ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ

ΤΜΗΜΑ ΜΗΧΑΝΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ

Διπλωματική Εργασία

Αλγόριθμος δρομολόγησης οχήματος με ενιαίο διαμέρισμα για κ=2 προιόντα

υπό

ΓΟΥΡΖΟΥΛΙΔΗ ΣΠΥΡΟΥ

Υπεβλήθη για την εκπλήρωση μέρους των

απαιτήσεων για την απόκτηση του

Διπλώματος Μηχανολόγου Μηχανικού

2014

Εγκρίθηκε από τα Μέλη της Τριμελούς Εξεταστικής Επιτροπής:

Πρώτος Εξεταστής (Επιβλέπων)	Δρ. Δημήτριος Παντελής Καθηγητής, Τμήμα Μηχανολόγων Μηχανικών Πανεπιστήμιο Θεσσαλίας
Δεύτερος Εξεταστής	Δρ. Γεώργιος Λυμπερόπουλος Καθηγητής, Τμήμα Μηχανολόγων Μηχανικών Πανεπιστήμιο Θεσσαλίας
Τρίτος Εξεταστής	Δρ. Γεώργιος Σαχαρίδης Καθηγητής, Τμήμα Μηχανολόγων Μηχανικών Πανεπιστήμιο Θεσσαλίας

Ευχαριστίες

Πρώτα απ' όλα, θέλω να ευχαριστήσω τον επιβλέποντα της διπλωματικής εργασίας μου, κ. Δημήτρη Παντελή, για την πολύτιμη βοήθεια και καθοδήγησή του κατά τη διάρκεια της δουλειάς μου.

Επίσης, είμαι ευγνώμων στα υπόλοιπα μέλη της εξεταστικής επιτροπής, Καθηγητές κ. Γεώργιο Λυμπερόπουλο και κ. Γεώργιο Σαχαρίδη για την ανάγνωση της εργασίας μου και για τις πολύτιμες υποδείξεις τους.

Πάνω απ'όλα, είμαι ευγνώμων στους γονείς μου και στους αδερφούς μου, στους οποίους και αφιερώνω αυτή την εργασία, για την αγάπη και την υποστήριξή τους όλα αυτά τα χρόνια.

Γουρζουλίδης Σπύρος

Αλγόριθμος δρομολόγησης οχήματος με ενιαίο διαμέρισμα

Γουρζουλίδης Σπύρος

Πανεπιστήμιο Θεσσαλίας,

Τμήμα Μηχανολόγων Μηχανικων, 2014

Επιβλέπων Καθηγητής: Δρ. Δημήτριος Παντελής

Περίληψη

Η εργασία αυτή ασχολείται με το πρόβλημα διαχείρισης και δρομολόγησης αποθεμάτων, και εστιάζει στην παρουσίαση και την αξιολόγηση ενός αλγορίθμου δρομολόγησης οχήματος με ενιαίο διαμέρισμα. Παρουσιάζει καταρχήν τα βασικά ζητήματα και έννοιες προβλημάτων βελτιστοποίησης, και αναλύει το πρόβλημα της Δρομολόγησης Διαχείρισης Αποθεμάτων της επιχειρησιακής έρευνας σαν πρόβλημα Δρομολόγησης Οχημάτων. Επιπλέον, παρουσιάζει τα βασικά χαρακτηριστικά του προβλήματος του πλανόδιου πωλητή (TSP) και του προβλήματος δρομολόγησης οχημάτων (VRP). Πέρα όμως από τη θεωρητική αυτή προσέγγιση, στα πλαίσια της εργασίας υλοποιήθηκε και ένας συγκεκριμένος αλγόριθμος δρομολόγησης οχημάτων, για το πρόβλημα που αφορά την περίπτωση του προμηθευτή που πρέπει να προμηθεύει με προϊόντα τους πελάτες του σε διαφορετικές πόλεις. Σε αυτό το πλαίσιο, αυτό που έγινε προσπάθεια να γίνει μέσω αυτού του αλγορίθμου είναι να ελαχιστοποιηθεί το κόστος μεταφοράς για τις N πόλεις που πρέπει να επισκεφθεί το φορτηγό του προμηθευτή, δεδομένου ότι έχουμε δύο μόνο διαφορετικά προϊόντα, σε σχέση και με την χωρητικότητα Q του φορτηγού που κάνει τις μεταφορές.

Σε αυτό το πλαίσιο, έγιναν αρκετές προσομοιώσεις με διαφορετικά χαρακτηριστικά δικτύου, και παρουσιάστηκαν τα αποτελέσματα αναφορικά με τη δρομολόγηση του φορτηγού. Εξετάστηκαν περιπτώσεις με διαφορετικό πλήθος πόλεων αλλά και διαφορετικές τιμές χωρητικότητας του φορτηγού, σε διάφορους συνδυασμούς. Σε κάθε περίπτωση προέκυψε ο κατάλληλος συνδυασμός αποφάσεων από τον οδηγό του φορτηγού, σε σχέση φυσικά και με τη ζήτηση ανά προϊόν σε κάθε πόλη, ώστε να ελαχιστοποιηθεί το κόστος μεταφοράς.

ΠΙΝΑΚΑΣ ΠΕΡΙΕΧΟΜΕΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

1 Εισαγωγή

1.1 Γενικά για ζητήματα βελτιστοποίησης

Ο χώρος της Βελτιστοποίησης (optimization) στα εφαρμοσμένα μαθηματικά αναφέρεται στην αναζήτηση βέλτιστων παραμέτρων ενός (συνήθως πολύπλοκου) συστήματος. Προβλήματα Βελτιστοποίησης απαντώνται σε πολλά επιστημονικά πεδία όπως η Φυσική, η Χημεία, τα Οικονομικά κλπ. Συνήθως ένα πρόβλημα βελτιστοποίησης δίνεται σαν πρόβλημα ελαχιστοποίησης ή μεγιστοποίησης μιας συνάρτησης μιας ή πολλών μεταβλητών. Σε απλά προβλήματα μπορούν να χρησιμοποιηθούν αναλυτικές μέθοδοι επίλυσης. Όμως, όσο αυξάνεται η μη γραμμικότητα των συναρτήσεων, η χρήση προσεγγιστικών αλγορίθμων γίνεται επιτακτική.

1.1.1 Βασικές Έννοιες

Σχεδόν πάντα στο χώρο της βελτιστοποίησης έχουμε κάποια κριτήρια καταλληλότητας μιας λύσης. Με αυτά ξεχωρίζουμε τις επιθυμητές από τις υπόλοιπες υποψήφιες λύσεις του προβλήματος. Τα κριτήρια αυτά εξαρτώνται πάντα από το πρόβλημα και από τον χρήστη. Μερικές βασικές έννοιες στον χώρο της βελτιστοποίησης είναι:

Περιορισμοί (constraints):

Προκύπτουν είτε από το ίδιο το πρόβλημα είτε τίθενται από τον χρήστη, περιορίζοντας το πλήθος των πιθανών λύσεων. Συνήθως είναι ανισοτικοί αλλά συναντάμε και ισοτικούς.

Εφικτή λύση (feasible solution):

Καλείται μια λύση που ικανοποιεί όλους τους περιορισμούς του προβλήματος.

Ολική ή καθολική βελτιστοποίηση (global optimization):

Αναζήτηση της καλύτερης δυνατής λύσης (βέλτιστης) ανάμεσα σε όλες τις εναλλακτικές

Τοπική βελτιστοποίηση (local optimization):

Αναζήτηση της καλύτερης δυνατής λύσης (υπό - βέλτιστης) μέσα σε ένα υποσύνολο όλων των εναλλακτικών

Αντικειμενική συνάρτηση (objective function):

Είναι η συνάρτηση που προκύπτει από την μοντελοποίηση του φυσικού προβλήματος, ώστε να καταστεί δυνατή η επίλυσή του με κάποιο αλγόριθμο. Έτσι, η διαδικασία της βελτιστοποίησης ανάγεται στην εύρεση του ελάχιστου (minimum) ή του μέγιστου (maximum) της αντικειμενικής συνάρτησης, καθώς και του αντίστοιχου ελαχιστοποιητή (minimizer) ή μεγιστοποιητή (maximizer) στον οποίο επιτυγχάνεται αυτή η τιμή

Ελαχιστοποίηση ή μεγιστοποίηση:

Πρόκειται για δύο ισοδύναμα προβλήματα. Η διαφορά μεταξύ τους έγκειται στην αναζήτηση ελάχιστου ή μεγίστου, αντίστοιχα. Επομένως δύναται να χρησιμοποιηθεί οποιαδήποτε από τις δύο περιπτώσεις, ανάλογα με το πρόβλημα.

Αναλυτική επίλυση:

Επιτυγχάνεται μόνο σε περιπτώσεις απλών προβλημάτων όπου οι μαθηματικοί χειρισμοί είναι εφικτοί. Στα υπόλοιπα προβλήματα των οποίων η μαθηματική επίλυση είναι δύσκολη ή ανέφικτη χρησιμοποιούνται προσεγγιστικοί αλγόριθμοι

Αριθμητική – προσεγγιστική επίλυση:

Ενδείκνυται σε περιπτώσεις πολύπλοκων προβλημάτων όπου η αναλυτική επίλυση είναι επίπονη ή αδύνατη. Πραγματοποιείται με χρήση αλγορίθμων που συνήθως ξεκινούν από μια αρχική συνθήκη και παράγουν επαναληπτικά ακολουθίες σημείων που προσεγγίζουν την λύση (δηλαδή τον ελαχιστοποιητή της αντικειμενικής συνάρτησης

1.2 Εισαγωγικά στοιχεία Επιχειρησιακής Έρευνας και Επίλυσης Προβλημάτων

1.2.1 Γενικά

Ο όρος Επιχειρησιακή Έρευνα (Ε.Ε) προήλθε από τον Αγγλικό όρο Operations Research (O.R) και αναφέρεται στα προβλήματα που ασχολούνται με διεξαγωγή ή συντονισμό επιχειρήσεων ή δραστηριοτήτων σε μια επιχείρηση. Χαρακτηριστικά παραδείγματα είναι το εμπόριο, ο δημόσιος τομέας, η υγεία, ο στρατός, η βιομηχανία κτλ. Ο γενικός ορισμός, από την πλευρά της επιστημονικής προσέγγισης έγκειται στη λήψη αποφάσεων και στη συστηματική μελέτη των προβλημάτων αυτών, όπου η επίλυση τους με χρήση μαθηματικών εργαλείων κρίνεται απαραίτητη. Η Ε.Ε προέρχεται από τα τέλη του 19ου αιώνα, συγκεκριμένα από την βιομηχανική επανάσταση. Την περίοδο εκείνη παρατηρήθηκε μια ραγδαία αύξηση των επιχειρήσεων ακολουθούμενη έπειτα από την δημιουργία μεγάλων πολυεθνικών. Επιπλέον, ήταν η πρώτη φορά που έγινε καταμερισμός των διοικητικών δράσεων σε πολλά τμήματα αντίστοιχα.

Από αυτή την διάσπαση και την εξειδίκευση προέκυψαν ποικίλα νέα προβλήματα στην βιομηχανία. Τα τμήματα της εταιρείας μετατράπηκαν σε αυτόνομες οντότητες με διαφορετικούς στόχους όπως:

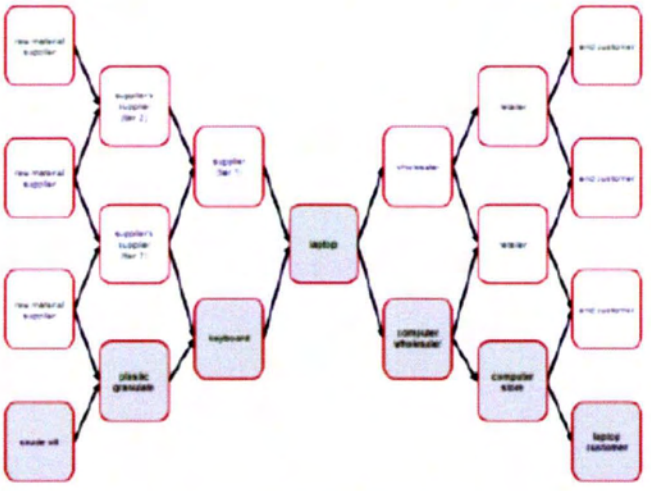
- Τμήμα παραγωγής: Στόχος η μέγιστη αξιοποίηση του ανθρώπινου δυναμικού
- Οικονομικές υπηρεσίες: Στόχος η δημιουργία καθαρού κέρδους
- Τμήμα πωλήσεων: Στόχος η διαθεσιμότητα των προϊόντων
- Τμήμα μάρκετινγκ: Στόχος η δημιουργία αποτελεσματικού δικτύου διανομής και προώθησης.

Μερικά από τα προβλήματα που δημιουργήθηκαν όμως εξαιτίας του κατακερματισμού, ήταν η δύσκολη κατανομή του ανθρώπινου δυναμικού, της διαχείριση των υλικών για το σύνολο της επιχείρησης, καθώς και η ανάγκη επίλυσης προβλημάτων κατανομής πόρων με τον πιο αποδοτικό τρόπο.

Όπως ήταν εύλογο, έτσι δημιουργήθηκαν οι πρώιμες συνθήκες εμφάνισης της Ε.Ε.

1.2.2 Γενική μεθοδολογία επίλυσης προβλημάτων

Σε γενικές γραμμές η μεθοδολογία που ακολουθείται στην Ε.Ε αρχικά είναι η παρατήρηση του συστήματος και η διατύπωση του αντίστοιχου προβλήματος. Ακολουθεί η δημιουργία του μαθηματικού μοντέλου που περιγράφει το πρόβλημα, καθώς επίσης και η ακριβής αναπαράσταση των χαρακτηριστικών αυτού. Στην συνέχεια έρχεται ο έλεγχος της καταλληλότητα του μοντέλου και η επίλυση του με την χρήση κάποιου μαθηματικού μοντέλου. Ακολουθεί η επαλήθευση της καταλληλότητας της λύσης που βρέθηκε.



Εικόνα 1: Παράδειγμα σύνθετου δικτύου εφοδιαστικής αλυσίδας.

1.2.3 Το πρόβλημα της Διαχείρισης Εφοδιαστικής Αλυσίδας (SCM)

Το πρόβλημα της Διαχείρισης Εφοδιαστικής Αλυσίδας (Δ.Ε.Α) αναφέρεται στη διαχείριση ενός δικτύου από διασυνδεδεμένες επιχειρήσεις όπου συμμετέχουν στην παροχή προϊόντων ή πακέτων, τα οποία ζητούνται

από τους τελικούς καταναλωτές στην εφοδιαστική αλυσίδα. Ένα παράδειγμα ενός σύνθετου δικτύου Δ.Ε.Α βλέπουμε στην Εικόνα 1. Ο όρος "Εφοδιαστικής Αλυσίδα ". εισήχθηκε από τον Keith Oliver, σύμβουλο στην Booz Allen Hamilton, ο οποίος τον χρησιμοποίησε σε μια συνέντευξή του το 1982 στο Financial Times. Άρχισε έτσι να γίνεται όλο και ποιο γνωστός. Ουσιαστική άνθιση γνώρισε στα μέσα του 1990, όταν ξεκίνησε να εμφανίζεται όλο και πιο συχνά σε διάφορα άρθρα και βιβλία. Στις αρχές του 1990 κατείχε ήδη εξέχουσα θέση στον τομέα των οικονομικών επιστημών.

Μερικοί επίσημοι ορισμοί που έχουν δοθεί για το πρόβλημα της Δ.Ε.Α είναι οι εξής:

- Η διαχείριση της εξερχόμενης και της εισερχόμενης ροής των προστιθέμενων υλικών, των τελικών προϊόντων και των συσχετιζόμενων πληροφοριών μεταξύ των προμηθευτών, εταιριών, μεταπωλητών και τελικών καταναλωτών αποκαλείται Δ.Ε.Α.
- Η ΔΕΑ είναι ο συστηματικός στρατηγικός συντονισμός των συστημάτων διαχείρισης και των τακτικών, στο πλαίσιο μιας συγκεκριμένης εταιρίας ή επιχείρησης, για τους σκοπούς της βελτίωσης της μακροπρόθεσμης απόδοσης αυτών, καθώς επίσης και της αλυσίδας εφοδιασμού.

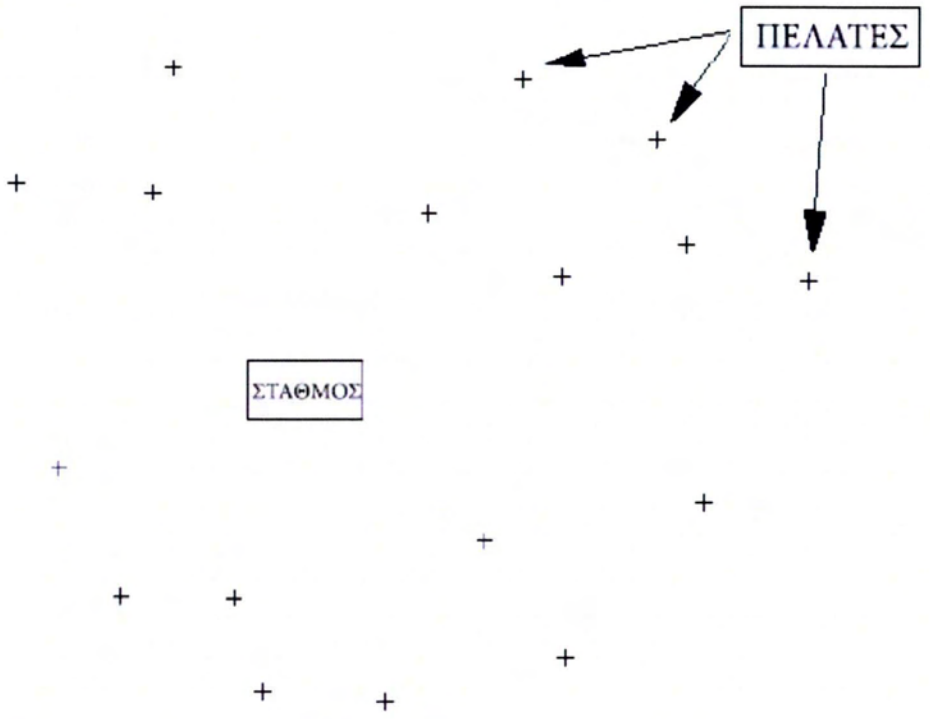
1.2.4 Το πρόβλημα της Διαχείρισης Αποθέματος - από πλευράς προμηθευτή (VMI)

Το πρόβλημα της Διαχείρισης Αποθέματος (Vendor-Manged Inventory) - από πλευράς προμηθευτή (Π.Δ.Α) αποτελεί μια οικογένεια επιχειρησιακών μοντέλων, στα οποία ο αγοραστής και το προϊόν προσφέρουν ορισμένες πληροφορίες στον προμηθευτή του συγκεκριμένου προϊόντος, και ο προμηθευτής είναι αυτός που αναλαμβάνει την διατήρηση της συμφωνημένης απογραφής του υλικού. Επιπλέον, ένας τρίτος πάροχος μπορεί να αναμειχθεί στην διαδικασία με σκοπό να εξασφαλίσει την απαιτούμενη ποσότητα αποθέματος αλλάζοντας την ζήτηση και τα διάφορα κενά του ανεφοδιασμού.

