Πανεπιστήμιο Πειραιώς
Σχολή Ναυτιλίας και Βιομηχανίας
Τμήμα Βιομηχανικής Διοίκησης και Τεχνολογίας
Μεταπτυχιακό πρόγραμμα στην Βιομηχανική Διοίκηση και Τεχνολογία
Κατεύθυνση :Διοίκηση Logistics

Πειραιάς 2016

Επιβλέπουσα Καθηγήτρια: Σοφιανοπούλου Στέλλα

Διερεύνηση τεχνικών επίλυσης προβλήματος διανομής με χρονικά παράθυρα πελατών και ομοειδή φορτηγά

ΜΕΤΑΠΤΥΧΙΑΚΗ ΔΙΑΤΡΙΒΗ

ΧΑΡΑΛΑΜΠΟΣ ΑΝΤΩΝΙΑΔΗΣ

Περίληψη

Ο σκοπός αυτής της εργασίας είναι διττός. Αρχικός και βασικότερος σκοπός είναι να διερευνηθούν οι προγραμματιστικές απαιτήσεις για την λύση ενός προβλήματος Vehicle Routing με Time Windows βασιζόμενο σε πραγματικά δεδομένα από το χώρο της βιομηχανίας. Πέρα από την δημιουργία προγράμματος με τη χρήση ευρετικών και μεταευρετικών αλγόριθμων για την επίλυση της δρομολόγησης, δεύτερος στόχος είναι η διερεύνηση των σύγχρονων διαδικτυακών εργαλείων, για την ανεύρεση των γεωγραφικών στιγμάτων, τον υπολογισμό των μεταξύ τους αποστάσεων, την απεικόνιση των λύσεων σε χάρτες.

Ευχαριστίες

Στους ανθρώπους που συνεχίζουν να προσφέρουν ανιδιοτελώς, την Εύη, τους γονείς μου.

Table of Contents

Εισαγωγή

Το πρόβλημα δρομολόγησης είναι ένα συνδυαστικό πρόβλημα βελτιστοποίησης που προκύπτει πολύ συχνά σε καθημερινές εφαρμογές σχετικές με τις μεταφορές, τα logistics και στη χρονοδρομολόγηση. Η οικογένεια των VRP (Vehicle Routing Problem) προβλημάτων ανήκει στην κατηγορία των NP-hard (Nondeterministic Polynomial Time Hard) προβλημάτων είναι πολύ δύσκολα στην επίλυση και ειδικά όταν αυξάνει το πλήθος των πελατών. Έχουν αντιμετωπισθεί με πολλές τεχνικές συμπεριλαμβάνοντας τόσο ακριβείς όσο και ευρετικές μεθόδους όμως το πολύ υψηλό υπολογιστικό κόστος των πρώτων μεθόδων και η πολύ χαμηλή απόδοσή τους σε μεγάλα προβλήματα έχει οδηγήσει την έρευνα να επικεντρωθεί στους στοχαστικούς αλγόριθμους που είναι ικανοί να προσφέρουν λύσεις ικανοποιητικές σε μεγάλα προβλήματα, όχι κατά ανάγκη βέλτιστες σε σχετικά καλό υπολογιστικό χρόνο.

Τέτοιου είδους προβλήματα αντιμετωπίζουμε σε τομείς όπως τα οικονομικά, η μηχανική και η βιομηχανία και απαιτούν το σχεδιασμό αποδοτικών αλγόριθμων. Όταν το πρόβλημα είναι υψηλής πολυπλοκότητας, όπως ένα NP-hard πρόβλημα, είναι γενικά αποδεκτή η εφαρμογή τεχνικών – μεθόδων ευρετικών και μεταευρετικών γιατί προσφέρουν προσεγγιστικές λύσεις και βοήθεια στη λήψη αποφάσεων των στελεχών της βιομηχανίας σε πολύπλοκα προβλήματα με ταχύ τρόπο.

Λόγω του γεγονότος ότι το VRP και όλες οι παραλλαγές του είναι προβλήματα της κλάσης NP-Hard, η επίλυσή τους είναι ιδιαίτερα δύσκολη και σε ορισμένες περιπτώσεις η ακριβής επίλυση μέσω ενός εξονυχιστικού ελέγχου όλων των πιθανών διαδρομών είναι ανέφικτη. Για τον λόγο αυτό έχουν αναπτυχθεί ορισμένοι αλγόριθμοι που δεν βρίσκουν την βέλτιστη λύση αλλά λύση που προσεγγίζει την βέλτιστη, παράγοντας μια ή περισσότερες ικανοποιητικές διαδρομές. Οι βασικές κατηγορίες των αλγορίθμων αυτών με κριτήριο αν είναι ακριβείς ή προσεγγιστικοί αλγόριθμοι και βάσει του χρόνου επίλυσης παρουσιάζονται στην συνέχεια.

Ακριβείς αλγόριθμοι

Οι ακριβείς αλγόριθμοι εύρεσης της βέλτιστης λύσης ελέγχουν συστηματικά όλες τις πιθανές διαδρομές και επιλέγουν την καλύτερη. Παρότι είναι οι μόνοι αλγόριθμοι που βρίσκουν την βέλτιστη λύση, όσο μεγαλώνει το μέγεθος του προβλήματος τόσο αυξάνει και ο χρόνος επίλυσής του

φτάνοντας έτσι σε μεγάλα προβλήματα με μη αποδεκτό χρόνο επίλυσης του προβλήματος. Μέχρι στιγμής οι καλύτεροι ακριβείς αλγόριθμοι περιορίζονται στο να λύνουν προβλήματα μέχρι 100 πελάτες (Fukasawa et al., 2006). Κλασικές εργασίες επικεντρωμένες στους ακριβείς αλγόριθμους βρίσκουμε από τους: Golden and Assad (1986,1988), Desrochers (1988), Solomon and Desrosies (1988), Desrosies et al. (1995), Cordeau et al. (2001), Larsen (1999), Cook and rich (1999).

Ευρετικοί αλγόριθμοι (heuristics)

Οι ευρετικοί αλγόριθμοι προσφέρουν καλές λύσεις με χαμηλό υπολογιστικό. Για να είναι ένας ευρετικός αλγόριθμος αποτελεσματικός πρέπει η απόκλιση της λύσης που δίνει από την βέλτιστη λύση να μην είναι μεγάλη, η εύρεση της λύσης να επιτυγχάνεται εύκολα και η λογική στην οποία στηρίζονται οι κανόνες που οδηγούν στην λύση να είναι σωστή. Οι ευρετικοί αλγόριθμοι χωρίζονται σε αλγόριθμους κατασκευής δρομολόγησης και σε αλγόριθμους βελτίωσης δρομολόγησης, τέτοιοι είναι οι αλγόριθμοι απληστίας (greedy algorithms), οι προσεγγιστικοί αλγόριθμοι (approximation algorithms) και οι αλγόριθμοι τοπικής αναζήτησης (local search algorithms).

Αξιολόγηση ευρετικών αλγόριθμων

Κριτήρια αξιολόγησης αποτελούν ο υπολογιστικός χρόνος, η ποιότητα της λύσης, η ευκολία της εφαρμογής, η ευελιξία, η αξιοπιστία της λύσης. Η αξιοπιστία της λύσης (robustness) υποδηλώνει ότι ο αλγόριθμος δεν θα πρέπει να είναι υπερβολικά ευαίσθητος σε τροποποιήσεις του προβλήματος και να αποδίδει καλά σε κάθε παράδειγμα και κυρίως να έχει επαναληψιμότητα στα αποτελέσματα που δίνει σε ένα συγκεκριμένο πρόβλημα. Το τελευταίο είναι πολύ σημαντικό γιατί οι ευρετικοί αλγόριθμοι δεν είναι αιτιοκρατικοί (nondeterministic) και κάθε φορά η λύση που προσφέρουν δεν είναι εξ ορισμού η ίδια, κάτι που δημιουργεί δυσκολία στην ανάλυση και επεξεργασία των αποτελεσμάτων. Συνήθως αξιολογούνται μόνο τα καλύτερα αποτελέσματα, ενώ θα έπρεπε να συγκρίνονται οι μέσοι όροι των αποτελεσμάτων και να αναφέρονται και οι χειρότερες επιδόσεις.

Η ποιότητα της λύσης καθορίζεται από την τιμή της αντικειμενικής συνάρτησης και το πόσο κοντά είναι στην βέλτιστη λύση. Η ποιότητα της λύσης είναι ένας συμβιβασμός με τον υπολογιστικό χρόνο. Η απόδοση των ευρετικών αλγόριθμων μπορεί να μετρηθεί με την γραφική παράσταση της ποιότητας της λύσης ως προς τον υπολογιστικό χρόνο. Σε μία τέτοια γραφική παράσταση, τα σημεία πέρα από

οποία δεν υπάρχει καλύτερη λύση και στους 2 άξονες είναι το βέλτιστο Pareto. Γενικά για να εκτιμήσεις την ποιότητα της λύσης πρέπει να δοκιμάσεις τον αλγόριθμο σε πολλά διαφορετικά παραδείγματα, σε διαφορετικές τιμές παραμέτρων και να μετρηθεί η γενικότερη απόδοση.

Σχετικά με το VRPTW ο πλέον συνήθης τρόπος αξιολόγησης ενός αλγορίθμου είναι τα αποτελέσματα που παρήχθησαν από τον Solomon (1987) σε 56 προβλήματα αναφοράς. Αυτά τα προβλήματα έχουν 100 πελάτες, 1 κέντρο διανομής, περιορισμούς χωρητικότητας, χρονικά παράθυρα και περιορισμό στην χρονική διάρκεια των δρομολογίων. Οι κλάσεις C1 και C2 (clustered) έχουν πελάτες χωρισμένους σε clusters, ενώ στις κλάσεις R1 και R2 (random) οι πελάτες έχουν τυχαίες θέσεις, οι κλάσεις RC1 και RC2 έχουν πελάτες σε clusters και σε τυχαίες θέσεις. Κάθε κλάση έχει από 8 εώς 12 διαφορετικές εκδόσεις με διαφορές μόνο στα χρονικά παράθυρα όπου μπορεί 25% ή 50% ή 75% ή 100% των πελατών να έχουν χρονικά παράθυρα. Οι κλάσεις C1, R1 και RC1 μικρές χωρητικότητες και χρόνους εξυπηρέτησης, ενώ οι C2, R2, RC2 έχουν μεγαλύτερες χωρητικότητες και λιγότερα οχήματα, παντού οι αποστάσεις και οι χρόνοι υπολογίζονται από τις Ευκλείδειες αποστάσεις. Τα αποτελέσματα αξιολογούνται με ιεράρχηση των τιμών των αντικειμενικών συναρτήσεων, όπου πρωτευόντως αξιολογούμε το πλήθος των οχημάτων και δευτερευόντως την απόσταση ή το χρόνο. Έχει παρατηρηθεί ότι η μείωση των απαιτούμενων οχημάτων μπορεί να επιφέρει αύξηση στα διανυόμενη απόσταση.

Route construction heuristics

Οι μέθοδοι ακολουθίας κατασκευάζουν ένα δρομολόγιο και μετά το επόμενο, ενώ οι παράλληλες κατασκευάζουν ταυτόχρονα όλα τα δρομολόγια.

Savings heuristic : αρχικά κάθε πελάτης εξυπηρετείται από ξεχωριστό δρομολόγιο και σε κάθε επανάληψη ενώνονται τα δρομολόγια, Clarke and Wright (1964)

savings $S_{ij} = d_{i0} + d_{0j} - d_{ij}$, d_{i0} =*κόστος διαδρομής i-0* d_{0j} = *κόστος διαδρομής i-0* d_{ij} *κόστος διαδρομής i-j*

Time-oriented heuristic : αρχίζει η δρομολόγηση με τον κοντινότερο πελάτη στην αποθήκη και σε κάθε επανάληψη προστίθεται ο επόμενος κοντινότερος πελάτης

I1 heuristic : ένα δρομολόγιο ξεκινάει με ένα πελάτη και οι υπόλοιποι προστίθενται σε αυτό το δρομολόγιο. Ο αρχικός πελάτης για μια νέα διαδρομή μπορεί να καθοριστεί με διάφορα κριτήρια (π.χ. ο πιο απομακρυσμένος από την αποθήκη πελάτης, ο πελάτης με το νωρίτερο χρονικό παράθυρο ή ο πελάτης με τη μεγαλύτερη ζήτηση). Για κάθε υποψήφιο πελάτη εκτός δρομολογίου υπολογίζεται

το κόστος εισαγωγής του στην διαδρομή ως σταθμισμένος μέσος όρος της πρόσθετης απόστασης και του πρόσθετου χρόνου. Στο επόμενο στάδιο επιλέγετε ο πελάτης για τον οποίο η διαφορά ανάμεσα στο κόστος εισαγωγής του σε μία νέα διαδρομή και ανάμεσα στην εισαγωγή του στην υπό κατασκευή διαδρομή είναι η μεγαλύτερη. Ο συγκεκριμένος πελάτης εισάγετε στη διαδρομή και έπειτα και ο υπολογισμός ξαναξεκινάει μέχρι να εξαντληθεί η δυνατότητα του φορτηγού σε χρονικά περιθώρια ή σε περιορισμός χωρητικότητας φορτίου. Νέοι πόροι δημιουργούνται με κάθε νέο δρομολόγιο ή πελάτη. Είναι πολύ σημαντικό να βρεθεί ο κατάλληλος σταθμισμένος μέσος όρος για προβλήματα του πραγματικού κόσμου. Μια καλή αρχική τιμή για την ρύθμιση του αλγόριθμου μπορεί να βρεθεί στην αρχική εργασία του Solomon (1987).

I2 heuristic : επιλέγει τον πελάτη που μειώνει τη συνολική απόσταση και χρόνο

I3 heuristic : επιλέγει τον πελάτη που είναι πιο επείγον να εξυπηρετηθεί.

Solution improvements heuristics

Για το σχεδιασμό ενός αλγόριθμου τοπικού βέλτιστου πρέπει να καθοριστούν τα παρακάτω, δηλαδή πως βρίσκεται η αρχική λύση, με ποιο μηχανισμό γίνεται η αλλαγή του δρομολογίου, το κριτήριο αποδοχής και η δοκιμή τερματισμού. Τα αποτελέσματα εξαρτώνται σε μεγάλο βαθμό από την αρχική λύση και από τον μηχανισμό επιλογής νέων γειτνιάσεων. Οι πιο συνήθεις είναι οι edge-exchange αλγόριθμοι, όπου σε ένα δρομολόγιο αντικαθιστούμε *k* ακμές του με άλλες *k* ακμές και ονομάζονται *k-exchange* και ένα δρομολόγιο που δεν μπορεί να βελτιωθεί άλλο από *k-exchange,* λέγετε *k-optimal*. Για να πιστοποιήσεις την *k-optimality* απαιτείται χρόνος O(n^k).

2-exchange ή 2-opt βελτιώνει ένα δρομολόγιο αλλάζοντας 2 ακμές μεταξύ τους.

2-opt* **heuristic** είναι ο συνδυασμός 2 δρομολογίων έτσι ώστε οι τελευταίοι πελάτες ενός δρομολογίου εισάγονται μετά από τους πρώτους πελάτες ενός άλλου δρομολογίου έτσι ώστε να διατηρείται η φορά της διαδρομής.

