, κάθε αίτηση σε αυτό το πρόβλημα έχει δικό του σημείο φόρτωσης, διαφορετικό σημείο εκφόρτωσης και ποικίλες απαιτήσεις στον όγκο αντικειμένων που πρέπει να τοποθετηθούν στο διαθέσιμο χώρο κάθε οχήματος. Επομένως, σε αυτή την εκδοχή του VRP υπάρχουν αρκετά προβλήματα να λυθούν, όπως το να χωρέσουν τα προϊόντα/αντικείμενα/πελάτες σε όλα τα διαθέσιμα οχήματα και να βρεθεί η κατάλληλη διαδρομή για την ταχύτερη μεταφορά.
- **VRP με διαμοιρασμένη μεταφορά (SDVRP)**: Σε ορισμένες περιπτώσεις είναι πιο συμφέρον ένας σταθμός να λάβει τα προϊόντα από διάφορα οχήματα. Στις περιπτώσεις που συμβαίνει αυτό, οι απαιτήσεις του σταθμού σε προϊόντα είναι ίσες με τη μέγιστη χωρητικότητα ενός οχήματος, και προκειμένου να ελαχιστοποιηθεί η απόσταση, ο σταθμός ανεφοδιάζεται από δύο ή και περισσότερα οχήματα με κάποια αναλογία 60-40, 50-50 κλπ.

Όλα τα παραπάνω, είναι περιπτώσεις που δεν θα μας απασχολήσουν περαιτέρω. Στη συνέχεια, θα μιλήσουμε αναλυτικά για την περίπτωση του VRP με την οποία ασχολούμαστε σε αυτή την εργασία, το CVRP.

2.3 VRP με περιορισμό χωρητικότητας

Σε αυτή την εργασία ασχολούμαστε με μία ιδιαίτερη περίπτωση του VRP. Αυτή η περίπτωση ονομάζεται CVRP και βρισκόμαστε αντιμέτωποι με το πρόβλημα του χώρου. Πιο αναλυτικά, στη διάθεσή μας έχουμε ένα συγκεκριμένο αριθμό οχημάτων και στόχος είναι η κάλυψη των απαιτήσεων όλων των πόλεων/πελατών, στον καλύτερο δυνατό χρόνο, ελαχιστοποιώντας την συνολική απόσταση που θα διανύσουν όλα τα οχήματα. Πρέπει εδώ να γίνει σαφές ότι όλα τα οχήματα που βρίσκονται στη διάθεση μας είναι παρόμοια, επομένως ο διαθέσιμος χώρος για τα προϊόντα που θα μεταφερθούν είναι ίδιος για όλα τα οχήματα. Βέβαια, το πρόβλημα «χώρος», μπορεί να αντιμετωπιστεί με δύο απλές μεθόδους:

- **Διάθεση περισσότερων οχημάτων**: Σε ορισμένες περιπτώσεις έχουμε τη δυνατότητα, να χρησιμοποιήσουμε αρκετά οχήματα, ώστε η ζήτηση να καλυφθεί επαρκώς. Αυτή η περίπτωση μας δίνει την ικανότητα να χωρίσουμε τα δρομολόγια με τον καλύτερο δυνατό τρόπο προκειμένου να έχουμε το ελάχιστο κόστος απόστασης.
- **Δυνατότητα ανεφοδιασμών**: Σε ορισμένα προβλήματα, μας ζητείται να βρούμε την βέλτιστη διαδρομή που θα ακολουθήσουν τα οχήματα για να ολοκληρώσουν τη μεταφορά των προϊόντων, δίνοντας μας όμως σαν δεδομένο ότι η συνολική χωρητικότητα των διαθέσιμων οχημάτων δεν επαρκεί για να καλύψει τη ζήτηση απευθείας. Έτσι, ένα ή περισσότερα από τα οχήματα θα πρέπει να επιστρέψουν στο σταθμό ανεφοδιασμού και να παραλάβουν τα υπόλοιπα προϊόντα που απομένουν να παραδοθούν. Η διαδικασία του ανεφοδιασμού μπορεί να επαναληφθεί και παραπάνω από μία φορά, ανάλογα με το μέγεθος της ζήτησης που περιγράφει το πρόβλημα.

Στην εργασία αυτή ασχολούμαστε με την πρώτη περίπτωση. Σε όλα τα πειράματα ο αριθμός των οχημάτων και η χωρητικότητα τους είναι επαρκής για να καλύψει όλη τη ζήτηση των πόλεων/πελατών. Με άλλα λόγια, τα οχήματα σε καμία περίπτωση δεν επιστρέφουν στην πόλη εφοδιασμού και είναι πάντα αρκετά σε αριθμό, προκειμένου αυτό να εξασφαλιστεί.

2.4 Βιβλιογραφική Ανασκόπηση

Έχουν περάσει 60 χρόνια από τη στιγμή που οι Dantzig και Ramser (Dantzig & Ramser, 1959), συστήσανε στο κόσμο το Πρόβλημα Δρομολόγησης Οχημάτων. Προκειμένου να τονίσουν τη σημαντικότητά του, χρησιμοποίησαν ως παράδειγμα την ανάγκη για μεταφορά πετρελαίου στα βενζινάδικα. Προτείνανε τη πρώτη αλγοριθμική προσέγγιση και ένα μαθηματικό μοντέλο προγραμματισμού για την επίλυση αυτού του προβλήματος. Έπειτα από χρόνια, συγκεκριμένα το 1965, οι Clarke και Wright (Clarke & Wright, 1964), παρουσιάσανε έναν αποτελεσματικό ευριστικό αλγόριθμο, που βελτίωσε την προσέγγιση των Dantzig και Ramser.

Από εκείνη τη στιγμή και μετά, πολλοί είναι αυτοί που έχουν προτείνει και άλλους ευριστικούς ή και άπληστους αλγόριθμους για την εύρεση της βέλτιστης και πιο ακριβής λύση των διάφορων περιπτώσεων του Προβλήματος Δρομολόγησης Οχημάτων. Υπάρχουν πολλά άρθρα που έχουν δημοσιευτεί για το VRP. Οι Laporte και Nobert (Laporte & Nobert, 1987) παρουσίασαν μία εκτενή έρευνα που ήταν εξ ολοκλήρου αφιερωμένη στις ακριβείς μεθόδους επίλυσης για VRP προβλήματα, και έδωσαν μία πλήρη και τεκμηριωμένη ανάλυση για τη κατάσταση των πραγμάτων κοντά στη δεκαετία του ‘80. Από εκεί και πέρα κι άλλοι εκδώσανε ακριβείς αλγόριθμους για προβλήματα VRP αλλά συνήθως με ευριστικές μεθόδους. Στη συνέχεια, θα αναλύσουμε εκτενώς τις διάφορες μεθόδους επίλυσης των Προβλημάτων Δρομολόγησης Οχημάτων.

- **Branch and Bound**: Η μέθοδος branch-and-bound χρησιμοποιήθηκε πολλές φορές τις τελευταίες δεκαετίες για την λύση Προβλημάτων Δρομολόγησης Οχημάτων και των διάφορων παραλλαγών του. Τις περισσότερες φορές, η λύση απευθυνόταν σε Α συμμετρικά Προβλήματα Δρομολόγησης Οχημάτων με περιορισμούς χωρητικότητας (ACVRP) και CVRP με περιορισμό αποστάσεων. Παρόλα αυτά, οι αλγόριθμοι αυτοί ακόμη αποτελούν έργο τέχνης. Οι Laporte, Mercure και Nobert (Laporte, Mercure, & Nobert, 1986) ήταν αυτοί που πρότειναν τον πρώτο αλγόριθμο branch-and-bound για προβλήματα ACVRP. Όπως αναφέρθηκε και στην ενότητα 2.1, το CVRP είναι παράγωγο του Προβλήματος του Πλανόδιου Πωλητή, ζητώντας τον προσδιορισμό του ελάχιστου κόστους επίσκεψης όλων των κόμβων, ακριβώς μία φορά, σε ένα Κύκλο Χάμιλτον. Από εκεί και πέρα, με την εκτεταμένη και επιτυχή δουλειά που έγινε για την εύρεση ακριβούς λύσης για το πρόβλημα TSP, προέκυψαν πολλές ακριβείς προσεγγίσεις για το CVRP. Μέχρι τα τέλη της δεκαετίας το ’80, οι πιο αποτελεσματικές ακριβείς προσεγγίσεις για το CVRP ήταν κυρίως branch-and-bound αλγόριθμοι. Τα τελευταία χρόνια, προτάθηκαν πιο εκλεπτυσμένα όρια (bounds), βασισμένα στις εξισώσεις Lagrance, οι οποίες αυξήσαν σε μέγεθος τα προβλήματα, των οποίων η βέλτιστη λύση μπορεί να βρεθεί με branch-and-bound αλγόριθμους. Οι πιο πρόσφατοι branch-and-bound αλγόριθμοι που μπορούσαν να λύσουν μικρού μεγέθους προβλήματα προτάθηκαν από τους Christofides, Mingozzi και Toth (Christofides, Mingozzi, & Toth, 1981).

- **Branch and Cut**: Αυτή η μέθοδος είναι πολύ επιτυχημένη στο να βρίσκει τη βέλτιστη λύση σε μεγάλα προβλήματα που συσχετίζονται με το Συμμετρικό Πρόβλημα Πλανόδιου Πωλητή (STSP). Βέβαια, το μέγεθος της έρευνας που έχει πραγματοποιηθεί για την εύρεση λύσης σε προβλήματα CVRP με αυτή τη μέθοδο δεν συγκρίνεται σε καμία περίπτωση με την έρευνα που έχει γίνει για τα STSP προβλήματα. Το μέγεθος της έρευνας που έχει γίνει σε αλγόριθμους branch-and-cut για προβλήματα CVRP, είναι πολύ μικρό και δεν υπάρχουν αρκετές δημοσιεύσεις. Παρόλα αυτά, σύμφωνα με τους Caprara και Fischetti (Caprara & Fischetti, 1997), οι αλγόριθμοι branch-and-cut έχουν μεγάλο ποσοστό επιτυχίας στην εύρεση λύσης για συνδυαστικά προβλήματα βελτιστοποίησης, αλλά μπορεί να έχουν και πολύ χαμηλή απόδοση.

- **Heuristics**: Οι Clarke and Wright (Clarke & Wright, 1964) έχουν δημιουργήσει ίσως τους πιο ευρέως διαδεδομένους ευριστικούς αλγόριθμους για VRP προβλήματα. Πολλά είδη ευριστικών αλγορίθμων έχουν προταθεί για την λύση VRP προβλημάτων. Οι δύο κύριες κατηγορίες είναι οι κλασικοί ευριστικοί και οι μεθευριστικοί, τους οποίους θα αναλύσουμε στη συνέχεια. Οι περισσότερες έρευνες σε κλασικούς ευριστικούς αλγόριθμους έγιναν μεταξύ των δεκαετιών '60 και '90. Οι αλγόριθμοι αυτοί, πραγματοποιούν μία σχετικά περιορισμένη διερεύνηση για εύρεση λύσης και σε συνήθως βρίσκουν ικανοποιητικές λύσεις σε λογικούς υπολογιστικούς χρόνους. Μερικοί ευριστικοί αλγόριθμοι έχουν την δυνατότητα να προσαρμοστούν στο μέγεθος πραγματικών προβλημάτων και ταυτοχρόνως να βρίσκουν λύση αρκετά ικανοποιητική. Για αυτό το λόγο είναι ιδιαίτερα διαδεδομένοι στο εμπόριο. Οι περισσότεροι από τους αλγόριθμους που σχεδιάστηκαν για VRP προβλήματα μπορούν να χρησιμοποιηθούν απευθείας και για CVRP προβλήματα. Επιπλέον, μπορούν να επεκταθούν και για περιπτώσεις που μπορεί να υπάρχουν περιορισμοί στη μέγιστη απόσταση που μπορεί να καλυφθεί από τα οχήματα (DCVRP), ακόμη και αν αυτό δεν γίνεται ξεκάθαρο από την αρχή του προβλήματος. Οι περισσότεροι ευριστικοί αλγόριθμοι μπορούν να λειτουργήσουν με απροσδιόριστο αριθμό οχημάτων, αλλά υπάρχουν εξαιρέσεις σε αυτό τον κανόνα.

- **Metaheuristics**: Τα τελευταία χρόνια έχουν ερευνηθεί πολλοί μεθευριστικοί αλγόριθμοι για Προβλήματα Δρομολόγησης Οχημάτων. Ιδιαίτερα αξιόλογη είναι η έρευνα των Gendreau, Laporte, και Potvin (Gendreau, 1997). Στους μεθευριστικούς αλγόριθμους, δίνεται έμφαση στην εκτεταμένη έρευνα στις πιο πολλά υποσχόμενες περιοχές για την εύρεση βέλτιστης λύσης. Αυτοί οι αλγόριθμοι, συνήθως συνδυάζουν εξελιγμένους κανόνες αναζήτησης γειτονικών κόμβων, δομές μνήμης και συνεχείς συνδυασμούς λύσεων. Η ποιότητα των λύσεων αυτών των αλγόριθμων είναι πολύ καλύτερη από τις λύσεις των κλασικών ευριστικών αλγόριθμων. Ως αντίτιμο αυτής της ποιότητας, οι μεθευριστικοί αλγόριθμοι ξοδεύουν περισσότερο υπολογιστικό χρόνο για την εύρεση λύσης. Επιπλέον, οι διαδικασίες που ακολουθούν αυτοί οι αλγόριθμοι, συνήθως χρειάζονται πολύ λεπτό συντονισμό των παραμέτρων, ο οποίος μπορεί να καταστήσει την επέκταση τους σε άλλες καταστάσεις ιδιαίτερα δύσκολη. Σε αντίθεση με τους κλασικούς ευριστικούς αλγόριθμους, οι μεθευριστικοί, επιτρέπουν την διερεύνηση και μη εφικτών λύσεων κατά τη διαδικασία εύρεσης λύσης. Οι πιο γνωστοί μεθευριστικοί αλγόριθμοι που έχουν αναπτυχθεί για VRP προβλήματα, μπορούν να εντοπίσουν καλύτερες τοπικές βέλτιστες λύσεις από τους κλασικούς ευριστικούς. Παρόλα αυτά, αυτό τους καθιστά αρκετά χρονοβόρους.

2.5 Εφαρμογές του VRP στην καθημερινή ζωή

Το VRP είναι ένα από τα πιο συνηθισμένα προβλήματα στο τομέα των μεταφορών. Εφαρμογές του Προβλήματος Δρομολόγησης Οχημάτων μπορούμε να βρούμε στη μεταφορά προϊόντων ακόμη και ανθρώπων σε ορισμένες περιπτώσεις. Υπάρχουν είδη μεταφορών όπου δίνεται η δυνατότητα να γίνει μεταφορά πολλών μονάδων με ένα όχημα ενώ ταυτόχρονα σε άλλα είδη, η μεταφορά γίνεται σε ατομικό βαθμό. Στη συνέχεια, θα δούμε μερικές από τις πιο γνωστές κατηγορίες μεταφορών που υπάρχουν μέχρι σήμερα. Θα τις αναλύσουμε προκειμένου να καταλάβουμε πόσο διαφορετικές είναι μεταξύ τους κάποιες περιπτώσεις και ποιοι είναι οι παράγοντες που οδηγούν σε αυτή τη διαφορετικότητά τους. Πιο αναλυτικά υπάρχουν:

- **Γενικού τύπου μεταφορικές**: Γενικού τύπου μεταφορικές: Ο ανεφοδιασμός όλων των ειδών καταστημάτων γίνεται κυρίως από φορτηγά. Συνήθως, υπάρχουν φορτηγά τα οποία είναι αρμόδια για την μεταφορά προϊόντων σε διάφορους προορισμούς και όχι απαραίτητα σε ένα. Τα φορτηγά αυτά, τις περισσότερες φορές θα πρέπει να εφοδιάσουν του πελάτες με τα προϊόντα πριν να αρχίσουν οι ώρες λειτουργίας, επομένως ο χρόνος παίζει μεγάλο ρόλο στην ομαλή διεκπεραίωση της μεταφοράς. Οι οδηγοί, πρέπει να είναι σε θέση να γνωρίζουν, ποια είναι η καλύτερη διαδρομή που πρέπει να ακολουθήσουν με σκοπό να ολοκληρώσουν την μεταφορά στον ταχύτερο δυνατό χρόνο και πριν το χρονικό όριο που καθορίζουν οι πελάτες. Τέλος, σε αυτή τη περίπτωση τα οχήματα είναι σχεδιασμένα με συγκεκριμένο αριθμό χωρητικότητας, με αποτέλεσμα η επαναλαμβανόμενη φόρτωση του οχήματος να αποτελεί ένα πιθανό παράγοντα επιβράδυνσης της διαδικασίας.

- **Ασθενοφόρα**: Τα ασθενοφόρα διαφέρουν από τις κοινές μεταφορικές. Στην περίπτωση του ασθενοφόρου, το όχημα ξεκινάει από τον αρχικό σταθμό (Νοσοκομείο), κατευθύνεται προς τον ασθενή και επιστρέφει ξανά στο σταθμό. Οδηγούμαστε λοιπόν στο συμπέρασμα ότι το ασθενοφόρο έχει πάντα ίδιο σταθμό αναχώρησης και ίδιο σταθμό επιστροφής. Παρόλα αυτά το ασθενοφόρο, κάθε φορά που ξεκινάει ένα δρομολόγιο έχει διαφορετικό σταθμό παραλαβής ενός ασθενή. Επιπλέον, ως γνωστόν τα ασθενοφόρα έχουν χώρο μόνο για έναν βαριά τραυματισμένο άνθρωπο. Επομένως, στη περίπτωση αυτή, αντιμετωπίζουμε μία διαφορετική διαχείριση των μεταφορών. Πιο αναλυτικά, για την σωστή και ταχύτατη εξυπηρέτηση των ασθενών, τα ασθενοφόρα καλούνται, πέρα από το να ακολουθήσουν την ταχύτατη δυνατή διαδρομή, να συντονιστούν σωστά ανάλογα με τη διαθεσιμότητα και το χρόνο που χρειάζεται το καθένα για να ολοκληρώσει την μεταφορά.

- **Ταξί**: Τα ταξί, αποτελούν μία παρόμοια περίπτωση με τα ασθενοφόρα. Φυσικά, τα ταξί μπορούν να μεταφέρουν έως και τέσσερις πελάτες αλλά χωρίς να μεταφέρει το καθένα σε διαφορετικό προορισμό. Επιπλέον, τα ταξί έχουν πάντα διαφορετικό σταθμό αναχώρησης, διαφορετικό σταθμό παραλαβής πελατών και διαφορετικό σταθμό άφιξης. Βέβαια, εδώ το αποτέλεσμα μοιάζει με αλυσίδα, αφού κάθε φορά ο σταθμός άφιξης ενός δρομολογίου και σταθμός αναχώρησης του επόμενου είναι ιδανικά ο ίδιος. Στη περίπτωση των ταξί, στόχος είναι η ταχύτατη εξυπηρέτηση των πελατών που θα οδηγήσει στην αύξηση των δρομολογίων σε μία μέρα. Πιο συγκεκριμένα, τα ταξί έχουν ως κύριο μέλημα να καταφέρουν να βγάλουν περισσότερα κέρδη από ότι χρέη για τα συνολικά δρομολόγια. Αν, όπως προαναφέραμε, δεν έχουμε την ιδανική περίπτωση όπου τα δρομολόγια αποτελούν αλυσίδα μεταξύ τους, τότε πρέπει να γίνει όσο το δυνατόν ταχύτερη ανάληψη πελατών για την αύξηση του δείκτη κέρδη/κόστη.