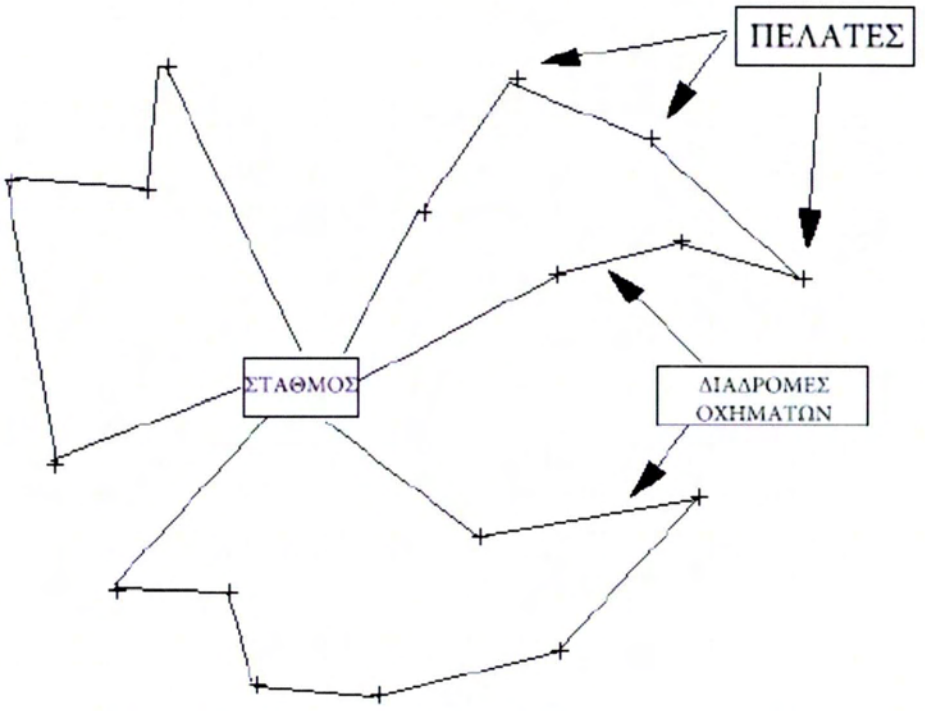
1.2.5 Το πρόβλημα της Δρομολόγησης Οχημάτων (VR)

Το πρόβλημα Δρομολόγησης Οχημάτων (Δ.Ο) αποτελεί ένα συνδυαστικό πρόβλημα βελτιστοποίησης ακέραιου προγραμματισμού που επιδιώκει την εξυπηρέτηση μεγάλου αριθμού πελατών με τη χρήση ενός στόλου οχημάτων. Προτάθηκε από τους Dantzig και Ramser το 1959 και είναι ένα από τα πιο σημαντικά προβλήματα στον τομέα των μεταφορών και της διανομής. Στο πρόβλημα αυτό έχουμε την διανομή διάφορων προϊόντων σε διαφορετικούς πελάτες με τη χρήση ενός στόλου οχημάτων με σκοπό την ικανοποίηση της εκάστοτε ζήτησης από τον κάθε πελάτη σε ένα πεπερασμένο χρονικό ορίζοντα. Στόχος είναι η ελαχιστοποίηση του κόστους διανομής των προϊόντων στους τελικούς πελάτες. Πολλές μέθοδοι έχουν προταθεί κατά καιρούς για την αναζήτηση λύσεων στο πρόβλημα, αλλά συνήθως για σχετικά μικρού μεγέθους προβλήματα, καθώς για την εύρεση μιας λύσης σε ένα μεγάλο και σύνθετο πρόβλημα απαιτείται μεγάλη υπολογιστική πολυπλοκότητα.

Ας υποθέσουμε ότι έχουμε την εξής κατάσταση που απεικονίζεται στην επόμενη εικόνα, όπου έχουμε έναν σταθμό περιτριγυρισμένο από πελάτες οι οποίοι προμηθεύονται από τον σταθμό διάφορα προϊόντα. Ο σταθμός αναλαμβάνει την διεργασία κατασκευής των διαδρομών που πρόκειται να ακολουθηθεί από τα οχήματα για την μεταφορά των προϊόντων στους πελάτες. Το πρόβλημα αυτό είναι γνωστό ως πρόβλημα Δ.Ο. Επιπλέον, στη $2^{η}$ κατά σειρά εικόνα βλέπουμε τις διαδρομές των οχημάτων από τον σταθμό προς τους πελάτες. Έτσι, το πρόβλημα Δ.Ο μπορεί να οριστεί σαν ένα πρόβλημα σχεδιασμού διαδρομών των οχημάτων με γνωστή χωρητικότητα, με σκοπό την δρομολόγηση των προϊόντων στους διάφορους πελάτες σε έναν πεπερασμένο χρονικό ορίζοντα, καλύπτοντας την εκάστοτε ζήτηση του κάθε πελάτη.



Εικόνα 2: Σταθμός και πελάτες



Εικόνα 3: Οι διαδρομές των οχημάτων από τον σταθμό προς τους πελάτες

Επιπλέον, οι διαδρομές των οχημάτων έχουν σχεδιαστεί έτσι ώστε να ελαχιστοποιούν την ολική απόσταση που πρόκειται να διανύσει το κάθε όχημα.

Το πρόβλημα της Δ.Ο ακολουθείται από μια μεγάλη ιστορία συστηματικής μελέτης, πρωτοεμφανίστηκε και θεωρήθηκε για πρώτη φορά σε μια ακαδημαϊκή εργασία από τους Dantzig και Ramser όπως αναφέραμε και δημοσιεύτηκε στις αρχές του 1950. Έκτοτε, έχει τραβήξει την προσοχή πολλών ακαδημαϊκών και μη λόγω της μεγάλης πρακτικής εφαρμογής του καθώς και του θεωρητικού ενδιαφέροντος.

1.3 Το πρόβλημα της Δρομολόγησης Διαχείρισης Αποθεμάτων της επιχειρησιακής έρευνας σαν πρόβλημα Δρομολόγησης Οχημάτων

Ο κύριος στόχος των προβλημάτων Δ.Ε.Α είναι η σωστή διευθέτηση των διαφόρων φάσεων της εφοδιαστικής αλυσίδας. Αναλυτικότερα, ενσωματώνοντας τις αποφάσεις στο σχεδίασμά των διαφορετικών δραστηριοτήτων έχει ως αποτέλεσμα μια ουσιώδες βελτίωση στην συνολική απόδοση. Ένα σχετικό παράδειγμα ενσωματωμένων και συντονισμένων αποφάσεων μπορεί να βρεθεί στα προβλήματα Π.Δ.Α. όπου οι πελάτες κάνουν διαθέσιμες τις πληροφορίες του αποθέματος στους προμηθευτές τους (διανομείς), οι οποίοι έπειτα αναλαμβάνουν την ευθύνη της απόφασης στο πότε θα αποφασίσουν να εξυπηρετήσουν και ποιους πελάτες. Συνεπώς, ο προμηθευτής έχει να επιλέξει πόσο συχνά, πότε, και σε ποίες ποσότητες θα τροφοδοτηθούν οι διαφορετικοί πελάτες του. Αυτή η ολοκληρωμένη διαχείριση των αποθεμάτων και διανομής προσφέρει μεγαλύτερη ευελιξία και ευκολία στον σχεδίασμά αποτελεσματικότερων διαδρομών για τα οχήματα μας, ενώ παράλληλα πραγματοποιείται βελτιστοποίηση στην αποθήκη καθ όλη την διάρκειά της αλυσίδας ανεφοδιασμού.

Το βασικό πρόβλημα βελτιστοποίησης που πρέπει να αντιμετωπιστεί από τον προμηθευτή, είναι η ταυτόχρονη λήψη αποφάσεων σχετικά με τις ποσότητες ανεφοδιασμού, καθώς επίσης και των διαδρομών που πρόκειται να ακολουθήσουν τα οχήματα μας με σκοπό την επίσκεψη όλων των

πελατών. Αυτό είναι γνωστό ως το πρόβλημα της Δρομολόγησης Διαχείρισης Αποθέματος (Δ.Δ.Α). Το Δ.Δ.Α είναι από τα πιο ενδιαφέροντα και τα πιο δύσκολα προβλήματα στο χώρο της βελτιστοποίησης αλυσίδας ανεφοδιασμού και τυπικά θεωρείται ότι λειτουργεί υπό έναν κεντρικό σταθμό οχημάτων, εφοδιάζοντας ένα αριθμό από γεωγραφικά κατανεμημένους διάσπαρτους πελάτες, σε μια συγκεκριμένη χρονική περίοδο (Campbell et al, 1998).

Η βιβλιογραφία των προβλημάτων Δ.Δ.Α είναι εκτενής και περιλαμβάνει αρκετές παραλλαγές του προβλήματος, κυρίως όσον αφορά την φύση της ζήτησης από τους πελάτες με την έννοια της ντετερμινιστικότητας, σχολαστικότητας κτλ, καθώς και το μήκος του χρονικού ορίζοντα, πεπερασμένο η μη.

Ενδεικτικά μπορούμε να αναφέρουμε μερικές παρόμοιες δουλειές στον προαναφερθείσα τομέα όπως οι εξής:

- Μονής περιόδου ΔΔΑ με στοχαστική ή ντετερμινιστική ζήτηση

- Πολλαπλών περιόδων πεπερασμένου ορίζοντα ΔΔΑ με σταθερή ή δυναμική ζήτηση

- Μη – πεπερασμένου ορίζοντα ΔΔΑ με ντετερμινιστική ή στοχαστική ζήτηση

Επιπρόσθετο υλικό στον τομέα των προβλημάτων Δ.Δ.Α υπάρχει διαθέσιμο σε πληθώρα ερευνών (παραδείγματος χάριν Campbell και Savelsbergh, 2004, Chan et al, 1998, Chien et al, 1989, Dorigo και Caro, 1999, Dorigo και Gambardella, 1997, Dorigo και Stutzle, 2004)

2 Προβλήματα Δρομολόγησης

Στο Κεφάλαιο αυτό παραθέτουμε μερικές σημαντικές περιπτώσεις ζητημάτων δρομολόγησης. Τα προβλήματα για τα οποία θα μιλήσουμε, μελετώνται διεξοδικά από τους σύγχρονους ερευνητές και παίζουν καθοριστικό ρόλο σε διάφορες περιοχές της επιστήμης (διαδίκτυο, κατανεμημένα συστήματα, traffic management, bandwidth management κ.α). Σκοπός του κεφαλαίου αυτού είναι μια σύντομη θεωρητική επισκόπηση και μια πρώτη εισαγωγή στις αλγοριθμικές έννοιες που διέπουν προβλήματα όπως το Πρόβλημα του Πλανόδιου Πωλητή (TSP), το Πρόβλημα Δρομολόγησης Οχημάτων (VRP) και το Πρόβλημα του Συντομότερου Μονοπατιού (SPP). Στο κεφάλαιο που ακολουθεί διεισδύουμε σε πιο συγκεκριμένο ζήτημα δρομολόγησης, το μελετούμε και εξετάζουμε τους αλγορίθμους που οδηγούν στη σχεδίαση της βέλτιστης διαδρομής, που είναι και ο αντικειμενικός σκοπός του Προβλήματος της Δρομολόγησης.

2.1 Είδη Προβλημάτων Δρομολόγησης

Το πρόβλημα της σχεδίασης της βέλτιστης διαδρομής μπορεί να τεθεί υπό τη μορφή μιας πληθώρας περιπτώσεων, που σχετίζονται με το αν για παράδειγμα η ζητούμενη διαδρομή είναι κλειστή ή ανοιχτή (μονοπάτι ή κύκλος), αν τα κόστη μεταβάλλονται ή είναι σταθερά, αν η τοπολογία του δικτύου αλλάζει με το χρόνο κ.α. Έτσι λοιπόν, προκύπτει η ανάγκη κατηγοριοποίησης των διαφόρων προβλημάτων δρομολόγησης, οπότε και μελετούμε το κάθε πρόβλημα ξεχωριστά. Παρακάτω δίνουμε τις συνοπτικές περιγραφές ορισμένων από τα ποιο γνωστά προβλήματα που συναντούμε στην περιοχή αυτή.

2.2 Το Πρόβλημα του Πλανόδιου Πωλητή (Travelling Salesman Problem - TSP)

Το πλέον διαδεδομένο αυτό πρόβλημα δρομολόγησης, απαντάται συχνά στην περιοχή της Θεωρητικής Πληροφορικής και της Μαθηματικής Βελτιστοποίησης και μας λέει το εξής: Δοθείσας μιας λίστας πόλεων και των

μεταξύ τους αποστάσεων, να βρεθεί η συντομότερη διαδρομή, που επισκέπτεται κάθε πόλη ακριβώς μία φορά, και καταλήγει στην πόλη από την οποία ξεκινά (αφετηρία).

Το πρόβλημα διατυπώθηκε για πρώτη φορά το 1930, και αποτελεί βασικό αντικείμενο μελέτης και ανάλυσης για πολλά ακόμη προβλήματα βελτιστοποίησης. Η απλότητα της διατύπωσης του προβλήματος είναι παραπλανητική. Και αυτό γιατί το TSP είναι ένα από τα πιο εντόνως μελετημένα προβλήματα των Υπολογιστικών Μαθηματικών, αφού καμία γενική μέθοδος επίλυσης του δεν έχει βρεθεί μέχρι σήμερα. Παρότι όμως το TSP παρουσιάζει μεγάλη δυσκολία ως προς την υπολογισιμότητα και πολυπλοκότητά του, ένας μεγάλος αριθμός ευρετικών αλγορίθμων έχουν προταθεί, παρουσιάζοντας λύσεις και για την περίπτωση που υπάρχουν αρκετές πόλεις σαν δεδομένο του προβλήματος.

Το TSP έχει διάφορες εφαρμογές, ακόμα και στην πιο πρωταρχική μορφή του, όπως είναι η σχεδίαση γεωγραφικών διαδρομών στα οδικά δίκτυα, τα οικονομικά μαθηματικά και η κατασκευή μικροκυκλωμάτων. Η ελαφριά παραλλαγή του, προσδίδει ακόμα περισσότερες εφαρμογές στο πρόβλημα, όπως για παράδειγμα, η αλληλουχία των βάσεων στο DNA. Στις παραπάνω εφαρμογές, η έννοια πόλη αντιστοιχεί σε πελάτες, σημεία συναλλαγής ή τμήματα DNA, και η απόσταση μεταφράζεται σε χρόνους μετάβασης, κόστη, ή μέτρα ομοιότητας μεταξύ τμημάτων DNA. Σε επιπρόσθετες εφαρμογές, μπορεί να έχουμε περιορισμούς, όπως η ύπαρξη χρονοπαραθύρων (time windows), ή περιορισμένους πόρους (limited resources), οπότε το πρόβλημα γίνεται σαφώς δυσκολότερο.

Στη θεωρία της Υπολογιστικής Πολυπλοκότητας, το TSP κατατάσσεται στην NP κλάση προβλημάτων. Αυτό σημαίνει ότι δεν υπάρχει αλγόριθμος που να επιλύει το πρόβλημα σε πολυωνυμικό χρόνο, δηλαδή σε χρόνο που εξαρτάται πολυωνυμικά από το πλήθος των πόλεων που περιέχει σαν δεδομένο το πρόβλημα. Επομένως όταν αυξάνονται οι πόλεις, ο απαιτούμενος χρόνος μεγαλώνει εκθετικά, δηλαδή ένας υπολογιστής θα χρειαζόταν μερικά χρόνια για να λύσει ακριβώς, ένα πρόβλημα μερικών εκατοντάδων πόλεων!

2.2.1 Περιγραφή του Προβλήματος TSP

Το TSP πρόβλημα περιγράφεται από ένα γράφο, του οποίου οι κορυφές αναπαριστούν τις πόλεις, οι ακμές τις δυνατές μεταβάσεις και μια λίστα από κόστη επί των ακμών, που σχετίζονται άμεσα με τις αποστάσεις μεταξύ των πόλεων. Μια διαδρομή σε ένα πρόβλημα TSP συνιστά γραφοθεωρητικά ένα κύκλο Hamilton και η βέλτιστη διαδρομή που αναζητούμε, είναι ο κύκλος Hamilton ελαχίστου μήκους. Συχνά το μοντέλο εφαρμόζεται σε ένα πλήρη γράφο, δηλαδή σε ένα γράφο που κάθε κορυφή του συνδέεται μέσω ακμών με όλες τις άλλες κορυφές.

Ανάλογα με τις απαιτήσεις του προβλήματος, συναντούμε το TSP σε δύο εκδοχές, αυτές του συμμετρικού και ασύμμετρου προβλήματος. Στην πρώτη, το κόστος μετάβασης ανάμεσα σε δύο πόλεις είναι ανεξάρτητο της κατεύθυνσης της κίνησης, δηλαδή δεν αλλάζει είτε μεταβαίνουμε από την κορυφή A προς την B, είτε αντίστροφα. Έχουμε επομένως να μελετήσουμε έναν μη κατευθυνόμενο γράφο. Στην ασύμμετρη περίπτωση, τα μονοπάτια μπορεί να μην είναι τα ίδια αν αλλάζουμε την κατεύθυνση της κίνησης, ή ακόμα τα κόστη μπορεί να είναι διαφορετικά, οπότε ο γράφος είναι κατευθυνόμενος. Εδώ το πρόβλημα είναι συνθετότερο. Η ασυμμετρία, μπορεί να περιγράφει καταστάσεις κυκλοφοριακής σύγκρουσης (traffic collisions), μονόδρομους καθώς και διαδρομές ανάμεσα σε πόλεις με διαφορετικές τιμές κόστους εισιτηρίων κ.α.

