ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΑΡΑΓΩΓΗΣ ΚΑΙ ΔΙΟΙΚΗΣΗΣ

Επίλυση του Προβλήματος Δρομολόγησης Οχημάτων με Χρονικά Παράθυρα με τον Αλγόριθμο Περιορισμένης Αναζήτησης (Tabu Search)

Συγγραφέας:

ΚΟΛΛΙΑΣ ΚΩΝΣΤΑΝΤΙΝΟΣ

Επιβλέπων Καθηγητής:

ΔΡ. ΜΑΡΙΝΑΚΗΣ ΙΩΑΝΝΗΣ

Χανιά 2011

ΠΕΡΙΕΧΟΜΕΝΑ

Περίληψη

Στη παρούσα διπλωματική κατασκευάζεται ένας Αλγόριθμος στο περιβάλλον προγραμματισμού Matlab, ο οποίος επιλύει το Πρόβλημα Δρομολόγησης Οχημάτων με Χρονικά Παράθυρα. Στον Αλγόριθμο που δημιουργήθηκε χρησιμοποιούνται κατά κύριο λόγο ευρετικές και μεθευρετικές μέθοδοι αναζήτησης. Στις ακόλουθες σελίδες λοιπόν θα αναλυθεί τόσο η φιλοσοφία και ο τρόπος λειτουργίας του Αλγορίθμου όσο και η απόδοση του σε ένα σύνολο πρότυπων προβλημάτων.

1. Εισαγωγή

Η παγκοσμιοποίηση, καθώς και η ραγδαία εξέλιξη της τεχνολογίας και των παραγωγικών διαδικασιών, οδηγούν σε έναν συνεχώς αυξανόμενο ανταγωνισμό μεταξύ

των επιχειρήσεων. Οι απαιτήσεις των πελατών, ως προς την ποιότητα, την τεχνική υποστήριξη αλλά και την άμεση εξυπηρέτηση διαρκώς εκτείνονται, ενώ μέσω του ηλεκτρονικού επιχειρείν δίνεται η δυνατότητα σε κάθε εταιρεία να απευθύνεται σε ένα αρκετά μεγαλύτερο αγοραστικό κοινό το οποίο σε αρκετές περιπτώσεις είναι παγκόσμιας κλίμακας, γεγονός που δεν συνέβαινε πριν την εμφάνιση του Διαδικτύου.

Στόχος λοιπόν της κάθε επιχείρησης είναι να χρησιμοποιήσει τους πόρους που διαθέτει ώστε να αυξήσει το μερίδιο αγοράς της και κατ' επέκταση τα κέρδη της υπερισχύοντος των ανταγωνιστών του αντίστοιχου κλάδου.

Προς πραγματοποίηση του παρακάτω σκοπού οι διαθέσιμοι πόροι μπορούν να χρησιμοποιηθούν ως εξής:

- Επένδυση στη διαφήμιση: με αυτό τον τρόπο επιχειρείται αύξηση των πωλήσεων γνωστοποιώντας το προϊόν – υπηρεσία στην αγορά και παρουσιάζοντάς το με τρόπο που να φαίνεται ως μία πολύ αξιόλογη αγοραστική επιλογή.
- Επέκταση της επιχείρησης
- Επένδυση στην έρευνα: η επιχείρηση, διαθέτει πόρους για βελτίωση των παραγωγικών διαδικασιών και των υπηρεσιών που προσφέρει.

Εφόσον η έρευνα αποδώσει οι παραγωγικές διαδικασίες και υπηρεσίες της επιχείρησης θα είναι πιο αποδοτικές με αποτέλεσμα τη μείωση του κόστους. Η μείωση του κόστους μπορεί να χρησιμοποιηθεί από την επιχείρηση με δύο τρόπους: Είτε μειώνοντας την τιμή του προϊόντος, προσελκύοντας έτσι περισσότερους πελάτες, είτε διατηρώντας την τιμή του προϊόντος με αποτέλεσμα την αύξηση του ποσοστού κέρδους επί των πωλήσεων.

Εάν αναλογιστεί κανείς ότι κάθε μέρα η τεχνολογία εξελίσσεται, γίνεται εύκολα κατανοητό ότι κάθε επιχείρηση πρέπει να βρίσκεται σε διαρκή προσπάθεια ώστε να παραμείνει ανταγωνιστική. Είναι πλέον εμφανές πως η βελτίωση των παραγωγικών διαδικασιών είναι θέμα υψίστης σημασίας για την βιωσιμότητα κάθε επιχείρησης. Η εφοδιαστική αλυσίδα λοιπόν και ο σωστός σχεδιασμός της δύναται να δημιουργήσει ισχυρά πλεονεκτήματα και να καθορίσει τις σχέσεις υπεροχής στην αγορά.

[1] Η εφοδιαστική αλυσίδα είναι η διαδικασία του σχεδιασμού, της εφαρμογής και του ελέγχου της αποτελεσματικής ροής και αποθήκευσης πρώτων υλών, ημικατεργασμένων προϊόντων, τελικών προϊόντων και της σχετιζόμενης πληροφορίας από το σημείο προέλευσης στο σημείο κατανάλωσης με σκοπό τη συμμόρφωση στις απαιτήσεις των πελατών. Η αποστολή της εφοδιαστικής αλυσίδας είναι να μεταφερθούν τα κατάλληλα προϊόντα ή οι κατάλληλες υπηρεσίες στο κατάλληλο μέρος, την κατάλληλη ώρα και στην επιθυμητή κατάσταση και συγχρόνως να πραγματοποιείται μέγιστη απόδοση για την εταιρεία. Οι δραστηριότητες της εφοδιαστικής αλυσίδας, λειτουργούν ως σύνδεσμος

μεταξύ της παραγωγής και της κατανάλωσης και των τοποθεσιών που βρίσκονται οι αγοραστές ή οι προμηθευτές που χωρίζονται τοπικά και χρονικά.

Μερικές από τις πιο βασικές λειτουργίες της διαχείρισης της εφοδιαστικής αλυσίδας είναι η επιλογή των προμηθευτών, ο έλεγχος των αποθεμάτων, η διαχείριση των αποθηκών, η διαχείριση των υλικών ανάμεσα σε όλες τις λειτουργίες, η επιλογή της κατάλληλης αποθήκης, η διάταξη των αποθηκών και φυσικά η μεταφορά των πρώτων υλών και των προϊόντων ανάμεσα στους προμηθευτές, τις αποθήκες, τα καταστήματα και φυσικά προς τους τελικούς καταναλωτές. Καθώς, λόγω των παγκοσμιοποιημένων παραγωγικών διαδικασιών, οι πηγές πρώτων υλών, τα εργοστάσια και τα σημεία πώλησης δεν βρίσκονται στα ίδια σημεία αλλά αντιθέτως ενδέχεται να βρίσκονται σε διαφορετικές ηπείρους, οι δραστηριότητες της εφοδιαστικής αλυσίδας επαναλαμβάνονται πολλές φορές πριν φτάσει ένα προϊόν στην αγορά[1].

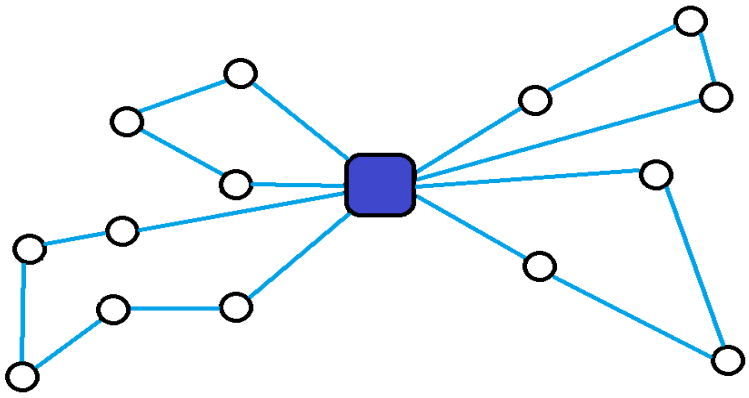
Τα αποθέματα και οι μεταφορές αποτελούν κύριες δραστηριότητες της εφοδιαστικής αλυσίδας καθώς απορροφούν το μεγαλύτερο κόστος στην επιχείρηση. Η μεταφορά φορτίου έχει παρατηρηθεί ότι απορροφά μεταξύ του 1/3 και των 2/3 του συνολικού κόστους της εφοδιαστικής αλυσίδας. Για να βελτιώσουμε τον συγκεκριμένο τομέα, οφείλουμε τόσο να εξελίξουμε τα μέσα μεταφοράς, όσο και να βελτιώσουμε τις μεθόδους με τις οποίες μπορούμε να υπολογίσουμε ποιες διαδρομές πρέπει να ακολουθηθούν ώστε να έχουμε λιγότερη σπατάλη πόρων. Σημαντική είναι η προώθηση λιγότερο ενεργοβόρων τρόπων μεταφοράς φορτίων, καθώς και η χρήση τεχνολογιών αξιοποίησης των ανανεώσιμων πηγών ενέργειας που μας χαρίζονται απλόχερα.

Στην παρούσα εργασία γίνεται μια προσπάθεια βελτίωσης του σχεδιασμού των μεταφορών με σκοπό την μείωση του κόστους. Κάτι τέτοιο επιτυγχάνεται μέσω της χρήσης όσο το δυνατόν λιγότερων μέσων μεταφοράς, εργατικού δυναμικού και ενεργειακών απαιτήσεων για την μεταφορά ορισμένης ποσότητας προϊόντος, σε ορισμένα σημεία και σε ορισμένα χρονικά διαστήματα.

2. Εισαγωγή στο Πρόβλημα VRPTW

2.1 Το Πρόβλημα Δρομολόγησης Οχημάτων.

Το Πρόβλημα Δρομολόγησης Οχημάτων (Vehicle Routing Problem) ή εν συντομία VRP είναι μία γενική ονομασία που έχει αποδοθεί σε ένα σύνολο ομοειδών προβλημάτων. Στα προβλήματα αυτά καλούμαστε να εξυπηρετήσουμε ένα σύνολο πελατών γεωγραφικά διασκορπισμένων στο χώρο, σε μία συγκεκριμένη χρονική περίοδο, χρησιμοποιώντας έναν στόλο οχημάτων τα οποία έχουν κοινή αφετηρία μια συγκεκριμένη αποθήκη. Ο αντικειμενικός στόχος είναι να εξυπηρετηθούν όλοι οι πελάτες, εκπληρώνοντας τις εκάστοτε απαιτήσεις, με τον πιο οικονομικό τρόπο, ενώ οι διαδρομές των οχημάτων θα πρέπει να ξεκινούν και να τερματίζουν στην αποθήκη. Ακόμη κάθε όχημα επισκέπτεται κάθε πελάτη μία μόνο φορά καθώς και κάθε διαδρομή μπορεί να εκτελεστεί μόνο από ένα όχημα. Για να αναπαραστήσουμε ένα δίκτυο μεταφοράς αναπαριστούμε τους πελάτες με κόμβους και τις διαδρομές μεταξύ των πελατών με τόξα, όπως φαίνεται στο παρακάτω σχήμα.



Το πρόβλημα Δρομολόγησης οχημάτων επεκτείνεται όσο οι παράμετροι που πρέπει να εξεταστούν κατά την επίλυση του προβλήματος αυξάνουν. Έχοντας λοιπόν διάφορους συνδυασμούς απαιτήσεων και περιορισμών, προκύπτουν ορισμένες παραλλαγές του Προβλήματος Δρομολόγησης Οχημάτων. Η περίπτωση για παράδειγμα όπου ο αριθμός των αποθηκών είναι μεγαλύτερος του ένα αποτελεί μία επέκταση του Προβλήματος Δρομολόγησης Οχημάτων. Το συγκεκριμένο πρόβλημα συναντάται συχνά στην πράξη ενώ στη βιβλιογραφία αναφέρεται ως Multidepot Vehicle Routing Problem.

2.2 Το Πρόβλημα Δρομολόγησης Οχημάτων με Χρονικά Παράθυρα

Η ιδιαιτερότητα του Προβλήματος Δρομολόγησης Οχημάτων με Χρονικά Παράθυρα (Vehicle Routing Problem with Time Windows) είναι ότι για κάθε πελάτη-κόμβο, υπάρχει ένα χρονικό παράθυρο μέσα στο οποίο πρέπει να εξυπηρετηθεί. Τέτοιου είδους περιορισμοί συναντώνται πολύ συχνά σε πρακτικά προβλήματα. Στην περίπτωση των εμπορικών καταστημάτων για παράδειγμα, η τροφοδοσία δεν μπορεί να συμβεί οποιαδήποτε στιγμή εντός της ημέρας. Ιδιαίτερα σε κεντρικές περιοχές πόλεων ορίζονται απ΄το κράτος συγκεκριμένες πρωινές ώρες κατά τις οποίες η διαδικασία εξυπηρέτησης των καταστημάτων δε παρεμποδίζει την κυκλοφορία των υπόλοιπων οχημάτων και των πεζών. Οι ώρες λοιπόν που επιλέγονται θα πρέπει να απέχουν από τις ώρες αιχμής. Πολλές βέβαια επιχειρήσεις ορίζουν οι ίδιες τα χρονικά παράθυρα που τους συμφέρουν, ανάλογα με το προϊόν που μεταφέρεται και την διαδικασία παράδοσης.

Όπως αναφέρθηκε, στη συγκεκριμένη εργασία γίνεται ανάλυση και επίλυση του προβλήματος VRPTW με χρήση του αλγόριθμου Tabu search. Προϋπόθεση όμως για τον σωστό προγραμματισμό του αλγόριθμου είναι η μελέτη και η πλήρης κατανόηση του προβλήματος. Ας εξετάσουμε λοιπόν το πρόβλημα πιο αναλυτικά.

Έχουμε ένα σύνολο πελατών τους οποίους καλούμαστε να εξυπηρετήσουμε, κάθε ένας πελάτης έχει μία συγκεκριμένη ζήτηση η οποία εκφράζεται σε μονάδες προϊόντος. Σε περίπτωση όπου δεν έχουμε διακριτές μονάδες εμπορεύματος όπως στις περιπτώσεις μεταφοράς ρευστών, τότε η ζήτηση εκφράζεται συνήθως σε μονάδες μέτρησης όγκου (L) για τα ρευστά και σε μονάδες μέτρησης μάζας (kg)για τα στερεά.

Ακόμη για κάθε πελάτη υπάρχει ένας συγκεκριμένος χρόνος εξυπηρέτησης. Για παράδειγμα, εάν μία επιχείρηση διαθέτει προηγμένα μέσα φόρτωσης-εκφόρτωσης, είναι λογικό να έχει μικρότερο χρόνο εξυπηρέτησης.

Τέλος, κάθε πελάτης έχει συγκεκριμένο χρονικό παράθυρο μέσα στο οποίο επιτρέπεται να συμβεί η άφιξη του οχήματος μεταφοράς. Κάθε χρονικό παράθυρο χαρακτηρίζεται από έναν ενωρίτερο και έναν βραδύτερο χρόνο.

Η αποθήκη, καθώς αποτελεί επίσης κόμβο, διαθέτει χρονικό Παράθυρο, το οποίο ουσιαστικά εκφράζει το μέγιστο χρονικό διάστημα που δύναται να διαρκεί κάθε διαδρομή. Δηλαδή, κάθε ένα από τα οχήματα που χρησιμοποιούνται θα πρέπει να έχει επιστρέψει στην αποθήκη σε χρονική στιγμή μικρότερη από τον βραδύτερο χρόνο του

«Παραθύρου» της Βάσης. Ακόμη, ο χρόνος εξυπηρέτησης της βάσης συνήθως θεωρείται μηδενικός, όπως ακριβώς συμβαίνει και με την ζήτηση.

Το σύνολο λοιπόν των πελατών θα πρέπει να εξυπηρετηθεί από έναν στόλο οχημάτων. Όπως είναι φανερό, ο αριθμός των οχημάτων που έχουμε την δυνατότητα να χρησιμοποιήσουμε είναι περιορισμένος. Συγκεκριμένη επίσης είναι και η χωρητικότητα του κάθε οχήματος. Σαφώς κάποιο όχημα με φορτίο μικρότερο της ζήτησης κάποιου κόμβου δεν έχει την δυνατότητα να εξυπηρετήσει τον κόμβο αυτό. Η χωρητικότητα εκφράζεται στις ίδιες μονάδες με την ζήτηση.

Στόχος του προβλήματος είναι να χρησιμοποιήσουμε όσο το δυνατόν λιγότερα οχήματα για την εξυπηρέτηση όλων των κόμβων καθώς και να ελαχιστοποιήσουμε την συνολική διανυθείσα απόσταση. Θα πρέπει κάθε διαδρομή να είναι κυκλική με αρχή και τέλος την αποθήκη. Ενώ από κάθε πελάτη, θα διέρχεται μόνο ένας «κύκλος».

Έστω λοιπόν M ο αριθμός των κόμβων-πελατών και ν ο αριθμός των οχημάτων. Ονομάζουμε $\kappa_{\alpha\beta}$ την απόσταση απευθείας μετάβασης από τον κόμβο α στον β. Ενώ με X αναπαριστούμε την χωρητικότητα κάθε οχήματος. Το πρόβλημα λοιπόν αποτελεί ένα πρόβλημα ακέραιου προγραμματισμού. Η αντικειμενική συνάρτηση είναι:

$$min \sum_{\theta=1}^{M} \sum_{\lambda=1}^{M} \sum_{\rho=1}^{\nu} \kappa_{\theta\lambda}\, \delta_{\theta\lambda\rho}$$

Όπου $\delta_{\theta\lambda\rho}$ η μεταβλητή απόφασης που υποδηλώνει ποια διαδρομή θα εκτελεστεί και από ποιο όχημα. Παρακάτω ακολουθούν οι περιορισμοί του προβλήματος

$$\delta_{\theta\lambda\rho} \in \{0,1\} \qquad \forall\, \rho \in 1, \dots, \nu \;\kappa\alpha\iota$$

$$\forall\, \theta \in 1, \dots, M \;\kappa\alpha\iota \qquad (1)$$

$$\forall\, \lambda \in 1, \dots, M$$

όπου:

$$\delta_{\theta\lambda\rho} = \begin{cases} 1, Αν\ το\ όχημα\ \rho\ από\ τον\ κόμβο\ \theta\ κατευθύνεται\ προς\ τον\ κόμβο\ \lambda \\ 0, Αλλιώς \end{cases}$$

$$n_a \leq t_{a,\rho} \leq v_a \qquad \forall\ \alpha\ \in 1, \dots, M\ \ και \qquad (2)$$

$$\forall\ \rho\ \in 1, \dots, \nu$$

$$\sum_{\theta=1}^{M} \sum_{\lambda=1}^{\nu} \delta_{\theta\lambda\rho} = 1 \qquad \forall\ \lambda\ \in 2, \dots, M \qquad (3)$$

$$\sum_{\lambda=1}^{M} \delta_{\theta\lambda\rho} - \sum_{\lambda=1}^{M} \delta_{\theta\lambda\rho} = 0 \qquad \forall\ \theta\ \in 1, \dots, M\ και \qquad (4)$$

$$\forall\ \rho \in 1, \dots, \nu$$

$$\sum_{\theta=1}^{M} \sum_{\lambda=1}^{M} \delta_{\theta\lambda\rho} \zeta_\theta \leq X \qquad \forall\ \rho \in 1, \dots, \nu \qquad (5)$$

$$\sum_{\lambda=1}^{M}\sum_{\rho=1}^{\nu}\delta_{1\lambda\rho}=\nu \qquad (6)$$

$$\sum_{\theta,\lambda\in SM}\delta_{\theta\lambda\rho}\leq |M|-1 \qquad \forall\ \rho\in 1,\dots,\nu \quad \kappa\alpha\iota \qquad (7)$$

$$\forall\ SM \subseteq \{2,\dots,M\}$$

Στον πρώτο περιορισμό βλέπουμε ότι η μεταβλητή απόφασης μπορεί να πάρει δύο τιμές. 1 εάν η συγκεκριμένη διαδρομή θλ εκτελείται από το όχημα ρ και 0 εάν κάτι τέτοιο δεν συμβαίνει. Στον περιορισμό (2) εξασφαλίζεται ότι η άφιξη του οχήματος ν στον κόμβο α θα συμβεί εντός του χρονικού παραθύρου που ορίζεται για τον κόμβο α. Προφανώς με n_α συμβολίζεται ο ενωρίτερος χρόνος ενώ με ν_α ο βραδύτερος. Στον περιορισμό 3 βλέπουμε πως κάθε κόμβος εξυπηρετείται μία μόνο φορά και κάθε διαδρομή εκτελείται το πολύ από ένα όχημα. Στον περιορισμό 4 βλέπουμε ότι το όχημα που εισέρχεται στον κόμβο είναι και εκείνο που εξέρχεται από αυτόν. Στον περιορισμό 5 απαιτείται το άθροισμα της ζήτησης των κόμβων μιας διαδρομής να είναι μικρότερο από την μέγιστη χωρητικότητα του οχήματος που την εκτελεί. Στον έκτο περιορισμό φαίνεται ότι ο αριθμός των οχημάτων που αναχωρούν από την αποθήκη είναι ίσος με ν ενώ ο τελευταίος περιορισμός δείχνει ότι ο αριθμός των διαδρομών σε μία κυκλική διαδρομή θα πρέπει να είναι μικρότερος του αριθμού των κόμβων μειωμένο κατά 1.

3. Βασικοί Αλγόριθμοι που χρησιμοποιήθηκαν

Η γενική διαδικασία που ακολουθήθηκε για την επίλυση του προβλήματος είναι η εξής: Αρχικά δημιουργούμε μία εφικτή λύση με την χρήση ενός ευρετικού αλγορίθμου και στη συνέχεια γίνεται προσπάθεια ώστε να βελτιώσουμε την λύση αυτή, με την βοήθεια ενός μεθευρετικού αλγόριθμου. Όπως θα δούμε στη συνέχεια, για την κατασκευή της αρχικής λύσης χρησιμοποιήθηκε ένας Αλγόριθμος απληστίας, ενώ για την βελτίωσή της χρησιμοποιήθηκε ο αλγόριθμος τοπικής αναζήτησης Tabu Search.

3.1 Αλγόριθμος Απληστίας

Οι Αλγόριθμοι απληστίας χρησιμοποιούνται για την δημιουργία εφικτών υποβέλτιστων λύσεων και είναι σχετικά απλής λογικής. Η λύση αναπτύσσεται βηματικά. Αρχίζοντας από μια μερική λύση, για παράδειγμα έναν κόμβο, προχωρούμε ελέγχοντας ποιος από τους υπόλοιπους κόμβους μπορεί να ακολουθήσει, χωρίς να παραβιαστεί κανένας περιορισμός, ενώ παράλληλα μεγιστοποιείται ή ελαχιστοποιείται ανάλογα κάποια συνάρτηση, η οποία μπορεί να αποτελεί και την αντικειμενική συνάρτηση. Με αυτό τον τρόπο όλες οι ενδιάμεσες μερικές λύσεις είναι πάντοτε εφικτές. Όταν ολοκληρωθεί η διαδικασία θα έχει δημιουργηθεί η πλήρης λύση, δηλαδή ένα σύνολο διαδρομών οι οποίες εξυπηρετούν όλους τους κόμβους τηρώντας παράλληλα τους περιορισμούς του προβλήματος. Όμως το αποτέλεσμα του αλγόριθμου δεν είναι το βέλτιστο, παρόλο που σε κάθε βήμα επιλέγεται να προστεθεί στη τρέχουσα διαδρομή ο κόμβος εκείνος που οδηγεί στην καλύτερη μερική εφικτή λύση.

Είναι σύνηθες, το αποτέλεσμα ενός ευρετικού αλγόριθμου να αποτελεί δεδομένο εισαγωγής σε έναν μεθευρετικό αλγόριθμο. Ο μεθευρετικός αλγόριθμος βελτιώνει σταδιακά την λύση που παραλαμβάνει από τον αλγόριθμο απληστίας πραγματοποιώντας εναλλαγές στην αλληλουχία εξυπηρέτησης των κόμβων. Η τακτική που ακολουθείται για την πραγματοποίηση της βελτιστοποίησης είναι χαρακτηριστικό κάθε μεθευρετικού αλγόριθμου.

3.2 Περιορισμένη Αναζήτηση (Tabu Search)

Ο αλγόριθμος περιορισμένης αναζήτησης είναι ίσως ο πιο γνωστός μεθευρετικός αλγόριθμος. Ο Tabu Search, όπως συνηθίζεται να λέγεται, χρησιμοποιεί έναν ευρετικό αλγόριθμο ώστε να μετακινηθεί από την μία λύση στην άλλη. Δηλαδή, υπολογίζει τις εναλλαγές των κόμβων που θα μπορούσαν να συμβούν στην τρέχουσα λύση και

εφαρμόζεται η κίνηση η οποία είναι η καλύτερη ή τουλάχιστον η λιγότερο χειρότερη. Η πιθανότητα όμως να παγιδευτεί αυτή η διαδικασία σε τοπικό ελάχιστο είναι αρκετά μεγάλη, σε όλους τους μεθευρετικούς αλγόριθμους. Κάτι τέτοιο θα είχε ως αποτέλεσμα να επαναλαμβάνονται περιοδικά συγκεκριμένες κινήσεις-εναλλαγές ώστε να επανεμφανίζονται οι ίδιες τρέχουσες λύσεις και έτσι να μην συμβαίνει καμία βελτίωση της αντικειμενικής συνάρτησης. Το σημαντικό χαρακτηριστικό λοιπόν του αλγόριθμου περιορισμένης αναζήτησης είναι τρόπος με τον οποίο αποφεύγονται αυτές οι παγίδες των τοπικών ελαχίστων. Η στρατηγική που ακολουθείται εκμεταλλεύεται τις πληροφορίες μιας πρόσκαιρης μνήμης του αλγόριθμου. Δηλαδή ο αλγόριθμος «θυμάται» τις τελευταίες κινήσεις που πραγματοποιήθηκαν και απαγορεύει την επανεκτέλεσή τους. Η λίστα στην οποία αποθηκεύονται οι πρόσφατες κινήσεις ονομάζεται λίστα περιορισμένης αναζήτησης (Tabu List) ενώ αυτή η διαδικασία ονομάζεται μνήμη μικρής περιόδου. Με αυτή την μέθοδο αποκλείουμε την επανεμφάνιση ιδίων λύσεων μέσα σε σύντομο χρονικό διάστημα. Ωστόσο υπάρχει μία περίπτωση κατά την οποία οι παραπάνω περιορισμοί αγνοούνται. Ονομάζεται κριτήριο απενεργοποίησης περιορισμών και δίνει την δυνατότητα επιλογής και εκτέλεσης μιας περιορισμένης κίνησης, εφόσον οδηγεί σε λύση η οποία υπερέχει της, έως εκείνη τη στιγμή, βέλτιστης. Σημαντικός παράγοντας στην επιτυχία της μεθόδου είναι η διάρκεια της μνήμης, δηλαδή για πόσα βήματα θα θεωρείται περιορισμένη μια κίνηση που μόλις επιλέχτηκε. Ενδέχεται η διάρκεια της μνήμης να επιλέγεται ανάλογα με το μέγεθος του προβλήματος, δηλαδή με τον αριθμό των προς εξυπηρέτηση κόμβων, είτε να αλλάζει δυναμικά κατά την διάρκεια του προγράμματος. Πολλές φορές επίσης εφαρμόζεται και η στρατηγική της διάχυσης, μηχανισμός ο οποίος προσπαθεί να οδηγήσει την αναζήτηση σε ανεξερεύνητες περιοχές. Συνήθως χρησιμοποιεί μια μνήμη μακράς διάρκειας, όπου αποθηκεύονται πληροφορίες από τις περιοχές που έχουν ήδη εξερευνηθεί. Η πιο σημαντική στρατηγική διαφοροποίησης ελέγχει τη συχνότητα επιλογής κάθε εναλλαγής κόμβων – κίνηση στο παρελθόν- και επιβάλει την εφαρμογή κινήσεων που δεν είχαν συμβεί έως εκείνη τη στιγμή, ανεξάρτητα από το αν βελτιστοποιούν ή όχι τη λύση του προβλήματος. Κάτι τέτοιο επιτυγχάνεται με τη χρήση της μνήμης συχνοτήτων όπου, σε κάθε βήμα του αλγορίθμου, αποθηκεύεται η κίνηση που εκτελέστηκε.

Η περιορισμένη αναζήτηση δεν συγκλίνει με φυσικό τρόπο, επομένως καθορίζεται ένας αριθμός επαναλήψεων που πρέπει να εκτελέσει το πρόγραμμα πριν τον τερματισμό του. Ακολουθεί ο ψευδοκώδικας του αλγορίθμου

Αλγόριθμος Περιορισμένη Αναζήτηση

Αρχικοποίηση

Κατασκευή μιας αρχικής λύσης

Υπολογισμός της συνάρτησης κόστους της λύσης

S* = So ! Αρχικοποίηση της βέλτιστης λύσης

f(S*) = f(So)

Κύρια φάση

Do While κάποιο κριτήριο σταματήματος δεν έχει ικανοποιηθεί

Υπολογισμός μίας γειτονικής λύσης S'

if f(S') **<** f**(**S***)* **then**

S* = S'

f* = f(S')

endif

Αποθήκευσε την τελευταία κίνηση στη λίστα περιορισμένων υποψηφίων

(ταυτόχρονα αν έχει συμπληρωθεί το μέγεθος της λίστας διέγραψε την παλαιότερη εγγραφή)

Κάλεσε κάθε k επαναλήψεις τη στρατηγική εντατικοποίησης

if f(S intersification)> f(S*) **then**

S* = S intersification

f* = f*(S intersification)

endif

Endo

Επέστρεψε τη βέλτιστη λύση S*.

Στη συνέχεια θα παρουσιαστούν αναλυτικά οι αλγόριθμοι που κατασκευάστηκαν κατά την επίλυση του Προβλήματος Δρομολόγησης Οχημάτων με χρονικά παράθυρα.

