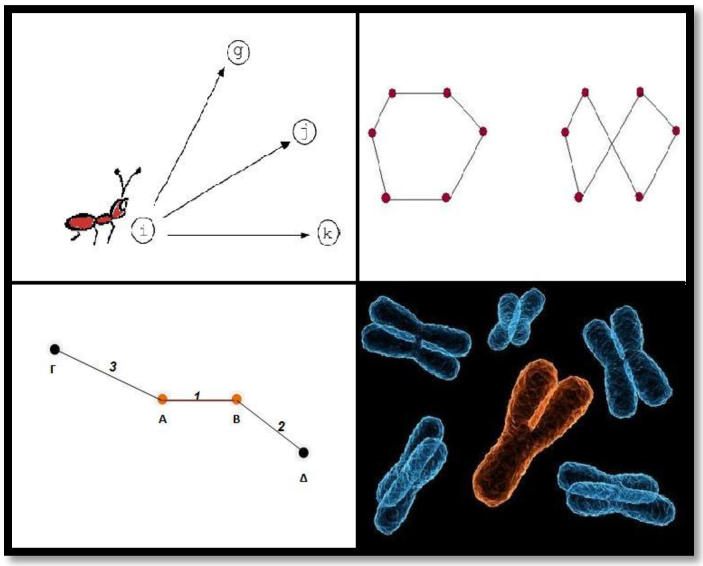
ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

Τμήμα Ηλεκτρονικών Μηχανικών και Μηχανικών Υπολογιστών

ΘΕΜΑ

ΑΝΑΠΤΥΞΗ ΥΠΟΛΟΓΙΣΤΙΚΟΥ ΣΥΣΤΗΜΑΤΟΣ ΓΙΑ ΤΗΝ ΕΠΙΛΥΣΗ ΠΡΟΒΛΗΜΑΤΩΝ ΕΦΟΔΙΑΣΤΙΚΗΣ ΑΛΥΣΙΔΑΣ



Διπλωματική εργασία:

ΒΕΝΙΖΕΛΟΣ ΑΘΑΝΑΣΙΟΣ

Επιβλέπων καθηγητής: **ΣΤΑΥΡΑΚΑΚΗΣ ΓΕΩΡΓΙΟΣ**

Δεύτερος Εξεταστής : **ΜΑΡΙΝΑΚΗΣ ΙΩΑΝΝΗΣ**

Τρίτος Εξεταστής : **ΖΕΡΒΑΚΗΣ ΜΙΧΑΗΛ**

ΧΑΝΙΑ 2010

Πρόλογος

Με την παρούσα διπλωματική εργασία κλείνει ο κύκλος σπουδών μου στο τμήμα Ηλεκτρονικών Μηχανικών και Μηχανικών Υπολογιστών του Πολυτεχνείου Κρήτης. Η εκπόνηση αυτής της εργασίας ξεκίνησε τον Μάιο του 2010 και ολοκληρώθηκε τον Οκτώβρη του 2010, υπό την επίβλεψη του κ.Ιωάννη Μαρινάκη.

Στο σημείο αυτό θα ήθελα να ευχαριστήσω όλους όσους συνέβαλλαν στην προσπάθειά μου αυτή, με οποιοδήποτε τρόπο.

Αρχικά θα ήθελα να ευχαριστήσω τους γονείς μου για την πολύτιμή τους βοήθεια και για τη στήριξή τους όλα αυτά τα χρόνια, επειδή χωρίς αυτούς δεν θα μπορούσα να φτάσω σε αυτό το σημείο, καθώς επίσης τους συγγενείς και τους φίλους μου για τη συμπαράστασή τους όλο αυτό τον καιρό.

Τέλος θα ήθελα να ευχαριστήσω τον επιβλέποντά μου, Ιωάννη Μαρινάκη, για τις πολύτιμες συμβουλές και την καθοδήγηση που μου παρείχε για την επιτυχή ολοκλήρωση της διπλωματικής μου εργασίας.

Περιεχόμενα

Εισαγωγή……………………………………………………………………………5

Κεφάλαιο 1:

Logistics και εφοδιαστική αλυσίδα………………………………………......6

1.1 Αναφορά στην εφοδιαστική αλυσίδα…………………………….………….....6

1.1.1 Ορισμός εφοδιαστικής αλυσίδας……………………….…………....6

1.1.2 Σκοπός εφοδιαστικής αλυσίδας………………………………….…..7

1.2 Ορισμός Logistics……………………………………………………………….8

1.2.1 Στόχοι και συστατικά της στρατηγικής των Logistics……………8

1.2.2 Χρησιμότητα των logistics στις επιχειρήσεις……………….........11

1.3 Logistics και εφοδιαστική αλυσίδα……………………………………………12

1.3.1Ψηφιακές τεχνολογίες σε logistics και εφοδιαστική αλυσίδα………………………………………………………………….....13

1.3.2 Οφέλη από logistics και εφοδιαστική αλυσίδα……………………14

Κεφάλαιο 2:

Διατύπωση του προβλήματος………………………………….…………........15

2.1 Πρόβλημα του πλανόδιου πωλητή (traveling salesman problem (TSP))…………………………………………..15

Κεφάλαιο 3:

Περιγραφή αλγορίθμων επίλυσης……………………………………………....19

3.1 Εισαγωγή…………………………………………………………………………..19

3.2 Ευρετικές Μέθοδοι…………………………………………………………………19

3.2.1 Αλγόριθμοι απληστίας………………………………………………19

3.2.1.1 Πλανόδιος πωλητής- (Standard Greedy Algorithm)……20

3.2.1.2 Πλανόδιος πωλητής- (Greedy Algorithm with the Round Robin Strategy) – Πλησιέστερος Γείτονας……………...21

3.2.2 Αλγόριθμοι τοπικής αναζήτησης (N - opt) …………………………22

3.2.2.1 Η μέθοδος N_opt – Περιγραφή …………………………......23

3.3 Μεθευρετικές μέθοδοι……………………………………………………........25

3.3.1 Αλγόριθμος Βελτιστοποίησης Αποικίας Μυρμηγκιών (Ant Colony Optimization)…….……..…………………..……......25

3.3.2 Γενετικοί Αλγόριθμοι –(Genetic Algorithms (GA))………..…………33

3.4 Εφαρμογή των αλγορίθμων που υλοποιήθηκαν……………………….........39

Κεφάλαιο 4:

Λογισμικά που χρησιμοποιήθηκαν ………………………………………………67

4.1 Το λογισμικό της MATLAB R2009b…………………………………………….67

4.2 Το λογισμικό του MICROSOFT SQL SERVER 2008………………………….69

4.3 Το λογισμικό του MICROSOFT VISUAL STUDIO 2010………………………69

Κεφάλαιο 5:

Αποτελέσματα …………………………………………………………………….85

Παράρτημα Α ………………………………………………………………………111

Βιβλιογραφία…………………………………………………………………..........112

Εισαγωγή

Η εφοδιαστική αλυσίδα είναι η διαδικασία του σχεδιασμού, της εφαρμογής και του ελέγχου της αποτελεσματικής ροής και αποθήκευσης πρώτων υλών, ημικατεργασμένων προϊόντων, τελικών προϊόντων και της σχετιζόμενης πληροφορίας από το σημείο προέλευσης στο σημείο κατανάλωσης με σκοπό τη συμμόρφωση στις απαιτήσεις των πελατών.

Στην παρούσα διπλωματική εργασία θα ασχοληθούμε με την επίλυση προβλημάτων που αντιμετωπίζουν οι επιχειρήσεις οι οποίες ασχολούνται με παραγωγή και διανομή προϊόντων και πιο συγκεκριμένα θα προχωρήσουμε στην εξέταση σύνθετων προβλημάτων της εφοδιαστικής κάνοντας έτσι τα πρώτα βήματα προς μια πιο ολοκληρωμένη αντιμετώπιση της διαχείρισης της εφοδιαστικής αλυσίδας.

Ένα από αυτά τα προβλήματα είναι το Πρόβλημα του Πλανόδιου Πωλητη ή αλλιώς Travelling Salesman Problem (TSP). Με την επίλυση αυτού του προβλήματος επιτυγχάνουμε τον υπολογισμό της βέλτιστης διαδρομής , με σκοπό την εξυπηρέτηση ενός συνόλου πελατών.

Συνοπτικά:

- Στο Κεφάλαιο 1 γίνεται μία αναφορά των θεμάτων με τα οποία ασχολούνται τα logistics και η εφοδιαστική αλυσίδα.

- Στο Κεφάλαιο 2 γίνεται η διατύπωση του προβλήματος του πλανόδιου πωλητή ή αλλιώς Travelling Salesman Problem (TSP) .

- Στο Κεφάλαιο 3 γίνεται μια αναλυτική περιγραφή των αλγορίθμων που χρησιμοποιήθηκαν (ευρετικών και μεθευρετικών) για την επίλυση του προβλήματος TSP.

- Στο Κεφάλαιο 4 γίνεται , μία αναφορά στα λογισμικά που χρησιμοποιήθηκαν , για να υλοποιηθεί η διεπαφή στην οποία εκτελόυμε το κάθε πρόβλημα.

- Στο Κεφάλαιο 5 θα δούμε τα αποτελέσματα που προκύπτουν.

- Στο Παράρτημα Α , παρατίθενται τα δεδομένα που χρησιμοποιήθηκαν.

- Στη Βιβλιογραφία παρατίθενται , όλες εκείνες οι πηγές στις οποίες στηρίχτηκαν πολλές , από τις επιλογές μας , για την υλοποίηση των αλγορίθμων .

Κεφάλαιο 1:

Logistics και εφοδιαστική αλυσίδα

1.1 Αναφορά στην εφοδιαστική αλυσίδα

Οδηγούμενοι από την παγκοσμιοποίηση και τις διαρκώς εκτεινόμενες απαιτήσεις των πελατών, η Εφοδιαστική Αλυσίδα (supply chain) παίζει καθοριστικό ρόλο στη δημιουργία πλεονεκτήματος για όλες τις επιχειρήσεις. Η ανάπτυξη μιας πλήρους και ανταποκρινόμενης Εφοδιαστικής Αλυσίδας που να καλύπτει τις απαιτήσεις των πελατών και να εξασφαλίζει αύξηση τόσο των μεριδίων της αγοράς όσο και της κερδοφορίας είναι κρίσιμη.

Σήμερα, οι επιχειρήσεις μετατοπίζουν το κέντρο βάρος των απαιτήσεών τους σε επιχειρηματικό λογισμικό που είναι σε θέση να υποστηρίζουν επιχειρηματικές διαδικασίες (business processes) και επιχειρηματικές ροές **(**workflow) πέρα από την αυτονόητη υποστήριξη εμπορολογιστικών διαδικασιών.

Οι επιχειρήσεις επιζητούν την καλύτερη υποστήριξη και διαχείριση της εφοδιαστικής αλυσίδας (Supply Chain Management), με ένα ολοκληρωμένο και ενιαίο τρόπο, ώστε να βελτιώσουν την ικανοποίηση του πελάτη (μειώνοντας τους χρόνους παράδοσης μέσω καλύτερου προγραμματισμού της αναμενόμενης ζήτησης και ελαχιστοποιώντας τα σφάλματα κατά την εκτέλεση των παραγγελιών) και να μειώσουν το κόστος (μειώνοντας τα αποθέματα και ελέγχοντας καλύτερα τις διαδικασίες προμηθειών και της παραγωγικής διαδικασίας).

Γίνεται ορατό ότι στο άμεσο μέλλον ο επιχειρηματικός ανταγωνισμός δεν θα γίνεται σε επίπεδο επιχειρήσεων αλλά Εφοδιαστικής Αλυσίδας, καθώς και το e-Business και η Τεχνολογία της Πληροφορικής θα αλλάζουν δραστικά τις απατήσεις και τους κανόνες του επιχειρείν. [25]

1.1.1 Ορισμός εφοδιαστικής αλυσίδας

Ο όρος "εφοδιαστική αλυσίδα" περιγράφει το πλέγμα διαδικασιών που απαιτούνται ώστε ένα προϊόν να περάσει από τη φάση της παραγωγής στη φάση της κατανάλωσης. Μεταξύ των διαδικασιών αυτών ξεχωρίζει η παραγωγή, η τυποποίηση, η αποθήκευση, η διακίνηση και η διάθεση του προϊόντος.

Επίσης, με τον όρο αυτό εννοούμε όχι μόνο τη ροή υλικών από τον προμηθευτή πρώτων υλών ή τον κατασκευαστή μέχρι τον τελικό καταναλωτή, αλλά παράλληλα και τη ροή πληροφοριών μεταξύ των μελών της ίδιας αλυσίδας. Η διαχείρισή της γίνεται σε δύο επίπεδα :

- **Επίπεδο προγραμματισμού:** στο επίπεδο αυτό, αναλύονται τα δεδομένα προμηθειών, αναλώσεων παραγωγής, αποθεματοποίησης και πωλήσεων, γίνονται προβλέψεις και πλάνα πάνω στα οποία βασίζεται ο προγραμματισμός.
- **Επίπεδο εκτέλεσης:** στο στάδιο αυτό εκτελείται το πλάνο που έχει καθοριστεί στο επίπεδο προγραμματισμού και παρακολουθείται η εξέλιξή του βάσει των δεδομένων και πληροφοριών που συλλέγονται από όλο το εύρος της Εφοδιαστικής Αλυσίδας.

1.1.2 Σκοπός εφοδιαστικής αλυσίδας

Η Εφοδιαστική Αλυσίδα αποτελεί ένα βασικό συντελεστή για την εφαρμογή του μάνατζμεντ. Ασχολείται με τον σχεδιασμό και την υλοποίηση των αποφάσεων και της στρατηγικής ενός οργανισμού, λαμβάνοντας υπόψη μοντέλα και τεχνικές λήψης αποφάσεων, τη σύγχρονη τεχνολογία της πληροφορικής και των επικοινωνιών, και το επιχειρησιακό περιβάλλον.

Επιπλέον, συμπεριλαμβάνει το συντονισμό και τη συνεργασία με όλους τους εταίρους του καναλιού εφοδιασμού, πού μπορεί να είναι προμηθευτές, μεσάζοντες, εταιρείες παροχής υπηρεσιών και πελάτες. Κατ' ουσία, η Διαχείριση Εφοδιαστικής Αλυσίδας ενοποιεί και ολοκληρώνει το σχεδιασμό, τις προμήθειες, την παραγωγή, την αποθήκευση, τη μεταφορά και τις πωλήσεις τόσο μέσα στις επιχειρήσεις όσο και μεταξύ αυτών.

Ο αντικειμενικός λοιπόν σκοπός της Διαχείρισης της Εφοδιαστικής Αλυσίδας είναι η αύξηση της συνολικής κερδοφορίας κατά μήκος της αλυσίδας που συνεπάγεται την αύξηση της κερδοφορίας όλων των εταίρων της. Αυτό επιτυγχάνεται με την κατανόηση και ικανοποίηση των πελατειακών αναγκών στον απαιτούμενο χρόνο, και με την προσφορά προϊόντων υψηλής προστιθέμενης αξίας και ανταγωνιστικού κόστους. Για την επίτευξη των παραπάνω στόχων, απαραίτητα χαρακτηριστικά των εφοδιαστικών αλυσίδων που ανταγωνίζονται μέσα στο σύγχρονο παγκοσμιοποιημένο περιβάλλον είναι η ευελιξία και η ταχεία προσαρμοστικότητα τους στις δυναμικά μεταβαλλόμενες συνθήκες.

1.2 Ορισμός logistics

Ο όρος logistics αποτελεί πολυσήμαντη και πολυσύνθετη έννοια, καλύπτοντας μια τεράστια γκάμα διαδικασιών σχεδιασμού, υλοποίησης και ελέγχου στο επιχειρηματικό πεδίο. Τα βασικά στοιχεία που συνυφαίνουν τα logistics είναι η διοίκηση και ο στρατηγικός σχεδιασμός της επιχείρησης, η βέλτιστη αξιοποίηση των έμψυχων (ανθρώπινων) και των άψυχων (υλικών) πόρων της, η παραγωγή, η αποθήκευση και η διανομή των αγαθών, από την πρώτη ύλη μέχρι το έτοιμο προϊόν και από την παραγωγή στο ράφι. Θεωρητικά τα logistics εξυπηρετούν την κερδοφορία μιας επιχείρησης, εξασφαλίζοντας τη συνεχή διαθεσιμότητα των προϊόντων και των λοιπών πόρων της, επιτρέποντας παράλληλα την ομαλή ροή επιτέλεσης των διαδικασιών που αναφέρθηκαν παραπάνω.

Τα logistics αποσκοπούν στην παραγωγή προϊόντων με όσο το δυνατόν χαμηλότερο κόστος, στη διατήρηση των προϊόντων με τον καλύτερο δυνατό τρόπο, στην πλήρη αξιοποίηση των υλικών μέσων της επιχείρησης, στη μεταφορά των προϊόντων με το χαμηλότερο δυνατό κόστος και τις μικρότερες δυνατές καθυστερήσεις και τελικά στην επίτευξη κερδοφορίας και οικονομίας κλίμακος για την επιχείρηση.

Αν και αφορούν σε κάθε είδους επιχειρηματικό τομέα και κάθε είδους επιχείρηση, εκεί που βρίσκουν κατεξοχήν πρόσφορο έδαφος εφαρμογής είναι οι επιχειρήσεις που δραστηριοποιούνται στην εφοδιαστική αλυσίδα (εμπορικές, μεταφορικές, παραγωγικές, εταιρίες αποθήκευσης κ.ά.), ανεξαρτήτως του τομέα της οικονομίας στον οποίο ανήκουν. [27]

1.2.1 Στόχοι και συστατικά της στρατηγικής των Logistics

Γενικότερα η στρατηγική των logistics έχει τρεις στόχους:

1. Τη μείωση του κόστους
2. Τη μείωση του απασχολούμενου κεφαλαίου και
3. Τη βελτίωση της εξυπηρέτησης

Η μείωση του κόστους είναι στρατηγική που στοχεύει στην ελαχιστοποίηση των μεταβλητών δαπανών που σχετίζονται με τη μεταφορά και την αποθήκευση, πράγμα όπου συνήθως επιτυγχάνεται με την επιλογή εναλλακτικών τρόπων μεταφοράς και αποθήκευσης. Στην περίπτωση αυτή πρωταρχικός στόχος είναι η μεγιστοποίηση του κέρδους διατηρώντας το επίπεδο εξυπηρέτησης σε σταθερά επίπεδα.

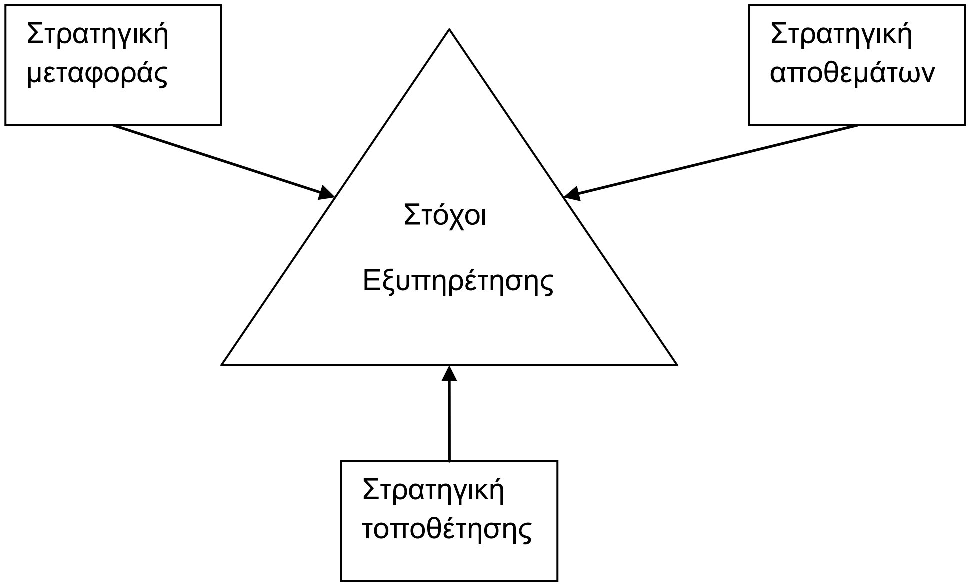
Η μείωση, τώρα, του απασχολούμενου κεφαλαίου είναι στρατηγική που στοχεύει στην ελαχιστοποίηση του επιπέδου του επενδυμένου κεφαλαίου στο σύστημα των logistics με κίνητρο την μεγιστοποίηση της απόδοσης, πράγμα το οποίο με διάφορους τρόπους όπως η κατευθείαν αποστολή αποφεύγοντας την αποθήκευση, προτιμώντας τη μη ιδιοκτησία των αποθηκών ή ακόμη χρησιμοποιώντας τρίτους ως προμηθευτές των υπηρεσιών των logistics. Στην περίπτωση αυτή είναι δυνατόν να αυξηθούν τα μεταβλητά κόστη σε σχέση με στρατηγικές υψηλότερων επιπέδων επένδυσης.

Η βελτίωση της εξυπηρέτησης είναι στρατηγικές που αναγνωρίζουν ότι τα οφέλη εξαρτώνται από το επίπεδο της παρεχόμενης εξυπηρέτησης των logistics. Είναι δυνατό παρά την αλματώδη αύξηση στα κόστη που επιφέρουν τα αυξημένα επίπεδα εξυπηρέτησης να προκύψουν κέρδη από τις πωλήσεις τα οποία να ξεπεράσουν την αύξηση του συνολικού κόστους. Για να είναι επιτυχημένη μια τέτοια στρατηγική εξυπηρέτησης θα πρέπει να αναπτύσσεται σε αντίθεση με την παρεχόμενη εξυπηρέτηση από τους ανταγωνιστές.

Ο σχεδιασμός των logistics χειρίζεται τέσσερις μεγάλες κατηγορίες προβλημάτων οι οποίες είναι οι εξής:

1. Τα επίπεδα εξυπηρέτησης πελατών
2. Την χωροθέτηση των εγκαταστάσεων
3. Τις αποφάσεις για την αποθεματοποίηση
4. Τις αποφάσεις για τη μεταφορά των προϊόντων

Οι τέσσερις αυτές κατηγορίες συνδέονται μεταξύ τους με τον τρόπο που φαίνεται στο πιο κάτω σχήμα:



Σχήμα 1.2 Το "Τρίγωνο" στη λήψη αποφάσεων των logistics. [5]

Κάθε μια από τις παραπάνω κατηγορίες έχει σημαντική επίδραση στο σχεδιασμό του συστήματος όπως εξηγείται στη συνέχεια.

Οι στόχοι εξυπηρέτησης πελατών καθορίζουν τα παρεχόμενα επίπεδα εξυπηρέτησης τα οποία επιδρούν περισσότερο από οποιονδήποτε άλλο παράγοντα στο σχεδιασμό του συστήματος, καθότι υψηλά επίπεδα εξυπηρέτησης επιτρέπουν συγκεντρωμένα αποθέματα σε λίγες αποθήκες και τη χρήση λιγότερο δαπανηρών μέσων μεταφοράς, ενώ τα χαμηλά επίπεδα το αντίθετο. Παρόλα αυτά, τα ανώτατα επίπεδα εξυπηρέτησης εκτινάσσουν το κόστος των logistics στα ύψη κάνοντας έτσι το πρόβλημα του καθορισμού των επιπέδων εξυπηρέτησης πολύ δύσκολο.

Η στρατηγική τοποθέτησης των εγκαταστάσεων, δηλαδή η γεωγραφική τοποθέτηση τους αποτελεί ένα προσχέδιο πάνω στο οποίο θα κινηθούν τα logistics, διότι το δρομολόγιο με το οποίο τα προϊόντα οδηγούνται στην αγορά καθορίζεται από τον αριθμό, τον τόπο και το μέγεθος των εγκαταστάσεων και την ζήτηση της αγοράς. Στο πρόβλημα αυτό πρέπει να εξεταστούν οι κινήσεις κάθε προϊόντος και το σχετικό κόστος από την πηγή ή τα ενδιάμεσα σημεία αποθήκευσης στην περιοχή κατανάλωσης, η επιλογή των οποίων επιδρά στο συνολικό κόστος διανομής. Η όλη ουσία του προβλήματος έγκειται στην εύρεση των αναθέσεων εγκατάστασης ελαχίστου ή μεγίστου κόστους.

Οι αποφάσεις αποθεμάτων αναφέρονται στον τρόπο με τον οποίο διαχειρίζονται τα αποθέματα. Υπάρχουν πολλές στρατηγικές όπως η προώθηση αποθεμάτων σε σημεία αποθήκευσης ή η συμπλήρωση των σημείων αυτών με ανατροφοδοτήσεις που ακολουθούν συγκεκριμένους κανόνες ή η τοποθέτηση αποθεμάτων σε τοπικές ή περιφερειακές αποθήκες με βάση συγκεκριμένα προϊόντα ή τέλος με διάφορες μεθόδους συνεχούς ελέγχου των αποθεμάτων. Η συγκεκριμένη πολιτική που ακολουθεί η εταιρία επηρεάζει τις αποφάσεις τοποθέτησης εγκαταστάσεων και γι' αυτό θα πρέπει να υπολογίζονται στην στρατηγική των logistics.

Οι στρατηγικές μεταφοράς εμπεριέχουν επιλογές μέσων, μεγέθη φορτίων και δρομολόγησης και προγραμματισμού και είναι αποφάσεις που επηρεάζουν την εγγύτητα των αποθηκών στους πελάτες επηρεάζοντας με αυτό τον τρόπο την τοποθεσία τους. Ακόμη και τα επίπεδα των αποθεμάτων μέσω των μεγεθών των φορτίων επηρεάζουν τις αποφάσεις μεταφοράς.

Συνοψίζοντας, τα παραπάνω προβλήματα όπως κατανέμονται στις τέσσερις μεγάλες κατηγορίες αποτελούν σημαντικές περιοχές σχεδιασμού λόγω της επίδρασης που έχουν στην κερδοφορία, στη ροή κεφαλαίων και στην απόδοση των επενδύσεων της εταιρίας. Πρέπει να τονιστεί ότι κάθε περιοχή απόφασης δεν πρέπει να μελετάται ξεχωριστά αλλά σε σχέση με τις άλλες για να λαμβάνεται υπόψη το αποτέλεσμα της εξισορρόπησης τους.

1.2.2 Χρησιμότητα των logistics στις επιχειρήσεις

Το τμήμα των logistics φαίνεται να παίζει σημαντικό ρόλο στην οργάνωση των επιχειρήσεων στις μέρες μας. Είναι βέβαια αδιανόητη η ύπαρξη μιας επιχείρησης χωρίς να διαθέτει, έστω και με στοιχειώδη οργάνωση, κάποιο τμήμα logistics. Όλες οι εταιρίες και οι οργανισμοί διαθέτουν άλλωστε αποθήκες όπου κρατούν τα προϊόντα τους και έναν αριθμό αποθεμάτων.

Ο στόχος για τις σύγχρονες επιχειρήσεις φαίνεται να είναι εκτός βέβαια από το να είναι βιώσιμες και να αυξάνουν την κερδοφορία τους, να καταφέρουν να ενδυναμώσουν τους τομείς των logistics και των μεταφορών, οι οποίοι φαίνεται να έχουν μεγάλη δυναμική ανάπτυξης, ιδιαίτερα στη χώρα μας τα τελευταία χρόνια. Δυστυχώς πλήττονται άμεσα από την απουσία κατάλληλα εκπαιδευμένου προσωπικού αλλά και τεχνογνωσίας. Άμεσο αποτέλεσμα είναι η υιοθέτηση από κάποιους πιο τολμηρούς επιχειρηματίες κάποιων νέων τεχνολογιών στο τομέα των logistics, οι οποίες αργότερα αντιγράφονται από τους ανταγωνιστές τις περισσότερες φορές με λανθασμένο τρόπο.

Σε κάθε περίπτωση όμως και σε δεδομένη χρονική στιγμή, κάποιος θα επενδύσει σε σύγχρονες τεχνολογίες στον τομέα των logistics, οι οποίες μπορεί να αφορούν στην κατασκευή νέων εξελιγμένων αποθηκευτικών κέντρων σε ότι αφορά όχι μόνο στον τρόπο αποθήκευσης αλλά και στα μέσα για τη γενικότερη οργάνωση της αποθήκης, σε συστήματα ελέγχου των μεταφορών αλλά και ηλεκτρονικά συστήματα εντοπισμού των φορτίων καθοδόν για τον προορισμό τους σε όποιο σημείο και αν βρίσκονται, αλλά και σε συστήματα σήμανσης και αυτοματοποιημένης συλλογής πληροφοριών.

1.3 Logistics και εφοδιαστική αλυσίδα

Όπως διαφαίνεται από τα παραπάνω, logistics και εφοδιαστική αλυσίδα συνδέονται άρρηκτα. Η εφοδιαστική αλυσίδα αποτελεί το βασικότερο πεδίο εφαρμογής των logistics, τα οποία αποτελούν το βασικότερο ζητούμενο για την ορθολογικοποίηση και την επιτυχία των διαδικασιών της εφοδιαστικής αλυσίδας. Τα logistics απαντούν στο πώς πρέπει να οργανωθούν οι διαδικασίες της εφοδιαστικής αλυσίδας - π.χ. πώς πρέπει να γίνεται η διακίνηση των προϊόντων, με τι συχνότητα πρέπει να εκτελούνται οι παραδόσεις, μέσω ποιου δρομολογίου κ.λπ.

Τα τελευταία χρόνια, νέες και καινοτόμες ψηφιακές τεχνολογίες έχουν εισέλθει στο χώρο των logistics και έχουν επιφέρει θεαματικές αλλαγές: εκεί που η απογραφή της αποθήκης απαιτούσε μολύβι, χαρτί και αρκετό χρόνο, τώρα πραγματοποιείται αυτόματα μέσω φορητών τερματικών (σκάνερ) και ηλεκτρονικών υπολογιστών. Η κακή οργάνωση της αποθήκης, ο ανεφοδιασμός χωρίς πρόγραμμα και σύστημα και τόσα άλλα αρνητικά, έχουν δώσει τη θέση τους σε ολοκληρωμένα συστήματα υψηλής ευφυΐας και αποτελεσματικότητας.

1.3.1 Ψηφιακές τεχνολογίες σε logistics και εφοδιαστική αλυσίδα

Οι ψηφιακές τεχνολογίες που απαντώνται συχνότερα στα logistics και την εφοδιαστική αλυσίδα είναι οι ακόλουθες:

α) Συστήματα πληροφορικής: Είναι εξειδικευμένες εφαρμογές λογισμικού, που αναλαμβάνουν να εξυπηρετήσουν το σύνολο των διαδικασιών της εφοδιαστικής αλυσίδας. Οι πιο γνωστές είναι τα συστήματα επιχειρηματικού σχεδιασμού (Enterprise Resource Planning - ERP) και τα πληροφοριακά συστήματα διαχείρισης της Εφοδιαστικής Αλυσίδας (Supply Chain Execution - SCE). Η συνηθέστερη μορφή των συστημάτων SCE είναι τα προγράμματα διαχείρισης αποθηκών (Warehouse Management System - WMS), τα οποία εν πολλοίς ταυτίζονται με τα συστήματα SCE.

β) Τεχνολογίες αναγνώρισης και κτήσης δεδομένων: Είναι εξειδικευμένες τεχνολογικές υποδομές (hardware και software), που συλλέγουν την πληροφορία τη στιγμή της δημιουργίας της -σε όλα τα στάδια της εφοδιαστικής αλυσίδας, λ.χ. μέσα στην αποθήκη- και τη μεταβιβάζουν στο εκάστοτε πρόγραμμα (λ.χ. WMS) για επεξεργασία. Τέτοιες υποδομές είναι τα φορητά τερματικά χειρός, τα τερματικά περονοφόρων οχημάτων, οι τεχνολογικές λύσεις Αυτόματης Αναγνώρισης και Κτήσης Δεδομένων (Automatic Identification and Data Capture - AIDC), στις οποίες ανήκουν ο γραμμωτός κώδικας (barcode), οι "έξυπνες" κάρτες, τα συστήματα αναγνώρισης χαρακτήρων και οι εφαρμογές ασύρματης αναγνώρισης, ευρύτερα γνωστές με το ακρωνύμιο RFID (Radio Frequency IDentification).

γ) Συστήματα Τηλεματικής: Είναι τεχνολογίες που χρησιμοποιούνται στις μεταφορές και αποτελούνται από πολλά μέρη hardware (πομποδέκτες, κεραίες, μικροϋπολογιστές, τηλεπικοινωνιακά δίκτυα, δορυφόροι) και software (συστήματα GIS, πρωτόκολλα επικοινωνίας), με βασική λειτουργία την καταγραφή της γεωγραφικής θέσης του οχήματος σε πραγματικό χρόνο και την απεικόνισή της σε ηλεκτρονικό υπολογιστή. Χάρη σ' αυτά, ο επιχειρηματίας μπορεί π.χ. να βλέπει ανά πάσα στιγμή πού βρίσκονται τα οχήματα και τα εμπορεύματά του, ενώ οι δυνατότητες σύνδεσης και αξιοποίησης των τεχνολογιών της πρώτης και της δεύτερης κατηγορίας είναι απεριόριστες.

δ) Υποδομές δικτύων: Ο λόγος για τα ενσύρματα και τα ασύρματα τοπικά δίκτυα, που συνήθως βρίσκονται σε μια αποθήκη εξυπηρετώντας τη μετάδοση των δεδομένων από τις διάφορες φορητές συσκευές, τους υπολογιστές κ.λπ. Τα δίκτυα αυτά αποτελούνται από υπολογιστές, καλωδίωση ή ασύρματα σημεία πρόσβασης (access points).

1.3.2 Οφέλη από logistics και εφοδιαστική αλυσίδα

Τα οφέλη που απορρέουν από την ενσωμάτωση νέων τεχνολογιών στα logistics και την εφοδιαστική αλυσίδα είναι σε γενικές γραμμές τα ακόλουθα:

α) Καλύτερη εκμετάλλευση των υλικών (άψυχων) πόρων της επιχείρησης.

Υλικοί πόροι θεωρούνται τα οχήματα, οι αποθηκευτικοί χώροι, ο εξοπλισμός κ.λπ. Για παράδειγμα, η χρήση ενός συστήματος τηλεματικής στα οχήματα της επιχείρησης (διαχείριση στόλου, fleet management) έχει ως αποτέλεσμα πιο οργανωμένες κινήσεις και λιγότερα δρομολόγια. Σχετικά με τον αποθηκευτικό χώρο, ένα σύστημα WMS και η εγκατάσταση κάποιου ασύρματου τοπικού δικτύου έχουν ως αποτέλεσμα αφενός την αξιοποίηση κάθε σπιθαμής της αποθήκης, αφετέρου τη γρηγορότερη επιτέλεση των διαδικασιών μέσα σ' αυτήν. Λόγου χάρη, το σκάνερ διαβάζει το γραμμωτό κώδικα μιας κούτας και στέλνει αυτόματα την πληροφορία (τι περιέχει η κούτα) σε κεντρικό υπολογιστή εφοδιασμένο με σύστημα WMS, μέσω του ασύρματου τοπικού δικτύου.

β) Καλύτερη αξιοποίηση των έμψυχων πόρων της επιχείρησης.

Εδώ εντάσσονται όχι μόνο οι εργαζόμενοι αλλά και οι πελάτες, οι προμηθευτές κ.λπ. Για παράδειγμα, ένα σύστημα ERP ή WMS ενημερώνει σχετικά με το ποιοι είναι οι επικερδείς πελάτες, εξασφαλίζει πολύτιμες εργατοώρες για το προσωπικό και συντελεί στην καλύτερη οργάνωση των εισερχόμενων ροών από τους προμηθευτές. Ο ενδιαφερόμενος δεν χρειάζεται πλέον να ασχολείται με τον έλεγχο του στοκ, αφού αυτό το έχει αναλάβει το ίδιο το σύστημα, η απογραφή αποθήκης γίνεται με το πάτημα ενός κουμπιού, ενώ το ίδιο απαιτείται για να μάθουμε τα έσοδα, τα έξοδα και τα κέρδη για μία ημέρα ή ένα μήνα. Συγχρόνως, γνωρίζει ποια είδη διακινούνται περισσότερο και αναλόγως διαμορφώνει τις παραγγελίες του.

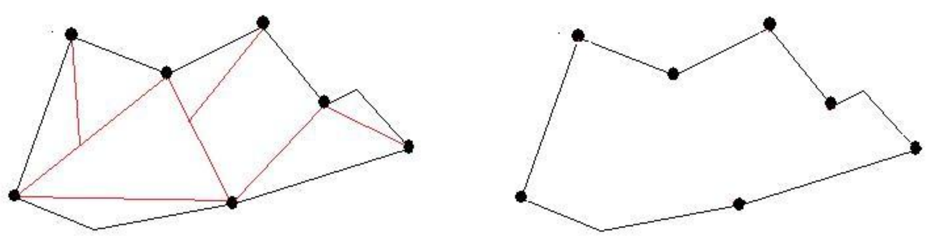
Στο επιχειρηματικό περιβάλλον, όπως αυτό διαμορφώνεται σήμερα, ζητούμενο της διοίκησης δεν είναι ο λεπτομερής έλεγχος κάθε τμήματος της επιχείρησης -καθώς κάτι τέτοιο απαιτεί πολλή ενέργεια και χρόνο- αλλά η αυτοματοποίηση διαδικασιών με τη χρήση τεχνολογικών εργαλείων. Οι τεχνολογίες που εξετάσαμε παρέχουν πλήθος πληροφοριών στη διοίκηση και τα στελέχη της επιχείρησης, ώστε να είναι δυνατή η λήψη ορθών και άμεσων αποφάσεων, καθώς και η χάραξη μακρόπνοης αναπτυξιακής στρατηγικής.

Κεφάλαιο 2:

Διατύπωση του προβλήματος

2.1 Πρόβλημα πλανόδιου πωλητή (traveling salesman problem (TSP))

Το πρόβλημα του πλανόδιου πωλητή αφορά την εύρεση της συντομότερης διαδρομής **< route >** (σε χρόνο εκτέλεσης της κεντρικής μονάδας επεξεργασίας **< CPU_time >** και της ευκλείδιας απόστασης του συνολικού μονοπατιού **< rc - route cost >**) για ένα όχημα (ή πωλητή) με αφετηρία κάποιο σημείο, π.χ. ένα κέντρο διανομής , και επιστροφή στο ίδιο σημείο αφού επισκεφθεί έναν σταθερό αριθμό πελατών ακριβώς μια φορά τον καθένα. Η αντίληψη που έχουμε για αυτή τη διαδρομή είναι αυτή του παρακάτω σχήματος , ένας κύκλος δηλαδή που διέρχεται από τους κόμβους που αντιστοιχούν στο σημείο αφετηρίας και τους πελάτες ακριβώς μια φορά.



(α) (β)

Σχήμα 2.1: Η πραγματική εικόνα μπορεί να διαφέρει από την αφαιρετική εικόνα ενός γραφήματος : οι χρωματιστές γραμμές στο (α) αντιστοιχούν στο πραγματικό οδικό δίκτυο. Η κυκλική διαδρομή (β) θα πρέπει να ερμηνευθεί σωστά στο πραγματικό οδικό δίκτυο (α).

Για τη διαμόρφωση ενός μαθηματικού προτύπου υποθέτουμε συνήθως ότι οι κόμβοι ανήκουν σ' ένα μη-διατεταγμένο γράφημα που είναι πλήρες. Ας είναι $i = 2,\dots,n$ οι κόμβοι των πελατών και $i = 1$ ο κόμβος αφετηρίας. Από την υπόθεση, κάθε μη-διατεταγμένο ζεύγος { i , j }με $i \neq j$, i = 1,... ,n, $j = 1,\dots,n$, αντιστοιχεί σ' ένα σύνδεσμο ή ακμή του γραφήματος. Σε κάθε τέτοιο σύνδεσμο αντιστοιχίζουμε ένα σταθμό Cij που είναι ίσο με το κόστος (όπως και αν νοείται αυτό) της διαδρομής του οχήματος από το i στο j ή αντίστροφα. Επειδή ένας τέτοιος σύνδεσμος δεν αντιστοιχεί πάντοτε σε κάποιο φυσικό τμήμα δρόμου (βλέπε Σχήμα 2.1) θα υποθέσουμε ότι τα σταθμά έχουν υπολογισθεί έτσι ώστε ν' αντιστοιχούν στην διαδρομή ελαχίστου κόστους μεταξύ των δύο κόμβων, οπότε ικανοποιούν τις δύο συνθήκες :

1.συμμετρία : $C_{ij} = C_{ji}$, $i \neq j$, $i = 1,....,n \quad j = 1,...,n$

2.τριγωνική ανισότητα : $C_{ij} \leq C_{ik} + C_{kj}$, $i \neq j \neq k$, $i = 1,....,n$, $k = 1,...,n$

οι οποίες υπό αυτές τις συνθήκες είναι φυσιολογικές, καθώς η μεν πρώτη απεξαρτεί το κόστος του απευθείας ταξιδιού μεταξύ i και j από την κατεύθυνση, ενώ η δεύτερη διατυπώνει ότι ο συντομότερος δρόμος μεταξύ δύο σημείων είναι απευθείας.

