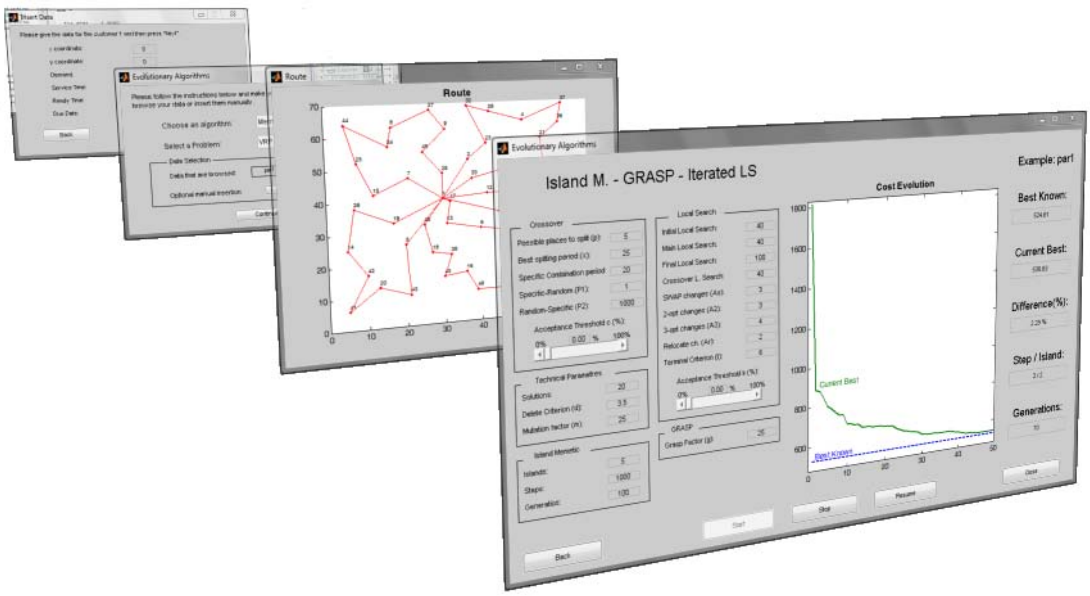
ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΑΡΑΓΩΓΗΣ ΚΑΙ ΔΙΟΙΚΗΣΗΣ

Μεταπτυχιακή Διατριβή

Ρογδάκης Ιωάννης

"Ένα Σύστημα Υποστήριξης Αποφάσεων για την επίλυση προβλημάτων Εφοδιαστικής Αλυσίδας με την χρήση Εξελικτικών Αλγορίθμων "



Επιβλέπων: Μαρινάκης Ιωάννης

ΧΑΝΙΑ 2010

Αφιερώνεται στη μνήμη της γιαγιάς μου

Ασπασίας Ρογδάκη

ΠΕΡΙΕΧΟΜΕΝΑ

Περίληψη

Ο αλγόριθμοι που κατασκευάστηκαν στα πλαίσια αυτής της διατριβής είναι μία καινοτομία στο χώρο της συνδυαστικής βελτιστοποίησης. Ανήκουν στις μεθευρετικές μεθόδους επίλυσης προβλημάτων συνδυαστικής βελτιστοποίησης. Βέβαια, εσωτερικά συνδυάζουν αρμονικά ευρετικές και μεθευρετικές μεθόδους. Με τις μεθευρετικές προσεγγίζονται τα διάφορα τοπικά βέλτιστα και με τις ευρετικές μεθόδους εντοπίζονται με ακρίβεια. Συνήθως οι καθαρές ευρετικές μέθοδοι εγκλωβίζονται σε τοπικά ελάχιστα, γι' αυτό η ποιότητα των λύσεων που επιτυγχάνονται με τις μεθευρετικές μεθόδους είναι πολύ καλύτερη από την ποιότητα που δίνουν οι απλές ευρετικές μέθοδοι.

1. ΕΙΣΑΓΩΓΗ

Στα πλαίσια της παρούσας μεταπτυχιακής διατριβής κατασκευάστηκε ένα Σύστημα Υποστήριξης Αποφάσεων για την επιτυχή επίλυση 5 τύπων προβλημάτων εφοδιαστικής αλυσίδας (TSP, VRP, Open VRP, Time Windows VRP, Multiple Depot VRP) με 10 από τις αποτελεσματικότερες μεθόδους που μπορούν να χρησιμοποιηθούν σήμερα:

1) Μιμητικός Αλγόριθμος (Memetic Algorithm)
2) Μιμητικός Αλγόριθμος με χρήση Άπληστης Τυχοποιημένης Προσαρμοστικής Αναζήτησης (Memetic with GRASP Algorithm)
3) Μιμητικός Αλγόριθμος Πολλαπλών Πληθυσμών-Νησιών (Island Memetic Algorithm)
4) Μιμητικός Αλγόριθμος Πολλαπλών Πληθυσμών-Νησιών με χρήση Άπληστης Τυχοποιημένης Προσαρμοστικής Αναζήτησης (Island Memetic with GRASP Algorithm)
5) Αλγόριθμος Άπληστης Τυχοποιημένης Προσαρμοστικής Αναζήτησης (Greedy Randomized Adaptive Search Procedure)
6) Επαναληπτική Τοπική Αναζήτηση (Iterated Local Search)
7) Μιμητικός Αλγόριθμος με χρήση Επαναληπτικής Τοπικής Αναζήτησης (Memetic Algorithm with Iterated Local Search)
8) Άπληστη Τυχοποιημένη Προσαρμοστική Αναζήτηση στο Μιμητικό Αλγόριθμο με χρήση Επαναληπτικής Τοπικής Αναζήτησης (GRASP in Memetic Algorithm and Iterated Local Search)
9) Μιμητικός Αλγόριθμος Πολλαπλών Πληθυσμών-Νησιών με χρήση Επαναληπτικής Τοπικής Αναζήτησης (Island Memetic Algorithm with Iterated Local Search)
10) Επαναληπτική Τοπική Αναζήτηση στο Μιμητικό Αλγόριθμο Πολλαπλών Πληθυσμών-Νησιών με χρήση Άπληστης Τυχοποιημένης Προσαρμοστικής Αναζήτησης (Iterated Local Search in Island Memetic with GRASP)

Το Σύστημα Υποστήριξης Αποφάσεων είναι φιλικό προς το χρήστη. Τον ωθεί να εισάγει τα σωστά δεδομένα με το σωστό τρόπο, δίνοντάς του οδηγίες για κάθε πιθανή του κίνηση. Επίσης κατά την κατασκευή του συγκεκριμένου συστήματος προβλέφθηκαν όλα τα πιθανά λάθη που μπορεί να κάνει κάποιος χρήστης. Οποιαδήποτε περίπτωση λάθους κίνησης αποτρέπεται αυτόματα από το σύστημα και αυτόματα δίνονται οδηγίες στο χρήστη για το πώς θα δράσει σωστά. Στο συγκεκριμένο σύστημα υπάρχει η δυνατότητα να φορτωθούν τα προς επεξεργασία δεδομένα από διάφορους

τύπους αρχείων είτε να εισαχθούν χειροκίνητα (από το χρήστη) όλα τα δεδομένα για οποιοδήποτε πραγματικό πρόβλημα και στη συνέχεια να πραγματοποιηθεί η βελτιστοποίηση με οποιονδήποτε από τους διαθέσιμους αλγορίθμους.

Εφαρμόσαμε το σύστημα και τους αλγορίθμους μας πάνω σε παγκοσμίως γνωστά προβλήματα (instances) τα οποία χρησιμοποιούνται για τη σύγκριση αλγορίθμων προβλημάτων εφοδιαστικής αλυσίδας σε παγκόσμια κλίμακα. Σε αρκετά από τα instances ισοφαρίστηκε το κόστος της καλύτερης λύσης που έχει βρεθεί σε παγκόσμια κλίμακα (συγκεκριμένα για το πρόβλημα VRP στα vrpnc1, vrpnc6, vrpnc11 και vrpnc12). Στα υπόλοιπα instances (του VRP) κρατήσαμε το κόστος σε πάρα πολύ χαμηλά επίπεδα, ο μέσος όρος απόκλισης από το παγκόσμιο βέλτιστο ήταν μόλις 1,3%.

2. Τα προβλήματα που επιλύονται από το Σ.Υ.Α.

2.1 Το πρόβλημα του περιπλανώμενου πωλητή (TSP)

Σε προβλήματα τύπου TSP υπάρχει ένα όχημα το οποίο θα πρέπει να εξυπηρετήσει όλη τη ζήτηση του συνόλου των πελατών. Το συγκεκριμένο όχημα ξεκινάει από και καταλήγει στη μοναδική αποθήκη που υπάρχει. Επίσης, κάθε όχημα επισκέπτεται μονάχα μία φορά κάθε πελάτη. Οι περιορισμοί του προβλήματος εκτός από τα λειτουργικά σημεία του προβλήματος εξασφαλίζουν ότι ικανοποιείται πλήρως η ζήτηση των πελατών. Στόχος είναι η ελαχιστοποίηση του συνολικού μεταφορικού κόστους με την προϋπόθεση ότι πληρούνται όλοι οι περιορισμοί.

Οι περιορισμοί έχουν να κάνουν με την χωρητικότητα του οχήματος και με τα ιδιαίτερα χαρακτηριστικά του δικτύου διανομής. Το όχημα έχει συγκεκριμένη χωρητικότητα, όσον αφορά στο βάρος, ή τον όγκο, ή αριθμό παλετών ή προϊόντων που μπορεί να μεταφέρει. Το κόστος χρήσης του οχήματος είναι ευθέως ανάλογο με την συνολική απόσταση που διανύει.

Τα δεδομένα που αντιστοιχούν σε κάθε έναν πελάτη έχουν να κάνουν με τις συντεταγμένες που βρίσκεται ο κάθε πελάτης και τη ζήτησή του. Το συνολικό κόστος προκύπτει από τα μεταφορικά κόστη από πελάτη σε πελάτη και από τα κόστη για τις 2 μεταφορές από και προς την αποθήκη. Το συνολικό κόστος για τη μετάβαση από έναν κόμβο α σε έναν κόμβο β, (έστω $c_{\alpha\beta}$) ανήκει στο μέρος της συντομότερης διαδρομής, το οποίο ξεκινάει από τον κόμβο α και καταλήγει στον β. Αναλόγως ο χρόνος μετάβασης $\tau_{\alpha\beta}$, προκύπτει από το άθροισμα των χρόνων μετάβασης όλων των επιμέρους μεταβάσεων (από πελάτη σε πελάτη).

Παρακάτω φαίνεται η μοντελοποίηση που ακολουθήθηκε για το συγκεκριμένο πρόβλημα. Έστω Χ η χωρητικότητα του φορτηγού.

$$\min \sum_{\alpha=1}^{M} \sum_{\beta=1}^{M} \kappa_{\alpha\beta}\, \delta_{\alpha\beta} \qquad (1)$$

Υπό:

Όλες οι μεταβλητές απόφασης είναι δυαδικές (0 ή 1):

$$\delta_{\alpha\beta} \in \{0,1\} \qquad \forall\, \alpha \in 1,\dots,M \quad \kappa\alpha\iota \quad \forall\, \beta \in 1,\dots,M \tag{2}$$

Από την αποθήκη αναχωρεί μονάχα 1 όχημα

$$\sum_{\alpha=1}^{M} \delta_{1\alpha} = 1 \tag{3}$$

Περιορισμός διαδρομών από και προς πελάτη και την αποθήκη:

$$\sum_{\alpha,\beta \in N} \delta_{\alpha\beta} = |N| - 1 \qquad \forall\, N \subseteq \{2,\dots,M\} \tag{4}$$

Κάθε πελάτης εξυπηρετείται μονάχα μία φορά:

$$\sum_{\alpha=1}^{M} \delta_{\alpha\beta} = 1 \qquad \forall\, \alpha \in 2,\dots,M \quad \kappa\alpha\iota \quad \forall\, \beta \in 2,\dots,M \tag{5}$$

Εξασφαλίζεται ότι το ίδιο όχημα που εισέρχεται σε έναν συγκεκριμένο κόμβο, στην συνέχεια εξέρχεται από τον ίδιο κόμβο:

$$\sum_{\beta=1}^{M} \delta_{\alpha\beta} - \sum_{\beta=1}^{M} \delta_{\beta\alpha} = 0 \quad \gamma\iota\alpha\ \kappa\acute{\alpha}\theta\varepsilon\ \alpha \in 1,\dots,M \tag{6}$$

όπου:

$$\delta_{\alpha\beta} = \begin{cases} 1, & Αν\ το\ όχημα\ κατευθύνεται\ στον\ κόμβο\ \beta\ αμέσως\ μετά\ τον\ κόμβο\ \alpha \\ 0, & αλλιώς \end{cases}$$

2.2 Το Πρόβλημα Δρομολόγησης Οχημάτων (VRP)

Σε ένα πρόβλημα τύπου V.R.P. (Vehicle Routing Problem) αναζητούνται οι διαδρομές που πρέπει να ακολουθήσει κάθε ένα από τα διαθέσιμα οχήματα για να εξυπηρετηθούν όλοι οι πελάτες (ζήτηση). Κάθε όχημα ξεκινάει από την αποθήκη και καταλήγει πάλι σε αυτήν. Συνεπώς οι περιορισμοί του προβλήματος αφορούν την ικανοποίηση της ζήτησης όλων των πελατών. Τέλος, επιδιώκεται η ελαχιστοποίηση του μεταφορικού κόστους μεταφοράς των προϊόντων.

Κάθε όχημα επισκέπτεται μονάχα μία φορά κάθε πελάτη. Όπως επίσης και κάθε διαδρομή (που ξεκινάει από την αποθήκη και καταλήγει σε αυτήν) εκτελείται από ένα μόνο όχημα. Οι περιορισμοί περιγράφουν τις αποστάσεις μεταξύ των πελατών, τη συνολική διανυόμενη απόσταση για κάθε ένα όχημα, την χωρητικότητα των οχημάτων καθώς επίσης οτιδήποτε περιγράφει τα χαρακτηριστικά του οδικού δικτύου διανομής. Δεν υπάρχει περιορισμός για τον αριθμό των πελατών που πρέπει να επισκεφτεί κάθε όχημα, εκτός του ότι πρέπει να επισκεφτεί τουλάχιστον έναν πελάτη.

Η συνολική ζήτηση των πελατών αλλά και η ζήτηση του κάθε πελάτη καθορίζουν τον αριθμό των οχημάτων που θα χρειαστούν. Κάθε όχημα έχει συγκεκριμένη χωρητικότητα η οποία εκφράζεται σαν ένα άνω όριο σχετικά με το βάρος, ή τον όγκο, ή αριθμό παλετών ή προϊόντων που μπορεί να μεταφέρει το κάθε όχημα. Το κόστος που συνδέεται με την χρήση του εκάστοτε οχήματος, στην μοντελοποίηση που ακολουθήθηκε στην παρούσα μεταπτυχιακή διατριβή, είναι ευθέως ανάλογο με την απόσταση που διανύει το κάθε όχημα.

Τα δεδομένα για τον κάθε πελάτη αφορούν τις συντεταγμένες θέσης του και τη ζήτησή του η οποία πρέπει να ικανοποιηθεί εξ' ολοκλήρου. Σε κάθε διαδρομή πρέπει να ικανοποιούνται λειτουργικοί περιορισμοί που έχουν να

κάνουν με την ποσότητα των μεταφερόμενων προϊόντων, και τα χαρακτηριστικά των οχημάτων. Για παράδειγμα, σε κάθε διαδρομή η μεταφερόμενη ποσότητα δεν θα πρέπει να ξεπερνά την χωρητικότητα του οχήματος. Οι οδηγοί των οχημάτων είναι απαραίτητο να ικανοποιούν διάφορους περιορισμούς που έχουν να κάνουν με συμβόλαια εργατικών ενώσεων καθώς και κανονισμούς της εταιρείας για την οποία εργάζονται. Το σύνολο των περιορισμών που αναφέρεται στα οχήματα μπορεί να περικλείει και περιορισμούς που προκύπτουν από τους οδηγούς. Ο χρόνος εργασίας κατά τη διάρκεια της ημέρας, ο αριθμός και η διάρκεια των διαλειμμάτων και οι υπερωρίες αποτελούν ορισμένους μόνο από τους "κανόνες" τους οποίους πρέπει να ληφθούν υπόψη κατά τη διαδικασία μοντελοποίησης για τον καθορισμό των περιορισμών.

Για τον έλεγχο των περιορισμών και τον υπολογισμό του κόστους του κάθε δρομολογίου, αθροίζονται τα κόστη μεταφοράς από τον ένα πελάτη στον άλλον καθώς και τα κόστη μεταφοράς από και προς την αποθήκη. Το κόστος μετάβασης από έναν κόμβο α σε έναν κόμβο β, είναι το $c_{\alpha\beta}$ το οποίο ανήκει στη συντομότερη διαδρομή μεταξύ των κόμβων α και β. Ταυτόχρονα και ο χρόνος μετάβασης $\tau_{\alpha\beta}$, ισούται με το άθροισμα των χρόνων μετάβασης όλων των επιμέρους μεταβάσεων (από πελάτη σε πελάτη και από και προς την αποθήκη) που ανήκουν στην συντομότερη διαδρομή μεταξύ των πελατών α και β.

Η αντικειμενική συνάρτηση αφορά στην ελαχιστοποίηση του συνολικού κόστους μεταφοράς, το οποίο εξαρτάται από τον συνολικό χρόνο μεταφοράς ή την συνολική απόσταση που διανύεται από το σύνολο των οχημάτων. Ο πρώτος περιορισμός αφορά στην χωρητικότητα του κάθε οχήματος, όπου σε κάθε δρομολόγιο ο συνολική ποσότητα των αγαθών όπου φορτώνονται στο κάθε φορτηγό δεν πρέπει να υπερβαίνουν την χωρητικότητα του συγκεκριμένου οχήματος. Επίσης υπάρχει και ο χρονικός περιορισμός, σύμφωνα με τον οποίο ο συνολικός χρόνος που απαιτεί μία ολοκληρωμένη διαδρομή δεν πρέπει να υπερβαίνει ένα συγκεκριμένο όριο.

Για το πρόβλημα δρομολόγησης οχημάτων ακολουθείται η μοντελοποίηση που περιγράφεται παρακάτω. Έστω ότι οι πελάτες και η αποθήκη αντιστοιχούν σε κάποιους κόμβους με συγκεκριμένες συντεταγμένες στο χώρο και μαζί με τα τόξα που ενώνουν τους κόμβους των πελατών και της αποθήκης αποτελούν ένα γράφημα A(K,T). Οι κόμβοι K=(2,...,M) αντιπροσωπεύουν τους πελάτες και την αποθήκη αντιστοιχεί στον κόμβο 1. Έστω ότι ο αριθμός των οχημάτων είναι ν όπου κάθε όχημα έχει χωρητικότητα Χ. Ο πίνακας με τα κόστη (κ) έχει διαστάσεις MxM και $\kappa_{\alpha\beta}$ είναι το κόστος μεταφοράς από τον κόμβο α στον κόμβο β. Η ζήτηση του

κόμβου α είναι ζ_α και φυσικά η ζήτηση του κόμβου 1 (δηλαδή της αποθήκης) είναι ζ_1=0. Στόχος είναι η εύρεση του βέλτιστου δ, των διαδρομών λοιπόν εκείνων που ελαχιστοποιούν το συνολικό κόστος. Άρα, η αντικειμενική συνάρτηση θα έχει ως εξής:

$$\min \sum_{\alpha=1}^{M} \sum_{\beta=1}^{M} \sum_{\gamma=1}^{\nu} \kappa_{\alpha\beta}\, \delta_{\alpha\beta\gamma} \qquad (7)$$

Οι περιορισμοί θα έχουν ως εξής:

Όλες οι μεταβλητές απόφασης είναι δυαδικές (0 ή 1):

$$\delta_{\alpha\beta\gamma} \in \{0,1\} \qquad \forall\, \gamma \in 1, \dots, \nu \quad \kappa\alpha\iota \quad \forall\, \alpha \in 1, \dots, M \quad \kappa\alpha\iota \quad \forall\, \beta \in 1, \dots, M \qquad (8)$$

Από την αποθήκη αναχωρούν ν οχήματα:

$$\sum_{\beta=1}^{M} \sum_{\gamma=1}^{\nu} \delta_{1\beta\gamma} = \nu \qquad (9)$$

Όλα τα οχήματα έχουν χωρητικότητα ίση με Χ:

$$\sum_{\alpha=1}^{M} \sum_{\beta=1}^{M} \delta_{\alpha\beta\gamma}\, \zeta_\alpha \le X \qquad \forall\, \gamma \in 1, \dots, \nu \qquad (10)$$

Περιορισμός διαδρομών:

$$\sum_{\alpha,\beta \in N} \delta_{\alpha\beta\gamma} \leq |N| - 1 \qquad \forall\, \gamma \in 1, \dots, \nu \;\; \kappa\alpha\iota$$

$$\forall\, N \subseteq \{2, \dots, M\} \qquad (11)$$

Κάθε πελάτης εξυπηρετείται μονάχα μία φορά

$$\sum_{\alpha=1}^{M} \sum_{\gamma=1}^{\nu} \delta_{\alpha\beta\gamma} = 1 \qquad \forall\, \alpha \in 2, \dots, M \qquad \kappa\alpha\iota$$

$$\forall\, \beta \in 2, \dots, M \qquad \kappa\alpha\iota$$

$$\forall\, \gamma \in 1, \dots, \nu \qquad (12)$$

Εξασφαλίζεται ότι το ίδιο όχημα που εισέρχεται σε έναν συγκεκριμένο κόμβο, στην συνέχεια εξέρχεται από τον ίδιο κόμβο:

$$\sum_{\beta=1}^{M} \delta_{\alpha\beta\gamma} - \sum_{\beta=1}^{M} \delta_{\beta\alpha\gamma} = 0 \quad \forall\, \alpha \in 1, \dots, M \quad \kappa\alpha\iota$$

$$\forall\, \gamma \in 1, \dots, \nu \qquad (13)$$

όπου:

$$\delta_{\alpha\beta\gamma} = \begin{cases} 1, & Αν\ το\ όχημα\ \gamma\ πάει\ στον\ κόμβο\ \beta\ αμέσως\ μετά\ τον\ κόμβο\ \alpha \\ 0, & αλλιώς \end{cases}$$

2.3 Το Ανοικτό Πρόβλημα Δρομολόγησης Οχημάτων (OVRP)

Σε προβλήματα τύπου OVRP (Open Vehicle Routing Problem) ακολουθείται η ίδια μοντελοποίηση με τα προβλήματα VRP με τη διαφορά ότι το όχημα που χρησιμοποιείται σε κάθε δρομολόγιο δεν απαιτείται να επιστρέψει πίσω στην αποθήκη. Κατά τα υπόλοιπα οι ισχύουν οι ίδιοι περιορισμοί που ισχύουν και στο απλό πρόβλημα VRP.

Και εδώ στόχος είναι η ελαχιστοποίηση του συνολικού κόστους μεταφορών, πράγμα που εξασφαλίζεται από την αντικειμενική συνάρτηση. Το συνολικό κόστος κι εδώ εξαρτάται από τον συνολικό χρόνο μεταφοράς ή την συνολική απόσταση που διανύεται από το σύνολο των οχημάτων.

Δεν θα πρέπει σε καμία περίπτωση να υπερβαίνεται η μέγιστη χωρητικότητα του κάθε οχήματος, αλλά ούτε και ο συνολικός χρόνος που απαιτείται για ένα δρομολόγιο δεν θα πρέπει να υπερβαίνει το όριο που θέτουν οι χρονικοί περιορισμοί.

Έστω λοιπόν ένα γράφημα A(K,T) όπου K είναι το σύνολο των κόμβων (πελάτες και αποθήκη) και T είναι το σύνολο των τόξων (διαδρομών) μεταξύ των κόμβων. Οι κόμβοι K=(2,...,M) πάλι αντιστοιχούν στους πελάτες και την αποθήκη αντιστοιχεί στον κόμβο 1. Ο πίνακας με τα κόστη (κ) έχει διαστάσεις MxM και κ_{ij} είναι το κόστος μεταφοράς από τον κόμβο i στον κόμβο j. Η ζήτηση του κόμβου i είναι ζ_i και h ζήτηση του κόμβου 1 είναι 0. Έστω ότι ο αριθμός των οχημάτων είναι v όπου κάθε όχημα έχει χωρητικότητα X.

Αντικειμενική Συνάρτηση: Εύρεση του βέλτιστου δ για την ελαχιστοποίηση του συνολικού κόστους:

$$\min \sum_{\alpha=1}^{M} \sum_{\beta=1}^{M} \sum_{\gamma=1}^{v} \kappa_{\alpha\beta}\, \delta_{\alpha\beta\gamma} \qquad (14)$$

Οι περιορισμοί θα έχουν ως εξής:

Το όχημα που εξέρχεται της αποθήκης δεν εισέρχεται ξανά σε αυτήν, στο τέλος του δρομολογίου.

$$\sum_{\beta=1}^{M} \delta_{1\beta\gamma} - \sum_{\beta=1}^{M} \delta_{\beta 1\gamma} = 1 \quad \forall\ \gamma \in 1, \dots, \nu \qquad (15)$$

Περιορισμός εισαγωγής – εξαγωγής οχημάτων για τους κόμβους των πελατών:

$$\sum_{\beta=1}^{M} \delta_{\alpha\beta\gamma} - \sum_{\beta=1}^{M} \delta_{\beta\alpha\gamma} \leq 0 \quad \forall\ \alpha \in 2, \dots, M \quad \kappa\alpha\iota \quad \forall\ \gamma \in 1, \dots, \nu \qquad (16)$$

Όλες οι μεταβλητές απόφασης είναι δυαδικές (0 ή 1):

$$\delta_{\alpha\beta\gamma} \in \{0,1\} \qquad \forall\ \gamma \in 1, \dots, \nu \quad \kappa\alpha\iota \quad \forall\ \alpha \in 1, \dots, M \quad \kappa\alpha\iota \quad \forall\ \beta \in 1, \dots, M \qquad (17)$$

Από την αποθήκη αναχωρούν ν οχήματα:

$$\sum_{\beta=1}^{M} \sum_{\gamma=1}^{\nu} \delta_{1\beta\gamma} = \nu \qquad (18)$$

Όλα τα οχήματα έχουν χωρητικότητα ίση με X:

$$\sum_{\alpha=1}^{M}\sum_{\beta=1}^{M}\delta_{\alpha\beta\gamma}\,\zeta_{\alpha} \le X \qquad \forall\,\gamma \in 1,\dots,\nu \tag{19}$$

Περιορισμός διαδρομών:

$$\sum_{\alpha,\beta \in N}\delta_{\alpha\beta\gamma} \le |N| - 2 \qquad \forall\,\gamma \in 1,\dots,\nu \;\; \kappa\alpha\iota$$

$$\forall\, N \subseteq \{2,\dots,M\} \tag{20}$$

Κάθε κόμβος πελάτη εξυπηρετείται μονάχα μία φορά

$$\sum_{\alpha=1}^{M}\sum_{\gamma=1}^{\nu}\delta_{\alpha\beta\gamma} = 1 \qquad \forall\,\alpha \in 2,\dots,M \quad \kappa\alpha\iota$$

$$\forall\,\beta \in 2,\dots,M \quad \kappa\alpha\iota$$

$$\forall\,\gamma \in 1,\dots,\nu \tag{21}$$

όπου:

$$\delta_{\alpha\beta\gamma} = \begin{cases} 1, & Αν\ το\ όχημα\ \gamma\ πάει\ στον\ κόμβο\ \beta\ αμέσως\ μετά\ τον\ κόμβο\ \alpha \\ 0, & αλλιώς \end{cases}$$

2.4 Το πρόβλημα VRP με Χρονικά Παράθυρα (VRPTW)

Στον τύπο προβλημάτων VRPTW (Vehicle Routing Problem with Time Windows), η εξυπηρέτηση των πελατών πρέπει να γίνει μέσα σε ορισμένα χρονικά πλαίσια, μέσα σε χρονικά παράθυρα που ορίζονται από τον ενωρίτερο χρόνο και τον βραδύτερο χρόνο. Κι εδώ το σύνολο των κόμβων αντιπροσωπεύουν τους πελάτες και την αποθήκη (στην μοντελοποίηση που έχει ακολουθηθεί σε αυτή την εργασία, η αποθήκη αναπαριστάται από τον κόμβο 1). Υπάρχουν δύο κύριοι τύποι προβλημάτων VRPTW. στην πρώτη περίπτωση υπάρχουν 'χαλαρά' χρονικά παράθυρα όπου μπορεί να παραβιαστεί το όριο του ενωρίτερου χρόνου και να ξεκινήσει η εξυπηρέτηση αμέσως μόλις φθάσει το όχημα στον πελάτη. Στην δεύτερη περίπτωση δεν μπορεί να παραβιαστεί κανένα από τα χρονικά παράθυρα και σε περίπτωση που το όχημα φθάσει νωρίτερα στον πελάτη, θα περιμένει μέχρι τον ενωρίτερο χρόνο του χρονικού παράθυρου του συγκεκριμένου πελάτη.

Έστω ότι ο αριθμός των κόμβων είναι Μ και ο αριθμός των οχημάτων έστω ότι είναι ν. Ο πίνακας που περιέχει τα κόστη μεταφορών (*κ*) είναι διαστάσεων α x α, όπου $\kappa_{\alpha\beta}$ είναι το κόστος μεταφοράς από τον κόμβο *α* στον κόμβο *β*. Η ζήτηση του κόμβου α είναι ζ_{α} (όπου σύμφωνα με τη μοντελοποίηση αυτής της εργασίας ισχύει ζ_1=0, αφού με τον αριθμό 1 αναπαρίσταται η αποθήκη). Κάθε όχημα έχει χωρητικότητα Χ. Το πρόβλημα που ζητείται να επιλυθεί είναι ένα πρόβλημα ακέραιου γραμμικού προγραμματισμού. Σκοπός της επίλυσης ενός προβλήματος VRPTW είναι η ελαχιστοποίηση της εξής αντικειμενικής συνάρτησης που φαίνεται παρακάτω, αξίζει να σημειωθεί ότι στη παρούσα εργασία έγινε επίλυση του προβλήματος VRPTW με χαλαρά χρονικά παράθυρα:

$$\min \sum_{\theta=1}^{M} \sum_{\lambda=1}^{M} \sum_{\rho=1}^{\nu} \kappa_{\theta\lambda}\, \delta_{\theta\lambda\rho} \qquad (22)$$

Επίσης πρέπει να ικανοποιούνται οι εξής περιορισμοί:

$$\delta_{\theta\lambda\rho} \in \{0,1\} \qquad \forall\, \rho \in 1,\dots,\nu \quad \kappa\alpha\iota \quad \forall\, \theta \in 1,\dots,M \quad \kappa\alpha\iota \quad \forall\, \lambda \in 1,\dots,M \tag{23}$$

όπου:

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

$$n_a \le t_{a,\rho} \le v_a \qquad \forall\, \alpha \in 1,\dots,M \quad \kappa\alpha\iota \quad \forall\, \rho \in 1,\dots,\nu \tag{24}$$

$$\sum_{\theta=1}^{M}\sum_{\lambda=1}^{\nu} \delta_{\theta\lambda\rho} = 1 \qquad \forall\, \lambda \in 2,\dots,M \tag{25}$$

$$\sum_{\lambda=1}^{M} \delta_{\theta\lambda\rho} - \sum_{\lambda=1}^{M} \delta_{\theta\lambda\rho} = 0 \quad \forall\, \theta \in 1,\dots,M \quad \kappa\alpha\iota \quad \forall\, \rho \in 1,\dots,\nu \tag{26}$$

$$\sum_{\theta=1}^{M}\sum_{\lambda=1}^{M} \delta_{\theta\lambda\rho}\, \zeta_\theta \le X \qquad \forall\, \rho \in 1,\dots,\nu \tag{27}$$

$$\sum_{\lambda=1}^{M}\sum_{\rho=1}^{\nu}\delta_{1\lambda\rho} = \nu \qquad (28)$$

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

$$\forall\, SM \subseteq \{2,\dots,M\} \qquad (29)$$

Επεξήγηση περιορισμών:

Ο περιορισμός 24 εξασφαλίζει για κάθε όχημα m, η εξυπηρέτηση σε κάθε πελάτη α αρχίζει μέσα στα χρονικά παράθυρα που έχουν οριστεί. Ο 23ος περιορισμός εξασφαλίζει ότι όλες οι μεταβλητές απόφασης είναι δυαδικές (συγκεκριμένα με πιθανές τιμές 0,1). Ο 25ος και 26ος, δείχνουν ότι ο κάθε κόμβος επισκέπτεται μία μόνο φορά και ότι το όχημα που εισέρχεται σε έναν κόμβο είναι το ίδιο με εκείνο που εξέρχεται από αυτόν. Ο περιορισμός 27 εξασφαλίζει ότι τα οχήματα έχουν χωρητικότητα ίση με Q. Ακόμα, ο 28ος περιορισμός δείχνει ότι από την αποθήκη φεύγουν g οχήματα. Τέλος ο 29ος περιορισμός θέτει το άνω όριο για το πλήθος των διαδρομών μέσα σε ένα συγκεκριμένο δρομολόγιο (κύκλο), το οποίο δεν θα πρέπει να υπερβαίνει τον αριθμό των κόμβων του δρομολογίου μειωμένο κατά 1.

2.5 Το Πρόβλημα Δρομολόγησης Οχημάτων με πολλές Αποθήκες (OVRP)

Σε προβλήματα τύπου MDVRP (Multiple Depot Vehicle Routing Problem) υπάρχουν παραπάνω από μία αποθήκες. Κάθε πελάτης εξυπηρετείται μονάχα μία φορά και θα εξυπηρετηθεί από εκείνη την αποθήκη που συμφέρει κάθε φορά με βάση τα μεταφορικά κόστη. Έστω ότι υπάρχουν φ αποθήκες. Αν χρησιμοποιηθούν για τις υπόλοιπες μεταβλητές οι ίδιοι

συμβολισμοί με τις προηγούμενες μεθόδους η μοντελοποίηση θα έχει ως εξής:

$$\min \sum_{\alpha=1}^{M} \sum_{\beta=1}^{M} \sum_{\gamma=1}^{\nu} \sum_{\pi=1}^{\varphi} \kappa_{\alpha\beta}\, \delta_{\alpha\beta\gamma\pi} \qquad (30)$$

Οι περιορισμοί θα έχουν ως εξής:

Όλες οι μεταβλητές απόφασης είναι δυαδικές (0 ή 1):

$$\delta_{\alpha\beta\gamma\pi} \in \{0,1\} \qquad \forall\, \gamma \in 1, \dots, \nu \quad \kappa\alpha\iota$$
$$\forall\, \alpha \in 1, \dots, M \quad \kappa\alpha\iota$$
$$\forall\, \beta \in 1, \dots, M \quad \kappa\alpha\iota$$
$$\forall\, \pi \in 1, \dots, \varphi \qquad (31)$$

Από το σύνολο των αποθηκών αναχωρούν ν οχήματα

$$\sum_{\pi=1}^{\varphi} \sum_{\beta=1}^{M} \sum_{\gamma=1}^{\nu} \delta_{1\beta\gamma\pi} = \nu \qquad (32)$$

Όλα τα οχήματα έχουν χωρητικότητα ίση με Χ:

$$\sum_{\alpha=1}^{M} \sum_{\beta=1}^{M} (\delta_{\alpha\beta\gamma\pi} * \zeta_{\alpha}) \leq X \qquad \forall\, \pi \in 1, \dots, \varphi \quad \kappa\alpha\iota$$
$$\forall\, \gamma \in 1, \dots, \nu \qquad (33)$$

Περιορισμός διαδρομών:

$$\sum_{\alpha,\beta \in N} \delta_{\alpha\beta\gamma\pi} \leq |N| - 1 \qquad \forall\, \pi \in 1, \dots, \varphi \quad \kappa\alpha\iota$$

$$\forall\, \gamma \in 1, \dots, \nu \quad \kappa\alpha\iota$$

$$\forall\, N \subseteq \{2, \dots, M\} \qquad (34)$$

Κάθε πελάτης εξυπηρετείται μονάχα μία φορά

$$\sum_{\alpha=1}^{M} \sum_{\gamma=1}^{\nu} \delta_{\alpha\beta\gamma\pi} = 1 \qquad \forall\, \alpha \in 2, \dots, M \quad \kappa\alpha\iota$$

$$\forall\, \alpha \in 2, \dots, M \quad \kappa\alpha\iota$$

$$\forall\, \beta \in 2, \dots, M \quad \kappa\alpha\iota$$

$$\forall\, \gamma \in 1, \dots, \nu \qquad (35)$$

Εξασφαλίζεται ότι το ίδιο όχημα που εισέρχεται σε έναν συγκεκριμένο κόμβο, στην συνέχεια εξέρχεται από τον ίδιο κόμβο.

$$\sum_{\beta=1}^{M} \delta_{\alpha\beta\gamma\pi} - \sum_{\beta=1}^{M} \delta_{\beta\alpha\gamma\pi} = 0 \quad \forall\, \pi \in 1, \dots, \varphi \quad \kappa\alpha\iota$$

$$\forall\, \alpha \in 1, \dots, M \quad \kappa\alpha\iota$$

$$\forall\, \gamma \in 1, \dots, \nu \qquad (36)$$

όπου:

$$\delta_{\alpha\beta\gamma\pi} = \begin{cases} 1, & Αν\ το\ όχημα\ \gamma\ που\ εφοδιάστηκε\ από\ την\ \pi\ αποθήκη \\ & πάει\ στον\ κόμβο\ \beta\ αμέσως\ μετά\ τον\ κόμβο\ \alpha \\ 0, & αλλιώς \end{cases}$$

3. Αλγόριθμοι που επιλύονται από το Σύστημα Υποστήριξης Αποφάσεων

3.1 Εισαγωγή - Εξελίξεις στο χώρο των μεθευρετικών αλγορίθμων

Στη δημοσίευση Braysy et al. [15] προτείνεται μία νέα ντετερμινιστική μεθευρετική ανόπτηση για το πρόβλημα VRP μεταβλητού στόλου με σταθερά κόστη και χρονικά παράθυρα. Τέτοιοι μεθευρετικοί αλγόριθμοι χωρίζονται σε τρεις φάσεις. Καταρχήν δημιουργούνται αρχικές λύσεις με ευρετικούς αλγορίθμους που συνδυάζουν τεχνικές διαφοροποίησης και "μηχανισμούς εκμάθησης", Στη συνέχεια γίνεται μία προσπάθεια για τη μείωση των διαδρομών της αρχικής λύσης με μία εκ νέου διαδικασία τοπικής αναζήτησης και τέλος η λύση της δεύτερης φάσης βελτιώνεται ακόμα περισσότερο από ένα σετ τεσσάρων μεθόδων τοπικής αναζήτησης που ενσωματώνονται στα πλαίσια μίας ντετερμινιστικής ανόπτησης. Τα υπολογιστικά πειράματα στα δοκιμαστικά παραδείγματα LS168 δείχνουν ότι αυτή η προτεινόμενη μέθοδος υπερτερεί των αποτελεσμάτων που είχαν δημοσιευθεί νωρίτερα και μάλιστα βελτιώνει σχεδόν όλες τις μέχρι τότε γνωστές βέλτιστες λύσεις. Μία από τις πρώτες προσεγγίσεις τέτοιου είδους είναι ο γενετικός αλγόριθμος που προτάθηκε από Ochi et al. [3] για το πρόβλημα VRP (μεταβλητού στόλου με σταθερά κόστη) όπου δημιουργείται ένας αρχικός πληθυσμός λύσεων με την ευρετική μέθοδο SWAP.

Ο ίδιος αλγόριθμος δοκιμάστηκε με παράλληλο προγραμματισμό στη δημοσίευση Ochi et al. [4]. Και στις δύο δημοσιεύσεις όμως δεν αναφέρονται λεπτομέρειες για τις υπολογιστικές δοκιμές. Για αυτή την οικογένεια προβλημάτων αναπτύχθηκαν προσεγγίσεις της μεθόδου "Tabu Search" από τους Osman και Salhi [23], Gendreau et al. [7], και Wassan και Osman [8]. Όλοι αυτοί οι αλγόριθμοι ήταν επεκτάσεις προηγούμενων προσεγγίσεων για προβλήματα VRP (μεταβλητού στόλου με σταθερά κόστη αλλά και ετερογενή προβλήματα VRP με μεταβαλλόμενα κόστη ανάλογα με τα οχήματα), χρησιμοποιώντας έλεγχο εφικτότητας για κάθε ένα πρόβλημα και εκτίμηση της αντικειμενικής συνάρτησης. Συγκεκριμένα οι Osman και Salhi [23] χρησιμοποίησαν μία και μόνο ανταλλαγή μεταξύ γειτονικών λύσεων μαζί με ένα απλό μηχανισμό λίστας "tabu", ενώ οι Wassan και Osman [8] ανέμειξαν διάφορες αποδοτικές στρατηγικές για να βελτιώσουν την ποιότητα του αλγορίθμου.

Διάφοροι ευαίσθητοι μηχανισμοί αναζήτησης μεταξύ "γειτονικών" λύσεων βασισμένοι σε λ ανταλλαγές μεταξύ τους συνδυάζονται με αποδοτικές τεχνικές

διαχείρισης δεδομένων για το χειρισμό των λιστών “tabu”. Η μέθοδος “tabu search” στη δημοσίευση Gendreau et al. [7], περιλαμβάνει έναν κλασικό αλγόριθμο βασισμένο σε “γείτονες με καλά χαρακτηριστικά” με προσαρμόσιμους μηχανισμούς μνήμης των Rochat and Taillard [9]. Τα αποτελέσματα του αλγορίθμου των Osman και Salhi [23] (στα παραδείγματα G20) είχαν ένα μέσο ποσοστιαίο σφάλμα περίπου 0,68% με βάση τις καλύτερες μέχρι τότε δημοσιευμένες λύσεις. Ο αλγόριθμος από Gendreau et al. [7] δοκιμάστηκε στα παραδείγματα G12, έχοντας ένα μέσο ποσοστιαίο σφάλμα ίσο με 0,24%, με μέσο χρόνο υπολογισμού 765 δευτερόλεπτα σε ένα Sun Sparcstation 10.

Ο ίδιος αλγόριθμος δοκιμάστηκε επίσης στα παραδείγματα T8, έχοντας ένα μέσο ποσοστιαίο σφάλμα ίσο με 0.09% με μέσο χρόνο υπολογισμού ίσο με 1151 δευτερόλεπτα. Η μέθοδος ‘tabu search’ των Wassan και Osman [8] παράγει καλά αποτελέσματα στα παραδείγματα G20 καθώς και στα παραδείγματα T8. Συγκεκριμένα στα παραδείγματα G20, το μέσο ποσοστιαίο σφάλμα είναι ίσο με 0.41% και ο μέσος χρόνος υπολογισμού είναι ίσος με 1215 δευτερόλεπτα. Στα παραδείγματα T8, το μέσο ποσοστιαίο σφάλμα είναι ίσο με 0.47% και ο μέσος χρόνος υπολογισμού είναι ίσος με 2098 δευτερόλεπτα σε έναν Sun Sparc 1000. Επίσης έχουν αναπτυχθεί δύο μεθευρετικοί αλγόριθμοι για το πρόβλημα VRP (ετερογενές VRP με κόστη που εξαρτώνται από τα οχήματα) από Tarantilis et al. [10, 11] βασισμένοι στην αποδοχή των λύσεων με βάση ένα κατώφλι: και οι δύο μέθοδοι ξεκινούν από μία λύση προερχόμενη από έναν ευρετικό αλγόριθμο, και στην συνέχεια ακολουθείται μία επαναληπτική διαδικασία αποδοχής με βάση τα κατώφλια. Σε αυτή τη διαδικασία, δημιουργείται επαναληπτικά μία τυχαία λύση στη “γειτονιά” της τρέχουσας λύσης επίσης υπολογίζεται και η τιμή για το κατώφλι η οποία αναπαριστά τη βελτίωση σε σχέση με την τιμή της τρέχουσας λύσης. Η νέα λύση μπορεί να γίνει αποδεκτή μετά από μία διαδικασία σύγκρισης της τιμής του κατωφλίου της με τις τιμές των κατωφλίων των M μέχρις στιγμής καλύτερων λύσεων (οι οποίες είναι αποθηκευμένες σε μία λίστα).

Σε αυτές τις δύο δημοσιεύσεις έχουν προταθεί διαφορετικοί τρόποι για την ανανέωση της λίστας με τα κατώφλια. Οι μέθοδος στην Tarantilis et al. [10] δοκιμάστηκε στα παραδείγματα T8 και παρήγαγε αποτελέσματα με μέση απόκλιση 0,79% σε σχέση με τις μέχρις τότε καλύτερες λύσεις με μέσο χρόνο υπολογισμού τα 223 δευτερόλεπτα σε υπολογιστή Pentium III, των 550 MHz. Τα αποτελέσματα της μεθόδου από Tarantilis et al. [11] είναι ελαφρώς καλύτερα στα παραδείγματα T8 με μέση απόκλιση 0,62% και μέσο χρόνο υπολογισμού 607 δευτερόλεπτα σε υπολογιστή Pentium II/400. Στην δημοσίευση Li et al. [12] προτείνεται μια παρόμοια προσέγγιση η οποία είναι παραλλαγή της μεθευρετικής προσομοιωμένης ανόπτησης. Η μέθοδός τους δοκιμάστηκε στην δημοσίευση Golden et al. [13] στα παραδείγματα T8 και σε πενταβάθμια παραδείγματα 200 έως 360 πελατών. Στα προβλήματα T8,

πέτυχαν μέση ποσοστιαία απόκλιση ίση με 0,03% με μέσο χρόνο υπολογισμού 286 δευτερόλεπτα σε έναν υπολογιστή Athlon 1 GHz. Επίσης προτείνεται μια προσέγγιση “αποσύνθεσης-σύνθεσης” για το πρόβλημα VRP μεταβλητού στόλου με σταθερά κόστη και χρονικά παράθυρα στη δημοσίευση Dell’Amico et al. [14].

Συγκεκριμένα, χρησιμοποιείται μια παράλληλη διαδικασία εισαγωγής για την παροχή μιας αρχικής λύσης και για την πιθανή συμπλήρωση των μερικών λύσεων που παράγονται στο βήμα της “αποσύνθεσης”. Αυτό το βήμα πραγματοποιείται επιλέγοντας μία διαδρομή για να καταργηθεί. Η προτεινόμενη προσέγγιση υπερτερούσε από τους αλγορίθμους των Liu και Shen [15] και Dullaert et al. [16] όσον αφορά στα παραδείγματα LS168. Υπάρχουν δύο μεθευρετικές μέθοδοι για το πρόβλημα VRP όπου οι τύποι των οχημάτων εξαρτώνται από τους πελάτες που εξυπηρετούν. Αυτές οι δύο μεθευρετικές μέθοδοι βασίζονται στην αναγωγή του προβλήματος σε γενικό πρόβλημα διαδρομών.