4. Περιγραφή του Αλγορίθμου

4.1 Ανάλυση Προβλήματος και Αρχικοποίηση Μεταβλητών

Στόχος μας είναι να δημιουργήσουμε έναν αλγόριθμο που επιλύει το VRPTW. Αρχικά λοιπόν θα πρέπει να εξετάσουμε το πρόβλημα και να αποθηκεύσουμε τα δεδομένα. Έχουμε ένα σύνολο κόμβων, κάθε κόμβος έχει συγκεκριμένη ζήτηση, συγκεκριμένη θέση στο χώρο, συγκεκριμένο χρόνο εξυπηρέτησης, καθώς και ένα χρονικό παράθυρο μέσα στο οποίο εξυπηρετείται. Ακόμη έχουμε ένα στόλο οχημάτων όπου κάθε ένα έχει συγκεκριμένη χωρητικότητα. Για την αποθήκευση των παραπάνω στοιχείων χρησιμοποιούμε πίνακες μονοδιάστατους και δυσδιάστατους. Συνήθως τα αρχικά δεδομένα είναι κατανεμημένα σε πίνακες του προγράμματος Excel, επομένως γίνεται χρήση της συνάρτησης xlsread ώστε να διαβαστούν και να αποθηκευτούν σε πίνακες στην Matlab.Για παράδειγμα, η εντολή demand=xlsread (‘input_data.xls’,1,’D2:D102’), διαβάζει από το αρχείο input_data τα στοιχεία από΄το κελί D2 έως το κελί D102 και τα αποθηκεύει στον πίνακα demand. Με αυτόν τον τρόπο, δημιουργείται ένας μονοδιάστατος πίνακας (demand), όπου κάθε στοιχείο του αντιστοιχεί στη ζήτηση του αντίστοιχου κόμβου. Ομοίως εργαζόμαστε και για τα υπόλοιπα δεδομένα.

Με αυτό τον τρόπο λοιπόν σχηματίζονται οι παρακάτω δισδιάστατοι πίνακες.

Demand: πίνακας ζήτησης

Service: πίνακας χρόνων εξυπηρέτησης

Capacity: χωρητικότητα σχημάτων

Max_vehicle_number: μέγιστος αριθμός οχημάτων

e: πίνακας ενωρίτερων χρόνων

l: πίνακας βραδύτερων χρόνων

x: πίνακας τετμημένης

y: πίνακας τεταγμένης

Όπως βλέπουμε πιο πάνω το e(i), l (i) θα αποτελεί το χρονικό παράθυρο του κόμβου i. Σε πολλά προβλήματα επίσης, η χωρητικότητα των οχημάτων είναι κοινή επομένως

αποθηκεύεται ως μία σταθερά και όχι ως πίνακας. Στο συγκεκριμένο αλγόριθμο θεωρείται ως αποθήκη ο κόμβος 1, επομένως service(1)=0 και demand(1)=0. Επίσης, όπως γίνεται φανερό πιο πάνω, μας δίδονται η τετμημένη και η τεταγμένη κάθε κόμβου. Όμως θα πρέπει να υπολογίσουμε τις αποστάσεις μεταξύ των κόμβων. Κάτι τέτοιο γίνεται εφικτό με τις ακόλουθες εντολές

```
for i=1: length (x)
    for j=1: length (y)
        d(i,j) - [ (x(j) - x(i)) ^2 + (y(j) - y(i)) ^2 ] ^(1/2)
    end
end
```

Ως αποτέλεσμα έχουμε τον πίνακα αποστάσεων d όπου d(i,j) είναι η απόσταση του κόμβου i από τον κόμβο j. Τα προβλήματα στα οποία ισχύει d(i,j)=d(j,i) ονομάζονται συμμετρικά. Όμως, σύμφωνα με το πρόβλημα, δεν επιτρέπεται η παραμονή του οχήματος σε έναν κόμβο μετά την εξυπηρέτησή του. Επομένως θα πρέπει να απειριστούν οι αποστάσεις d(i,i) ώστε να αποκλειστεί αυτή η δυνατότητα. Η απαγόρευση παραμονής συμβαίνει με έμμεσο τρόπο. Επειδή το κριτήριο που θέλουμε να βελτιστοποιήσουμε είναι το κόστος είναι ασύμφορο να επιλέξουμε την διαδρομή i,i η οποία έχει άπειρο κόστος.

Δηλαδή

```
for  i=l:size(d,1)
        d(i,i)=inf;
end
```

Αφού λοιπόν εισαχθούν τα δεδομένα και οργανωθούν με τις προαναφερθείσες διαδικασίες, γίνεται αρχικοποίηση όλων των μεταβλητών που θα χρησιμοποιηθούν στον ευρετικό αλγόριθμο. Θα χρειαστούμε λοιπόν τις παρακάτω μεταβλητές:

load: είναι ένας μονοδιάστατος πίνακας όπου κάθε στοιχείο του αναπαριστά το φορτίο του οχήματος αφού εξυπηρετηθεί ο αντίστοιχος κόμβος. Δηλαδή το load(4) είναι το φορτίο που απομένει στο όχημα που εξυπηρετεί τον κόμβο 4, μετά την διαδικασία φόρτωσης-εκφόρτωσης στον κόμβο 4.

starting-time: μονοδιάστατος πίνακας όπου αποθηκεύεται η χρονική στιγμή έναρξης της εξυπηρέτησης του αντίστοιχου κόμβου. Θεωρούμε ότι starting-time(1)=0 καθώς τα οχήματα ξεκινούν από τον κόμβο 1 τη στιγμή t=0s

node-waiting-time: μονοδιάστατος πίνακας όπου αποθηκεύεται ο χρόνος αναμονής σε κάθε κόμβο, μέχρι την έναρξη της εξυπηρέτησής του.

eding-vehicle-time:μονοδιάστατος πίνακας όπου αποθηκεύεται η χρονική στιγμή που επιστρέφει στην αποθήκη, το αντίστοιχο όχημα.

Load-left:μονοδιάστατος πίνακας όπου αποθηκεύεται το φορτίο με το οποίο επιστρέφει στη βάση κάθε όχημα

Waiting-time: μονοδιάστατος πίνακας όπου αποθηκεύεται ο συνολικός χρόνος αναμονής κάθε οχήματος σε όλους τους κόμβους που εξυπηρέτησε.

Short-path: δισδιάστατος πίνακας όπου αποθηκεύεται η αρχική εφικτή λύση που προκύπτει από τον αλγόριθμο απληστίας. Κάθε γραμμή του πίνακα αναπαριστά τη διαδρομή ενός οχήματος. Το άθροισμα των γραμμών μας φανερώνει τον αριθμό των οχημάτων που χρειάστηκαν για την εξυπηρέτηση όλων των κόμβων.

Cost: μονοδιάστατος πίνακας όπου αποθηκεύεται η συνολική διανυθείσα απόσταση κάθε οχήματος

Cost-sum: η συνολική απόσταση που διανύθηκε από όλα τα οχήματα. Αποτελεί το μέγεθος εκείνο που προσπαθούμε να ελαχιστοποιήσουμε.

4.2 Δημιουργία Αρχικής Εφικτής Λύσης

Για την δημιουργία μιας αρχικής εφικτής λύσης, θα χρησιμοποιήσουμε έναν αλγόριθμο απληστίας και συγκεκριμένα τον αλγόριθμο πλησιέστερου γείτονα.

Στο συγκεκριμένο Πρόβλημα Δρομολόγησης Οχημάτων καλούμαστε να εξυπηρετήσουμε με την ζήτηση όλων των κόμβων, εντός των χρονικών περιορισμών σε χρονικό διάστημα που δεν θα υπερβαίνει τον βραδύτερο χρόνο της αποθήκης l(1). Με μία σύντομη επισκόπηση των δεδομένων ενός τέτοιου προβλήματος γίνεται εύκολα αντιληπτό πως είναι αδύνατη η εξυπηρέτηση όλων των κόμβων από ένα μόνο όχημα. Επομένως, θα πρέπει να υπολογίσουμε πόσα δρομολόγια θα έχουμε και ποια διαδρομή θα εκτελέσει κάθε δρομολόγιο. Κάτι τέτοιο καθίσταται εφικτό με την βοήθεια του αλγόριθμου του Πλησιέστερου Γείτονα. Επιτυγχάνεται δηλαδή σταδιακή δημιουργία της Αρχικής Λύσης. Σε κάθε βήμα του αλγόριθμου ελέγχεται ποίοι κόμβοι θα μπορούσαν να προστεθούν στο τρέχον δρομολόγιο και από αυτούς επιλέγεται ο κοντινότερος κόμβος ως προς τον τελευταίο της διαδρομής. Ας εξετάσουμε όμως την παραπάνω διαδικασία πιο αναλυτικά.

Όπως έχουμε προαναφέρει, κάθε όχημα ξεκινά από την αποθήκη την χρονική στιγμή t=0. Επομένως ο πρώτος κόμβος κάθε δρομολογίου είναι πάντοτε ο κόμβος 1. Το πρώτο σχήμα ξεκινά από τον κόμβο 1. Για να βρούμε ποιος πελάτης θα εξυπηρετηθεί πρώτος θα πρέπει να ελέγξουμε ποιοι μπορούν να εξυπηρετηθούν. Σε αυτό το βήμα, εφόσον τα δεδομένα του προβλήματος δεν είναι εσφαλμένα όλοι οι κόμβοι θα μπορούν να εξυπηρετηθούν. Αυτό συμβαίνει καθώς το σχήμα μόλις ξεκίνησε από την αποθήκη με πλήρες φορτίο. Δηλαδή μπορεί να εξυπηρετήσει τη ζήτηση κάθε κόμβου αλλά και το χρονικό παράθυρο κάθε κόμβου, αφού θα πρέπει να ισχύει d(1,node) <= l(node) για κάθε κόμβο (node) αλλιώς το πρόβλημα είναι αδύνατο. Στο βήμα λοιπόν αυτό θα επιλεγεί ο κόμβος εκείνος που βρίσκεται πιο κοντά στη βάση, σύμφωνα με το κριτήριο του Πλησιέστερου Γείτονα. Ακολούθως θα πρέπει να υπολογιστούν το φορτίο που απέμεινε στο φορτηγό μετά την εξυπηρέτηση του πρώτου πελάτη, το χρόνο έναρξης της εξυπηρέτησης, το χρόνο αναμονής πριν την εξυπηρέτηση καθώς και το τρέχων κόστος της διαδρομής. Έστω δηλαδή ότι επιλέγεται ο κοντινότερος στη βάση κόμβος node. Τότε αποθηκεύεται στην τρέχουσα διαδρομή: short_path(vehicle,2)=node; όπου vehicle ο αριθμός του οχήματος που εκτελεί το συγκεκριμένο δρομολόγιο που αναπτύσσουμε.

Ενημέρωση φορτίου: load(node) = load(p_node) - demand(node)

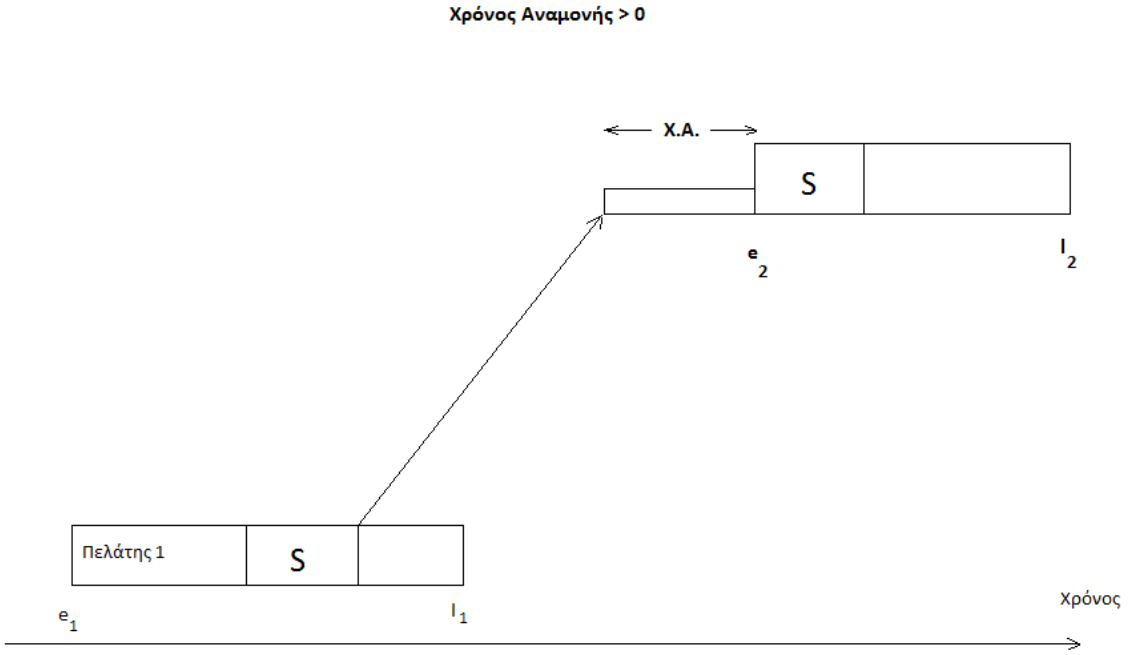
όπου p_node είναι ο προηγούμενος κόμβος, στο συγκεκριμένο βήμα δηλαδή p_node=1 και load(1)=capacity.

Ενημέρωση χρόνου έναρξης:

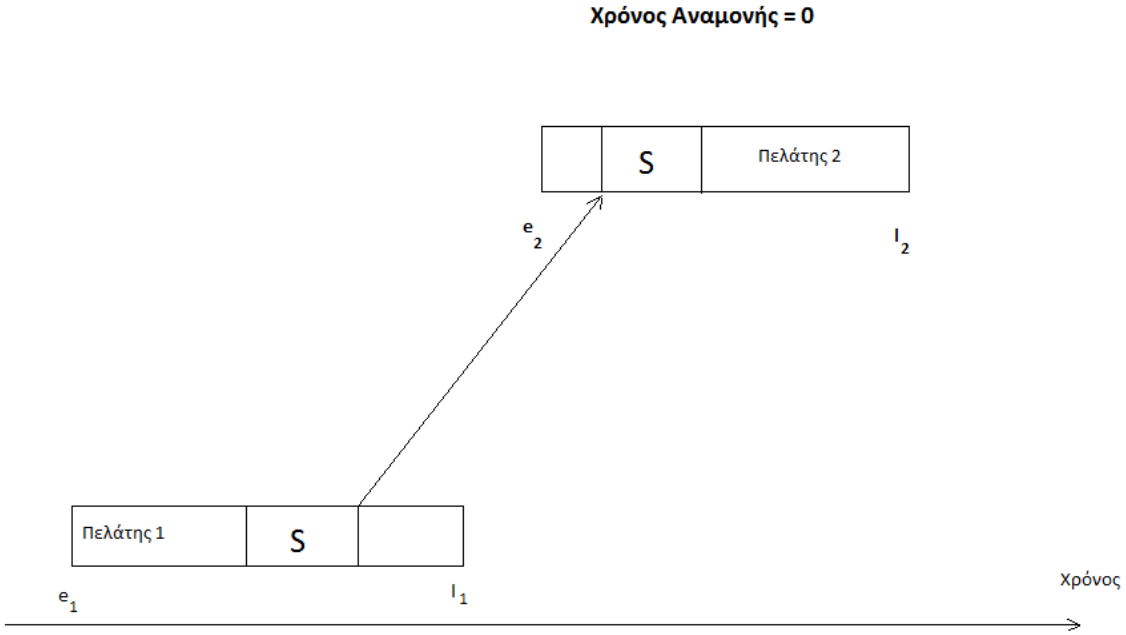
starting_time (node) =max (e (node), starting_time (p_node) +service (p_node) +

+d (p_ node, node));

Σύμφωνα με το πρόβλημα εάν το όχημα φτάσει πριν τον ενωρίτερο χρόνο εξυπηρέτησης στον κόμβο, περιμένει έως τη στιγμή e(node), ενώ εάν η άφιξή του είναι εντός του χρονικού παραθύρου, η εξυπηρέτηση ξεκινά αμέσως. Στην πρώτη περίπτωση (Σχήμα 1) ο χρόνος έναρξης της εξυπηρέτησης συμπίπτει με τον ενωρίτερο χρόνο, ενώ στη δεύτερη περίπτωση (Σχήμα 2) συμπίπτει με τον χρόνο άφιξης στον κόμβο.



Σχήμα 1



Σχήμα 2

Με παρόμοια λογική υπολογίζεται και ο χρόνος αναμονής στο συγκεκριμένο κόμβο

Node_waiting_time (node) =max (0, starting_time (node)-starting_time (p-node)

-serevice(p-node)-d (p-node, node)) ;

Όπως ειπώθηκε και πιο πάνω ο χρόνος έναρξης στον κόμβο 1 είναι μηδέν. Δηλαδή

starting_time(1)=0

Επίσης ενημερώνεται και το κόστος, δηλαδή η απόσταση που έχει διανύσει το όχημα έως τον κόμβο που μόλις προστέθηκε στη διαδρομή. Δηλαδή

cost (vehicle) =cost (vehicle) +d (p_node, node);

Όπου το αρχικό cost(vehicle) ισούται με το 0.

Αφού πραγματοποιηθεί η παραπάνω ενημέρωση, θα πρέπει να αποκλείσουμε το ενδεχόμενο να επιλεγεί ως επόμενος πελάτης ένας κόμβος που έχουμε ήδη επισκεφτεί. Επομένως, θέτουμε ως άπειρη την απόσταση μεταξύ του τελευταίου κόμβου και όλων όσων έχουν εξυπηρετηθεί προηγουμένως. Με αυτό τον τρόπο, εκμεταλλευόμενοι το

κριτήριο του πλησιέστερου γείτονα που επιλέγει τον κοντινότερο κόμβο, αποκλείουμε την επανεκλογή οποιουδήποτε κόμβου.

Όπως έγινε φανερό πιο πάνω η χιλιομετρική απόσταση μεταξύ δύο κόμβων, χρησιμοποιείται και σαν χρονική απόσταση στους υπολογισμούς των χρόνων έναρξης και αναμονής.

Έπειτα προχωρούμε στο επόμενο βήμα. Θέτουμε p_node=node και αρχίζοντας από τον κοντινότερο κόμβο, ως προς τον τελευταίο της διαδρομής, ελέγχουμε εάν θα μπορούσε να εξυπηρετηθεί από το συγκεκριμένο όχημα στις παρούσες συνθήκες φορτίου και χρόνου. Ο έλεγχος συνεχίζεται μέχρι να βρεθεί κάποιος κόμβος που να μπορεί να προστεθεί στην διαδρομή χωρίς να καταπατώνται οι περιορισμοί.

Στον έλεγχο που πραγματοποιείται, απαιτείται:

- Ο χρόνος έναρξης της εξυπηρέτησης του νέου κόμβου να μην υπερβαίνει τον βραδύτερο χρόνο εξυπηρέτησης του. Δηλαδή:
  (Starting-time (p_node) +service (p_node) +d (p_node, node)) <= l (node)

- Η ζήτηση του κόμβου να μην υπερβαίνει το τρέχον φορτίο του οχήματος δηλαδή:
  Load (p_node)-demand (node)>=0

- Τέλος θα πρέπει να υπάρχει χρόνος ώστε το όχημα να επιστρέψει εγκαίρως στην αποθήκη μετά την εξυπηρέτηση του νέου κόμβου. Δηλαδή:
  Starting_time (node) +service (node) +d (node, 1) <l (1)

Όταν βρεθεί κάποιος κόμβος που να μπορεί να προστεθεί στο δρομολόγιο, τότε ενημερώνονται όλες οι μεταβλητές και το συγκεκριμένο βήμα επαναλαμβάνεται. Εάν όμως δεν βρεθεί κάποιος κόμβος που να μπορεί να προστεθεί τότε θέτουμε τον κόμβο 1 ως τον τελευταίο κόμβο του δρομολογίου και υπολογίζονται:

- το φορτίο με το οποίο επέστρεψε το όχημα στη βάση load-left(vehicle)
- ο χρόνος επιστροφής στη βάση ending_vehicle_time(vehicle)
- το κόστος του δρομολογίου cost(vehicle)=cost(vehicle)+d(p-node,1)
- το άθροισμα του κόστους όσων δρομολογίων έχουν σχηματιστεί μέχρι την παρούσα στιγμή cost_sum=cost_sum+cost(vehicle)

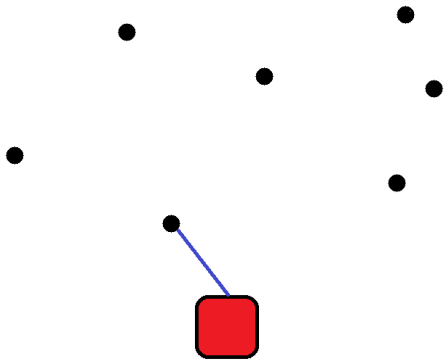
Τέλος απειρίζονται οι αποστάσεις μεταξύ όλων των κόμβων και των κόμβων του δρομολογίου που μόλις σχηματίστηκε ώστε να μην εξυπηρετηθούν ξανά. Εφόσον δεν έχουν κατανεμηθεί όλοι οι κόμβοι σε κάποια διαδρομή, ένα νέο δρομολόγιο ξεκινά από

τον κόμβο 1 με ένα νέο όχημα τη στιγμή t=0. Δεν πρέπει να ξεχνάμε ότι τα δρομολόγια εκτελούνται ταυτόχρονα, παρόλο που κατασκευάζονται διαδοχικά.

Η διαδικασία που περιγράφηκε αναπαρίσταται στα ακόλουθα Σχήματα.

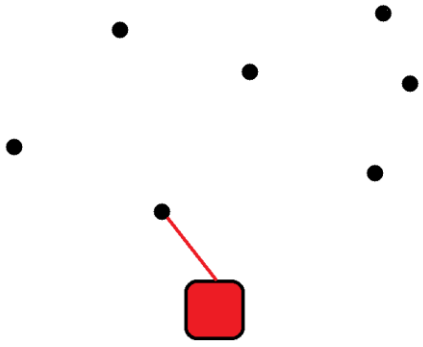
Βημα 1

Επιλέγεται ο κοντινότερος στη βάση κόμβος



Σχήμα 3

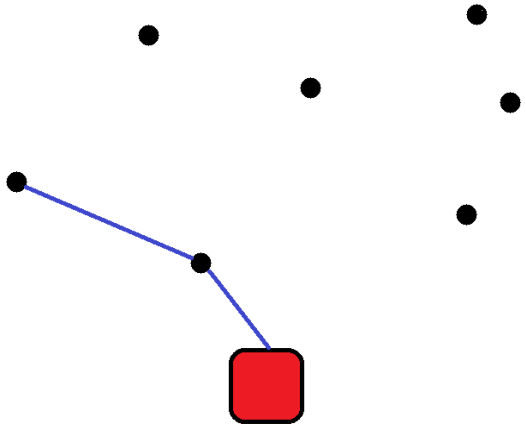
Απειρίζονται οι διαδρομές με κόκκινο χρώμα



Σχήμα 4

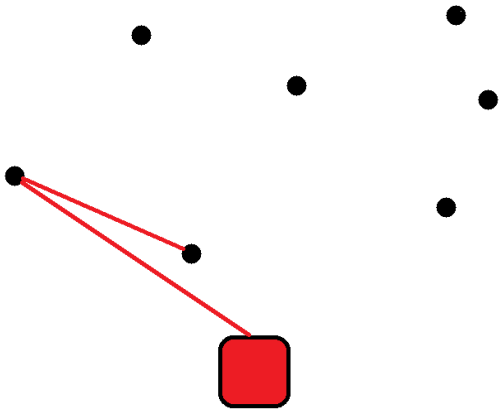
Βήμα 2

Επιλέγεται ο κοντινότερος κόμβος που μπορεί να εξυπηρετηθεί



Σχήμα 5

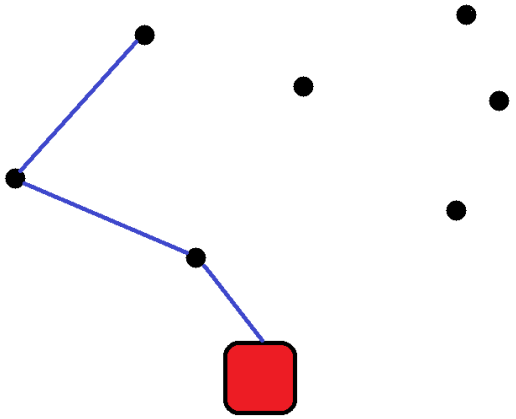
Απειρίζονται οι ακόλουθες διαδρομές ώστε να μην μπορεί το όχημα να επισκεφτεί τους κόμβους που έχει ήδη εξυπηρετήσει



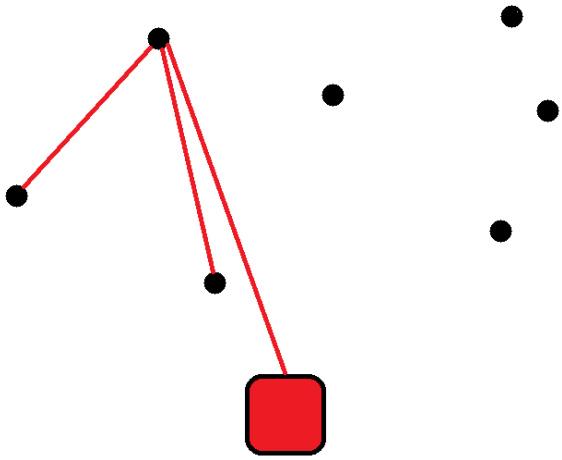
Σχήμα 6

Βήμα 3

Ομοίως



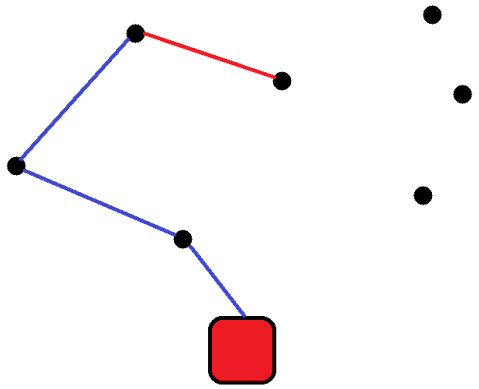
Σχήμα 7



Σχήμα 8

Βήμα 4

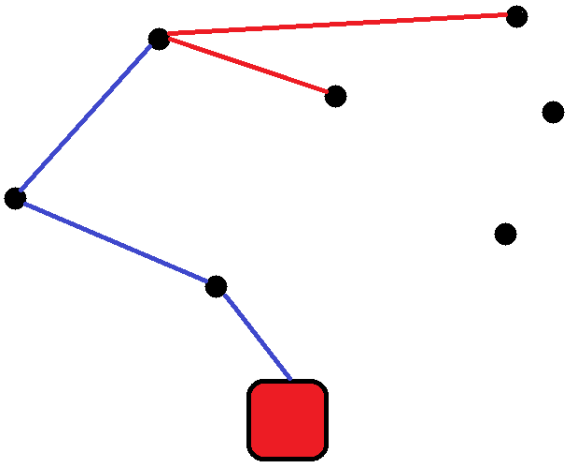
Ελέγχεται ο κοντινότερος στον τελευταίο κόμβο πελάτης για το αν θα μπορούσε να εξυπηρετηθεί. Το όχημα, είτε λόγω φορτίου, είτε λόγω χρόνου, αδυνατεί να εξυπηρετήσει τον συγκεκριμένο πελάτη. Όπως φαίνεται στο παρακάτω σχήμα, η διαδρομή προς αυτόν απειρίζεται, ώστε κατά την επανάληψη του συγκεκριμένου βήματος να ελεγχθεί ο κόμβος με την αμέσως μεγαλύτερη απόσταση από τον τελευταίο, που δεν έχει εξυπηρετηθεί ακόμα.



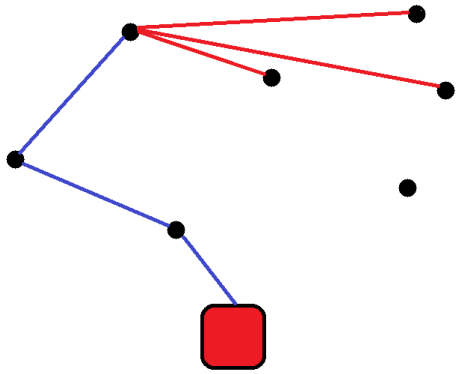
Σχήμα 9

Βήμα 4 Επανάληψη

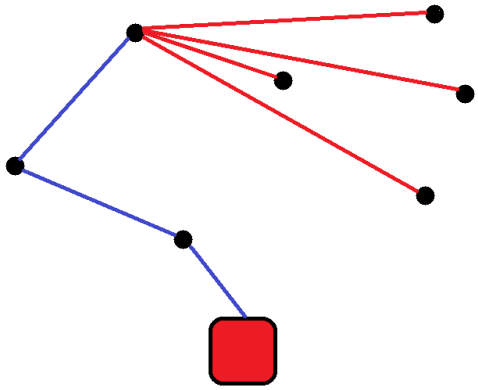
Όπως βλέπουμε στις επόμενες τρείς επαναλήψεις (Σχήμα 10 έως 12) κανένας από τους υπόλοιπους κόμβους δεν μπορεί να εξυπηρετηθεί από το τρέχον δρομολόγιο. Τέλος το όχημα επιστρέφει αναγκαστικά στην αποθήκη Σχήμα 13.



