ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ ΠΑΡΑΓΩΓΗΣ ΚΑΙ ΔΙΟΙΚΗΣΗΣ

ΔΙΑΔΙΚΑΣΙΑ ΑΠΛΗΣΤΗΣ ΤΥΧΑΙΟΠΟΙΗΜΕΝΗΣ ΠΡΟΣΑΡΜΟΣΤΙΚΗΣ ΑΝΑΖΗΤΗΣΗΣ ΓΙΑ ΤΟ ΧΡΟΝΙΚΩΣ ΕΞΑΡΤΩΜΕΝΟ ΠΡΟΒΛΗΜΑ ΔΡΟΜΟΛΟΓΗΣΗΣ ΟΧΗΜΑΤΩΝ

GREEDY RANDOMIZED ADAPTIVE SEARCH PROCEDURE FOR THE TIME DEPENDENT VEHICLE ROUTING PROBLEM

Νασούλης Σταμάτιος

Επιβλέπων Καθηγητής: Δρ. Μαρινάκης Ιωάννης

Χανιά, 2021

Ευχαριστίες

Αρχικά, θα ήθελα να ευχαριστήσω τον κύριο Μαρινάκη Ιωάννη για την άψογη συνεργασία και την ουσιαστική καθοδήγηση καθόλη τη διάρκεια εκπόνησης της διπλωματικής μου εργασίας.

Έπειτα, θα ήθελα να ευχαριστήσω την οικογένειά μου για την αμέριστη υποστήριξη καθόλη τη διάρκεια της φοίτησής μου στο Πολυτεχνείο Κρήτης, καθώς και για τις θυσίες που έχουν κάνει όλα αυτά τα χρόνια.

Τέλος, θα ήθελα να ευχαριστήσω τους φίλους μου για την παρέα, τους προβληματισμούς και τις αναμνήσεις που μοιραστήκαμε κατά τη διάρκεια των φοιτητικών μας χρόνων.

Περιεχόμενα

Περίληψη

Τα τελευταία χρόνια παρατηρείται έντονα η τάση των επιχειρήσεων να επενδύσουν στη Διαχείριση της Εφοδιαστικής Αλυσίδας. Πράγματι, η Διαχείριση της Εφοδιαστικής Αλυσίδας, ή εν συντομία τα Logistics, αποτελούν πλέον βασικό πυλώνα στον στρατηγικό σχεδιασμό των επιχείρησεων κι όχι άδικα. Η βελτιστοποίηση των μεταφορών και των αποθεμάτων έχει πάψει να θεωρείται απλώς ένα ανταγωνιστικό πλεονέκτημα, αφού πλέον αποτελεί επιτακτική ανάγκη για τη βιωσιμότητα των επιχειρήσεων. Η παρούσα διπλωματική εργασία πραγματεύεται ένα από τα σημαντικότερα προβλήματα της Εφοδιαστικής Αλυσίδας, αυτό της δρομολόγησης οχημάτων. Πιο συγκεκριμένα, πρόκειται για το Χρονικώς Εξαρτώμενο Πρόβλημα Δρομολόγησης Οχημάτων (Time Dependent Vehicle Routing Problem), το οποίο προσπαθεί να προσεγγίσει ρεαλιστικά μία πραγματική κατάσταση με στόχο τη βελτιστοποίηση του κόστους. Με την έννοια πραγματική κατάσταση νοούνται όλες εκείνες οι παράμετροι που ενδέχεται να επηρεάσουν τον τρόπο και τους χρόνους των μεταφορών, όπως για παράδειγμα η κυκλοφοριακή συμφόρηση ή η κατάσταση του οδικού δικτύου. Γι΄ αυτό, οι χρόνοι μετάβασης εξαρτώνται από τη χρονική στιγμή της ημέρας κατά την οποία πραγματοποιείται μία μεταφορά, καθώς η μέρα χωρίζεται στα κατάλληλα χρονικά διαστήματα. Έπειτα, ο αλγόριθμος, που χρησιμοποείται για την επίλυση του προβλήματος, είναι ο GRASP (Greedy Randomized Adaptive Search Procedure). Ο GRASP πρόκειται για μία επαναληπτική διαδικασία, η οποία αποτελείται από δύο φάσεις. Στην πρώτη φάση κατασκευάζεται μία πιθανή λύση μέσω μίας τυχαιοποιημένης συνάρτησης απληστίας. Η συνάρτηση, αυτή, σε κάθε επανάληψη επιλέγει τυχαία τον επόμενο πελάτη προς εξυπηρέτηση μέσα από μία λίστα περιορισμένων υποψηφίων (Restricted Candidate List). Στη δεύτερη φάση, η λύση που κατασκευάστηκε, ακολουθεί μία διαδικασία τοπικής αναζήτησης, προκειμένου να βελτιωθεί το συνολικό της κόστος. Εν τέλει, συγκρίνοτας όλες τις πιθανές λύσεις που κατασκευάστηκαν, προκύπτει το τελικό αποτέλεσμα. Δηλαδή, η βέλτιστη λύση μαζί με το αντίστοιχο κόστος.

1 ΕΙΣΑΓΩΓΗ

1.1 Η Εφοδιαστική Αλυσίδα (Supply Chain)

Με τον όρο Εφοδιαστική Αλυσίδα νοούνται όλες εκείνες οι ενέργειες που αφορούν τη διαδικασία μεταφοράς ενός προϊόντος ή μίας υπηρεσίας από το σημείο παραγωγής έως και το σημείο διανομής. Βασικός στόχος είναι η αύξηση της κερδοφορίας της επιχείρησης και η βέλτιστη κατά το δυνατόν ικανοποίηση των πελατών της. Σήμερα, σε μία παγκοσμιοποιημένη και ψηφιακή οικονομία, η Εφοδιαστική Αλυσίδα αποτελείται από ένα σύνολο επιχειρήσεων ή οργανισμών οι οποίοι είναι υπεύθυνοι για την παραγωγή ενός προϊόντος, ξεκινώντας από τις πρώτες ύλες έως και την τελική του διάθεση στον πελάτη. Επομένως, γίνεται κατανοητό ότι ο ανταγωνισμός περνάει σε συλλογικό επίπεδο και η εύρυθμη λειτουργία της Εφοδιαστικής Αλυσίδας αποτελεί επιτακτική ανάγκη για τη βιωσιμότητα των επιχειρήσεων.

Οι βασικές λειτουργίες της Εφοδιαστικής Αλυσίδας συνοψίζονται ως εξής:

- Προμήθειες
- Παραγωγή
- Μεταφορές
- Αποθήκευση
- Διανομή
- Εξυπηρέτηση Πελατών

1.2 Διαχείριση Εφοδιαστικής Αλυσίδας

Η Διαχείριση της Εφοδιαστικής Αλυσίδας αναφέρεται στον σχεδιασμό και τη διαχείρηση όλων εκείνων των ενεργειών που αφορούν τις διαδικασίες προμήθειας, παραγωγής και διανομής. Επιπλέον, περιλαμβάνει τον συντονισμό και τη συνεργασία όλων των εταίρων του καναλιού εφοδιασμού, οι οποίοι μπορεί να είναι προμηθευτές, μεσάζοντες, εταιρείες παροχής υπηρεσιών και πελάτες. Ουσιαστικά, η Διαχείριση της Εφοδιαστικής Αλυσίδας συγκεντρώνει και ολοκληρώνει τις ενέργειες που σχετίζονται με τον σχεδιασμό, τις προμήθειες, την παραγωγή, την αποθήκευση, τη μεταφορά και τις πωλήσεις.

Επειδή η Εφοδιαστική Αλυσίδα δραστηριοποείται σε ένα περιβάλλον δυναμικά μεταβαλλόμενο, η ταχύτητα με την οποία οι επιχειρήσεις αντιλαμβάνονται και ανταποκρίνονται στις πελατειακές ανάγκες είναι ιδιαίτερα σημαντική για την κερδοφορία τους. Εξίσου σημαντική είναι και η ευελιξία τους, έτσι ώστε να μπορούν να καλύπτουν τις ανάγκες των πελατών έγκαιρα και αποτελεσματικά. Συνεπώς, η συνεχής εγρήγορση και η άμεση συμμόρφωση των επιχειρήσεων στα πλαίσια των καινούριων αναγκών, που προκύπτουν με ταχύτατο ρυθμό, αποτελούν σημαντικές παραμέτρους για την εξασφάλιση του ανταγωνιστικού πλεονεκτήματος και κατ΄ επέκταση της κερδοφορίας τους.

1.3 Logistics

Τα Logistics αποτελούν τμήμα της Διαχείρισης της Εφοδιαστικής Αλυσίδας που σχεδιάζει, υλοποιεί και ελέγχει την αποδοτική και αποτελεσματική ροή και αποθήκευση των προϊόντων, των υπηρεσιών και των πληροφοριών από το σημείο παραγωγής έως το τελικό σημείο διάθεσης. Αξίζει να σημειωθεί ότι η ροή αυτή μπορεί να είναι κανονική ή και αντίστροφη. Πρακτικά, τα Logistics αφορούν την παράδοση του σωστού προϊόντος στο σωστό μέρος τη σωστή στιγμή στη σωστή ποσότητα με βασική επιδίωξη την ικανοποίηση των πελατών. Παρακάτω αναφέρονται ενδεικτικά μερικοί τομείς που βρίσκουν εφαρμογή τα Logistics:

- Business Logistics
- Systems Logistics
- Medical Logistics
- Military Logistics
- Environmental Logistics

Οι τέσσερις βασικοί πυλώνες των Logistics αναλύονται εν συντομία παρακάτω και έχουν ως εξής:

- Οι **απαιτήσεις** που αφορούν την ανάλυση, τη σύνθεση και τον καθορισμό όλων εκείνων των πόρων που είναι απαραίτητοι για την υλοποίηση ενός στόχου.
- Ο **εφοδιασμός** που αφορά την εξασφάλιση όλων εκείνων των πόρων, π.χ υλικά, μηχανήματα, συστήματα, ανθρώπινο δυναμικό, εκπαίδευση ανθρώπινου δυναμικού, που είναι απαραίτητοι για την υλοποίηση του στόχου.

- Ο **σχεδιασμός** που περιλαμβάνει την ανάπτυξη του σχεδιασμού, τον έλεγχο και την αξιολόγησή του.
- Η **συντήρηση** που αφορά την προστασία, τη διατήρηση και την ανάκτηση όλων των διατεθιμένων πόρων.

2 ΔΡΟΜΟΛΟΓΗΣΗ ΟΧΗΜΑΤΩΝ

2.1 Το Πρόβλημα Δρομολόγησης Οχημάτων

Το Πρόβλημα Δρομολόγησης Οχημάτων (Vehicle Routing Problem - VRP) αποτελεί ένα από τα σημαντικότερα προβλήματα της Διαχείρισης της Εφοδιαστικής Αλυσίδας, το οποίο αφορά τη διανομή / παραλαβή προϊόντων στους / από τους πελάτες. Το ίδιο πρόβλημα απασχολεί έντονα κι άλλους «συγγενικούς» κλάδους, όπως αυτοί της Συνδυαστικής Βελτιστοποίησης, των Μεταφορών και εν γένει της Επιχειρησιακής Έρευνας. Τα τελευταία χρόνια, το συγκεκριμένο πρόβλημα συγκεντρώνει εξαιρετικό ενδιαφέρον τόσο από πλευρά έρευνας όσο και ανάπτυξης. Κι αυτό, γιατί η εύρεση του βέλτιστου συνόλου διαδρομών βοηθά σημαντικά τους αποφασίζοντες να μειώσουν το λειτουργικό κόστος της Εφοδιαστικής Αλυσίδας και κατά συνέπεια να αυξήσουν την κερδοφορία της. Επιπλέον, κατά τη διαδικασία της μοντελοποίησης οι αποφασίζοντες έχουν τη δυνατότητα να «εξατομικεύσουν» το πρόβλημα. Πιο συγκεκριμένα, μπορούν να προσαρμόσουν το πρόβλημα στις συνθήκες και τους περιορισμούς που το διέπουν, έτσι ώστε να υπάρχει η κατά το δυνατόν ρεαλιστικότερη προσομοιώση του πραγματικού προβλήματος. Ενδεικτικά, ορισμένοι περιορισμοί που αφορούν ένα πρόβλημα μπορεί να είναι οι πελάτες, τα οχήματα, τα δρομολόγια, η κυκλοφοριακή ροή, η κατάσταση του οδικού δικτύου, η κατανάλωση ενέργειας, κτλ.

Στην παραπάνω παράγραφο γίνεται λόγος για την εύρεση ενός συνόλου βέλτιστων λύσεων. Αυτό συμβαίνει, γιατί από τη πρώτη φορά που παρουσιάστηκε το συγκεκριμένο πρόβλημα (Dantzig and Ramster, 1959) αποδείχθηκε ότι πρόκειται για ένα μη πολυωνυμικά δύσκολο (Non-Polynomial hard - NP-hard) πρόβλημα ακόμα και στις πιο απλές μορφές του. Επομένως, είναι πρακτικά αδύνατο να βρεθεί ένα βέλτιστο σύνολο λύσεων υπό ρεαλιστικές συνθήκες. Για τον λόγο αυτό, έχει αναπτυχθεί μία μεγάλη ποικιλία από ευρετικούς, μεθευρετικούς, εξελικτικούς και αλγορίθμους εμπνευσμένους από τη φύση για την επίλυση του προβλήματος δρομολόγησης οχημάτων.

2.2 Το Χρονικώς Εξαρτώμενο Πρόβλημα Δρομολόγησης Οχημάτων

Το Χρονικώς Εξαρτώμενο Πρόβλημα Δρομολόγησης Οχημάτων (Time Dependent Vehicle Routing Problem - TDVRP) αποτελεί μία παραλλαγή του κλασι-

κού προβλήματος δρομολόγησης οχημάτων, το οποίο αφορά τη βέλτιστη δρομολόγηση ενός στόλου από οχήματα δεδομένης χωρητικότητας όταν οι χρόνοι ταξιδιού είναι χρονικά εξαρτώμενοι. Με την έννοια χρονικά εξαρτώμενοι νοείται ότι ο χρόνος, που απαιτείται για να πραγματοποιηθεί κάθε δεδομένη διαδρομή, εξαρτάται από την ώρα της ημέρας που το ταξίδι ξεκινά από τον αρχικό κόμβο. Το TDVRP πρόκειται για ένα συνεχές πρόβλημα βελτιστοποίησης που προέκυψε από την ανάγκη μίας ρεαλιστικής προσέγγισης του προβλήματος δρομολόγησης οχημάτων σε αστικά περιβάλλοντα. Πιο συγκεκριμένα, στις περισσότερες αστικές περιοχές του κόσμου παρατηρείται έντονα το φαινόμενο της κυκλοφοριακής συμφόρησης, η οποία επηρεάζει σημαντικά τους χρόνους ταξιδιού κατά τη διάρκεια των ωρών αιχμής. Παράλληλα, οι ιδιαιτερότητες του οδικού δικτύου, που δύναται να συναντηθούν από περιοχή σε περιοχή, προκαλούν κι αυτές σημαντικές διαφοροποιήσεις στους τελικούς χρόνους των ταξιδιών. Για τους λόγους αυτούς, λοιπόν, αναπτύχθηκε το TDVRP με αποτέλεσμα οι λύσεις που προκύπτουν να ανταποκρίνονται όσο πιο ρεαλιστικά γίνεται στο πραγματικό πρόβλημα. Τέλος, αξίζει να σημειωθεί ότι το TDVRP μπορεί να υπόκειται σε χρονικούς περιορισμούς, όπως για παράδειγμα τα σκληρά χρονικά παράθυρα για τους πελάτες.

Το TDVRP με σκληρά χρονικά παράθυρα μπορεί να περιγραφεί ως εξής: Έστω $G = (V, A)$ ένα γράφημα, όπου $A = (v_i, v_j) : i \neq j \wedge i, j \in V$ είναι το σύνολο τόξων και $V = (v_0, ..., v_{n+1})$ το σύνολο των κόμβων. Οι κορυφές v_0 και v_{n+1} συμβολίζουν την αποθήκη η οποία αποτελεί τη βάση για τα οχήματα δεδομένης χωρητικότητας q_{max}. Κάθε κόμβος στο V έχει μία αντίστοιχη ζήτηση $q_i \geq 0$, ένα χρόνο εξυπηρέτησης $g_i \geq 0$ και ένα χρονικό παράθυρο εξυπηρέτησης $[e_i, l_i]$ (η αποθήκη έχει $q_0 = 0$ και $g_0 = 0$). Το σύνολο των κόμβων $C = (v_1, .., v_n)$ καθορίζει το σύνολο των n πελατών. Ο χρόνος άφιξης ενός οχήματος στον πελάτη $i, i \in C$ συμβολίζεται με a_i και ο χρόνος αναχώρησης από αυτόν με b_i. Κάθε τόξο (v_i, v_j) έχει μία αντίστοιχη σταθερή απόσταση $d_{ij} \geq 0$ και ένα χρόνο ταξιδιού $t_{ij}(b_i) \geq 0$ ο οποίος είναι συνάρτηση του χρόνου αναχώρησης από τον πελάτη i. Το σύνολο των διαθέσιμων οχημάτων συμβολίζεται με K. Το κόστος ανά μονάδα της διάρκειας της διαδρομής συμβολίζεται με c_t. Το κόστος ανά μονάδα της διανυόμενης απόστασης συμβολίζεται με c_d. Επίσης, υποτίθεται ότι το πρόβλημα είναι εφικτό. Δηλαδή είναι πάντα εφικτό να εξυπηρετηθεί κάθε πελάτης, ξεκινώντας από την αποθήκη.

Ο πρωτεύον αντικειμενικός στόχος του TDVRP είναι η ελαχιστοποίηση του αριθμού των διαδρομών, όπου ο βέλτιστος αριθμός διαδρομών είναι άγνωστος. Ο δευτερεύων αντικειμενικός στόχος είναι η ελαχιστοποίηαη του συνολικού χρόνου ή της απόστασης. Οι μεταβλητές απόφασης του προβλήματος είναι δύο:

- x_{ij}^k είναι μία δυαδική μεταβλητή απόφασης που υποδεικνύει αν το όχημα k ταξιδεύει μεταξύ των πελατών i και j.
- y_i^k υποδεικνύει το χρόνο έναρξης της εξυπηρέτησης του πελάτη i από το όχημα k.

2.2.1 Μοντελοποίηση

Το TDVRP μοντελοποιείται ως εξής:

Πρωτεύον αντικειμενικός στόχος:
$\min \sum_{k\in K} \sum_{j\in C} x_{0j}^k$ (1)

Δευτερεύων αντικειμενικός στόχος:
$\min c_d \sum_{k\in K} \sum_{(i,j)\in A} d_{ij}^k x_{ij}^k + c_t \sum_{k\in K} \sum_{j\in C} (y_{n+1}^k - y_0^k) x_{0j}^k$ (2)

υπό

- $\sum_{i\in C} q_i \sum_{j\in V} x_{ij}^k \leq q_{max}, \forall k \in K$ (3)
- $\sum_{k\in K} \sum_{j\in V} x_{ij}^k = 1, \forall i \in C$ (4)
- $\sum_{i\in V} x_{il}^k - \sum_{i\in V} x_{lj}^k = 0, \forall l \in C, \forall k \in K$ (5)
- $x_{i0}^k = 0, x_{n+1,i}^k = 0, \forall i \in V, \forall k \in K$ (6)
- $\sum_{j\in V} x_{0j}^k = 1, \forall k \in K$ (7)
- $\sum_{j\in V} x_{j,n+1} = 1, \forall k \in K$ (8)
- $e_i \sum_{j\in V} x_{ij}^k \leq y_i^k, \forall i \in V, \forall k \in K$ (9)
- $l_i \sum_{j\in V} x_{ij}^k \geq y_i^k, \forall i \in V, \forall k \in K$ (10)
- $x_{ij}^k [y_i^k + g_i + t_{ij}(y_i^k + g_i)] \leq y_j^k, \forall (i,j) \in A, \forall k \in K$ (11)
- $x_{ij}^k \in \{0,1\}, \forall (i,j) \in A, \forall k \in K$ (12)
- $y_i^k \in R, \forall i \in V, \forall k \in K$ (13)

Οι σχέσεις (1) και (2) καθορίζουν τον πρωτεύον και τον δευτερεύον αντικειμενικό στόχο, αντίστοιχα. Οι περιορισμοί καθορίζονται ως εξής:

- η χωρητικότητα δεν πρέπει να υπερβαίνεται (3)
- όλοι οι πελάτες πρέπει να εξυπηρετούνται (4)
- εάν το όχημα φτάσει σε έναν πελάτη, πρέπει να φύγει κι από αυτόν τον πελάτη (5)
- οι διαδρομές πρέπει να ξεκινούν και να τερματίζουν στην αποθήκη (6)
- κάθε όχημα φεύγει από την αποθήκη και επιστρέφει σε αυτήν ακριβώς μία φορά (7) και (8), αντίστοιχα
- οι χρόνοι εξυπηρέτησης πρέπει να ικανοποιούν τους χρόνους έναρξης του χρονικού παραθύρου (9) και λήξης (10) του χρονικού παραθύρου
- ο χρόνος έναρξης της εξυπηρέτησης πρέπει να επιτρέπει για χρόνο ταξιδιού μεταξύ πελατών (11)

Το είδος και ο χώρος τιμών των μεταβλητών απόφασης καθορίζεται από τις σχέσεις (12) και (13), αντίστοιχα.