- **Διανομή φαγητού**: Σε αυτού του είδους τις περιπτώσεις, ο χρόνος είναι πολύ σημαντικός για τους εμπόρους. Συνήθως, οι εταιρείες έχουν μόνο ένα όχημα αρμόδιο για τις μεταφορές και ο διαθέσιμος χώρος για τα προϊόντα είναι πολύ μικρός. Επομένως, ο διανομέας πρέπει να επιλέξει τις διαδρομές κατάλληλα, έτσι ώστε όλα τα δρομολόγια να χρειάζονται τον ίδιο χρόνο για να ολοκληρωθούν. Όπως και πριν, να αναφέρουμε και εδώ, ότι ο σταθμός αναχώρησης και ο σταθμός άφιξης είναι ο ίδιος όπως και στη περίπτωση των ασθενοφόρων. Σε αυτή τη περίπτωση αλλάζει σε κάθε δρομολόγιο ο σταθμός που θα μεταφέρει τα προϊόντα ο υπάλληλος. Συνήθως, οι μεταφορείς υπολογίζουν μόνοι τους το μήκος των αποστάσεων που θα ακολουθήσουν και καταλήγουν στο κατά τη γνώμη τους καλύτερο δρομολόγιο χωρίς να είναι ουσιαστικά και το καλύτερο.

- **Οχήματα άμεσης δράσης**: Είναι τα οχήματα που είναι αρμόδια για την βοήθεια όσων τα αυτοκίνητα παρουσιάζουν βλάβη στο δρόμο, σε βαθμό που καθιστά αδύνατη τη μετακίνηση τους. Επιπλέον, είναι τα οχήματα που παραλαμβάνουν δύο αυτοκίνητα μετά από ατυχήματα προκειμένου να μην εμποδίζουν την ομαλή κυκλοφορία των υπολοίπων στο δρόμο. Τα οχήματα άμεσης δράσης, έχουν την δυνατότητα να μεταφέρουν έως και δύο οχήματα. Σε ορισμένες περιπτώσεις λαμβάνονται κλήσεις για δύο περιστατικά σε δύο διαφορετικές περιοχές. Προκειμένου να μη δημιουργηθεί κυκλοφοριακή συμφόρηση, τα οχήματα άμεσης δράσης πρέπει να εκμεταλλευτούν όσο το δυνατόν καλύτερα τις παρακάμψεις των δρόμων και να ακολουθήσουν τις συντομότερες διαδρομές για να παραλάβουν τα οχήματα με βλάβη από την τοποθεσία τους. Όπως και στα ταξί, ο σταθμός αναχώρησης, ο σταθμός από όπου παραλαμβάνονται τα οχήματα και ο σταθμός που θα τα αφήσουν είναι διαφορετικός. Τα οχήματα με τη βλάβη μπορεί να βρίσκονται σε ένα μεγάλο εύρος μίας περιοχής, επομένως το όχημα θα πρέπει να το παραλάβει από εκεί. Στη συνέχεια, ο πελάτης επιλέγει τη περιοχή που θέλει να επιστραφεί το όχημα του. Τέλος, το όχημα εάν λάβει επόμενη κλήση, πρέπει να συνεχίσει, για να παραλάβει τον επόμενο πελάτη κοκ. Φτάνουμε στο συμπέρασμα ότι, οι αποφάσεις που καλείται να πάρει ο οδηγός του οχήματος άμεσης ανάγκης για την βέλτιστη διαδρομή αλλάζουν συνεχώς και πρέπει να είναι, ανά πάσα στιγμή, σε εγρήγορση για την εύρεση του καλύτερου μονοπατιού.

Κεφάλαιο 3

Παρουσίαση Μοντέλων

3.1 Κλασικό Μοντέλο

Έστω ότι έχουμε μία εταιρεία διανομής προϊόντων με $i \in R$ όπου $R = \{1, ..., R\}$, δηλαδή ο αριθμός των πελατών/πόλεων στις οποίες πρέπει να γίνει η διανομή. Η εταιρεία έχει στη διάθεσή της $k \in K$ οχήματα, με $K = \{1, ..., K\}$. Επίσης, έστω ότι κάθε όχημα έχει μία συγκεκριμένη χωρητικότητα προϊόντων, την οποία θα αναπαριστήσουμε με C, και κάθε πόλη έχει το δικό της αριθμό παραγγελιών, τις οποίες θα αναπαριστήσουμε με d_i.

Προκειμένου να αναπαριστήσουμε τις διαδρομές που θα ακολουθήσουν κάθε ένα από τα οχήματα, έχουμε έναν πίνακα $X = \{x_{ijk}\}$ στον οποίο το i αντιπροσωπεύει την πόλη από την οποία ξεκινά ένα όχημα, j την πόλη προορισμού και k το όχημα που κάνει τη συγκεκριμένη διαδρομή. Για τις παραμέτρους που αναπαριστούν τις πόλεις στο συγκεκριμένο συμβολισμό ισχύει $i, j \in V, i \neq j$, με $V = \{0, 1, ..., V\}$, δηλαδή όλες τις πόλεις προορισμού συμπεριλαμβανομένου και της πόλης από την οποία ξεκινάνε και καταλήγουν όλα τα οχήματα. Προκειμένου να γνωρίζουμε ποια πόλη επισκέφθηκε το κάθε όχημα, χρησιμοποιούμε το πίνακα $Y = \{y_{ik}\}$, όπου i είναι η πόλη που επισκέφθηκε το όχημα k.

Στον πίνακα 3.1 , βλέπουμε το πώς διαμορφώνονται τα στοιχεία των πινάκων X και Y. Πιο αναλυτικά, όπως βλέπουμε, έστω ότι στη κάθετη στήλη είναι η πόλη από την οποία ξεκινάει μία διαδρομή το πρώτο όχημα που έχουμε στη διάθεσή μας. Υποθέτοντας ότι το όχημα ξεκινάει από την Πόλη 1 και κατευθύνεται στη Πόλη 2. Ο πίνακας X τότε θα πρέπει να έχει ένα στοιχείο με $i = 1$, $j = 2$ και $k = 1$. Επιπλέον, ο πίνακας Υ θα πρέπει να έχει ένα στοιχείο, το οποίο να αποδεικνύει ότι

Πίνακας 3.1: Στοιχεία πινάκων X και Υ σύμφωνα με το κλασικό μοντέλο

Προς Από	**Πόλη 1**	**Πόλη 2**	**Πόλη 3**	**Επίσκεψη Πρώτου Οχήματος ($\mathbf{y}_{i1}$)**
Πόλη 1	-	x_{121}	x_{131}	y_{11}
Πόλη 2	x_{211}	-	x_{231}	y_{21}
Πόλη 3	x_{311}	x_{321}	-	y_{31}

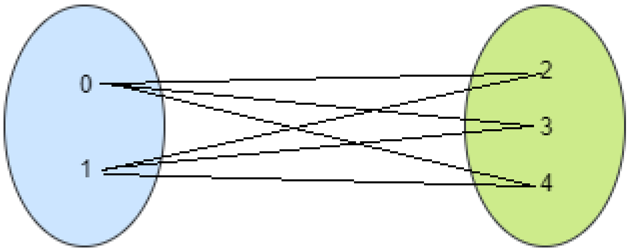
το όχημα 1 επισκέφθηκε την Πόλη 1. Με τον ίδιο τρόπο, όπως φαίνεται στον πίνακα, πρέπει να καλυφθούν όλες οι περιπτώσεις όπου κάθε όχημα ξεκινά από μία πόλη και καταλήγει σε μία άλλη.

Τα στοιχεία των πινάκων X και Y είναι δυαδικές μεταβλητές. Για τα στοιχεία του X, όταν κάποιο έχει την τιμή μηδέν τότε το όχημα k δεν θα πάει από την πόλη i στη πόλη j, ενώ στην αντίθετη περίπτωση, η μεταφορά θα γίνει κανονικά. Παρομοίως, όταν κάποιο στοιχείο του Y πίνακα έχει την τιμή μηδέν, τότε το όχημα k δεν θα επισκεφτεί την πόλης i.

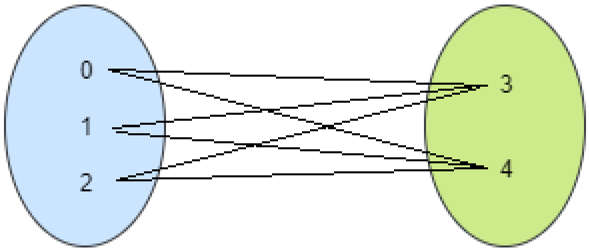
Τέλος, για το περιορισμό που μας αποτρέπει τις επανεπισκέψεις σε μία πόλη θα χρειαστούμε τον πίνακα S, ο οποίος είναι μεταβλητός και ισχύει $S \subseteq R$. Το μέγεθος του πίνακα S, καθορίζεται από τον τύπο $2 \leq |S| \leq |R| - 2$, δηλαδή ο S έχει πάντα τουλάχιστον δύο στοιχεία και το πολύ δύο λιγότερα από τον αριθμό των στοιχείων του πίνακα R.

Για να γίνει πιο κατανοητή η έννοια του πίνακα S, θα προβούμε σε ένα αναλυτικό παράδειγμα. Υποθέτουμε ότι έχουμε ένα περιβάλλον όπου |R|□5, επομένως ο πίνακας S μπορεί να έχει από 2 έως 3 στοιχεία. Οι συνδυασμοί που προκύπτουν υπολογίζονται με τη μέθοδο επιμεριστικής ιδιότητας μεταξύ των στοιχείων του πίνακα S και τον εναπομείναντων στοιχείων. Στο σχήμα 3.1, αναπαριστούμε την πρώτη περίπτωση που προκύπτει σε αυτό το παράδειγμα. Στο πρώτο στάδιο υπολογισμού, ο πίνακας S έχει δύο στοιχεία τα οποία είναι οι πόλεις 0 και 1. Έτσι, από τον επιμερισμό των στοιχείων, προκύπτουν οι συνδυασμοί: 02, 03, 04, 12, 13, 14. Στη συνέχεια, τα στοιχεία του πίνακα S αλλάζουν, και τη θέση του 1 παίρνει το 2, οπότε τα νέα στοιχεία του πίνακα S είναι τα 0 και 2. Επομένως, το 1 μεταφέρεται στην απέναντι μεριά και ξεκινάμε πάλι την δημιουργία συνδυασμών. Όταν έχουμε πλέον ολοκληρώσει όλους τους συνδυασμούς με τον αριθμό των στοιχείων στον πίνακα S να είναι 2, τότε αυξάνουμε τα στοιχεία σε 3 και ξεκινάμε πάλι την αντιστοιχία των αριθμών, όπως φαίνεται στο σχήμα 3.2. Είναι προφανές ότι, σε περίπτωση που ο αριθμός των πόλεων/πελατών είναι μεγαλύτερος, τότε ο μέγιστος αριθμός των στοιχείων του πίνακα S ξεπερνάει τα τρία.

Σχήμα 3.1: Συσχέτιση μεταξύ των στοιχείων του πίνακα S και των υπολοίπων όταν $|S| = 2$



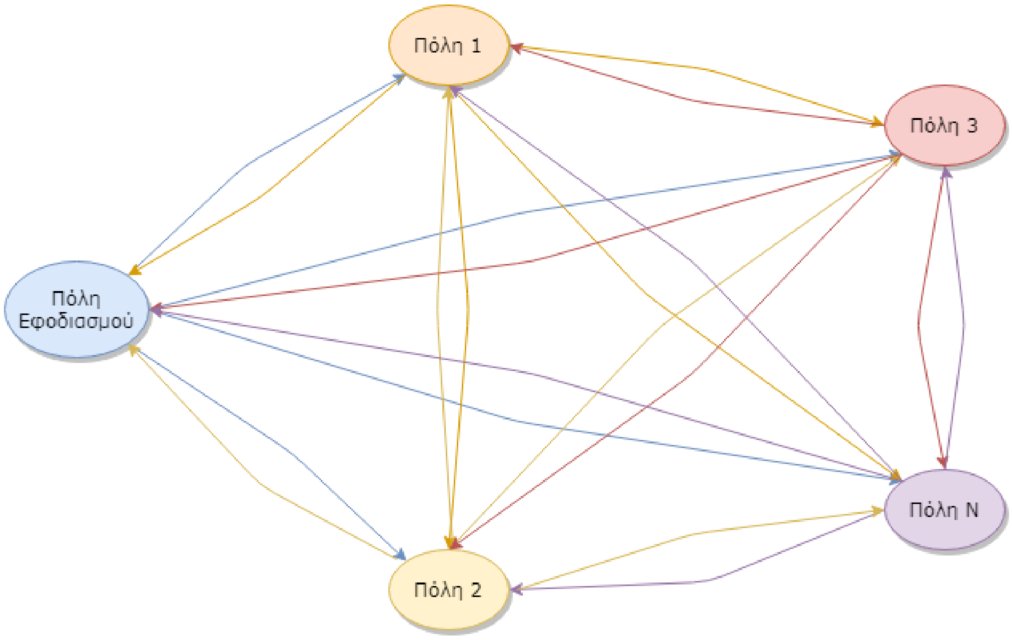
Σχήμα 3.2: Συσχέτιση μεταξύ των στοιχείων του πίνακα S και των υπολοίπων όταν $|S| = 3$



Στο σχήμα 3.3, θα δούμε μια αναπαράσταση από τη δημιουργία ενός υποθετικού διαγράμματος του κλασικού μοντέλου που περιγράφουμε. Πιο συγκεκριμένα, στο σχήμα απεικονίζονται όλες οι πόλεις που πρέπει να επισκεφθούν τα οχήματα μιας εταιρείας. Επομένως, πρέπει να σχηματίσουμε όλες τις διαδρομές από την πόλη ανεφοδιασμού προς όλες τις υπόλοιπες πόλεις, και από κάθε πόλη προς όλες τις υπόλοιπες αντίστοιχα. Έτσι, καταλήγουμε σε ένα σχήμα όπως το παρακάτω. Φυσικά, όταν το μοντέλο αποφασίσει για το ποια είναι η βέλτιστη λύση, άρα και ποιες είναι οι διαδρομές που θα ακολουθήσει το κάθε όχημα, τότε στο σχήμα θα απεικονίζονται μόνο οι διαδρομές που έγιναν από τα οχήματα. Παρακάτω θα δούμε σχηματικά ένα παράδειγμα, προκειμένου να γίνει κατανοητό το πως διαμορφώνεται το διάγραμμα.

Στη συνέχεια, θα δούμε την αντικειμενική συνάρτηση και το μαθηματικό μοντέλο του πρώτου μοντέλου. Σύμφωνα με την αντικειμενική συνάρτηση, γίνεται γνωστό στο πρόγραμμα ποιες είναι οι συνολικές μεταβλητές που θα χρησιμοποιηθούν κατά τη διάρκεια της μοντελοποίησης για τη δημιουργία περιορισμών, όπως επίσης για το ποιο είναι το κόστος κάθε διαδρομής ξεχωριστά. Έπειτα, σύμφωνα με το μαθηματικό μοντέλο που ακολουθεί κάτω από την αντικειμενική συνάρτηση, προκύπτουν οι περιορισμοί, τους οποίους το πρόγραμμα θα επεξεργαστεί

Σχήμα 3.3: Γράφημα κλασσικού μοντέλου για N αριθμό πόλεων



προκειμένου να βγάλει τη βέλτιστη λύση. Παρακάτω βλέπουμε την αντικειμενική συνάρτηση και τους περιορισμούς:

$$\min_z = \sum_{i \in V} \sum_{j \in V} \sum_{k \in K} c_{ij} \cdot x_{ijk} \tag{3.1}$$

$$\forall i \in R : \sum_{k \in K} y_{ik} = 1 \tag{3.2}$$

$$\sum_{k \in K} y_{0k} = K \tag{3.3}$$

$$\forall k \in K, i \in V : \sum_{j \in V} x_{ijk} = \sum_{j \in V} x_{jik} = y_{ik} \tag{3.4}$$

$$\forall k \in K : \sum_{i \in V} d_i \cdot y_{ik} \leq C \tag{3.5}$$

$$\forall S \subseteq R, h \in S, k \in K : \sum_{i \in S} \sum_{i \notin S} x_{ijk} \geq y_{hk} - 1 \tag{3.6}$$

$$\forall i \in V, j \in V : \sum_{k \in K} (x_{ijk} + x_{jik}) \leq 1 \tag{3.7}$$

Πρώτα, παρατηρούμε την αντικειμενική συνάρτηση (3.1) στην οποία, ως c_{ij} ορίζουμε το κόστος σε απόσταση που θα χρειαστεί ένα όχημα k για να ταξιδέψει από την πόλη i στην πόλη j. Ο αριθμός αυτός, προκύπτει από την Ευκλείδεια απόσταση που έχουν μεταξύ τους δύο πόλεις πάνω στο χάρτη, δηλαδή την μεταξύ τους απόσταση σε ευθεία γραμμή. Η αντικειμενική συνάρτηση υποδηλώνει τον κύριο σκοπό του VRP, που δεν είναι άλλος από την ελαχιστοποίηση του κόστους των διαδρομών, δηλαδή την απόσταση που θα διανύσουν τα οχήματα συνολικά.

Ο περιορισμός (3.2), εξασφαλίζει την επίσκεψη ενός μόνο οχήματος k σε κάθε πόλη $i \in R$. Είναι προφανές ότι, αν ένα όχημα από αυτά που βρίσκονται στη διάθεσή μας επισκεφθεί μία πόλη και καλύψει τη ζήτησή της, τότε δεν υπάρχει λόγος για επίσκεψη άλλου οχήματος. Επομένως, ο περιορισμός αυτός αποτρέπει την επίσκεψη δύο ή περισσότερων οχημάτων στην ίδια πόλη.

Όπως είναι λογικό, μία εταιρεία έχει συγκεντρωμένα σε ένα μέρος τα οχήματά της. Το μέρος αυτό δεν μπορεί να είναι άλλο από το σταθμό ανεφοδιασμού. Όλα τα οχήματα προκειμένου να ξεκινήσουν τα δρομολόγια θα πρέπει να εφοδιαστούν με προϊόντα, επομένως για εξοικονόμηση χρόνου όλα τα οχήματα χρησιμοποιούν αυτό το σταθμό ως εκκίνηση του δρομολογίου. Έτσι, ο περιορισμός (3.3) χρησιμοποιείται για να δείξουμε ότι όλα τα οχήματα ξεκινούν από την πόλη $i = 0$, η οποία αποτελεί την αφετηρία. Στον τρίτο περιορισμό, εξασφαλίζεται η καλή σύνδεση των διαδρομών κάθε οχήματος. Πιο συγκεκριμένα, θέλουμε να εξασφαλίσουμε την ομαλή επεξεργασία των στοιχείων από το πρόγραμμα και να αποφύγουμε λανθασμένες διαδρομές στα αποτελέσματα. Έτσι, προκειμένου το πρόγραμμα να μην δημιουργήσει δρομολόγια που δεν έχουν λογική συνοχή μεταξύ τους και συγκεκριμένα δρομολόγια που δεν ξεκινούν και καταλήγουν στη πόλη ανεφοδιασμού, η χρήση του περιορισμού (3.3) είναι σημαντική.