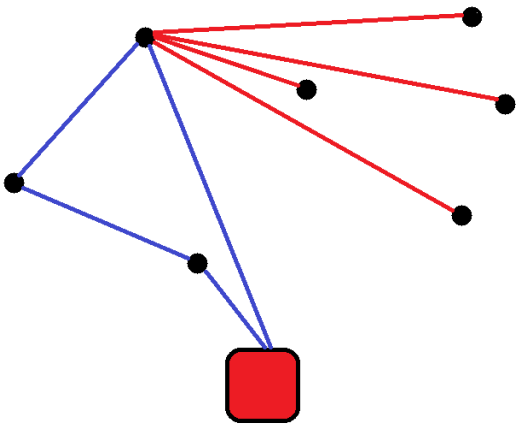
Σχήμα 10



Σχήμα 11



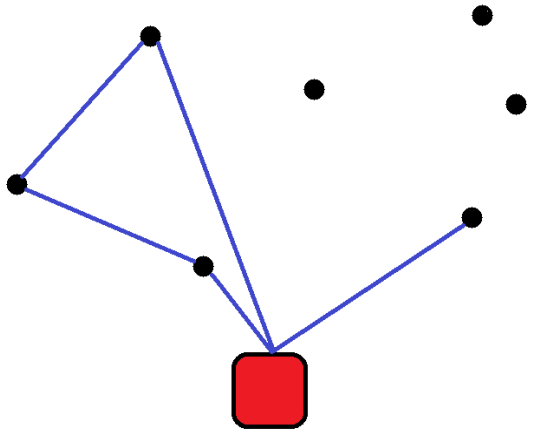
Σχήμα 12



Σχήμα 13

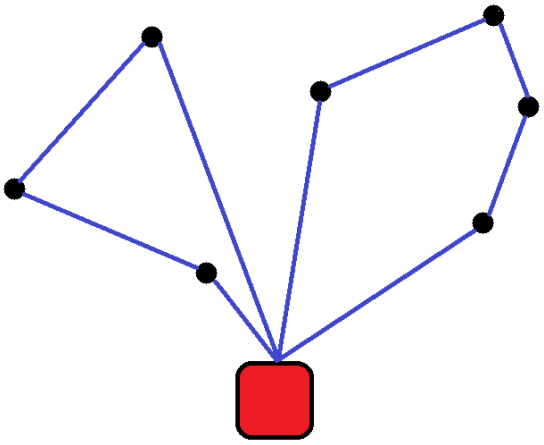
Βήμα 5

Ένα νέο δρομολόγιο ξεκινά με τον κοντινότερο στην αποθήκη πελάτη που δεν έχει εξυπηρετηθεί.



Σχήμα 14

Ομοίως εκτελούνται και τα επόμενα βήματα με αποτέλεσμα να προκύψει η ακόλουθη Εφικτή Λύση, η οποία διαθέτει δύο δρομολόγια.



Σχήμα 15

Ο αλγόριθμος που μόλις περιγράφηκε δημιουργεί εφικτές λύσεις, οι οποίες όμως απέχουν σημαντικά από το βέλτιστο. Σε ορισμένες περιπτώσεις μάλιστα η λύση που κατασκευάζεται από τον αλγόριθμο του Πλησιέστερου Γείτονα χρησιμοποιεί διπλάσιο σε αριθμό οχήματα για την εξυπηρέτηση όλων των κόμβων από ότι η Βέλτιστη Λύση. Για τον λόγο αυτό έγινε μία προσπάθεια αλλαγής του αλγόριθμου ώστε να επιτυγχάνει καλύτερα αποτελέσματα. Ο νεοτερισμός που φέρει ο κώδικας που τελικά χρησιμοποιήθηκε εντοπίζεται στο κριτήριο επιλογής του κόμβου που προστίθεται, σε κάθε βήμα, στο τρέχον δρομολόγιο. Όπως αναφέρθηκε, ο αλγόριθμος του πλησιέστερου Γείτονα επιλέγει πελάτες ανάλογα με την απόστασή τους από τον τελευταίο κόμβο της διαδρομής. Στον νέο αλγόριθμο που κατασκευάστηκε λόγω της ανάγκης για δημιουργία μιας καλύτερης αρχικής λύσης, οι κόμβοι αξιολογούνται τόσο ως προς την απόστασή τους από τον τελευταία εισηγμένο κόμβο όσο και από το χρονικό διάστημα μεταξύ της στιγμής άφιξης του οχήματος στον εκάστοτε κόμβο και του μέσου του αντίστοιχου χρονικού παραθύρου. Με αυτό τον τρόπο επιτυγχάνεται καλύτερος χρονικός προγραμματισμός στην εξυπηρέτηση των κόμβων κάθε διαδρομής, ο χρόνος αναμονής κάθε οχήματος μειώνεται σημαντικά ενώ χρησιμοποιούνται λιγότερα οχήματα για την εξυπηρέτηση όλων των κόμβων. Τα παραπάνω γίνονται άμεσα αντιληπτά από την εφαρμογή του αλγόριθμου. Σε ορισμένα προβλήματα μάλιστα η αρχική λύση που δημιουργούταν χρησιμοποιούσε τον ελάχιστο δυνατό αριθμό οχημάτων, ενώ το κόστος διαδρομής είχε αρκετά μικρή απόκλιση από το βέλτιστο, γεγονός που δεν συνέβαινε με τον αλγόριθμο του Πλησιέστερου Γείτονα. Ωστόσο, στόχος του ευρετικού αλγόριθμου είναι η δημιουργία μιας εφικτής λύσης και όχι η βελτιστοποίησή της. Μια καλύτερη αρχική λύση όμως δίνει περισσότερες πιθανότητες στον Αλγόριθμο τοπικής αναζήτησης να συνθέσει διαδρομές οι οποίες δεν έχουν μεγάλη απόκλιση από τις βέλτιστες. Στην τελευταία ενότητα όπου παρατίθενται τα αποτελέσματα του Αλγορίθμου, υπάρχει πίνακας οπου γίνεται σύγκριση των Αρχικών λύσεων που δημιουργεί ο Παραλλαγμένος και ο συμβατικός αλγόριθμος του Πλησιέστερου Γείτονα.

4.3 Αλγόριθμος Περιορισμένης Αναζήτησης (Tabu Search)

Ο Αλγόριθμος Περιορισμένης Αναζήτησης καλείται να βελτιστοποιήσει την εφικτή λύση που βρέθηκε στο αρχικό στάδιο του προγράμματος. Για να επιτευχθεί ο συγκεκριμένος στόχος ακολουθείται η παρακάτω διαδικασία.

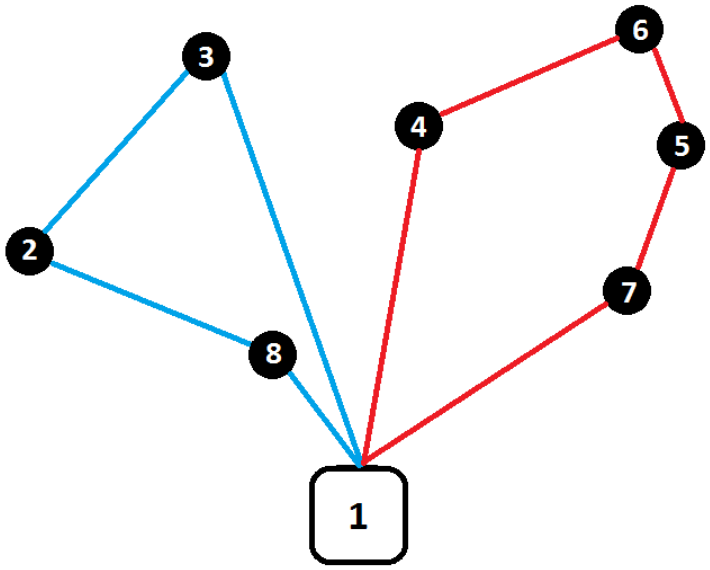
Δημιουργία «Γειτονιάς»

Η πρώτη διαδικασία που εκτελείται στον tabu search είναι η δημιουργία της γειτονιάς της τρέχουσας λύσης. Τρέχουσα λύση ονομάζεται η λύση που έχει προκύψει ύστερα από εναλλαγές στην αλληλουχία των κόμβων της Αρχικής Λύσης. Στην πρώτη επανάληψη του αλγόριθμου η Αρχική Λύση θεωρείται τρέχουσα. Γειτονιά θεωρείται το σύνολο των λύσεων που προκύπτουν από μία εναλλαγή κόμβων στην τρέχουσα λύση. Δηλαδή η «γειτονιά» περιέχει όλες τις δυνατές κινήσεις που μπορεί να επιλέξει ο αλγόριθμος. Ας δούμε όμως την διαδικασία πιο αναλυτικά.

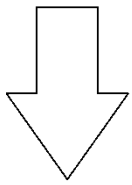
Για να βρούμε μία γειτονική λύση της τρέχουσας, πρέπει όπως είπαμε να εκτελέσουμε κάποια αλλαγή κόμβων. Στη συγκεκριμένη εργασία επιλέχτηκε η αλλαγή αυτή να συμβαίνει μεταξύ κόμβων που ανήκουν σε διαφορετικά δρομολόγια. Δηλαδή ένας κόμβος από κάποια διαδρομή αντικαθίσταται από έναν άλλο κόμβο κάποιας άλλης διαδρομής και το αντίστροφο. Αλλά και μεταξύ γειτονικών διαδοχικών κόμβων (swap). Οι διαδικασίες αυτές αναπαρίστανται σχηματικά στα ακόλουθα παραδείγματα .

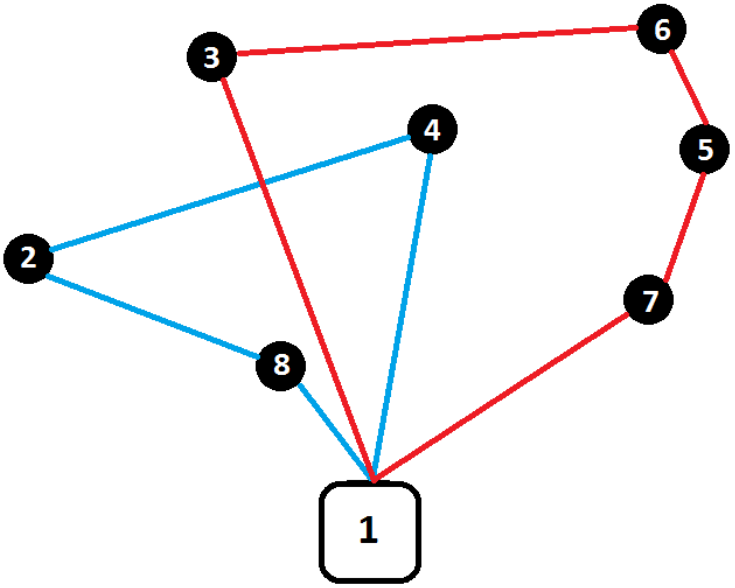
Παράδειγμα 1: Εναλλαγή κόμβων που ανήκουν σε διαφορετικά δρομολόγια

Λύση **πριν** την εναλλαγή	Λύση **μετά** την εναλλαγή
1 8 2 3 1	1 8 2 4 1
1 7 5 6 4 1	1 7 5 6 3 1



Όχημα 1
Όχημα 2





Όχημα 1
Όχημα 2

Παράδειγμα 2: Εναλλαγή διαδοχικών κόμβων που ανήκουν στο ίδιο δρομολόγιο.

Λύση **πριν** την εναλλαγή

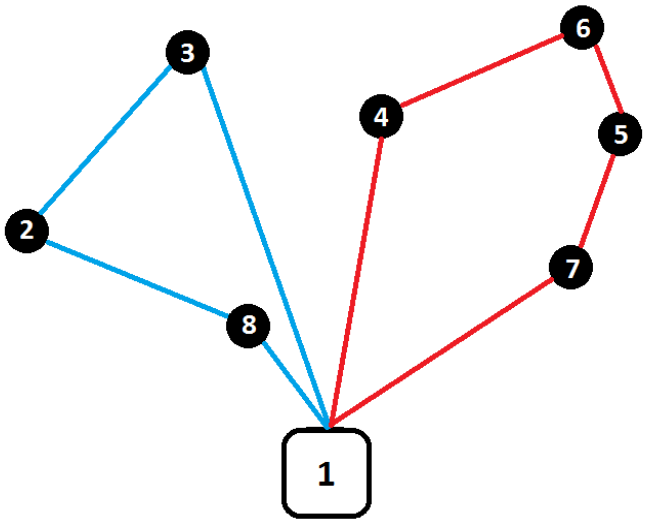
1 8 2 3 1

1 7 5 6 4 1

Λύση **μετά** την εναλλαγή

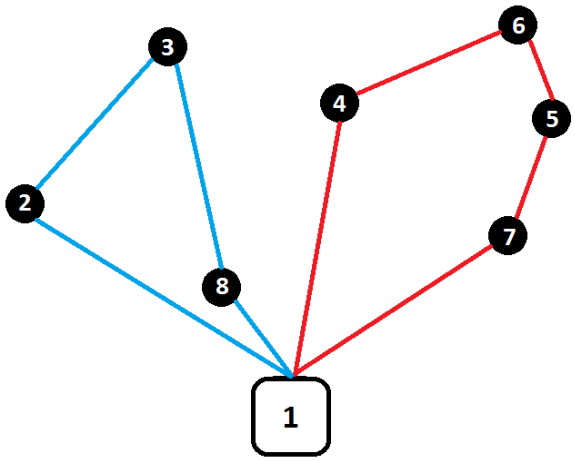
1 8 3 2 1

1 7 5 6 4 1



Όχημα 1
Όχημα 2

Swap



Όχημα 1
Όχημα 2

Πιο συγκεκριμένα, για κάθε κόμβο της τρέχουσας λύσης, βρίσκουμε τους κοντινότερους του, των οποίων ο αριθμός αλλάζει δυναμικά ανάλογα με το μέγεθος του προβλήματος και τις παρούσες συνθήκες και υποχρεωτικά δεν ανήκουν στην ίδια διαδρομή με τον κόμβο. Για παράδειγμα εάν έχουμε συνολικά 100 κόμβους, επιλέγουμε τους 50 κοντινότερους κάθε κόμβου, ενώ εάν έχουμε ένα πρόβλημα 20 κόμβων, ελέγχονται οι 10 κοντινότεροι του καθενός. Ακολούθως εξετάζουμε εάν η εναλλαγή μεταξύ του κόμβου και καθενός από τους κοντινότερους του μπορεί να οδηγήσει σε μία εφικτή γειτονική λύση. Στις περιπτώσεις όπου οι νέες διαδρομές που προκύπτουν παραμένουν εφικτές υπολογίζεται το κόστος της νέας λύσης. Τα κόστη των εφικτών γειτονικών λύσεων αποθηκεύονται σε έναν πίνακα, όπου nbr_cost (i,j) είναι το κόστος της νέας λύσης που προκύπτει από την αλλαγή θέσης του κόμβου i με τον κόμβο j στην τρέχουσα λύση.

Αφού λοιπόν δημιουργηθεί ο πίνακας nbr_cost, ο αλγόριθμος καλείται να επιλέξει ποια από τις εφικτές εναλλαγές θα πραγματοποιηθεί. Επειδή τα στοιχεία του πίνακα nbr_cost που αντιστοιχούν σε κινήσεις που παραβαίνουν τους περιορισμούς έχουν οριστεί ως άπειρο, η μικρότερη τιμή του nbr_cost αντιστοιχεί στην κίνηση που οδηγεί στην καλύτερη, ή τουλάχιστον στην λιγότερο κακή γειτονική λύση. Εάν λοιπόν το στοιχείο nbr_cost(α,b) είναι το ελάχιστο του πίνακα, τότε θεωρούμε ως καλύτερη κίνηση την εναλλαγή του κόμβου α με τον b. Στην περίπτωση που εκτελεστεί αυτή η αλλαγή, το κόστος της νέας λύσης θα είναι ίσο με την τιμή που ήταν αποθηκευμένη στο στοιχείο nbr_cost(α,b).

Αφού εντοπιστεί η βέλτιστη κίνηση, γίνεται έλεγχος για το εάν εμπεριέχεται στον Tabu πίνακα περιορισμένων κινήσεων. Εάν η συγκεκριμένη εναλλαγή είναι περιορισμένη, ελέγχεται ομοίως η επόμενη βέλτιστη κίνηση. Η διαδικασία τερματίζει με την εύρεση της καλύτερης μη περιορισμένης εναλλαγής. Υπάρχει όμως η περίπτωση να πληρείται το κριτήριο απενεργοποίησης περιορισμών. Στην περίπτωση αυτή η πρώτη βέλτιστη εναλλαγή που ελέγχεται, συναντάται στον πίνακα Tabu, όμως οδηγεί σε λύση με λιγότερο συνολικό κόστος διαδρομής από το έως εκείνη τη στιγμή Βέλτιστο. Όταν συμβαίνει κάτι τέτοιο παρακάμπτεται το γεγονός ότι η κίνηση είναι περιορισμένη και επιλέγεται η αντίστοιχη εναλλαγή.

Ακολούθως γίνεται έλεγχος για το εάν η εναλλαγή που επιλέχτηκε βελτιώνει το κόστος της τρέχουσας διαδρομής. Σε περίπτωση που κάτι τέτοιο δεν συμβαίνει επιλέγεται κίνηση με βάση τη Συχνότητα Εμφάνισης. Δηλαδή από τις εφικτές κινήσεις, επιλέγεται εκείνη η οποία έχει εκτελεστεί τις λιγότερες φορές, ανεξάρτητα από το κόστος της λύσης που παράγει.

Με τις παραπάνω ενέργειες επιλέγεται μία συγκεκριμένη κίνηση που είναι πολύ πιθανό να εκτελέσει ο αλγόριθμος. Είναι πιθανό και όχι βέβαιο καθώς έπειτα από την παραπάνω διαδικασία, ελέγχονται οι γειτονικές λύσεις που προκύπτουν από την

εναλλαγή γειτονικών κόμβων της ίδιας διαδρομής. Η διαδικασία παρουσιάζεται γραφικά στη συνέχεια. Η διαφορά με την προηγούμενη μέθοδο εύρεσης γειτονικών λύσεων, εντοπίζεται στο γεγονός ότι στην πρώτη περίπτωση οι κόμβοι που αλλάζουν θέση, ανήκουν υποχρεωτικά σε διαφορετικά δρομολόγια, δηλαδή εξυπηρετούνται από διαφορετικά οχήματα. Ακόμα ο έλεγχος εφικτότητας των εναλλαγών διαφέρει σημαντικά, καθώς στην πρώτη περίπτωση είναι αρκετά πιο πολύπλοκος. Εάν λοιπόν η βέλτιστη γειτονική λύση που προέκυψε από την δεύτερη διαδικασία (swap) παράγεται από μη περιορισμένη εναλλαγή και έχει κόστος μικρότερο του αντίστοιχου της τρέχουσας κίνησης, επιλέγεται και ακολούθως εκτελείται. Σε αντίθετη περίπτωση η κίνηση παραμένει η ίδια που είχε επιλεγεί πριν τον υπολογισμό των SWAP «γειτόνων». Πρέπει να αναφερθεί ότι και σε αυτή την περίπτωση ελέγχεται κριτήριο απενεργοποίησης των περιορισμών. Εάν δηλαδή η βέλτιστη swap κίνηση είναι περιορισμένη και οδηγεί σε λύση καλύτερη της Βέλτιστης, το γεγονός ότι αποτελεί κίνηση tabu δεν έχει πλέον καμία σημασία.

Τέλος ελέγχεται εάν η νέα λύση που προκύπτει από την εναλλαγή που επιλέχτηκε έχει συνολικό κόστος διαδρομής μικρότερο από την Βέλτιστη έως τώρα λύση. Επίσης μία αρκετά σημαντική διαδικασία που εκτελείται πριν το τέλος κάθε επανάληψης είναι η προσπάθεια ελαχιστοποίησης των οχημάτων που χρειάζονται για την εξυπηρέτηση όλων των κόμβων. Κάτι τέτοιο γίνεται εφικτό με την κατάργηση των μικρότερων δρομολογίων και την εξυπηρέτηση των κόμβων που ανήκουν σε αυτά με τη βοήθεια των υπολοίπων οχημάτων. Πάντοτε βέβαια η λύση που προκύπτει θα πρέπει να τηρεί όλους τους περιορισμούς. Το συγκεκριμένο όμως τμήμα του προγράμματος θα αναλυθεί σε ακόλουθες παραγράφους.

Ακολουθεί ο ψευδοκώδικας του αλγόριθμου όπου η λειτουργία του προγράμματος γίνεται πιο κατανοητή.

Αρχικοποίηση Μεταβλητών

Αποθήκευση Δεδομένων του Προβλήματος

Υπολογισμός Αποστάσεων Μεταξύ των Κόμβων

Εφαρμογή Παραλλαγμένου Αλγόριθμου Πλησιέστερου Γείτονα, μέχρι να εξυπηρετηθούν όλοι οι κόμβοι, για την δημιουργία μιας Αρχικής Υποβέλτιστης Λύσης.

Tabu Search

Η Αρχική Λύση θεωρείται ως Προσωρινή Λύση

Για (μέχρι να ολοκληρωθούν οι επαναλήψεις)

Υπολογίζονται οι γειτονικές λύσεις, οι οποίες προκύπτουν εναλλαγές κόμβων που ανήκουν σε διαφορετικά δρομολόγια της προσωρινής λύσης.

Επιλέγεται η γειτονική λύση με το ελάχιστο Συνολικό Κόστος διαδρομής.

Μέχρι (η κίνηση που οδηγεί στην επιλεγμένη γειτονική λύση είναι στη λίστα περιορισμένων διαδρομών και το κόστος της είναι μεγαλύτερο από το βέλτιστο)

Επιλέγεται η γειτονική λύση με το αμέσως μεγαλύτερο κόστος

Τέλος Μέχρι

Αν (το κόστος της επιλεγμένης γειτονικής λύσης είναι μεγαλύτερο από το κόστος της τρέχουσας λύσης)

Επιλέγεται η γειτονική λύση που προκύπτει από την κίνηση που έχει εφαρμοστεί τις λιγότερες φορές στο παρελθόν από τον αλγόριθμο.

Τέλος Αν

Υπολογίζεται η καλύτερη swap γειτονική λύση, που προκύπτει από εναλλαγή γειτονικών κόμβων της ίδιας διαδρομής στην προσωρινή λύση.

Αν (Η swap γειτονική λύση έχει μικρότερο κόστος από την επιλεγμένη γειτονική λύση)

> **Αν** (η εναλλαγή που οδηγεί στην swap γειτονική λύση δεν είναι tabu κίνηση ή εάν είναι tabu και το κόστος της swap γειτονικής λύσης είναι μικρότερο του κόστους της Βέλτιστης Διαδρομής).

Επιλέγεται ως γειτονική λύση η Swap γειτονική λύση.

> **Τέλος Αν**

Τέλος Αν

Η κίνηση που οδηγεί στην επιλεγμένη γειτονική λύσει εισάγεται στον πίνακα περιορισμών κινήσεων.

Ο πίνακας Συχνοτήτων «απομνημονεύει» το γεγονός ότι η συγκεκριμένη κίνηση εφαρμόστηκε ακόμη μια φορά.

Η επιλεγμένη γειτονική λύση τίθεται ως τρέχουσα λύση

Ελέγχεται εάν κάποια από τις τέσσερις μικρότερες διαδρομές της τρέχουσας λύσης θα μπορούσε να αποσυντεθεί και οι κόμβοι της να εξυπηρετηθούν από τα υπόλοιπα οχήματα, χωρίς παράβαση των περιορισμών.

Αν (το κόστος της επιλεγμένης γειτονικής Λύσης είναι μικρότερο του Βέλτιστου)

Αποθηκεύεται ως Βέλτιστη Λύση η Επιλεγμένη Γειτονική Λύση

Αποθηκεύεται ως Βέλτιστο Κόστος το κόστος της Επιλεγμένης Γειτονικής Λύσης

Τέλος Αν

Τέλος Για

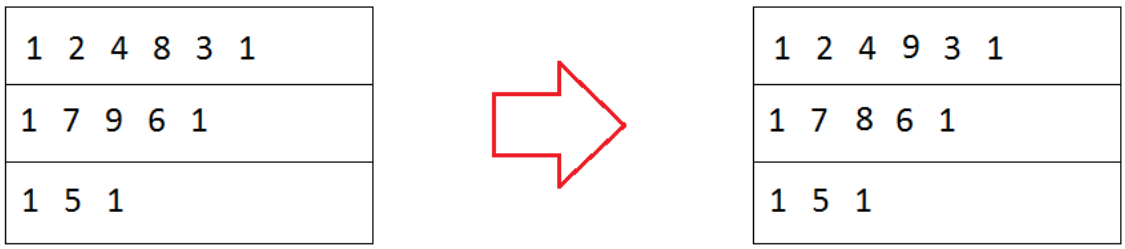
Εκτύπωση Αποτελεσμάτων, Χρονικής Διάρκειας

4.4 Αλγόριθμος Ελαχιστοποίησης Αριθμού Οχημάτων

Κατά την μελέτη του Μεθευρετικού Αλγόριθμου Περιορισμένης Αναζήτησης Tabu Search, γίνεται άμεσα αντιληπτό το γεγονός ότι ο αριθμός των οχημάτων που χρησιμοποιούνται στην αρχική λύση θα είναι ο ίδιος με τον αριθμό των οχημάτων της Βέλτιστης Λύσης που θα κατασκευάσει ο Tabu Search. Αυτό συμβαίνει καθώς ο Αλγόριθμος πραγματοποιεί εναλλαγές στις θέσεις μεταξύ κόμβων, διατηρώντας κάθε φορά τόσο τον αριθμό οχημάτων που χρησιμοποιούνται, όσο και τον αριθμό των κόμβων που εξυπηρετεί κάθε όχημα.

Για παράδειγμα:

Εναλλαγή Κομβων Διαφορετικών Δρομολογίων



Ο κόμβος 8 του πρώτου δρομολογίου αλλάζει θεση με τον κόμβο 9 του δεύτερου δρομολογίου

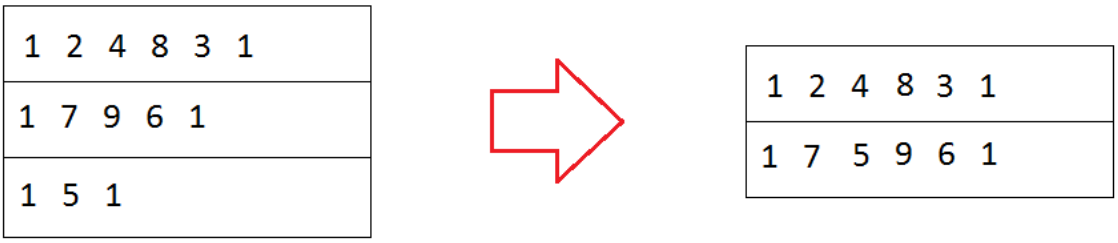
Όπως βλέπουμε στην παραπάνω τρέχουσα λύση συμβαίνει μια εναλλαγή κόμβων που ανήκουν σε διαφορετικά δρομολόγια και συγκεκριμένα του 8 και του 9. Εύκολα παρατηρεί κανείς πως ο αριθμός των οχημάτων που χρησιμοποιούνται για την εξυπηρέτηση και των οκτώ κόμβων παραμένει ίσος με 3 ενώ ο αριθμός των κόμβων που συντάσσονται σε κάθε δρομολόγιο παραμένει επίσης σταθερός. Ενώ λοιπόν στην αρχική λύση χρησιμοποιείται ένας πολύ μεγάλος αριθμός οχημάτων, ο οποίος απέχει πολύ από τον ελάχιστο που θα μπορούσαμε να έχουμε, τότε και η βέλτιστη λύση που θα υπολογιστεί θα είναι καταδικασμένη να χρησιμοποιεί και εκείνη έναν πολύ μεγάλο στόλο οχημάτων. Ενδεικτικά σε ορισμένα προβλήματα η Αρχική Λύση που κατασκευάζονταν από τον Αλγόριθμο του Πλησιέστερου Γείτονα χρησιμοποιούσε μέχρι και τριπλάσιο αριθμό οχημάτων από τον ελάχιστο δυνατό. Όπως είναι λογικό κάτι τέτοιο καθιστά αδύνατη την όποια σημαντική βελτιστοποίηση θέλουμε να επιτύχουμε.

Ένα πολύ σημαντικό τμήμα του αλγόριθμου που κατασκευάστηκε για τις ανάγκες της συγκεκριμένης εργασίας είναι εκείνο που ελαχιστοποιεί τον αριθμό οχημάτων που χρησιμοποιούνται στην τρέχουσα λύση. Είναι ίσως το πιο απαιτητικό στάδιο από άποψη σχεδιασμού και προγραμματισμού, όμως χωρίς αυτό τα βέλτιστα αποτελέσματα του αλγόριθμου δεν θα ήταν άξια λόγου. Στην συνέχεια λοιπόν θα παρουσιαστούν οι βασικές ενέργειες που εκτελούνται για την επίτευξη μείωσης των δρομολογίων.

Οι κόμβοι του δρομολογίου που θα καταργηθεί, θα εξυπηρετηθούν από τα υπόλοιπα οχήματα και θα καταλάβουν ενδιάμεσες θέσεις, μεταξύ γειτονικών διαδοχικών κόμβων άλλων δρομολογίων.

Για παράδειγμα:

Μείωση απαιτούμενων οχημάτων



Στην πρώτη λύση χρησιμοποιούνται 3 οχήματα ενώ στη δεύτερη 2.
Ο κόμβος 5 εξυπηρετείτε πλέον απο το όχημα 2, με αποτέλεσμα το τριτο δρομολόγιο να καθίσταται αχρείαστο και να καταργείται.