Αν ορίσουμε τις δυαδικές μεταβλητές

$$x_{ij} \begin{cases} 1, & \text{εάν το όχημα κάνει χρήση του συνδέσμου \{I , j\}} \\ 0, & \text{αλλιώς, } \forall \text{ i, } \forall \text{ j, i} \neq \text{ j} \end{cases}$$

τότε το πρότυπο διαμορφώνεται ως εξής:

$$\min \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij} x_{ij} \qquad (2.1)$$

$$\text{υπό} \quad \sum_{j=1}^{n} x_{ij} = 2 \text{, i=1,...,n} \qquad (2.2)$$

$$\sum_{i \in C}\sum_{i \in \bar{C}} x_{ij} \geq 1, \quad \forall C \subset \{1,....,n\}, C \neq \varnothing \qquad (2.3)$$

$$x_{ij} \in \{0,1\}, \forall i, \forall j, i \neq j \qquad (2.4)$$

Οι περιορισμοί (2.2) επιβάλλουν στην λύση να έχει δύο συνδέσμους σε κάθε κόμβο, έτσι ώστε το όχημα να εισέλθει κατά μήκος του ενός και να εξέλθει κατά μήκος του άλλου (Σχήμα 2.1[β]), ενώ οι περιορισμοί (2.3) αποβλέπουν στην εξάλειψη κυκλικών διαδρομών (υποκύκλων) που δεν διέρχονται απ' όλους τους κόμβους απαιτώντας από κάθε εν δυνάμει υπόκυκλο, που αντιπροσωπεύεται από ένα κατάλληλο μη- κενό υποσύνολο C των κόμβων, να διαθέτει στην λύση τουλάχιστον ένα σύνδεσμο που οδηγεί στο συμπληρωματικό του υποσύνολο $\bar{C} = \{1,...,n\} \setminus C$.

Όταν η φορά του οχήματος, εάν δηλαδή κινείται από το i στο j ή από το j στο i, παίζει ρόλο, τότε το γράφημα αποτελείται από τόξα που αντιστοιχούν στα διατεταγμένα ζεύγη (i,j) και (j,i) αντίστοιχα και ως τέτοια $(i,j) \neq (j, \ i)$. Υπό αυτές τις συνθήκες η συνθήκη συμμετρίας δεν ισχύει οπότε $cij \neq cji$ τουλάχιστον για κάποια i και j. Το μαθηματικό πρότυπο σε αυτήν την περίπτωση διαμορφώνεται από το (2.1)-(2.4), αντικαθιστώντας όμως τον περιορισμό (2.2) με δύο νέους περιορισμούς:

$$\sum_{j=1}^{n} x_{ij} = 1, i = 1,...,n \qquad (2.5)$$

$$\sum_{i=1}^{n} x_{ij} = 1, j = 1,...,n \qquad (2.6)$$

Όπου ο μεν πρώτος εγγυάται ότι το όχημα εξέρχεται από τον κόμβο i χρησιμοποιώντας ένα ακριβώς τόξο προς κάποιον άλλο κόμβο j, ενώ ο δεύτερος εγγυάται ότι το όχημα εισέρχεται στον κόμβο j χρησιμοποιώντας ένα ακριβώς τόξο από κάποιον άλλο κόμβο i.

Αμφότερα τα πρότυπα έχουν ένα μειονέκτημα: ο αριθμός των περιορισμών (2.3) αυξάνει εκρηκτικά με τον αριθμό κόμβων. Υπάρχει η δυνατότητα διατύπωσης προτύπου με περιορισμένο αριθμό περιορισμών. Για τον λόγο αυτό θα θεωρήσουμε ότι στη αφετηρία παράγεται μια μονάδα κάποιου αγαθού για κάθε πελάτη και ότι τα αγαθά αυτά δεν αναμειγνύονται. Επιπρόσθετα θα δημιουργήσουμε έναν ακόμα πελάτη $n_0 = n+1$ δημιουργώντας ένα αντίγραφο της αφετηρίας (μαζί με τα τόξα που της ανήκουν). Έτσι για κάθε τόξο (i,j) ορίζουμε τις μεταβλητές y_{ij}^{k} που αναφέρει την ποσότητα (από τη μία μονάδα) που διακινείται στο τόξο με προορισμό τον πελάτη k, k=2,…, n_0. Το πρότυπο δύναται τότε να διατυπωθεί ως ακολούθως:

$$\min \quad \sum_{i=1}^{n_0} \sum_{j=1}^{n_0} c_{ij} x_{ij} \qquad (2.7)$$

$$\text{υπό} \quad \sum_{j=1}^{n_0} x_{ij} = 1, i = 1,...,n \qquad (2.8)$$

$$\sum_{i=1}^{n} x_{ij} = 1, j = 2,...,n_0 \qquad (2.9)$$

$$\sum_{j=1}^{n_0} y_{ij}^k - \sum_{j=1}^{n_0} y_{ji}^k = \begin{cases} 1, & \text{εάν } i=1 \\ -1, & \text{εάν } i=k, \forall i, \forall k \\ 0, & \text{αλλιώς} \end{cases} \qquad (2.10)$$

$$y_{ij}^k \le x_{ij}, \forall i, \forall j, i \ne j, \forall k \qquad (2.11)$$

$$y_{ij}^k \ge 0, \forall i, \forall j, i \ne j \qquad (2.12)$$

$$x_{i,j} \in \{0,1\}, \forall i, \forall j, i \ne j \qquad (2.13)$$

Η αντικειμενική συνάρτηση 2.7 και οι περιορισμοί 2.8 - 2.9 και 2.13 παραμένουν στην ουσία οι ίδιοι με πριν, ενώ οι περιορισμοί 2.3 έχουν αντικατασταθεί από τους 2.10, 2.11 και 2.12 που ο αριθμός τους $0(\eta^3)$ συγκριτικά με τον αριθμό $0(2^\eta)$ των προηγουμένων προτύπων. Φυσικά το επιτύχαμε αυτό εισάγοντας $O(\eta^3)$ νέες μεταβλητές.

Οι περιορισμοί 2.10 είναι οι γνωστές εξισώσεις ισορροπίας της ροής και εγγυώνται ότι η μία μονάδα από το σημείο αφετηρίας με προορισμό τον πελάτη *k* αφικνείται στον προορισμό της χωρίς ν' απορροφηθεί ενδιάμεσους κόμβους ί. Οι περιορισμοί αυτοί διατυπώνονται για κάθε κόμβο i και κάθε πελάτη *k* στην μορφή *εκροή από τον κόμβο ί — εισροή στον κόμβο ί.* Τέλος οι περιορισμοί 2.11 εγγυώνται ότι το αγαθό με προορισμό τον πελάτη *k* κινείται μόνον σε τόξα για τα οποία Xij = 1, τόξα με άλλα λόγια που έχουν επιλεγεί για την βέλτιστη λύση.

Στο πρότυπο αυτό κάθε εφικτή λύση έχει αναγκαστικά, όπως το επιβάλλουν οι περιορισμοί 2.8 και 2.9, ένα τόξο εισόδου και ένα τόξο εξόδου για κάθε έναν από τους κόμβους με εξαίρεση τον κόμβο αφετηρίας 1 όπου ζητούμε μόνον ένα τόξο εξόδου και το αντίγραφο του n_0 όπου απαιτούμε μόνον ένα τόξο εισόδου. Οι περιορισμοί ροών απαιτούν την επιλογή αυτών των τόξων με τέτοιο τρόπο ώστε να υπάρχει δυνατότητα για κάθε αγαθό να αφιχθεί στον προορισμό του και άρα πρέπει η εφικτή λύση να προδιαγράφει μία πλήρη διαδρομή από την αφετηρία σε κάθε πελάτη συμπεριλαμβανομένου και του n_0. Για την ικανοποίηση αυτής της απαίτησης αρκούν n_0 - 1 τόξα για την δημιουργία μιας διαδρομής που από την αφετηρία εξυπηρετεί όλους τους n_0 - 1 πελάτες. Η ελαχιστοποίηση της αντικειμενικής συνάρτησης, και εφόσον όλα τα Cij είναι θετικά, δεν πρόκειται να επιτρέψει την χρήση περισσοτέρων τόξων. Άρα η βέλτιστη λύση θα αποτελείται από n τόξα τα οποία, εάν συμπτύξουμε την αφετηρία και το αντίγραφό της, θα μας δώσουν τον ζητούμενο κύκλο του πλανόδιου πωλητή.

Κεφάλαιο 3:

Περιγραφή αλγορίθμων επίλυσης

3.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα παρουσιάσουμε διάφορες ευρετικές και μεθευρετικές μεθόδους βάση των οποίων είναι δυνατή η επίλυση όλων των προβλημάτων TSP που έχουμε αναφέρει στο κεφάλαιο 2, και όχι μόνο. Οι παρακάτω μέθοδοι έχουν προταθεί από διάφορους ερευνητές και μερικές από αυτές αναπτύχθηκαν τα τελευταία χρόνια βάση προηγούμενων αποτελώντας βελτίωσή τους.

3.2 Ευρετικές Μέθοδοι

Η επίλυση ενός προβλήματος συνδυαστικής βελτιστοποίησης γίνεται ολοένα και δυσκολότερη όσο αυξάνει το μέγεθος του προβλήματος και πολλές φορές το να προσπαθούμε να βρούμε την ολικά βέλτιστη λύση σε λογικό χρόνο είναι πρακτικά αδύνατο. Για να επιλύσουμε προβλήματα αυτής της μορφής συχνά καταφεύγουμε σε διαφορετικές τεχνικές που μας οδηγούν σε μία σχεδόν βέλτιστη, αλλά ικανοποιητική λύση. Μια λύση ενός ευρετικού αλγόριθμου γίνεται αποδεκτή αν ικανοποιεί κάποια κριτήρια όπως η ποιότητα της λύσης, δηλαδή η απόκλιση της από τη βέλτιστη, η ευκολία απόκτησης μιας λύσης, η λογική πάνω στην οποία στηρίζονται οι κανόνες του ευρετικού αλγόριθμου που χρησιμοποιήθηκαν για να οδηγηθούμε στην λύση.

Για κάποιο πρόβλημα συνδυαστικής βελτιστοποίησης δεν υπάρχει μονάχα ένας ευρετικός αλγόριθμος που να δίνει τη βέλτιστη λύση, αλλά έχουν αναπτυχθεί αρκετοί αλγόριθμοι οι οποίοι συγκρινόμενοι μεταξύ τους μας οδηγούν ολοένα και σε καλύτερες λύσεις.

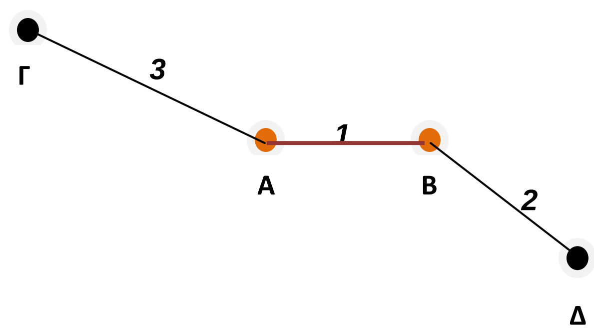
Οι ευρετικές μέθοδοι που χρησιμοποιήθηκαν είναι οι ακόλουθοι :

1. **Αλγόριθμοι απληστίας** (greedy algorithms)
2. **Αλγόριθμοι τοπικής αναζήτησης** (local search algorithms)

3.2.1 Αλγόριθμοι απληστίας [7]

Η λογική των αλγορίθμων απληστίας είναι πολύ απλή : κάποιος ξεκινάει από μια μερική μη – εφικτή λύση και σε κάθε βήμα καθορίζει μια ή περισσότερες μεταβλητές μέχρι να βρεθεί μια εφικτή λύση , όπου αυτή είναι η ζητούμενη λύση του προβλήματος . Γενικά η αρχή απληστίας είναι η αρχή ανάπτυξης μυωπικών κατασκευαστικών αλγορίθμων , αλγορίθμων δηλαδή που κατασκευάζουν μία εφικτή λύση για δεδομένο πρόβλημα σε χρόνο πολυωνυμικό ως προς το μέγεθος των δεδομένων . Στην ουσία ακολουθούν την **ιδέα του μικρού αδελφού** : Ότι φαίνεται προς στιγμή καλύτερο ως επιλογή υποθέτουμε ότι είναι και συνολικά η καλύτερη επιλογή .

3.2.1.1 Πλανόδιος πωλητής- (Standard Greedy Algorithm)



Σχήμα 3.2.1.1.1 *: Παράδειγμα εφαρμογής του αλγορίθμου απληστίας*

Παρακάτω παρατίθονται αναλυτικά τα βήματα που ακολουθεί ο αλγόριθμος :

ΒΗΜΑ 1^{ο}

Αρχικά υπολογίζεται η ευκλείδια απόσταση κάθε κόμβου από τους υπολοίπους. Με αυτόν τον τρόπο βρίσκεται και κρατείται η μικρότερη απόσταση μεταξύ δύο κόμβων , όπως για παράδειγμα στο *Σχήμα 3.2.1.1.1,* η μικρότερη απόσταση είναι το μονοπάτι **Α** ⟶ **Β** , καθώς η ευκλείδια απόσταση είναι 1 .

ΒΗΜΑ 2^{ο}

Σε αυτό το βήμα επιλέγεται η κατεύθυνση προς την οποία θα κινηθούμε έπειτα . Υπολογίζεται από το κάθε άκρο του μονοπατιού που έχουμε βρεί , τον επόμενο κόμβο που βρίσκεται στην μικρότερη ευκλείδια απόσταση. Έπειτα υπολογίζεται η συνολική ευκλείδια απόσταση του κάθε μονοπατιού και επιλέγεται η μικρότερη . Έτσι χρησιμοποιώντας την *Εικόνα 3.2.1.1.1* σαν παράδειγμα έχουμε :

1) **Α ⟶ Β ⟶ Δ : 1 + 2 = 3**
2) **Β ⟶ Α ⟶ Γ : 1 + 3 = 4**

Από τα παραπάνω αντιλαμβάνεται κανείς ότι , το μονοπάτι με την μικρότερη ευκλείδια απόσταση είναι το 1^{ο} .

ΒΗΜΑ 3^{ο}

Από τον τελευταίο κόμβο που σταματήσαμε στο Βήμα 2 , συγκεκριμένα στο παράδειγμά μας τον κόμβο **Δ** , υπολογίζεται ο επόμενος κόμβος που βρίσκεται στην μικρότερη ευκλείδια απόσταση .

<u>ΒΗΜΑ 4^o</u>

Επαναλαμβάνουμε το Βήμα 3, μέχρις ότου όλοι οι κόμβοι να βρίσκονται στο μονοπάτι.

3.2.1.2 Πλανόδιος πωλητής- (Greedy Algorithm with the Round Robin Strategy) – Πλησιέστερος Γείτονας

Σε αυτή την ευρετική διαδικασία, ο πωλητής ξεκινά από κάποια πόλη και μετά επισκέπτεται την πόλη που είναι πλησιέστερη στην αρχική. Από εκεί επισκέπτεται την πλησιέστερη πόλη που δεν έχει ακόμη επισκεφτεί, μέχρις ότου όλες οι πόλεις περιληφθούν στην διαδρομή του, και τότε επιστρέφει στην αρχική πόλη.

Παρακάτω παρατίθενται αναλυτικά τα βήματα που ακολουθεί ο αλγόριθμος :

<u>ΒΗΜΑ 1^o</u>

Στο 1^o πρώτο Βήμα επιλέγεται σαν αρχή του μονοπατιού , τυχαία , ένας κόμβος.

<u>ΒΗΜΑ 2^o</u>

Βρίσκεται ο κόμβος που είναι πλησιέστερα στον τελευταίο που προστέθηκε στην διαδρομή . Προστείθεται και αυτός ο κόμβος στο μονοπάτι.

<u>ΒΗΜΑ 3^o</u>

Επαναλαμβάνεται το Βήμα 2, μέχρις ότου όλες οι κορυφές να βρίσκονται στο μονοπάτι.

<u>ΒΗΜΑ 4^o</u>

Αποθηκεύεται αρχικά η συνολική διαδρομή . Στην συνέχεια επιλέγεται σαν αρχικός κόμβος , ένας άλλος κόμβος από αυτόν που επιλέχθηκε στο βήμα 1. Με αυτόν τον τρόπο επιτυγχάνεται το γεγονός όλοι οι κόμβοι να επιλεχθούν σαν αρχικοί . Επειτα συνεχίζουμε με το Βήμα 3 .

<u>ΒΗΜΑ 5^o</u>

Αν δεν υπάρχει άλλος κόμβος για να επιλεχθεί στην αρχική θέση , ο αλγόριθμός επιλέγει εκείνη την συνολική διαδρομή με το μικρότερη συνολική ευκλείδια απόσταση . Και έπειτα τερματίζει.

3.2.2 Αλγόριθμοι τοπικής αναζήτησης (N - opt)

Η τοπική αναζήτηση βασίζεται στην αρχαιότερη μέθοδο βελτιστοποίησης, στην μέθοδο δοκιμής και σφάλματος. Η ιδέα είναι τόσο απλή που είναι εκπληκτικό πόσο επιτυχημένη έχει αποδειχτεί η τοπική αναζήτηση στην πράξη σε ένα πολύ μεγάλο αριθμό από προβλήματα συνδυαστικής βελτιστοποίησης. Στη συνέχεια περιγράφεται ένας γενικός αλγόριθμος.

Δοθέντος ενός προβλήματος βελτιστοποίησης (F,c), όπου F είναι ένα εφικτό σύνολο και c είναι το κόστος, επιλέγουμε τη γειτονιά :

$$\boldsymbol{N}: \boldsymbol{F} \rightarrow \boldsymbol{2^F}$$

στην οποία εφαρμόζεται αναζήτηση για να βρεθεί καλύτερη λύση από την αρχική αν υπάρχει κάποιο σημείο $s \epsilon N$ που να έχει μικρότερο κόστος από το κόστος που έχουμε αυτή τη στιγμή. Για όσο χρονικό διάστημα μία καλύτερη λύση υπάρχει, αντικαθιστούμε την υπάρχουσα και συνεχίζουμε την αναζήτηση μέχρι το σημείο που θα βρούμε κάποιο τοπικό ελάχιστο και η λύση που έχουμε δεν θα βελτιώνεται παραπάνω. Στη συνέχεια περιγράφεται ένας πολύ γενικός αλγόριθμος τοπικής αναζήτησης. Στον αλγόριθμο αυτό ξεκινάμε από μια αρχική εφικτή λύση $t \epsilon F$ και χρησιμοποιούμε την υπορουτίνα *improve(i)* για να ψάξουμε για μια καλύτερη λύση στη γειτονιά της αρχικής λύσης. Όσο βρίσκεται μια βελτιωμένη λύση, την εφαρμόζουμε και επαναλαμβάνουμε την διαδικασία αναζήτησης από τη νέα λύση. Όταν φτάσουμε σε τοπικό ελάχιστο σταματάμε.

διαδικασία local search
begin

t μια αρχική λύση του προβλήματος

do while βρίσκετε μια βελτιωμένη λύση (*improve(t))*

t = improve(t)

return t
end

Ένα πολύ σημαντικό χαρακτηριστικό της τοπικής αναζήτησης είνα το γεγονός οτι μπορεί να εκτελείται από πολλά διαφορετικά αρχικά σημεία και να επιλέγεται το καλύτερο σαν βέλτιστη λύση. Σε αυτές τις περιπτώσεις μια άλλη απόφαση που πρέπει να παρθεί είναι *πόσα* θα είναι τα αρχικά σημεία που θα πρέπει να επιλέξουμε. Το επόμενο πρόβλημα που πρέπει να επιλυθεί για να είμαστε σε θέση να πούμε ότι η τοπική αναζήτηση που εφαρμόζουμε οδηγεί σε ικανοποιητικά αποτελέσματα είναι το ότι θα πρέπει να γίνει σωστή επιλογή της γειτονιάς που θα πραγματοποιηθεί η αναζήτηση.

Τέλος, το σημαντικότερο ίσως στοιχείο για την επιτυχία της διαδικασίας είναι η επιλογή της μεθόδου που θα χρησιμοποιηθεί για την αναζήτηση. Στην βιβλιογραφία του προβλήματος υπάρχουν αρκετές μέθοδοι που έχουν εφαρμοστεί για την επίλυση του προβλήματος. Στη συνέχεια, θα δούμε μια από τις σημαντικότερες εκ των οποίων είναι η μέθοδος N-opt.

Πριν παρατεθούν αναλυτικά τα βήματα εκτέλεσης , οφείλουμε αρχικά να επιλέξουμε τις παραμέτρους του αλγορίθμου , όπως υλοποιήθηκαν στα πλάισια της διπλωματικής. Οι παράμετροι που εμείς επιλέγουμε είναι οι εξής :

Παράμετροι N_opt	Εξήγηση Παραμέτρων	Εύρος τιμών
Number of items to swap	Αριθμός των ακμών που θα διαγραφούν και στην συνέχεια θα επανασυνδεθούν	2 - 4
Iterations	Αριθμός επαναλήψεων εκτέλεσης αλγορίθμου.	5 - 100.000

Πίνακας N_opt_1: *Οι παράμετροι που επιλέγονται για τον N_opt αλγόριθμο*

Στην συνέχεια αναλύεται ο αλγορίθμος N_opt , από τον οποίο μας ενδιαφέρουν μόνο οι περιπτώσεις των : 2_opt , 3_opt και 4_opt καθώς με παραπάνω διαγραφή ακμών δεν υπάρχει περαιτέρω βελτίωση [8] .

3.2.2.1 Η μέθοδος N_opt - Περιγραφή

Η μέθοδος αυτή αποτελείται γενικά από τη διαγραφή **N (2,3,4)** ακμών και την επανασύνδεση **N** μονοπατιών με διαφορετικό τρόπο για να καθορίσουμε μια καινούργια διαδρομή. Η διαδικασία της μεθόδου είναι η ακόλουθη:

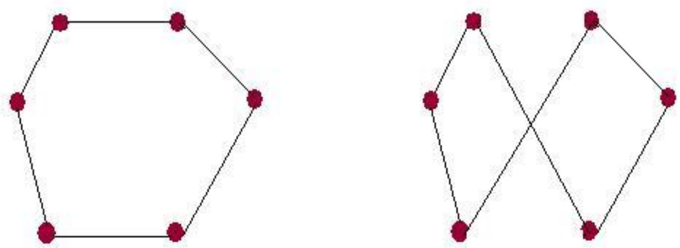
<u>ΒΗΜΑ 1ο</u>

Αν δεν μας δίνεται μία αρχική διαδρομή, τότε δημιουργούμε τυχαία μία, με βάση των αριθμό των κόμβων που ήδη γνωρίζουμε .

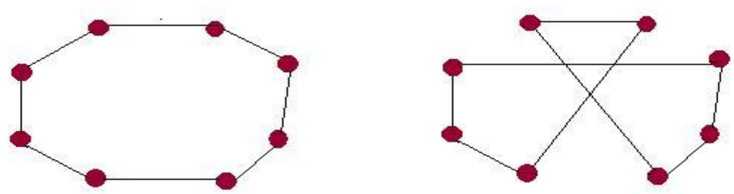
<u>ΒΗΜΑ 2ο</u>

Για μια τυχαία επιλογή μονοπατιού (στο οποίο έχει απορίψει ο αρχικός κόμβος και ο τελικός, οι οποίοι και συμπίπτουν) εκτελούμε τόσες N-opt κινήσεις , όσες μας υποδηλώνει ο αριθμός των επαναλήψεων (**iterations**) που έχει επιλεχθεί. Αν με αυτό τον τρόπο δύναται να μειωθεί το κόστος της συνολικής διαδρομής, τότε επιλέγεται η καλύτερη N-opt κίνηση και εφαρμόζονται οι αλλαγές στο μονοπάτι T.

Παρακάτω , παραθέτουμε δύο εικόνες, στις οποίες αναπαριστάται η εφαρμογής ενός 2_opt και ενός 3_opt αλγορίθμου :



***Εικόνα N_opt_1*:** Εφαρμογή του αλγορίθμου 2-opt



***Εικόνα N_opt_2*:** Εφαρμογή του αλγορίθμου 3-opt

3.3 Μεθευρετικές μέθοδοι

Τα τελευταία χρόνια οι μεθευρετικές μέθοδοι χρησιμοποιούνται ευρύτατα για την επίλυση πολύπλοκων προβλημάτων βελτιστοποίησης. Λόγω της πολυπλοκότητας που διέπει τους αλγορίθμους των μεθευρετικών μεθόδων ακόμη και σήμερα δεν έχει δοθεί επίσημα ένας ορισμός τους. Ένας πιθανός ορισμός που θα μπορούσε να δοθεί για τους μεθευρετικούς αλγορίθμους είναι ο ακόλουθος:

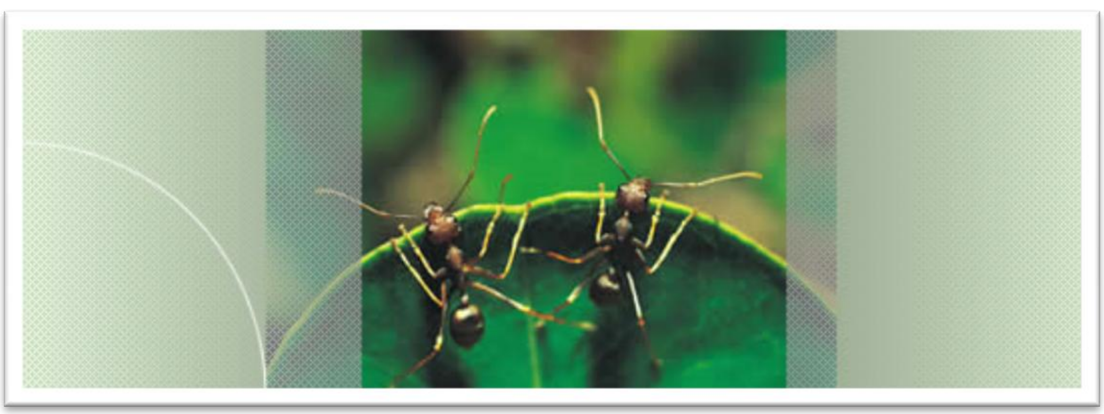
"Ένας μεθευρετικός αλγόριθμος μπορεί να περιγραφεί ως ένα γενικό πλαίσιο εργασίας αλγορίθμων (algorithmic framework) το οποίο μπορεί να εφαρμοσθεί σε διαφορετικά προβλήματα βελτιστοποίησης και με μικρές τροποποιήσεις να υιοθετηθεί για την επίλυση συγκεκριμένων προβλημάτων."

Οι μεθευρετικές μέθοδοι έχουν ως κύριο στόχο την εξονυχιστική εξερεύνηση των διαστημάτων πιθανών λύσεων. Η ποιότητα των λύσεων που επιτυγχάνονται με τις μεθόδους αυτές είναι πολύ καλύτερη από την ποιότητα που δίνουν οι κλασσικές ευρετικές μέθοδοι.

Για την επίλυση του προβλήματος του πλανόδιου πωλητή (T.S.P) χρησιμοποιήθηκαν οι ακόλουθοι μεθευρητικοί αλγόριθμοι:

1. **Αλγόριθμος Βελτιστοποίησης Αποικίας Μυρμηγκιών** (Ant Colony Optimization)
2. **Γενετικοί Αλγόριθμοι** – (Genetic Algorithms (GA))

3.3.1 Αλγόριθμος Βελτιστοποίησης Αποικίας Μυρμηγκιών (Ant Colony Optimization)



Εικόνα 3.3.1.1: *Εικόνα μυρμηγκιών σε αναζήτηση τροφής, αφήνοντας πίσω μια ποσότητα φερομόνης.*

Η Βελτιστοποίηση Αποικίας Μυρμηγκιών (Ant Colony Optimization ACO)) είναι μια σχετικά νέα στρατηγική για την επίλυση προβλημάτων συνδυαστικής βελτιστοποίησης. Η Βελτιστοποίηση Αποικίας Μυρμηγκιών είναι ένα σύστημα που μιμείται τη συμπεριφορά των πραγματικών μυρμηγκιών κατά τη διαδικασία της εύρεσης της τροφής τους.

Τα μυρμήγκια αναπτύσσουν μια τεχνική για να βρουν τη συντομότερη διαδρομή από την φωλιά τους προς την πηγή της τροφής τους και αντίθετα. Τα μυρμήγκια ξεκινούν την αναζήτηση της τροφής γύρω από την πηγή με τυχαίο τρόπο και καθώς κινούνται αφήνουν μια ποσότητα μιας ουσίας που ονομάζεται φερομόνη και με αυτό τον τρόπο μαρκάρουν το μονοπάτι που έχουν διανύσει. Η ποσότητα της φερομόνης στο κάθε μονοπάτι εξαρτάται από την απόσταση, την ποιότητα και την ποσότητα της τροφής που βρέθηκε. Το επόμενο μυρμήγκι που θα φύγει από τη φωλιά του είναι πολύ πιθανό να ακολουθήσει τη φερομόνη που θα υπάρχει σε κάποιο μονοπάτι, αφήνοντας μια ποσότητα φερομόνης στο ίδιο μονοπάτι. Καθώς η ποσότητα φερομόνης στο συγκεκριμένο μονοπάτι όλο και αυξάνεται όλο και περισσότερα μυρμήγκια ακολουθούν αυτό το μονοπάτι. Όμως καθώς η ώρα περνάει η φερομόνη, ιδιαίτερα από τα μονοπάτια που δεν πηγαίνουν πολλά μυρμήγκια, ελαττώνεται. Τελικά από όλα τα υπόλοιπα μονοπάτια η φερομόνη εξαφανίζεται και όλα τα μυρμήγκια ακολουθούν τελικά το ίδιο μονοπάτι, που είναι και η βέλτιστη ή η σχεδόν βέλτιστη λύση. Η κύρια ιδέα της βελτιστοποίησης Αποικίας Μυρμηγκιών είναι να μοντελοποιηθεί το πρόβλημα εύρεσης μονοπατιού ελαχίστου κόστους σε ένα γράφημα. Κάθε μυρμήγκι αποτελεί μια λύση για το πρόβλημα. Ο αλγόριθμος αποτελείται από ένα αριθμό από επαναλήψεις , όπου σε κάθε επανάληψη κάθε μυρμήγκι ξεκινάει να κατασκευάσει μια λύση με βάση την εμπειρία που έχει απόκτηθει από τις λύσεις που δημιούργησαν τα προηγούμενα μυρμήγκια . Στο τέλος των επαναλήψεων όλα ή σχεδόν όλα τα μυρμήγκια ακολουθούν την ίδια διαδρομή η οποία και κρατείται ως βέλτιστη λύση .

Πριν παρατεθούν αναλυτικά τα βήματα εκτέλεσης , οφείλουμε αρχικά να επιλέξουμε τις παραμέτρους του αλγορίθμου ,όπως υλοποιήθηκαν στα πλάισια της διπλωματικής. Οι παράμετροι που εμείς επιλέγουμε είναι οι εξής :

Παράμετροι ACO	Εξήγηση Παραμέτρων	Εύρος τιμών
K (ants)	Αριθμός των μυρμηγκιών	5 - 1000
Iterations	Αριθμός επαναλήψεων εκτέλεσης αλγορίθμου.	500 - 100.000
Pheromone Coefficient (a)	Συντελεστής φερομόνης ($\mathbf{\tau_{ij}}$)	1 - 10
Heuristic Function Coefficient (b)	Συντελεστής ομοιότητας ($\mathbf{n_{ij}}$)	1 - 10
Pheromone Evaporation Rate (%)	Το ποσοστό (%) της φερομόνης που εξατμίζεται σε κάθε επανάληψη του αλγορίθμου.	1 - 90 (%)
Path Rejection Threshold	Το κατώφλι με βάση το οποίο απορίπτουμε ένα μονοπάτι.	1 - 90
Local Search (2-opt) 'Optional'	Τοπική αναζήτηση καλύτερου μονοπατιού, με τον ευρετικό αλγόριθμο (2-opt) 'Επιλόγη'	0 ή 1 (ΟΧΙ) ή (ΝΑΙ)

Πίνακας ACO_1 : *Οι παράμετροι του αλγορίθμου ACO*

Στον παραπάνω πίνακα , φαίνονται το πλήθος των παραμέτρων που εμείς επιλέγουμε να χρησιμοποιήσουμε σε κάθε, νέο πρόβλημα TSP , που μας παρουσιάζεται .Όπως , παρατηρούμε , οι τιμές της κάθε παραμέτρου , έχουν ένα κάτω και ένα άνω άκρο, οι οποίες, έχουν επιλεχθεί έτσι για να καλύπτουν τα πλαίσια της διπλωματικής .

Στην συνέχεια, παρατίθενται αναλυτικά τα βήματα εκτέλεσης που ακολουθεί ο αλγόριθμος :

<u>ΒΗΜΑ 1^{o}</u>

Ορίζονται αρχικά οι απαραίτητοι πίνακες :

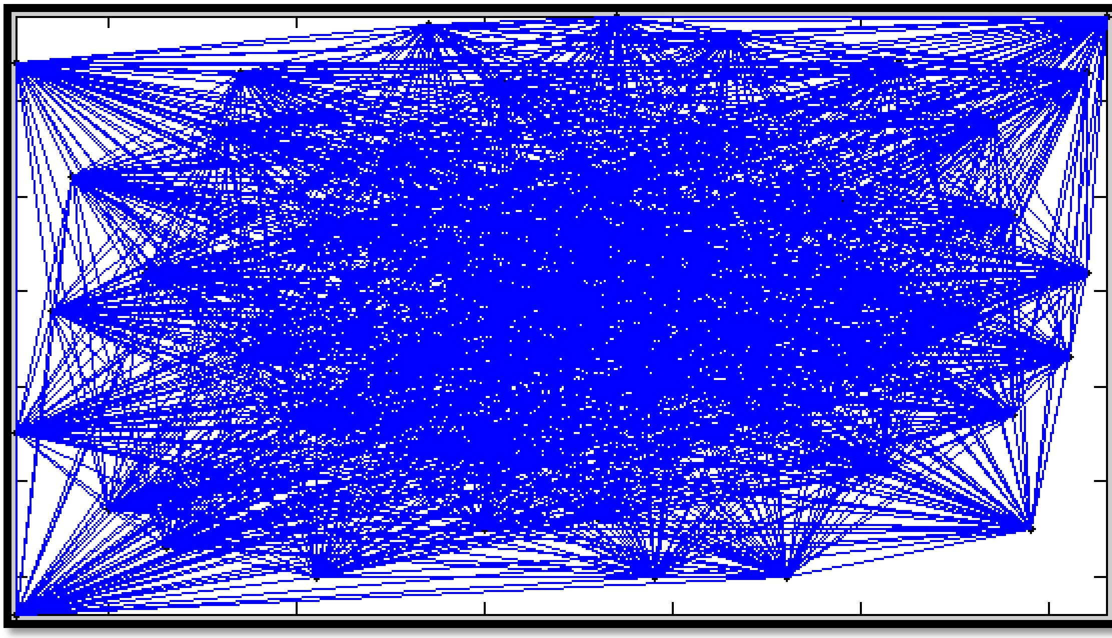
$$\boldsymbol{Sim(ij)} = \mathbf{1}/\boldsymbol{d_{ij}}$$

Όπου $\boldsymbol{Sim(ij)}$ έιναι ένας $N \times N$ πίνακας (όπου N είναι το σύνολο των κόμβων του προβλήματος **TSP**), καλείται πίνακας ομοιότητας και πρόκειται για την ευρετική συνάρτηση του προβλήματος **TSP** . Και όπου $\boldsymbol{d_{ij}}$ έιναι ένας $N \times N$ πίνακας, καλείται πίνακας απόστασης και πρόκειται για την ευκλείδια απόσταση ανάμεσα στους κόμβους **i** και **j** .

Στην συνέχεια υλοποιείται ο πίνακας $\mathbf{T_0}$, ο οποίος συμβολίζει τα αρχικά επίπεδα της φερομόνης και πιο συγκεκριμένα την επιθυμία του μυρμηγκιού να επισκεφθεί μετά την πόλη **i** την πόλη **j** .

<u>ΒΗΜΑ 2^{o}</u>

Κατασκευάζεται ο γράφος **WG** -(όπου πρόκειται για έναν $N \times N$ πίνακα)**.** Ο γράφος πάνω στον οποίο κινούνται τα μυρμήγκια είναι μη κατευθυνόμενος, διαστάσεων $N \times N$ και οι ακμές του ενώνουν κάθε σημείο της διαδρομής με όλα τα υπόλοιπα. Ένα παράδειγμα γράφου αυτής της μορφής φαίνεται στο Σχήμα *ACO_1 :*



Σχήμα ACO_1 : *Η μορφή του μη κατευθυνόμενου γράφου , πάνω στον οποίο θα κινηθούν τα μυρμήγκια πριν την εκκίνηση του αλγορίθμου.*

<u>ΒΗΜΑ 3^ο</u>

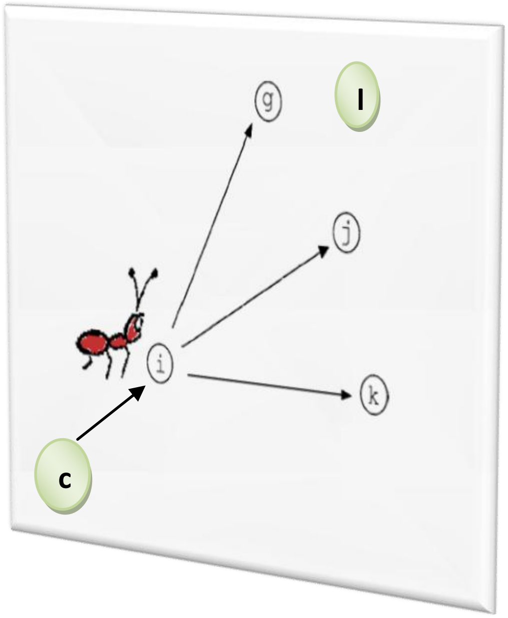
Τοποθετούνται τα μυρμήγκια σε τυχαίους κόμβους πάνω στο γράφο .

Παρατηρήσεις :

- Ο αριθμός των μυρμηγκιών που θα χρησιμοποιηθούν είναι ανάλογος με το πλήθος των σημείων που αποτελούν την διαδρομή και αντιστρόφος ανάλογος του ποσοστού εξάτμισης της φερομόνης.
- Επιπλέον ο αριθμός των μυρμηγκιών είναι ανάλογος του κατωφλιού απόριψης (**PRR**) ,καθώς όσο αυξάνεται ο αριθμός του κατωφλιού , ολοένα και περισσότερα μυρμήγκια χρειάζονται για να αφήνουν μεγαλύτερες ποσότητες φερομόνης , εξασφαλίζοντας έτσι να μην αποριφθούν ακμές που ανήκουν στην βέλτιστη διαδρομή.

<u>ΒΗΜΑ 4^ο</u>

Κινείται κάθε μυρμήγκι χωριστά έως ότου περάσει από όλους τους κόμβους του γράφου **WG** . Φυσικά δεν είναι ανάγκη να διασχύσει όλες τις δυνατές ακμές , όπως είδαμε στο *Σχήμα ACO_1* . Η επιλογή του επόμενου κόμβου είναι μια διαδικασία η οποία εξηγείται αναλυτικά κάτωθεν:



Σχήμα ACO_2 :*Επιλογή επόμενου κόμβου από το ένα μυρμήγκι.*

Αρχικά βρίσκονται όλοι οι δυνατοί κόμβοι που μπορούν να διαλεχτούν σαν επόμενοι,από το υπάρχον κόμβο.

Ένα μυρμήγκι μπορεί να κινηθεί μόνο σε επόμενους κόμβους :

Α) Τους οποίους δεν έχει επισκεφθεί ακόμα μέχρι την τρέχουσα επανάληψη.

Β) Στους οποίους έχει πρόσβαση από τον τρέχοντα κόμβο μεσω μιας ακμής.