CROSS- Exchange : η βασική ιδέα είναι ότι από μία διαδρομή αφαιρούμε 2 ακμές, το ίδιο γίνετε και σε μία δεύτερη διαδρομή. Οι ενδιάμεσες διαδρομές, που περιέχουν ένα αυθαίρετο αριθμό πελατών, ανταλλάζονται με αποτέλεσμα τη δημιουργία 2 νέων διαδρομών διατηρώντας την αρχική φορά των διαδρομών.

Μεταευρετικοί αλγόριθμοι (metaheuristics)

Οι μεταευρετικοί αλγόριθμοι όπως και οι ευρετικοί δίνουν λύσεις που είναι κοντά στην βέλτιστη λύση. Οι μεταευρετικοί όμως είναι αλγόριθμοι υψηλού επιπέδου, αποδοτικότεροι από τους ευρετικούς (Yang, 2010c) και μπορούν να χρησιμοποιηθούν για την επίλυση σχεδόν όλων των προβλημάτων βελτιστοποίησης (Talbi, 2009).

Οι μεταευρετικοί αλγόριθμοι συνήθως μοντελοποιούν κάποιο φαινόμενο που υπάρχει στην φύση και είναι προσαρμοστικοί.

Σε κάθε περίπτωση πάντως, οι καλύτερες συνολικά λύσεις από μεταευρετικούς αλγόριθμους, προκύπτουν από προβλήματα με καλύτερες αρχικές λύσεις και με την βοήθεια των ευρετικών αλγόριθμοι απληστίας που φαίνεται να αποδίδουν καλύτερα από τους αλγόριθμους τοπικής αναζήτησης.

Tabu Search αλγόριθμοι (TS)

Αποτελεί μία μεταευρετική μέθοδο τοπικής – περιορισμένης αναζήτησης που πρωτοπαρουσιάστηκε από τον Glover (1986). Εξερευνά τον χώρο των λύσεων μετακινούμενος από μία λύση *S* σε μία καλύτερη λύση στο υποσύνολο των γειτονικών λύσεων *N(s),* επιτρέπει νέες χειρότερες λύσεις ώστε να μπορέσει να ξεφύγει από τοπικά ελάχιστα και για να αποφύγει την επιστροφή σε προηγούμενες λύσεις που είχαν εξερευνηθεί πρόσφατα τις δηλώνει ως απαγορευμένες ή tabu. Η διάρκεια που οι λύσεις παραμένουν σε κατάσταση tabu μπορεί να αλλάζει ανά χρονικά διαστήματα. Η κατάσταση tabu μπορεί υπό συγκεκριμένες συνθήκες να ξεπερασθεί (κριτήριο φιλοδοξίας – aspiration criterion). Οι περισσότερες προσεγγίσεις με TS χρησιμοποιούν εξειδικευμένες στρατηγικές της εκμετάλλευσης (exploitation) και της εξερεύνησης (exploration) για να κατευθύνουν την αναζήτηση. Η εκμετάλλευση αφορά την αναζήτηση σε μία μικρή περιοχή στην οποία έχει ήδη εντοπιστεί μία καλή λύση. Η εξερεύνηση σχετίζεται με την αναζήτηση σε όλο τον χώρο λύσεων με την βοήθεια της τυχαιότητας η οποία δεν επιτρέπει στον αλγόριθμο να εγκλωβίζεται σε τοπικά βέλτιστα.

Genetic - Γενετικοί αλγόριθμοι (GA)

Η βασική ιδέα αναπτύχθηκε από τον Holland (1975) και η πρώτη εφαρμογή σε πολύπλοκα υπολογιστικά προβλήματα από τους De Jong (1975) και Goldberg (1975). Ο αλγόριθμος εξελίσσει ένα πληθυσμό χρωμοσωμάτων δημιουργώντας νέες γενιές απογόνων μέσα από μία επαναληπτική διαδικασία μέχρι να ικανοποιηθούν κριτήρια σύγκλισης. Τέτοια κριτήρια μπορεί να είναι μέγιστος αριθμός νέων γενεών. Το καλύτερο χρωμόσωμα δίνει την λύση. Τα νέα χρωμοσώματα δημιουργούνται μέσα από 4 φάσεις: Αντιπροσώπευση – η κωδικοποίηση τμήματος μίας λύσης ως χρωμόσωμα, Επιλογή – τυχαία επιλογή δύο γονέων από τον συνολικό πληθυσμό, Αναπαραγωγή- με χρήση των χρωμοσωμάτων δημιουργούνται νέοι απόγονοι, Μετάλλαξη – είναι η τυχαία τροποποίηση ενός γονιδίου και συνήθως γίνεται με μικρή πιθανότητα. Το πλήθος των χρωμοσωμάτων που διατηρούνται ή αντικαθίστανται ορίζεται από την στρατηγική του GA.

Οι Thangiah et al. (1991) ήταν οι πρώτοι που παρουσίασαν εφαρμογή του GA στο VRPTW, όπου ο GA αρχικά δημιουργεί clusters με πελάτες και μετά κάνει την δρομολόγηση ανά cluster με ευρετικούς αλγόριθμους με εισαγωγή του πελάτη με το χαμηλότερο κόστος και στη συνέχεια εφαρμόζει *λ-exchanges* για βελτίωση της ποιότητας της λύσης. Τα τελευταία χρόνια έχουν δημιουργηθεί εφαρμογές που συνδυάζουν GA με ευρετικούς αλγόριθμους κατασκευής και βελτίωσης, αλλά και με άλλους μεταευρετικούς αλγόριθμους όπως Tabu Search, δημιουργώντας ένα υβριδικό σύστημα.

Simulated Annealing (SA) - προσομοιωμένη ανόπτηση

Οι Chiang and Russel (1996) χρησιμοποιούν μία προσέγγιση της λύσης του VRPTW με προσομοιωμένη ανόπτηση. Ο SA είναι μία στοχαστική τεχνική χαλάρωσης που προέρχεται από τη στατιστική μηχανική. Η λύση S' γίνετε αποδεκτή σαν νέα λύση αν $\Delta \leq 0$, όπου $\Delta = C(S') - C(S)$. Για μπορέσει να ξεφύγει από το τοπικό ελάχιστο, επιτρέπει την αύξηση (χειροτέρευση) της αντικειμενικής συνάρτησης με μία πιθανότητα ίση με $e^{-\Delta/T}$ αν $\Delta > 0$, όπου T είναι η παράμετρος αντίστοιχη της θερμοκρασίας η οποία ενώ ξεκινάει από μία σχετικά υψηλή τιμή σε μία θερμοκρασία κοντά στο μηδέν. Χαρακτηριστικό είναι ότι σε πολύ υψηλές θερμοκρασίες κάθε πιθανή κατάσταση (λύση) έχει ίδιες σχεδόν πιθανότητες να επιλεγεί, ενώ σε χαμηλές θερμοκρασίες μόνο οι καταστάσεις με χαμηλή ενέργεια έχουν υψηλή πιθανότητα να επιλεγούν. Αν μία νέα κατάσταση έχει υψηλότερη ενέργεια κατά *d* Joules, η πιθανότητα να επιλεχθεί είναι ίση με $\exp(-\frac{d}{kT})$, όπου *k* είναι η σταθερά *Boltzmann* και *T* η απόλυτη θερμοκρασία. Αυτή η διαδικασία ονομάζεται βρόχος *metropolis*. Η

μείωση της θερμοκρασίας προσδιορίζεται από ένα πρόγραμμα ψύξης το οποίο ορίζει τον ρυθμό της πτώσης σε κάθε στάδιο του αλγόριθμου, όπως πρωτοπαρουσιάστηκε από Metropolis (1953). Οι Tanghia et al. (1994) ανέπτυξαν ένα αλγόριθμο δύο φάσεων, στην πρώτη φάση γίνετε χρήση ευρετικού αλγόριθμου κατασκευής με βάση την εισαγωγή του πελάτη με το μικρότερο κόστος και στη δεύτερη φάση χρήση λ-interchange και αξιολόγηση με SA ή με συνδυασμό SA και Tabu Search. Τα πλεονεκτήματα του SA συνοψίζονται σε: α. Μπορεί να συνεργαστεί με αυθαίρετα συστήματα και με συναρτήσεις κόστους β. Στατιστικά μπορεί να εντοπίσει τη βέλτιστη λύση γ. Είναι σχετικά εύκολη η κωδικοποίησή του ακόμη και για πολύπλοκα προβλήματα δ. Δίνει γενικά «καλές» λύσεις. Μειονεκτήματα του SA είναι: α. ένα πρόγραμμα ψύξης τύπου 1/log *k* είναι πολύ αργό ειδικά όταν είναι χρονοβόρος ο υπολογισμός της λύσης β. σε προβλήματα όπου υπάρχουν λίγα τοπικά βέλτιστα ο SA δεν αποτελεί ιδανική επιλογή επειδή αποδεικνύεται χρονοβόρος γ. οι ευρετικές μέθοδοι ειδικά σχεδιασμένες μπορεί να είναι καλύτερες από γενικές μεθόδους.

Ant Colony optimization (ACO)

Ο αλγόριθμος βασίζεται στην επικοινωνία και συνεργασία των μυρμηγκιών ώστε να μπορούν να βρίσκουν μονοπάτια από την φωλιά τους για τις πηγές τροφής. Η επικοινωνία γίνετε με φερομόνες τις οποίες εναποθέτουν στο μονοπάτι. Ένα αυτόνομο μυρμήγκι γενικά κινείται τυχαία, αλλά με την παρουσία φερομόνης ακολουθεί το μονοπάτι στο οποίο ενισχύει την ποσότητα φερομόνης. Έτσι η πιθανότητα να ακολουθηθεί το μονοπάτι αυξάνει ανάλογα με το πλήθος τον μυρμηγκιών που το ακολουθούν. Η φερομόνη στον αλγόριθμο αντιπροσωπεύει την μνήμη του συστήματος σε μονοπάτια που δημιουργήθηκαν από καλές προηγούμενες λύσεις. Το κάθε μυρμήγκι διαλέγει το επόμενο κομμάτι της λύσης βασιζόμενο σε μία ευρετική αξιολόγησή του και στην ποσότητα φερομόνης αντιπροσωπευόμενη από ένα συντελεστή συσχετιζόμενο με αυτή. Οι Gambarella et al. (1999) χρησιμοποίησαν δύο συνεργατικές αποικίες μυρμηγκιών. Η πρώτη έχει στόχο να ελαχιστοποίηση το πλήθος οχημάτων ενώ το δεύτερο ελαχιστοποιεί την συνολική απόσταση.

Δύο μετρήσεις συσχετίζονται με την κάθε ακμή, η ελκυστικότητα N_{ij} και η φερομόνη στο μονοπάτι T_{ij}. Τα μονοπάτια κατασκευάζονται με τη χρήση του αλγόριθμου κοντινότερου γείτονα με πιθανοθεωρητικούς κανόνες. Κατά την αναζήτηση τα ίχνη φερομόνης ενημερώνονται τοπικά και συνολικά. Για να βελτιωθεί η δημιουργία τυχαίων αρχικών λύσεων γίνετε χρήση των MLS (multistart local search), και ILS (iterated local search).

Γλώσσες Προγραμματισμού και αλγόριθμοι.

Όπως αναφέραμε παραπάνω για την σύγκριση της απόδοσης των αλγόριθμων βασική παράμετρος είναι ο υπολογιστικός χρόνος και για να είναι συγκρίσιμος θα πρέπει να ληφθεί υπόψη και η γλώσσα προγραμματισμού. Οι περισσότεροι αλγόριθμοι έχουν σχεδιαστεί σε γλώσσα προγραμματισμού C, αλλά βρίσκουμε προτάσεις και σε Java ή Fortran. Η Java υπολογίζεται ότι είναι 5 φορές αργότερη από την C, ενώ δεν υπάρχουν ουσιαστικές διαφορές ανάμεσα σε C και Fortran. Το Matlab είναι 3 φορές πιο αργό από τη Java. Εφαρμογές υπάρχουν και σε VBA με τις αναφορές να ισχυρίζονται ότι είναι τουλάχιστον 3 φορές πιο αργή από την C, που αποτελεί βέβαια το τίμημα της ευελιξίας έναντι της απόδοσης.

VRP μια σύνοψη

Το πρόβλημα του περιοδεύοντος πωλητή Traveling Salesman Problem (TSP) είναι ένα πρόβλημα όπου μόνο ένα όχημα – ο πωλητής πρέπει να επισκεφθεί όλους τους πελάτες. Πιο πρακτικές εφαρμογές δυστυχώς περιλαμβάνουν επιπλέον περιορισμούς όπως η ζήτηση του πελάτη να ξεπερνάει την χωρητικότητα του οχήματος ή ότι υπάρχει η δυνατότητα να μειωθεί η συνολική διανυόμενη απόσταση με την χρήση άλλου ενός επιπλέον οχήματος. Με αυτόν τον τρόπο εισήχθηκε η έννοια του VRP από τους Dantzig and Ramser (1959) σαν μια επέκταση του TSP. Πολλές εκδόσεις του VRP έχουν προταθεί τα τελευταία χρόνια για να συμπεριλάβουν επιπλέον πτυχές στο πρόβλημα, για να συμπεριλάβουν χαρακτηριστικά του δικτύου διανομής, της ζήτησης του πελάτη, του στόλου, του κόστους ή του πλήθους των αντικειμενικών συναρτήσεων κάτι που κάνει ακόμη πιο δύσκολη την επίλυση του προβλήματος.

CVRPTW - Capacitated Vehicle Routing Problem with Time Windows

Η εκδοχή του VRP, το Capacitated Vehicle Routing Problem with (hard) Time Windows (CVRPTW or VRPTW) είναι μια πολύ σημαντική εκδοχή του VRP που αποτελείται από την δρομολόγηση ενός ομογενή στόλου οχημάτων με ίδια ακριβώς χωρητικότητα και έδρα ένα μοναδικό κέντρο διανομής το οποίο λειτουργεί μέσα σε συγκεκριμένα χρονικά πλαίσια. Τα οχήματα χρησιμοποιούνται για να

επισκέπτονται και να εξυπηρετούν τις ζητήσεις ενός πλήθους πελατών κατανεμημένων γεωγραφικά στο χώρο και κάθε δρομολόγιο αρχίζει και τελειώνει στο κέντρο διανομής και δεν εξυπηρετεί συνολική ζήτηση μεγαλύτερη από την δυναμικότητα του κάθε φορτηγού. Οι πελάτες μπορούν να εξυπηρετηθούν από μόνο ένα φορτηγό και μόνο μία φορά, έχουν προκαθορισμένες ζητήσεις προϊόντων και ώρες εξυπηρέτησης. Η απόσταση ανάμεσα στους πελάτες υπολογίζετε και το σύνολό της καθορίζει τον χρόνο εξυπηρέτησης. Κάθε πελάτης έχει ένα καθορισμένο χρονικό παράθυρο το οποίο καθορίζει πότε είναι ο νωρίτερος χρόνος εξυπηρέτησης και πότε το αργότερο χρονικό σημείο στο οποίο πρέπει να αρχίσει η εξυπηρέτησή του. Έτσι το πρόβλημα βελτιστοποίησης ανάγεται στον καθορισμό της σειράς εξυπηρέτησης όλων των πελατών, με στόχο την ελαχιστοποίηση της διανυόμενης απόστασης και την ικανοποίηση των περιορισμών δυναμικότητας των φορτηγών και των χρονικών παραθύρων.