3 ΑΛΓΟΡΙΘΜΟΙ ΕΠΙΛΥΣΗΣ ΠΡΟΒΛΗΜΑΤΩΝ ΔΡΟΜΟΛΟΓΗΣΗΣ ΟΧΗΜΑΤΩΝ

3.1 Αλγόριθμοι Κατασκευής μίας Αρχικής Λύσης

Οι Αλγόριθμοι Κατασκευής μίας Αρχικής Λύσης βασίζονται σε μία πολύ απλή αρχή, την αρχή της απληστίας. Σύμφωνα με την αρχή της απληστίας, ο αλγόριθμος ξεκινάει από μία μερική μη - εφικτή λύση και σε κάθε βήμα επιλέγει σύμφωνα με το τι είναι βέλτιστο για το συγκεκριμένο βήμα, έως ότου κατασκευαστεί μία εφικτή λύση. Η λύση, αυτή, θα είναι και η γενική λύση του προβλήματος.

Όσον αφορά τα προβλήματα Συνδυαστικής Βελτιστοποίησης, έστω n είσοδοι και ζητούμενο ένα υποσύνολο που να ικανοποιεί κάποιους περιορισμούς. Οποιοδήποτε υποσύνολο προκύπτει το οποίο ικανοποεί τους συγκεκριμένους περιορισμούς, τότε ονομάζεται εφικτή λύση. Η εφικτή, αυτή, λύση προϋποθέτει τη βελτιστοποίηση μίας αντικειμενικής συνάρτησης (είτε ελαχιστοποίηση είτε μεγιστοποίηση). Επειδή στα Προβλήματα Δρομολόγησης Οχημάτων, συνήθως, απαιτείται η ελαχιστοποίηση του κόστους, παρακάτω γίνεται λόγος για ελαχιστοποίηση της αντικειμενικής συνάρτησης και τοπικά ελάχιστα. Έτσι, μέσα από μία επαναληπτική διαδικασία η οποία πληροί τις παραπάνω προϋποθέσεις και στηρίζεται στην αρχή της απληστίας, προκύπτει μία αρχική λύση. Οι κυριότεροι Αλγόριθμοι Κατασκευής μίας Αρχικής Λύσης, είναι:

- Ο Αλγόριθμος του Πλησιέστερου Γείτονα
- Ο Αλγόριθμος των Εξοικονομήσεων των Clarke & Wright

3.2 Αλγόριθμοι Τοπικής Αναζήτησης

Οι Αλγόριθμοι Τοπικής Αναζήτησης βασίζονται στην αρχαιότερη ίσως μέθοδο βελτιστοποίησης, η οποία μπορεί πολύ απλά να περιγραφεί ως μέθοδος δοκιμής και σφάλματος. Αν και πρόκειται για μία τόσο απλή ιδέα, βρίσκει εφαρμογή στα περισσότερα προβλήματα Συνδυαστικής Βελτιστοποίησης, προσφέροντας ανέλπιστα καλά αποτελέσματα. Η λογική, που ακολουθεί, είναι η εξής: ξεκινώντας από μία εφικτή λύση και χρησιμοποιώντας μία «διαδικασία βελτίωσης»,

ο αλγόριθμος ψάχνει για μία καλύτερη λύση στη γειτονιά της αρχικής λύσης. Η διαδικασία επαναλαμβάνεται έως ότου προκύψει ένα τοπικό ελάχιστο, δηλαδή η υπάρχουσα λύση δεν βελτιώνεται άλλο.

Η επιτυχία των Αλγορίθμων Τοπικής Αναζήτησης εξαρτάται μεν από την ορθή επιλογή της γειτονιάς αναζήτησης, αλλά το σημαντικότερο είναι η μέθοδος η οποία χρησιμοποιείται.

3.2.1 Μέθοδος 1-0 Relocate

Η μέθοδος 1-0 Relocate αποτελείται από μία απλή διαγραφή ενός πελάτη από μία διαδρομή και την επανατοποθέτησή του σε μία άλλη διαδρομή, με στόχο να βελτιωθεί η τρέχουσα λύση. Η συγκεκριμένη μέθοδος προσπαθεί να βελτιώσει την ανάθεση των πελατών σε οχήματα. Παρόμοιας αρχής αλγόριθμοι, είναι:

- Μέθοδος 1-1 Exchange
- Μέθοδος 1-2 Exchange

3.2.2 Μέθοδος 2 - Opt

Η συγκεκριμένη μέθοδος πρακτικά αποτελείται από τη διαγραφή δύο ακμών και ύστερα την επανασύνδεση δύο μονοπατιών με διαφορετικό τρόπο, έτσι ώστε να προκύψει μία καινούρια διαδρομή. Πιο αναλυτικά, ξεκινώντας από μία διαδρομή και εν συνεχεία εξετάζοντας όλες τις πιθανές 2 - Opt κινήσεις, γίνεται προσπάθεια να βελτιωθεί η τρέχουσα διαδρομή. Εάν η νέα διαδρομή βελτιώνει την ήδη υπάρχουσα, τότε συντελούνται οι τρέχουσες αλλαγές και επιλέγται η αντίστοιχη διαδρομή. Η διαδικασία επαναλαμβάνεται, έως ότου δεν μπορεί να βρεθεί εκ νέου βελτίωση. Αξίζει να σημειωθεί, ότι η εν λόγω μέθοδος προσπαθεί να βελτιώσει τη δρομολόγηση των οχημάτων σε μία διαδρομή. Άλλοι αλγόριθμοι, που ανήκουν στην ίδια κατηγορία, είναι:

- Μέθοδος 3 - Opt
- Μέθοδος Or - Opt

3.3 Μεθευρετικοί Αλγόριθμοι

Οι Μεθευρετικοί Αλγόριθμοι αποτελούν τη λύση σε ένα πολύ κοινό πρόβλημα που εμφανίζουν οι Αλγόριθμοι Τοπικής Αναζήτησς. Πιο συγκεκριμένα, οι Αλγόριθμοι Τοπικής Αναζήτησης πολύ σύχνα συγκλίνουν και «εγκλωβίζονται» σε κάποιο τοπικό ελάχιστο, με αποτέλεσμα ο αλγόριθμος να μην μπορεί να βελτιώσει επιπλέον τη λύση. Οι Μεθευρετικοί Αλγόριθμοι, λοιπόν, έρχονται να δώσουν τη λύση σε αυτό το πρόβλημα. Ένας πολύ καλός διαχωρισμός γίνεται σύμφωνα με τις πόσες λύσεις χρησιμοποιούν. Έτσι, χωρίζονται σε αλγορίθμους που χρησιμοποιούν μία λύση και κάνουν αναζήτηση στη γειτονιά της λύσης αυτής, ή σε αλγορίθμους που χρησιμοποιούν έναν πληθυσμό από λύσεις και προσπαθούν να κάνουν αναζήτητση σε όλον τον χώρο λύσεων.

Οι αλγόριθμοι που χρησιμοποιούν μία λύση έχουν πολύ ισχυρές δυνατότητες εντατικοποίησης, ενώ οι αλγόριθμοι που χρησιμοποιούν έναν πληθυσμό λύσεων έχουν πολύ ισχυρές δυνατότητες εξερεύνησης. Όπως και να έχει, όμως, οι εν λόγω αλγόριθμοι μπορούν να χωριστούν σε τέσσερις βασικές κατηγορίες, κρίνοντας από τη μέθοδο που χρησιμοποιούν για να αποφύγουν το τοπικό ελάχιστο.

- Επαναληπτικές διαδικασίες που αρχίζουν από διαφορετικές αρχικές λύσεις, π.χ Διαδικασία Άπληστης Τυχαιοποιημένης Προσαρμοστικής Αναζήτησης (Greedy Randomized Adaptive Search Procedure - GRASP).

- Αλγόριθμοι που δέχονται γειτονικές κινήσεις που δεν βελτιώνουν τη λύση, π.χ Προσομοιωμένη Ανόπτηση (Simulated Annealing), Περιορισμένη Αναζήτηση (Tabu Search).

- Αλγόριθμοι που αλλάζουν τη γειτονιά αναζήτησης, π.χ Αλγόριθμος Μεταβλητής Γειτονιάς Αναζήτησης (Variable Neighborhood Search - VNS) , Αλγόριθμος Επέκτασης της Γειτονιάς Αναζήτησης (Expanding Neighborhood Search - ENS).

- Αλγόριθμοι που αλλάζουν την αντικειμενική συνάρτηση ή κάποια από τα δεδομένα του προβλήματος, π.χ Αλγόριθμος Καθοδηγούμενης Τοπικής Αναζήτησης (Guided Local Search).

3.3.1 Διαδικασία Άπληστης Τυχαιοποιημένης Προσαρμοστικής Αναζήτησης (Greedy Randomized Adaptive Search Procedure)

Η Διαδικασία Άπληστης Τυχαιοποιημένης Προσαρμοστικής Αναζήτησης (Greedy Randomized Adaptive Search Procedure - GRASP) αποτελεί μία επαναληπτική διαδικασία, όπου κάθε επανάληψη αντιστοιχεί και σε μία πιθανή λύση. Πιο συγκεκριμένα, κάθε επανάληψη αποτελείται από δύο φάσεις.

Στην πρώτη φάση, κατασκευάζεται μία αρχική λύση μέσω μίας τυχαιοποιημένης συνάρτησης απληστίας. Η συγκεκριμένη συνάρτηση χρησιμοποιεί μία λίστα περιορισμένων υποψηφίων (Restricted Candidate List - RCL), η οποία περιέχει ένα συγκεκριμένο αριθμό από τους καλύτερους επόμενους προορισμούς. Από τη λίστα, αυτή, επιλέγεται τυχαία ο επόμενος προορισμός. Η συγκεκριμένη φάση, σε «ελεύθερη μετάφραση», αποτελεί μία ανεπτυγμένη μορφή του Αλγορίθμου του Πλησιέστερου Γείτονα, η οποία διαθέτει στρατηγική και μία δόση τυχαιότητας. Στη δεύτερη φάση, η λύση, που κατασκευάστηκε προηγουμένως, υποβάλλεται σε μία διαδικασία τοπικής αναζήτησης με σκοπό να βελτιωθεί. Τελικά, το αποτέλεσμα του αλγορίθμου είναι η καλύτερη λύση που κατασκευάστηκε κατά τη διάρκεια όλων των επαναλήψεων.

Ο ψευδοκώδικας του αλγορίθμου, που περιγράφεται παραπάνω, είναι ο εξής:

Greedy Randomized Adaptive Search Procedure
Ανάγνωση Δεδομένων
Αρχικοποίηση
$c(s^*) = \infty$
Repeat
  κατασκευή μίας αρχικής λύσης s
  εφαρμογή τοπικής αναζήτησης στην s
  if $c(s) < c(s^*)$ ***then***
    $s^* = s$
  endif
Untill όσο το κριτήριο τερματισμού δεν ικανοποιείται
Επέστρεψε τη βέλτιστη λύση s^*

4 ΠΕΡΙΓΡΑΦΗ ΚΑΙ ΕΠΙΛΥΣΗ ΤΟΥ ΧΡΟΝΙΚΩΣ ΕΞΑΡΤΩΜΕΝΟΥ ΠΡΟΒΛΗΜΑΤΟΣ ΔΡΟΜΟΛΟΓΗΣΗΣ ΟΧΗΜΑΤΩΝ ΜΕ **GRASP**

Σκοπός του κεφαλαίου είναι να γίνει κατανοητό τόσο το πρόβλημα που επιλύεται όσο και η διαδικασία που ακολουθείται από την άντληση των δεδομένων έως και την εξαγωγή των τελικών αποτελεσμάτων. Γι΄ αυτόν τον λόγο εξηγείται αναλυτικά ο τρόπος διαμόρφωσης των σετ δεδομένων, καθώς και τα επιμέρους χαρακτηριστικά των βασικών μεταβλητών που χρησιμοποιούνται. Επιπλέον, για την εξαγωγή των τελικών αποτελεσμάτων απαιτείται η σύγκριση μίας αρχικής λύσης με τη λύση που προκύπτει από τον GRASP. Η αρχική, αυτή, λύση κατασκευάζεται με τον αλγόριθμο του Πλησιέστερου Γείτονα. Τόσο η υλοποίηση του GRASP όσο και του Πλησιέστερου Γείτονα παρουσιάζεται αναλυτικά στη συνέχεια, ενώ και οι δύο αλγόριθμοι έχουν αναπτυχθεί στο προγραμματιστικό περιβάλλον του Matlab.

4.1 Περιγραφή και Μοντελοποίηση του Προβλήματος

Το πρόβλημα, με το οποίο ασχολείται η παρούσα εργασία, πρόκειται για ένα τυπικό Χρονικώς Εξαρτώμενο Πρόβλημα Δρομολόγησης Οχημάτων χωρίς χρονικά παράθυρα. Η μέρα χωρίζεται σε τρία χρονικά διαστήματα, τα οποία ενδεικτικά είναι 1-300, 301-600 και 601-900. Τα χρονικά διαστήματα παραμένουν ίδια για κάθε περίπτωση που εξετάζεται. Ομοίως, ίδια παραμένει η μέγιστη επιτρεπτή χωρητικότητα των οχημάτων και ο μέγιστος επιτρεπτός χρόνος διαδρομής. Οι δύο, αυτές, μεταβλητές αποτελούν και τους δύο βασικούς περιορισμούς του προβλήματος. Φυσικά, προκειμένου να θεωρηθεί μία λύση εφικτή, πρέπει να έχουν επισκεφθεί όλοι οι κόμβοι ακριβώς μία φορά και το όχημα να έχει επιστρέψει στην αποθήκη μετά το τέλος της διαδρομής του. Περιορισμός για τον αριθμό των οχημάτων, που χρησιμοποιούνται, δεν υπάρχει στη συγκεκριμένη εργασία.

Ο αντικειμενικός στόχος του προβλήματος είναι η ελαχιστοποίηση του κόστους. Ο πίνακας αποστάσεων είναι και πίνακας αμοιβών. Επομένως, με την ελαχιστοποίηση της απόστασης επιτυγχάνεται και η ελαχιστοποίηση του κόστους.

Παρόμοια Προβλήματα Δρομολόγησης Οχημάτων έχουν επιλυθεί στο παρελθόν και ορισμένα από αυτά λαμβάνονται υπόψιν για την υλοποίηση της παρούσας εργασίας. Τα εν λόγω προβλήματα αναφέρονται στη βιβλιογραφία της εργασίας.

4.1.1 Περιγραφή των Βασικών Μεταβλητών

Αρχικά, στη μεταβλητή n αποθηκεύεται το πλήθος των κόμβων - πελατών. Ο πελάτης, υπ΄ αριθμόν, 1 συμβολίζει την αποθήκη. Έπειτα, στη μεταβλητή capacity αποθηκεύεται η μέγιστη επιτρεπτή χωρητικότητα των οχημάτων, ενώ στη μεταβλητή max_route_time αποθηκεύται ο μέγιστος επιτρεπτός χρόνος διαδρομής. Επιπλέον, στο διάνυσμα visited_nodes αποθηκεύονται οι κόμβοι οι οποίοι έχουν επισκεφθεί ή όχι, χρησιμοποιώντας τις τιμές 1 ή 0, αντίστοιχα. Στη συνέχεια, στους πίνακες demand, srv_time και dist αποθηκεύονται οι τιμές της ζήτησης, των χρόνων εξυπηρέτησης και των αποστάσεων για κάθε πελάτη, αντίστοιχα. Τέλος, στους πίνακες travel_time_a, tavel_time_b και travel_travel_c αποθηκεύονται οι τιμές των χρόνων μετάβασης ανάλογα με τη χρονική στιγμή της ημέρας. Δηλαδή, ο πίνακας travel_travel_a αντιστοιχεί στις χρονικές στιγμές 1-300, ο travel_time_b στις χρονικές στιγμές 301-600 και ο travel_time_c στις χρονικές στιγμές 601-900.

Οι χρόνοι μετάβασης παρουσιάζουν σημαντικές διαφορές ανάλογα με το χρονικό διάστημα της ημέρας. Γι΄ αυτό, είναι πολύ σημαντική η χρονική στιγμή που θα ξεκινήσει το όχημα από έναν κόμβο - πελάτη με προορισμό τον επόμενο. Η χρονική, αυτή, στιγμή αποθηκεύται στη μεταβλητή time_tracker, έτσι ώστε να επιλέγεται ο σωστός πίνακας χρόνων μετάβασης κάθε φορά αλλά και να ελέγχεται ο περιορισμός του μέγιστου επιτρεπτού χρόνου διαδρομής. Επίσης, πολύ σημαντική είναι η περίπτωση στην οποία ο χρόνος αναχώρησης και ο χρόνος άφιξης βρίσκονται ανάμεσα σε δύο διαφορετικά χρονικά διαστήματα. Για παράδειγμα, αν ένα όχημα αναχωρήσει από έναν κόμβο - πελάτη τη χρονική στιγμή 296 και φτάσει στον προορισμό του τη χρονική στιγμή 305, τότε για τον υπολογισμό του χρόνου μετάβασης λαμβάνονται υπόψιν 4 μονάδες από τον travel_time_a και οι υπόλοιπες από τον travel_time_b. Το ίδιο ισχύει και στη περίπτωση που ένα όχημα επιστρέφει στην αποθήκη και ο χρόνος μετάβασης αντιστοιχεί στην αλλαγή δύο χρονικών διαστημάτων. Στις παραπάνω περιπτώσεις, λοιπόν, ο χρόνος μετάβασης υπολογίζεται μέσω της μεταβλητής fixed ή depot_fixed και αποθηκεύεται στη depot_time, όταν πρόκειται για επιστροφή στην αποθήκη.

Έπειτα, στη μεταβλητή atm_capacity αποθηκεύεται η στιγμιαία χωρητικότητα του οχήματος, πράγμα που βοηθά στον έλεγχο του περιορισμού της χωρητι-

κότητας. Ο κόμβος που βρίσκεται κάθε δεδομένη χρονική στιγμή ο αλγόριθμος συμβολίζεται με atm_node, ο επόμενος πιθανός προορισμός με next_node και με depot συμβολίζεται ο κόμβος από τον οποίο πραγματοποιείται επιστροφή στην αποθήκη. Στο διάνυσμα route κατασκευάζεται και αποθηκεύεται κάθε πιθανή λύση, ενώ στη μεταβλητή route_cost υπολογίζεται και αποθηκεύεται το αντίστοιχο συνολικό κόστος. Επιπλέον, χρησιμοποιούνται οι βοηθητικές μεταβλητές stop και feasibility, οι οποίες παίρνουν τις τιμές 0 ή 1. Η stop χρησιμοποιείται ως κριτήριο τερματισμού, ενώ η feasibility χρησιμοποιείται ως κριτήριο εφικτότητας.

Όσα περιγράφονται παραπάνω ισχύουν τόσο για τον Πλησιέστερου Γείτονα όσο και για τον GRASP. Όμως, για την υλοποίηση του GRASP είναι απαραίτητη η χρήση μερικών επιπλέον μεταβλητών και πινάκων. Στην πρώτη φάση του αλγορίθμου, οι πίνακες possible_routes και possible_route_costs χρησιμοποιούνται για την αποθήκευση κάθε πιθανής λύσης που κατασκευάζεται μαζί με αντίστοιχο συνολικό της κόστος. Επίσης, κάθε φορά που μία λύση εκχωρείται στους εν λόγω πίνακες, η μεταβλητή a_it αυξάνεται κατά μία μονάδα. Έτσι, είναι κάθε στιγμή γνωστό το πλήθος των λύσεων που βρίσκονται μέσα στους συγκεκριμένους πίνακες. Ακολουθεί η δεύτερη φάση του αλγορίθμου, όπου οι παραπάνω λύσεις περνούν από την διαδικασία της τοπικής αναζήτησης. Στους πίνακες new_possible_routes και new_possible_route_costs αποθηκεύονται οι νέες λύσεις μαζί με τα αντίστοιχα κόστη που προκύπτουν μετά την πρώτη προσπάθεια βελτίωσης. Ύστερα, ακουλουθεί η δεύτερη και τελευταία προσπάθεια βελτίωσης για την κατασκευή ακόμα καλύτερων τελικών πιθανών λύσεων. Τα αποτελέσματα που προκύπτουν αποθηκεύονται στους πίνακες final_possible_routes και final_possible_route_costs. Τέλος, συγκρίνοντας όλες τις πιθανές τελικές λύσεις, εντοπίζεται η βέλτιστη η οποία εκχωρείται στο διάνυσμα global_best_route μαζί με το αντίστοιχο κόστος στη μεταβλητή global_best_route_cost.