Η πρώτη μέθοδος είναι η “tabu search” που προτάθηκε από τους Cordeau and Laporte [20] για την παραλλαγή του προβλήματος VRP με “χρονικά παράθυρα”. Το πρόβλημα ανάγεται σε ένα πρόβλημα με χρονικές περιόδους όπου κάθε τύπος οχήματος σχετίζεται με μία διαφορετική περίοδο-μέρα και επιβάλλεται ότι κάθε πελάτης πρέπει να εξυπηρετηθεί μία φορά σε μία από τις μέρες που αντιστοιχούν στο συμβατό όχημα. Στη συνέχεια το πρόβλημα λύνεται χρησιμοποιώντας τη μέθοδο “tabu search” που προτείνεται στην δημοσίευση Cordeau et al. [21]. Η μέση ποσοστιαία απόκλιση ισούται με 1,48% με μέσο χρόνο υπολογισμού τα 12 δευτερόλεπτα σε έναν Sun Ultra 2 (300 MHz). Ο δεύτερος μεθευρετικός αλγόριθμος για το πρόβλημα VRP όπου οι τύποι των οχημάτων εξαρτώνται από τους πελάτες που εξυπηρετούν, δημιουργήθηκε από τους Pisinger και Ropke [19]. Μετέτρεψαν το πρόβλημα σε ένα πρόβλημα “συλλογής-διανομής” με χρονικά παράθυρα, το οποίο λύθηκε χρησιμοποιώντας ένα προσαρμοστικό αλγόριθμο αναζήτησης . Η μέθοδος τους υπερέχει ξεκάθαρα όλως των προηγούμενων που περιγράφηκαν πριν, όσον αφορά στα αποτελέσματα και ο μέσος χρόνος υπολογισμού ήταν 162 δευτερόλεπτα σε έναν υπολογιστή Pentium 4, των 3 GHz. [20]

3.2 Άπληστη Τυχοποιημένη Προσαρμοστική Αναζήτηση με (GRASP)

Η μέθοδος της Άπληστης Τυχοποιημένης Προσαρμοστικής Αναζήτησης (GRASP, Greedy Randomized Adaptive Search Procedure) πραγματοποιείται σε δύο σκέλη. Στο πρώτο σκέλος χρησιμοποιείται μια τυχοποιημένη συνάρτηση απληστίας η οποία δίνει καλά διαφοροποιημένες λύσεις (σχετικά καλής ποιότητας) και στο δεύτερο σκέλος χρησιμοποιούνται αποτελεσματικές μέθοδοι τοπικής αναζήτησης (όπως 2-opt, 3-opt, SWAP, Relocate και Path-Relinking), με τις οποίες επιτυγχάνονται τοπικά και ολικά βέλτιστα.

Ο ψευδοκώδικας της μέθοδος της Άπληστης Τυχοποιημένης Προσαρμοστικής Αναζήτησης έχει ως εξής:

ΓΙΑ n βήματα

Εφαρμογή Τυχοποιημένης Συνάρτησης Απληστίας (για δημιουργία

νέας λύσης)

Αποθήκευση λύσης και του αντίστοιχου κόστους

ΤΕΛΟΣ ΓΙΑ

ΓΙΑ *(όλες τις λύσεις)*

Εφαρμογή Τοπικής Αναζήτησης

Υπολογισμός της αντικειμενικής συνάρτησης (κόστους) της λύσης

ΑΝ *(κόστος νέας λύσης < κόστος παλαιότερης)*

Αντικατάσταση παλαιάς λύσης και του αντίστοιχου κόστους

ΤΕΛΟΣ ΑΝ

ΤΕΛΟΣ ΓΙΑ

3.2.1Τυχοποιημένη Μέθοδος Απληστίας

Η τυχοποιημένη μέθοδος απληστίας πραγματοποιείται σε τόσα βήματα όσοι είναι οι κόμβοι. Σε κάθε βήμα η λύση που κατασκευάζεται εμπλουτίζεται με έναν επιπλέον κόμβο. Στο τέλος προκύπτει μία εφικτή λύση που συνήθως όμως δεν αντιστοιχεί σε τοπικό (ή ολικό βέλτιστο).

Τα δεδομένα που χρησιμοποιούνται για την υλοποίηση της συγκεκριμένης μεθόδου αφορούν τη ζήτηση του κάθε κόμβου, τους χρόνους διαδρομών και εξυπηρέτησης κτλ. Έτσι εξασφαλίζεται ότι ικανοποιούνται οι περιορισμοί και βρίσκονται οι κόμβοι που αντιστοιχούν στα μικρότερα μεταφορικά κόστη, κατά τη διαδικασία σταδιακής κατασκευής της λύσης του πρώτου βήματος (με την τυχοποιημένη άπληστη μέθοδο). Σε κάθε επιμέρους βήμα του πρώτου σκέλους της μεθόδου επιλέγεται ένας από τους κόμβους που περιγράφηκαν πριν.

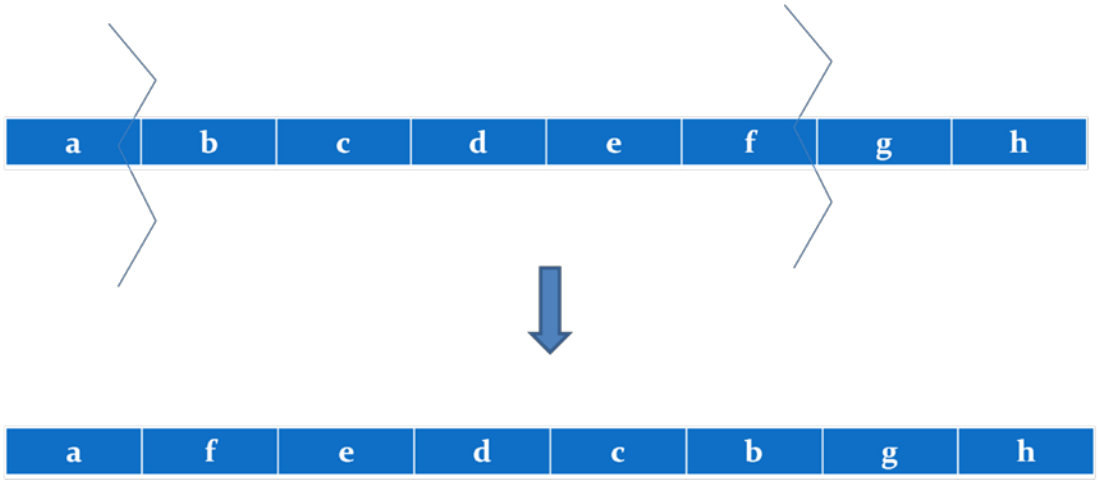
Για τη δημιουργία μίας λύσης με τη συγκεκριμένη μέθοδο ακολουθείται η εξής διαδικασία: Διαμορφώνουμε σταδιακά τη λύση προσθέτοντας σε κάθε βήμα έναν κόμβο. Ο κόμβος που προστίθεται σε κάθε βήμα είναι ένας από τους κόμβους που αντιστοιχεί στα μικρότερα κόστη. Μόλις διαμορφωθεί η λύση, εφαρμόζεται πάνω της τοπική αναζήτηση για την περαιτέρω βελτίωσή της. Συγκεκριμένα εργαζόμαστε ως εξής: Ξεκινάμε να δημιουργούμε μια λύση ψάχνοντας στην αρχή τους πιο κοντινούς κόμβους από τον κόμβο έναρξης 1 (την αποθήκη δηλαδή). Στη συνέχεια επιλέγεται ένας από αυτούς τους κόμβους για προορισμός του φορτηγού και προστίθεται στην πρώτη θέση του διανύσματος της λύσης. Έστω ότι επιλέξαμε τον κόμβο x και τον προσθέσαμε στην πρώτη θέση του διανύσματος της λύσης. Στο επόμενο βήμα από τους υπόλοιπους κόμβους, εντοπίζονται οι πιο κοντινοί ως προς τον κόμβο x. Εάν οι χωρικοί, οι χρονικοί περιορισμοί και τα χρονικά παράθυρα το επιτρέπουν τότε το φορτηγό θα επισκεφθεί κάποιον από αυτούς τους πιο κοντινούς κόμβους, ο οποίος θα τοποθετηθεί στην δεύτερη θέση του διανύσματος της λύσης που κατασκευάζεται, αλλιώς είτε το φορτηγό θα επισκεφθεί οποιοδήποτε άλλο κόμβο που επιτρέπουν οι περιορισμοί, είτε θα επιστρέψει στον κόμβο 1 (αποθήκη), από εκεί θα αναχωρήσει προς έναν σχετικά κοντινό κόμβο. Αυτή η διαδικασία συνεχίζεται έως ότου σχηματιστεί ολόκληρο το διάνυσμα της λύσης, το οποίο στέλνεται σαν όρισμα εξόδου προς το κύριο πρόγραμμα όπου θα χρησιμοποιηθεί για τη συνέχεια της διαδικασίας (στις μεθόδους τοπικής αναζήτησης δηλαδή).

3.2.2 Τοπική Αναζήτηση

3.2.2.1 Η μέθοδος 2-opt

Η μέθοδος 2-opt εφαρμόζεται ως εξής:

Το διάνυσμα της λύσης χωρίζεται σε δύο σημεία, (είτε τυχαία σημεία, είτε σε σημεία όπου υπάρχουν ακριβές μεταβάσεις). Στη συνέχεια αντιστρέφεται η σειρά των κόμβων που βρίσκονται ανάμεσα στα συγκεκριμένα σημεία. Η διαδικασία αναπαρίσταται στην παρακάτω εικόνα:

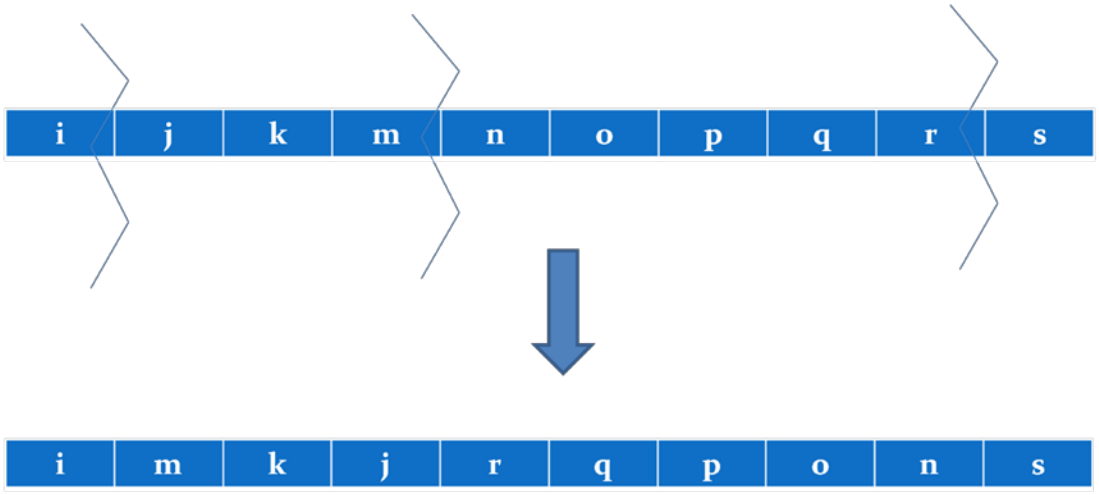


Όπως φαίνεται και από την παραπάνω εικόνα, το νέο διάνυσμα που προκύπτει, διαφέρει από την αρχική λύση μόνο στα στοιχεία που βρίσκονται ανάμεσα στα στοιχεία a και g. Αυτή η λύση είναι σίγουρα μία εφικτή λύση. Στη συνέχεια υπολογίζεται το κόστος της. Αν το κόστος της συγκεκριμένης λύσης είναι καλύτερο από το κόστος της αρχικής τότε η λύση θα αποθηκευθεί για να χρησιμοποιηθεί στο επόμενο βήμα της μεθόδου.

Στο επόμενο βήμα, αναζητείται με τον ίδιο τρόπο μια νέα καλύτερη λύση. Αυτή η διαδικασία επαναλαμβάνεται μέχρι το σημείο όπου η λύση δεν θα μπορεί να βελτιωθεί άλλο ή μέχρι να συμπληρωθεί ο μέγιστος αριθμός βημάτων της μεθόδου 2-opt.

3.2.2.2 Η μέθοδος 3-opt

Στη μέθοδο 3-opt το διάνυσμα της λύσης χωρίζεται σε 3 σημεία αντί για 2. Έπειτα για κάθε ένα από τα ενδιάμεσα μέρη της λύσης πραγματοποιείται αντιστροφή των κόμβων τους.



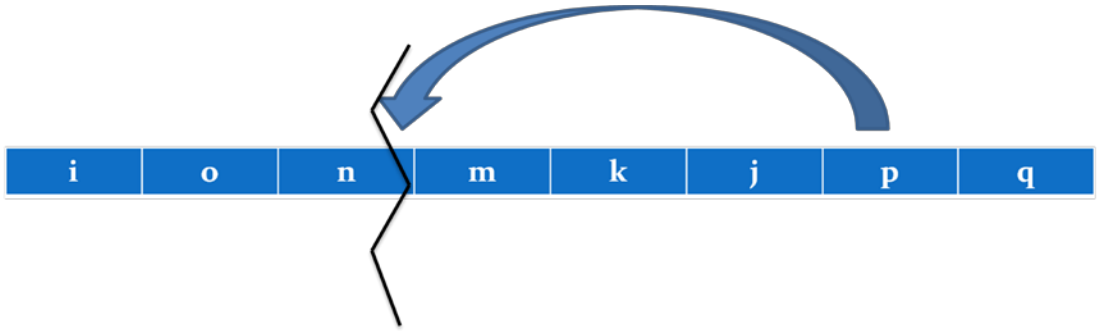
Το νέο διάνυσμα που προκύπτει φαίνεται στην παραπάνω εικόνα και είναι πάλι μία εφικτή λύση. Για να ελεγχθεί αν είναι καλύτερη από τη λύση του προηγούμενου βήματος, υπολογίζεται το κόστος της. Αν το κόστος της συγκεκριμένης λύσης είναι καλύτερο από το κόστος της προηγούμενης τότε θα αποθηκευθεί, για να χρησιμοποιηθεί στην υπόλοιπη διαδικασία.

Με τον ίδιο τρόπο, στο επόμενο βήμα, αναζητείται νέα καλύτερη λύση (θέτοντας σαν αρχική λύση, τη λύση που αποθηκεύθηκε στο προηγούμενο βήμα). Αυτή η διαδικασία επαναλαμβάνεται μέχρι το σημείο όπου η λύση δεν θα μπορεί να βελτιωθεί άλλο ή μέχρι να συμπληρωθεί ο μέγιστος αριθμός βημάτων της μεθόδου 3-opt.

3.2.2.3 Η μέθοδος RELOCATE

Στη μέθοδο RELOCATE μεταφέρεται ένα συγκεκριμένο στοιχείο του διανύσματος της λύσης, σε άλλη θέση, μεταθέτοντας τα στοιχεία που βρίσκονται ενδιάμεσα της αρχικής και της τελικής θέσης του

προαναφερθέντος στοιχείου κατά μία θέση αντίθετα προς τη φορά μεταφοράς του συγκεκριμένο στοιχείου. Σχηματικά η διαδικασία έχει ως εξής:



Και εδώ υπάρχει η δυνατότητα να χωριστεί η λύση, με μεγαλύτερη πιθανότητα, σε σημεία που υπάρχουν ακριβές μεταβάσεις. Έτσι υπολογίζονται για το αρχικό μας διάνυσμα οι ακριβότερές μεταβάσεις από κόμβο σε κόμβο. Στη συνέχεια επιλέγεται με μεγαλύτερη πιθανότητα μία από αυτές τις μεταβάσεις για να καταργηθεί. Έστω ότι η διαδρομή που θα καταργηθεί είναι η n – m. Οι πιθανότητες βελτίωσης του κόστους της νέας λύσης αυξάνονται όταν ο κόμβος μεταφερθεί (με μεγαλύτερη πιθανότητα) ανάμεσα στους κόμβους n και m

1) Δεν δημιουργεί μεγάλα μεταφορικά κόστη

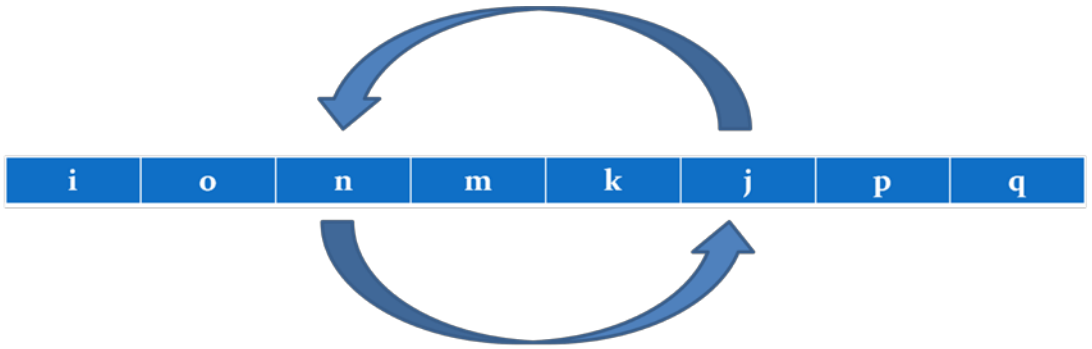
2)Αντιστοιχούσε σε μεγάλα μεταφορικά κόστη στην παρούσα μορφή της λύσης

Ομοίως λοιπόν με τις προηγούμενες μεθόδους υπολογίζονται τα κόστη των μεταβάσεων που μας ενδιαφέρουν και στη συνέχεια να εφαρμόζεται η διαδικασία όπως περιγράφηκε πριν. Η νέα λύση θα διαφέρει από την προηγούμενη στο ότι ο κόμβος p τοποθετήθηκε ανάμεσα στους κόμβους n και m και οι κόμβοι που βρίσκονταν ανάμεσα στην αρχική και την τελική θέση του κόμβου p μεταφέρθηκαν κατά μία θέση προς την αντίθετη δηλαδή κατεύθυνση από αυτήν της μετακίνησης του κόμβου p. Υπολογίζεται το κόστος της νέας εφικτής λύσης και αν το κόστος της συγκεκριμένης λύσης είναι καλύτερο από το κόστος της προηγούμενης τότε θα αποθηκευθεί, για να χρησιμοποιηθεί στο επόμενο βήμα.

Αυτή η διαδικασία επαναλαμβάνεται μέχρι το σημείο όπου η λύση δεν βελτιώνεται περεταίρω ή μέχρι να συμπληρωθεί ο μέγιστος αριθμός βημάτων της μεθόδου.

3.2.2.4 Η μέθοδος SWAP

Η μέθοδος SWAP εφαρμόζεται, αλλάζοντας θέσεις στο διάνυσμα της λύσης σε δύο κόμβους. Συνεπώς οι μεταβάσεις που μας ενδιαφέρουν για κάποιο κόμβο είναι η διαδρομή που τον ενώνει με τον προηγούμενό του κόμβο αλλά και η διαδρομή που τον ενώνει με τον επόμενο κόμβο. Έτσι προστίθεται το κόστος της διαδρομής που καταλήγει στον συγκεκριμένο κόμβο με το κόστος της διαδρομής που ξεκινάει από τον ίδιο κόμβο και το αποτέλεσμα της πρόσθεσης τοποθετείται σε ένα διάνυσμα που περιέχει τα κόστη για όλους τους κόμβους. Αφού υπολογιστούν τα κόστη που προαναφέρθηκαν, υπάρχει η δυνατότητα να αλλάξουμε θέσεις, με μεγαλύτερη πιθανότητα, σε κόμβους που αντιστοιχούν σε ακριβά κόστη. Η διαδικασία συνοπτικά έχει εξής:

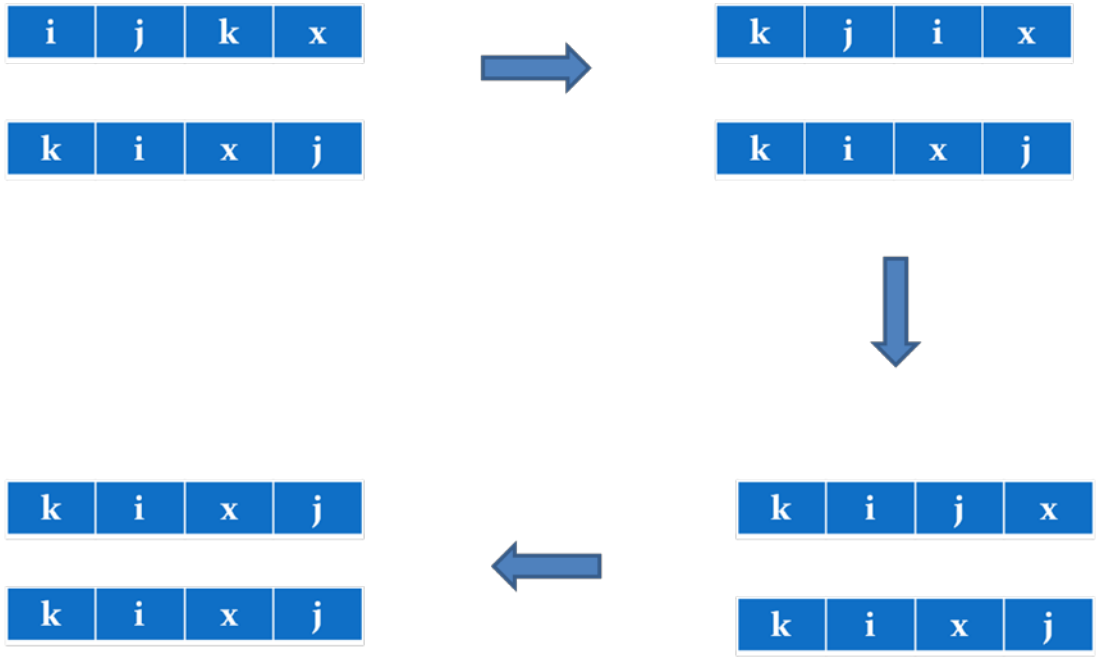


Το νέο διάνυσμα που θα προκύπτει (η νέα λύση δηλαδή), θα διαφέρει από την αρχική μόνο κατά δύο στοιχεία, τα j και n, όπου το ένα αντικατέστησε το άλλο. Η νέα λύση είναι πάντα εφικτή και αν το κόστος της είναι καλύτερο ή από το κόστος της προηγούμενης τότε θα αποθηκευθεί για να χρησιμοποιηθεί στο επόμενο βήμα. Και εδώ η διαδικασία σταματάει στο σημείο όπου η λύση δεν βελτιώνεται άλλο ή μέχρι να συμπληρωθεί ο μέγιστος αριθμός βημάτων.

3.2.2.5 Η μέθοδος Path-Relinking

Στη μέθοδο Path Relinking, χρησιμοποιούνται δύο εφικτές και σχετικά καλές λύσεις. Αξίζει να σημειωθεί ότι υπάρχει μεγαλύτερη πιθανότητα από δύο ήδη καλές λύσεις να προκύψει μία πολύ καλής ποιότητας λύση.

Η μέθοδος Path Relinking σχηματικά έχει ως εξής:



Σε κάθε βήμα της συγκεκριμένης διαδικασίας ένα στοιχείο της πρώτης λύσης γίνεται ίδιο με το αντίστοιχο στοιχείο της δεύτερης λύσης, πχ όπως φαίνεται στο παραπάνω σχήμα το i της πρώτης λύσης αντικαταστάθηκε από το k. Για να συνεχίσει η πρώτη λύση να είναι εφικτή θα πρέπει και το k (που αρχικά βρισκόταν στην τρίτη θέση) να αντικατασταθεί από το i. Σε κάθε βήμα επίσης υπολογίζεται το κόστος της λύσης που προκύπτει και η καλύτερη λύση (μέχρι εκείνη τη στιγμή) αποθηκεύεται. Αυτή η διαδικασία επαναλαμβάνεται έως ότου η πρώτη λύση γίνει ίδια με τη δεύτερη. Το τελικό αποτέλεσμα της συγκεκριμένης μεθόδου είναι η καλύτερη λύση όλων των βημάτων.

Η απόδοση της συγκεκριμένης μεθόδου βελτιώθηκε στην παρούσα διατριβή αφού κατά την εφαρμογή του αλγορίθμου αποθηκεύονταν οι καλύτερες λύσεις και στο τέλος το τελικό αποτέλεσμα ήταν μία λύση από τις καλύτερες. Με αυτόν τον τρόπο επιβραδύνεται ο αλγόριθμος, αλλά αποφεύγεται ευκολότερα ο εγκλωβισμός σε τοπικό βέλτιστο.

3.3 Μιμητικός Αλγόριθμος (Memetic Algorithm)

Ο Μιμητικός Αλγόριθμος που εφαρμόστηκε στα πλαίσια της παρούσας διατριβής, όπως και οι υπόλοιπες μέθοδοι, είναι μια μεθευρετική μέθοδος βελτιστοποίησης προβλημάτων εφοδιαστικής αλυσίδας. Λειτουργεί με παρόμοιο τρόπο με το Γενετικό Αλγόριθμο (χρησιμοποιώντας διαδικασίες διασταύρωσης, μετάλλαξης κτλ) και διαφέρει από αυτόν μονάχα στο ότι στον Μιμητικό αλγόριθμο χρησιμοποιείται και τοπική αναζήτηση.

Συνεπώς το πλεονέκτημα του μιμητικού αλγορίθμου έναντι του γενετικού είναι το ότι μπορεί να προσεγγίσει ευκολότερα και γρηγορότερα ένα συγκεκριμένο τοπικό βέλτιστο το οποίο μπορεί να είναι και ολικό βέλτιστο. Έτσι στην πλειοψηφία των περιπτώσεων ο Μιμητικός αλγόριθμος οδηγεί σε πολύ καλύτερες λύσεις από αυτές του γενετικού, με αποτέλεσμα πολλές φορές να απαιτεί περισσότερο χρόνο για να συγκλίνει κάπου (αφού συγκλίνει σε καλύτερες λύσεις). Και οι δύο αλγόριθμοι μετά από έναν συγκεκριμένο αριθμό βημάτων (γενεών) συγκλίνουν σε ένα τοπικό ή ολικό βέλτιστο.

Αρχικά δημιουργείται μια ομάδα αρχικών λύσεων. Στην συνέχεια ταξινομούνται οι λύσεις με βάση το κόστος τους, ξεκινώντας από την καλύτερη και καταλήγοντας στην χειρότερη. Εφαρμόζεται η διαδικασία της διασταύρωσης για όλες τις λύσεις (με τη σειρά), από την οποία πιθανότατα προκύπτει μία μη εφικτή λύση. Η συγκεκριμένη διαδικασία περιγράφεται παρακάτω. Αμέσως μετά εφαρμόζεται η διαδικασία της μετάλλαξης για να μετατραπούν σε εφικτές οι λύσεις που προέκυψαν από τη διαδικασία της διασταύρωσης.

Μετά εφαρμόζεται η μέθοδος Path Relinking μεταξύ των δύο πρώτων λύσεων της ομάδας, δηλαδή μεταξύ των δύο καλύτερων λύσεων της ομάδας.

Στο τέλος αν η τελική λύση που προκύπτει μετά και από την τοπική αναζήτηση είναι καλύτερη ή λίγο χειρότερη από το ($1^{η}$) αρχική λύση, θα αποθηκευθεί για να χρησιμοποιηθεί στη συνέχεια της διαδικασίας, αλλιώς δεν την κρατάμε.

Αφού οι λύσεις ταξινομούνται με βάση το κόστος τους, αν υπάρχουν ίδιες ή σχεδόν ίδιες λύσεις στην ομάδα, θα γίνουν γειτονικές ή έστω θα έρθουν πολύ κοντά μετά την ταξινόμηση. Έτσι, στα πλαίσια της παρούσας διατριβής, ελέγχεται επιπλέον αν όλα τα πιθανά ζεύγη γειτονικών και κοντινών λύσεων έχουν πολύ μεγάλες ομοιότητες. Εάν βρεθούν λύσεις που μοιάζουν πάρα πολύ, τότε διαγράφονται οι χειρότερες. Η συγκεκριμένη διαδικασία εγγυάται ένα πολύ καλύτερο αποτέλεσμα παρ' όλο που επιβραδύνει τον αλγόριθμο.

Υπάρχει μεγάλη πιθανότητα ο αριθμός των λύσεων της ομάδας να μειωθεί πάρα πολύ. Σε αυτή την περίπτωση τα κενά συμπληρώνονται με τυχαίες λύσεις στις οποίες εφαρμόζεται τοπική αναζήτηση.

Η ίδια διαδικασία συνεχίζεται για όλα τα βήματα (γενιές), μέχρι να συμπληρωθεί ο μέγιστος αριθμός γενεών ή να επέλθει σύγκλιση. Στα πλαίσια της παρούσας διατριβής εφαρμόστηκε επιπλέον και η εξής διαδικασία: Εάν , για έναν μεγάλο αριθμό βημάτων, δεν αλλάξουν οι καλύτερες λύσεις της ομάδας, τότε οι λύσεις δεν διασταυρώνονται με τη σειρά (αφού έχουν ταξινομηθεί με βάση το κόστος τους), αλλά με τυχαίο τρόπο. Θα συνεχίσουν να διασταυρώνονται τυχαία, μέχρι να αλλάξουν οι καλύτερες λύσεις της ομάδας, οπότε και διασταυρώνονται ξανά με τη σειρά.

Σε ψευδοκώδικα ο Μιμητικός αλγόριθμος έχει ως εξής:

*Δημιουργία **n** Αρχικών Λύσεων*

***ΓΙΑ** (έναν αριθμό γενεών)*

***ΓΙΑ** (όλες τις λύσεις)*

Εφαρμογή διασταύρωσης σε όλες τις δυάδες λύσεων

Εφαρμογή Μετάλλαξης

Εφαρμογή Τοπικής Αναζήτησης

Υπολογισμός της αντικειμενικής συνάρτησης της λύσης

ΤΕΛΟΣ ΓΙΑ

*Επιλογή των **n** καλύτερων λύσεων (για την επόμενη γενιά)*

ΤΕΛΟΣ ΓΙΑ

Επιλογή της καλύτερης λύσης

3.3.1 Διασταύρωση (Crossover)

Η διαδικασία της διασταύρωσης δύο λύσεων (έστω των και y) έχει ως εξής: Το διάνυσμα της μίας λύσης χωρίζεται σε δύο επιμέρους διανύσματα (τα οποία δεν είναι κατ' ανάγκη ιδίου μεγέθους). Στη συνέχεια το διάνυσμα της δεύτερης λύσης χωρίζεται στο αντίστοιχο σημείο που χωρίστηκε και το πρώτο.

Στη συνέχεια θα δημιουργηθούν δύο νέες λύσεις (έστω η a και η b). Η πρώτη θα εμπεριέχει το πρώτο μέρος της λύσης x και το δεύτερο μέρος της λύσης y. Αντίστοιχα το διάνυσμα της δεύτερης λύσης (της b δηλαδή) θα απαρτίζεται από το πρώτο μέρος της λύσης y και το δεύτερο μέρος της λύσης x. Οι δύο νέες λύσεις είναι οι "απόγονοι"

Στην πλειοψηφία των περιπτώσεων οι νέες λύσεις είναι μη εφικτές. Αυτό το πρόβλημα λύνεται με τη διαδικασία της "μετάλλαξης", όπου μέσω της συγκεκριμένης διαδικασίας μία μη εφικτή λύση γίνεται εφικτή.

3.3.2 Μετάλλαξη (Mutation)

Σύμφωνα με τη μοντελοποίηση που έχει ακολουθηθεί κάθε κόμβος κατέχει μονάχα μία θέση στο διάνυσμα κάθε εφικτής λύσης. Μετά τη διαδικασία της διασταύρωσης, τα διανύσματα των μη εφικτών λύσεων θα περιέχουν ορισμένους κόμβους δύο φορές και άλλους καμία.

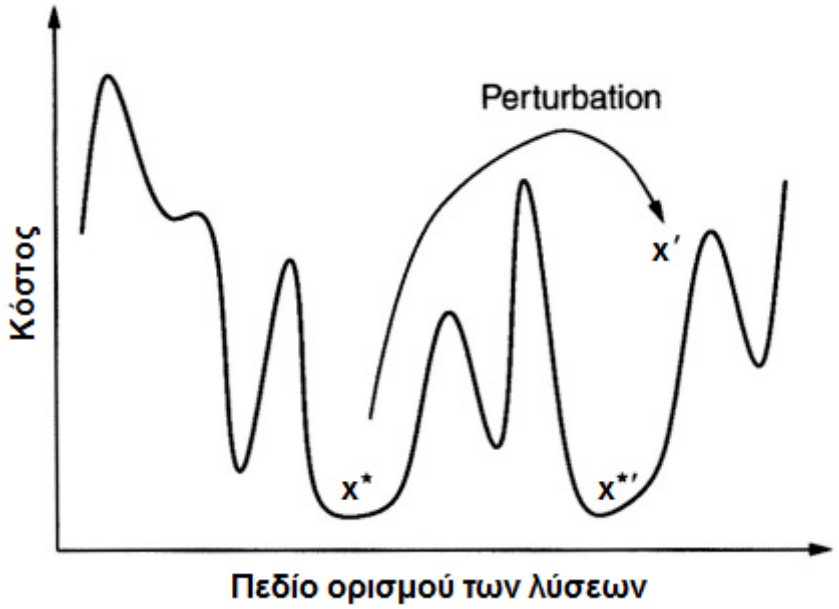
Για τους κόμβους που περισσεύουν (αφού εντοπίζονται δύο φορές στο διάνυσμα της μη εφικτής λύσης) εισάγονται σε ένα διάνυσμα (έστω το e), οι θέσεις τους στο διάνυσμα της μη εφικτής λύσης. Οι κόμβοι που δεν εντοπίζονται καμία φορά εισάγονται σε ένα άλλο διάνυσμα (έστω το r).

Στη συνέχεια εισάγονται οι κόμβοι του διανύσματος r στις θέσεις (του διανύσματος της μη εφικτής λύσης) που εμπεριέχονται στο διάνυσμα e. Αξίζει να σημειωθεί ότι οι κόμβοι τοποθετούνται σε τέτοιες θέσεις ώστε να μην δημιουργούνται λύσεις πολύ μεγάλου κόστους. Αυτό επιτυγχάνεται υπολογίζοντας τα κόστη μετάβασης που προκύπτουν από την κάθε δυνατή εισαγωγή κόμβου στο διάνυσμα της λύσης.

3.4 Επαναληπτική Τοπική Αναζήτηση (Iterated Local Search)

Η τοπική αναζήτηση είναι ένα από τα σημαντικότερα κομμάτια ενός αλγορίθμου βελτιστοποίησης και απαραίτητο εργαλείο για την αποτελεσματικότητά του. Δυστυχώς όμως μετά από παρατεταμένη χρήση τοπικής αναζήτησης ο αλγόριθμος βελτιστοποίησης συγκλίνει σε κάποιο τοπικό βέλτιστο, από το οποίο είναι δύσκολο να “ξεφύγει”. Γι’ αυτό λοιπόν το λόγο προτάθηκε η μέθοδος Iterated Local Search, της οποίας η λειτουργία παρουσιάζεται παρακάτω:

Βασικός σκοπός της μεθόδου iterated local search είναι, η αναζήτηση της λύσης να επικεντρωθεί σταδιακά σε όσο το δυνατό περισσότερα πλεονεκτικά και διαφορετικά μεταξύ τους σημεία του πεδίου ορισμού των λύσεων, έτσι ώστε το τοπικό βέλτιστο ενός σημείου (από αυτά που έχουμε επικεντρωθεί) να είναι και το ολικό βέλτιστο του προβλήματος που επιλύεται. Για να οδηγηθούμε σε ένα πλεονεκτικό σημείο του πεδίου ορισμού των πιθανών λύσεων, χρησιμοποιούμε τη διαδικασία “perturbation”. Η συγκεκριμένη ενέργεια αναπαριστάται γραφικά παρακάτω:



Όπως φαίνεται και από το σχήμα, με τη διαδικασία “Perturbation” μεταφερόμαστε σε ένα (αρκετά) διαφορετικό σημείο του πεδίου ορισμού των λύσεων με την ελπίδα ότι από το νέο σημείο θα μεταφερθούμε εύκολα σε ένα νέο τοπικό βέλτιστο το οποίο μπορεί να είναι καλύτερο από το προηγούμενο ή ακόμα να είναι και ολικό βέλτιστο.

Βέβαια αξίζει να σημειωθεί ότι η μεταβολή που επιτυγχάνεται με τη διαδικασία perturbation, τις περισσότερες φορές, δεν είναι αντιστρεπτή, μια ιδιότητα που δεν χρειάζεται βέβαια στα προβλήματα που επιλύονται στην παρούσα μεταπτυχιακή διατριβή.

Η μέθοδος iterated local search έχει ως εξής:

1. Δημιουργία αρχικής λύσης (x_0)
2. Εφαρμογή τοπικής αναζήτησης στη λύση x_0, για την παραγωγή της λύσης x^*.
3. Πραγματοποίηση της παρακάτω επαναληπτικής διαδικασίας έως ότου ικανοποιηθεί το κριτήριο τερματισμού (π.χ. να μην πραγματοποιηθεί περαιτέρω βελτίωση της λύσης για έναν συγκεκριμένο αριθμό βημάτων):
    a. Εφαρμογή της διαδικασίας Perturbation στη λύση x^* με βάση τη μέχρις στιγμής πορεία επίλυσης του προβλήματος. Εξαγωγή της λύσης x'.
    b. Πραγματοποίηση τοπικής αναζήτησης στη λύση x' με αποτέλεσμα την παραγωγή της λύσης $x^{*'}$.
    c. Λαμβάνεται υπόψη η παρούσα λύση x^*, η λύση που προέκυψε από το προηγούμενο βήμα $x^{*'}$ καθώς και το σύνολο των ανανεωμένων τιμών των παραμέτρων του προβλήματος για να διαπιστωθεί αν ικανοποιείται το κριτήριο (Acceptance Criterion) για την ανανέωση της τιμής του x^* με την τιμή $x^{*'}$ (π.χ. αν το κόστος της $x^{*'}$ είναι μικρότερο από το κόστος της x^*. Αν ικανοποιείται το παραπάνω κριτήριο τότε γίνεται η ανανέωση που αναφέρθηκε και επιστρέφουμε στο βήμα “3.a”, σε αντίθετη περίπτωση επιστρέφουμε στο βήμα “3.a” χωρίς την ανανέωση που προαναφέρθηκε.

Τα βασικά ερωτήματα που πρέπει να μας απασχολήσουν κατά τη κατασκευή της μεθόδου iterated local search είναι 4 και είναι τα εξής:

1. Με ποια μέθοδο θα δημιουργηθούν οι αρχικές λύσεις και πως θα τις επεξεργαστούμε.
2. Ποια ή ποιες μέθοδοι τοπικής αναζήτησης θα χρησιμοποιηθούν.
3. Πως θα εφαρμοστεί η διαδικασία Perturbation (π.χ. ποιος τρόπος (τυχαίος ή συγκεκριμένος) θα επιλεχθεί για να διαφοροποιηθεί η λύση

και σε τι ποσοστό θα διαφοροποιηθεί η λύση. (αυτό το ερώτημα όπως και τα υπόλοιπα θα απαντηθεί στη συνέχεια).

4. Ποιο είναι το κριτήριο που πρέπει να ικανοποιείται για να πραγματοποιηθεί η ανανέωση της τιμής του x^* με την τιμή $x^{*\prime}$ στο βήμα “3.a”.

Συνοπτικά η διαδικασία σε ψευδοκώδικα έχει ως εξής:

Δημιουργία αρχικής λύσης x_0

Εξαγωγή της λύσης x^* από εφαρμογή τοπικής αναζήτησης στη x_0

ΜΕΧΡΙ (να ικανοποιηθεί το κριτήριο τερματισμού)

Εξαγωγή της λύσης x' από εφαρμογή της διαδικασίας “perturbation”

στη x^*

Εξαγωγή της λύσης $x^{*'}$ από εφαρμογή τοπικής αναζήτησης στη x'

ΑΝ (κόστος($x^{*'}$)< κόστος(x'))

$x' = x^{*'}$

ΤΕΛΟΣ ΑΝ

ΤΕΛΟΣ ΜΕΧΡΙ

Στα παρακάτω υποκεφάλαια επεξηγούνται με κάθε λεπτομέρεια οι διαδικασίες και οι σημαντικότερες παράμετροι της μεθόδου Iterated Local Search.

3.4.1 Αρχική λύση

Οι συνηθέστεροι και πιο ενδεδειγμένοι τρόποι δημιουργίας αρχικής λύσης είναι η τυχαία μέθοδος, με την οποία δημιουργείται εντελώς τυχαία μία λύση η οποία βελτιώνεται στη συνέχεια με τοπική αναζήτηση.

Ένας έξυπνος τρόπος δημιουργίας αρχικής λύσης είναι με τη μέθοδο GRASP. Για την υλοποίηση της συγκεκριμένης μεθόδου χρησιμοποιούνται δεδομένα που αφορούν στη ζήτηση του κάθε κόμβου, τους χρόνους διαδρομών και εξυπηρέτησης κτλ. Τέτοιου είδους δεδομένα χρειάζονται για να

υπολογιστεί πότε ένα φορτηγό αδυνατεί να επισκεφθεί κάποια άλλη πόλη λόγω των χωρικών, χρονικών περιορισμών ή χρονικών παραθύρων όπου υφίστανται. Σαν χωρικοί περιορισμοί εννοούνται αυτούς που αναφέρονται στην χωρητικότητα των οχημάτων μεταφοράς π.χ. φορτηγών. Με αυτόν λοιπόν τον τρόπο ξεκινάμε από μία ομάδα καλών λύσεων. Αυτός ο τρόπος έχει δύο πλεονεκτήματα:

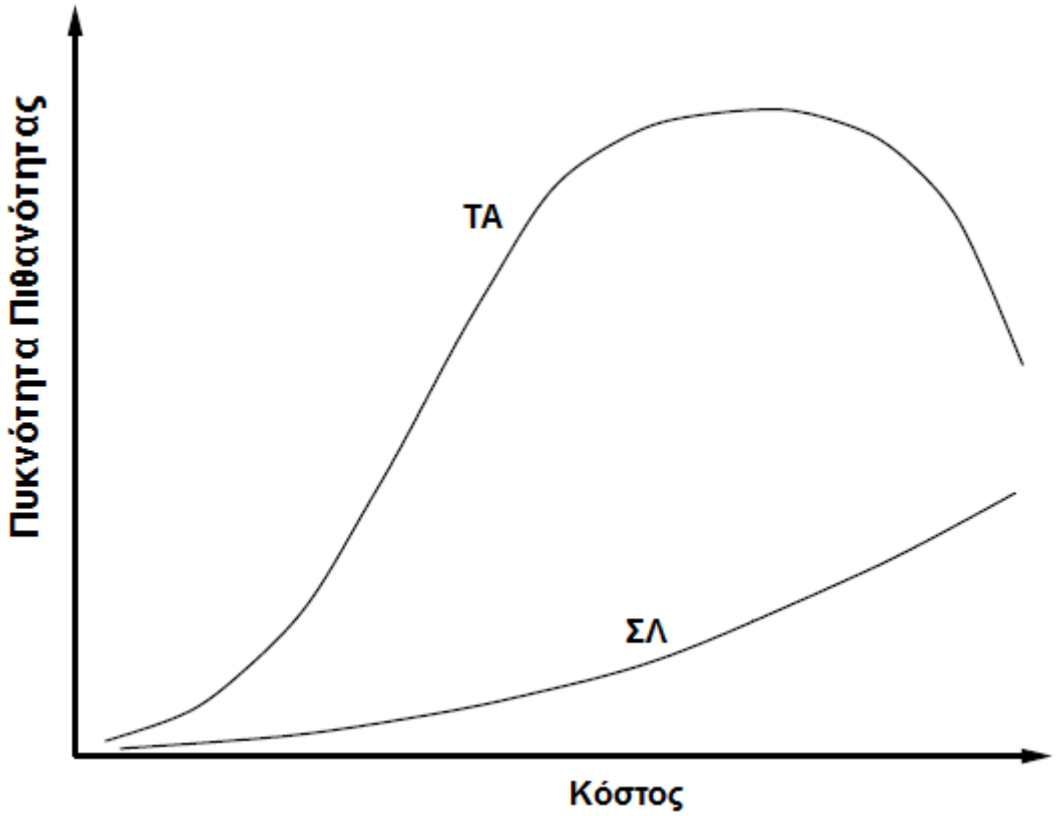
1.Προκύπτουν λύσεις καλής ποιότητας

2.Οι καλές λύσεις προκύπτουν πολύ πιο γρήγορα από τον τυχαίο τρόπο δημιουργίας λύσεων (και με πολύ λιγότερο υπολογιστικό φόρτο), αφού δεν απαιτείται εκτεταμένη τοπική αναζήτηση (όπως στον τυχαίο τρόπο) για την εξαγωγή της τελικής λύσης.

Το μοναδικό αρνητικό στοιχείο αυτής της μεθόδου είναι ότι οι λύσεις που προκύπτουν δεν έχουν τόσο μεγάλη μεταβλητότητα μεταξύ τους όσο αυτές του τυχαίου τρόπου δημιουργίας λύσεων. Με αποτέλεσμα, αν χρησιμοποιείται και σε άλλα μέρη του αλγορίθμου η μέθοδος GRASP μπορεί να μας δίνει παρόμοιες λύσεις σε όλα τα σημεία, πράγμα που εξαρτάται από το πεδίο εφαρμογής του αλγορίθμου GRASP καθώς και τη φύση του προβλήματος

3.4.2 Τοπική Αναζήτηση

Η χρήση τοπικής αναζήτησης επιταχύνει τη διαδικασία βελτιστοποίησης αφού μας δίνει σχετικά πιο εύκολα (με μεγαλύτερη πιθανότητα) καλής ποιότητας λύσεις από μία μέθοδο παραγωγής τυχαίων λύσεων. Αυτό φαίνεται και σχηματικά παρακάτω:



Στο παραπάνω σχήμα με ΤΑ συμβολίζονται οι λύσεις που επιτυγχάνονται με τοπική αναζήτηση και με ΣΛ συμβολίζεται το πεδίο των λύσεων του προβλήματος που επιλύεται.

Η τοπική αναζήτηση γίνεται με τυχαία επιλογή μίας εκ των εξής 4 μεθόδων: 2-opt, 3-opt, Relocate και Swap. Συνήθως η τοπική αναζήτηση παίζει τον καθοριστικότερο ρόλο στη μέθοδο iterated local search, όσο αποδοτικότερη δηλαδή η μέθοδος τοπικής αναζήτησης που χρησιμοποιείται τόσο καλύτερα τα αποτελέσματα της μεθόδου ILS. Οι τρόποι λειτουργίας κάθε μίας από τις 4 μεθόδους τοπικής αναζήτησης παρουσιάστηκαν παραπάνω.