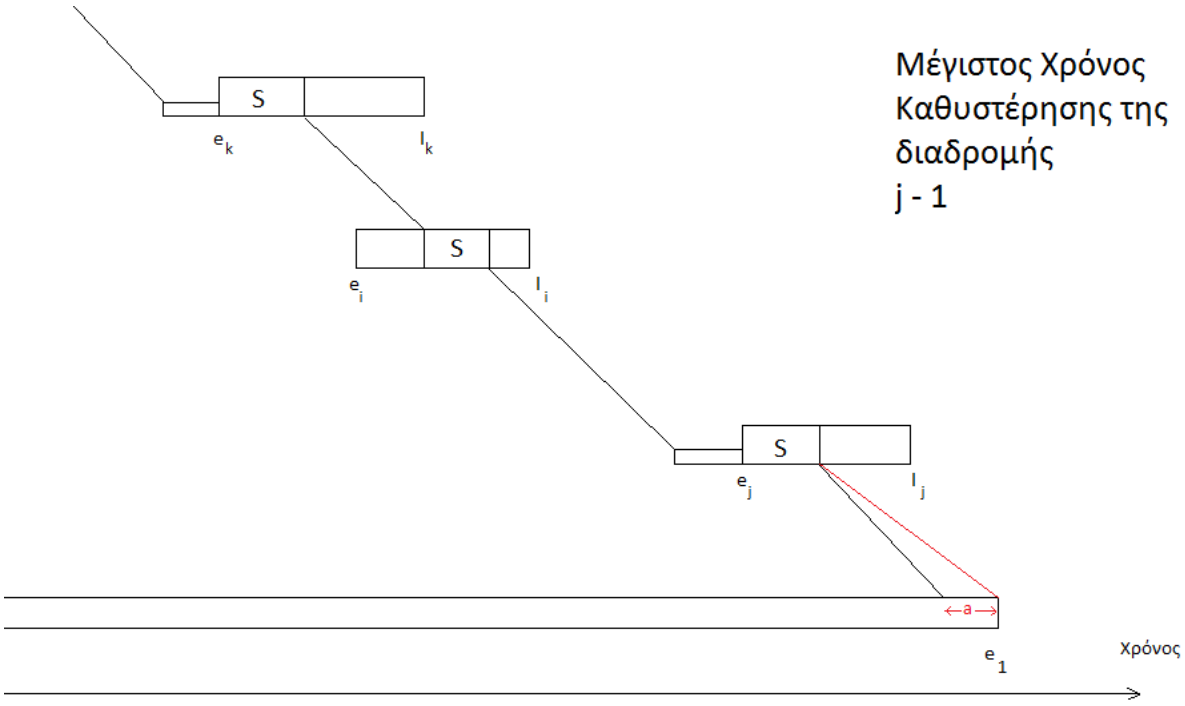
Παραπάνω βλέπουμε ότι το τελευταίο δρομολόγιο καταργείται και ο κόμβος 5 εξυπηρετείται πλέον από το σχήμα 2 σε θέση που βρίσκεται μεταξύ των κόμβων 7 και 9. Στη νέα λύση δεν χρησιμοποιούνται τρία οχήματα αλλά δύο. Για να γίνει όμως μία τέτοιου είδους αλλαγή στην τρέχουσα λύση θα πρέπει να μην καταπατώνται οι περιορισμοί ζήτησης και χρόνου.

Αρχικά λοιπόν υπολογίζεται πόσο χρόνο θα μπορούσε να αργήσει κάθε διαδρομή, της τρέχουσας λύσης καθώς οι κόμβοι του δρομολογίου που καταργείται τοποθετούνται ενδιάμεσα στους υπόλοιπους. Το χρονικό διάστημα μέγιστης καθυστέρησης της διαδρομής εξαρτάται άμεσα από τα χρονικά παράθυρα των κόμβων που εξυπηρετούνται μετά από την συγκεκριμένη μετάβαση, στο ίδιο δρομολόγιο.

Εάν έχουμε για παράδειγμα το δρομολόγιο 1 2 4 8 3 1, είχαμε για κάποιο λόγο καθυστέρηση στη διαδρομή 4-8, θα επηρεάζονταν οι χρόνοι εξυπηρέτησης και αναμονής των κόμβων 8 και 3. Ενώ, θα υπήρχε ενδεχόμενο να μην ήταν δυνατόν να εξυπηρετηθούν καθώς το όχημα φτάνει καθυστερημένα στους κόμβους αυτούς και

ενδέχεται ο νέος χρόνος άφιξης να είναι μεγαλύτερος του βραδύτερου χρόνου του εκάστοτε χρονικού παραθύρου.

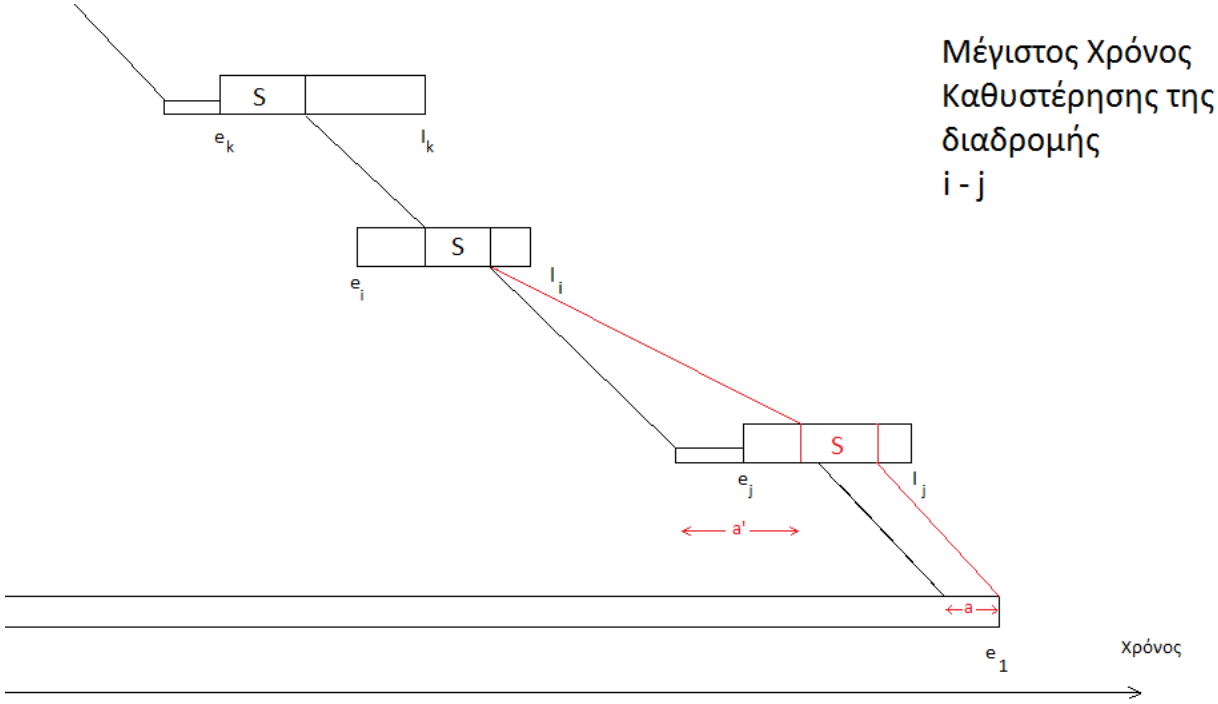
Ο μέγιστος χρόνος καθυστέρησης θα είναι ο μεγαλύτερος δυνατός για τον οποίο τηρούνται οι χρονικοί περιορισμοί στους ακόλουθους κόμβους. Για να υπολογίσουμε λοιπόν κάτι τέτοιο θα πρέπει να αρχίσουμε από την τελευταία διαδρομή κάθε δρομολογίου. Ακολούθως γίνεται αναπαράσταση της διαδικασίας για καλύτερη κατανόηση. Στο παρακάτω σχήμα απεικονίζονται τα χρονικά παράθυρα των τελευταίων κόμβων ενός δρομολογίου. Αρχικά λοιπόν υπολογίζουμε τη μέγιστη καθυστέρηση της τελευταίας διαδρομής j-1. Με κόκκινο χρώμα αναπαρίσταται η περίπτωση όπου το όχημα καλύπτει οριακά τους χρονικούς περιορισμούς λόγω αργοπορίας. Βλέπουμε λοιπόν ότι η τελευταία διαδρομή μπορεί να καθυστερήσει κατά α ώστε το όχημα να προλάβει οριακά να επιστρέψει στην αποθήκη (κόμβος 1) εντός του χρονικού παραθύρου, πριν τον βραδύτερο χρόνο επιστροφής.



Εν συνεχεία υπολογίζεται η μέγιστη καθυστέρηση της προτελευταίας διαδρομής i-j. Όπως βλέπουμε και στο ακόλουθο σχήμα θα είναι ίση με χρόνο αναμονής στον κόμβο j + (ελάχιστο μεταξύ του α και του (l(j) – χρονική στιγμή έναρξης της εξυπηρέτησης του κόμβου j).

Στη συγκεκριμένη περίπτωση θα έχουμε:

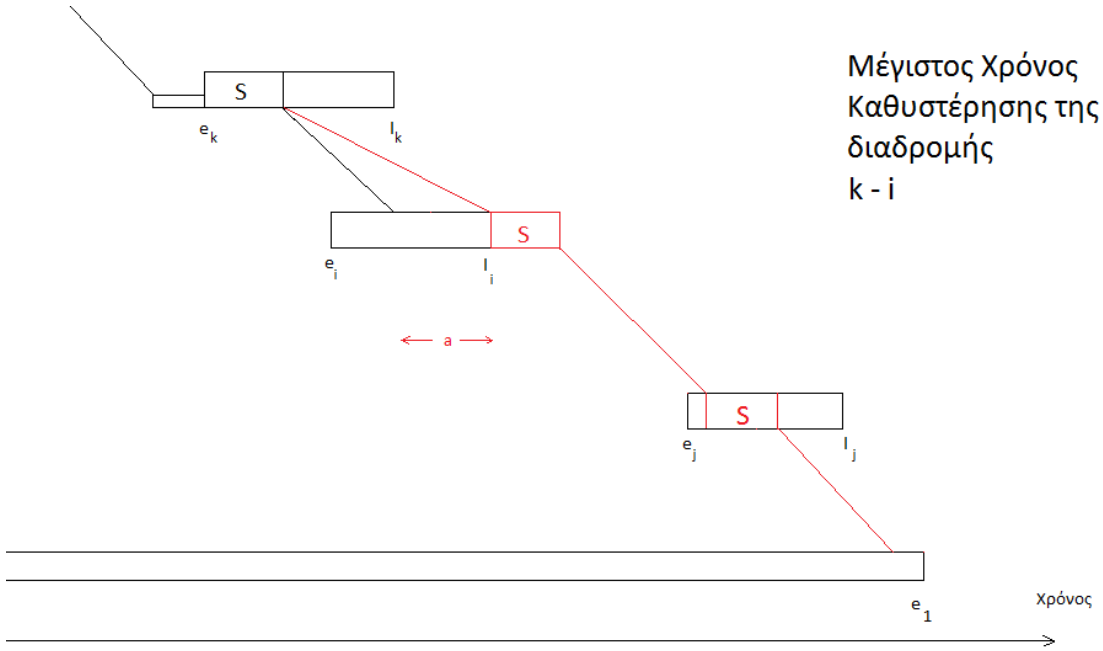
μέγιστη καθυστέρηση διαδρομής (l - j) = (χρόνος αναμονής στο j) + (α)



Ακολούθως θέτουμε α΄= μέγιστη καθυστέρηση (i-j) και προχωρούμε στην διαδρομή κ-i όπου θα έχουμε:

> μέγιστη καθυστέρηση διαδρομής (κ-i)=l(i) – χρονική στιγμή περάτωσης της εξυπηρέτησης του κόμβου i.

Η συγκεκριμένη περίπτωση φαίνεται γραφικά παρακάτω:



Στη γενική περίπτωση έχουμε:

max_delay_time(node-I, node-j)=node_waiting_time(node j)+

+min(α , l(node j)-starting_time(node j)),

Όπου max_delay_time ο πίνακας στον οποίο αποθηκεύεται μέγιστη δυνατή καθυστέρηση μεταξύ διαδοχικών κόμβων node-i, node-j. Για κόμβους που δεν είναι διαδοχικοί ως προς την εξυπηρέτησή τους το αντίστοιχο στοιχείο του πίνακα max-delay-time τίθεται ίσο με 0.

Όλη η παραπάνω διαδικασία συνεχίζεται για όλες τις διαδρομές όλων των δρομολογίων.

Υπολογίσαμε λοιπόν πόσο θα μπορούσε να καθυστερήσει κάθε διαδρομή ώστε να μπορέσουμε εν συνεχεία να ελέγξουμε εάν υπάρχει δυνατότητα εξυπηρέτησης ενός κόμβου που ανήκει σε ένα δρομολόγιο που καταργείται στο χρονικό διάστημα μέγιστης καθυστέρησης κάποιας διαδρομής. Στο ακόλουθο σχήμα εμφανίζεται ένας νέος κόμβος μεταξύ των i και j.

Ήρθε λοιπόν η στιγμή όπου θα γίνει προσπάθεια εξάλειψης ενός δρομολογίου. Αρχικά επιλέγουμε το μικρότερο σε εξυπηρετούμενους κόμβους δρομολόγιο.

Ακολούθως ελέγχουμε εάν κάθε κόμβος που εξυπηρετείται από αυτό το μονοπάτι μπορεί να εξυπηρετηθεί από κάποιο άλλο όχημα διατηρώντας την τρέχουσα λύση

εφικτή. Εφόσον όλοι οι κόμβοι μπορούν να εξυπηρετηθούν από κάποιο άλλο όχημα το δρομολόγιο καταργείται και αποθηκεύεται η νέα λύση σαν τρέχουσα.

Ακόμη ενημερώνονται και τα υπόλοιπα στοιχεία όπως χρόνοι έναρξης εξυπηρέτησης, χρόνοι αναμονής και φορτία. Εάν κάποιος κόμβος δεν μπορεί να εξυπηρετηθεί ελέγχουμε με τον ίδιο τρόπο το αμέσως μεγαλύτερο δρομολόγιο. Η διαδικασία αυτή επαναλαμβάνεται στο συγκεκριμένο αλγόριθμο τέσσερις φορές.

5. Αποτελέσματα Αλγόριθμου

Με την χρήση όλων των διαδικασιών και των αλγορίθμων που περιγράφηκαν, επιτεύχθηκε η κατασκευή ενός αλγορίθμου για την επίλυση του Προβλήματος Δρομολόγησης Οχημάτων με Χρονικά Παράθυρα αλλά και όλων των απλούστερων προβλημάτων VRP. Δηλαδή, θέτοντας τα κατάλληλα αρχικά δεδομένα και περιορισμούς καθίσταται εφικτή η επίλυση παρεμφερών προβλημάτων. Εάν για παράδειγμα κληθούμε να αντιμετωπίσουμε το πρόβλημα OVRP, όπου το όχημα δεν απαιτείται να επιστρέψει στη βάση, οι μόνες αλλαγές που θα πρέπει να συμβούν στα αρχικά δεδομένα είναι να τεθούν τα χρονικά παράθυρα των πελατών (εκτός της απόθηκης) ίσα με [0,inf] και οι αποστάσεις d(node,1) ίσες με μηδέν για κάθε κόμβο(εκτός του 1).

Για να ελεγχθεί όμως η επιτυχία του προγράμματος που δημιουργήθηκε εν τέλει, θα πρέπει να διερευνηθεί κατά πόσο η λύση που παράγει είναι άξια λόγου. Επιλύθηκαν λοιπόν ορισμένα προβλήματα των οποίων οι βέλτιστες λύσεις ήταν γνωστές εκ των προτέρων. Έπειτα υπολογίστηκε η απόκλιση του κόστους της λύσης που παρήγαγε ο αλγόριθμος από την αντίστοιχη παγκοσμίως βέλτιστη λύση του κάθε παραδείγματος.

Όλα τα πρότυπα προβλήματα τα οποία επιλύθηκαν με τον Αλγόριθμο που κατασκευάστηκε στην συγκεκριμένη εργασία έχουν ορισμένα κοινά στοιχεία στα δεδομένα τους. Ο αριθμός των προς εξυπηρέτηση πελατών είναι ίσος με 100, ο αριθμός των διαθέσιμων οχημάτων είναι 25 και έχουν χωρητικότητα ιση με 200 μονάδες προϊόντος. Ενώ οι συντεταγμένες, το χρονικό παράθυρο, η ζήτηση και ο χρόνος εξυπηρέτησης κάθε πελάτη αλλάζουν σε κάθε πρόβλημα. Επίσης τα προβλήματα διακρίνονται σε τρείς κατηγορίες ανάλογα με τον τρόπο που διατάσσονται οι πελάτες

στον χώρο. Δηλαδή έχουμε την κατηγορία προβλημάτων Centered, όπου οι κόμβοι σχηματίζουν ορισμένες περιοχές μεγαλύτερης συγκέντρωσης γύρω από την αποθήκη. Στα προβλήματα τύπου Random η διάταξη των κόμβων στο χώρο είναι τυχαία, ενώ στα προβλήματα τύπου Random Centered συναντούμε μια μίξη των δύο προηγούμενων τύπων. Το είδος κάθε προβλήματος αναγνωρίζεται από τα αρχικά γράμματα που συναντούμε στην ονομασία του. Ακολούθως παρατίθενται τόσο οι λύσεις που παράχθηκαν σε κάθε, όσο και οι αποκλείσεις από τις αντίστοιχες βέλτιστες.

5.1 Πρόβλημα C101

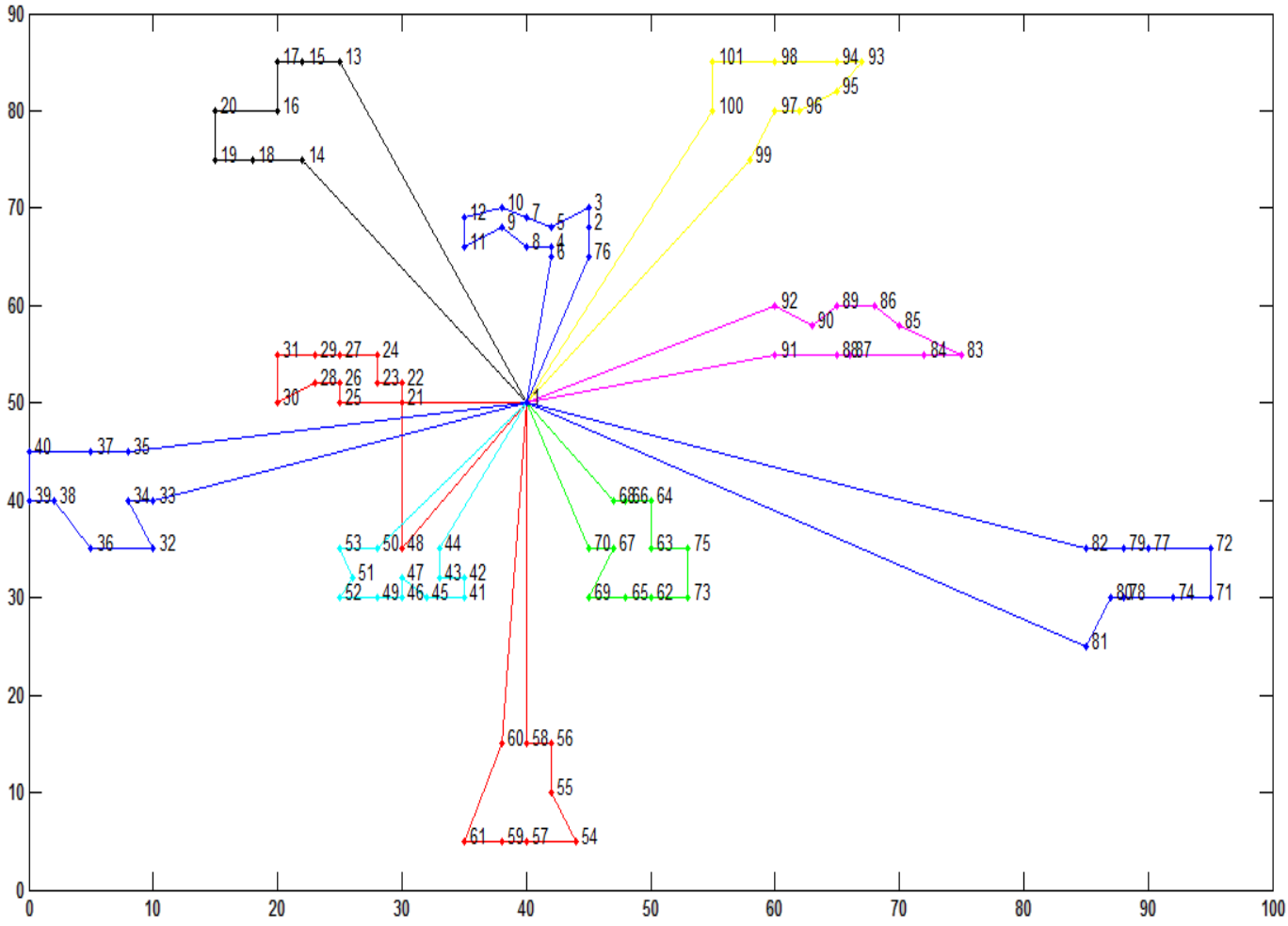
Στο συγκεκριμένο πρόβλημα, έχουμε n=100 κόμβους προς εξυπηρέτηση. Για τον σκοπό αυτό, διαθέτουμε στόλο 25 οχημάτων ενώ μας δίδεται τόσο η ζήτηση, όσο και το χρονικό παράθυρο καθενός από του κόμβους. Ακόμη, τα οχήματα θεωρούνται όμοια, με ίδια χωρητικότητα, ίση με 200, δηλαδή capacity=200.

Στο πρόβλημα C101 λοιπόν έγινε χρήση του παραλλαγμένου Αλγόριθμου του Πλησιέστερου γείτονα που οδήγησε στην κατασκευή μίας αρχικής λύσης, η οποία δεν βελτιώθηκε αργότερα από τον Αλγόριθμο Tabu Search. Κάτι τέτοιο συνέβη μόνο σε δύο παραδείγματα από όλα όσα επιλύθηκαν για τις ανάγκες της συγκεκριμένης διπλωματικής εργασίας. Στη γενική περίπτωση, η αρχική λύση απέχει αρκετά τόσο σε κόστος, όσο και σε αριθμό χρησιμοποιούμενων οχημάτων από την τελική λύση που κατασκευάζει ο Tabu Search με την βοήθεια του αλγορίθμου Less Vehicle.

Επομένως, στο συγκεκριμένο παράδειγμα έχουμε την ίδια αρχική και τελική λύση, η οποία παρουσιάζεται στον ακόλουθο πίνακα.

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

Το συνολικό κόστος διαδρομής της παραπάνω λύσης είναι Cost=852.948201 ενώ η παγκοσμίως βέλτιστη λύση του συγκεκριμένου παραδείγματος έχει κόστος Best Solution Cost=828.94. Δηλαδή έχουμε μία απόκλιση 2,9%. Ακόμη, χρησιμοποιούνται 10 οχήματα ακριβώς όπως και στη βέλτιστη λύση ,Ακολουθεί η γραφική αναπαράσταση της λύσης.



5.2 Πρόβλημα C102

Στο πρόβλημα C102,όπως και στα προηγούμενα, έχουμε n=100, Capacity=200, maximum vehicle number=25. Η λύση που προκύπτει έχει Κόστος= 973.24 με χρήση 10 οχημάτων, ενώ το βέλτιστο είναι 828.94 με 10 οχήματα. Δηλαδή η απόκλιση από το βέλτιστο, σε αυτή την περίπτωση θα είναι 17.4%. Ακολουθεί η γραφική αναπαράσταση της λύσης

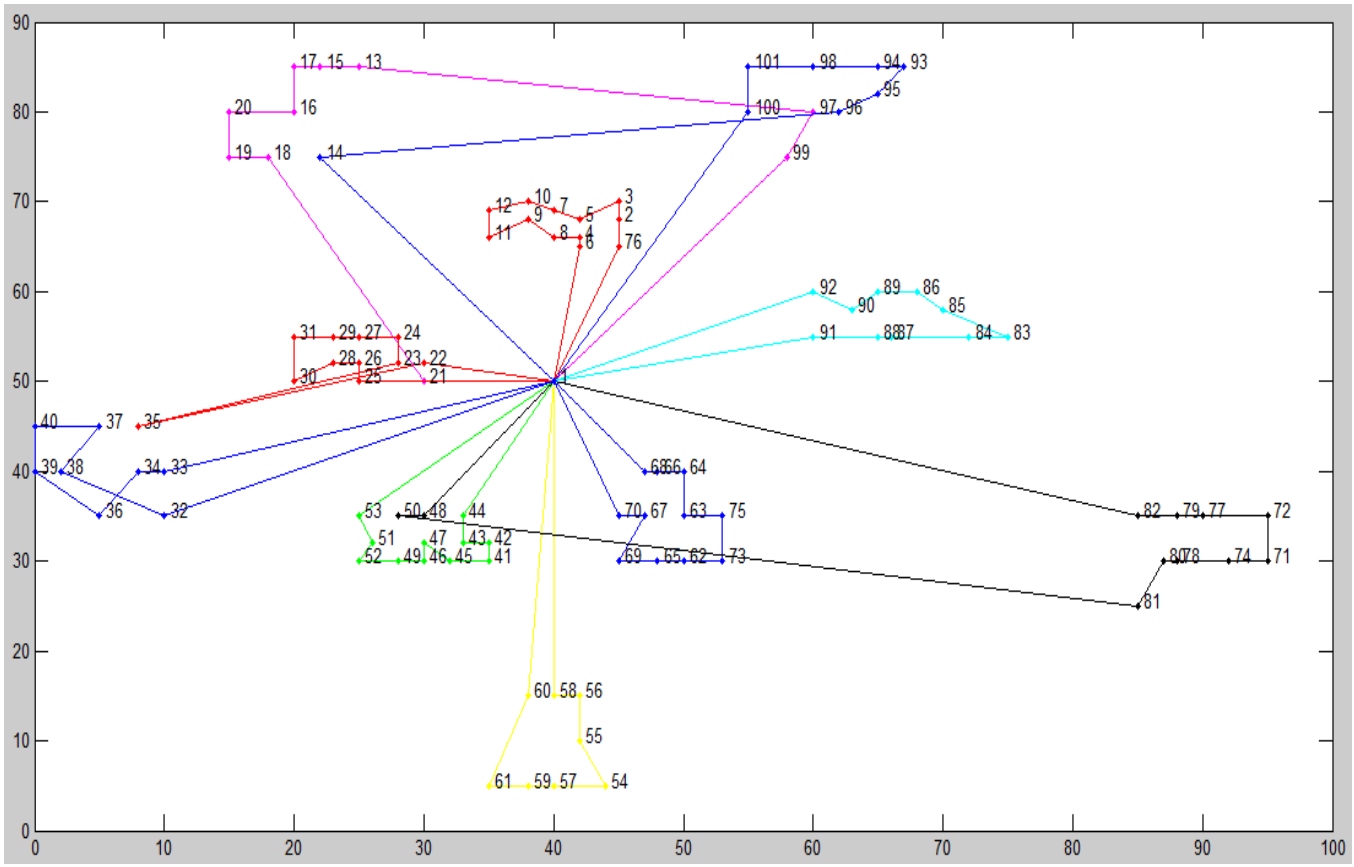
0

1	6	4	8	9	11	12	10	7	5	3	2	76	1
1	68	66	64	63	75	73	62	65	69	67	70	1	
1	44	43	42	41	45	47	46	49	52	51	53	1	0
1	91	88	87	84	83	85	86	89	90	92	1		
1	21	18	19	20	16	17	15	13	97	99	1	0	0
1	58	56	55	54	57	59	61	60	1				
1	82	79	77	72	71	74	78	80	81	50	48	1	0
1	25	26	28	30	31	29	27	24	23	35	22	1	
1	14	96	95	93	94	98	101	100	1	0	0	0	0
1	33	34	36	39	40	37	38	32	1				

Κόστος Λύσης = 973.24

Βέλτιστο Κόστος = 828.94

Απόκλιση από το Βέλτιστο = 0.174


0 10 20 30 40 50 60 70 80 90 100
0 10 20 30 40 50 60 70 80 90
17 15 13
20 16
19 18
14
101 98 94 93
95
100 97 96
99
12 10 7 5 3
11 9 8 6 4 2 76
92 89 86
90 85
91 88 87 84 83
31 29 27 24
28 26 23 22
30 25 21
40 37 35
39 38 34 33
36 32
53 50 48 44
51 47 43 42
52 49 46 45 41
68 66 64
70 67 63 75
69 65 62 73
82 79 77 72
80 78 74 71
81
60 58 56
55
61 59 57 54

5.3 Πρόβλημα C103

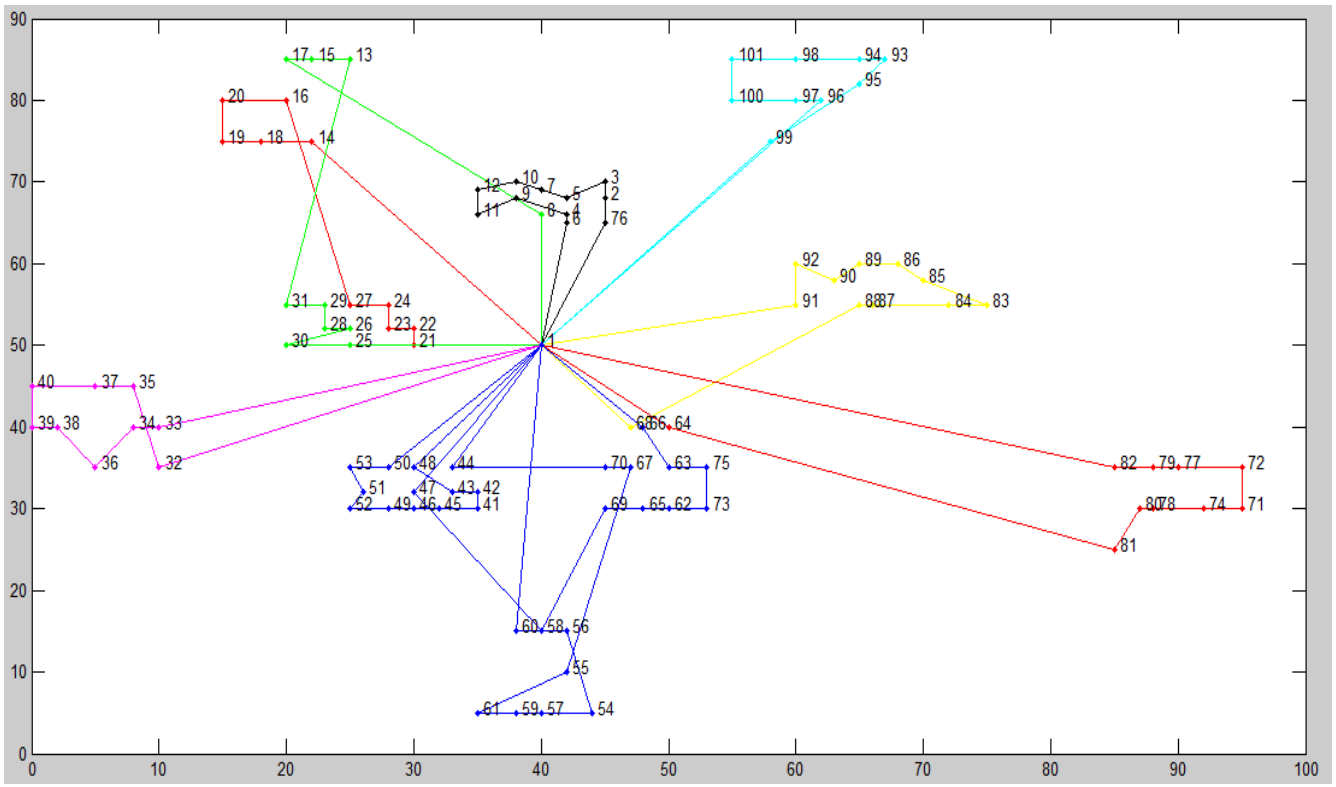
Στο πρόβλημα C103,όπως και στα προηγούμενα, έχουμε n=100, Capacity=200, maximum vehicle number=25. Η λύση που προκύπτει έχει Κόστος=968.97 με χρήση 10 οχημάτων, ενώ το βέλτιστο είναι 828.06 με 10 οχήματα. Δηλαδή η απόκλιση από το βέλτιστο, σε αυτή την περίπτωση θα είναι 17.0%. Ακολουθεί η γραφική αναπαράσταση της λύσης