Επομένως ο συνδυασμός των Α και Β μας δίνει τους επόμενους έγκυρους κόμβους.

Το μυρμήγκι του *Σχήματος ACO_2* βρίσκεται στον κόμβο (**i**) . Με βάση των συνδυασμό των Α και Β , οι επόμενοι έγκυροι κόμβοι είναι οι : (**g**) , (**j**) , (**k**) . Διότι, ο κόμβος (**c**) έχει αποριφθεί λόγω του ότι, είναι ο προηγούμενος κόμβος τον οποίον έχει ήδη επισκεφθει. Ενώ ο κόμβος (**l**) έχει επίσης αποριφθεί , καθώς δεν ενώνεται με τον υπάρχον κόμβο με μία ακμή.

Στη συνέχεια , για να επιλεχθεί ο ένας και μοναδικός επόμενος κόμβος από τους υπάρχοντες έγκυρους κόμβους ακολουθείται η εξής διαδικασία:

1. Επιλέγεται αρχικά, ένας τυχαίος αριθμός **(q)** μεταξύ του διαστήματος **0 και 1**.
2. Ορίζεται έπειτα ένα κατώφλι **(q_0) .**
3. Και τέλος αποφασίζουμε ως εξής :

Θεωρώντας πρώτα, ως **(i)** την υπάρχουσα θέση του μυρμηγκιού και **(j)** μία από τις επόμενες έγκυρες θέσεις , έπειτα συκρίνεται ο τυχαίος αριθμός **(q)** με το κατώφλι **(q_0)** και καταλήγουμε στα εξής :

- Αν ο τυχαίος αριθμός **(q)** είναι μικρότερος από το κατώφλι **(q_0)** τότε κερδίζει εκείνος ο συνδυασμός των κόμβων **(i)** και **(j)** ο οποίος μεγιστοποιεί το ακόλουθο γινόμενο :

$$\tau_{ij} \times n_{ij}^{b}$$

Όπου τ_{ij} είναι η ποσότητα της φερομόνης που υπάρχει ανάμεσα στον κόμβο **(i)** και **(j)**. Ενώ n_{ij}^{b} είναι ο πίνακας ομοιότητας που αναφέρουμε παραπάνω $n_{ij} = 1/_{d_{ij}}$ υψωμένος στον συντελεστή ομοιότητας **b** .

- Αν ο τυχαίος αριθμός **(q)** είναι μεγαλύτερος και ίσος από το κατώφλι **(q_0)** τότε κερδίζει εκείνος ο συνδυασμός των κόμβων **(i)** και **(j)** ο οποίος μεγιστοποιεί το ακόλουθο πηλίκο :

$$\tau_{ij}^{a} \times n_{ij}^{b} \Big/ \sum(\tau_{ij}^{a} \times n_{ij}^{b})$$

Όπου τ_{ij} είναι η ποσότητα της φερομόνης που υπάρχει ανάμεσα στον κόμβο **(i)** και **(j)** υψωμένη στον συντελεστή φερομόνης **α**. Ενώ n_{ij}^{b} είναι ο πίνακας ομοιότητας που αναφέρουμε παραπάνω $n_{ij} = 1/_{d_{ij}}$ υψωμένος στον συντελεστή ομοιότητας **b** .

Παρατηρήσεις :

- Όταν οι ευκλείδιες απόστασεις μεταξύ των κόμβων , έχουν μεγάλες διαφορές ,επιλέγουμε συντελεστή ομοιότητας **b > α** τον συντελεστή φερομόνης .Αυτό είναι λογικό , γιατί με αυτόν τον τρόπο και με βάση την παραπάνω συνάρτηση επιλογής του επόμενου κόμβου , δίνουμε μεγαλυτερη βάση στις αποστάσεις των κόμβων και επομένως προκύπτουν καλύτερα αποτελέσματα.

- Αντιθέτως , όταν οι ευκλείδιες απόστασεις μεταξύ των κόμβων , έχουν μικρές διαφορές , επιλέγουμε συντελεστή ομοιότητας **b < α** τον συντελεστή φερομόνης .Με αυτόν τον τρόπο δίνουμε βάση στα επίπεδα φερομόνης που υπάρχουν στις ακμές του γράφου μας .

- Επιλέγοντας, ωστόσο ,συντελεστή ομοιότητας **b** μεγαλύτερο από τον συντελεστή φερομόνης **α** ,η συμπεριφορά του **ACO ,** αρχίζει να μοιάζει με αυτή ενός **greedy αλγορίθμου** [4] . Αυτό το αντιλαμβανόμαστε , καθώς με την παραπάνω κίνηση , δίνουμε έμφαση αποκλειστικά και μόνο στις ευκλείδιες αποστάσεις των κόμβων, δεδομένου βέβαια ότι έχουμε επιλέξει συντελεστή ομοιότητας **b > 1.**

- Σε περίπτωση που, ο επιλεγμένος συνδιασμός παραμέτρων δεν είναι ικανός να παράγει αξιόπιστη λύση, εξατμίζεται η φερομόνη από παραπάνω ακμές από τις ελάχιστες επιθυμητές. Επομένως , όταν ο αριθμός των ακμών πέσει κάτω από (**N-1**), όπου **N** ο αριθμός των σημείων της διαδρομής, η εκτέλεση του αλγορίθμου σταματά, καθώς από αυτή την επανάληψη και έπειτα, καμία λύση δεν αντιστοιχεί σε κλειστό μονοπάτι.

ΒΗΜΑ 5ο

Στο τέλος της κίνησης των μυρμηγκιών , κάθε ένα μυρμήγκι έχει φτιάξει και ένα μονοπάτι το οποίο θεωρεί ως βέλτιστο .

ΒΗΜΑ 6ο

Στο σημείο αυτό αν θέλουμε μπορούμε να βελτιώσουμε τις υπάρχουσες λύσεις (μονοπάτια) , εκτελώντας την ευρετική μέθοδο της τοπικής αναζήτησης (**Local search**) την λειτουργία της οποίας εξηγούμε αναλυτικά στο υποκεφάλαιο 3.2.2.1 .

Παρατηρήσεις :

- Ο αριθμός των επαναλήψεων που χρησιμοποιείται για την 2_opt τοπική αναζήτηση, *στον αλγόριθμο του ACO* , είναι ανάλογος του αριθμού των σημείων (**N**) .

ΒΗΜΑ 7ο

Εξατμίζεται από όλες τις ακμές σε κάθε επανάληψη του αλγορίθμου, το ποσοστό της φερομόνης **(PER %)** που έχουμε επιλέξει .

Παρατηρήσεις :

- Αν έχουμε επιλέξει , λίγες επαναλήψεις και μεγάλο ποσοστό φερομόνης , το οποίο εξατμίζουμε , αντιλαμβανόμαστε πως τα αποτελέσματα μας δεν θα είναι αρκετά καλά . Αυτό συμβαίνει γιατί τα μυρμήγκια, μέσα σε λίγες επαναλήψεις και με την φερομόνη να εξατμίζεται σε κάθε επανάληψη πολύ γρήγορα , δεν θα πρόλάβουν να ανακαλύψουν την βέλτιστη διαδρομή.

- Γι' αυτόν ακριβώς τον λόγο επιλέγουμε πολλές επαναλήψεις και μικρό ποσοστό εξάτμισης της φερομόνης , έχοντας έτσι σαν αποτέλεσμα πιο βέλτιστες διαδρομές . Από την μία, βέβαια η **CPU** μας χρειάζεται για την εκτέλεση του αλγορίθμου , περισσότερο χρόνο , ενώ από την άλλη έχουμε μικρότερα και επομένως βέλτιστα μονοπάτια.

ΒΗΜΑ 8ο

Ταξινομούνται τα μυρμήγκια με βάση το κόστος λύσης (το μήκος του συνολικού μονοπατιού) , σε αύξουσα σειρά (από το μικρότερο στο μεγαλύτερο).

<u>ΒΗΜΑ 9ο</u>

Επιλέγεται το **25%** [4] από τα ταξινομημένα μυρμήγκια Έτσι εξασφαλίζεται ότι ο γράφος θα ενημερωθεί μόνο από τα μυρμήγκια με τις καλύτερες λύσεις. Με αυτό τον τρόπο, τα πιο «έξυπνα» μυρμήγκια θα ενημερώσουν τα υπόλοιπα ότι έχουν βρεί πιο σύντομο μονοπάτι.

Παρατηρήσεις :

- Ένας άλλος λόγος επομένως , που θα πρέπει να επιλέξουμε μεγάλη ποσότητα μυρμηγκιών (**K**) στις παραμέτρους μας , είναι ότι από τον συνολικό αριθμό των μυρμηγκιών μόνο αυτά που έχουν τις καλύτερες λύσεις θα εναποθέσουν φερομόνη στον γράφο.

<u>ΒΗΜΑ 10ο</u>

Ακολουθώντας την κάθε διαδρομή που διένυσαν το **25%** του συνόλου των μυρμηγκιών ξεχωριστά, προσθέτουμε ένα ποσό φερομόνης $\boldsymbol{\Delta^{\kappa}_{\tau_{ij}}}$ το οποίο είναι ανάλογο, ενός σταθερού παράγοντα $\boldsymbol{Q}$ και αντιστρόφως ανάλογο με το συνολικό μήκος της διαδρομής $\boldsymbol{L_{\kappa}}$ που έχει διανύσει κάθε μυρμήγκι **(κ).** Τα προηγούμενα φαίνονται και παρακάτω :

$$\boldsymbol{\Delta^{\kappa}_{\tau_{ij}} = Q \times {}^{1}/_{L_{\kappa}}}$$

Παρατηρήσεις :

- Όσο μικρότερη διαδρομή έχει διανύσει ένα μυρμήγκι , τόσο μεγαλύτερη ποσότητα φερομόνης προστίθεται στην διαδρομή . Έτσι , με αυτήν την διαδικασία στην επόμενη επανάληψη τα μυρμήγκια θα ακολουθήσουν εκείνη την διαδρομή που έχει το μεγαλύτερη ποσότητα φερομόνης και κατ' επέκταση την διαδρομή με την μικρότερη απόσταση.

<u>ΒΗΜΑ 11ο</u>

Ελέγχεται στην συνέχεια από τον συνολικό γράφο **WG** (στον οποίο όπως έχουμε ήδη προανάφερει κρατάμε τα επίπεδα της φερομόνης σε κάθε ακμή) η ποσότητα της υπάρχουσας φερομόνης. Αν είναι λιγότερη από το κατώφλι (**PRR - Path rejection Rate**) , τότε αφαιρείται από τον γράφο **WG .**

Παρατηρήσεις :

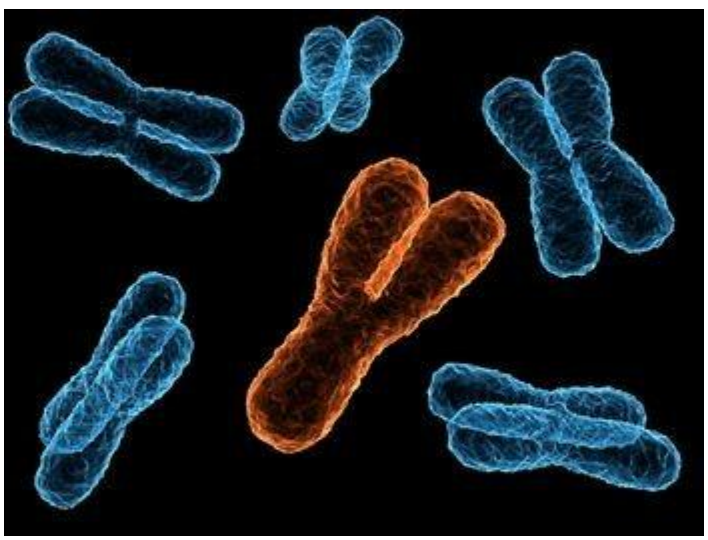
- Επιλέγουμε το κατώφλι της φερομόνης μας (**PRR**), να είναι αρκετά μικρό , εξασφαλίζοντας με αυτόν τον τρόπο ότι δεν θα απορριφθούν ακμές που ανήκουν στην βέλτιστη διαδρομή.

- Αν ο αριθμός των ακμών πέσει κάτω από (**N-1**), ο αλγόριθμος τερματίζεται.

ΒΗΜΑ 12^{o}

Τέλος , τοποθετούνται τα μυρμήγκια ένα κόμβο πρίν τον τελευταίο , ο οποίος είναι και αρχικός μας κόμβος , (για να έιναι ‘φυσικά’ η διαδρομή του πλανόδιου πωλητή μας κλείστη) και εκτελούνται τα βήματα πάλι από την αρχή .

3.3.2 Γενετικοί Αλγόριθμοι –(Genetic Algorithms (GA))



Εικόνα GA_1: *Εικόνα χρωμοσωμάτων , παραλληλίζοντάς τα με το σύνολο των δυνατών διαδρομών , με βάση το πρόβλημα του TSP.*

Η ιδέα της μίμησης της διαδικασίας εξέλιξης στην αναζήτηση για καλύτερες λύσεις οδήγησαν στην ανάπτυξη του πεδίου των Γενετικών Αλγορίθμων. Ο πρώτος γενετικός αλγόριθμος αναπτύχθηκε από τον John H. Holland την δεκαετία του 1960, για να δώσει τη δυνατότητα στους ηλεκτρονικούς υπολογιστές να παράγουν λύσεις σε δύσκολα προβλήματα αναζήτησης και συνδυαστικής βελτιστοποίησης, όπως η ελαχιστοποίηση συναρτήσεων και η μηχανική μάθηση . Οι γενετικοί αλγόριθμοι μιμούνται τη διαδικασία εξέλιξης στη φύση. Βασίζονται σε μια μίμηση της βιολογικής διαδικασίας στην οποία νέοι και καλύτεροι πληθυσμοί μεταξύ διαφορετικών ειδών αναπτύσσονται κατά τη διάρκεια της εξέλιξης. Έτσι, σε αντίθεση με τους περισσότερους ευρετικούς αλγορίθμους, οι γενετικοί αλγόριθμοι κατά τη διάρκεια της αναζήτησης μιας καλύτερης λύσης χρησιμοποιούν πληροφορίες από ένα πληθυσμό λύσεων,που ονομάζονται άτομα (individuals).

Το πρώτο πράγμα που μας ενδιαφέρει στους γενετικούς αλγορίθμους είναι η σωστή κωδικοποίηση των λύσεων. Κάθε ένα μέλος του πληθυσμού (individual) αντιπροσωπεύει μια λύση του προβλήματος βελτιστοποίησης. Τα χαρακτηριστικά των ατόμων του πληθυσμού αναπαριστούνται από το χρωμόσωμα (chromosome) ή αλλιώς γονιδίωμα (genome).Τα χαρακτηριστικά αυτά αφορούν τις μεταβλητές του προβλήματος βελτιστοποίησης. Κάθε μεταβλητή ονομάζεται γονίδιο (gene) και είναι η μικρότερη ποσότητα πληροφορία.Η ανάθεση μιας τιμής από την επιτρεπόμενη περιοχή της αντίστοιχης μεταβλητής αναφέρεται σαν αλληλομορφή (allele). Τα χαρακτηριστικά των ατόμων μπορούν να διαιρεθούν σε δύο κλάσεις πληροφοριών: στον γονότυπο (genotype)και στον φαινότυπο (phenotype).Ο γονότυπος περιγράφει την γενετική σύνθεση των ατόμων όπως κληρονομούνται από τους γονείς τους. Ο φαινότυπος δείχνει πως το άτομο μοιάζει, δηλαδή δείχνει μόνο τα κληρονομούμενα χαρακτηριστικά που βασίζονται στην συμπεριφορά των γονιών τους. Ένα πολύ σημαντικό χαρακτηριστικό στον σχεδιασμό ενός γενετικού αλγορίθμου είναι η εύρεση της κατάλληλης αναπαράστασης της υποψήφιας λύσης. Η αποτελεσματικότητα και η πολυπλοκότητα του αλγορίθμου αναζήτησης εξαρτάται από την αναπαράσταση της λύσης.

Ένας γενετικός αλγόριθμος είναι μια στοχαστική επαναληπτική διαδικασία που διατηρεί το μέγεθος του πληθυσμού σταθερό σε κάθε επανάληψη, η οποία ονομάζεται γενιά (generation). Η βασική τους λειτουργία είναι το ταίριασμα δύο λύσεων με στόχο να δημιουργήσουν μια νέα λύση. Για να δημιουργηθεί μια νέα λύση εφαρμόζεται ένας τελεστής που ονομάζεται μετάλλαξη (mutation). Ο οποίος λαμβάνει τιμές από 1 μέχρι 4. Με βάση την επιλογή της μετάλλαξης που θα κάνουμε , θα ακολουθήσουμε και διαφορετικό τρόπο για να προκύψουν οι απόγονοι (offspring). Κάθε άτομο στον πληθυσμό αντιπροσωπεύει μια υποψήφια λύση για το πρόβλημα επιλογής υποσυνόλου χαρακτηριστικών.

Πριν παρατεθούν αναλυτικά τα βήματα εκτέλεσης , οφείλουμε αρχικά να επιλέξουμε τις παραμέτρους του γενετικού αλγορίθμου , όπως υλοποιήθηκαν στα πλάισια της διπλωματικής. Οι παράμετροι που εμείς επιλέγουμε είναι οι εξής :

Παράμετροι GA	Εξήγηση Παραμέτρων	Εύρος τιμών
Chromosomes Population Size	Αριθμός των χρωμοσωμάτων που θα χρησιμοποιήσουμε	5 - 200
Number of Iterations	Αριθμός επαναλήψεων εκτέλεσης αλγορίθμου.	5 - 100.000
Batch Size	Το μέγεθος της παρτίδας των χρωμοσωμάτων , που περιέχει κάθε ομάδα. Π.χ(Αν τα χρωμοσώματα είναι 60 και η παρτίδα 6, τότε έχουμε 10 ομάδες των 6 χρωμοσωμάτων).	4 - 50
Mutation Function (Optional)	Συναρτήσεις μετάλλαξης	Επιλογή
Standard Mutation Operators (swap, shift left/right ,flip)	Μετάλλαξη με (ανταλλαγή 2 κόμβων, ολίσθηση αριστερή/δεξιά, αναστροφή ενός μέρους του χρωμοσώματος)	0
The 2-opt local search technique	Μετάλλαξη με 2_opt τοπική αναζήτηση	1
The Greedy Algorithm	Μετάλλαξη με εφαρμογή του αλγορίθμου απληστίας	2
The Greedy Algorithm with Round Robin Strategy	Μετάλλαξη με εφαρμογή του αλγορίθμου απληστίας – Πλησιέστερος Γείτονας	3

Πίνακας GA_1 : *Οι παράμετροι του αλγορίθμου του γενετικού*

Στον παραπάνω πίνακα , φαίνονται το πλήθος των παραμέτρων που εμείς επιλέγουμε να χρησιμοποιήσουμε σε κάθε, νέο πρόβλημα TSP , που μας παρουσιάζεται .Όπως , παρατηρούμε , οι τιμές της κάθε παραμέτρου , έχουν ένα κάτω και ένα άνω άκρο, οι οποίες, έχουν επιλεχθεί έτσι για να καλύπτουν τα πλαίσια της διπλωματικής .

Στην συνέχεια, παρατίθενται αναλυτικά τα βήματα εκτέλεσης που ακολουθεί ο αλγόριθμος :

<u>ΒΗΜΑ 1ο</u>

Σχεδιάζεται αρχικά ένας πίνακας , ο οποίος καλείται **Evolution (Εξέλιξη των χρωμοσωμάτων)** και στον οποίο στην συνέχεια της εκτέλεσης του αλγορίθμου, αποθηκεύεται σε κάθε επανάληψη το κόστος του κάθε χρωμοσώματος (μονοπατιού) .

<u>ΒΗΜΑ 2ο</u>

Ελέγχεται αν ο αριθμός των χρωμοσωμάτων (**Chromosomes Population Size**) , διαιρείται ακριβώς με τον αριθμό της παρτίδας (**batch size**) , που έχουμε επιλέξει . Σε περίπτωση που δεν διαιρείται ακριβώς , στρογγυλοποιείται ο αριθμός των χρωμοσωμάτων στον αμέσως επόμενο ακέραιο αριθμό με τον οποίο προκύπτει υπόλοιπο **0** .

Παρατηρήσεις :

- Αν ο πληθυσμός των χρωμοσωμάτων που έχουμε επιλέξει είναι μικρός, δεν θα είναι αντιπροσωπευτικός , όλων των δυνατών λύσεων του προβλήματος TSP . Με αυτόν τον τρόπο η αναζήτηση θα περιοριστεί αναγκαστικά πολύ και η λύση που θα βρεθεί μπορεί να μην είναι ικανοποιητική.

- Αν αντιθέτως ο αριθμός που έχουμε επιλέξει , είναι πολύ μεγάλος , καμία μέθοδος βελτιστοποίησης δεν μπορεί να ψάξει αποτελεσματικά και γρήγορα (όσον αφορά τον χρόνο εκτέλεσης της CPU) μέσα σε αυτόν .

<u>ΒΗΜΑ 3ο</u>

Στο βήμα αυτό δημιουργούνται τόσες τυχαίες διαδρομές με **(N+1)** κόμβους (δηλαδή ο αριθμός των σημείων μας σύν το αρχικό μας σημείο) , όσος είναι ο αριθμός των χρωμοσωμάτων (**Chromosomes Population Size**) που έχουμε επιλέξει . Οι διαδρομές αυτές , έπειτα μπαίνουν σε τυχαία σειρά .

<u>ΒΗΜΑ 4ο</u>

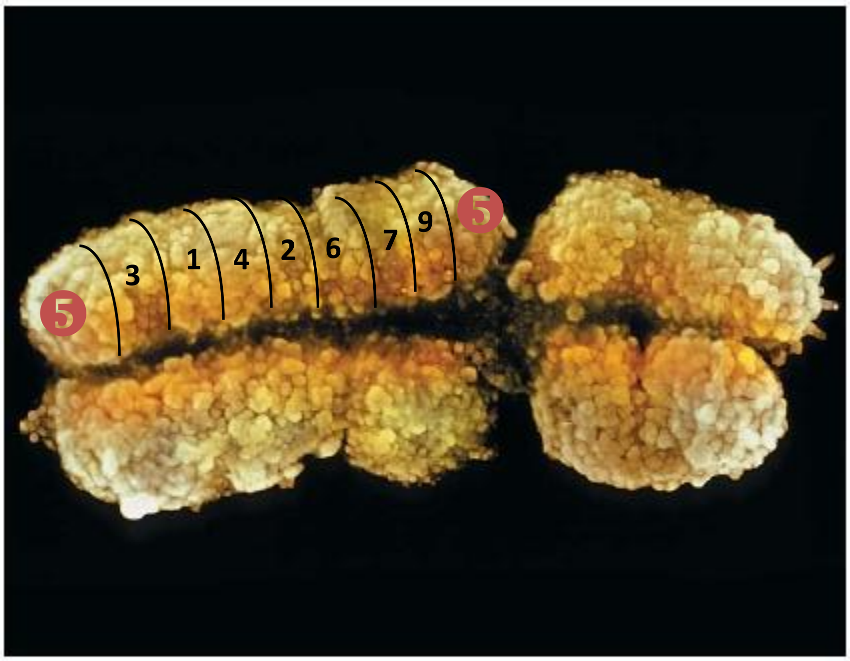
- Κατηγοριοποιούνται τα χρωμοσωμάτα σε ομάδες :

$$\boldsymbol{\pi\lambda\acute{\eta}\theta o\varsigma\ o\mu\acute{\alpha}\delta\omega\nu} = \boldsymbol{\alpha\rho\iota\theta\mu\acute{o}\varsigma\ \chi\rho\omega\mu o\sigma\omega\mu\acute{\alpha}\tau\omega\nu}\ (\boldsymbol{chromosomes\ size})\ /\ \boldsymbol{\pi\lambda\acute{\eta}\theta o\varsigma\ \pi\alpha\rho\tau\acute{\iota}\delta\alpha\varsigma}\ (\boldsymbol{batch\ size})$$

- Υπολογίζεται το συνολικό κόστος (ευκλείδεια απόσταση) όλων των χρωμοσωμάτων από κάθε ομάδα . Με αυτόν τον τρόπο χρίζεται σαν συνάρτηση καταλληλότητας (**fitness function**) η συνάρτηση κόστους .

➢ Έπειτα ταξινομούνται σε αύξουσα σειρά και επιλέγεται από κάθε ομάδα εκείνο το χρωμόσωμα με το μικρότερο κόστος (μονοπάτι), το οποίο χρίζεται αυτομάτως ως **γονέας (parent)** με βάση τον οποίο θα παραχθούν οι **απόγονοι** του (**offspring**).

<u>ΒΗΜΑ 5^{o} – Στάδιο Μετάλλαξης (Mutation stage)</u> <u>[5]</u>



Εικόνα GA_2 : Ένα χρωμόσωμα στο οποίο έχει αποτυπωθεί τυχαία μία διαδρομή.

Με βάση την συνάρτηση της μετάλλαξης που έχει επιλεχθεί έχουμε τα ακόλουθα :

1) *<u>Standard Mutation Operators (swap, shift left/right ,flip) – Μετάλλαξη με (ανταλλαγή 2 κόμβων, ολίσθηση αριστερή/δεξιά, αναστροφή ενός μέρους του χρωμοσώματος)</u>*

Οι γενετικοί τελεστές, από τους οποίους σε κάθε γενιά προκύπτουν τα χρωμοσώματα είναι οι ακόλουθοι , (όπως για παράδειγμα το χρωμόσωμα της *Εικόνας GA_2)* **:**

1. **Flip:** Επιλέγονται τυχαία πάνω στο χρωμόσωμα του γονέα , 2 σημεία (κόμβους) και στην συνέχεια με βάση ένα κέντρο, το οποίο

μπορεί να είναι , είτε ένας αριθμός (π.χ: 3 **2** 5 και γίνεται 5 **2** 3), είτε νοητό μεταξύ 2 σημείων – κόμβων (π.χ: 5 7 **|** 8 9 & γίνεται 9 8 **|** 7 5) , εκτελείται μια αναστροφή σε όλα τα σημεία που ξεκινούν από το μικρότερο σημείο (το 5 είναι το $1^{ο}$ σημείο που είναι μικρότερο από το $4^{ο}$ σημείο που είναι το 9, **1 < 4**) εκ των δύο και καταλήγουν στο μεγαλύτερο .

2. **Swap :** Επιλέγονται τυχαία πάνω στο χρωμόσωμα του γονέα , 2 σημεία (κόμβους) και στην συνέχεια με βάση ένα κέντρο, το οποίο μπορεί να είναι , είτε ένας αριθμός (π.χ: 3 **2** 5 και γίνεται 5 **2** 3) , είτε νοητό μεταξύ 2 σημείων – κόμβων (π.χ: 5 7 **|** 8 9 & γίνεται 9 7 **|** 8 5) , εκτελείται μια αναστροφή .

3. **Slide right :** Επιλέγονται τυχαία πάνω στο χρωμόσωμα του γονέα , 2 σημεία (κόμβους) και στην συνέχεια με βάση το μεγαλύτερο σημείο , όλα τα σημεία (κόμβους) , πού βρίσκονται λογικά στην αριστερή του πλευρά, τα σείρουμε στην δεξιά πλευρά. (π.χ: 6 3 2 **5** και γίνεται **5** 6 3 2) .

4. **Slide left :** Επιλέγονται τυχαία πάνω στο χρωμόσωμα του γονέα , 2 σημεία (κόμβους) και στην συνέχεια με βάση το μικρότερο σημείο , όλα τα σημεία (κόμβους) , πού βρίσκονται λογικά στην δεξιά του πλευρά, τα σείρουμε στην αριστερή πλευρά. (π.χ: **6** 3 2 5 και γίνεται 3 2 5 **6**) .

2) The 2-opt local search technique – Μετάλλαξη με 2-opt τοπική αναζήτηση [6]

Σε αυτόν τον τύπο της μετάλλαξης , χρησιμοποιείται η ευρετική μέθοδος 2_opt που έχει περιγραφθεί στην ενότητα 3.2.2.1 για τοπική αναζήτηση μιας πιο βέλτιστης λύσης .

3) The Greedy Algorithm – Μετάλλαξη με αλγόριθμο απληστίας [6]

Σε αυτόν τον τύπο της μετάλλαξης , χρησιμοποιείται ο ευρετικός αλγόριθμος απληστίας που έχει περιγραφθεί στην ενότητα 3.2.1 για αναζήτηση μιας πιο βέλτιστης λύσης .

4) The Greedy Algorithm with Round Robin Strategy – Μετάλλαξη με αλγόριθμο απληστίας με την στρατηγική του πλησιέστερου γείτονα [6]

Σε αυτόν τον τύπο της μετάλλαξης , χρησιμοποιείται ο ευρετικός αλγόριθμος απληστίας με την στρατηγική του πλησιέστερου γείτονα, που έχει περιγραφθεί στην ενότητα 3.2.1 για αναζήτηση μιας πιο βέλτιστης λύσης .

3.4 Εφαρμογή των αλγορίθμων που υλοποιήθηκαν

Το παράδειγμα που χρησιμοποιούμε σε όλους τους αλγορίθμους που υλοποιήθηκαν είναι το εξής **:**

Eil 51

Ακολουθούν τα δεδομένα του παραδείγματος :

X	**Y**
37	52
49	49
52	64
20	26
40	30
21	47
17	63
31	62
52	33
51	21
42	41
31	32
5	25
12	42
36	16
52	41
27	23
17	33
13	13
57	58
62	42
42	57
16	57
8	52
7	38
27	68
30	48
43	67
58	48
58	27
37	69
38	46
46	10
61	33
62	63
63	69
32	22
45	35
59	15
5	6
10	17
21	10
5	64
30	15
39	10
32	39
25	32
25	55
48	28
56	37
30	40

Πίνακας 3.4.1 : Παραπάνω απεικονίζονται οι συντεταγμένες των σημείων-κόμβων που θα χρησιμοποιηθούν από όλους τους αλγορίθμους.

Η εφαρμογή του **Standard Greedy Algorithm – Ά πλειστου Αλγορίθμου** φαίνεται έμπρακτα, παρακάτω:

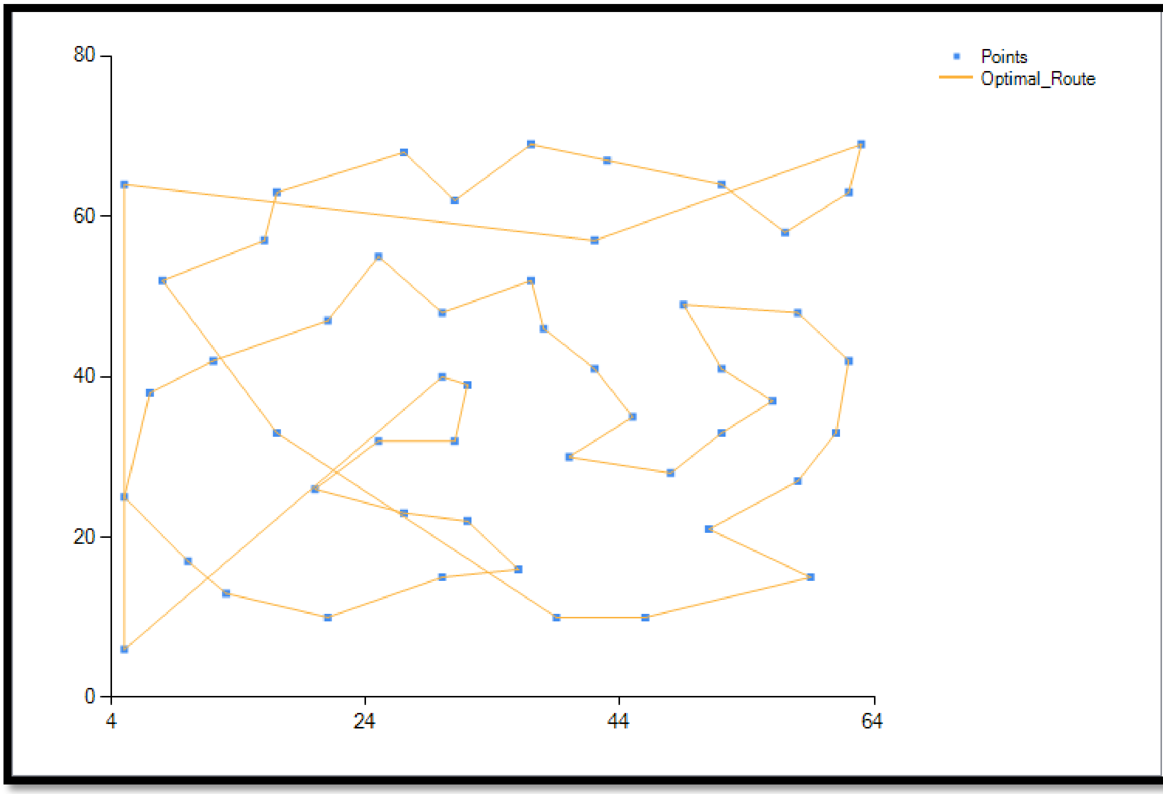
Ακολουθούν τα αποτελέσματα του παραδείγματος :

Dataset	Eil_greedy.xlsx
Solution Route - Η τελική διαδρομή :	51 46 12 47 4 17 37 15 44 42 19 41 13 25 14 6 48 27 1 32 11 38 5 49 9 50 16 2 29 21 34 30 10 39 33 45 18 24 23 7 26 8 31 28 3 20 35 36 22 43 40 51
Κόστος διαδρομής σε μονάδες ευκλέιδιας απόστασης :	523,4082
Cpu time – Χρόνος επεξεργασίας σε δευτερόλεπτα(sec) :	0,0936006

Πίνακας 3.4.2 : *Τα αποτελέσματα του παραδείγματος Eil_51 εφαρμόζοντας τον αλγόριθμο απληστίας.*

Παραπάνω απεικονίζονται , οι κόμβοι της τελικής διαδρομής που υπολογίστηκε σαν βέλτιστη από τον αλγόριθμο απληστίας , ενώ παράλληλα κρατούνται το κόστος της συνολικής διαδρομής και ο χρόνος εκτέλεσης του αλγορίθμου .

Ακολουθεί η γραφική αναπαράσταση της τελικής (βέλτιστης) διαδρομής του παραδείγματος :



Σχήμα 3.4.1 : Η τελική διαδρομή του αλγορίθμου απληστίας

Παραπάνω απεικονίζεται , ένας μη κατευθυνόμενος γράφος ,ο οποίος αποτελείται από ($\mathrm{N} - 1 = 51 - 1 = 50$) ευθύγραμμα τμήματα τα οποία ενώνουν αμφιμονοσήμαντα δύο και μοναδικούς κόμβους .Όπου Ν = 51 είναι το σύνολο των σημείων που έχει το συγκεκριμένο παράδειγμα. Επομένως , ο μη κατευθυνόμενος γράφος του σχήματος , αντιστοιχεί στην τελική διαδρομή που υπολογίστηκε από τον αλγόριθμο απληστίας .

Παρατηρήσεις :

- Ο αλγόριθμος απληστίας, συμβουλευόμενοι τον *πίνακα 3.4.2* , στο *1^ο^ βήμα* εκτέλεσης επέλεξε σαν πλησιέστερο ευθύγραμμο τμήμα , το μονοπάτι (**46 – 12**). Στο *2^ο^ βήμα* με βάση αυτό το μονοπάτι επέλεξε στην συνέχεια να κατευθυνθεί προς τον κόμβο (**47**) , δεδομένου ότι βρίσκεται πλησιέστερα στον τελευταίο κόμβο που πρόστέθηκε στην διαδρομή . Στο *3^ο^ βήμα* επιλέγεται όμοια ο αμέσως πλησιέστερος κόμβος . Στο 4^ο^ και τελευταίο *βήμα* επαναλαμβάνεται το βήμα 3 τόσες φορές όσες χρειάζεται για συμπεριληφθούν όλοι οι κόμβοι στη τελική διαδρομή . Τέλος ο τελευταίος ευρισκόμενος κόμβος ενώνεται με ένα

ευθύγραμμο τμήμα με τον αρχικό κόμβο . Πιο συγκεκριμένα, σχεδιαστικά , στο παράδειγμά μας **Eil 51** , το 3^{o} βήμα επαναλαμβάνεται :
Δεδομένου ότι έχουμε **51** κόμβους , αφαιρείται **1** κόμβος ο οποίος λαμβάνει θέση αρχικού κόμβου . Στην συνέχεια αφαιρούνται άλλοι **2** με βάση το βήμα 2 . Εν κατακλείδι τελικά μας απομένουν **48** κόμβοι .Άρα το 3^{o} βήμα θα εκτελεστεί **48** φορές , για να συνπληρωθεί η συνολική διαδρομή.

- Παρόλο , που ο χρόνος εκτέλεσης του αλγορίθμου είναι πολύ γρήγορος , το κόστος διαδρομής που υπολόγισε , απέχει πολύ από το βέλτιστο (**+ 22,86%** απόκλιση από το βέλτιστο) .Γι' αυτόν , ακριβώς τον λόγο , συνίσταται η χρήση του στην τοπική αναζήτηση μιας περισσότερο βέλτιστης λύσης , σε σχέση με αυτή που του δώθηκε σαν είσοδο .

Η εφαρμογή του **Greedy Algorithm with the Round Robin Strategy – Άπλειστου Αλγορίθμου με χρήση της στρατηγικής Round Robin** φαίνεται παρακάτω :

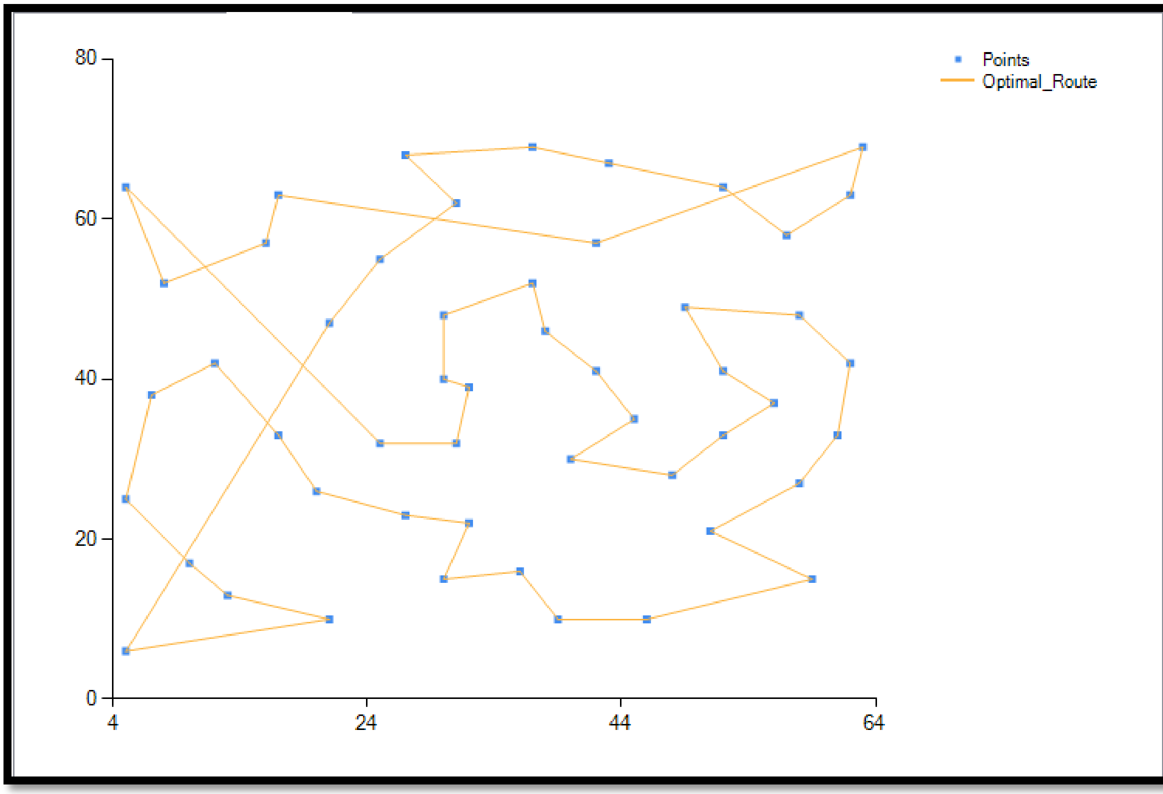
Ακολουθούν τα αποτελέσματα του παραδείγματος :

Dataset:	eil_greedy_rb_res.xlsx
Solution Route – Η τελική διαδρομή:	47 12 46 51 27 1 32 11 38 5 49 9 50 16 2 29 21 34 30 10 39 33 45 15 44 37 17 4 18 14 25 13 41 19 42 40 6 48 8 26 31 28 3 20 35 36 22 7 23 24 43 47
Route Cost - Κόστος διαδρομής σε μονάδες ευκλέιδιας απόστασης :	505,7737
Cpu time - Χρόνος επεξεργασίας σε δευτερόλεπτα(sec):	0,2340015

Πίνακας 3.4.3 : *Τα αποτελέσματα του παραδείγματος Eil_51 εφαρμόζοντας τον αλγόριθμο απληστίας με την στρατηγική του Round Robin.*

Παραπάνω απεικονίζονται , οι κόμβοι της τελικής διαδρομής που υπολογίστηκε σαν βέλτιστη από τον αλγόριθμο, ενώ παράλληλα κρατούνται το κόστος της συνολικής διαδρομής και ο χρόνος εκτέλεσης του αλγορίθμου.