Η λύση που προκύπτει μετά από κάθε μέθοδο τοπικής αναζήτησης είναι σίγουρα μία εφικτή λύση. Στη συνέχεια υπολογίζεται το κόστος της και αν το κόστος της συγκεκριμένης λύσης είναι καλύτερο από το κόστος της αρχικής τότε αποθηκεύεται, εάν όχι τότε συνεχίζουμε στο επόμενο βήμα με την προηγούμενη λύση.

3.4.3 Διαταραχή - Perturbation

Διαταραχή - “Perturbation” είναι η διαδικασία κατά την οποία μία καλής ποιότητας λύση μετατρέπεται ελαφρώς (με σκοπό να βελτιωθεί περεταίρω) και έτσι αποφεύγεται ο “εγκλωβισμός” σε τοπικό βέλτιστο.

Εφαρμόζοντας απλά τοπική αναζήτηση, κάποια στιγμή θα βρεθούμε με μαθηματική ακρίβεια σε τοπικό βέλτιστο. Το οποίο τοπικό βέλτιστο πιθανότατα δεν θα είναι ολικό βέλτιστο με αποτέλεσμα αν δεν καταφέρουμε να “ξεφύγουμε” από το συγκεκριμένο τοπικό βέλτιστο δεν θα επιτύχουμε ποτέ το ολικό βέλτιστο.

Η διαδικασία “Perturbation” λοιπόν λύνει το πρόβλημα που περιγράφτηκε πριν. Οι πιο συνηθισμένη μέθοδος για “perturbation” έχει ως εξής:

1. Μετατρέπεται ένα μέρος της λύσης με τυχαίο τρόπο. Πρέπει το ποσοστό της λύσης που θα αλλάξει να είναι τέτοιο ώστε να ξεφύγουμε από την περιοχή τοπικού βέλτιστου (της παρούσας λύσης που εξετάζουμε) και να οδηγηθούμε σε μια περιοχή σχετικά καλών λύσεων. Έπειτα με εφαρμογή τοπικής αναζήτησης συνήθως προκύπτει μία καλή λύση η οποία μπορεί να είναι καλύτερη από την παρούσα λύση ή ακόμα και να είναι το ολικό βέλτιστό του προβλήματος. Άρα το συγκεκριμένο ποσοστό διαφοροποίησης της λύσης δεν θα πρέπει να είναι ούτε πολύ μικρό ούτε πολύ μεγάλο. Αν το ποσοστό είναι πολύ μικρό τότε δεν θα καταφέρουμε να βγούμε από την περιοχή του τοπικού βέλτιστου της υπάρχουσας λύσης και αν το ποσοστό είναι πολύ μεγάλο τότε χάνονται πολλά από τα καλά χαρακτηριστικά της λύσης μας και είναι δύσκολο να βρεθούμε σε περιοχή καλών λύσεων και ακόμα δυσκολότερο να οδηγηθούμε σε καλύτερη λύση με νέα εφαρμογή τοπικής αναζήτησης. Το ποσοστό της λύσης που μετατρέπεται μπορεί να μεταβάλλεται καθ’ όλη τη διάρκεια της διαδικασίας εύρεσης του βέλτιστου. Γι παράδειγμα μπορούμε να ξεκινήσουμε από ένα ποσοστό p_{min} .

2. Εάν μέσω εφαρμογής της διαδικασίας “Perturbation” και της μετέπειτα εφαρμογής τοπικής αναζήτησης, η λύση που εξετάζουμε δεν βελτιωθεί περαιτέρω, τότε αυξάνεται το ποσοστό διαφοροποίησης της λύσης (p) κατά μία σταθερή ποσότητα, αλλιώς παραμένει να έχει την τιμή p_{min} .

3. Αν χρειαστεί φθάνουμε έως και το p_{max} που έχουμε ορίσει, εάν δεν βελτιωθεί η λύση τότε ξαναγυρνάμε στο p_{min} .

Η διαδικασία “Perturbation” που ακολουθήθηκε στα πλαίσια της παρούσας μεταπτυχιακής διατριβής είχε τη μορφή που περιγράφτηκε πριν με τα βήματα 1 έως 2. Η συγκεκριμένη μορφή είναι η συνηθέστερη και καταλληλότερη για τα προβλήματα που επιλύει το Σύστημα Υποστήριξης Αποφάσεων που κατασκευάστηκε. Τέλος η συγκεκριμένη μέθοδος “Perturbation” είναι από τις ταχύτερες και αποτελεσματικότερες που μπορούν να χρησιμοποιηθούν γι αυτά προβλήματα.

3.4.4 Κριτήριο Αποδοχής (Acceptance Criterion)

Το Acceptance Criterion (δηλαδή το κριτήριο για την ανανέωση της τιμής του x^* με την τιμή $x^{*'}$ στο βήμα 3.c της διαδικασίας Perturbation), είναι ένας από τους σημαντικότερους παράγοντες στη διαδικασία αναζήτησης της βέλτιστης λύσης στο πεδίο των λύσεων τοπικής αναζήτησης. Με το Acceptance Criterion θα πρέπει να διασφαλίζεται ότι η νέα λύση $x^{*'}$ είναι αρκετά διαφοροποιημένη από την x^* αλλά κυρίως να είναι πολύ καλής ποιότητας. Το δεύτερο κριτήριο ικανοποιείται αν επιλεγεί η καλύτερη λύση (δηλαδή σε περίπτωση που επιδιώκουμε στο μέγιστο (100% των περιπτώσεων) την καλή ποιότητα της λύσης, η αντικατάσταση της x^* με την $x^{*'}$ θα γίνει μόνο εάν το κόστος της $x^{*'}$ είναι μικρότερο από το κόστος της x^*. Η διαφοροποίηση ελέγχεται από το πόσο (σε τι ποσοστό) διαφέρουν οι λύσεις x^* και $x^{*'}$.

Το Acceptance Criterion μπορεί να υλοποιηθεί με διάφορους τρόπους, παρακάτω παρουσιάζονται οι 4 επικρατέστεροι:

1. Η λύση γίνεται δεκτή αν και μόνο εάν είναι καλύτερη από την αντίστοιχη λύση του προηγούμενου βήματος. Μόνο σε αυτή την περίπτωση η λύση x^* αντικαθίσταται από τη λύση $x^{*'}$. Αλλιώς συνεχίζουμε με την x^*.

2. Εδώ παρουσιάζεται η ακραία και εντελώς αντίθετη στρατηγική από την προηγούμενη, όπου η νέα λύση $x^{*'}$ γίνεται πάντα δεκτή ανεξάρτητα από το αν το κόστος της είναι μικρότερο από το αντίστοιχο κόστος της λύσης x^*. Εδώ φαίνεται ξεκάθαρα ότι δίνεται ιδιαίτερη έμφαση στη διαφοροποίηση των λύσεων.

3. Στην τρίτη περίπτωση παρουσιάζεται μία ενδιάμεση στρατηγική όπου εάν το κόστος της λύσης $x^{*'}$ είναι μικρότερο από το αντίστοιχο

κόστος της λύσης x^* τότε δεχόμαστε τη λύση $x^{*'}$ και συνεχίζουμε με αυτήν στο επόμενο βήμα της διαδικασίας. Εάν όμως το κόστος της λύσης $x^{*'}$ είναι χειρότερο από το αντίστοιχο κόστος της λύσης x^* τότε δεχόμαστε την $x^{*'}$ με πιθανότητα $e^{(Κόστος\ x^* - Κόστος\ x^{*'})/Θ}$ όπου Θ είναι μία παράμετρος η οποία μειώνεται κατά την εξέλιξη της διαδικασίας όπως την παράμετρο της θερμοκρασίας στην προσομοιωμένη ανόπτηση (simulated annealing). Αν η παράμετρος Θ είναι πολύ μεγάλη τότε τείνουμε προς τη στρατηγική 2 που περιγράφτηκε πριν και αντίστοιχα αν η παράμετρος Θ είναι πολύ μικρή τότε τείνουμε προς την 1η στρατηγική. Σε αυτή την περίπτωση μία καλή εφαρμογή της στρατηγικής θα ήταν να αυξάνεται η τιμή του Θ, όταν διαπιστωθεί ότι η λύση δεν βελτιώνεται περαιτέρω με χρήση πολύ χαμηλών τιμών του Θ, με σκοπό να επέλθει η επιθυμητή διαφοροποίηση των λύσεων. Στη συνέχεια θα καταχωρηθεί ξανά μία μικρή τιμή στο Θ, και θα συνεχιστεί η συγκεκριμένη διαδικασία με τον ίδιο ακριβώς τρόπο.

4. Τέλος η συνήθως η πιο αποδοτική στρατηγική έχει ως εξής: χρησιμοποιείται η στρατηγική 1 που περιγράφθηκε πριν έως ότου σταματήσει να είναι αποδοτική. Σε εκείνο το σημείο πραγματοποιείται μια “επανεκκίνηση” με μία λύση τυχαία προσδιορισμένη είτε προσδιορισμένη με τη μέθοδο GRASP.

Στα πλαίσια της υλοποίησης της παρούσας μεταπτυχιακής διατριβής αποδοτικότερη αποδείχθηκε να είναι η 1η στρατηγική για το Acceptance Criterion και γι’ αυτό χρησιμοποιήθηκε η συγκεκριμένη στρατηγική.

3.5 Μιμητικός Αλγόριθμος με χρήση Άπληστης Τυχοποιημένης Προσαρμοστικής Αναζήτησης (Memetic with GRASP Algorithm)

Αναμφίβολα η μέθοδος GRASP είναι μια διαδικασία που δίνει πάντα γρήγορα λύσεις σχετικά καλής ποιότητας. Αν αυτή η χρήσιμη ιδιότητα της μεθόδου GRASP συνδυαστεί με τις ανάγκες του Μιμητικού Αλγορίθμου για τυχαίες καλής ποιότητας λύσεις που παράγονται όσο το δυνατό πιο γρήγορα, το αποτέλεσμα είναι ένας πολύ αποτελεσματικός αλγόριθμος.

Ο ψευδοκώδικας της συγκεκριμένης μεθόδου έχει ως εξής:

ΓΙΑ n βήματα

Δημιουργία λύσης από Εφαρμογή GRASP

Αποθήκευση λύσης και του αντίστοιχου κόστους

ΤΕΛΟΣ ΓΙΑ

ΓΙΑ *(έναν αριθμό γενεών)*

ΓΙΑ *(όλες τις λύσεις)*

Εφαρμογή διασταύρωσης σε όλες τις δυάδες λύσεων

Εφαρμογή Μετάλλαξης

Εφαρμογή Τοπικής Αναζήτησης

ΑΝ *(ισχύει το κριτήριο για δημιουργία m νέων λύσεων)*

ΓΙΑ *m βήματα*

Εφαρμογή GRASP [για διαφοροποίηση των λύσεων]

ΤΕΛΟΣ ΓΙΑ

ΤΕΛΟΣ ΑΝ

Υπολογισμός της αντικειμενικής συνάρτησης της λύσης

ΤΕΛΟΣ ΓΙΑ

Επιλογή των n καλύτερων λύσεων (για την επόμενη γενιά)

ΤΕΛΟΣ ΓΙΑ

Επιλογή της καλύτερης λύσης

3.6 Μιμητικός Αλγόριθμος Πολλαπλών Πληθυσμών – Νησιών (Island Memetic Algorithm)

Η διαφορά του Μιμητικού Αλγορίθμου Πολλαπλών Πληθυσμών – Νησιών από τον Μιμητικό Αλγόριθμο έγκειται στο γεγονός ότι εξελίσσονται παράλληλα κάποιες ομάδες λύσεων (“νησιά”) και μετά από κάποιο συγκεκριμένο αριθμό βημάτων ή όταν επέλθει σύγκλιση, γίνονται ορισμένες

ανταλλαγές λύσεων μεταξύ των διαφόρων “νησιών”. Οι ανταλλαγές μπορούν να γίνουν κυκλικά (δηλαδή από το $1^{ο}$ στο $2^{ο}$ νησί, από το $2^{ο}$ στο $3^{ο}$ κοκ) ή στην τύχη. Επίσης καλό είναι οι νέες λύσεις σε ένα νησί να είναι μερικές από τις καλύτερες του νησιού από το οποίο προήλθαν και να αντικαθιστούν μερικές από τις χειρότερες λύσεις του νέου νησιού. Πράγμα που ακολουθείται και στον αντίστοιχο αλγόριθμο που υλοποιήθηκε στο πλαίσια της παρούσας μεταπτυχιακής διατριβής. Αξίζει να σημειωθεί ότι, η τοπική αναζήτηση μπορεί να εφαρμοστεί είτε μετά είτε πριν τη μετανάστευση είτε και στα δύο σημεία.

Ο Μιμητικός Αλγόριθμος Πολλαπλών Πληθυσμών – Νησιών είναι μία μεθευρετική μέθοδος επίλυσης προβλημάτων συνδυαστικής βελτιστοποίησης και καλύπτει τα μειονεκτήματα του Γενετικού Αλγορίθμου Πολλαπλών Πληθυσμών – Νησιών (Island Genetic Algorithm). Ο Γενετικός Αλγόριθμος Πολλαπλών Πληθυσμών – Νησιών συνήθως σε πολύ χειρότερες λύσεις από αυτέ που συγκλίνει ο Μιμητικός Αλγόριθμος Πολλαπλών Πληθυσμών Νησιών. Αυτό συμβαίνει επειδή ο Μιμητικός Αλγόριθμος Πολλαπλών Πληθυσμών Νησιών περιλαμβάνει στις υπορουτίνες του μεθόδους Τοπικής Αναζήτησης εκτός από τις βασικές διαδικασίες του Γενετικού Αλγορίθμου Πολλαπλών Πληθυσμών – Νησιών (διασταύρωση – μετάλλαξη – μετανάστευση κτλ). Συνδυάζει μεθόδους τοπικής αναζήτησης καθώς και μεθευρετικές μεθόδους. Χρησιμοποιώντας μία μεθευρετική μέθοδο πραγματοποιείται μια ευρεία αναζήτηση στο χώρο των λύσεων και με τις ευρετικές μεθόδους (τοπικής αναζήτησης) εντοπίζονται τα διάφορα τοπικά βέλτιστα τα οποία μπορεί να είναι και ολικά.

Ο Μιμητικός Αλγόριθμος Πολλαπλών Πληθυσμών – Νησιών βελτιώνει τις λύσεις του με τοπική αναζήτηση ωθώντας τον αλγόριθμο συνήθως σε πολύ καλύτερα αποτελέσματα. Αφού όμως ο αλγόριθμος βρίσκει όλο και καλύτερες λύσεις επόμενο είναι ότι η αναζήτηση θα διαρκέσει περισσότερο. Το γεγονός ότι καθυστερεί λίγο να συγκλίνει ήταν αναμενόμενο, λόγω της πολυπλοκότητας του συγκεκριμένου αλγορίθμου (αφού χρησιμοποιεί επιπλέον μεθόδους τοπικής αναζήτησης).

Σε ψευδοκώδικα ο Μιμητικός Αλγόριθμος Πολλαπλών Πληθυσμών – Νησιών έχει ως εξής:

ΓΙΑ (όλα τα νησιά)

*Δημιουργία **n** Αρχικών Λύσεων*

ΤΕΛΟΣ ΓΙΑ

***ΓΙΑ** (έναν αριθμό βημάτων)*

***ΓΙΑ** (όλα τα “νησιά”)*

ΓΙΑ *(έναν αριθμό γενεών)*

ΓΙΑ *(όλες τις λύσεις(**n**))*

Εφαρμογή διασταύρωσης σε όλες τις δυάδες λύσεων.

Εφαρμογή Μετάλλαξης.

Εφαρμογή Τοπικής Αναζήτησης.

Υπολογισμός της αντικειμενικής συνάρτησης της λύσης.

ΤΕΛΟΣ ΓΙΑ

*Επιλογή των **n** καλύτερων λύσεων (για την επόμενη*

γενιά).

ΤΕΛΟΣ ΓΙΑ

ΤΕΛΟΣ ΓΙΑ

Εφαρμογή Μετανάστευσης

ΤΕΛΟΣ ΓΙΑ

Αρχικοποίηση βέλτιστο κόστος: c= ∞

ΓΙΑ *(i=1 έως "το πλήθος των νησιών")*

ΑΝ *(κόστος καλύτερης λύσης i-οστού νησιού<c*

Βέλτιστη Λύση = καλύτερη λύση i-οστού νησιού

βέλτιστο κόστος = κόστος καλύτερης λύσης i-οστού νησιού

ΤΕΛΟΣ ΑΝ

ΤΕΛΟΣ ΓΙΑ

3.7 Μιμητικός Αλγόριθμος Πολλαπλών Πληθυσμών – Νησιών με χρήση Άπληστης Τυχοποιημένης Προσαρμοστικής Αναζήτησης (Island Memetic with GRASP Algorithm)

Ακόμα και ο Μιμητικός Αλγόριθμος Πολλαπλών Πληθυσμών Νησιών έχει ανάγκη από σχετικά καλές και καλά διαφοροποιημένες λύσεις οι οποίες να μπορούν να παραχθούν ταχύτατα. Η συγκεκριμένη ανάγκη υπάρχει σε διάφορα σημεία του αλγορίθμου. Στη συγκεκριμένη μεταπτυχιακή διατριβή στις αρχικές λύσεις για κάθε ομάδα-νησί, κάθε μία αρχική λύση προέρχεται είτε από τη μέθοδο GRASP (με μία πιθανότητα p), είτε με τυχαίο τρόπο (με πιθανότητα 1- p φυσικά).

Αξίζει να σημειωθεί ότι ο τυχαίος τρόπος εισάγεται για να αυξηθεί η διαφοροποίηση (μεταβλητότητα) των λύσεων, με το σκοπό να μην πραγματοποιηθεί σύγκλιση (σε πρόωρο στάδιο) σε μία σχετικά κακής ποιότητας λύση.

Σε ψευδοκώδικα ο Μιμητικός Αλγόριθμος Πολλαπλών Πληθυσμών – Νησιών έχει ως εξής (όπου p η πιθανότητα χρήσης GRASP έναντι τυχαίας μεθόδου) :

ΓΙΑ όλα τα “νησιά”
    ΓΙΑ όλες τις λύσεις του νησιού (**n**)
        Εξαγωγή τυχαίου αριθμού x (με ομοιόμορφη καταν.) στο διάστημα (0,1)
        ΑΝ x<p
            Δημιουργία λύσης με τυχαίο τρόπο
        ΑΛΛΙΩΣ
            Δημιουργία λύσης από Εφαρμογή GRASP
        ΤΕΛΟΣ ΑΝ
        Αποθήκευση λύσης και του αντίστοιχου κόστους

    ΤΕΛΟΣ ΓΙΑ

ΤΕΛΟΣ ΓΙΑ

ΓΙΑ *(έναν αριθμό βημάτων)*

    ΓΙΑ *(όλα τα νησιά)*

        ΓΙΑ *(έναν αριθμό γενεών)*

            ΓΙΑ *(όλες τις λύσεις του “νησιού”)*

Διασταύρωση σε όλες τις δυάδες λύσεων

Εφαρμογή Μετάλλαξης

Εφαρμογή Τοπικής Αναζήτησης

Υπολογισμός της κόστους της λύσης

ΤΕΛΟΣ ΓΙΑ

*Επιλογή των **n** καλύτερων λύσεων (για επόμενη γενιά)*

ΤΕΛΟΣ ΓΙΑ

ΤΕΛΟΣ ΓΙΑ

Εφαρμογή Μετανάστευσης

ΤΕΛΟΣ ΓΙΑ

Αρχικοποίηση βέλτιστο κόστος: c= ∞

***ΓΙΑ** (i=1 έως "το πλήθος των νησιών")*

***ΑΝ** (κόστος καλύτερης λύσης i-οστού νησιού<c*

Βέλτιστη Λύση = καλύτερη λύση i-οστού νησιού

βέλτιστο κόστος = κόστος καλύτερης λύσης i-οστού νησιού

ΤΕΛΟΣ ΑΝ

ΤΕΛΟΣ ΓΙΑ

3.8 Μιμητικός Αλγόριθμος με χρήση Επαναληπτικής Τοπικής Αναζήτησης (Memetic Algorithm with Iterated Local Search)

Η συγκεκριμένη μέθοδος συνδυάζει τα πλεονεκτήματα της μεθόδου Iterated Local Search με αυτά του Μιμητικού Αλγορίθμου. Ο Μιμητικός Αλγόριθμος χρησιμοποιεί τοπική αναζήτηση και αυτό είναι το πλεονέκτημά του έναντι του γενετικού αλγορίθμου. Χρησιμοποιώντας Iterated Local Search επιταχύνεται ο αλγόριθμος κατά πολύ. Αφού όταν εφαρμόζεται η συγκεκριμένη μέθοδος, το αποτέλεσμα είναι λύσεις σχετικά καλής ποιότητας (στη γειτονιά των καλών λύσεων της αναζήτησης) και μάλιστα λύσεις οι οποίες προκύπτουν σε πολύ μικρό χρονικό διάστημα.

Ο ψευδοκώδικας για τη μέθοδο Memetic Algorithm with Iterated Local Search έχει ως εξής:

ΓΙΑ n βήματα

Δημιουργία και αποθήκευση τυχαίας λύσης x

Εφαρμογή τοπικής αναζήτησης στην x

ΤΕΛΟΣ ΓΙΑ

ΓΙΑ *(έναν αριθμό γενεών)*

ΓΙΑ *(όλες τις λύσεις)*

Εφαρμογή διασταύρωσης σε όλες τις λύσεις με τη σειρά

Εφαρμογή Μετάλλαξης

Εφαρμογή Iterated Local Search

Υπολογισμός της αντικειμενικής συνάρτησης της λύσης

ΤΕΛΟΣ ΓΙΑ

Επιλογή των n καλύτερων λύσεων (για την επόμενη γενιά)

ΤΕΛΟΣ ΓΙΑ

Επιλογή της καλύτερης λύσης

3.9 Άπληστη Τυχοποιημένη Προσαρμοστική Αναζήτηση στο Μιμητικό Αλγόριθμο με χρήση Επαναληπτικής Τοπικής Αναζήτησης (GRASP in Memetic Algorithm and Iterated Local Search)

Στη συγκεκριμένη μέθοδο συνδυάζονται τρεις πολύ αποτελεσματικές υπορουτίνες. Η μέθοδος GRASP παράγει σχετικά γρήγορα λύσεις αρκετά καλής ποιότητας (συγκρινόμενες με τις εντελώς τυχαίες λύσεις κατά την κατασκευή των αρχικών λύσεων. Επίσης στην περίπτωση που η λύση που προκύπτει από τη διασταύρωση των λύσεων(απόγονος) ταυτίζεται με έναν από τους “γονείς” (αρχικές λύσεις), τότε φυσικά δεν λαμβάνεται υπόψη για το επόμενο βήμα αφού προϋπήρχε ήδη στον συγκεκριμένο πληθυσμό λύσεων και αντί για τη λύση που αντιστοιχεί στον απόγονο λαμβάνεται υπόψη για το

επόμενο βήμα μία λύση που παράγεται είτε με τυχαίο τρόπο είτε με τη μέθοδο GRASP.

Με την χρήση υπορουτίνας Μιμητικού Αλγορίθμου επιτυγχάνεται ένα αρκετά μεγάλο εύρος αναζήτησης λύσεων, για την αποφυγή εγκλωβισμού σε τοπικό ελάχιστο και με την υπορουτίνα με Iterated Local Search επιταχύνεται ο αλγόριθμος κάνοντας αποτελεσματικότερη τη διαδικασία της τοπικής αναζήτησης.

Ο ψευδοκώδικας της συγκεκριμένης μεθόδου έχει ως εξής:

ΓΙΑ n βήματα

Δημιουργία τυχαίας μεταβλητής x (με ομοιόμορφη καταν.) στο διάστημα (0,1)

ΑΝ x<(πιθανότητα χρήσης GRASP)

Δημιουργία λύσης από Εφαρμογή GRASP

ΑΛΛΙΩΣ

Δημιουργία λύσης με τυχαίο τρόπο

ΤΕΛΟΣ ΑΝ

Αποθήκευση λύσης και του αντίστοιχου κόστους

ΤΕΛΟΣ ΓΙΑ

ΓΙΑ *(έναν αριθμό γενεών)*

ΓΙΑ *(όλες τις λύσεις)*

Εφαρμογή διασταύρωσης σε όλες τις δυάδες λύσεων

Εφαρμογή Μετάλλαξης

Εφαρμογή Iterated Local Search

ΑΝ *(ισχύει το κριτήριο για δημιουργία m νέων λύσεων)*

ΓΙΑ *m βήματα*

Εφαρμογή GRASP [για διαφοροποίηση των λύσεων]

ΤΕΛΟΣ ΓΙΑ

ΤΕΛΟΣ ΑΝ

Υπολογισμός της αντικειμενικής συνάρτησης της λύσης

ΤΕΛΟΣ ΓΙΑ

Επιλογή των n καλύτερων λύσεων (για την επόμενη γενιά)

ΤΕΛΟΣ ΓΙΑ

Επιλογή της καλύτερης λύσης

3.10 Μιμητικός Αλγόριθμος Πολλαπλών Πληθυσμών-Νησιών με χρήση Επαναληπτικής Τοπικής Αναζήτησης (Island Memetic Algorithm with Iterated Local Search)

Η συγκεκριμένη μέθοδος είναι αρκετά αποτελεσματική αφού εξελίσσονται ταυτόχρονα ομάδες λύσεων και σε κάθε βήμα γίνονται ανταλλαγές μερικών εκ των καλύτερων λύσεων μεταξύ των ομάδων. Έτσι αυξάνεται και η διαφοροποίηση των λύσεων με αποτέλεσμα να αποφεύγεται ο γρήγορος εγκλωβισμός σε τοπικό βέλτιστο. Με την υπορουτίνα που περιέχει Iterated Local Search ο αλγόριθμος επιταχύνεται αισθητά και παράγονται γρήγορα λύσεις πολύ καλής ποιότητας και μάλιστα είναι αρκετά διαφοροποιημένες μεταξύ τους.

Σε ψευδοκώδικα η μέθοδος Island Memetic with Iterated Local Search έχει ως εξής:

ΓΙΑ όλα τα “νησιά”

ΓΙΑ *m βήματα*

Δημιουργία τυχαίας λύσης r

Εξαγωγή r’ από εφαρμογή τοπικής αναζήτησης στη λύση r

Αποθήκευση λύσης r’ και του κόστους της

ΤΕΛΟΣ ΓΙΑ

ΤΕΛΟΣ ΓΙΑ

ΓΙΑ *(έναν αριθμό βημάτων)*

ΓΙΑ *(όλα τα νησιά)*

ΓΙΑ *(έναν αριθμό γενεών)*

ΓΙΑ *(όλες τις λύσεις)*

"Διασταύρωση" σε όλες τις δυάδες λύσεων

Εφαρμογή Μετάλλαξης

Εφαρμογή Iterated Local Search

Υπολογισμός της αντικ. συνάρτησης της λύσης

ΤΕΛΟΣ ΓΙΑ

Επιλογή των m καλύτερων λύσεων (για επόμενη γενιά)

ΤΕΛΟΣ ΓΙΑ

ΤΕΛΟΣ ΓΙΑ

Εφαρμογή Μετανάστευσης

ΤΕΛΟΣ ΓΙΑ

ΓΙΑ *(i=1 έως πλήθος των "νησιών")*

Επιλογή και Αποθήκευση καλύτερης λύσης της i-οστής ομάδας-νησιού

ΤΕΛΟΣ ΓΙΑ

Επιλογή της καλύτερης λύσης όλων των ομάδων με βάση το κόστος

3.11 Επαναληπτική Τοπική Αναζήτηση στο Μιμητικό Αλγόριθμο Πολλαπλών Πληθυσμών-Νησιών με χρήση Άπληστης Τυχοποιημένης Προσαρμοστικής Αναζήτησης (Iterated Local Search in Island Memetic with GRASP)

Η τελευταία αυτή μέθοδος είναι η αποτελεσματικότερη από τις προηγούμενες. Αρχικά παράγονται λύσεις είτε με τη μέθοδο GRASP είτε με εντελώς τυχαίο τρόπο. Πράγμα που αυξάνει τη μεταβλητότητα (διαφοροποίηση) μεταξύ των αρχικών σχετικά καλής ποιότητας λύσεων. Η διαφοροποίηση των λύσεων συντηρείται σε καλά επίπεδα την περισσότερη ώρα αναζήτησης αφού χρησιμοποιείται η υπορουτίνα για Island Memetic. Για τον ίδιο λόγο επίσης και οι λύσεις που παράγονται είναι πολύ καλής

ποιότητας αφού χρησιμοποιείται και τοπική αναζήτηση, με αποτέλεσμα να εντοπίζονται ευκολότερα τα τοπικά βέλτιστα τα οποία μπορεί να είναι και ολικά. Επειδή όμως χρησιμοποιείται και η υποροutίνα Iterated Local Search η τοπική αναζήτηση εκτός από το γεγονός ότι επιταχύνεται παράγει και λύσεις πολύ καλής ποιότητας, ιδιαίτερα μετά τα πρώτα βήματα του αλγορίθμου, αφού ο αλγόριθμος δεν ξεφεύγει αρκετά από τη “γειτονιά” τω καλών λύσεων όπου βρίσκεται.

Ο ψευδοκώδικας για τη συγκεκριμένη μέθοδο έχει ως εξής:

ΓΙΑ (όλα τα “νησιά”)

ΓΙΑ (n βήματα)
Εξαγωγή τυχαίου αριθμού x (με ομοιόμορφη κατ.) στο διάστημα (0,1)
ΑΝ x<p
Δημιουργία λύσης με τυχαίο τρόπο
ΑΛΛΙΩΣ
Δημιουργία λύσης από Εφαρμογή GRASP
ΤΕΛΟΣ ΑΝ
Αποθήκευση λύσης και του αντίστοιχου κόστους

ΤΕΛΟΣ ΓΙΑ

ΤΕΛΟΣ ΓΙΑ

ΓΙΑ *(έναν αριθμό βημάτων)*

ΓΙΑ *(τον αριθμό των νησιών)*

ΓΙΑ *(έναν αριθμό γενεών)*

ΓΙΑ *(όλες τις λύσεις του “νησιού”)*

Εφαρμογή διασταύρωσης σε όλες τις δυάδες λύσεων.

Εφαρμογή Μετάλλαξης.

Εφαρμογή Iterated Local Search.

Υπολογισμός της αντικειμενικής συνάρτησης της λύσης.

ΤΕΛΟΣ ΓΙΑ

Επιλογή των n καλύτερων λύσεων (για την επόμενη γενιά)

ΤΕΛΟΣ ΓΙΑ

ΤΕΛΟΣ ΓΙΑ

Εφαρμογή Μετανάστευσης

ΤΕΛΟΣ ΓΙΑ

Αρχικοποίηση βέλτιστο κόστος: c= ∞

***ΓΙΑ** (i=1 έως πλήθος των "νησιών")*

***ΑΝ** (κόστος καλύτερης λύσης i-οστού νησιού<c*

Βέλτιστη Λύση = καλύτερη λύση i-οστού νησιού

βέλτιστο κόστος = κόστος καλύτερης λύσης i-οστού νησιού

ΤΕΛΟΣ ΑΝ

ΤΕΛΟΣ ΓΙΑ

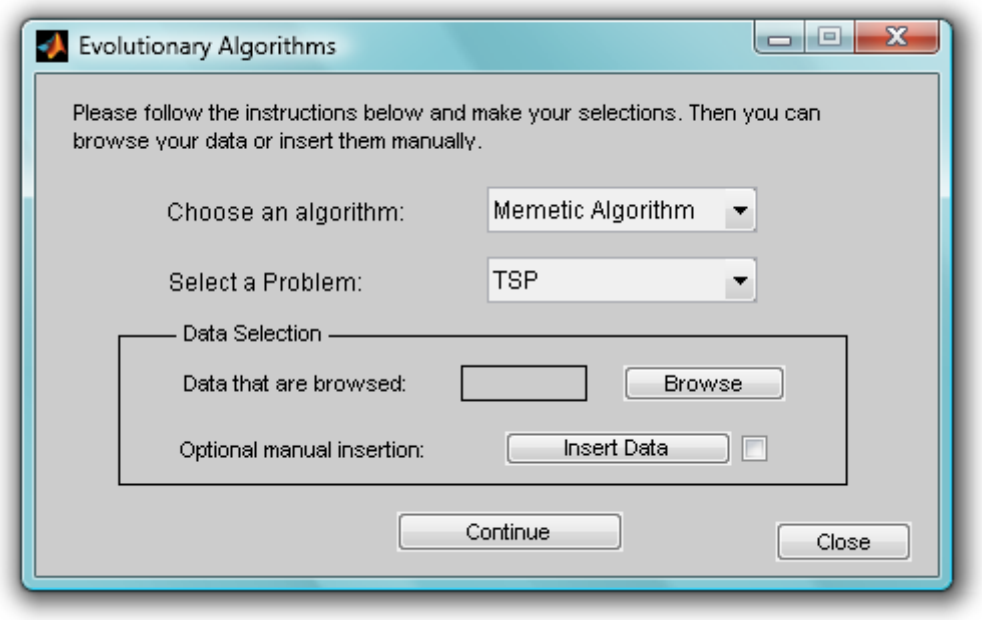
4. Παρουσίαση του Συστήματος Υποστήριξης Αποφάσεων για την βελτιστοποίηση προβλημάτων Εφοδιαστικής Αλυσίδας

4.1 Εισαγωγή

Το Σύστημα Υποστήριξης Αποφάσεων έχει κατασκευαστεί με τέτοιο τρόπο ώστε να δίνει πρώτη προτεραιότητα στις ανάγκες του χρήστη, χωρίς να χρειάζεται ο χρήστης να είναι ένα καλά εξειδικευμένο άτομο, αφού για οποιαδήποτε πιθανή κίνηση του χρήστη, δίνονται οδηγίες σε “tooltips”. Με αυτόν τον τρόπο ο χρήστης ωθείται να εισάγει τα σωστά δεδομένα με το σωστό τρόπο. Επίσης κατά την κατασκευή του συγκεκριμένου συστήματος προβλέφθηκαν όλα τα πιθανά λάθη που μπορεί να κάνει κάποιος χρήστης. Οποιαδήποτε περίπτωση λάθους κίνησης αποτρέπεται αυτόματα από το σύστημα και αυτόματα δίνονται οδηγίες στο χρήστη για το πώς θα δράσει σωστά. Στο συγκεκριμένο σύστημα υπάρχει η δυνατότητα να φορτωθούν τα προς επεξεργασία δεδομένα από διάφορους τύπους αρχείων είτε να εισαχθούν χειροκίνητα (από το χρήστη) όλα τα δεδομένα για οποιοδήποτε πραγματικό πρόβλημα και στη συνέχεια να πραγματοποιηθεί η βελτιστοποίηση με οποιονδήποτε από τους διαθέσιμους αλγορίθμους.

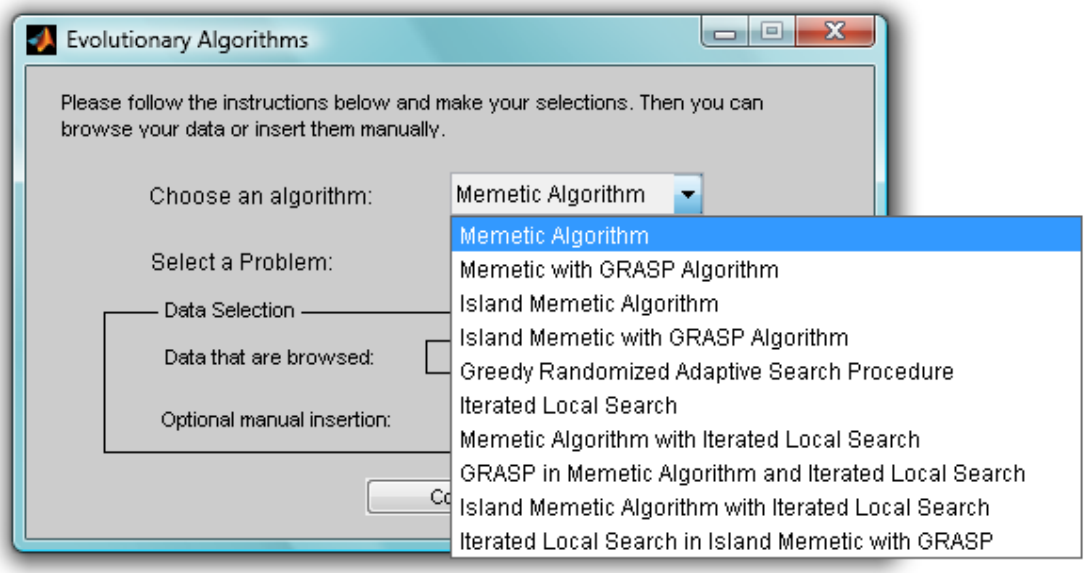
4.2 Επιλογή προβλήματος και μεθόδου

Παρακάτω παρουσιάζεται το πρώτο παράθυρο του Συστήματος Υποστήριξης Αποφάσεων.

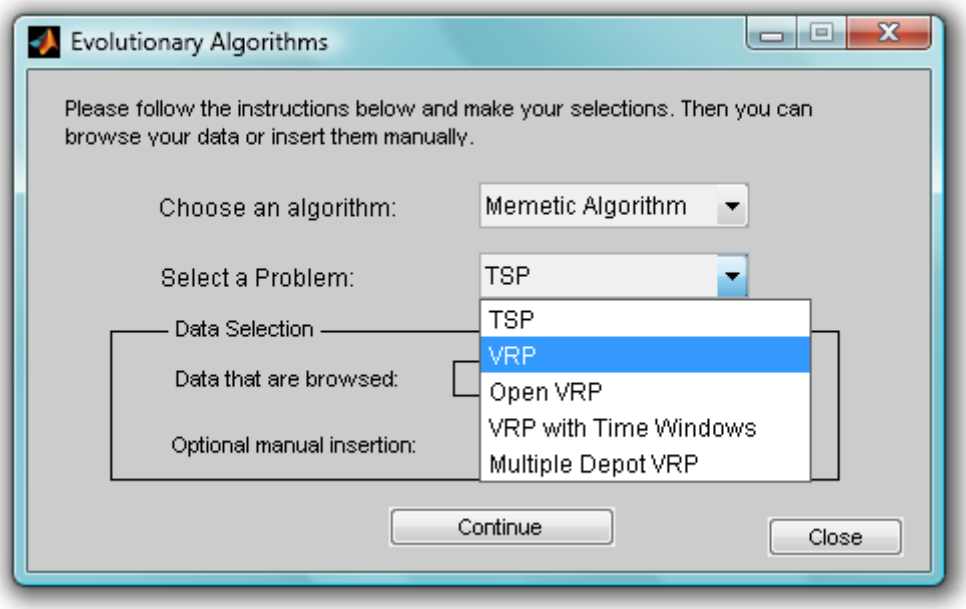


Στο πρώτο πεδίο επιλέγεται ο αλγόριθμος που επιθυμούμε να χρησιμοποιηθεί στη διαδικασία επίλυσης. Οι διαθέσιμοι αλγόριθμοι είναι οι εξής: Memetic Algorithm, Memetic with GRASP Algorithm, Island Memetic Algorithm, Island Memetic with GRASP Algorithm, Greedy Randomized Adaptive Search Procedure, Iterated Local Search, Memetic Algorithm with Iterated Local Search, GRASP in Memetic Algorithm and Iterated Local Search, Island Memetic Algorithm with Iterated Local Search, Iterated Local Search in Island Memetic with GRASP.

Από default (όπως φαίνεται και στην προηγούμενη εικόνα) έχει επιλεγεί η μέθοδος του Μιμητικού Αλγορίθμου (Memetic Algorithm). Ο χρήστης έχει τη δυνατότητα να επιλέξει οποιονδήποτε από τους διαθέσιμους αλγορίθμους από το pop-up menu του πρώτου πεδίου, όπως φαίνεται και στην επόμενη εικόνα:



Για να επιλεγεί η κατηγορία προβλημάτων στην οποία ανήκει το πρόβλημα που θα επιλυθεί χρησιμοποιείται το pop-up menu του δεύτερου πεδίου (“Select a problem”).

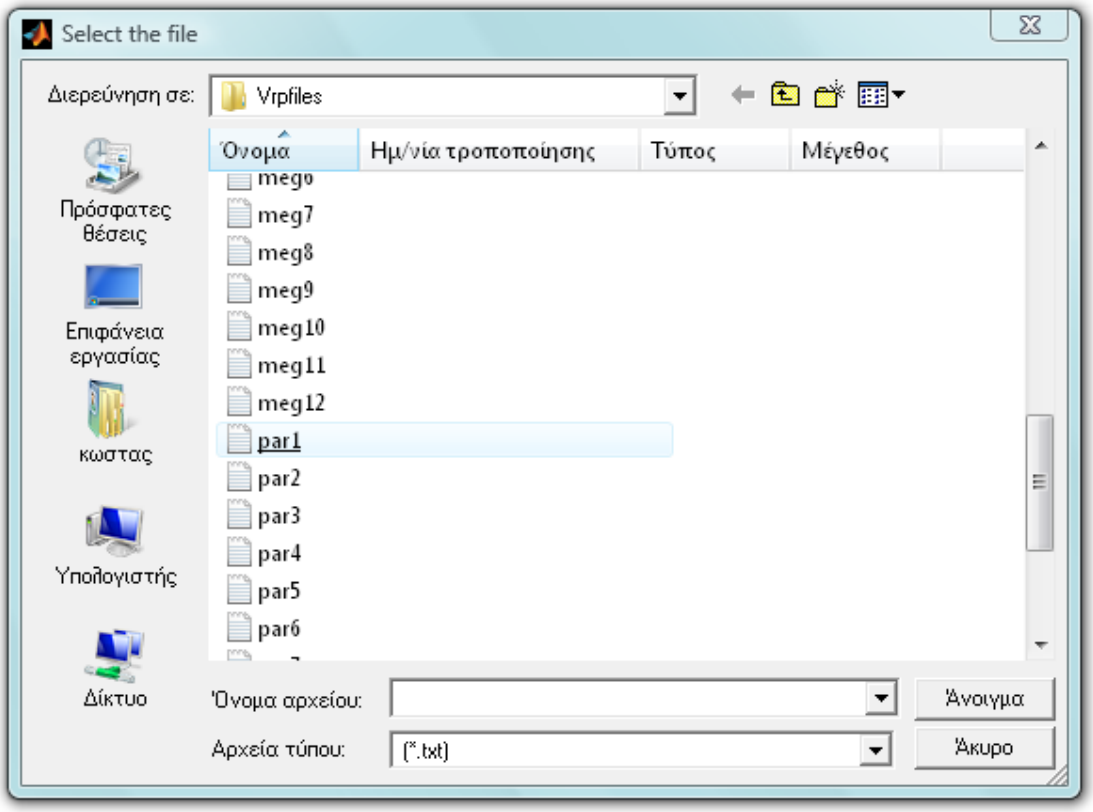


Στο συγκεκριμένο πεδίο από default είναι επιλεγμένος ο τύπος προβλημάτων TSP, και ο χρήστης έχει τη δυνατότητα να επιλέξει οποιονδήποτε άλλο από τους διαθέσιμους τύπους προβλημάτων. Όπως φαίνεται και στην παραπάνω εικόνα, το Σύστημα Υποστήριξης Αποφάσεων που κατασκευάστηκε στα πλαίσια αυτής της διατριβής, επιλύει 5 από τους συνηθέστερους τύπους προβλημάτων εφοδιαστικής αλυσίδας: TSP, VRP, Open VRP , VRP με Time Windows και Multiple Depot VRP.

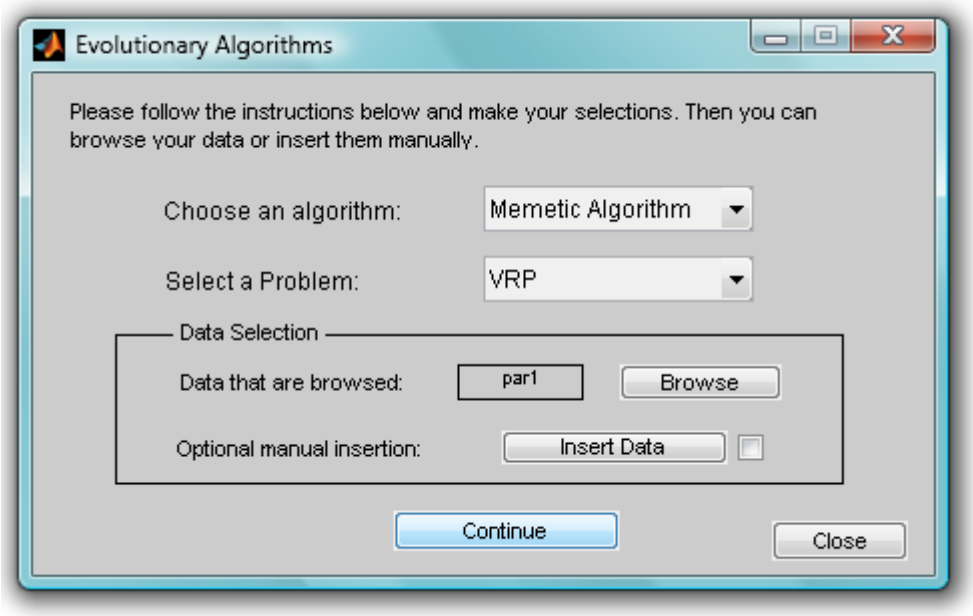
4.3 Εισαγωγή Δεδομένων

Η εισαγωγή των δεδομένων μπορεί να γίνει με δύο εναλλακτικούς τρόπους. Είτε με επιλογή (από το χρήστη) του αρχείου που περιέχει όλα τα δεδομένα του εκάστοτε προβλήματος, είτε με σταδιακή χειροκίνητη εισαγωγή των δεδομένων μέσα από ένα ειδικά σχεδιασμένο interface, στο οποίο δίνονται οδηγίες για οποιαδήποτε κίνηση του χρήστη και αποτρέπεται αυτόματα κάθε πιθανό λάθος.

Παρακάτω φαίνεται το παράθυρο για την επιλογή του αρχείου που περιέχει τα δεδομένα. Το συγκεκριμένο παράθυρο ανοίγει πατώντας το κουμπί “Browse”, οπότε ανοίγει ένας φάκελος που περιέχει όλα τα αρχεία για τον τύπο προβλημάτων που έχει επιλεγεί. Στη συνέχεια ο χρήστης επιλέγει το αρχείο που επιθυμεί και φορτώνονται όλα τα δεδομένα.