Το Πρόβλημα του Πλανόδιου Πωλητή, μπορεί να είναι εφοδιασμένο με μια μετρική επί του συνόλου των κορυφών. Ένα απλό ενδεχόμενο μιας τέτοιας περίπτωσης, είναι να έχουμε σε ισχύ την τριγωνική ανισότητα για τις αποστάσεις μεταξύ τριών διαφορετικών κορυφών i,j και k. Πρακτικά αυτό μπορεί να σημαίνει ότι, στο δίκτυο δεν υπάρχουν συντομεύσεις, δηλαδή ότι η απευθείας μετάβαση από την A στη B δεν έχει μεγαλύτερη απόσταση, από ότι να γίνει η μετάβαση διαμέσου της κορυφής C. Ισχύει λοιπόν εδώ ότι: $C_ij<C_{ik}+C_kj$. Προφανώς όμως, σε πολλά προβλήματα δρομολόγησης, οι αποστάσεις δεν ικανοποιούν την τριγωνική ανισότητα. Για παράδειγμα, διαδρομές που καλύπτονται αεροπορικά μπορεί να είναι ταχύτερες, παρότι οι αποστάσεις είναι μεγαλύτερες.

Εξ' ορισμού του, το TSP δεν επιτρέπει ο πωλητής (salesman) να

επισκεφθεί πόλεις, περισσότερο από μία φορά, αλλά πολλές εφαρμογές δεν χρειάζονται αυτόν τον περιορισμό. Σε αυτές τις περιπτώσεις, ένα συμμετρικό ενδεχόμενο του προβλήματος, το οποίο δεν φέρει μετρική, μπορεί να αναχθεί σε ένα άλλο, που φέρει μια μετρική σχέση, ώστε να βοηθηθεί ο σχεδιαστής κατά τη διαδικασία εύρεσης της βέλτιστης διαδρομής. Έτσι λοιπόν, μπορούμε να αντικαταστήσουμε τον αρχικό γράφο με έναν πλήρη γράφο, στον οποίο η απόσταση Cij αντικαθίσταται από το κόστος του συντομότερου μονοπατιού (Shortest Path), ανάμεσα στης κορυφές i και j, στον αρχικό γράφο.

2.2.2 Υπολογιστική Πολυπλοκότητα και Αλγόριθμοι Επίλυσης στο TSP

Το TSP, όπως ήδη αναφέρθηκε, κατατάσσεται στην οικογένεια των προβλημάτων που παρουσιάζουν πολυπλοκότητα NP τύπου. Για το λόγο αυτό οι βασικές αρχές που ακολουθούμε για την επίλυση του είναι οι ακόλουθες:

Περαιτέρω, έχει αποδειχθεί πως το TSP είναι πλήρες ως προς την κλάση πολυπλοκότητας FP^{np}. Στη θεωρία της Υπολογιστικής Πολυπλοκότητας, ένα function problem είναι ένα πρόβλημα, για το οποίο μία και μόνο έξοδος αναμένεται για κάθε είσοδο. Αλλά αυτή η έξοδος είναι συνθετότερη από εκείνη ενός προβλήματος απόφασης (decision problem), δηλαδή δεν περιορίζεται σε δύο καταστάσεις τύπου NAI ή OXI. Έτσι λοιπόν, με τον όρο FP^{np}, υπονοούμε την κλάση όλων των function problems που επιλύονται από μια μηχανή Turing σε πολυωνυμικό χρόνο, αλλά στην περίπτωση που η πολυπλοκότητα είναι τύπου NP, υπάρχει μια μηχανή χρησμού (oracle machine) που οδηγεί το πρόβλημα στην επίλυσή του, μέσω λήψης αποφάσεων.

Η πιο άμεση διαδικασία για την ακριβή λύση ενός προβλήματος TSP είναι να δοκιμάσουμε όλες τις δυνατές διαδρομές (permutations), και να αποφανθούμε για το ποια είναι η οικονομικότερη. Μπορούμε να χρησιμοποιήσουμε την τεχνική του brute- force search ή αλλιώς γνωστή και σαν exhaustive search. Η μέθοδος αυτή που είναι γενικά αδύναμη (trivial), περιλαμβάνει τη συστηματική απαρίθμηση όλων των δυνατών υποψηφίων λύσεων και έλεγχο αυτών, ως προς το αν ικανοποιούν τις αρχές-προυποθέσεις του προβλήματος. Τα δύο βασικά πλεονεκτήματα του brute-search αλγορίθμου είναι πως αφενός είναι αρκετά απλός στην υλοποίησή

του, αφετέρου δε βρίσκει πάντα τη λύση, εφόσον υπάρχει. Ωστόσο η πολυπλοκότητά του είναι ανάλογη του αριθμού των πιθανών λύσεων, γεγονός που καθιστά την εφαρμογή του αλγορίθμου απαγορευτική για προβλήματα μεγάλου μεγέθους. Πράγματι, ο απαιτούμενος χρόνος, χρησιμοποιώντας τη μέθοδο της απαρίθμησης όλων των δυνατών διαδρομών, παρουσιάζει πολυπλοκότητα της τάξης του O(^), όπου n είναι ο αριθμός των πόλεων που πρέπει να επισκεφτεί ο πωλητής. Ο Δυναμικός Προγραμματισμός έδωσε έναν άλλο τρόπο εύρεσης της ακριβούς λύσης, που παρουσιάζει πολυπλοκότητα της τάξης του $O(n^2 2^n)$ (Bellman).

Γίνεται λοιπόν απόλυτα αντιληπτό, πως για τις διάφορες εφαρμογές, καθίσταται αναγκαίος ο σχεδιασμός αλγορίθμων που προσεγγίζουν τη βέλτιστη διαδρομή, και μάλιστα το πετυχαίνουν αυτό σε σχετικά ικανοποιητικό χρόνο. Έτσι είναι χρήσιμη μια άλλη προσέγγιση του προβλήματος. Αν δοθεί ένα στιγμιότυπο ενός TSP, με n κόμβους, τότε κάθε διαδρομή που περνά από κάθε κόμβο ακριβώς μία φορά είναι μια υποψήφια λύση, και το κόστος της αποτελεί ένα άνω φράγμα (upper bound) για το ελάχιστο (βέλτιστο) κόστος. Οι αλγόριθμοι που δουλεύουν σε πολυωνυμικό χρόνο, δίνοντας n στο σύνολο εφικτές λύσεις, (δηλαδή δίνουν λύσεις που αποτελούν άνω φράγματα για τη βέλτιστη λύση) λέγονται προσεγγιστικοί αλγόριθμοι. Στη γενική περίπτωση, οι αλγόριθμοι αυτοί παράγουν λύσεις χωρίς όμως καμία εγγύηση για το πόσο απέχουν οι λύσεις αυτές από τη βέλτιστη λύση του προβλήματος.

2.2.3 Το Δυναμικό TSP (Dynamic TSP)

Η δυναμική εκδοχή του TSP αποτελεί σενάριο δρομολόγησης, το οποίο είναι πιο κοντά στις πρακτικές εφαρμογές απ' ότι το κλασσικό (στατικό) TSP. Το δυναμικό TSP συνίσταται στην εύρεση του κύκλου Hamilton ελαχίστου κόστους σε έναν γράφο G, του οποίου τα χαρακτηριστικά είναι δυναμικά στο χρόνο. Αυτό μπορεί να σημαίνει ότι τα κόστη των ακμών αλλάζουν κατά τη διάρκεια της δρομολόγησης. Ακόμα, είναι δυνατόν να έχουμε την προσθήκη νέων κόμβων στο δίκτυο, γεγονός που συνεπάγεται επανασχεδίαση της βέλτιστης διαδρομής, ώστε να συμπεριληφθούν στον κύκλο Hamilton οι κόμβοι που εισήλθαν δυναμικά στο δίκτυο. Το δυναμικό TSP βρίσκει εξαιρετική απήχηση σε διάφορους τομείς της έρευνας, όπως για παράδειγμα οι τηλεπικοινωνίες, ο έλεγχος ρομπότ, η βελτιστοποίηση δικτύων

υπολογιστών κ.α.

Δεδομένου ότι το δυναμικό TSP σχετίζεται άμεσα με το κλασσικό TSP, πολλά χαρακτηριστικά του είναι παρόμοια με αυτά του συμβατικού προβλήματος. Ωστόσο, η δυναμική θεώρηση που έχουμε εδώ, προσδίδει στο DTSP (Dynamic TSP) αυτοτελείς ιδιότητες, που αναφέρουμε ευθύς αμέσως.

Πρώτα από όλα, θα μπορούσαμε να πούμε ότι το DTSP περιγράφει ένα κινούμενο προφίλ στατικών προβλημάτων. Η παρατήρηση αυτή είναι καθαρά εποπτική. Πράγματι για δύο διαφορετικές χρονικές στιγμές, τα στιγμιότυπα του DTSP αποτελούν δύο ξεχωριστά προβλήματα δρομολόγησης, τα οποία όμως δεν είναι αποκομμένα μεταξύ τους. Γίνεται λοιπόν αντιληπτό, πως το DTSP παρουσιάζει έναν επίκαιρο χαρακτήρα (actuality) ως προς τη χρονική εξέλιξη της διαδικασίας της δρομολόγησης (routing process).

Μια δεύτερη παρατήρηση που κάνουμε, είναι πως το DTSP παρουσιάζει μια συνέχεια (continuity). Και πάλι εδώ βασιζόμαστε στην εποπτεία του προβλήματος. Το DTSP αλλάζει μερικώς (partially) και ποσοτικώς (quantitatively) μέσα στο χρόνο, και η εξέλιξη αυτή συμβαίνει κατά τρόπο συνεχή και αβίαστο.

Το DTSP χαρακτηρίζεται από ευρωστία (robustness). Αυτό σημαίνει ότι καταστάσεις που δεν είναι αναμενόμενες για το σύστημα, όπως για παράδειγμα διαγραφή ή εισαγωγή κόμβου στο γράφο, αντιμετωπίζονται με γρήγορη ανταπόκριση (δηλαδή άμεσα).

Τέλος, είναι κατανοητό πως στο DTSP αναζητούμε τη βέλτιστη λύση μέσα σε λογικά πλαίσια χρόνου. Το πρόβλημα πρέπει δηλαδή να χαρακτηρίζεται και από αποτελεσματικότητα (effectiveness).

2.2.4 Το πιθανοτικό TSP (Probabilistic TSP)

Πολλές φορές, τα δεδομένα σε ένα πρόβλημα δρομολόγησης αποτελούν στατιστικά μεγέθη. Αυτό σημαίνει ότι ο σχεδιαστής οφείλει να καταστρώσει τη βέλτιστη διαδρομή στα πλαίσια των στατιστικών διακυμάνσεων των χαρακτηριστικών που περιγράφουν το πρόβλημα. Έτσι λοιπόν, θα μιλήσουμε εδώ για μια πιθανοτική (στοχαστική) εκδοχή του TSP, που αποτελεί προέκταση του κλασσικού TSP, όπως ακριβώς και το DTSP που αναφέραμε προηγουμένως.

Το PTSP (Probabilistic TSP) ορίζεται σε ένα γράφο G=(N,A), όπου N είναι το σύνολο των κόμβων και A το σύνολο των ακμών του γράφου. Κάθε κόμβος είναι παρών με μια πιθανότητα p_i. Κόμβοι που εμφανίζονται με πιθανότητα 1, αναφέρονται ως «μαύροι κόμβοι», ενώ όλοι οι άλλοι ως «άσπροι». Το κόστος c_{ij}, ορίζεται για κάθε πλευρά e που ανήκει στο A. Σκοπός μας είναι η εύρεση της μέσης (expected) οικονομικότερης διαδρομής.

Όταν λοιπόν αποφασιστεί ποια διαδρομή θα ακολουθηθεί, κάθε «άσπρος» κόμβος είναι παρών ή όχι. Οι κόμβοι που δεν είναι παρόντες στο γράφο, μπορούν να αμεληθούν και ο πωλητής θα μεταβεί στον αμέσως επόμενο κόμβο που αιτείται εξυπηρέτησης.

Οι εφαρμογές του PTSP απαντώνται σε διάφορα πεδία της έρευνας. Ένα γνωστό παράδειγμα εφαρμογής αφορά ένα σύνολο πελατών που πρέπει να εξυπηρετηθεί σε καθημερινή βάση, χωρίς όμως να απαιτούν όλοι οι πελάτες εξυπηρέτηση μέσα στην ίδια μέρα. Έχει μάλιστα αναπτυχθεί ένα πρόγραμμα δρομολόγησης οχήματος (vehicle routing program) προκειμένου να προσφέρονται υπηρεσίες σε ανθρώπους τρίτης ηλικίας, το οποίο βασίζεται στη θεώρηση ενός PTSP. Η λίστα των προς εξυπηρέτηση ηλικιωμένων άλλαζε με ένα μέσο ρυθμό 14% ανά μήνα. Το “meals on wheels” πρόγραμμα περιλάμβανε έναν πλήρους απασχόλησης εργαζόμενο, ο οποίος ήταν υπεύθυνος για τη διαχείριση των πόρων του συστήματος, τη σχεδίαση των menu διατροφής, την επίβλεψη της ποιότητας των υπηρεσιών και φυσικά την παράδοση των φορτίων προς τους ηλικιωμένους πελάτες του προγράμματος. Με άλλα λόγια, η διαχείριση του συστήματος ήταν αρκετά πολύπλοκη, γεγονός που επέβαλλε την αναγκαιότητα για απλοποίηση και βέλτιστο σχεδιασμό του προγράμματος παράδοσης των υπηρεσιών που αναφέραμε. Έτσι αναπτύχθηκε ένα σύστημα δρομολόγησης βασιζόμενο σε έναν αρκετά απλό ευριστικό αλγόριθμο, ο οποίος όμως αποδίδει ικανοποιητικές λύσεις κατά μέσο όρο. Η κύρια ιδέα επικεντρώνεται στη χρήση της “space filling curve”, μιας καμπύλης δηλαδή που επισκέπτεται όλα τα σημεία που βρίσκονται εντός μιας δομικής περιοχής (unit square). Σε πρώτη φάση υπολογίζεται η σχετική θέση Θ καθενός από τους πελάτες, ως προς την καμπύλη πλήρωσης χώρου. Η διαδρομή τότε βρίσκεται ταξινομώντας τις τιμές του Θ (sorting). Η τοποθέτηση του “the meals on wheels” προβλήματος

οδηγεί σε ένα PTSP, αφού όλοι οι κόμβοι κατά μήκος της καμπύλης πλήρωσης χώρου μπορεί να είναι πιθανές θέσεις εξυπηρέτησης ζήτησης.

Το παραπάνω σχήμα δρομολόγησης βελτίωσε τους χρόνους διαδρομής κατά 13% και ο διανομέας κατάφερε να μειώσει τις πέντε διαδρομές σε τέσσερις. Άλλο παράδειγμα του PTSP είναι η δρομολόγηση ενός ταχυδρόμου. Το σύνολο των πελατών ορίζεται στην αρχή του προβλήματος, όμως δεν λαμβάνουν μήνυμα όλοι οι πελάτες κάθε μέρα, οπότε κάποιοι μπορούν να παραλειφθούν από τον ταχυδρόμο.

2.3 Το Πρόβλημα Δρομολόγησης Οχημάτων (Vehicle Routing Problem-VRP)

Το Πρόβλημα Δρομολόγησης Οχημάτων (Vehicle Routing Problem-VRP) αποτελεί ένα από τα πιο σημαντικά προβλήματα που συναντούμε στην περιοχή της Συνδυαστικής Βελτιστοποίησης (Combinatorial Optimization) και εντάσσεται στην κατηγορία των προβλημάτων Ακέραιου Προγραμματισμού. Τα προβλήματα αυτά αποτελούν συνήθως ειδικές περιπτώσεις προβλημάτων Γραμμικού Προγραμματισμού (Linear Programming), στα οποία όμως οι μεταβλητές λαμβάνουν επιτρεπτές τιμές από ένα σύνολο ακεραίων αριθμών. Η βασική παρατήρηση για τον Ακέραιο Προγραμματισμό έγκειται στο ότι, διάφορα προβλήματά του παρουσιάζουν πολυπλοκότητα NP-τύπου, δηλαδή δεν έχει βρεθεί για αυτά αλγόριθμος που να τα επιλύει σε πολυωνυμικό χρόνο.

Η διατύπωση του προβλήματος είναι αρκετά απλή. Ζητούμε τον καθορισμό του συνόλου βέλτιστων διαδρομών για έναν στόλο οχημάτων, προκειμένου να εξυπηρετηθούν κάποιοι πελάτες που εντοπίζονται σε καθορισμένους κόμβους ενός δικτύου. Τα οχήματα αναχωρούν από μια ή περισσότερες αποθήκες ανεφοδιασμού (depots), στις οποίες και επιστρέφουν με το πέρας της αποστολής τους.

Η θεωρητική έρευνα και οι πρακτικές εφαρμογές στο πεδίο δρομολόγησης οχημάτων ξεκίνησαν το 1959 με το «πρόβλημα αποστολής φορτηγών» (truck dispatching problem), το οποίο παρουσιάστηκε από τους Dantzig και Ramser. Ο ορισμός του προβλήματος αυτού έλεγε ότι: «Να βρεθεί

η βέλτιστη δρομολόγηση (σχεδιασμός διαδρομής (tour), δηλαδή) ενός στόλου φορτηγών διανομής καυσίμου, μεταξύ ενός τερματικού σταθμού ανεφοδιασμού και ενός μεγάλου αριθμού σταθμών εξυπηρέτησης, οι οποίοι τροφοδοτούνται με καύσιμο από τον σταθμό ανεφοδιασμού». Χρησιμοποιώντας μια μέθοδο βασισμένη στις αρχές του Γραμμικού Προγραμματισμού, οι Dantzig και Ramser κατέληξαν σε μια προσεγγιστικά βέλτιστη λύση που περιλάμβανε τέσσερις διαδρομές για ένα πρόβλημα δώδεκα σταθμών εξυπηρέτησης. Μερικά χρόνια αργότερα και συγκεκριμένα το 1964, οι Clarke και Wright πρότειναν έναν ευριστικό αλγόριθμο βασισμένο στην άπληστη μέθοδο (greedy method), ο οποίος πέτυχε καλύτερες προσεγγίσεις από αυτές των Dantzig και Ramser. Ακολουθώντας τις παραπάνω μελέτες, εκατοντάδες αλγορίθμων και μοντέλων προτάθηκαν για την ακριβή και προσεγγιστική επίλυση διαφόρων περιπτώσεων VRP. Σήμερα, υπάρχουν διαθέσιμα στην αγορά αρκετά πακέτα λογισμικού που χρησιμοποιούνται για την επίλυση πραγματικών προβλημάτων διαχείρισης στόλων οχημάτων.