Αξίζει να σημειωθεί ότι στην πρώτη φάση του GRASP, οι λύσεις που κατασκευάζονται μπορεί να είναι διαφορετικών διαστάσεων. Επειδή το γεγονός αυτό δημιουργεί πρόβλημα στην εκτέλεση του κώδικα, όλοι οι πίνακες που περιγράφονται στην προηγούμενη παράγραφο χρησιμοποιούνται και με το πρόθεμα «extended». Ουσιαστικά, λειτουργούν ως «ασφαλιστική δικλείδα» για την απρόσκοπτη λειτουργία του κωδικά, εξυπηρετώντας τους ίδιους ακριβώς σκοπούς που περιγράφονται παραπάνω. Επίσης, κάθε φορά που γίνεται χρήση των «extended» πινάκων, η μεταβλητή b_it αυξάνεται κατά μία μονάδα. Έτσι, είναι κάθε στιγμή γνωστό και το πλήθος των «extended» λύσεων. Επιπλέον, η βοηθητική μεταβλητή flag παίρνει την τιμή 1, ώστε κατά την τοπική αναζήτηση να γίνεται χρήση των αντίστοιχων πινάκων. Ομοίως, η βοηθητική μεταβλητή trigger πα-

ίρνει την τιμή 1, έτσι ώστε κατά τη σύγκριση των τελικών πιθανών λύσεων να λαμβάνονται υπόψιν και οι «extended» περιπτώσεις.

4.2 Περιγραφή και Διαμόρφωση των Δεδομένων

Τα δεδομένα προέρχονται από ένα παρόμοιο πρόβλημα δρομολόγησης οχημάτων, όπου αντικειμενικός στόχος είναι η μείωση των ρύπων που εκπέμπουν τα οχήματα κατά τις μετακινήσεις. Οι αποστάσεις βασίζονται σε πραγματικές μετρήσεις και αφορούν τυχαίες πόλεις του Ηνωμένου Βασιλείου. Εκτός από τις αποστάσεις, παρέχονται δεδομένα για τη μέγιστη επιτρεπτή χωρητικότητα των οχημάτων που αντιστοιχεί σε 3650 μονάδες προϊόντος, τους χρόνους εξυπηρέτησης των πελατών και τους χρόνους των χρονικών παραθύρων. Για τους χρόνους εξυπηρέτησης χρησιμοποιείται το ακέραιο μέρος της διαίρεσης με το 100, ενώ οι χρόνοι των χρονικών παραθύρων παραλείπονται, αφού το πρόβλημα της παρούσας εργασίας δεν περιορίζεται από χρονικά παράθυρα. Ο μέγιστος επιτρεπτός χρόνος διαδρομής ορίζεται από τον χρήστη και στην προκειμένη ισούται με 900 χρονικές μονάδες.

Τα δεδομένα, ανάλογα με τη διάσταση, μπορούν να χωριστούν σε ομάδες. Στη παρούσα εργασία εξετάζονται τρεις ομάδες οι οποίες αποτελούνται από 11, 51 και 101 κόμβους, αντίστοιχα. Για κάθε διάσταση υπάρχουν είκοσι διαθέσιμα σετ δεδομένων, τα οποία λαμβάνονται ως τετράδες. Επομένως, για κάθε διάσταση υπάρχουν πέντε διαθέσιμες τετράδες. Από το πρώτο σετ κάθε τετράδας προκύπτει ο πίνακας αποστάσεων, κρατώντας το ακέραιο μέρος της διαίρεσης με το 1000. Οι υπόλοιπες τρεις τετράδες αντιστοιχούν στους χρόνους μετάβασης, ομοίως κρατώντας το ακέραιο μέρος της διαίρεσης με το 1000, οι οποίοι είναι χωρισμένοι σε τρία χρονικά διαστήματα. Ενδεικτικά, το δεύτερο σετ αντιστοιχεί στο χρονικό διάστημα 1-300, το τρίτο στο 301-600 και το τέταρτο στο 601-900. Ακολουθώντας, λοιπόν, την παραπάνω διαδικασία, δημιουργούνται τελικά πέντε σετ δεδομένων για κάθε διάσταση τα οποία είναι κατάλληλα διαμορφωμένα για το πρόβλημα της εργασίας.

4.3 Κατασκευή Αρχικής Λύσης με τον Αλγόριθμο του Πλησιέστερου Γείτονα

Ο Πλησιέστερος Γείτονας αποτελεί ίσως τον απλούστερο αλγόριθμο για την κατασκευή μίας αρχικής λύσης. Σε κάθε βήμα επιλέγει σύμφωνα με το τι είναι

βέλτιστο εκείνη τη στιγμή, προϋποθέτοντας πάντα την ικανοποίηση των περιορισμών και την βελτιστοποίηση μίας αντικειμενικής συνάρτησης. Το γεγονός, αυτό, τον καθιστά αρκετά όμοιο με τον GRASP. Πρακτικά, θέτοντας τη λίστα περιορισμένων υποψηφίων ίση με 1 και παραλείποντας τη διαδικασία τοπικής αναζήτησης, είναι οι ίδιοι αλγόριθμοι. Με αυτήν την προσέγγιση, λοιπόν, έγινε η ανάπτυξη του Πλησιέστερου Γείτονα.

Στην αρχή του αλγορίθμου, πραγματοποιείται η άντληση των δεδομένων μέσα από ένα αρχείο Excel και ακολουθεί η αρχικοποίηση όλων των βασικών μεταβλητών. Επίσης, κατά την αρχικοποίηση, η αποθήκη ορίζεται ως εναρκτήριος κόμβος. Έπειτα, ξεκινά η επαναληπτική διαδικασία για την κατασκευή μίας εφικτής αρχικής λύσης.

Όσο η βοηθητική μεταβλητή stop έχει την τιμή 0, σημαίνει ότι το όχημα δεν έχει επισκεφθεί ακόμα όλους τους κόμβους. Επομένως, η διαδικασία μπορεί να συνεχιστεί με την εύρεση του πλησιέστερου κόμβου. Ο πλησιέστερος κόμβος υπολογίζεται μέσω της συνάρτησης «GreedyRandomization.m», όπου η λίστα των περιορισμένων υποψηφίων έχει προφανώς μέγεθος 1. Εκτός από τον πλησιέστερο κόμβο next_node, η συνάρτηση επιστρέφει και τη βοηθητική μεταβλητή stop. Όταν η stop πάρει την τιμή 1, σημαίνει ότι το όχημα έχει επισκεφθεί όλους τους κόμβους και η επαναληπτική διαδικασία πρέπει να τερματιστεί. Διαφορετικά, η διαδικασία συνεχίζεται με τον έλεγχο των περιορισμών για τον κόμβο που έχει επιλεχθεί.

Αρχικά, ελέγχεται ο περιορισμός της χωρητικότητας μέσω της συνάρτησης «CapacityConstraint.m». Κάθε φορά που καλείται αυτή η συνάρτηση, η ζήτηση του εξεταζόμενο κόμβου αθροίζεται στη στιγμιαία χωρητικότητα. Ύστερα, αν η στιγμιαία χωρητικότητα ξεπερνά το μέγιστο επιτρεπτό όριο, τότε η βοηθητική μεταβλητή feasibility παίρνει την τιμή 0. Διαφορετικά, παίρνει την τιμή 1 που σημαίνει ότι η διαδικασία μπορεί να συνεχιστεί με τον έλεγχο του επόμενου περιορισμού. Σε κάθε περίπτωση, όμως, η συνάρτηση επιστρέφει τη στιγμιαία χωρητικότητα atm_capacity, τον εξεταζόμενο κόμβο next_node και τη βοηθητική μεταβλητή feasibility.

Έπειτα, εφόσον η feasibility έχει την τιμή 1, η διαδικασία συνεχίζεται με τον έλεγχο του μέγιστου επιτρεπτού χρόνου διαδρομής μέσω της συνάρτησης «TimeConstraint.m». Κατά τη συνάρτηση, αυτή, σε πρώτη φάση προσδιορίζεται ποιος πίνακας χρόνων μετάβασης θα χρησιμοποιηθεί ανάλογα με την τιμή της time_tracker. Μετά, ελέγχεται αν ο χρόνος μετάβασης αντιστοιχεί στην αλλαγή δύο χρονικών διαστημάτων. Εάν αντιστοιχεί στην αλλαγή δύο χρονικών διαστημάτων, τότε υπολογίζεται μέσω της μεταβλητής fixed. Διαφορετικά,

παραμένει ως έχει σύμφωνα με τον πίνακα των χρόνων μετάβασης που ανήκει. Εν συνεχεία, τόσο ο χρόνος μετάβασης όσο και ο χρόνος εξυπηρέτησης του εξεταζόμενου κόμβου αθροίζονται στην time_tracker και υπολογίζεται ο χρόνος επιστροφής στην αποθήκη. Ομοίως, εάν ο χρόνος επιστροφής αντιστοιχεί στην αλλαγή δύο χρονικών διαστημάτων, τότε υπολογίζεται μέσω της μεταβλητής depot_fixed και εκχωρείται στη depot_time. Αλλιώς, η μεταβλητή depot_time παίρνει την τιμή του πίνακα των χρόνων μετάβασης που της αντιστοιχεί σύμφωνα με την time_tracker. Ύστερα, ελέγχεται η συνθήκη εφικτότητας, η οποία πρακτικά είναι η εξής: time_tracker $\leq$ max_route_time - depot_time. Εάν η συνθήκη ικανοποιείται, τότε η feasibility παίρνει την τιμή 1, ενώ στην αντίθετη περίπτωση παίρνει την τιμή 0. Εν τέλει, η συνάρτηση επιστρέφει τις feasibility, time_tracker καθώς και τον εξεταζόμενο κόμβο next_node.

Τέλος, εφόσον η τιμή της feasibility εξακολουθεί να είναι 1, σημαίνει ότι ο εξεταζόμενος κόμβος ικανοποιεί τους δύο περιορισμούς και η μετάβαση σε αυτόν κρίνεται εφικτή. Έτσι, εκχωρείται στο διάνυσμα route και το κόστος της μετάβασης προσμετράται στο συνολοκό κόστος route_cost. Επιπλέον, εκχωρείται στις μεταβλητές next_node και depot, ώστε η διαδικασία να συνεχιστεί από αυτόν. Κατ΄ αυτόν τον τρόπο, λοιπόν, κατασκευάζεται η αρχική λύση. Αξίζει να σημειωθεί, όμως, ότι οποιαδήποτε στιγμή η βοηθητική μεταβλητή feasibility πάρει την τιμή 0, πραγματοποιείται αναγκαστική επιστροφή στην αποθήκη και η διαδικασία συνεχίζεται από αυτήν. Το κόστος της επιστροφής σαφώς προσμετράται στο συνολικό, ενώ παράλληλα μηδενίζονται οι τιμές των atm_capacity και time_tracker.

Όταν η stop πάρει την τιμή 1, σημαίνει ότι έχουν επισκεφθεί όλοι οι κόμβοι και η επαναληπτική διαδικασία σταματά. Τότε πραγματοποιείται επιστροφή στην αποθήκη από τον τελευταίο κόμβο που έχει εκχωρηθεί στο διάνυσμα route, αφού μία εφικτή λύση προϋποθέτει την επιστροφή του οχήματος σε αυτήν μετά το τέλος της διαδρομής του. Φυσικά, το κόστος επιστροφής προσμετράται κι αυτό στο συνολικό. Αφού, λοιπόν, πραγματοποιηθεί κι αυτή η διαδικασία, η κατασκευή της αρχικής λύση έχει πλέον ολοκληρωθεί.

4.4 Υλοποίηση του αλγορίθμου GRASP

Ο GRASP, όπως έχει ανεφερθεί και στο προηγούμενο κεφάλαιο, αποτελεί μία ανεπτυγμένη μορφή του Πλησιέστερου Γείτονα. Αρκεί να τεθεί η λίστα περιορισμένων υποψηφίων μεγαλύτερη του 1, καθώς και να πραγματοποιηθεί η φάση της τοπικής αναζήτησης. Έτσι, λοιπόν, για την ανάπτυξη του GRASP

χρησιμοποιήθηκαν οι αλγόριθμοι τοπικής αναζήτησης 1-0 Relocate και 2 - Opt, ενώ για το μέγεθος της λίστας περιορσμένων υποψηφίων χρησιμοποιήθηκαν δύο διαφορετικές τιμές (2, 3) ανάλογα με το σετ των δεδομένων.

4.4.1 Φάση Κατασκευής Πιθανών Λύσεων

Η φάση της κατασκευής των πιθανών λύσεων είναι παρόμοια με την κατασκευή της αρχικής λύσης μέσω του Πλησιέστερου Γείτονα. Ουσιαστικά, η λογική και τα βήματα που ακολουθούνται για την κατασκευή της λύσης είναι τα ίδια. Δηλαδή, η λύση κατασκευάζεται στο διάνυσμα route και το αντίστοιχο κόστος υπολογίζεται στη μεταβλητή route_cost. Αφού ολοκληρωθεί η διαδικασία της κατασκευής, τα αποτελέσματα αποθηκεύονται στους πίνακες possible_routes και possible_route_costs. Φυσικά, εάν οι διαστάσεις των λύσεων διαφέρουν, τότε χρησιμοποιούνται και οι «extended» πίνακες. Επομένως, σε κάθε επανάληψη του GRASP, μία λύση εκχωρείται στους παραπάνω πίνακες και εν τέλει το πλήθος των πιθανών λύσεων είναι ίσο με τον αριθμό των επαναλήψεων που έχουν επιλεχθεί από τον χρήστη.

Η ειδοποιός διαφορά, ωστόσο, έγκειται στους ελέγχους των περιορισμών. Πιο συγκεκριμένα, κάθε πιθανή παραβίαση των περιορισμών δεν «συνοδεύεται» με απευθείας επιστροφή στην αποθήκη, αλλά ελέγχονται πρώτα οι υπόλοιποι κόμβοι που βρίσκονται μέσα στη λίστα περιορισμένων υποψηφίων. Οι εν λόγω κόμβοι συμβολίζονται με alternative_node_1 και alternative_node_2, ενώ προκύπτουν από τη συνάρτηση «GreedyRandomization.m». Επομένως, ανάλογα με το μέγεθος της λίστας περιορισμένων υποψηφίων, η συνάρτηση πλέον επιστρέφει τους ανάλογους εναλλακτικούς κόμβους, καθώς και το πλήθος τους restricted_cols.

Στον έλεγχο της χωρητικότητας, λοιπόν, εάν η ζήτηση του εξεταζόμενου κόμβου next_node δεν μπορεί να εξυπηρετηθεί, τότε ελέγχονται οι εναλλακτικοί κόμβοι. Πρώτα, ελέγχεται ο alternative_node_1 και εφόσον μπορεί να εξυπηρετηθεί, παίρνει τη θέση του next_node. Επιπλέον, η atm_capacity παίρνει κι αυτή την ανάλογη τιμή. Ο alternative_node_2 ελέγχεται μόνο στην περίπτωση που έχουν απορριφθεί οι δύο προηγούμενοι και εφόσον μπορεί να εξυπηρετηθεί, τότε οι next_node και atm_capacity διαμορφώνονται αντιστοίχως. Στην περίπτωση, όμως, που κανείς από τους τρεις δεν μπορεί να εξυπηρετηθεί, πραγματοποιείται αναγκαστική επιστροφή στην αποθήκη και ακολουθούν οι ανάλογες ενέργειες που περιγράφονται στον Πλησιέστερο Γείτονα.

Αξίζει να σημειωθεί ότι ακόμα και στην περίπτωση που ένας κόμβος μπορεί

να εξυπηρετηθεί, οι έλεγχοι για τους υπόλοιπους πραγματοποιούνται κανονικά και τα αποτελεσμάτα αποθηκεύονται στις ανάλογες μεταβλητές. Πιο αναλυτικά, χρησιμοποιούνται οι βοηθητικές μεταβλητές alternative_atm_capacity_1, help_1 και alternative_atm_capacity_2, help_2. Οι μεταβλητές help_1, help_2 παίρνουν την τιμή 1, όταν ο αντίστοιχος κόμβος μπορεί να εξυπηρετηθεί και την τιμή 0, όταν δεν μπορεί. Το γεγονός, αυτό, βοηθά σε κάθε πιθανή παραβίαση του επόμενου περιορισμού, ώστε να είναι γνωστό ποιος εναλλακτικός κόμβος έχει νοήμα να ελεγχθεί και ποιος όχι. Επίσης, αξίζει να σημειωθεί ότι το πλήθος των εναλλακτικών κόμβων δεν είναι σταθερό σε κάθε επανάληψη. Γι΄ αυτόν τον λόγο, χρησιμοποιείται η μεταβλητή restricted_cols, που πρακτικά αποθηκεύει το πλήθος των εναλλακτικών κόμβων σε κάθε επανάληψη, ώστε να εκτελείται απρόσκοπτα ο κώδικας. Τελικά, η συνάρτηση «CapacityConstraint.m» επιστρέφει πλέον και τις βοηθητικές μεταβλητές που περιγράφονται παραπάνω.

Στον έλεγχο του μέγιστου επιτρεπτού χρόνου διαδρομής, ομοίως, ελέγχονται οι εναλλακτικοί κόμβοι στην περίπτωση που ο εξεταζόμενος κόμβος next_node δεν μπορεί να εξυπηρετηθεί. Στο σημείο, αυτό, χρησιμοποιούνται οι βοηθητικές μεταβλητές alternative_atm_capacity_1, help_1 και alternative_atm_capacity_2, help_2. Έλεγχος για κάποιον εναλλακτικό κόμβο πραγματοποιείται μόνο στην περίπτωση που η αντίστοιχη μεταβλητή help έχει την τιμή 1. Δεδομένης της παραπάνω προϋπόθεσης, λοιπόν, ελέγχεται πρώτα ο alternative_node_1 και εφόσον μπορεί να εξυπηρετηθεί, παίρνει τη θέση του next_node. Επιπλέον, οι τιμές των atm_capacity και time_tracker διαμορφώνονται σύμφωνα με τα δεδομένα που αντιστοιχούν στον συγκεκριμένο κόμβο. Ο alternative_node_2 ελέγχεται μόνο στην περίπτωση που έχουν απορριφθεί οι δύο προηγούμενοι και εφόσον μπορεί να εξυπηρετηθεί, τότε εκτελούνται οι αντίστοιχες ενέργειες σύμφωνα με τα δικά του δεδομένα. Φυσικά, στην περίπτωση που κανείς κόμβος δεν μπορεί να εξυπηρετηθεί, πραγματοποιείται αναγκαστική επιστροφή στην αποθήκη η οποία «συνοδευέται» με τις ανάλογες ενέργειες. Τελικά, η συνάρτηση «TimeConstraint.m» επιστρέφει πλέον και τη στιγμιαία χωρητικότητα atm_capacity.

Κάθε φορά που ελέγχεται ένας εναλλακτικός κόμβος που μπορεί να εξυπηρετηθεί, η τιμή της feasibility, όπως είναι λογικό, παίρνει την τιμή 1. Εφόσον η feasibility έχει την τιμή 1 και στο τελευταίο στάδιο, που αφορά την εκχώρηση του εξεταζόμενοι κόμβου στη διαδρομή και τον υπολογισμό του συνολικού κόστους, τότε πραγματοποιούνται οι ίδιες ενέργειες που περιγράφονται και στον Πλησιέστερο Γείτονα. Ομοίως, οι ίδιες ενέργειες πραγματοποιούνται και στην περίπτωση που έχει την τιμή 0. Συνεπώς, η κατασκευή της πιθανής λύσης πραγματοποιείται κατά τον ίδιο τρόπο. Αφού ολοκληρωθεί, όμως, η κατασκευή της

πιθανής λύσης, γίνεται έλεγχος της διάστασής της. Έτσι, κάθε πιθανή λύση αποθηκεύεται στον σωστό πίνακα, ενώ οι βοηθητικές μεταβλητές a_it, b_it, flag και trigger διαμορφώνονται αναλόγως. Εν τέλει, αρχικοποιούνται όλες οι τιμές των βασικών μεταβλητών, ώστε να ξεκινήσει η καινούρια επανάληψη και κατ΄ επέκταση η καινούρια πιθανή λύση από την αρχή.

4.4.2 Φάση Τοπικής Αναζήτησης

Η φάση της τοπικής αναζήτησης ξεκινά με τον αλγόριθμο 1-0 Relocate, ο οποίος υλοποιείται μέσω της συνάρτησης «Relocate.m» ή «ExtendedRelocate.m». Η πρώτη συνάρτηση χρησιμοποιείται για τις πιθανές λύσεις που έχουν εκχωρηθεί στους πίνακες possible_routes και possible_route_costs, ενώ η δεύτερη συνάρτηση χρησιμοποιείται για τις πιθανές λύσεις που έχουν εκχωρηθεί στους αντίστοιχους «extended» πίνακες. Επίσης, η δεύτερη συνάρτηση καλείται μόνο στην περίπτωση που η βοηθητική μεταβλητή flag έχει πάρει την τιμή 1. Έτσι, σε κάθε επανάληψη καλείται η κατάλληλη συνάρτηση για την βελτίωση της πιθανής λύσης. Ο διαχωρισμός, αυτός, έγκειτα στις διαφορετικές διαστάσεις των πιθανών λύσεων που μπορεί να προκύψουν κατά την πρώτη φάση του GRASP. Επομένως, οι δύο συναρτήσεις εκτελούν ακριβώς την ίδια διαδικασία. Γι΄ αυτόν τον λόγο, αρκεί να γίνει ανάλαυση μόνο για τη μία συνάρτηση. Παρακάτω αναλύεται η «Relocate.m». Η μόνη διαφορά, που εντοπίζεται στη συνάρτηση «ExtendedRelocate.m», είναι η χρήση των «extended» πινάκων κατά την είσοδο και την έξοδο.