Το VRPTW έχει αποτελέσει το επίκεντρο εκτεταμένων μελετών γιατί αποτελεί ένα από τα πιο δύσκολα προβλήματα επίλυσης και βελτιστοποίησης με σημαντικές οικονομικές επιπτώσεις στις διαδικασίες των logistics, ιδιαίτερα λόγω της σημασίας που έχουν στα επίκαιρα συστήματα παραγωγής just-in-time και στον αυξημένο απαιτούμενο συντονισμό των διεργασιών της εφοδιαστικής αλυσίδας. Τα logistics και ειδικότερα η διανομή προϊόντων είναι πολύ συχνά στο επίκεντρο ακαδημαϊκών ερευνών και αποτελούν βασικό πεδίο για τις επιχειρήσεις γιατί σχετίζονται με το ύψος των αποθεμάτων, τις αποφάσεις των εργοστασίων παραγωγής και το κόστος διανομής το οποίο αποτελεί ένα σημαντικό ποσοστό του συνολικού κόστους logistics. Έχουν προταθεί ακριβείς αλγόριθμοι για την επίλυση του προβλήματος όπως οι τεχνικές χαλάρωσης Langrange, οι μέθοδοι column ή delayed column generation και ο δυναμικός προγραμματισμός.

Σε μεσαία και μεγάλα προβλήματα VRP οι ακριβείς μέθοδοι παράγουν μη ικανοποιητικά αποτελέσματα και ιδανικές τεχνικές είναι οι ευρετικές και οι μεταευρετικές τεχνικές όπως οι γενετικοί αλγόριθμοι, tabu search, μιμητικοί αλγόριθμοι, αλγόριθμοι συμπεριφοράς σμήνους κ.α. με πολύ καλές λύσεις σε σύντομους χρόνους.

Multi objectives VRPTW

Ακόμη πιο κοντά στα πρακτικά προβλήματα δρομολόγησης βρίσκονται οι διατυπώσεις του VRP που συμπεριλαμβάνουν πολλαπλές αντικειμενικές συναρτήσεις που πρέπει να βελτιστοποιηθούν ταυτόχρονα.

Τα πιο μελετημένα τέτοιου είδους προβλήματα είναι αυτά όπου έχουμε ιεραρχική προσέγγιση στη μείωση αρχικά των χρησιμοποιούμενων οχημάτων και ύστερα στη μείωση των συνολικών διανυθέντων αποστάσεων, όπου ο αλγόριθμος βρίσκει - μειώνει τον αριθμό των χρησιμοποιούμενων οχημάτων και στη συνέχεια βελτιστοποιεί την διανυθείσα απόσταση. Υπάρχει όμως και η αντίθετη ιεραρχική αντιμετώπιση, όμως αυτές οι δυο αντικειμενικές συναρτήσεις είναι αλληλοσυνδεόμενες.

Τέλος υπάρχουν και άλλες προσεγγίσεις με αντικειμενικές συναρτήσεις που προσπαθούν να μειώσουν το συνολικό κόστος υπό την έννοια της απόστασης ή του χρόνου, την μείωση του μεγαλύτερου δρομολογίου ή την μείωση της ανομοιογένειας των δρομολογίων. Το καλό είναι ότι ο υπεύθυνος μπορεί να ρυθμίσει το συσχετισμό – την βαρύτητα των αντικειμενικών συναρτήσεων και να προσαρμόσει τον μεταξύ τους συμβιβασμό ώστε να προσαρμόσει τις λύσεις στα δικά του κριτήρια. Ιδιαίτερο ενδιαφέρον παρουσιάζει και η μορφοποίηση του προβλήματος που εξασφαλίζει ισορροπία ανάμεσα στα δρομολόγια με την έννοια των διανυόμενων αποστάσεων ή την έννοια του μεταφερόμενου φορτίου.

VRPTW μοντελοποίηση.

Ο μαθηματικός ορισμός των αντικειμενικών συναρτήσεων και των περιορισμών του VRPTW , συμπεριλαμβανομένου των συχνότερα χρησιμοποιούμενων συμβολισμών όπως η διαδρομή, το κέντρο διανομής, του πελάτη, των οχημάτων μπορεί να μοντελοποιηθεί ως πρόβλημα της θεωρίας των γράφων.

G = (V, E), όπου *G* είναι ο γράφος, ένα διατεταγμένο ζεύγος

V = {1,……,N}, όπου *V* το σύνολο των κορυφών

$e \in E\{(i,j): ij \in V\}$, όπου *E* το σύνολο των ακμών

Παράμετροι:

α_j *= το νωρίτερο που μπορεί να εξυπηρετηθεί ο πελάτης j*

b_j *= το αργότερο που μπορεί να εξυπηρετηθεί ο πελάτης j*

C_{ij} *= το κόστος μεταφοράς από τον κόμβο i στον κόμβο j (σε απόσταση ή χρόνο)*

d_j *= η ζήτηση του πελάτη j*

K = το μέγιστο πλήθος οχημάτων που μπορούν να χρησιμοποιηθούν

N = το πλήθος των πελατών(=2,…,N) μαζί με το κέντρο διανομής(=1)

Q^k *= είναι η χωρητικότητα του οχήματος k*

Η μαθηματική μοντελοποίηση του προβλήματος του VRP έχει ως εξής:

Μεταβλητές:

$$X_{ij}^k = \begin{cases} 1 \ \alpha\nu \ \tau o \ \acute{o}\chi\eta\mu\alpha \ \ k \ \kappa\acute{a}\nu\epsilon\iota \ \tau\eta\nu \ \delta\iota\alpha\delta\rho o\mu\acute{\eta} \ \alpha\pi\acute{o} \ \tau o\nu \ \kappa\acute{o}\mu\beta o \ i \ \sigma\tau o\nu \ \kappa\acute{o}\mu\beta o \ j \\ 0 \ \ \sigma\varepsilon \ \kappa\acute{a}\theta\varepsilon \ \acute{a}\lambda\lambda\eta \ \pi\varepsilon\rho\acute{\iota}\pi\tau\omega\sigma\eta \end{cases}$$

Αντικειμενική συνάρτηση:

$$\varepsilon\lambda\alpha\chi\iota\sigma\tau o\pi o\acute{\iota}\eta\sigma\eta \ \tau\eta\varsigma \quad TD = \sum_{k=1}^{K}\sum_{i=1}^{N}\sum_{j=1}^{N} CijX_{ij}^k \qquad (1)$$

$$X_{ii}^k = 0 \qquad (\forall i \ \in \{1, \dots . , N\}, \forall k \ \in \{1, \dots . , K\}) \qquad (2)$$

$$X_{ij}^k \ \in \{1,0\} \qquad (\forall i, j \ \in \{1, \dots . , N\}, \forall k \ \in \{1, \dots . , K\}) \qquad (3)$$

$$\sum_{k=1}^{K}\sum_{i=1}^{N} X_{ij}^k = 1 \qquad (\forall j \ \in \{2, \dots . , N\}) \qquad (4)$$

$$\sum_{i=1}^{N}\sum_{j=2}^{N} X_{ij}^k \, d_j \ \leq Q^k \quad (\forall k \ \in \{1, \dots . , K\}) \qquad (5)$$

$$\sum_{k=1}^{K}\sum_{j=2}^{N} X_{1j}^k \ \leq K \qquad (6)$$

$$\sum_{j=2}^{N} X_{1j}^k \ - \sum_{j=2}^{N} X_{j1}^k = 0 \qquad (\forall k \ \in \{1, \dots . , K\}) \qquad (7)$$

$$\alpha_j \leq S_{kj} \ \leq b_j \qquad (\forall i, j \ \in \{1, \dots . , N\}, \forall k \ \in \{1, \dots . , K\}) \qquad (8)$$

$$S_{ki} \ + \ C_{ij} - L \ (\ 1 - \ X_{ij}^k) \leq S_{kj} \qquad (\forall i, j \ \in \{1, \dots . , N\}, \forall k \ \in \{1, \dots . , K\}) \qquad (9)$$

(1) Η Αντικειμενική Συνάρτηση του προβλήματος

(2) Δηλώνει ότι ένα όχημα πρέπει να κινείται από ένα κόμβο σε έναν άλλο

(3) Αν το X_{ij}^{k} =1 το όχημα k επισκέπτεται τον κόμβο i και μετά τον j

(4) Δηλώνει ότι ένας πελάτης μπορεί να επισκεφθεί μόνο από ένα όχημα

(5) Ο περιορισμός ότι το όχημα k δεν μπορεί να μεταφέρει συνολικό φορτίο μεγαλύτερο από Q^k σε ένα δρομολόγιο

(6) Δηλώνει ότι υπάρχουν K δρομολόγια παράδοσης με εκκίνηση το depot

(7) Δηλώνει ότι κάθε όχημα ξεκινάει και επιστρέφει στο depot

(8) S_{kj} είναι το άθροισμα των αποστάσεων που διανύει το όχημα πριν αφιχθεί στον πελάτη j. Ο περιορισμός εξασφαλίζει την τήρηση των χρονικών παράθυρων

(9) L είναι ένας μεγάλος αριθμός που εξασφαλίζει ότι το όχημα k αν κινείτε από τον πελάτη i στον j, δεν μπορεί να αφιχθεί πριν από $S_{ki} + C_{ij}$, το S_{kj} είναι ή χρονική στιγμή που το όχημα k ξεκινάει να εξυπηρετεί τον πελάτη j.

Το πρόβλημα CVRPTW

Τα δεδομένα που χρησιμοποιούνται στην παρούσα μελέτη είναι πραγματικά και έχουν συλλεγεί από εταιρεία που δραστηριοποιείτε στο χώρο της διανομής τροφίμων. Τα διαθέσιμα στοιχεία αφορούν το πρόγραμμα διανομής μίας ολόκληρης χρονιάς ενώ έγινε ανάλυση των δεδομένων για να βρεθεί το μέσο φορτίο ανά φορτηγό ανά ημέρα, το μέσο πλήθος σημείων παράδοσης καθώς και τα μέγιστα καi τα ελάχιστα. Επιλέχθηκε από όλες της ημέρες διανομής μία τυπική ημέρα κατά την οποία η ζήτηση των πελατών να είναι κοντά στην μέση ζήτηση, αποφεύγοντας έτσι ακραία γεμάτα φορτηγά ή αντίστοιχα ακραία άδεια φορτηγά. Τα διαθέσιμα φορτηγά διανομής είναι έξι (6) και είναι όλα ιδίου τύπου με μέγιστη χωρητικότητα 3000 μονάδων. Το κέντρο διανομής είναι ένα και τα φορτηγά πρέπει να επιστρέφουν στο κέντρο διανομής μετά το τέλος του ημερήσιου έργου τους. Το κάθε φορτηγό πρέπει να κάνει μόνο ένα δρομολόγιο ανά ημέρα, δηλαδή δεν μπορεί να επιστρέφει και να ξαναφορτώνει προϊόντα προς τους πελάτες και φυσικά πρέπει να χρησιμοποιηθεί δεδομένου ότι η εταιρεία απασχολεί 6 οδηγούς. Ο χρόνος εργασίας κάθε φορτηγού είναι 10 ώρες, από 7:00 έως 15:00 και οι 103 πελάτες από τους 113 δεν έχουν χρονικά παράθυρα ή έχουν χρονικά παράθυρα 8 ωρών ενώ 10 πελάτες έχουν 2ωρα χρονικά παράθυρα. Από την ανάλυση των στοιχείων προκύπτει ότι περίπου το 10% των πελατών έχει χρονικά παράθυρα εξυπηρέτησης.

Οπότε συνοψίζοντας σχετικά με την φύση της ζήτησης:

Έχουμε μόνο παραδόσεις, χωρίς επιστροφές, μόνο ένα είδος προϊόντος και είναι γνωστή η ζήτηση πριν την δρομολόγηση.

Σχετικά με τον στόλο των οχημάτων:

Είναι ομογενής με περιορισμό βάρους-μονάδων.

Σχετικά με τα χρονικά παράθυρα:

Τύπου hard.

Απαιτούμενα στοιχεία για την δόμηση του προβλήματος

1. Διευθύνσεις πελατών και κέντρου διανομής
2. Γεωγραφικές θέσεις πελατών και κέντρου διανομής
3. Αποστάσεις ανάμεσα σε όλα τα σημεία παράδοσης και την αποθήκη
4. Χρόνος απαιτούμενος για την διάνυση όλων των αποστάσεων
5. Πλήθος φορτηγών
6. Χωρητικότητα φορτηγών
7. Ωράριο εργασίας αποθήκης
8. Ωράριο εργασίας οδηγών
9. Επιτρεπόμενος χρόνος οδήγησης
10. Χρονικά παράθυρα πελατών
11. Απαιτούμενος χρόνος για την παράδοση της παραγγελίας ανά πελάτη
12. Ζήτηση ανά σημείο παράδοσης

Γεωκωδικοποίηση Δεδομένων για VRP

Γεωκωδικοποίηση Geocoding είναι η διαδικασία της μετατροπής μίας περιγραφής μίας διεύθυνσης, δηλαδή από διεύθυνση (οδός, Ταχ. Κωδ., κτ.λ.) σε γεωγραφικές συντεταγμένες και το αντίστροφο. Υπάρχουν πολλές δυνατότητες την σημερινή εποχή σε αυτό το πεδίο, μέσω Διεπαφών Προγραμματισμού Εφαρμογών ή APIs (Application Programming Interfaces) είναι δηλαδή η διεπαφή προγραμματιστικών διαδικασιών που παρέχει ένα λειτουργικό σύστημα, βιβλιοθήκη ή εφαρμογή προκειμένου να επιτρέπει να γίνονται προς αυτά αιτήσεις από άλλα προγράμματα και ανταλλαγή δεδομένων. Ένας από τους βασικούς σκοπούς μίας διεπαφής είναι να ορίζει και να διατυπώνει το

σύνολο των λειτουργιών - υπηρεσιών που μπορεί να παρέχει μια βιβλιοθήκη ή ένα λειτουργικό σύστημα σε άλλα προγράμματα, χωρίς να επιτρέπει πρόσβαση στον κώδικα που υλοποιεί αυτές τις υπηρεσίες. Η διεπαφή, ένα «συμβόλαιο κλήσης» μεταξύ καλούντος και καλούμενου, διαχωρίζει την προγραμματιστική υλοποίηση κάποιων υπηρεσιών από τη χρήση τους. Στο πεδίο της δρομολόγησης τα APIs που χρησιμοποιούμε ανήκουν στην κατηγορία mapping APIs, μιλάμε για υπηρεσίες όπως Google Geocoding API, Cloudcode Geocoding API, Bing Maps API, MapQuest API κ.τ.λ.. Η κάθε εφαρμογή διαφέρει ως προς την ταχύτητα, την αξιοπιστία των αποτελεσμάτων, τα όρια των αναζητήσεων ανά ημέρα και το κόστος χρήσης ανάλογα αν είναι να γίνει χρήση χαρτών εμπορικών ή freeware. Επομένως, για την εύρεση δεδομένων χιλιομετρικών αποστάσεων (km) ή χρονικών αποστάσεων (min) πρέπει να αντληθούν πληροφορίες από μία βάση δεδομένων ενός Σύστηματος Γεωγραφικής Πληροφόρησης (GIS) κάτι που απαιτεί επένδυση αν οι απαιτήσεις σε ταχύτητα και όγκο δεδομένων είναι αυξημένες.