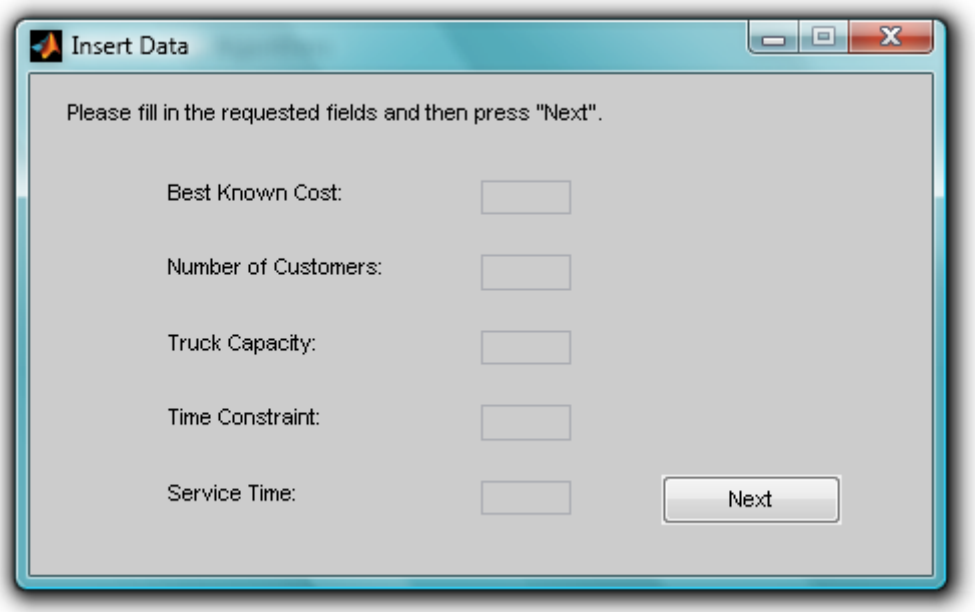


Στη συνέχεια το όνομα του αρχείου που επιλέχθηκε εμφανίζεται δίπλα στο κουμπί “Browse”, όπως φαίνεται στην επόμενη εικόνα.



Ο παραπάνω τρόπος είναι ιδιαίτερα χρήσιμος για την περίπτωση όπου ο χρήστης επιθυμεί να “τρέξει” ένα από τα instances που χρησιμοποιούνται σε παγκόσμια κλίμακα για την σύγκριση αλγορίθμων εφοδιαστικής αλυσίδας. Το “par1” που φαίνεται στην παραπάνω εικόνα είναι ένα από αυτά τα instances. Εάν ο χρήστης επιθυμεί να εισάγει τα δεδομένα με το δεύτερο τρόπο πατάει

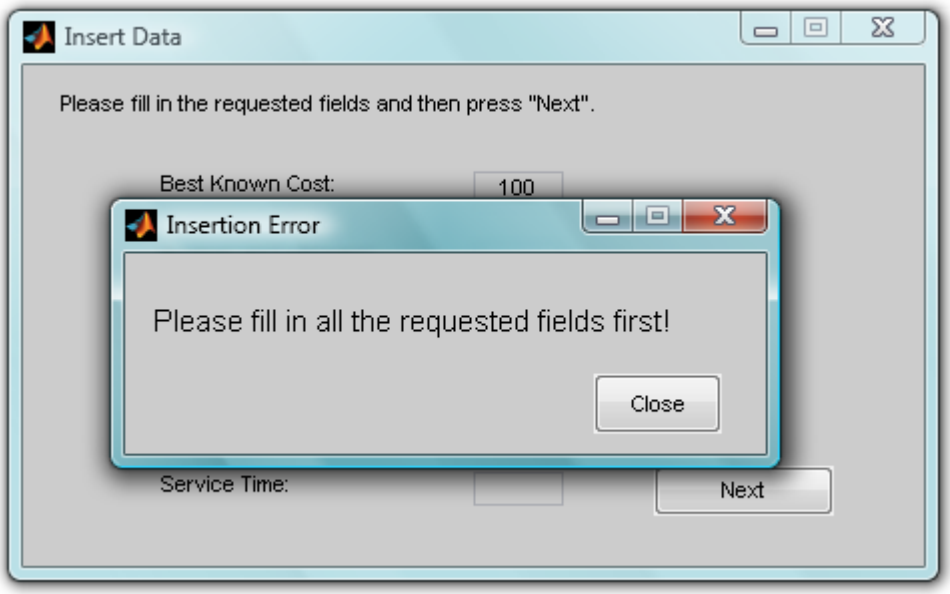
το κουμπί “Insert Data” και προκύπτει το παράθυρα που φαίνεται στην επόμενη εικόνα.



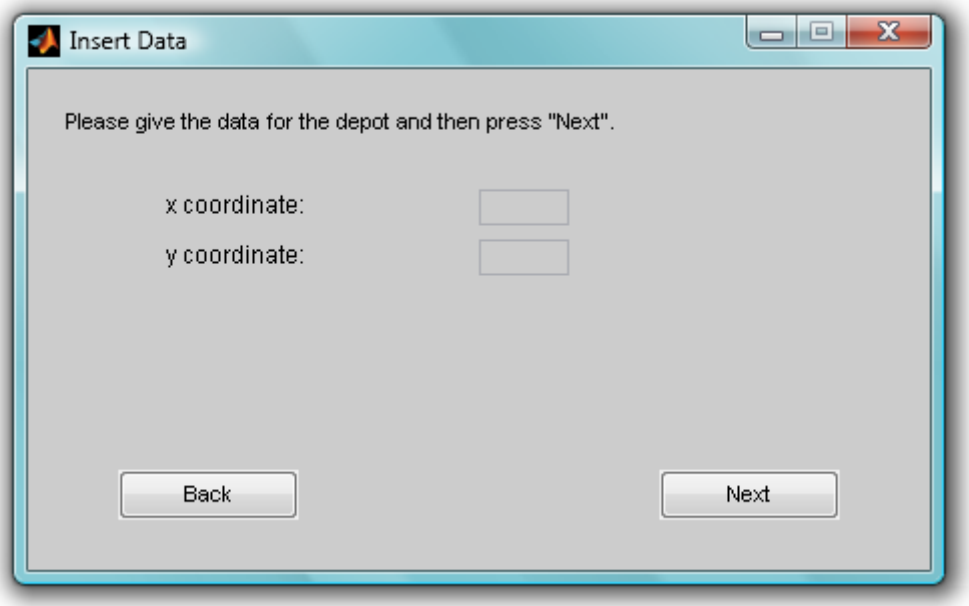
Στη συνέχεια συμπληρώνονται τα απαραίτητα πεδία. Ανάλογα με τα δεδομένα που εισάγονται στο συγκεκριμένο παράθυρο (πχ. αριθμός πελατών) προκύπτουν και τα νέα παράθυρα που απαιτούνται για την εισαγωγή και των υπολοίπων δεδομένων.

Εάν δεν συμπληρωθεί, για παράδειγμα, ο αριθμός πελατών (“Number of Customers”), κ α ο χρήστης επιλ ξει να συνεχίσει τη συμπλήρωση των υπολοίπων δεδομένων, το σύστημα αναγνωρίζει αυτομάτως το λάθος, το αποτρέπει και εμφανίζει ένα μήνυμα λάθους που προτρέπει το χρήστη να συμπληρώσει τα απαραίτητα δεδομένα πρώτα, έτσι ώστε να συμπληρωθούν σωστά τα επόμενα πεδία.

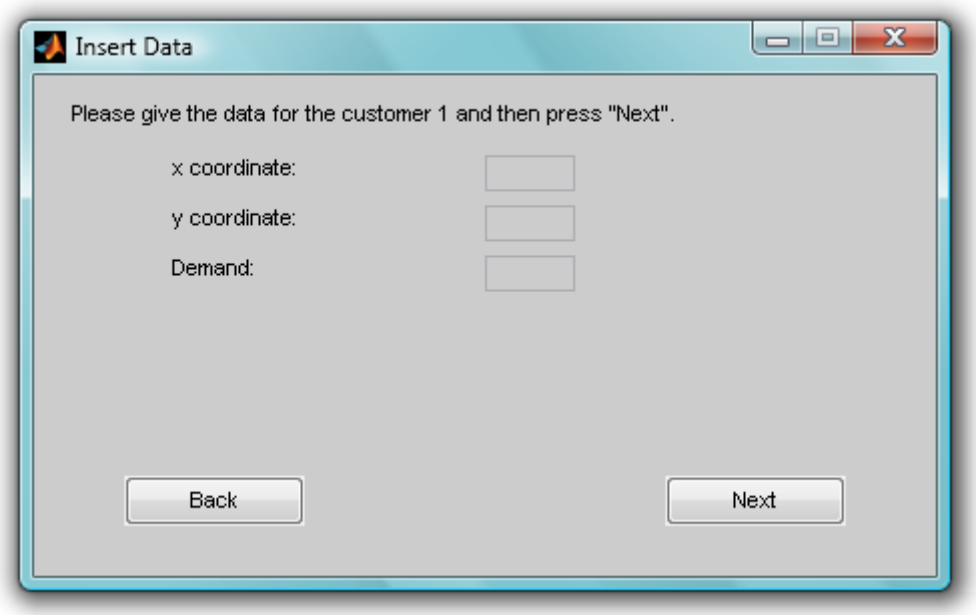
Για παράδειγμα, αν δεν συμπληρωθεί πρώτα ο αριθμός των πελατών, δεν γίνεται να εισαχθούν τα δεδομένα για κάθε πελάτη, για τον απλό λόγο του ότι αν δεν έχει εισαχθεί ο αριθμός τους το σύστημα δεν ‘γνωρίζει’ που θα σταματήσει η διαδικασία εισαγωγής δεδομένων για τους πελάτες. Ένα παράδειγμα μηνύματος λάθους, του τύπου που αναφέρθηκε πριν, είναι αυτό που φαίνεται στην επόμενη εικόνα:



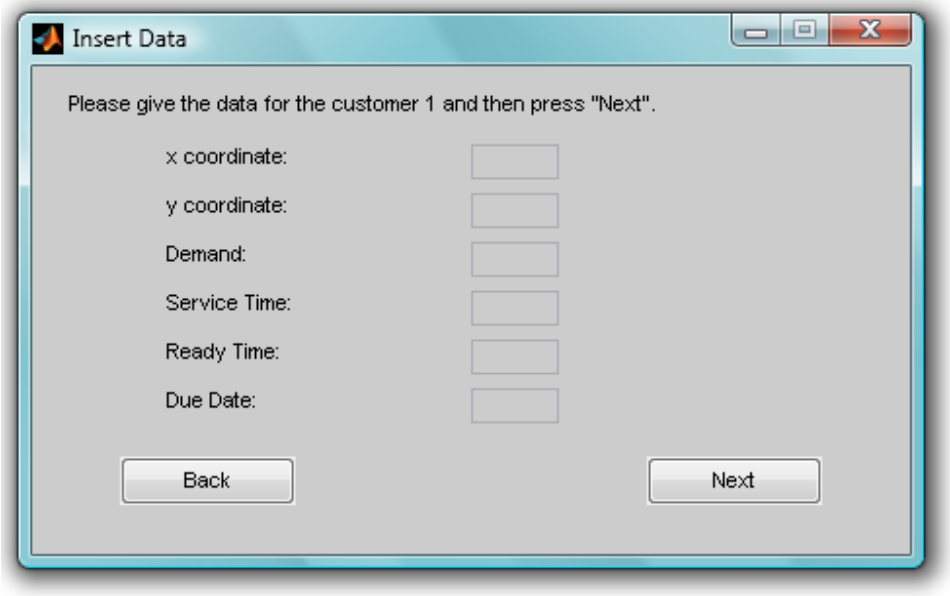
Όταν συμπληρωθούν τα απαραίτητα στοιχεία στο πρώτο παράθυρο, ο χρήστης πατώντας “Next” μεταφέρεται στο παράθυρο για την εισαγωγή των δεδομένων που έχουν να κάνουν με την αποθήκη, (όταν ο τύπος προβλήματος που έχει επιλεγεί είναι ένας από τους TSP, VRP, Open VRP, VRP με Time Windows). Στην περίπτωση του προβλήματος VRP τα μόνα δεδομένα που απαιτούνται για την αποθήκη είναι οι συντεταγμένες θέσης της. Το παράθυρα που προκύπτει σε αυτήν την περίπτωση είναι το εξής:



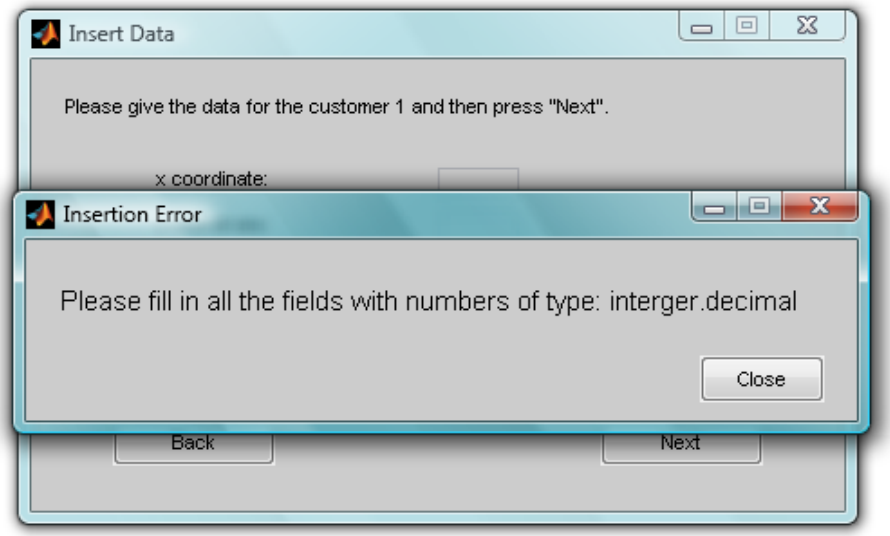
Αξίζει να σημειωθεί ότι καθ' όλη τη διάρκεια της διαδικασίας εισαγωγής δεδομένων ο χρήστης μπορεί να πλοηγηθεί με τα πλήκτρα "Back" και "Next" για να διορθώσει δεδομένα που εισήγαγε πριν ή για να προχωρήσει στην εισαγωγή άλλων δεδομένων. Σε καμία περίπτωση βέβαια η διαδικασία δεν ολοκληρώνεται επιτυχώς (με το αντίστοιχο ενημερωτικό μήνυμα) αν δεν εισαχθούν όλα τα δεδομένα σωστά. Σε αυτό το σημείο αξίζει να σημειωθεί ότι στην περίπτωση των προβλημάτων Multiple Depot VRP όπου υπάρχουν παραπάνω από μία αποθήκες, πρώτα ζητούνται τα δεδομένα για κάθε μία αποθήκη ξεχωριστά και στη συνέχεια τα δεδομένα που έχουν να κάνουν με τους πελάτες. Στην περίπτωση των προβλημάτων τύπου VRP τα δεδομένα για τους πελάτες εισάγονται ως εξής:



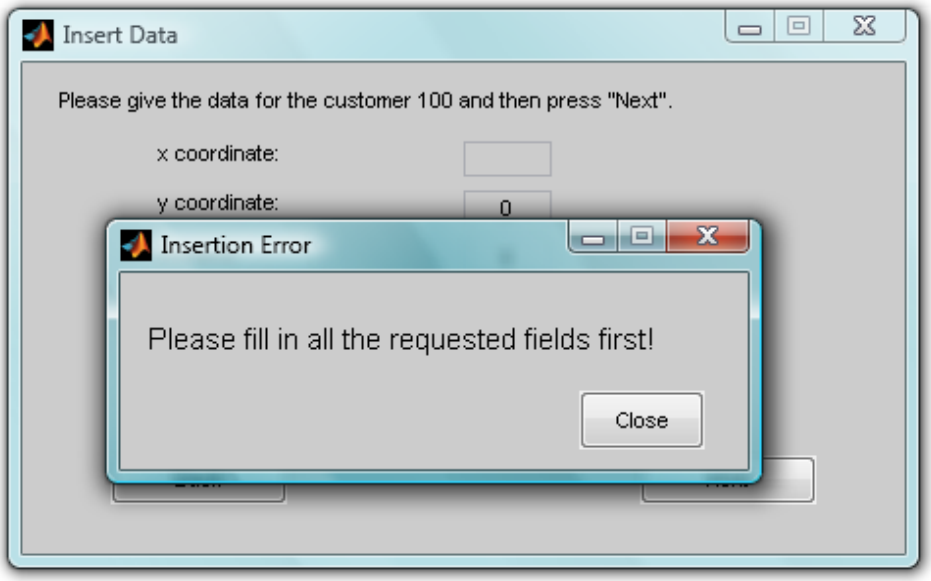
Εάν είχε επιλεγεί ο τύπος προβλημάτων VRP με Time Windows, τα δεδομένα για κάθε πελάτη θα εισάγονταν ως εξής:



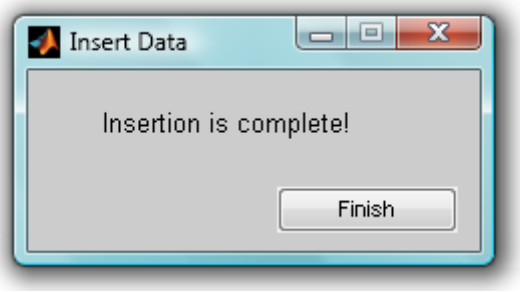
Σε οποιοδήποτε πεδίο η καταχώρηση της τιμής που έχει πληκτρολογηθεί γίνεται μόλις ο χρήστης πατήσει κάπου αλλού τον κέρσορα, τότε γίνεται αυτόματα η καταχώρηση αν είναι σωστή. Κατά την εισαγωγή των δεδομένων όμως, είναι πολύ εύκολο να γίνει κάποια λάθος πληκτρολόγηση. Εάν για παράδειγμα, αντί για τον αριθμό 4 ο χρήστης κατά λάθος πληκτρολογήσει το χαρακτήρα ‘e’ που βρίσκεται κάτω από το κουμπί του αριθμού 4, όταν επιχειρήσει να καταχωρήσει την συγκεκριμένη τιμή, το σύστημα θα αποτρέψει το λάθος του χρήστη και θα εμφανίσει ένα μήνυμα που τον προτρέπει να εισάγει τη σωστή τιμή, όπως φαίνεται στην παρακάτω εικόνα:



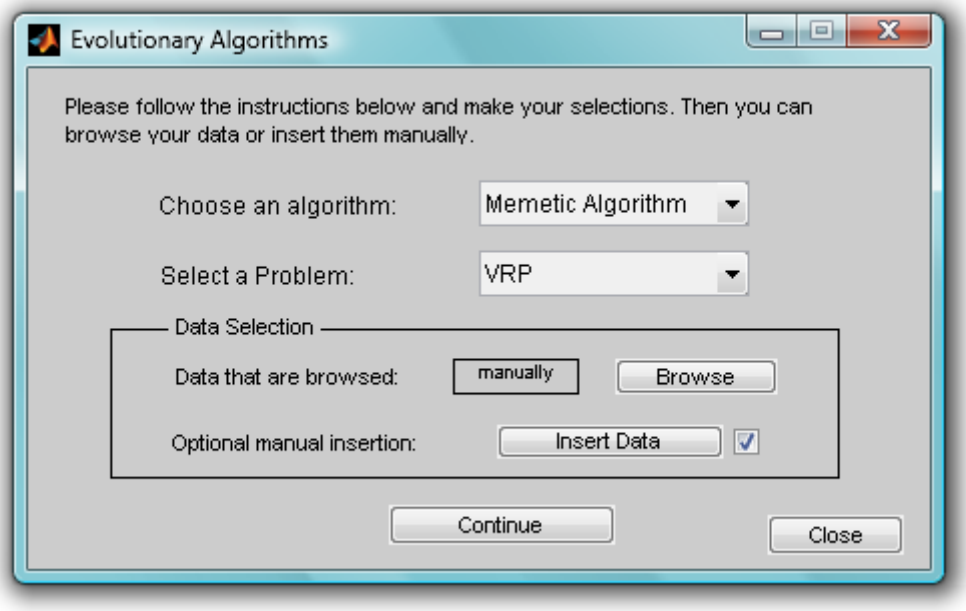
Εάν στο παράθυρα που αντιστοιχεί στον τελευταίο πελάτη ο χρήστης επιλέξει “Next”, και εάν δεν έχουν καταχωρηθεί τα δεδομένα σε όλα τα πεδία, θα εμφανιστεί το εξής μήνυμα λάθους:



Στην περίπτωση που ο χρήστης εισάγει όλα τα δεδομένα σωστά, εμφανίζεται το παρακάτω ενημερωτικό μήνυμα:



Επίσης στο πεδίο “Data that are browsed” εμφανίζεται η ένδειξη “manually”, πράγμα που υποδεικνύει ότι χρησιμοποιήθηκε και ολοκληρώθηκε επιτυχώς η διαδικασία σταδιακής χειροκίνητης εισαγωγής δεδομένων, πράγμα που φαίνεται κ α από το check-box που βρίσκεται δίπλα από το κουμπί “Insert Data”. Το συγκεκριμένο check-box έχει το σήμα (v, tick), πράγμα που φαίνεται και στην επόμενη εικόνα:



Ο λόγος ύπαρξης του συγκεκριμένου check-box δεν είναι μόνο για να επιβεβαιωθεί η επιτυχής σταδιακή χειροκίνητη εισαγωγή δεδομένων. Χρησιμεύει και στην περίπτωση που ο χρήστης μετά από μια επιτυχή σταδιακή χειροκίνητη εισαγωγή δεδομένων, μετανιώσει και επιθυμήσει να φορτώσει δεδομένα από ένα αρχείο. Σε αυτήν την περίπτωση φορτώνει τα δεδομένα από το αρχείο που επιθυμεί (πατώντας “Browse” και επιλέγοντάς το

συγκεκριμένο αρχείο) και στη συνέχεια αφαιρεί το σήμα (v, tick) από το συγκεκριμένο check-box.

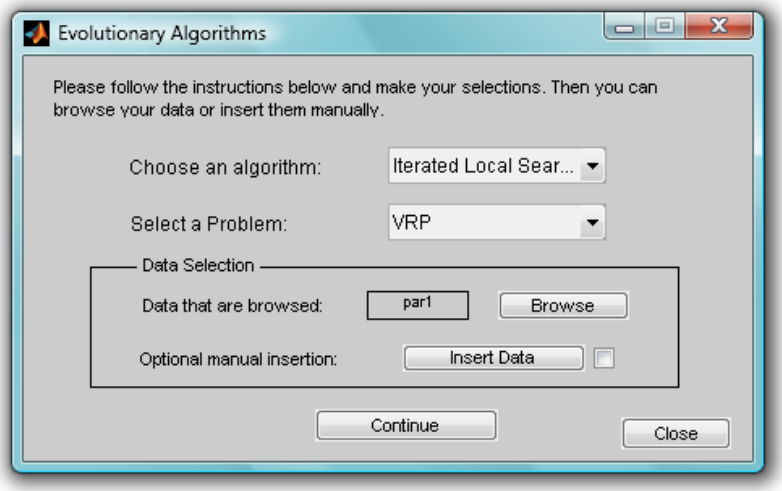
Εάν για κάποιο λόγο ο χρήστης πληκτρολογήσει λάθος αριθμό πελατών ή αριθμό αποθηκών (για την περίπτωση των προβλημάτων Multiple Depot VRP), δεν χρειάζεται να πληκτρολογήσει ξανά όλα τα δεδομένα από την αρχή.

Για παράδειγμα, εάν ο χρήστης έχει δώσει λανθασμένα αριθμό πελατών 110 ενώ οι πελάτες είναι 100 και καταλάβει το λάθος του αφού έχει εισάγει τα δεδομένα και για τους 100 πελάτες. Υπάρχει η δυνατότητα να διορθώσει το 110 σε 100 χωρίς να χρειάζεται να αλλάξει κάτι άλλοή να πληκτρολογήσει κάτι ξανά.

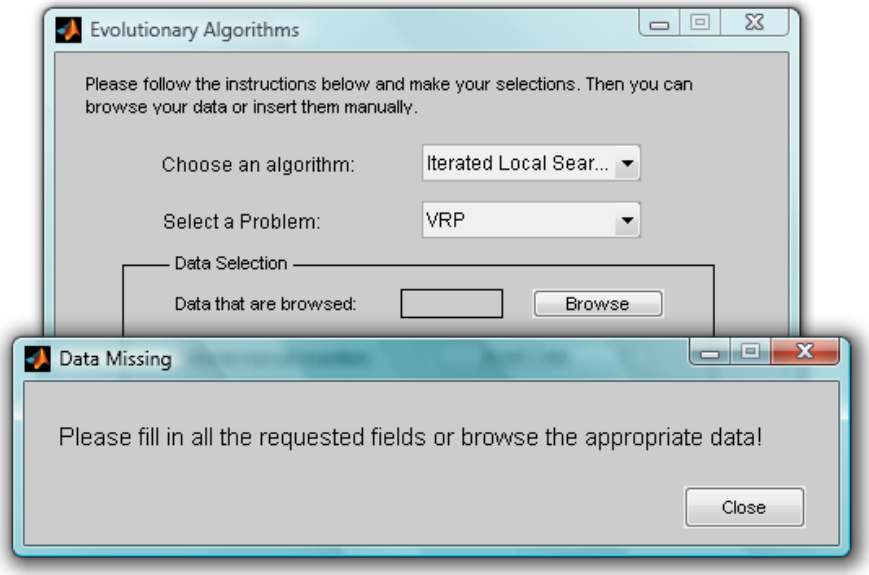
Αντίστοιχα αν γίνει το αντίθετο λάθος και ο χρήστης έχει εισάγει στοιχεία για 100 πελάτες, και μετά την καταχώρηση του συνόλου των στοιχείων (όταν δηλαδή έχει φύγει από το interface για την σταδιακή χειροκίνητη εισαγωγή δεδομένων) επιθυμεί να εισάγει στοιχεία για άλλους 10 πελάτες. Τότε μπορεί να εισέλθει ξανά στο interface για την σταδιακή χειροκίνητη εισαγωγή δεδομένων και να εισάγει απλά τα στοιχεία για τους επιπλέον πελάτες, τα υπόλοιπα θα έχουν ήδη αποθηκευθεί.

4.4 Επίλυση Προβλήματος

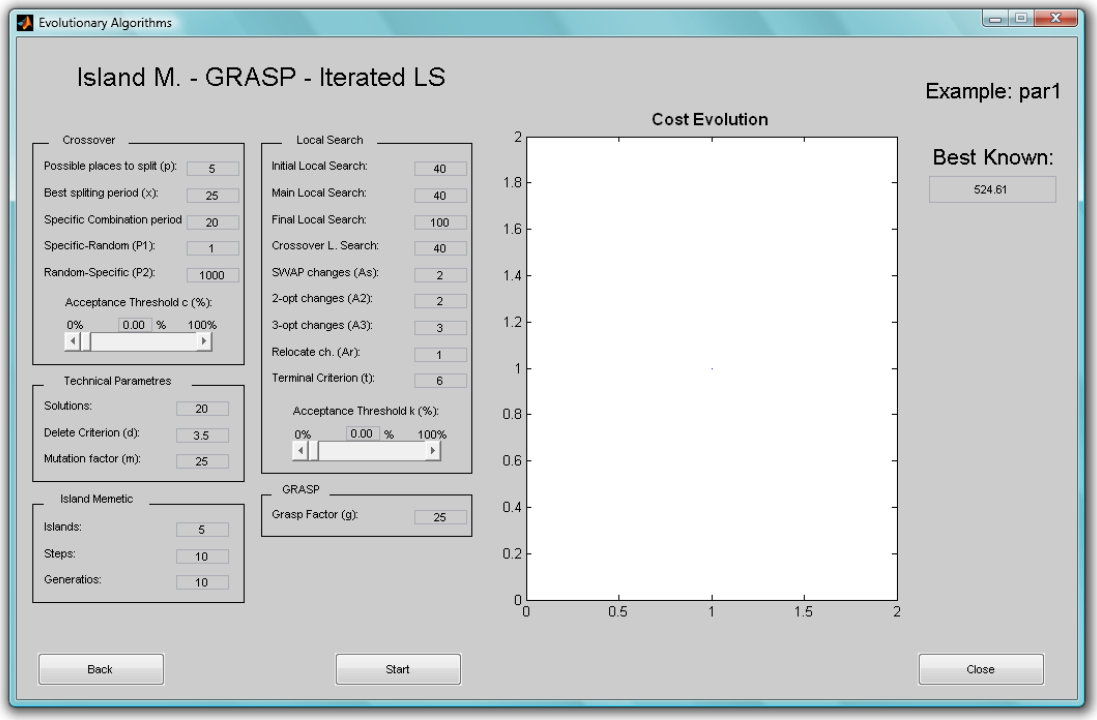
Έστω ότι επιλέγεται να λυθεί ένα πρόβλημα τύπου VRP, με τη μέθοδο Iterated Local Search in Island Memetic with GRASP. Έστω επίσης ότι το πρόβλημα είναι τυποποιημένο instance: par1, το οποίο είναι ένα από τα προβλήματα που χρησιμοποιούνται σε παγκόσμια κλίμακα για τη σύγκριση αλγορίθμων προβλημάτων εφοδιαστικής αλυσίδας. Το παράθυρο επιλογής προβλήματος, μεθόδου και δεδομένων θα είναι ως εξής:



Στη συνέχεια ο χρήστης επιλέγει “Continue” και μεταφέρεται στο επόμενο παράθυρο. Αξίζει να σημειωθεί ότι εάν δεν έχουν συμπληρωθεί όλα τα απαραίτητα πεδία, δεν ανοίγει το επόμενο παράθυρο και αντί για αυτό εμφανίζεται το εξής μήνυμα:



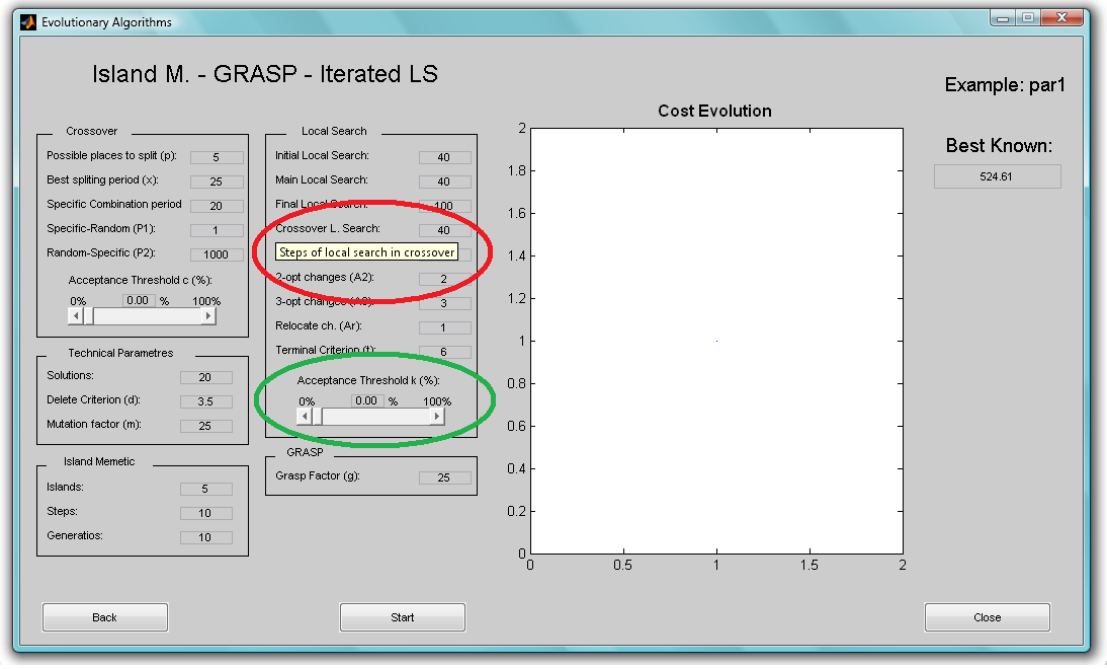
Το επόμενο παράθυρο έχει ως εξής:



Όπως φαίνεται στο παραπάνω παράθυρο, το όνομα του αλγορίθμου που θα χρησιμοποιηθεί αναγράφεται στην πάνω αριστερή πλευρά. Στην πάνω δεξιά πλευρά αναγράφεται το όνομα του παραδείγματος που επιλύεται (για την περίπτωση που λύνεται κάποιο ευρέως διαδεδομένο πρόβλημα – instance). Στην περίπτωση όπου λύνεται ένα συγκεκριμένο πρόβλημα του οποίου τα δεδομένα έχουν εισαχθεί με τον $2^{ο}$ εναλλακτικό τρόπο (της σταδιακής χειροκίνητης εισαγωγής), στην πάνω δεξιά πλευρά του παραθύρου αναγράφεται “Specific”.

Ακριβώς κάτω από το όνομα του προβλήματος αναγράφεται το κόστος της μέχρις στιγμής καλύτερης λύσης που έχει βρεθεί για το πρόβλημα που λύνεται.

Στο αριστερό μέρος του παραθύρου βρίσκονται όλες οι παράμετροι που μπορεί να αλλάξει ο χρήστης. Οι συγκεκριμένες παράμετροι χρησιμοποιούνται στον αλγόριθμο βελτιστοποίησης και είναι μερικές από τις σημαντικότερες, όσον αφορά στην επιρροή τους ως προς την αποτελεσματικότητα του αλγορίθμου. Υπάρχει η δυνατότητα να χρησιμοποιούνται πάντα οι βέλτιστες τιμές των παραμέτρων και να μην αφήνεται στο χρήστη η ευθύνη επιλογής των τιμών τους. Πολλές φορές όμως η βέλτιστη τιμή κάποιας παραμέτρου εξαρτάται και από το πρόβλημα το οποίο επιλύεται. Επειδή λοιπόν υπάρχει η δυνατότητα επίλυσης οπουδήποτε προβλήματος (και οποιουδήποτε μεγέθους φυσικά), υπάρχει και η δυνατότητα της αλλαγής των τιμών ορισμένων παραμέτρων, οι οποίες είναι μερικές από τις σημαντικότερες και πιο ευπαθείς (όσον αφορά τη μεταβλητότητα της βέλτιστης τιμής τους ανάλογα με το πρόβλημα που επιλύεται). Με αυτόν τον τρόπο ο κάθε χρήστης μπορεί εύκολα να πειραματιστεί και να προσεγγίσει τις βέλτιστες τιμές των παραμέτρων για το εκάστοτε πρόβλημα.



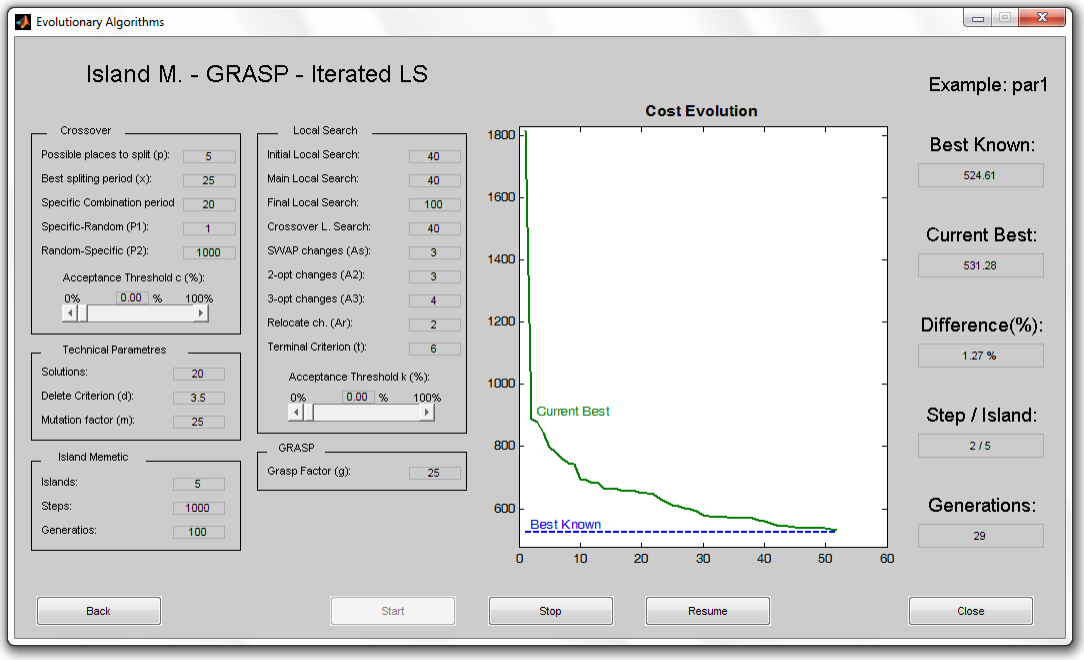
Στην παραπάνω εικόνα φαίνεται ότι ακόμα και στην περίπτωση που ο χρήστης δεν είναι ειδικός σε θέματα βελτιστοποίησης προβλημάτων εφοδιαστικής αλυσίδας, μπορεί να χρησιμοποιήσει το παρόν σύστημα, και να πειραματιστεί με τις τιμές των παραμέτρων που αναφέρθηκαν πριν. Αυτό συμβαίνει επειδή δίνονται ειδικές οδηγίες ξεχωριστά για κάθε παράμετρο με τη βοήθεια εργαλείων τύπου “tooltip”, όπως αυτό που βρίσκεται στον κόκκινο κύκλο της προηγούμενης εικόνας.

Για τη διαχείριση παραμέτρων που εκφράζονται σε ποσοστά %, έχουν εισαχθεί ειδικά διαμορφωμένα “sliders” (όπως αυτό στον πράσινο κύκλο της προηγούμενης εικόνας). Τα συγκεκριμένα sliders κυμαίνονται από το 0% έως το 100%, και η τιμή τους μπορεί να εισαχθεί είτε πληκτρολογώντας το νούμερο στο πεδίο πάνω από το slider (κυρίως για τις περιπτώσεις που απαιτείται μεγάλη ακρίβεια), είτε χρησιμοποιώντας τα βέλη ή την μπάρα του slider.

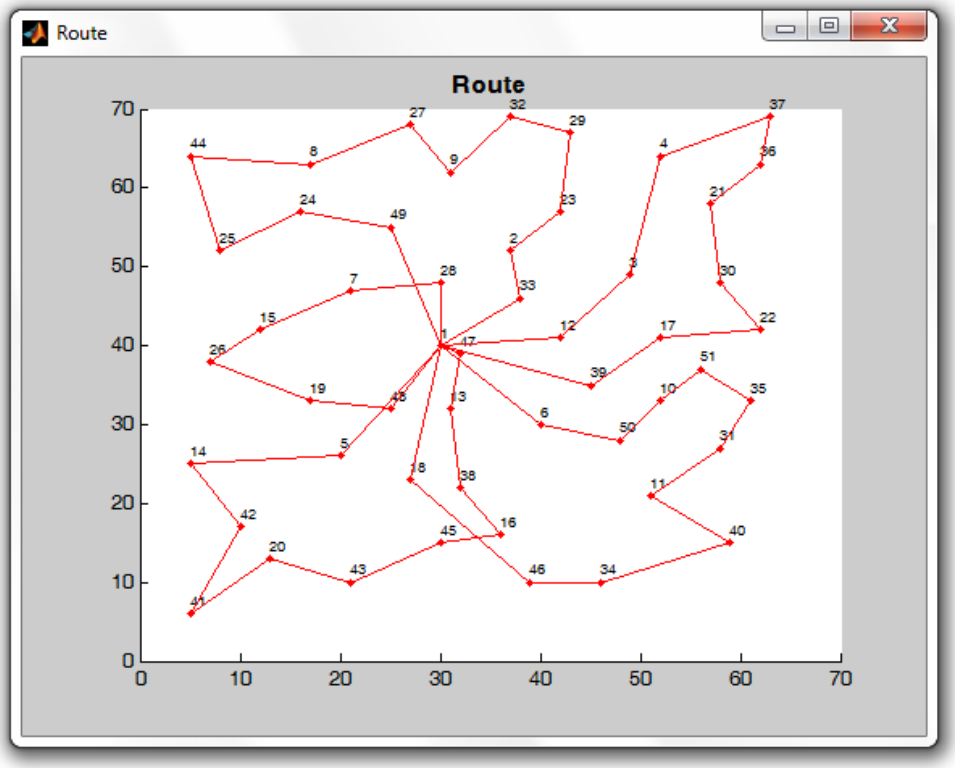
Για την ευκολότερη χρήση του συστήματος οι παράμετροι έχουν ομαδοποιηθεί. Έτσι όπως φαίνεται και στην παραπάνω εικόνα, οι παράμετροι έχουν χωριστεί σε 5 ομάδες. Οι συγκεκριμένες ομάδες έχουν ως εξής:

1. Crossover (Διασταύρωση)
2. Local Search (Τοπική Αναζήτηση)
3. GRASP (Greedy Randomized Adaptive Search Procedure, Άπληστη Τυχοποιημένη Προσαρμοστική Αναζήτηση)
4. Island Memetic (παράμετροι μιμητικού αλγορίθμου πολλαπλών πληθυσμών – νησιών)
5. Technical Parameters (λοιπές τεχνικές – προγραμματιστικές παράμετροι)

Εάν ο χρήστης διαπιστώσει ότι έπρεπε να εισάγει διαφορετικά δεδομένα στο προηγούμενο παράθυρο, υπάρχει η δυνατότητα να επιστρέψει στο προηγούμενο βήμα πατώντας το πλήκτρο “Back”. Για να ξεκινήσει η διαδικασία βελτιστοποίησης ο χρήστης απλά πατάει το πλήκτρο “Start”.



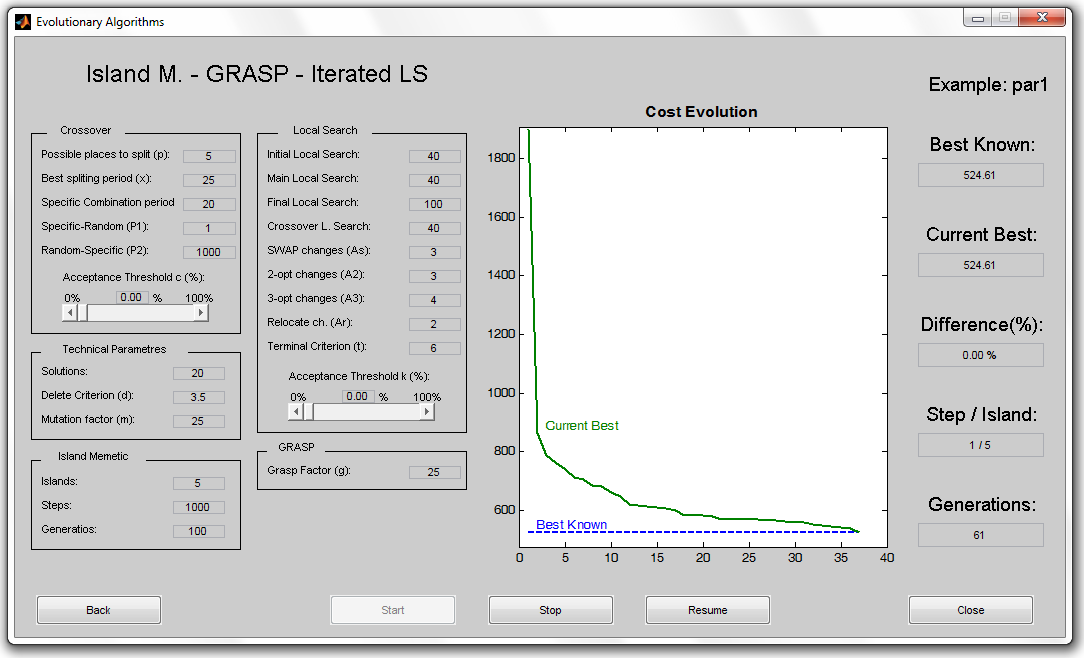
Στο παραπάνω παράθυρο φαίνεται μια ενδιάμεση κατάσταση την ώρα που γίνεται η αναζήτηση. Εάν ο χρήστης πατήσει το πλήκτρο “Stop”, γίνεται παύση στην εκτέλεση του προγράμματος, αποθηκεύεται σε ένα αρχείο excel η μέχρις στιγμής καλύτερη λύση και σχεδιάζεται η μέχρις στιγμής βέλτιστη λύση για την πλήρη απεικόνιση της κατάστασης μέχρι τη στιγμή που πατήθηκε το “Stop”. Για την περίπτωση του παραπάνω σχήματος σχεδιάστηκε η εξής λύση:



Στη συνέχεια ο χρήστης έχει την επιλογή να πατήσει “Resume” (ή να κλείσει απλά το σχήμα που απεικονίζει τη λύση) και να συνεχιστεί η αναζήτηση από το σημείο που σταμάτησε.

Επίσης οποιαδήποτε στιγμή το θελήσει ο χρήστης μπορεί να επιστρέψει πίσω για να αλλάξει οτιδήποτε επιθυμεί πατώντας το πλήκτρο “Back”, ή ακόμα και να κλείσει το πρόγραμμα πατώντας το πλήκτρο “Close”. Και στις δύο περιπτώσεις (επειδή διακόπτεται η αναζήτηση) η βέλτιστη λύση που έχει βρεθεί αποθηκεύεται σε ένα αρχείο excel και επίσης σχεδιάζεται σε ένα figure όπως αυτό της προηγούμενης εικόνας.

Στο παρακάτω σχήμα φαίνεται η περίπτωση όπου το σύστημα που κατασκευάστηκε στα πλαίσια της παρούσας μεταπτυχιακής διατριβής εντόπισε την καλύτερη λύση που έχει βρεθεί μέχρι στιγμής σε παγκόσμια κλίμακα. Όπου στο συγκεκριμένο παράδειγμα (par1) το κόστος της συγκεκριμένης λύσης είναι 524.61, όπως φαίνεται και στο πρώτο πεδίο (το “Best Known” δηλαδή), το οποίο βρίσκεται στη δεξιά πλευρά του παραθύρου.