Αρχικά, η πιθανή λύση, που εισέρχεται στη συνάρτηση, εκχωρείται στα διανύσματα new_best_route και help_route. Επιπλέον, το αντίστοιχο κόστος εκχωρείται στη μεταβλητή new_best_route_cost. Έπειτα, ξεκινά η επαναληπτική διαδικασία. Το διάνυσμα, που χρησιμοποιείται για την υλοποίηση των αλλαγών κατά τη διάρκεια του αλγορίθμου, είναι το help_route. Έτσι, σε κάθε επανάλήψη επιλέγεται ένας τυχαίος κόμβος, ο οποίος δεν μπορεί να είναι σε καμία περίπτωση ο εναρκτήριος. Στη συνέχεια, ο κόμβος, αυτός, διαγράφεται από την διαδρομή και επανατοποθετείται σε μία άλλη τυχαία θέση, η οποία πάλι δεν μπορεί να αντιστοιχεί στην αφετηρία της διαδρομής. Για παράδειγμα, για μία λύση της μορφής [1 2 3 4 1 5 6 7 1], κάποιες πιθανές αλλαγές μπορεί να είναι οι εξής: [1 2 4 1 3 5 6 7 1, [1 2 7 3 4 1 5 6 1], [1 2 3 6 4 1 5 7 1], κ.ο.κ. Κατ΄ αυτόν τον τρόπο, προκύπτει μία νέα λύση, η οποία μένει να ελεγχθεί για το αν βελτιώνει το κόστος και αν ικανοποιεί τους περιορισμούς. Γι΄ αυτό, υπολογίζεται το συνολικό της κόστος total_route_cost και συγκρίνεται με το στιγμιαία καλύτερο κόστος, το οποίο κάθε φορά αποθηκεύεται στη μετα-

βλητή new_best_route_cost. Εφόσον είναι μικρότερο, η διαδικασία συνεχίζεται με τον έλεγχο των περιορισμών. Διαφορετικά, απορρίπτεται και το διάνυσμα help_route αποκτά και πάλι τη στιγμιαία καλύτερη λύση new_best_route.

Πρώτα, ελέγχεται ο περιορισμός της χωρητικότητας. Ο έλεγχος πραγματοποιείται για κάθε διαδρομή ξεχωριστά, αθροίζοντας τη ζήτηση των κόμβων που βρίσκονται σε αυτή. Για παράδειγμα, για τη λύση της μορφής [1 2 3 4 1 5 6 7 1], θα γίνει έλεγχος των διαδρομών [1 2 3 4 1] και [1 5 6 7 1]. Το άθροισμα της ζήτησης των κόμβων 2, 3 και 4 πρέπει να είναι μικρότερο ή ίσο της μέγιστης επιτρεπτής χωρητικότητας, ενώ το ίδιο ισχύει και για τους κόμβους 5, 6 και 7. Εφόσον δεν υπάρχει κάποια παραβίαση του περιορισμού, η διαδικασία συνεχίζεται με τον έλεγχο για τον μέγιστο επιτρεπτό χρόνο διαδρομής. Διαφορετικά, η λύση απορρίπτεται και γίνεται ξανά ανάθεση του new_best_route στο help_route.

Ύστερα, ελέγχεται ο περιορισμός για τον μέγιστο επιτρεπτό χρόνο διαδρομής. Ομοίως, ο έλεγχος πραγματοποιείται για κάθε διαδρομή ξεχωριστά. Αυτή τη φορά, όμως, αθροίζονται οι χρόνοι μετάβασης και εξυπηρέτησης από κόμβο σε κόμβο. Η συνθήκη που πρέπει να ικανοποιείται είναι η ίδια με αυτήν της συνάρτησης «TimeConstraint.m», δηλαδή η help_time $\leq$ max_route_time - depot_time. Η help_time έχει την ίδια λειτουργία με τη time_tracker, απλά χρησιμοποιείται με άλλο όνομα για να εκτελείται απρόσκοπτα ο κώδικας. Επίσης, ο υπολογισμός των χρόνων μετάβασης γίνεται ακριβώς με τον ίδιο τρόπο που περιγράφεται στη συνάρτηση «TimeConstraint.m». Στη συνέχεια, εφόσον ικανοποιείται κι αυτός ο περιορισμός, η λύση κρίνεται εφικτή και εκχωρείται στο διάνυσμα new_best_route. Παράλληλα, το αντίστοιχο κόστος εκχωρείται στη μεταβλητή new_best_route_cost. Στην περίπτωση, όμως, που δεν ικανοποιείται ο περιορισμός, η λύση απορρίπτεται και ακολουθείται η ανάλογη διαδικασία που προβλέπεται κατά την απόρριψη.

Τέλος, η συνάρτηση επιστρέφει και αποθηκεύει τα παραπάνω αποτελέσματα στους πίνακες new_possible_routes και new_possible_route_costs. Η επαναληπτική διαδικασία σταματά, όταν ολοκληρωθεί ο αριθμός των επαναλήψεων που έχει ορίσει ο χρήστης.

Η φάση της τοπικής αναζήτησης συνεχίζεται με τον αλγόριθμο 2 - Opt, ο οποίος υλοποιείται μέσω της συνάρτησης «Opt.m» ή «ExtendedOpt.m». Ομοίως, η συνάρτηση «ExtendedOpt.m» αντιστοιχεί στις «extended» πιθανές λύσεις και καλείται μόνο στην περίπτωση που η βοηθητική μεταβλητή flag έχει πάρει την τιμή 1. Σαφώς, οι δύο συναρτήσεις εκτελούν ακριβώς την ίδια διαδικασία, με τη μόνη διαφορά να εντοπίζεται και πάλι στην είσοδο και την έξοδο των

πινάκων που χρησιμοποιούνται. Γι΄ αυτό, αρκεί να αναλυθεί η μία εκ των δύο συναρτήσεων. Στην προκειμένη αναλύεται η «Opt.m».

Αρχικά, η πιθανή λύση , που εισέρχεται στη συνάρτηση, αντλείται από τους πίνακες new_possible_routes και new_possible_route_costs. Έπειτα, εκχωρείται στα διανύσματα final_possible_route και help_route_2, ενώ το αντίστοιχο κόστος εκχωρείται στη μεταβλητή final_possible_route_cost. Στη συνέχεια, ξεκινά η επαναληπτική διαδικασία. Το help_route_2 χρησιμοποιείται για την υλοποίηση όλων των πιθανών 2 - Opt κινήσεων. Έτσι, σε κάθε επανάληψη επιλέγονται δύο τυχαίοι κόμβοι που ανήκουν στην ίδια διαδρομή και ορίζουν ένα μονοπάτι. Οι κόμβοι, αυτοί, δεν μπορούν να είναι σε καμία περίπτωση η αποθήκη. Ύστερα, το συγκεκριμένο μονοπάτι αντιστρέφεται και δημιουργείται μία διαφορετική αλληλουχία κόμβων. Η διαδικασία, αυτή, επαναλαμβάνεται για όλες τις διαδρομές της εξεταζόμενης λύσης. Για παράδειγμα, για μία λύση της μορφής [1 2 3 4 5 1 6 7 8 9 1], πρώτα γίνοται αλλαγές στη διαδρομή [1 2 3 4 5 1] και εν συνεχεία στη διαδρομή [1 6 7 8 9 1]. Ενδεικτικά, οι πιθανές 2 - Opt κινήσεις μπορούν να είναι: [1 2 5 4 3 1], [1 9 8 7 6 1], [1 5 4 3 2 1], κ.ο.κ. Επομένως, προκύπτει και μία νέα λύση, της οποίας μένει να ελεγχθεί το κόστος και η συμμόρφωσή της με τους περιορισμούς.

Αρχικά, υπολογίζεται το συνολικό της κόστος στη μεταβλητή total_route_cost_2 και συγκρίνεται με το στιγμιαία καλύτερο final_possible_route_cost . Εφόσον είναι μικρότερο, η διαδικασία συνεχίζεται με τον έλεγχο για τον μέγιστο επιτρεπτό χρόνο διαδρομής. Διαφορετικά, η λύση απορρίπτεται και το help_route_2 αποκτά και πάλι τη στιγμιαία καλύτερη λύση final_possible_route. Αξίζει να σημειωθεί ότι ο περιορισμός της χωρητικότητας δεν χρειάζεται να ελεγχθεί, αφού οι 2 - Opt κινήσεις πραγματοποιούνται αυστηρά μέσα στις ίδιες διαδρομές. Συνεπώς, η συνολική ζήτηση κάθε διαδρομής δεν γίνεται να αλλάξει. Όσον αφορά τον περιορισμό του μέγιστου επιτρεπτού χρόνου διαδρομής, ελέγχεται όπως ακριβώς και στον 1-0 Relocate.

Τέλος, αφού ολοκληρωθούν οι επαναλήψεις που έχει ορίσει ο χρήστης, τα αποτελέσματα των final_possible_route, final_possible_route_cost επιστρέφουν και αποθηκεύονται στους πίνακες final_possible_routes, final_possible_route_costs, αντίστοιχα.

4.4.3 Εύρεση Βέλτιστης Λύσης

Η βέλτιστη λύση προκύπτει μέσα από μία διαδικασία σύγκρισης όλων των τελικών πιθανών λύσεων που έχουν κατασκευαστεί. Αρχικά, μέσω σύγκρισης

εντοπίζεται η καλύτερη λύση από τους πίνακες final_possible_routes και final_possible_route_costs. Έπειτα, αποθηκεύεται στο διάνυσμα global_best_route και το αντίστοιχο κόστος στη μεταβλητή global_best_route_cost. Στη συνέχεια, εάν η βοηθητική μεταβλητή trigger έχει πάρει την τιμή 1, τότε εξετάζονται και οι πίνακες extended_final_possible_routes, extended_final_possible_route_costs. Η μεταβλητή global_best_route_cost συγκρίνεται με τα κόστη του πίνακα extended_final_possible_route_costs και κάθε φορά αποθηκεύεται το καλύτερο κόστος μαζί με την αντίστοιχη λύση. Εν τέλει, ο κώδικας επιστρέφει τα βέλτιστα αποτελέσματα global_best_route και global_best_route_cost.

4.5 Ψευδοκώδικας

Greedy Randomized Adaptive Search Procedure
Ανάγνωση Δεδομένων
Αρχικοποίηση
Repeat
  κατασκευή μίας αρχικής λύσης route
  εφαρμογή 1-0 Relocate στην route
  if $c(new\_route) < c(route)$ ***then***
    $route = new\_route$
  endif
  εφαρμογή 2 - Opt στην route
  if $c(new\_route) < c(route)$ ***then***
    $route = new\_route$
  endif
  $final = [final; route]$
Untill όσο το κριτήριο τερματισμού δεν ικανοποιείται
$global\_best\_route = final()$
$global\_best\_route\_cost = c(final())$
Do while *μέχρι να εξεταστούν όλα τα στοιχεία του final*
  if $c(final()) < global\_best\_route\_cost$ ***then***
    $global\_best\_route = final()$
    $global\_best\_route\_cost = c(final())$
  endif
End do
Επέστρεψε τη βέλτιστη λύση *global_best_route, global_best_route_cost*

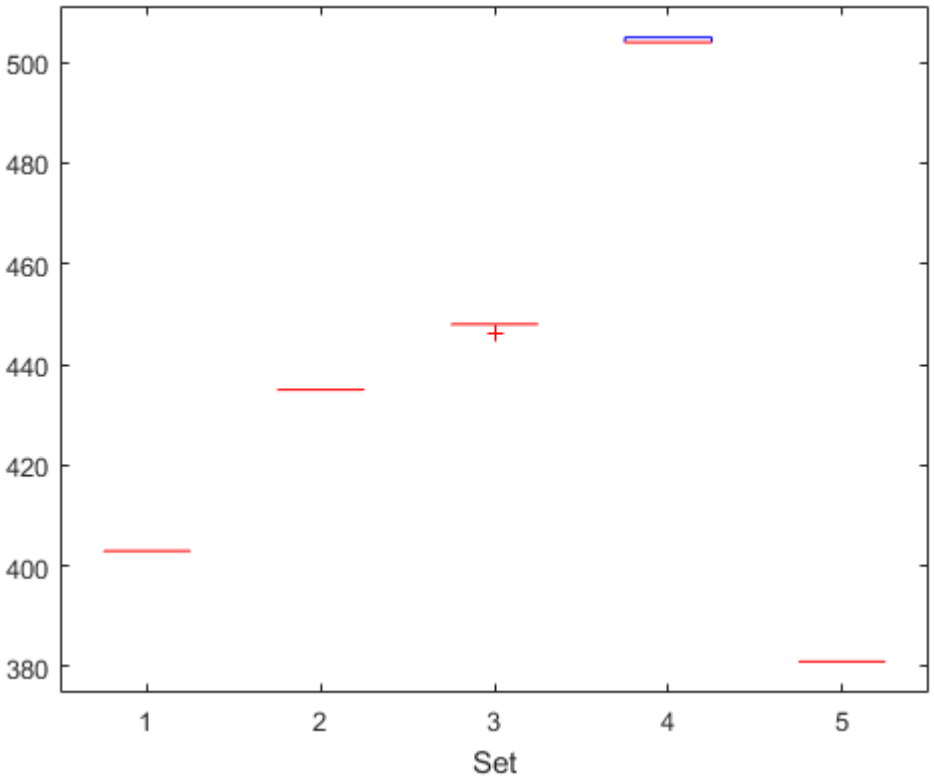
5 ΕΞΑΓΩΓΗ ΑΠΟΤΕΛΕΣΜΑΤΩΝ ΚΑΙ ΣΧΟΛΙΑΣΜΟΣ

Στο παρακάτω κεφάλαιο γίνεται σύγκριση και σχολιασμός των αποτελεσμάτων που προέκυψαν κατά την υλοποίηση του Πλησιέστερου Γείτονα και του GRASP. Τα αποτελέσματα χωρίζονται σε τρεις ενότητες, όπου η κάθε μία αντιστοιχεί και σε μία διαφορετική διάσταση των σετ δεδομένων. Επιπλέον, τα αποτελέσματα του GRASP παρουσιάζουν τη βέλτιστη τιμή, που προέκυψε, ύστερα από δέκα διαφορετικά «τρεξίματα» του αλγορίθμου.

5.1 Σετ Δεδομένων 11 Κόμβων

Στην παρακάτω ενότητα, ο GRASP πραγματοποίησε 200 επαναλήψεις, πράγμα που σημαίνει ότι κατασκευάστηκαν 200 πιθανές λύσεις. Το μέγεθος της λίστας περιορισμένων υποψηφίων ορίστηκε ίσο με 3, ενώ οι αλγόριθμοι τοπικής αναζήτησης πραγματοποίησαν 50 επαναλήψεις έκαστος. Επίσης, όλες οι πιθανές λύσεις ήταν της ίδιας διάστασης, με αποτέλεσμα να μην γίνει χρήση των «extended» πινάκων. Παρακάτω, παρουσιάζονται τα συγκεντρωτικά αποτέλεσματα των πέντε σετ δεδομένων, ύστερα από τα δέκα διαφορετικά «τρεξίματα», καθώς και το αντίστοιχο θηκόγραμμα με την ελάχιστη, την ενδιάμεση, τη μέγιστη τιμή και τις τιμές του πρώτου και του τρίτου τεταρτημορίου.

Set	Πλ. Γείτονας	Βελτ. GRASP	Μ.Ο GRASP	Τυπ. Αποκλ. GRASP
1	447	403	403	0.0000
2	560	435	435	0.0000
3	532	446	447.6	0.8000
4	552	504	504.3	0.4583
5	466	381	381	0.0000



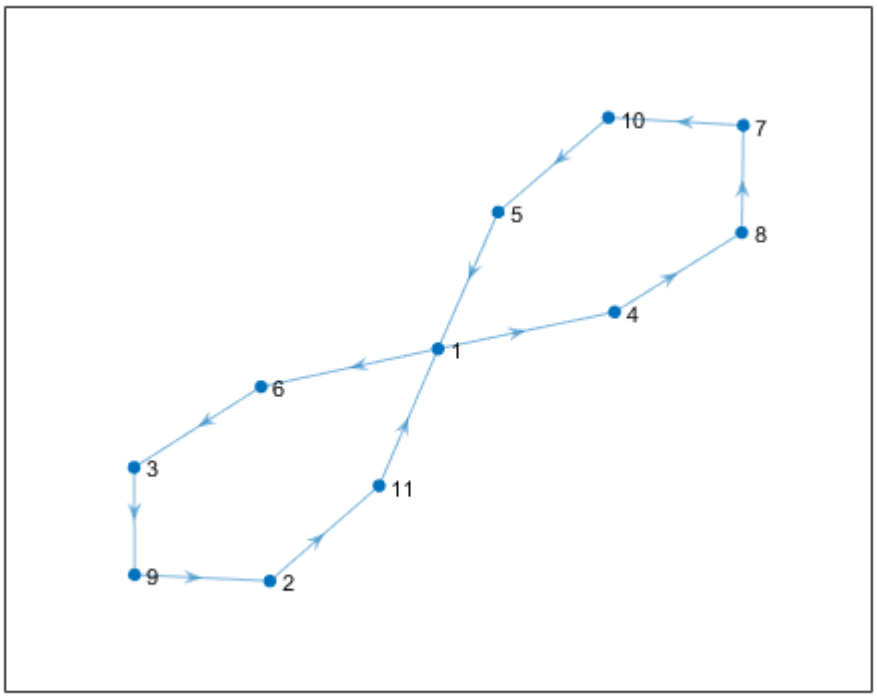
Στη συνέχεια, παρουσιάζονται συγκεντρωτικά και σχηματικά τόσο οι λύσεις του Πλησιέστερου Γείτονα όσο και οι βέλτιστες λύσεις του GRASP.

Πλησιέστερος Γείτονας (RCL = 1)		
set	route	route_cost
1	1 1	447
2	1 1	560
3	1 1	532
4	1 1	552
5	1 1	466

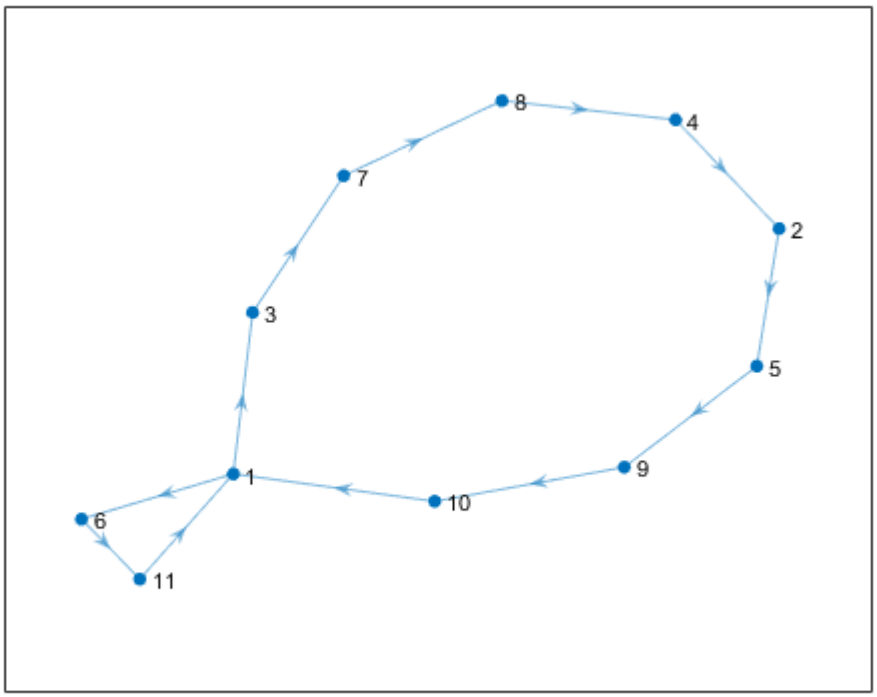
GRASP (RCL = 3)		
set	route	route_cost
1	1 1	403
2	1 1	435
3	1 1	446
4	1 1	504
5	1 1	381

Για τον **Πλησιέστερο Γείτονα:**

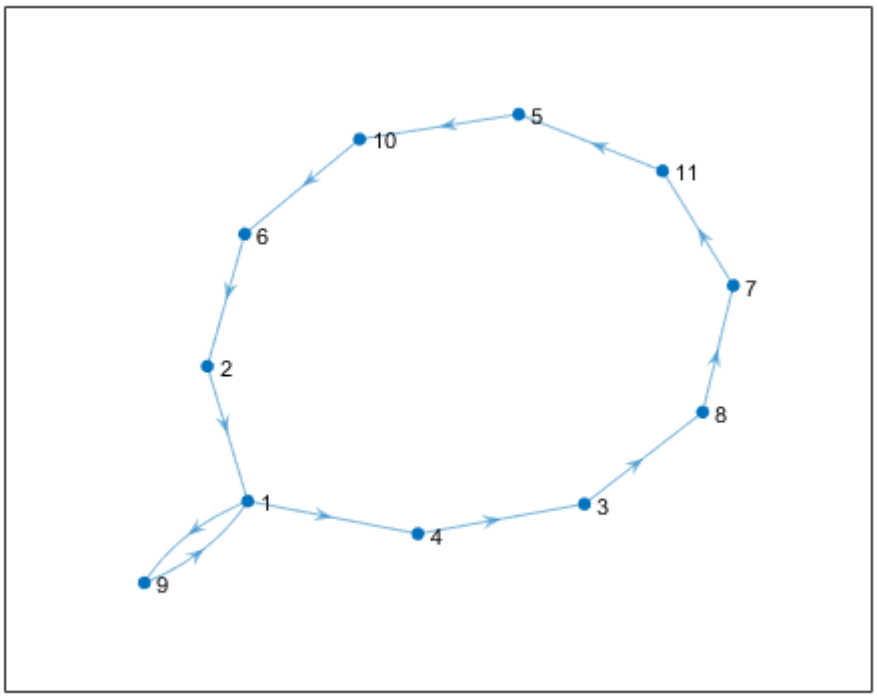
- **Σετ 1:** 1 6 3 9 2 11 1 4 8 7 10 5 1