Ακολουθεί η γραφική αναπαράσταση της τελικής (βέλτιστης) διαδρομής του παραδείγματος :



Σχήμα 3.4.2 : Η τελική διαδρομή του αλγορίθμου απληστίας με την στρατηγική του Round Robin.

Παραπάνω απεικονίζεται , ένας μη κατευθυνόμενος γράφος ,ο οποίος αποτελείται από ($\mathrm{N} - 1 = 51 - 1 = 50$) ευθύγραμμα τμήματα τα οποία ενώνουν αμφιμονοσήμαντα δύο και μοναδικούς κόμβους .Όπου N = 51 είναι το σύνολο των σημείων που έχει το συγκεκριμένο παράδειγμα. Επομένως ,ο μη κατευθυνόμενος γράφος του σχήματος , αντιστοιχεί στην τελική διαδρομή που υπολογίστηκε από τον συγκεκριμένο αλγόριθμο .

<u>Παρατηρήσεις :</u>

- Ο αλγόριθμος απληστίας με την στρατηγική του πλησιέστερου γείτονα, συμβουλευόμενοι τον *πίνακα 3.4.3,* μετά από (*N = 51 σημεία)* τυχαίες επιλογές , κατέληξε στην επιλογή του σαν αρχικό κόμβο , τον κόμβο (**47**). Αυτό συνέβη , επειδή , ο αλγόριθμος επιλέγοντας τυχαία σαν αρχικό κόμβο τον κόμβο (**47**) , οδηγήθηκε τελικά σε ένα συνολικό μονοπάτι το οποίο ήταν το μικρότερο σε ευκλείδια απόσταση από τα υπόλοιπα . Στην συνέχεια , χρησιμοποιώντας την στρατηγική του πλησιέστερου γείτονα , επέλεξε σαν επόμενο κόμβο τον κόμβο (**12**) , δεδομένου ότι βρίσκεται πλησιέστερα στον αρχικό κόμβο που πρόστέθηκε στην διαδρομή . Με τον ίδιο ακριβώς τρόπο, επιλέχθηκαν και οι υπόλοιποι κόμβοι της διαδρομής. Τέλος ο τελευταίος ευρισκόμενος κόμβος ενώνεται με ένα ευθύγραμμο τμήμα με τον αρχικό κόμβο . Επομένως αρχή και τέλος συμπίπτουν .

- Παρόλο , που ο χρόνος εκτέλεσης του αλγορίθμου είναι πολύ γρήγορος , το κόστος διαδρομής που υπολόγισε , απέχει πολύ από το βέλτιστο (**+ 18,72%** απόκλιση από το βέλτιστο) .Γι' αυτόν , ακριβώς τον λόγο , συνίσταται η χρήση του στην τοπική αναζήτηση μιας περισσότερο βέλτιστης λύσης , σε σχέση με αυτή που του δώθηκε σαν είσοδο .

- Ωστόσο, το κόστος του αλγόριθμου απληστίας με την στρατηγική του πλησιέστερου γείτονα , είναι μικρότερο συγκριτικά με το κόστος του αλγόριθμου απληστίας . Αυτό , βέβαια , είναι πολύ λογικό καθώς με την στρατηγική του πλησιέστερου γείτονα επιλέγονται όλοι οι κόμβοι, σαν αρχικοί (και οι **51**) . Εκείνο , τέλος στο οποίο μειονεκτεί συγκριτικά με τον απλό αλγόριθμο απληστίας , έιναι η μικρή διαφορά που έχουν στον χρόνο επεξεργασίας , αν και γενικά ο χρόνος εκτέλεσης του , είναι πολύ γρήγορος .

Η εφαρμογή του **N - opt Algorithm – N - opt Αλγορίθμου για N = 2** φαίνεται έμπρακτα, παρακάτω :

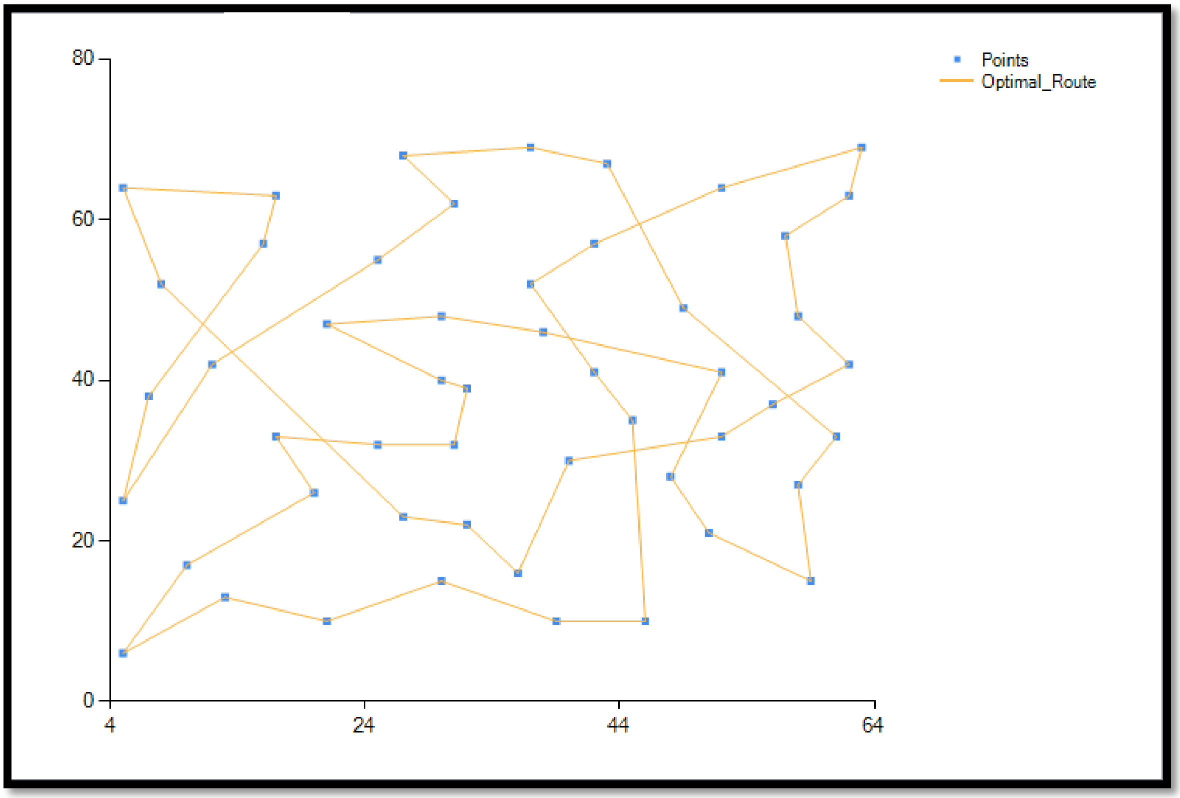
Ακολουθούν τα αποτελέσματα του παραδείγματος :

Dataset:	**eil_2_opt_better_2_res.xlsx**
Solver Used:	N-opt
Maxiter – Αριθμός επαναλήψεων	30000
route_start – Αρχικός κόμβος	1
N	2
Solution Route - Η Τελική διαδρομή:	1 22 3 36 35 20 29 21 50 9 5 15 37 17 24 43 7 23 25 13 14 48 8 26 31 28 2 34 30 39 10 49 16 32 27 6 51 46 12 47 18 4 41 40 19 42 44 45 33 38 11 1
Route Cost - Κόστος διαδρομής σε μονάδες ευκλείδιας απόστασης:	563,0058
Cpu time (sec) – Χρόνος επεξεργασίας σε δευτερόλεπτα :	61,13679

Πίνακας 3.4.4 : *Τα αποτελέσματα του παραδείγματος Eil_51 εφαρμόζοντας τον αλγόριθμο 2_opt.*

Παραπάνω απεικονίζονται , οι κόμβοι της τελικής διαδρομής που υπολογίστηκε σαν βέλτιστη από τον αλγόριθμο, ενώ παράλληλα κρατούνται : ο αριθμός των επαναλήψεων , ο αρχικός κόμβος , ο αριθμός του N=2, το κόστος της συνολικής διαδρομής και ο χρόνος εκτέλεσης του αλγορίθμου.

Ακολουθεί η γραφική αναπαράσταση της τελικής (βέλτιστης) διαδρομής του παραδείγματος :



Σχήμα 3.4.3: Η τελική διαδρομή του αλγορίθμου 2_opt.

Παραπάνω απεικονίζεται , ένας μη κατευθυνόμενος γράφος ,ο οποίος αποτελείται από ($N - 1 = 51 - 1 = 50$) ευθύγραμμα τμήματα τα οποία ενώνουν αμφιμονοσήμαντα δύο και μοναδικούς κόμβους .Όπου Ν = 51 είναι το σύνολο των σημείων που έχει το συγκεκριμένο παράδειγμα. Επομένως ,ο μη κατευθυνόμενος γράφος του σχήματος , αντιστοιχεί στην τελική διαδρομή που υπολογίστηκε από τον συγκεκριμένο αλγόριθμο .

<u>*Παρατηρήσεις :*</u>

- Ο αλγόριθμος 2_opt , συμβουλευόμενοι τον *πίνακα 3.4.4,* έχει επιλέξει σαν αρχικό κόμβο τον κόμβο (**1**) . Η τιμή αυτή έχει επιλέχθεί στον σχεδιασμό του αλγορίθμου μας [8] . Σε αντίθετη περίπτωση, που ο αλγόριθμος 2_opt , ‘συγκεκριμένα’ χρησιμοποιείται από έναν άλλο μεθευρετικό αλγόριθμο, τότε στην περίπτωση αυτή δίνεται σαν είσοδο στην μέθοδο 2_opt, μία διαδρομή στην οποία χρειάζεται να εφαρμοστεί τοπική αναζήτηση για την έυρεση μιας νέας διαδρομής με μικρότερο κόστος (ευκλείδια απόσταση) . Διατηρώντας , επομένως σαν αρχικό και τελικό κόμβο τον (**1**) , επιλέγεται στην συνέχεια τυχαία μια διαδρομή , η οποία περιλαμβάνει τους υπόλοιπους **50** κόμβους . Έπειτα , εκτελούμε **30.000 φορές** την 2_opt μέθοδο .Κατά την οποία, διαγράφουμε τόσα ευθύγραμμα τμήματα , όσα μας υποδηλώνει η παράμετρος **N (= 2)** και έπειτα τα συνδέουμε με τυχαίο και διαφορετικό τρόπο .Στο τέλος κάθε επανάληψης συγκρίνουμε το κόστος της διαδρομής που δημιουργήθηκε , με το κόστος της προηγούμενης διαδρομής . Στην περίπτωση , επομένως ,που το νέο κόστος , είναι μικρότερο από το προηγούμενο ,το αποθηκεύουμε για να χρησιμοποιηθεί σαν αντικείμενο σύγκρισης με τα επόμενα . Με τον τρόπο αυτό , στο τέλος , έχει παραμείνει αποθηκευμένο το μικρότερο κόστος , μέσα στο πλήθος των **30.000** επαναλήψεων .

- Επιλέξαμε **30.000 επανάληψεις** , ούτως ώστε να γίνουν πολλές 2_opt κινήσεις και να πλησιάσει το κόστος της συνολικής διαδρομής μας το βέλτιστο . Ωστόσο , η απόκλιση μας από το βέλτιστο είναι μεγάλη **(+32,15 %),** παρόλο που για τόσες επαναλήψεις ο αλγόριθμός 2_opt χρειάστηκε αρκετό χρόνο για καταλήξει στο τελικό αποτέλεσμα **.** Γι' αυτόν , ακριβώς τον λόγο , συνίσταται η χρήση του, από άλλους μεθευρετικούς αλγορίθμους , στην τοπική αναζήτηση μιας περισσότερο βέλτιστης λύσης , σε σχέση με αυτή που του δίνεται σαν είσοδο .Αυτό , είναι κατανοητό , καθώς με είσοδο μία διαδρομή με μικρότερο – καλύτερο , κόστος ,μετά την τοπική βελτιστοποίηση , πιθανολογικά και μόνο θα παραχθούν καλύτερα αποτελέσματα.

- Τέλος , συγκριτικά με τα αποτέλεσματα των άλλων ευρετικών αλγορίθμων , το κόστος της διαδρομής που παράχθηκε από τον 2_opt , δεν είναι μικρότερο - καλύτερο από τους αλγορίθμους απληστίας . Αυτό συμβαίνει επειδή ο αλγόριθμος 2_opt , σε κάθε επανάληψη χρησιμοποιεί ένα μόνο 'τέχνασμα' , συγκριτικά με τους άλλους . Επομένως , εκείνο που θα άλλαζε λίγο την κατάσταση , θα ήταν η χρησιμοποίηση περισσότερων επαναλήψεων , γεγονός που θα έκανε πολύ μεγάλο το χρόνο εκτέλεσης .

Η εφαρμογή του **N - opt Algorithm – N - opt Αλγορίθμου για N = 3** φαίνεται έμπρακτα, παρακάτω:

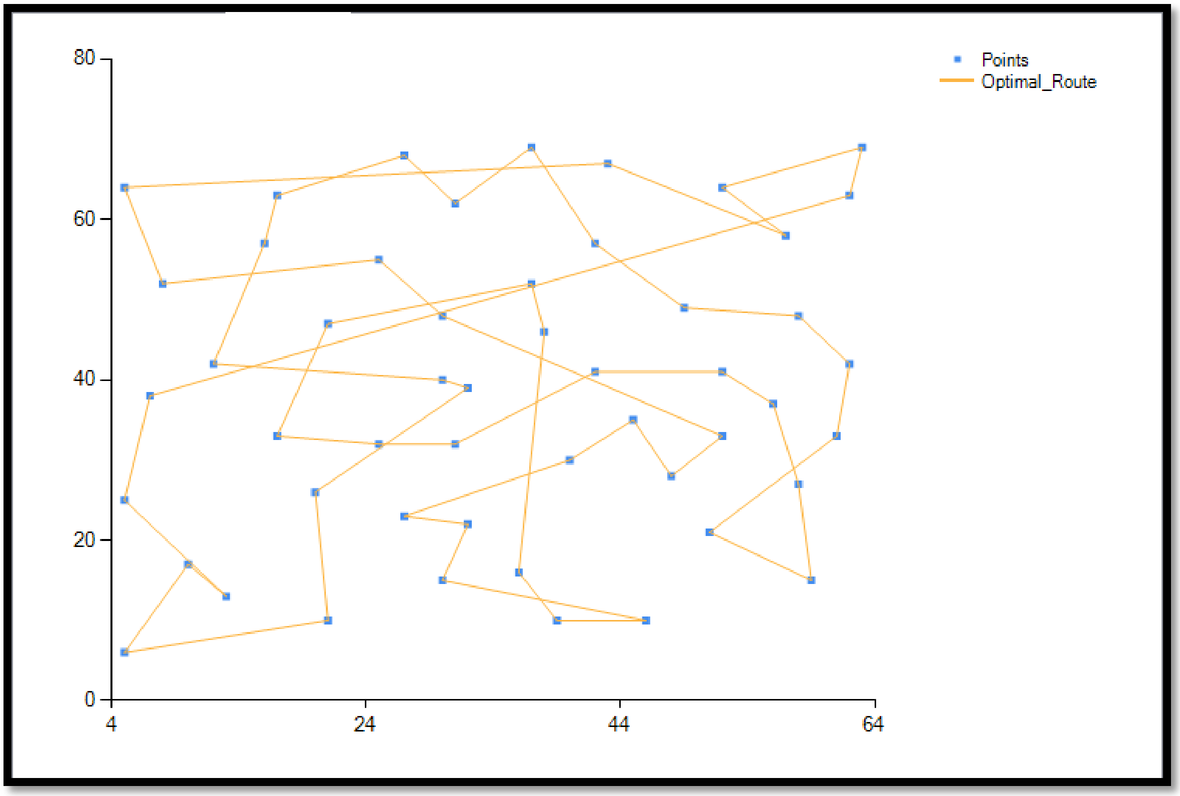
Ακολουθούν τα αποτελέσματα του παραδείγματος :

Dataset:	eil_3_opt_best.xlsx
Solver Used:	N-opt
Maxiter – Αριθμός επαναλήψεων	30000
route_start – Αρχικός κόμβος	1
N	3
Solution Route - Η Τελική διαδρομή:	1 32 15 45 33 44 37 17 5 38 49 9 27 48 24 43 28 20 3 36 35 25 13 19 41 40 42 4 46 51 14 23 7 26 8 31 22 2 29 21 34 10 39 30 50 16 11 12 47 18 6 1
Route Cost - Κόστος διαδρομής σε μονάδες ευκλείδιας απόστασης:	657,3744
Cpu time (sec) – Χρόνος επεξεργασίας σε δευτερόλεπτα :	58,62518

Πίνακας 3.4.5 : *Τα αποτελέσματα του παραδείγματος Eil_51 εφαρμόζοντας τον αλγόριθμο 3_opt.*

Παραπάνω απεικονίζονται , οι κόμβοι της τελικής διαδρομής που υπολογίστηκε σαν βέλτιστη από τον αλγόριθμο, ενώ παράλληλα κρατούνται : ο αριθμός των επαναλήψεων , ο αρχικός κόμβος , ο αριθμός του N=3, το κόστος της συνολικής διαδρομής και ο χρόνος εκτέλεσης του αλγορίθμου.

Ακολουθεί η γραφική αναπαράσταση της τελικής (βέλτιστης) διαδρομής του παραδείγματος :



Σχήμα 3.4.4: Η τελική διαδρομή του αλγορίθμου 3_opt.

Παραπάνω απεικονίζεται , ένας μη κατευθυνόμενος γράφος ,ο οποίος αποτελείται από ($N - 1 = 51 - 1 = 50$) ευθύγραμμα τμήματα τα οποία ενώνουν αμφιμονοσήμαντα δύο και μοναδικούς κόμβους .Όπου N = 51 είναι το σύνολο των σημείων που έχει το συγκεκριμένο παράδειγμα. Επομένως ,ο μη κατευθυνόμενος γράφος του σχήματος , αντιστοιχεί στην τελική διαδρομή που υπολογίστηκε από τον συγκεκριμένο αλγόριθμο .

<u>Παρατηρήσεις :</u>

- Ο αλγόριθμος 3_opt , συμβουλευόμενοι τον *πίνακα 3.4.5,* έχει επιλέξει σαν αρχικό κόμβο τον κόμβο (**1**) . Η τιμή αυτή έχει επιλέχθεί στον σχεδιασμό του αλγορίθμου μας [8] . Διατηρώντας , επομένως σαν αρχικό και τελικό κόμβο τον (**1**) , επιλέγεται στην συνέχεια τυχαία μια διαδρομή , η οποία περιλαμβάνει τους υπόλοιπους **50** κόμβους . Έπειτα , εκτελούμε **30.000 φορές** την 3_opt μέθοδο .Κατά την οποία, διαγράφουμε τόσα ευθύγραμμα τμήματα , όσα μας υποδηλώνει η παράμετρος **N (= 3)** και έπειτα τα συνδέουμε με τυχαίο και διαφορετικό τρόπο .Στο τέλος κάθε επανάληψης συγκρίνουμε το κόστος της διαδρομής που δημιουργήθηκε , με το κόστος της προηγούμενης διαδρομής . Στην περίπτωση , επομένως ,που το νέο κόστος , είναι μικρότερο από το προηγούμενο ,το αποθηκεύουμε για να χρησιμοποιηθεί σαν αντικείμενο σύγκρισης με τα επόμενα . Με τον τρόπο αυτό , στο τέλος , έχει παραμείνει αποθηκευμένο το μικρότερο κόστος , μέσα στο πλήθος των **30.000** επαναλήψεων .

❖ Επιλέξαμε **30.000 επανάληψεις** , ούτως ώστε να γίνουν πολλές 3_opt κινήσεις και να πλησιάσει το κόστος της συνολικής διαδρομής μας το βέλτιστο . Ωστόσο , η απόκλιση μας από το βέλτιστο είναι πολύ μεγάλη **(+54,31 %),** παρόλο που για τόσες επαναλήψεις ο αλγόριθμός 3_opt χρειάστηκε αρκετό χρόνο για καταλήξει στο τελικό αποτέλεσμα **.** Γι' αυτόν , ακριβώς τον λόγο , συνίσταται η χρήση του, από άλλους μεθευρετικούς αλγορίθμους , στην τοπική αναζήτηση μιας περισσότερο βέλτιστης λύσης , σε σχέση με αυτή που του δίνεται σαν είσοδο .

❖ Τέλος , συγκριτικά με τα αποτέλεσματα των άλλων ευρετικών αλγορίθμων , το κόστος της διαδρομής που παράχθηκε από τον 3_opt , δεν είναι μικρότερο - καλύτερο από τον 2_opt και τους αλγορίθμους απληστίας .

Η εφαρμογή του **N - opt Algorithm – N - opt Αλγορίθμου και για N = 4** φαίνεται έμπρακτα, παρακάτω :

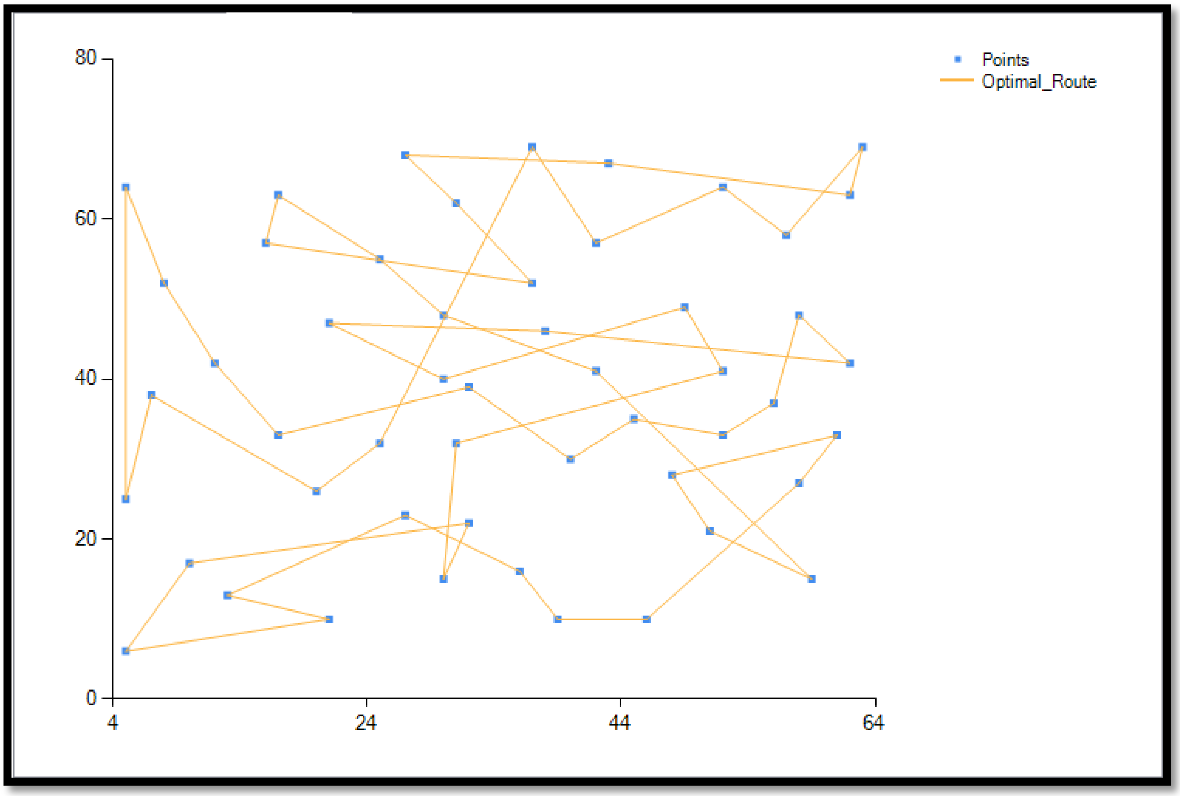
Ακολουθούν τα αποτελέσματα του παραδείγματος :

Dataset:	**eil_4_opt_best.xlsx**
Solver Used:	N-opt
Maxiter – Αριθμός επαναλήψεων	30000
route_start – Αρχικός κόμβος	1
N	4
Solution Route - Η Τελική διαδρομή:	1 8 26 28 35 36 20 3 22 31 47 4 25 13 43 24 14 18 46 5 38 9 50 29 21 32 6 51 2 16 12 44 37 41 40 42 19 17 15 45 33 30 34 49 10 39 11 27 48 7 23 1
Route Cost - Κόστος διαδρομής σε μονάδες ευκλείδιας απόστασης:	705,6201
Cpu time (sec) – Χρόνος επεξεργασίας σε δευτερόλεπτα :	60,98079

Πίνακας 3.4.6: *Τα αποτελέσματα του παραδείγματος Eil_51 εφαρμόζοντας τον αλγόριθμο 4_opt.*

Παραπάνω απεικονίζονται , οι κόμβοι της τελικής διαδρομής που υπολογίστηκε σαν βέλτιστη από τον αλγόριθμο, ενώ παράλληλα κρατούνται : ο αριθμός των επαναλήψεων , ο αρχικός κόμβος , ο αριθμός του N=4, το κόστος της συνολικής διαδρομής και ο χρόνος εκτέλεσης του αλγορίθμου.

Ακολουθεί η γραφική αναπαράσταση της τελικής (βέλτιστης) διαδρομής του παραδείγματος :



Σχήμα 3.4.5: Η τελική διαδρομή του αλγορίθμου 3_opt.

Παραπάνω απεικονίζεται , ένας μη κατευθυνόμενος γράφος ,ο οποίος αποτελείται από ($\mathrm{N} - 1 = 51 - 1 = 50$) ευθύγραμμα τμήματα τα οποία ενώνουν αμφιμονοσήμαντα δύο και μοναδικούς κόμβους .Όπου N = 51 είναι το σύνολο των σημείων που έχει το συγκεκριμένο παράδειγμα. Επομένως ,ο μη κατευθυνόμενος γράφος του σχήματος , αντιστοιχεί στην τελική διαδρομή που υπολογίστηκε από τον συγκεκριμένο αλγόριθμο .

Παρατηρήσεις :

❖ Για τον αλγόριθμο 4_opt ισχύουν τα ίδια που έχουμε αναφέρει και για τους άλλους N_opt αλγορίθμους .

Επιλέξαμε και στα 3 παραδείγματα N_opt αλγορίθμων **30.000 επανάληψεις** ,για να τα συγκρίνουμε μεταξύ τους . Όπως , παρατηρούμε , συγκρίνοντας τα αποτελέσματα , ο 2_opt , είναι εκείνος που για τον ίδιο αριθμό επαναλήψεων κατέληξε σε διαδρομή με μικρότερο κόστος από τους υπόλοιπους και για τον ίδιο χρόνο εκτέλεσης ≈ 60 δευτερόλεπτα. Για αυτό τον λόγο επιτρέπεται εύρος τιμών της παραμέτρου N , από 2 μέχρι και 4 . [8]

Η εφαρμογή του **Αλγορίθμου Βελτιστοποίησης Αποικίας Μυρμηγκιών (Ant Colony Optimization)** φαίνεται έμπρακτα, παρακάτω:

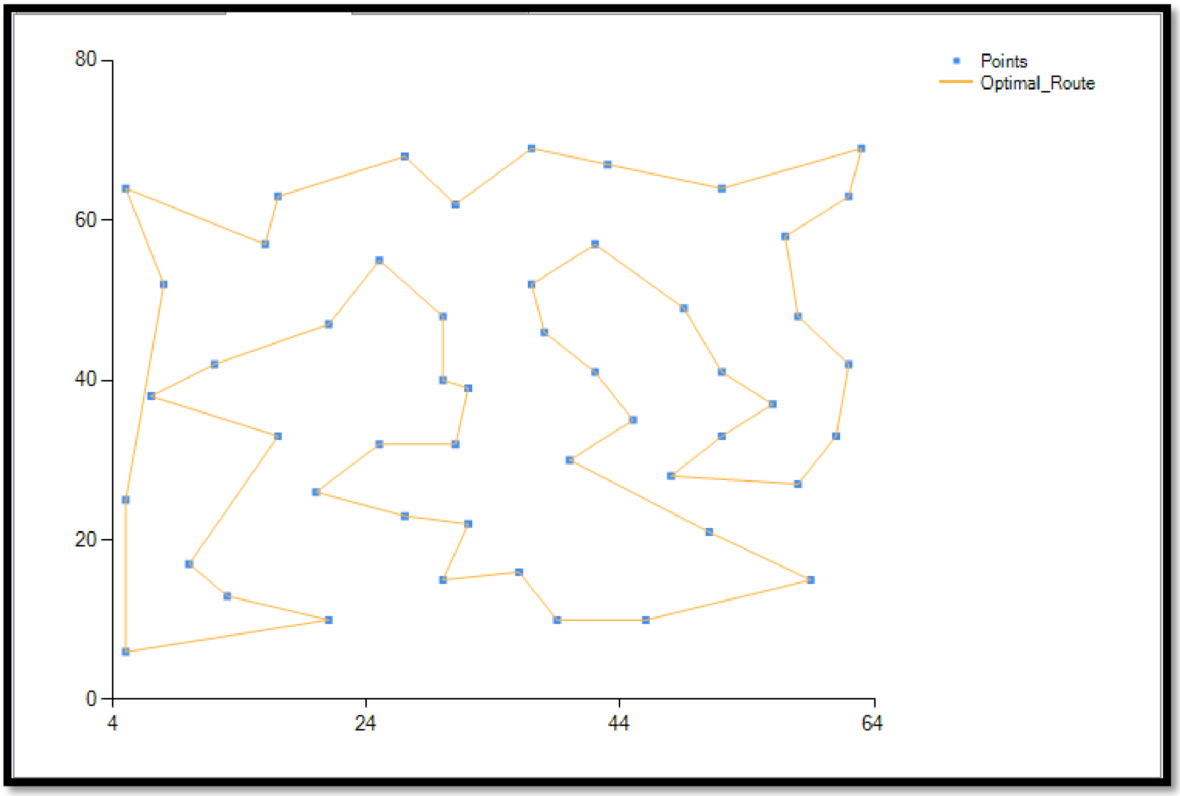
Ακολουθούν τα αποτελέσματα του παραδείγματος :

Dataset:	eil_best_manyiterations.xlsx
Solver Used:	ACO
Ants – Μυρμήγκια	100
Maxiter – Επαναλήψεις	1000
Ls – Τοπική Αναζήτηση	1
a – Συντελεστής φερομόνης	4
b – Συντελεστής Ομοιότητας	2
PER – Το ποσοστό της φερομόνης που εξατμίζεται σε κάθε επανάληψη	1
PRR – Το κατώφλι της φερομόνης , με βάση το οποίο απορίπτουμε ένα μονοπάτι	1
Q – Σταθερός παράγοντας	10
Solution Route – Η τελική διαδρομή :	40 42 19 41 18 25 14 6 48 27 51 46 12 47 4 17 37 44 15 45 33 39 10 5 38 11 32 1 22 2 16 50 9 49 30 34 21 29 20 35 36 3 28 31 8 26 7 23 43 24 13 40
Route Cost – Κόστος της διαδρομής:	461,5882
Cpu time (sec) – Χρόνος επεξεργασίας σε δευτερόλεπτα :	3493,72

***Πίνακας 3.4.7 :** Τα αποτελέσματα του παραδείγματος Eil_51 εφαρμόζοντας τον αλγόριθμο του ACO.*

Παραπάνω απεικονίζονται , οι κόμβοι της τελικής διαδρομής που υπολογίστηκε σαν βέλτιστη από τον αλγόριθμο, ενώ παράλληλα κρατούνται : ο αριθμός των μυρμηγκιών , ο αριθμός των επαναλήψεων , αν έχει γίνει τοπική αναζήτηση, ο συντελεστής φερομόνης , ο συντελεστής ομοιότητας , το PER, το PRR, το Q, η τελική διαδρομή , το κόστος της συνολικής διαδρομής και ο χρόνος εκτέλεσης του αλγορίθμου.

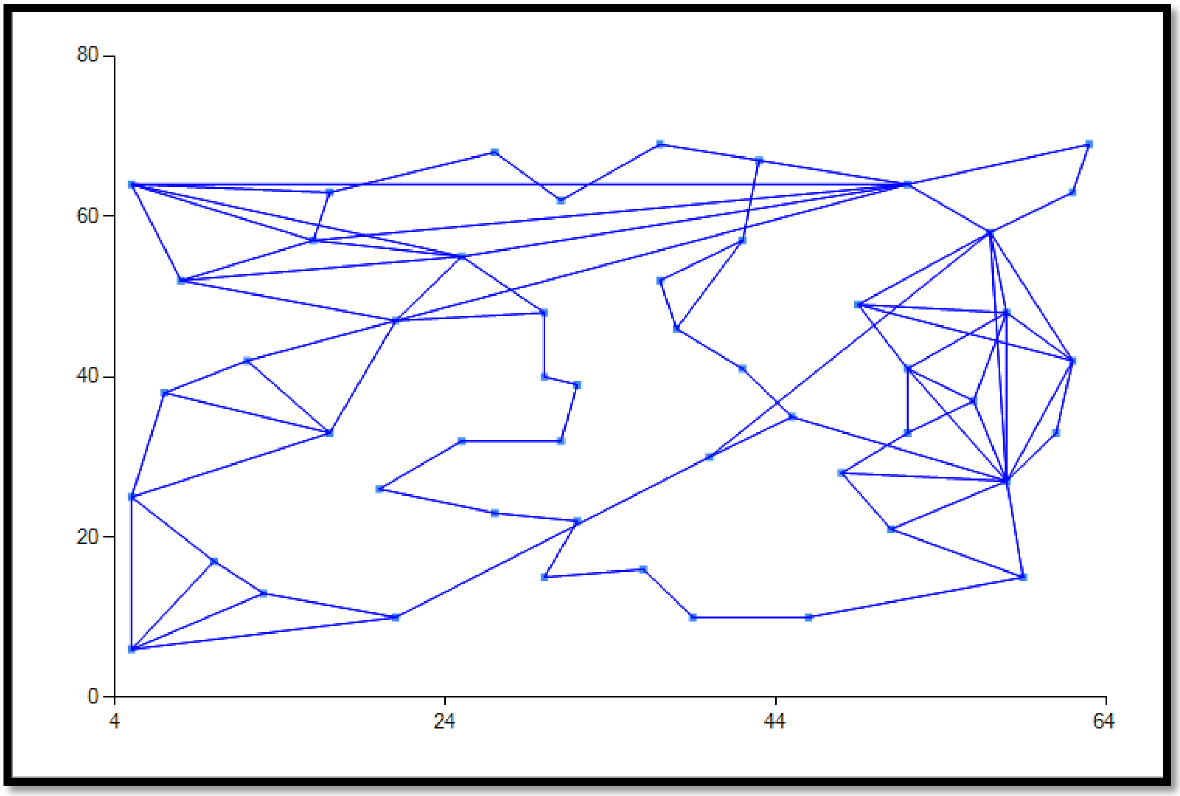
Ακολουθεί η γραφική αναπαράσταση της τελικής (βέλτιστης) διαδρομής του παραδείγματος :



Σχήμα 3.4.6: Η τελική διαδρομή του αλγορίθμου του ACO.

Παραπάνω απεικονίζεται , ένας μη κατευθυνόμενος γράφος ,ο οποίος αποτελείται από (50) ευθύγραμμα τμήματα τα οποία ενώνουν αμφιμονοσήμαντα δύο και μοναδικούς κόμβους .Επομένως ,ο μη κατευθυνόμενος γράφος του σχήματος , αντιστοιχεί στην τελική διαδρομή που υπολογίστηκε από τον συγκεκριμένο αλγόριθμο .

Ακολουθεί η γραφική αναπαράσταση του πίνακα γειτνίασης (**adjacent matrix**) του παραδείγματος :



Σχήμα 3.4.7: *Ο πίνακας γειτνίασης του αλγορίθμου του ACO.*

Παραπάνω απεικονίζεται ο πίνακας γειτνίασης του αλγορίθμου του ACO .

- Ο πίνακας γειτνίασης , κρατάει όλες τις ακμές του γράφου οι οποίες έχουν επίπεδα φερομόνης μεγαλύτερα από το κατώφλι απόριψης (**PRR**) μετά και την τελευταία επανάληψη του αλγορίθμου .
- Ο πίνακας γειτνίασης , είναι ενδεικτικός της ποιότητας των παραμέτρων που έχουν επιλεχθεί . Πιο συγκεκριμένα , αν οι ακμές που περιέχει είναι ίδιες ,ή και λίγο παραπάνω , αυτό υποδηλώνει ότι οι παράμετροι που έχουν επιλεχθεί είναι καλοί .

<u>Παρατηρήσεις :</u>

- Ο αλγόριθμος του ACO , συμβουλευόμενοι και τον *πίνακα 3.4.7* , στο 1^{o} βήμα , σχεδιάζει τον πίνακα ομοιότητας και τον αρχικό πίνακα της φερομόνης , τον οποίο αρχικοποιεί με την τιμή **100** , σε όλες τις θέσεις του πίνακα , εκτός από την κύρια διαγώνιο του που την μηδενίζουμε .

- Αντιλαμβάνεται κανείς δηλαδή , ότι στην αρχή του αλγορίθμου μας θα έχουν όλες οι ακμές του γράφου μας την ίδια τιμή φερομόνης (**100**) και άρα τα μυρμήγκια θα έχουν την ίδια ‘’επιθυμία’’ για να επισκεφθούν όλες τις πόλεις . Στην συνέχεια , αφού σχεδιαστεί ο γράφος με όλους τους κόμβους ενωμένους μεταξύ τους , τοποθετούνται τα **100 μυρμήγκια** σε τυχαίους κόμβους πάνω στο γράφο .

- Τα μηρμήγκια επιλέχθηκαν **100** , για δύο λόγους : Πρώτον , για να είναι περισσότερα από τους κόμβους (**51**) και με αυτόν τον τρόπο να υπάρχει μεγαλύτερη πιθανότητα να βρεθεί η βέλτιστη διαδρομή. Δεύτερον , επειδή , από τα **100** μυρμήγκια , μόνο το **25%** από αυτά θα προτιμηθούν για την εύρεση της διαδρομής .

- Στο επόμενο βήμα, κάθε μυρμήγκι έχει διανύσει και μία διαδρομή , την οποία θεωρεί σαν βέλτιστη . Σχεδιαστικά , το κατώφλι ($\mathbf{q_0}$) , με βάση το οποίο θα επιλεχθεί από τα μυρμήγκια ο επόμενος κόμβος , έχει την τιμή **(0,8)** [4] **.**

- Έχει επιλεχθεί , **συντελεστής φερομόνης a = 4** και **συντελεστής ομοιότητας b = 2** . Οι τιμές αυτές , επιλέχθηκαν , επειδή , ύστερα από παρατήρηση, οι αποστάσεις μεταξύ των κόμβων έχουν μικρές διαφορές και επομένως δόθηκε περισσότερη έμφαση στο ‘ένστικτο’ – φερομόνη των μυρμηγκιών , παρά την απόσταση των κόμβων μεταξύ τους .

- Στο σημείο αυτό , επειδή έχει επιλεχθεί η χρήση της τοπικής αναζήτησης , με την 2_opt μέθοδο , για βελτιστοποίηση των **100** διαθέσιμων μονοπατιών που έχουν σχεδιάσει αντίστοιχα τα 100 μυρμήγκια .Ο αριθμός των επαναλήψεων που χρησιμοποιούμε στην 2_opt μέθοδο , είναι **(20 * N = 20 * 51 = 1020 επαναλήψεις) ,** πέραν του **20 * N ,** ο αλγόριθμος , δεν έδωσε βελτίωση στα αποτελέσματα και αύξησε κατά πολύ τον χρόνο εκτέλεσης του αλγορίθμου.