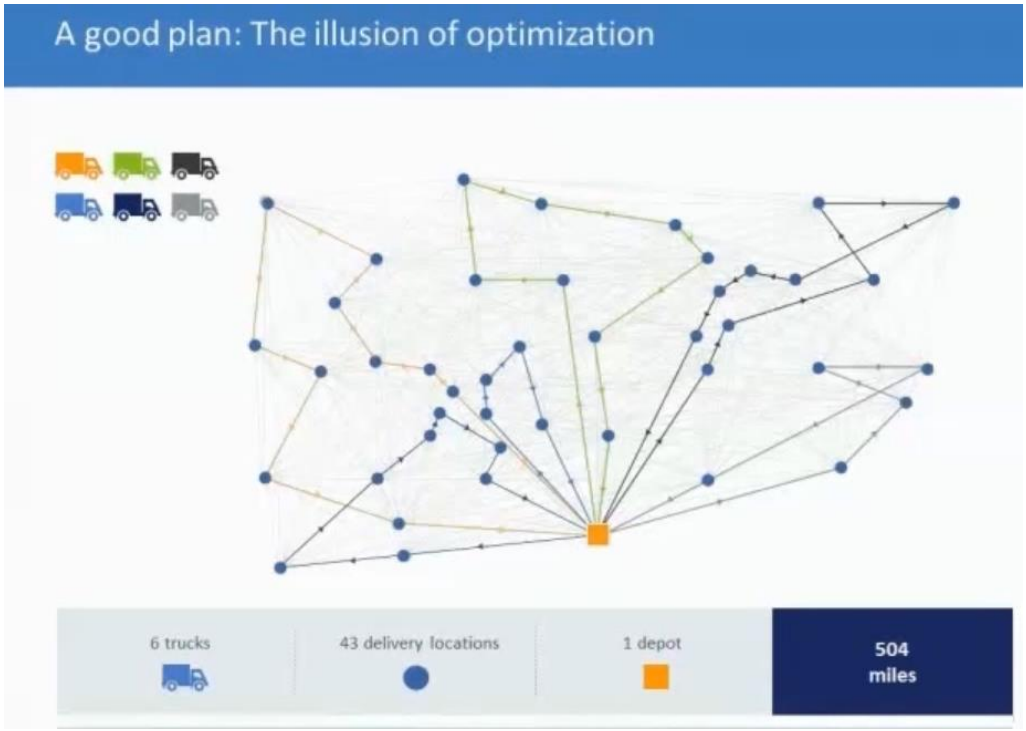
Τα αποτελέσματα ενός αλγόριθμου συνήθως είναι αριθμητικές τιμές, όπως το CNV – Cumulative Number of Vehicles ή το CTD – Cumulative Total Distance. Όμως για έναν υπεύθυνο σχεδιασμού του πλάνου διανομής, οι παραπάνω δείκτες αποτελούν απλώς ένα σύνολο και τίποτα παραπάνω. Ένας πραγματικός χρήστης θέλει εύκολα να καταλάβει το κόστος της επιχείρησης, να αλλάζει τα δεδομένα και να οπτικοποιεί τις λύσεις. Για την λύση είναι απαραίτητη η χρήση τουλάχιστον 3 υπηρεσιών των δημόσιων GIS : 1. Geocoding 2. Directions, που δίνει την απόσταση, τον χρόνο οδήγησης και οδηγίες ανάμεσα σε 2 σημεία 3. Static Maps, που δίνει χάρτες πολλών σημείων που καθορίζονται από το κεντρικό σημείο και την κλίμακα.

Τα αποτελέσματα, μίας τέτοιας εφαρμογής, μπορεί να είναι μέχρι και 20% καλύτερα σε σχέση με την δρομολόγηση χωρίς τη χρήση λογισμικού.

Εταιρίες προσφέρουν πλατφόρμες που ενοποιούν το routing με τα συστήματα ERP, ακόμη και με την διαχείριση αποθεμάτων, το πρόγραμμα παραγωγής και την διαχείριση ανθρώπινου δυναμικού προσφέροντας μία συνολική λύση βελτιστοποίησης.

Η βελτιστοποίηση ενός στόλου 100 φορτηγών, 1000 παραδόσεων και 1 depot μπορεί να δώσει βελτίωση κατά 4800km, 10% λιγότερα φορτηγά, 7% λιγότερα km και εξοικονόμηση 1M € ετησίως με αντίστοιχα οφέλη και για το περιβάλλον ή την επιβάρυνση της κυκλοφορίας.

Παρακάτω βλέπουμε ένα παράδειγμα αντίστοιχο με το πρόβλημα της εργασίας (6 φορτηγά), μια καλή λύση, χωρίς τη χρήση βελτιστοποίησης, βασιζόμενη στη λογική του clustering των πελατών ανά φορτηγό.



Όμως η νέα λύση με την χρήση αλγόριθμου προσφέρει 8% βελτίωση του CTD και 33% βελτίωση του CNV, χωρίς να δείχνει βέλτιστη.

Ψευδοκώδικας Αλγόριθμου

Παρακάτω ακολουθεί ο ψευδοκώδικας του αλγόριθμου που δημιουργήθηκε σε GNU Octave 4.0.3 και έτρεξε σε υπολογιστή με επεξεργαστή i7-6500U @ 2.50GHz και 4GB Ram.

Τα βασικά τμήματα του αλγόριθμου παρουσιάζονται στο παράρτημα μαζί με τα αντίστοιχα σχόλια.

Η βασική ιδέα είναι ότι η δρομολόγηση μπορεί να απεικονίζεται σε ένα πίνακα της παρακάτω μορφής:

	Cust(1)	Cuct(2)	Cuct(…)	Cuct(113)	
Veh(1)	1	0	0	0	Sum(veh(1)<113)
Veh(…)	0	1	0	0	Sum(veh(…)<113)
Veh(6)	0	0	1	1	Sum(veh(6)<113)
	0<=Sum(cust(1))<=1	0<=Sum(cust(1))<=1	0<=Sum(cust(1))<=1	0<=Sum(cust(1))<=1	

Κάθε νέος πελάτης αναθέτετε τυχαία σε κάθε φορτηγό και πρέπει να ισχύουν οι περιορισμοί του φορτίου και των χρονικών παραθύρων.

Τότε δημιουργούμε ένα πίνακα tab2 που απεικονίζει την σειρά επίσκεψης πελατών ανά φορτηγό

	1	2	3	4	5	6	7	8	9	10
Veh(1)	1	10	15	23	29	100	11	112	-	-
Veh(…)										
Veh(6)	113	24	18	55	3	67	-	-	-	-

ένα αντίστοιχο πίνακα sk που απεικονίζει την σειρά επίσκεψης πελατών ανά φορτηγό και τον χρόνο (min) που μπορεί το φορτηγό να φύγει από τον κάθε πελάτη

	1	2	3	4	5	6	7	8	9	10
Veh(1)	1 (480)	10 (490)	15 (510)	23 (550)	29 (590)	100(620)	11 (660)	112(700)	-	-
Veh(…)										
Veh(6)	113(480)	24 (580)	18 (620)	55 (660)	3 (699)	67 (888)	-	-	-	-

ένα αντίστοιχο πίνακα ticket που απεικονίζει την σειρά επίσκεψης πελατών ανά φορτηγό και το συσσωρευτικό φορτίο (π.χ. kg) που έχει εξυπηρετήσει το φορτηγό όταν φεύγει από τον κάθε πελάτη

	1	2	3	4	5	6	7	8	9	10
Veh(1)	1 (50)	10 (80)	15 (110)	23 (150)	29 (190)	100(220)	11 (360)	112(700)	-	-
Veh(…)										
Veh(6)	113(80)	24 (180)	18 (220)	55 (260)	3 (399)	67 (588)	-	-	-	-

1. Πλήθος Φορτηγών = 6
2. Περιορισμός βάρους φορτηγών = 3000
3. Κέντρο διανομής = 1 (114)
4. Πλήθος πελατών = 113
5. Συντεταγμένες πελατών
6. Δημιουργία πίνακα αποστάσεων (km) ανάμεσα σε όλους τους κόμβους.
7. Ταχύτητα κίνησης φορτηγού = 40 km/h
8. Δημιουργία πίνακα χρονικών αποστάσεων (min) ανάμεσα σε όλους τους κόμβους.
9. Δημιουργία πίνακα ζήτησης ανά πελάτη
10. Δημιουργία πίνακα χρονικών παραθύρων πελατών = 10 X 2ωρα + 103 X 8ωρα τύπου hard
11. Χρόνος εργασίας φορτηγών = 10 h
12. Χρόνος εξυπηρέτησης ανά πελάτη = 15 min
13. Ορισμός αρχικού και τελευταίου κόμβου = το κέντρο διανομής (114)
14. Δημιουργία αρχικής λύσης
15. Υπολογισμός πλήθους πελατών ενταγμένων στον πίνακα λύσης.
16. Αν πλήθος πελατών = 113, τότε πήγαινε στο 27
17. Μετρητής κύκλων = Μετρητής + 1
18. Εισαγωγή τυχαία ενός φορτηγού [1-6]
19. Εισαγωγή τυχαία ενός πελάτη [1-113]
20. Υπολογισμός αντικειμενικής συνάρτησης σε (min) = Χρόνος εκκίνησης από προηγούμενο κόμβο (min) + Χρόνος διαδρομής (min) + Χρόνος εξυπηρέτησης ανά πελάτη (min)
21. Υπολογισμός συνολικού φορτίου ανά φορτηγό = Αρχικό φορτίο φορτηγού + ζήτηση νέου πελάτη
22. Έλεγχος περιορισμού βάρους
23. Έλεγχος χρονικού παράθυρου του πελάτη
24. Ικανοποίηση περιορισμών;
25. Ναι, τότε ένταξε το νέο πελάτη στον πίνακα της λύσης και πήγαινε στο 15, αν ο μετρητής είναι <1000
26. Όχι, τότε επιστροφή στην προηγούμενη λύση και πήγαινε στο 15.
27. Αποθήκευση πίνακα Αρχικής Λύσης με Obj(0)
28. Βελτιστοποίηση Αρχικής Λύσης Simulated Annealing
29. Μετρητής Kiter=0, Αρχική θερμοκρασία T=100
30. Ανάγνωση kiter και T

31. Αν T>10, τότε συνέχισε αλλιώς πήγαινε στάδιο 39
32. αν kiter = 10 τότε νέα θερμοκρασία T=T/log(4.5) και kiter = 0
33. Μετρητής kiter = kiter + 1
34. Ξαναξεκίνα εύρεση λύσης (στάδιο 14)
35. Σύγκριση τιμής αντικειμενικής συνάρτησης νέας λύσης Obj(1)
36. Αν Obj(1)<obj(0), τότε υιοθέτησε νέα λύση
37. Αλλιώς υπολόγισε τυχαίο αριθμό p0 από [0,1] και αν p0<exp(-(obj1-obj0)/T), τότε υιοθέτησε τη νέα λύση.
38. Πήγαινε στάδιο 30
39. Αποθήκευση πίνακα Βελτιστοποιημένης Λύσης με Obj(0)
40. <u>Βελτιστοποίηση Λύσης με Simulated Annealing ανά φορτηγό</u>
41. Αρχικός Πίνακας Φορτηγών Φ=[1-6]
42. Επιλογή τυχαία φορτηγού X=[1-6]
43. Δημιουργία νέου πίνακα φορτηγών = Φ - X
44. Αν Φ=0 τότε πήγαινε στάδιο 62, αλλιώς 45 - συνέχισε
45. Για φορτηγό X ανάγνωση πίνακα-γραμμής λύσης από στάδιο 39
46. Τυχαία επιλογή 2 πελατών από την λύση
47. Αντιμετάθεση των πελατών στη σειρά εξυπηρέτησης
48. Υπολογισμός χρόνων άφιξης και έλεγχος τήρησης χρονικών παραθύρων
49. Αν ικανοποιεί περιορισμό υπολογισμός ΑΣ
50. Μετρητής Kiter=0, Αρχική θερμοκρασία T=100
51. Ανάγνωση kiter και T
52. Αν T>2, τότε συνέχισε αλλιώς πήγαινε στάδιο 57
53. αν kiter = 100 τότε νέα θερμοκρασία T=T/log(4.5) και kiter = 0
54. Μετρητής kiter = kiter + 1
55. Ξαναξεκίνα εύρεση λύσης (στάδιο 41)
56. Σύγκριση τιμής αντικειμενικής συνάρτησης νέας λύσης Obj(1)
57. Αν Obj(1)<obj(0), τότε υιοθέτησε νέα λύση
58. Αλλιώς υπολόγισε τυχαίο αριθμό p0 από [0,1] και αν p0<exp(-(obj1-obj0)/T), τότε υιοθέτησε τη νέα λύση.
59. Πήγαινε στάδιο 30
60. Αποθήκευση πίνακα Βελτιστοποιημένης Λύσης Φορτηγού X
61. Επιστροφή στάδιο 42
62. Αποθήκευση νέου πίνακα Βελτιστοποιημένης Λύσης

Αποτελέσματα

Από τις διευθύνσεις των πελατών μέσω εφαρμογής batchgeocoding προκύπτουν οι συντεταγμένες των πελατών σε δεκαδικές μοίρες (decimal degrees – DD) που είναι συμβατές και με πολλα API και GIS. Για τον υπολογισμό των αποστάσεων χρησιμοποιήθηκε η φόρμουλα Haversine :

dlon = lon2 - lon1

dlat = lat2 - lat1

a = (sin(dlat/2))^2 + cos(lat1) * cos(lat2) * (sin(dlon/2))^2

c = 2 * arcsin(min(1,sqrt(a)))

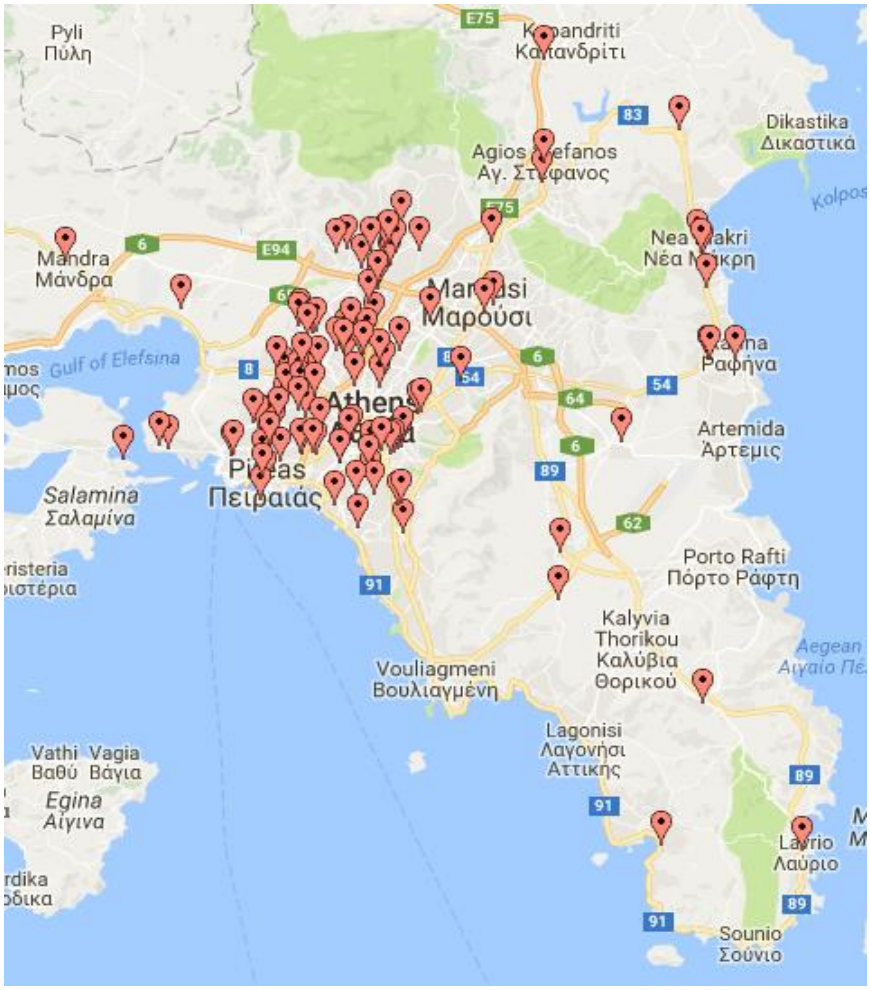
d = R * c , R=6373km για την περιοχή της Ελλάδας

Τα παραπάνω δεδομένα τροφοδότησαν το πρόβλημα. Το πρόβλημα εξυπηρέτησης 113 πελατών από 6 φορτηγά αν υποθέσουμε ότι ισοκατανέμουμε με περίπου 19 παραδόσεις ανά δρομολόγιο έχει 10^38 δυνατές λύσεις. Ο αλγόριθμος αρχικά δημιουργεί μία αρχική τυχαία λύση, κατόπιν τρέχει μεταευρετικό SA και η λύση που προκύπτει μετά από 7min δίνει τιμή αντικειμενικής συνάρτησης 2454min. Στη δεύτερη φάση ανά φορτηγό εκτελείτε βελτιστοποίηση με χρήση Solution improvements heuristics τύπου 2-exchange σε συνδυασμό με SA. Τα αποτελέσματα της 2^{ης} φάσης είναι κατά 31% βελτιωμένο σε σχέση με το αρχικό με τιμή ΑΣ=1692 σε περίπου 10min λειτουργίας.