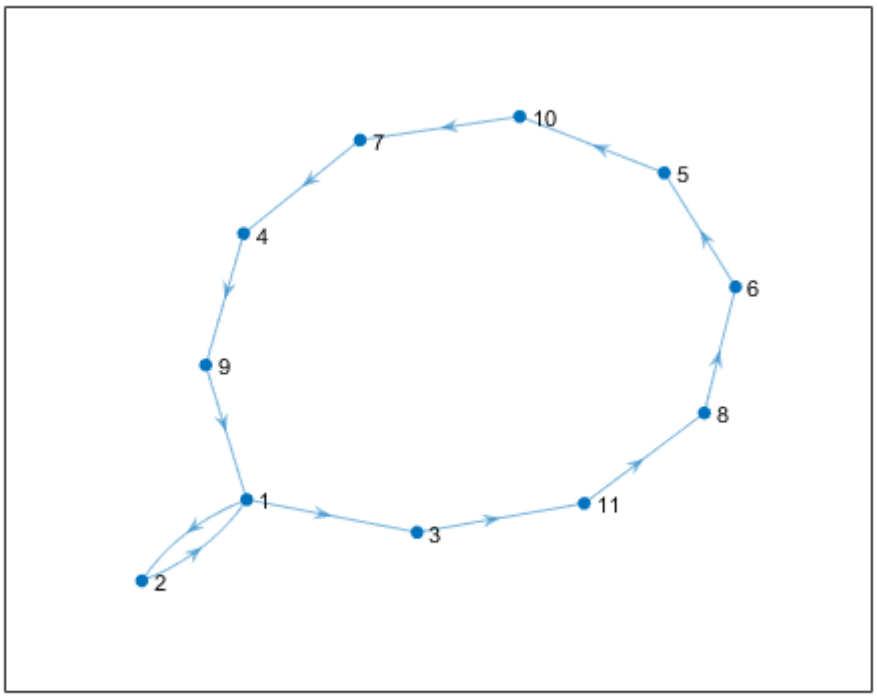
- **Σετ 2:** 1 3 7 8 4 2 5 9 10 1 6 11 1



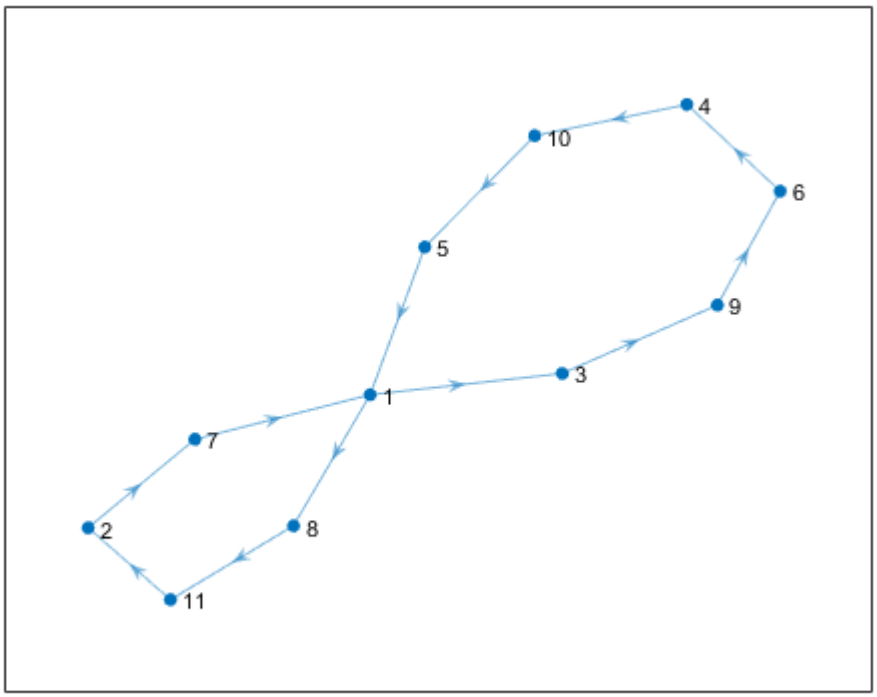
- **Σετ 3:** 1 4 3 8 7 11 5 10 6 2 1 9 1



- **Σετ 4:** 1 3 11 8 6 5 10 7 4 9 1 2 1

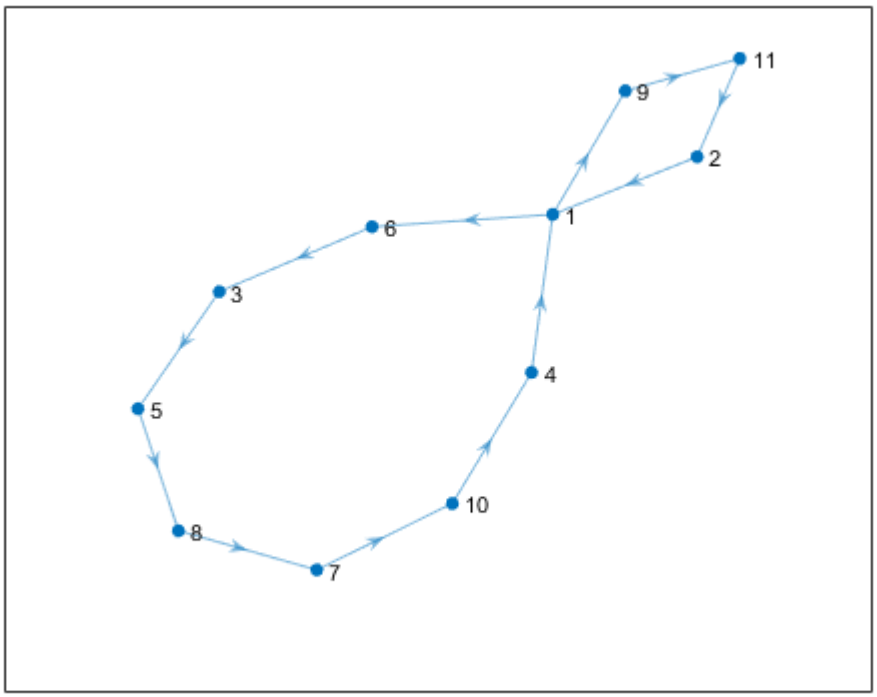


- **Σετ 5:** 1 8 11 2 7 1 3 9 6 4 10 5 1

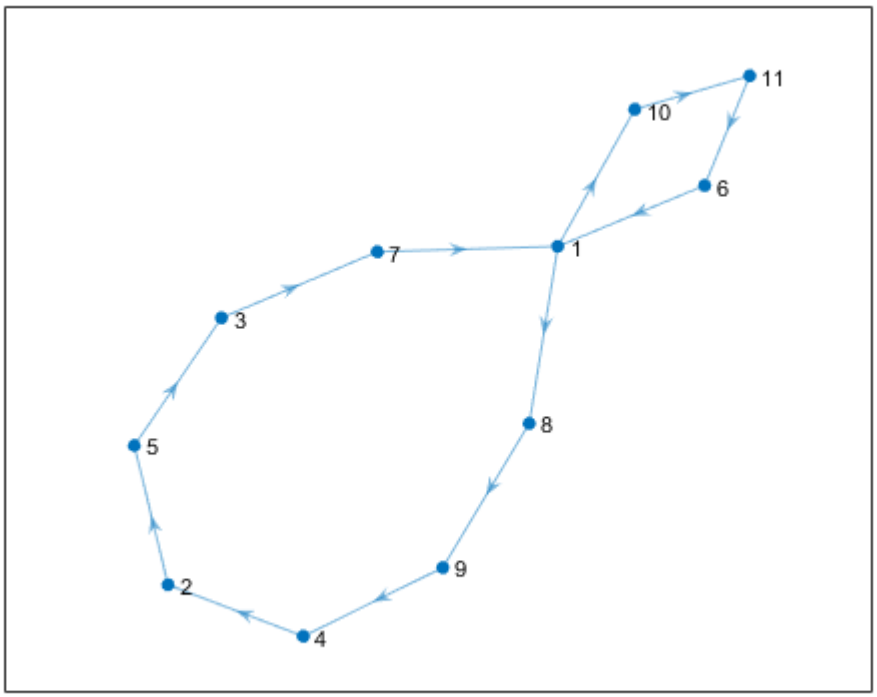


Για τον **GRASP:**

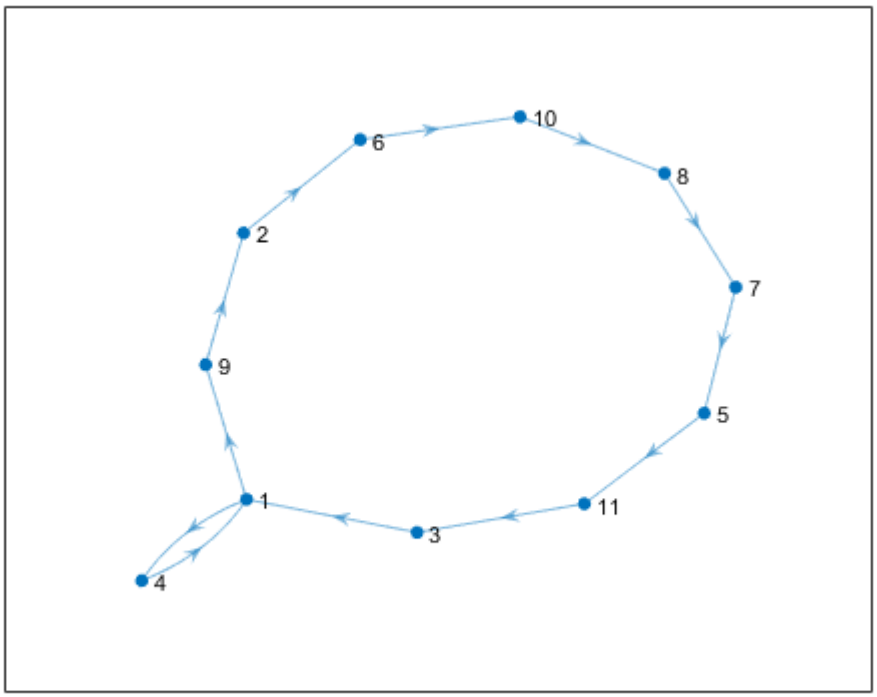
- **Σετ 1:** 1 6 3 5 8 7 10 4 1 9 11 2 1



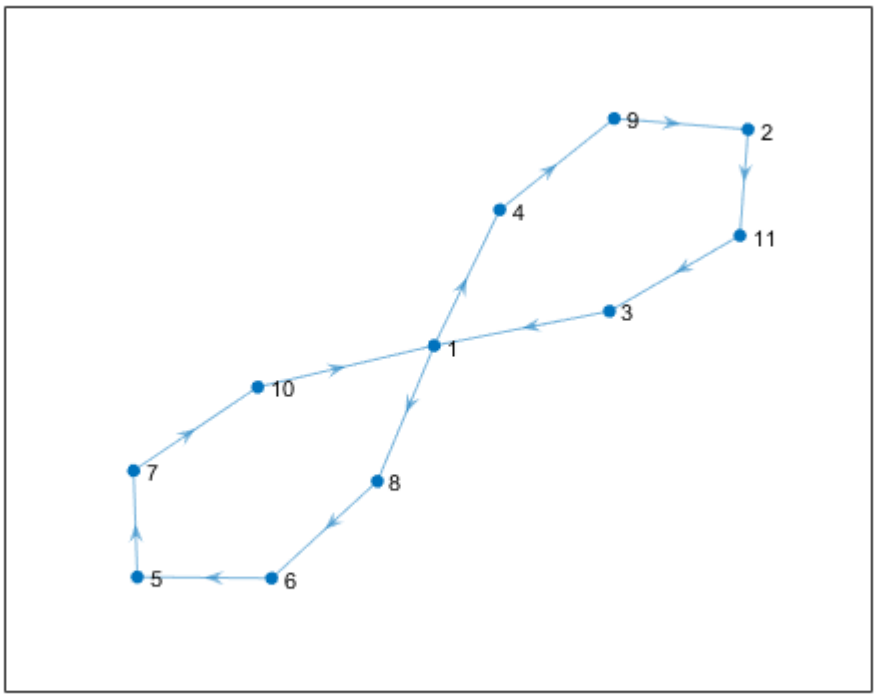
- **Σετ 2:** 1 8 9 4 2 5 3 7 1 10 11 6 1



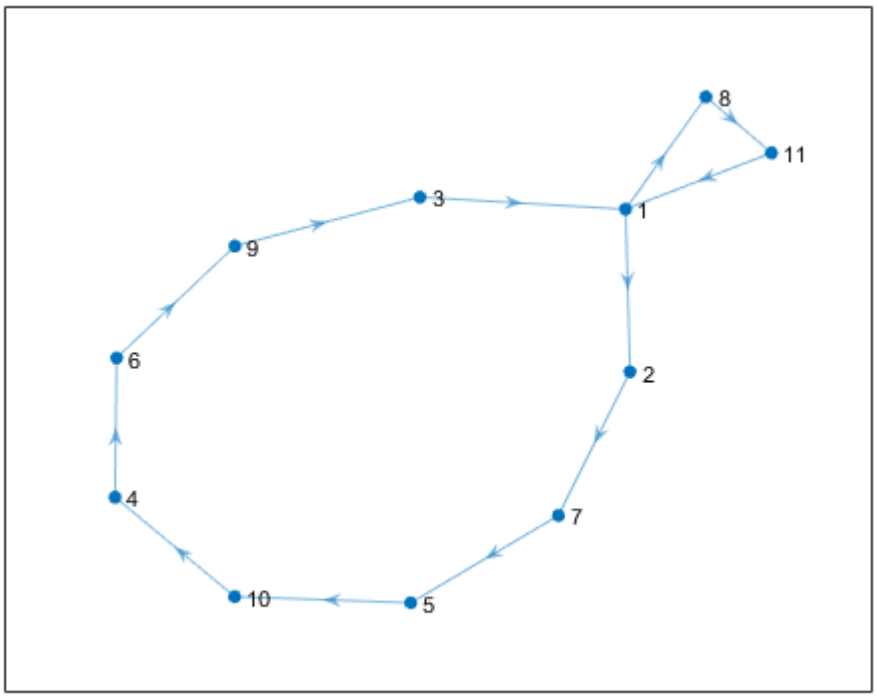
- **Σετ 3:** 1 4 1 9 2 6 10 8 7 5 11 3 1



- **Σετ 4:** 1 4 9 2 11 3 1 8 6 5 7 10 1



- **Σετ 5:** 1 8 11 1 2 7 5 10 4 6 9 3 1



Φαίνεται, λοιπόν, ότι ο GRASP βελτιώνει σημαντικά τα κόστη. Επιπλέον, στη συγκεκριμένη διάσταση, όπου ο αριθμός των κόμβων είναι μικρός, μπορούν να διακριθούν σχετικά εύκολα τόσο οι αλλαγές του 1-0 Relocate όσο και οι 2 - Opt κινήσεις. Γι΄ αυτό, έχει νόημα να αναλυθούν ξεχωριστά τα αποτελέσματα σε κάθε φάση του GRASP. Παρακάτω, παρουσιάζονται ενδεικτικά μερικές λύσεις στις αντίστοιχες φάσεις από το πρώτο σετ δεδομένων.

possible_routes		
set	route	route_cost
1	1 9 6 4 2 8 7 1 10 3 11 5 1	749
1	1 3 6 9 4 2 10 1 7 8 5 11 1	633
1	1 6 3 2 11 4 10 1 7 5 9 8 1	662
1	1 3 2 11 4 7 5 1 8 9 6 10 1	675
1	1 6 2 11 4 7 5 8 1 9 3 10 1	558

new_possible_routes		
set	route	route_cost
1	1 9 5 8 7 10 4 1 6 3 2 11 1	468
1	1 3 6 4 11 2 1 9 8 5 7 10 1	480
1	1 6 3 2 11 4 1 9 7 10 8 5 1	470
1	1 2 11 4 9 1 10 7 5 8 3 6 1	444
1	1 2 11 4 8 1 9 6 3 5 7 10 1	495

final_possible_routes		
set	route	route_cost
1	1 9 4 10 7 8 5 1 6 3 2 11 1	446
1	1 3 6 4 11 2 1 9 8 10 7 5 1	461
1	1 6 3 4 11 2 1 9 8 10 7 5 1	460
1	1 2 11 4 9 1 10 7 8 5 3 6 1	428
1	1 2 11 4 8 1 9 10 7 5 3 6 1	476

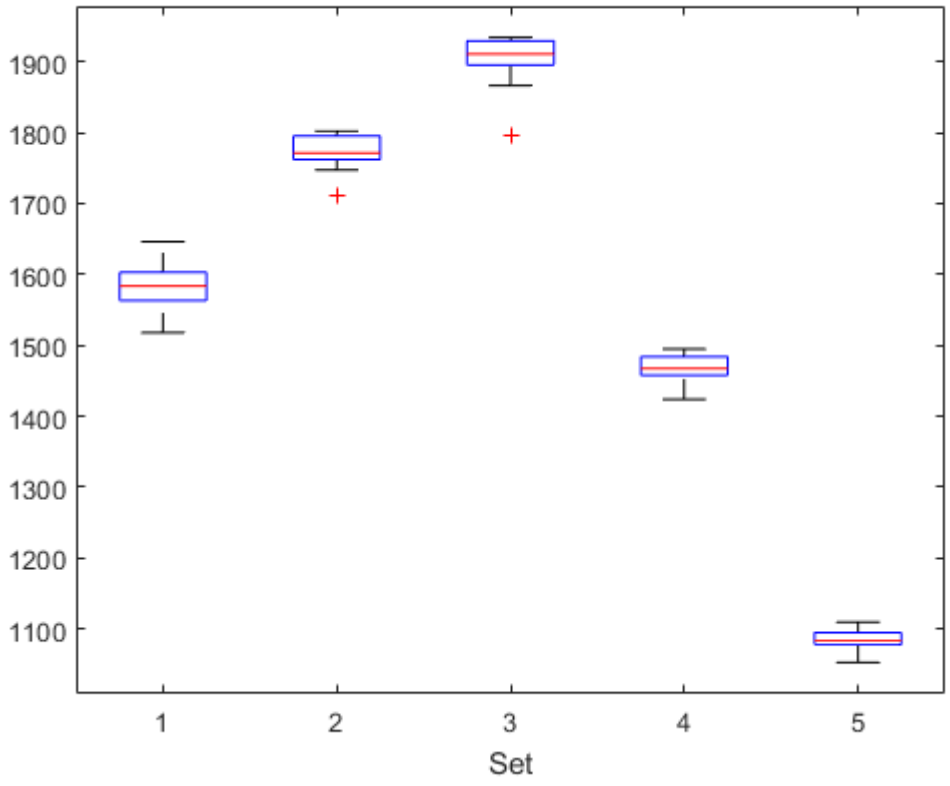
Για παράδειγμα, στην πρώτη λύση [1 9 6 4 2 8 7 1 10 3 11 5 1] οι κόμβοι 2, 6 και 5, 10 αλλάζουν διαδρομές κατά τον 1-0 Relocate, με αποτέλεσμα η νέα λύση να διαμορφώνεται ως εξής: [1 9 5 8 7 10 4 1 6 3 2 11 1]. Κατά τον 2 - Opt, αρχικά, επιλέγονται οι κόμβοι 5, 4 και στη συνέχεια αντιστρέφεται το μονοπάτι 5 - 8 - 7 - 10 - 4. Εν τέλει, προκύπτει η εξής τελική πιθανή λύση: [1 9 4 10 7 8 5 1 6 3 2 11 1]. Επίσης, φαίνεται ότι το κόστος, αν και αρχίζει από μία υψηλή τιμή, μειώνεται σημαντικά μετά από κάθε αλγόριθμο τοπικής αναζήτησης. Μάλιστα, η αρχική τιμή είναι υψηλότερη από αυτήν του Πλησιέστερου Γείτονα, κάτι το οποίο αποδίδεται στην τυχαιότητα του GRASP λόγω του μεγέθους της λίστας περιορισμένων υποψηφίων.