- Στην συνέχεια, εξατμίζουμε , από όλες τις ακμές που έχουν δημιουργήσει οι 100 διαδρομές των μυρμηγκιών , το **1%** της φερομόνης που έχει κάθε ακμή . Έχουμε, επιλέξει **1000** επαναλήψεις εκτέλεσης του αλγόρίθμου και μικρό ποσοστό εξάτμισης της φερομόνης σε κάθε επανάληψη , για να έχουμε καλύτερα αποτελέσματα, να προλάβουν δηλαδή τα μυρμήγκια να ανακαλύψουν την βέλτιστη διαδρομή.

- Έπειτα , αφού ταξινομήσουμε , τα 100 μυρμήγκια σε αύξουσα σειρά, επιλέγουμε τα 25% καλύτερα από αυτά και συγκεκριμένα τα **25** .

- Η ποσότητα της φερομόνης που προστίθεται στις διαδρομές των 25 αυτών μυρμηγκιών είναι ανάλογη του σταθερού παράγοντα **Q** ,ο οποίος έχει επιλεχθεί ίσος με **(10)** [4] και αντιστρόφως ανάλογη της απόστασης της διαδρομής . Επομένως , μεγαλύτερη ποσότητα φερομόνης , προστίθεται στις μικρότερες σε απόσταση διαδρομές .

❖ Έχει , επιλεχθεί κατώφλι απόριψης (PRR) ίσο με **1** . Επιλέχθηκε , τόσο μικρό , για να διατηρηθούν μέχρι και το τέλος των επαναλήψεων οι ακμές που ανήκουν στην βέλτιστη διαδρομή. Αν τυχόν , το σύνολο των ακμών πέσει κάτω από **50 ,** στο συγκεκριμένο πρόβλημα , ο αλγόριθμος τερματίζεται .

❖ Τέλος , για καλύτερα αποτελέσματα , επιλέχθηκε να τοποθετηθούν τα μυρμήγκια , έναν κόμβο πρίν τελευταίο και αρχικό παράλληλα .

❖ Τα αποτελέσματά , μας επιβεβαίωσαν ακράδαντα . Είναι τα καλύτερα αποτελέσματα που βγήκαν , συγκρτιτικά με τους υπόλοιπους αλγορίθμους . Όσο , για την απόκλιση τους από το βέλτιστο μονοπάτι , είναι μόλις (**+8,2%**).Ο χρόνος , ωστόσο εκτέλεσης του αλγορίθμου , ύστερα κιόλας και από την προσθήκη των **20 * N** επαναλήψεων της 2_opt μεθόδου , σε κάθε επανάληψη του αλγορίθμου ACO (1000) , ήταν μεγάλος (**3493,72 sec** = **58.21 λεπτά**) .

Η εφαρμογή του **Γενετικού Αλγορίθμου – (Genetic Algorithm)** φαίνεται έμπρακτα, παρακάτω:

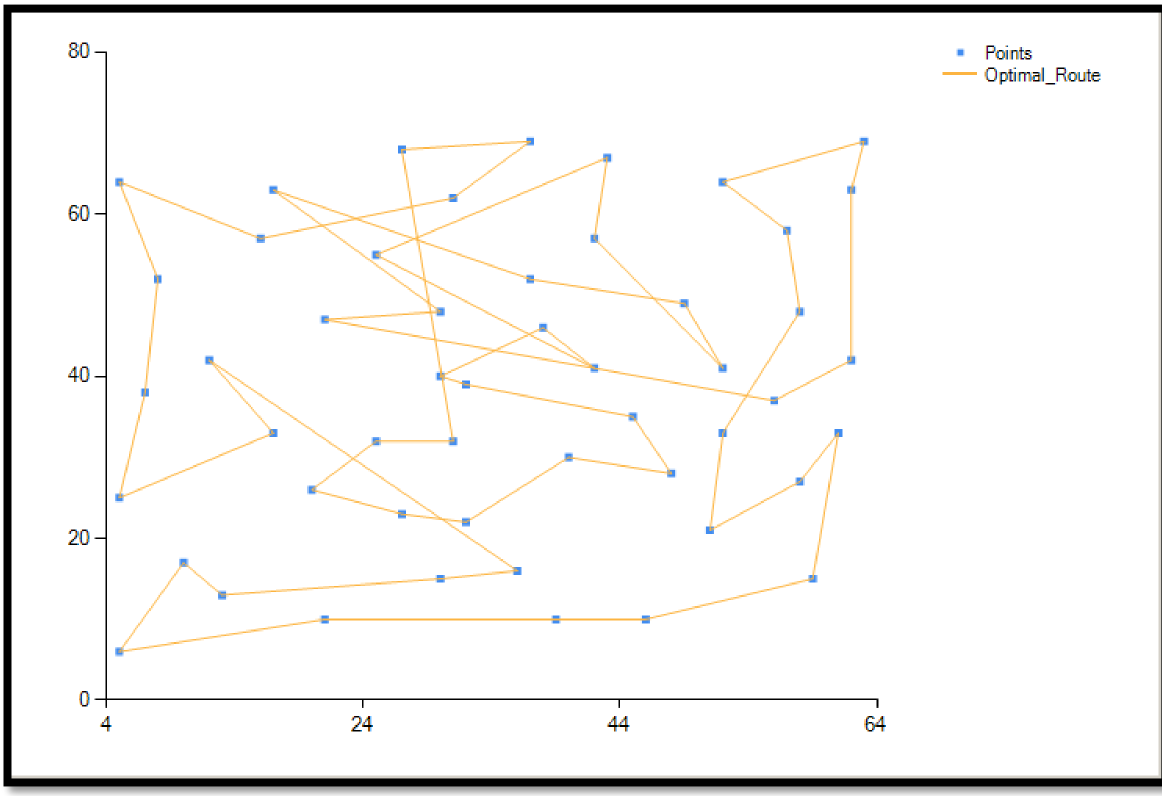
Ακολουθούν τα αποτελέσματα του παραδείγματος :

Dataset:	eil_ga_swap.xlsx
Solver Used:	Genetic Algorithm
Npopulace – Αριθμός Χρωμοσωμάτων	200
Iterations – Επαναλήψεις	3000
batch_size – Το μέγεθος της παρτίδας	4
mutation_type – Τύπος μετάλλαξης	0
Solution Route – Η Τελική διαδρομή :	30 34 39 33 45 42 40 41 19 44 15 14 18 13 25 24 43 23 8 31 26 12 47 4 17 37 5 49 38 46 51 32 11 48 28 22 16 2 1 7 27 6 50 21 35 36 3 20 29 9 10 30
Route Cost – Το κόστος της διαδρομής:	674,3351
Cpu time (sec) – Ο χρόνος εκτέλεσης σε δευτερόλεπτα:	40,84106

Πίνακας 3.4.8 : *Τα αποτελέσματα του παραδείγματος Eil_51 εφαρμόζοντας τον αλγόριθμο το γενετικό με τύπο μετάλλαξης 0.*

Παραπάνω απεικονίζονται , οι κόμβοι της τελικής διαδρομής που υπολογίστηκε σαν βέλτιστη από τον αλγόριθμο, ενώ παράλληλα κρατούνται : ο αριθμός των χρωμοσωμάτων , ο αριθμός των επαναλήψεων , το μέγεθος της παρτίδας που έχει κάθε ομάδα, ο τύπος της μετάλλαξης , η τελική διαδρομή , το κόστος της συνολικής διαδρομής και ο χρόνος εκτέλεσης του αλγορίθμου. Στο συγκεκριμένο, παράδειγμα ,χρησιμοποιήθηκε τύπος μετάλλαξης (**0**) , πιο συγκεκριμένα (swap / right - left shift / flip).

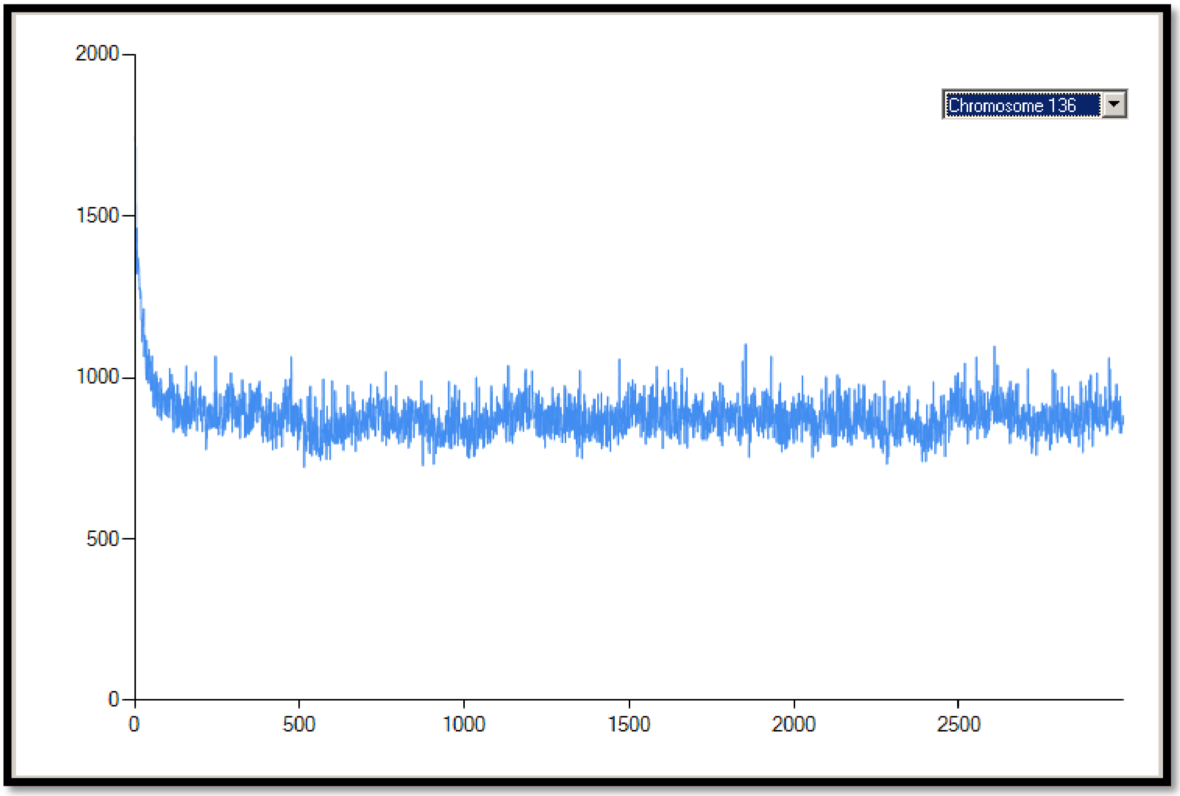
Ακολουθεί η γραφική αναπαράσταση της τελικής (βέλτιστης) διαδρομής του παραδείγματος , για **τύπο μετάλλαξης 0** :



Σχήμα 3.4.8: *Η τελική διαδρομή του γενετικού αλγορίθμου.*

Παραπάνω απεικονίζεται , ένας μη κατευθυνόμενος γράφος ,ο οποίος αποτελείται από (50) ευθύγραμμα τμήματα τα οποία ενώνουν αμφιμονοσήμαντα δύο και μοναδικούς κόμβους .Επομένως ,ο μη κατευθυνόμενος γράφος του σχήματος , αντιστοιχεί στην τελική διαδρομή που υπολογίστηκε από τον συγκεκριμένο αλγόριθμο .

Ακολουθεί η γραφική αναπαράσταση του πίνακα που κρατάει , *το κόστος κάθε χρωμοσώματος σε κάθε επανάληψη* **,** του παραδείγματος :



Σχήμα 3.4.9: *Το κόστος κάθε χρωμοσώματος στο σύνολο των επαναλήψεων.*

Παραπάνω απεικονίζεται ο πίνακας που κρατάει , *το κόστος κάθε χρωμοσώματος σε κάθε επανάληψη* του αλγορίθμου του γενετικού .

- Ο παραπάνω πίνακας , απεικονίζει την εξέλιξη του κάθε χρωμοσώματος , μέσα στο σύνολο , των επαναλήψεων .

Παρατηρήσεις :

- Ο γενετικός αλγόριθμος , συμβουλευόμενοι και τον *πίνακα 3.4.8* , αφού σχεδιάσει τον πίνακα της εξέλιξης των χρωμοσωμάτων , στην συνέχεια ελέγχεται αν ο αροθμός των χρωμοσωμάτων που έχουμε επιλέξει διαιρείται ακριβώς με τον αριθμό της παρτίδας . Στην προκειμένη περίπτωση , τα **200** χρωμοσώματα , διαιρούνται ακριβώς με το **4** τον αριθμό της παρτίδας (**200 / 4 = 50**) . Επομένως , έχουμε **50 ομάδες** που αντιστοιχούν **4 χρωμοσώματα σε κάθε ομάδα .**

- Έπειτα , δημιουργούνται **200** διαδρομές , όσος είναι ο αριθμός των χρωμοσωμάτων , και τοποθετούνται τυχαία .Αντιλαμβανόμαστε , επομένως , ότι λόγος που επιλέξαμε 200 χρωμοσώματα, είναι επειδή θέλουμε να έχουμε πολλές διαδρομές , μεγιστοποιώντας έτσι την πιθανότητα να βρούμε την βέλτιστη .
- Στην συνέχεια , αφού τα τοποθετηθούν σε 50 ομάδες των 4 σε κάθε ομάδα , επιλέγεται από κάθε ομάδα η διαδρομή – χρωμόσωμα , με το μικρότερο , με βάση το οποίο θα μεταλλαχθούν τα υπόλοιπα . Είναι , προφανές , επομένως , πως η χρησιμοποίηση 4 (η μικρότερη δυνατή επιλογή) χρωμοσωμάτων ανά ομάδα , επιλέχθηκε επειδή , στην περίπτωση , πού το χρωμόσωμα με το μικρότερο κόστος , δεν είναι τελικά τόσο καλό (πολύ μακριά δηλαδή από το βέλτιστο) , οι απόγονοι που θα δημιουργήσει , μέσω της μετάλλαξης , είναι οι λιγότεροι δυνατοί , δηλαδή 4 .
- Σε αυτό το παράδειγμα , χρησιμοποίθηκε τύπος μετάλλαξης 0 . Παρατηρούμε , ότι το κόστος της διαδρομής που υπολογίστηκε , είναι πολύ μακριά από το βέλτιστο (**+58,2%**) , παρόλο που ο αριθμός των επαναλήψεων είναι αρκετά μεγάλος (**3000**) . Αυτό , ωστόσο , είναι λογικό , καθώς , ο τύπος αυτός της μετάλλαξης στηρίζεται σε τύχαίες επιλογές δύο σημείων , τυχαίες αναστροφες αυτών , σκοπεύοντας έτσι , να υπάρξει βελτίωση του κόστους . Όλα αυτά τα παρατηρούμε και στο *σχήμα 3.4.9* , καθώς μέσα στην πάροδο των επαναλήψεων η τιμή του κάθε χρωμοσώματος δεν βελτιώνεται ιδιαίτερα , αλλά πάλεται μεταξύ ενός μικρού εύρους τιμών κόστους.

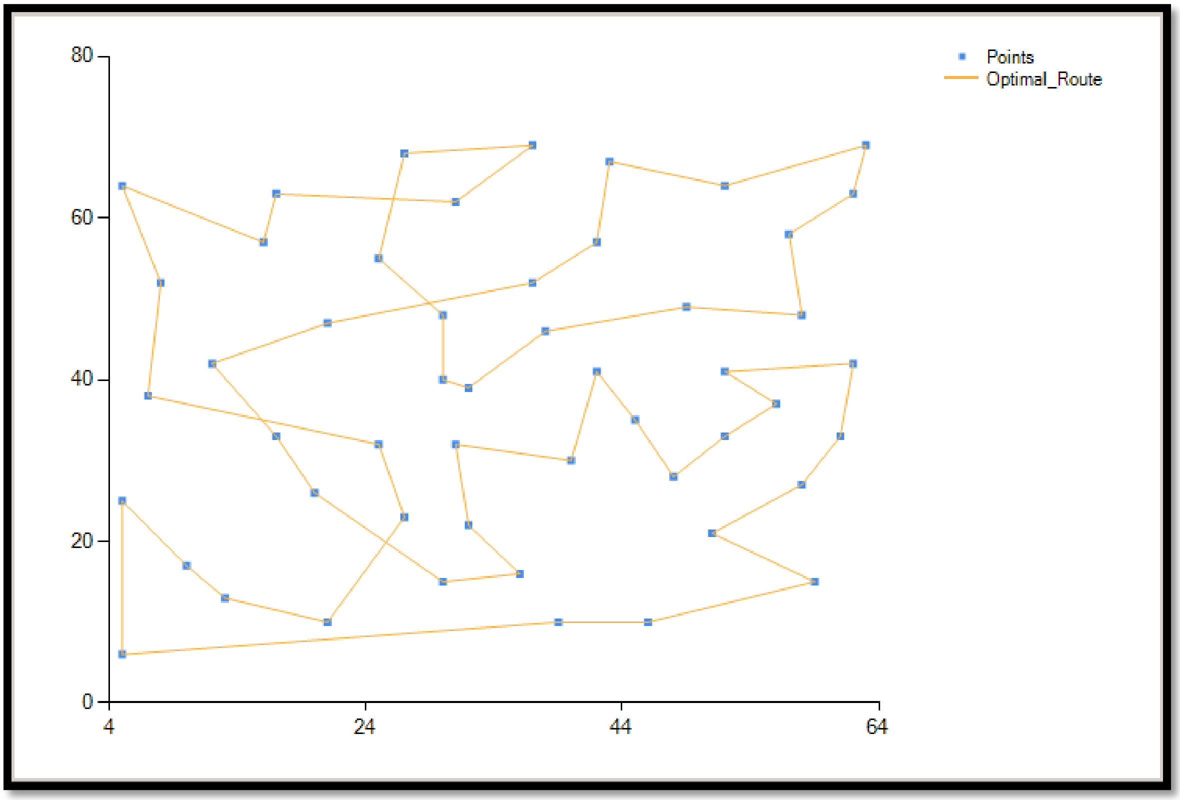
Ακολουθούν τα αποτελέσματα του παραδείγματος **Eil_51** , για **τύπο μετάλλαξης 1** :

Dataset:	eil_ga_2_opt.xlsx
Solver Used:	Genetic Algorithm
Npopulace – Αριθμός Χρωμοσωμάτων	200
Iterations – Επαναλήψεις	3000
batch_size – Το μέγεθος της παρτίδας	4
mutation_type – Τύπος μετάλλαξης	1
Solution Route – Η Τελική διαδρομή :	8 7 23 43 24 25 47 17 42 19 41 13 40 45 33 39 10 30 34 21 16 50 9 49 38 11 5 12 37 15 44 4 18 14 6 1 22 28 3 36 35 20 29 2 32 46 51 27 48 26 31 8
Route Cost – Το κόστος της διαδρομής:	503,6623
Cpu time (sec) – Ο χρόνος εκτέλεσης σε δευτερόλεπτα:	416,281

Πίνακας 3.4.9 : *Τα αποτελέσματα του παραδείγματος Eil_51 εφαρμόζοντας τον αλγόριθμο το γενετικό με τύπο μετάλλαξης 1.*

Παραπάνω απεικονίζονται , οι κόμβοι της τελικής διαδρομής που υπολογίστηκε σαν βέλτιστη από τον αλγόριθμο, ενώ παράλληλα κρατούνται : ο αριθμός των χρωμοσωμάτων , ο αριθμός των επαναλήψεων , το μέγεθος της παρτίδας που έχει κάθε ομάδα, ο τύπος της μετάλλαξης , η τελική διαδρομή , το κόστος της συνολικής διαδρομής και ο χρόνος εκτέλεσης του αλγορίθμου. Στο συγκεκριμένο, παράδειγμα ,χρησιμοποιήθηκε τύπος μετάλλαξης (**1**) , πιο συγκεκριμένα (2_opt τοπική αναζήτηση).

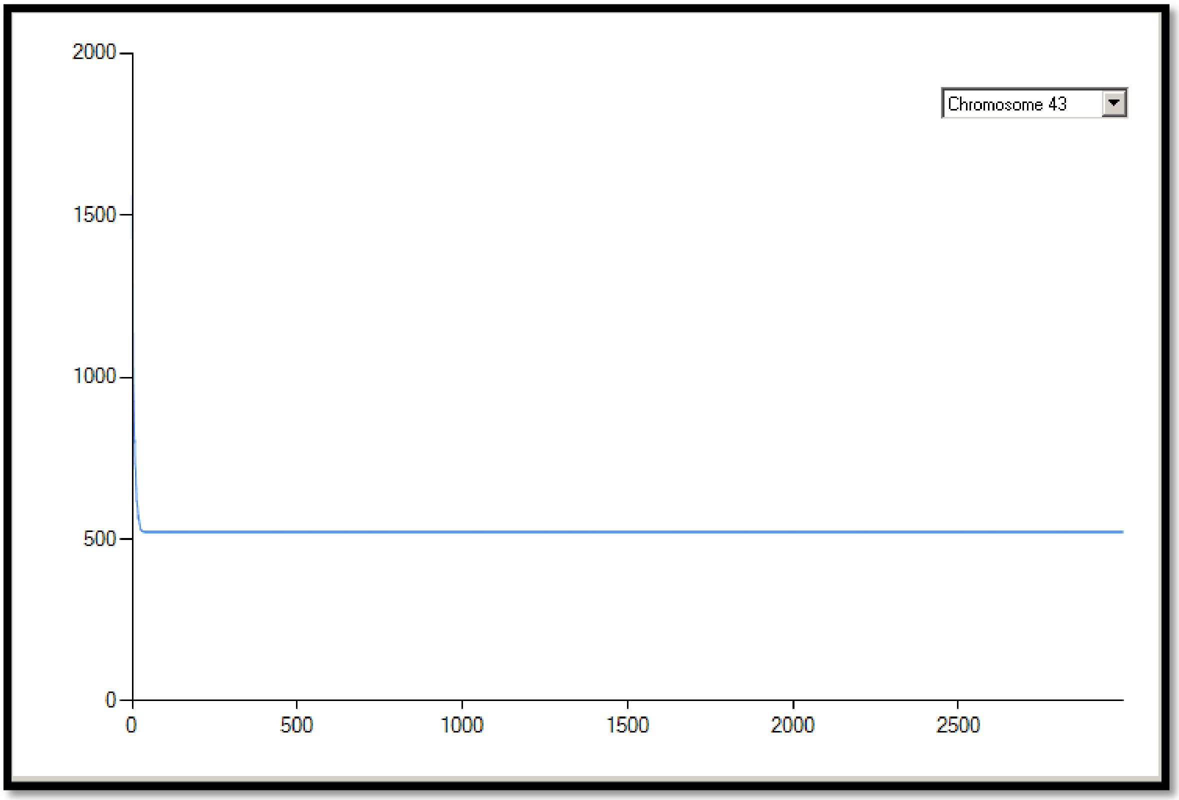
Ακολουθεί η γραφική αναπαράσταση της τελικής (βέλτιστης) διαδρομής του παραδείγματος , για **τύπο μετάλλαξης 1** :



Σχήμα 3.4.10: *Η τελική διαδρομή του γενετικού αλγορίθμου.*

Παραπάνω απεικονίζεται , ένας μη κατευθυνόμενος γράφος ,ο οποίος αποτελείται από (50) ευθύγραμμα τμήματα τα οποία ενώνουν αμφιμονοσήμαντα δύο και μοναδικούς κόμβους .Επομένως ,ο μη κατευθυνόμενος γράφος του σχήματος , αντιστοιχεί στην τελική διαδρομή που υπολογίστηκε από τον συγκεκριμένο αλγόριθμο .

Ακολουθεί η γραφική αναπαράσταση του πίνακα που κρατάει , *το κόστος κάθε χρωμοσώματος σε κάθε επανάληψη* **,** του παραδείγματος :



Σχήμα 3.4.11: *Το κόστος κάθε χρωμοσώματος στο σύνολο των επαναλήψεων.*

Παραπάνω απεικονίζεται ο πίνακας που κρατάει , *το κόστος κάθε χρωμοσώματος σε κάθε επανάληψη* του αλγορίθμου του γενετικού .

<u>Παρατηρήσεις :</u>

- Σε αυτό το παράδειγμα , χρησιμοποίθηκε τύπος μετάλλαξης 1 . Παρατηρούμε , ότι το κόστος της διαδρομής που υπολογίστηκε , δεν είναι μακριά από το βέλτιστο (**+18,02%**) . Παράλληλα ,το κόστος της συνολικής διαδρομής είναι μικρότερο συγκριτικά με τον τύπο μετάλλαξης 0. Αυτό , είναι λογικό , καθώς , ο τύπος αυτός της μετάλλαξης παρόλο πού και αυτός, στηρίζεται σε τύχαίες επιλογές δύο σημείων και τυχαίες αναστροφες αυτών , στον ίδιο αριθμό επαναλήψεων η τοπική αναζήτηση 2_opt σε κάθε επανάληψη , εκτελείται **N** φορές, όπου **N = 51** ο αριθμός των χρωμοσωμάτων . Γι’ αυτό το λόγο ο χρόνος εκτέλεσης του γενετικού αλγορίθμου με τύπο μετάλλαξης 1 είναι διπλάσιος από το γενετικό με τύπο 0 . Όλα τα παραπάνω, τα παρατηρούμε και στο *σχήμα 3.4.11* , καθώς μέσα στην πάροδο των επαναλήψεων η τιμή του κάθε χρωμοσώματος βελτιώνεται καλύτερα αποκτώντας , τελικά μία τιμή σταθερή , που είναι και ο στόχος μας .

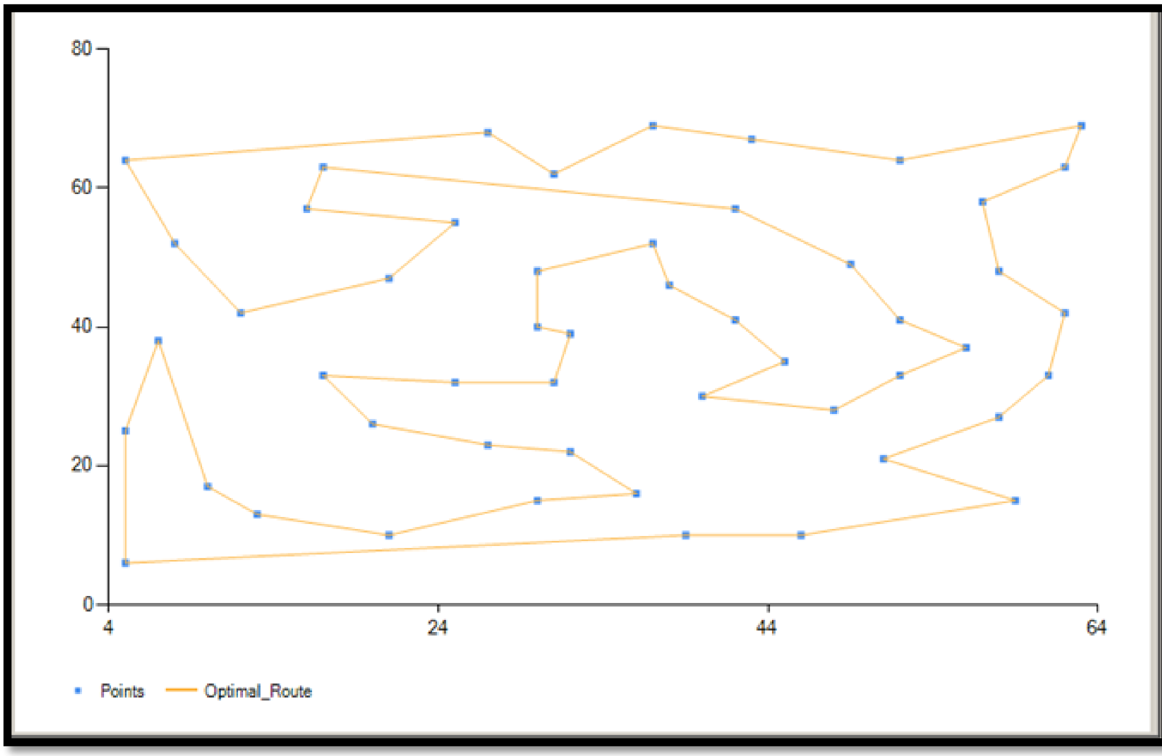
Ακολουθούν τα αποτελέσματα του παραδείγματος **Eil_51** , για **τύπο μετάλλαξης 2** :

Dataset:	**eil_ga_greedy.xlsx**
Solver Used:	Genetic Algorithm
Npopulace – Αριθμός Χρωμοσωμάτων	200
Iterations – Επαναλήψεις	3000
batch_size – Το μέγεθος της παρτίδας	4
mutation_type – Τύπος μετάλλαξης	2
Solution Route – Η Τελική διαδρομή :	41 19 42 44 15 37 17 4 18 47 12 46 51 27 1 32 11 38 5 49 9 50 16 2 22 7 23 48 6 14 24 43 26 8 31 28 3 36 35 20 29 21 34 30 10 39 33 45 40 13 25 41
Route Cost – Το κόστος της διαδρομής:	492,3026
Cpu time (sec) – Ο χρόνος εκτέλεσης σε δευτερόλεπτα:	956,875

Πίνακας 3.4.10 : *Τα αποτελέσματα του παραδείγματος Eil_51 εφαρμόζοντας τον αλγόριθμο το γενετικό με τύπο μετάλλαξης 2.*

Παραπάνω απεικονίζονται , οι κόμβοι της τελικής διαδρομής που υπολογίστηκε σαν βέλτιστη από τον αλγόριθμο, ενώ παράλληλα κρατούνται : ο αριθμός των χρωμοσωμάτων , ο αριθμός των επαναλήψεων , το μέγεθος της παρτίδας που έχει κάθε ομάδα, ο τύπος της μετάλλαξης , η τελική διαδρομή , το κόστος της συνολικής διαδρομής και ο χρόνος εκτέλεσης του αλγορίθμου. Στο συγκεκριμένο, παράδειγμα ,χρησιμοποιήθηκε τύπος μετάλλαξης (**2**) , πιο συγκεκριμένα (greedy αλγόριθμο).

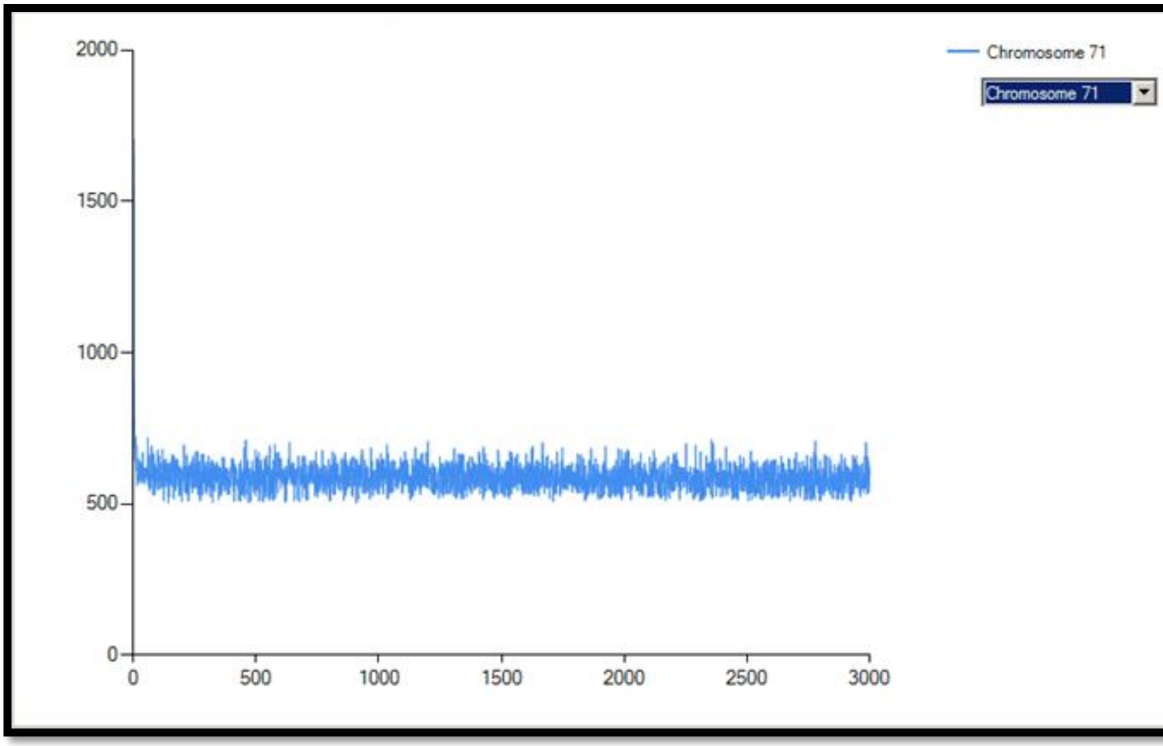
Ακολουθεί η γραφική αναπαράσταση της τελικής (βέλτιστης) διαδρομής του παραδείγματος , για **τύπο μετάλλαξης 2**:



Σχήμα 3.4.12: *Η τελική διαδρομή του γενετικού αλγορίθμου.*

Παραπάνω απεικονίζεται , ένας μη κατευθυνόμενος γράφος ,ο οποίος αποτελείται από (50) ευθύγραμμα τμήματα τα οποία ενώνουν αμφιμονοσήμαντα δύο και μοναδικούς κόμβους .Επομένως ,ο μη κατευθυνόμενος γράφος του σχήματος , αντιστοιχεί στην τελική διαδρομή που υπολογίστηκε από τον συγκεκριμένο αλγόριθμο .

Ακολουθεί η γραφική αναπαράσταση του πίνακα που κρατάει , *το κόστος κάθε χρωμοσώματος σε κάθε επανάληψη* **,** του παραδείγματος :



Σχήμα 3.4.13: *Το κόστος κάθε χρωμοσώματος στο σύνολο των επαναλήψεων.*

Παραπάνω απεικονίζεται ο πίνακας που κρατάει , *το κόστος κάθε χρωμοσώματος σε κάθε επανάληψη* του αλγορίθμου του γενετικού .

Παρατηρήσεις :

- Σε αυτό το παράδειγμα , χρησιμοποίθηκε τύπος μετάλλαξης 2 . Παρατηρούμε , ότι το κόστος της διαδρομής που υπολογίστηκε , πλησίαζει το βέλτιστο (**+11,97%**) . Παράλληλα ,το κόστος της συνολικής διαδρομής είναι μικρότερο συγκριτικά με τους παραπάνω τύπους μετάλλαξης . Αυτό , είναι λογικό , καθώς , ο τύπος αυτός της μετάλλαξης στηρίζεται σε μετάλλαξη με τον αλγόριθμο απληστίας , οποίος από μόνος του καταλήγει σε καλύτερα αποτελέσματα συγκριτικά με την τοπική αναζήτηση 2_opt. Όλα τα παραπάνω, τα παρατηρούμε και στο *σχήμα 3.4.13* , καθώς μέσα στην πάροδο των επαναλήψεων η τιμή του κάθε χρωμοσώματος πάλεται μεν μεταξύ ενός μικρού εύρους τιμών, το οποίο ωστόσο αποκτά τιμές μικρότερες από ότι του 2_opt .

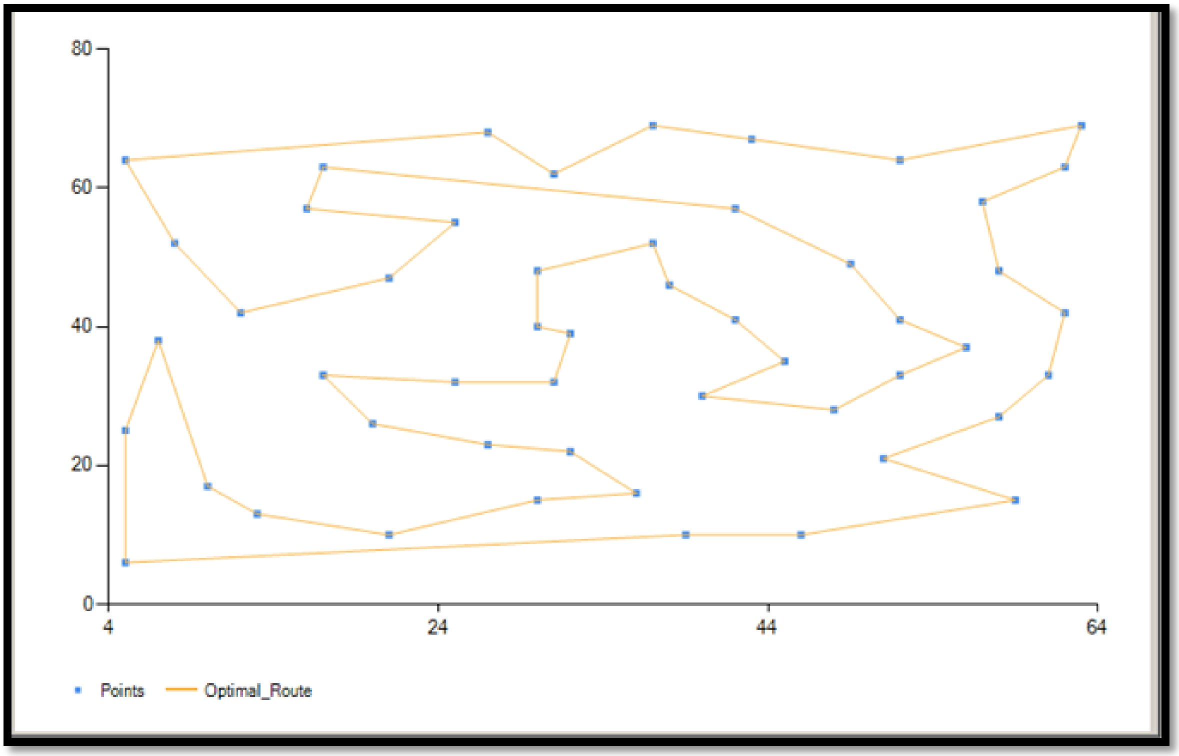
Ακολουθούν τα αποτελέσματα του παραδείγματος **Eil_51** , για **τύπο μετάλλαξης 3** :

Dataset:	eil_ga_greedy.xlsx
Solver Used:	Genetic Algorithm
Npopulace – Αριθμός Χρωμοσωμάτων	200
Iterations – Επαναλήψεις	3000
batch_size – Το μέγεθος της παρτίδας	4
mutation_type – Τύπος μετάλλαξης	3
Solution Route – Η Τελική διαδρομή :	25 13 41 40 19 42 47 12 46 51 27 1 32 11 38 5 49 9 21 29 20 35 36 3 28 31 8 26 22 2 16 50 34 30 10 39 33 45 15 44 37 17 4 18 14 6 48 23 7 43 24 25
Route Cost – Το κόστος της διαδρομής:	457,0596
Cpu time (sec) – Ο χρόνος εκτέλεσης σε δευτερόλεπτα:	5253,271

Πίνακας 3.4.11 : *Τα αποτελέσματα του παραδείγματος Eil_51 εφαρμόζοντας τον γενετικό αλγόριθμο με τύπο μετάλλαξης 3.*

Παραπάνω απεικονίζονται , οι κόμβοι της τελικής διαδρομής που υπολογίστηκε σαν βέλτιστη από τον αλγόριθμο, ενώ παράλληλα κρατούνται : ο αριθμός των χρωμοσωμάτων , ο αριθμός των επαναλήψεων , το μέγεθος της παρτίδας που έχει κάθε ομάδα, ο τύπος της μετάλλαξης , η τελική διαδρομή , το κόστος της συνολικής διαδρομής και ο χρόνος εκτέλεσης του αλγορίθμου. Στο συγκεκριμένο, παράδειγμα ,χρησιμοποιήθηκε τύπος μετάλλαξης (**3**) , πιο συγκεκριμένα (greedy αλγόριθμο).