Η επιστημονική κοινότητα έστρεψε έντονα την προσοχή της στη μελέτη του VRP, ιδιαίτερα τις τελευταίες τρεις με τέσσερις δεκαετίες. Και αυτό γιατί στα χρόνια αυτά, σημειώθηκε αρκετά έντονη αύξηση της πολυπλοκότητας των Κατανεμημένων Συστημάτων (Distributed Systems). Ένα κατανεμημένο σύστημα αποτελείται από γεωγραφικά ανεξάρτητες, αυτόνομες υπολογιστικές συσκευές, που έχουν τη δυνατότητα να επικοινωνούν μεταξύ τους και να λειτουργούν συντονισμένα για την επίτευξη ενός κοινού στόχου. Σε αντίθεση με τα κεντρικοποιημένα συστήματα, ο σχεδιασμός των κατανεμημένων συστημάτων, η κατανόηση της λειτουργίας τους υπό ιδιαίτερες συνθήκες καθώς και η ανάλυση της συμπεριφοράς τους απαιτεί ειδικές γνώσεις και ικανότητες.

Το VRP που θα συζητήσουμε εδώ, φέρει ως επίκεντρο την ιδέα της διανομής αγαθών ανάμεσα σε ένα σύνολο αποθηκών (depots) και ένα σύνολο γεωγραφικά διασκορπισμένων πελατών (customers). Μπορούμε να θεωρήσουμε τις θέσεις των πελατών σαν κόμβους εξυπηρέτησης πάνω σε ένα (οδικό) δίκτυο, στο οποίο σημειώνουμε προφανώς και όλες τις δυνατές μεταβάσεις (ακμές) που ενώνουν τους κόμβους εξυπηρέτησης μεταξύ τους,

αλλά και με τους σταθμούς ανεφοδιασμού επίσης.

Τα μοντέλα και οι αλγόριθμοι που προτείνονται για τη λύση του VRP μπορούν να χρησιμοποιηθούν επαρκώς και στην περίπτωση γενικότερων ζητημάτων πάνω στα συστήματα μεταφοράς. Η συλλογή στερεών αποβλήτων (solid waste collection), η δρομολόγηση σχολικών λεωφορείων (school bus routing), η μεταφορά ατόμων με αναπηρία (transportation of handicapped persons) και η διαχείριση μονάδων

συντήρησης (management of maintenance units) αποτελούν μερικά παραδείγματα πραγματικών προβλημάτων στα οποία εφαρμόζονται παρεμφερείς τεχνικές επίλυσης με αυτές που χρησιμοποιούμε στο VRP.

2.3.1 Περιγραφή του Προβλήματος VRP

Όπως είδαμε παραπάνω, ένα θεμελιώδες πρόβλημα δρομολόγησης είναι αυτό του Πλανόδιου Πωλητή (TSP). Υπενθυμίζουμε ότι, στο πρόβλημα αυτό ένας πλανόδιος πωλητής πρέπει να επισκεφτεί ένα σύνολο πόλεων και να επιστρέψει στην πόλη από την οποία ξεκίνησε, ελαχιστοποιώντας την απόσταση που συνολικά διανύει. Προφανώς, ο πωλητής πρέπει να επισκεφτεί κάθε πόλη μία και μόνο φορά, διαφορετικά η λύση δεν είναι βέλτιστη. Το VRP αποτελεί μια επέκταση του TSP, με την έννοια ότι στο VRP πρέπει να βρεθούν m το πλήθος βέλτιστες διαδρομές οχημάτων, στις οποίες κάθε όχημα ξεκινάει από έναν κόμβο-αποθήκη (depot), διατρέχει ένα υποσύνολο κόμβων πελατών (customers) σε καθορισμένη σειρά και επιστρέφει στον αρχικό κόμβο.

Στους κόμβους πελατών μπορεί να έχουμε ποικίλες μορφές εξυπηρέτησης, ενώ και οι χωρητικότητες των οχημάτων μπορεί να διαφέρουν. Τα οχήματα πρέπει να επισκεφθούν τα σημεία εξυπηρέτησης ακριβώς μία φορά και η συνολική ζήτηση φορτίου (total demand) πάνω σε μία διαδρομή δεν πρέπει να ξεπερνά τη χωρητικότητα (capacity) του οχήματος. Ο αντικειμενικός σκοπός του VRP είναι η ελαχιστοποίηση του κόστους για κάθε μία από τις m διαδρομές (overall distribution costs). Στις πραγματικές εφαρμογές δρομολόγησης, όμως, προκύπτουν περιορισμοί που καθιστούν το πρόβλημα πολυπλοκότερο. Αυτοί οι παράπλευροι περιορισμοί μπορεί για παράδειγμα να δεσμεύουν το συνολικό χρόνο που

απαιτείται για τη δρομολόγηση, έτσι ώστε αυτός να κινείται εντός ενός προκαθορισμένου διαστήματος. Ακόμα, μπορεί να σχετίζονται με χρονοπαράθυρα (time windows) μέσα στα οποία πρέπει να ξεκινήσει η εξυπηρέτηση (service). Το τελευταίο αυτό είδος προβλήματος δρομολόγησης αναφέρεται ως VRPTW (Vehicle Routing Problem with Time Windows). Περιπτώσεις τέτοιου είδους, συναντά κανείς στα πλαίσια των προβλημάτων χρονοπρογραμματισμού οχημάτων (Vehicle Scheduling). Επιπλέον, όταν έχουμε να μελετήσουμε ζητήματα στα οποία υπάρχουν πολλαπλές αποθήκες (depots) και εμπορεύματα (commodities), η πολυπλοκότητα της επίλυσης αυξάνει υπερβολικά.

Όπως το TSP, έτσι και το VRP μπορεί να παρασταθεί με τη βοήθεια ενός γράφου. Στα σχήματα που ακολουθούν, δείχνουμε τους κόμβους εισόδου ενός παραδείγματος δρομολόγησης οχημάτων, καθώς και την αντίστοιχη έξοδο, δηλαδή τις οικονομικότερες διαδρομές που ακολουθούν τρία στον αριθμό, οχήματα εξυπηρέτησης ζήτησης.

Μιλώντας πιο αυστηρά, το VRP είναι ένα συνδυαστικό πρόβλημα το πεδίο ορισμού του οποίου, είναι το σύνολο των ακμών ενός γράφου G(V,E). Για το πρόβλημα αυτό ορίζουμε:

> $V=\{vo,v1,...,v_n\}$ είναι ένα σύνολο κορυφών, όπου θεωρούμε την αποθήκη ανεφοδιασμού τοποθετημένη στον κόμβο v_0 και έστω $V' = V\backslash\{v_0\}$ το σύνολο n πόλεων.

> $A=\{(v_i,vj)/\ v_i,vj\ e\ V;\ i<>j)$, δηλαδή A υποσύνολο V X V είναι ένα σύνολο ακμών

> C είναι ένας πίνακας από μη αρνητικά κόστη ή αποστάσεις c_ij μεταξύ των πελατών v_i και vj.

> d είναι ένα διάνυσμα που περιέχει τις ποσότητες ζήτησης των πελατών.

> R_i είναι η διαδρομή του i-οστού οχήματος.

> m είναι το πλήθος των οχημάτων. Οποιαδήποτε διαδρομή αντιστοιχεί σε ένα μόνο όχημα.

Αν ισχύει $c_ij=cj_i$ για κάθε (v_i,vj) e A το πρόβλημα καλείται συμμετρικό και

αρκεί τότε να αντικαταστήσουμε το σύνολο A με το σύνολο ακμών E=={(v_i,vj)/ v_i,vj e V; i<j). Με κάθε κορυφή v_i e V ' συσχετίζουμε μια ποσότητα q_i που παριστάνει το φορτίο που πρέπει να παραδοθεί στον κόμβο u_i.

Θα θεωρήσουμε ακόμη ένα χρόνο εξυπηρέτησης δi που απαιτείται για την εκφόρτωση της ποσότητας q_i στον κόμβο u_i. Είναι προφανές ότι η συνολική διάρκεια οποιασδήποτε διαδρομής δεν πρέπει να ξεπερνά ένα δοσμένο φράγμα D, οπότε μπορούμε να θεωρήσουμε για παράδειγμα ότι τα κόστη cij αντιστοιχούν στους χρόνους μετάβασης του οχήματος από ακμή σε ακμή του γράφου.

Μια δυνατή λύση (feasible solution) του VRP αποτελείται από:

• μια διαμέριση Rk (k=1->m) του V.

• μια μεταλλαγή (permutation) σ_i του συνόλου R_i U{v_0}, που να δηλώνει τη διαδοχή στην εξυπηρέτηση των πελατών, πάνω στη διαδρομή i.

Το κόστος μιας συγκεκριμένης διαδρομής R_i={v_0,v_1,.,v_{m+1}}, όπου v_0 = v_{m+1} (depot), δίνεται από το άθροισμα:

$$C(R_i) = \sum_{i=0}^{m} c_{i,i+1} + \sum_{i=0}^{m} \delta_i$$

Η διαδρομή Ri είναι μέλος μιας υποψήφιας λύσης, αν το όχημα i σταματά ακριβώς μια φορά σε κάθε κόμβο εξυπηρέτησης και η συνολική διάρκεια της διαδρομής δεν ξεπερνά ένα προκαθορισμένο κατώφλι D, δηλαδή έχουμε ότι $C(R_i)$<D. Τέλος, το κόστος της λύσης S του προβλήματος λαμβάνεται ίσο με $F_{VRP}(S)=\sum_{i=1}^{m} C(R_i)$

2.3.2 Παραλλαγές του VRP

Οι πρακτικές εφαρμογές, που έχουμε για το VRP, χαρακτηρίζονται από διάφορους παράπλευρους περιορισμούς. Για το λόγο αυτό υπάρχουν στη βιβλιογραφία, αρκετές παραλλαγές vehicle routing προβλημάτων. Ορισμένες από τις παραλλαγές αυτές, θα αναφέρουμε ευθύς αμέσως.

• Capacitated VRP (CVRP): Το CVRP είναι μια περίπτωση VRP, στην οποία ένας στόλος οχημάτων ίδιας χωρητικότητας (uniform capacity), πρέπει να εξυπηρετήσει γνωστές ζητήσεις πελατών (known customer demands), για

ένα δεδομένο και μοναδικό είδος εμπορεύματος (single commodity). Για όλα τα οχήματα υπάρχει μία και μόνο αποθήκη (depot) και στόχος μας είναι η σχεδίαση των διαδρομών ελαχίστου κόστους. Το CVRP μοιάζει με το πρωταρχικό VRP, για το οποίο μιλήσαμε προηγουμένως, με τον επιπλέον περιορισμό ότι όλα τα οχήματα έχουν την ίδια χωρητικότητα και εξυπηρετούν την παράδοση ενός είδους εμπορεύματος. Μπορούμε επομένως να δώσουμε την τυπική περιγραφή του CVRP:

1. Ο αντικειμενικός σκοπός του CVRP είναι η ελαχιστοποίηση του αριθμού των οχημάτων του στόλου και του αθροίσματος των χρόνων μετάβασης από κόμβο σε κόμβο (κόστη). Η συνολική ζήτηση των εμπορευμάτων για κάθε διαδρομή δεν πρέπει να ξεπερνά τη χωρητικότητα του κάθε οχήματος.

2. Ένα σύνολο διαδρομών στο CVRP συνιστά μια υποψήφια λύση, αν ικανοποιείται η προϋπόθεση που αναφέραμε ακριβώς παραπάνω για τη χωρητικότητα των οχημάτων.

3. Τα χαρακτηριστικά 1 και 2 ορίζουν ότι: Η λύση για το CVRP είναι ίδια με αυτή του VRP, με την προϋπόθεση ότι υπάρχει μια ποσότητα Q (χωρητικότητα οχήματος), τέτοια ώστε για κάθε διαδρομή R_i να ισχύει (με di παριστάνουμε τη συνολική ζήτηση στη διαδρομή Ri).

• Multiple Depots VRP (MDVRP): Μια εταιρεία διανομών μπορεί να διαθέτει πολλές αποθήκες, από τις οποίες εξυπηρετεί την πελατειακή ζήτηση που της παρουσιάζεται. Αν οι κόμβοι των πελατών διαμερίζονται σε υποσύνολα (clusters), που φέρουν ως γεωγραφικό τους επίκεντρο τις αποθήκες της εταιρείας, τότε το πρόβλημα της διανομής μπορεί να αναχθεί σε ένα σύνολο ανεξάρτητων μεταξύ τους VRPs. Αν όμως, οι πελάτες και οι αποθήκες διαπλέκονται μεταξύ τους, τότε πρέπει να επιλύσουμε ένα MDVRP. Ένα τέτοιο πρόβλημα απαιτεί την ανάθεση των πελατών στις διαθέσιμες αποθήκες. Σε κάθε αποθήκη υπάρχει ένας στόλος οχημάτων. Κάθε όχημα αναχωρεί από μια αποθήκη, εξυπηρετεί τους πελάτες που έχουν ανατεθεί στη συγκεκριμένη αποθήκη, και επιστρέφει στο σημείο από το οποίο ξεκίνησε. Δίνουμε την τυπική περιγραφή του MDVRP:

1. Ο αντικειμενικός σκοπός του MDVRP είναι η ελαχιστοποίηση του

αριθμού των οχημάτων του στόλου και του αθροίσματος των χρόνων μετάβασης από κόμβο σε κόμβο (κόστη). Η συνολική ζήτηση των εμπορευμάτων κάθε διαδρομής εξυπηρετείται από συγκεκριμένη αποθήκη, που γνωστοποιείται αμέσως μετά τη διαδικασία της ανάθεσης (assignment).

2. Ένα σύνολο διαδρομών στο MDVRP συνιστά μια υποψήφια λύση, αν κάθε διαδρομή ικανοποιεί τις προϋποθέσεις του πρωταρχικού VRP και επιπλέον ξεκινά και καταλήγει στην ίδια αποθήκη.

3. Τα χαρακτηριστικά 1 και 2 ορίζουν ότι: Το σύνολο των κόμβων στο MDVRP γράφεται σαν $V=\{v_1,^,v_n\}U\ V_0$ όπου $V_0=\{v_{01},..\ ,,v_{0d}\}$ είναι το σύνολο των κόμβων των αποθηκών. Μια διαδρομή i ορίζεται ως $Ri=\{d,vi,...,v_m,d\}$, όπου d e V_0. Το κόστος της διαδρομής υπολογίζεται όπως και στην περίπτωση του VRP.

• Periodic VRP (PVRP): Στο κλασσικό VRP, η σχεδίαση της βέλτιστης δρομολόγησης αφορά στο χρονικό διάστημα μιας ημέρας. Στην περίπτωση του PVRP όμως, πρέπει να αποφασίσουμε για τις διαδρομές του στόλου δεδομένου ότι η διαδικασία της εξυπηρέτησης διαρκεί M ημέρες. Δίνουμε την τυπική περιγραφή του PVRP:

1. Ο αντικειμενικός σκοπός του MDVRP είναι η ελαχιστοποίηση του αριθμού των οχημάτων του στόλου και του αθροίσματος των χρόνων μετάβασης από κόμβο σε κόμβο (κόστη).

2. Ένα σύνολο διαδρομών στο PVRP συνιστά μια υποψήφια λύση, αν κάθε διαδρομή ικανοποιεί τις προϋποθέσεις του κλασσικού VRP. Επιπλέον, οποιοδήποτε όχημα δεν επιτρέπεται να επιστρέψει στην αποθήκη αναχώρησής του, την ίδια μέρα κατά την οποία ξεκίνησε την προγραμματιζόμενη διαδρομή του από την αποθήκη αυτή. Με το πέρας της περιόδου των M ημερών, κάθε πελάτης θα πρέπει να έχει εξυπηρετηθεί τουλάχιστον μία φορά.

3. Για κάθε πελάτη είναι δεδομένη η ημερήσια ζήτηση φορτίου. Η εξυπηρέτηση σε κάθε κόμβο πρέπει να γίνει εξολοκλήρου σε μία επίσκεψη και από ένα μόνο όχημα. Αν η περίοδος M=1, το PVRP μεταπίπτει σε περίπτωση κλασσικού VRP. Κάθε κόμβος στο PVRP, πρέπει να προσπελαστεί k φορές, όπου 1<k<M. Στο κλασσικό PVRP, η ημερήσια

ζήτηση κάθε πελάτη είναι σταθερή.

Το PVRP μπορεί να αντιμετωπιστεί σαν ένα πρρόβλημα, στο οποίο ζητούμε ένα σύνολο διαδρομών για κάθε ημέρα έτσι ώστε οι συνολικοί περιορισμοί να ικανοποιούνται και να πετυχαίνουμε το ελάχιστο κόστος διαδρομών.

4. Το PVRP μπορεί να μελετηθεί και από τη σκοπιά ενός συνδυαστικού προβλήματος βελτιστοποίησης, πολλών επιπέδων (multi-level combinatorial optimization problem)

Split Delivery VRP (SDVRP): Το SDVRP αποτελεί μια χαλάρωση του VRP, με την έννοια ότι στο SDVRP επιτρέπεται η εξυπηρέτηση ενός πελάτη από διαφορετικά οχήματα, αν αυτό επιφέρει μείωση του συνολικού κόστους της δρομολόγησης. Η χαλάρωση αυτή είναι εξαιρετικά σημαντική όταν το μέγεθος της ζήτησης των πελατών είναι συγκρίσιμο με την χωρητικότητα των οχημάτων. Η εύρεση της λύσης στο SDVRP είναι αρκετά πιο πολύπλοκη διαδικασία από την αντίστοιχη του κλασσικού VRP. Δίνουμε την τυπική περιγραφή του SDVRP:

1. Ο αντικειμενικός σκοπός του MDVRP είναι η ελαχιστοποίηση του αριθμού των οχημάτων του στόλου και του αθροίσματος των χρόνων μετάβασης από κόμβο σε κόμβο (κόστη).