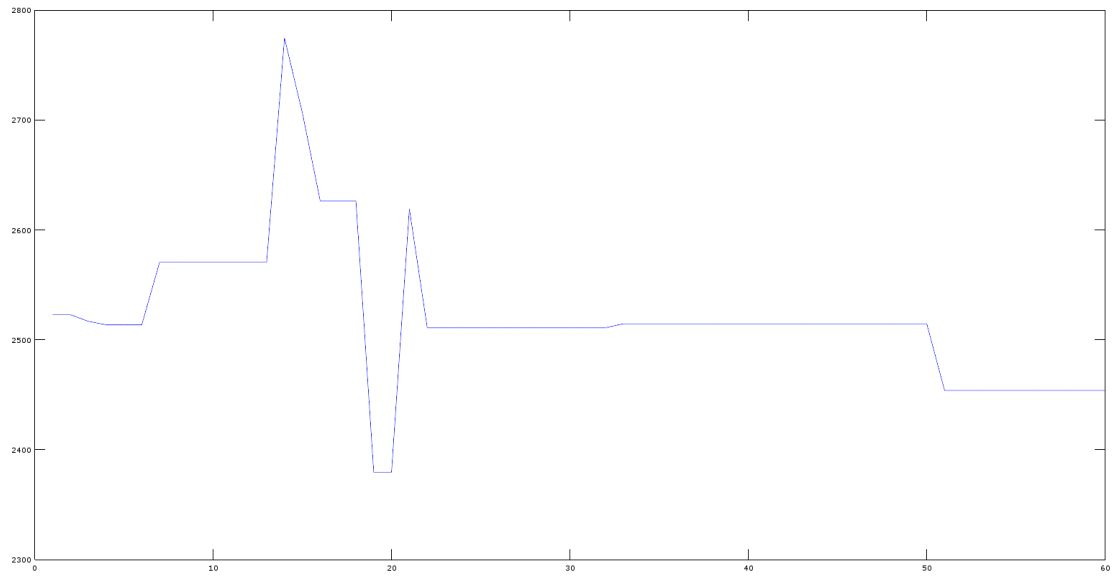
Στο παράρτημα στον πίνακα 1 μπορούμε να δούμε ανά πελάτη τα χρονικά παράθυρα, τη ζήτηση, τις συντεταγμένες ανά πελάτη.

Στον πίνακα 2 είναι η 1^{η} λύση δρομολόγησης όπου ανά φορτηγό βλέπουμε τη σειρά εξυπηρέτησης πελατών, την ώρα άφιξης στον πελάτη και το συνολικό κόστος του κάθε φορτηγού ανά πελάτη.

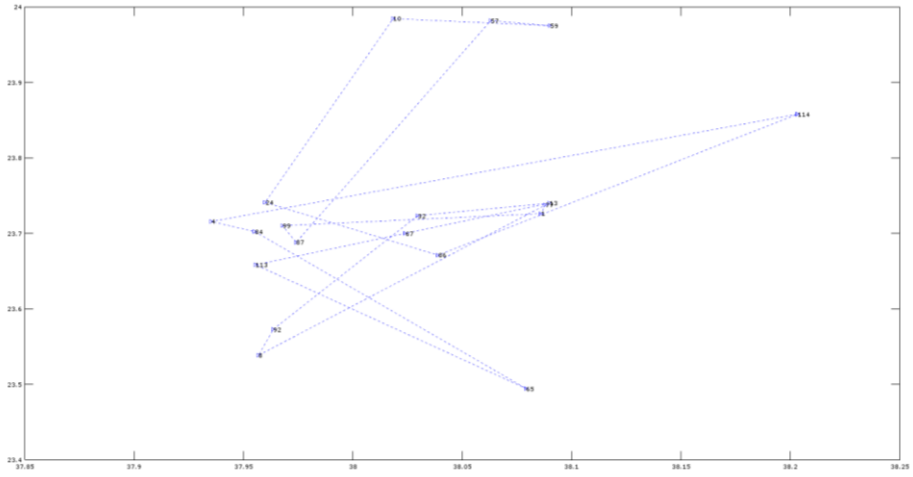
Στον πίνακα 3 είναι η 2^{η} λύση δρομολόγησης μετά τη βελτιστοποίηση με αλγόριθμο 2-exchange και SA όπου και πάλι ανά φορτηγό βλέπουμε τη σειρά εξυπηρέτησης πελατών, την ώρα άφιξης στον πελάτη και το συνολικό κόστος του κάθε φορτηγού ανά πελάτη.



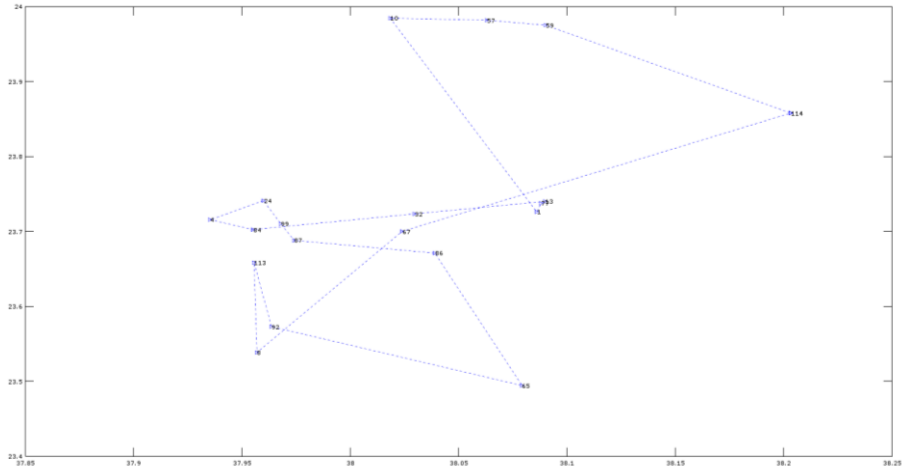
Χάρτης 1: Σύνολο Σημείων Παράδοσης



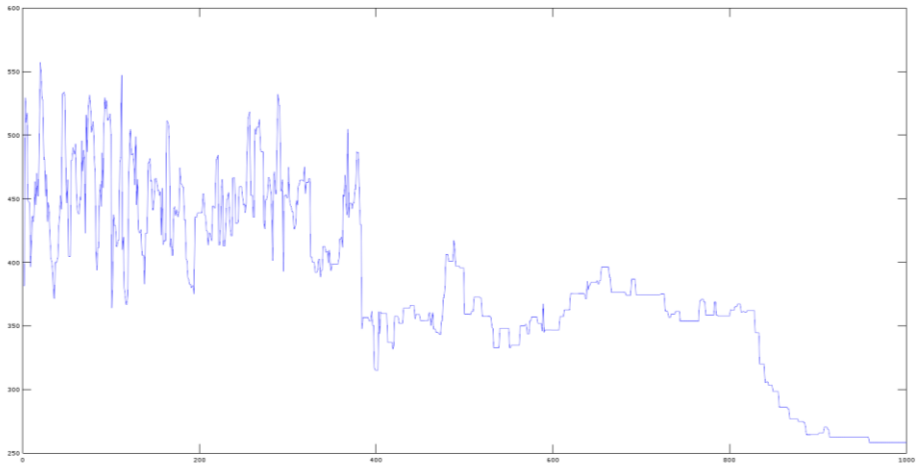
Γράφημα 1 Πτώσης κόστους Δρομολόγησης κατά το 1ο στάδιο Sim. Ann.



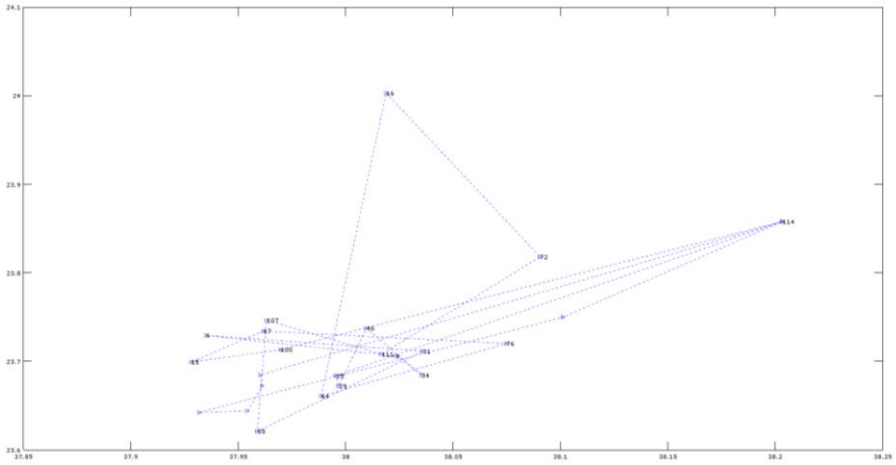
Γράφημα 2 Δρομολόγηση Φορτηγού 1 αρχική λύση



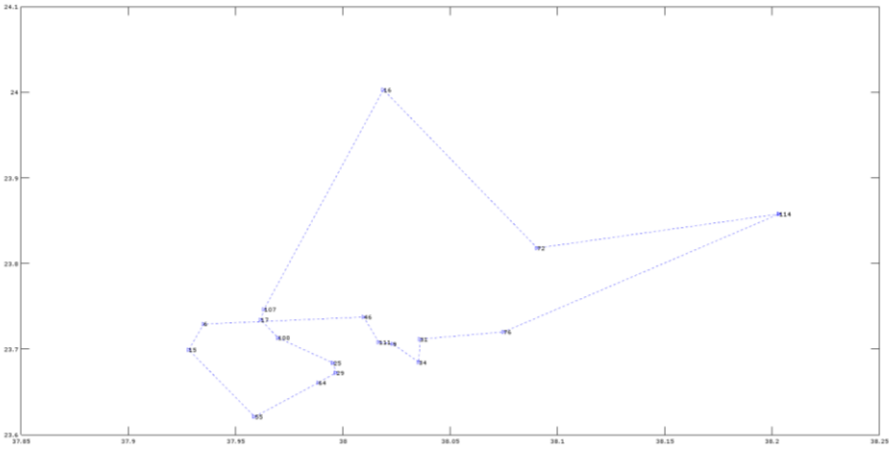
Γράφημα 3 Δρομολόγηση Φορτηγού 1 τελική λύση



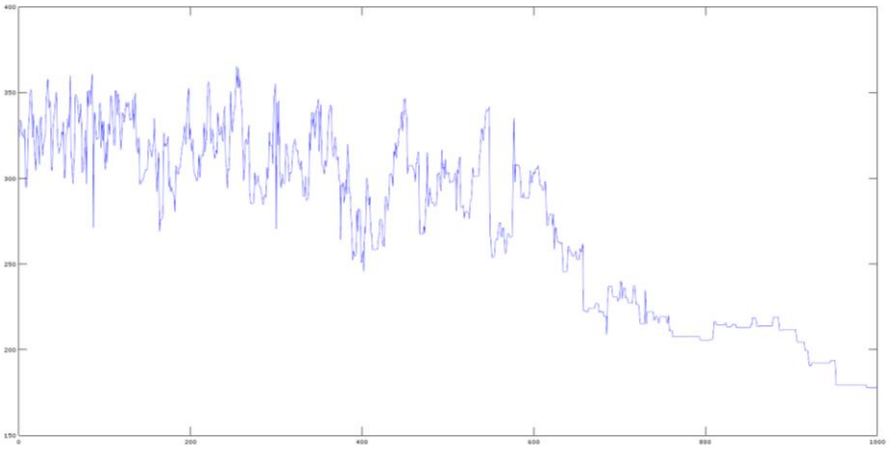
Γράφημα 4 Πτώση κόστους Δρομολόγησης Φορτηγού 1 κατά το 2ο στάδιο Sim. Ann.