Ο περιορισμός (3.4) είναι αυτός που καθορίζει τον αριθμό των πόλεων που μπορεί να εξυπηρετήσει κάθε όχημα βάσει της χωρητικότητας που διαθέτει. Είναι σημαντικό να γνωρίζει το πρόγραμμα ότι οι απαιτήσεις κάθε πόλης/πελάτη πρέπει να καλύπτονται από ένα όχημα και ολοκληρωμένα. Δηλαδή, εάν ένα όχημα έχει στο δρομολόγιο του μία πόλη, θα πρέπει να έχει αρκετά αποθέματα, ώστε να προμηθεύσει τον πελάτη πλήρως. Η συμπλήρωση προϊόντων για την ικανοποίηση των απαιτήσεων μιας πόλης από δεύτερο όχημα αποκλείεται, σύμφωνα με τον περιορισμό (3.4).

Τέλος, χωρίς τους κατάλληλους περιορισμούς, το πρόγραμμα θα μπορεί να δημιουργήσει επαναλαμβανόμενα δρομολόγια σε μία πόλη δημιουργώντας ένα ατέρμονα βρόγχο. Επίσης, όπως είπαμε και προηγουμένως, όταν ένα όχημα επισκέπτεται μία πόλη τότε κανένα άλλο όχημα δεν μπορεί να επισκεφθεί ξανά αυτή τη πόλη. Οι περιορισμοί (3.6) και (3.7) είναι αυτοί που χρειάζονται, έτσι ώστε να εξαλειφθεί η πιθανότητα επανεπισκέψεων ενός ή περισσοτέρων οχημάτων στην ίδια πόλη.

3.2 Μοντέλο βασισμένο σε διαγράμματα αποφάσεων

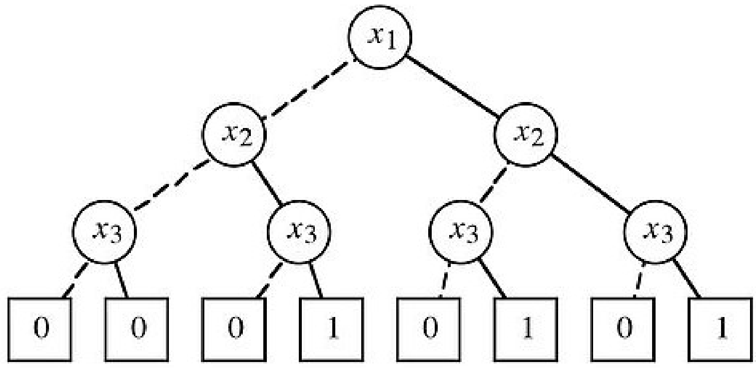
Το δεύτερο μοντέλο είναι μία διαφορετική προσέγγιση του CVRP επηρεασμένη από τη λογική των διαγραμμάτων αποφάσεων (Bergman, Cire, van Hoeve, & Hooker, 2016) (Hooker, 2013). Τα διαγράμματα αποφάσεων έχουν χρησιμοποιηθεί στο παρελθόν για τον σχεδιασμό κυκλωμάτων και επαλήθευσής τους, για λογισμικά και υλικά ενός υπολογιστή και επίσης, για πολλών ειδών έρευνες. Παρόλα αυτά, μπορούν να χρησιμοποιηθούν επιτυχώς και για την βελτιστοποίηση ορισμένων προβλημάτων. Τα διαγράμματα αποφάσεων δίνουν την δυνατότητα στο χρήστη να επιλέξει την επόμενη κίνησή τους, βάση της απόφασης που οι ίδιοι θα λάβουν. Τα πιο διαδεδομένα είναι τα δυαδικά διαγράμματα αποφάσεων (Binary Decision Diagrams). Σε αυτού του είδους τα διαγράμματα, κάθε κόμβος έχει δύο παιδία. Η απόφαση μεταξύ 1 και 0 καθορίζει σε ποιο παιδί θα συνεχίσει ο χρήστης, για να βρει τη βέλτιστη λύση. Στο σχήμα 3.4, παρουσιάζεται ένα δυαδικό διάγραμμα αποφάσεων. Το διάγραμμα, απεικονίζει τα αποτελέσματα των αποφάσεων σε τρία κρίσιμα σημεία και το πόσες διαφορετικές λύσεις μπορούν να προκύψουν σε ένα τόσο μικρό πρόβλημα.

Στην περίπτωσή μας, κάθε κόμβος έχει για πεδία τον ακριβή αριθμό πόλεων/πελατών που εξετάζουμε σε κάθε πρόβλημα, πλην της πόλης από την οποία εφοδιάζονται όλα τα οχήματα. Σε αυτήν την περίπτωση, η επίσκεψη στο επόμενο παιδί καθορίζεται από 0 και 1 για κάθε μετάβαση. Πιο αναλυτικά, ας υποθέσουμε ότι ένα όχημα βρίσκεται σε μία πόλη X. Σύμφωνα με το μοντέλο μας, το όχημα αυτό

[1] https://cs.uni-paderborn.de/ceg/teaching/courses/ws-201617/seminar-binary-decision-diagrams/

Σχήμα 3.4: Διαδικό Διάγραμμα Αποφάσεων[1]

x_1	x_2	x_3	f
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1



Πίνακας 3.2: Στοιχεία πινάκων X και Υ σύμφωνα με το δεύτερο μοντέλο

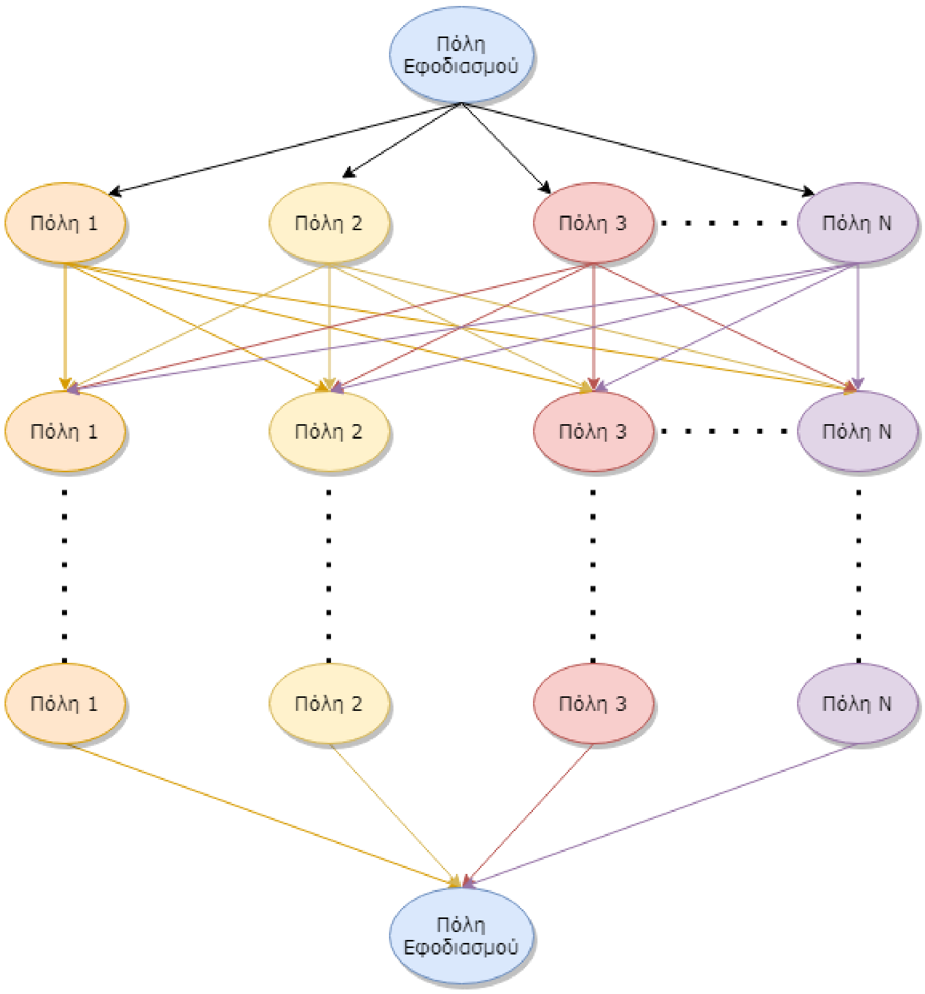
Προς Από	**Πόλη 1**	**Πόλη 2**	**Πόλη 3**	**Επίσκεψη Πρώτου Οχήματος ($\mathbf{y}_{i1}$)**
Πόλη 1	x_{111}	x_{121}	x_{131}	y_{11}
Πόλη 2	x_{211}	x_{221}	x_{231}	y_{21}
Πόλη 3	x_{311}	x_{321}	x_{331}	y_{31}

μπορεί να αποφασίσει με 0 και 1 για κάθε μία πόλη, αν θα την επισκεφτεί. Βέβαια, την τιμή 1 θα μπορεί να πάρει μόνο μία πόλη καθώς, όπως είναι φυσιολογικό, το όχημα μπορεί να επισκεφθεί μία πόλη κάθε φορά. Το ίδιο πράγμα θα γίνει για όλα τα οχήματα με την άφιξή τους σε μία πόλη, και κατ' επέκταση σε κάθε πόλη θα υπάρχει ένα νέος κύκλος αποφάσεων που θα πρέπει να ληφθούν.

Πριν συνεχίσουμε στην παρουσίαση της σχηματικής επεξήγησης του δεύτερου μοντέλου, θα ήταν χρήσιμο να εξηγήσουμε το πως διαμορφώνεται και εδώ ο πίνακας που δημιουργήσαμε στο κλασικό μοντέλο. Ο πίνακας 3.2 , απεικονίζει μία υποθετική κατάσταση του πίνακα διαδρομών X και του πίνακα, που μας επιτρέπει να γνωρίζουμε ποια οχήματα επισκέφθηκαν κάθε πόλη του πίνακα Y. Όπως μπορεί κανείς εύκολα να παρατηρήσει, η διαγώνιος αυτή τη φορά δεν είναι άδεια και αυτό συμβαίνει γιατί πλέον συνυπολογίζονται οι μεταβλητές που υποδηλώνουν τις διαδρομές που μεταφέρουν ένα όχημα από την μία πόλη στην ίδια.

Στο σχήμα 3.5, όλα τα οχήματα βρίσκονται στοιχισμένα στη πόλη εφοδιασμού, και εκεί ξεκινάνε την λήψη αποφάσεων για την συνέχεια στην επόμενη πόλη. Το πρόγραμμα σε αυτό το στάδιο, θα αποφασίσει σε ποια πόλη θα κατευθυνθεί στη συνέχεια το κάθε όχημα. Ο παράγοντας που καθορίζει αυτήν την απόφαση στο δικό μας πρόβλημα, είναι κυρίως η απόσταση που απέχει το όχημα από την επόμενη πόλη. Αφού, αποφασιστεί από το πρόγραμμα η πόλη που θα επισκεφτούν τα οχήματα και ολοκληρωθεί η διαδρομή αυτή, τότε το πρόγραμμα καλείται να αποφασίσει εκ νέου για τους αμέσως επόμενος σταθμούς που θα επισκεφτεί κάθε όχημα. Αυτό ακολουθεί επαναλαμβανόμενα μέχρι τα οχήματα να επισκεφθούν όλες τις πόλεις και να καλύψουν τη ζήτηση, οπότε τα οχήματα θα πρέπει να επιστρέψουν πάλι στην πόλη εφοδιασμού. Υπάρχει όμως πάντα η πιθανότητα ένα όχημα να επισκεφτεί τις πιο απαιτητικές σε ζήτηση πόλεις και να τελειώσουν τα αποθέματά του νωρίτερα από τα υπόλοιπα οχήματα. Σε αυτή τη περίπτωση, το όχημα αυτό θα επιστρέψει στη πόλη εφοδιασμού και το πρόγραμμα θα παίρνει αποφάσεις για τις διαδρομές των υπολοίπων οχημάτων από εκείνο το χρονικό σημείο.

Σχήμα 3.5: Γράφημα μοντέλου βασισμένο σε διαγράμματα αποφάσεων για N αριθμό πόλεων



Προκειμένου να κάνουμε το παραπάνω μοντέλο να λειτουργήσει, θα πρέπει να χρησιμοποιήσουμε αρκετά διαφορετικούς περιορισμούς αλλά και διαφορετικά όρια. Παρά τις ορισμένες διαφορές που παρουσιάζουν τόσο στη λογική όσο και στη τροπολογία τους τα δύο μοντέλα, ορισμένοι από τους περιορισμούς που χρειάστηκαν για το δεύτερο μοντέλο είναι παρόμοιοι με τους περιορισμούς του κλασικού μοντέλου.

Στη συνέχεια παρουσιάζουμε την αντικειμενική συνάρτηση του μοντέλου που καθορίζει το κόστος όλων των πιθανών διαδρομών που πρόκειται να ακολουθήσει κάθε όχημα, καθώς επίσης καθορίζει και όλες τις μεταβλητές με τις οποίες το μοντέλο θα δημιουργήσει τους περιορισμούς. Επίσης, ακολουθεί το μαθηματικό μοντέλο σύμφωνα με το οποίο διαμορφώνει όλους τους περιορισμούς που θα δοθούν στο πρόγραμμα προκειμένου αυτό να εκτυπώσει την βέλτιστη λύση.

$$\min_{z} = \sum_{i\in V}\sum_{j\in V}\sum_{k\in K} c_{ij} \cdot x_{ijk} \tag{3.8}$$

$$\forall i \in R : \sum_{k\in K} y_{ik} = 1 \tag{3.9}$$

$$\forall i \in R : \sum_{k\in K} y_{iik} = 0 \tag{3.10}$$

$$\sum_{k\in K} y_{0k} = K \tag{3.11}$$

$$\forall i \in R : \sum_{k\in K} (x_{0ik} + x_{i0k}) \leq 1 \tag{3.12}$$

$$\forall k \in K, i \in R : \sum_{j\in R} x_{ijk} = \sum_{j\in R} x_{jik} = y_{ik} \tag{3.13}$$

$$\sum_{k\in K}\sum_{j\in R}\sum_{i\in R} x_{ijk} \leq K \tag{3.14}$$

$$\sum_{k\in K}\sum_{j\in R} x_{0jk} = \sum_{k\in K}\sum_{i\in R} x_{i0k} = K \tag{3.15}$$

$$\forall k \in K : \sum_{i\in R} d_i \cdot y_{ik} \leq C \tag{3.16}$$

$$\forall S \subseteq R, h \in S, k \in K : \sum_{i\in S}\sum_{i\notin S} x_{ijk} \geq y_{hk} - 1 \tag{3.17}$$

$$\forall i \in V, j \in V : \sum_{k \in K} (x_{ijk} + x_{jik}) \leq 1 \tag{3.18}$$

Σε αυτό το μοντέλο, η αντικειμενική συνάρτηση (3.8) δεν παρουσιάζει σημαντικές διαφορές από αυτή του πρώτου μοντέλου, εκτός από το ότι για τις πόλεις δεν ισχύει $i \neq j$. Βέβαια, δεν μπορεί επίσης να υπάρχει η πιθανότητα τα οχήματα να κατευθύνονται από τη πόλη εφοδιασμού κατευθείαν στην πόλη εφοδιασμού, επομένως εξαλείφεται η περίπτωση όπου $i = j = 0$. Όπως είναι λογικό, η αντικειμενική συνάρτηση δηλώνει ότι, στόχος του μοντέλου είναι η ελαχιστοποίηση του αποτελέσματος, δηλαδή το συνολικό κόστος όλων των διαδρομών να είναι το μικρότερο δυνατό, πάντα στα πλαίσια που ικανοποιούν τους περιορισμούς.

Ο περιορισμός (3.9), όπως και στο προηγούμενο μοντέλο φροντίζει κάθε πόλη να την επισκεφθεί ένα μόνο όχημα. Είναι ένας σημαντικός κανόνας και πρέπει να τηρηθεί αυστηρά. Με αυτό τον περιορισμό το μοντέλο, αποκλείει την περίπτωση συνωστισμού πολλών οχημάτων σε έναν σταθμό, και φυσικά την περίπτωση ένα όχημα να επισκεφθεί μία πόλη/πελάτη, του οποίου η ζήτηση έχει καλυφθεί. Κάτι τέτοιο, είναι πολύ πιθανό να συμβεί στην περίπτωση που ένα όχημα έχει ολοκληρώσει τα δρομολόγια του και οδεύει προς την πόλη εφοδιασμού. Εάν το πρόγραμμα αντιληφθεί ότι υπάρχει μικρότερη διαδρομή για επιστροφή στη πόλη εφοδιασμού από την άμεση επιστροφή του σε αυτή, που όμως είναι διαμέσου ενός σταθμού του οποίου η ζήτηση έχει καλυφθεί, τότε είναι φυσικό να επιλέξει αυτήν προκειμένου να ελαχιστοποιήσει το κόστος. Ένα τέτοιο συμβάν, θα παραποιούσε σημαντικά το συνολικό αποτέλεσμα της λύσης.

Όπως προαναφέραμε, η σημαντική διαφορά του μοντέλου βασισμένου σε διαγράμματα αποφάσεων σε σχέση με το κλασικό μοντέλο είναι, ότι το δεύτερο μοντέλο λαμβάνει υπόψιν όλες τις πιθανές διαδρομές από όλες τις πόλεις προς όλες τις πόλεις. Έτσι, όταν ένα όχημα βρίσκεται για παράδειγμα στη πόλη 1, ανάμεσα στις πόλεις που πιθανόν θα συνεχίσει το δρομολόγιό του, είναι πάλι η πόλη 1. Το πρόβλημα εδώ είναι ότι, βάση της Ευκλείδειας απόστασης, μέσω της οποίας προκύπτει η απόσταση μίας πόλης προς όλες τις άλλες, η επίσκεψη ενός οχήματος από μία πόλη στην ίδια είναι μηδενική. Αυτό θα οδηγούσε το πρόγραμμα σε ατέρμον βρόγχο, μεταφέροντας τα οχήματα στην ίδια πόλη συνεχώς. Γι' αυτό το λόγο χρησιμοποιούμε τον περιορισμό (3.10), ο οποίος θέτει μηδέν τις διαδρομές μεταξύ ίδιων πόλεων.

Στον περιορισμό (3.11), καθορίζουμε ως πόλη αφετηρία όλων των οχημάτων την πόλη μηδέν. Είναι σημαντικό, το πρόγραμμα να γνωρίζει ποια θα είναι η πόλη από την οποία όλα τα οχήματα θα πρέπει να ξεκινούν όλα τα δρομολόγια. Σε αντίθετη περίπτωση, το πρόγραμμα θα μπορούσε να ξεκινήσει από οποιαδήποτε πόλη τις διαδρομές των οχημάτων και να καταλήγουν στην πόλη του εφοδιασμού, πράγμα που αρχικά θα φαινόταν σαν ιδανική λύση για το πρόβλημα, εάν έδινε την βέλτιστη λύση βάση του κόστους διαδρομών, αλλά στη πραγματικότητα θα δημιουργούσε σημαντικά προβλήματα στην γενική εικόνα του προβλήματος. Είναι περιττό να αναφερθεί ότι αυτό το αποτέλεσμα είναι αντίθετο με τη γενική τροπολογία του Προβλήματος Δρομολόγησης Οχημάτων.