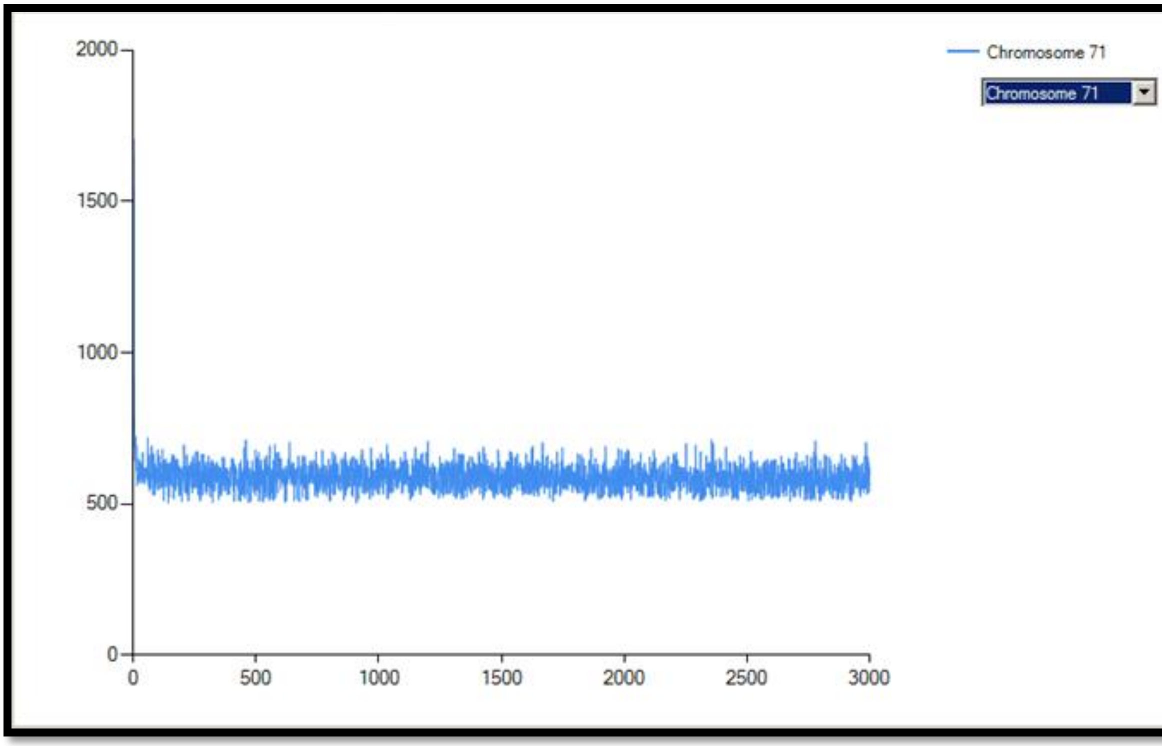
Ακολουθεί η γραφική αναπαράσταση της τελικής (βέλτιστης) διαδρομής του παραδείγματος , για **τύπο μετάλλαξης 3**:



Σχήμα 3.4.14: *Η τελική διαδρομή του γενετικού αλγορίθμου.*

Παραπάνω απεικονίζεται , ένας μη κατευθυνόμενος γράφος ,ο οποίος αποτελείται από (50) ευθύγραμμα τμήματα τα οποία ενώνουν αμφιμονοσήμαντα δύο και μοναδικούς κόμβους .Επομένως ,ο μη κατευθυνόμενος γράφος του σχήματος , αντιστοιχεί στην τελική διαδρομή που υπολογίστηκε από τον συγκεκριμένο αλγόριθμο .

Ακολουθεί η γραφική αναπαράσταση του πίνακα που κρατάει , *το κόστος κάθε χρωμοσώματος σε κάθε επανάληψη* **,** του παραδείγματος :



Σχήμα 3.4.15: *Το κόστος κάθε χρωμοσώματος στο σύνολο των επαναλήψεων.*

Παραπάνω απεικονίζεται ο πίνακας που κρατάει , *το κόστος κάθε χρωμοσώματος σε κάθε επανάληψη* του αλγορίθμου του γενετικού .

Παρατηρήσεις :

- Σε αυτό το παράδειγμα , χρησιμοποίθηκε τύπος μετάλλαξης 3 . Παρατηρούμε , ότι το κόστος της διαδρομής που υπολογίστηκε , πλησίαζει ακόμα περισσότερο το βέλτιστο (**+7,27%**) . Παράλληλα ,το κόστος της συνολικής διαδρομής είναι μικρότερο συγκριτικά με όλους τους παραπάνω τύπους μετάλλαξης . Αυτό , είναι λογικό , καθώς , ο τύπος αυτός της μετάλλαξης στηρίζεται σε μετάλλαξη με τον αλγόριθμο απληστίας με την στρατηγική του πλησιέστερου γείτονα , ο οποίος από μόνος του καταλήγει σε καλύτερα αποτελέσματα συγκριτικά με τους υπόλοιπους. Όλα τα παραπάνω, τα παρατηρούμε και στο *σχήμα 3.4.15* , καθώς μέσα στην πάροδο των επαναλήψεων η τιμή του κάθε χρωμοσώματος πάλεται μεν και εδώ μεταξύ ενός μικρού εύρους τιμών, το οποίο ωστόσο αποκτά τιμές μικρότερες από τα παραπάνω .

Κεφάλαιο 4:

Λογισμικά που χρησιμοποιήθηκαν

Τα λογισμικά που χρησιμοποιήθηκαν , για την υλοποίηση του υπολογιστικού μας συστήματος , είναι τα ακόλουθα :

1) **MATLAB R2009b - (Υλοποίηση του κώδικα)**

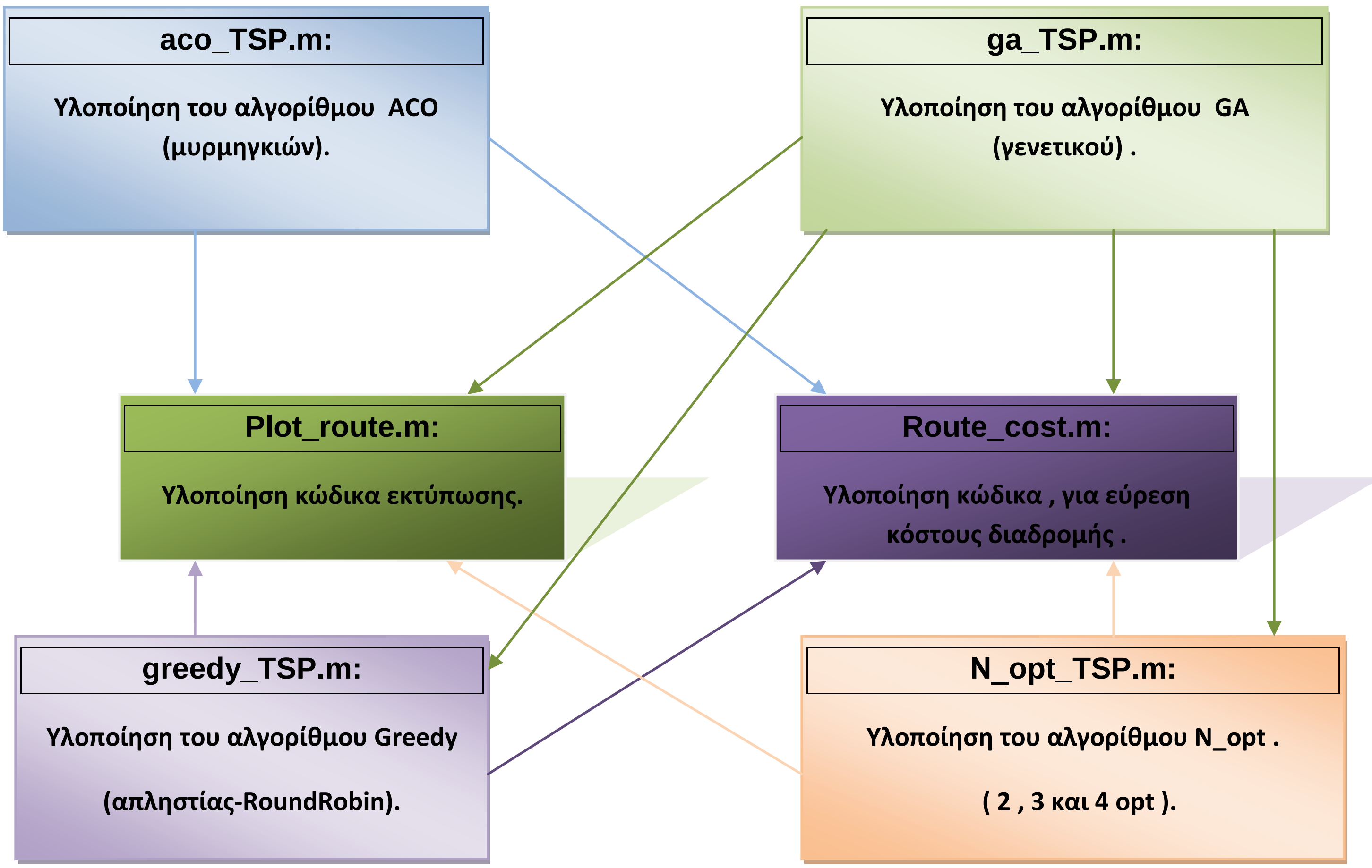
2) **MICROSOFT SQL SERVER 2008 – (Χρησιμοποίηση Βάσης Δεδομένων)**

3) **MICROSOFT VISUAL BASIC 2010 EXPRESS - (Υλοποίηση της τελικής διεπαφής)**

4.1 Το λογισμικό της MATLAB R2009b

Στο λογισμικό της Matlab , υλοποίηθηκε , το σχεδιαστικό κομμάτι του κώδικα των αλγορίθμων , του κώδικα του κόστους διαδρομής καθώς και του κώδικα σχεδίασης της διαδρομής .

Οι προηγούμενες συνδέσεις , φαίνονται αναλυτικά παρακάτω:



Σχήμα 4.1.1 : *Οι συνδέσεις που υλοποιήθηκαν στο λογισμικό του Matlab.*

Στο παραπάνω σχήμα φαίνονται , αναλυτικά οι συνδέσεις που υλοποιήθηκαν στον κώδικα της MATLAB . Φαίνεται , καθαρά ποιες συναρτήσεις καλούνε , άλλες συναρτήσεις . Η κατεύθυνση του βέλους , υποδηλώνει την συνάρτηση που θα καλεστεί .

4.2 Το λογισμικό του MICROSOFT SQL SERVER 2008

Στο λογισμικό του **MICROSOFT SQL SERVER 2008** , υλοποίηθηκε ,μία βάση δεδομένων , στην οποία αποθηκεύονται τα εξής :

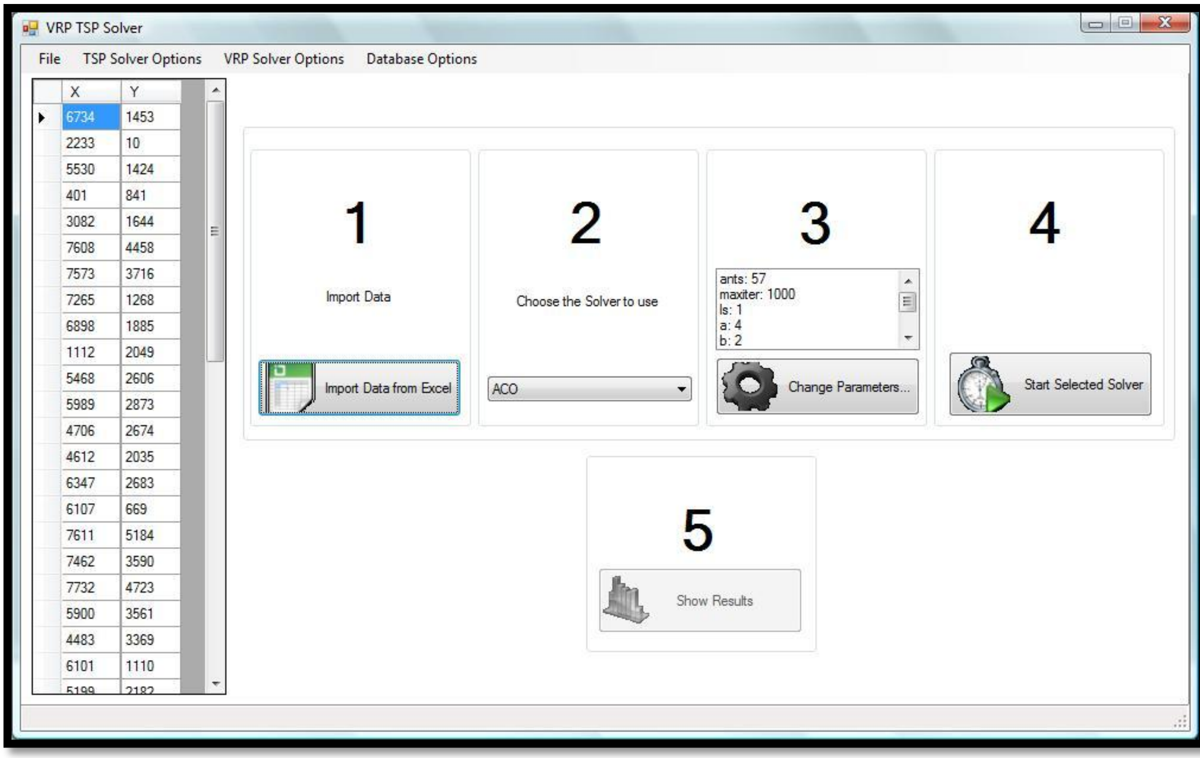
- Αρχικά αποθηκευόνται τα δεδομένα του προβλήματος TSP και πιο συγκεκριμένα οι συντεταγμένες του κάθε κόμβου.

- Έπειτα , αποθηκεύονται , οι παράμετροι του κάθε αλγορίθμου , αν τυχόν διαθέτει . Πιο συγκεκριμένα , για τον Greedy δεν κρατάει κάποια , παράμετρο , καθώς δεν έχει .

- Τέλος , αποθηκεύεται , η τελική διαδρομή του προβλήματος , το συνολικό κόστος της διαδρομής καθώς και ο συνολικός χρόνος επεξεργασίας . Στην περίπτωση του αλγορίθμου ACO κρατείται και ο πίνακας γειτνίασης . Ενώ , στην περίπτωση του γενετικού αλγορίθμου , κρατείται το κόστος κάθε χρωμοσώματος σε κάθε επανάληψη .

4.3 Το λογισμικό του MICROSOFT VISUAL BASIC 2010 EXPRESS

Στο λογισμικό του **MICROSOFT VISUAL MICROSOFT VISUAL BASIC 2010 EXPRESS** , σχεδιάστηκε ,η τελική διεπαφή (interface) . Το περιβάλλον , δηλαδή το οποίο , χρησιμοποιούμε , για να επιλέξουμε , τα δεδομένα μας , τους αλγορίθμους που θέλουμε να χρησιμοποιήσουμε , τις παραμέτρους που θα χρησιμοποιήσουμε και πολλά άλλα , τα οποία εικονίζονται παρακάτω :



Εικόνα 4.3.1:

Στην παραπάνω , εικόνα , φαίνεται το περιβάλλον – η διεπαφή στην οποία δουλεύουμε και στην οποία επιλέγουμε τα δεδομένα , τις παραμέτρους του κάθε αλγορίθμου , που θα χρησιμοποιήσουμε . Ενώ τέλος, μας παρουσιάζονται τα αποτέλεσματα εικονικά . Ακολουθώντας απλά, τα 5 βήματα που φαίνονται παραπάνω , λαμβάνουμε την τελική διαδρομή μαζί με το συνολικό της κόστος και τον χρόνο επεξεργασίας του αλγορίθμου που χρησιμοποιήσαμε.

X	Y
6734	1453
2233	10
5530	1424
401	841
3082	1644
7608	4458
7573	3716
7265	1268
6898	1885
1112	2049
5468	2606
5989	2873
4706	2674
4612	2035
6347	2683
6107	669
7611	5184
7462	3590
7732	4723
5900	3561
4483	3369
6101	1110
5199	2182

<u>Εικόνα 4.3.2:</u>

Στην παραπάνω εικόνα φαίνονται οι συντεταγμένες των σημείων του προβλήματος TSP, οι οποίες φορτώθηκαν μέσα από αρχείο του Excel.

Εικόνα 4.3.3:

Στην παραπάνω εικόνα απεικονίζεται το 1^{o} Βήμα που ακολουθούμε , πατώντας το πλήκτρο , για εισαγωγή δεδομένων μέσα από αρχείο του Excel.

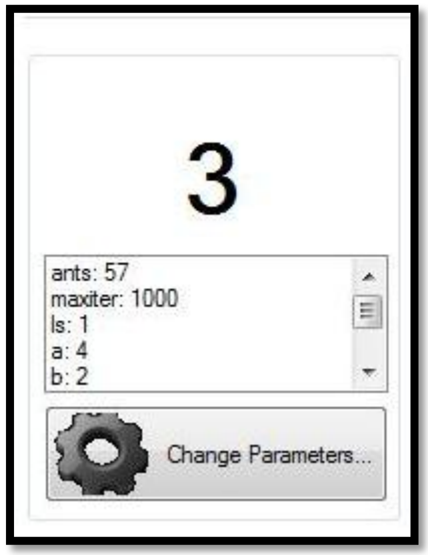
Στην παραπάνω εικόνα απεικονίζονται όλες οι επιλογές που μπορείς να κάνεις.

Εικόνα 4.3.4:

***Στην παραπάνω εικόνα απεικονίζεται το 2^o Βήμα που ακολουθούμε , πατώντας το πλήκτρο* , επιλέγουμε αλγόριθμο , εμφανίζοντας την *εικόνα 4.3.5* .**

Εικόνα 4.3.5:

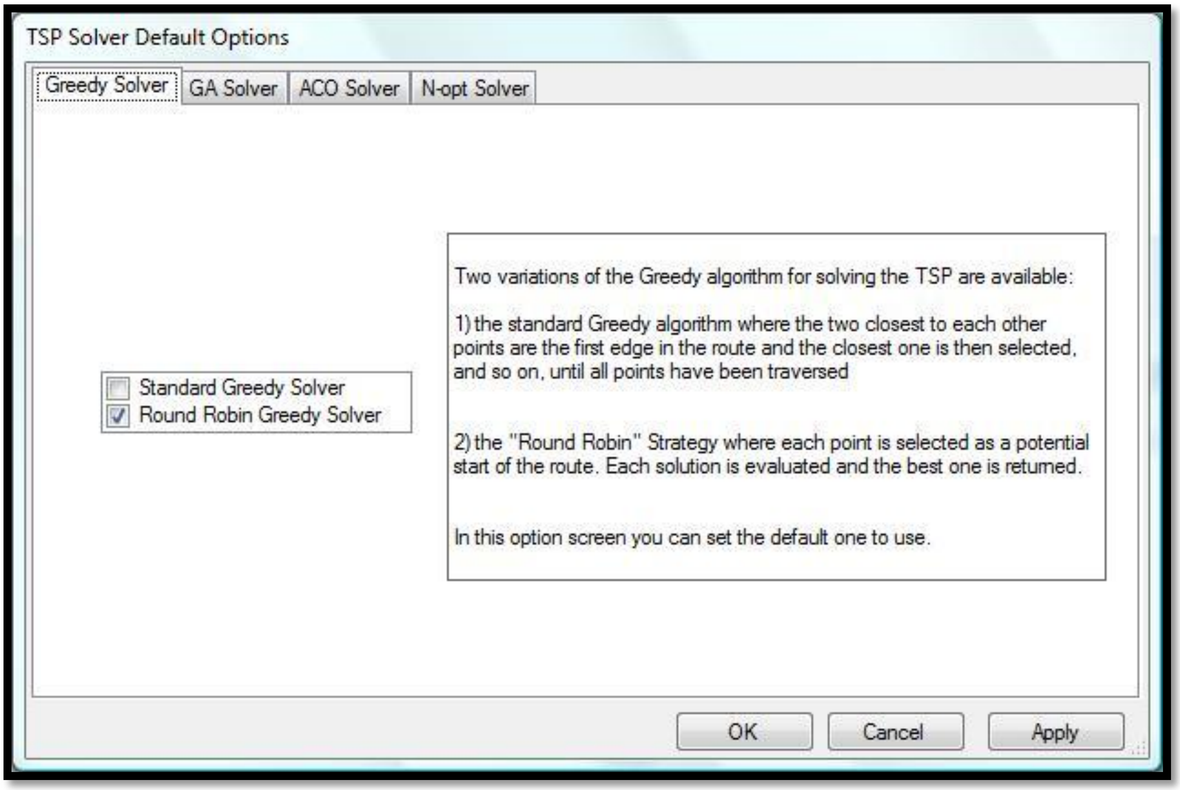
***Στην παραπάνω εικόνα απεικονίζονται όλοι οι δυνατοί αλγόριθμοι που μπορείς να χρησιμοποιήσεις*.**



<u>Εικόνα 4.3.6:</u>

Στην παραπάνω εικόνα απεικονίζεται το 3ο Βήμα που ακολουθούμε , πατώντας το πλήκτρο **, επιλέγουμε τις παραμέτρους του αλγορίθμου, που έχουμε επιλέξει . Ανάλογα τι αλγόριθμο επιλέξαμε στο βήμα 2, εμφανίζονται οι ακόλουθες εικόνες :**

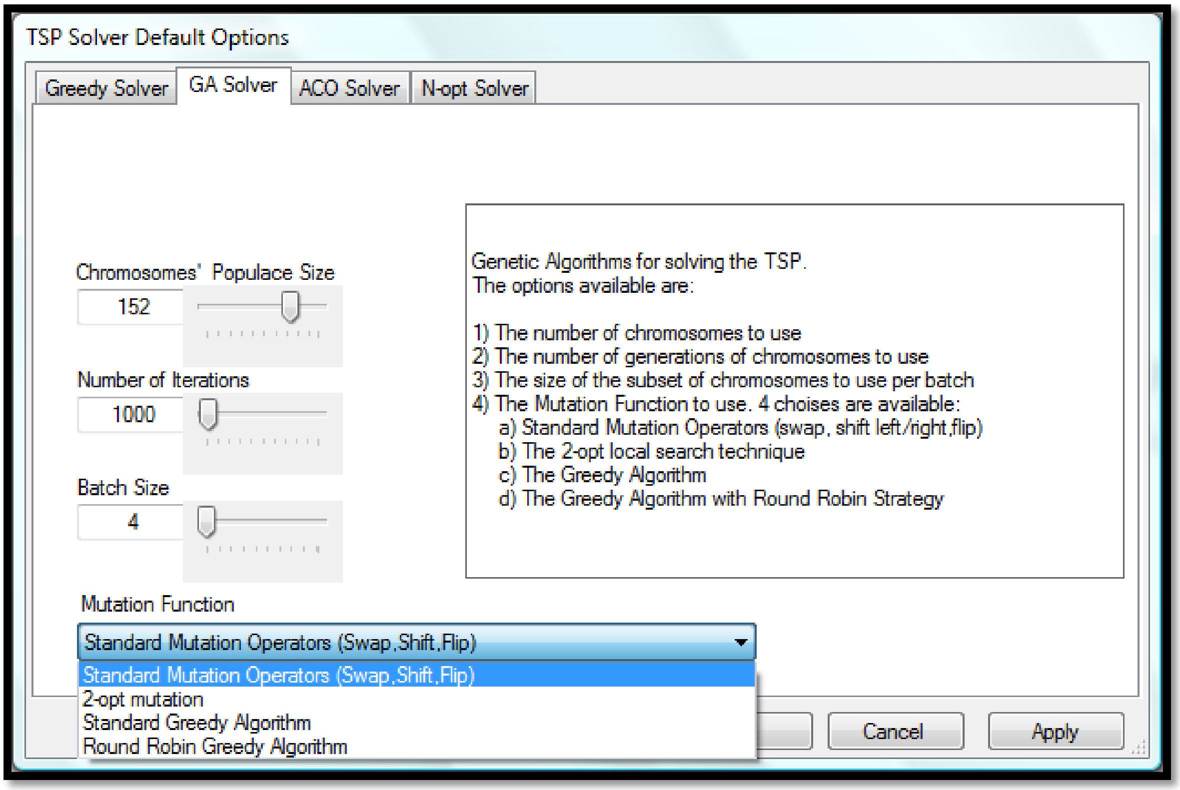
Για επιλογή Greedy (άπληστου) εμφανίζεται η εικόνα :



<u>Εικόνα 4.3.7 :</u>

Στην παραπάνω εικόνα απεικονίζονται οι παράμετροι του αλγορίθμου απληστίας . Επιλέγουμε αν θα εκτελεστεί ο απλός ο greedy , ή ο greedy with Round Robin , τικάροντας απλά στο διπλανό τους κουτάκι . Ενώ τέλος , για να αποθηκευτούν οι επιλογές που έγιναν , πατάμε το πλήκτρο Apply και έπειτα το πλήκτρο OK , σε αντίθετη περίπτωση πατάμε το πλήκτρο Cancel.

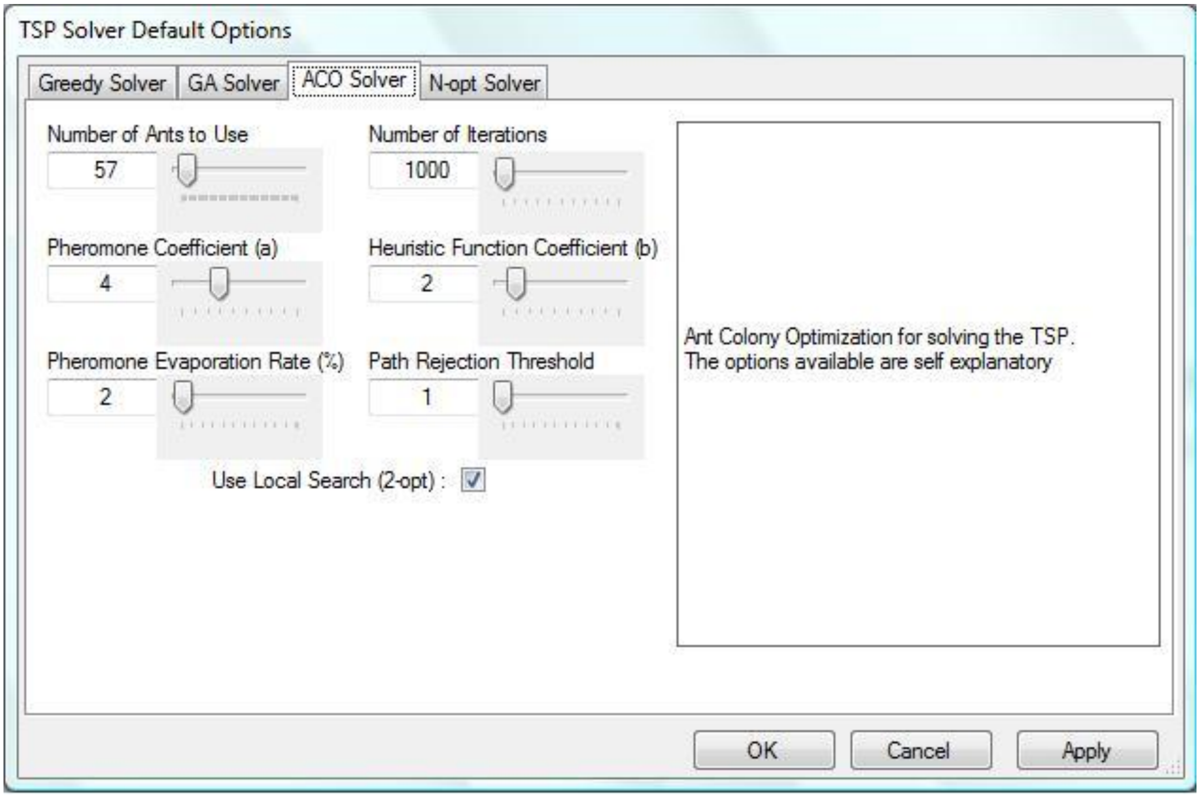
Για επιλογή Genetic algorithm (γενετικού αλγορίθμου) εμφανίζεται η εικόνα :



<u>Εικόνα 4.3.8 :</u>

Στην παραπάνω εικόνα απεικονίζονται οι παράμετροι του γενετικού αλγορίθμου. Επιλέγουμε τις τιμές των παραμέτρων του αλγορίθμου, είτε σύροντας το βελάκι , είτε πληκτρολογώντας τες. Επίσης , επιλέγουμε τον τύπο της μετάλλαξης , τσεκάροντας μία από τις δυνατές συναρτήσεις μετάλλαξης που φαίνονται παραπάνω . Ενώ τέλος , για να αποθηκευτούν οι επιλογές που έγιναν , πατάμε το πλήκτρο Apply και έπειτα το πλήκτρο OK , σε αντίθετη περίπτωση πατάμε το πλήκτρο Cancel.

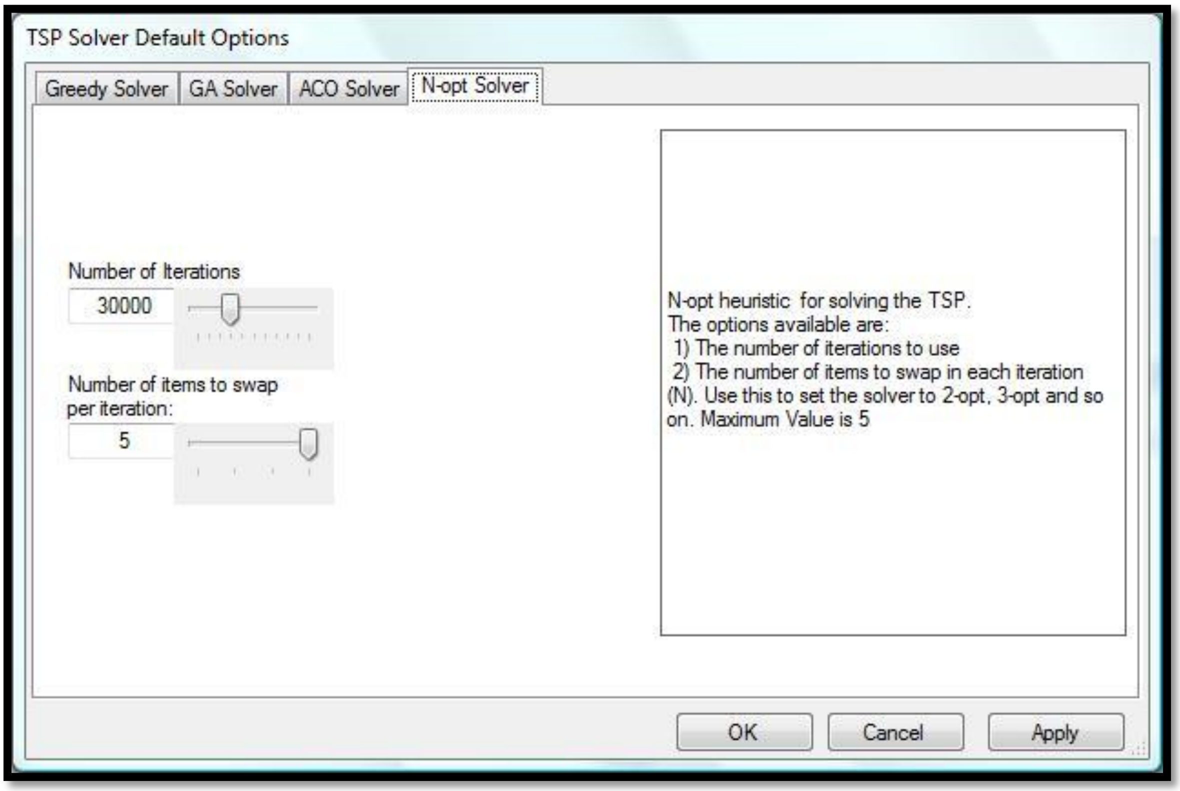
Για επιλογή ACO (αλγορίθμου βελτιστοποίησης μυρμηγκιών) εμφανίζεται η εικόνα :



<u>Εικόνα 4.3.9 :</u>

Στην παραπάνω εικόνα απεικονίζονται οι παράμετροι του αλγορίθμου βελτιστοποίησης μυρμηγκιών . Επιλέγουμε τις τιμές των παραμέτρων του αλγορίθμου, είτε σύροντας το βελάκι , είτε πληκτρολογώντας τες . Ενώ τέλος , για να αποθηκευτούν οι επιλογές που έγιναν , πατάμε το πλήκτρο Apply και έπειτα το πλήκτρο OK , σε αντίθετη περίπτωση πατάμε το πλήκτρο Cancel.

Και για επιλογή N_opt (N_opt αλγορίθμου) εμφανίζεται η εικόνα :

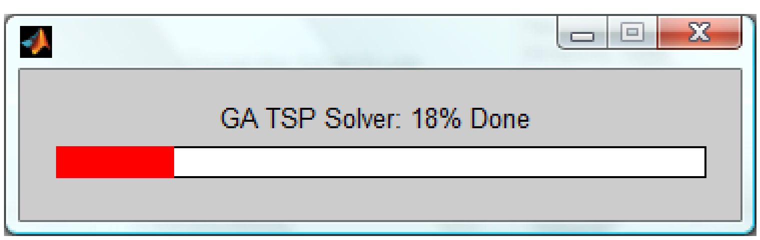


<u>Εικόνα 4.3.10 :</u>

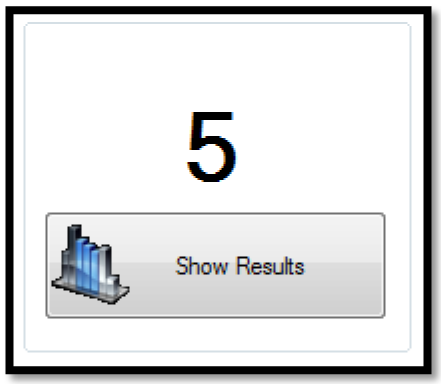
Στην παραπάνω εικόνα απεικονίζονται οι παράμετροι N_opt αλγορίθμου. Επιλέγουμε τις τιμές των παραμέτρων του αλγορίθμου, είτε σύροντας το βελάκι , είτε πληκτρολογώντας τες . Ενώ τέλος , για να αποθηκευτούν οι επιλογές που έγιναν , πατάμε το πλήκτρο Apply και έπειτα το πλήκτρο OK , σε αντίθετη περίπτωση πατάμε το πλήκτρο Cancel.

<u>Εικόνα 4.3.11:</u>

Στην παραπάνω εικόνα απεικονίζεται το $4^{ο}$ Βήμα που ακολουθούμε , πατώντας το πλήκτρο , ξεκινάει η εκτέλεση του αλγορίθμου , που έχει επιλεγεί, ενώ παράλληλα εμφανίζεται η μπάρα εξέλιξης σε ποσοστό επί της εκατό , στον αλγόριθμο που έχουμε επιλέξει , για παράδειγμα :



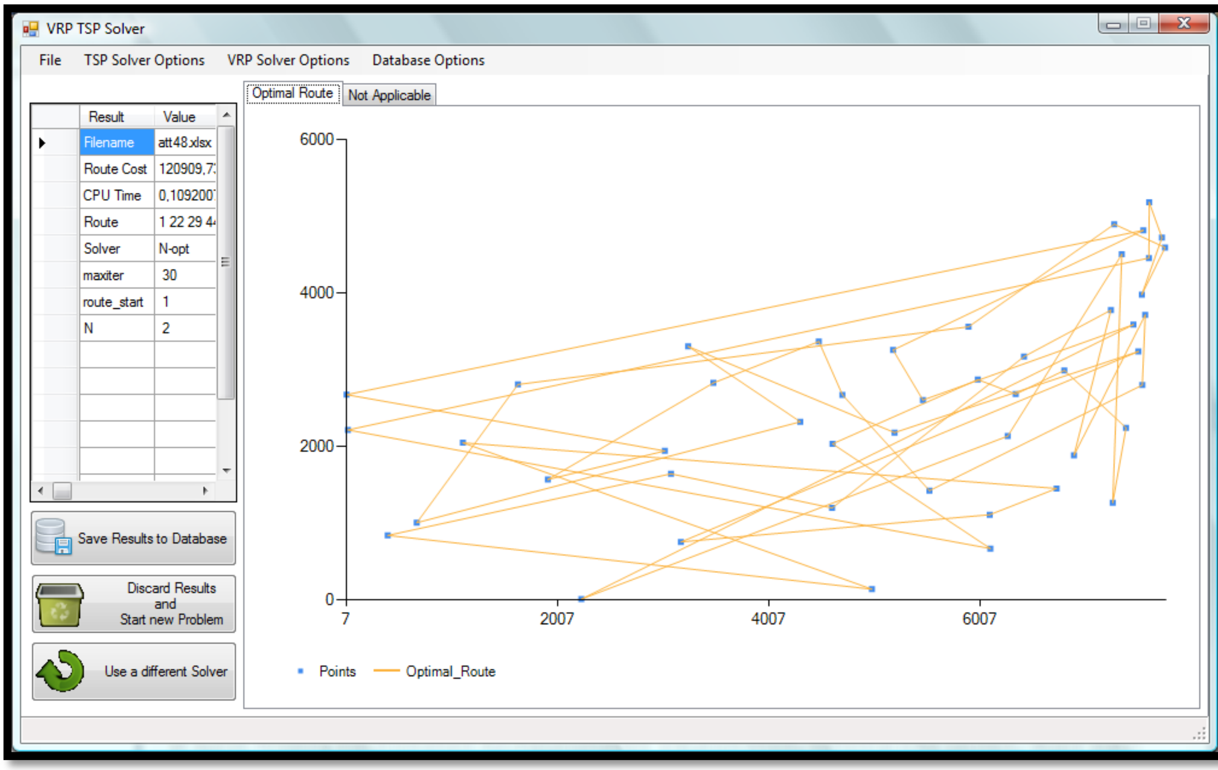
<u>Εικόνα 4.3.12:</u>



<u>Εικόνα 4.3.13:</u>

Στην παραπάνω εικόνα απεικονίζεται το 5^{o} Βήμα , και τελευταίο . Πατώντας το πλήκτρο (*το οποίο ενεργοποιείται αφού τελειώσει η εκτέλεση του αλγορίθμου)* **, εμφανίζονται τα αποτελέσματα του αλγορίθμου , που έχει επιλεγεί.**

Για όλους τους αλγορίθμους , εμφανίζεται η ακόλουθη εικόνα :

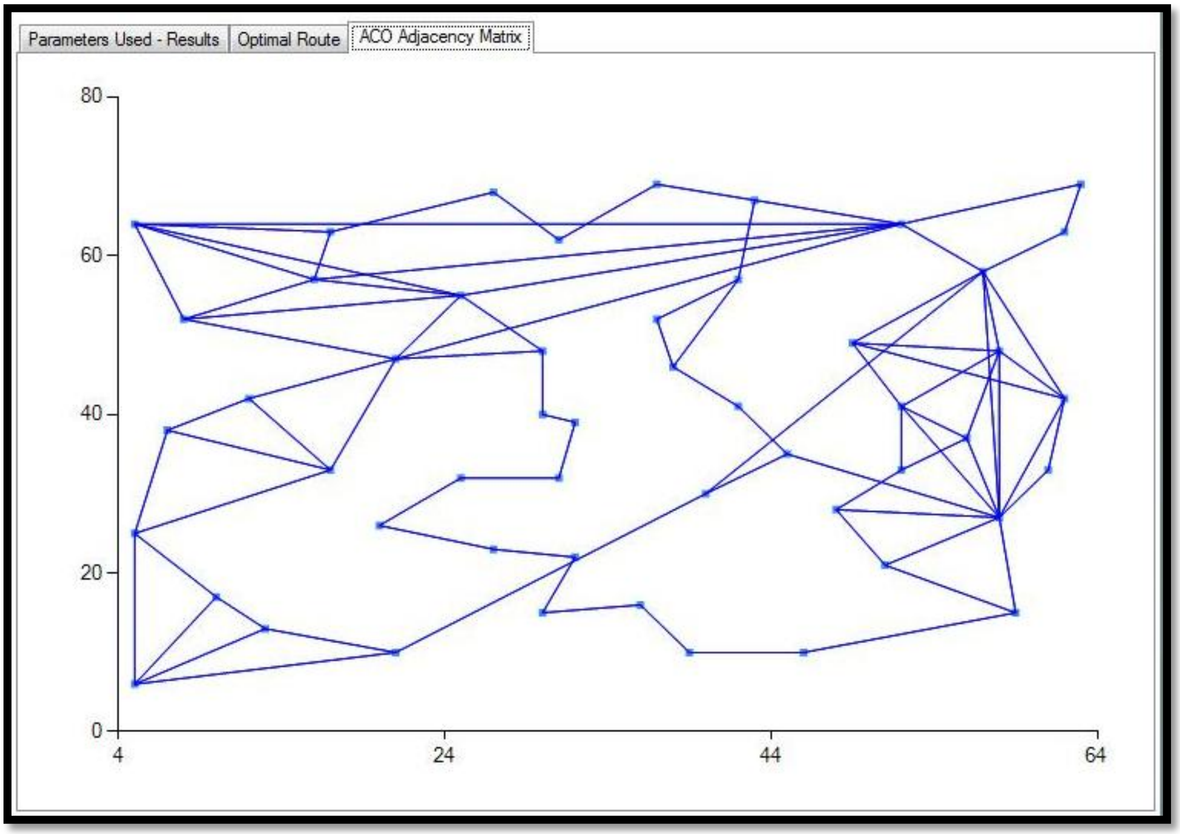


Εικόνα 4.3.14:

Στην παραπάνω εικόνα απεικονίζεται η τελική διαδρομή σχεδιασμένη στο καρτεσιανό επίπεδο, ενώ στο αριστερό παράθυρο της εικόνας , απεικονίζεται , το όνομα του αρχείου που εκτελέστηκε , η τελική διαδρομή με την καταγραφή των κόμβων (αριθμημένων), το κόστος της συνολικής διαδρομής , ο χρόνος εκτέλεσης του προβλήματος και τέλος οι παράμετροι που έχει ο κάθε αλγόριθμος (αν έχει) . Πατώντας το πλήκτρο Save Results to Database ,αποθηκεύουμε με αυτό τον τρόπο τα αποτελέσματα στην βάση δεδομένων , με σκοπό να τα καλέσουμε ξανά και να τα επεξεργαστούμε. Πατώντας το πλήκτρο Discard Results and start new problem , αδιαφορούμε για τα αποτελέσματα και ξεκινάμε εκτέλεση νέου προβλήματος . Τέλος πατώντας το πλήκτρο

Use a different solver , χρησιμοποιούμε για το ίδιο πρόβλημα , άλλο αλγόριθμο , χωρίς να αποθηκευθούν τα τελευταία αποτελέσματα.