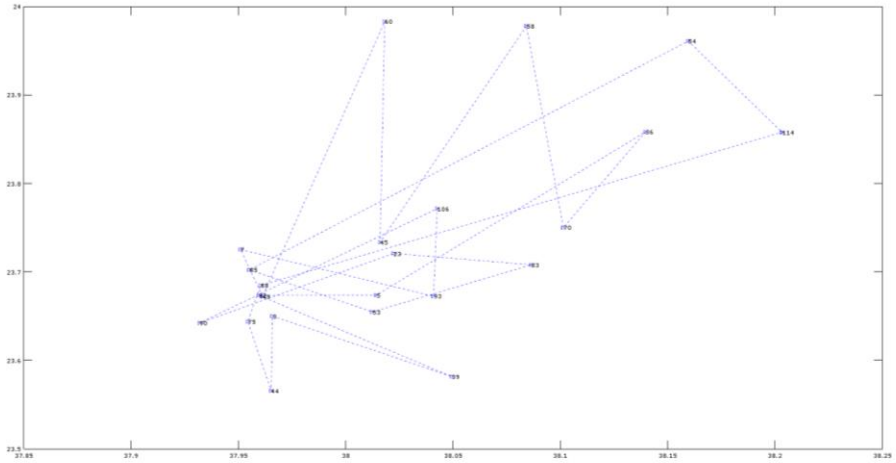
Γράφημα 5 Δρομολόγηση Φορτηγού 2 αρχική λύση



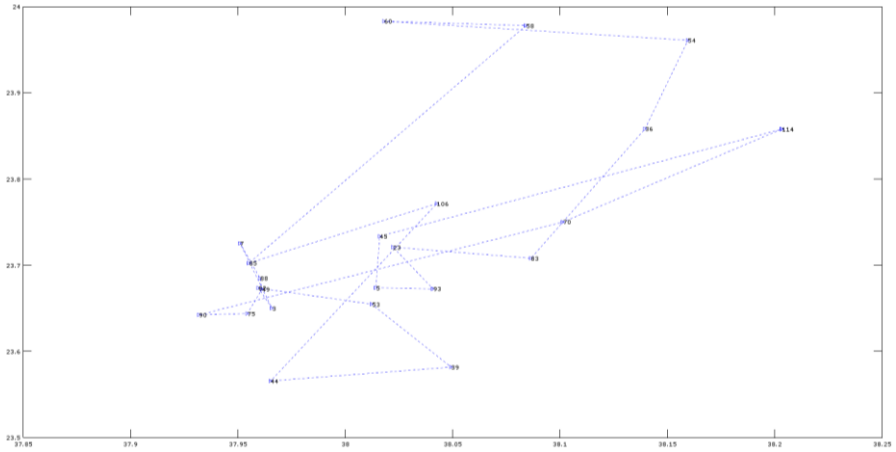
Γράφημα 6 Δρομολόγηση Φορτηγού 2 τελική λύση



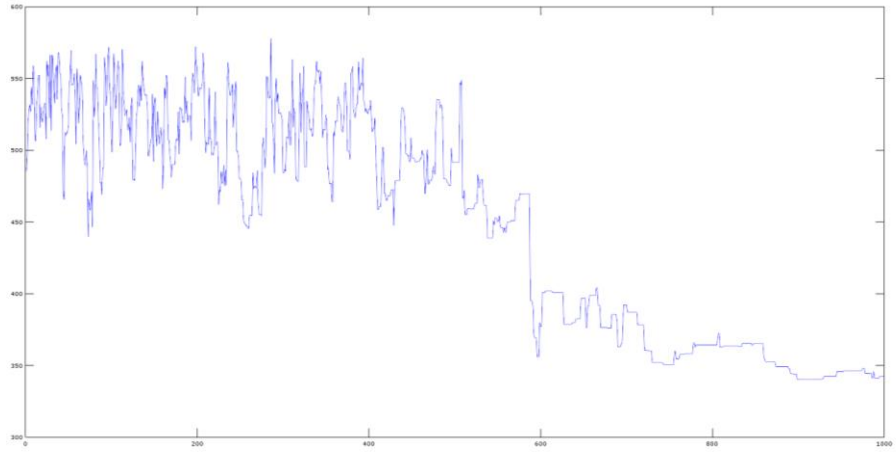
Γράφημα 7 Πτώση κόστους Δρομολόγησης Φορτηγού 2 κατά το 2ο στάδιο Sim. Ann.



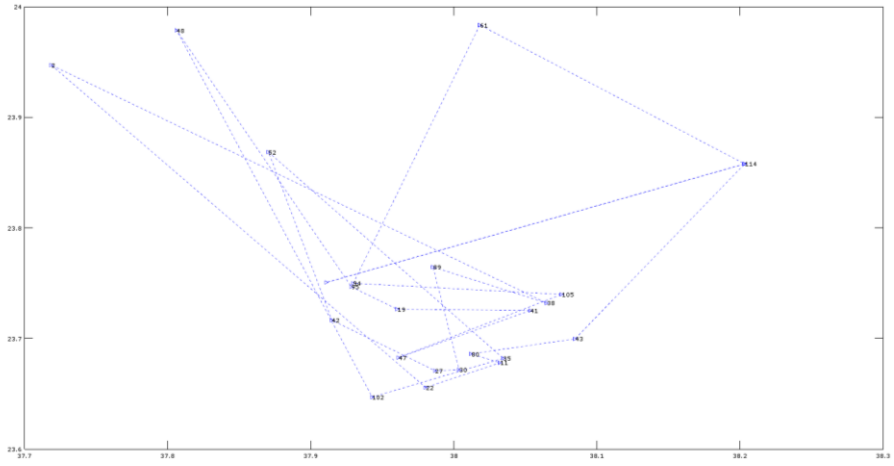
Γράφημα 8 Δρομολόγηση Φορτηγού 3 αρχική λύση



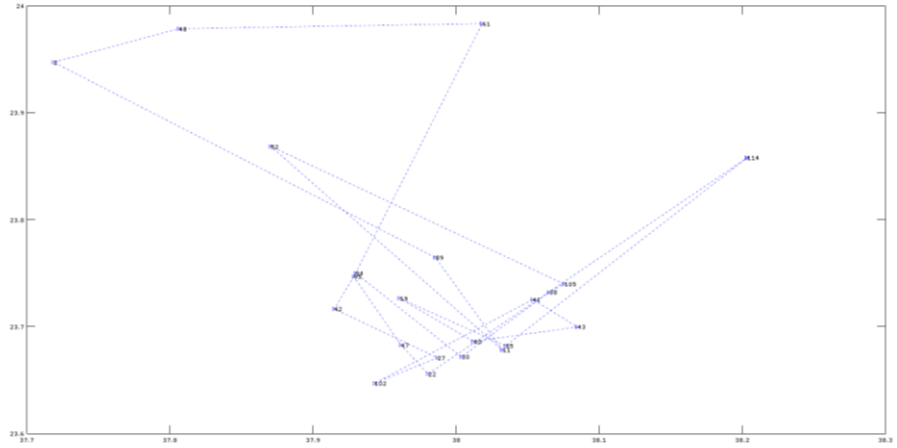
Γράφημα 9 Δρομολόγηση Φορτηγού 3 τελική λύση



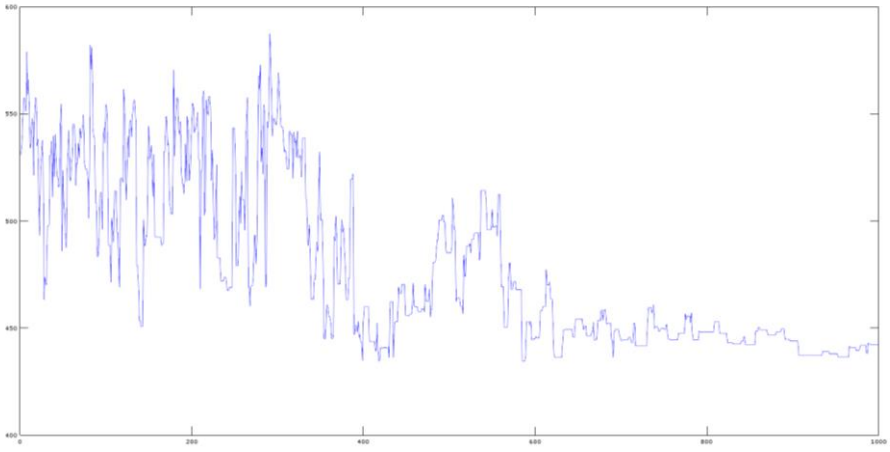
Γράφημα 10 Πτώση κόστους Δρομολόγησης Φορτηγού 3 κατά το 2^{ο} στάδιο Sim. Ann.



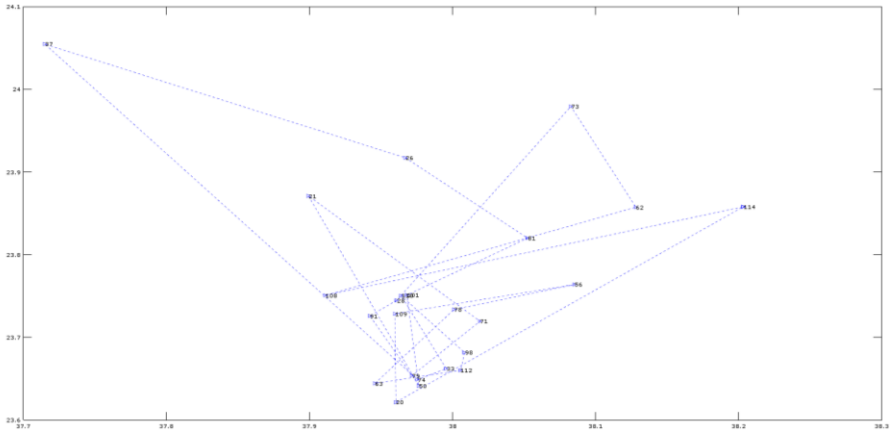
Γράφημα 11 Δρομολόγηση Φορτηγού 4 αρχική λύση



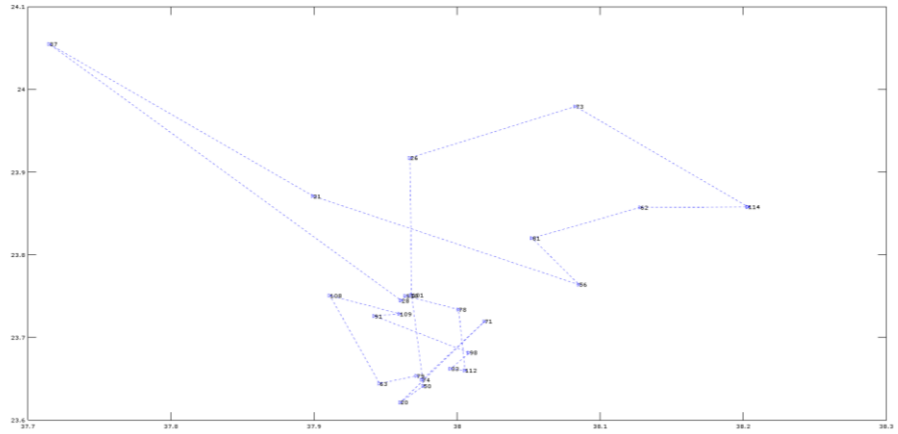
Γράφημα 12 Δρομολόγηση Φορτηγού 4 τελική λύση



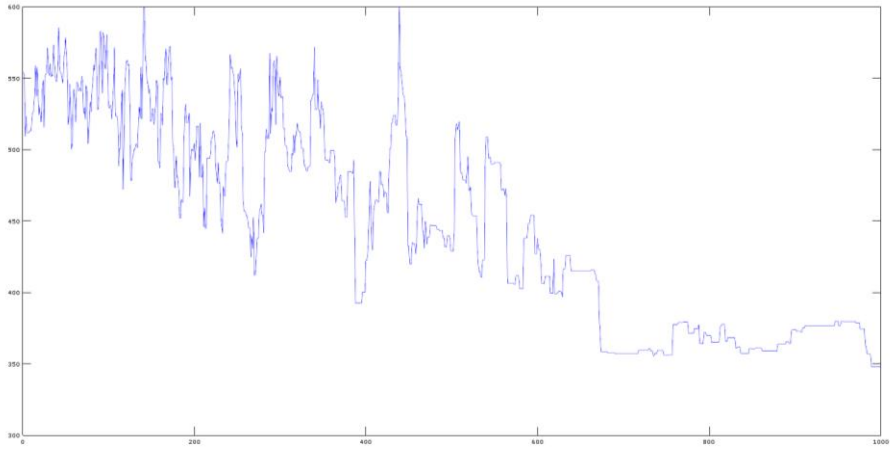
Γράφημα 13 Πτώση κόστους Δρομολόγησης Φορτηγού 4 κατά το 2^{ο} στάδιο Sim. Ann.



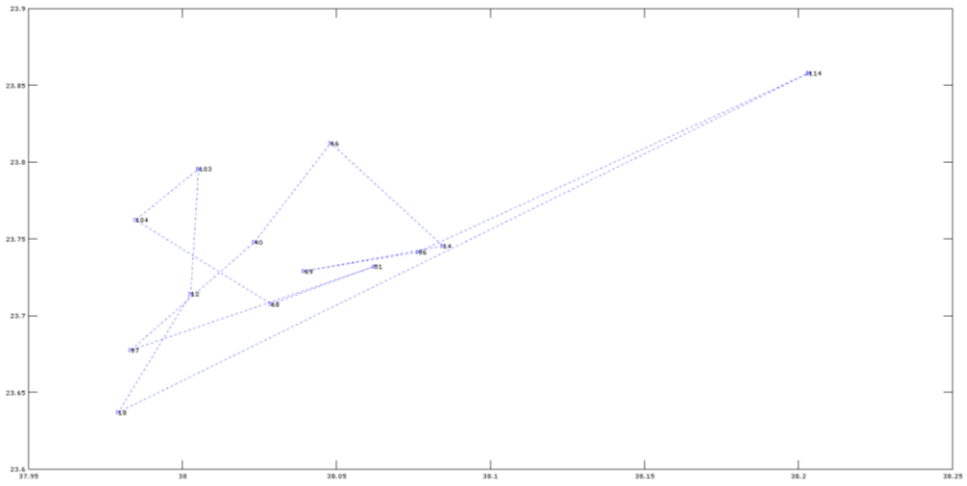
Γράφημα 14 Δρομολόγηση Φορτηγού 5 αρχική λύση



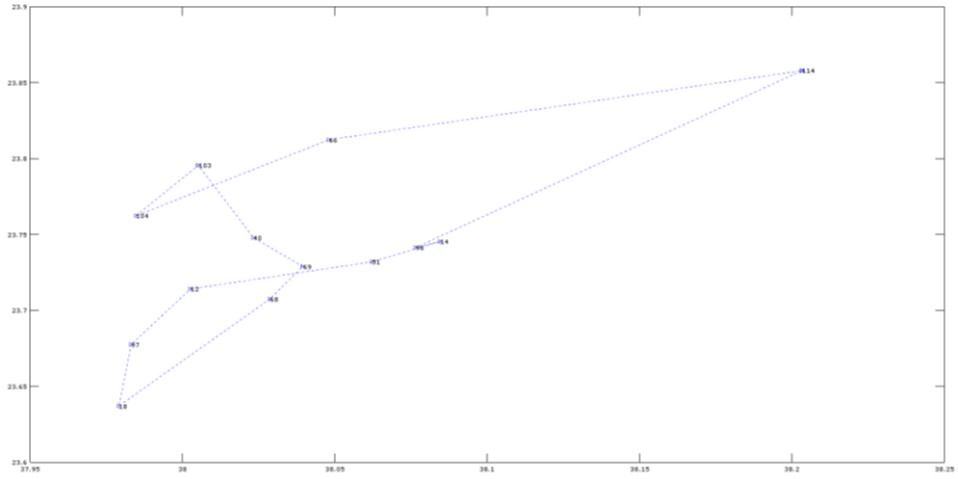
Γράφημα 15 Δρομολόγηση Φορτηγού 5 τελική λύση



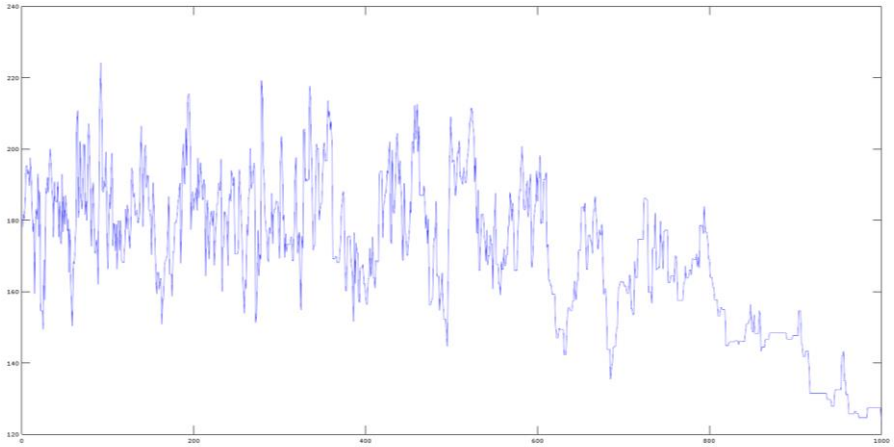
Γράφημα 16 Πτώση κόστους Δρομολόγησης Φορτηγού 5 κατά το 2^{ο} στάδιο Sim. Ann.