Στον περιορισμό (3.12) , απαγορεύουμε την περίπτωση όπου ένα όχημα επισκέπτεται μόνο μία πόλη και επιστρέφει πάλι στη πόλη αφετηρία, αφού αυτό βάση της λογικής μας θεωρείται επαναληπτική διαδρομή την οποία και αποκλείουμε σε παρακάτω περιορισμό. Σε μία ιδανική περίπτωση για την ελαχιστοποίηση του κόστους όλων των διαδρομών, η σύντομη διαδρομή ενός οχήματος για την κάλυψη της ζήτησης μιας πόλης, ακόμη και αν αυτό σημαίνει να επισκεφτεί μόνο μία πόλη, κάνει δελεαστική την απόφαση του προγράμματος να δημιουργήσει ένα τέτοιο δρομολόγιο για κάποιο όχημα. Στη χειρότερη περίπτωση, το πρόγραμμα θα μπορούσε να επιτρέψει την επανάληψη αυτής της τεχνικής, για να καλύψει την ζήτηση των κοντινότερων πόλεων/πελατών με ένα όχημα το οποίο θα επιστρέφει στην πόλη εφοδιασμού κάθε φορά που ολοκληρώνει μία παραγγελία. Σε αυτό το σημείο, θα δούμε έναν περιορισμό που χρησιμοποιήθηκε και στο κλασικό μοντέλο για την αποτροπή ενός μείζονος προβλήματος. Ο περιορισμός αυτός, είναι αρμόδιος για την καλή συνοχή του συνολικού δρομολογίου κάθε οχήματος. Ο περιορισμός (3.13), συνδέει τις διαδρομές ενός οχήματος μεταξύ τους, έτσι ώστε το πρόγραμμα να είναι σε θέση να αντιλαμβάνεται ότι, κάθε φορά που κάποιο όχημα καταφθάνει σε μία πόλη, μπορεί να φύγει από αυτή χωρίς προβλήματα. Με άλλα λόγια, όταν ένα όχημα καλύψει τη ζήτηση ενός πελάτη και το πρόγραμμα πρέπει να αποφασίσει ποιος θα είναι ο επόμενος σταθμός του οχήματος, τότε το πρόγραμμα βλέποντας ποιες είναι οι υπολειπόμενες πόλεις/πελάτες που χρειάζονται να καλυφθούν, θα πρέπει να γνωρίζει από ποια πόλη προέρχεται το όχημα, για να κατευθυνθεί στην επόμενη. Σε αντίθετη περίπτωση, υπήρχε η πιθανότητα τα οχήματα να ακολουθούν διαδρομές χωρίς συνέχεια και αυτό θα είχε ως συνέπεια τα αποτελέσματα να είναι εσφαλμένα.

Ο επόμενος περιορισμός, ο περιορισμός (3.14) είναι μία καινούργια ιδέα, αποκλειστικά δημιουργημένος για το μοντέλο βασισμένο στα διαγράμματα αποφάσεων. Πιο αναλυτικά, σε κάθε σειρά – όπως φαίνεται στο σχήμα 3.5 – όπου το πρόγραμμα αποφασίζει για τον επόμενο σταθμό κάθε οχήματος το μοντέλο, θέτει τον μέγιστο αριθμό των συνολικών μεταβάσεων όλων των οχημάτων προς τις επόμενες πόλεις στον συνολικό αριθμό των οχημάτων που βρίσκεται στη διάθεση μας. Αυτός ο περιορισμός είναι σημαντικός, διότι σε περίπτωση που ένα όχημα ολοκληρώσει το δρομολόγιό του ή τα αποθέματα του τελειώσουν και αναγκαστεί να επιτρέψει στο σταθμό εφοδιασμού έχοντας επισκεφτεί λιγότερες πόλεις/πελάτες από τα άλλα οχήματα, τότε ο συνολικός αριθμός μεταβάσεων σε επόμενες πόλεις, θα είναι μικρότερος από τον συνολικό αριθμό οχημάτων και κάτι τέτοιο γίνεται αποδεκτό από το πρόγραμμα. Στην περίπτωση όμως που ο περιορισμός αυτός δεν υπήρχε, για να περιορίσει τον μέγιστο αριθμό μεταβάσεων, τότε το πρόγραμμα θα μπορούσε να δώσει εντολή σε κάποιο όχημα να ακολουθήσει δύο δρομολόγια ταυτόχρονα και πιο συγκεκριμένα, το δρομολόγιο που θα οδηγούσε στην ίδια πόλη σε συνδυασμό με κάποια άλλη που δεν έχει εξυπηρετηθεί. Με αυτό το τρόπο, οι δρομολογημένες διαδρομές θα ξεπερνούσαν τον συνολικό αριθμό των οχημάτων, επομένως το μοντέλο με τον έκτο περιορισμό αποκλείει οποιαδήποτε τέτοια περίπτωση.

Στον περιορισμό (3.15) , κάνουμε σαφές στο πρόγραμμα ότι τα οχήματα που θα ξεκινήσουν από την πόλη εφοδιασμού, καθώς επίσης τα οχήματα που θα καταλήξουν σε αυτή είναι ίσα σε αριθμό, και ακριβώς όσα και τα οχήματα που έχουμε στη διάθεσή μας για διανομή προϊόντων. Πιο αναλυτικά, είναι σημαντικό το πρόγραμμα να γνωρίζει ότι για να επιτύχουμε τη βέλτιστη λύση, θα πρέπει να χρησιμοποιήσουμε όλο το δυναμικό μιας εταιρίας και να δημιουργηθούν δρομολόγια για όλα τα οχήματα. Επιπλέον, εφόσον έχουμε δώσει την εντολή στο πρόγραμμα να δημιουργήσει δρομολόγια για όλα τα οχήματα, θα πρέπει ταυτοχρόνως το πρόγραμμα να είναι σε θέση να μπορεί να επιστρέψει επίσης όλα τα οχήματα στη πόλη εφοδιασμού. Έτσι, με τον έβδομο περιορισμό, συνολικά, επιτυγχάνουμε την ολοκληρωτική εκμετάλλευση όλων των διαθέσιμων οχημάτων προκειμένου να πετύχουμε το βέλτιστο αποτέλεσμα και ταυτοχρόνως δεν αφήνουμε περιθώρια στο πρόγραμμα να αφήσει κάποιο όχημα σε πόλη διαφορετική από την πόλη ανεφοδιασμού.

Ο περιορισμός (3.16), δεν παρουσιάζει σημαντικές διαφορές από τον αντίστοιχο περιορισμό, του κλασικού μοντέλου. Πιο συγκεκριμένα, είναι ο περιορισμός που δίνει τη δυνατότητα στο πρόγραμμα να γνωρίζει ποιες πόλεις/πελάτες μπορεί να επισκεφθεί κάθε όχημα βάση των απαιτήσεων τους. Με άλλα λόγια, ο μέγιστος αριθμός πόλεων των οποίων την ζήτηση μπορεί να καλύψει κάθε όχημα καθορίζεται από το αριθμό χωρητικότητας κάθε οχήματος, πλην των απαιτήσεων κάθε πόλης/πελάτη. Με αυτό το τρόπο, το πρόγραμμα γνωρίζει από την αρχή ποιες πόλεις σε σύνολο θα αναθέσει και σε ποιο όχημα, προκειμένου να καλυφθεί πλήρως η ζήτηση. Έτσι, αποφεύγεται η υποψία όπου ένα όχημα θα καταφθάσει σε μία πόλη/πελάτη, μη έχοντας στη διάθεσή του αρκετά προϊόντα για να καλύψει τη ζήτηση.

Οι περιορισμοί (3.17) και (3.18) είναι αρμόδιοι για την αποφυγή οποιασδήποτε επίσκεψης ενός οχήματος σε πόλη που ή το ίδιο ή κάποιο άλλο από τα οχήματα έχει επισκεφθεί στο παρελθόν παραδίδοντας τα απαραίτητα προϊόντα, για να καλύψει τον πελάτη. Πρέπει να γίνει εδώ σαφές ότι, η επίσκεψη οποιουδήποτε οχήματος σε μία πόλη/πελάτη του οποίου η ζήτηση έχει καλυφθεί, συνήθως για την μείωση του συνολικού κόστους ενός δρομολογίου, δεν είναι αποδεκτή από το γενικό Πρόβλημα Δρομολόγησης Οχημάτων και πρέπει να αποφεύγεται. Για αυτόν ακριβώς το λόγο, χρησιμοποιούμε αυτούς τους δύο περιορισμούς.

3.3 Διαφορές μεταξύ μοντέλων

Τα δύο μοντέλα παρουσιάζουν σημαντικές διαφορές μεταξύ τους. Βέβαια, το δεύτερο μοντέλο είναι βασισμένο στο κλασικό, καθώς όπως έχουμε αναφέρει σε παραπάνω κεφάλαιο, το μοντέλο βασισμένο στα διαγράμματα αποφάσεων είναι η προσπάθειά μας να πετύχουμε, τα ίδια και καλύτερα αποτελέσματα από ένα κλασικό μοντέλο με τη διαφορά ότι, σε αυτό το μοντέλο ο τρόπος σκέψης, αποτελεί παραλλαγή. Και τα δύο μοντέλα χρησιμοποιούν παρόμοιους περιορισμούς, αλλά συνήθως τα όρια τον περιορισμών είναι αυτό που κάνει τη μεγάλη διαφορά.

Συγκεκριμένα, στη περίπτωση του δεύτερου μοντέλου ορισμένες μεταβλητές πρέπει να ληφθούν υπόψιν για την καλή επεξεργασία των δεδομένων, ενώ κάποια άλλα στοιχεία πρέπει βάσει λογικής να μην υπολογιστούν.

Πιο αναλυτικά, κάθε πόλη έχει στην πορεία τις διαδρομές οι οποίες οδηγούν στην ίδια πάλι πόλη, άρα και αυτές οι διαδρομές θα πρέπει να ληφθούν υπόψιν στον υπολογισμό του μοντέλου, εν αντιθέσει με το κλασικό μοντέλο που δεν παρουσιάζεται τέτοιο ενδεχόμενο. Έτσι, προκειμένου να είμαστε σε θέση να προσθέσουμε κάποιους περιορισμούς, για να αποτρέψουμε την πιθανότητα δημιουργίας ατέρμων βρόγχου όπου ένα όχημα υποθετικά επισκέπτεται συνεχώς την ίδια πόλη, δημιουργούμε πολύ περισσότερους περιορισμούς και μεταβλητές. Παράλληλα, το δεύτερο μοντέλο μας είναι εμφανέστατα πιο φορτωμένο με περιορισμούς, από ότι το κλασικό μοντέλο. Είναι φυσικό, ένα μοντέλο που απαιτείται να λάβει υπόψιν του περισσότερες μεταβλητές, ιδιαίτερα ένα μοντέλο το οποίο χρησιμοποιεί μία όχι και τόσο στα μέτρα του Προβλήματος Δρομολόγησης Οχημάτων μέθοδο - όπως είναι τα διαγράμματα αποφάσεων - να έχει πολλούς περιορισμούς προκειμένου να καλύψει όλες τις πιθανές περιπτώσεις, που μπορεί να προκύψουν. Εύκολα λοιπόν φτάνει κανείς στο συμπέρασμα ότι, με όλους τους περιορισμούς που πρέπει να περιπλέξει το μοντέλο μας, θα χρειαστεί και πολύ χρόνο για να επεξεργαστεί τα δεδομένα και να καταλήξει στη βέλτιστη λύση.

Ωστόσο, βασική διαφορά των δύο μοντέλων είναι ότι, το γράφημά μας δεν είναι ένα τυπικό γράφημα για CVRP προβλήματα, αλλά πρόκειται για μία παραλλαγή, όπως προαναφέρθηκε, με επιρροή από τα διαγράμματα αποφάσεων. Έτσι, σε αντίθεση με το κλασικό μοντέλο, οι διαθέσιμες διαδρομές από και προς την πόλη εφοδιασμού (Πόλη 0), υπολογίζονται μία φορά για κάθε όχημα και δεν χρειάζεται να λαμβάνονται υπόψιν κάθε φορά που ένα όχημα πρόκειται να μεταβεί από μία πόλη στην επόμενη. Κάτι τέτοιο δεν σημαίνει απαραίτητα ότι, επειδή οι διαδρομές αυτές υπολογίζονται μία φορά, τότε τα δρομολόγια όλων των οχημάτων θα περιέχουν τον ίδιο αριθμό πόλεων πριν επιστρέψουν στην πόλη εφοδιασμού.

Επομένως, φτάνουμε στο συμπέρασμα ότι, μπορεί το δεύτερο μοντέλο να επεξεργάζεται πολύ περισσότερους περιορισμούς και να πρέπει να λάβει υπόψιν του περισσότερες μεταβλητές αλλά το εύρος αναζήτησής του για κάθε μεταφορά οχήματος από τη μία πόλη στη επόμενη, περιορίζεται κατά μία ολόκληρη πόλη και συγκεκριμένα στην πόλη εφοδιασμού που υπολογίζεται μόνο σε σημαντικές περιπτώσεις. Φυσικά, αυτό δεν σημαίνει απαραίτητα ότι, το πρόγραμμα θα ολοκληρώσει την εύρεση βέλτιστης λύσης σε καλύτερο χρόνο από το κλασικό μοντέλο, όπως θα δούμε και στο επόμενο κεφάλαιο αλλά, με τις κατάλληλες τροποποιήσεις σε κάποια μελλοντική δουλειά θα μπορούσαμε να κάνουμε το μοντέλο μας πιο ανταγωνιστικό.

3.4 Παραδείγματα

Προκειμένου να γίνει καλύτερα κατανοητό το CVRP, θα αναπτύξουμε δύο απλά παραδείγματα και έτσι θα δούμε τη συμπεριφορά των μοντέλων καθώς και τα γραφήματα που δημιουργούνται. Με αυτόν τον τρόπο, θα είμαστε σε θέση να αντιληφθούμε ποια μέθοδο σκέψης ακολουθεί κάθε μοντέλο, τι μεταβλητές επεξεργάζονται, τι αποφάσεις καλούνται να πάρουν καθώς επίσης και τι μορφή αποκτά το γενικό διάγραμμα απεικόνισης των δρομολογίων που αποφάσισε το πρόγραμμα να αναθέσει στα οχήματα. Να σημειωθεί εδώ ότι για προφανής λόγους, τα χαρακτηριστικά των πειραμάτων που έχουν τα παραδείγματα, όπως ο αριθμός πόλεων, οι απαιτήσεις κάθε πόλης, η μέγιστη χωρητικότητα των οχημάτων καθώς επίσης και η απόσταση που απέχουν όλες οι πόλεις μεταξύ τους, είναι τα ίδια και για τα δύο μοντέλα.

3.4.1 Παράδειγμα Πρώτο

Στο παράδειγμα που ακολουθεί, θα αναλύσουμε την διαδικασία εύρεσης της βέλτιστης διαδρομής ενός οχήματος, που βρίσκεται στη διάθεση μίας υποθετικής εταιρείας διανομής προϊόντων, για να καλύψει τη ζήτηση τεσσάρων πόλεων/πελατών. Το όχημα είναι ικανό να μεταφέρει 30 προϊόντα σε κάθε δρομολόγιο. Όπως έχουμε προαναφέρει, τα οχήματα δεν επιστρέφουν στην πόλη αφετηρία για ανεφοδιασμό προϊόντων, επομένως, στο παράδειγμα θέτουμε συνολική ζήτηση όλων των πελατών μικρότερη από 30. Τα δύο μοντέλα, θα προσπαθήσουν να βρουν την βέλτιστη διαδρομή που πρέπει να ακολουθήσει το όχημα, προκειμένου να ολοκληρώσει την παράδοση των προϊόντων στο μικρότερο δυνατό χρόνο. Ο πίνακας 3.3, παρουσιάζει αναλυτικά όλα τα κόστη των διαδρομών από και προς όλες τις πόλεις, καθώς και τις απαιτήσεις κάθε πόλης/πελάτη.

Πίνακας 3.3: Πίνακας Πρώτου Παραδείγματος

	Πόλη 0 (Εφοδιασμού)	**Πόλη 1**	**Πόλη 2**	**Πόλη 3**	**Πόλη 4**	**Απαιτήσεις**
Πόλη 0 (Εφοδιασμού)	-	1	3	2	4	0
Πόλη 1	1	-	1	2	4	5
Πόλη 2	3	1	-	2	2	7
Πόλη 3	2	2	2	-	1	8
Πόλη 4	4	4	2	1	-	9

Σύμφωνα με αυτά τα στοιχεία, τα δύο μοντέλα θα καταλήξουν στη βέλτιστη λύση.