2. Ένα σύνολο διαδρομών στο PVRP συνιστά μια υποψήφια λύση, αν κάθε διαδρομή ικανοποιεί τις προϋποθέσεις του κλασσικού VRP. Ισχύει όμως εδώ ότι, ένας πελάτης μπορεί να εξυπηρετηθεί από περισσότερα του ενός οχήματα.

3. Μπορούμε να μετατρέψουμε το κλασσικό VRP σε SDVRP, επιβάλλοντας τον διαχωρισμό των παραδόσεων, αφού πρώτα διαμοιράσουμε τις αιτήσεις των πελατών σε επιμέρους αδιαίρετα υποσύνολα αιτήσεων.

VRP with Backhauls (VRPB): Το VRPB είναι μια προέκταση του CVRP, στην οποία το σύνολο των πελατών, $V\backslash\{v_0\}$, διαμοιράζεται σε δύο υποσύνολα. Το πρώτο υποσύνολο, έστω L, περιέχει n πελάτες «εμπρόσθιας εξυπηρέτησης» (linehaul customers), καθένας από τους οποίους ζητά να του παραδοθεί μια συγκεκριμένη ποσότητα φορτίου. Το δεύτερο υποσύνολο, έστω B, περιέχει m πελάτες «οπίσθειας εξυπηρέτησης» (backhaul

customers), από τους οποίους κάποια οχήματα πρέπει να παραλάβουν ορισμένη ποσότητα εισερχόμενου φορτίου. Οι κόμβοι αριθμώνται έτσι ώστε L={1,2,..,n} και B={n+1,..,n+m}. Στο VRPB υπάρχει μια προτεραιότητα ανάμεσα στους linehaul και backhaul πελάτες: όταν σε μια διαδρομή επίκειται να γίνει εξυπηρέτηση και των δύο ειδών πελατών, πάντα εξυπηρετούνται πρώτα οι linehaul πελάτες. Διαφορετικά, δεν ξεκινά η εξυπηρέτηση των backhaul πελατών. Μια μη αρνητική ποσότητα di, συμβολίζει το φορτίο παράδοσης ή παραλαβής που σχετίζεται με τον κόμβο i ενώ για τον κόμβο ανεφοδιασμού ισχύει ότι d0=0. Όταν ο πίνακας κόστους είναι μη συμμετρικός, το πρόβλημα αναφέρεται ως Asymmetric VRP With Backhauls (AVRPB). Δίνουμε την τυπική περιγραφή του VRPB:

1. Ο αντικειμενικός σκοπός του VRPB είναι η εύρεση ενός συνόλου, K το πλήθος απλών κλειστών διαδρομών (κύκλων) ελαχίστου κόστους, για το οποίο: (i) κάθε κύκλος διέρχεται από τον κόμβο ανεφοδιασμού (depot), (ii) ο κόμβος κάθε πελάτη ανήκει αποκλειστικά και μόνο σε μια κλειστή διαδρομή, (iii) η συνολική ζήτηση τόσο των backhaul όσο και των linehaul πελατών που εντοπίζονται στους κόμβους μιας κλειστής διαδρομής δεν ξεπερνούν (ξεχωριστά) τη χωρητικότητα C του οχήματος και (iv) σε κάθε κλειστή διαδρομή οι linehaul πελάτες προηγούνται των backhaul πελατών, εφόσον υπάρχουν.

2. Ένα σύνολο διαδρομών στο VRPB συνιστά μια υποψήφια λύση, αν όλες οι παραλαβές για κάθε διαδρομή ολοκληρώνονται προτού τα οχήματα παραλάβουν επιστροφές φορτίου. Ταυτόχρονα, τα περιθώρια χωρητικότητας κάθε οχήματος δεν παραβιάζονται είτε από τους linehaul είτε από τους backhaul κόμβους.

3. Το κόστος μιας διαδρομής υπολογίζεται όπως και στην περίπτωση του κλασσικού VRP, με τον επιπλέον περιορισμό ότι μια διαδρομή είναι δυνατή (feasible) αν και μόνον αν είναι delivery feasible, pick-up feasible και load-feasible. Καταρχήν, θα ορίσουμε με p το διάνυσμα που περιγράφει τις αιτήσεις των πελατών ως προς την επιστροφή φορτίου. Δίνουμε λοιπόν τις εξής περιγραφές:

Delivery feasible: Με τον όρο αυτό αναφερόμαστε στο ότι η

συνολική ποσότητα των εμπορευμάτων σε μια διαδρομή δεν πρέπει να υπερβαίνει την χωρητικότητα C του οχήματος. Για μια διαδρομή $R_i=\{v_0,v_1,.,v_{m+1}\}$ θα πρέπει να έχουμε ότι $C_d(v_k) < C$ και $C_d(v_{k+1}) > C$, όπου με $C_d(v_k)$ δηλώνουμε τη συνολική ποσότητα εμπορευμάτων που παραδόθηκε σε όλους τους πελάτες κατά μήκος μιας διαδρομής που ξεκινά από τον κόμβο v0 και καταλήγει στον κόμβο

$$v_k: C_d(v_k) = \sum_{v_i \in P(1,v_k)} d_i.$$

Με P(1,vk) συμβολίζουμε τους πελάτες που επισκέφτηκε το όχημα κατά μήκος της διαδρομής από τον κόμβο v0 μέχρι και τον κόμβο vk (περιλαμβάνεται και ο κόμβος v_k).

Pick up feasible: Ο περιορισμός αυτός εξασφαλίζει ότι το όχημα φέρει αρκετή χωρητικότητα ώστε να παραλάβει τα φορτία επιστροφής όλων των πελατών πάνω σε μια διαδρομή. Εδώ έχουμε ότι $C_p(v_k) < C$ και $C_p(v_{k+1}) > C$, όπου με Cp(vk) δηλώνουμε τη συνολική ποσότητα εμπορευμάτων που παραλείφθηκε από όλους τους πελάτες κατά μήκος μιας διαδρομής που ξεκινά από τον κόμβο v0 και καταλήγει στον κόμβο

$$v_k: \ C_p(v_k) = \sum_{v_i \in P(1,v_k)} p_i$$

(περιλαμβάνεται και ο κόμβος vk).

Load feasible: Ο περιορισμός της χωρητικότητας του οχήματος μπορεί να παραβιαστεί σε οποιονδήποτε κόμβο της διαδρομής. Μια τέτοια παραβίαση εξαρτάται από την σειρά των πελατών. Έστω $L(v_k)$ το φορτίο του οχήματος μετά από την προσπέλαση του κόμβου v_k. Υποθέτουμε ότι το όχημα φέρει ένα αρχικό φορτίο L(1) < C. Τότε το φορτίο του οχήματος σε κάθε σημείο της διαδρομής θα είναι $L(v_k)=C_p(v_k)+L(1)-C_d(v_k)$. Η εξίσωση αυτή δείχνει πως το τρέχον φορτίο μπορεί να ξεπερνά την χωρητικότητα C του οχήματος. Αυτό σημαίνει πως η προκύπτουσα διαδρομή είναι μη εφικτή διότι το όχημα δεν μπορεί να εξυπηρετήσει τον επόμενο κόμβο v_{k+1}. Άρα μια διαδρομή είναι load feasible αν $L(v_k) < C$ και $L(v_{k+1}) > C$.

• VRP with PickUp and Delivery (VRPPD): Στην περίπτωση του VRP with PickUp and Delivery, κάθε πελάτης i σχετίζεται με δύο ποσότητες d_i και p_i, που αναπαριστούν τη ζήτηση παραλαβής και επιστροφής φορτίου

αντίστοιχα. Αρκετά συχνά, μια μόνο ποσότητα, η διαφορά di-pi, αρκεί για να χαρακτηριστεί η αίτηση του κάθε πελάτη (η διαφορά αυτή μπορεί να είναι και αρνητική). Για κάθε πελάτη i, υπάρχει μια αποθήκη αποστολής φορτίου O_i και μια αποθήκη παραλαβής φορτίου D_i. Υποτίθεται ότι σε κάθε κόμβο πελάτη η παράδοση φορτίου από το όχημα συμβαίνει προτού ο πελάτης αποθέσει το φορτίο επιστροφής. Για το λόγο αυτό θα ισχύει και εδώ ότι $L(v_k)=C_p(v_k)+L(1)-C_d(v_k)$, σύμφωνα με τις παραπάνω επεξηγήσεις. Μπορούμε να θεωρήσουμε εδώ ότι η τυπική περιγραφή του VRPPD ακολουθεί τα χαρακτηριστικά 1,2 και 3 όπως και προηγουμένως.

• VRP With Satellite Facilities (VRPSF): Μια σημαντική εκδοχή του VRP είναι αυτή που προβλέπει τη μεσολάβηση δορυφόρων που ενημερώνουν το σύστημα ως προς το αν είναι αναγκαία η αναπλήρωση οχημάτων. Όταν είναι δυνατό, η μεσολάβηση δορυφόρων επιτρέπει τη συνέχιση της διαδικασίας εξυπηρέτησης από τα οχήματα, χωρίς να είναι απαραίτητη η επιστροφή τους στην κεντρική αποθήκη ανεφοδιασμού.

2.3.3 Τεχνικές Επίλυσης στο VRP

Παραθέτουμε στο σημείο αυτό, ορισμένες τεχνικές επίλυσης για το πρόβλημα δρομολόγησης οχημάτων. Σχεδόν όλες από αυτές αποτελούν ευριστικές και μετα- ευριστικές μεθόδους, δεδομένου ότι κανείς ακριβής αλγόριθμος δεν μπορεί να εγγυηθεί την εύρεση της βέλτιστης λύσης, ειδικά όταν το πλήθος των κόμβων είναι σχετικά μεγάλο. Αυτό οφείλεται στο ότι το VRP είναι ένα NP-hard πρόβλημα. Θα δώσουμε λοιπόν μια κατηγοριοποίηση των τεχνικών επίλυσης, που χρησιμοποιούνται ευρύτατα μέχρι σήμερα.

2.3.3.1 Ακριβής Μέθοδος

Όπως υπονοεί και το όνομά της, η μέθοδος αυτή προβλέπει τη συστηματική απαρίθμηση κάθε πιθανής λύσης μέχρις ότου βρεθεί η βέλτιστη δυνατή. Μια από τις πιο γνωστές, ακριβείς τεχνικές για το CVRP είναι η K-tree μέθοδος που έχει χρησιμοποιηθεί για την επίλυση CVRP 71 κόμβων (Branch and Bound Technique-Fischer 1994). Δοσμένου ενός γράφου με n+1 κόμβους, ένα K-δέντρο είναι ένα σύνολο n+K ακμών που επιλύει κάθε δυνατή

διαδρομή στο γράφο. Το VRP μπορεί να αντιμετωπιστεί σαν το πρόβλημα της εύρεσης του ελαχίστου K-δέντρου όταν ο βαθμός του κόμβου ανεφοδιασμού είναι 2K, ο βαθμός κάθε κόμβου εξυπηρέτησης είναι 2 και ικανοποιούνται οι περιορισμοί ως προς τη χωρητικότητα των οχημάτων. Παρόλο που η K-tree τεχνική έχει εφαρμοστεί με επιτυχία σε σύστημα 71 κόμβων, υπάρχουν περιπτώσεις με μικρότερο αριθμό κόμβων για τις οποίες η τεχνική αυτή δεν έχει δώσει λύσεις ακόμα. Προκειμένου να πετύχουμε λύσεις για τις περιπτώσεις αυτές, ή για να μειώσουμε τον υπολογιστικό χρόνο πρέπει να ανατρέξουμε σε ευριστικές μεθόδους.

Η K-tree τεχνική χρησιμοποιεί μια Branch-and-Bound διαδικασία βάσει της οποίας το πρόβλημα διαχωρίζεται, διορθώνοντας τις προσπτώσεις των ακμών του γράφου για τα επιλεγόμενα υποσύνολα των διαμοιρασμένων πελατών εξυπηρέτησης. Οι παράπλευροι περιορισμοί υπόκεινται σε δυϊσμό (dualized) προκειμένου να πετύχουμε μια χαλάρωση κατά Lagrange (Langrangian relaxation), οπότε το VRP ανάγεται στην εύρεση του ελαχίστου (κατά βαθμό κόμβου ανεφοδιασμού) K δέντρου.

Η K-tree τεχνική μπορεί να γενικευτεί προκειμένου να επιλύσουμε πιο ρεαλιστικά σενάρια VRP, στα οποία για παράδειγμα έχουμε ασσύμετρα κόστη, χρονοπαράθυρα και οχήματα διαφορετικής χωρητικότητας.

Ένας branch-and-bound αλγόριθμος χρησιμοποιεί τη στρατηγική του «διαίρει και βασίλευε». Διαχωρίζει το χώρο των λύσεων σε υποσύνολα και έπειτα εφαρμόζει τη διαδικασία της βελτιστοποίησης σε κάθε υποσύνολο χωριστά. Χρησιμοποιώντας τη λογική του branch-and-bound, εξετάζουμε αρχικά όλο τον χώρο των λύσεων S. Κατά τη φάση που φράσσουμε τις λύσεις, χαλαρώνουμε τους περιορισμούς του προβλήματος. Έτσι λοιπόν, δεχόμαστε λύσεις που δεν ανήκουν στο εφικτό σύνολο λύσεων S. Η χαλάρωση οδηγεί στην εκτίμηση ενός κάτω φράγματος της τιμής της βέλτιστης λύσης. Αν η λύση που δίνει η χαλάρωση ανήκει στο S ή έχει τιμή ίση με αυτή μιας λύσης s e S, τότε έχουμε βρει τη βέλτιστη λύση.

Διαφορετικά, καθορίζουμε n υποσύνολα $\{S_1,S_2,.,S_n\}$ του S τέτοια ώστε

$$\bigcup_{i=1}^{n} S_i = S$$

Κάθε ένα από τα υποσύνολα αυτά καλείται υποπρόβλημα. Ορισμένες

φορές τα $\{S_1,S_2,..,S_n\}$ καλούνται παιδιά του S. Προσθέτουμε τα παιδιά του S στη λίστα των υποψηφίων υποπροβλημάτων (αυτών δηλαδή που αναμένουν εξέταση). Η διαδικασία αυτή λέγεται branching. Για να συνεχίσουμε τον αλγόριθμο, επιλέγουμε ένα από τα υποψήφια υποπροβλήματα και το εξετάζουμε. Υπάρχουν τέσσερα δυνατά αποτελέσματα:

1. Αν βρούμε μια εφικτή λύση καλύτερη από τη λύση, την αντικαθιστούμε με τη λύση αυτή και συνεχίζουμε.

2. Μπορεί να βρούμε ότι το υποπρόβλημα δεν έχει λύση, οπότε το απορρίπτουμε.

3. Διαφορετικά, συγκρίνουμε το κάτω φράγμα του υποπροβλήματος με το καθολικό άνω φράγμα, που είναι η τιμή της καλύτερης εφικτής λύσης που έχουμε βρει μέχρι και το σημείο αυτό. Αν το κάτω φράγμα είναι μεγαλύτερο ή ίσο της τιμής αυτής, τότε μπορεί και πάλι να απορρίψουμε το υποπρόβλημα.

4. Εν τέλει, αν δεν μπορούμε να απορρίψουμε το υποπρόβλημα, επιβάλλεται να χρησιμοποιήσουμε την τεχνική branch και να προσθέσουμε τα παιδιά αυτού του υποπροβλήματος στη λίστα των ενεργών υποψηφίων. Συνεχίζουμε κατά αυτόν τον τρόπο, έως ότου η λίστα των υποψηφίων υποπροβλημάτων αδειάσει. Τότε θα έχουμε πετύχει τη βέλτιστη λύση του προβλήματος.

2.3.3.2 Ευριστικές Μέθοδοι

Οι ευριστικές μέθοδοι επιτελούν μια σχετικά περιορισμένη εξερεύνηση του χώρου των λύσεων και παράγουν αποτελεσματικές προσεγγίσεις, που χαρακτηρίζονται από χαμηλή υπολογιστική πολυπλοκότητα. Θα αναφέρουμε ονομαστικά ορισμένες βασικές ευριστικές τεχνικές που χρησιμοποιούνται στο VRP.

• Κατασκευαστικές μέθοδοι (Constructive Methods): Σταδιακά χτίζουμε μια εφικτή λύση ενώ ταυτόχρονα παρατηρούμε τις μεταβολές στο κόστος. Δεν βελτιστοποιούμε τη λύση σε κάθε φάση της τεχνικής. Κυριότερες μελέτες στις κατασκευαστικές μεθόδους:

- Savings (Clark and Wright 1964)

- ¬ Matching Based
- ¬ Multi-Route Improvement Heuristics
  - Thomson and Psaraftis (1993)
  - Van Breedam (1994)
  - Kinderwater and Savelsbergh (1997)

Αλγόριθμος 2 φάσεων (2-Phase Algorithm): Το πρόβλημα αποσυντίθεται στα δύο βασικά του χαρακτηριστικά: (i) διαμέριση του συνόλου των κορυφών σε υποσύνολα εφικτών διαδρομών και (ii) επί της ουσίας κατασκευή της διαδρομής, με πιθανούς βρόχους ανάδρασης ανάμεσα στις δύο φάσεις. Κυριότερες μελέτες για τον αλγόριθμο 2 φάσεων:

- ¬ Cluster-First, Route-Second Algorithms
  - Fisher and Jaikumar (1981)
  - The Petal Algorithm *S* The Sweep Algorithm
  - Taillard (1993)
- ¬ Route-First, Cluster-Second Algoriithms

2.3.4 Το Δυναμικό VRP (DVRP)

Το συμβατικό πρόβλημα δρομολόγησης αναφέρεται αλλιώς και ως SVRP (Static Vehicle Routing Problem), καθότι όλη η πληροφορία που σχετίζεται με το σχεδιασμό της δρομολόγησης είναι εκ των προτέρων γνωστή, προτού πριν ξεκινήσουν τα οχήματα τις διαδρομές εξυπηρέτησης στους κόμβους. Η πληροφορία αυτή δεν αλλάζει από τη στιγμή που αποφασίσουμε ποιοι δρόμοι περιέχονται στο σχήμα της βέλτιστης δρομολόγησης.