5.2 Σετ Δεδομένων 51 Κόμβων

Στη συγκεκριμένη ενότητα, ο GRASP εκτέλεσε 1000 επαναλήψεις, ενώ οι αλγόριθμοι τοπικής αναζήτησης εκτέλεσαν 250 επαναλήψεις έκαστος. Προηγήθηκαν δοκιμές στις 200, 50 και 500, 125 επαναλήψεις, αντίστοιχα, αλλά τα αποτελέσματα ήταν πολύ κοντά σε αυτά του Πλησιέστερου Γείτονα. Γι΄ αυτό, αποφασίστηκε να αυξηθεί ο αριθμός των επαναλήψεων, προκειμένου να περιοριστεί ο παράγοντας της τυχαιότητας και να κατασκευαστούν παράπανω πιθανές λύσεις. Πράγματι, οι επιπλέον πιθανές λύσεις εξασφάλισαν ένα ικανοποιητικό πλήθος καλύτερων τελικών λύσεων, μέσα από τις οποίες προέκυψε τελικά η βέλτιστη. Επίσης, το μέγεθος της λίστας περιορισμένων υποψηφίων ορίστηκε ίσο με 3,

εκτός από το 4ο σετ δεδομένων. Στο εν λόγω σετ, το μέγεθος της λίστας περιορισμένων υποψηφίων ελάβε την τιμή 2, καθώς μετά από αρκετές δοκιμές με διαφορετικά μεγέθη και αριθμό επαναλήψεων, διαπιστώθηκε ότι υπό αυτές τις παραμέτρους προκύπτει μία ικανοποιητική βελτίωση. Παρακάτω, παρουσιάζονται τα συγκεντρωτικά αποτέλεσματα των πέντε σετ δεδομένων, ύστερα από τα δέκα διαφορετικά «τρεξίματα», καθώς και το αντίστοιχο θηκόγραμμα με την ελάχιστη, την ενδιάμεση, τη μέγιστη τιμή και τις τιμές του πρώτου και του τρίτου τεταρτημορίου.

Set	Πλ. Γείτονας	Βελτ. GRASP	Μ.Ο GRASP	Τυπ. Αποκλ. GRASP
1	1604	1518	1581.7	34.7737
2	1898	1712	1770.1	25.7816
3	2058	1796	1899.9	39.9911
4	1464	1424	1467.9	19.0497
5	1242	1053	1083.8	14.6205



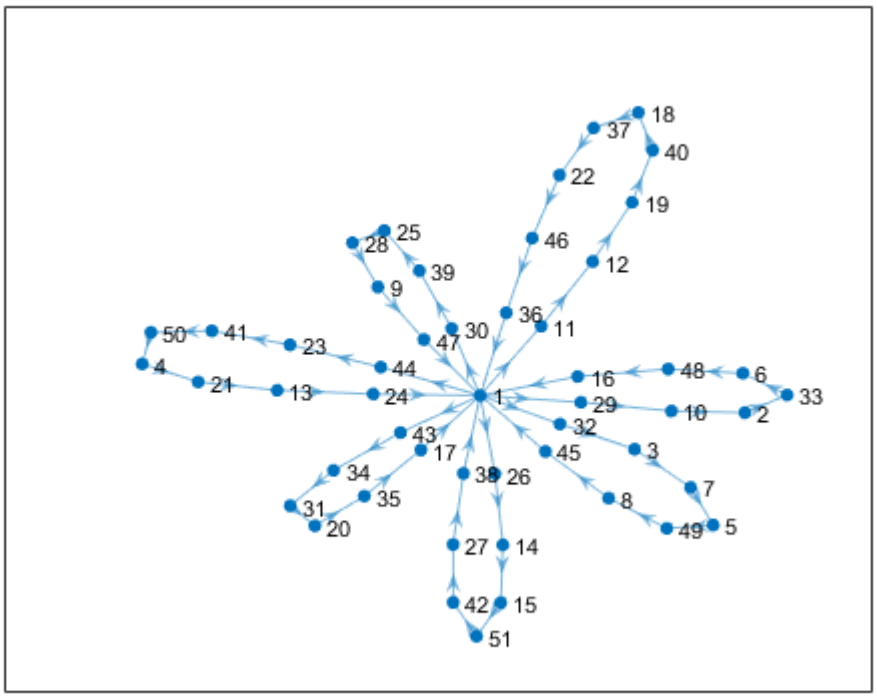
Στη συνέχεια, παρουσιάζονται συγκεντρωτικά και σχηματικά τόσο οι λύσεις του Πλησιέστερου Γείτονα όσο και οι βέλτιστες λύσεις του GRASP.

Πλησιέστερος Γείτονας (RCL = 1)		
set	route	route_cost
1	1 1	1604
2	1 1	1898
3	1 1	2058
4	1 1	1464
5	1 1	1242

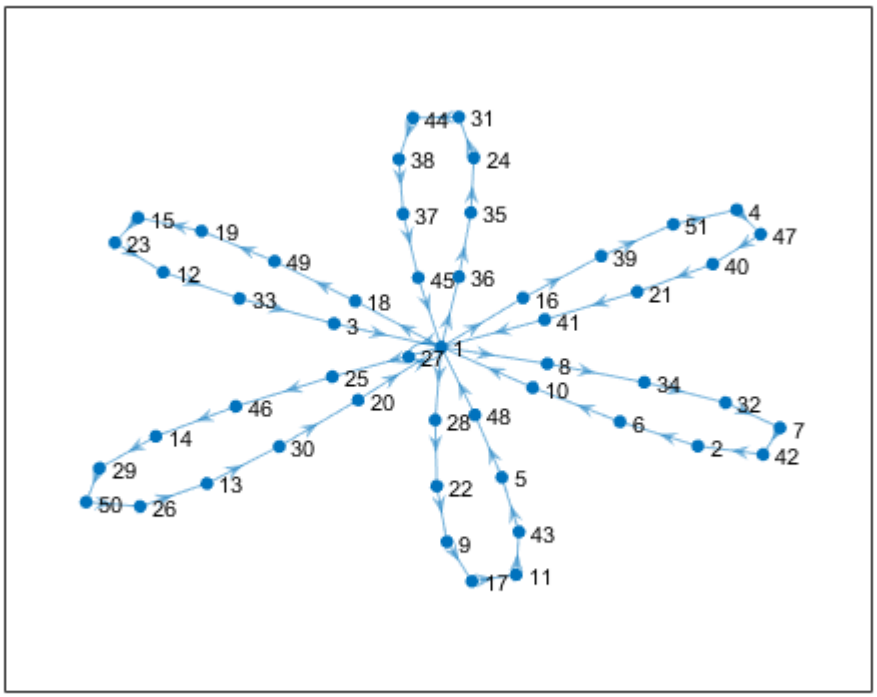
GRASP (RCL = 2, 3)		
set	route	route_cost
1	1 1	1518
2	1 1	1712
3	1 1	1796
4	1 1	1424
5	1 1	1053

Για τον **Πλησιέστερο Γείτονα:**

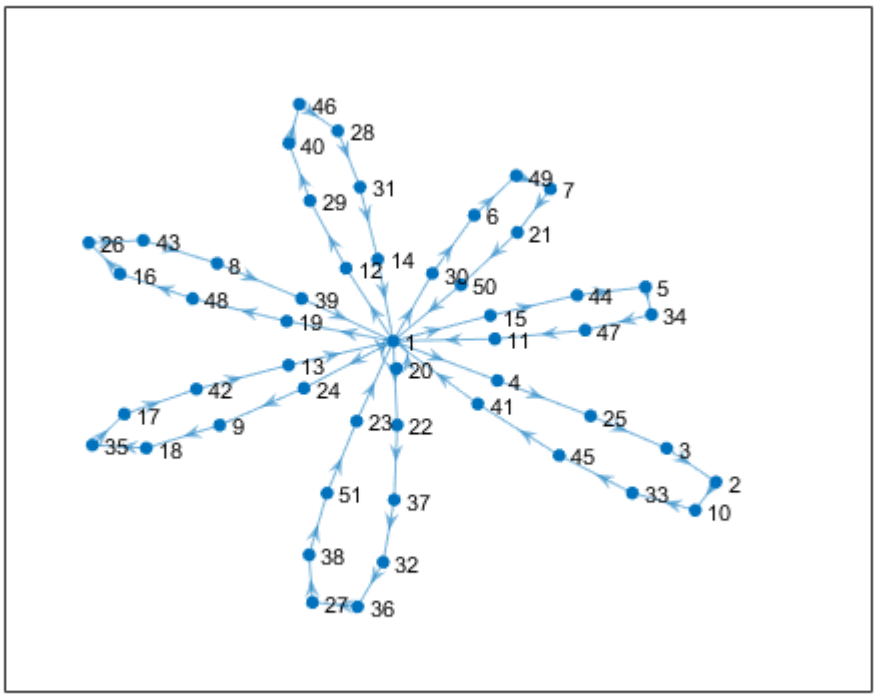
- **Σετ 1:** 1 44 23 41 50 4 21 13 24 1 43 34 31 20 35 17 1 29 10 2 33 6 48 16 1 11 12 19 40 18 37 22 46 36 1 26 14 15 51 42 27 38 1 30 39 25 28 9 47 1 32 3 7 5 49 8 45 1



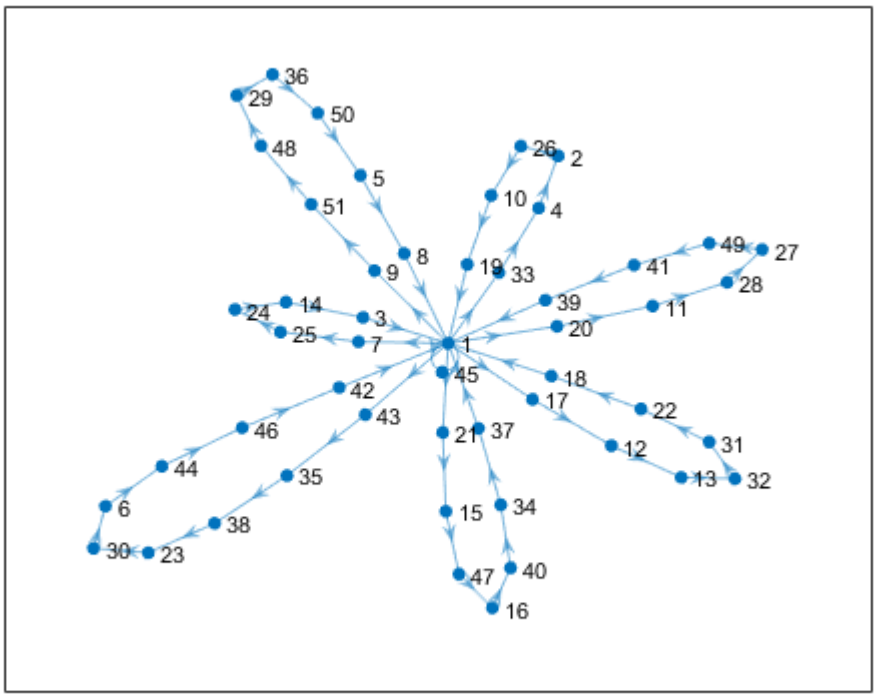
- **Σετ 2:** 1 25 46 14 29 50 26 13 30 20 1 28 22 9 17 11 43 5 48 1 16 39 51 4 47 40 21 41 1 36 35 24 31 44 38 37 45 1 18 49 19 15 23 12 33 3 1 8 34 32 7 42 2 6 10 1 27 1



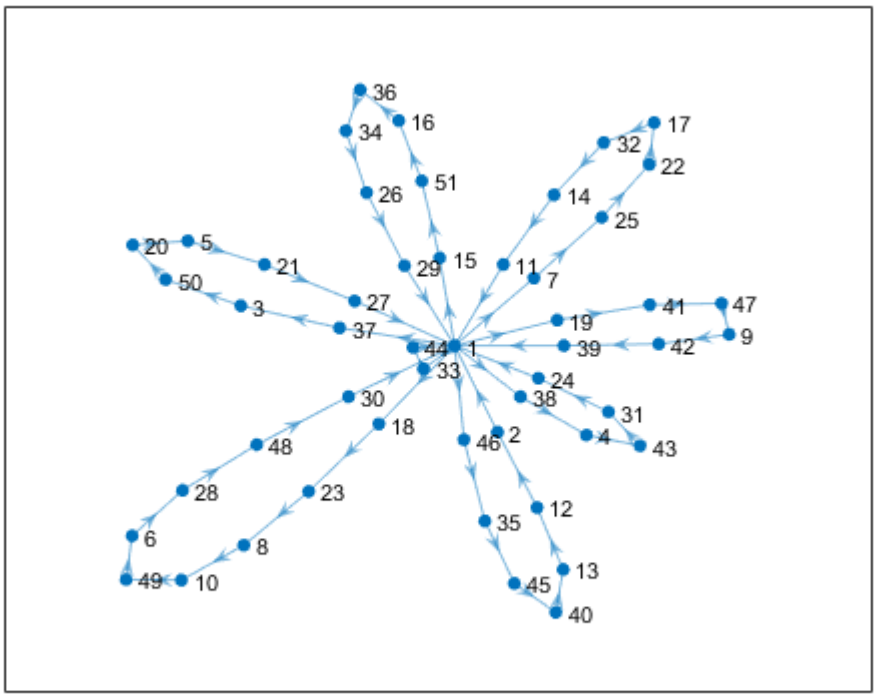
- **Σετ 3:** 1 24 9 18 35 17 42 13 1 30 6 49 7 21 50 1 22 37 32 36 27 38 51 23 1 19 48 16 26 43 8 39 1 4 25 3 2 10 33 45 41 1 12 29 40 46 28 31 14 1 15 44 5 34 47 11 1 20 1



- **Σετ 4:** 1 7 25 24 14 3 1 43 35 38 23 30 6 44 46 42 1 33 4 2 26 10 19 1 17 12 13 32 31 22 18 1 21 15 47 16 40 34 37 1 9 51 48 29 36 50 5 8 1 20 11 28 27 49 41 39 1 45 1

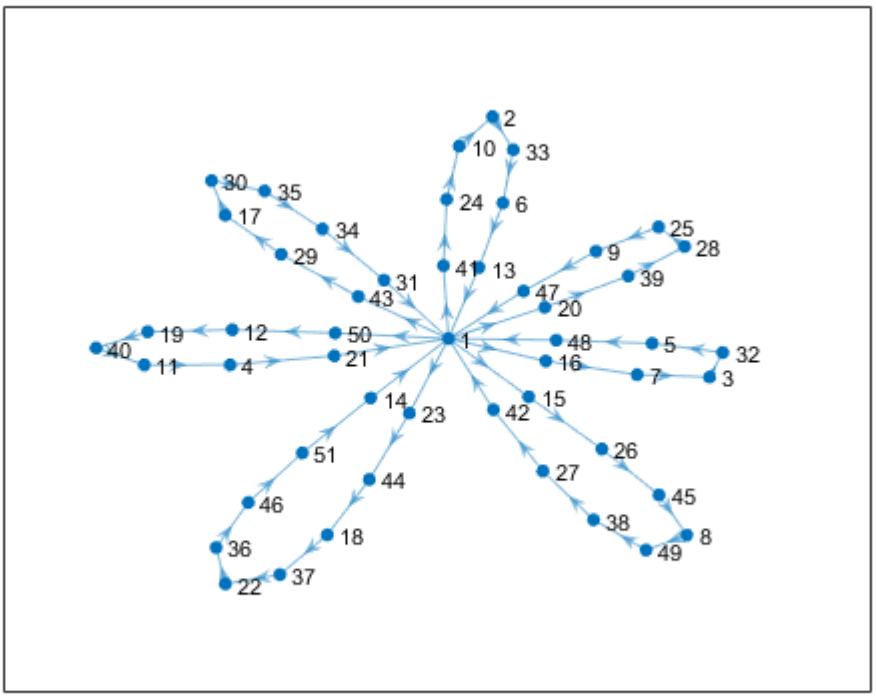


- **Σετ 5:** 1 7 25 22 17 32 14 11 1 19 41 47 9 42 39 1 46 35 45 40 13 12 2 1 37 3 50 20 5 21 27 1 38 4 43 31 24 1 18 23 8 10 49 6 28 48 30 1 15 51 16 36 34 26 29 1 33 44 1

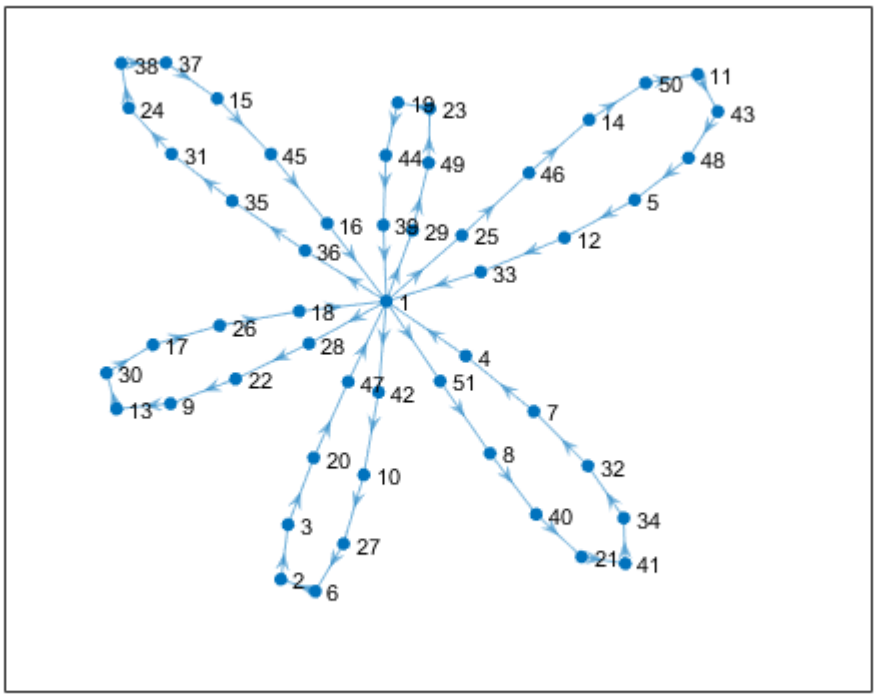


Για τον **GRASP:**

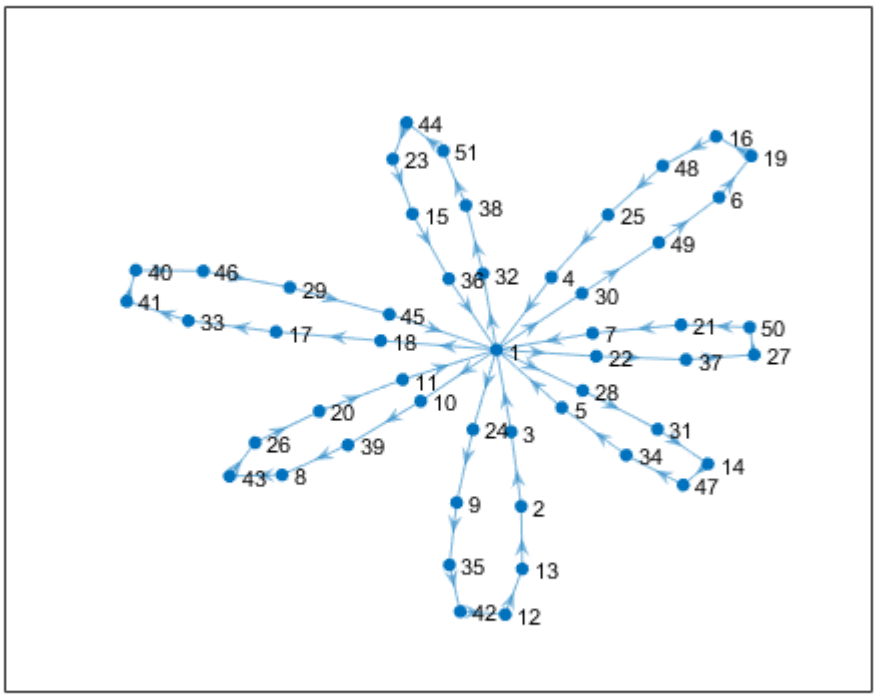
- **Σετ 1:** 1 50 12 19 40 11 4 21 1 43 29 17 30 35 34 31 1 41 24 10 2 33 6 13 1 23 44 18 37 22 36 46 51 14 1 20 39 28 25 9 47 1 15 26 45 8 49 38 27 42 1 16 7 3 32 5 48 1



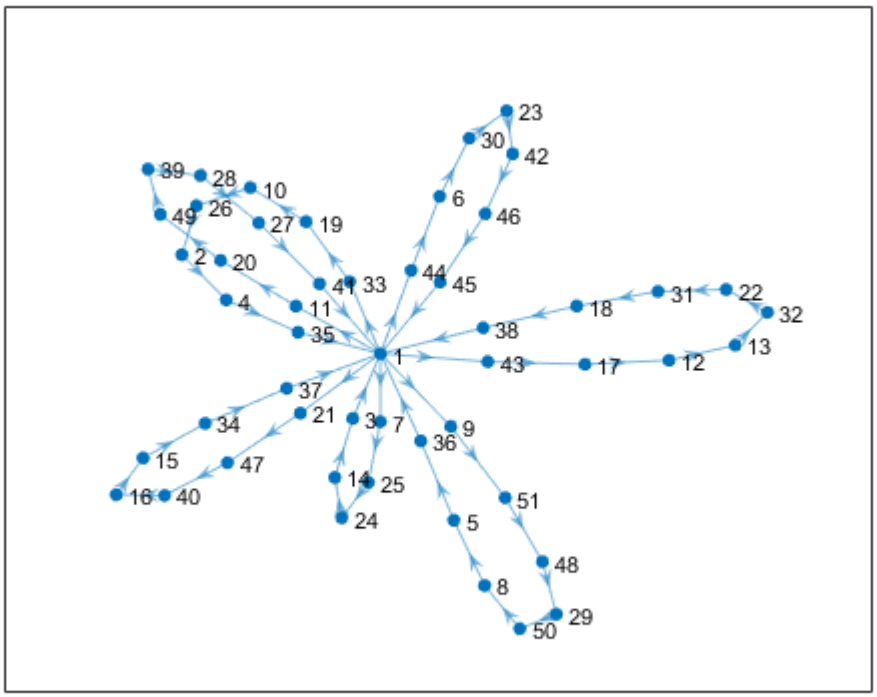
- **Σετ 2:** 1 28 22 9 13 30 17 26 18 1 25 46 14 50 11 43 48 5 12 33 1 36 35 31 24 38 37 15 45 16 1 29 49 23 19 44 39 1 51 8 40 21 41 34 32 7 4 1 42 10 27 6 2 3 20 47 1