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

Κόστος Λύσης = 968.97

Βέλτιστο Κόστος = 828.06

Απόκλιση από το Βέλτιστο = 0.17


0 10 20 30 40 50 60 70 80 90 100
0 10 20 30 40 50 60 70 80 90
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101

5.4 Πρόβλημα C104

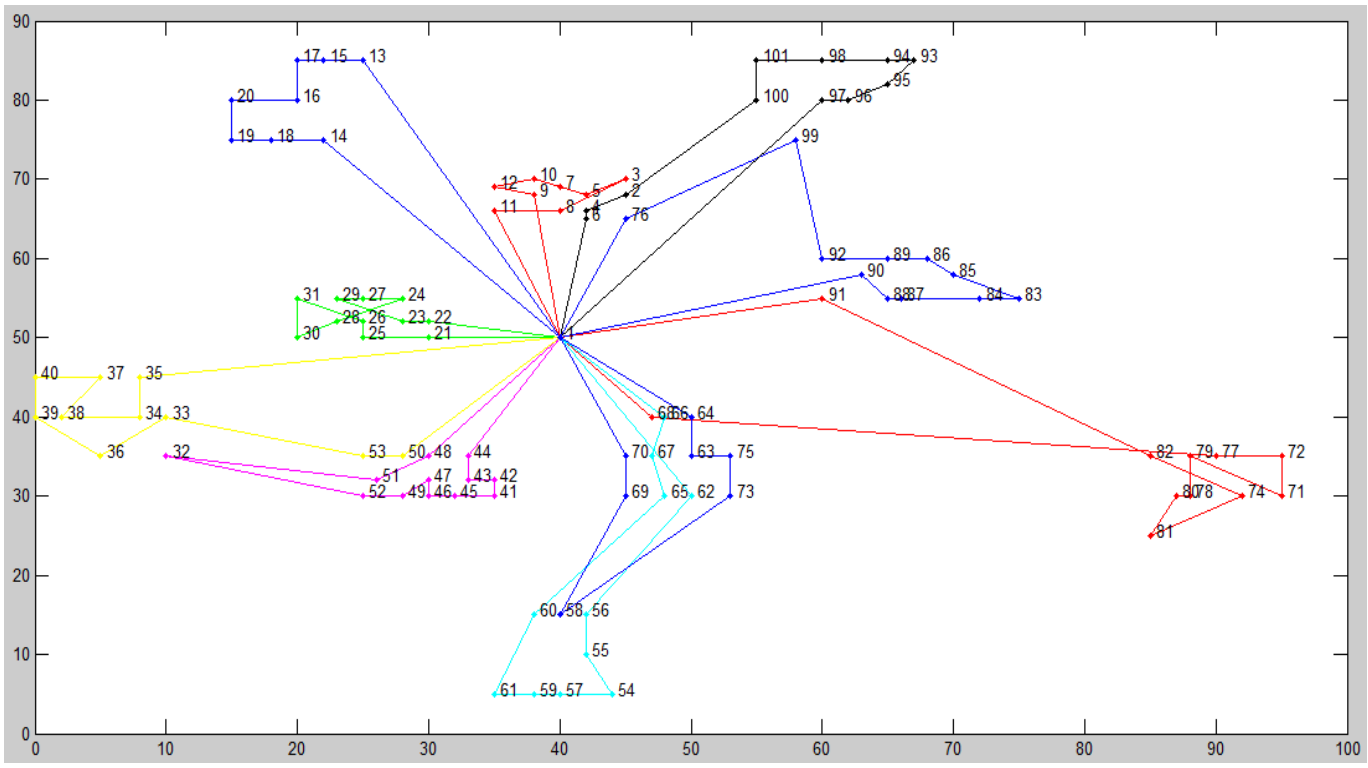
Στο πρόβλημα C104,όπως και στα προηγούμενα, έχουμε n=100, Capacity=200, maximum vehicle number=25. Η λύση που προκύπτει έχει Κόστος=968.48 με χρήση 10 οχημάτων, ενώ το βέλτιστο είναι 824.78 με 10 οχήματα. Δηλαδή η απόκλιση από το βέλτιστο, σε αυτή την περίπτωση θα είναι 16.8%. Ακολουθεί η γραφική αναπαράσταση της λύσης

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

Κόστος Λύσης = 968.48

Βέλτιστο Κόστος = 824.78

Απόκλιση από το Βέλτιστο = 0.168

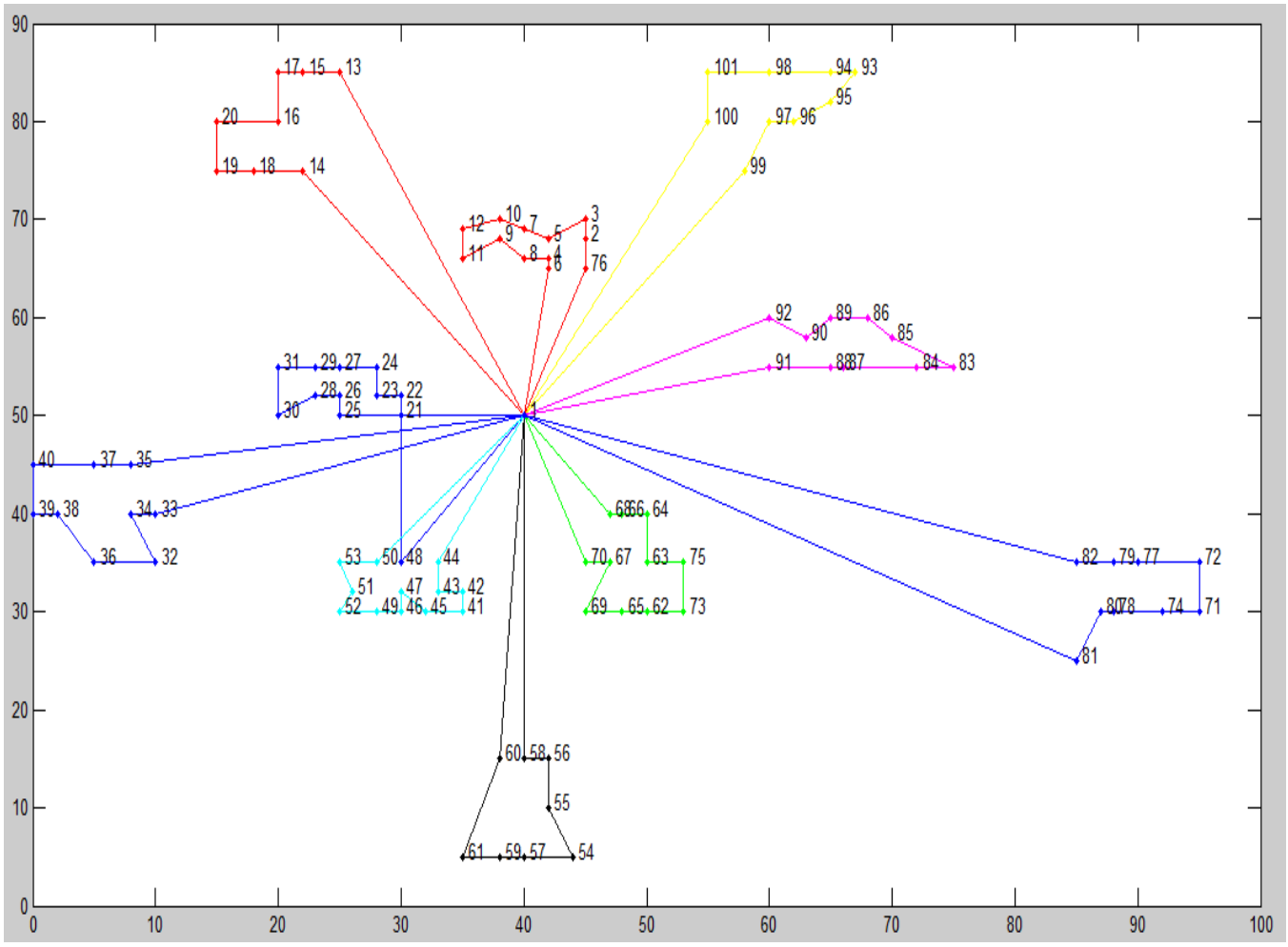

0 10 20 30 40 50 60 70 80 90 100
0 10 20 30 40 50 60 70 80 90
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101

5.5 Πρόβλημα C105

Σύμφωνα με τα δεδομένα του προβλήματος C105, έχουμε n=100 κόμβους προς εξυπηρέτηση, 25 οχήματα και χωρητικότητα ίση με 200. Στο C105 έχουμε παρόμοια περίπτωση με το πρόβλημα C101. Δηλαδή ο παραλλαγμένος αλγόριθμος πλησιέστερου γείτονα οδηγεί σε μια αρχική λύση η οποία βρίσκεται πολύ κοντά στο βέλτιστο και ο αλγόριθμος Tabu Search δεν μπορεί να την βελτιώσει. Βέβαια ο αριθμός των επαναλήψεων που χρησιμοποιήθηκε στα συγκεκριμένα δύο παραδείγματα ήταν αρκετά μικρός, ίσως και ανεπαρκής. Ενδέχεται, εάν είχαμε χρησιμοποιήσει περισσότερες επαναλήψεις, η λύση να βελτιωνόταν ακόμη περισσότερο. Ακολούθως παρουσιάζονται οι διαδρομές που κατασκευάστηκαν από τον αλγόριθμο κατά την επίλυση του προβλήματος C105.

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

Το συνολικό κόστος διαδρομής της παραπάνω λύσης είναι Cost=852.948201 ενώ η παγκοσμίως βέλτιστη λύση του συγκεκριμένου παραδείγματος έχει κόστος Best Solution Cost=828.94. Δηλαδή έχουμε μία απόκλιση 2.9%. Επίσης χρησιμοποιούνται 10 οχήματα, ακριβώς όπως και στη βέλτιστη λύση. Ακολουθεί η γραφική αναπαράσταση της λύσης.


0 10 20 30 40 50 60 70 80 90 100
0 10 20 30 40 50 60 70 80 90
17 15 13
20 16
19 18 14
12 10 7 5 3
9 2
11 8 4 6 76
101 98 94 93
95
100 97 96
99
92 89 86
90 85
91 88 87 84 83
31 29 27 24
28 26 23 22
30 25 21
40 37 35
39 38 34 33
36 32
53 50 48 44
51 47 43 42
52 49 46 45 41
68 66 64
70 67 63 75
69 65 62 73
82 79 77 72
81 78 74 71
60 58 56
55
61 59 57 54

5.6 Πρόβλημα C106

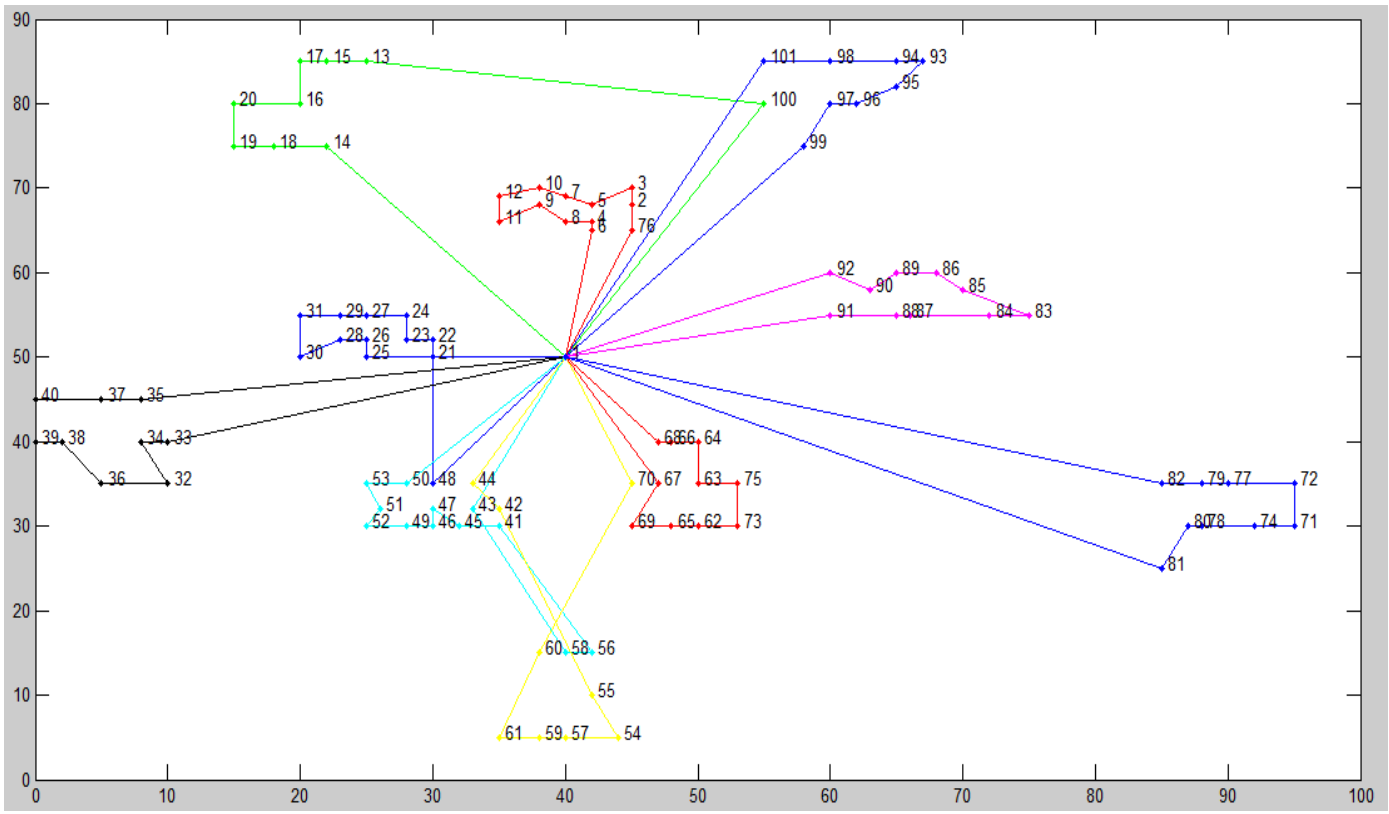
Στο πρόβλημα C106,όπως και στα προηγούμενα, έχουμε n=100, Capacity=200, maximum vehicle number=25. Η λύση που προκύπτει έχει Κόστος=912.99 με χρήση 10 οχημάτων, ενώ το βέλτιστο είναι 828.94 με 10 οχήματα. Δηλαδή η απόκλιση από το βέλτιστο, σε αυτή την περίπτωση θα είναι 10.1%. Ακολουθεί η γραφική αναπαράσταση της λύσης

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

Κόστος Λύσης = 909.25

Βέλτιστο Κόστος = 828.94

Απόκλιση από το Βέλτιστο = 0.097


0 10 20 30 40 50 60 70 80 90 100
0 10 20 30 40 50 60 70 80 90
17 15 13
20 16
19 18 14
101 98 94 93
95
100 97 96
99
12 10 7 3
9 5 2
11 8 4 6 76
92 89 86
90 85
91 88 87 84 83
31 29 27 24
28 26 23 22
30 25 21
40 37 35
39 38 34 33
36 32
53 50 48 44
51 47 43 42
52 49 46 45 41
70 67 63 75
69 65 62 73
82 79 77 72
81 74 71
60 58 56
55
61 59 57 54

5.7 Πρόβλημα C107

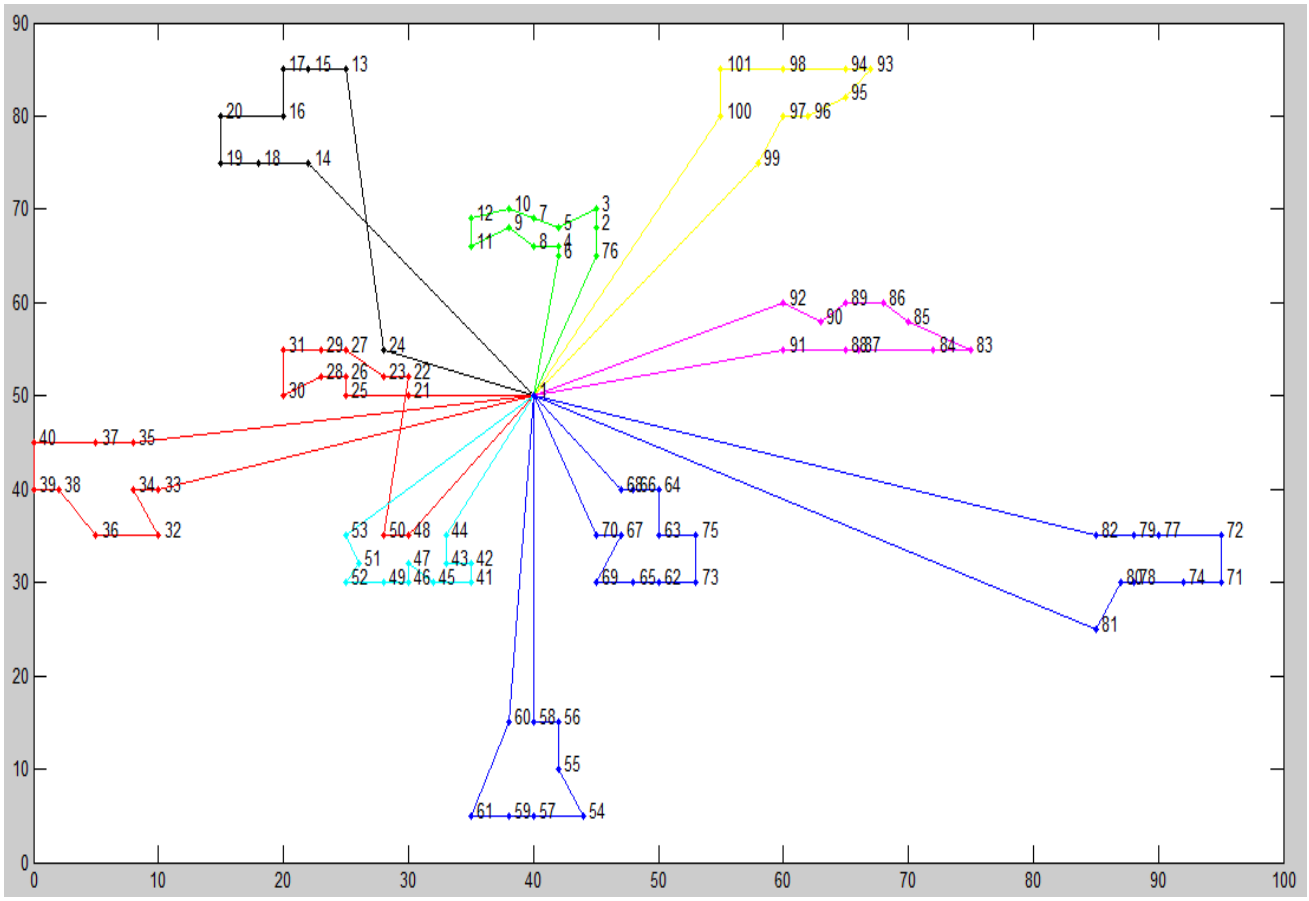
Στο πρόβλημα C107,όπως και στα προηγούμενα, έχουμε n=100, Capacity=200, maximum vehicle number=25, ενώ η ζήτηση, το χρονικό παράθυρο καθώς και η γεωγραφική θέση κάθε κόμβου είναι διαφορετική. Η αρχική λύση που δημιουργήθηκε είχε κόστος 1142.4, ενώ με την διαδικασία του αλγόριθμου βελτιστοποίησης η τελική λύση έχει κόστος 857.38. Η αντίστοιχη βέλτιστη είναι 828.94 δηλαδή έχουμε απόκλιση 3.4%. Ενώ χρησιμοποιούνται 10 οχήματα ακριβώς όπως και στη βέλτιστη λύση. Ακολουθεί η γραφική αναπαράσταση της λύσης.

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

Κόστος Λύσης = 857.38

Βέλτιστο Κόστος = 828.94

Απόκλιση από το Βέλτιστο = 0.034


0 10 20 30 40 50 60 70 80 90 100
0 10 20 30 40 50 60 70 80 90
17 15 13
20 16
19 18 14
101 98 94 93
95
100 97 96
99
12 10 7 3
9 5 2
11 8 4 76
6
92 89 86
90 85
91 88 87 84 83
31 29 27 24
28 26 23 22
30 25 21
40 37 35
39 38 34 33
36 32
53 50 48 44
51 47 43 42
52 49 46 45 41
70 67 63 75
69 65 62 73
82 79 77 72
80 78 74 71
81
60 58 56
55
61 59 57 54

5.8 Πρόβλημα C108

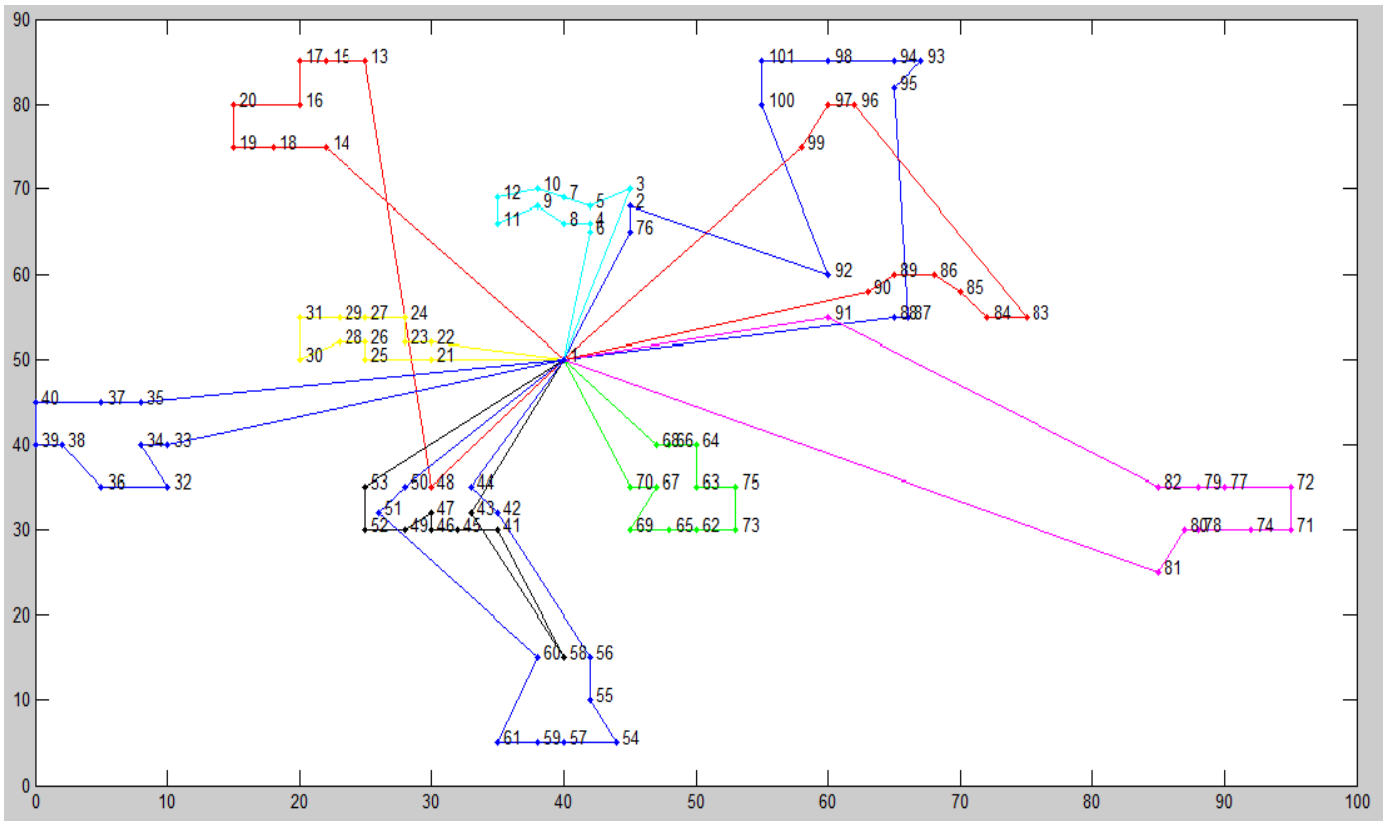
Στο πρόβλημα C108,όπως και στα προηγούμενα, έχουμε n=100, Capacity=200, maximum vehicle number=25. Η λύση που προκύπτει έχει Κόστος=966.50 με χρήση 10 οχημάτων, ενώ το βέλτιστο είναι 828.94 με 10 οχήματα. Δηλαδή η απόκλιση από το βέλτιστο, σε αυτή την περίπτωση θα είναι 16.6%. Ακολουθεί η γραφική αναπαράσταση της λύσης

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

Κόστος Λύσης = 966.50

Βέλτιστο Κόστος = 828.94

Απόκλιση από το Βέλτιστο = 0.166


0 10 20 30 40 50 60 70 80 90 100
0 10 20 30 40 50 60 70 80 90
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101

5.9 Πρόβλημα C109

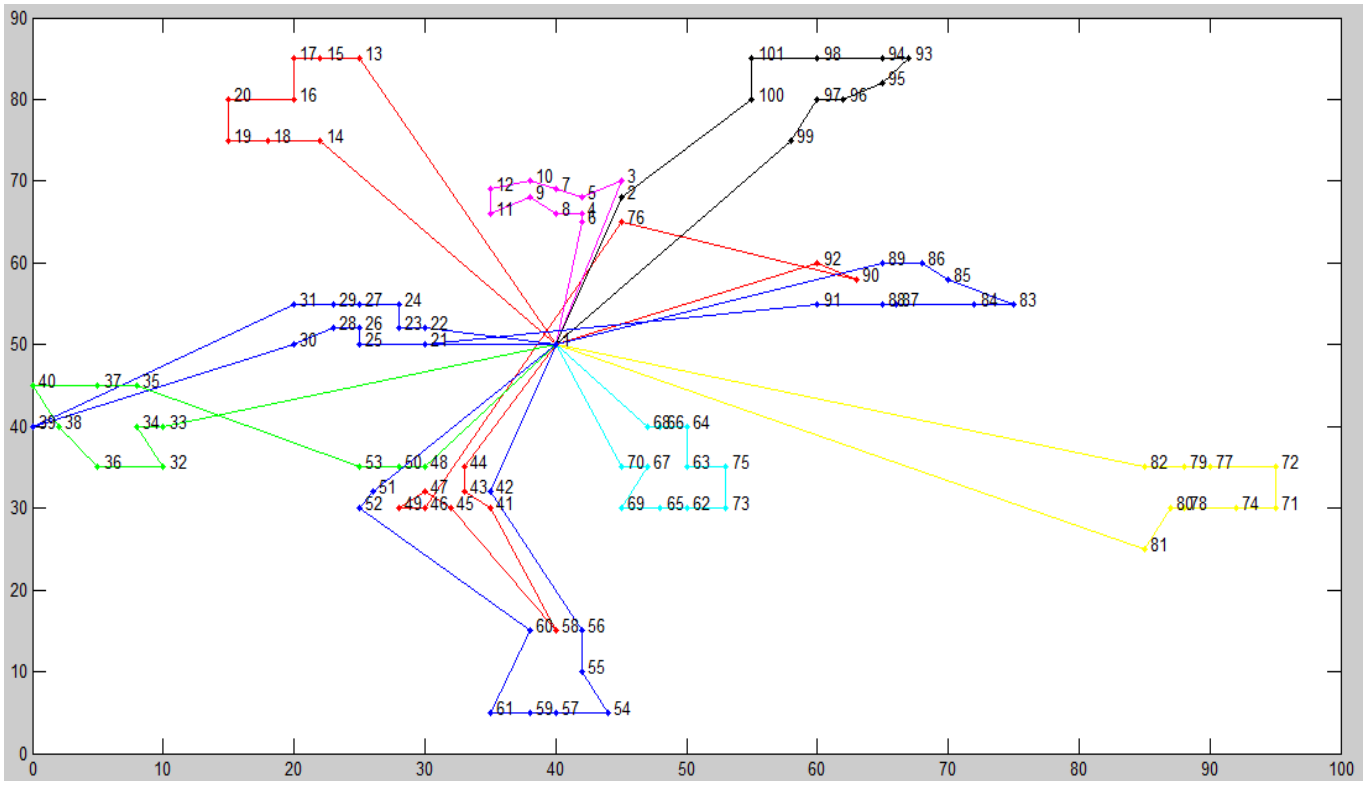
Στο πρόβλημα C109,όπως και στα προηγούμενα, έχουμε n=100, Capacity=200, maximum vehicle number=25. Η λύση που προκύπτει έχει Κόστος=969.42 με χρήση 10 οχημάτων, ενώ το βέλτιστο είναι 828.94 με 10 οχήματα. Δηλαδή η απόκλιση από το βέλτιστο, σε αυτή την περίπτωση θα είναι 16.9%. Ακολουθεί η γραφική αναπαράσταση της λύσης