Η βέλτιστη απόσταση που αποφασίζει το κλασικό μοντέλο είναι ίση με 7. Το όχημα ξεκινάει από την πόλη εφοδιασμού και κατευθύνεται στην πόλη 1, όπου και καλύπτει την ζήτηση της πρώτης πόλης. Στη συνέχεια, το πρόγραμμα ελέγχει εκ νέου τις αποστάσεις από την πόλη που βρίσκεται το όχημα προς όλες τι άλλες και αποφασίζει ως επόμενη πόλη κατεύθυνσης την πόλη 2. Σε αυτό το σημείο υπάρχει μία ισοτιμία στις αποστάσεις μεταξύ των πόλεων 3 και 4 ως προς την απόσταση που απέχουν από την πόλη 2. Το πρόγραμμα ελέγχει ποια θα είναι η συνολική βέλτιστη απόσταση που θα διανύσει ελέγχοντας και τις δύο πιθανές διαδρομές, και αποφασίζει να συνεχίσει στην πόλη 4 και έπειτα στην πόλη 3, πριν επιστρέψει στην πόλη ανεφοδιασμού για να ολοκληρώσει το όχημα το δρομολόγιό του. Όπως εύκολα μπορεί να παρατηρήσει κανείς ότι, αν το όχημα από την πόλη 2 συνέχιζε στην πόλη 3 και στη συνέχεια στην 4, στην ολοκλήρωση του δρομολογίου θα διένυε πολύ μεγαλύτερη απόσταση. Στο σχήμα 3.6 βλέπουμε την πορεία που ακολούθησε το όχημα βάση του κλασικού μοντέλου. Απο κάτω από το σχήμα θα δούμε και το πως διαμορφώνονται η αντικειμενική συνάρτηση και οι περιορισμοί του πρώτου μοντέλου σύμφωνα με το πρώτο παράδειγμα.

$$\min\ 1x_{011} + 3x_{021} + 2x_{031} + 4x_{041} + 1x_{101} + 1x_{121} + 2x_{131} + 4x_{141} + 3x_{201}$$

$$+ 1x_{211} + 2x_{231} + 2x_{241} + 2x_{301} + 2x_{311} + 2x_{321} + 1x_{341} + 4x_{401} + 4x_{411} + 2x_{421} + 1x_{431}$$

$$\text{s.t. } y_{01} = 1$$

$$y_{11} = 1$$

$$y_{21} = 1$$

$y_{31} = 1$

$y_{41} = 1$

$x_{011} + x_{021} + x_{031} + x_{041} = x_{101} + x_{201} + x_{301} + x_{401} = y_{01}$

$x_{101} + x_{121} + x_{131} + x_{141} = x_{011} + x_{211} + x_{311} + x_{411} = y_{11}$

$x_{201} + x_{211} + x_{231} + x_{241} = x_{021} + x_{121} + x_{321} + x_{421} = y_{21}$

$x_{301} + x_{311} + x_{321} + x_{341} = x_{031} + x_{131} + x_{231} + x_{431} = y_{31}$

$x_{401} + x_{411} + x_{421} + x_{431} = x_{041} + x_{141} + x_{241} + x_{341} = y_{41}$

$y_{01} + y_{11} + y_{21} + y_{31} + y_{41} <= 29$

$x_{021} + x_{031} + x_{041} + x_{121} + x_{131} + x_{141} >= y_{01} - 1$

$x_{021} + x_{031} + x_{041} + x_{121} + x_{131} + x_{141} >= y_{11} - 1$

$x_{011} + x_{031} + x_{041} + x_{211} + x_{231} + x_{241} >= y_{01} - 1$

$x_{011} + x_{031} + x_{041} + x_{211} + x_{231} + x_{241} >= y_{21} - 1$

$x_{011} + x_{021} + x_{041} + x_{311} + x_{321} + x_{341} >= y_{01} - 1$

$x_{011} + x_{021} + x_{041} + x_{311} + x_{321} + x_{341} >= y_{31} - 1$

$x_{011} + x_{021} + x_{031} + x_{411} + x_{421} + x_{431} >= y_{01} - 1$

$x_{011} + x_{021} + x_{031} + x_{411} + x_{421} + x_{431} >= y_{41} - 1$

$x_{101} + x_{131} + x_{141} + x_{201} + x_{231} + x_{241} >= y_{11} - 1$

$x_{101} + x_{131} + x_{141} + x_{201} + x_{231} + x_{241} >= y_{21} - 1$

$x_{101} + x_{121} + x_{141} + x_{301} + x_{321} + x_{341} >= y_{11} - 1$

$x_{101} + x_{121} + x_{141} + x_{301} + x_{321} + x_{341} >= y_{31} - 1$

$x_{101} + x_{121} + x_{131} + x_{401} + x_{421} + x_{431} >= y_{11} - 1$

$x_{101} + x_{121} + x_{131} + x_{401} + x_{421} + x_{431} >= y_{41} - 1$

$x_{201} + x_{211} + x_{241} + x_{301} + x_{311} + x_{341} >= y_{21} - 1$

$x_{201} + x_{211} + x_{241} + x_{301} + x_{311} + x_{341} >= y_{31} - 1$

$x_{201} + x_{211} + x_{231} + x_{401} + x_{411} + x_{431} >= y_{21} - 1$

$x_{201} + x_{211} + x_{231} + x_{401} + x_{411} + x_{431} >= y_{41} - 1$

$x_{301} + x_{311} + x_{321} + x_{401} + x_{411} + x_{421} >= y_{31} - 1$

$x_{301} + x_{311} + x_{321} + x_{401} + x_{411} + x_{421} >= y_{41} - 1$

$$x_{031} + x_{041} + x_{131} + x_{141} + x_{231} + x_{241} >= y_{01} - 1$$
$$x_{031} + x_{041} + x_{131} + x_{141} + x_{231} + x_{241} >= y_{11} - 1$$
$$x_{031} + x_{041} + x_{131} + x_{141} + x_{231} + x_{241} >= y_{21} - 1$$
$$x_{021} + x_{041} + x_{121} + x_{141} + x_{321} + x_{341} >= y_{01} - 1$$
$$x_{021} + x_{041} + x_{121} + x_{141} + x_{321} + x_{341} >= y_{11} - 1$$
$$x_{021} + x_{041} + x_{121} + x_{141} + x_{321} + x_{341} >= y_{31} - 1$$
$$x_{021} + x_{031} + x_{121} + x_{131} + x_{421} + x_{431} >= y_{01} - 1$$
$$x_{021} + x_{031} + x_{121} + x_{131} + x_{421} + x_{431} >= y_{11} - 1$$
$$x_{021} + x_{031} + x_{121} + x_{131} + x_{421} + x_{431} >= y_{41} - 1$$
$$x_{011} + x_{041} + x_{211} + x_{241} + x_{311} + x_{341} >= y_{01} - 1$$
$$x_{011} + x_{041} + x_{211} + x_{241} + x_{311} + x_{341} >= y_{21} - 1$$
$$x_{011} + x_{041} + x_{211} + x_{241} + x_{311} + x_{341} >= y_{31} - 1$$
$$x_{011} + x_{031} + x_{211} + x_{231} + x_{411} + x_{431} >= y_{01} - 1$$
$$x_{011} + x_{031} + x_{211} + x_{231} + x_{411} + x_{431} >= y_{21} - 1$$
$$x_{011} + x_{031} + x_{211} + x_{231} + x_{411} + x_{431} >= y_{41} - 1$$
$$x_{011} + x_{021} + x_{311} + x_{321} + x_{411} + x_{421} >= y_{01} - 1$$
$$x_{011} + x_{021} + x_{311} + x_{321} + x_{411} + x_{421} >= y_{31} - 1$$
$$x_{011} + x_{021} + x_{311} + x_{321} + x_{411} + x_{421} >= y_{41} - 1$$
$$x_{101} + x_{141} + x_{201} + x_{241} + x_{301} + x_{341} >= y_{11} - 1$$
$$x_{101} + x_{141} + x_{201} + x_{241} + x_{301} + x_{341} >= y_{21} - 1$$
$$x_{101} + x_{141} + x_{201} + x_{241} + x_{301} + x_{341} >= y_{31} - 1$$
$$x_{101} + x_{131} + x_{201} + x_{231} + x_{401} + x_{431} >= y_{11} - 1$$
$$x_{101} + x_{131} + x_{201} + x_{231} + x_{401} + x_{431} >= y_{21} - 1$$
$$x_{101} + x_{131} + x_{201} + x_{231} + x_{401} + x_{431} >= y_{41} - 1$$
$$x_{101} + x_{121} + x_{301} + x_{321} + x_{401} + x_{421} >= y_{11} - 1$$
$$x_{101} + x_{121} + x_{301} + x_{321} + x_{401} + x_{421} >= y_{31} - 1$$
$$x_{101} + x_{121} + x_{301} + x_{321} + x_{401} + x_{421} >= y_{41} - 1$$
$$x_{201} + x_{211} + x_{301} + x_{311} + x_{401} + x_{411} >= y_{21} - 1$$

$x_{201} + x_{211} + x_{301} + x_{311} + x_{401} + x_{411} >= y_{31} - 1$

$x_{201} + x_{211} + x_{301} + x_{311} + x_{401} + x_{411} >= y_{41} - 1$

$x_{011} + x_{101} <= 1$

$x_{021} + x_{201} <= 1$

$x_{031} + x_{301} <= 1$

$x_{041} + x_{401} <= 1$

$x_{101} + x_{011} <= 1$

$x_{121} + x_{211} <= 1$

$x_{131} + x_{311} <= 1$

$x_{141} + x_{411} <= 1$

$x_{201} + x_{021} <= 1$

$x_{211} + x_{121} <= 1$

$x_{231} + x_{321} <= 1$

$x_{241} + x_{421} <= 1$

$x_{301} + x_{031} <= 1$

$x_{311} + x_{131} <= 1$

$x_{321} + x_{231} <= 1$

$x_{341} + x_{431} <= 1$

$x_{401} + x_{041} <= 1$

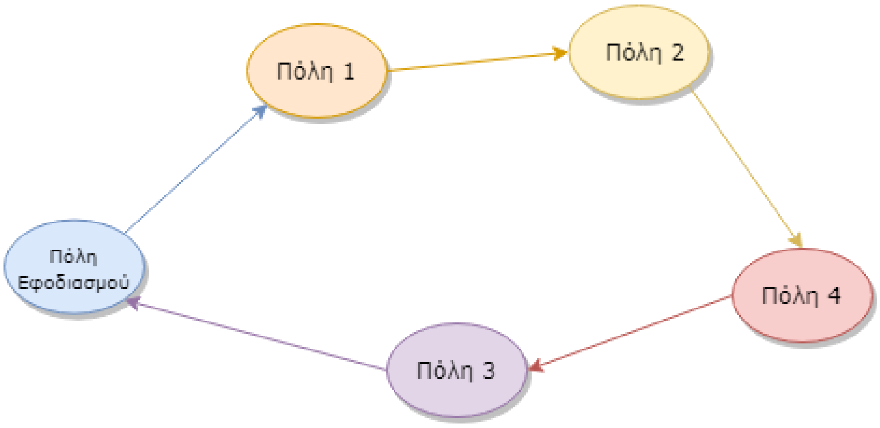
$x_{411} + x_{141} <= 1$

$x_{421} + x_{241} <= 1$

$x_{431} + x_{341} <= 1$

Το μοντέλο βασισμένο στα διαγράμματα αποφάσεων, αποφάσισε εξίσου ότι η βέλτιστη απόσταση που θα διανύσει το όχημα είναι 7. Βλέπουμε εδώ λοιπόν ότι, τα δύο μοντέλα συμφωνούν στην απόφασή του για την βέλτιστη τιμή που είναι άλλωστε και το ζητούμενο. Στο συγκεκριμένο παράδειγμα, τα δύο μοντέλα αποφάσισαν την ίδια διαδρομή για να ακολουθήσει το όχημα. Αυτό, μπορεί να μη συμβαίνει

Σχήμα 3.6: Διαμόρφηση Διαγράμματος Πρώτου Μοντέλο για Παράδειγμα 1



πάντα. Όπως θα δούμε σε επόμενο παράδειγμα, στη περίπτωση που η βέλτιστη λύση μπορεί να επιτευχθεί από δύο ή και περισσότερες διαφορετικές διαδρομές, δεν αποκλείεται να βρεθούμε με δύο εξίσου σωστές βέλτιστες διαδρομές, μία από κάθε μοντέλο ξεχωριστά οι οποίες όμως να είναι πάντα στα νόμιμα πλαίσια, σύμφωνα με τους κανονισμούς του CVRP. Εκτός από τη λύση, είναι ενδιαφέρον να δούμε πόσο διαφορετικά διαμορφώνεται το σχήμα 3.7. Σε αντίθεση με το πρώτο, όπως βλέπουμε, ακολουθεί πιστά την λογική του διαγράμματος αποφάσεων. Σε κάθε περίπτωση, κατά την οποία το όχημα καταφθάνει σε μία πόλη, το όχημα αναλύει όλες τις διαθέσιμες διαδρομές ως επόμενο σταθμό, και αποφασίζει ποια είναι η επόμενη πόλη όπου θα συνεχίσει το δρομολόγιο το όχημα. Στο τέλος, αφού έχει ικανοποιήσει τις απαιτήσεις όλων των πόλεων/πελατών, απομένει το όχημα που έκανε την διαδρομή να επιστρέφει στη πόλη εφοδιασμού. Πριν από το σχήμα θα δούμε την αντικειμενική συνάρτηση και τους περιορισμοούς του δεύτερου μοντέλου, σύμφωνα με το πρώτο παράδειγμα.

$$
\begin{aligned}
&\text{min } 1x_{011} + 3x_{021} + 2x_{031} + 4x_{041} + 1x_{101} + 0x_{111} + 1x_{121} \\
&+ 2x_{131} + 4x_{141} + 3x_{201} + 1x_{211} + 0x_{221} + 2x_{231} + 2x_{241} + 2x_{301} \\
&+ 2x_{311} + 2x_{321} + 0x_{331} + 1x_{341} + 4x_{401} + 4x_{411} + 2x_{421} + 1x_{431} \\
&+ 0x_{441} \\
&\text{s.t. } y_{01} = 1 \\
&y_{11} = 1
\end{aligned}
$$

$y_{21} = 1$

$y_{31} = 1$

$y_{41} = 1$

$y_{111} = 0$

$y_{221} = 0$

$y_{331} = 0$

$y_{441} = 0$

$x_{101} + x_{011} <= 1$

$x_{201} + x_{021} <= 1$

$x_{301} + x_{031} <= 1$

$x_{401} + x_{041} <= 1$

$x_{101} + x_{111} + x_{121} + x_{131} + x_{141} = x_{011} + x_{111} + x_{211} + x_{311}$

$+ x_{411} = y_{11}$

$x_{201} + x_{211} + x_{221} + x_{231} + x_{241} = x_{021} + x_{121} + x_{221} + x_{321}$

$+ x_{421} = y_{21}$

$x_{301} + x_{311} + x_{321} + x_{331} + x_{341} = x_{031} + x_{131} + x_{231} + x_{331}$

$+ x_{431} = y_{31}$

$x_{401} + x_{411} + x_{421} + x_{431} + x_{441} = x_{041} + x_{141} + x_{241} + x_{341}$

$+ x_{441} = y_{41}$

$x_{101} + x_{011} <= 1$

$x_{201} + x_{021} <= 1$

$x_{301} + x_{031} <= 1$

$x_{401} + x_{041} <= 1$

$x_{111} + x_{121} + x_{131} + x_{141} + x_{211} + x_{221} + x_{231} + x_{241} + x_{311} + x_{321} + x_{331}$

$+ x_{341} + x_{411} + x_{421} + x_{431} + x_{441} <= 1$

$x_{011} + x_{021} + x_{031} + x_{041} = x_{101} + x_{201} + x_{301} + x_{401} = 1$