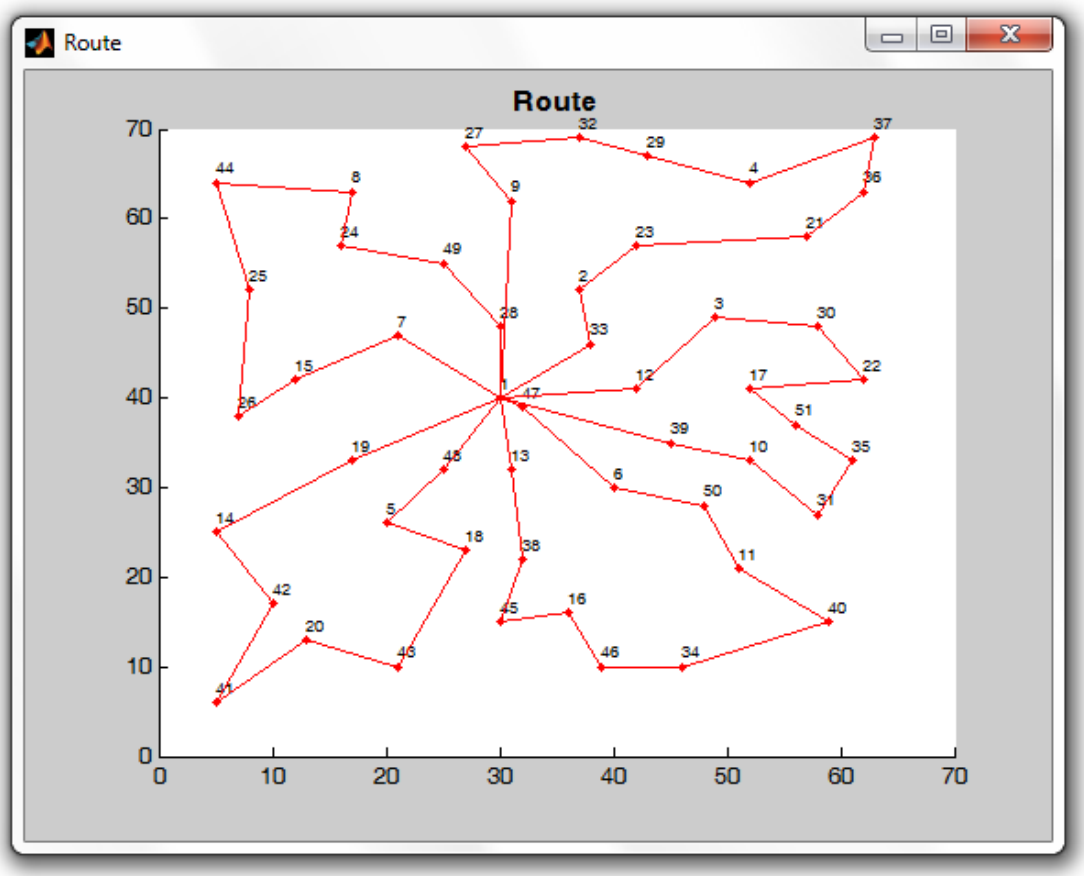


Όσον αφορά τα υπόλοιπα πεδία που βρίσκονται στη δεξιά πλευρά του παραθύρου, το δεύτερο (δηλ. το “Current Best”) περιέχει το κόστος της μέχρι στιγμής καλύτερης λύσης που έχει υπολογιστεί από το σύστημα από τη στιγμή που άρχισε η αναζήτηση. Το πεδίο “Difference(%)” περιέχει την % διαφορά της καλύτερης λύσης, που έχει υπολογιστεί από τη στιγμή που άρχισε να τρέχει το σύστημα, με την καλύτερη λύση που έχει βρεθεί μέχρι στιγμής σε παγκόσμια κλίμακα.

Στο επόμενο πεδίο (“Step/Island ”) φαίνεται το βήμα (step) στο οποίο βρίσκεται ο αλγόριθμος και σε ποια ομάδα λύσεων – (ή αλλιώς “νησί-island”) γίνονται οι υπολογισμοί την παρούσα στιγμή. Τέλος στο πεδίο “Generations” φαίνεται σε ποια γενιά έχει φτάσει μέχρι στιγμής η αναζήτηση (για τη συγκεκριμένη ομάδα λύσεων και για το συγκεκριμένο βήμα φυσικά).

Το σχήμα του κεντρικού παραθύρου (“Cost Evolution”), την εξέλιξη, στο χρόνο, του κόστους της καλύτερης λύσης της παρούσας διαδικασίας αναζήτησης. Αυτό φαίνεται από την πράσινη γραφική παράσταση, η οποία όπως φαίνεται και από την εικόνα έχει το όνομα “Current Best”. Η μπλε ευθεία αναπαριστά το κόστος της μέχρις στιγμής καλύτερης λύσης που έχει βρεθεί σε παγκόσμια κλίμακα και έχει αυτή τη μορφή που φαίνεται στο παραπάνω σχήμα για να φαίνεται και γραφικά που πρέπει να συγκλίνει ο αλγόριθμος.

Η γραφική αναπαράσταση της βέλτιστης λύσης (με 0% απόκλιση από το παγκόσμιο βέλτιστο) που βρέθηκε στην παραπάνω περίπτωση είναι η εξής:



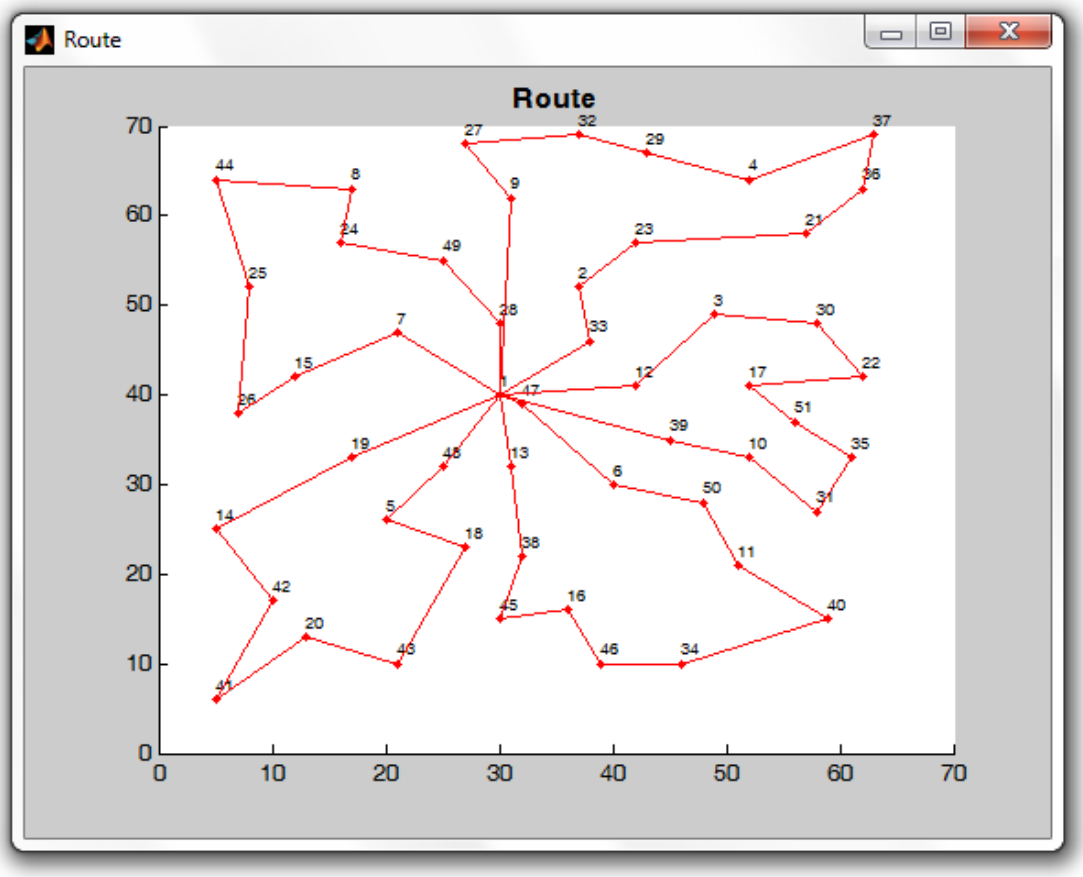
Σε κάθε κόμβο αντιστοιχεί ένας αριθμός, ο οποίος είναι ο αύξων αριθμός του εκάστοτε κόμβου. Στο συγκεκριμένο παράδειγμα ο κόμβος 1 αντιστοιχεί στην αποθήκη και οι υπόλοιποι στους πελάτες.

5. Παρουσίαση Αποτελεσμάτων

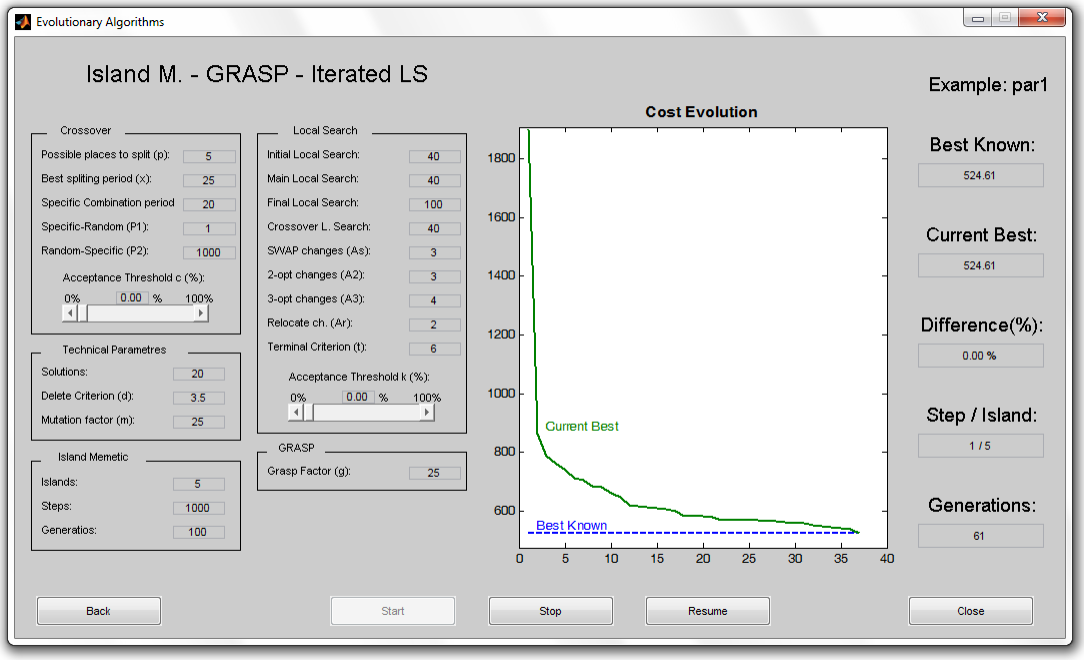
5.1 Εισαγωγή

Το σύστημα που κατασκευάστηκε στα πλαίσια της παρούσας μεταπτυχιακής διατριβής, είναι απόλυτα φιλικό προς το χρήστη, του δίνει οδηγίες σε κάθε του κίνηση και αποτρέπει κάθε πιθανό του λάθος. Σε κάθε περίπτωση λάθους κίνησης του χρήστη εκτός από την αποτροπή του λάθους, το σύστημα αυτόματα δίνει ξανά άλλες οδηγίες στο χρήστη για το πώς θα δράσει σωστά στην εκάστοτε περίπτωση.

Τα αποτελέσματα του εκάστοτε προβλήματος παρουσιάζονται με την παρακάτω μορφή (πχ για το παράδειγμα par1):



Η εξέλιξη του κόστους καθ' όλη τη διάρκεια της αναζήτησης, καθώς και η πλήρης περιγραφή της κατάστασης στην οποία βρίσκεται ο αλγόριθμος την εκάστοτε στιγμή, φαίνεται από το παρακάτω παράθυρο:



5.2 Συγκεντρωτικά Αποτελέσματα

VRP:

Όσον αφορά στα αποτελέσματα που βρέθηκαν για το πρόβλημα VRP, η μέση απόκλιση από το παγκόσμιο βέλτιστο είναι 1,31%. Τα αποτελέσματα για το συγκεκριμένο πρόβλημα παρουσιάζονται συνοπτικά στον επόμενο πίνακα:

ΠΑΡΑΔΕΙΓΜΑ	ΚΟΜΒΟΙ	ΧΩΡΗΤΙΚΟΤΗΤΑ ΦΟΡΤΗΓΩΝ	ΠΕΡΙΟΡΙΣΜΟΣ ΔΙΑΔΡΟΜΗΣ	ΜΕΧΡΙ ΣΤΙΓΜΗΣ ΒΕΛΤΙΣΤΟ	ΑΠΟΤΕΛΕΣΜΑ ΑΛΓΟΡΙΘΜΩΝ ΜΑΣ	ΑΠΟΚΛΙΣΗ(%) ΑΠΟ ΤΟ ΜΕΧΡΙ ΣΤΙΓΜΗΣ ΒΕΛΤΙΣΤΟ
1	51	160	-	524,61	**524,61**	**0%**
2	76	140	-	835,26	835,77	0,06%
3	101	200	-	826,14	832,27	0,74%
4	151	200	-	1028,42	1057,24	2,80%
5	200	200	-	1291,29	1355,69	4,96%
6	51	160	200	555,43	**555,43**	**0%**
7	76	140	160	909,68	921,09	1,25%
8	101	200	230	865,94	871,71	0,67%
9	151	140	160	1162,55	1186,36	2,05%
10	200	200	200	1395,85	1452,39	4,05%
11	101	200	-	1042,11	**1042,11**	**0%**
12	121	200	-	819,56	**819,5575**	**0%**
13	101	200	720	1541,14	1563,8016	1,47%
14	121	200	1040	866,37	867,17	0,09%

OVRP: Για το πρόβλημα OVRP τα αποτελέσματα έχουν ως εξής:

ΠΑΡΑΔΕΙΓΜΑ	ΚΟΜΒΟΙ	ΧΩΡΗΤΙΚΟΤΗΤΑ ΦΟΡΤΗΓΩΝ	ΠΕΡΙΟΡΙΣΜΟΣ ΔΙΑΔΡΟΜΗΣ	ΜΕΧΡΙ ΣΤΙΓΜΗΣ ΒΕΛΤΙΣΤΟ	ΑΠΟΤΕΛΕΣΜΑ ΑΛΓΟΡΙΘΜΩΝ ΜΑΣ	ΑΠΟΚΛΙΣΗ(%) ΑΠΟ ΤΟ ΜΕΧΡΙ ΣΤΙΓΜΗΣ ΒΕΛΤΙΣΤΟ
1	51	160	-	412,95	**412,9568**	**0,00%**
2	76	140	-	564,06	568,6032	0,81%
3	101	200	-	639,25	663,6062	3,81%
4	151	200	-	733,13	795,2329	8,47%
5	200	200	-	868,44	950,8875	9,49%
6	51	160	200	412,95	**412,9568**	**0,00%**
7	76	140	160	566,93	570,7721	0,68%
8	101	200	230	640,89	694,9458	8,43%
9	151	140	160	741,44	790,6126	6,63%
10	200	200	200	871,58	938,0832	7,63%
11	101	200	-	678,54	693,6481	2,23%
12	121	200	-	534,24	534,7104	0,09%
13	101	200	720	836,55	885,0669	5,80%
14	121	200	1040	552,64	555,7429	0,56%

TSP: Για το πρόβλημα TSP τα αποτελέσματα έχουν ως εξής:

ΠΑΡΑΔΕΙΓΜΑ	ΚΟΜΒΟΙ	ΜΕΧΡΙ ΣΤΙΓΜΗΣ ΒΕΛΤΙΣΤΟ	ΑΠΟΤΕΛΕΣΜΑ ΑΛΓΟΡΙΘΜΩΝ ΜΑΣ	ΑΠΟΚΛΙΣΗ(%) ΑΠΟ ΤΟ ΜΕΧΡΙ ΣΤΙΓΜΗΣ ΒΕΛΤΙΣΤΟ
eil	51	426	428,8718	0,67%
berlin52	52	7542	7716,6859	2,26%
eil76	76	538	552,0201	2,61%
kroA100	100	21282	21322,7366	0,19%
d198	198	15780	16214,0126	2,75%
kroA200	200	29368	30514,3481	3,90%

VRPTW: Για το πρόβλημα VRPTW τα αποτελέσματα έχουν ως εξής:

ΠΑΡΑΔΕΙΓΜΑ	ΚΟΜΒΟΙ	ΧΩΡΗΤΙΚΟΤΗΤΑ ΦΟΡΤΗΓΩΝ	ΜΕΧΡΙ ΣΤΙΓΜΗΣ ΒΕΛΤΙΣΤΟ	ΑΠΟΤΕΛΕΣΜΑ ΑΛΓΟΡΙΘΜΩΝ ΜΑΣ	ΑΠΟΚΛΙΣΗ(%) ΑΠΟ ΤΟ ΜΕΧΡΙ ΣΤΙΓΜΗΣ ΒΕΛΤΙΣΤΟ
C101	100	200	828,94	**828,9368**	**0%**
C102	100	200	828,94	848,6984	2,38%
C103	100	200	828,06	832,1807	0,50%
C104	100	200	824,78	835,9920	1,36%
C105	100	200	828,94	857,2659	3,42%
C106	100	200	828,94	891,9596	7,60%

5.3 Αναλυτικά αποτελέσματα για το Πρόβλημα Δρομολόγησης Οχημάτων (VRP) και το Ανοικτό Πρόβλημα Δρομολόγησης Οχημάτων (OVRP)

Για τα προβλήματα VRP και OVRP χρησιμοποιήθηκαν τα ίδια παραδείγματα (par1 - par14), που χρησιμοποιούνται παγκοσμίως για τη σύγκριση αλγορίθμων επίλυσης προβλημάτων εφοδιαστικής αλυσίδας. Κάθε κόμβος (πελάτη ή αποθήκης) αντιστοιχεί σε συγκεκριμένες συντεταγμένες θέσης και σε συγκεκριμένη ζήτηση (η ζήτηση για τις αποθήκες φυσικά είναι ίση με 0). Σε όλα αυτά τα παραδείγματα ο κόμβος 1 αντιστοιχεί στην αποθήκη. Αναλυτικά για κάθε παράδειγμα οι καλύτερες λύσεις για το πρόβλημα VRP και το OVRP έχουν ως εξής:

5.3.1 Παράδειγμα 1

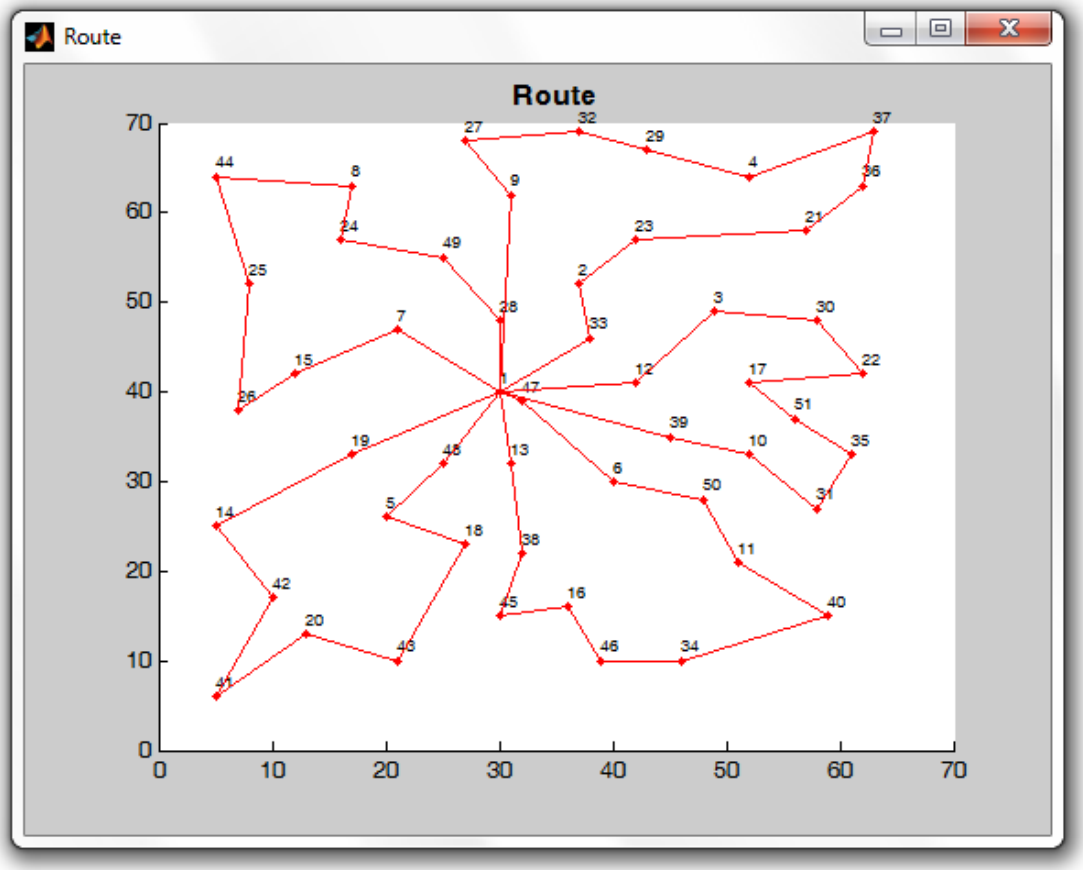
Στο συγκεκριμένο παράδειγμα υπάρχουν 51 κόμβοι. Δεν υπάρχουν χρονικοί περιορισμοί και όλα τα διαθέσιμα φορτηγά έχουν την χωρητικότητα ίση με 160.

VRP:

Η βέλτιστη λύση λοιπόν που βρέθηκε για το συγκεκριμένο παράδειγμα έχει κόστος ίσο με 524,6111 (ίδιο με αυτό της καλύτερης λύσης που έχει βρεθεί σε παγκόσμια κλίμακα). Η συγκεκριμένη λύση, αποτελείται από πέντε διαδρομές που ξεκινούν από και καταλήγουν στον κόμβο 1 ο οποίος αντιστοιχεί στην αποθήκη. Οι διαδρομές αυτές είναι οι εξής:

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

Γραφικά η λύση έχει ως εξής:



OVRP:

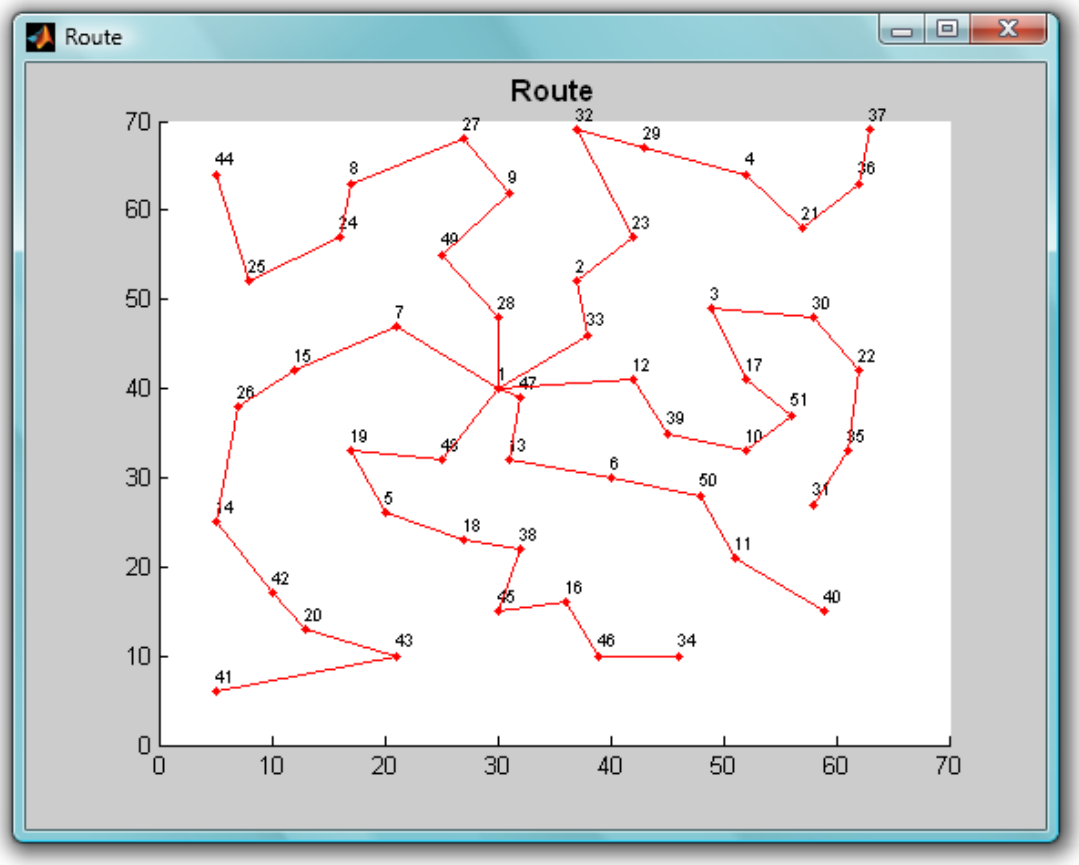
Η λύση που προέκυψε έχει κόστος 412,9568 και η απόκλισή της από το μέχρι στιγμής βέλτιστο (412,95) είναι 0,00%:

$$\frac{412{,}9568 - 412{,}95}{412{,}95} = 0{,}00\ \%$$

Η συγκεκριμένη λύση αποτελείται από 6 διαδρομές οι οποίες έχουν ως εξής:

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

Γραφικά η συγκεκριμένη λύση έχει ως εξής:



5.3.2 Παράδειγμα 2

Στο παράδειγμα 2, υπάρχουν 76 κόμβοι. Όλα τα διαθέσιμα φορτηγά έχουν χωρητικότητα ίση με 140 και δεν υπάρχουν χρονικοί περιορισμοί.

VRP:

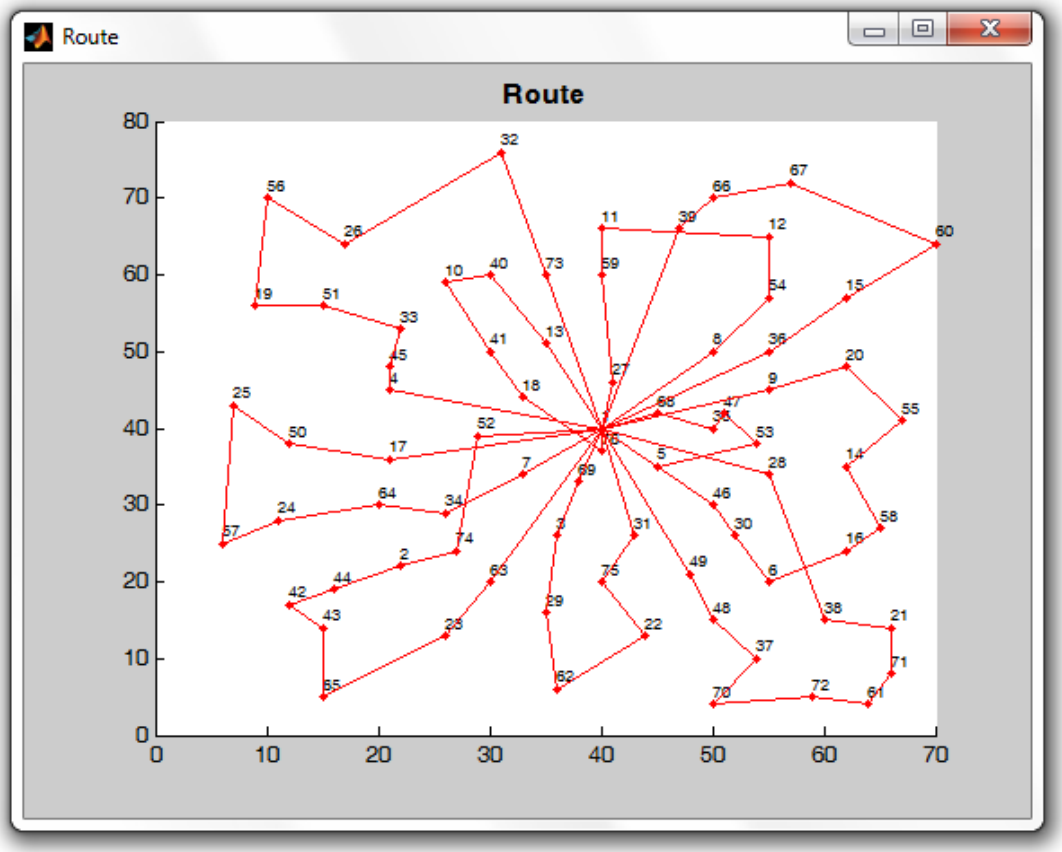
Η λύση που βρέθηκε είχε κόστος 835,77 και η απόκλισή της από το μέχρι στιγμής βέλτιστο σε παγκόσμια κλίμακα (835,26) είναι η εξής:

$$\frac{835{,}77 - 835{,}26}{835{,}26} = 0{,}06\%$$

Η συγκεκριμένη λύση αποτελείται από δέκα διαδρομές οι οποίες έχουν ως εξής:

1	31	75	22	62	29	3	69	1			
1	9	20	55	14	58	16	6	30	46	1	
1	4	45	33	51	19	56	26	32	73	1	
1	13	40	10	41	18	76	1				
1	7	34	64	24	57	25	50	17	1		
1	63	23	65	43	42	44	2	74	52	1	
1	36	15	60	67	66	39	1				
1	8	54	12	11	59	27	1				
1	68	35	47	53	5	1					
1	28	38	21	71	61	72	70	37	48	49	1

Η γραφική απεικόνιση της συγκεκριμένης λύσης έχει ως εξής:



OVRP:

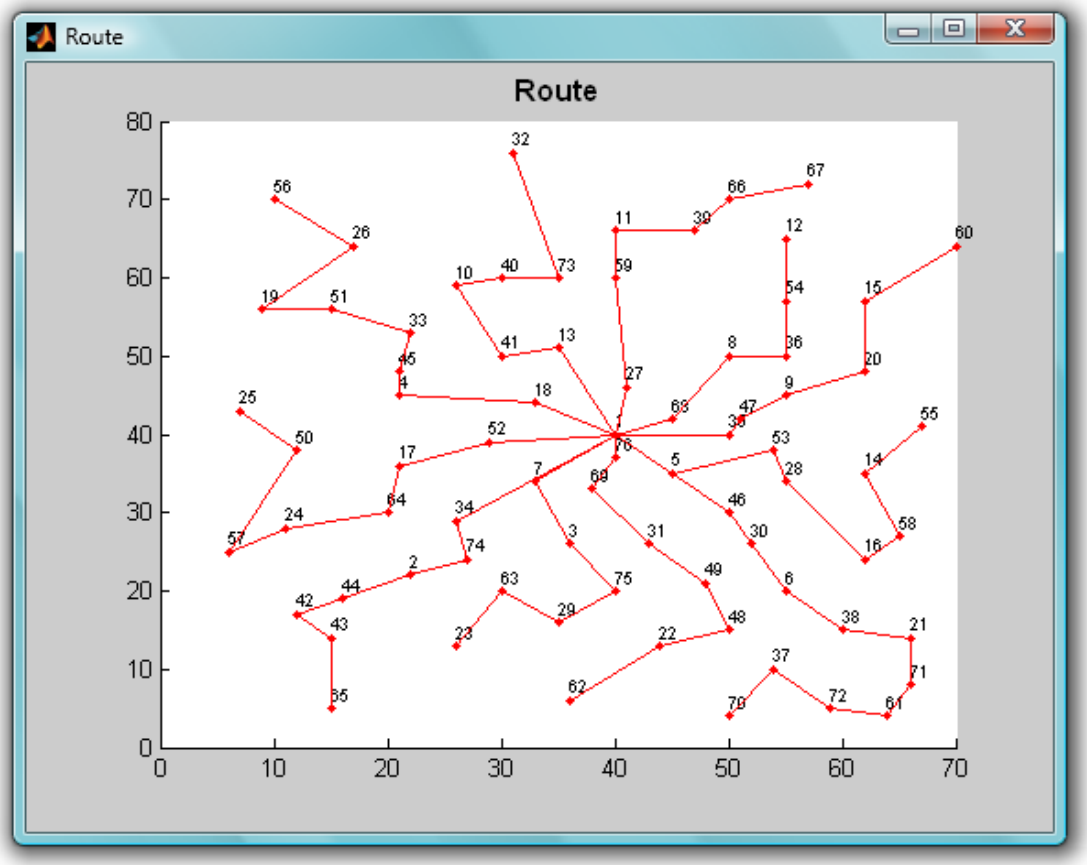
Η λύση που προέκυψε είχε κόστος 568,6032 . Η απόκλισή της από το μέχρι στιγμής βέλτιστο που έχει βρεθεί σε παγκόσμια κλίμακα (564,06) είναι:

$$\frac{568{,}60 - 564{,}06}{564{,}06} = 0{,}81\%$$

Η συγκεκριμένη λύση αποτελείται από 11 διαδρομές, που είναι οι εξής:

1	76	69	31	49	48	22	62			
1	46	30	6	38	21	71	61	72	37	70
1	34	74	2	44	42	43	65			
1	27	59	11	39	66	67				
1	35	47	9	20	15	60				
1	13	41	10	40	73	32				
1	5	53	28	16	58	14	55			
1	18	4	45	33	51	19	26	56		
1	7	3	75	29	63	23				
1	52	17	64	24	57	50	25			
1	68	8	36	54	12					

Η γραφική αναπαράσταση της λύσης έχει ως εξής:



5.3.3 Παράδειγμα 3

Στο παράδειγμα 3 υπάρχουν 101 κόμβοι. Δεν υπάρχουν χρονικοί περιορισμοί και όλα τα διαθέσιμα φορτηγά χωρητικότητα ίση με 200.

VRP:

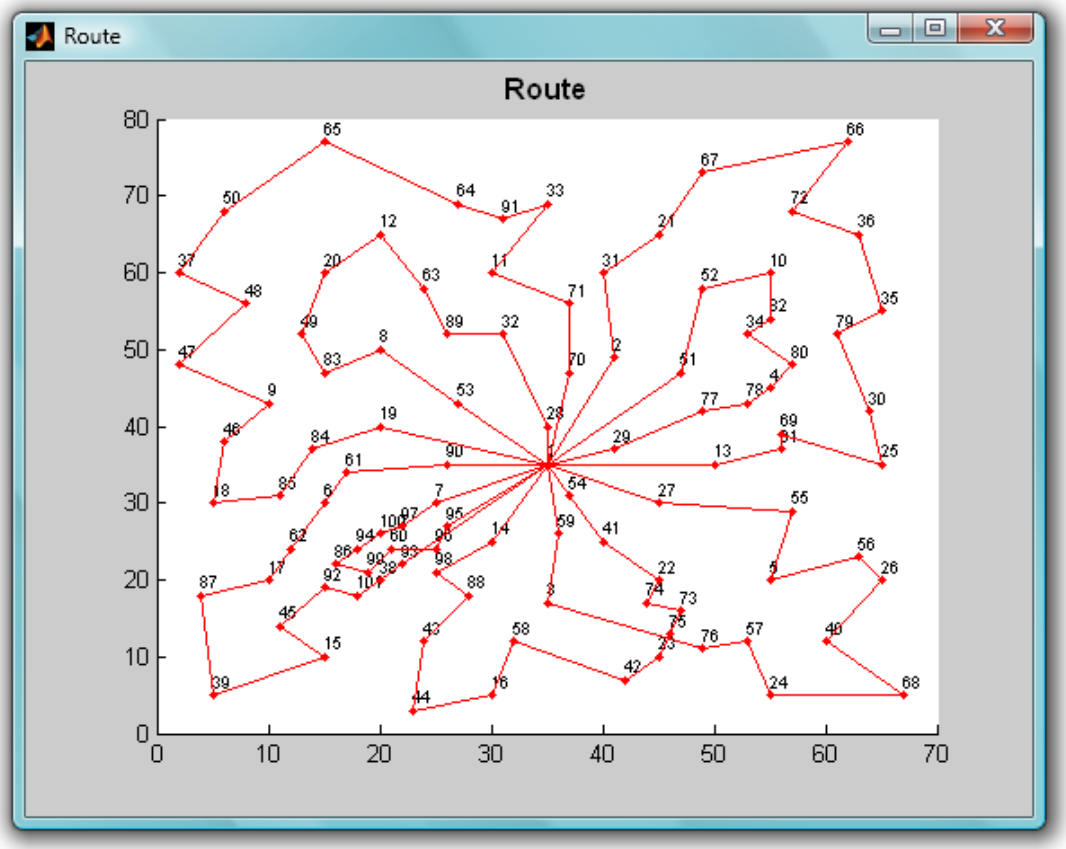
Η λύση που προέκυψε έχει κόστος 832,2714 . Η απόκλισή της από τη βέλτιστη λύση που έχει βρεθεί σε παγκόσμια κλίμακα (826,14) είναι:

$$\frac{832{,}2714 \ - 826{,}14}{826{,}14} = 0{,}74\%$$

Η συγκεκριμένη λύση αποτελείται από 8 διαδρομές οι οποίες έχουν ως εξής:

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

Γραφικά η συγκεκριμένη λύση έχει ως εξής:



OVRP:

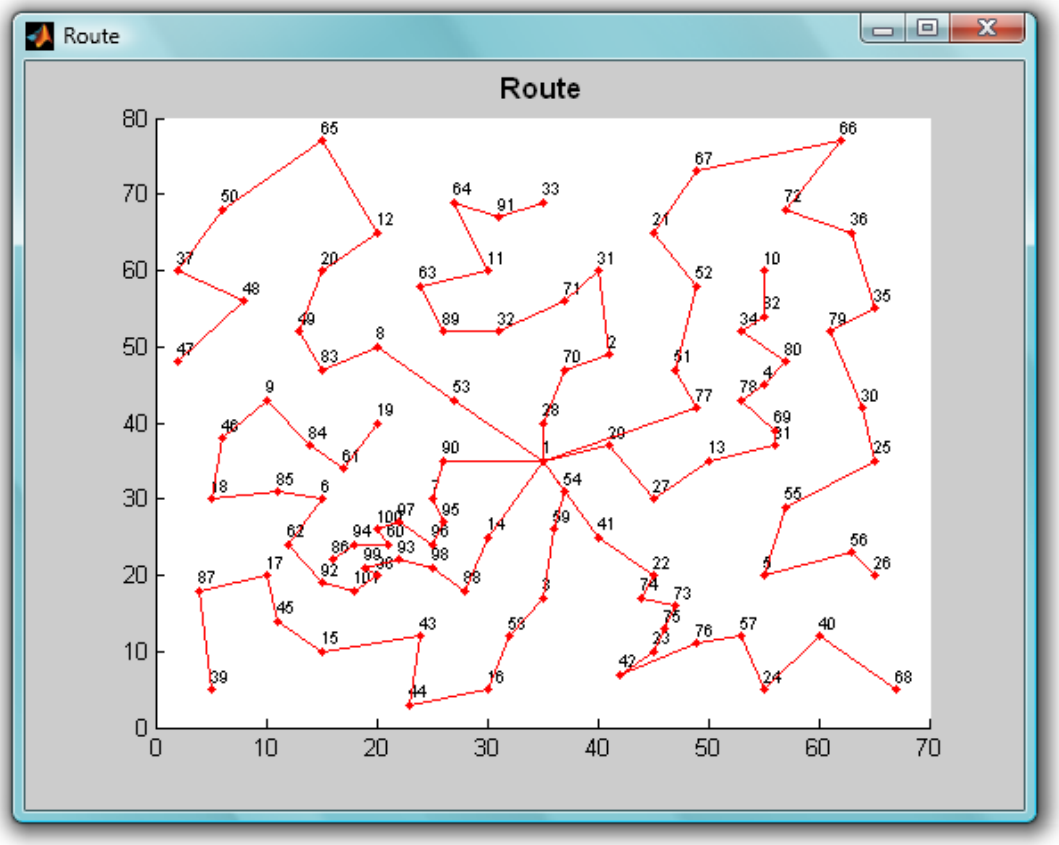
Η λύση που βρέθηκε έχει κόστος 663,6062 . Η απόκλισή της από τη μέχρι στιγμής βέλτιστη λύση που έχει βρεθεί (639,25) είναι:

$$\frac{663{,}60 - 639{,}25}{639{,}25} = 3{,}81\%$$

Η συγκεκριμένη λύση αποτελείται από 8 διαδρομές και είναι οι εξής:

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

Σχηματικά η λύση έχει ως εξής:



5.3.4 Παράδειγμα 4

Στο συγκεκριμένο παράδειγμα υπάρχουν 151 κόμβοι Επίσης όλα τα διαθέσιμα φορτηγά έχουν την ίδια χωρητικότητα, όπου είναι ίση με 200. Ακόμη, δεν υπάρχουν χρονικοί περιορισμοί.

VRP:

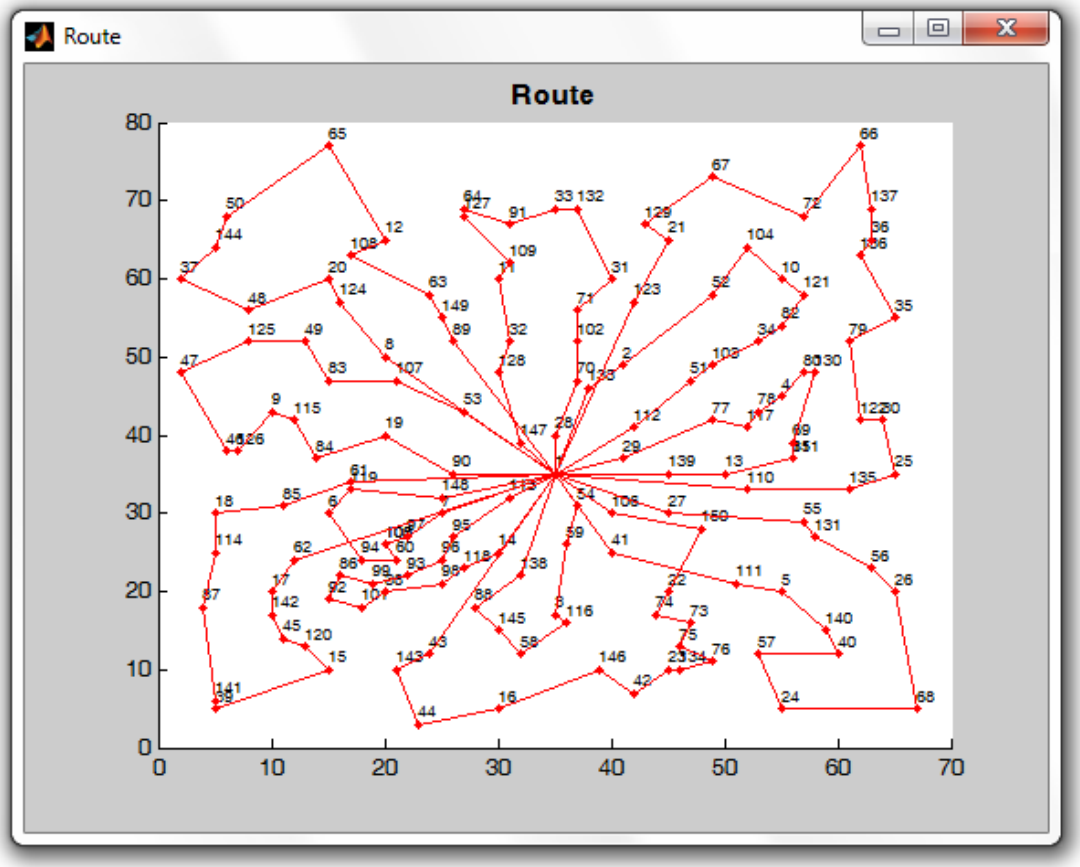
Η λύση που προέκυψε είχε κόστος 1057,24 . Η απόκλισή της από το μέχρι στιγμής βέλτιστο σε παγκόσμια κλίμακα (1028,42) είναι:

$$\frac{1057{,}24 - 1028{,}42}{1028{,}42} = 2{,}80\%$$

Η συγκεκριμένη λύση αποτελείται από δώδεκα διαδρομές, που είναι οι εξής:

1	139	13	151	81	69	130	80	4	78	117	77	29	1				
1	53	107	83	49	125	47	46	126	9	115	84	19	90	1			
1	8	124	20	48	37	144	50	65	12	108	63	149	89	1			
1	112	51	103	34	82	121	10	104	52	2	133	1					
1	54	59	3	116	58	145	88	138	1								
1	106	150	22	74	73	75	76	134	23	42	146	16	44	143	43	1	
1	41	111	5	140	40	57	24	68	26	56	131	55	27	1			
1	61	85	18	114	87	141	39	15	120	45	142	17	62	1			
1	113	95	96	93	99	86	92	101	38	98	118	14	1				
1	148	119	6	94	60	100	105	97	7	1							
1	123	21	129	67	72	66	137	36	136	35	79	122	30	25	135	110	1
1	28	70	102	71	31	132	33	91	64	127	109	11	32	128	147	1	

Η γραφική αναπαράσταση της λύσης έχει ως εξής:



OVRP:

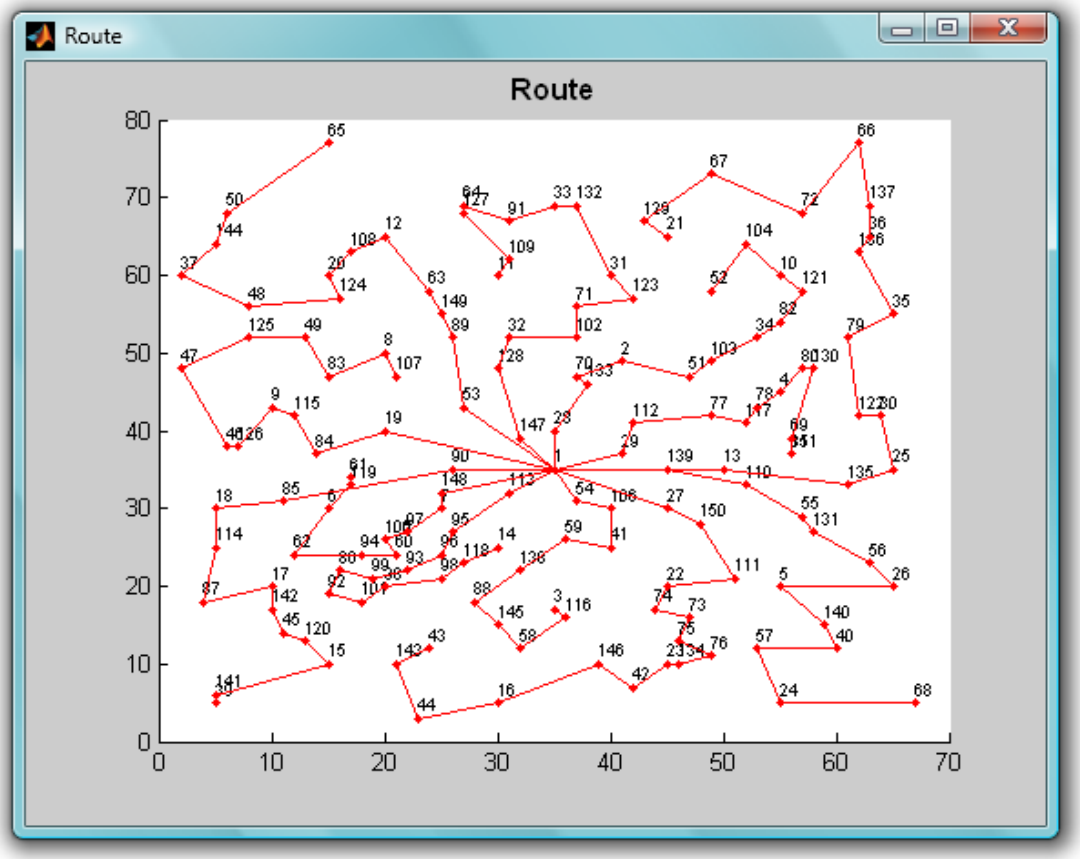
Η λύση είχε κόστος 795,2329 . Η απόκλισή της από τη μέχρι στιγμής βέλτιστη λύση σε παγκόσμια κλίμακα (733,13) είναι:

$$\frac{795{,}23 - 733{,}13}{733{,}13} = 8{,}47\%$$

Η λύση αποτελείται από 12 διαδρομές, τις εξής:

1	53	89	149	63	12	108	20	124	48	37	144	50	65	1			
1	19	84	115	9	126	46	47	125	49	83	8	107	1				
1	29	112	77	117	78	4	80	130	69	81	151	1					
1	28	133	70	2	51	103	34	82	121	10	104	52	1				
1	54	106	41	59	138	88	145	58	116	3	1						
1	27	150	111	22	74	73	75	76	134	23	42	146	16	44	143	43	1
1	139	110	55	131	56	26	5	140	40	57	24	68	1				
1	90	85	18	114	87	17	142	45	120	15	141	39	1				
1	113	95	96	93	99	86	92	101	38	98	118	14	1				
1	13	135	25	30	122	79	35	136	36	137	66	72	67	129	21	1	
1	148	7	97	105	100	60	94	62	6	119	61	1					
1	147	128	32	102	71	123	31	132	33	91	64	127	109	11	1		

Η γραφική αναπαράσταση της λύσης έχει ως εξής:



5.3.5 Παράδειγμα 5

Υπάρχουν 200 κόμβοι. Επίσης, θεωρείται ότι όλα τα διαθέσιμα φορτηγά έχουν χωρητικότητα ίση με 200. Τέλος, δεν υπάρχουν χρονικοί περιορισμοί.