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

Κόστος Λύσης = 969.42

Βέλτιστο Κόστος = 828.94

Απόκλιση από το Βέλτιστο = 0.169


0 10 20 30 40 50 60 70 80 90 100
0 10 20 30 40 50 60 70 80 90
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101

5.10 Πρόβλημα R101

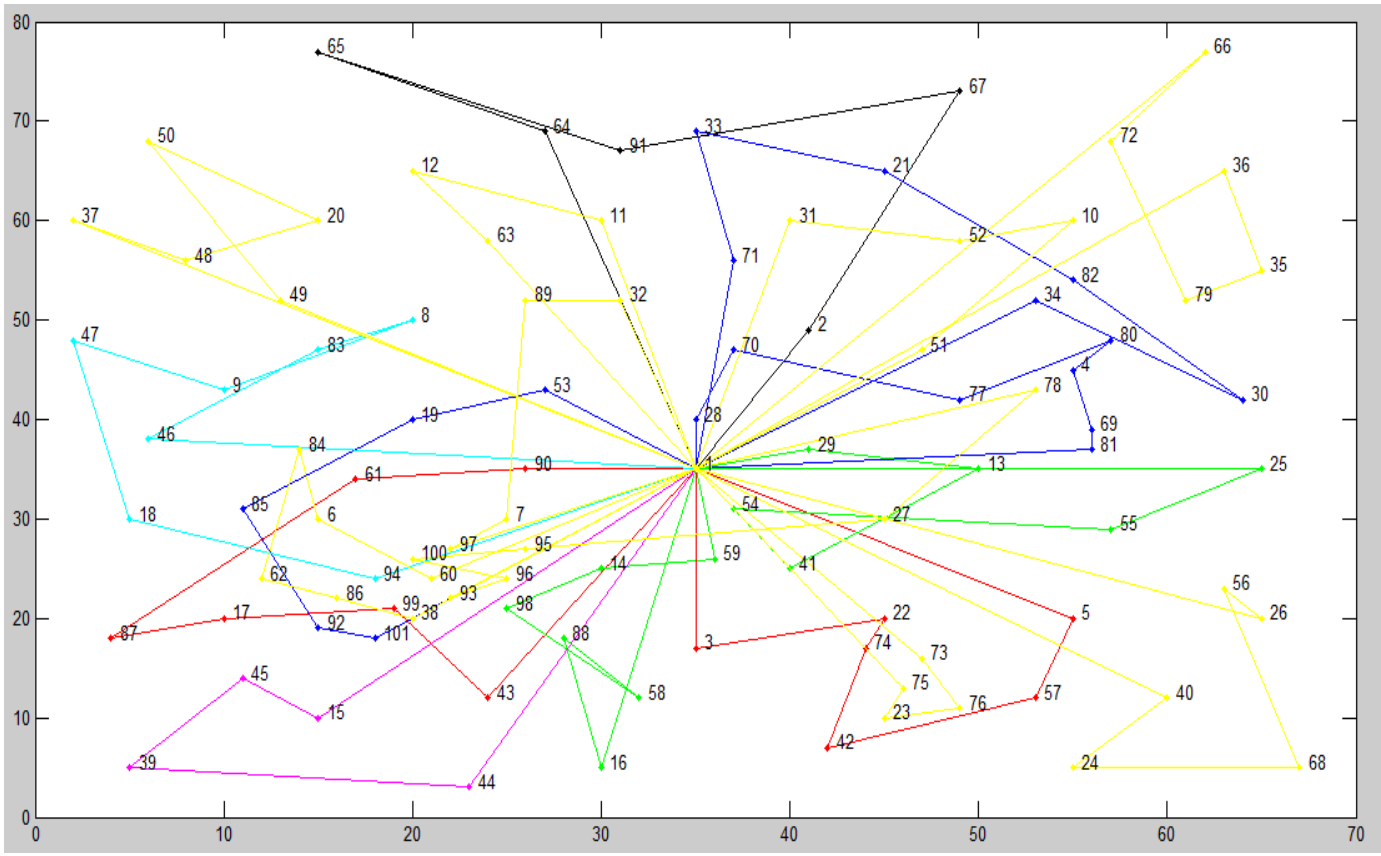
Στο πρόβλημα R101,όπως και στα προηγούμενα, έχουμε n=100, Capacity=200, maximum vehicle number=25. Η λύση που προκύπτει έχει Κόστος=1750.59 με χρήση 19 οχημάτων, ενώ το βέλτιστο είναι 1645.79 με 19 οχήματα. Δηλαδή η απόκλιση από το βέλτιστο, σε αυτή την περίπτωση θα είναι 6.3%. Ακολουθεί η γραφική αναπαράσταση της λύσης

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

Κόστος Λύσης = 1750.59

Βέλτιστο Κόστος = 1645.79

Απόκλιση από το Βέλτιστο = 0.063


0
10
20
30
40
50
60
70
80
0
10
20
30
40
50
60
70
65
66
67
64
91
33
50
72
12
21
36
37
20
11
31
10
63
52
71
35
48
82
49
89
32
34
79
8
2
47
83
70
51
80
4
9
53
77
78
30
19
28
46
84
69
81
29
90
1
13
25
61
85
6
7
54
18
27
55
100
97
95
59
94
60
96
14
41
62
86
93
98
56
26
99
38
22
5
17
92
101
88
3
74
87
73
45
43
58
75
57
40
15
23
76
42
39
16
24
68
44

5.11 Πρόβλημα R102

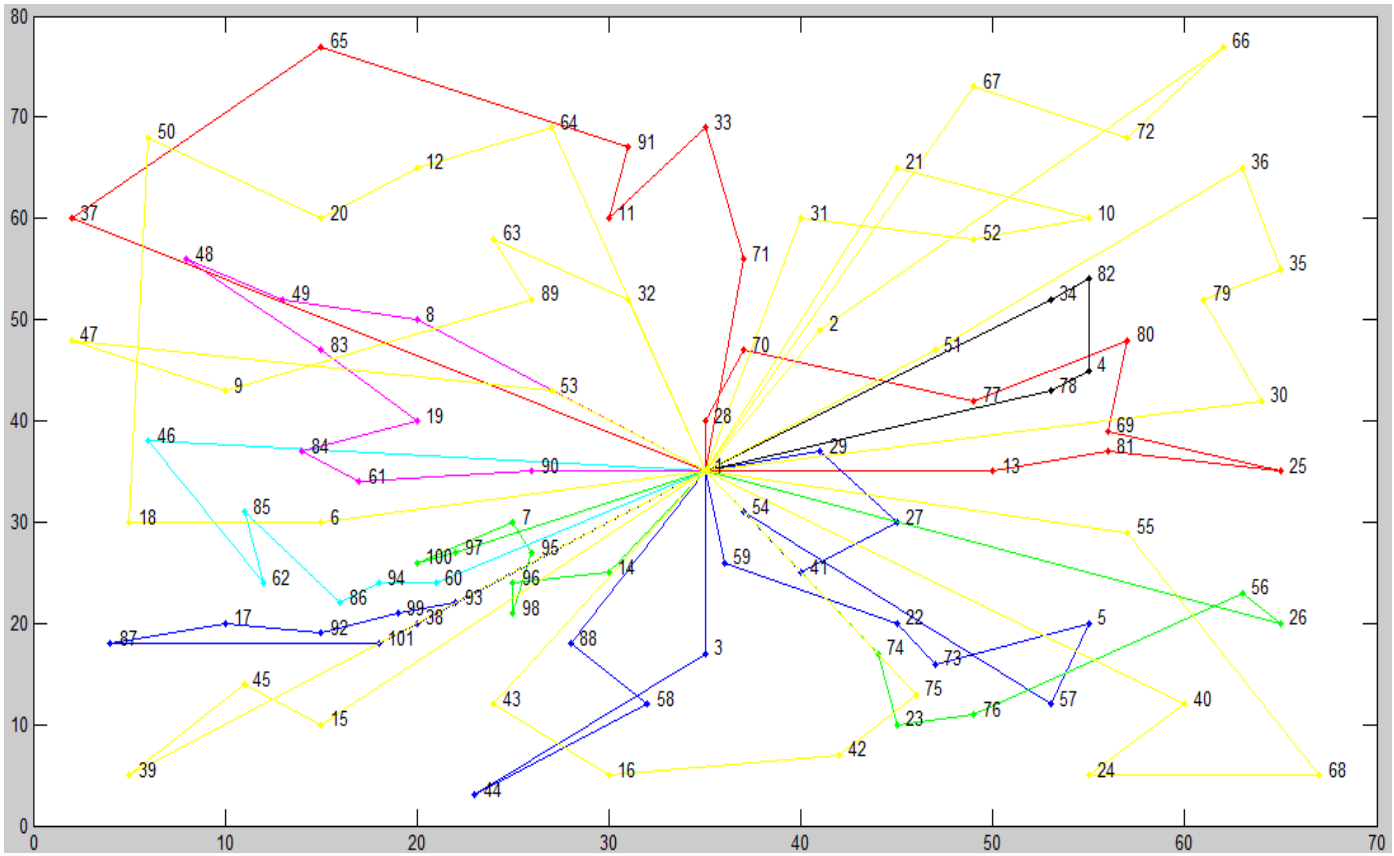
Στο πρόβλημα R102,όπως και στα προηγούμενα, έχουμε n=100, Capacity=200, maximum vehicle number=25. Η λύση που προκύπτει έχει Κόστος=1622.68 με χρήση 18 οχημάτων, ενώ το βέλτιστο είναι 1486.12 με 17 οχήματα. Δηλαδή η απόκλιση από το βέλτιστο, σε αυτή την περίπτωση θα είναι 9.1%. Ακολουθεί η γραφική αναπαράσταση της λύσης

1	28	70	77	80	69	25	81	13	1	0
1	29	27	41	54	57	5	73	22	59	1
1	97	100	7	95	98	96	14	1		
1	46	62	85	86	94	60	1			
1	8	49	48	83	19	84	61	90	1	
1	2	66	72	67	1					
1	34	82	4	78	1					
1	37	65	91	11	33	71	1			
1	93	99	92	17	87	101	38	1		
1	88	58	44	3	1					
1	74	23	76	56	26	1				
1	31	52	10	21	1					
1	64	12	20	50	18	6	1			
1	43	16	42	75	1					
1	30	79	35	36	51	1				
1	32	63	89	9	47	53	1			
1	15	45	39	1						
1	40	24	68	55	1					

Κόστος Λύσης = 1622.48

Βέλτιστο Κόστος = 1486.12

Απόκλιση από το Βέλτιστο = 0.091


0
10
20
30
40
50
60
70
80
0
10
20
30
40
50
60
70
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101

5.12 Πρόβλημα R103

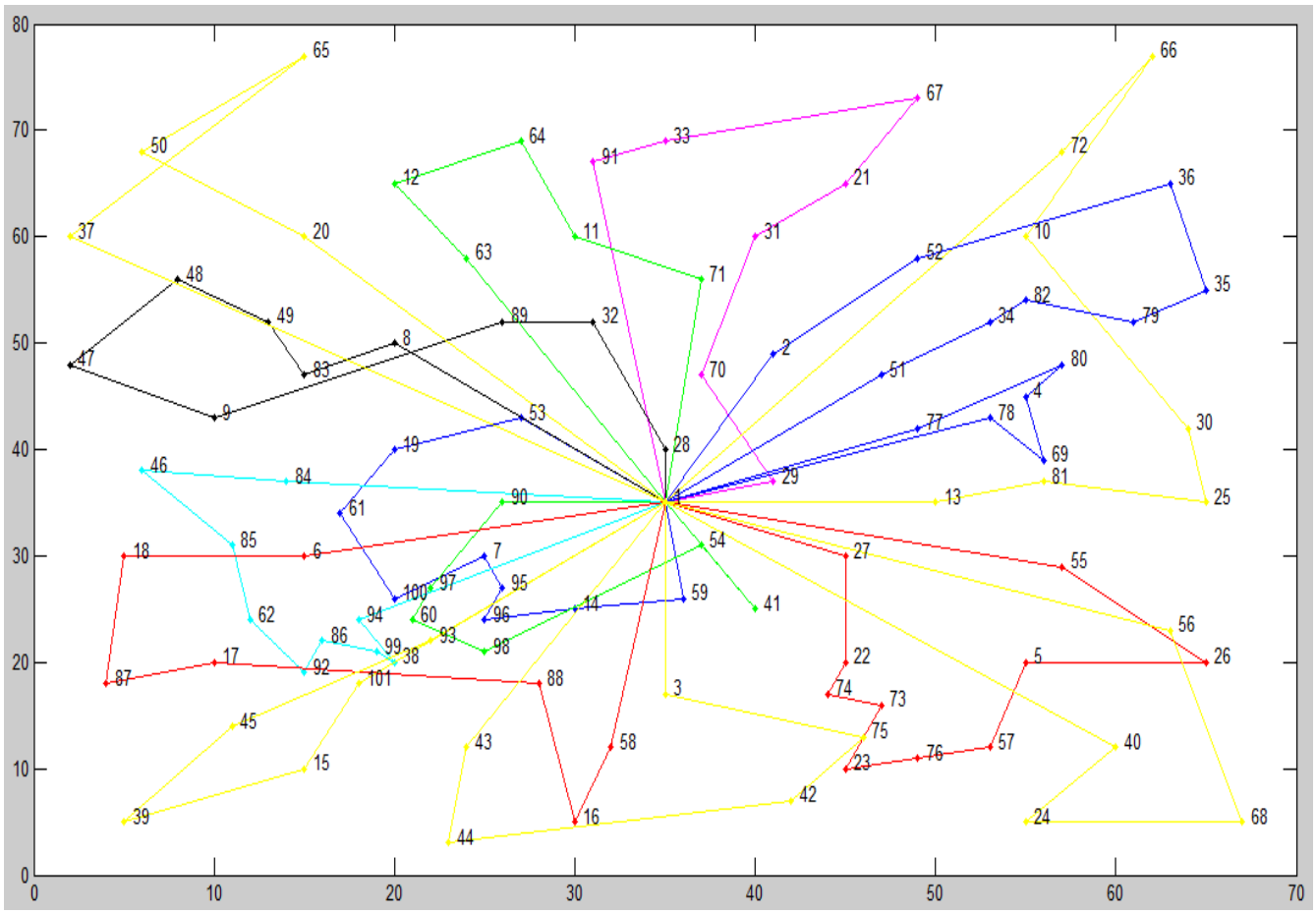
Στο πρόβλημα R103,όπως και στα προηγούμενα, έχουμε n=100, Capacity=200, maximum vehicle number=25. Η λύση που προκύπτει έχει Κόστος=1371.73 με χρήση 15 οχημάτων, ενώ το βέλτιστο είναι 1292.68 με 13 οχήματα. Δηλαδή η απόκλιση από το βέλτιστο, σε αυτή την περίπτωση θα είναι 6%. Ακολουθεί η γραφική αναπαράσταση της λύσης.

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

Κόστος Λύσης = 1371.73

Βέλτιστο Κόστος = 1292.68

Απόκλιση από το Βέλτιστο = 0.060


0
10
20
30
40
50
60
70
80
65
66
67
50
64
91
33
72
12
21
36
37
20
11
31
10
63
52
71
48
35
82
49
89
32
34
79
47
8
2
80
83
51
4
9
53
77
78
30
19
28
69
46
84
29
81
90
13
25
61
85
18
6
7
54
27
55
97
95
100
59
41
62
94
60
14
56
86
96
99
93
98
22
5
26
17
38
92
101
88
3
74
73
87
45
75
43
58
57
40
15
23
76
39
42
24
68
16
44

5.13 Πρόβλημα R104

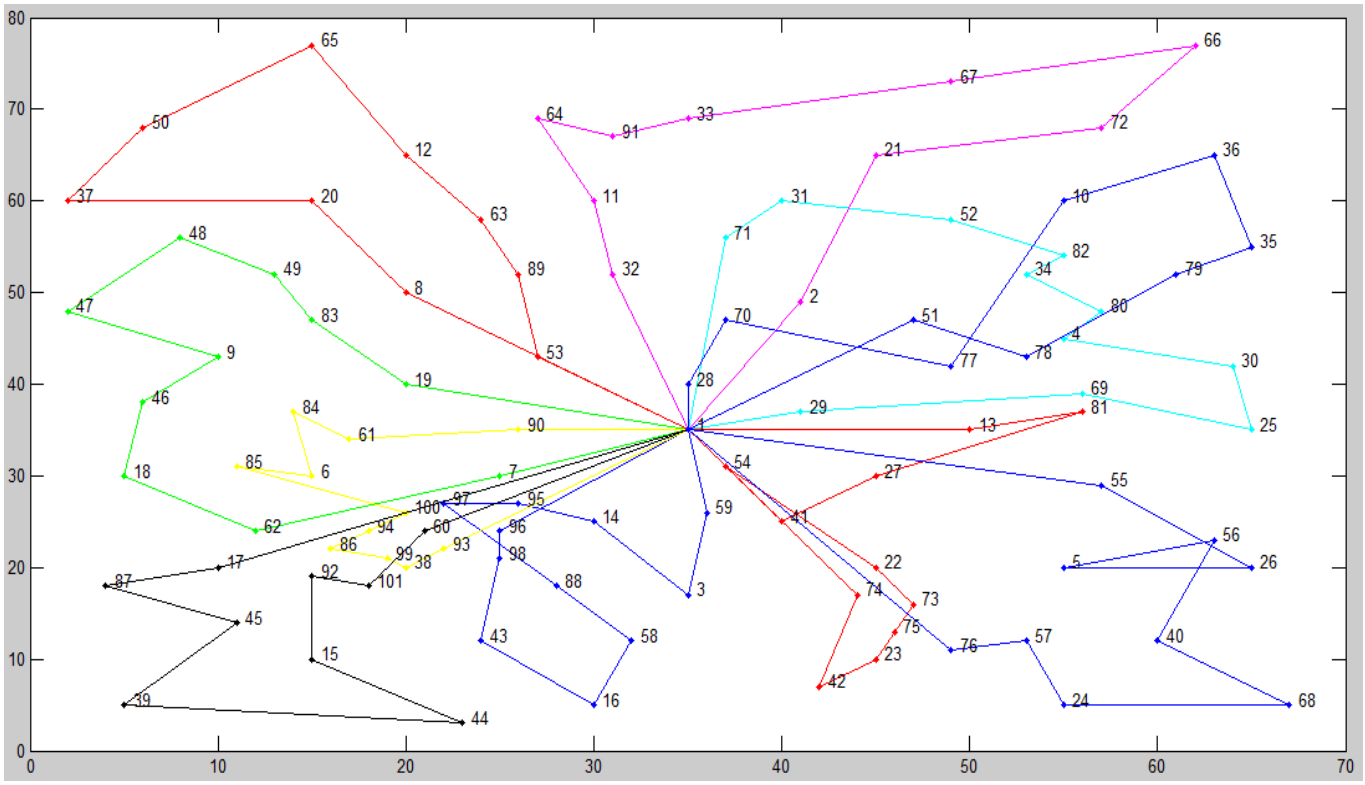
Στο πρόβλημα R104,όπως και στα προηγούμενα, έχουμε n=100, Capacity=200, maximum vehicle number=25. Η λύση που προκύπτει έχει Κόστος=1079.19 με χρήση 10 οχημάτων, ενώ το βέλτιστο είναι 982.01 με 10 οχήματα. Δηλαδή η απόκλιση από το βέλτιστο, σε αυτή την περίπτωση θα είναι 9.9%. Ακολουθεί η γραφική αναπαράσταση της λύσης

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

Κόστος Λύσης = 1079.19

Βέλτιστο Κόστος = 982.01

Απόκλιση από το Βέλτιστο = 0.099


0
10
20
30
40
50
60
70
80
65
66
67
50
64
91
33
72
12
21
36
37
20
63
11
31
10
52
48
89
32
71
82
35
49
34
79
47
8
2
80
83
70
51
4
53
77
78
30
9
19
28
69
46
84
29
81
61
90
13
25
85
6
54
55
18
7
27
59
100
97
95
41
14
56
94
60
96
62
86
99
38
93
22
5
26
17
98
92
101
88
74
73
87
3
45
75
76
57
40
43
58
23
15
42
24
68
39
16
44

5.14 Πρόβλημα R105

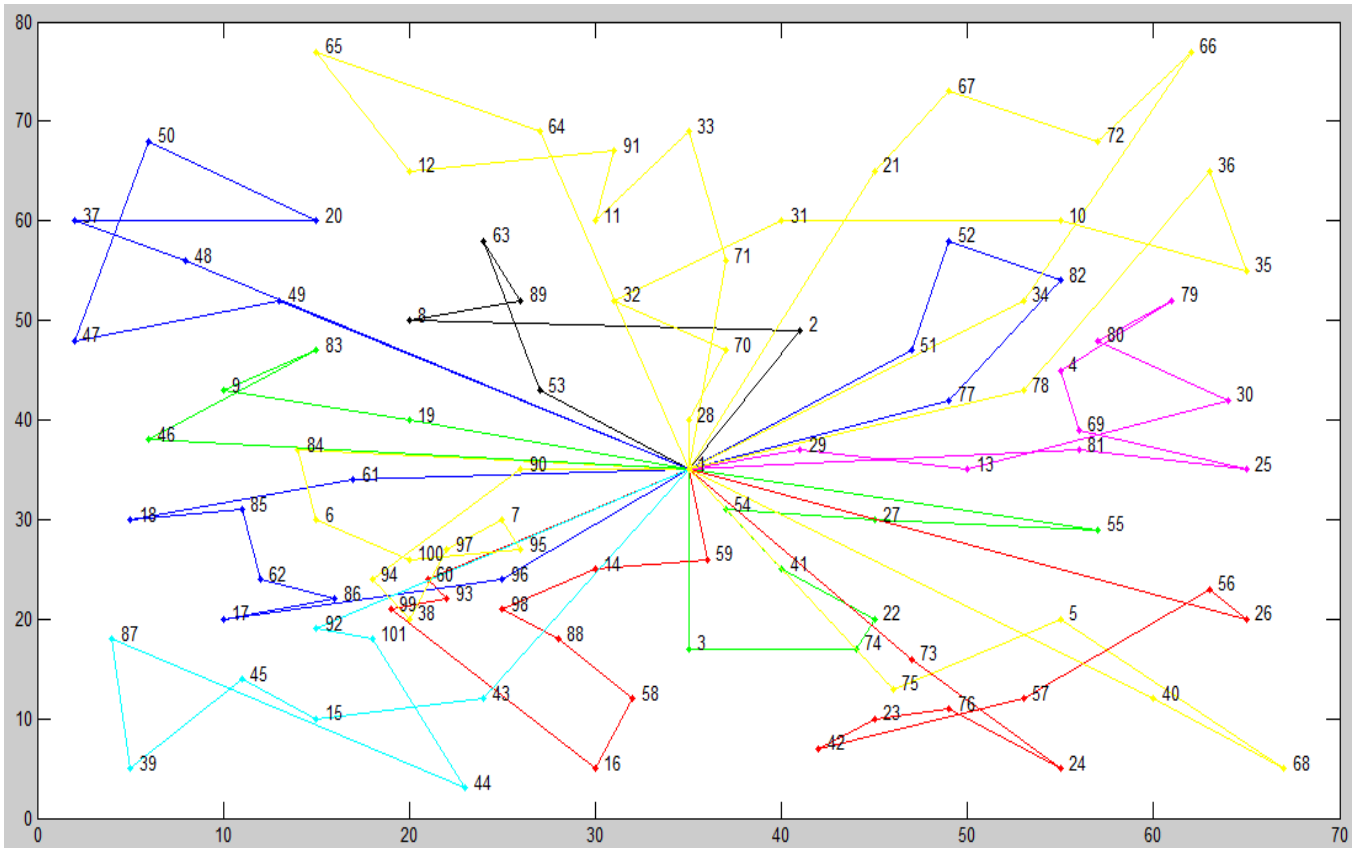
Στο πρόβλημα R105,όπως και στα προηγούμενα, έχουμε n=100, Capacity=200, maximum vehicle number=25. Η λύση που προκύπτει έχει Κόστος=1477.82 με χρήση 15 οχημάτων, ενώ το βέλτιστο είναι 1377.11 με 14 οχήματα. Δηλαδή η απόκλιση από το βέλτιστο, σε αυτή την περίπτωση θα είναι 7.3%. Ακολουθεί η γραφική αναπαράσταση της λύσης

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

Κόστος Λύσης = 1477.82

Βέλτιστο Κόστος = 1377.11

Απόκλιση από το Βέλτιστο =0.073


0
10
20
30
40
50
60
70
80
65
66
67
64
91
33
72
12
21
36
50
11
31
10
37
20
63
52
48
71
35
82
89
32
34
79
49
8
2
80
47
70
51
4
83
53
77
78
30
9
28
19
46
84
29
69
81
13
25
90
61
18
85
6
7
54
27
55
100
97
95
59
41
62
94
60
96
14
56
86
93
99
38
98
22
5
26
17
92
101
88
3
74
73
87
45
75
40
43
58
15
57
23
76
42
24
39
16
68
44

5.15 Πρόβλημα R106

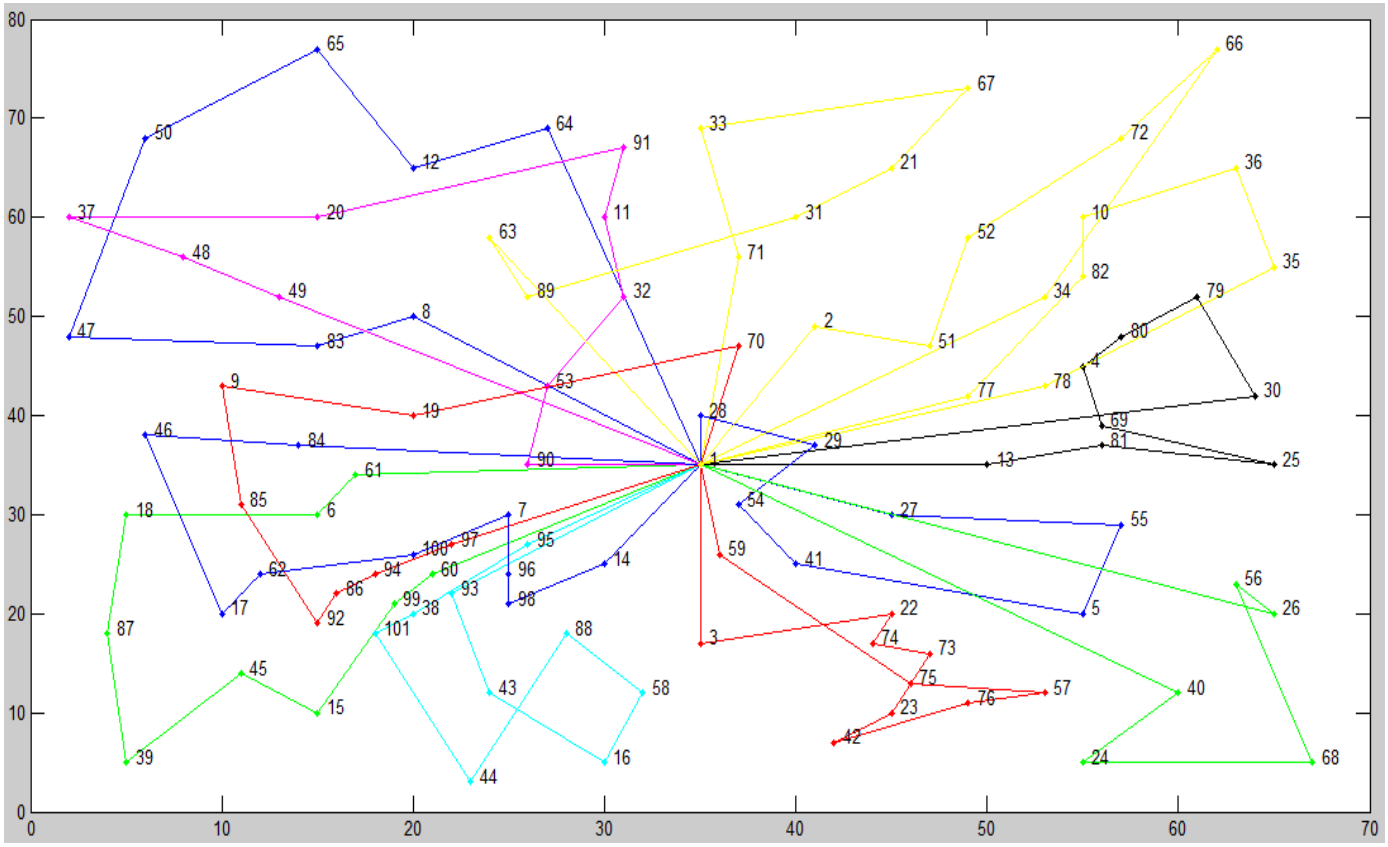
Στο πρόβλημα R106,όπως και στα προηγούμενα, έχουμε n=100, Capacity=200, maximum vehicle number=25. Η λύση που προκύπτει έχει Κόστος=1324.93 με χρήση 13 οχημάτων, ενώ το βέλτιστο είναι 1252.03 με 12 οχήματα. Δηλαδή η απόκλιση από το βέλτιστο, σε αυτή την περίπτωση θα είναι 5.8%. Ακολουθεί η γραφική αναπαράσταση της λύσης

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

Κόστος Λύσης = 1324.93

Βέλτιστο Κόστος = 1252.03

Απόκλιση από το Βέλτιστο =0.058


0
10
20
30
40
50
60
70
80
65
50
12
64
91
37
20
11
48
49
8
47
83
63
89
32
33
67
21
31
71
72
66
36
52
10
35
82
34
79
80
4
30
69
81
25
13
2
51
70
53
9
19
28
29
1
77
78
46
84
90
61
85
6
18
7
54
27
55
59
41
5
56
26
62
94
86
92
100
97
95
60
96
98
14
99
38
93
101
17
87
45
15
39
43
88
58
16
44
3
22
74
73
75
57
76
23
42
40
24
68

5.16 Πρόβλημα R107

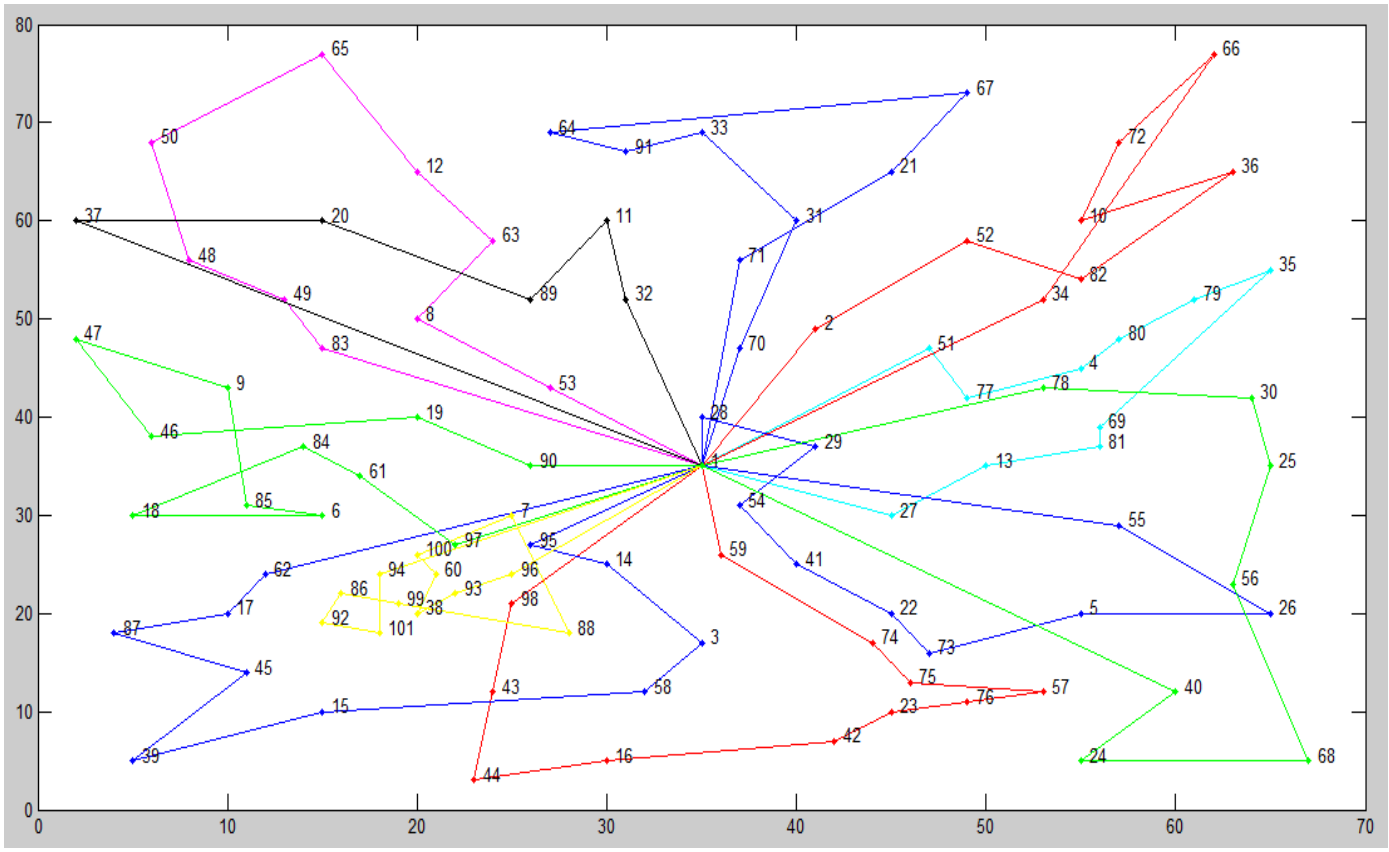
Στο πρόβλημα R107 έχουμε n=100, Capacity=200, maximum vehicle number=25. Η λύση που προκύπτει έχει Κόστος=1176.11 με χρήση 11 οχημάτων, ενώ το βέλτιστο είναι 1104.66 με 10 οχήματα. Δηλαδή η απόκλιση από το βέλτιστο, σε αυτή την περίπτωση θα είναι 6.4%. Ακολουθεί η γραφική αναπαράσταση της λύσης.