$y_{11} + y_{21} + y_{31} + y_{41} <= 29$

$x_{021} + x_{031} + x_{041} + x_{121} + x_{131} + x_{141} >= y_{01} - 1$

$$x_{021} + x_{031} + x_{041} + x_{121} + x_{131} + x_{141} >= y_{11} - 1$$

$$x_{011} + x_{031} + x_{041} + x_{211} + x_{231} + x_{241} >= y_{01} - 1$$

$$x_{011} + x_{031} + x_{041} + x_{211} + x_{231} + x_{241} >= y_{21} - 1$$

$$x_{011} + x_{021} + x_{041} + x_{311} + x_{321} + x_{341} >= y_{01} - 1$$

$$x_{011} + x_{021} + x_{041} + x_{311} + x_{321} + x_{341} >= y_{31} - 1$$

$$x_{011} + x_{021} + x_{031} + x_{411} + x_{421} + x_{431} >= y_{01} - 1$$

$$x_{011} + x_{021} + x_{031} + x_{411} + x_{421} + x_{431} >= y_{41} - 1$$

$$x_{101} + x_{131} + x_{141} + x_{201} + x_{231} + x_{241} >= y_{11} - 1$$

$$x_{101} + x_{131} + x_{141} + x_{201} + x_{231} + x_{241} >= y_{21} - 1$$

$$x_{101} + x_{121} + x_{141} + x_{301} + x_{321} + x_{341} >= y_{11} - 1$$

$$x_{101} + x_{121} + x_{141} + x_{301} + x_{321} + x_{341} >= y_{31} - 1$$

$$x_{101} + x_{121} + x_{131} + x_{401} + x_{421} + x_{431} >= y_{11} - 1$$

$$x_{101} + x_{121} + x_{131} + x_{401} + x_{421} + x_{431} >= y_{41} - 1$$

$$x_{201} + x_{211} + x_{241} + x_{301} + x_{311} + x_{341} >= y_{21} - 1$$

$$x_{201} + x_{211} + x_{241} + x_{301} + x_{311} + x_{341} >= y_{31} - 1$$

$$x_{201} + x_{211} + x_{231} + x_{401} + x_{411} + x_{431} >= y_{21} - 1$$

$$x_{201} + x_{211} + x_{231} + x_{401} + x_{411} + x_{431} >= y_{41} - 1$$

$$x_{301} + x_{311} + x_{321} + x_{401} + x_{411} + x_{421} >= y_{31} - 1$$

$$x_{301} + x_{311} + x_{321} + x_{401} + x_{411} + x_{421} >= y_{41} - 1$$

$$x_{031} + x_{041} + x_{131} + x_{141} + x_{231} + x_{241} >= y_{01} - 1$$

$$x_{031} + x_{041} + x_{131} + x_{141} + x_{231} + x_{241} >= y_{11} - 1$$

$$x_{031} + x_{041} + x_{131} + x_{141} + x_{231} + x_{241} >= y_{21} - 1$$

$$x_{021} + x_{041} + x_{121} + x_{141} + x_{321} + x_{341} >= y_{01} - 1$$

$$x_{021} + x_{041} + x_{121} + x_{141} + x_{321} + x_{341} >= y_{11} - 1$$

$$x_{021} + x_{041} + x_{121} + x_{141} + x_{321} + x_{341} >= y_{31} - 1$$

$$x_{021} + x_{031} + x_{121} + x_{131} + x_{421} + x_{431} >= y_{01} - 1$$

$$x_{021} + x_{031} + x_{121} + x_{131} + x_{421} + x_{431} >= y_{11} - 1$$

$$x_{021} + x_{031} + x_{121} + x_{131} + x_{421} + x_{431} >= y_{41} - 1$$

$x_{011} + x_{041} + x_{211} + x_{241} + x_{311} + x_{341} >= y_{01} - 1$

$x_{011} + x_{041} + x_{211} + x_{241} + x_{311} + x_{341} >= y_{21} - 1$

$x_{011} + x_{041} + x_{211} + x_{241} + x_{311} + x_{341} >= y_{31} - 1$

$x_{011} + x_{031} + x_{211} + x_{231} + x_{411} + x_{431} >= y_{01} - 1$

$x_{011} + x_{031} + x_{211} + x_{231} + x_{411} + x_{431} >= y_{21} - 1$

$x_{011} + x_{031} + x_{211} + x_{231} + x_{411} + x_{431} >= y_{41} - 1$

$x_{011} + x_{021} + x_{311} + x_{321} + x_{411} + x_{421} >= y_{01} - 1$

$x_{011} + x_{021} + x_{311} + x_{321} + x_{411} + x_{421} >= y_{31} - 1$

$x_{011} + x_{021} + x_{311} + x_{321} + x_{411} + x_{421} >= y_{41} - 1$

$x_{101} + x_{141} + x_{201} + x_{241} + x_{301} + x_{341} >= y_{11} - 1$

$x_{101} + x_{141} + x_{201} + x_{241} + x_{301} + x_{341} >= y_{21} - 1$

$x_{101} + x_{141} + x_{201} + x_{241} + x_{301} + x_{341} >= y_{31} - 1$

$x_{101} + x_{131} + x_{201} + x_{231} + x_{401} + x_{431} >= y_{11} - 1$

$x_{101} + x_{131} + x_{201} + x_{231} + x_{401} + x_{431} >= y_{21} - 1$

$x_{101} + x_{131} + x_{201} + x_{231} + x_{401} + x_{431} >= y_{41} - 1$

$x_{101} + x_{121} + x_{301} + x_{321} + x_{401} + x_{421} >= y_{11} - 1$

$x_{101} + x_{121} + x_{301} + x_{321} + x_{401} + x_{421} >= y_{31} - 1$

$x_{101} + x_{121} + x_{301} + x_{321} + x_{401} + x_{421} >= y_{41} - 1$

$x_{201} + x_{211} + x_{301} + x_{311} + x_{401} + x_{411} >= y_{21} - 1$

$x_{201} + x_{211} + x_{301} + x_{311} + x_{401} + x_{411} >= y_{31} - 1$

$x_{201} + x_{211} + x_{301} + x_{311} + x_{401} + x_{411} >= y_{41} - 1$

$x_{011} + x_{101} <= 1$

$x_{021} + x_{201} <= 1$

$x_{031} + x_{301} <= 1$

$x_{041} + x_{401} <= 1$

$x_{101} + x_{011} <= 1$

$x_{121} + x_{211} <= 1$

$x_{131} + x_{311} <= 1$

$$x_{141} + x_{411} <= 1$$

$$x_{201} + x_{021} <= 1$$

$$x_{211} + x_{121} <= 1$$

$$x_{231} + x_{321} <= 1$$

$$x_{241} + x_{421} <= 1$$

$$x_{301} + x_{031} <= 1$$

$$x_{311} + x_{131} <= 1$$

$$x_{321} + x_{231} <= 1$$

$$x_{341} + x_{431} <= 1$$

$$x_{401} + x_{041} <= 1$$

$$x_{411} + x_{141} <= 1$$

$$x_{421} + x_{241} <= 1$$

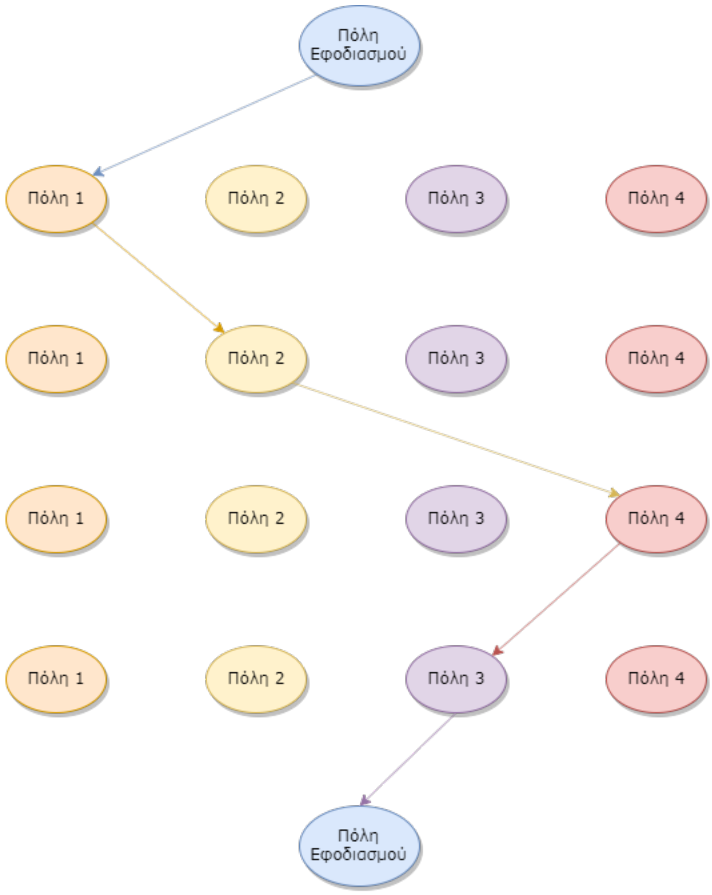
$$x_{431} + x_{341} <= 1$$

3.4.2 Παράδειγμα Δεύτερο

Στο δεύτερο παράδειγμα, θα δούμε ένα λίγο μεγαλύτερο πρόβλημα, όμως αυτήν τη φορά με 5 πόλεις/πελάτες. Επιπλέον, η εταιρία που αναλύουμε σε αυτό το παράδειγμα έχει στο στόλο της δύο οχήματα, τα οποία καλούνται να καλύψουν την ζήτηση ταυτόχρονα. Τα δύο οχήματα έχουν μέγιστη χωρητικότητα 20 προϊόντων, ενώ οι απαιτήσεις που καλούνται να καλύψουν τα δύο οχήματα είναι αρκετά μεγαλύτερη από 20 προϊόντα, με αποτέλεσμα να μη μπορεί να καλυφθεί από ένα μόνο όχημα. Παρακάτω, βλέπουμε τον πίνακα 3.4 με τα αναλυτικά κόστη κάθε διαδρομής για όλες τις πόλεις που έχουν να επισκεφτούν τα οχήματα, όπως επίσης βλέπουμε και τις απαιτήσεις κάθε πελάτη.

Το πρώτο μοντέλο εκτυπώνει ως βέλτιστη συνολική διαδρομή για τα δύο οχήματα 15. Αυτή τη φορά, το πρόγραμμα έχει να διαχειριστεί δρομολόγια για δύο οχήματα των οποίων οι διαδρομές, όπως έχουμε προαναφέρει, δεν πρέπει να συγκρούονται μεταξύ τους. Το πρώτο όχημα, είναι αυτό που θα ακολουθήσει την μεγαλύτερη

Σχήμα 3.7: Διαμόρφηση Διαγράμματος Δέυτερου Μοντέλο για Παράδειγμα 1



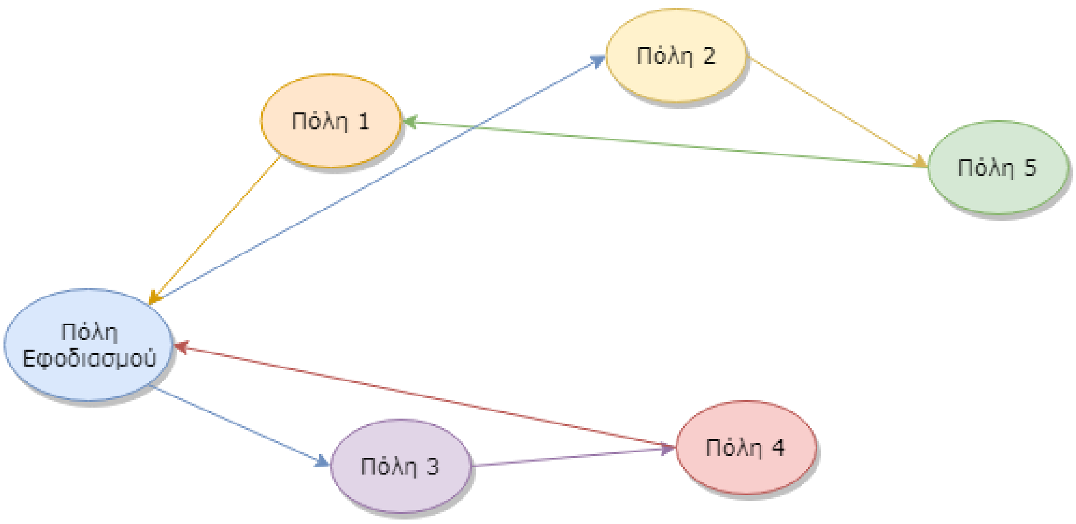
διαδρομή περνώντας πρώτα από την πόλη 2, συνεχίζει στην πόλη 5 και πριν ολοκληρώσει το δρομολόγιο επισκέπτεται την πόλη 1. Στο τέλος, το πρώτο όχημα επιστρέφει στη πόλη εφοδιασμού, έχοντας απόθεμα δύο προϊόντων στη διάθεσή του. Αυτό δεν σημαίνει απαραίτητα ότι υπάρχει λάθος, αφού κανένας κανονισμός του CVRP δεν απαγορεύει σε ένα όχημα να ολοκληρώσει το δρομολόγιό του, χωρίς να έχει παραδώσει όλα τα προϊόντα που μεταφέρει. Στη συνέχεια, βλέπουμε το δρομολόγιο του δεύτερου οχήματος, το οποίο ακολούθησε μια πιο μικρή διαδρομή και εξυπηρέτησε πόλεις 3 και 4, παραδίδοντας 17 προϊόντα και επιστρέφει με 3 προϊόντα απόθεμα, όπως και στο πρώτο όχημα. Στο σχήμα 3.8, βλέπουμε το διάγραμμα που δημιουργήθηκε. Να σημειωθεί εδώ ότι και για τα δύο μοντέλα, η δημιουργία της αντικειμενικής συνάρτησης και των περιορισμών έγινε με την ίδια λογική όπως και στο πρώτο παράδειγμα.

Το δεύτερο μοντέλο σε αυτό το παράδειγμα δημιουργεί μία ελαφρώς διαφορετική λύση από αυτή του πρώτου, χωρίς όμως να επηρεάζει την λύση του προβλήματος, βγάζοντας διαφορετικό αποτέλεσμα. Όπως στο κλασικό μοντέλο έτσι και εδώ, η

Πίνακας 3.4: Πίνακας Δεύτερου Παραδείγματος

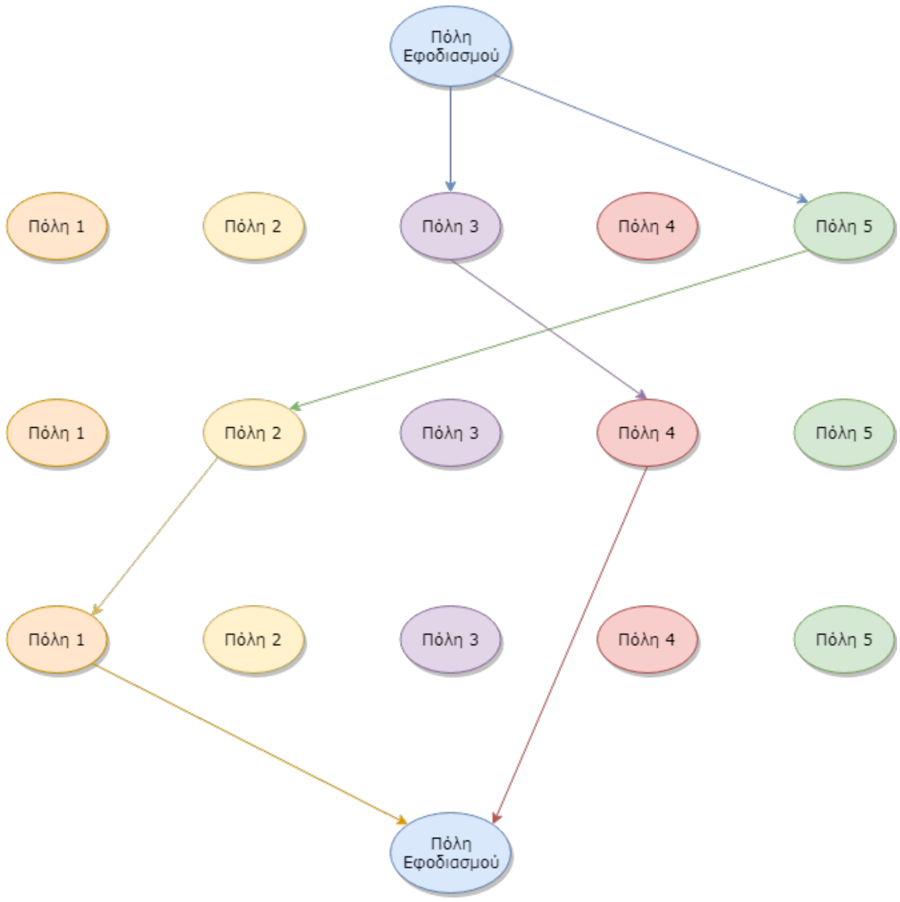
	Πόλη 0 (Εφοδιασμού)	Πόλη 1	Πόλη 2	Πόλη 3	Πόλη 4	Πόλη 5	Απαιτήσεις
Πόλη 0 (Εφοδιασμού)	-	1	3	2	4	5	0
Πόλη 1	1	-	1	2	4	3	5
Πόλη 2	3	1	-	2	2	1	7
Πόλη 3	2	2	2	-	1	3	8
Πόλη 4	4	4	2	1	-	2	9
Πόλη 5	5	3	1	3	2	-	6

Σχήμα 3.8: Διαμόρφηση Διαγράμματος Πρώτου Μοντέλο για Παράδειγμα 2



συνολική απόσταση που θα διανύσουν τα δύο οχήματα είναι ίση με 15. Σύμφωνα με το σχήμα 3.9, η διαδρομή του πρώτου οχήματος, το πρόγραμμα αποφασίζει να στείλει το όχημα αρχικά στη πόλη 5, ύστερα στη πόλη 2 και τέλος, πριν από την πόλη εφοδιασμού να περάσει από την πόλη 1. Η αλλαγή αυτή δεν έχει ούτε θετική αλλά ούτε και αρνητική επίδραση στη βέλτιστη λύση. Αν παρατηρήσουμε καλά τον πίνακα 3.4, θα δούμε ότι οι πόλεις 5 και 2 έχουν το ίδιο κόστος, εάν κατευθυνθούμε από την μία στην άλλη, και αντίστροφα. Εδώ, πιστώνεται η παρατήρησή μας στα αποτελέσματα του πρώτου οχήματος, δηλαδή ότι τα δύο μοντέλα μπορούν να ακολουθήσουν διαφορετικές διαδρομές, δίχως να έχουν διαφορετική βέλτιστη τιμή.

Σχήμα 3.9: Διαμόρφωση Διαγράμματος Δέυτερου Μοντέλο για Παράδειγμα 2



Κεφάλαιο 4

Πειραματικό Μέρος

Το κεφάλαιο αυτό είναι αφιερωμένο στην διεξοδική ανάλυση κυρίως των δύο μοντέλων και των απαιτήσεών τους για μία ολοκληρωμένη μοντελοποίηση ενός οποιουδήποτε προβλήματος. Συνοπτικά, στο κεφάλαιο αυτό θα δούμε αρχικά τον συνολικό αριθμό μεταβλητών που δημιουργούν τα δύο μοντέλα, καθώς επίσης και το συνολικό αριθμό περιορισμών που θα προκύψουν έπειτα από επεξεργασία όλων των μεταβλητών. Στη συνέχεια, θα παρουσιάσουμε τα εργαλεία που χρησιμοποιήσαμε για την εργασία. Έπειτα, θα αναλύσουμε το πρόγραμμα CPLEX, που ήταν το πρόγραμμα που βοήθησε στη συνολική επίτευξη των στόχων αυτής της εργασίας. Τέλος, θα δούμε αναλυτικά όλα τα αποτελέσματα από τα πειράματα που εκτελέσαμε κατά τη διάρκεια εκπόνησης της έρευνας.

4.1 Απαιτήσεις κάθε μοντέλου και δημιουργία μεταβλητών και περιορισμών

Τα δύο μοντέλα μας, όπως έχουμε αναφέρει σε προηγούμενα κεφάλαια, παρουσιάζουν σημαντικές διαφορές στη δημιουργία των περιορισμών. Το κλασικό μοντέλο, υπολογίζει την μετάβαση των οχημάτων προς την πόλη εφοδιασμού κάθε φορά που υπάρχει ανάγκη για απόφαση ποιας πόλης θα ακολουθήσει μετά την εξυπηρέτηση μίας άλλης. Από την άλλη πλευρά, στο δεύτερο μοντέλο, λαμβάνουμε υπόψιν πολύ περισσότερες μεταβάσεις και επίσης δεν μεταφερόμαστε οποιαδήποτε στιγμή, με οποιοδήποτε όχημα, στην πόλη ανεφοδιασμού.

Το αποτελέσματα των πινάκων 4.1 ,4.2 και 4.3 , αντικατοπτρίζουν τον αριθμό των μεταβλητών που δημιουργεί κάθε μοντέλο και το πόσοι περιορισμοί προκύπτουν

Πίνακας 4.1: Πίνακας απαιτήσεων για πειράματα με ένα(1) όχημα

	Πρώτο Μοντέλο		Δεύτερο Μοντέλο	
Αριθμός Πόλεων	Μεταβλητές	Περιορισμοί	Μεταβλητές	Περιορισμοί
10	100	5.141	109	5.160
15	225	245.791	239	245.820
20	400	10.485.801	419	10.485.840
25	625	419.430.451	649	419.430.500

Πίνακας 4.2: Πίνακας απαιτήσεων για πειράματα με δύο(2) οχήματα

	Πρώτο Μοντέλο		Δεύτερο Μοντέλο	
Αριθμός Πόλεων	Μεταβλητές	Περιορισμοί	Μεταβλητές	Περιορισμοί
10	200	10.172	218	10.189
15	400	491.342	478	491.369
20	800	20.971.182	838	20.971.219
25	1250	838.860.252	1298	838.860.299

από το συνδυασμό όλων αυτών των μεταβλητών, για την απόφαση της βέλτιστης λύσης. Ο υποθετικός αριθμός πόλεων που εξετάζουμε στους παρακάτω πίνακες είναι 10, 15, 20 και 25, ενώ ο αριθμός των διαθέσιμων οχημάτων είναι από 1 έως και 3.

Αρχικά, γίνεται αντιληπτό ότι το μοντέλο βασισμένο στα διαγράμματα αποφάσεων δημιουργεί σταθερά περισσότερους περιορισμούς από το κλασικό μοντέλο. Επίσης, όπως είναι φυσικό, το δεύτερο μοντέλο διαχειρίζεται περισσότερες μεταβλητές. Κάτι τέτοιο, θα μπορούσε να οδηγήσει στην υπερβολική διαφορά μεταξύ των συνολικών περιορισμών που δημιουργούν τα δύο μοντέλα και να κάνει το δεύτερο μοντέλο μας ανούσιο για αποτελεσματική και γρήγορη επίλυση προβλημάτων, σε σύγκριση με το κλασικό μοντέλο. Αντιθέτως, όμως, όπως έχουμε αναφέρει στο Κεφάλαιο 3, όπου παρουσιάσαμε τα δύο μοντέλα αναλυτικά, το μοντέλο βασισμένο στα διαγράμματα αποφάσεων σε κάθε περιορισμό υπολογίζει μία λιγότερη περίπτωση. Την περίπτωση της μεταφοράς ενός οχήματος από την

Πίνακας 4.3: Πίνακας απαιτήσεων για πειράματα με τρία(3) οχήματα

	Πρώτο Μοντέλο		Δεύτερο Μοντέλο	
Αριθμός Πόλεων	Μεταβλητές	Περιορισμοί	Μεταβλητές	Περιορισμοί
10	300	15.203	327	15.218
15	675	736.893	717	736.918
20	1200	31.456.563	1257	31.456.598
25	1875	1.258.290.053	1947	1.258.290.098

πόλη που βρίσκεται προς την πόλη εφοδιασμού σε οποιοδήποτε χρονικό σημείο. Αυτό ελαφραίνει κάπως το βάρος του δεύτερου μοντέλου. Δεν είναι όμως αρκετό για να αποδειχθεί ταχύτερο από το κλασικό μοντέλο και χρήζει δουλειάς προκειμένου να βελτιωθεί.

Και στα δύο μοντέλα μπορούμε να παρατηρήσουμε ότι, καθώς ανεβαίνουμε σε αριθμό πόλεων ο αριθμός περιορισμών που δημιουργούν τα δύο μοντέλα αυξάνεται ραγδαία. Εν αντιθέσει με των αριθμό των πόλεων, ο αριθμός των οχημάτων που μπορεί να έχει μία εταιρεία στη διάθεσή της, δεν επηρεάζει τόσο σημαντικά τον αριθμό των περιορισμών όσο αυξάνονται. Σε γενικό βαθμό, όπως μπορεί κανείς εύκολα να παρατηρήσει ότι, από το σημείο που φτάνουμε στις 25 πόλεις προς αναζήτηση βέλτιστης λύσης, ο αριθμός περιορισμών που δημιουργούνται είναι πολύ μεγάλος ακόμη και στη περίπτωση που τις παραγγελίες ολοκληρώνει μόνο ένα όχημα. Με λίγα λόγια, στον τομέα επιβάρυνσης του συστήματος με περιορισμούς, ο παράγοντας που επιβραδύνει το συνολικό χρόνο απόφασης βέλτιστης λύσης είναι με μεγάλη διαφορά οι πόλεις/πελάτες που πρέπει να εξυπηρετήσει μια εταιρία, παρά ο αριθμός των οχημάτων που διαθέτει αυτή η εταιρεία στο στόλο της.