VRP:

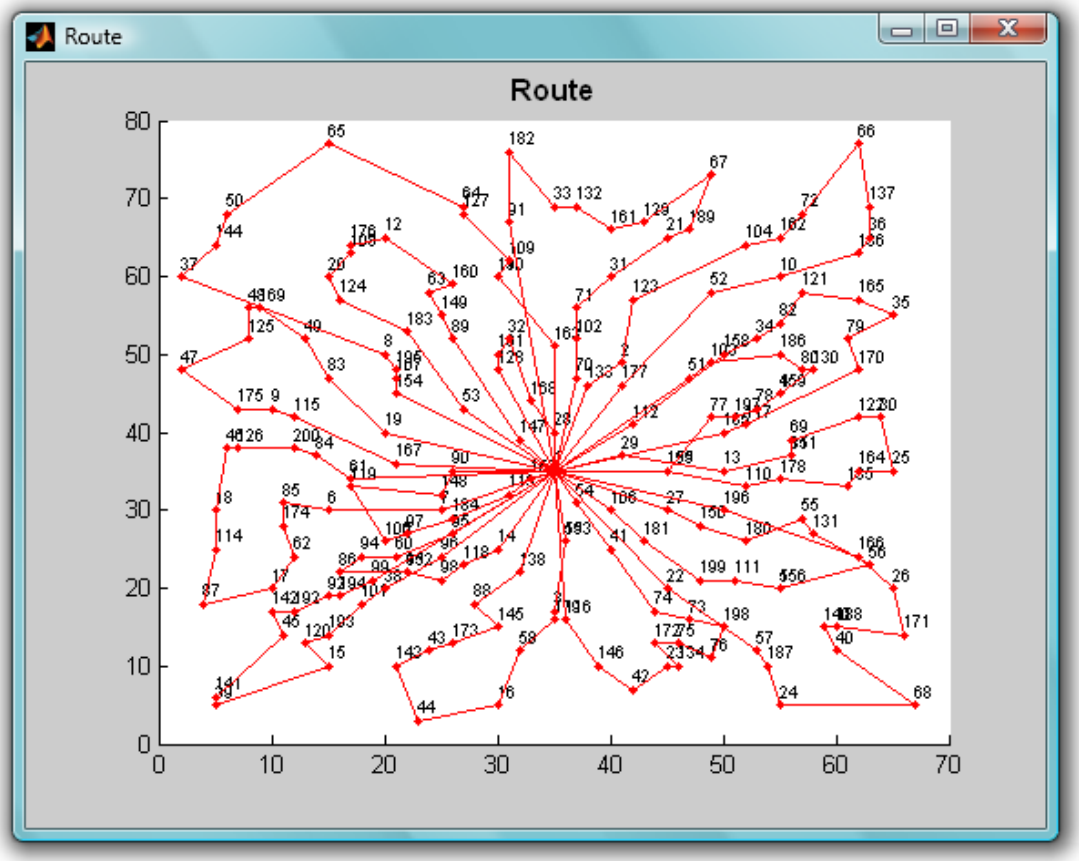
Η λύση που βρέθηκε έχει κόστος 1355,69 και η απόκλισή της από το μέχρι στιγμής βέλτιστο σε παγκόσμια κλίμακα (1291,29) είναι:

$$\frac{1355{,}69 - 1291{,}29}{1291{,}29} = 4{,}96\%$$

Η λύση αποτελείται από 17 διαδρομές. Οι 17 αυτές διαδρομές έχουν ως εξής:

1	157	113	184	97	105	100	119	148	90	1					
1	116	146	42	23	134	172	75	76	198	73	74	41	54	1	
1	196	166	26	171	188	140	40	68	24	187	57	22	1		
1	163	11	190	109	127	64	65	50	144	37	8	195	107	154	1
1	70	102	71	31	21	189	67	129	161	132	33	182	91	1	
1	19	83	49	169	48	125	47	175	9	115	167	1			
1	133	2	123	104	162	72	66	137	36	136	10	52	177	1	
1	139	110	178	135	164	25	30	122	69	151	81	13	29	1	
1	112	158	34	82	121	165	35	79	170	117	185	1			
1	155	77	197	78	4	159	130	80	186	103	51	1			
1	147	89	149	63	160	12	176	108	20	124	183	53	1		
1	138	88	145	173	43	143	44	16	58	179	3	153	59	1	
1	106	181	199	111	156	5	56	131	55	180	150	27	1		
1	128	191	32	168	28	1									
1	99	194	92	192	142	45	141	39	15	120	193	101	38	96	1
1	14	118	98	152	93	86	94	60	95	1					
1	61	84	200	126	46	18	114	87	17	62	174	85	6	7	1

Στο επόμενο σχήμα παρουσιάζεται γραφικά η λύση:



OVRP:

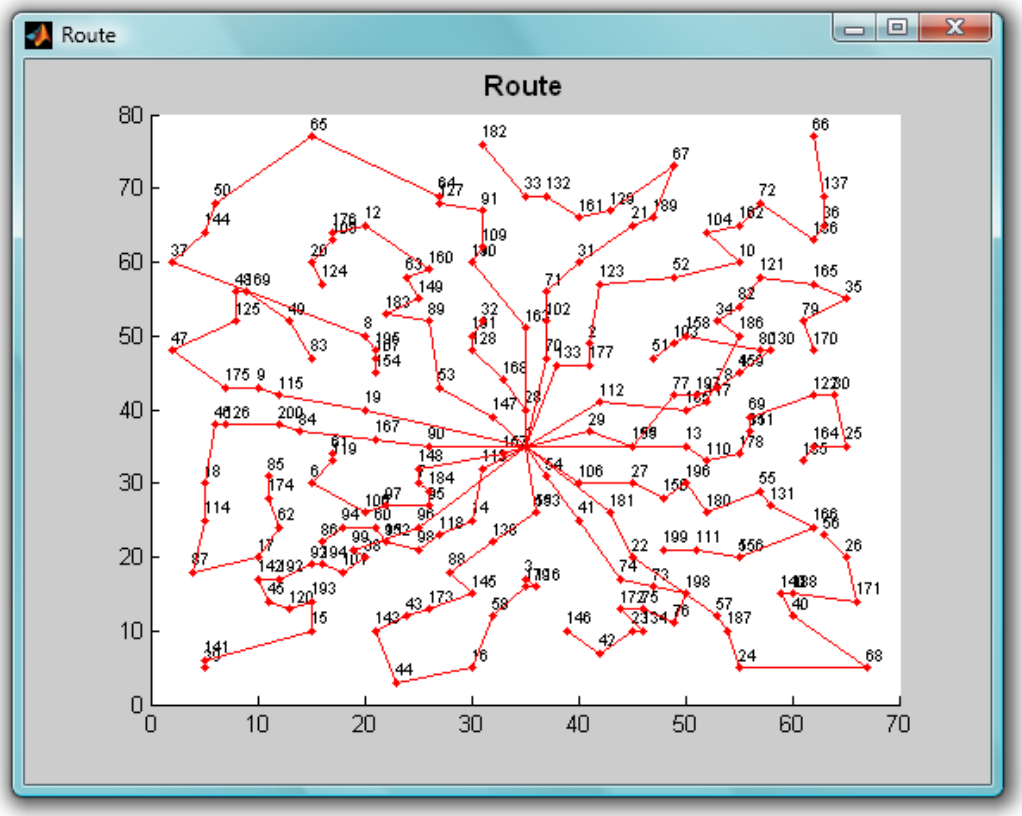
Η λύση για το συγκεκριμένο παράδειγμα έχει κόστος 950,8875 και η απόκλισή της από το μέχρι στιγμής βέλτιστο σε παγκόσμια κλίμακα (868,44) είναι:

$$\frac{950{,}89 - 868{,}44}{868{,}44} = 9{,}49\%$$

Η λύση αποτελείται από 17 διαδρομές. Οι 17 αυτές διαδρομές έχουν ως εξής:

1	157	148	7	184	95	97	105	100	6	119	61	1				
1	181	22	57	187	24	68	40	140	188	171	26	56	1			
1	54	41	74	73	198	76	75	172	134	23	42	146	1			
1	163	11	190	109	91	127	64	65	50	144	37	8	195	107	154	1
1	70	102	71	31	21	189	67	129	161	132	33	182	1			
1	19	115	9	175	47	125	48	169	49	83	1					
1	133	177	2	123	52	10	104	162	72	136	36	137	66	1		
1	29	139	13	110	178	81	151	69	122	30	25	164	135	1		
1	112	185	117	186	34	82	121	165	35	79	170	1				
1	155	77	197	78	4	159	130	80	158	103	51	1				
1	147	53	89	183	149	63	160	12	176	108	20	124	1			
1	90	167	84	200	126	46	18	114	87	17	62	174	85	1		
1	113	14	118	98	152	93	60	94	86	1						
1	153	59	138	88	145	173	43	143	44	16	58	179	116	3	1	
1	106	27	150	196	180	55	131	166	5	156	111	199	1			
1	96	99	38	101	194	92	192	142	45	120	193	15	141	39	1	
1	28	168	128	191	32	1										

Στο παρακάτω σχήμα αναπαρίσταται γραφικά η λύση:



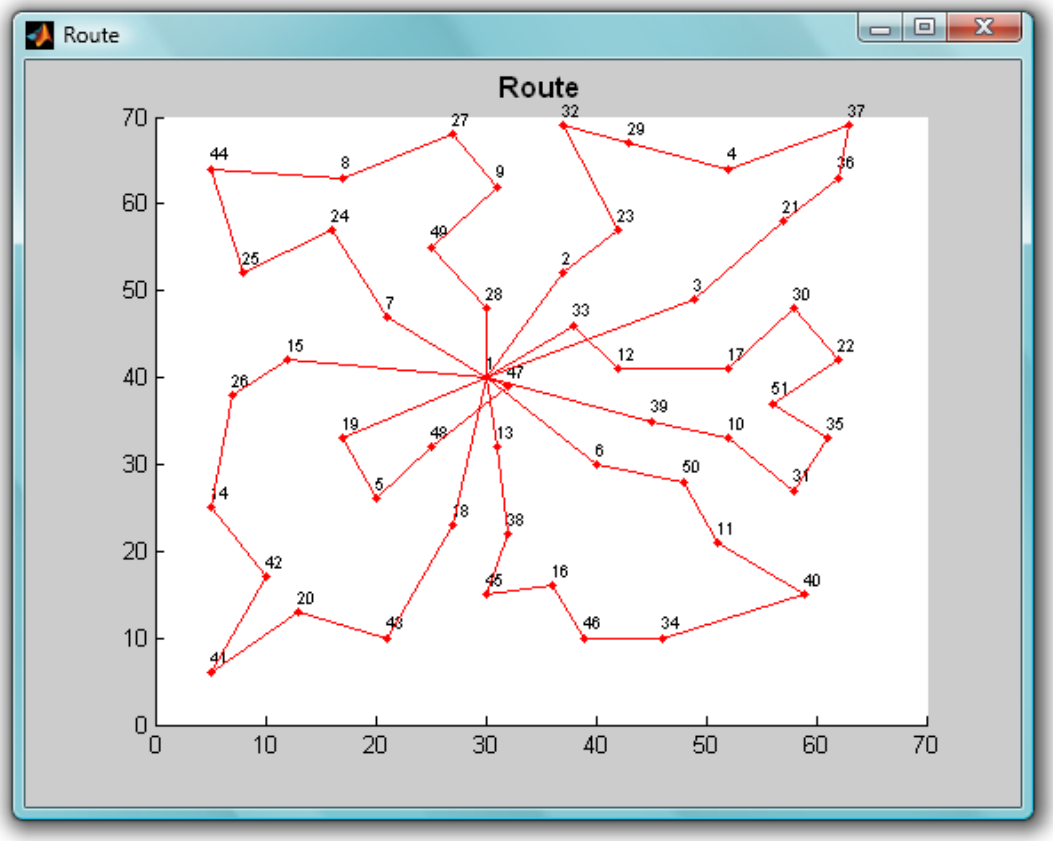
5.3.6 Παράδειγμα 6

Στο παράδειγμα 6 υπάρχουν 51 κόμβοι. Όλα τα διαθέσιμα φορτηγά έχουν χωρητικότητα ίση με 160. Επίσης , κανένα δρομολόγιο δεν πρέπει να έχει μήκος πάνω από 200.

Η λύση του συστήματος που κατασκευάστηκε στα πλαίσια της παρούσας μεταπτυχιακής διατριβής έχει κόστος ίδιο με αυτό της βέλτιστης λύσης που έχει βρεθεί σε παγκόσμια κλίμακα. Η βέλτιστη λύση που βρέθηκε έχει κόστος ίσο με 555,4302 και αποτελείται από 6 διαδρομές, οι οποίες ξεκινούν και καταλήγουν στον κόμβο 1. Η λύση είναι η εξής:

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

Η γραφική της απεικόνιση είναι η εξής:



OVRP:

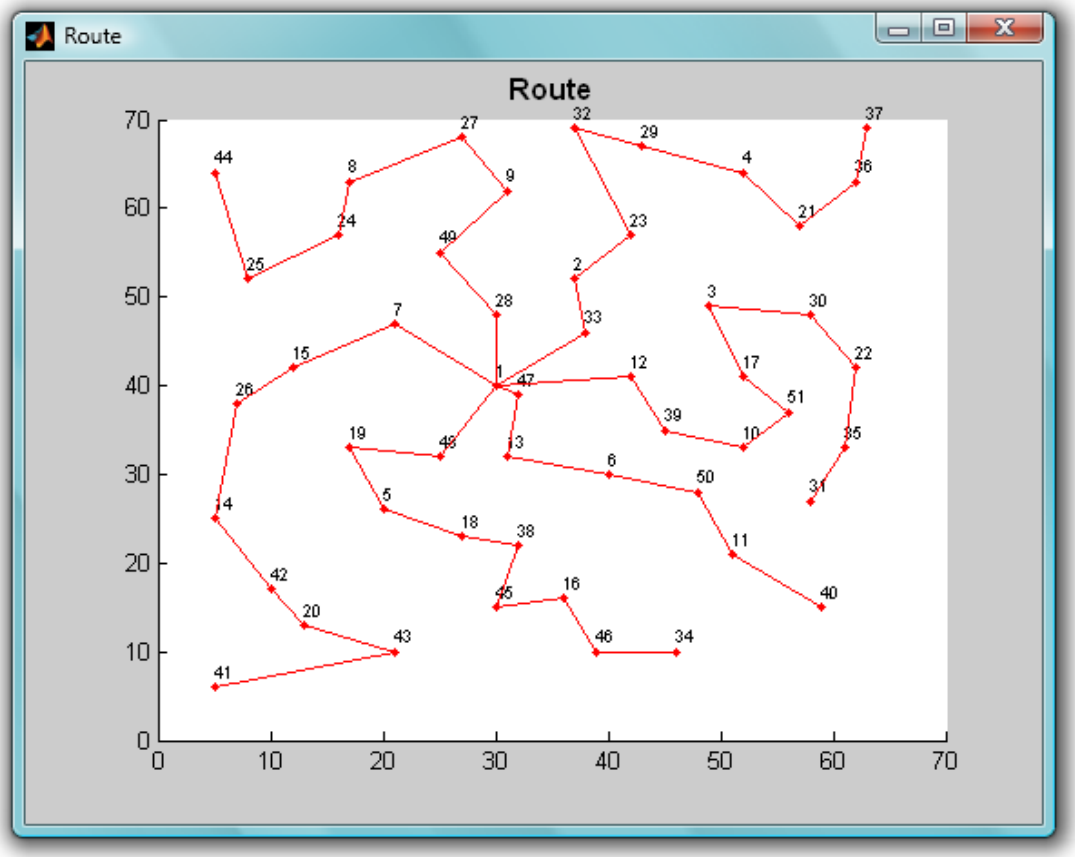
Η λύση που προέκυψε έχει κόστος 412,9568 και η απόκλισή της από τη μέχρι στιγμής βέλτιστη λύση σε παγκόσμια κλίμακα (412,95) είναι 0,00%:

$$\frac{412{,}9568 - 412{,}95}{412{,}95} = 0{,}00\ \%$$

Η συγκεκριμένη λύση αποτελείται από 6 διαδρομές οι οποίες έχουν ως εξής:

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

Γραφικά η συγκεκριμένη λύση έχει ως εξής:



5.3.7 Παράδειγμα 7

Στο παράδειγμα 7 υπάρχουν 76 κόμβοι. Επίσης, όλα τα διαθέσιμα φορτηγά έχουν την ίδια χωρητικότητα, η οποία είναι ίση με 140. Τέλος, καμία διαδρομή δεν πρέπει να έχει συνολικό μήκος πάνω από 160.

VRP:

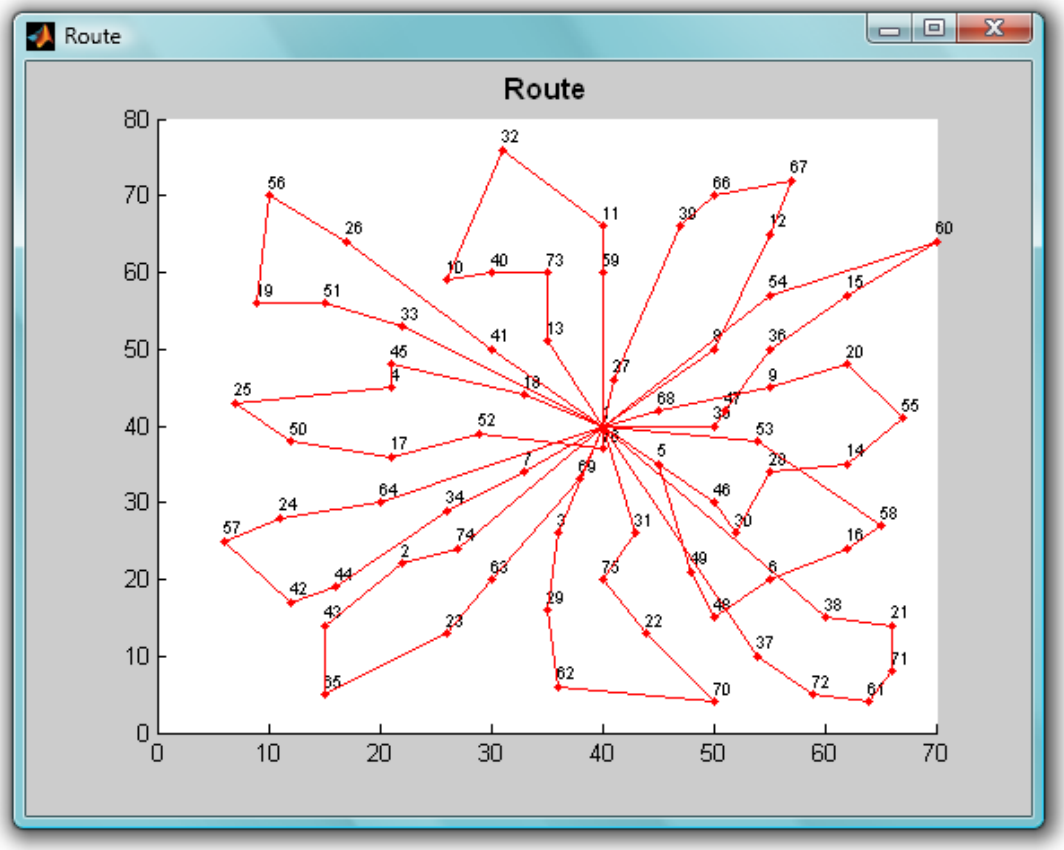
Η λύση που βρέθηκε έχει κόστος 921,09 . Η απόκλισή της από τη μέχρι στιγμής βέλτιστη λύση που έχει βρεθεί σε παγκόσμια κλίμακα (909,68) είναι:

$$\frac{921{,}09 - 909{,}68}{909{,}68} = 1{,}25\%$$

Η συγκεκριμένη λύση αποτελείται από 12 διαδρομές και είναι οι εξής:

1	33	51	19	56	26	41	1		
1	7	34	44	42	57	24	64	1	
1	27	39	66	67	12	8	1		
1	68	9	20	55	14	28	30	46	1
1	37	72	61	71	21	38	1		
1	76	52	17	50	25	4	45	18	1
1	35	47	36	15	60	54	1		
1	53	58	16	6	48	49	5	1	
1	59	11	32	10	40	73	13	1	
1	74	2	43	65	23	63	69	1	
1	3	29	62	70	22	75	31	1	
1	33	51	19	56	26	41	1		

Σχηματικά η λύση έχει ως εξής:



OVRP:

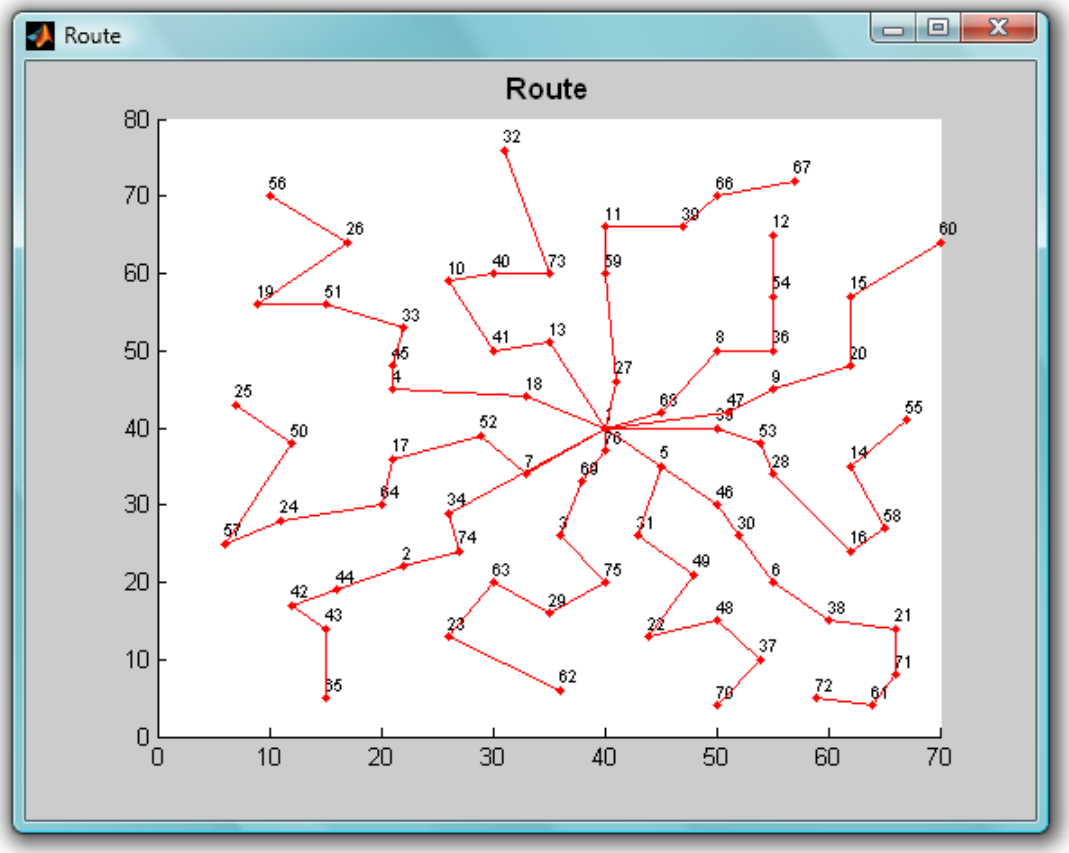
Η λύση που προέκυψε είχε κόστος 570,7721 . Η απόκλισή της από το μέχρι στιγμής βέλτιστο που έχει βρεθεί σε παγκόσμια κλίμακα (566,93) είναι:

$$\frac{570,77 - 566,93}{566,93} = 0,68\%$$

Η συγκεκριμένη λύση αποτελείται από 11 διαδρομές, που είναι οι εξής:

1	46	30	6	38	21	71	61	72
1	7	52	17	64	24	57	50	25
1	5	31	49	22	48	37	70	
1	27	59	11	39	66	67		
1	13	41	10	40	73	32		
1	47	9	20	15	60			
1	35	53	28	16	58	14	55	
1	34	74	2	44	42	43	65	
1	68	8	36	54	12			
1	18	4	45	33	51	19	26	56
1	76	69	3	75	29	63	23	62

Σχηματικά η λύση έχει ως εξής:



5.3.8 Παράδειγμα 8

Στο παράδειγμα 8 υπάρχουν 101 κόμβοι και θεωρείται ότι όλα τα διαθέσιμα φορτηγά έχουν χωρητικότητα ίση με 200. Επίσης, καμία διαδρομή δεν πρέπει να έχει συνολικό μήκος πάνω από 230.

VRP:

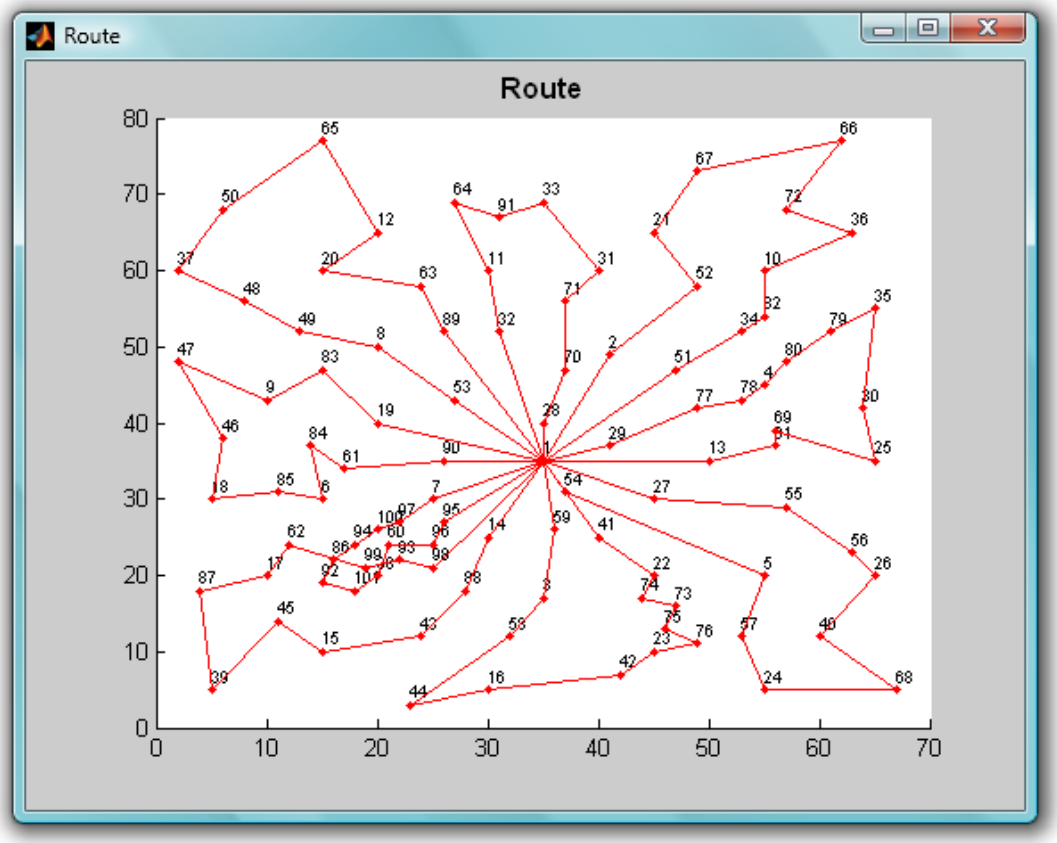
Η λύση που προέκυψε είχε κόστος 871,71 και η απόκλισή της από το μέχρι στιγμής βέλτιστο παγκοσμίως(865,94) είναι:

$$\frac{871{,}71 - 865{,}94}{865{,}94} = 0{,}67\ \%$$

Η λύση για το 7ο παράδειγμα αποτελείται από 9 διαδρομές. Οι συγκεκριμένες 9 διαδρομές έχουν ως εξής:

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

Σχηματικά η συγκεκριμένη λύση έχει ως εξής:



OVRP:

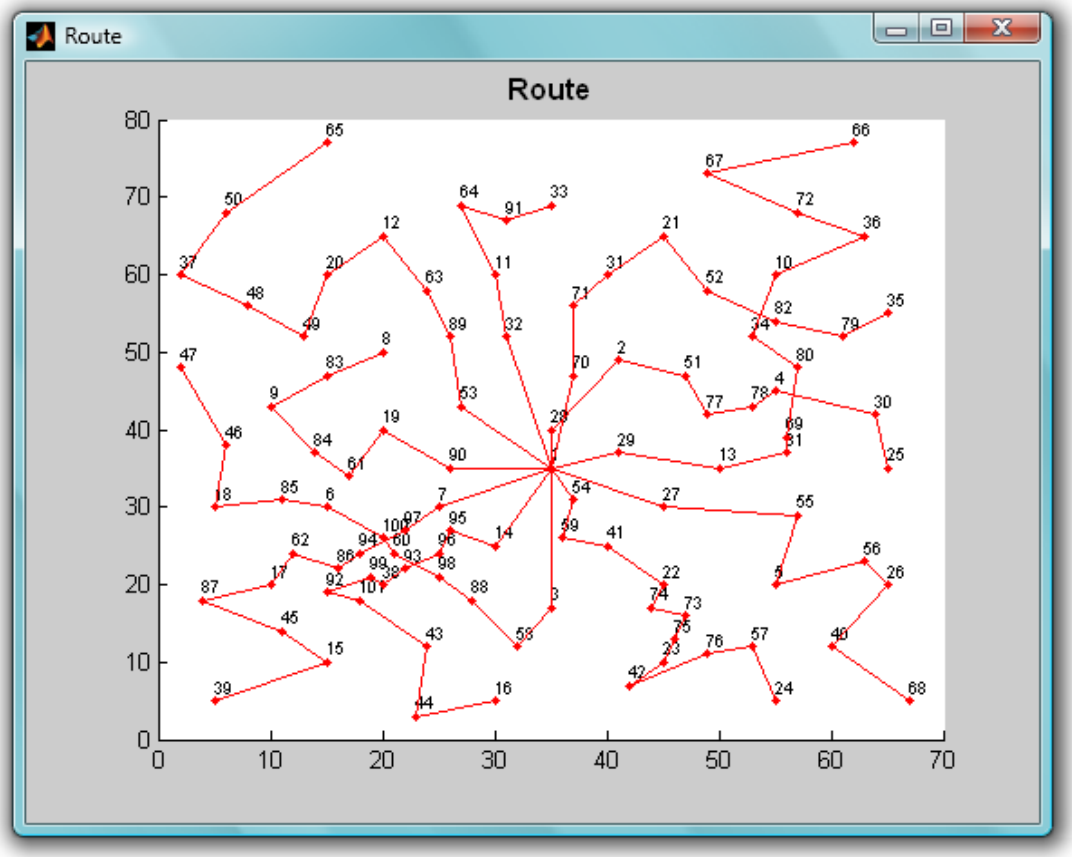
Η λύση για το συγκεκριμένο παράδειγμα έχει κόστος 694,9458 και η απόκλισή της από το μέχρι στιγμής βέλτιστο σε παγκόσμια κλίμακα (640,89) είναι:

$$\frac{694{,}95 - 640{,}89}{640{,}89} = 8{,}43\ \%$$

Η συγκεκριμένη λύση αποτελείται από 11 διαδρομές, οι οποίες έχουν ως εξής:

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

Η λύση γραφικά έχει ως εξής:



5.3.9 Παράδειγμα 9

Σε αυτό το παράδειγμα υπάρχουν 76 κόμβοι. Όλα τα διαθέσιμα φορτηγά έχουν την ίδια χωρητικότητα, η οποία είναι ίση με 140. Τέλος, καμία διαδρομή δεν πρέπει να έχει συνολικό μήκος πάνω από 160.

VRP:

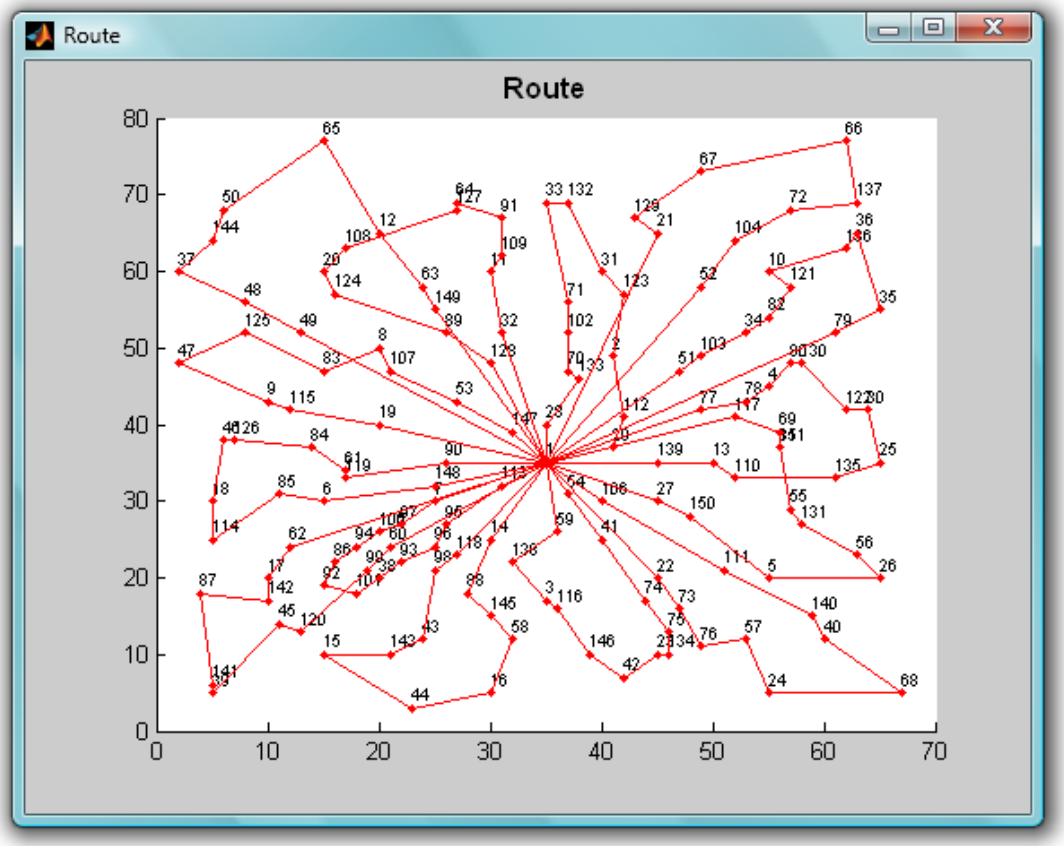
Η λύση που βρέθηκε είχε κόστος ίσο με 1186,36. Η απόκλισή της από τη μέχρι στιγμής βέλτιστη λύση που έχει βρεθεί σε παγκόσμια κλίμακα (1162,55) είναι:

$$\frac{1186{,}36 - 1162{,}55}{1162{,}55} = 2{,}05\ \%$$

Η λύση για το 9^{O} παράδειγμα αποτελείται από 16 διαδρομές οι οποίες έχουν ως εξής:

1	49	48	37	144	50	65	12	63	149	1				
1	32	11	109	91	64	127	108	20	124	89	128	1		
1	19	115	9	47	125	83	8	107	53	147	1			
1	113	95	96	93	38	101	92	86	94	100	105	97	7	1
1	54	41	74	75	134	23	42	146	116	3	138	59	1	
1	106	111	140	40	68	24	57	76	73	22	1			
1	28	133	70	102	71	33	132	31	123	2	112	29	1	
1	77	78	4	80	130	122	30	25	135	110	13	139	1	
1	51	103	34	82	121	10	136	36	35	79	1			
1	90	119	61	84	126	46	18	114	85	6	148	1		
1	21	129	67	66	137	72	104	52	1					
1	60	99	120	45	39	141	87	142	17	62	1			
1	27	150	5	26	56	131	55	151	81	69	117	1		
1	118	98	43	143	15	44	16	58	145	88	14	1		
1	49	48	37	144	50	65	12	63	149	1				
1	32	11	109	91	64	127	108	20	124	89	128	1		

Γραφικά, η συγκεκριμένη λύση έχει ως εξής:



OVRP:

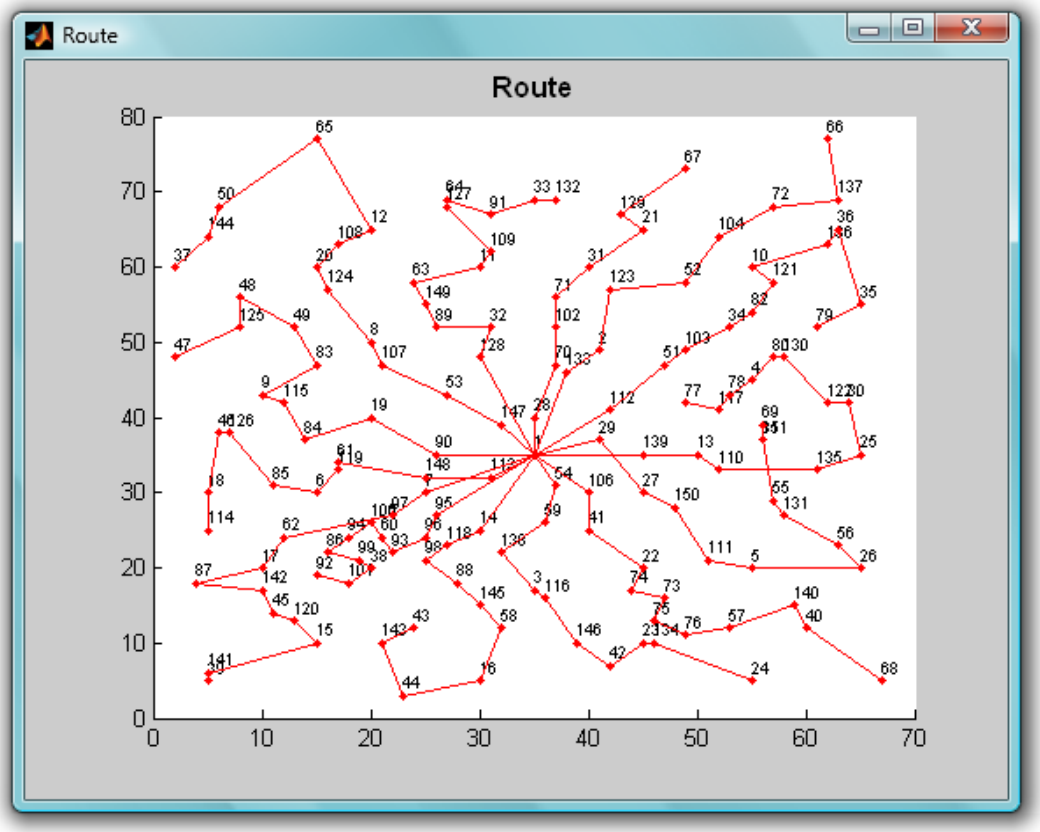
Η λύση που εντοπίστηκε έχει κόστος: 790,6126. Η απόκλισή της από το μέχρι στιγμής βέλτιστο(741,44) είναι:

$$\frac{790{,}61 - 741{,}44}{741{,}44} = 6{,}63\ \%$$

Η συγκεκριμένη λύση αποτελείται από 14 διαδρομές, κι είναι οι εξής:

1	147	53	107	8	124	20	108	12	65	50	144	37	1	
1	128	32	89	149	63	11	109	127	64	91	33	132	1	
1	90	19	84	115	9	83	49	48	125	47	1			
1	95	96	93	60	105	100	94	86	99	38	101	92	1	
1	106	41	22	74	73	75	76	57	140	40	68	1		
1	54	59	138	3	116	146	42	23	134	24	1			
1	28	70	102	71	31	21	129	67	1					
1	139	13	110	135	25	30	122	130	80	4	78	117	77	1
1	112	51	103	34	82	121	10	136	36	35	79	1		
1	133	2	123	52	104	72	137	66	1					
1	113	148	61	119	6	85	126	46	18	114	1			
1	7	97	62	17	87	142	45	120	15	141	39	1		
1	29	27	150	111	5	26	56	131	55	81	151	69	1	
1	14	118	98	88	145	58	16	44	143	43	1			

Γραφικά, η λύση για το 9^{ο} παράδειγμα έχει ως εξής



5.3.10 Παράδειγμα 10

Στο συγκεκριμένο παράδειγμα υπάρχουν 200 κόμβοι και θεωρείται ότι όλα τα διαθέσιμα φορτηγά έχουν χωρητικότητα ίση με 200. Επίσης, καμία διαδρομή δεν πρέπει να έχει συνολικό μήκος πάνω από 200.

VRP:

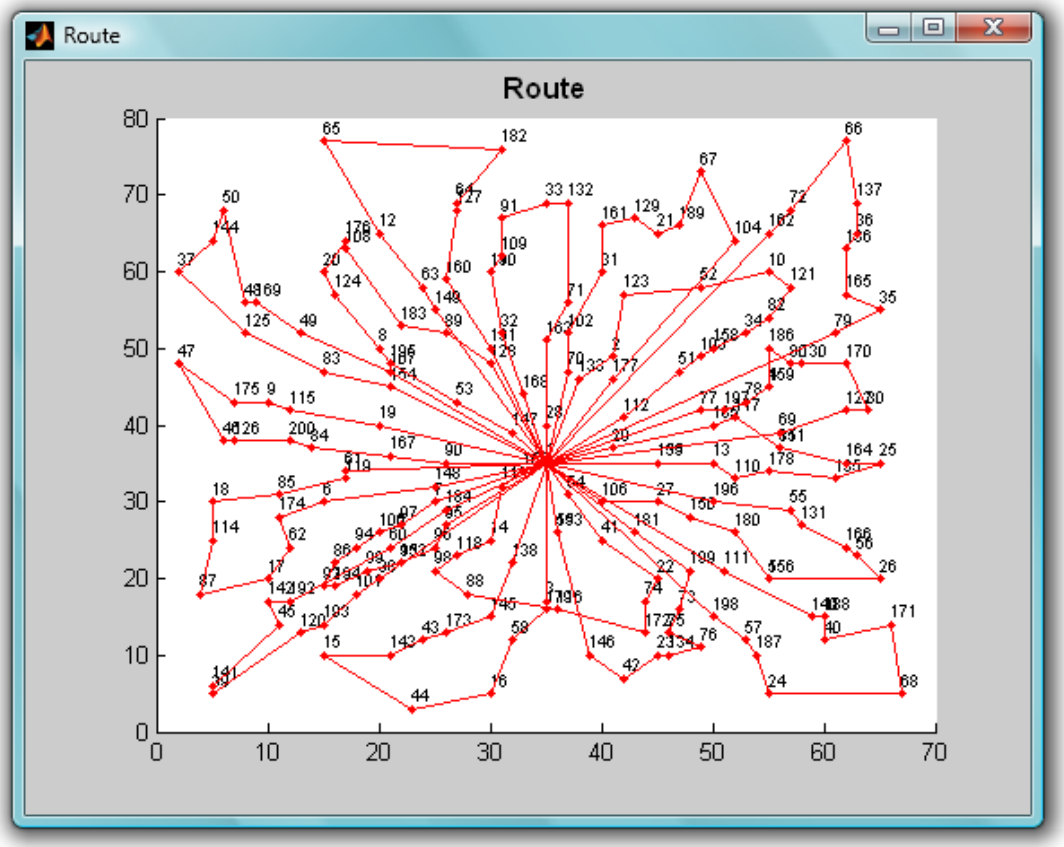
Η λύση που προέκυψε είχε κόστος 1452,39. Η απόκλισή της από το μέχρι στιγμής βέλτιστο αποτέλεσμα από προηγούμενες έρευνες σε παγκόσμια κλίμακα (1395,85) είναι:

$$\frac{1452{,}39 - 1395{,}85}{1395{,}85} = 4{,}05\%$$

Η συγκεκριμένη λύση αποτελείται από 20 δρομολόγια, τα εξής:

1	139	155	13	110	178	135	25	164	151	81	117	185	29	1
1	181	199	73	75	76	134	23	42	146	153	59	1		
1	95	96	152	93	99	194	92	86	94	100	105	97	7	1
1	157	113	14	118	98	88	116	172	74	22	41	54	1	
1	147	53	195	8	124	20	176	108	183	89	128	1		
1	79	35	165	136	36	137	66	72	162	1				
1	90	167	84	200	126	46	47	175	9	115	19	1		
1	184	60	192	142	45	141	39	120	193	101	38	1		
1	154	83	125	37	144	50	48	169	49	107	1			
1	149	63	12	65	182	64	127	160	191	1				
1	198	57	187	24	68	171	40	188	140	111	1			
1	77	197	78	4	159	186	80	130	170	30	122	69	1	
1	168	32	11	190	109	91	33	132	71	163	28	1		
1	61	119	85	18	114	87	17	62	174	6	148	1		
1	196	55	131	166	56	26	156	5	180	150	27	106	1	
1	70	102	31	161	129	21	189	67	104	177	1			
1	133	2	123	52	10	121	82	34	158	103	51	112	1	
1	3	179	58	16	44	15	143	43	173	145	138	1		

Γραφικά η λύση του παραδείγματος 10 έχει ως εξής:



OVRP:

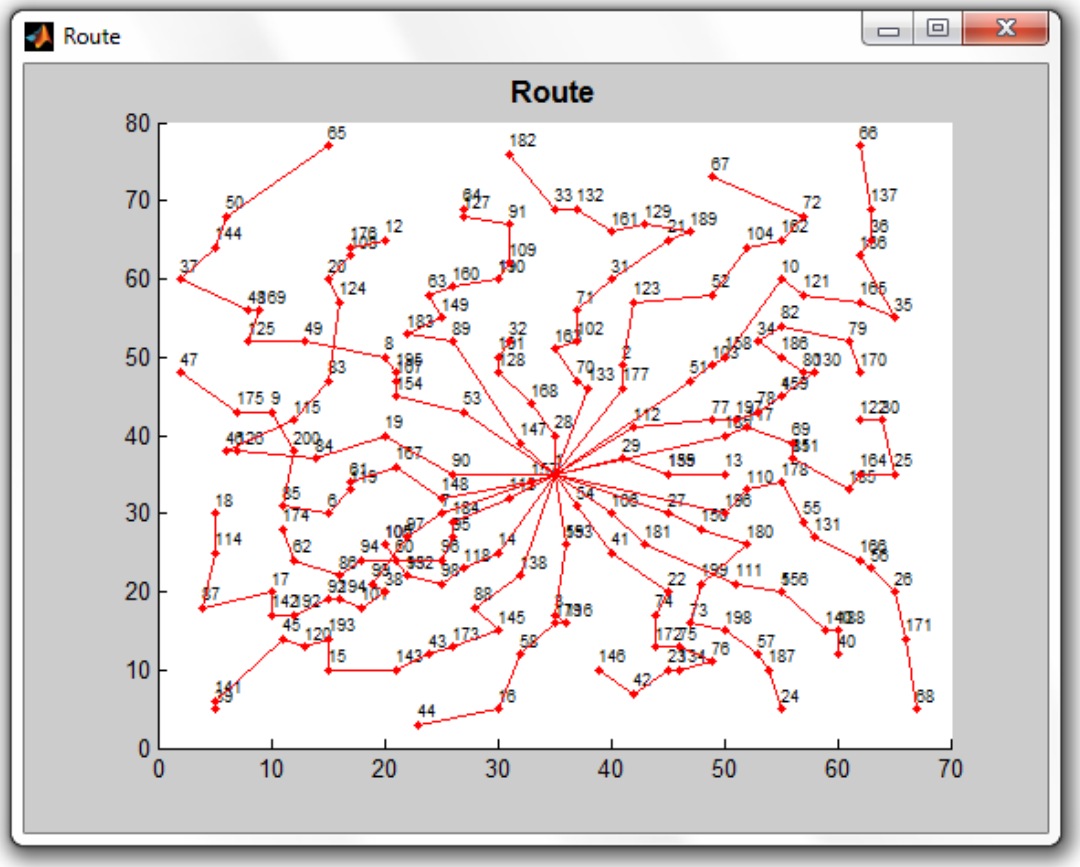
Η λύση για το 10^{ο} παράδειγμα έχει κόστος 938,0832. Η απόκλισή της από τη μέχρι στιγμής βέλτιστη λύση από έρυνες σε παγκόσμια κλίμακα (871,58) είναι:

$$\frac{938{,}08 - 871{,}58}{871{,}58} = 7{,}63\%$$

Η συγκεκριμένη λύση αποτελείται από 20 δρομολόγια, τα εξής:

1	185	117	69	81	151	135	164	25	30	122	1			
1	112	77	197	78	159	4	130	80	186	34	82	79	170	1
1	106	181	111	156	5	140	188	40	1					
1	14	118	98	93	152	60	105	100	1					
1	113	184	95	96	94	86	62	174	1					
1	54	41	22	74	172	75	76	134	23	42	146	1		
1	27	150	180	199	73	198	57	187	24	1				
1	196	110	178	55	131	166	56	26	171	68	1			
1	90	19	84	126	46	115	83	124	20	108	176	12	1	
1	153	59	3	116	179	58	16	44	1					
1	7	97	99	38	101	194	92	192	142	17	87	114	18	1
1	53	154	107	195	8	49	125	169	48	37	144	50	65	1
1	147	89	183	149	63	160	11	190	109	91	127	64	1	
1	133	70	163	102	71	31	21	189	129	161	132	33	182	1
1	29	155	139	13	1									
1	157	148	167	61	119	6	85	200	9	175	47	1		
1	28	168	128	191	32	1								
1	138	88	145	173	43	143	15	193	120	45	141	39	1	

Η γραφική αναπαράσταση της λύσης έχει ως εξής:



5.3.11 Παράδειγμα 11

Στο συγκεκριμένο παράδειγμα υπάρχουν 121 κόμβοι. Επίσης, όλα τα διαθέσιμα φορτηγά έχουν χωρητικότητα ίση με 200. Τέλος, δεν υπάρχουν χρονικοί περιορισμοί.