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

Κόστος Λύσης = 1176.11

Βέλτιστο Κόστος = 1104.66

Απόκλιση από το Βέλτιστο = 0.064


0
10
20
30
40
50
60
70
80
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101

5.17 Πρόβλημα R108

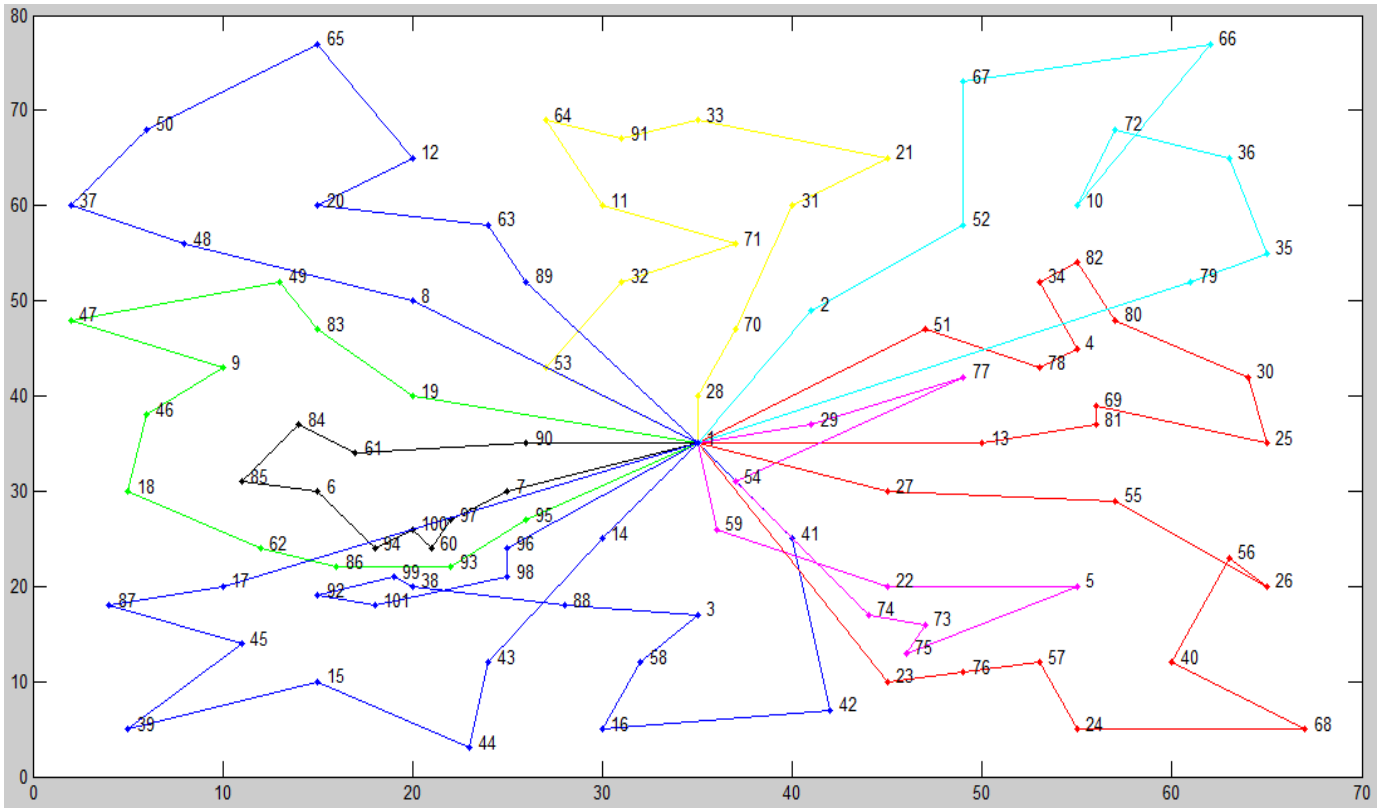
Στο πρόβλημα R108,όπως και στα προηγούμενα, έχουμε n=100, Capacity=200, maximum vehicle number=25. Η λύση που προκύπτει έχει Κόστος=1037.15 με χρήση 10 οχημάτων, ενώ το βέλτιστο είναι 960.88 με 9 οχήματα. Δηλαδή η απόκλιση από το βέλτιστο, σε αυτή την περίπτωση θα είναι 7.9%. Ακολουθεί η γραφική αναπαράσταση της λύσης

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

Κόστος Λύσης = 1037.15

Βέλτιστο Κόστος = 960.88

Απόκλιση από το Βέλτιστο = 0.079


0
10
20
30
40
50
60
70
80
65
50
12
37
20
63
48
89
49
8
47
83
9
19
46
84
61
90
18
85
6
7
62
100
97
95
94
60
96
86
99
93
98
17
38
92
101
88
87
3
45
14
15
43
58
39
44
16
42
64
91
33
21
11
31
71
32
53
70
28
2
67
52
66
72
36
10
35
82
34
79
80
4
51
78
77
30
29
69
81
13
25
1
54
27
55
59
41
56
22
5
26
74
73
75
23
76
57
40
24
68

5.18 Πρόβλημα R109

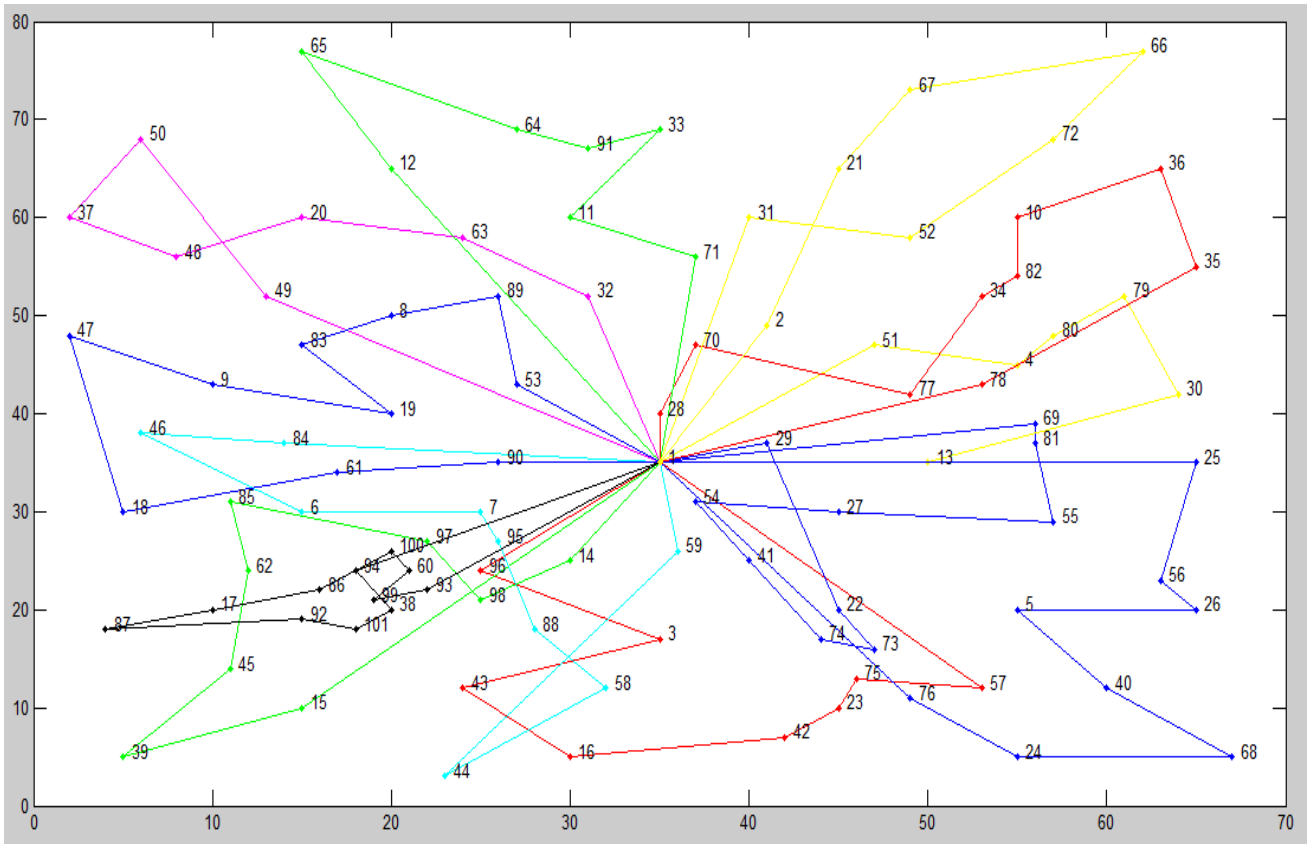
Στο πρόβλημα R109,όπως και στα προηγούμενα, έχουμε n=100, Capacity=200, maximum vehicle number=25. Η λύση που προκύπτει έχει Κόστος=1253.33 με χρήση 12 οχημάτων, ενώ το βέλτιστο είναι 1194.73 με 11 οχήματα. Δηλαδή η απόκλιση από το βέλτιστο, σε αυτή την περίπτωση θα είναι 4.9%. Ακολουθεί η γραφική αναπαράσταση της λύσης

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

Κόστος Λύσης = 1253.33

Βέλτιστο Κόστος = 1194.73

Απόκλιση από το Βέλτιστο = 0.049


0
10
20
30
40
50
60
70
80
0
10
20
30
40
50
60
70
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101

5.19 Πρόβλημα R110

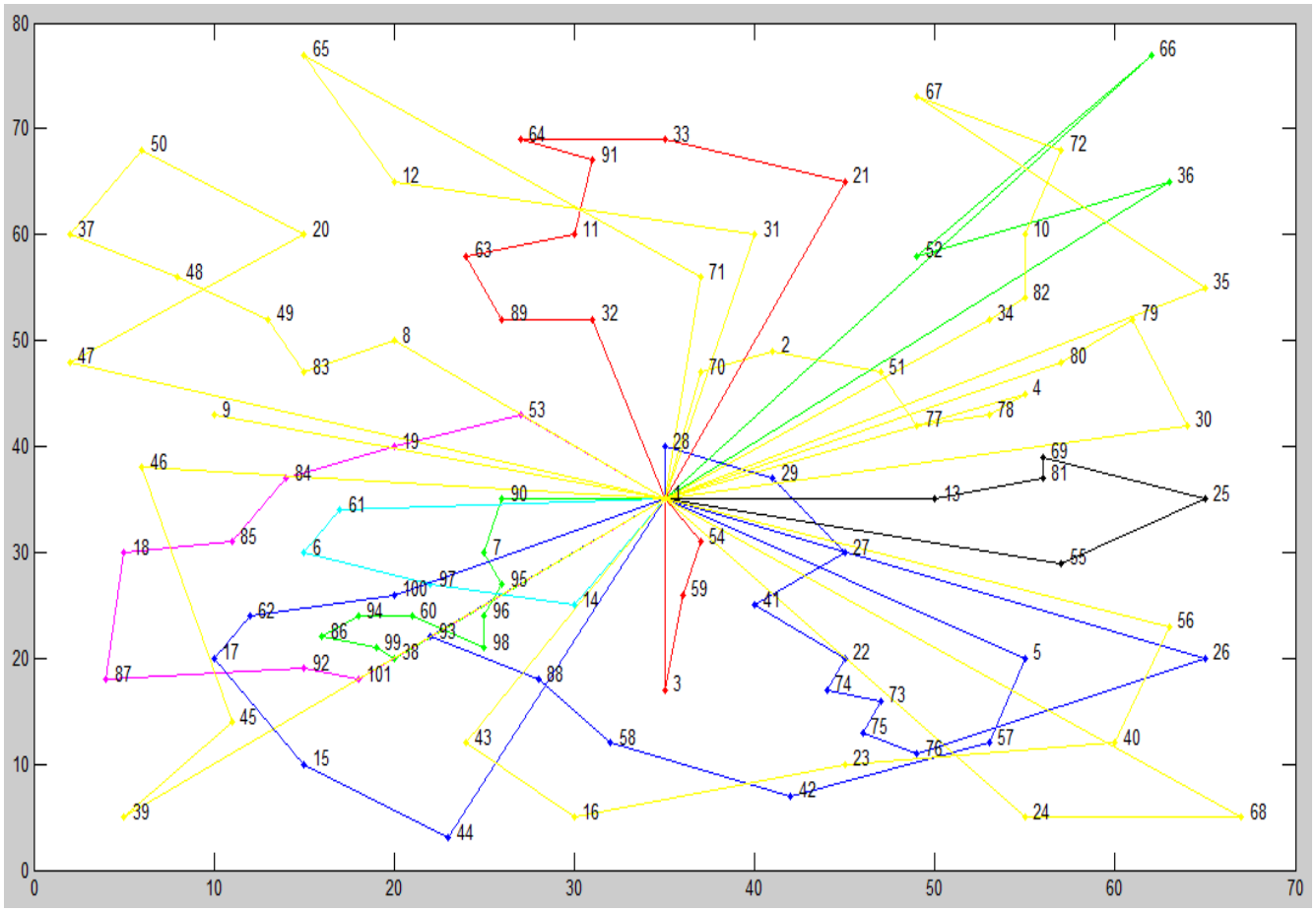
Στο πρόβλημα R110,όπως και στα προηγούμενα, έχουμε n=100, Capacity=200, maximum vehicle number=25. Η λύση που προκύπτει έχει Κόστος=1186.98 με χρήση 11 οχημάτων, ενώ το βέλτιστο είναι 1118.84 με 10 οχήματα. Δηλαδή η απόκλιση από το βέλτιστο, σε αυτή την περίπτωση θα είναι 6%. Ακολουθεί η γραφική αναπαράσταση της λύσης.

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

Κόστος Λύσης = 1186.98

Βέλτιστο Κόστος = 1118.84

Απόκλιση από το Βέλτιστο = 0.060


0
10
20
30
40
50
60
70
80
65
66
67
50
64
33
72
91
21
12
36
37
20
11
31
10
52
63
71
48
35
82
49
89
32
34
79
47
8
2
80
83
70
51
4
9
53
77
78
30
19
28
69
46
84
29
81
90
13
25
61
6
7
54
27
18
85
55
100
97
95
59
41
14
62
94
60
96
56
86
99
93
98
22
5
26
17
92
38
101
88
3
74
87
73
45
43
58
75
57
40
15
23
76
42
39
16
24
68
44

5.20 Πρόβλημα R111

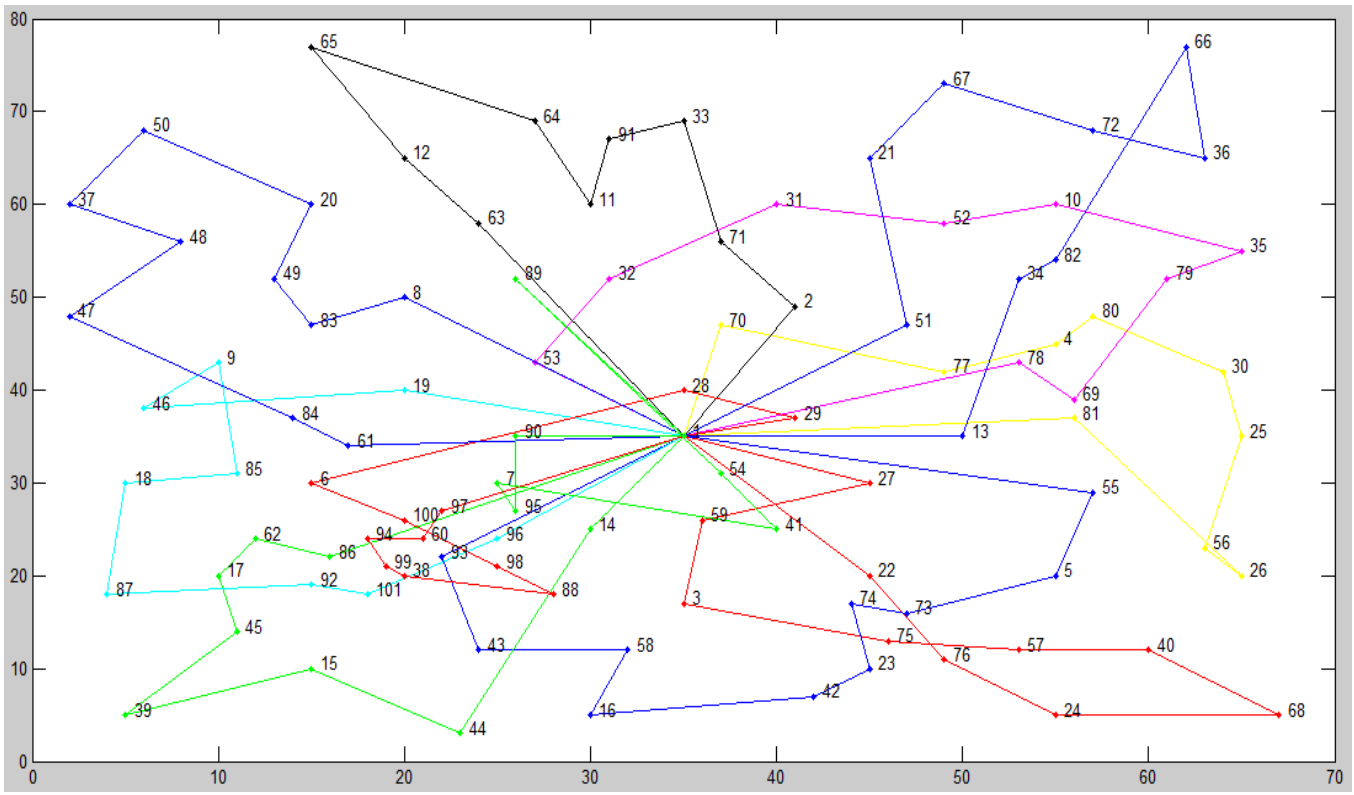
Στο πρόβλημα R111,όπως και στα προηγούμενα, έχουμε n=100, Capacity=200, maximum vehicle number=25. Η λύση που προκύπτει έχει Κόστος=1186.98 με χρήση 11 οχημάτων, ενώ το βέλτιστο είναι 1096.72 με 10 οχήματα. Δηλαδή η απόκλιση από το βέλτιστο, σε αυτή την περίπτωση θα είναι 8.2%. Ακολουθεί η γραφική αναπαράσταση της λύσης

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

Κόστος Λύσης = 1186.98

Βέλτιστο Κόστος = 1096.72

Απόκλιση από το Βέλτιστο = 0.082


0
10
20
30
40
50
60
70
80
65
64
91
33
66
67
72
36
50
12
11
21
37
20
63
31
52
10
48
71
35
49
89
32
82
34
79
47
8
83
2
70
51
80
4
9
53
77
78
30
19
28
69
46
84
29
81
90
13
25
61
1
6
7
54
27
55
18
85
100
97
95
59
41
62
94
60
96
14
56
86
99
93
98
22
17
38
5
87
92
101
88
3
74
73
26
45
75
57
40
43
58
76
15
23
39
16
42
24
68
44

5.21 Πρόβλημα R112

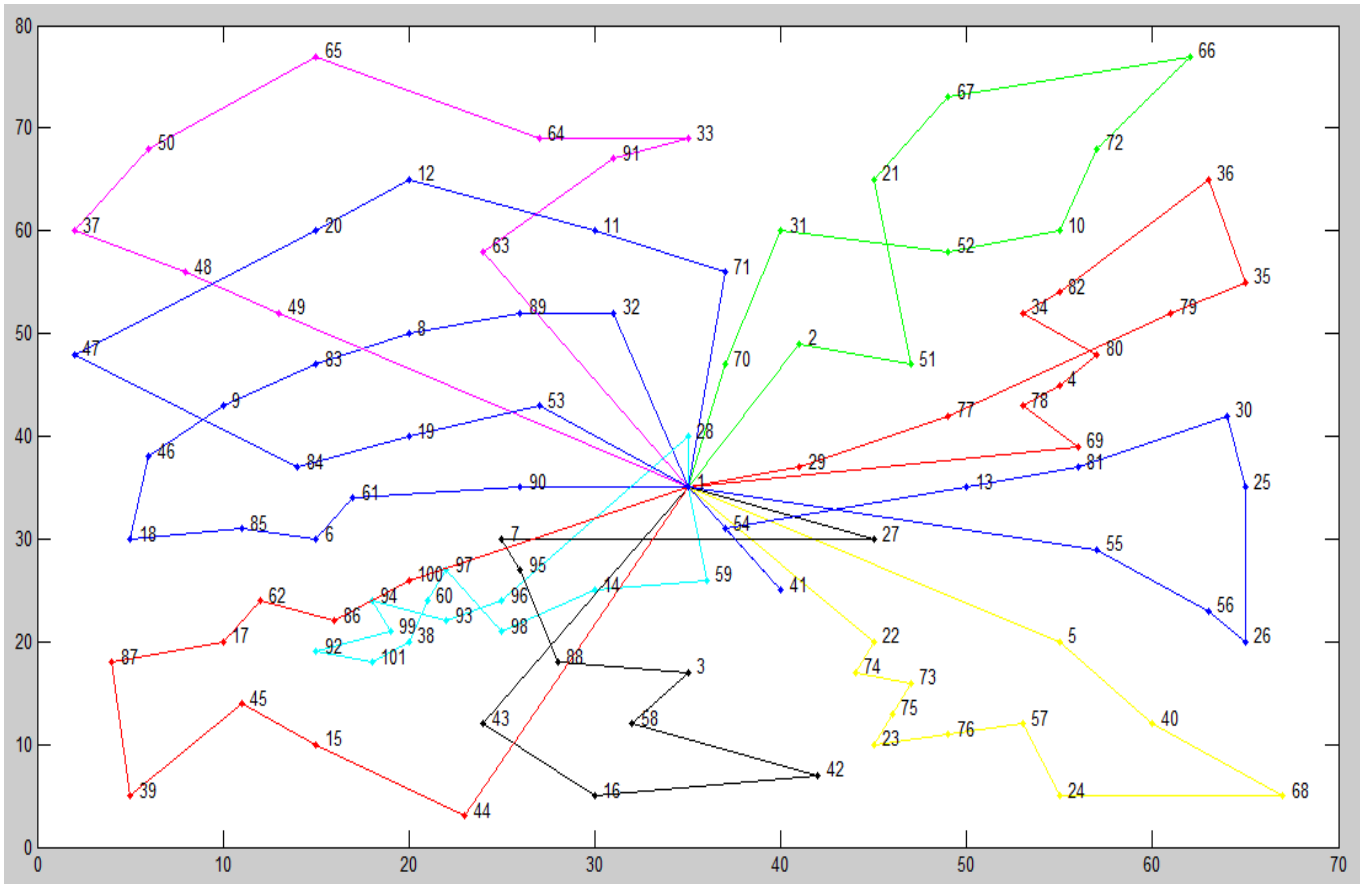
Στο πρόβλημα R112,όπως και στα προηγούμενα, έχουμε n=100, Capacity=200, maximum vehicle number=25. Η λύση που προκύπτει έχει Κόστος=1060.51 με χρήση 10 οχημάτων, ενώ το βέλτιστο είναι 982.14 με 9 οχήματα. Δηλαδή η απόκλιση από το βέλτιστο, σε αυτή την περίπτωση θα είναι 7.9%. Ακολουθεί η γραφική αναπαράσταση της λύσης.