Με τον όρο «πληροφορία» εννοούμε οτιδήποτε σχετικό περιλαμβάνει τα χαρακτηριστικά των πελατών που μας ενδιαφέρουν, όπως για παράδειγμα τη γεωγραφική τους θέση, τον χρόνο που απαιτείται για την κατατόπου εξυπηρέτησή τους και φυσικά τη ζήτηση φορτίου για κάθε έναν τους. Επιπλέον, πληροφορίες του συνολικού (κατανεμημένου) συστήματος απαιτούνται να είναι γνωστές: οι χρόνοι μετάβασης από πελάτη σε πελάτη

πρέπει να είναι δοσμένοι στο σχεδιαστή, προκειμένου να εξαχθεί η βέλτιστη δρομολόγηση για κάθε υποσύνολο σταθμών εξυπηρέτησης.

Στα πιο πραγματικά σενάρια που αφορούν τη δρομολόγηση οχημάτων η συνδεόμενη με το πρόβλημα πληροφορία μεταβάλλεται δυναμικά από τη στιγμή που οι πρώτες διαδρομές σχεδιαστούν. Επίσης, η διαδικασία δρομολόγησης ξεκινά χωρίς να είναι γνωστή η συνολική πληροφορία που απαιτείται προκειμένου να σχεδιαστούν οι διαδρομές. Προφανώς, το Δυναμικό Πρόβλημα Δρομολόγησης είναι πιο σύνθετο σε σχέση με το συμβατικό πρόβλημα που αναφέραμε. Αν με P(DVRP) συμβολίσουμε την κλάση των δυναμικών προβλημάτων δρομολόγησης, και με P(SVRP) την κλάση των αντιστοίχων συμβατικών τότε θα έχουμε ότι P(SVRP) c P(DVRP).

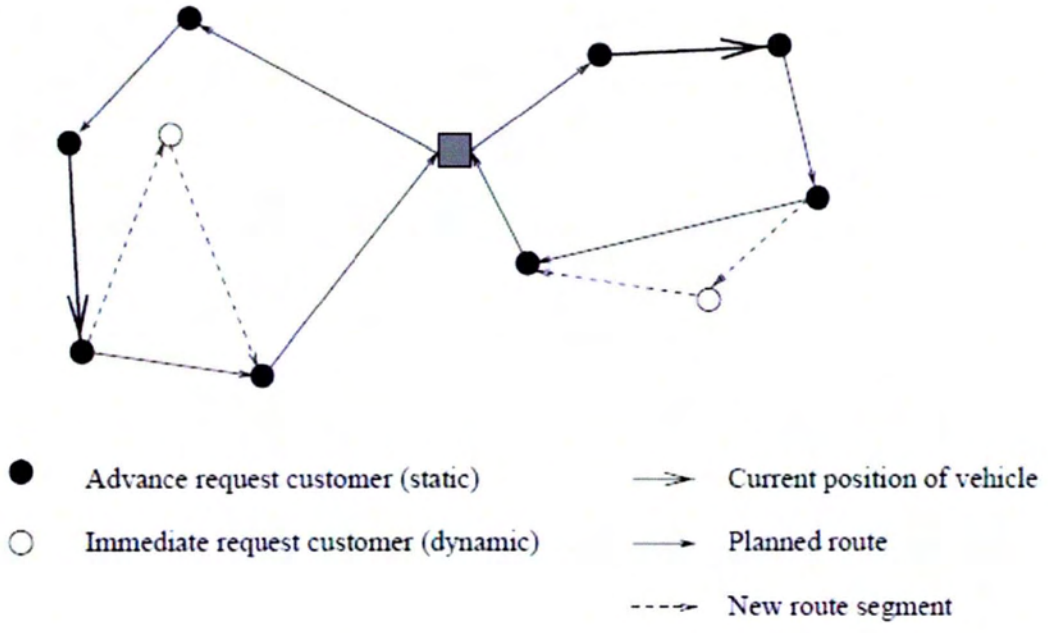
Το DVRP επιλύεται συνήθως με on-line αλγόριθμους που δουλεύουν σε πραγματικό χρόνο (real time), από τη στιγμή που γνωστοποιηθούν οι αιτήσεις των πελατών. Όπως και το SVRP έτσι και το DVRP κατατάσσεται στην κλάση των NP-hard προβλημάτων.

Λόγω της αυξημένης πολυπλοκότητας του DVRP, έχουν αναπτυχθεί προσεγγιστικές τεχνικές (ευριστικές και μετα-ευριστικές μέθοδοι) που δίνουν τη βέλτιστη λύση σε μια περιοχή του χώρου των λύσεων. Έτσι λοιπόν, χρησιμοποιούνται έντονα εδώ αλγόριθμοι σαν και αυτούς που αναφέραμε παραπάνω για το συμβατικό VRP με τη διαφορά ότι αρκετοί από αυτούς έχουν εμπλουτιστεί με προσθήκες δυναμικών πολιτικών (dynamic policies) προκειμένου να ανταποκρίνονται στα απαιτούμενα χαρακτηριστικά του προβλήματος.

Αν με p_t συμβολίσουμε το τρέχον στιγμιότυπο ενός δυναμικού προβλήματος δρομολόγησης, τότε με x_t θα θεωρούμε εκείνη τη λύση, που είναι γνωστή σαν δοκιμαστική λύση (tentative solution). Η δοκιμαστική λύση αναφέρεται στο τρέχον σύνολο εισόδων του συστήματος, και μόνο σ' αυτό. Αν κατά τη διάρκεια της δρομολόγησης καμία νέα αίτηση δεν δοθεί στο σχεδιαστή λέμε ότι η δοκιμαστική λύση είναι και η βέλτιστη λύση του προβλήματος.

Στην Εικόνα 4 δίνουμε ένα απλό παράδειγμα ενός προβλήματος δυναμικής δρομολόγησης. Στο παράδειγμα αυτό, δύο οχήματα, που αρχικά

δεν φέρουν φορτία, πρέπει να εξυπηρετήσουν κάποιους πελάτες, μερικοί εκ των οποίων εμφανίζονται με δεδομένη ζήτηση και μερικοί με ζήτηση που μεταβάλλεται κατά τη διάρκεια της δρομολόγησης. Οι πελάτες δεδομένης ζήτησης αναπαριστώνται με κόμβους μαύρου χρώματος, ενώ οι υπόλοιποι με κόμβους λευκού χρώματος. Τα συμπαγή βέλη δείχνουν τις διαδρομές που κατασκεύασε ο σχεδιαστής (dispatcher), πριν τα οχήματα αναχωρήσουν από την αποθήκη ανεφοδιασμού. Τα δύο βέλη έντονου μαύρου χρώματος δείχνουν τις θέσεις των οχημάτων τη χρονική στιγμή που καταφθάνει η δυναμική ζήτηση. Θεωρητικά, οι νέοι πελάτες θα πρέπει να προστεθούν πάνω στο υπάρχον δίκτυο εξυπηρέτησης, χωρίς να αλλάξει η σειρά των πελατών που δεν έχουν παραλάβει τη ζήτηση τους και βέβαια όλο αυτό θα πρέπει να γίνει με την ελάχιστη καθυστέρηση. Η περίπτωση αυτή αναπαρίσταται στον δεξιό υπογράφο του σχήματος. Στην πραγματικότητα όμως η εισαγωγή νέων πελατών επιφέρει επανασχεδίαση της σειράς εξυπηρέτησης των πελατών που δεν έχουν παραλάβει τη ζήτησή τους.



Εικόνα 4: Δυναμικό σενάριο δρομολόγησης οχημάτων, που περιλαμβάνει 8 κόμβους δεδομένης ζήτησης και 2 κόμβους μεταβαλλόμενης ζήτησης.

Σε γενικές γραμμές, όσο πιο περίπλοκο και σύνθετο είναι ένα πρόβλημα δρομολόγησης οχημάτων, τόσο πιο δύσκολη και σύνθετη θα είναι η

αντιμετώπιση της προσθήκης δυναμικών κόμβων εξυπηρέτησης. Για παράδειγμα, η προσθήκη νέων πελατών σε ένα πρόβλημα δρομολόγησης με χρονοπαράθυρα (time-windows) αποτελεί δυσκολότερη περίπτωση επίλυσης απ' ότι ένα δυναμικό πρόβλημα χωρίς χρονοπεριορισμούς. Ας σημειώσουμε ότι ένα δυναμικό σύστημα δρομολόγησης μπορεί να απορρίπτει τις αιτήσεις εξυπηρέτησης νέων πελατών, εφόσον αποφασίσει ότι η ενσωμάτωσή τους στο σχήμα της επικείμενης δρομολόγησης καταστρατηγεί τη βελτιστότητα του σχεδιασμού. Συχνά, μια τέτοιου είδους πολιτική περιλαμβάνει την προσφορά εξυπηρέτησης προς τους πελάτες που απορρίφθηκαν, αμέσως μόλις το σύστημα ολοκληρώσει τη διαδικασία εξυπηρέτησης για τους πελάτες που είχαν γίνει αποδεκτοί στο σχήμα της βέλτιστης δρομολόγησης.

Συχνά όταν μελετούμε συστήματα βέλτιστης δυναμικής δρομολόγησης οχημάτων, χρησιμοποιούμε τυχαία δεδομένα (randomly generated data). Αυτό γίνεται για δύο λόγους κυρίως. Ο πρώτος λόγος είναι ότι η τυχαιότητα των δεδομένων υποβοηθά στην κατασκευή συνόλων δεδομένων (datasets), στα οποία υπάρχει μια διευθέτηση των παραμέτρων του συστήματος. Ο δεύτερος λόγος έχει να κάνει με την πληρότητα των πληροφοριών που έχουμε για το πρόβλημα. Στα πραγματικά προβλήματα δρομολόγησης, είναι αρκετές οι φορές που ο σχεδιαστής δεν δύναται να έχει τις συντεταγμένες των οχημάτων και τους χρόνους άφιξης των αιτήσεων. Για το λόγο αυτό, μοντελοποιούμε το σύστημα βάσει τυχαίων δεδομένων έτσι ώστε να είμαστε σε θέση να λάβουμε αποφάσεις υπό συνθήκες έλλειψης πραγματικών δεδομένων.

Ο χρόνος αντίδρασης του συστήματος αποτελεί πολύ βασική παράμετρο που επηρεάζει την αξιοπιστία του σχεδιασμού. Ο χρόνος αυτός αφορά στο διάστημα ανάμεσα στην άφιξη της αίτησης και το τέλος του χρονοπλαισίου που ορίζει την πιο καθυστερημένη στιγμή για να ξεκινήσει η εξυπηρέτηση. Ο χρόνος αντίδρασης για κάθε κόμβο εξυπηρέτησης αποτελεί ένα μέτρο του κατά πόσο επείγει η εξυπηρέτηση στον κόμβο αυτό. Ο συνολικός χρόνος αντίδρασης του συστήματος εκφράζει την ένταση ή αλλιώς τον δυναμισμό (dynamism) του συστήματος. Γενικά, όσο πιο δυναμικό είναι το σύστημα τόσο πιο δύσκολο είναι για το σχεδιαστή να πετύχει μικρό χρόνο αντίδρασης.

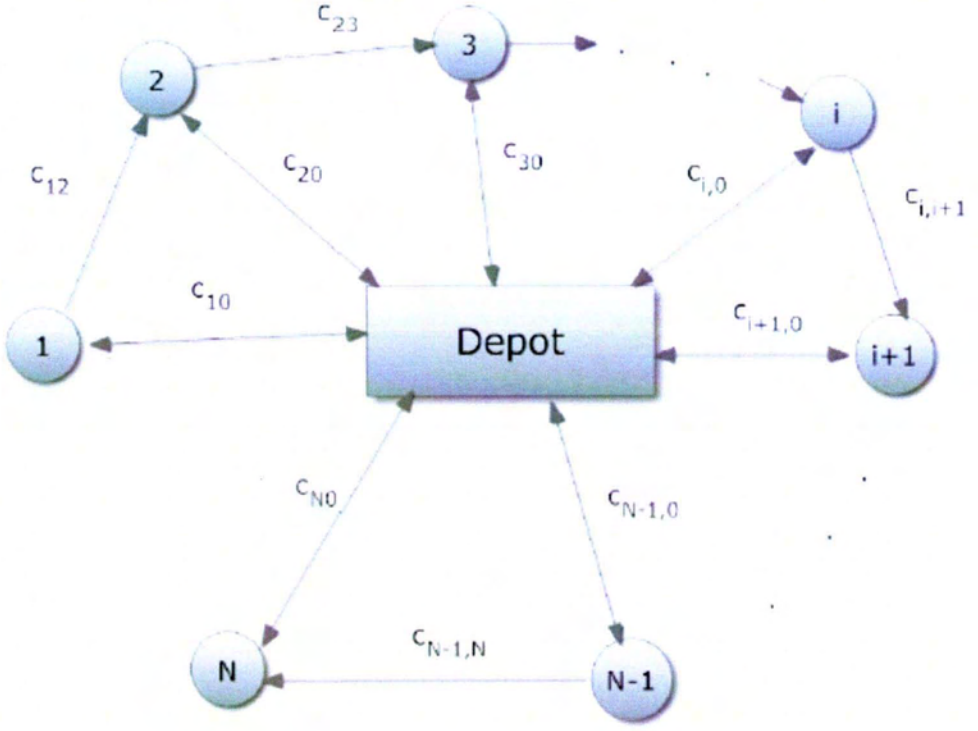
3 Παρουσίαση προβλήματος και αλγορίθμου

3.1 Σύντομη περιγραφή προβλήματος

Το πρόβλημα που καλούμαστε να λύσουμε σε αυτήν την εργασία, αφορά την περίπτωση του προμηθευτή που πρέπει να προμηθεύει με προϊόντα τους πελάτες του σε διαφορετικές πόλεις. Σε αυτό το πλαίσιο, αυτό που πρέπει να γίνει είναι να ελαχιστοποιηθεί το κόστος μεταφοράς για τις N πόλεις που πρέπει να επισκεφθεί το φορτηγό του προμηθευτή.

Συγκεκριμένα, στο πρόβλημά μας το φορτηγό μεταφέρει 2 προϊόντα, έστω Α προϊόν και Β προϊόν. Ανάλογα με την ζήτηση για την πρώτη πόλη θα φορτώσει x μονάδες από το Α και y από το Β. Φτάνοντας στην πρώτη πόλη αν δεν καλύψει την ζήτηση της τότε θα ξαναγυρίσει πίσω στην αποθήκη για να φορτώσει ενώ αν καλύψει την ζήτηση θα πάρει απόφαση για το αν θα πάει στην επόμενη πόλη με ότι του έχει μείνει ή θα επιστρέψει πίσω στην αποθήκη για να ξαναφορτώνεται.

Αυτό επαναλαμβάνεται μέχρι να καλύψει την ζήτηση και στις N πόλεις και τέλος να επιστρέψει πίσω στην αποθήκη. Το πρόβλημα αυτό, θα αντιμετωπιστεί με δυναμικό προγραμματισμό ξεκινώντας από την τελευταία πόλη οπού πρέπει μόνο να καλύψουμε την ζήτηση της N πόλης και να επιστρέψει ο πωλητής στην αποθήκη. Μετά υπολογίζουμε για την N-1 πόλη, και ούτω καθεξής. Το πρόβλημα περιγράφεται στην εικόνα που ακολουθεί:



Εικόνα 5: Παρουσίαση προβλήματος

Στην επόμενη ενότητα παρουσιάζεται αναλυτικά ο αλγόριθμος επίλυσης.

3.2 Παρουσίαση αλγορίθμου επίλυσης

Ο αλγόριθμος επίλυσης του παραπάνω προβλήματος βασίζεται καταρχήν στις εξής παραδοχής – μεγέθη, δεδομένου πως έχουμε 2 προϊόντα στην περίπτωσή μας:

z_i, i = 1,2 είναι οι ποσότητες των προϊόντων 1 και 2 που έχει ο πωλητής μετά την επίσκεψη στον κάθε πελάτη (δηλαδή στην κάθε πόλη). Αρνητικές τιμές της μεταβλητής αυτής σημαίνουν ότι ο πελάτης έμεινε ανικανοποίητος. Σε κάθε περίπτωση ισχύει για τις τιμές αυτές ο εξής περιορισμός:

$$|z_1 + z_2| \leq Q$$

όπου Q είναι η χωρητικότητα του φορτηγού σε προϊόντα.

Επίσης ορίζεται η ζήτηση που έχει ο κάθε πελάτης για κάθε προϊόν: ξ_1^j, ξ_2^j όπου j η πόλη του πελάτη (j = 1 ◊ N).

Αντίστοιχα ορίζονται και τα κόστη μετάβασης τόσο από πόλη σε πόλη ($c_{i,i+1}$) όσο και από κάθε πόλη στην αποθήκη ($c_{i,0}$). Για τα κόστη αυτά ισχύει η εξής συνθήκη:

$$c_{i,i+1} \leq c_{i0} + c_{0,i+1}\,, \qquad i = 1, \ldots . N - 1.$$

Ορίζουμε επίσης την μεταβλητή ζ ως εξής:

$\zeta = \sum_{i=1}^{2} \bar{z}_i$

όπου $\bar{z}_i = min\{0, z_i\}$.

Άρα όταν το ζ είναι αρνητικό, τότε υπήρχε έλλειψη σε ένα προϊόν, ενώ όταν είναι 0, δεν υπήρχε έλλειψη.

Οι επιλογές δράσης του πωλητή, είναι οι εξής, ανάλογα με το αν υπήρχε τελικά έλλειψη σε κάποιο προϊόν ή όχι:

Αν έχει έλλειψη (ζ<0), τότε ο πωλητής μπορεί είτε να επιστρέψει μια φορά στην αποθήκη, να γεμίσει προϊόντα, και αφού εξυπηρετήσει την πόλη i να προχωρήσει με ό,τι του έμεινε στην πόλη i+1, είτε να επιστρέψει δύο φορές

στην αποθήκη, μια για να εξυπηρετήσει την πόλη i και άλλη μια για να γεμίσει προϊόντα και να προχωρήσει στην πόλη i+1.