4.2 Εργαλεία που χρησιμοποιήσαμε

Κατά τη διάρκεια εκπόνησης της εργασίας, χρησιμοποιήσαμε πολλά προγράμματα για την επιτυχία των επιθυμητών αποτελεσμάτων. Μερικά προγράμματα είχαν μεγαλύτερο ρόλο από κάποια άλλα. Επίσης, ορισμένα προγράμματα αποτέλεσαν καθημερινά εργαλεία για την διευκόλυνση κατανόησης ορισμένων στοιχείων αλλά πιο σημαντικά στην ολοκλήρωση της εργασίας.

Αρχικά, έγινε χρήση του LINGO (είχαμε στη διάθεσή μας την έκδοση 18.0). Το LINGO [1], χρησιμοποιήθηκε για την πρώιμη δοκιμή των αποτελεσμάτων των δύο μοντέλων. Είναι ένα πρόγραμμα που λειτουργεί ως λύτης προβλημάτων γραμμικού ακέραιου προγραμματισμού. Μας έδωσε την δυνατότητα να βελτιώσουμε το μοντέλο βασισμένο σε διαγράμματα αποφάσεων, και να καταλήξουμε στην τελική μορφή του έπειτα από συνεχείς δοκιμές. Με το LINGO, κάναμε σύγκριση των συνολικών περιορισμών που δημιουργεί το μοντέλο, είδαμε ποιοι περιορισμοί έχουν την μεγαλύτερη σημασία, κάναμε τις απαραίτητες διορθώσεις προκειμένου

[1] https://www.lindo.com/index.php/products/lingo-and-optimization-modeling

η λύση που θα έχουμε στα χέρια μας να είναι πάντα εφικτή και βέλτιστη, έτσι ώστε να συνεχίσουμε στην αυτοματοποίηση δημιουργίας του μοντέλου με το κύριο πρόγραμμα.

Στη συνέχεια, θα μιλήσουμε για το γενικό πρόγραμμα δημιουργίας και επεξεργασίας του κώδικα των δύο μοντέλων. Το Visual Studio (χρησιμοποιήσαμε την έκδοση 15.9), είναι το πρόγραμμα με το οποίο συγκεντρώσαμε όλα τα στοιχεία που είχαμε, το συνδέσαμε με το πρόγραμμα που μας έδωσε τη βέλτιστη λύση το CPLEX, και καταφέραμε να αυτοματοποιήσουμε την διαδικασία βελτιστοποίησης. Η δουλειά του Visual Studio ήταν να διαβάσει τα αρχεία που περιείχαν τα στοιχεία των πειραμάτων, να τα αναλύσει σε πίνακες που το CPLEX μπορεί να επεξεργαστεί, και να μας εκτυπώσει το αποτέλεσμα. Η συγγραφή του κώδικα έγινε σε γλώσσα C.

4.3 Ο λύτης CPLEX

Το CPLEX[2], είναι ένα χαρακτηριστικό του περιβάλλοντος βελτιστοποίησης IBM ILOG, και προσφέρει τις καλύτερες επιδόσεις για την επίλυση προβλημάτων που εκφράζονται ως μαθηματικά μοντέλα προγραμματισμού. Είναι συμβατό με το Visual Studio και ο συνδυασμός των δύο αυτών προγραμμάτων μας δίνει τη δυνατότητα να γράψουμε τον κώδικα με τον οποίο κάναμε τη σύγκριση των δύο μοντέλων. Είναι συμβατό με πολλές γλώσσες προγραμματισμού ανάμεσά του η C, η C++, η Java, η Python και η MATLAB. Εμείς επιλέξαμε την C για την υλοποίηση της εργασίας. Στη συνέχεια, θα αναλύσουμε τις πιο σημαντικές συναρτήσεις του CPLEX και θα εξηγήσουμε την χρησιμότητά τους.

CPXopenCPLEX(status): Κάνει την αρχικοποίηση του CPLEX περιβάλλοντος. Η CPXopenCPLEX είναι η πρώτη συνάρτηση που πρέπει να καλεστεί. Επιστρέφει έναν δείκτη στο περιβάλλον CPLEX. Αυτός ο δείκτης χρησιμοποιείται σε παρακάτω συναρτήσεις. Εάν υπάρξει κάποιο πρόβλημα, η τιμή που επιστρέφεται είναι NULL. Σε περίπτωση που η συνάρτηση δεν αντιμετωπίσει σημαντικά προβλήματα, τότε ο δείκτης status παίρνει την τιμή μηδέν.

CPXcreateprob(env, status, "myprob"): Η συνάρτηση CPXcreateprob δημιουργεί ένα αντικείμενο προβληματος CPLEX στο περιβάλλον CPLEX. Το τρίτο στοιχείο

[2] `https://www.ibm.com/products/ilog-cplex-optimization-studio?mhsrc=ibmsearch_a&mhq=CPLEX`

στις εισόδους της συνάρτησης δηλώνει το όνομα του αρχείου που θα δημιουργήσει το CPLEX, το οποίο θα έχει μορφή ακέραιου γραμμικού προγραμματισμού. Το πρόβλημα που δημιουργείται είναι ένα LP (Linear Programming) πρόβλημα ελαχιστοποίησης με κανέναν περιορισμό, καμία μεταβλητή και έναν άδειο πίνακα περιορισμών. Νέες μεταβλητές ή περιορισμοί μπορούν να δημιουργηθούν με τις συναρτήσεις, CPXnewrows ή CPXnewcols αντίστοιχα. Τα άλλα δύο στοιχεία εισόδου της συνάρτησης είναι, ένας δείκτης με το αποτέλεσμα της συνάρτησης CPXopenCPLEX και η κατάσταση της συνάρτησης. Όπως και πριν, σε περίπτωση που παρουσιαστεί πρόβλημα επιστρέφεται NULL, ενώ σε αντίθετη περίπτωση το status παίρνει τιμή 0.

populatebyrow(env, lp): Είναι η βασική συνάρτηση που χρησιμοποιούμε για να γεμίσουμε τους πίνακες που διαχειρίζεται το CPLEX, κατά σειρά. Στο χρήστη, δίνεται από την αρχή η δυνατότητα επιλογής ανάμεσα σε γέμισμα κατά σειρά, κατά στήλη (populatebycolumn) και κατά μη μηδενικά στοιχεία (populatebynonzero). Τα δύο ορίσματα της συνάρτησης είναι οι δείκτες με τιμή ίση με τα αποτελέσματα των συναρτήσεων CPXopenCPLEX και CPXcreateprob.

CPXmipopt(env, lp): Αφού δημιουργηθεί το πρόβλημα μεικτών ακεραίων στην CPXcreateprob, και συμπληρωθούν οι απαραίτητοι πίνακες στην populatebyrow, η CPXmipopt, βρίσκει τη βέλτιστη λύση στο πρόβλημα. Λύση LP δεν υπάρχει ακριβώς μετά την χρήστη της CPXmipopt. Τα συνολικά στοιχεία της λύσης του προβλήματος προκύπτουν από την χρήση και της CPXchgprobtype στη συνέχεια του κώδικα. Πιθανό προβλήματα που μπορεί να παρουσιάσει το πρόγραμμα, είναι η δέσμευση περισσότερης από την διαθέσιμη μνήμη, άρα θα εκτυπωθεί CPXERR_ NO_ MEMORY. Ένα άλλο πρόβλημα που δημιουργείται συνήθως, είναι να συμπληρωθούν οι πίνακες του CPLEX, με λάθος στοιχεία, όπως για παράδειγμα την εκχώρηση boolean μεταβλητής σε πίνακα με integer στοιχεία. Σε αυτή τη περίπτωση, το πρόγραμμα θα εκτυπώσει CPXERR_ NO_ PROBLEM.

CPXgetstat(env, lp): Δέχεται ως ορίσματα τους δείκτες με τα αποτελέσματα των συναρτήσεων CPXopenCPLEX και CPXcreateprob. Χρησιμοποιείται για να ενημερώσει το χρήστη για το είδος του προβλήματος βελτιστοποίησης, δηλαδή αν είναι γραμμικού ακέραιου προγραμματισμού ή μεικτού ακέραιου γραμμικού προγραμματισμού. Σε περίπτωση που επιστραφεί η τιμή 0 τότε σημαίνει ότι προέκυψε προβληματική κατάσταση, ή κάποια αλλαγή στη πιο πρόσφατη κατάσταση της λύσης, μπορεί να παράβει του κανόνες της βελτιστοποίησης. Σε περίπτωση που εκτυπωθεί CPX_ STAT_ NUM_ BEST, σημαίνει ότι ο αλγόριθμος δεν μπορούσε να συγκλίνει στις ζητούμενες ανοχές λόγω αριθμητικών δυσκολιών.

CPXgetobjval(env, lp, objval): Η συνάρτηση χρησιμοποιείται για να είναι σε θέση το πρόγραμμα να γνωρίζει, ποια είναι η βέλτιστη τιμή του προβλήματος. Ως ορίσματα βλέπουμε πρώτα την τιμή που έδωσε η συνάρτηση CPXopenCPLEX στον δείκτη env, στη συνέχεια την τιμή της συνάρτησης CPXcreateprob από τον δείκτη lp και τέλος τον δείκτη του οποίου η τιμή είναι ίση με τη βέλτιστη τιμή. Η συνάρτηση CPXgetobjval επιστρέφει στον δείκτη status την τιμή 0, εάν εκτελέστηκε επιτυχημένα και διαφορετική του μηδενός, όταν αντιμετωπίστηκε κάποιο πρόβλημα.

CPXwriteprob(env, lp, "myprob", LP): Εφόσον βρεθεί η βέλτιστη λύση, το CPLEX, με την χρήση της συνάρτησης CPXwriteprob, δημιουργεί ένα αρχείο στο οποίο γράφει με μορφή γραμμικού προγραμματισμού την κατάσταση του προβλήματος. Για άλλη μια φορά, τα δύο πρώτα ορίσματα τις συνάρτησης είναι τα env και lp. Το επόμενο όρισμα, είναι ένα αλφαριθμητικό που υποδηλώνει τον όνομα του αρχείου, μέσα στο οποίο θα γραφτεί το πρόβλημα. Το τελευταίο όρισμα, είναι ένα εξίσου αλφαριθμητικό που καθορίζει τον τύπο που θα έχει το αρχείο μέσα στο οποίο θα γραφεί το πρόβλημα. Το αλφαριθμητικό αυτό στη προκειμένη περίπτωση είναι LP, που ως γνωστό σημαίνει ότι η αναπαράσταση του προβλήματος θα έχει τη μορφή ακέραιου γραμμικού προγραμματισμού. Μία άλλη τιμή είναι η RLP, δηλαδή το πρόβλημα να είναι πάλι γραμμένο ως πρόβλημα ακέραιου γραμμικού προγραμματισμού αλλά όλα τα ονόματα να είναι αλλαγμένα σε γενικευμένα ονόματα. Επιπλέον, δίνεται η δυνατότητα στο χρήστη να αναπαρασταθούν τα στοιχεία του αρχείου σε έναν δυαδικό πίνακα και ένα αρχείο βάσης, χρησιμοποιώντας το αλφαριθμητικό SAV. Τέλος, να σημειωθεί ότι το τελευταίο αλφαριθμητικό μπορεί να αγνοηθεί, αλλάζοντας την τιμή σε NULL, εάν το αλφαριθμητικό του τρίτου ορίσματος έχει την ανάλογη κατάληξη στο τέλος του ονόματος του αρχείου, δηλαδή .sav, .lp, .rlp κτλ. Όπως και σε παραπάνω συναρτήσεις, η CPXwriteprob επιστρέφει 0, εάν δεν αντιμετωπίσει κάποιο πρόβλημα και μη μηδενική τιμή σε αντίθετη περίπτωση.

4.4 Εκπόνηση πειραμάτων και συμπεράσματα

Βάση των δοκιμών που κάναμε στη ενότητα 4.1, είναι εύκολο να αντιληφθεί κανείς ότι η τιμές στον πίνακα που θα ακολουθήσει, είναι και οι αναμενόμενες. Βέβαια, μπορεί να είναι ευνόητο το γεγονός ότι το πρώτο μοντέλο χρειάζεται λιγότερο χρόνο για την εξαγωγή της βέλτιστης λύσης αλλά, το ποσοστό, κατά το οποίο είναι ταχύτερο συγκρίσει με το δεύτερο μοντέλο, είναι αρκετά ενδιαφέρον.

Ο εξοπλισμός που χρειαστήκαμε για την εκπόνηση των παραπάνω πειραμάτων και την εξαγωγή των αποτελεσμάτων στο καλύτερο δυνατό χρόνο, αποτελείται από έναν επεξεργαστή Intel Core i7 3.4 GHz, με κεντρική μνήμη 32 GB και 8 πυρήνες, χρονισμό στα 3700MHz, μία L1 cache μνήμη κώδικα των 32KB σε κάθε πυρήνα, μία L1 cache μνήμη δεδομένων των 32KB σε κάθε πυρήνα, μία L2 cache μνήμη των 256KB σε κάθε πυρήνα, μία L3 cache μνήμη των 8MB και bandwidth μνήμης στα 21Gb/s ο οποίος λειτουργεί σε Microsoft Windows 7 64-bit.

Στον παρακάτω πίνακα, βλέπουμε τους χρόνους εκτέλεσης του κλασικού μοντέλου και του μοντέλου βασισμένο σε διαγράμματα αποφάσεων. Τα πειράματα που χρησιμοποιήσαμε δημιουργήθηκαν με τυχαίους αριθμούς. Το περιβάλλον στο οποίο υποθέτουμε ότι βρίσκονται οι πόλεις είναι ένας χάρτης με μήκος και πλάτος 100 χιλιομέτρων. Επομένως και οι τοποθεσίες των πόλεων/πελατών βρίσκονται σε απόλυτα τυχαίες θέσεις σε κάθε πείραμα ξεχωριστά. Για να επιτύχουμε τα καλύτερα αποτελέσματα χωρίς αμφιβολίες για το αν είναι σταθερά, οι τιμές των αποτελεσμάτων που φαίνονται στον παρακάτω πίνακα αποτελούνται από τον μέσο όρο πέντε ξεχωριστών πειραμάτων που τρέξαμε για κάθε περίπτωση. Στην προκειμένη περίπτωση, φαίνονται τα αποτελέσματα από 22 πειράματα, στην πραγματικότητα όμως έχουν γίνει δοκιμές για την ορθή λειτουργία των δύο μοντέλων σε 110 πειράματα.

Στην πρώτη στήλη του πίνακα που ακολουθεί, θα υποδηλώνεται το όνομα του κάθε πειράματος. Η τιμή δίπλα στο n υποδεικνύει τον αριθμό των πόλεων/πελατών που εξετάζουμε σε αυτό το πείραμα. Επίσης, ο αριθμός δίπλα στο k μας ενημερώνει για το πόσα οχήματα έχουν οι εταιρείες στη διάθεσή τους σε κάθε πείραμα, για την διανομή των προϊόντων. Τα πειράματα είναι για πόλεις από 5 έως 19 αυξάνοντας δύο κάθε φορά, και τα οχήματα εκτός από την πρώτη και την τελευταία περίπτωση

Πίνακας 4.4: Πίνακας Αποτελεσμάτων

		Πρώτο Μοντέλο	Δεύτερο Μοντέλο
Πείραμα	**Βέλτιστη Τιμή**	**Χρόνος Εύρεσης**	**Βέλτιστης Λύσης (sec)**
N-n5-k1	194	0,018	0,024
N-n5-k2	303	0,014	0,018
N-n7-k1	224	0,018	0,020
N-n7-k2	351	0,116	0,122
N-n7-k3	419	0,018	0,018
N-n9-k1	266	0,018	0,022
N-n9-k2	354	0,260	0,206
N-n9-k3	375	0,265	0,328
N-n11-k1	274	0,082	0,130
N-n11-k2	433	0,158	0,190
N-n11-k3	445	0,393	0,405
N-n13-k1	296	0,208	0,166
N-n13-k2	395	0,406	0,416
N-n13-k3	435	0,740	0,760
N-n15-k1	324	0,980	1,270
N-n15-k2	385	2,140	2,170
N-n15-k3	451	3,460	3,540
N-n17-k1	344	5,770	6,220
N-n17-k2	415	13,910	14,180
N-n17-k3	418	24,360	24,950
N-n19-k1	354	35,330	38,380
N-n19-k2	386	227,780	308,130

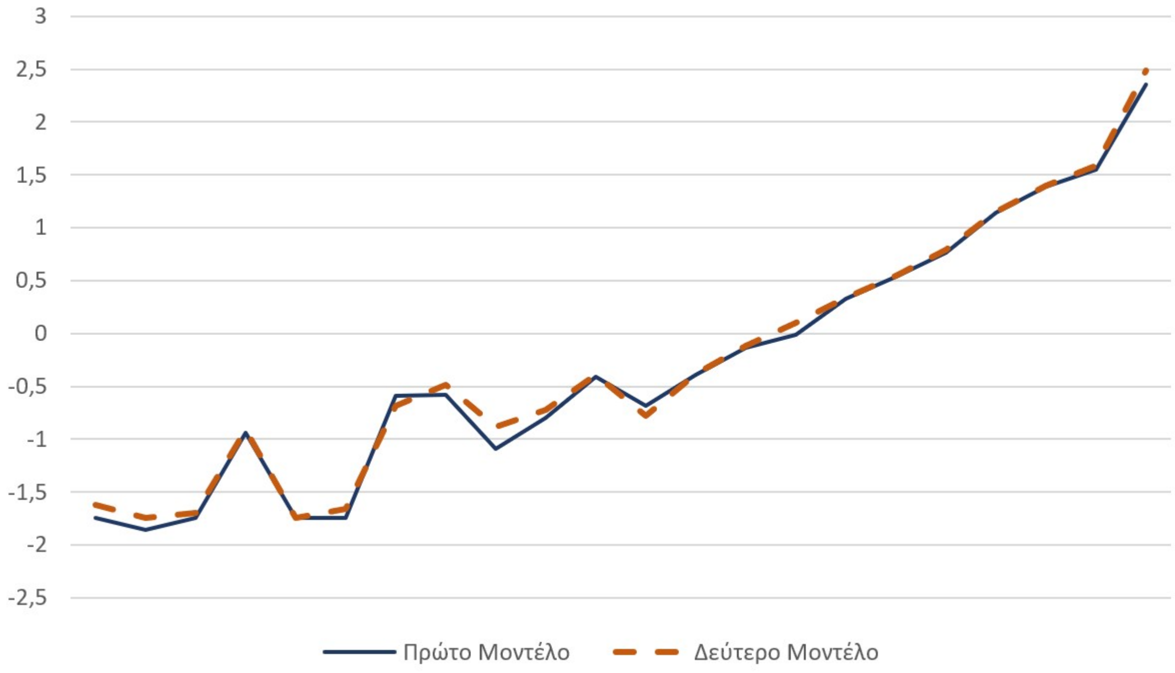
κυμαίνονται από 1 έως 3.