Γράφημα 17 Δρομολόγηση Φορτηγού 6 αρχική λύση



Γράφημα 18 Δρομολόγηση Φορτηγού 6 τελική λύση



Γράφημα 19 Πτώση κόστους Δρομολόγησης Φορτηγού 6 κατά το 2ο στάδιο Sim. Ann.

Βιβλιογραφία

1. A Simulated Annealing-Based Parallel Multi-Objective Approach To Vehicle Routing Problems With Time Windows. Raul Banow, Julio Ortega, Consolacion Gil, Antonia Fernandez, Fransisco De Toro (2012)
2. Vehicle Routing Problem With Time Windows, Part 1: Route Construction And Local Search Algorithms. Olli Braysy, Michel Gendreau (2005)
3. Vehicle Routing Problem With Time Windows, Part 2: Metaheuristics. Olli Braysy, Michel Gendreau (2005)
4. Artificial intelligence in supply chain management: theory and applications. Hokey Min (2010)
5. Minimisation of supply chain cost with embedded risk using computational intelligence approaches. Sri Krishna Kumar a , M.K. Tiwari b & Radu F. Babiceanu (2009)
6. A Branch-and-Cut Algorithm for the Capacitated Vehicle Routing Problem (lecture I), Bjorn Petersen (2006)
7. Formulations and exact algorithms for the vehicle routing problem with time windows. Brian Kallehauge (2007)
8. Solve the Vehicle Routing Problem with Time Windows via a Genetic Algorithm. Yaw Chang and Lin Chen (2007)
9. Vehicle Routing with Time Windows: Optimization and Approximation. M. Desrochers, J.K. Lenstra, M.W.P. Savelsbergh, F.Soumis (1988)
10. Vehicle Routing: Methods and Studies. B.L. Golden, A.A. Assad (1988)
11. Operational Research, Analysis and Applications, Michael Wilkes (1989)
12. Heuristic methods for vehicle routing problem with time windows. K.C. Tan, L.H. Lee, Q.L. Zhu, K. Ou (2000)
13. Solving Vehicle Routing Problems using Excel, Gunes Erdogan

Παράρτημα

customer	Window Start	Window end	duration	demand	latitude	longitude
114					38.20320	23.85778
86	480	960	480	191.667	38.03899	23.67100
24	480	960	480	77.667	37.95993	23.74100
10	480	960	480	4.667	38.01842	23.98443
59	480	960	480	24.000	38.09015	23.97508
57	480	960	480	7.000	38.06305	23.98183
87	480	960	480	19.333	37.97407	23.68799
99	480	960	480	10.333	37.96790	23.71009
1	480	960	480	10.333	38.08601	23.72602
77	480	960	480	5.000	38.08800	23.73740
8	480	960	480	21.333	37.95692	23.53847
92	480	960	480	8.000	37.96365	23.57281
32	480	960	480	173.333	38.02961	23.72335
13	480	960	480	36.000	38.08986	23.74002
67	600	720	120	9.000	38.02394	23.70002
113	480	960	480	16.000	37.95556	23.65823
65	480	960	480	140.333	38.07926	23.49419
84	720	840	120	4.667	37.95512	23.70237
4	720	840	120	45.667	37.93529	23.71553
114					38.20320	23.85778
114					38.20320	23.85778
25	480	960	480	46.667	37.99544	23.68382
29	480	960	480	24.000	37.99680	23.67236
46	480	960	480	95.000	38.00973	23.73752
34	480	960	480	16.333	38.03541	23.68466
9	480	960	480	9.667	38.02324	23.70618
6	480	960	480	4.000	37.93512	23.72928
31	480	960	480	9.000	38.03589	23.71166
55	480	960	480	4.667	37.95891	23.62094
107	480	960	480	26.000	37.96324	23.74647
111	480	960	480	11.667	38.01672	23.70811
72	480	960	480	13.667	38.09056	23.81806
16	480	960	480	4.667	38.01887	24.00310
64	480	960	480	18.333	37.98862	23.66089
76	600	720	120	10.333	38.07495	23.72011
17	480	960	480	34.333	37.96171	23.73419
15	480	960	480	4.667	37.92813	23.69922
100	660	780	120	23.667	37.96981	23.71317
114					38.20320	23.85778
114					38.20320	23.85778
54	480	960	480	13.667	38.15962	23.96097
85	480	960	480	16.000	37.95512	23.70237
53	480	960	480	10.333	38.01234	23.65477
83	480	960	480	13.667	38.08650	23.70807
23	480	960	480	126.667	38.02223	23.72099
90	480	960	480	125.000	37.93193	23.64233
106	480	960	480	13.667	38.04267	23.77163
93	480	960	480	9.000	38.04090	23.67252
7	480	960	480	6.333	37.95103	23.72538
49	480	960	480	7.667	37.96123	23.67219
60	480	960	480	63.333	38.01815	23.98306
45	600	720	120	15.000	38.01602	23.73353
58	480	960	480	18.333	38.08416	23.97802
70	480	960	480	17.333	38.10138	23.75008
36	480	960	480	12.333	38.13961	23.85824
5	480	960	480	13.000	38.01426	23.67376

82	480	960	480	141.667	37.95953	23.67353
39	480	960	480	82.000	38.04932	23.58175
3	480	960	480	11.667	37.96585	23.65010
44	480	960	480	53.333	37.96529	23.56568
75	840	960	120	5.000	37.95441	23.64395
88	840	960	120	4.667	37.96035	23.68478
114					38.20320	23.85778
114					38.20320	23.85778
35	480	960	480	15.333	38.03458	23.68205
19	480	960	480	17.333	37.96045	23.72652
80	480	960	480	32.667	38.01208	23.68611
43	480	960	480	16.667	38.08474	23.69967
41	480	960	480	6.333	38.05359	23.72539
102	480	960	480	83.333	37.94299	23.64698
27	480	960	480	55.000	37.98697	23.67089
42	480	960	480	85.000	37.91486	23.71653
94	480	960	480	10.000	37.92975	23.74996
30	480	960	480	42.000	38.00372	23.67163
38	480	960	480	23.333	38.06489	23.73206
105	480	960	480	4.667	38.07537	23.73986
52	480	960	480	36.667	37.87052	23.86858
11	480	960	480	16.667	38.03228	23.67808
89	480	960	480	21.333	37.98547	23.76462
2	480	960	480	23.667	37.71869	23.94732
48	480	960	480	10.000	37.80633	23.97870
61	480	960	480	10.667	38.01819	23.98350
95	480	960	480	45.667	37.92847	23.74671
47	480	960	480	9.333	37.96140	23.68282
22	840	960	120	12.333	37.98034	23.65565
114					38.20320	23.85778
114					38.20320	23.85778
62	480	960	480	9.333	38.12810	23.85733
81	480	960	480	6.333	38.05244	23.81994
56	480	960	480	9.333	38.08525	23.76370
21	480	960	480	37.333	37.89936	23.87097
37	480	960	480	14.000	37.71502	24.05470
28	480	960	480	10.000	37.96091	23.74456
110	480	960	480	26.333	37.96407	23.75022
78	480	960	480	5.667	38.00103	23.73360
112	480	960	480	19.333	38.00556	23.65995
33	480	960	480	26.333	37.99557	23.66205
98	480	960	480	116.667	38.00850	23.68134
91	480	960	480	12.667	37.94225	23.72584
109	480	960	480	9.000	37.95983	23.72852
108	480	960	480	9.667	37.91102	23.75069
63	480	960	480	1.667	37.94581	23.64419
79	480	960	480	42.000	37.97158	23.65344
74	480	960	480	1.667	37.97550	23.64871
71	480	960	480	12.667	38.01918	23.71934
20	480	960	480	70.333	37.96036	23.62133
50	660	780	120	107.333	37.97642	23.64122
101	480	960	480	9.000	37.96818	23.75102
26	480	960	480	9.333	37.96705	23.91715
73	480	960	480	26.000	38.08273	23.97928
114					38.20320	23.85778
114					38.20320	23.85778
18	480	960	480	30.000	37.97920	23.63701
12	480	960	480	210.667	38.00267	23.71402
103	480	960	480	28.333	38.00526	23.79546
104	480	960	480	37.000	37.98478	23.76227

68	480	960	480	4.667	38.02874	23.70749
51	480	960	480	4.333	38.06221	23.73193
97	480	960	480	52.333	37.98323	23.67753
40	480	960	480	23.667	38.02334	23.74778
66	480	960	480	317.667	38.04811	23.81237
14	480	960	480	9.333	38.08457	23.74528
69	480	960	480	26.667	38.03951	23.72893
96	480	960	480	9.667	38.07653	23.74137
114					38.20320	23.85778

Πίνακας 1 Χρονικά Παράθυρα, Ζήτηση, Συντεταγμένες ανά πελάτη

Πελάτης	Σειρά εξυπηρέτησης - Αρχική λύση	Φορτηγό	Άφιξη στον πελάτη	Κόστος αντικειμενικής συνάρτησης ανά φορτηγό
114	1	**1**	**443.219**	
86	2	1	480.000	
24	3	1	496.085	
10	4	1	529.549	
59	5	1	541.579	
57	6	1	546.185	
87	7	1	587.563	
99	8	1	590.646	
1	9	1	610.463	
77	10	1	611.995	
8	11	1	646.083	
92	12	1	650.738	
32	13	1	673.384	
13	14	1	683.672	
67	15	1	695.860	
113	16	1	708.523	
65	17	1	738.371	
84	18	1	772.691	
4	19	1	776.424	
114	20	**1**	**824.884**	**381.66**
114	1	2	**438.472**	
25	2	2	480.000	
29	3	2	481.523	
46	4	2	490.356	
34	5	2	498.519	
9	6	2	501.999	
6	7	2	517.012	
31	8	2	533.983	
55	9	2	551.511	
107	10	2	568.039	
111	11	2	578.289	
72	12	2	597.273	
16	13	2	624.368	
64	14	2	669.640	
76	15	2	686.013	
17	16	2	704.998	
15	17	2	712.248	
100	18	2	719.441	
114	19	**2**	**762.778**	**324.31**
114	1	3	**464.645**	
54	2	3	480.000	
85	3	3	528.149	
53	4	3	539.565	
83	5	3	553.783	
23	6	3	564.640	
90	7	3	582.915	
106	8	3	608.024	

93	9	3	621.051	
7	10	3	637.577	
49	11	3	644.778	
60	12	3	686.744	
45	13	3	719.546	
58	14	3	753.622	
70	15	3	783.690	
36	16	3	799.254	
5	17	3	831.261	
82	18	3	840.392	
39	19	3	859.628	
3	20	3	876.202	
44	21	3	887.307	
75	22	3	897.762	
88	23	3	903.223	
114	24	**3**	**949.690**	**485.04**
114	1	4	**444.996**	
61	2	4	480.000	
94	3	4	514.077	
105	4	4	538.409	
47	5	4	558.849	
41	6	4	575.216	
19	7	4	590.757	
95	8	4	596.717	
48	9	4	633.445	
102	10	4	682.725	
35	11	4	698.688	
52	12	4	735.450	
42	13	4	756.793	
27	14	4	770.239	
30	15	4	773.036	
89	16	4	785.636	
38	17	4	799.561	
2	18	4	863.902	
22	19	4	922.060	
11	20	4	931.212	
80	21	4	934.744	
43	22	4	946.996	
114	23	4	**975.666**	**530.67**
114	1	5	**428.938**	
20	2	5	480.000	
109	3	5	494.101	
56	4	5	515.531	
78	5	5	530.129	
63	6	5	545.068	
112	7	5	555.251	
98	8	5	558.105	
110	9	5	569.810	
33	10	5	582.541	
74	11	5	586.322	
37	12	5	655.241	
26	13	5	701.030	
81	14	5	720.168	
101	15	5	736.893	
50	16	5	751.401	
21	17	5	784.253	
71	18	5	812.494	
79	19	5	824.248	
91	20	5	834.956	
28	21	5	838.926	

73	22	5	875.870	
62	23	5	893.579	
108	24	5	932.415	
114	25	**5**	**983.164**	**554.23**
114	1	6	**432.687**	
18	2	6	480.000	
12	3	6	490.857	
103	4	6	501.572	
104	5	6	507.115	
68	6	6	517.395	
51	7	6	523.837	
97	8	6	538.830	
40	9	6	550.236	
66	10	6	559.677	
14	11	6	570.386	
69	12	6	578.204	
96	13	6	584.593	
114	14	**6**	**610.612**	**177.92**
	Συνολικό Κόστος Λύσης>>>>			**2,453.84**

Πίνακας 2 Λύση μετά την 1η βελτιστοποίηση: Σειρά εξυπηρέτησης πελατών, φορτηγό εξυπηρέτησης, κόστος ανά πελάτη

Πελάτης	Σειρά εξυπηρέτησης - Τελική λύση	Φορτηγό	Άφιξη στον πελάτη	Κόστος αντικειμενικής συνάρτησης ανά φορτηγό	Ποσοστό Βελτίωσης
114	1	**1**	**443.607**		
67	2	1	480.000		
8	3	1	504.006		
113	4	1	519.763		
92	5	1	531.081		
65	6	1	552.964		
86	7	1	577.144		
87	8	1	588.204		
99	9	1	591.287		
24	10	1	595.566		
4	11	1	600.869		
84	12	1	604.602		
32	13	1	617.334		
13	14	1	627.621		
77	15	1	628.083		
1	16	1	629.615		
10	17	1	665.390		
57	18	1	672.844		
59	19	1	677.450		
114	20	**1**	**701.793**	**258.19**	**-32%**
114	1	2	**451.981**		
76	2	2	480.000		
31	3	2	486.612		
34	4	2	490.161		
9	5	2	493.642		
111	6	2	494.760		
46	7	2	498.797		
6	8	2	511.292		
15	9	2	515.417		
55	10	2	526.925		
64	11	2	534.148		
29	12	2	536.183		
25	13	2	537.706		
100	14	2	543.465		
17	15	2	546.543		

107	16	2	548.179		
16	17	2	583.177		
72	18	2	610.271		
114	19	**2**	**629.786**	**177.80**	**-45%**
114	1	3	**444.753**		
45	2	3	480.000		
5	3	3	487.862		
93	4	3	492.310		
23	5	3	499.400		
83	6	3	510.256		
36	7	3	531.869		
54	8	3	545.756		
60	9	3	569.536		
58	10	3	580.571		
85	11	3	622.717		
106	12	3	639.931		
44	13	3	669.926		
39	14	3	684.105		
53	15	3	695.514		
82	16	3	704.663		
3	17	3	707.920		
7	18	3	718.128		
88	19	3	723.691		
49	20	3	725.354		
75	21	3	729.239		
90	22	3	732.995		
70	23	3	764.615		
114	24	**3**	**786.728**	**341.97**	**-29%**
114	1	4	**443.605**		
35	2	4	480.000		
19	3	4	493.682		
80	4	4	503.805		
43	5	4	516.057		
41	6	4	522.255		
102	7	4	543.393		
27	8	4	551.376		
42	9	4	564.822		
94	10	4	569.874		
30	11	4	585.950		
38	12	4	598.882		
105	13	4	600.908		
52	14	4	639.049		
11	15	4	675.880		
89	16	4	689.678		
2	17	4	740.280		
48	18	4	755.477		
61	19	4	790.831		
95	20	4	825.385		
47	21	4	835.428		
22	22	4	840.198		
114	23	4	**885.896**	**442.29**	**-17%**
114	1	5	**467.462**		
62	2	5	480.000		
81	3	5	493.545		
56	4	5	502.739		
21	5	5	536.810		
37	6	5	575.956		
28	7	5	633.861		
110	8	5	634.773		
78	9	5	641.315		