VRP:

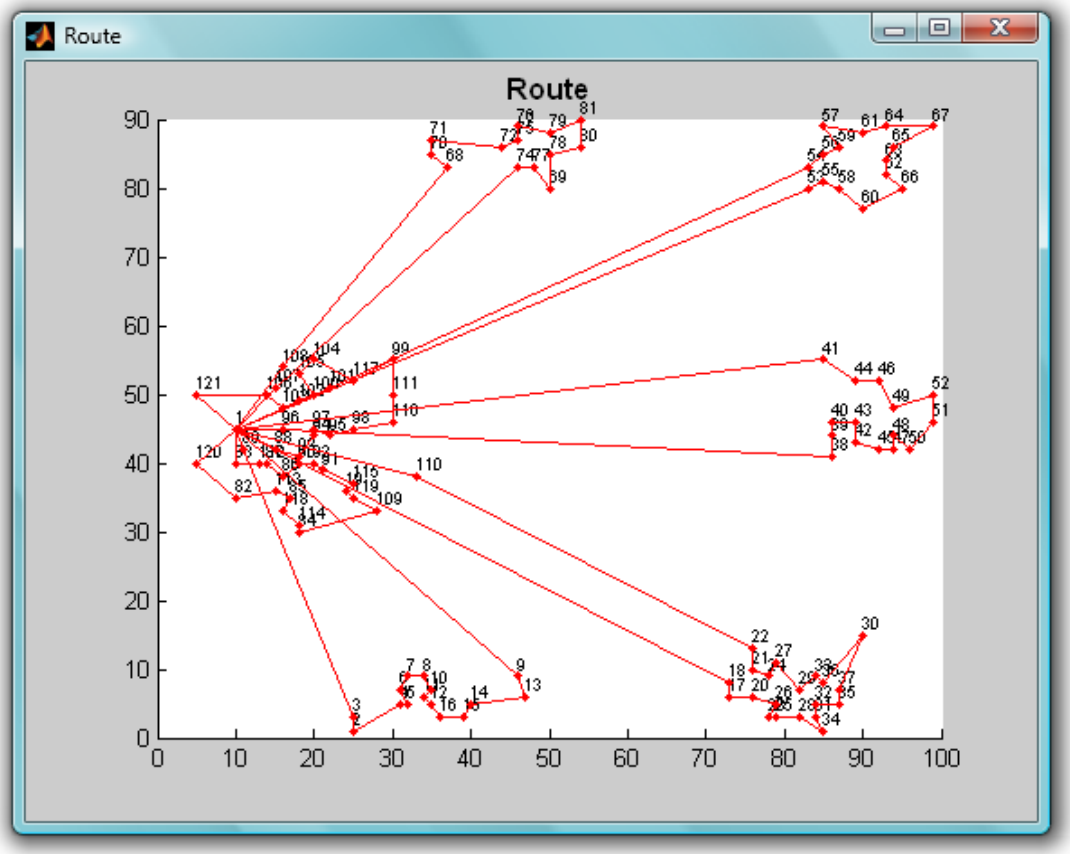
Η λύση που προέκυψε έχει κόστος 1042,1150 και η απόκλισή της από το μέχρι στιγμής βέλτιστο σε παγκόσμια κλίμακα (1042,11) είναι, σχεδόν 0%:

$$\frac{1042{,}1150 - 1042{,}11}{1042{,}11} = 0{,}00\ \%$$

Η συγκεκριμένη λύση αποτελείται από 7 διαδρομές οι οποίες έχουν ως εξής:

1	83	112	87	86	90	92	91	115	19	119	…	
…	109	84	114	118	85	113	82	120	1			
1	108	68	70	71	72	75	73	76	79	81	…	
…	80	78	69	77	74	107	1					
1	88	93	94	97	95	98	116	111	99	117	…	
…	104	105	100	102	103	106	121	1				
1	9	13	14	15	16	12	11	10	8	7	…	
…	6	5	4	2	3	89	1					
1	96	38	39	40	43	42	45	47	48	50	…	
…	51	52	49	46	44	41	1					
1	53	55	58	60	66	62	63	65	67	64	…	
…	61	57	59	56	54	101	1					
1	110	22	21	24	27	29	33	36	30	37	…	
…	35	32	31	34	28	25	23	26	20	17	18	1

Γραφικά η λύση για το παράδειγμα 11 έχει ως εξής:



OVRP:

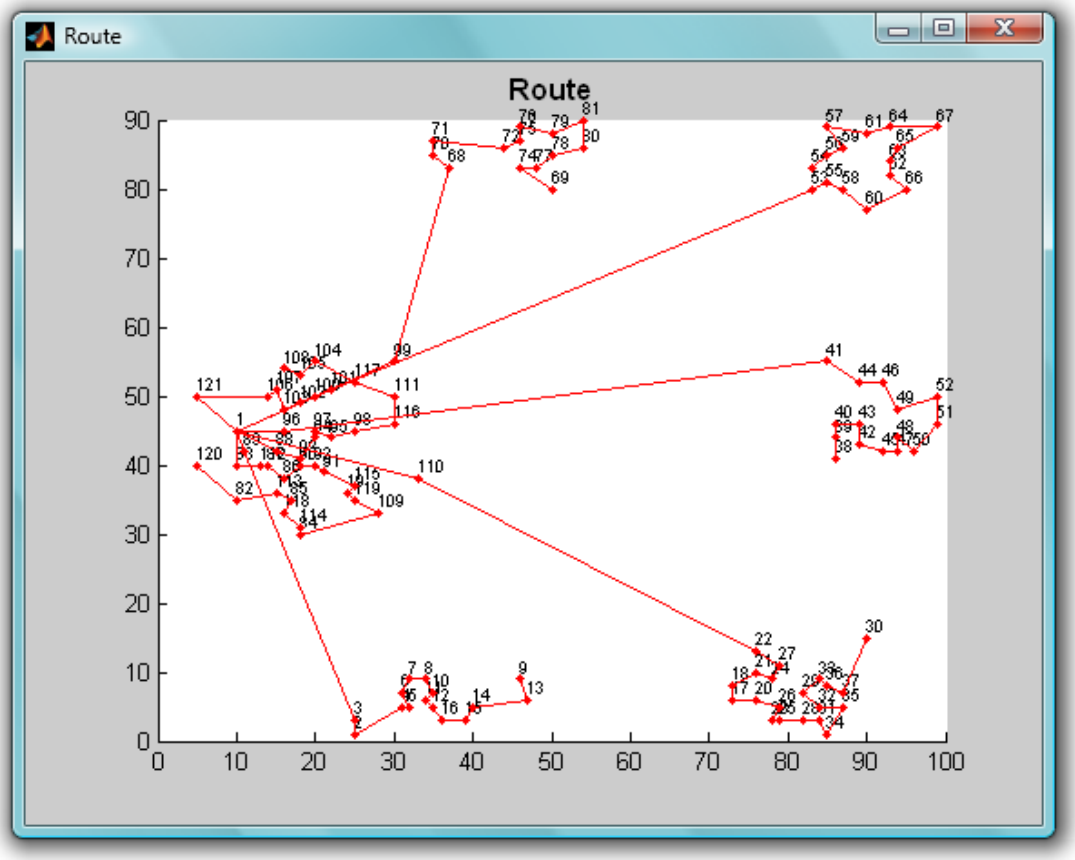
Η λύση που προέκυψε έχει κόστος 693,6481 και η απόκλισή της από το μέχρι στιγμής βέλτιστο (678,54) είναι:

$$\frac{693{,}65 - 678{,}54}{678{,}54} = 2{,}23\ \%$$

Η συγκεκριμένη λύση αποτελείται από 8 διαδρομές οι οποίες έχουν ως εξής:

1	88	93	94	97	95	98	116	111	117	104	…
…	105	108									
1	89	3	2	4	5	6	7	8	10	11	…
…	12	16	15	14	13	9					
1	99	68	70	71	72	75	76	73	79	81	…
…	80	78	77	74	69						
1	83	112	87	86	90	92	91	115	19	119	…
…	109	84	114	118	85	113	82	120			
101	53	55	58	60	66	62	63	65	67	101	…
…	64	61	57	59	56	54					
1	96	41	44	46	49	52	51	50	48	47	…
…	45	42	43	40	39	38					
1	110	22	27	24	21	18	17	20	26	23	…
…	25	28	31	34	35	32	29	33	36	37	30	…
1	121	106	107	103	102	100					

Η γραφική αναπαράσταση της λύσης για το παράδειγμα 11 έχει ως εξής:



5.3.12 Παράδειγμα 12

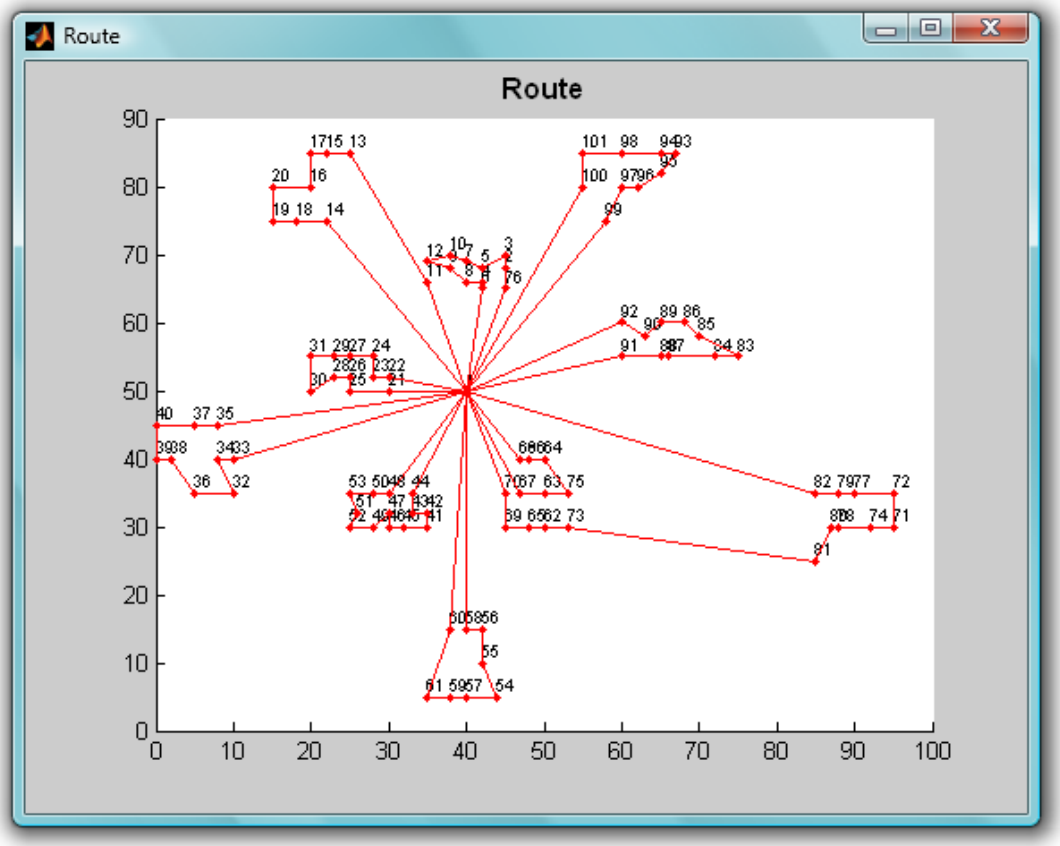
Στο συγκεκριμένο παράδειγμα υπάρχουν 101 κόμβοι κι όλα τα διαθέσιμα φορτηγά έχουν χωρητικότητα ίση με 200. Επίσης, δεν υπάρχουν χρονικοί περιορισμοί.

VRP:

Η λύση που βρέθηκε έχει κόστος ίδιο με το κόστος της μέχρι στιγμής βέλτιστης λύσης που έχει βρεθεί σε παγκόσμια κλίμακα (δηλαδή 819,56). Αυτή η λύση αποτελείται από τις εξής 10 διαδρομές:

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

Γραφικά η συγκεκριμένη λύση έχει ως εξής:



OVRP:

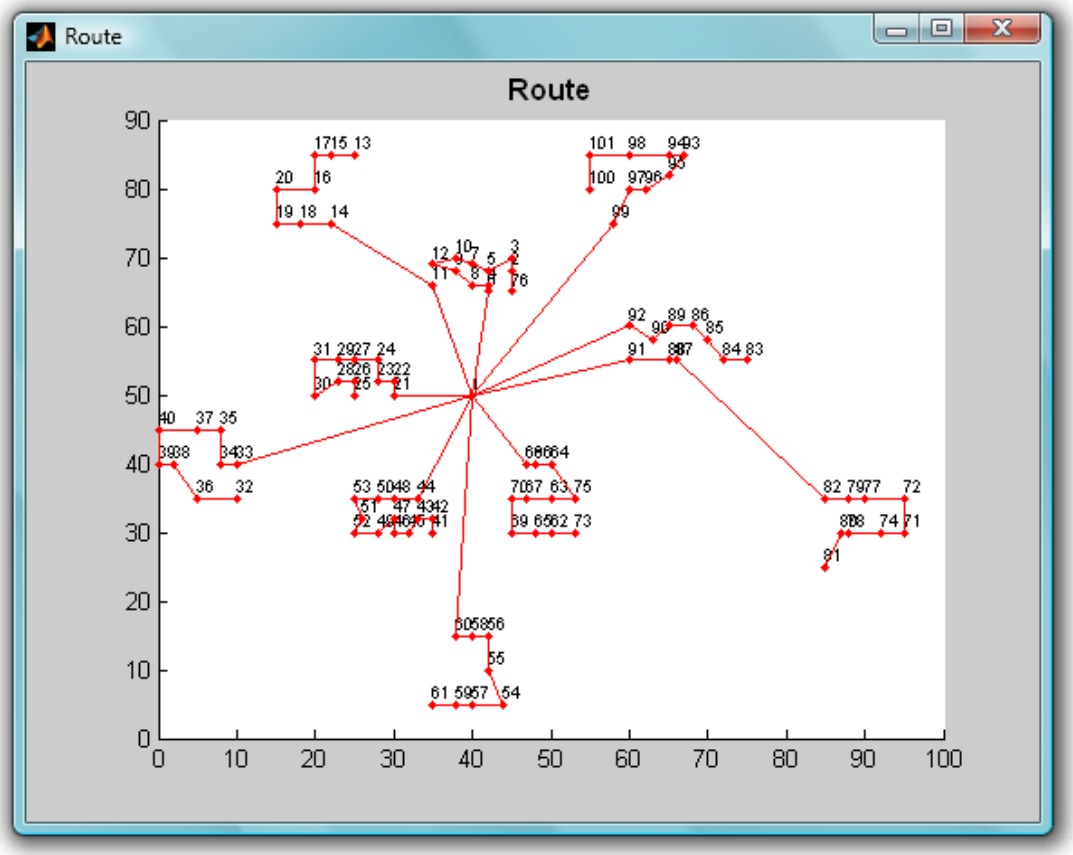
Η λύση που προέκυψε είχε κόστος 534,7104 και η απόκλισή της από το μέχρι στιγμής βέλτιστο σε παγκόσμια κλίμακα (534,24) είναι:

$$\frac{534{,}71 - 534{,}24}{534{,}24} = 0{,}09\%$$

Η λύση για το παράδειγμα 12 αποτελείται από 11 δρομολόγια και έχει ως εξής:

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

Η γραφική απεικόνιση της λύσης έχει ως εξής:



5.3.13 Παράδειγμα 13

Στο παράδειγμα 13 υπάρχουν 121 κόμβοι. Επίσης, θεωρείται ότι όλα τα διαθέσιμα φορτηγά έχουν χωρητικότητα ίση με 200. Ακόμα, κανένα δρομολόγιο δεν πρέπει να έχει συνολικό μήκος πάνω από 720.

VRP:

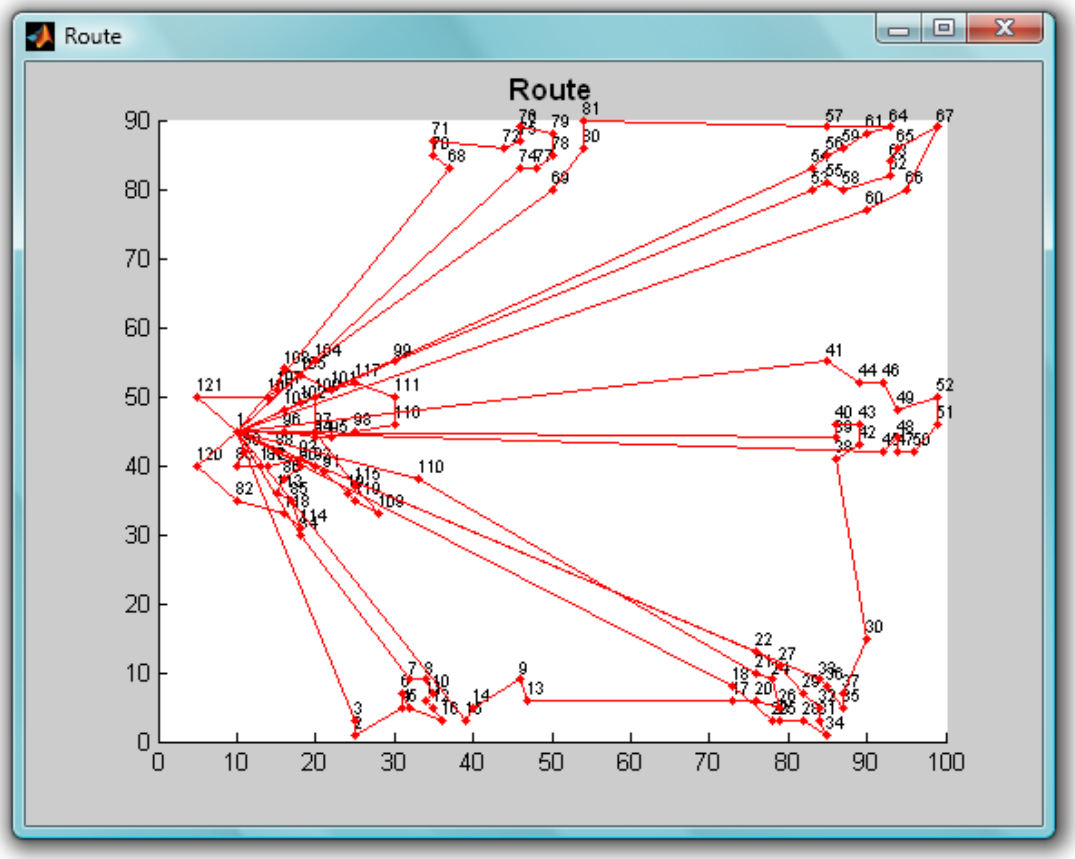
Η λύση που προέκυψε είχε κόστος 1563,8016. Η απόκλισή της από τη μέχρι στιγμής βέλτιστη λύση που έχει βρεθεί σε παγκόσμια κλίμακα (1541,14) είναι:

$$\frac{1563{,}8016 - 1541{,}14}{1541{,}14} = 1{,}47\%$$

Αυτή η λύση αποτελείται από 11 διαδρομές, Οι διαδρομές αυτές έχουν ως εξής:

1	54	56	59	61	64	57	81	80	69	104	1			
1	33	36	35	37	30	38	42	43	40	39	1			
1	41	44	46	49	52	51	50	47	48	45	1			
1	22	27	29	32	31	34	28	25	23	18	1			
1	74	77	78	79	73	76	75	72	71	70	68	1		
1	60	66	67	65	63	62	58	55	53	99	1			
1	103	102	100	97	115	109	119	19	91	92	88	1		
1	121	106	107	108	105	101	117	111	116	98	95	94	96	1
1	89	83	112	87	93	90	86	113	85	114	118	82	120	1
1	110	21	24	26	20	17	13	9	14	15	1			
1	3	2	4	6	5	16	12	11	10	8	7	84	1	

Γραφικά η συγκεκριμένη λύση έχει ως εξής:



OVRP:

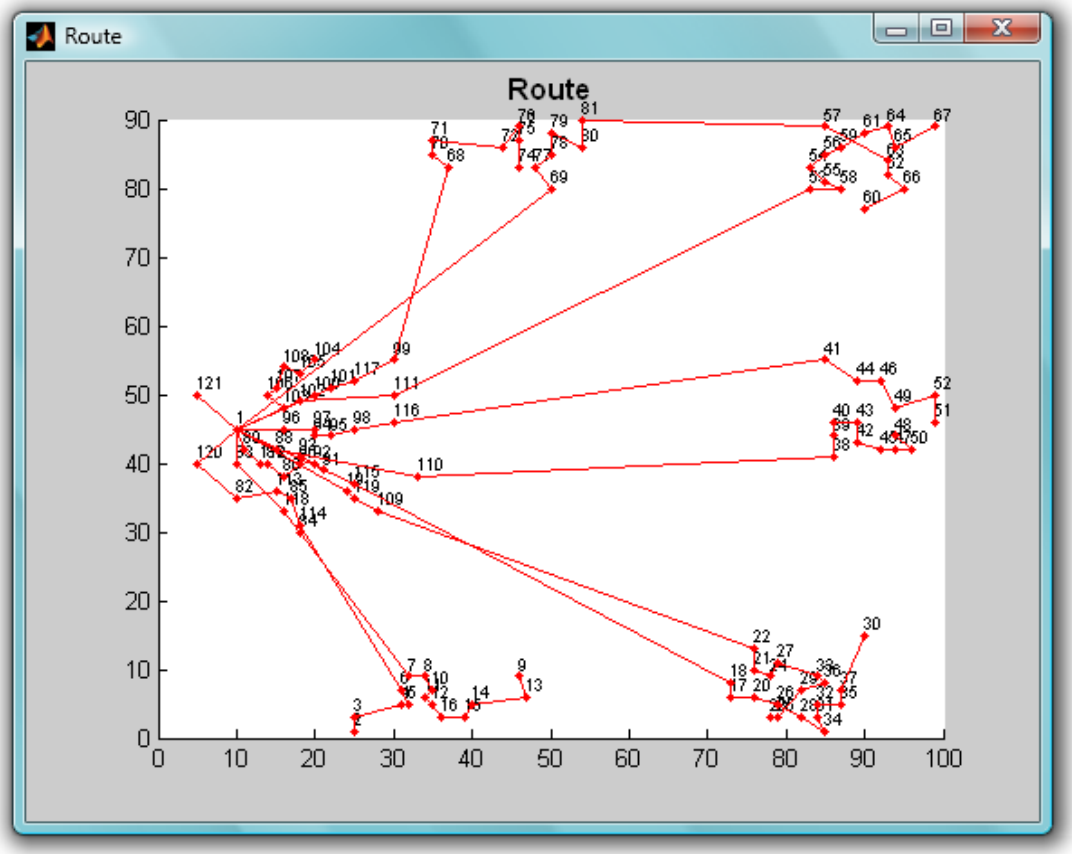
Η λύση που προέκυψε έχει κόστος 885,0669 και η απόκλισή της από το μέχρι στιγμής βέλτιστο (836,55) είναι:

$$\frac{885{,}07 - 836{,}55}{836{,}55} = 5{,}80\%$$

Αυτή η λύση αποτελείται από 12 διαδρομές, Οι διαδρομές αυτές έχουν ως εξής:

1	88	110	38	39	40	43	42	45	47	50	48	
1	121											
1	102	111	53	58	55	54	56	59	61	64	65	67
1	115	18	17	20	26	28	34	31	32	35	37	30
1	69	77	78	79	80	81	57	63	62	66	60	
1	19	119	109	22	21	24	27	33	36	29	25	23
1	100	101	117	99	68	70	71	72	76	73	75	74
1	89	112	87	86	90	93	92	91				
1	83	118	84	7	8	10	11	12	16	15	14	13	9
1	103	106	107	108	105	104						
1	120	82	113	85	114	6	5	4	3	2		
1	96	97	94	95	98	116	41	44	46	49	52	51

Γραφικά η συγκεκριμένη λύση έχει ως εξής:



5.3.14 Παράδειγμα 14

Στο συγκεκριμένο παράδειγμα υπάρχουν 121 κόμβοι. Καμία διαδρομή δεν πρέπει να έχει συνολικό μήκος πάνω από 720 μονάδες. Επίσης, θεωρείται ότι όλα τα διαθέσιμα φορτηγά έχουν χωρητικότητα, ίση με 200.

VRP:

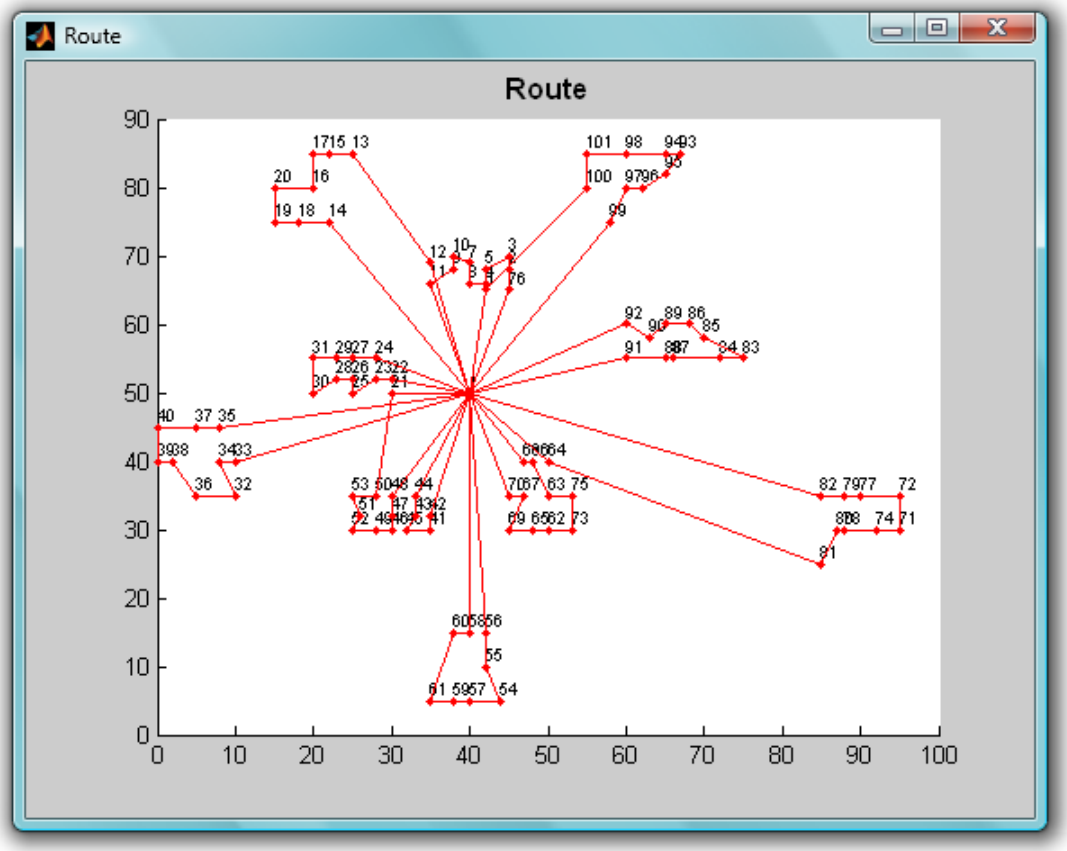
Η λύση που προέκυψε είχε κόστος 867,17 και η απόκλισή της από το μέχρι στιγμής βέλτιστο σε παγκόσμια κλίμακα (866,37) είναι:

$$\frac{867{,}17 - 866{,}37}{866{,}37} = 0{,}09\%$$

Η λύση για το παράδειγμα 14 αποτελείται από 11 δρομολόγια και έχει ως εξής:

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

Η γραφική απεικόνιση της λύσης έχει ως εξής:



OVRP:

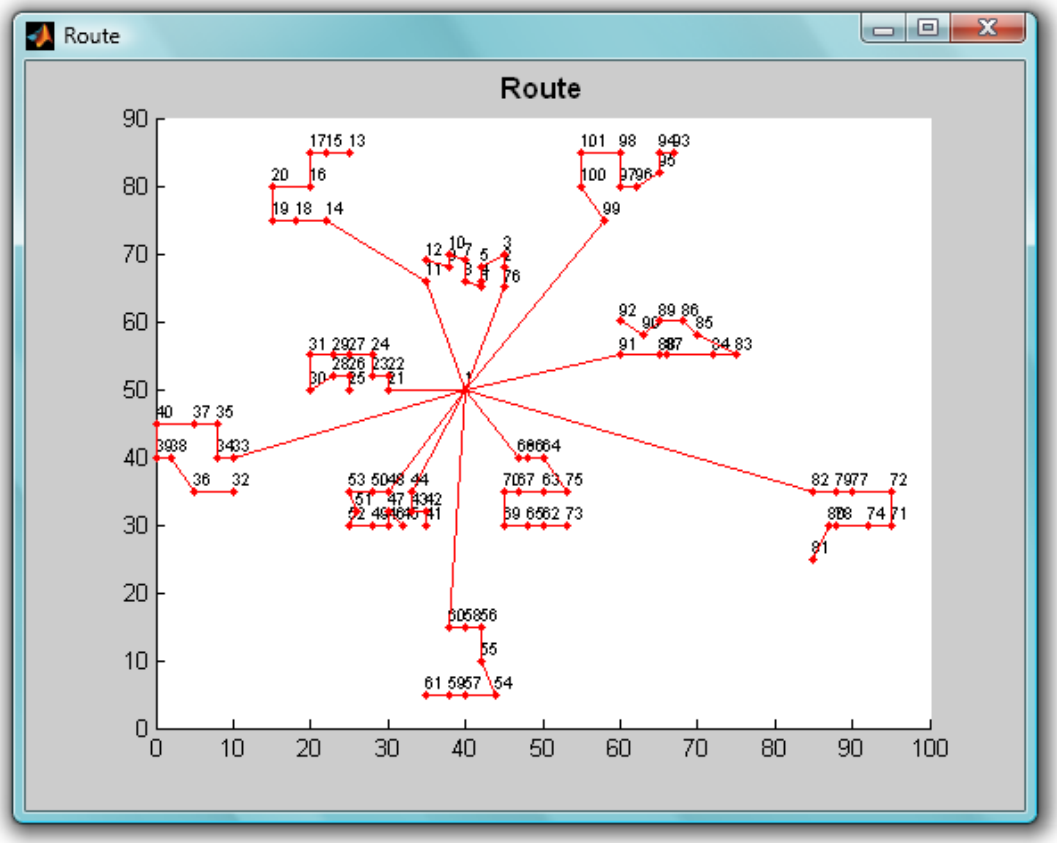
Η λύση που προέκυψε είχε κόστος 555,7429 και η απόκλισή της από το μέχρι στιγμής βέλτιστο σε παγκόσμια κλίμακα (552,64) είναι:

$$\frac{555{,}74 - 552{,}64}{552{,}64} = 0{,}56\%$$

Η λύση για το παράδειγμα 14 αποτελείται από 11 δρομολόγια και έχει ως εξής:

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

Η γραφική απεικόνιση της λύσης έχει ως εξής:



5.4 Αποτελέσματα για το Πρόβλημα του Περιπλανώμενου Πωλητή (TSP)

Στα πλαίσια της παρούσας μεταπτυχιακής διατριβής επιλύθηκαν τα εξής προβλήματα:

eil:

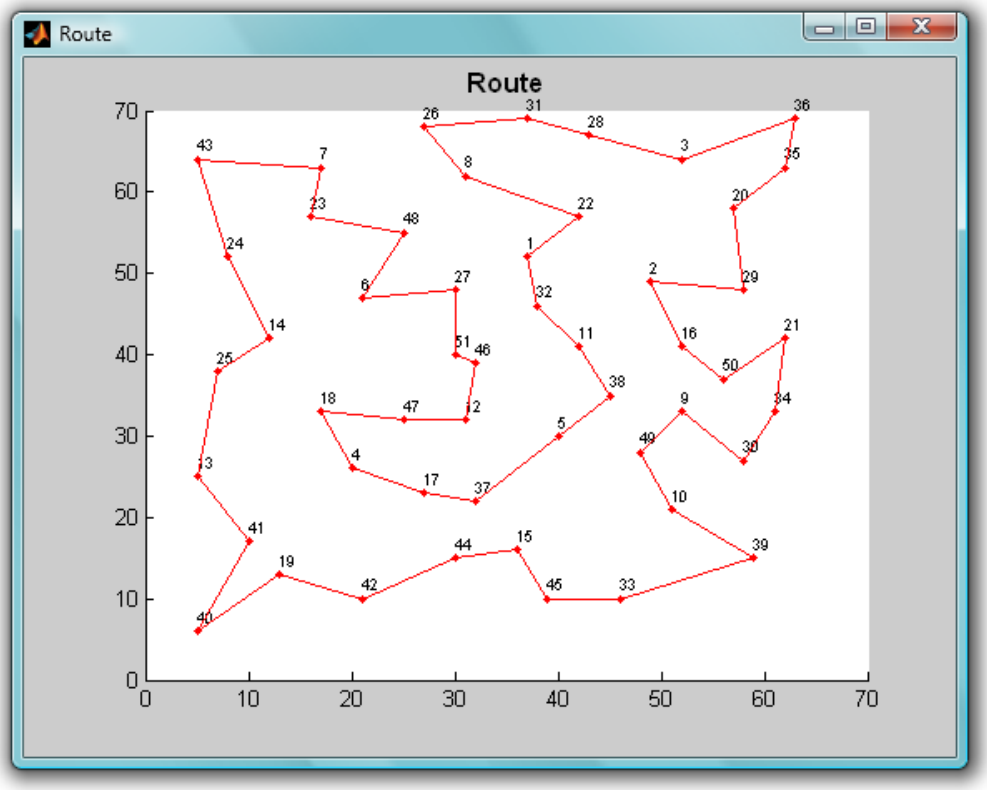
Στο παράδειγμα eil υπάρχουν 51 κόμβοι. Η λύση που προέκυψε έχει κόστος 428,8718 . Η απόκλισή της από τη βέλτιστη λύση που έχει βρεθεί σε παγκόσμια κλίμακα (426) είναι:

$$\frac{428{,}87 - 426}{426} = 0{,}67\%$$

Η συγκεκριμένη λύση έχει ως εξής:

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

Γραφικά η συγκεκριμένη λύση έχει ως εξής:



berlin52:

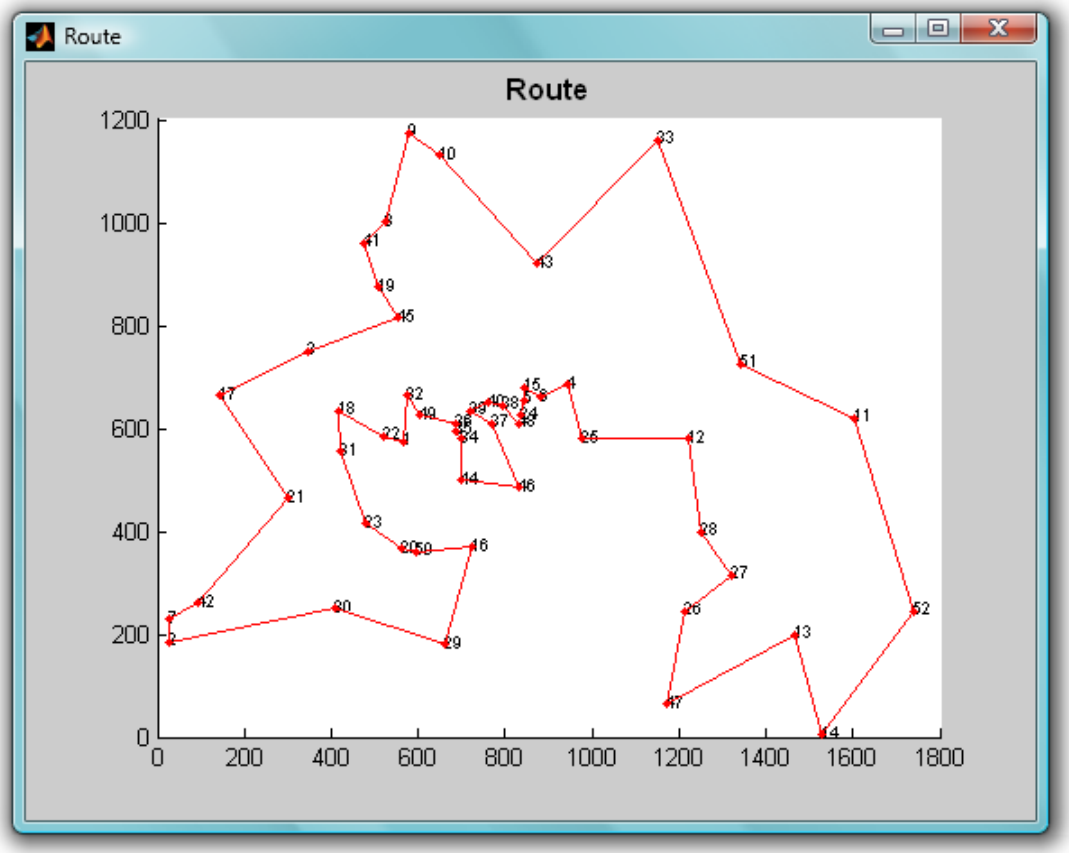
Στο συγκεκριμένο παράδειγμα υπάρχουν 52 κόμβοι. Η λύση έχει κόστος 7716,6859 . Η απόκλισή της από τη βέλτιστη λύση (7542) είναι:

$$\frac{7716{,}69\ -7542}{7542} = 2{,}26\%$$

Η λύση για το συγκεκριμένο παράδειγμα έχει ως εξής:

1	32	49	36	35	34	44	46	37	39	40	38	48	24	5	15	…
…	6	4	25	12	28	27	26	47	13	14	52	11	51	33	43	…
…	10	9	8	41	19	45	3	17	21	42	7	2	30	29	16	…
…	50	20	23	31	18	22	1									

Η γραφική αναπαράσταση της λύσης είναι η εξής:



eil76:

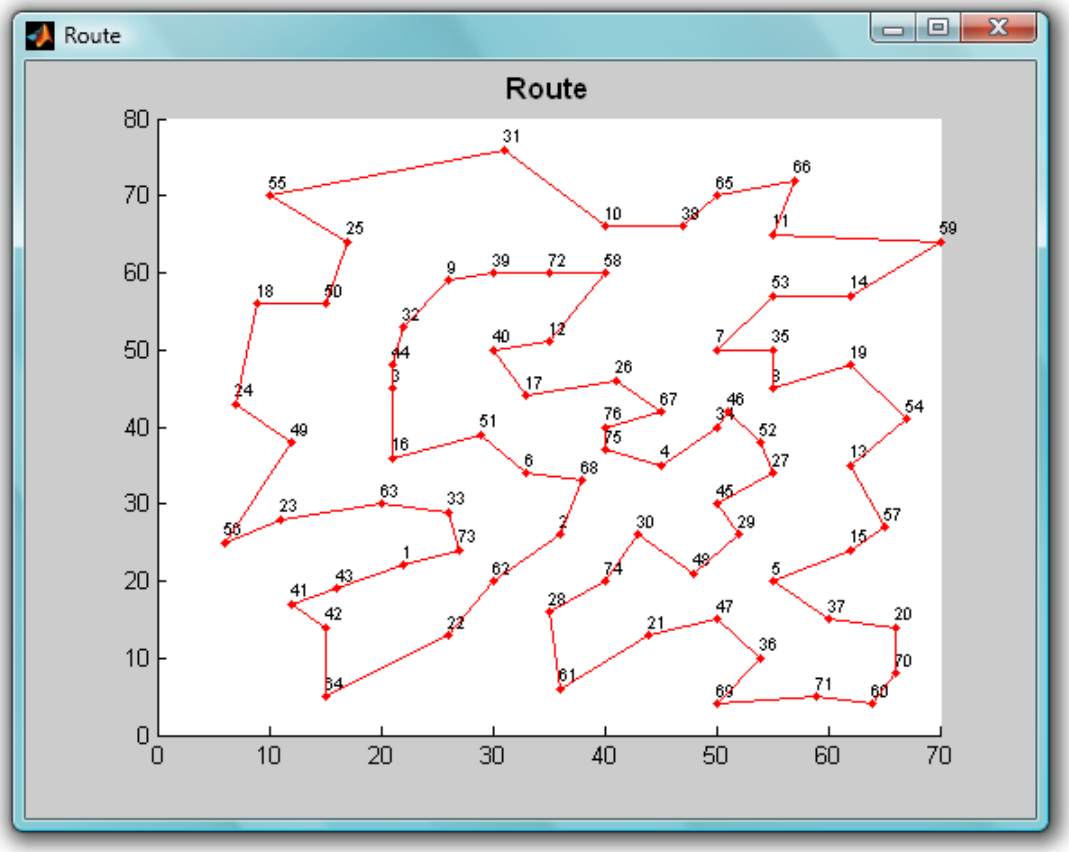
Σε αυτό το παράδειγμα υπάρχουν 76 κόμβοι. Η λύση για το συγκεκριμένο παράδειγμα έχει κόστος 552,0201 . Η απόκλισή της από τη βέλτιστη λύση (538) είναι:

$$\frac{552{,}02 \ - 538}{538} = 2{,}61\%$$

Η λύση για το παράδειγμα eil76 έχει ως εξής:

1	43	41	42	64	22	62	2	68	6	51	16	3	44	32	9	…
…	39	72	58	12	40	17	26	67	76	75	4	34	46	52	27	…
…	45	29	48	30	74	28	61	21	47	36	69	71	60	70	20	…
…	37	5	15	57	13	54	19	8	35	7	53	14	59	11	66	…
…	65	38	10	31	55	25	50	18	24	49	56	23	63	33	73	1

Η γραφική αναπαράσταση της λύσης είναι η εξής:



kroA100:

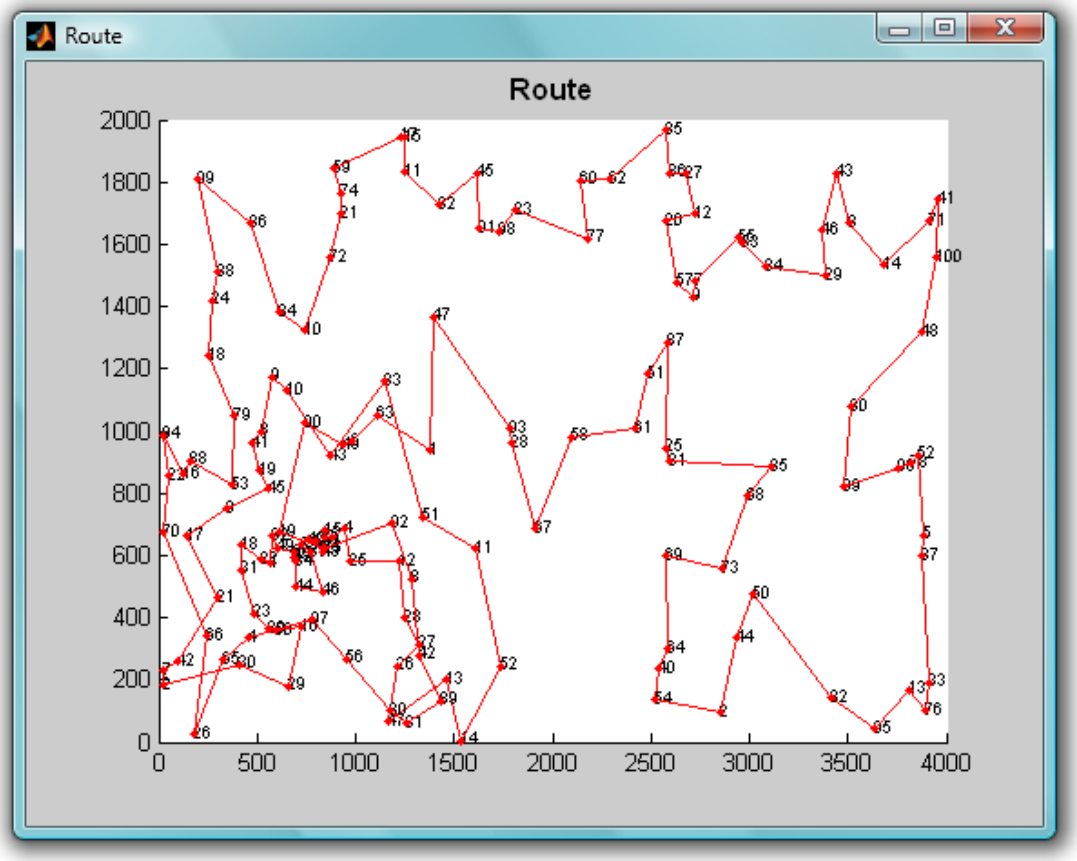
Στο παράδειγμα kroA100 υπάρχουν 100 κόμβοι. Η λύση που προέκυψε έχει κόστος 21322,7366 . Η απόκλισή της από τη βέλτιστη λύση που έχει βρεθεί σε παγκόσμια κλίμακα (21282) είναι:

$$\frac{21322{,}7366 - 21282}{21282} = 0{,}19\%$$

Η συγκεκριμένη λύση έχει ως εξής:

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

Γραφικά η συγκεκριμένη λύση έχει ως εξής:



d198:

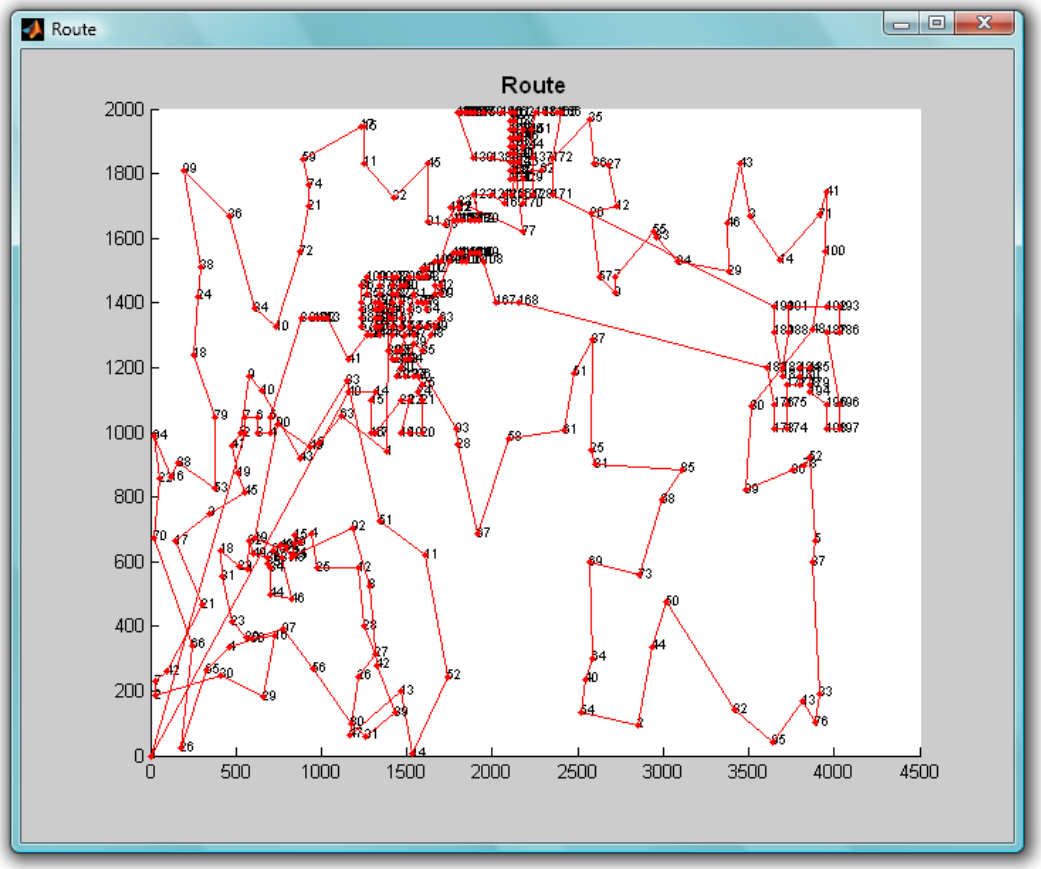
Στο συγκεκριμένο παράδειγμα υπάρχουν 198 κόμβοι. Η λύση έχει κόστος 16214,0126 . Η απόκλισή της από τη βέλτιστη λύση (15780) είναι:

$$\frac{16214{,}0126 - 15780}{15780} = 2{,}75\%$$

Η λύση για το συγκεκριμένο παράδειγμα έχει ως εξής:

1	40	14	15	16	17	23	22	18	19	20	21	24	25	26	27	…
…	28	29	30	33	34	39	35	48	49	63	50	51	47	52	65	
…	81	76	77	64	78	80	91	79	92	105	114	113	112	106	107	…
…	111	110	109	108	167	168	182	176	173	174	175	177	178	180	184	…
…	185	179	194	195	198	197	196	187	186	193	192	191	188	183	181	…
…	189	190	171	172	166	165	164	163	151	137	128	170	127	129	136	…
…	144	150	145	146	149	148	153	152	162	161	160	159	158	157	156	…
…	155	154	139	138	134	140	142	147	143	141	135	133	132	131	130	…
…	126	125	169	124	123	120	119	118	117	116	121	122	115	104	103	…
…	102	93	101	94	95	96	90	89	82	75	62	53	46	36	37	…
…	32	31	38	45	54	61	55	44	56	59	60	67	70	66	74	…
…	83	88	97	98	99	87	84	73	68	72	85	100	86	71	69	…
…	58	57	43	42	41	13	12	11	10	9	8	5	4	3	6	…
…	7	2	1													

Η γραφική αναπαράσταση της λύσης είναι η εξής:



kroA200:

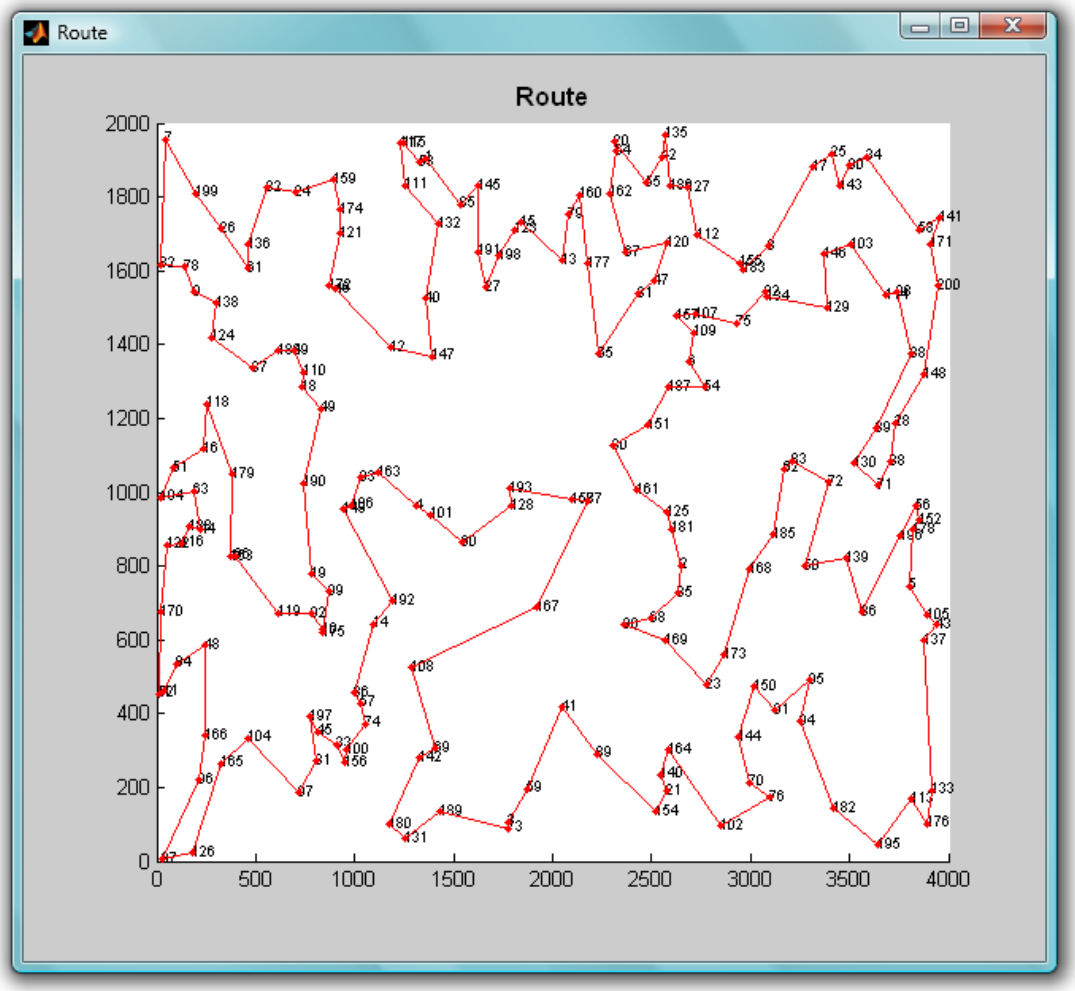
Σε αυτό το παράδειγμα υπάρχουν 200 κόμβοι. Η λύση για το συγκεκριμένο παράδειγμα έχει κόστος 30514,3481 . Η απόκλισή της από τη βέλτιστη λύση (29368) είναι:

$$\frac{30514{,}3481 \; - 29368}{29368} = 3{,}90\ \%$$

Η λύση για το παράδειγμα kroA200 έχει ως εξής:

1	53	115	117	111	132	40	147	12	46	172	121	174	159	24	32	...
...	136	61	26	199	7	82	78	9	138	124	37	184	29	110	18	...
...	49	190	19	99	175	10	92	119	66	153	179	118	16	51	194	...
...	63	44	188	116	122	170	52	11	84	48	166	96	87	126	165	...
...	104	97	81	197	45	33	156	100	74	57	36	14	192	149	106	...
...	93	163	4	101	60	128	193	158	77	167	108	69	142	180	131	...
...	189	73	3	59	41	89	154	21	140	164	102	76	70	144	150	...
...	91	95	94	182	195	113	176	133	137	43	105	5	178	152	56	...
...	196	86	139	50	72	83	62	185	168	173	23	169	30	68	35	...
...	2	181	125	161	80	151	187	54	6	109	157	107	75	22	134	...
...	129	146	103	114	98	88	39	130	71	38	28	148	200	171	141	...
...	58	34	90	143	25	17	8	183	155	112	127	186	135	42	55	...
...	20	64	162	67	120	47	31	65	177	160	79	13	15	123	198	...
...	27	191	145	85	1											

Η γραφική αναπαράσταση της λύσης είναι η εξής:



5.5 Αποτελέσματα για το Πρόβλημα Δρομολόγησης Οχημάτων με Χρονικά Παράθυρα (Time Windows VRP)

Για τα προβλήματα Δρομολόγησης Οχημάτων με Χρονικά Παράθυρα (VRPTW) χρησιμοποιήθηκαν τα παραδείγματα (C101 – C106), των 100 κόμβων. Κάθε κόμβος (πελάτη ή αποθήκης) αντιστοιχεί σε συγκεκριμένες συντεταγμένες θέσης, σε συγκεκριμένη ζήτηση (όπου η ζήτηση για τις αποθήκες είναι ίση με 0) και σε συγκεκριμένα χρονικά παράθυρα (ενωρίτερο χρόνο και βραδύτερο χρόνο). Σε όλα τα παραδείγματα ο κόμβος 1 αντιστοιχεί στην αποθήκη. Επίσης σε όλα τα παρακάτω παραδείγματα η χωρητικότητα των φορτηγών είναι ίση με 200. Αναλυτικά για κάθε παράδειγμα οι καλύτερες λύσεις για το πρόβλημα Δρομολόγησης Οχημάτων με Χρονικά Παράθυρα (VRPTW) έχουν ως εξής:

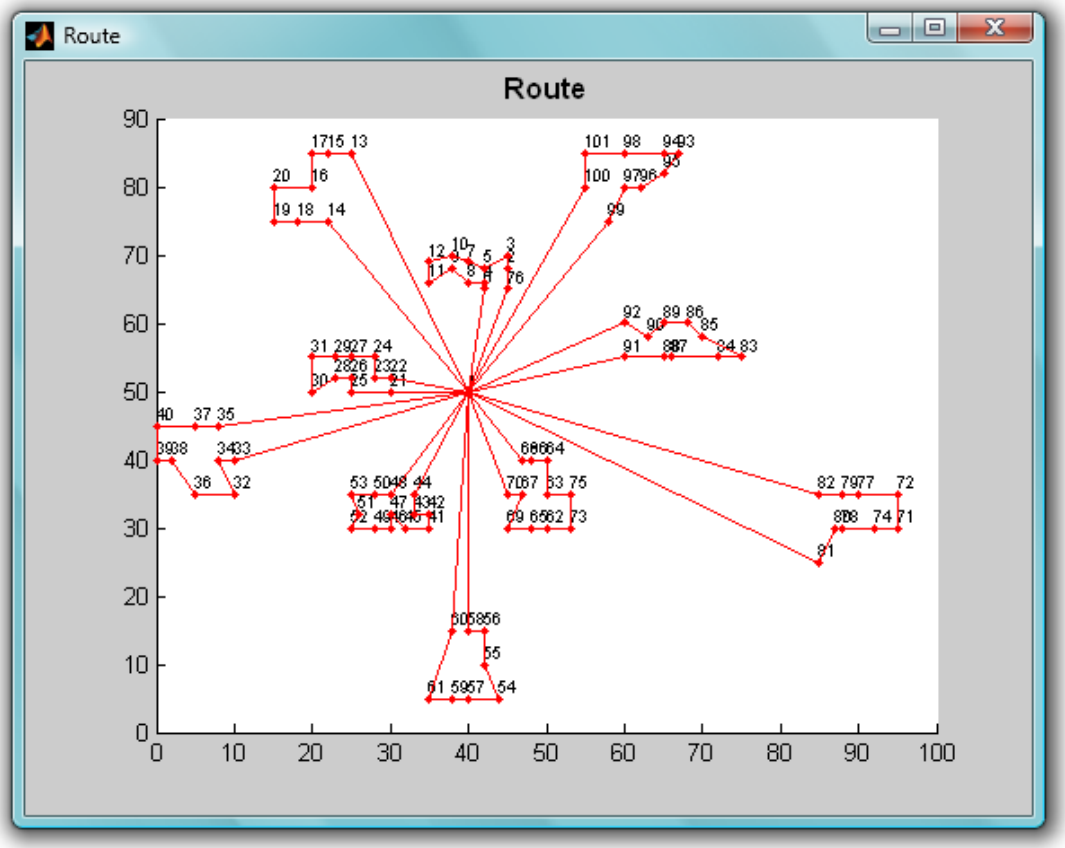
C101:

Η λύση που προέκυψε είχε κόστος 828,9369. Το συγκεκριμένο κόστος είναι το ίδιο με αυτό της μέχρι στιγμής βέλτιστης λύσης που έχει βρεθεί σε παγκόσμια κλίμακα (828,94).

Η λύση για το παράδειγμα C101 αποτελείται από 10 δρομολόγια και έχει ως εξής:

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

Η γραφική απεικόνιση της λύσης έχει ως εξής:



C102:

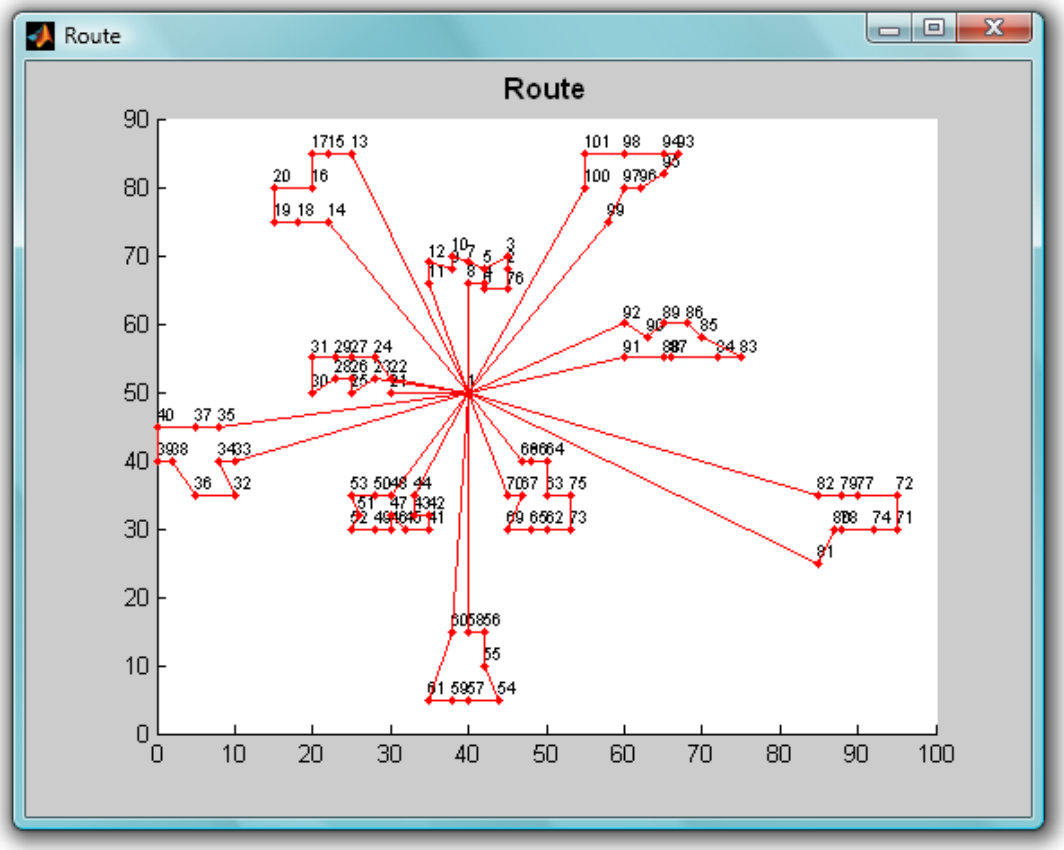
Η λύση που προέκυψε είχε κόστος 848,6984 και η απόκλισή της από το μέχρι στιγμής βέλτιστο σε παγκόσμια κλίμακα (828,94) είναι:

$$\frac{848{,}70 - 828{,}94}{828{,}94} = 2{,}38\%$$

Η λύση για το παράδειγμα C102 αποτελείται από 11 δρομολόγια και έχει ως εξής:

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

Η γραφική απεικόνιση της λύσης έχει ως εξής:



C103:

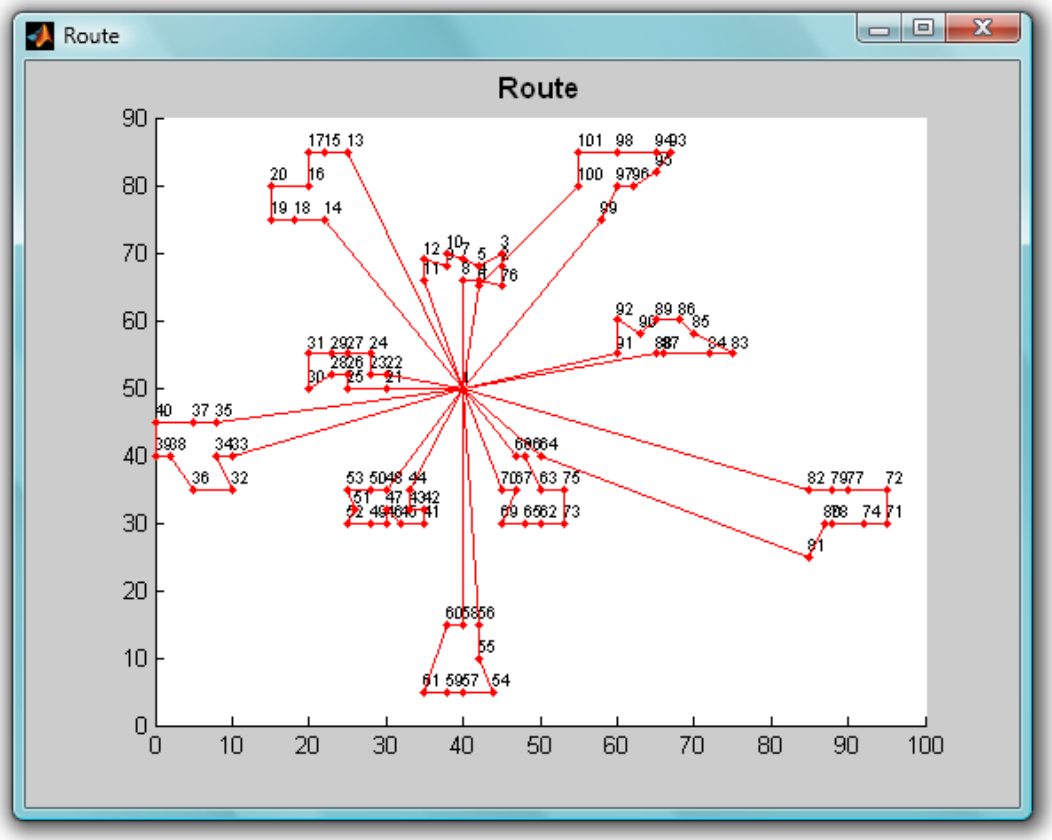
Η λύση που προέκυψε είχε κόστος 832,1877 και η απόκλισή της από το μέχρι στιγμής βέλτιστο παγκοσμίως(828,06) είναι:

$$\frac{832{,}19 - 828{,}06}{828{,}06} = 0{,}50\ \%$$

Η λύση για το παράδειγμα C103 αποτελείται από 10 διαδρομές. Οι συγκεκριμένες 10 διαδρομές έχουν ως εξής:

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

Σχηματικά η συγκεκριμένη λύση έχει ως εξής:



C104:

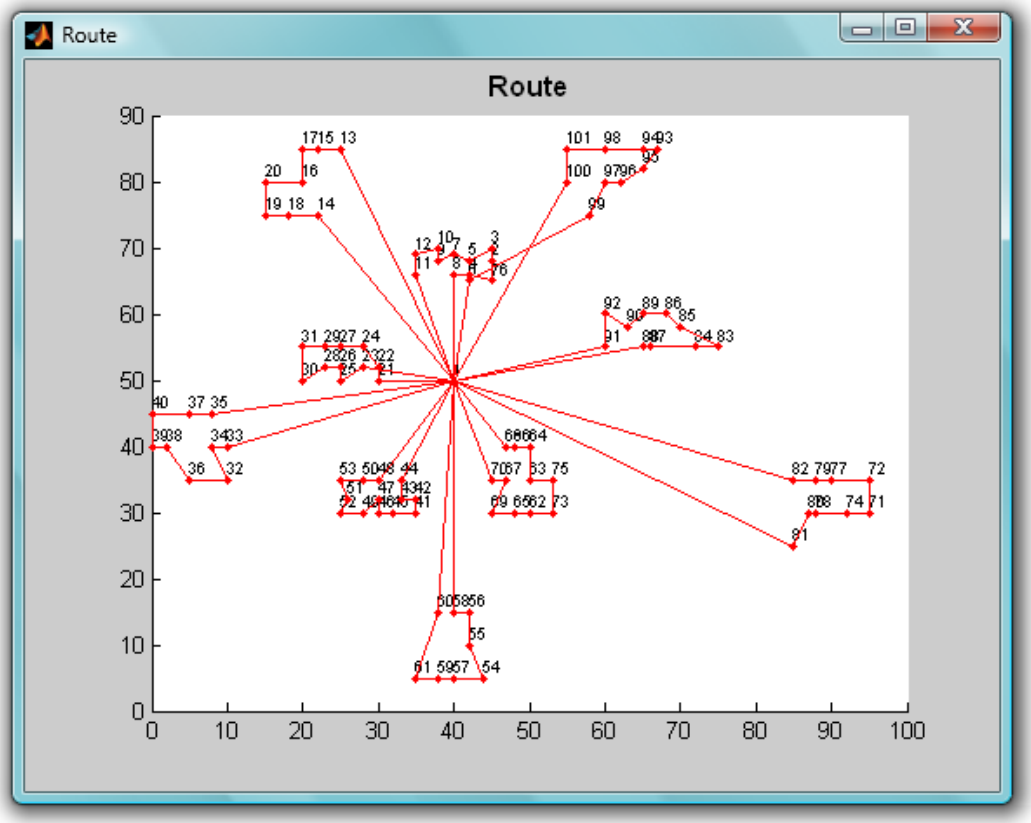
Η λύση που προέκυψε είχε κόστος 835,9920 και η απόκλισή της από το μέχρι στιγμής βέλτιστο σε παγκόσμια κλίμακα (824,78) είναι:

$$\frac{835{,}99 - 824{,}78}{824{,}78} = 1{,}36\%$$

Η λύση για το συγκεκριμένο παράδειγμα αποτελείται από 10 δρομολόγια και έχει ως εξής:

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

Η γραφική απεικόνιση της λύσης έχει ως εξής:



C105:

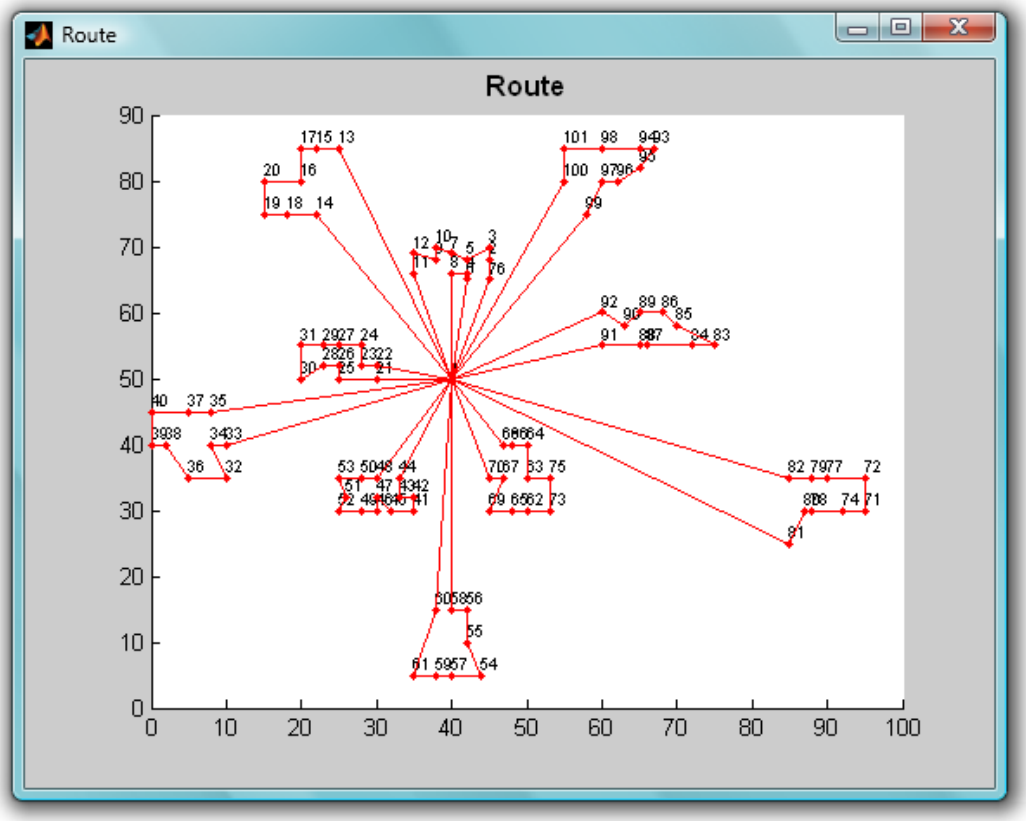
Η λύση που προέκυψε είχε κόστος 857,2659 και η απόκλισή της από το μέχρι στιγμής βέλτιστο σε παγκόσμια κλίμακα (828,94) είναι:

$$\frac{857{,}27 - 828{,}94}{828{,}94} = 3{,}42\%$$

Η λύση για το παράδειγμα C102 αποτελείται από 12 δρομολόγια και είναι η:

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

Η γραφική απεικόνιση της συγκεκριμένης λύσης έχει ως εξής:



C106:

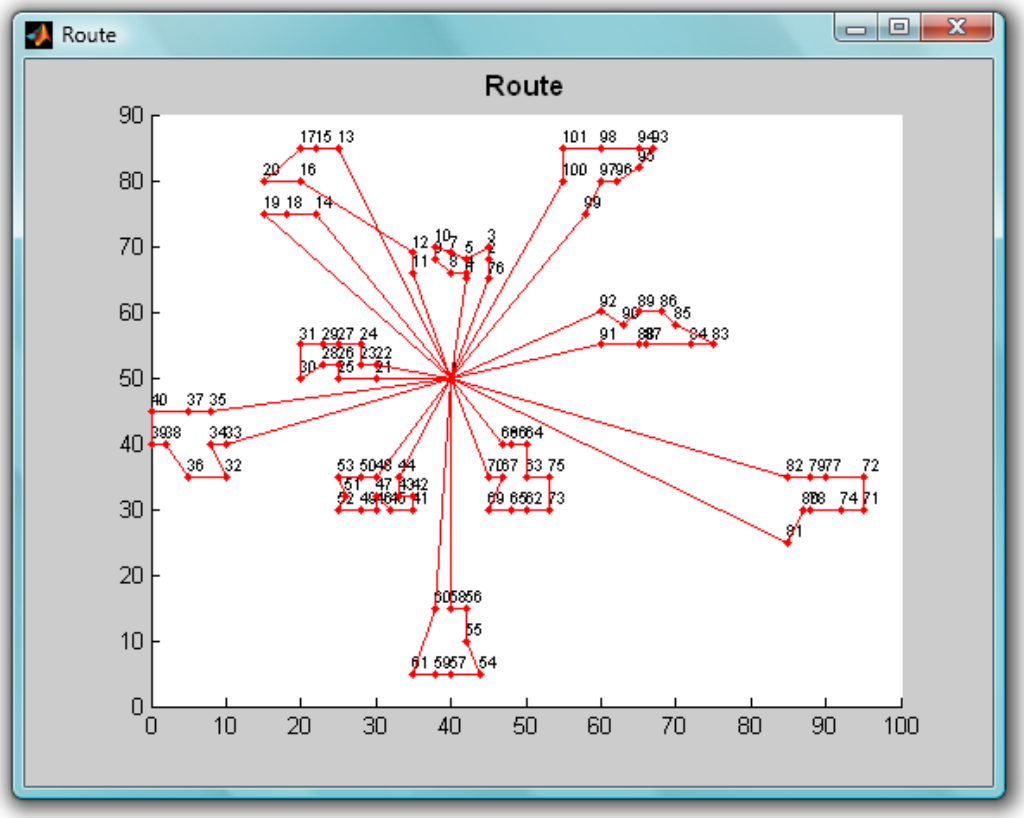
Η λύση που προέκυψε είχε κόστος 891,9596 και η απόκλισή της από το μέχρι στιγμής βέλτιστο σε παγκόσμια κλίμακα (828,94) είναι:

$$\frac{891{,}96 - 828{,}94}{828{,}94} = 7{,}60\%$$

Η λύση για το παράδειγμα C106 αποτελείται από 11 δρομολόγια και έχει ως εξής:

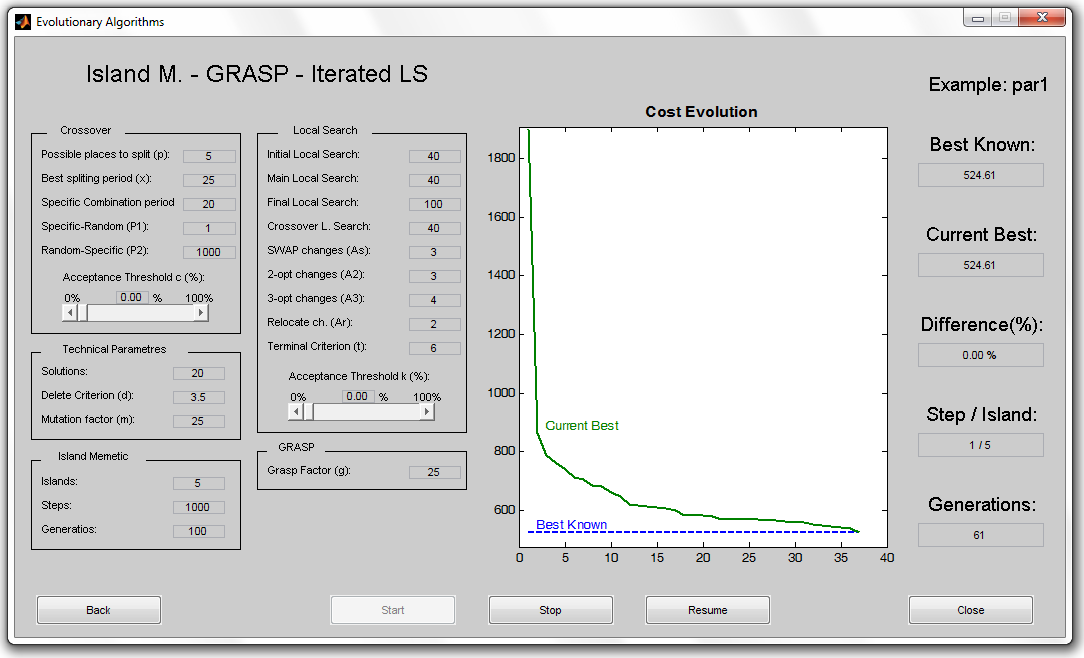
1	6	4	8	9	10	7	5	3	2	76	1			
1	99	97	96	95	93	94	98	101	100	1				
1	44	43	42	41	45	47	46	49	52	51	53	50	48	1
1	58	56	55	54	57	59	61	60	1					
1	68	66	64	63	75	73	62	65	69	67	70	1		
1	11	12	16	20	17	15	13	1						
1	91	88	87	84	83	85	86	89	90	92	1			
1	21	25	26	28	30	31	29	27	24	23	22	1		
1	33	34	32	36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

Η γραφική απεικόνιση της λύσης έχει ως εξής:



5.6 Σύγκριση Μεθόδων – Βέλτιστες Παράμετροι

Οι βέλτιστες τιμές των παραμέτρων του προβλήματος εξαρτώνται από πολλούς παράγοντες: τον τύπο του προβλήματος, τη μέθοδο που χρησιμοποιείται, το μέγεθος προβλήματος κτλ. Οι τιμές των παραμέτρων που αποδείχθηκαν αποτελεσματικότερες στην περίπτωσή μας είναι αυτές που φαίνονται στο επόμενο παράθυρο:



Βέβαια μέσα από το παράθυρο που φαίνεται παραπάνω, ο κάθε χρήστης έχει τη δυνατότητα να πειραματιστεί σχετικά με τις βέλτιστες τιμές των παραμέτρων και τις προσαρμόσει στην εκάστοτε περίπτωση.

Για τη σύγκριση των μεθόδων κατασκευάστηκε ένας συγκεντρωτικός πίνακας στον οποίο κάθε μέθοδος αντιπροσωπεύεται από τον αντίστοιχο κωδικό αριθμό, όπως φαίνεται στη σελίδα 6. Για τα προβλήματα VRP και OVRP, χρησιμοποιήθηκε το παράδειγμα par1 και για το πρόβλημα TSP χρησιμοποιήθηκε το παράδειγμα eil. Και τα δύο παραδείγματα χρησιμοποιούνται παγκοσμίως για τις συγκρίσεις αλγορίθμων εφοδιαστικής αλυσίδας.

	VRP			OVRP			TSP		
Μέθο-δος	Αποτέ-λεσμα	Βέλ-τιστο (par1)	Απόκλι-ση (%)	Αποτέ-λεσμα	Βέλ-τιστο (par1)	Απόκλι-ση (%)	Αποτέ-λεσμα	Βέλ-τιστο (eil)	Απόκλι-ση (%)
1	524,61	524,61	0,00%	412,9568	412,95	0,00%	433,60	426	1,78%
2	524,61	524,61	0,00%	412,9568	412,95	0,00%	431,32	426	1,25%
3	524,61	524,61	0,00%	412,9568	412,95	0,00%	429,53	426	0,83%
4	524,61	524,61	0,00%	412,9568	412,95	0,00%	429,48	426	0,82%
5	530,29	524,61	1,08%	424,3981	412,95	2,77%	430,24	426	1,00%
6	533,41	524,61	1,68%	425,2686	412,95	2,98%	435,92	426	2,33%
7	524,61	524,61	0,00%	412,9568	412,95	0,00%	432,99	426	1,64%
8	524,61	524,61	0,00%	412,9568	412,95	0,00%	429,11	426	0,73%
9	524,61	524,61	0,00%	412,9568	412,95	0,00%	428,98	426	0,70%
10	524,61	524,61	0,00%	412,9568	412,95	0,00%	428,87	426	0,67%

Από τον παραπάνω πίνακα φένεται ξεκάθαρα η υπεροχή της 10^{ης} μεθόδου (δηλαδή της Επαναληπτικής Τοπικής Αναζήτησης στο Μιμητικό Αλγόριθμο Πολλαπλών Πληθυσμών-Νησιών με χρήση Άπληστης Τυχοποιημένης Προσαρμοστικής Αναζήτησης (Iterated Local Search in Island Memetic with GRASP).

6. Συμπεράσματα

Η χρήση του Συστήματος Υποστήριξης Αποφάσεων που κατασκευάστηκε στα πλαίσια της παρούσας μεταπτυχιακής διατριβής είναι ιδιαίτερα εύκολη ακόμα και από κάποιον που δεν έχει ιδιαίτερη επαφή με το αντικείμενο των εξελικτικών αλγορίθμων. Για κάθε πιθανή κίνηση του χρήστη δίνονται οδηγίες και επίσης έχει προβλεφθεί κάθε πιθανό λάθος (και συνδυασμός λαθών) του χρήστη.

Σε περίπτωση λάθους κίνησης, αυτόματα αποτρέπεται η συγκεκριμένη κίνηση και δίνονται εκ νέου οδηγίες στο χρήστη για το πώς θα δράσει σωστά. Επίσης δίνεται η ευκαιρία στο χρήστη να πειραματιστεί με τις παραμέτρους του προβλήματος, έχοντας τη δυνατότητα να αλλάξει τις σημαντικότερες παραμέτρους για τον εκάστοτε αλγόριθμο.

Όσον αφορά στο αλγοριθμικό κομμάτι, το συγκεκριμένο σύστημα επιλύει οποιοδήποτε πραγματικό πρόβλημα αλλά και τα “instances” που χρησιμοποιούνται για τη σύγκριση των αλγορίθμων σε παγκόσμια κλίμακα.

Σχετικά με τα αποτελέσματα, όπως φαίνεται και από το προηγούμενο κεφάλαιο, οι αλγόριθμοι που κατασκευάστηκαν στα πλαίσια της παρούσας μεταπτυχιακής διατριβής ισοφάρισαν σε αρκετά “instances” το παγκόσμιο βέλτιστο και στα υπόλοιπα πέτυχαν πολύ μικρές αποκλίσεις. Ιδιαίτερα αποτελεσματική φάνηκε η μέθοδος της Επαναληπτικής Τοπικής Αναζήτησης στο Μιμητικό Αλγόριθμο Πολλαπλών Πληθυσμών-Νησιών με χρήση Άπληστης Τυχοποιημένης Προσαρμοστικής Αναζήτησης (Iterated Local Search in Island Memetic with GRASP). Η συγκεκριμένη μέθοδος είχε τις καλύτερες επιδόσεις από τις υπόλοιπες 9. Αυτό ήταν αναμενόμενο αφού η συγκεκριμένη μέθοδος συνδυάζει τη μέθοδο GRASP με συναρτήσεις τυχαιότητας αυξάνοντας τη μεταβλητότητα (διαφοροποίηση) μεταξύ των αρχικών λύσεων, οι οποίες είναι και καλής ποιότητας. Η διαφοροποίηση των λύσεων διατηρείται την περισσότερη ώρα αναζήτησης με τη χρήση της υπορουτίνας “Island Memetic”. Έτσι και οι λύσεις που παράγονται είναι πολύ καλής ποιότητας αφού χρησιμοποιείται τοπική αναζήτηση κι έτσι εντοπίζονται ευκολότερα τα τοπικά και ολικά βέλτιστα. Η χρήση της υπορουτίνας Iterated Local Search συντελεί στην ταχύτατη εύρεση πολύ καλής ποιότητας λύσεων.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] Yannis Marinakis - Athanasios Migdalas - Panos M. Pardalos. Multiple phase neighborhood Search—GRASP based on Lagrangean relaxation, random backtracking Lin–Kernighan and path relinking for the TSP. J Comb Optim 17: 134–156, 2007

[2] Ιωάννης Μαρινάκης, Αθανάσιος Μυγδαλάς. Σχεδιασμός και Βελτιστοποίηση Εφοδιαστικής Αλυσίδας. Εκδόσεις “σοφία”, Θεσσαλονίκη 2008

[3] L.S. Ochi, D.S. Vianna, L.M.A. Drummond, and A.O. Victor. An evolutionary hybrid metaheuristic for solving the vehicle routing problem with heterogeneous fleet. *Lecture notes in computer science*, 1391:187–195, 1998.

[4] L.S. Ochi, D.S. Vianna, L.M.A. Drummond, and A.O. Victor. A parallel evolutionary algorithm for the vehicle routing problem with heterogeneous fleet. *Parallel and Distributed Processing*, 1388:216–224, 1998.

[5] Ρογδάκης Ιωάννης. Μεμετικός Αλγόριθμος Πολλαπλών Πληθυσμών-Νησιών στο Πρόβλημα VRP. (ΔΙΠ) 2009

[6] Ανδρέου Χαράλαμπος. Βελτιστοποίηση Δρομολόγησης Οχημάτων Διανομής των κατεψυγμένων Προϊόντων της Παγκύπριας Εταιρίας Αρτοποιών. (ΔΙΠ) 2005.

[7] M. Gendreau, G. Laporte, C. Musaraganyi, and E.D. Taillard. A tabu search heuristic for the heterogeneous fleet vehicle routing problem. *Computers & Operations Research*, 26(12):1153–1173, 1999.

[8] N.A. Wassan and I.H. Osman. Tabu search variants for the mix fleet vehicle routing problem. *Journal of the Operational Research Society*, 53 (7):768–782, 2002.

[9] Y. Rochat and E.D. Taillard. Probabilistic diversification and intensification in local search for vehicle routing. *Journal of Heuristics*, 40:147–167, 1995.

[10] C.D. Tarantilis, C.T. Kiranoudis, and V.S. Vassiliadis. A list based threshold accepting metaheuristic for the heterogeneous fixed fleet vehicle routing problem. *Journal of the Operational Research Society*, 54(1):65–71,2003.

[11] C.D. Tarantilis, C.T. Kiranoudis, and V.S. Vassiliadis. A threshold accepting metaheuristic for the heterogeneous fixed fleet vehicle routing problem. *European Journal of Operational Research*, 152(1):148–158, 2004.

[12] F. Li, B.L. Golden, and E.A. Wasil. A record-to-record travel algorithm for solving the heterogeneous fleet vehicle routing problem. *Computers & Operations Research*, 34:2734–2742, 2007.

[13] B.L. Golden, E.Wasil, J. Kelly, and I.M. Chao. The impact of metaheuristic on solving the vehicle routing problem: algorithms, problem sets, and computational results. In T. Crainic and G. Laporte, editors, *Fleet Management and Logistics*, pages 33–56. Kluwer, Boston, MA, 1998.

[14] M. Dell'Amico, M. Monaci, C. Pagani, and D. Vigo. Heuristic approaches for the fleet size and mix vehicle routing problem with time windows. *Transportation Science*, 2007. To appear.

[15] F.H. Liu and S.Y. Shen. The fleet size and mix vehicle routing problem with time windows. *Journal of the Operational Research Society*, 50(7): 721–732, 1999.

[16] W. Dullaert, G.K. Janssens, K. Sorensen, and B. Vernimmen. New heuristics for the fleet size and mix vehicle routing problem with time windows. *Journal of the Operational Research Society*, 53(11):1232–1238, 2002.

[17] O. Bräysy, W. Dullaert, G. Hasle, D. Mester, and M. Gendreau. An effective multi-restart deterministic annealing methauristic for the fleet size and mix vehicle routing problem with time windows. *Transportation Science*, to appear, 2006.

[18] J.-F. Cordeau and G. Laporte. A tabu search algorithm for the site dependent vehicle routing problem with time windows. *INFOR*, 39:292–298, 2001.

[19] J.-F. Cordeau, M. Gendreau, and G. Laporte. A tabu search heuristic for periodic and multi–depot vehicle routing problems. *Networks*, 30:105–119, 1997.

[20] D. Pisinger and S. Ropke. A general heuristic for vehicle routing problems. *Comput. Oper. Res.*, 34(8):2403–2435, 2007.

[21] Roberto Baldacci, Maria Battarra, Daniele Vigo. *Routing a Heterogeneous Fleet of Vehicles* Bruce Golden, S. Raghavan, Edward Wasil, editors, The Vehicle Routing Problem: Latest advances and new challenges, pages 3-28, Springer Science+Business Media: New York, 2008.

[22] *Paolo Toth, Andrea Tramontani*. An Integer Linear Programming Local Search for Capacitated Vehicle Routing Problems*. In* Bruce Golden, S. Raghavan, Edward Wasil, editors, The Vehicle Routing Problem: Latest advances and new challenges, pages 275-296, Springer Science+Business Media: New York, 2008.

[23] I.H. Osman and S. Salhi. Local search strategies for the vehicle fleet mix problem. In V.J. Rayward-Smith, I.H. Osman, C.R. Reeves, and G.D. Smith, editors, *Modern Heuristic Search Methods*, pages 131–153. Wiley:Chichester, 1996