Στη δεύτερη στήλη, βλέπουμε τον αριθμό της μέσης βέλτιστης τιμής της απόστασης που θα διανύσουν τα οχήματα, που σύμφωνα και με τα δύο μοντέλα θα επιφέρουν την ταχύτατη διανομή των προϊόντων στις πόλεις/πελάτες. Είναι σημαντικό να ξέρουμε ότι, για την επίτευξη των σωστών αποτελεσμάτων, τα δύο μοντέλα θα πρέπει να έχουν κοινή βέλτιστη λύση. Σε αντίθετη περίπτωση, το ένα από τα δύο μοντέλα έχει δημιουργήσει λάθος δρομολόγια για το οχήματα και έτσι οδηγήθηκε σε λάθος βέλτιστη τιμή.

Τέλος, η τρίτη και τέταρτη στήλη, παρουσιάζει το χρόνο που χρειάστηκε το κάθε μοντέλο για την εύρεση της βέλτιστης λύσης. Βάσει αυτών των δύο στηλών έγινε και η σύγκριση της απόδοσης των δύο μοντέλων, που μας οδήγησε να προβούμε στα συμπεράσματα, τα οποία παραθέτουμε αμέσως μετά τον πίνακα που ακολουθεί.

Έπειτα από την ολοκλήρωση των πειραμάτων, είχαμε στα χέρια μας τον παραπάνω πίνακα με τις βέλτιστες τιμές των προβλημάτων και τους χρόνους

Σχήμα 4.1: Γραφική Παράσταση Αποτελεσμάτων ($\log_{10}$)



εκτύπωσής τους από το κάθε μοντέλο ξεχωριστά και το σχήμα 4.1, με τη γραφική παράσταση των αποτελεσματων. Για λόγους ευμορφίας, χρησιμοποιήσαμε το λογάριθμο με βάση το 10 στα αποτελέσματα των μοντέλων. Το ζητούμενο μας ήταν, να βρούμε αν το μοντέλο μας βασισμένο στα διαγράμματα αποφάσεων, είναι ανταγωνιστικό απέναντι στο κλασικό μοντέλο που χρησιμοποιείται για Προβλήματα Δρομολόγησης Οχημάτων. Σύμφωνα με τα αποτελέσματα, το πρώτο είναι μοντέλο είναι ταχύτερο σε γενικό βαθμό κατά 11.18%. Το ποσοστό αυτό είναι ένα υπό προϋποθέσεις καλό ποσοστό, αφού μας δίνει τη δυνατότητα με κατάλληλες τροποποιήσεις να το κάνουμε πιο ανταγωνιστικό.

Συγκεκριμένα, το κλασικό μοντέλο ήταν καλύτερο σε σχέση με το δεύτερο περίπου στα μισά από τα πειράματα που εκτελέσαμε. Πιο αναλυτικά, το κλασικό μοντέλο σε δέκα περιπτώσεις ήταν κατά 10% ταχύτερο από το δεύτερο. Επίσης, σε άλλα δέκα πειράματα το κλασικό μοντέλο ήταν λιγότερο από 10% ταχύτερο, πράγμα που σημαίνει ότι σε ορισμένα πειράματα κατεβήκαμε αρκετά παρακάτω σε διαφορά χρόνου από τον μέσο όρο, και σε αυτά τα πειράματα θα μπορούσαμε να εστιάσουμε για την βελτίωση του μοντέλου μας. Βέβαια, υπήρξε μία περίπτωση όπου τα δύο μοντέλα είχαν κατά μέσο όρο την ίδια απόδοση από άποψη χρόνου και επιπλέον μία περίπτωση κατά την οποία το δεύτερο μοντέλο ήταν ταχύτερο από το κλασικό μοντέλο κατά 22%. Η τελευταία αυτή περίπτωση, έχει το πιο ενθαρρυντικό αποτέλεσμα αφού μας αποδεικνύει ότι είναι εφικτό να δημιουργήσουμε ένα πιο βελτιωμένο μοντέλο, βασισμένο στα διαγράμματα αποφάσεων, που θα είναι σε θέση να βγάλει ταχύτερα αποτελέσματα από ένα κλασικό μοντέλο και έτσι, τα διαγράμματα αποφάσεων να αποτελέσουν ένα νέο τρόπο μελέτης των Προβλημάτων Δρομολόγησης Οχημάτων.

Είναι επίσης σημαντικό να παρατηρήσουμε ότι, στο τελευταίο πείραμα το οποίο είναι αναμφίβολα και το μεγαλύτερο πείραμα από άποψη απαιτήσεων αλλά και σε χρόνο που απαιτείται για να εξεταστεί όλο, έχουμε σημαντική διαφορά στην απόδοση των δύο μοντέλων. Σε αυτό το σημείο, τα αποτελέσματα μας δείχνουν ότι χρειάζονται σημαντικές βελτιώσεις, προκειμένου το δεύτερο μοντέλο να είναι πλήρως ανταγωνιστικό με το κλασικό μοντέλο. Επιπλέον, είναι αξιοσημείωτο ότι σε αυτό το πείραμα ο συνολικός χρόνος εύρεσης της βέλτιστης λύσης και για τα δύο μοντέλα ήταν υπερβολικά μεγαλύτερος σε σχέση με όλα τα άλλα πειράματα. Εδώ, πιστώνονται οι παρατηρήσεις μας, από την παρουσίαση που έγινε για τις απαιτήσεις κάθε μοντέλου στην ενότητα 4.1. Είναι αλήθεια ότι, όσο ο αριθμός των πόλεων που πρέπει να ερευνηθούν αυξάνεται, αυξάνεται όλο και περισσότερο ο χρόνος που χρειάζεται το πρόγραμμα να βρει την βέλτιστη λύση. Είναι σίγουρο ότι, σε πολύ μεγαλύτερα πειράματα το πρόγραμμα θα χρειαζόταν ώρες ακόμα και μία ολόκληρη μέρα για καταλήξει σε μία λύση.

Κεφάλαιο 5

Γενικά Συμπεράσματα

Στο 5ο και τελευταίο κεφάλαιο της εργασίας, θα παραθέσουμε τα γενικά συμπεράσματά μας από όλη την δουλειά που πράξαμε. Θα αναλύσουμε την γνώμη μας για μία πιο σωστή προσέγγιση μοντέλου, βασισμένο σε διαγράμματα αποφάσεων και θα σχολιάσουμε το Πρόβλημα Δρομολόγησης Οχημάτων, ως γενικό πρόβλημα στη καθημερινότητα. Ακόμη, θα μιλήσουμε για τις σκέψεις μας πάνω σε μελλοντικές δουλειές στο Πρόβλημα Δρομολόγησης Οχημάτων και ιδιαίτερα στο δεύτερο μοντέλο, οπού θα πούμε αναλυτικά ποιες είναι κατά τη γνώμη μας οι κατάλληλες μετατροπές που θα χρειαζόταν ένα τέτοιο μοντέλο, προκειμένου να γίνει αυτός ο τρόπος σκέψης πιο αποτελεσματικός σε τέτοιου είδους προβλήματα.

5.1 Συμπεράσματα

Αναλογιζόμενοι τα όσα έχουν γραφεί σε αυτήν την εργασία, δεν θα ήταν υπερβολή να πούμε ότι, το Πρόβλημα Δρομολόγησης Οχημάτων είναι ένα πολύ ενδιαφέρον πρόβλημα προς ενασχόληση. Οι διάφορες εκδοχές του δίνουν την δυνατότητα στον ερευνητή να επιλέξει ανάμεσα από μία μεγάλη ποικιλία, ποιο είναι το ακριβές πρόβλημα που θέλουν να ασχοληθούν. Εμείς για την εργασία μας επιλέξαμε την περίπτωση όπου η χωρητικότητα κάθε οχήματος αποτελεί το σημαντικότερο περιορισμό στην δημιουργία της βέλτιστης διαδρομής για τα οχήματα μίας εταιρείας.

Πολλοί επιστήμονες έχουν ασχοληθεί με το Πρόβλημα Δρομολόγησης Οχημάτων και τις διάφορες περιπτώσεις του, όπως αναφέρθηκε στο δεύτερο κεφάλαιο. Από το 1959 και έπειτα, οι ερευνητές μελετούν το πρόβλημα ανακαλύπτοντας νέα μοντέλα για την επίτευξη όλο και καλύτερης λύσης. Στην έρευνα που φέραμε εις πέρας, αποφασίσαμε να ασχοληθούμε με δύο μοντέλα. Το ένα εκ των δύο είναι ένα κλασικό μοντέλο, με το οποίο θα μπορούσαμε να γνωρίσουμε το Πρόβλημα Δρομολόγησης Οχημάτων, να πειραματιστούμε με τους περιορισμούς και να καταλάβουμε ποια είναι τα περιθώρια που μας δίνονται, προκειμένου να μελετήσουμε βαθύτερα το πρόβλημα και να δημιουργήσουμε ένα δικό μας μοντέλο.

Στη συνέχεια, επιχειρήσαμε να κάνουμε μία δική μας προσέγγιση για την εύρεση βέλτιστης λύσης σε Προβλήματα Δρομολόγησης Οχημάτων. Υιοθετήσαμε μία σχετικά πρωτότυπη ιδέα για την μελέτη τέτοιο είδους προβλημάτων, τα διαγράμματα αποφάσεων. Μέσω αυτών, δημιουργήσαμε ένα πρότυπο μοντέλο εύρεσης βέλτιστης λύσης, στηριζόμενοι στο κλασικό μοντέλο που χρησιμοποιήσαμε πρωτίστως. Έπειτα από αρκετές μετατροπές, δοκιμές και πειραματισμούς καταλήξαμε σε ένα συνολικό αποτέλεσμα. Προκειμένου να έχουμε μία γενική ιδέα για το αν, η πρότασή μας είναι ανταγωνιστική απέναντι σε ήδη δοκιμασμένα μοντέλα άλλων ερευνητών, εκτελέσαμε ορισμένα πειράματα συγκρίνοντας τα αποτελέσματα του μοντέλου μας με αυτά του κλασικού μοντέλου.

Τα αποτελέσματά που είχαμε στη διάθεση μας, είναι κάθε άλλο παρά αποθαρρυντικά. Θα μπορούσε κανείς να πει ότι τα αποτελέσματα ήταν και τα αναμενόμενα, αλλά σε ορισμένες περιπτώσεις, μας βοήθησαν να καταλάβουμε ότι οι προσπάθειές μας απέδωσαν καρπούς. Με άλλα λόγια, ενώ σε γενικό βαθμό το κλασικό μοντέλο αποδείχτηκε ταχύτερο από το μοντέλο που δημιουργήσαμε, το ποσοστό κατά το οποίο ήταν ταχύτερο, είναι ένα ποσοστό που μας οδηγεί στο συμπέρασμα ότι σίγουρα σε μία πιο εκτεταμένη ενασχόληση με το μοντέλο μας, θα μπορούσαμε να το κάνουμε ακόμα πιο ανταγωνιστικό. Ιδιαίτερα, σε δύο σημαντικές περιπτώσεις το μοντέλο μας είχε από τη μία ίση απόδοση με το κλασικό μοντέλο και από την άλλη καλύτερη απόδοση. Από αυτό καταλαβαίνουμε ότι, ένα τέτοιου είδους μοντέλο θα μπορούσε με τη κατάλληλη επιμέλεια να ριχτεί στη μάχη, για την εύρεση του ταχύτερου μοντέλου εύρεσης βέλτιστης λύσης για Προβλήματα Δρομολόγησης Οχημάτων.

Ως γενικό συμπέρασμα θα λέγαμε ότι, το Πρόβλημα Δρομολόγησης Οχημάτων είναι ένα πρόβλημα που θα μπορούσε να δεχτεί πολύ έρευνα, είναι ενδιαφέρον για να ασχοληθεί κάποιος και δίνει τη δυνατότητα στον ερευνητή να πειραματιστεί με πολλούς διαφορετικούς τρόπους. Επιπλέον, τα διαγράμματα αποφάσεων μπορούν να αποτελέσουν έναν πολλά υποσχόμενο τρόπο μελέτης και επίλυσης τέτοιου είδους προβλημάτων. Βέβαια, θα χρειαστεί αρκετή δουλειά πάνω σε αυτή τη προσέγγιση προκειμένου να επιτευχθούν τα επιθυμητά αποτελέσματα. Το σίγουρο πάντως είναι ότι, με τη κατάλληλη έρευνα και τη σωστή καθοδήγηση, τα αποτελέσματα της έρευνας να οδηγήσουν σε νέες προσεγγίσεις.

5.2 Μελλοντική Δουλειά

Έπειτα από την ολοκλήρωση αυτής της εργασίας, υπάρχουν οι προϋποθέσεις για την επίτευξη ακόμα καλύτερων αποτελεσμάτων. Φυσικά, μιλάμε για την βελτίωση του δεύτερο μοντέλου που παρουσιάσαμε, του μοντέλου βασισμένο σε διαγράμματα αποφάσεων. Ιδανικά, θα θέλαμε να βελτιώσουμε το μοντέλο σε αριθμό περιορισμών. Ένα υποσχόμενο μοντέλο, με αποτελέσματα παρόμοια ή και καλύτερα με αυτό που έχουμε στα χέρια μας, θα μπορούσε να υπάρξει και με λιγότερους περιορισμούς. Σε μελλοντικές έρευνες, θα ερευνήσουμε πιο διεξοδικά αυτή την πιθανότητα, με σκοπό την εύρεση βέλτιστης λύσης σε ακόμη πιο γρήγορους χρόνους.

Μία εξίσου ενδιαφέρουσα δουλειά, θα ήταν να ασχοληθούμε με την ένταξη των μοντέλων μας στη καθημερινότητά μας. Για παράδειγμα, θα μπορούσαμε να υλοποιήσουμε κώδικα που θα δέχεται ως είσοδο πραγματικές συντεταγμένες από πόλεις/πελάτες και να αποφασίζει ποια θα είναι η βέλτιστη διαδρομή που πρέπει να ακολουθήσουν τα οχήματα. Έτσι, αντί για τυχαία πειράματα που τρέξαμε αυτή τη φορά, θα έχουμε στη διάθεσή μας γνωστές περιοχές πάνω σε ένα χάρτη, θα γνωρίζουμε με ακρίβεια ποια είναι η πραγματική απόσταση μεταξύ δύο πόλεων/πελατών, τις κατευθύνσεις των δρόμων, παρακάμψεις για ταχύτερη μεταφορά καθώς επίσης και το αν ορισμένοι δρόμοι ενδείκνυνται για να περάσουν φορτηγά από αυτούς. Ως αποτέλεσμα, οι δοκιμές μας θα χαρακτηρίζονται από ρεαλισμό πράγμα που σημαίνει ότι θα μπορούμε να μελετήσουμε σε βάθος το Πρόβλημα Δρομολόγησης Οχημάτων.

Μία άλλη ενδιαφέρουσα ιδέα έχει να κάνει με την έκταση που έχει πάρει η τεχνολογία στην εποχή μας. Πηγαίνοντας ένα βήμα παρακάτω την ιδέα μας για πιο ρεαλιστική διερεύνηση του Προβλήματος Δρομολόγησης Οχημάτων, θα μπορούσαμε να βγάλουμε μία πιο εμπορική δουλειά προς το κοινό. Πιο αναλυτικά, θα μπορούσαμε να δημιουργήσουμε ένα πρόγραμμα με γραφικό περιβάλλον που θα δέχεται από τον χρήστη τις πόλεις/πελάτες που θέλει να επισκεφτεί για την μεταφορά προϊόντων και να επεξεργάζεται με κάποιο μοντέλο τα δεδομένα, δίνοντας του τη δυνατότητα να ακολουθήσει το συντομότερο δρομολόγιο, λαμβάνοντας υπόψιν όλους του περιορισμούς που μπορεί να έχουν οι διαδρομές.

Επιπλέον, όπως έχει αναφερθεί, το Πρόβλημα Δρομολόγησης Οχημάτων απευθύνεται και σε πιο ελεύθερης φύσης μεταφορές και είναι επίσης γνωστό ότι, η πρόσβαση κινητού τηλεφώνου είναι πολύ εύκολη και ευρύτατα διαδεδομένη. Σκεπτόμενοι αυτά τα δύο, θα μπορούσαμε να δημιουργήσουμε μία εφαρμογή, συμβατή με όλα τα σύγχρονα κινητά τηλέφωνα που θα βοηθούσε και τους μεταφορείς που τελούν δρομολόγια μικρότερων αποστάσεων. Για παράδειγμα, για την διανομή φαγητού ένας υπάλληλος καλύπτει μικρές αποστάσεις εντός πόλης, αλλά κάθε φορά επισκέπτεται διαφορετικούς πελάτες άρα και διαφορετικούς σταθμούς. Επομένως, μία εφαρμογή θα δέχεται τις οδούς από τους πελάτες, θα υπολογίζει τη βέλτιστη διαδρομή και θα βοηθάει τον χρήστη να καλύψει τις παραγγελίες στον ταχύτερο δυνατό χρόνο.

Όλα αυτά όπως αναφέρθηκε, ανήκουν στις μελλοντικές υποθετικές δουλειές που θα μπορούσαμε να κάνουμε αν επιχειρήσουμε να ασχοληθούμε πιο εκτεταμένα πάνω στο θέμα των μεταφορών. Η μέχρι τώρα δουλειά μας, μας βοήθησε ώστε να δημιουργηθούν αρκετές ιδέες για πιο ουσιαστική και ταυτόχρονα ρεαλιστική προσέγγιση στο θέμα.

Βιβλιογραφία

Bergman, D., Cire, A. A., van Hoeve, W.-J., & Hooker, J. N. (2016). Discrete optimization with decision diagrams. *INFORMS Journal on Computing*, *28*(1), 47–66.

Caprara, A., & Fischetti, M. (1997). Branch-and-cut algorithms. *Annotated Bibliographies in Combinatorial Optimization*, 45–63.

Christofides, N., Mingozzi, A., & Toth, P. (1981). Exact algorithms for the vehicle routing problem, based on spanning tree and shortest path relaxations. *Mathematical Programming*, *20*(1), 255–282.

Clarke, G., & Wright, J. W. (1964). Scheduling of vehicles from a central depot to a number of delivery points. *Operations Research*, *12*(4), 568–581.

Dantzig, G. B., & Ramser, J. H. (1959). The truck dispatching problem. *Management Science*, *6*(1), 80–91.

Desaulniers, G., & Groupe d'études et de recherche en analyse des décisions (Montréal, Q. (2000). *The vrp with pickup and delivery*. Groupe d'études et de recherche en analyse des décisions.

Gendreau, M. (1997). Vehicle routing: modern heuristics. *Local search in combinatorial optimization*.

Golden, B. L., Raghavan, S., & Wasil, E. A. (2008). *The vehicle routing problem: latest advances and new challenges* (Vol. 43). Springer Science & Business Media.

Hooker, J. N. (2013). Decision diagrams and dynamic programming. In *International conference on ai and or techniques in constriant programming for combinatorial optimization problems* (pp. 94–110).

Kallehauge, B., Larsen, J., Madsen, O. B., & Solomon, M. M. (2005). Vehicle routing problem with time windows. In *Column generation* (pp. 67–98). Springer.

Korf, R. E. (2002). A new algorithm for optimal bin packing. In *Aaai/iaai* (pp. 731–736).

Laporte, G., Mercure, H., & Nobert, Y. (1986). An exact algorithm for the asymmetrical capacitated vehicle routing problem. *Networks*, *16*(1), 33–46.

Laporte, G., & Nobert, Y. (1987). Exact algorithms for the vehicle routing problem. In *North-holland mathematics studies* (Vol. 132, pp. 147–184). Elsevier.

Toth, P., & Vigo, D. (2002). *The vehicle routing problem*. SIAM.