112	10	5	651.027		
33	11	5	652.715		
98	12	5	656.045		
91	13	5	668.553		
109	14	5	671.508		
108	15	5	680.157		
63	16	5	695.328		
79	17	5	699.797		
74	18	5	700.700		
71	19	5	712.505		
20	20	5	728.704		
50	21	5	732.448		
101	22	5	746.955		
26	23	5	768.807		
73	24	5	789.763		
114	25	**5**	**815.416**	**347.95**	**-37%**
114	1	6	**453.435**		
66	2	6	480.000		
104	3	6	492.450		
103	4	6	497.993		
40	5	6	504.949		
69	6	6	508.612		
68	7	6	511.955		
18	8	6	524.371		
97	9	6	529.741		
12	10	6	535.533		
51	11	6	545.742		
14	12	6	549.865		
96	13	6	551.301		
114	14	**6**	**577.392**	**123.96**	**-30%**
			Συνολικό Κόστος Λύσης>>>>	**1692.17**	**-31%**

Πίνακας 3 Λύση μετά την 2η βελτιστοποίηση: Σειρά εξυπηρέτησης πελατών, φορτηγό εξυπηρέτησης, κόστος ανά πελάτη, ποσοστό βελτιστοποίησης

Κώδικας 1 Session2.m : Εύρεση αρχικής λύσης δρομολόγησης

```
function [v1,tab2]=session2()
load coord2.mat % πίνακας ΣΥΝΤΕΤΑΓΜΕΝΩΝ ΠΕΛΑΤΩΝ
load constr1.mat % πίνακας ΑΠΟΣΤΑΣΕΩΝ σε km ΠΕΛΑΤΩΝ
load cost1.mat % πίνακας ΑΠΟΣΤΑΣΕΩΝ σε min ΠΕΛΑΤΩΝ
load demand.mat % πίνακας ΖΗΤΗΣΗΣ ΠΕΛΑΤΩΝ
nod=113; % όρισε αριθμός πελατών είναι nod=114-1=113
veh=6; % όρισε αριθμός φορτηγών είναι veh=6
start=114; % όρισε depot
note1=1; % Δημιουργία αρχικής λύσης:
while note1~=0 % όσο δεν ικανοποιούνται οι περιορισμοί συνέχισε
counter2=0;
```

```
count1=0;

tab=zeros(veh,nod);

tab2=[];

cust=[];

sk=[];

while sum(sum(tab))~=nod % Όσο το σύνολο του πίνακα δεν είναι ίσο με 113 τότε συνέχισε

   i1=randi(veh); % Δώσε τυχαία αριθμό φορτηγού i1 (1-6)
   j1=randi(nod); % Δώσε τυχαία αριθμό πελάτη j1 (1-113) σε κάθε αξιολογεί 1 πελάτη

   % σε κάθε loop αξιολογει μόνο ένα πελάτη for each loop only one node is accepted

   if sum(tab(i1,:))<nod && sum(tab(:,j1))==0 % Αν το άθροισμα του πίνακα-γραμμή (i1, : ) < 113 και αν το άθροισμα του πίνακα-στήλη είναι μηδέν

%    και το άθροισμα των στηλών είναι μηδέν, δηλ. ένα φορτηγό μπορεί να εξυπηρετήσει λιγότερους από 114 πελάτες και ένας πελάτης μόνο από ένα φορτηγό

     if sum(tab(i1,:))==0 % Και αν το άθροισμα του πίνακα-γραμμή (i1, : )=0 τότε:

     % accept first node and calculate sk (χρονικό παράθυρο πρώτου πελάτη)

     tab(i1,j1)=1;

     count1=count1+1;

     tab2(i1,count1)=j1; % φτιάχνω πίνακα [6X113] γραμμή =φορτηγό, στήλες = σειρά επίσκεψης πελατών

     sk(i1,1)=timewind(j1,1); % χρονικό παράθυρο πρώτου πελάτη

     else

     cust=tab2(i1,tab2(i1,:)~=0); % πελάτες οχήματος i1  // διάφορος του μηδενός

     pos=length(cust); %  θέση τελευταίου πελάτη

     skpred=sk(i1,pos); % δώσε μου το χρονικό παράθυρο του τελευταίου πελάτη που εξυπηρέτησε το φορτηγό

     sksuc=skpred+cost(cust(pos),j1); % χρονικό παράθυρο =χρονικό παράθυρο προηγούμενου πελάτη + χρόνος μεταφοράς από προηγούμενο πελάτη

     [ticket]=tonnage(i1,j1,tab2); % έλεγχος περιορισμού βάρους ανά φορτηγό <3000

     if sksuc>=timewind(j1,1) && sksuc<=timewind(j1,2) && ticket==0 % ΕΛΕΓΧΟΣ ΧΡΟΝΟΥ αν ο χρόνος άφιξης στον επόμενο πελάτη είναι ανάμεσα στο χρονικό του παράθυρο.

     tab(i1,j1)=1;

     count1=count1+1;

     tab2(i1,count1)=j1;
```

```
    sk(i1,pos+1)=sksuc; % καταχωρώ χρόνο άφιξης στον επόμενο πελάτη στον πίνακα των χρόνων
    end   % if sk(n,1)>=timewind(col(n),1) && sk(n,1)<=timewind(col(n),2)
    end   % if sum(tab(i1,:))==0
  end    % if sum(tab(i1,:))<nod && sum(tab(:,j1))==0
end   % while sum(sum(tab))~=nod
end % first while
%counter2
vv=[];
vv2=[];
vv3=[];
for n2=1:size(tab2,1) % για κάθε σειρά του πίνακα-όχημα
if any(tab2(n2,:)>0) % αν έχει το όχημα ανάθεση - πελάτη
vv=tab2(n2,tab2(n2,:)~=0);   % κατέγραψε τη σειρά επίσκεψης στους πελάτες
% we use rows to assign a truck with specific number
v4=zeros(1,length(vv)+2);
v4(1)=start; ; % αρχίζουμε από node = 114 – κέντρο διανομής
v4(end)=v4(1); % τελειώνουμε σε node = 114 – κέντρο διανομής
v4(2:end-1)=vv;
vv=v4;
vv2=[vv2,vv];
for n=1:length(vv) % πλήθος φορτηγών που έχουν ανατεθεί
vv3=[vv3,n2];
end
end
end
z=[vv2;vv3]'  ; % πίνακας γειτνίασης - adjacency matrix
v1=[];
v2=0;
for m=1:length(z)-1
        v2=v2+1; % μετρητής - counter
```

```
      v1(v2,:)=[z(m,1),z(m+1,1),z(m,2)];

end

nn=0;

for n3=1:length(v1)

if v1(n3,1)==v1(n3,2)

nn=nn+1;

n4(nn)=n3;

end

end

v1(n4,:)=[]; % αφαίρεσε τις διπλοεγγραφές στον πίνακα γειτνίασης

v1;

sk;

tab2;

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

trial=[];

count3=0;

for n1=1:size(sk,1)  % μέτρα το μήκος της γραμμής ανά φορτηγό

   for n2=1:length(sk) % για κάθε φορτηγό

    if sk(n1,n2)>0   % αν υπάρχει καταχώρηση

        count3=count3+1;

        nn=tab2(n1,tab2(n1,:)~=0); % αφαίρεσε τα μηδενικά

        trial(count3,:)=[sk(n1,n2),nn(n2),n1,timewind(nn(n2),:)]; % ο πίνακας trial μου δείχνει τι ώρα
εξυπηρετείτε ο κάθε πελάτης

    end

  end

end

save sk.mat
```

Κώδικας 2 Algorithm1.m : Βελτιστοποίηση αρχικής λύσης δρομολόγησης με Simulated Annealing

```
 [v0,tab2]=session2(); % Δημιούργησε αρχική λύση
T=100 ; % Όρισε αρχική θερμοκρασία
kiter=0; % μετρητής kiter
```

```
kiter2=0;
while T>10 % όσο η θερμοκρασία είναι πάνω από 10
while kiter<10 % κάνε n=10 επαναλήψεις πριν γίνει αλλαγή θερμοκρασίας
obj0=evalcost(v0); % αρχική τιμή αντικειμενικής συνάρτησης
 [v2,tab2]=session2(); % δώσε μία νέα λύση
obj1=evalcost(v2); % υπολόγισε τιμή αντικειμενικής συνάρτησης νέας λύσης

if obj1<obj0 % αν η νέα λύση είναι καλύτερη
   v0=v2; % υιοθέτησε νέα λύση
   tab3=tab2;

else
   p0=rand(1); % δώσε τυχαίο αριθμό από 0-1
   if p0<exp(-(obj1-obj0)/T); % αν ο τυχαίος αριθμός είναι μικρότερος από τη συνάρτηση
     v0=v2; % υιοθέτησε νέα λύση
     tab3=tab2;

   end
end
   kiter=kiter+1;
   kiter2=kiter2+1;
   y(kiter2)=obj1;
   z(kiter2)=obj0;

 [obj0,obj1,kiter2,T]
%  kiter2

% pause

end
  kiter=0;
  T=T/log(4.5); %% πρόγραμμα ψύξης - kiter2

end
toc
%figure(1)
%plot(y)
figure(2)
plot(z)
tab2=tab3;

more off
%plot1
save tab2.mat % σώσε τη λύση για περαιτέρω επεξεργασία στο project2
```

Κώδικας 3 project2.m : Καθορισμός δρομολόγησης φορτηγού Χ αρχικής λύσης που θα βελτιστοποιηθεί σε δεύτερη φάση με Simulated Annealing

```
function [z4]=project2()
% returns the matrix that will be optimized
%  WE MUST ALREADY HAVE SAVED ALGORITHM1 SOLUTION BEFORE WE BEGIN save tab2 to a file //
algorithm 1
```

```
load tab2
load cost1.mat
load constr1.mat % Χρονικά παράθυρα

start=114; % Τελευταίος κόμβος σαν depot

vv=[];
vv2=[];
vv3=[];

for n2=1:size(tab2,1) % για κάθε γραμμή του πίνακα λύση (όχημα)
if any(tab2(n2,:)>0)
vv=tab2(n2,tab2(n2,:)~=0);   % δώσε την σειρά επίσκεψης των πελατών από τυχαία διαδικασία
(session2.m)
% we use rows to assign a truck with specific number
v4=zeros(1,length(vv)+2);
v4(1)=start; % ορισμός αρχικού πελάτη starting point
v4(end)=v4(1); % ορισμός τελικού πελάτη ending point
v4(2:end-1)=vv;
vv=v4;

vv2=[vv2,vv];

for n=1:length(vv) % trucks assigned to delivery
vv3=[vv3,n2];
end

end
end
z=[vv2;vv3]' ;
%*************************************************************
% PLEASE INPUT MANUALLY THE NUMBER OF TRUCK AFTER '=='
% if second column of matrix z is equal to truck number return first column
z4=z(z(:,end)==4,1) ;
```

Κώδικας 4 saphase2.m : Βελτιστοποίηση δρομολόγησης φορτηγού Χ αρχικής λύσης με Simulated Annealing

```
clear,clc
tic
more off;
[z4]=project2(); %get initial solution

T=100 ; % Όρισε αρχική θερμοκρασία
kiter=0; % μετρητής
kiter2=0;

while T>2 % επανάληψη μέχρι η θερμοκρασία να είναι πολύ χαμηλή

while kiter<100 % κάνε επαναλήψεις για n=100 φορές πριν αλλάξεις θερμοκρασία
obj0=evaltwo(z4) ; % αξιολογεί την πρώτη λύση (ΑΣ)
```

```
 [znew]=createpermut(z4); % δώσε νέα λύση με κλήση του προγράμματος createpermut - αντιμετάθεση
obj1=evaltwo(znew) ; % αξιολόγησε τη νέα λύση (ΑΣ)
if obj1<obj0 % αν η νέα λύση είναι μικρότερη από την αρχική τότε κάνε την αρχική
   z4=znew;
% αν η νέα λύση είναι χειρότερη από την αρχική τότε αποδέξου τη νέα λύση αν η κατανομή Boltzmann στη τρέχουσα θερμοκρασία είναι μεγαλύτερη από τον τυχαίο αριθμό p0 = [0,1]
else
   p0=rand(1);
   if p0<exp(-(obj1-obj0)/T);

     z4=znew;


   end
end

   kiter=kiter+1;
   kiter2=kiter2+1;
   y(kiter2)=obj1;
   z(kiter2)=obj0;

 [obj0,obj1,kiter2,T]
%  kiter2


end % while kiter
% cooling schedule --- πρόγραμμα ψύξης
   kiter=0;
  T=T/log(4.5);  % kiter2

end % while T
obj0
toc
figure(1)
plot(z)
figure(2)
plottruck(z4)
```

Κώδικας 5 createpermut.m : Αντιμετάθεση πελατών στη δρομολόγηση φορτηγού Χ αρχικής λύσης

```
function [z4]=createpermut(z4)

load constr1.mat
load cost1.mat
% φόρτωσε την αρχική λύση obtain the solution that is already made
z4=z4(2:end-1);   % αφαίρεσε αρχικό και τελικό κόμβο δλδ αποθήκη
z5=z4;
%plottruck(z4) % plot initial solution
count10=0;
note2=1; % δημιούργησε τυχαίες αντιμεταθέσεις ανάμεσα σε 2 κόμβους του δρομολογίου
```

```
while note2==1;
note2=0;
randnum1=randi(length(z5));
randnum2=randi(length(z5));
while randnum1==randnum2 % if num1 equals to num2 give us two new numbers
randnum1=randi(length(z5));
randnum2=randi(length(z5));
end

% cross variables to create permutation
testvar1=z5(randnum1);
testvar2=z5(randnum2);
z5(randnum1)=testvar2;
z5(randnum2)=testvar1;
for n10=2:length(z5) % έλεγξε τα χρονικά παράθυρα για κάθε στοιχείο του νέου πίνακα λύση
pred=z5(n10-1);
succ=z5(n10);

if n10==2
timepred=timewind(pred,1);
timebox(1)=timepred;
else
timepred=timebox(n10-1); % contains routes total time
end

timesucc=timepred+cost(pred,succ);
timebox(n10)=timesucc; % φτιάξε τη συονολική διάρκεια της διαδρομής

% Χρονικοί περιορισμοί
if timesucc>=timewind(succ,1) && timesucc<=timewind(succ,2)
% Αποδέξου αλλαγές και προχώρα σε αξιολόγηση
else
note2=1;
% οι αλλαγές δεν είναι αποδεκτές και θα πρέπει να επιστρέψεις στην αρχική κατάσταση
%[timewind(succ,1),timewind(succ,2),timesucc]
end
end % for n10=2:length(z4)
if note2==1
% απέρριψε την λύση
else% αποδέξου την λύση//δλδ note2 παρέμεινε μηδέν//και το while θα σταματήσει, έτσι θα
επιστρέψει στον πίνακα z4
start=114;
z6=zeros(1,length(z5)+2);
z6(1)=start;
z6(2:end-1)=z5(:);
z6(end)=z6(1);
z4=z6;
end
end %while
```