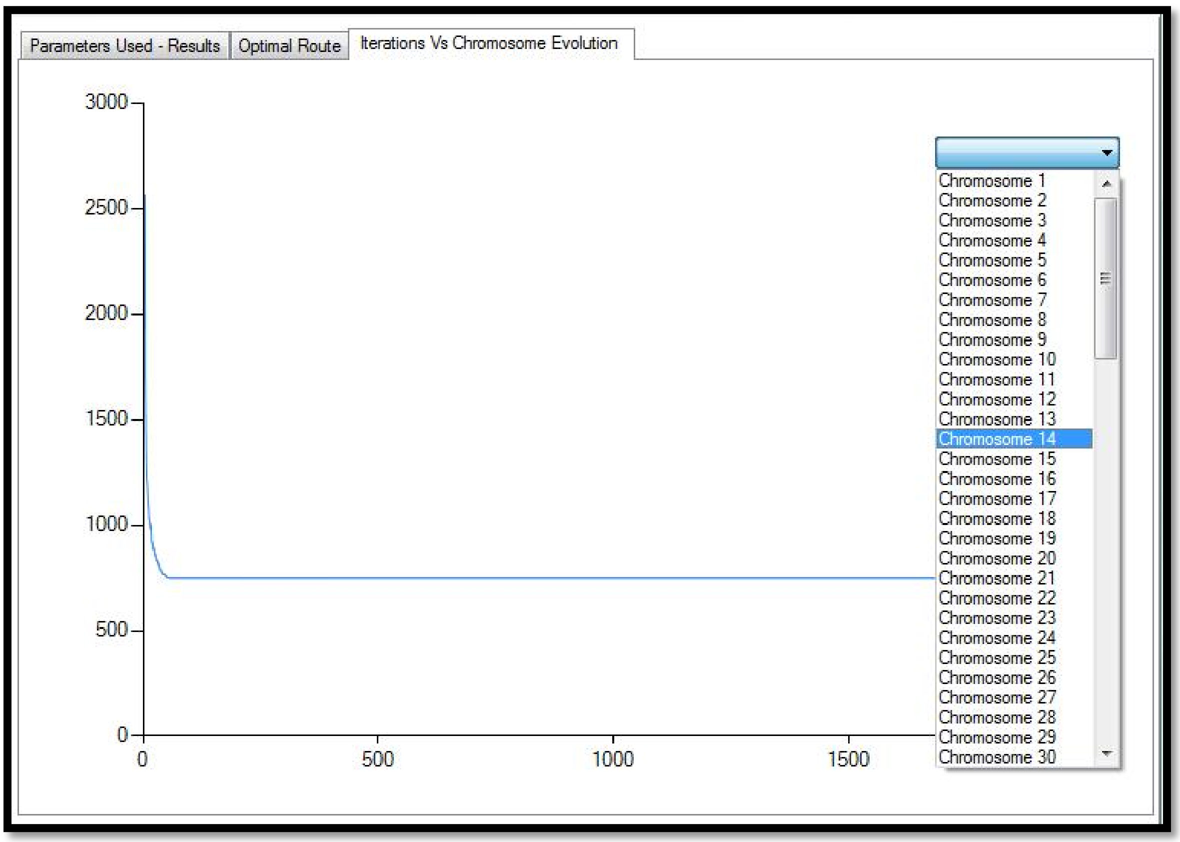
Ειδικά για τον αλγόριθμο του ACO , εμφανίζεται επιπλέον , η ακόλουθη εικόνα :



<u>Εικόνα 4.3.15:</u>

Στην παραπάνω εικόνα απεικονίζεται ο πίνακας γειτνίασης , ο οποίος , εμφανίζεται σαν επιπλέον παράθυρο μόνο στον αλγόριθμο του ACO , ενώ για τους άλλους αλγορίθμους δεν εμφανίζεται .

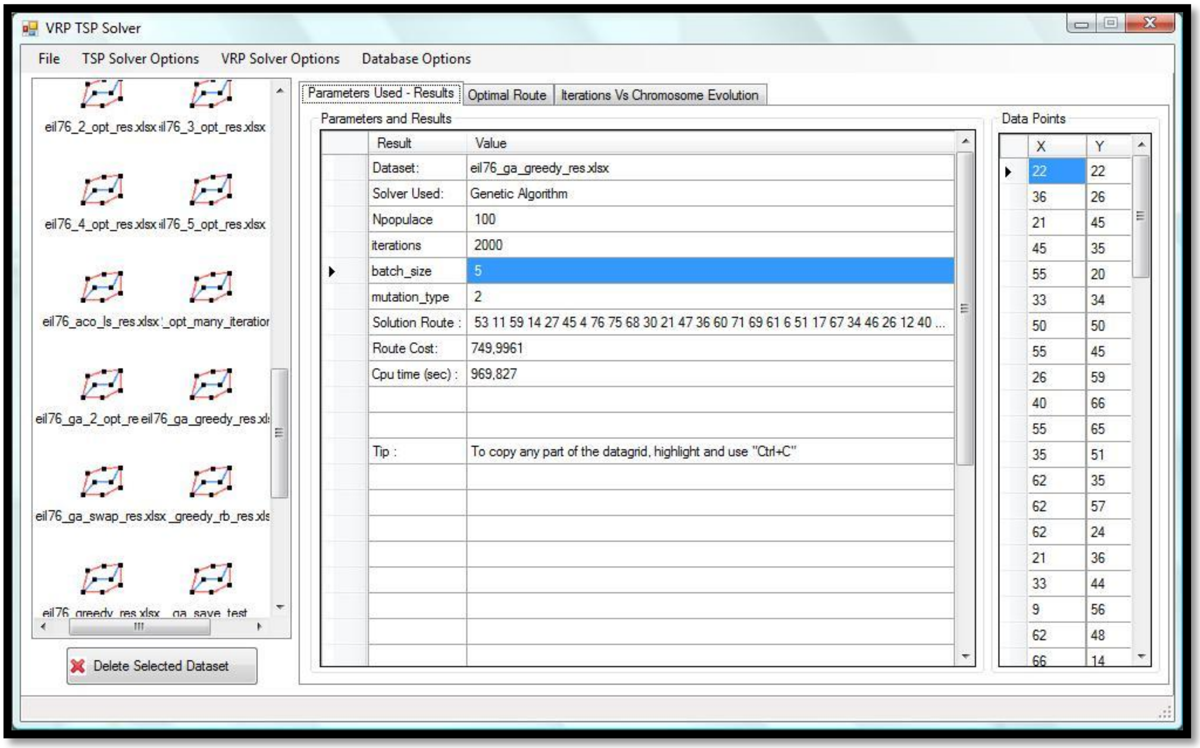
Ειδικά , επίσης για τον αλγόριθμο τον γενετικό , εμφανίζεται επιπλέον , η ακόλουθη εικόνα :



Εικόνα 4.3.16:

Στην παραπάνω εικόνα απεικονίζεται ο πίνακας του κόστους κάθε χρωμοσώματος , σε κάθε επανάληψη , ο οποίος , εμφανίζεται σαν επιπλέον παράθυρο μόνο στον αλγόριθμο τον γενετικό , ενώ για τους άλλους αλγορίθμους δεν εμφανίζεται . Ενώ , παράλληλα στην δεξιά μπάρα , μπορούν να επιλεγούν οι αναπαραστάσεις και άλλων χρωμοσωμάτων (τα οποία είναι αριθμημένα) .

Τέλος , επιλέγοντας από την κύρια γραμμή εργαλείων , από το μενού Database Options , την επιλογή Manage TSP Results , εμφανίζεται η ακόλουθη εικόνα :



Εικόνα 4.3.17:

Στην παραπάνω εικόνα απεικονίζονται στα αριστερά της , όλα τα αποθηκευμένα στην βάση , αποτελέσματα που έχουμε εκτελέσει . Επιλέγοντας, ένα , όπως φαίνεται και παραπάνω , εμφανίζονται ξανά όλες οι πληροφορίες του προβλήματος που επιλέξαμε . Ενώ τέλος , αν θέλουμε μπορούμε επιλέγοντας πρώτα ένα αποτέλεσμα , να πατήσουμε έπειτα το πλήκτρο Delete Selected Dataset και να διαγράψουμε το επιλεγμένο αποτέλεσμα .

Κεφάλαιο 5:

Αποτελέσματα

Παρακάτω , φαίνονται τα αποτελέσματα σε 10 επιλεγμένα παραδείγματα και οι αποκλίσεις τους από τα βέλτιστα :

Σημείωση : Όλα , τα παραδείγματα , εκτελούνται με τις ίδιες παραμέτρους που χρησιμοποιήσαμε στην περιγραφή του παραδείγματος **Eil_51** , λόγω του ότι τα αποτέλεσματα που προκύπτουν , είναι ενδεικτικά , των όσων ζητήθηκαν στα πλαίσια της διπλωματικής (*μέχρι +10%*).

Για αλγορίθμους απληστίας (**Greedy**) :

Δεδομένα	**Standard**		Ω(%)	**Round Robin**		Ω(%)	**Βέλτιστο Κόστος**
	CPU(sec)	Κόστος		CPU(sec)	Κόστος		
Eil_51	0,013	563,01	**+22,86%**	0.234	505,77	**+18,45%**	426
Eil_76	0,093	649,54	**+20,63%**	0,499	612,65	**+13,52%**	538
Pr_76	0,015	149121,30	**+37,82%**	0,405	130921,	**+21,04%**	108159
KroB_100	0,015	28585,88	**+34,31%**	0,686	25882,97	**+21,61%**	21282
KroA_100	0,001	27940,33	**+26,19%**	0,717	24698,5	**+11,54%**	22141
Berlin_52	0,001	9632,14	**+27,71%**	0,171	8182,19	**+8.48%**	7542
Rd_100	0,015	10920,93	**+38,05%**	0,717	9427,33	**+19,17%**	7910
Eil_101	0,015	768,20	**+22,09%**	0,733	736,36	**+17,01%**	629
Ch_130	0,015	7581,05	**+24,07%**	1,138	7198,74	**+17,80%**	6110
Ch_150	0,015	7782,47	**+19,20%**	1,747	7078,44	**+8,42%**	6528
A_280	0,062	3221,55	**+24,89%**	6,801	3094,27	**+19,96%**	2579

Πίνακας 5.1: Ο τελικός πίνακας των αποτελεσμάτων των αλγορίθμων απληστίας.

Στο παραπάνω πίνακα, φαίνονται τα αποτελέσματα των αλγορίθμων απληστίας. Όπως παρατηρούμε , σε κάθε περίπτωση , ο αλγόριθμος απληστίας με την στρατηγική του Round Robin , βγάζει καλύτερα αποτελέσματα , πιο κοντά δηλαδή στο βέλτιστο . Ωστόσο και πάλι , η απόκλισή του από το βέλτιστο είναι μεγαλύτερη του *10%*.

Για αλγόριθμους **N_opt** :

Δεδομένα	**2_opt** CPU	Κόστος	Ω(%)	**3_opt** CPU	Κόστος	Ω(%)	**4_opt** CPU	Κόστος	Ω(%)	**Βέλτιστο Κόστος**
Eil_51	0,093	523,40	**+32,15%**	58,62	657,37	**+54,31%**	60,98	705,62	**+65,62%**	426
Eil_76	7,532	827,89	**+53,17%**	7,83	1217,55	**+126,2%**	18,95	1204,44	**+120,3%**	538
Pr_76	8,361	212648,93	**+86,60%**	7,129	242437,18	**+124,14%**	14,258	265204,25	**+145,19%**	108159
KroB_100	9,032	70354,59	**+197,75%**	9,032	68997,02	**+196,52%**	37,471	77220,55	**+228,76%**	22141
KroA_100	27,814	46057,15	**+116,41%**	28,371	59644,68	**+170,25%**	56,597	69868,18	**+228,29%**	21282
Berlin_52	5,974	11432,29	**+51,57%**	5,990	13623,33	**+80,62%**	7,332	13852,28	**+83,66%**	7542
Rd_100	8,502	19708,75	**+135,47%**	28,657	20558,53	**+159,89%**	57,033	24242,32	**+204,47%**	7910
Eil_101	28,236	1074,13	**+70,74%**	28,782	1301,96	**+106,83%**	58,391	1577,03	**+150,71%**	629
Ch_130	36,176	14393,34	**+135,56%**	37,362	17601,02	**+188,06%**	151,055	20818,72	**+240,72%**	6110
A_280	111,243	10742,31	**+316,51%**	154,518	13791,14	**+434,74%**	256,057	14875,36	**+476,32%**	2579

Πίνακας 5.2: Ο τελικός πίνακας των αποτελεσμάτων των αλγορίθμων N_opt.

Στο παραπάνω πίνακα, φαίνονται τα αποτελέσματα των αλγορίθμων N_opt. Όπως παρατηρούμε , σε κάθε περίπτωση , ο αλγόριθμος για N = 2 , βγάζει καλύτερα αποτελέσματα , αν και είναι πολύ μακριά από το βέλτιστο . Ωστόσο και πάλι , η απόκλισή του από το βέλτιστο είναι μεγαλύτερη του *10%*. Στα παραδείγματα μετά και το Ch_130 χρησιμοποιήσαμε 10000 επαναλήψεις , λόγω του ότι έχουν περισσότερα σημεία .

Για αλγόριθμο **G_A** :

Δεδομένα	GA(τύπος1) CPU	Κόστος	Ω (%) +	GA(τύπος2) CPU	Κόστος	Ω (%) +	GA(τύπος3) CPU	Κόστος	Ω (%) +	GA(τύπος4) CPU	Κόστος	Ω (%) +	Βέλτιστο Κόστος
Eil_51	10,60	674,33	**58,21%**	416,281	503,66	**18,07%**	4074,82	477,63	**11,97%**	5253,27	457,05	**7,27%**	426
Eil_76	8,923	1045	**94,22%**	692,531	687,43	**27,69%**	440,68	586,50	**8,92%**	9016,2	572,63	**6,31%**	538
Pr_76	15,054	225515,195	**108,22%**	892,23	147713	**36,52%**	397,53	128656	**18,59%**	9415,32	121590	**12,41%**	108159
KroB_100	14,383	60239,67	**172,06%**	1322,38	35127,25	**58,65%**	523,55	26248,72	**18,54%**	30629,22	23657	**6,44%**	22141
KroA_100	13,868	57760,19	**171,14%**	1283,75	36308,52	**70,6%**	528,96	24845,41	**16,74%**	30570	22726	**6,78%**	21282
Berlin_52	13,260	10909,22	**44,06%**	556,73	9488,96	**25,8%**	291,15	7826,69	**3,76%**	5419,00	7868,29	**4,23%**	7542
Rd_100	15,506	22079,42	**179,28%**	10407,59	11468,45	**44,98%**	592,86	9017,45	**13,99%**	30782,12	9081,21	**14,62%**	7910
Eil_101	15,568	1437,32	**128,45%**	1300,67	877,14	**39,42%**	603,77	733,15	**16,53%**	48057,21	712,14	**13,19%**	629
Ch_130	46,921	18231,65	**198,37%**	78	17585,15	**187,82%**	798,42	6933,22	**13,46%**	50207,47	6621,15	**8,61%**	6110
A_280	18,379	13432,14	**420,82%**	46636,82	4909,48	**90,34%**	2424,72	3177,65	**18,58%**	69122,13	2918,12	**12,14%**	2579
AVERAGE									**14,05%**			**9,17%**	

Πίνακας 5.3: Ο τελικός πίνακας των αποτελεσμάτων των αλγορίθμων N_opt.

Όπως παρατηρούμε , σε κάθε περίπτωση , ο γενετικός αλγόριθμος, που χρησιμοποιεί σαν συνάρτηση μετάλλαξης τον αλγόριθμο απληστίας με Round Robin ,έχει μέσο όρο απόκλισης καλύτερο από όλους τους παραπάνω.

Για αλγόριθμο **ACO** :

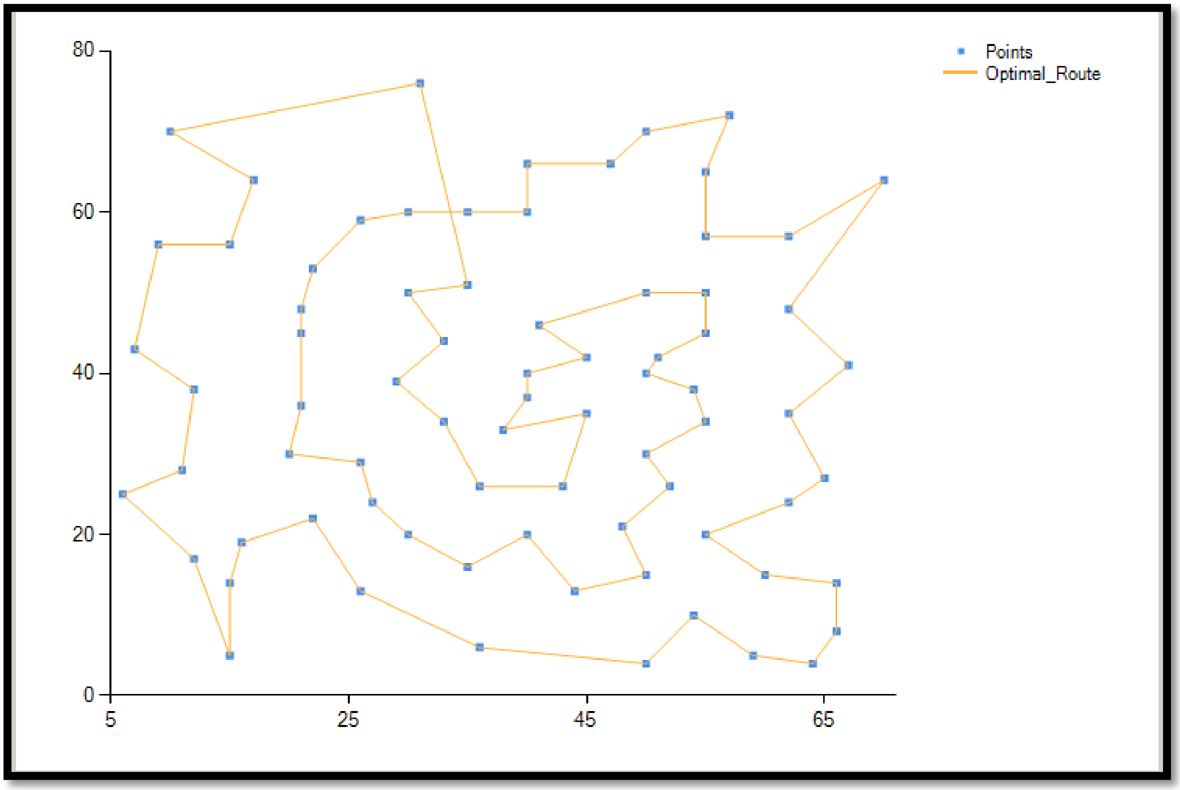
Δεδομένα	ACO_Ls CPU(sec)	Κόστος	Ω(%)	Βέλτιστο Κόστος
Eil_51	3493,72	447,05	**+4,8%**	426
Eil_76	5791,20	563,79	**+4,6%**	538
Pr_76	5709,44	111608,79	**+3,1%**	108159
KroB_100	12497,22	22152,55	**+0,004%**	22141
KroA_100	4177,91	23127,03	**+8,6%**	21282
Berlin_52	1869,11	7867,42	**+4,3%**	7542
Rd_100	19034,76	8745,49	**+9,2%**	7910
Eil_101	19191,67	683,98	**+8,4%**	629
Ch_130	26293,64	6603,25	**+8,04%**	6110
A_280	48117,82	2805,08	**+8,76%**	2579
AVERAGE			**+5,98%**	

Πίνακας 5.4: Ο τελικός πίνακας των αποτελεσμάτων του αλγορίθμου του ACO.

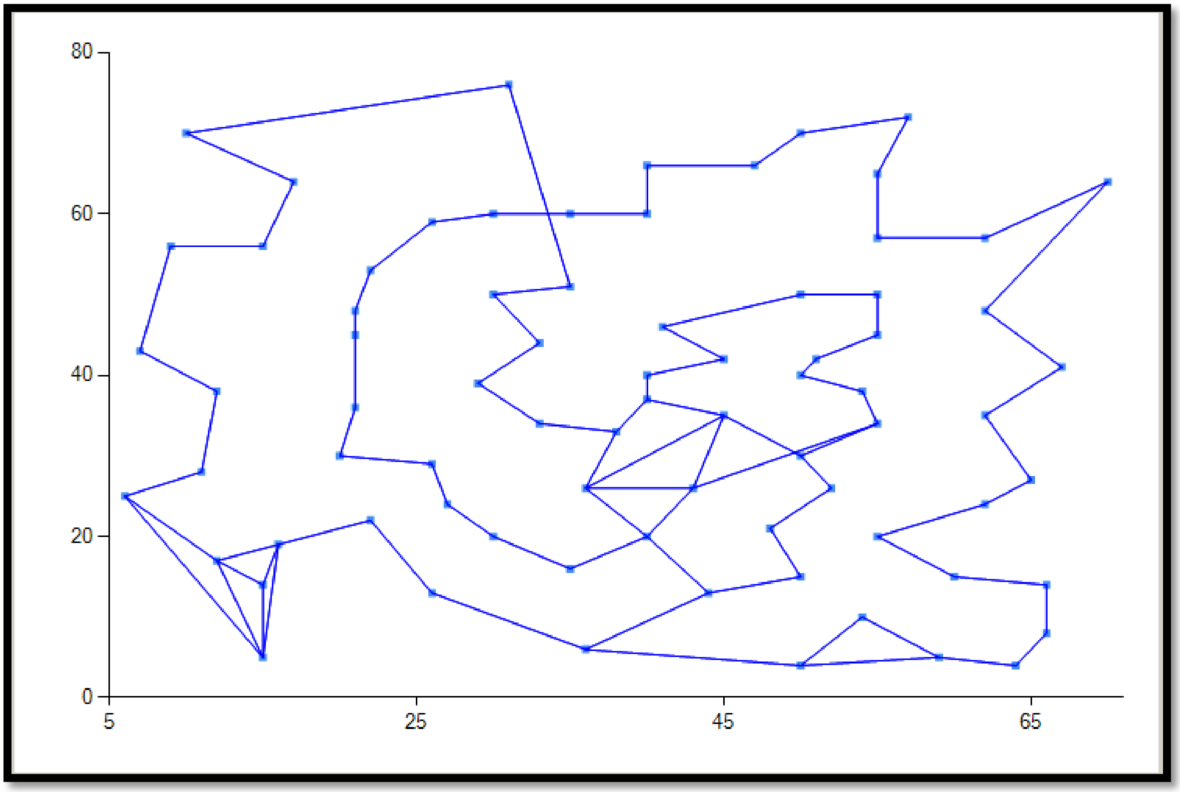
Στο παραπάνω πίνακα, φαίνονται τα αποτελέσματα του αλγορίθμου βελτιστοποίησης μυρμηγκιών. Όπως παρατηρούμε , ο αλγόριθμος των μυρμηγκιών, βγάζει τα καλύτερα αποτελέσματα σε κάθε παράδειγμα , από όλους τους παραπάνω αλγορίθμους. <u>Ενώ , η απόκλισή του , επιτυγχάνει τον σκοπό μας , δηλαδή μέχρι *+10%* από το βέλτιστο</u> . Οι παράμετροι τέλος , που χρησιμοποιούμε σε κάθε παράδειγμα , είναι ίδιοι με τους παραμέτρους που χρησιμοποιήσαμε στο Κεφάλαιο 3 , στα σημεία που περιγράφουμε αναλυτικά , το παράδειγμα Eil_51.

Παρακάτω , παρατίθενται τα σχήματα των βέλτιστων διαδρομών, μαζί με τους πίνακες γειτνίασης , του αλγορίθμου , του ACO , καθώς και του γεγεντικού , ο οποίος απέδωσε τα προσδοκούμενα αποτελέσματα :

Eil_76

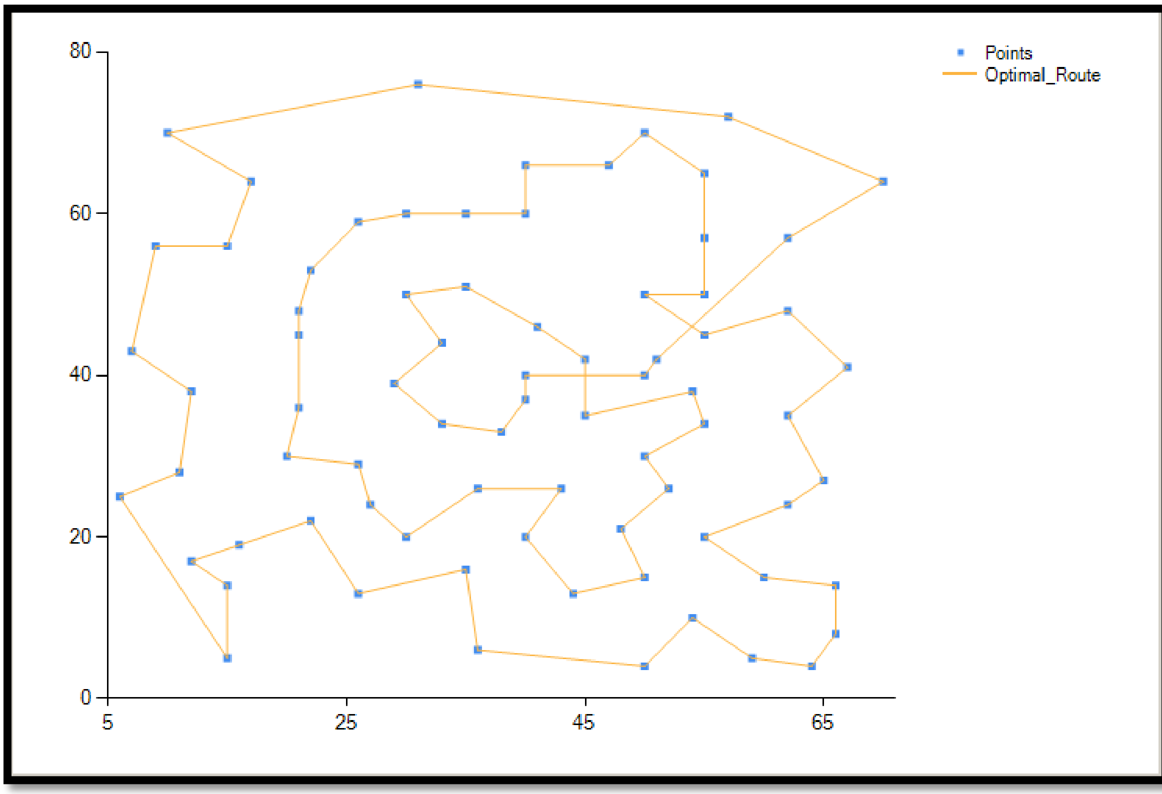


Σχήμα 5.1 : **Η τελική διαδρομή**

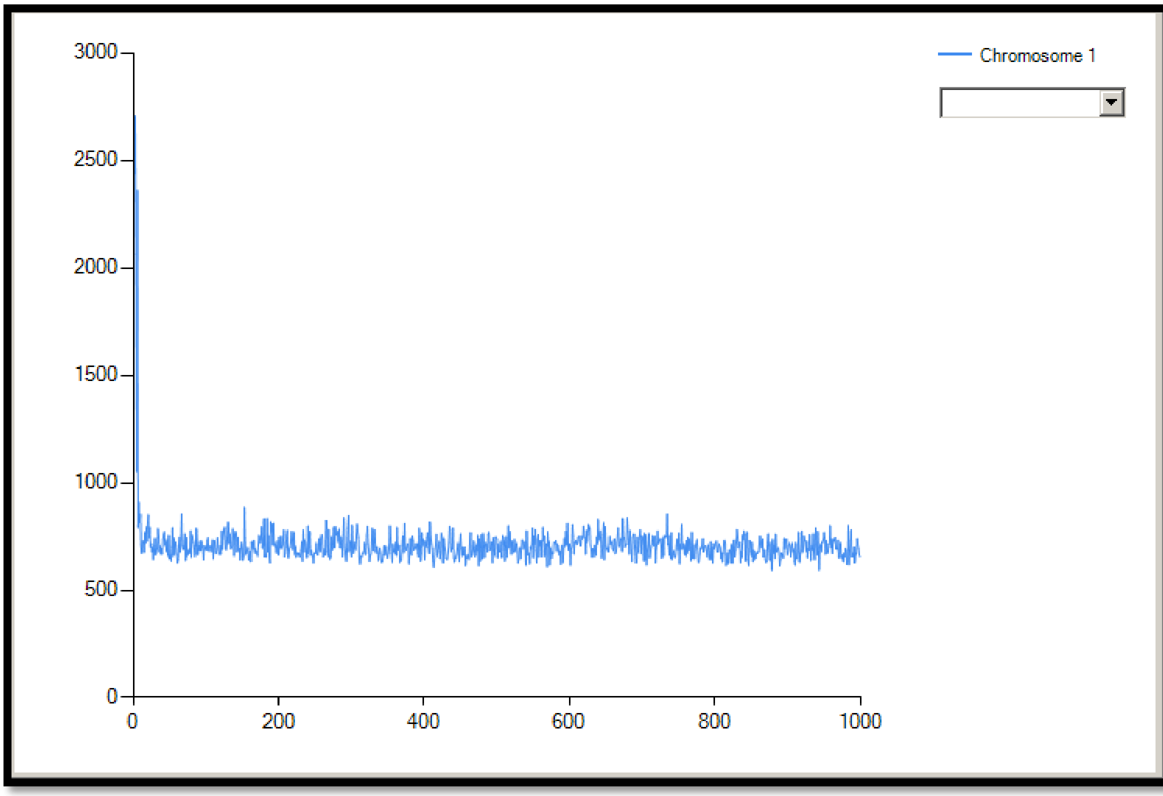


Σχήμα 5.2 : **Ο πίνακας γειτνίασης**

Για τον Γενετικό Αλγόριθμο με μετάλλαξη Greedy Round Robin έχουμε τα ακόλουθα σχήματα :