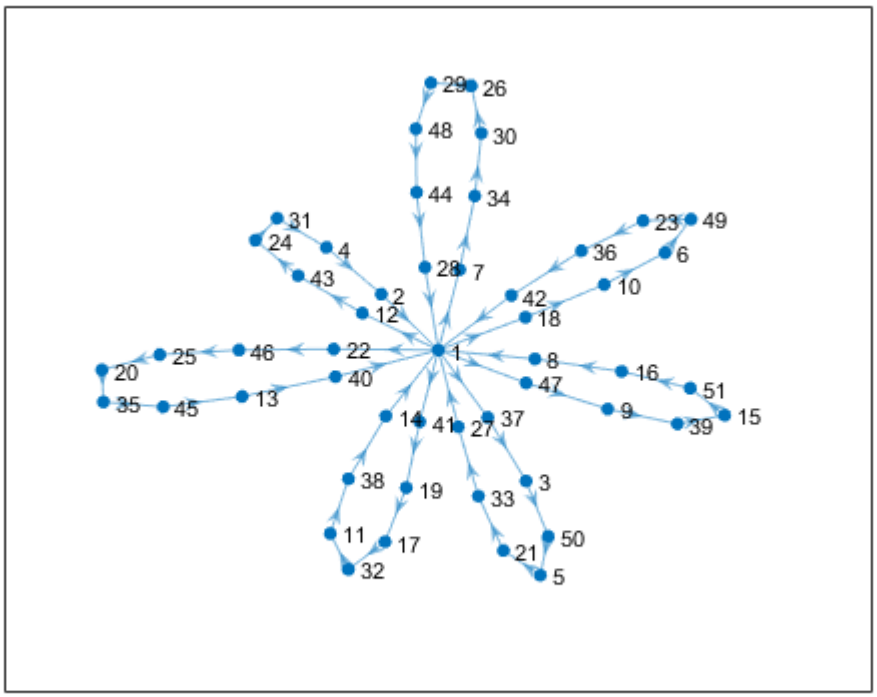
- **Σετ 3:** 1 30 49 6 19 16 48 25 4 1 22 37 27 50 21 7 1 24 9 35 42 12 13 2 3 1 18 17 33 41 40 46 29 45 1 32 38 51 44 23 15 36 1 10 39 8 43 26 20 11 1 28 31 14 47 34 5 1



- **Σετ 4:** 1 43 17 12 13 32 22 31 18 38 1 7 25 24 14 3 1 33 19 10 26 2 4 35 1 21 47 40 16 15 34 37 1 9 51 48 29 50 8 5 36 1 44 6 30 23 42 46 45 1 11 20 49 39 28 27 41 1



- **Σετ 5:** 1 22 46 25 20 35 45 13 40 1 41 19 17 32 11 38 14 1 47 9 39 15 51 16 8 1 37 3 50 5 21 33 27 1 12 43 24 31 4 2 1 18 10 6 49 23 36 42 1 7 34 30 26 29 48 44 28 1



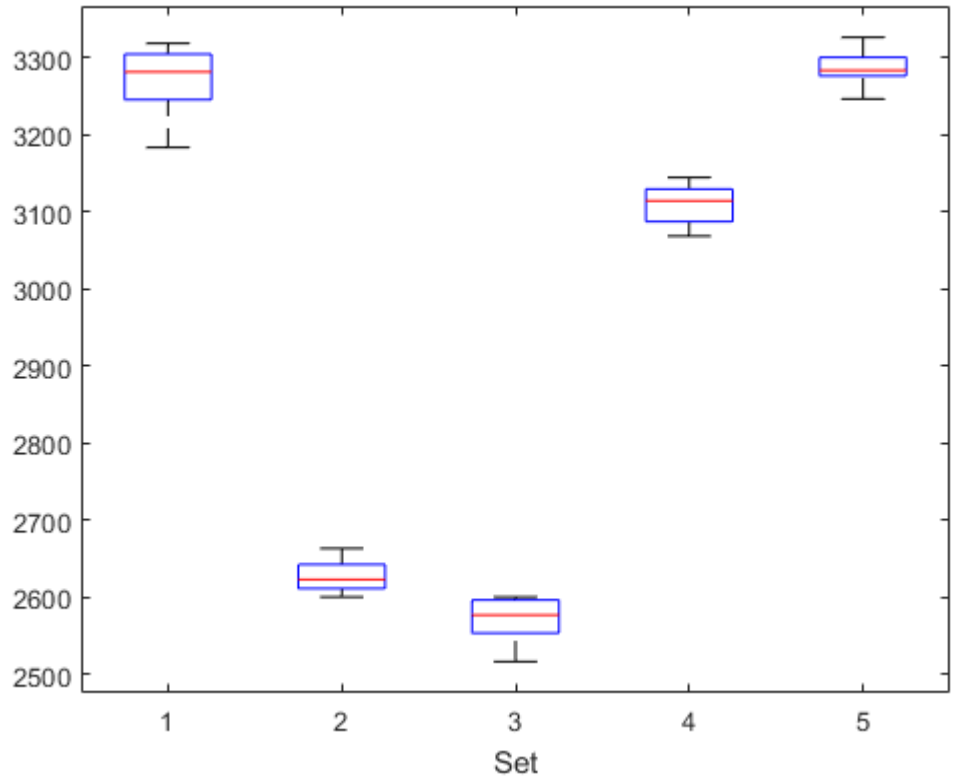
Φαίνεται, λοιπόν, ότι κι εδώ με την εφαρμογή του GRASP επιτυγχάνεται σημαντική βελτίωση στα κόστη. Επιπλέον, σε αυτή τη διάσταση γίνεται χρήση των «extended» πινάκων, αφού προκύπτουν πιθανές λύσεις διαφορετικών διαστάσεων.

5.3 Σετ Δεδομένων 101 Κόμβων

Στην ενότητα, αυτή, ο αριθμός των επαναλήψεων για τον GRASP «κλείδωσε» στις 2000 επαναλήψεις, ενώ για τους αλγορίθμους τοπικής αναζήτησης «κλείδωσε» στις 500 επαναλήψεις. Κι΄ αυτό, γιατί οι δοκιμές στις 500, 125 και 1000, 250 επαναλήψεις, αντίστοιχα, δεν επέφεραν την επιθυμητή βελτίωση. Τα αποτελέσματα μόλις βελτίωναν αυτά του Πλησιέστερου Γείτονα, με αποτέλεσμα να αυξηθεί ο αριθμός των επαναλήψεων αλλά και να «κλειδώσει» το μέγεθος της λίστας περιορισμένων υποψηφίων στην τιμή 2. Πράγματι, υπό αυτές τις συνθήκες, τα αποτελέσματα σημείωσαν αισθητή βελτίωση στα τρία από τα πέντε σετ δεδομένων. Πιο συγκεκριμένα, στα σετ 1,2 και 5 παρατηρήθηκε σημαντική

βελτίωση στα κόστη, ενώ στο 4ο σετ η βελτίωση κρίθηκε ικανοποιητική. Όμως, δεν συνέβη το ίδιο και στο 3ο σετ, όπου ο αλγόριθμος του GRASP δεν κατάφερε να επιστρέψει μία καλύτερη λύση. Ακόμη, σε ορισμένα σετ δεδομένων προέκυψαν πιθανές λύσεις τριών διαφορετικών διαστάσεων, με αποτέλεσμα οι «extended» πίνακες να μην επαρκούν. Έτσι, για τα σετ 2 και 4 χρησιμοποιήθηκαν οι επιπλέον πίνακες «v2extended», οι οποίοι επιτελούν ακριβώς το ίδιο έργο με τους «extended» πίνακες. Παρακάτω, παρουσιάζονται τα συγκεντρωτικά αποτέλεσματα των πέντε σετ δεδομένων, ύστερα από τα δέκα διαφορετικά «τρεξίματα», καθώς και το αντίστοιχο θηκόγραμμα με την ελάχιστη, την ενδιάμεση, τη μέγιστη τιμή και τις τιμές του πρώτου και του τρίτου τεταρτημορίου.

Set	Πλ. Γείτονας	Βελτ. GRASP	Μ.Ο GRASP	Τυπ. Αποκλ. GRASP
1	3371	3183	3271.2	37,9284
2	2703	2600	2626	19,5806
3	2507	2516	2571	26,5104
4	3075	3068	3109.6	25,1324
5	3393	3246	3287.4	21,3645



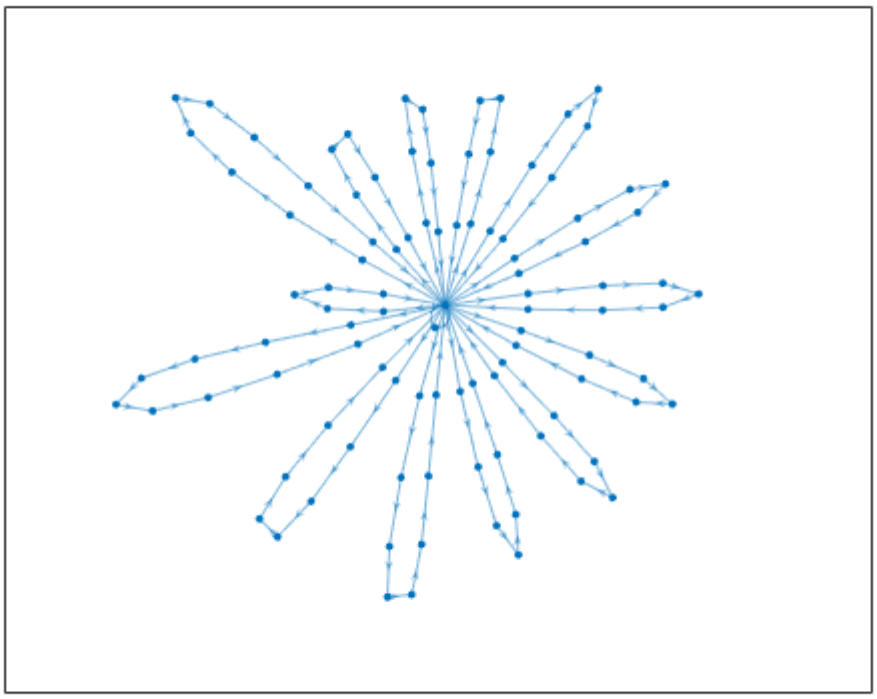
Στη συνέχεια, παρουσιάζονται συγκεντρωτικά και σχηματικά τόσο οι λύσεις του Πλησιέστερου Γείτονα όσο και οι βέλτιστες λύσεις του GRASP.

Πλησιέστερος Γείτονας (RCL = 1)		
set	route	route_cost
1	1 1	3371
2	1 1	2703
3	1 1	2507
4	1 1	3075
5	1 1	3393

GRASP (RCL = 2)		
set	route	route_cost
1	1 1	3183
2	1 1	2600
3	1 1	2516
4	1 1	3068
5	1 1	3246

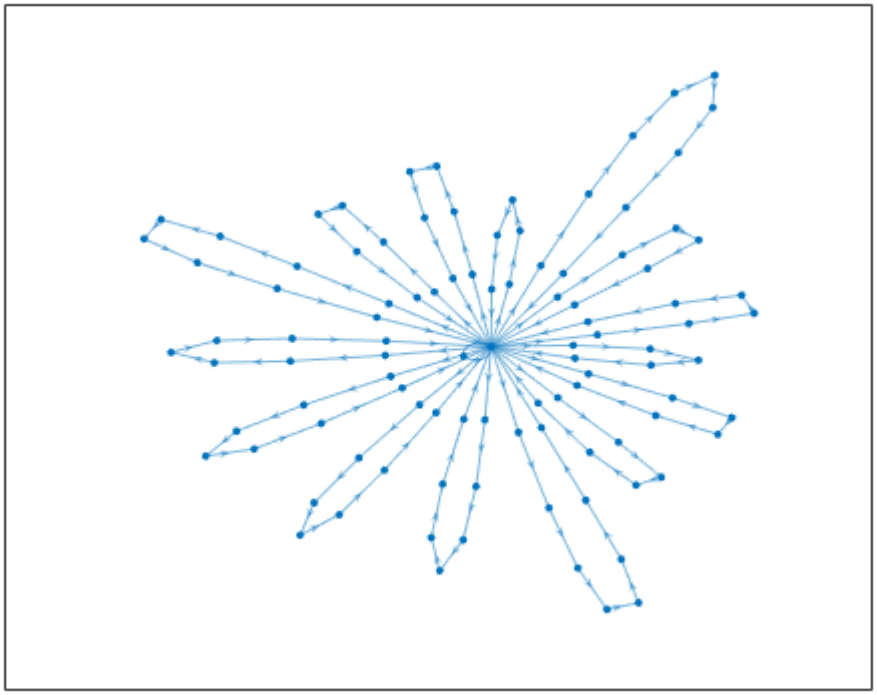
Για τον **Πλησιέστερο Γείτονα:**

- **Σετ 1:** 1 80 57 56 28 32 59 41 1 93 61 81 99 44 2 86 1 29 39 10 98 25 42 65 1 9 35 97 47 68 94 88 89 1 83 45 91 100 69 64 34 14 77 1 11 38 36 76 30 49 50 1 78 12 31 82 101 55 21 1 53 52 3 22 67 54 1 96 92 51 71 66 13 1 15 24 33 62 27 90 1 4 70 43 26 6 1 79 37 75 60 5 73 58 1 16 7 84 40 72 63 48 23 1 20 8 18 17 85 87 95 19 46 1 74 1

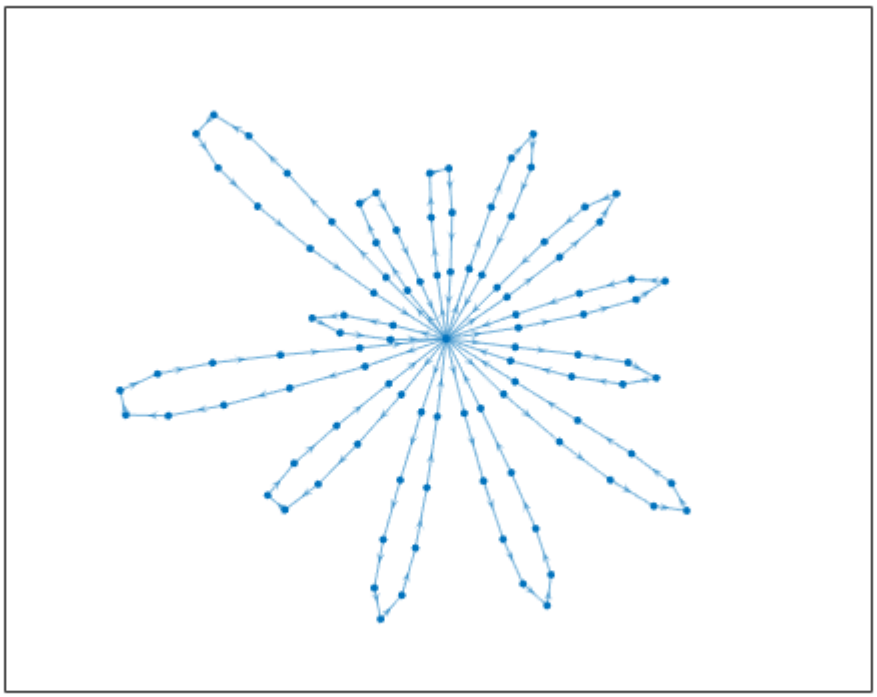


- **Σετ 2:** 1 15 20 28 86 63 16 1 101 66 62 22 12 1 59 98 77 21 27 7 1 45 23 11 83 50 37 1 90 39 25 81 43 60 34 1 46 2 64 6 92 8 1 13 73 96 71 72

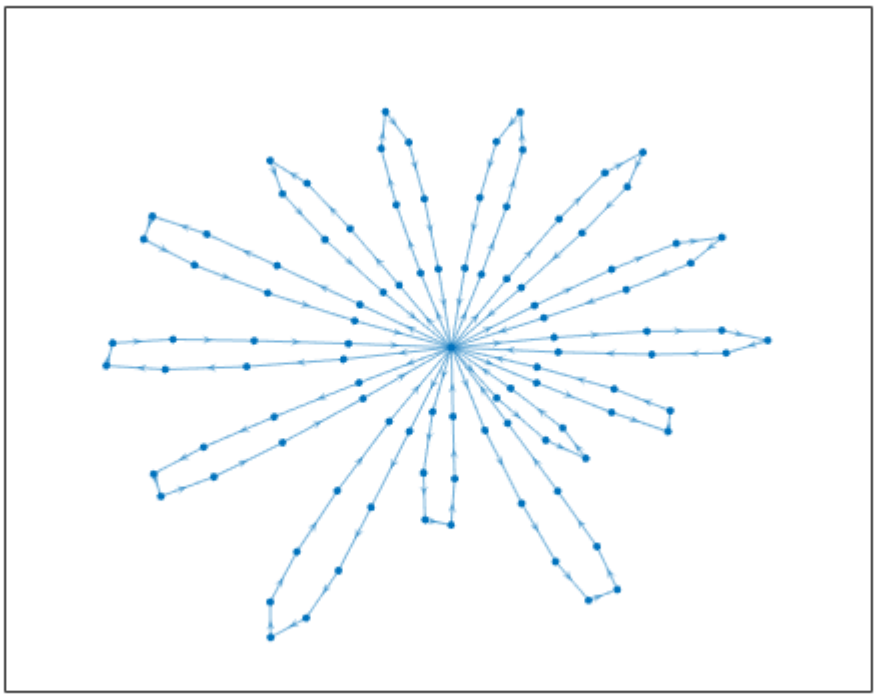
91 68 1 29 52 26 94 88 65 54 1 17 38 85 49 3 57 1 32 36 44 41 19 75 1 31 82 4 61 74 5 100 47 1 78 67 9 42 30 1 99 70 76 33 58 80 53 93 56 1 14 18 79 51 97 84 35 1 87 48 10 55 24 69 40 95 1 89 1



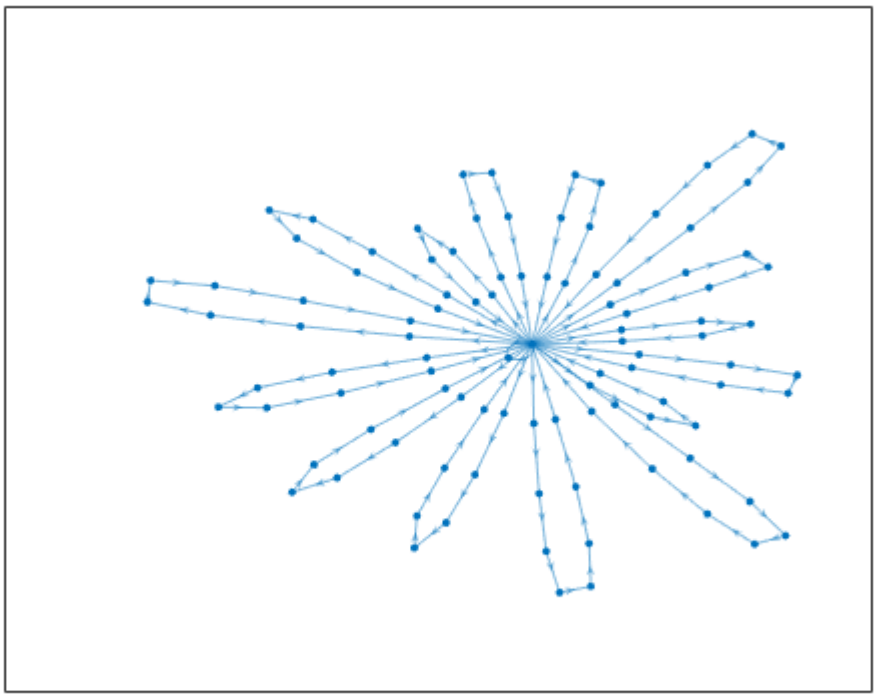
- **Σετ 3:** 1 94 51 64 33 31 49 1 4 74 5 53 32 18 16 1 52 35 79 96 93 40 20 1 22 38 6 100 63 60 7 26 36 1 19 84 23 44 92 72 15 43 8 1 81 27 89 88 47 25 1 75 50 30 39 70 14 9 37 1 97 34 77 76 45 1 48 24 10 90 13 41 67 1 87 80 29 42 59 98 56 46 58 1 66 83 85 11 57 62 91 54 17 12 1 2 86 65 28 61 78 101 1 69 99 68 21 71 95 55 3 73 82 1