Αν δεν έχει έλλειψη, τότε μπορεί είτε να πάει στην αποθήκη, να γεμίσει προϊόντα και μετά προχωράει στην επόμενη πόλη, είτε να προχωρήσει απευθείας στην επόμενη πόλη χωρίς να πάει στην αποθήκη.

Για να λύσουμε το πρόβλημα, και να ελαχιστοποιηθεί το κόστος, τότε πρέπει να ελαχιστοποιηθεί η συνάρτηση f(z), όπου για κάθε πόλη j ισχύουν τα εξής:

1) Αν ζ = 0, τότε:

$$f_j(z) = min\{H_j(\bar{z}), A_j\}$$

2) Αν ζ < 0, τότε:

$$f_j(\bar{z}) = 2c_{j0} + min\left\{H_j\left(\sum_{i=1}^{2} \bar{z}_i\right), A_j\right\}$$

όπου:

$$H_j(\bar{z}) = c_{j,j+1} + Ef_{j+1}(\bar{z} - \bar{\xi}^{j+1}), \qquad z_1, z_2 \geq 0$$

$$\widetilde{H}_j(\zeta) = c_{j,j+1} + \min_{\bar{\theta}:\sum_{i=1}^{2}\theta_i = Q+\zeta} Ef_{j+1}(\bar{\theta} - \bar{\xi}^{j+1}), \qquad \zeta < 0$$

$$A_j = c_{j0} + c_{j+1,0} + \min_{\bar{\theta}:\sum_{i=1}^{2}\theta_i = Q} Ef_{j+1}(\bar{\theta} - \bar{\xi}^{j+1})$$

Στις παραπάνω σχέσεις, με θ ορίζουμε την ποσότητα του κάθε προϊόντος που γεμίζει ο πωλητής από την αποθήκη.

Με χρήση δυναμικού προγραμματισμού, ξεκινάμε από την τελευταία πόλη, όπου ισχύει ότι:

$$f_N(\bar{z}) = c_{N0} + 2c_{N0}1(\textstyle\sum_{i=1}^{2} \bar{z}_i < 0)$$

Και τελικά το πρόβλημα λύνεται με την ελαχιστοποίηση της συνάρτησης για την $1^{η}$ πόλη, δηλαδή της:

$$f_0 = c_{10} + \min_{\bar{\theta}:\Sigma_{i=1}^{2}\theta_i=Q} Ef_1\left(\bar{\theta} - \bar{\xi}^1\right)$$

Ο κώδικας επίλυσης του προβλήματος, είναι διαθέσιμος στο παράρτημα.

4 Αποτελέσματα προσομοιώσεων

Σε αυτή την ενότητα θα παρουσιάσουμε τα αποτελέσματα των προσομοιώσεων του κώδικα, για κάποια διαφορετικά σενάρια.

4.1 1ο σενάριο: N = 10, Q = 6

Στο 1° σενάριο λειτουργίας, υποθέτουμε ότι έχουμε 10 πελάτες σε 10 πόλεις, και το φορτηγό μπορεί να μεταφέρει ως 6 τεμάχια συνολικά και από τα 2 προϊόντα. Η ζήτηση είναι διωνυμική με n=3 και p=0,4 .

Θεωρούμε πως τα κόστη μετάβασης από την κάθε πόλη στην επόμενη, καθώς και από την κάθε πόλη στην αποθήκη, υπολογίζονται με βάση τον παρακάτω πίνακα:

Πίνακας 1: Κόστη μετάβασης, 1° σενάριο

	Costj0									
Costj0	**0**	**6**	**7**	**8**	**7**	**5**	**4**	**8**	**6**	**5**
	6	0	9	0	0	0	0	0	0	0
	7	9	0	6	0	0	0	0	0	0
	8	0	6	0	9	0	0	0	0	0
	7	0	0	9	0	5	0	0	0	0
	5	0	0	0	5	0	7	0	0	0
	4	0	0	0	0	7	0	10	0	0
	8	0	0	0	0	0	10	0	9	0
	6	0	0	0	0	0	0	9	0	8
	5	0	0	0	0	0	0	0	8	0

Στον πίνακα αυτό, στην 1η γραμμή (και στην 1η στήλη αντίστοιχα) φαίνονται τα κόστη μετάβασης από την κάθε πόλη στην αποθήκη. Αντίστοιχα στις θέσεις (i, i+1) και (i+1,i) φαίνονται τα κόστη μετάβασης από την μια πόλη στην άλλη.

Τα αποτελέσματα της προσομοίωσης φαίνονται στον επόμενο πίνακα. Εκεί, φαίνεται με 5 διαφορετικούς αριθμούς, η επιλογή που κάνει ο πωλητής ώστε να έχουμε βέλτιστη απόδοση του αλγορίθμου, ανάλογα με την τιμή του z για κάθε ένα προϊόν.

Θυμίζουμε ότι τα z μπορούν να πάρουν θεωρητικά τιμές από -6 ως 6 (δηλαδή για κάθε προϊόν, μπορεί είτε να υπάρχει απόλυτη έλλειψη, να ήταν δηλαδή ο πωλητής γεμάτος με το άλλο προϊόν και η ζήτηση να ήταν όλη για αυτό το προϊόν) ή να περισσέψει ολόκληρη η ποσότητά του.

Παράλληλα όμως, το άθροισμα z1 + z2 πρέπει πάντα να είναι ανάμεσα στο -6 και το 6.

Επομένως, στον παρακάτω πίνακα, η τιμή μηδέν (0) αντιστοιχεί σε περιπτώσεις μη εφικτές αναφορικά με τις τιμές του z, και άρα σε περιπτώσεις που ο πωλητής δεν αντιμετωπίζει. Από εκεί και πέρα, οι άλλες τέσσερις τιμές αντιστοιχούν στις εξής περιπτώσεις / αποφάσεις για τον πωλητή:

1: Είναι η περίπτωση που $\zeta < 0$, και $H^{\sim} < A$. Ο έμπορος έχει έλλειψη τουλάχιστον ενός προϊόντος στην πόλη i, και επιλέγει να επιστρέψει μια φορά στην αποθήκη, να γεμίσει προϊόντα, και αφού εξυπηρετήσει την πόλη i να προχωρήσει με ό,τι του έμεινε στην πόλη i+1

2: Είναι η περίπτωση που $\zeta < 0$, και $H^{\sim} > A$. Ο έμπορος έχει έλλειψη τουλάχιστον ενός προϊόντος στην πόλη i, και επιλέγει να επιστρέψει δύο φορές στην αποθήκη, μια για να εξυπηρετήσει την πόλη i και άλλη μια για να γεμίσει προϊόντα και να προχωρήσει στην πόλη i+1

3: Είναι η περίπτωση που $\zeta = 0$. Εδώ ο πωλητής πάει στην αποθήκη, γεμίζει προϊόντα και μετά προχωράει στην επόμενη πόλη.

4: Είναι η $2^{η}$ περίπτωση που $\zeta = 0$. Εδώ ο πωλητής προχωράει στην επόμενη πόλη χωρίς να πάει στην αποθήκη.

Όπως βλέπουμε λοιπόν από τον πίνακα, ανάλογα με τις τιμές των z1, z2 ο πωλητής παίρνει διαφορετική απόφαση έτσι ώστε να ελαχιστοποιήσει το συνολικό κόστος μεταφοράς.

Πίνακας 2: Αποτέλεσμα προσομοίωσης, 1° σενάριο

Z2/Z1	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6
-6	0	0	0	0	0	0	2	2	2	2	2	2	2
-5	0	0	0	0	0	2	2	2	2	2	2	2	2
-4	0	0	0	0	2	2	2	2	2	2	2	2	2
-3	0	0	0	2	2	2	2	2	2	2	2	2	2
-2	0	0	2	2	2	2	2	2	2	2	2	2	2
-1	0	2	2	2	2	2	1	1	1	1	1	1	1
0	2	2	2	2	2	1	3	3	3	3	3	3	3
1	2	2	2	2	2	1	3	3	3	3	3	3	0
2	2	2	2	2	2	1	3	3	3	4	4	0	0
3	2	2	2	2	2	1	3	3	4	4	0	0	0
4	2	2	2	2	2	1	3	3	4	0	0	0	0
5	2	2	2	2	2	1	3	3	0	0	0	0	0
6	2	2	2	2	2	1	3	0	0	0	0	0	0

4.2 2ο σενάριο: N = 10, Q = 10

Στο 2° σενάριο λειτουργίας, υποθέτουμε ότι έχουμε 10 πελάτες σε 10 πόλεις, και το φορτηγό μπορεί να μεταφέρει ως 10 τεμάχια συνολικά και από τα 2 προϊόντα. Η ζήτηση είναι διωνυμική με n=3 και p=0,4 .

Θεωρούμε πως τα κόστη μετάβασης από την κάθε πόλη στην επόμενη, καθώς και από την κάθε πόλη στην αποθήκη, είναι ίδια με το προηγούμενο σενάριο.

Τα αποτελέσματα της προσομοίωσης φαίνονται στον επόμενο πίνακα, όπως και προηγουμένως. Εδώ ο πίνακας είναι 21X21, καθώς υπάρχουν περισσότεροι δυνατοί συνδυασμοί ανάμεσα στις ποσότητες z1, z2, αλλά και πάλι αποτυπώνονται οι επιλογές που κάνει ο πωλητής ανάλογα με τον συνδυασμό, ώστε να βελτιστοποιήσει την απόδοση και να ελαχιστοποιήσει το κόστος διαδρομών.

Πίνακας 3: Αποτέλεσμα προσομοίωσης, 2° σενάριο

Z2/Z1	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	10
-10	0	0	0	0	0	0	0	0	0	0	2	2	2	2	2	2	2	2	2	2	2
-9	0	0	0	0	0	0	0	0	0	2	2	2	2	2	2	2	2	2	2	2	2
-8	0	0	0	0	0	0	0	0	2	2	2	2	2	2	2	2	2	2	2	2	2
-7	0	0	0	0	0	0	0	2	2	2	2	2	2	2	2	2	2	2	2	2	2
-6	0	0	0	0	0	0	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
-5	0	0	0	0	0	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
-4	0	0	0	0	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
-3	0	0	0	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1
-2	0	0	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1	1
-1	0	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1
0	2	2	2	2	2	2	2	1	1	1	3	3	3	3	3	3	3	3	3	3	3
1	2	2	2	2	2	2	2	1	1	1	3	3	3	3	3	3	3	3	3	3	0
2	2	2	2	2	2	2	2	1	1	1	3	3	3	3	3	3	3	3	3	0	0
3	2	2	2	2	2	2	2	1	1	1	3	3	3	3	4	4	4	4	0	0	0
4	2	2	2	2	2	2	2	1	1	1	3	3	3	4	4	4	4	0	0	0	0
5	2	2	2	2	2	2	2	1	1	1	3	3	3	4	4	4	0	0	0	0	0
6	2	2	2	2	2	2	2	1	1	1	3	3	3	4	4	0	0	0	0	0	0
7	2	2	2	2	2	2	2	1	1	1	3	3	3	4	0	0	0	0	0	0	0
8	2	2	2	2	2	2	2	1	1	1	3	3	3	0	0	0	0	0	0	0	0
9	2	2	2	2	2	2	2	1	1	1	3	3	0	0	0	0	0	0	0	0	0
10	2	2	2	2	2	2	2	1	1	1	3	0	0	0	0	0	0	0	0	0	0

4.3 3ο σενάριο: N = 10, Q = 4

Στο 3° σενάριο λειτουργίας, υποθέτουμε ότι έχουμε 10 πελάτες σε 10 πόλεις, και το φορτηγό μπορεί να μεταφέρει ως 4 τεμάχια συνολικά και από τα 2 προϊόντα. Η ζήτηση είναι διωνυμική με n=2 και p=0,4 .

Θεωρούμε πως τα κόστη μετάβασης από την κάθε πόλη στην επόμενη, καθώς και από την κάθε πόλη στην αποθήκη, είναι ίδια με το προηγούμενο σενάριο.

Τα αποτελέσματα της προσομοίωσης φαίνονται στον επόμενο πίνακα, όπως και προηγουμένως. Εδώ ο πίνακας είναι 9Χ9, καθώς υπάρχουν λιγότεροι δυνατοί συνδυασμοί ανάμεσα στις ποσότητες z1, z2, αλλά και πάλι αποτυπώνονται οι επιλογές που κάνει ο πωλητής ανάλογα με τον συνδυασμό, ώστε να βελτιστοποιήσει την απόδοση και να ελαχιστοποιήσει το κόστος διαδρομών.

Πίνακας 4: Αποτέλεσμα προσομοίωσης, 3° σενάριο

Z2/Z1	-4	-3	-2	-1	0	1	2	3	4
-4	0	0	0	0	2	2	2	2	2
-3	0	0	0	2	2	2	2	2	2
-2	0	0	2	2	1	1	1	1	1
-1	0	2	2	1	3	3	3	3	3
0	2	2	2	1	3	3	3	3	3
1	2	2	2	1	4	4	4	4	0
2	2	2	2	1	4	4	4	0	0
3	2	2	1	1	4	4	0	0	0
4	2	1	1	1	4	0	0	0	0

4.4 4ο σενάριο: N = 20, Q = 6

Στο 4° σενάριο λειτουργίας, υποθέτουμε ότι έχουμε 20 πελάτες σε 20 πόλεις, και το φορτηγό μπορεί να μεταφέρει ως 6 τεμάχια συνολικά και από τα 2 προϊόντα. Η ζήτηση είναι διωνυμική με n=3 και p=0,4 .

Θεωρούμε πως τα κόστη μετάβασης από την κάθε πόλη στην επόμενη, καθώς και από την κάθε πόλη στην αποθήκη, υπολογίζονται με βάση τον παρακάτω πίνακα:

Πίνακας 5: Κόστη μετάβασης, 4° σενάριο

0	9	10	2	10	7	1	3	6	10	10	2	10	10	5	9	2	5	10	8
9	0	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	7	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	1	0	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
10	0	0	9	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	10	0	7	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	7	0	4	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	4	0	7	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	7	0	2	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	2	0	8	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	8	0	1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	1	0	3	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	3	0	1	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	1	0	9	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	9	0	7	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	7	0	4	0	0
5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	10	0

10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	1
8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

Τα αποτελέσματα της προσομοίωσης φαίνονται στον επόμενο πίνακα, όπως και προηγουμένως. Όπως και προηγουμένως, αποτυπώνονται οι επιλογές που κάνει ο πωλητής ανάλογα με τον συνδυασμό, ώστε να βελτιστοποιήσει την απόδοση και να ελαχιστοποιήσει το κόστος διαδρομών. Παρατηρούμε παραδείγματος χάριν ότι εδώ τυχαίνει ο πωλητής πάντα να έχει έλλειψη προϊόντων (θυμίζουμε ότι οι ανάγκες του κάθε πελάτη σε κάθε προϊόν προκύπτουν τυχαία κάθε φορά), οπότε όλες οι επιλογές του πωλητή είναι είτε «1» είτε «2».

Πίνακας 6: Αποτέλεσμα προσομοίωσης, 3^{ο} σενάριο

Z2/Z1	**-6**	**-5**	**-4**	**-3**	**-2**	**-1**	**0**	**1**	**2**	**3**	**4**	**5**	**6**
-6	0	0	0	0	0	0	2	2	2	2	2	2	2
-5	0	0	0	0	0	2	2	2	2	2	2	2	2
-4	0	0	0	0	2	2	2	2	2	2	2	2	2
-3	0	0	0	2	2	2	1	1	1	1	1	2	2
-2	0	0	2	2	2	1	1	1	1	1	1	2	2
-1	0	2	2	2	1	1	1	1	1	1	1	2	2
0	2	2	2	1	1	1	2	2	2	2	2	2	2
1	2	2	2	1	1	1	2	2	2	2	2	2	0
2	2	2	2	1	1	1	2	2	2	2	2	0	0
3	2	2	2	1	1	1	2	2	2	2	0	0	0
4	2	2	2	1	1	1	2	2	2	0	0	0	0
5	2	2	2	1	1	1	2	3	0	0	0	0	0
6	2	2	2	1	1	1	3	0	0	0	0	0	0

4.5 5ο σενάριο: N = 20, Q = 10

Στο 5° και τελευταίο σενάριο λειτουργίας, υποθέτουμε ότι έχουμε 20 πελάτες σε 20 πόλεις, και το φορτηγό μπορεί να μεταφέρει ως 10 τεμάχια συνολικά και από τα 2 προϊόντα. Η ζήτηση είναι διωνυμική με n=3 και p=0,4 .

Θεωρούμε πως τα κόστη μετάβασης από την κάθε πόλη στην επόμενη, καθώς και από την κάθε πόλη στην αποθήκη, είναι ίδια με το προηγούμενο σενάριο.

Τα αποτελέσματα της προσομοίωσης φαίνονται στον επόμενο πίνακα, όπως και προηγουμένως. Εδώ ο πίνακας είναι 21Χ21, καθώς υπάρχουν περισσότεροι δυνατοί συνδυασμοί ανάμεσα στις ποσότητες z1, z2, αλλά και πάλι αποτυπώνονται οι επιλογές που κάνει ο πωλητής ανάλογα με τον συνδυασμό, ώστε να βελτιστοποιήσει την απόδοση και να ελαχιστοποιήσει το κόστος διαδρομών.