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

Κόστος Λύσης = 1060.51

Βέλτιστο Κόστος = 982.14

Απόκλιση από το Βέλτιστο = 0.079


0
10
20
30
40
50
60
70
80
65
50
37
48
49
47
46
18
85
6
84
9
83
8
89
32
12
20
64
33
91
63
11
71
53
19
61
90
7
95
97
100
94
60
96
98
93
99
38
92
101
86
62
17
87
39
45
15
44
43
88
3
58
16
42
59
14
28
70
2
51
31
52
10
72
21
67
66
41
54
27
1
29
77
78
4
80
79
34
82
36
35
69
81
13
30
25
55
56
26
5
22
74
73
75
23
76
57
24
40
68

5.22 Πρόβλημα RC101

Στο πρόβλημα RC101,όπως και στα προηγούμενα, έχουμε n=100, Capacity=200, maximum vehicle number=25. Η λύση που προκύπτει έχει Κόστος=1827.33 με χρήση 17 οχημάτων, ενώ το βέλτιστο είναι 1696.94 με 14 οχήματα. Δηλαδή η απόκλιση από το βέλτιστο, σε αυτή την περίπτωση θα είναι 7.6%. Ακολουθεί η γραφική αναπαράσταση της λύσης

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

Κόστος Λύσης = 1827.33

Βέλτιστο Κόστος = 1696.94

Απόκλιση από το Βέλτιστο = 0.076

5.23 Πρόβλημα RC102

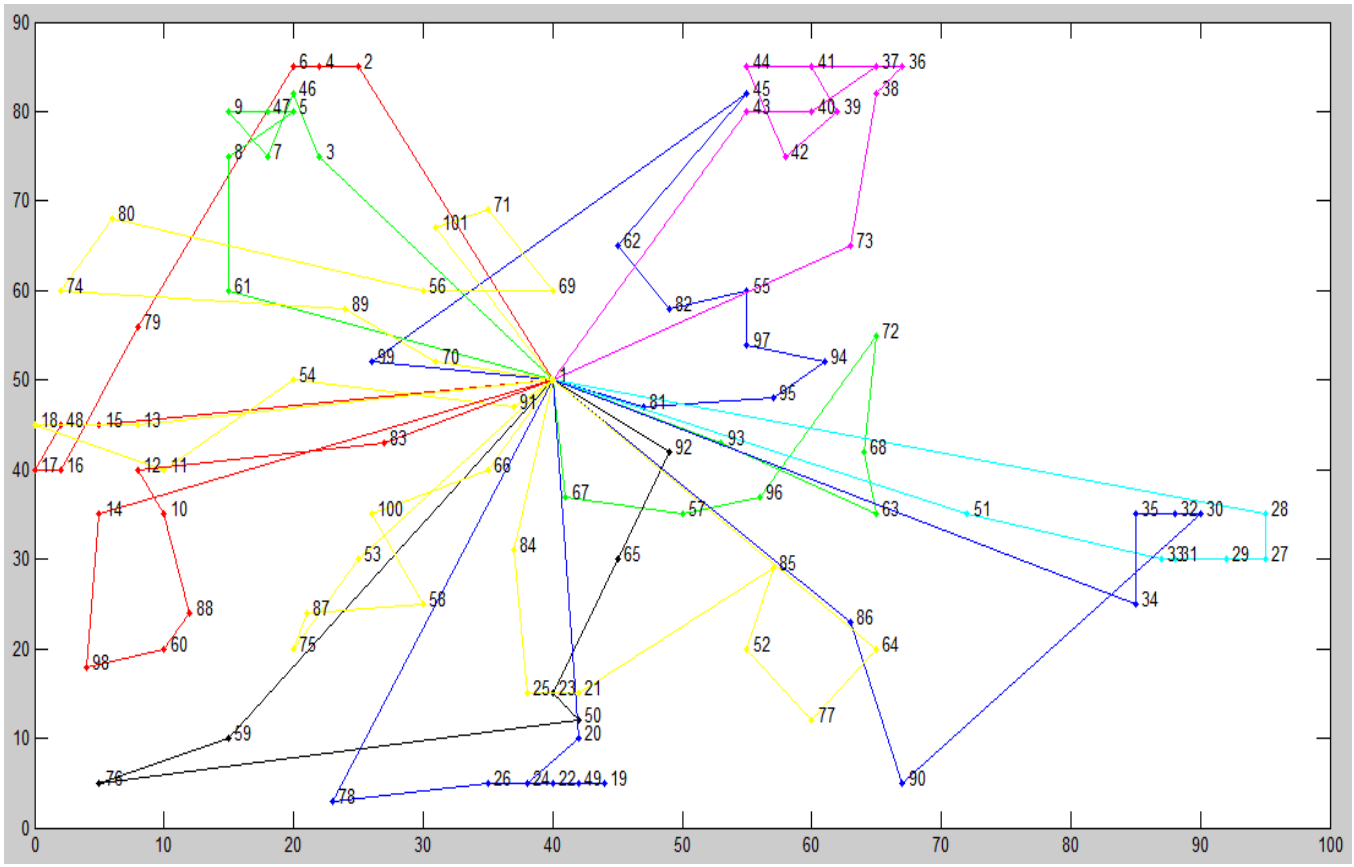
Στο πρόβλημα RC102, έχουμε n=100, Capacity=200, maximum vehicle number=25. Η λύση που προκύπτει έχει Κόστος=1673.18 με χρήση 14 οχημάτων, ενώ το βέλτιστο είναι 1554.75 με 12 οχήματα. Δηλαδή η απόκλιση από το βέλτιστο, σε αυτή την περίπτωση θα είναι 7.6%. Ακολουθεί η γραφική αναπαράσταση της λύσης.

1	15	48	17	16	79	6	4	2	1	0	0
1	99	45	62	82	55	97	94	95	81	1	
1	93	63	68	72	96	57	67	1	0	0	0
1	28	27	29	31	33	51	1				
1	43	40	37	41	39	42	44	36	38	73	1
1	91	54	11	18	13	1					
1	92	65	23	50	76	59	1	0	0	0	0
1	83	12	10	88	60	98	14	1			
1	34	35	32	30	90	86	1	0	0	0	0
1	20	24	49	19	22	26	78	1			
1	3	46	7	9	47	5	8	61	1	0	0
1	66	100	58	87	75	53	1				
1	70	89	74	80	56	69	71	101	1	0	0
1	64	77	52	85	21	25	84	1			

Κόστος Λύσης = 1673.18

Βέλτιστο Κόστος = 1554.75

Απόκλιση από το Βέλτιστο = 0.076


0 10 20 30 40 50 60 70 80 90 100
0 10 20 30 40 50 60 70 80 90
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101

5.24 Πρόβλημα RC103

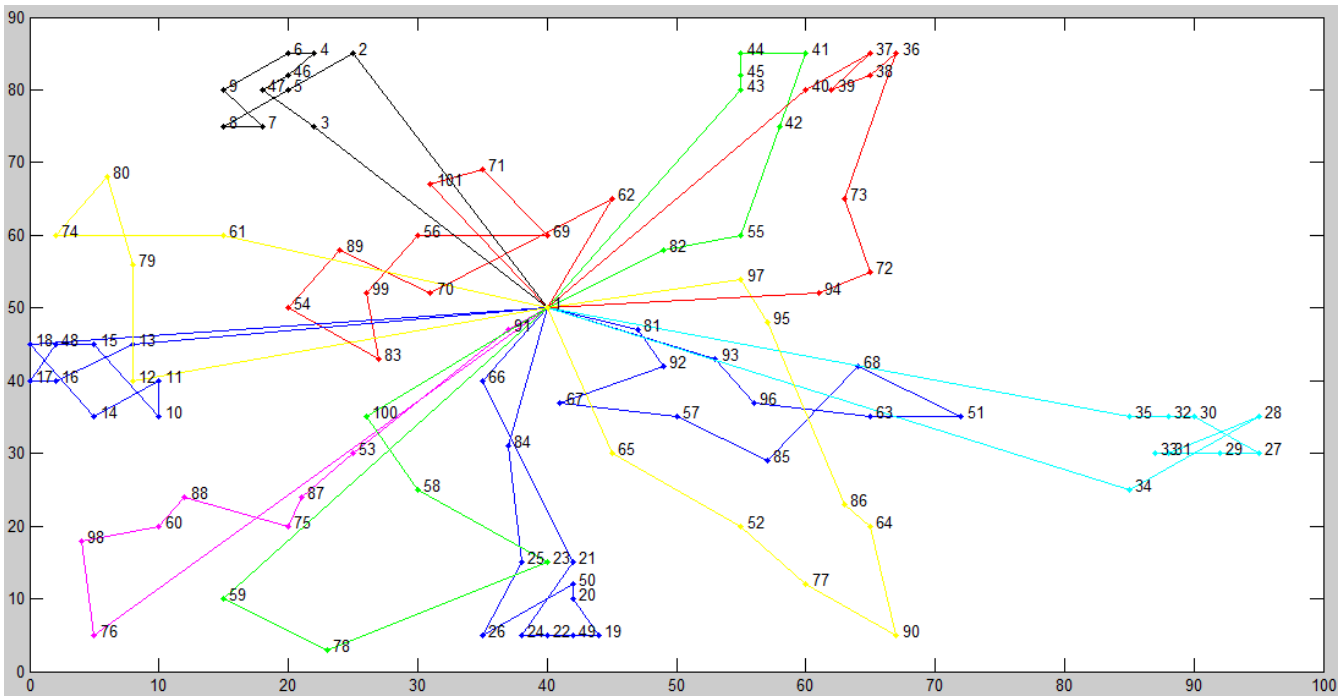
Στο πρόβλημα RC103,όπως και στα προηγούμενα, έχουμε n=100, Capacity=200, maximum vehicle number=25. Η λύση που προκύπτει έχει Κόστος=1409.09 με χρήση 12 οχημάτων, ενώ το βέλτιστο είναι 1261.67 με 11 οχήματα. Δηλαδή η απόκλιση από το βέλτιστο, σε αυτή την περίπτωση θα είναι 11.6%. Ακολουθεί η γραφική αναπαράσταση της λύσης

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

Κόστος Λύσης = 1409.09

Βέλτιστο Κόστος = 1261.67

Απόκλιση από το Βέλτιστο = 0.116


0 10 20 30 40 50 60 70 80 90 100
0 10 20 30 40 50 60 70 80 90
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101

5.25 Πρόβλημα RC104

Στο πρόβλημα RC104,όπως και στα προηγούμενα, έχουμε n=100, Capacity=200, maximum vehicle number=25. Η λύση που προκύπτει έχει Κόστος=1277.26 με χρήση 11 οχημάτων, ενώ το βέλτιστο είναι 1135.48 με 10 οχήματα. Δηλαδή η απόκλιση από το βέλτιστο, σε αυτή την περίπτωση θα είναι 12.4%. Ακολουθεί η γραφική αναπαράσταση της λύσης

1	91	66	67	84	65	57	96	85	86	64	52	92	1
1	81	93	95	94	72	97	55	82	69	56	101	71	1
1	89	3	7	8	54	11	14	17	15	13	1	0	0
1	48	18	16	12	10	98	76	59	1	0	0	0	0
1	70	99	100	88	60	75	87	53	83	1	0	0	0
1	68	31	29	27	28	30	32	1	0	0	0	0	0
1	73	42	45	40	38	36	37	41	44	43	62	1	0
1	63	51	77	90	21	23	1	0	0	0	0	0	0
1	61	79	74	80	9	47	5	46	6	4	2	1	0
1	58	25	24	22	49	19	20	50	26	78	1	0	0
1	39	35	33	34	1	0	0	0	0	0	0	0	0

Κόστος Λύσης = 1277.26

Βέλτιστο Κόστος = 1135.48

Απόκλιση από το Βέλτιστο = 0.124

5.26 Πρόβλημα RC105

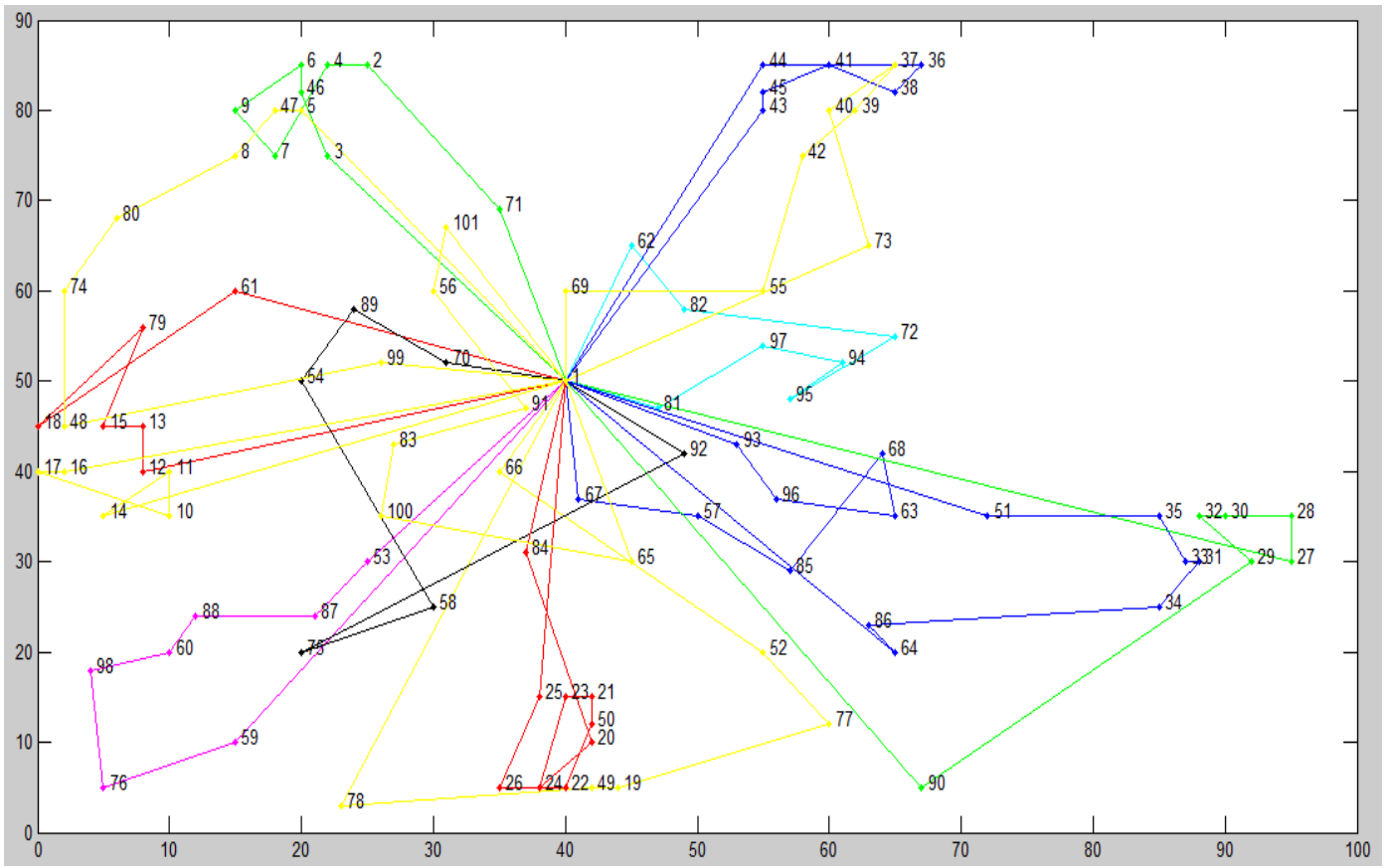
Στο πρόβλημα RC105 έχουμε n=100, Capacity=200, maximum vehicle number=25. Η λύση που προκύπτει έχει Κόστος=1695.41 με χρήση 15 οχημάτων, ενώ το βέλτιστο είναι 1629.44 με 13 οχήματα. Δηλαδή η απόκλιση από το βέλτιστο, σε αυτή την περίπτωση θα είναι 4%. Ακολουθεί η γραφική αναπαράσταση της λύσης.

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

Κόστος Λύσης = 1695.41

Βέλτιστο Κόστος = 1629.44

Απόκλιση από το Βέλτιστο = 0.040



5.27 Πρόβλημα RC106

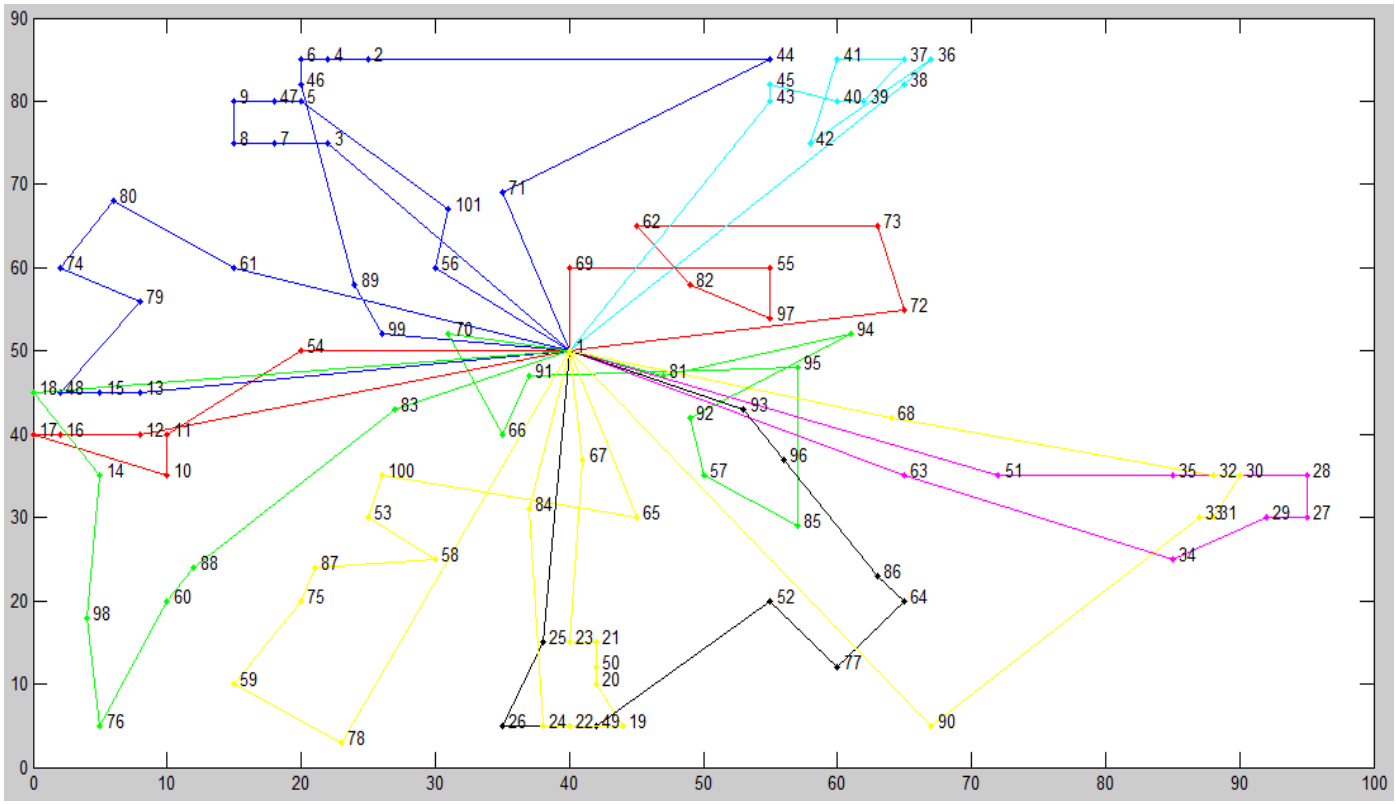
Στο πρόβλημα RC106 έχουμε n=100, Capacity=200, maximum vehicle number=25. Η λύση που προκύπτει έχει Κόστος=1534.65 με χρήση 13 οχημάτων, ενώ το βέλτιστο είναι 1424.73 με 11 οχήματα. Δηλαδή η απόκλιση από το βέλτιστο, σε αυτή την περίπτωση θα είναι 7.7%. Ακολουθεί η γραφική αναπαράσταση της λύσης.

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

Κόστος Λύσης = 1534.65

Βέλτιστο Κόστος = 1424.73

Απόκλιση από το Βέλτιστο = 0.077


0 10 20 30 40 50 60 70 80 90 100
0 10 20 30 40 50 60 70 80 90
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101

6. Σύνοψη Αποτελεσμάτων Αλγόριθμου

6.1 Πίνακας Αποτελεσμάτων Παραλλαγμένου Αλγορίθμου Πλησιέστερου Γείτονα

Στον ακόλουθο πίνακα βλέπουμε πως ο Παραλλαγμένος αλγόριθμος του Πλησιέστερου Γείτονα που αναπτύχθηκε, δημιουργεί καλύτερες αρχικές λύσεις από τον συμβατικό αλγόριθμο σε όλα σχεδόν τα προβλήματα στα οποία εφαρμόστηκε. Ενώ ο αριθμός των οχημάτων στις λύσεις που δημιουργεί είναι πάντοτε μικρότερος από τις αντίστοιχες λύσεις του συμβατικού αλγορίθμου.

Πρόβλημα	**Κόστος (Πλησιέστερου Γείτονα)**	Αριθμός οχημάτων (Πλησιέστερου Γείτονα)	**Κόστος (Νέου Αλγορίθμου)**	Αριθμός οχημάτων (Νέου Αλγορίθμου)	**Απόκλιση (Συμβατικής Λύσης από Νέα)**
C101	1779.3	21	852.9	10	-52.0%
C102	2126.3	20	1323.7	12	-37.7%
C103	2033.8	19	1779.1	12	-12.5%
C104	1625.0	13	1587.7	11	-2.3%
C105	2027.2	20	852.9	10	-57.9%
C106	1849.4	19	1151.3	11	-37.7%
C107	2257.6	19	1142.4	12	-49.4%
C108	1758.7	15	1729.3	13	-1.6%
C109	1749.7	14	1732.3	13	-0.9%
R101	2627.4	35	2292.3	29	-12.7%
R102	2343.0	33	2129.4	26	-9.1%
R103	2050.5	27	1985.5	23	-3.1%
R104	1580.5	19	1540.0	15	-2.5%
R105	2204.9	27	1932.2	21	-12.3%

R106	2245.3	29	1828.5	20	-18.5%
R107	1943.8	24	1674.6	16	-15.2%
R108	1503.5	16	1448.5	13	-3.6%
R109	1695.3	19	1774.4	18	4.6%
R110	1687.4	20	1785.2	16	5.8%
R111	1687.4	20	1785.2	16	5.8%
R112	1388.4	13	1339.8	12	-3.5%
RC101	2780.4	27	2225.5	22	-19.9%
RC102	2606.5	27	2054.1	20	-21.2%
RC103	2283.1	24	1909.8	16	-16.3%
RC104	1819.7	17	1702.2	14	-6.4%
RC105	2822.4	28	2250.1	21	-20.2%
RC106	2093.9	20	2049.1	17	-2.1%

6.2 Πίνακας Αποτελεσμάτων Tabu Search

Στον παρακάτω λοιπόν πίνακα βλέπουμε πόσο αποδοτικός είναι τελικά ο αλγόριθμος που δημιουργήσαμε.

Πρόβλημα	**Κόστος (Βέλτιστο)**	Αριθμός Οχημάτων (Βέλτιστο)	Κόστος Αρχικής Λύσης	**Κόστος Tabu Search**	Αριθμός Οχημάτων	Βελτίωση Αρχικής Λύσης	**Απόκλιση από το Βέλτιστο**
C101	**828.94**	10	852.94	**852.94**	10	0%	**2.9%**
C102	**828.94**	10	1323.72	**973.24**	10	26.4%	**17.4%**
C103	**828.06**	10	1779.13	**968.97**	10	45.5%	**17.0%**
C104	**824.78**	10	1587.72	**968.48**	10	39.0%	**16.8%**
C105	**828.94**	10	852.94	**852.94**	10	0%	**2.9%**
C106	**828.94**	10	1151.35	**909.25**	10	21.0%	**9.7%**
C107	**828.94**	10	1142.48	**857.38**	10	24.9%	**3.4%**

C108	**828.94**	10	1729.36	**966.50**	10	41.1%	**16.6%**
C109	**828.94**	10	1732.39	**969.42**	10	44.0%	**16.9%**
R101	**1645.79**	19	2292.33	**1750.59**	19	23.6%	**6.3%**
R102	**1486.12**	17	2129.41	**1622.68**	18	23.7%	**9.1%**
R103	**1292.68**	13	1985.50	**1371.73**	15	30.9	**6.0%**
R104	**982.01**	10	1540.06	**1079.19**	10	29.9%	**9.9%**
R105	**1377.11**	14	1932.25	**1477.82**	15	23.5%	**7.3%**
R106	**1252.03**	12	1828.55	**1324.93**	13	27.5%	**5.8%**
R107	**1104.66**	10	1674.67	**1176.11**	11	29.7%	**6.4%**
R108	**960.88**	9	1448.55	**1037.15**	10	28.4%	**7.9%**
R109	**1194.73**	11	1774.43	**1253.33**	12	29.3%	**6.4%**
R110	**1118.84**	10	1785.28	**1186.98**	11	33.5%	**6.0%**
R111	**1096.72**	10	1785.28	**1186.98**	11	33.5%	**8.2%**
R112	**982.14**	9	1339.84	**1060.51**	10	20.8%	**7.9%**
RC101	**1696.94**	14	2225.53	**1827.33**	17	17.8%	**7.6%**
RC102	**1554.75**	12	2054.12	**1673.18**	14	18.5%	**7.6%**
RC103	**1261.67**	11	1909.82	**1409.09**	12	26.2%	**11.6%**
RC104	**1135.48**	10	1702.26	**1277.26**	11	24.9%	**12.4%**
RC105	**1629.44**	13	2250.15	**1695.41**	15	24.6%	**4.0%**
RC106	**1424.73**	11	2049.14	**1534.65**	13	25.1%	**7.7%**

6.3 Ανάλυση Αποτελεσμάτων

Όπως βλέπουμε στον παραπάνω πίνακα τα αποτελέσματα του αλγορίθμου στα περισσότερα από τα προβλήματα στα οποία δοκιμάστηκε ήταν πολύ ικανοποιητικά. Στα περισσότερα προβλήματα η απόκλιση από το βέλτιστο δεν ξεπερνά το 8% ενώ σε ορισμένες περιπτώσεις η απόκλιση άγγιξε το 2.8%. Ο αριθμός τον επαναλήψεων που εκτελέστηκαν ώστε να έχουμε τα παραπάνω αποτελέσματα ήταν σχετικά μικρός, εφόσον ο Αλγόριθμος συγκλίνει με ταχύ ρυθμό προς το βέλτιστο. Σε ορισμένα προβλήματα μάλιστα χρησιμοποιήθηκαν λιγότερες από 5000. Είναι σχεδόν βέβαιο πως με περισσότερες επαναλήψεις θα είχαμε και καλύτερα αποτελέσματα.

Απ ’ότι λοιπόν είδαμε, ο Παραλλαγμένος Αλγόριθμος του Πλησιέστερου Γείτονα, σε συνεργασία με την Μεθευρετικό Αλγόριθμο Περιορισμένης Αναζήτησης και τον

Αλγόριθμο Ελαχιστοποίησης του Αριθμού των Οχημάτων έδωσαν ένα άρτιο αποτέλεσμα. Ο κώδικας που δημιουργήθηκε θα μπορούσε να έχει εφαρμογή σε προβλήματα όπου ο χρόνος αποτελεί μία σημαντική παράμετρο. Επειδή λοιπόν ο χρονικός προγραμματισμός και ακρίβεια στις μεταφορικές διαδικασίες είναι υψίστης σημασίας για τις περισσότερες σύγχρονες επιχειρήσεις, οι εφαρμογές στις οποίες θα μπορούσε να χρησιμοποιηθεί ο συγκεκριμένος αλγόριθμος είναι πολλές.

Ο κώδικας που αναπτύχθηκε στα πλαίσια της συγκεκριμένης Διπλωματικής Εργασίας έχει αρκετές δυνατότητες περαιτέρω εξέλιξης. Για παράδειγμα, στον Αλγόριθμο που αναλύθηκε στις παραπάνω σελίδες, παρατηρούμε πως ο αριθμός των κόμβων που εξυπηρετεί κάθε δρομολόγιο στην Αρχική Λύση παραμένει αμετάβλητος καθ' όλη την διάρκεια της βελτιστοποίησης. Εάν δηλαδή ένα δρομολόγιο δεν καταργηθεί κατά την περιορισμένη αναζήτηση, θα εξυπηρετεί υποχρεωτικά τον ίδιο αριθμό κόμβων σε όλα τα στάδια του αλγορίθμου. Το γεγονός αυτό λοιπόν θα μπορούσε να αλλάξει ώστε να υπάρχει δυνατότητα αφαίρεσης ενός κόμβου από ένα δρομολόγιο χωρίς την αντικατάσταση της θέσης αυτού από κάποιο άλλο κόμβο ενός άλλου δρομολογίου. Μια τέτοια εξέλιξη θα έδινε πολλά πλεονεκτήματα στον αλγόριθμο και είναι βέβαιο πως θα παρατηρούνταν ακόμη καλύτερες λύσεις.

Βιβλιογραφία

[1] Ιωάννης Μαρινάκης, Αθανάσιος Μυγδαλάς. Σχεδιασμός και Βελτιστοποίηση της Εφοδιαστικής Αλυσίδας. Εκδόσεις 'σοφία', Θεσσαλονίκη 2008.

[2] H.R.G Van Landeghem. ''A bi-criteria heuristic for the vehicle routing problem with time windows''. *European Journal of Operational Research 36 (1988) 217-226 North Holland*

[3] Jean Berger, Martin Salois and Regent Begin. ''A hybrid genetic algorithm for the

vehicle routing problem with time windows''. *Defence research Establishment Valcartier, Decision Support Technology Section, Canada.*

[4] Bruno-Laurent Garcia, Jean Yves Potvin and Jean-Mark Rousseau. ''A parallel implementation of the Tabu search heuristic for vehicle routing problems with time windows''. Computers Ops Res. Vol 21 No. 9,Elsevier Science Ltd, *University of Montreal, Canada* September 1994.

[5] Jean Yves Potvin and Jean-Mark Rousseau. ''A parallel route building algorithm for the vehicle routing and scheduling problem with time wind''. *University of Montreal, Canada* May 1991, European Journal of Operational Research 66.

[6] Philippe Badeau, Francois Guertin, Michel Gendreau, Jean Yves Potvin and Eric Taillard. ''A parallel tabu search heuristic for the vehicle routing problem with time windows ''. *Elsevier Science 1997.*

[7] *George Ioannou, Manolis Kritikos, Gregory Prastacos. ''A problem generator-solver heuristic for vehicle routing with soft time windows''. Athens University of Economics and Business, October 2002, The International Journal of Management Science.*

[8] Antoine Landrieu, Yazid Mati and Zdenek Binder. ''A tabu search heuristic for the single vehicle pickup and delivery problem with time windows''*. 2001 Kluwer Academic Publishers, Manufactured in The Netherlands, Journal of Intelligent Manufacturing.*

[9] S.C. Ho, D. Haugland. ‘‘A tabu search heuristic for the vehicle routing problem with time windows and split deliveries’’. *Department of Informatics, University of Bergen, Norway 2004 , Computers and Operations Research 31.*

[10] Jorg Homberger, Hermann Gehring, ‘‘A two-phase hybrid metaheuristic for the vehicle routing problem with time windows’’. *University of Hangen, Germany , January 2004,* European Journal of Operational Research 162*.*

[11] David Mester, Olli Rraysy. ‘‘Active guided evolution strategies for large-scale vehicle routing problems with time windows’’. *Computers and Operations Research 2005, Computers and Operations Research 32.*

[12] Jean Charles Creput, Abder Koukam, Jaroslaw Kozlak and Jan Lukasik. ‘‘An Evolutionary Approach to Pickup and Delivery Problem with Time Windows’’. *University of Technology of Belfort,France, AGH University of Science and Technology, Poland.*

[13] Hideki Hashimoto, Mitsunori Yagiura, Toshihide Ibaraki. ‘‘An iterated local search algorithm for the time-dependent vehicle routing problem with time windows’’. *Kyoto University, Japan, May 2007, Discrete Optimization 5.*

[14] Fuh-hwa Franklin Liu and Sheng-yuan Shen, ‘‘An overview of a heuristic for vehicle routing problem with time windows’’. *National Chiao Tung University, Hsinchu, Taiwan, Republic of Chin, Computers and Industrial Engineering 37 (1999) .*