- **Σετ 4:** 1 94 5 52 45 23 88 36 1 10 91 95 37 60 58 55 1 27 13 28 59 101 86 1 76 90 73 46 83 1 61 70 69 78 80 93 21 1 41 68 44 66 75 72 33 26 1 16 97 2 35 7 9 38 1 15 82 24 29 79 56 89 1 62 98 81 84 96 25 85 51 48 1 14 40 67 8 17 31 63 65 1 32 77 74 18 92 57 47 100 1 6 3 43 99 54 30 34 1 53 4 11 64 87 12 50 39 1 22 71 42 49 20 19 1

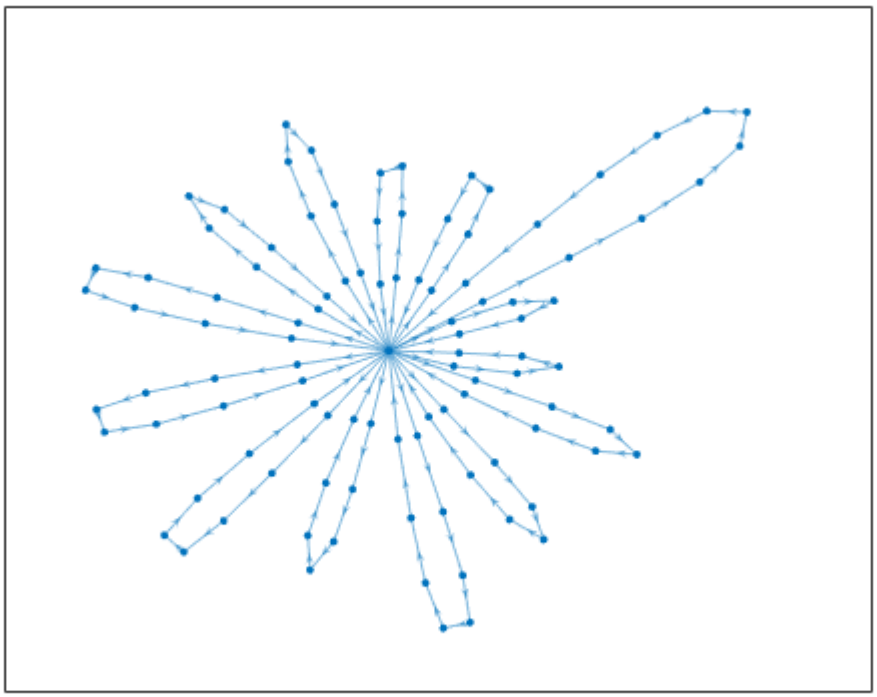


- **Σετ 5:** 1 61 60 73 22 20 29 21 38 1 6 56 88 19 32 43 55 1 57 90 99 37 39 89 36 100 1 64 27 70 16 28 1 81 78 53 26 14 47 1 87 30 10 84 77 35 9 46 1 59 76 44 80 11 1 45 31 18 4 50 67 1 48 49 54 74 42 75 86 1 82 15 17 66 69 7 1 101 62 8 91 34 1 68 83 97 58 92 98 72 12 1 41 3 79 5 23 40 52 1 71 24 33 95 93 96 85 1 2 51 63 65 25 13 1 94 1

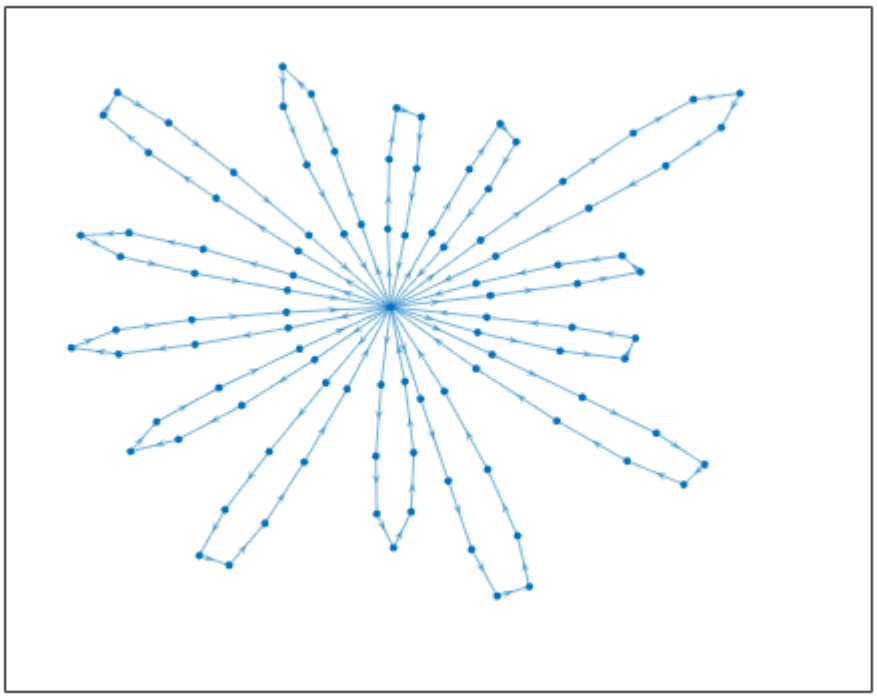


Για τον **GRASP:**

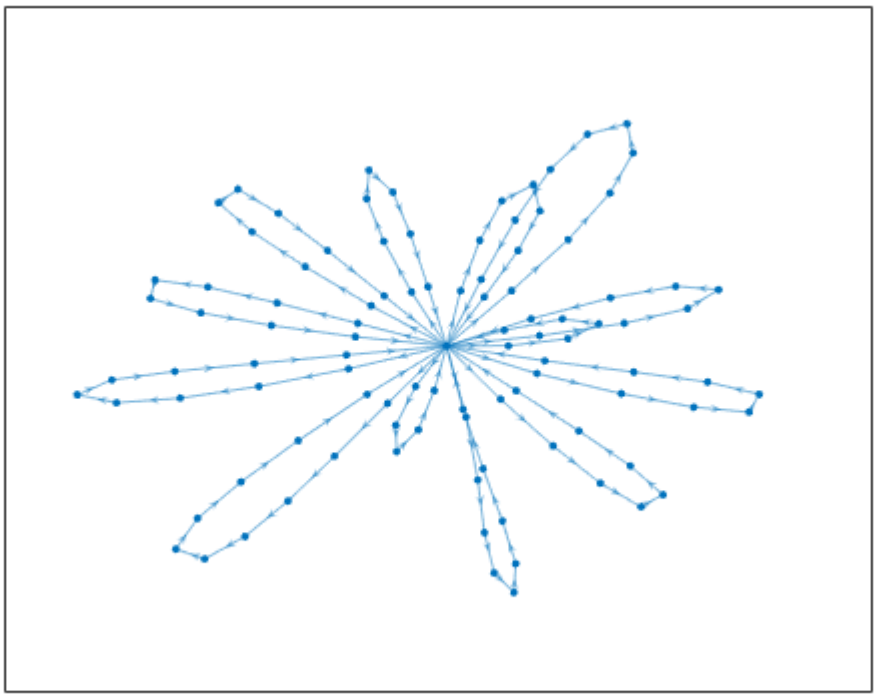
- **Σετ 1:** 1 80 56 28 32 59 41 9 1 93 81 61 25 10 39 29 1 83 45 91 100 69 64 14 94 34 77 57 1 11 36 67 3 22 1 42 79 37 75 60 70 4 98 1 78 12 101 55 7 16 54 52 1 99 44 2 86 31 82 21 1 38 53 49 50 84 30 76 1 35 97 68 88 63 89 24 1 47 72 40 48 23 62 33 66 1 65 5 73 58 17 85 95 87 1 96 51 92 71 27 13 1 15 90 19 46 74 43 1 6 26 8 18 20 1



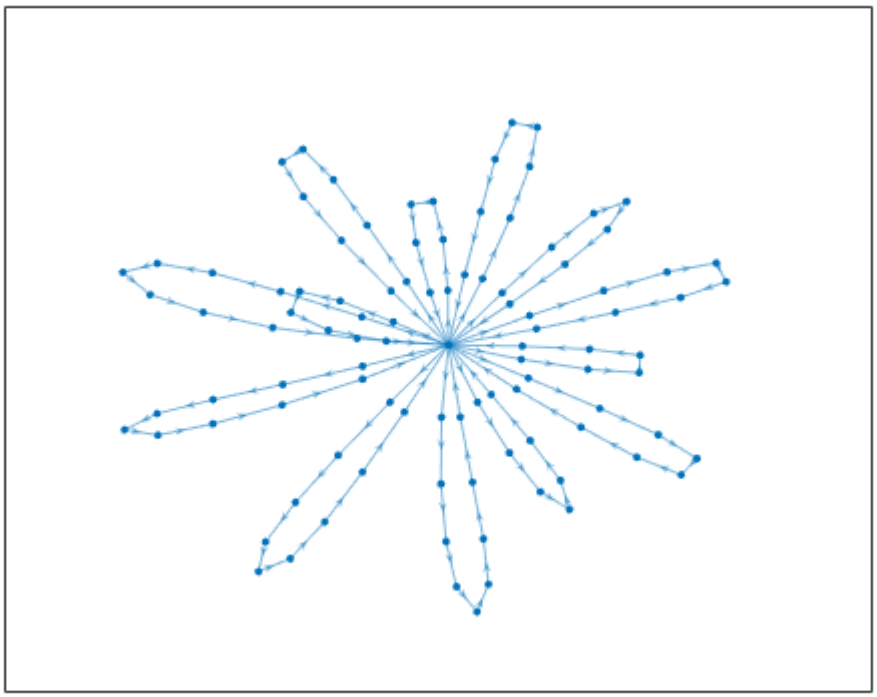
- **Σετ 2:** 1 20 15 28 46 90 63 86 1 59 8 92 16 96 73 13 1 101 66 17 62 38 49 31 85 1 7 21 77 27 98 71 1 45 12 23 11 83 81 25 43 1 22 3 37 50 57 100 1 29 26 94 97 51 84 79 14 1 2 6 64 78 60 34 39 1 52 19 41 36 44 32 1 67 9 42 95 30 47 1 72 99 91 68 87 48 10 70 33 1 88 65 54 35 55 24 80 76 1 82 4 61 74 5 89 75 1 18 69 40 53 56 93 58 1



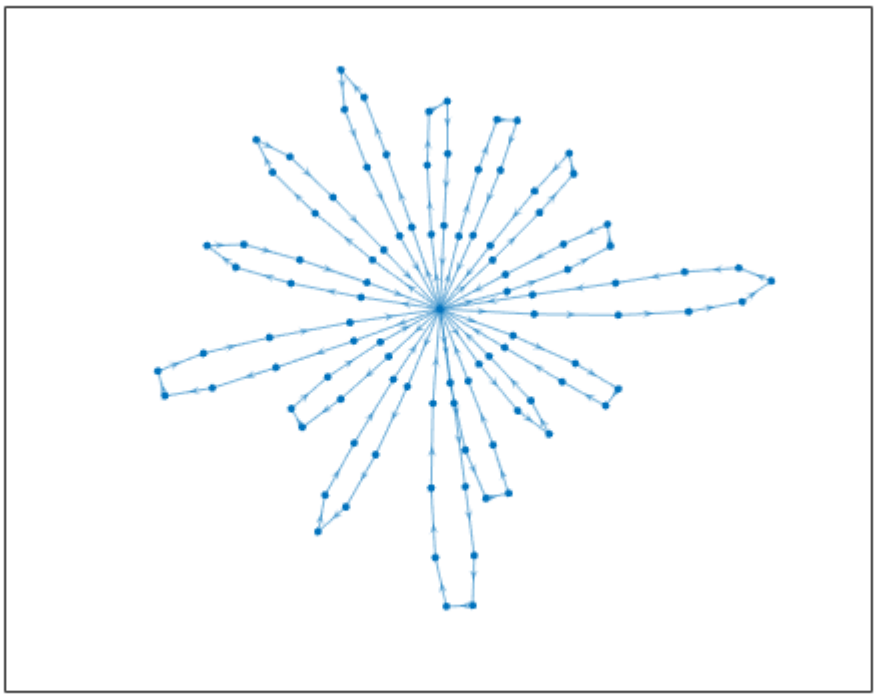
- **Σετ 3:** 1 51 94 64 31 33 49 18 32 1 4 74 5 47 25 53 87 1 22 52 35 93 96 79 20 40 1 38 100 63 60 7 36 78 26 101 82 1 19 84 23 92 72 15 44 81 27 1 6 89 88 43 8 66 30 70 1 75 16 97 34 77 1 50 39 14 9 83 11 85 1 76 45 80 29 42 41 67 1 48 90 13 99 69 68 21 71 95 1 98 56 86 58 46 65 28 61 1 2 59 73 55 3 12 17 54 91 1 24 10 57 62 37 1



- **Σετ 4:** 1 94 10 37 91 60 58 55 1 5 52 45 23 61 95 76 90 1 59 101 28 13 27 73 1 86 46 83 81 84 96 25 1 70 69 78 93 80 57 47 100 20 1 16 97 67 40 8 17 63 31 1 41 35 2 65 38 9 3 6 1 68 44 66 72 75 26 51 85 33 1 62 98 32 77 74 48 18 21 92 1 14 50 12 11 64 87 4 53 36 1 15 24 79 29 56 89 82 19 1 88 39 7 43 54 30 1 22 71 42 99 34 49 1



- **Σετ 5:** 1 6 56 43 32 19 88 1 61 60 90 99 73 22 100 20 1 81 78 53 18 14 47 1 39 89 36 75 86 17 15 1 64 70 27 59 48 21 1 87 30 10 77 84 72 12 9 35 1 26 101 7 66 82 37 1 45 31 4 96 95 93 67 1 29 38 74 54 49 51 1 16 68 83 97 58 92 1 42 69 8 91 23 5 79 3 1 76 28 44 80 11 2 1 50 24 46 71 33 55 98 1 41 62 40 34 52 13 57 1 85 25 94 63 65 1



Εξαιρώντας, λοιπόν, το 4ο σετ δεδομένων, φαίνεται ότι η εφαρμογή του GRASP επέφερε κι εδώ αισθητή βελτίωση στα κόστη.

5.4 Συμπεράσματα

Εξετάζοντας προσεχτικά τα αποτέλεσματα των προηγούμενων ενοτήτων, τα συμπεράσματα εστιάζονται στο «ισοζύγιο» τριών παραμέτρων. Το πλήθος των πελατών, ο αριθμός των επαναλήψεων στον GRASP αλλά και το μέγεθος της λίστας περιορισμένων υποψηφίων αποτελούν τρεις έννοιες στενά συνδεδεμένες μεταξύ τους, οι οποίες παίζουν καταλυτικό ρόλο στην ποιότητα των λύσεων του GRASP.

Αρχικά, το πλήθος των πελατών καθορίζει σε μεγάλο βαθμό τον αριθμό των επαναλήψεων του GRASP. Μάλιστα, παρατηρείται μία αναλογία ανάμεσα στις δύο παραμέτρους. Στα προβλήματα λίγων πελατών, ο GRASP εγγυάται εύκολο, γρήγορο και αξιόπιστο εντοπισμό της βέλτιστης λύσης ακόμα και με μικρό

αριθμό επαναλήψεων. Αυξάνοντας, όμως, τον αριθμό των πελατών, απαιτείται και η αύξηση των επαναλήψεων τόσο του GRASP όσο και των αλγορίθμων της τοπικής αναζήτησης. Με την αύξηση των επαναλήψεων στον GRASP και κατ΄ επέκταση των τελικών πιθανών λύσεων επιτυγχάνεται η κατά το δυνατόν μεγαλύτερη «κάλυψη» των πιθανών συνδυασμών στην αλληλουχία εξυπηρέτησης των πελατών. Ενώ, με την αύξηση των επαναλήψεων στην τοπική αναζήτηση, επιτυγχάνεται η κατά το δυνατόν καλύτερη ανάθεση των πελατών σε κάθε διαδρομή. Έτσι, η ποιότητα των λύσεων του GRASP διατηρείται σε ένα καλύτερο επίπεδο, συγκρίνοντας με τις λύσεις του Πλησιέστερου Γείτονα.

Έπειτα, το πλήθος των πελατών καθορίζει σε μεγάλο βαθμό και το μέγεθος της λίστας περιορισμένων υποψηφίων. Η σχέση, που συνδέει τις δύο παραμέτρους, χαρακτηρίζεται ως αντιστρόφως ανάλογη. Στα προβλήματα λίγων πελατών, το μέγεθος της λίστας περιορισμένων υποψηφίων έχει το περιθώριο να πάρει και μεγαλύτερες τιμές. Όμως, όσο αυξάνεται το πλήθος των πελατών τόσο καλείται να μειωθεί το μέγεθος της λίστας, προκειμένου η ποιότητα των λύσεων να διατηρηθεί σε ένα ικανοποιητικό επίπεδο. Η μείωση του μεγέθους της λίστας συνεπάγεται κατά κάποιον τρόπο και τη μείωση της τυχαιότητας στην ανάθεση των πελατών σε κάθε διαδρομή. Στην τυχαιότητα αυτή, άλλωστε, οφείλεται η ειδοποιός διαφορά του GRASP με τον Πλησιέστερο Γείτονα. Γι΄ αυτό, στα προβλήματα πολλών πελατών δεν αποτελεί έκπληξη το γεγονός ότι ο Πλησιέστερος Γείτονας ενδέχεται να δώσει καλύτερα αποτέλεσματα από τον GRASP.

Τέλος, τόσο το πλήθος των πελατών όσο και ο αριθμός των επαναλήψεων παίζουν σημαντικό ρόλο στον χρόνο εκτέλεσης του GRASP. Για προβλήματα μικρών διαστάσεων και λίγων επαναλήψεων, ο GRASP κυμαίνεται στους ίδιους περίπου χρόνους με τον Πλησιέστερο Γείτονα. Όμως, για προβλήματα μεγαλύτερης κλίμακας, ο GRASP μπορεί να αποτέλεσει μία ιδιαίτερα χρονοβόρα διαδικασία.

Βιβλιογραφία

[1] Ιωάννης Μαρινάκης, Μαγδαληνή Μαρινάκη, Αθανάσιος Μυγδαλάς. (2019, Οκτώβριος). Προβλήματα Δρομολόγησης Οχημάτων στη Διαχείρηση της Εφοδιαστικής Αλυσίδας. Αθήνα, ΕΚΔΟΣΕΙΣ ΝΕΩΝ ΤΕΧΝΟΛΟΓΙΩΝ

[2] Nicolas Rincon-Garcia, Ben Waterson, Tom J. Cherrett, Fernando Salazar-Arrieta. A metaheuristic for the time-dependent vehicle routing problem considering driving hours regulations – An application in city logistics. Transportation Research Part A 137 (2020) 429-446.

[3] Yixiao Huang, Lei Zhao, Tom Van Woensel, Jean-Philippe Gross. Time-dependent vehicle routing problem with path flexibility. Transportation Research Part B 95 (2017) 169-195.

[4] Miguel Andres Figliozzi. The time dependent vehicle routing problem with time windows: Benchmark problems, an efficient solution algorithm, and solution characteristics. Transportation Research Part E 48 (2012) 616-636.

[5] Mehmet Soysal, Mustafa Çimen. A Simulation Based Restricted Dynamic Programming approach for the Green Time Dependent Vehicle Routing Problem. Computers and Operations Research 88 (2017) 297-305.

[6] Merve Keskin, Gilbert Laporte, Bülent Çatay. Electric Vehicle Routing Problem with Time-Dependent Waiting Times at Recharging Stations. Computers and Operations Research 107 (2019) 77-94.

[7] Binbin Pan, Zhenzhen Zhang, Andrew Lim. A hybrid algorithm for time-dependent vehicle routing problem with time windows. Computers and Operations Research 128 (2021)105193.

[8] Houming Fan, Yueguang Zhang, Panjun Tian, Yingchun Lv, Hao Fan. Time-dependent multi-depot green vehicle routing problem with time windows considering temporal-spatial distance. Computers and Operations Research 129 (2021) 105211.

[9] Somayeh Allahyari, Saeed Yaghoubi, Tom Van Woensel. The secure time-dependent vehicle routing problem with uncertain demands. Computers and Operations Research 131 (2021) 105253.

[10] Maha Gmira, Michel Gendreau, Andrea Lodi, Jean-Yves Potvin. Tabu

search for the time-dependent vehicle routing problem with time windows on a road network. European Journal of Operational Research 288 (2021) 129-140.

[11] Michael Saint-Guillain, Célia Paquay, Sabine Limbourg. Time-dependent stochastic vehicle routing problem with random requests: Application to online police patrol management in Brussels. European Journal of Operational Research 292 (2021) 869-885.

[12] Binbin Pan, Zhenzhen Zhang, Andrew Lim. Multi-trip time-dependent vehicle routing problem with time windows. European Journal of Operational Research 291 (2021) 218-231.

[13] Changshi Liu, Gang Kou, Xiancheng Zhou, Yi Peng, Huyi Sheng, Fawaz E. Alsaadi. Time-dependent vehicle routing problem with time windows of city logistics with a congestion avoidance approach. Knowledge-Based Systems 188 (2020) 104813.

[14] http://www.apollo.management.soton.ac.uk/prplib.htm

[15] www.supplychain.gr

[16] www.terra.gr

[17] en.wikipedia.org