Πίνακας 7: Αποτέλεσμα προσομοίωσης, 5° σενάριο

Z2/Z1	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1	0	1	2	3	4	5	6	7	8	9	10
-10	0	0	0	0	0	0	0	0	0	0	2	2	2	2	2	2	2	2	2	2	2
-9	0	0	0	0	0	0	0	0	0	2	2	2	2	2	2	2	2	2	2	2	2
-8	0	0	0	0	0	0	0	0	2	2	2	2	2	2	2	2	2	2	2	2	2
-7	0	0	0	0	0	0	0	2	2	2	2	2	2	2	2	2	2	2	2	2	2
-6	0	0	0	0	0	0	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
-5	0	0	0	0	0	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
-4	0	0	0	0	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
-3	0	0	0	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1
-2	0	0	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1	1
-1	0	2	2	2	2	2	2	2	1	1	1	1	1	1	1	1	1	1	1	1	1
0	2	2	2	2	2	2	2	1	1	1	3	3	3	3	3	3	3	3	3	3	3
1	2	2	2	2	2	2	2	1	1	1	3	3	3	3	3	3	3	3	3	3	0
2	2	2	2	2	2	2	2	1	1	1	3	3	3	3	3	3	3	3	3	0	0
3	2	2	2	2	2	2	2	1	1	1	3	3	3	3	4	4	4	4	0	0	0
4	2	2	2	2	2	2	2	1	1	1	3	3	3	4	4	4	4	0	0	0	0
5	2	2	2	2	2	2	2	1	1	1	3	3	3	4	4	4	0	0	0	0	0
6	2	2	2	2	2	2	2	1	1	1	3	3	3	4	4	0	0	0	0	0	0
7	2	2	2	2	2	2	2	1	1	1	3	3	3	4	0	0	0	0	0	0	0
8	2	2	2	2	2	2	2	1	1	1	3	3	3	0	0	0	0	0	0	0	0
9	2	2	2	2	2	2	2	1	1	1	3	3	0	0	0	0	0	0	0	0	0
10	2	2	2	2	2	2	2	1	1	1	3	0	0	0	0	0	0	0	0	0	0

Βιβλιογραφία

I. *M. Campbell, L. Clarke, A. J. Kleywegt, and M. W. P. Savelsbergh. The inventory routing problem. In T. Crainic and G. Laporte, editors, Fleet Management and Logistics, pages 95-113. Kluwer Academic Publishers, Boston, 1998*

II. *M. Campbell and M. W. P. Savelsbergh. A decomposition approach for the inventory-routing problem. Transportation Science, 38(4):488-502, 2004.*

III. *L. Chan, A. Federgruen, and D. Simchi-Levi. Probabilistic analyses and practical algorithms for inventory-routing models. Operations Research, 46(1):96—106, 1998.*

IV. *T. W. Chien, A. Balakrishnan, and E. T. Wong. An integrated inventory allocation and vehicle routing problem. Transport Science, 23:67-76, 1989.*

V. *M. Dorigo and G. Di Caro. The ant colony optimization meta-heuristic. In D. Corne, M. Dorigo, and F. Glover, editors, New Ideas in Optimization, pages 11-32. McGraw- Hill, 1999.*

VI. *M. Dorigo and L. M. Gambardella. Ant colony system: A cooperative learning approach to the traveling salesman problem. IEEE Transactions on Evolutionary Computation, 1 (1) :53—66, 1997.*

VII. *M. Dorigo and T. Stutzle. Ant Colony Optimization. MIT Press, 2004.*

VIII. *Gianpaolo Ghiani, Antonella Quaranta, Chefi Triki, New Policies for the Dynamic Traveling Salesman Problem, Optimization Methods and Software, 2007.*

IX. *Zhao LIU, Lishan KANG, A Hybrid Algorithm of n-OPT and GA to Solve Dynamic TSP.*

X. *Paolo Toth, Daniele Vigo, The Vehicle Routing Problem, SIAM Monographs on Discrete Mathematics and Applications, 2002.*

XI. *Tonci Caric, Hrvoje Gold, Vehicle Routing Problem, In-Tech 2008.*

XII. *Christine Wei Wu, J. Christopher Beck, Kenneth N. Brown, Dynamic Vehicle Routing With Uncertain Customer Demand, Cork Constraint Computation Center, Department of Computer Science, University College Cork, Cork, Ireland.*

XIII. *Gilbert Laporte, Metaheuristics for the Vehicle Routing Problem: Fifteen Years of Research, Canada Research Chair in Distribution Management, HEC Montreal.*

XIV. *Bruce Golden, S. Raghavan, Edward Wasil, The Vehicle Routing Problem: Latest Advantages and New Challenges, Springer Science & Business Media, LLC, 2008.*

Παράρτημα: Ο κώδικας

```
%function R = R5914
clear all
N = 10;

Q = 6;

  costj0=[6 7 8 7 5 4 8 6 5 8];
  costij=[9 6 9 5 7 10 9 8 9];

% for i=1:N
%     costj0(i) = randi(10,1);
% end
%
% for i=1:(N-1)
%
%     temp(i) = randi(10,1);
%     while temp(i) > costj0(i)+costj0(i+1)
%         temp(i) = randi(10,1);
%     end
%     costij(i)=temp(i);
% end



for i = 1:N
    for j = 1:N
        if i == 1 && j > 1
            COST(i,j)=costj0(j-1);
        elseif i > 1 && j == 1
            COST(i,j)=costj0(i-1);
        elseif (i==j+1)&&(i>1 && j>1)
            COST(i,j)=costij(j-1);
        elseif (j == i+1)&&(i>1 && j>1)
            COST(i,j)=costij(i-1);
        else
            COST(i,j)=0;
        end
    end
end
ksi=binornd(3,0.4,2,N);

f = zeros(N,2*Q+1,2*Q+1);

%%% ----  ALGORITHM BEGINS -------------%%%

%% ΥΠΟΛΟΓΙΣΜΟΣ ΤΗΣ f(N)
for z1=-Q:Q
    for z2=-Q:Q
```

```
        if ((z1+z2)>= - Q) && ((z1+z2)<= Q) %% Συνθήκη στη
σελίδα 5 του άρθρου
            z1matrix=z1+Q+1; %% Για να γίνεται η αντίστοιχιση
στα indeces του πίνακα
            z2matrix=z2+Q+1;
            if (z1<0) || (z2<0)     %% Εξίσωση 6 του άρθρου
                f(N,z1matrix,z2matrix)=3*costj0(N);
            else
                f(N,z1matrix,z2matrix)=costj0(N);
            end
        end
    end
end

%% αρχικοποίηση μεταβλητών



fmin = 10000*ones(1,N);
fmin2 = 10000*ones(1,N);
fmin3 = 10000*ones(1,N);
fmin4 = 10000*ones(1,N);

H = zeros(1,N);
Htilda = zeros(1,N);
Htilda_zita= zeros(1,N);
x = zeros(4,4);
x1 = zeros(4,4);
x2 = zeros(4,4);
x3 = zeros(4,4);
x4 = zeros(4,4);
%% Υλοποίηση αλγορίθμου


for j2 = 1:N-1
    j= N-j2;
    zita = -1;
    % Υπολογισμός του A

                for theta1b = 0:Q
                    theta2b = Q-theta1b;

                    for ksi1=0:3
                      for ksi2=0:3
                          probabil_1 =
factorial(3)/(factorial(3-
ksi1)*factorial(ksi1))*(0.4^ksi1)*0.6^(3-ksi1);
                          probabil_2 =
factorial(3)/(factorial(3-
ksi2)*factorial(ksi2))*(0.4^ksi2)*0.6^(3-ksi2);
                    % υπολογισμός του νέου z (στην εξίσωση 5)

    %                       z1b=theta1b-ksi(1,j+1);
    %                       z2b=theta2b-ksi(2,j+1);
                          z1b=theta1b-ksi1;
```

```
                                z2b=theta2b-ksi2;
                                z1bmatrix=z1b+Q+1;
                                z2bmatrix=z2b+Q+1;

x(ksi1+1,ksi2+1)=f(j+1,z1bmatrix,z2bmatrix)*probabil_1*probabi
l_2;
%                                 if x(ksi1+1,ksi2+1)<fmin3(j+1)
%υπολογίζω την ελάχιστη τιμή της f
%                                     fmin3(j+1) =
x(ksi1+1,ksi2+1);
%                                 end
                            end
                        end
                        if sum(sum(x))<fmin3(j+1)
                            fmin3(j+1)=sum(sum(x));
                        end

                    end
            A(j)=costj0(j) + costj0(j+1) + fmin3(j+1); % εξίσωση 5
            fmin3(j+1)=1000;
            while zita>=-Q
                %Υπολογίζουμε το H~


                    for theta1 = 0:(Q+zita)
                        theta2 = (Q+zita)-theta1;

                        for ksi1=0:3
                         for ksi2=0:3
                                probabil_1 =
factorial(3)/(factorial(3-
ksi1)*factorial(ksi1))*(0.4^ksi1)*0.6^(3-ksi1);
                                probabil_2 =
factorial(3)/(factorial(3-
ksi2)*factorial(ksi2))*(0.4^ksi2)*0.6^(3-ksi2);
%                           z1=theta1-ksi(1,j+1);
%                           z2=theta2-ksi(2,j+1);
                        z1=theta1-ksi1;
                        z2=theta2-ksi2;
                        z1matrix=z1+Q+1;
                        z2matrix=z2+Q+1;


x1(ksi1+1,ksi2+1)=f(j+1,z1matrix,z2matrix)*probabil_1*probabil
_2;
%                           if x1(ksi1+1,ksi2+1)<fmin(j+1)
%υπολογίζω την ελάχιστη τιμή της f
%                               fmin(j+1) = x1(ksi1+1,ksi2+1);
%                               z1matrix_min=z1matrix;
%                               z2matrix_min=z2matrix;
%                           end

                    end

                        end
```

```
                    if sum(sum(x1))<fmin(j+1)
                        fmin(j+1)=sum(sum(x1));
                          z1matrix_min=z1matrix;
                          z2matrix_min=z2matrix;
                    end
            end
            Htilda_zita(j) = costij(j)+ fmin(j+1); % εξίσωση 4
            fmin(j+1)=1000;

            if A(j)< Htilda_zita(j)
            s(j)=zita+1;
            for z1=-Q:Q
                for z2=-Q:Q
                    z1matrix=z1+Q+1;
                    z2matrix=z2+Q+1;
                    if ((z1+z2)>= - Q) && ((z1+z2)<= Q)
                        if z1 < 0
                            z1tilda = z1;
                        else
                            z1tilda=0;
                        end
                        if z2 < 0
                            z2tilda = z2;
                        else
                            z2tilda=0;
                        end

                        if (s(j)<=(z1tilda+z2tilda))&&((z1tilda +
z2tilda)<0)

                                for theta1a =
0:(Q+z1tilda+z2tilda)
                                    theta2a = (Q+z1tilda+z2tilda)-
theta1a;

                                    for ksi1=0:3
                                       for ksi2=0:3
                                probabil_1 =
factorial(3)/(factorial(3-
ksi1)*factorial(ksi1))*(0.4^ksi1)*0.6^(3-ksi1);
                                probabil_2 =
factorial(3)/(factorial(3-
ksi2)*factorial(ksi2))*(0.4^ksi2)*0.6^(3-ksi2);
%                                    z1a=theta1a-ksi(1,j+1);
%                                    z2a=theta2a-ksi(2,j+1);
                                    z1a=theta1a-ksi1;
                                    z2a=theta2a-ksi2;
                                    z1amatrix=z1a+Q+1;
                                    z2amatrix=z2a+Q+1;

x2(ksi1+1,ksi2+1)=f(j+1,z1amatrix,z2amatrix)*probabil_1*probab
il_2;
%                                    if
x2(ksi1+1,ksi2+1)<fmin2(j+1) %υπολογίζω την ελάχιστη τιμή της
f
```

```
%                                          fmin2(j+1) =
x2(ksi1+1,ksi2+1);
%                                  end

                                end
                                    end
                                if sum(sum(x2))<fmin2(j+1)
                                    fmin2(j+1)=sum(sum(x2));
                                end
                            end
                                Htilda(j) = costij(j)+ fmin2(j+1);
                                fmin2(j+1)=1000;

f(j,z1matrix,z2matrix)=2*costj0(j)+ Htilda(j); %% case where
zita<0, Htilda<A. Ο έμπορος έχει ελλειψη τουλάχιστον ενός
προϊόντος στην πόλη i, και επιλέγει να επιστρέψει μια φορά
στην αποθήκη, να γεμίσει προϊόντα, και αφου εξυπηρετήσει την
πόλη i να προχωρήσει με ό,τι του έμεινε στην πόλη i+1
                                MATRIX(z1matrix,z2matrix) = 1;
                        else
f(j,z1matrix,z2matrix)=2*costj0(j)+A(j); %% case where zita<0,
Htilda>A Ο έμπορος έχει ελλειψη τουλάχιστον ενός προϊόντος
στην πόλη i, και επιλέγει να επιστρέψει δύο φορές στην
αποθήκη, μια για να εξυπηρετήσει την πόλη i και αλλη μια για
να γεμισει προϊοντα και να προχωρήσει στην πόλη i+1
                                MATRIX(z1matrix,z2matrix) = 2;
                        end
                    end
                end
            end
            zita=-Q-2;
        else
            zita = zita - 1;
        end
            end


    if zita == -Q-1
        s(j)=-Q;
            for z1=-Q:Q
                for z2=-Q:Q
                    z1matrix=z1+Q+1;
                    z2matrix=z2+Q+1;
                    if ((z1+z2)>= - Q) && ((z1+z2)<= Q)
                        if z1 < 0
                            z1tilda = z1;
                        else
                            z1tilda=0;
                        end
                        if z2 < 0
                            z2tilda = z2;
                        else
                            z2tilda=0;
                        end
```

```
                    if (z1tilda + z2tilda)<0

                      for theta1a = 0:(Q+z1tilda+z2tilda)
                        theta2a = (Q+z1tilda+z2tilda)-theta1a;
                                  for ksi1=0:3
                                     for ksi2=0:3
                             probabil_1 =
factorial(3)/(factorial(3-
ksi1)*factorial(ksi1))*(0.4^ksi1)*0.6^(3-ksi1);
                             probabil_2 =
factorial(3)/(factorial(3-
ksi2)*factorial(ksi2))*(0.4^ksi2)*0.6^(3-ksi2);
%                                  z1a=theta1a-ksi(1,j+1);
%                                  z2a=theta2a-ksi(2,j+1);
                                z1a=theta1a-ksi1;
                                z2a=theta2a-ksi2;
                                z1amatrix=z1a+Q+1;
                                z2amatrix=z2a+Q+1;

x3(ksi1+1,ksi2+1)=f(j+1,z1amatrix,z2amatrix)*probabil_1*probab
il_2;
%                                  if
x2(ksi1+1,ksi2+1)<fmin2(j+1) %υπολογίζω την ελάχιστη τιμή της
f
%                                       fmin2(j+1) =
x2(ksi1+1,ksi2+1);
%                                  end
                             end

                                  end
                    if sum(sum(x3))<fmin2(j+1)
                        fmin2(j+1)=sum(sum(x3));
                    end
                       end
                            Htilda(j) = costij(j)+ fmin2(j+1);
                            fmin2(j+1)=1000;

f(j,z1matrix,z2matrix)=2*costj0(j)+ Htilda(j); %% case where
zita<0, Htilda<A. Ο έμπορος έχει ελλειψη τουλάχιστον ενός
προϊόντος στην πόλη i, και επιλέγει να επιστρέψει μια φορά
στην αποθήκη, να γεμίσει προϊόντα, και αφου εξυπηρετήσει την
πόλη i να προχωρήσει με ό,τι του έμεινε στην πόλη i+1
                            MATRIX(z1matrix,z2matrix) = 1;
                    end
                end
            end
        end
  end


  for z1=0:Q
      z2 = Q-z1;
      while z2>=0
      z1matrix=z1+1+Q;
      %z2matrix=z2+1+Q;
```

```
                for ksi1=0:3
                   for ksi2=0:3
                       z1a=z1-ksi1;
                       z2a=z2-ksi2;
                       z1bmatrix=z1a+Q+1;
                       z2bmatrix=z2a+Q+1;
                            probabil_1 =
factorial(3)/(factorial(3-
ksi1)*factorial(ksi1))*(0.4^ksi1)*0.6^(3-ksi1);
                            probabil_2 =
factorial(3)/(factorial(3-
ksi2)*factorial(ksi2))*(0.4^ksi2)*0.6^(3-ksi2);

x4(ksi1+1,ksi2+1)=f(j+1,z1bmatrix,z2bmatrix)*probabil_1*probab
il_2;
                   end
                end
%                       if sum(sum(x4))<fmin4(j+1)
%                           fmin4(j+1)=sum(sum(x4));
%                       end
          H(j)=costij(j)+sum(sum(x4));

           if H(j) > A(j)
               s2(j,z1+1)=z2+1;
               for z2=0:Q-z1
                   z2matrix=z2+1+Q;
               if z2<s2(j,z1+1)

                   f(j,z1matrix,z2matrix)=A(j); %% case where
zita =0. Εδώ ο πωλητης πάει στην αποθήκη, γεμίζει προϊόντα και
μετα προχωράει στην επόμενη πολη
                   MATRIX(z1matrix,z2matrix) = 3;
               else

                for ksi1=0:3
                  for ksi2=0:3
                            probabil_1 =
factorial(3)/(factorial(3-
ksi1)*factorial(ksi1))*(0.4^ksi1)*0.6^(3-ksi1);
                            probabil_2 =
factorial(3)/(factorial(3-
ksi2)*factorial(ksi2))*(0.4^ksi2)*0.6^(3-ksi2);
                              x4(ksi1+1,ksi2+1)=f(j+1,z1-
ksi1+Q+1,z2-ksi2+Q+1)*probabil_1*probabil_2;
                  end
                end

                 H(j)=costij(j)+sum(sum(x4));

                   f(j,z1matrix,z2matrix)=H(j); %% case where
zita =0. Εδώ ο πωλητης προχωράει στην επόμενη πολη χωρίς να
πάει στην αποθήκη
                   MATRIX(z1matrix,z2matrix) = 4;
               end
```

```
                end
                z2=-2;
            else
                z2 = z2-1;
            end

            if z2 == -1
               s2(j,z1+1)=0;
               for z2=0:Q-z1
                 z2matrix=z2+1+Q;
                 for ksi1=0:3
                   for ksi2=0:3
                       z1a=z1-ksi1;
                       z2a=z2-ksi2;
                       z1bmatrix=z1a+Q+1;
                       z2bmatrix=z2a+Q+1;
                       probabil_1 = factorial(3)/(factorial(3-
ksi1)*factorial(ksi1))*(0.4^ksi1)*0.6^(3-ksi1);
                       probabil_2 = factorial(3)/(factorial(3-
ksi2)*factorial(ksi2))*(0.4^ksi2)*0.6^(3-ksi2);

x4(ksi1+1,ksi2+1)=f(j+1,z1bmatrix,z2bmatrix)*probabil_1*probab
il_2;
                   end
                 end
               end
              H(j)=costij(j)+sum(sum(x4));
              f(j,z1matrix,z2matrix)=H(j);
              MATRIX(z1matrix,z2matrix) = 4;
              end


      end

  end
end
```