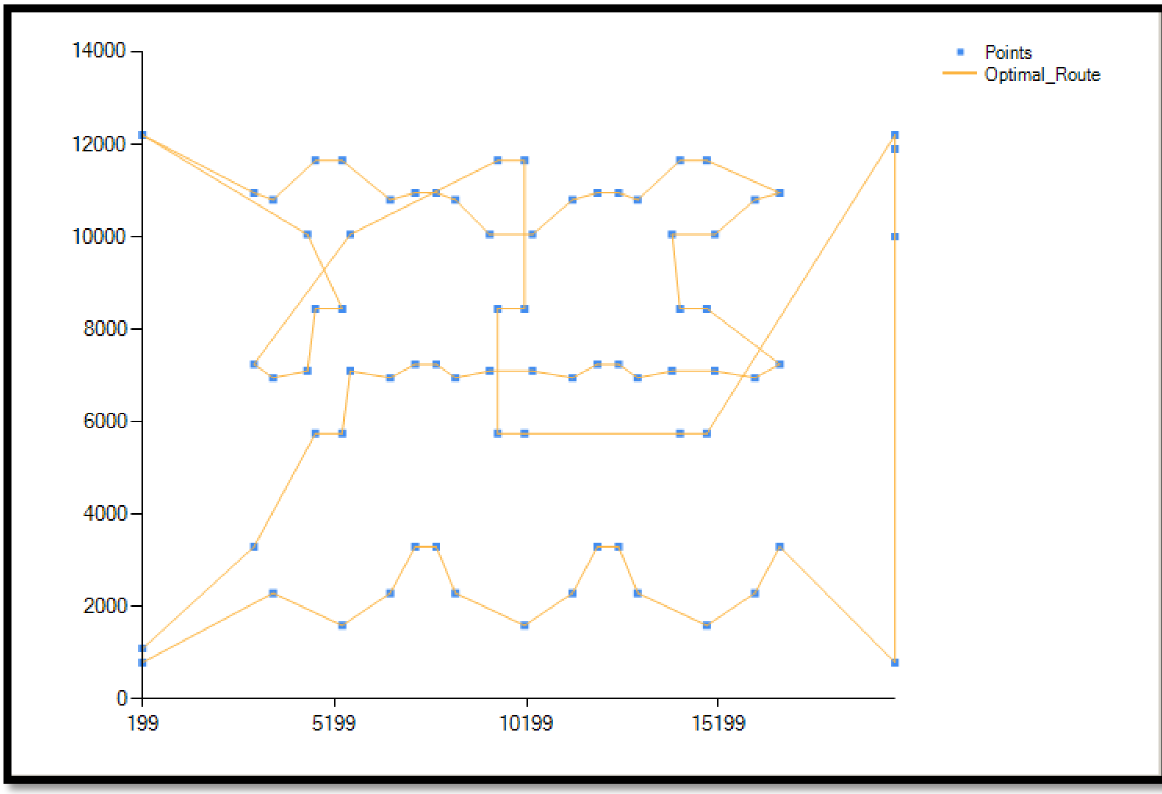


Σχήμα 5.3 : **Η τελική διαδρομή**

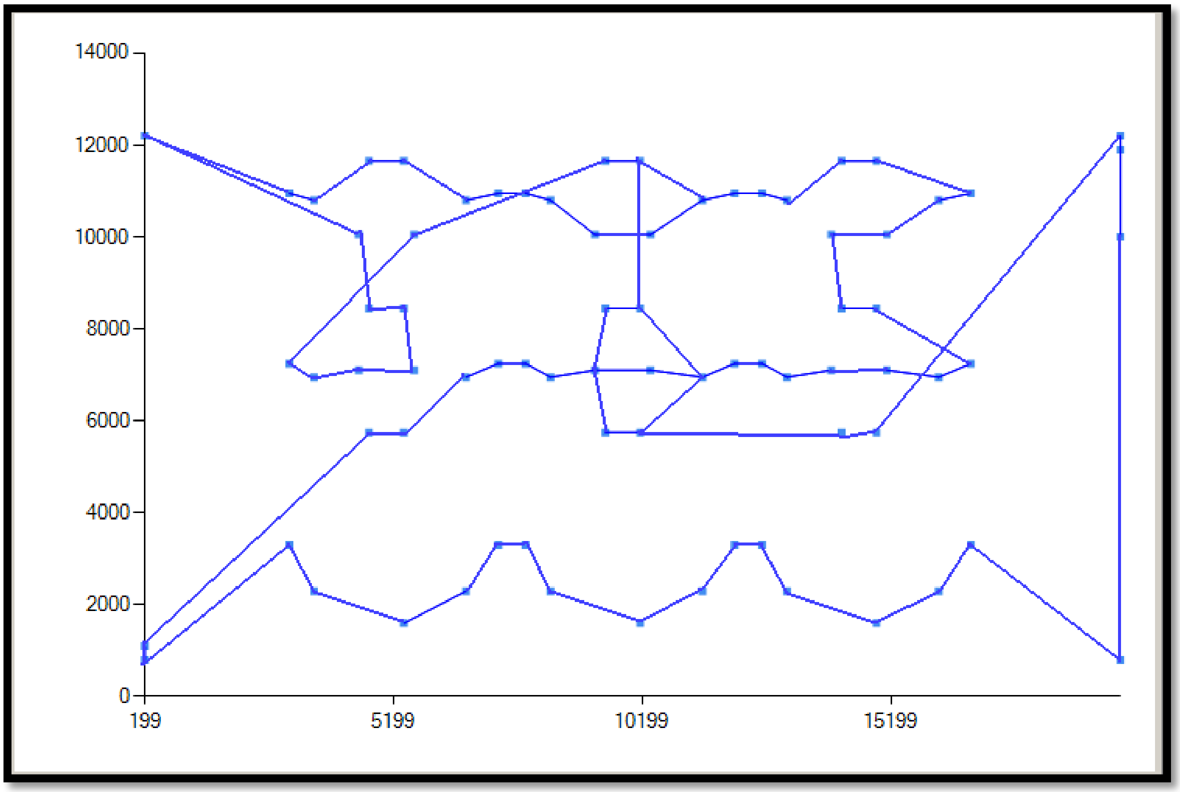


Σχήμα 5.4: **Η εξέλιξη των κόστων κάθε χρωμοσώματος**

Pr_76

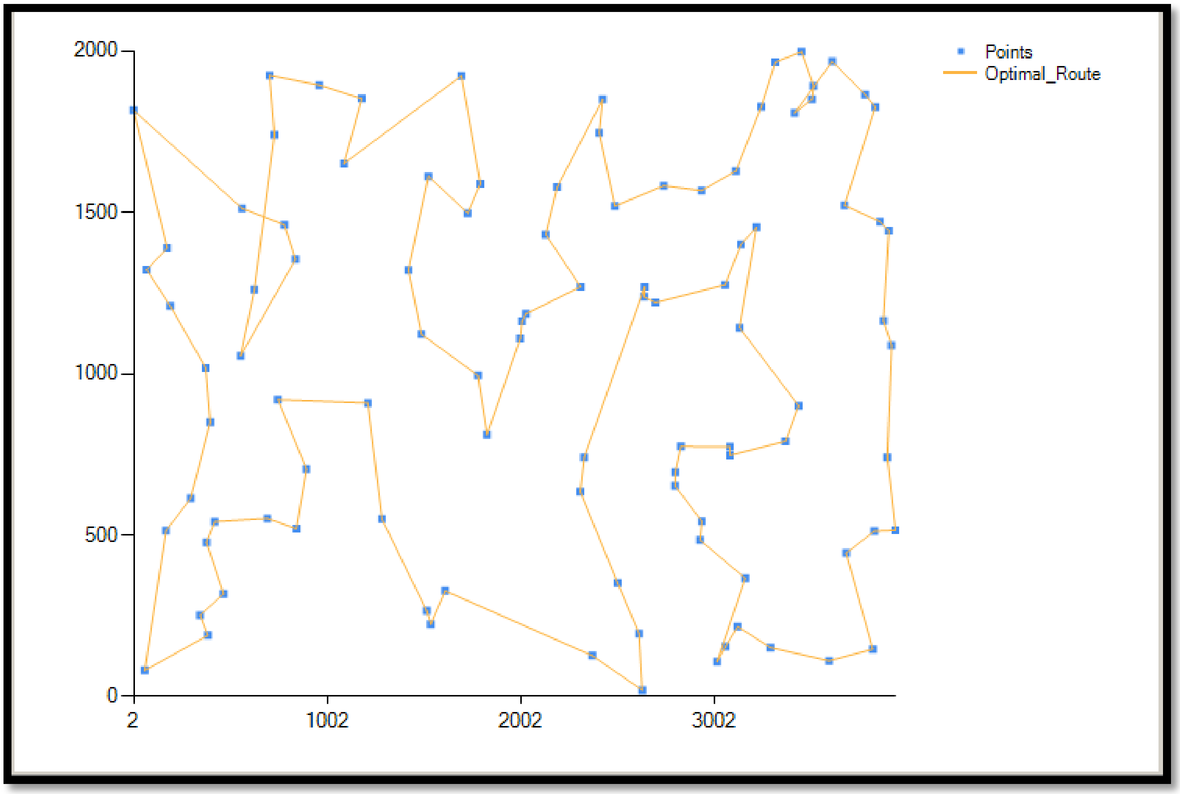


Σχήμα 5.5 : **Η τελική διαδρομή**

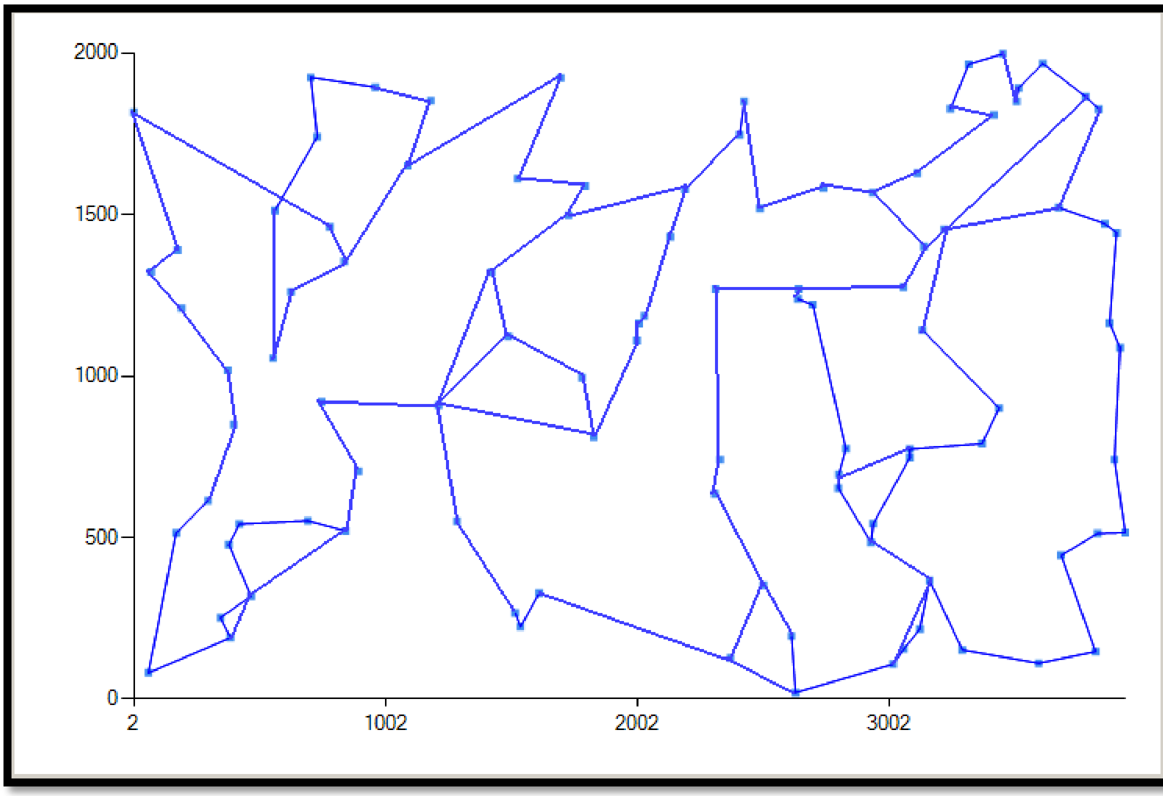


Σχήμα 5.6 : **Ο πίνακας γειτνίασης**

KroB_100

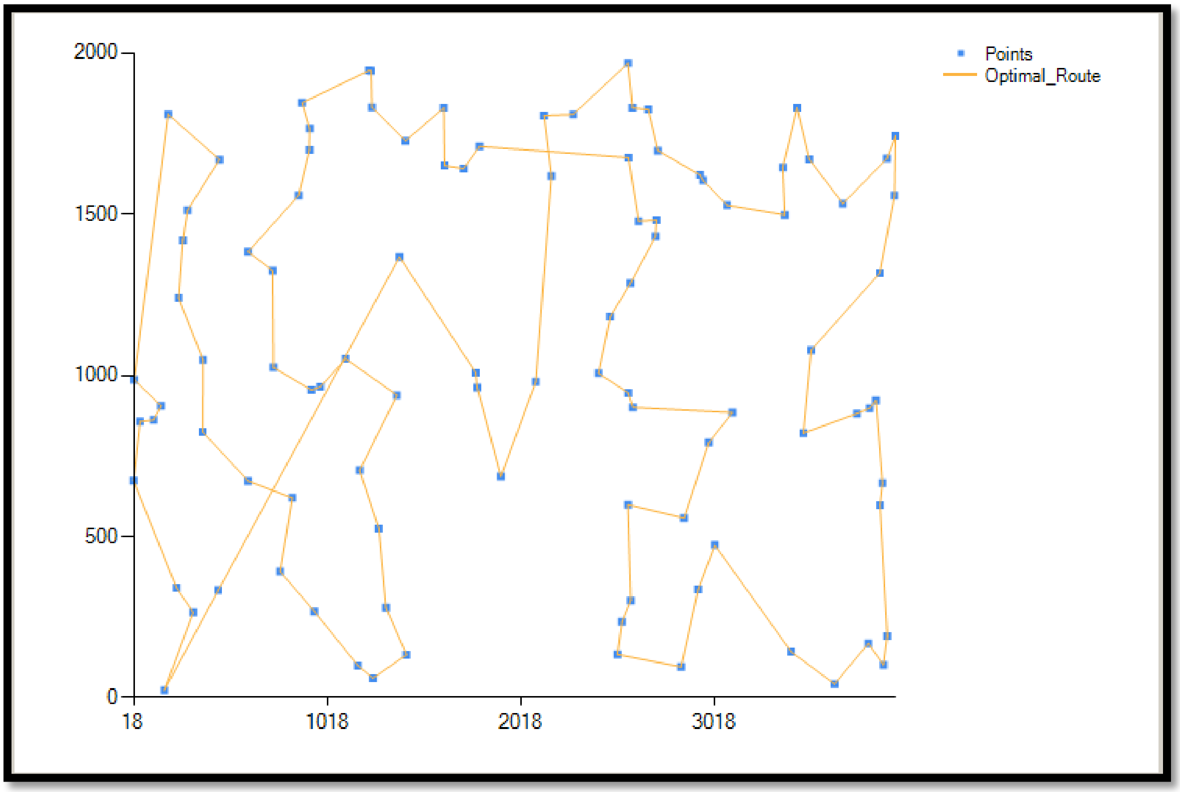


Σχήμα 5.7 : **Η τελική διαδρομή**

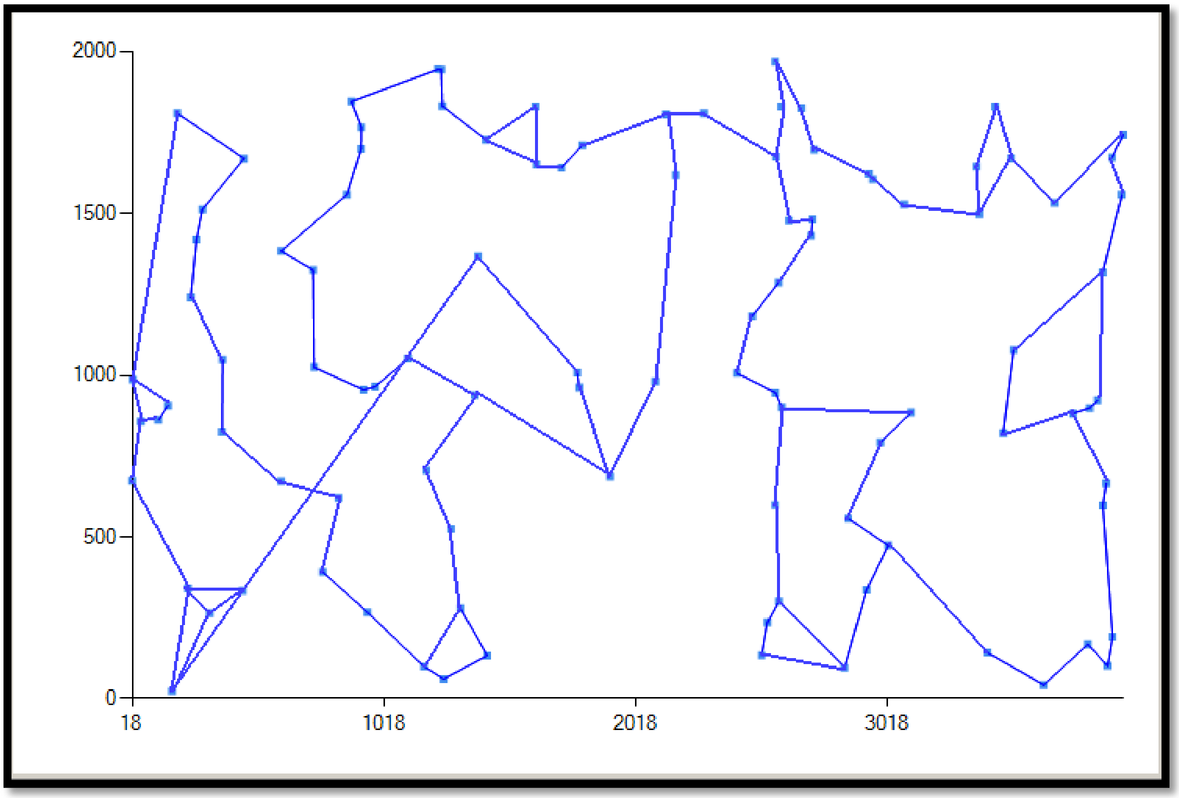


Σχήμα 5.8 : **Ο πίνακας γειτνίασης**

KroA_100

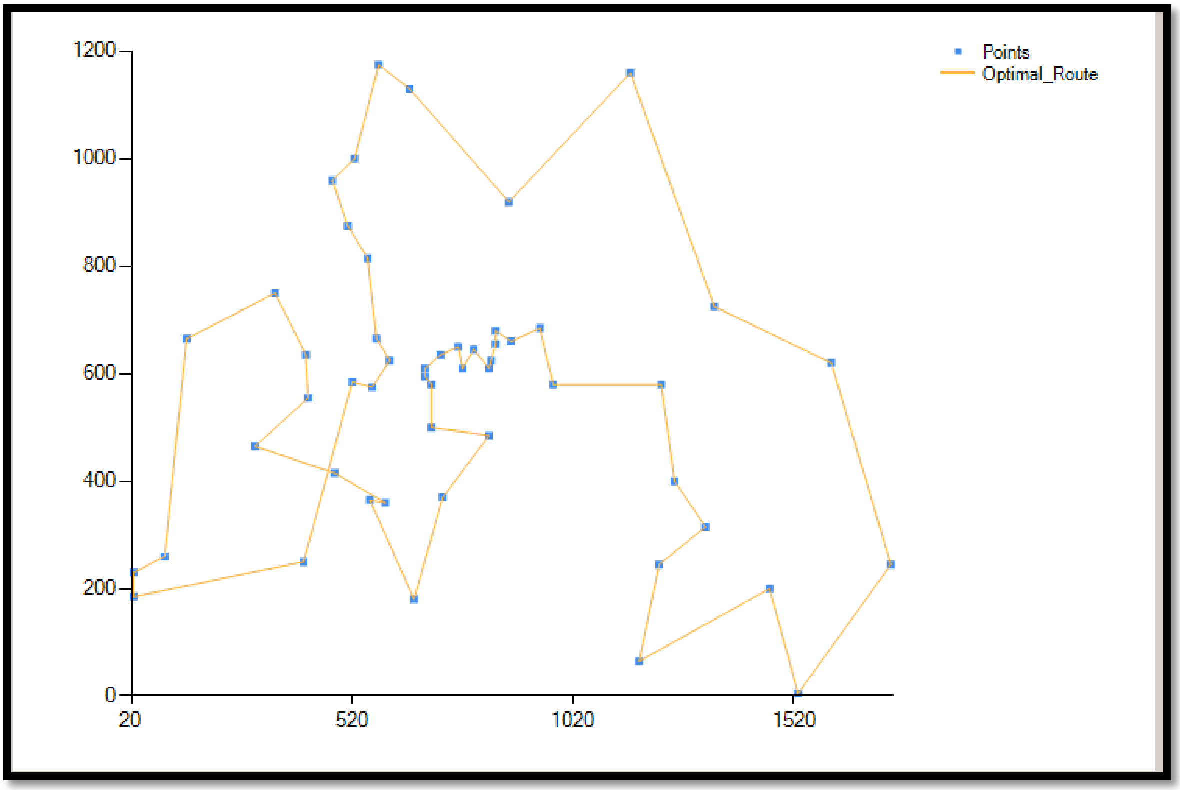


Σχήμα 5.9 : **Η τελική διαδρομή**

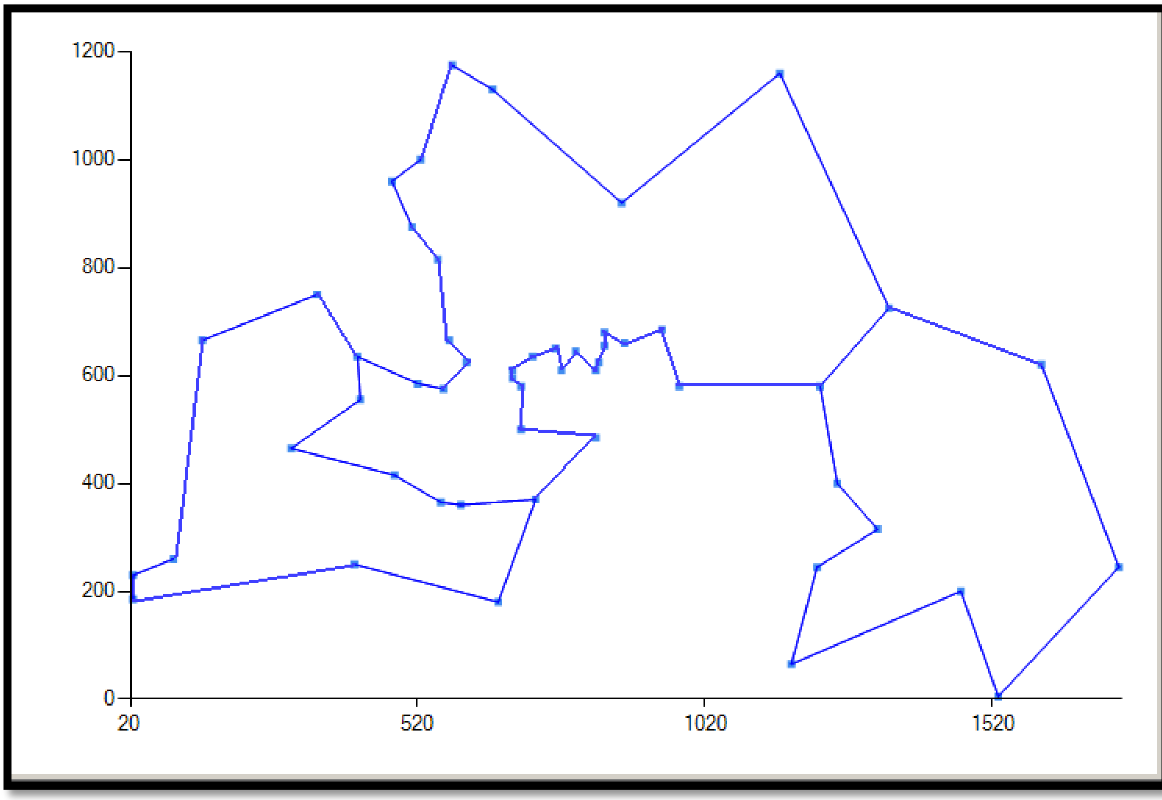


Σχήμα 5.10 : **Ο πίνακας γειτνίασης**

Berlin_52

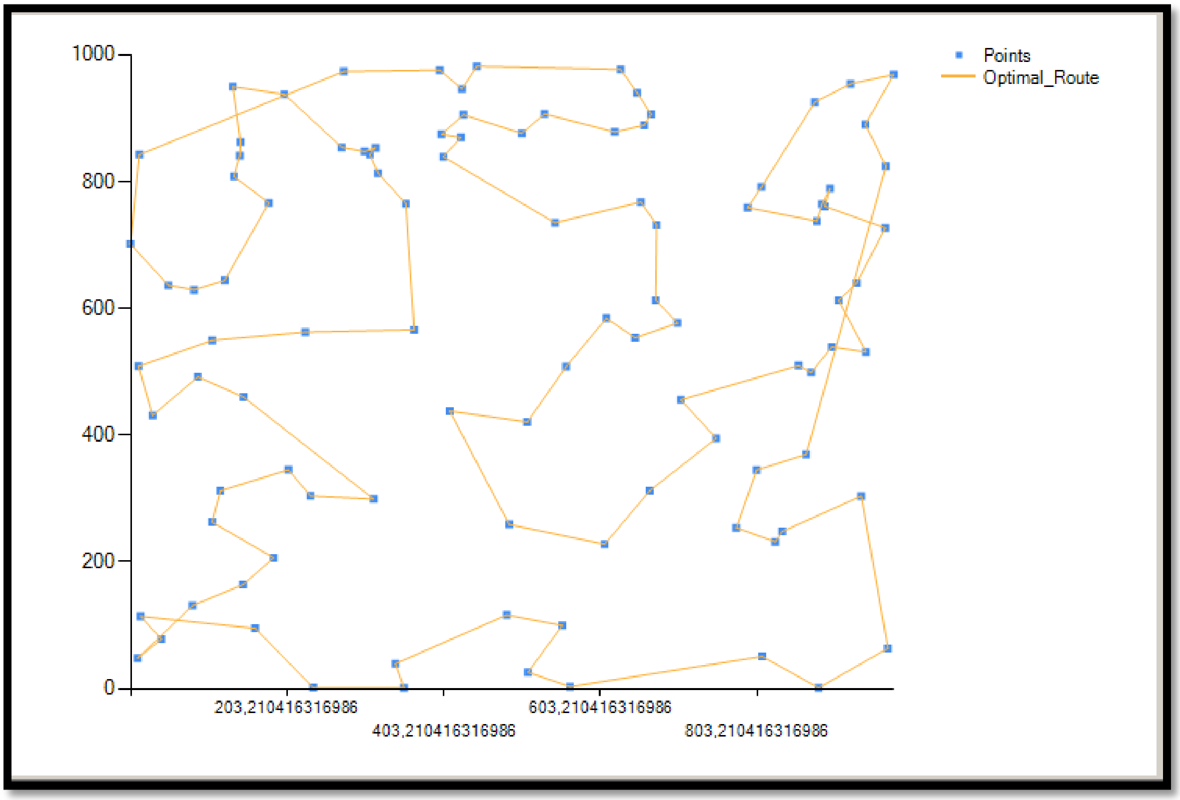


Σχήμα 5.11 : **Η τελική διαδρομή**

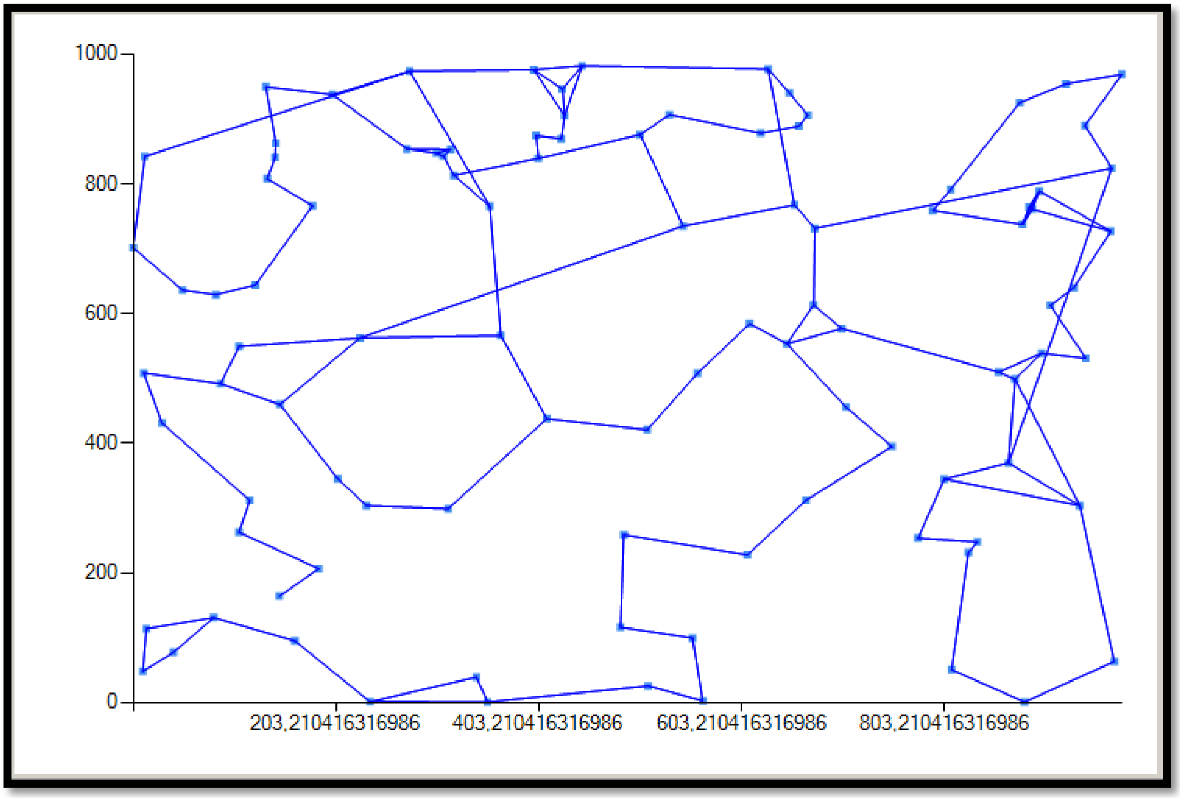


Σχήμα 5.12: **Ο πίνακας γειτνίασης**

Rd_100

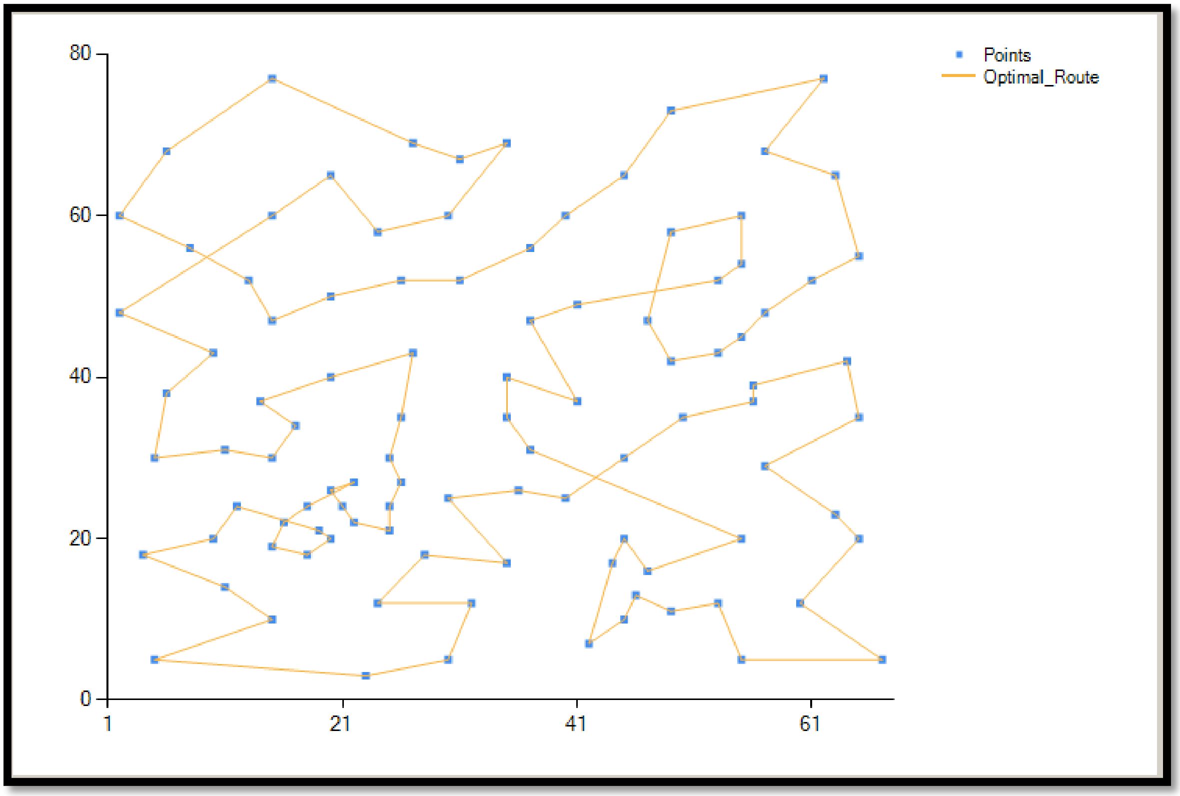


Σχήμα 5.13 : **Η τελική διαδρομή**

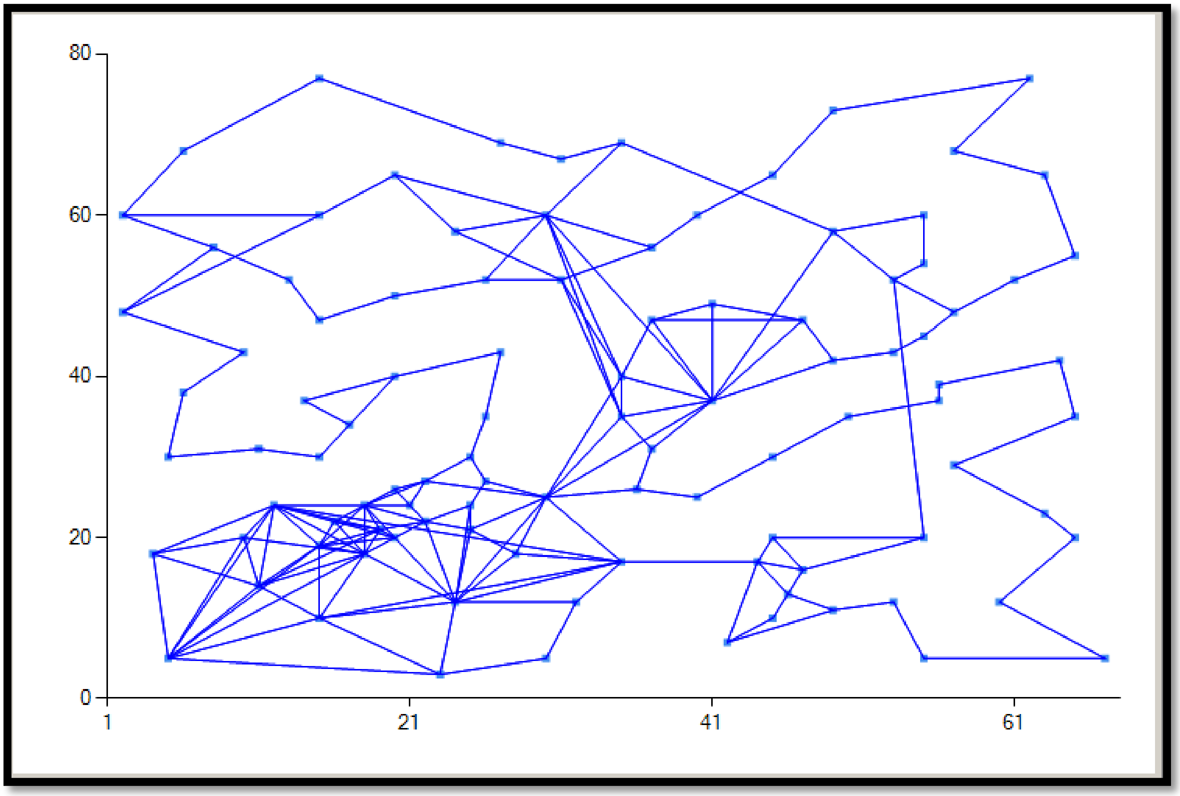


Σχήμα 5.14: **Ο πίνακας γειτνίασης**

Eil_101

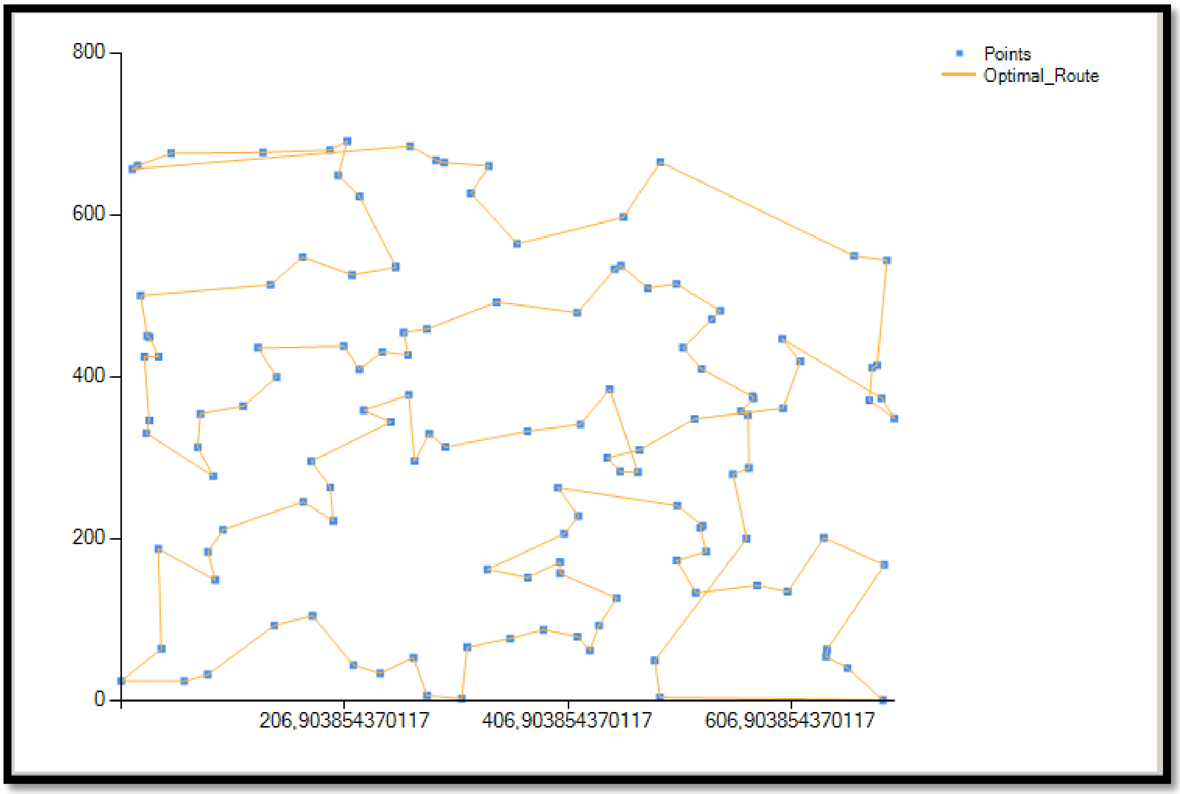


Σχήμα 5.15 : **Η τελική διαδρομή**

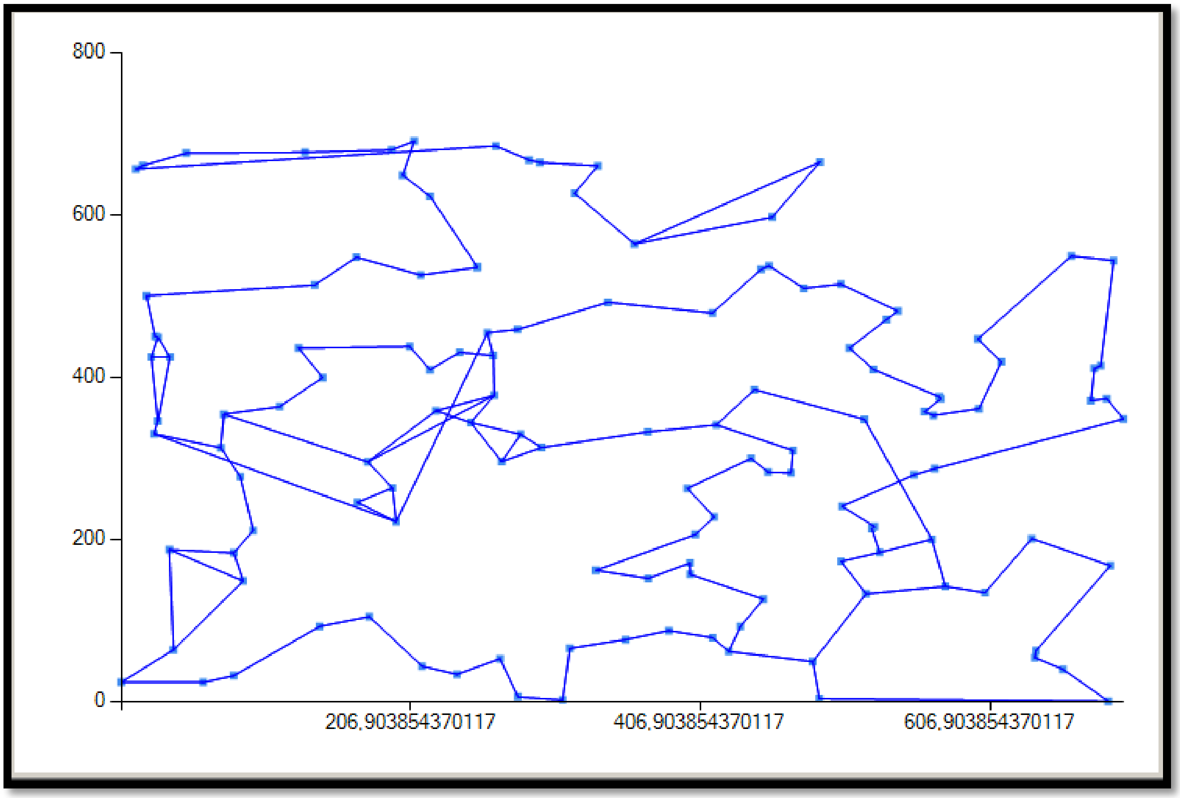


Σχήμα 5.16: **Ο πίνακας γειτνίασης**

Ch_130

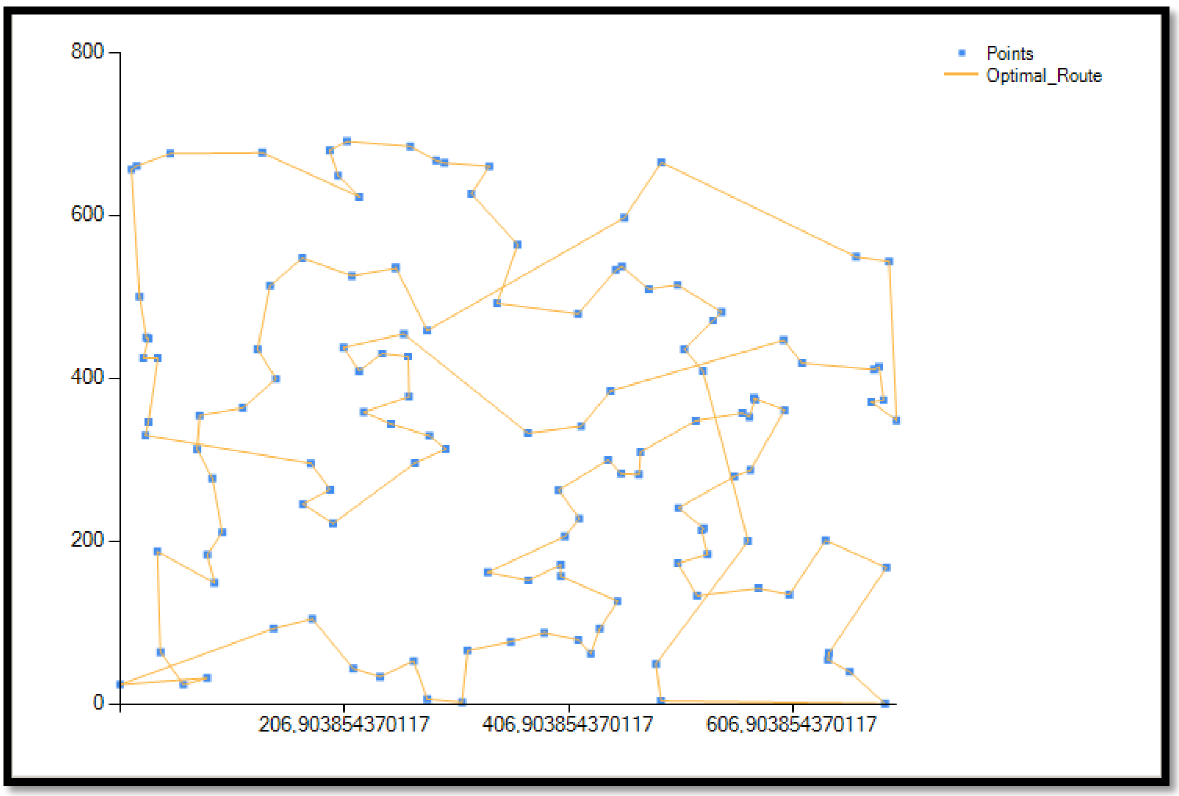


Σχήμα 5.17 : **Η τελική διαδρομή**

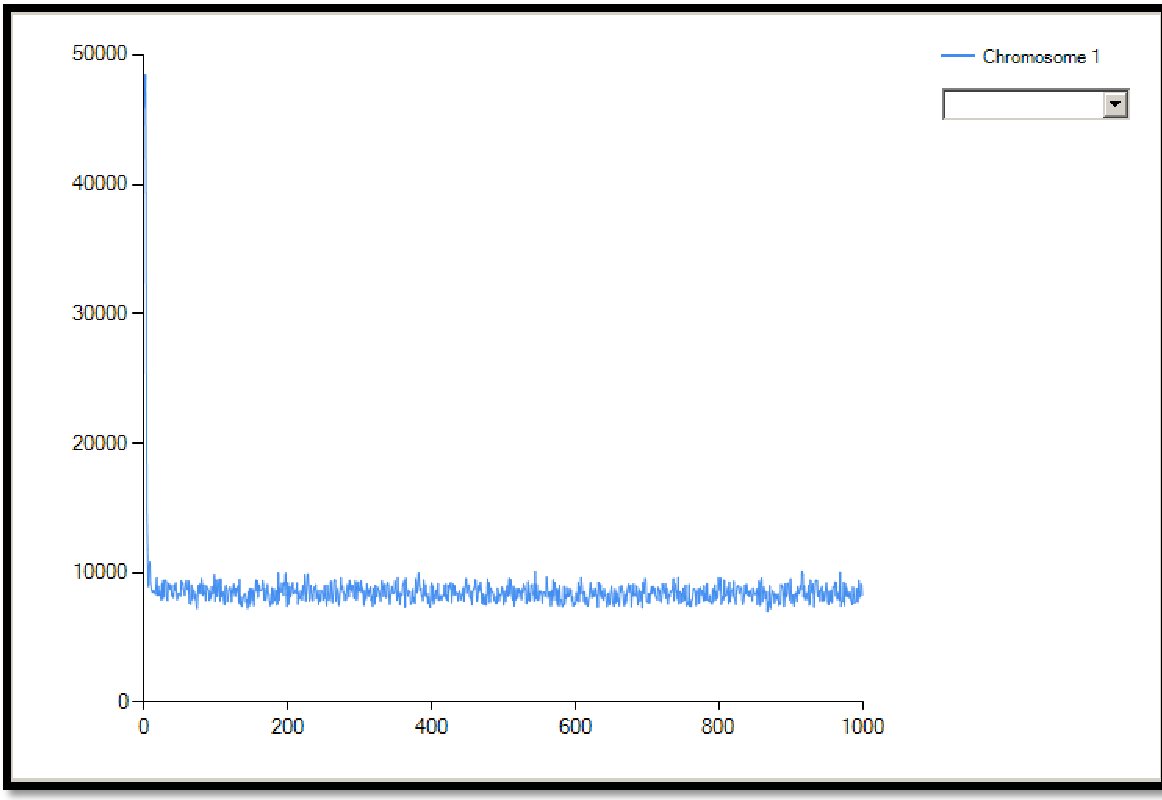


Σχήμα 5.18: **Ο πίνακας γειτνίασης**

Για τον Γενετικό Αλγόριθμο με μετάλλαξη Greedy Round Robin έχουμε τα ακόλουθα σχήματα :

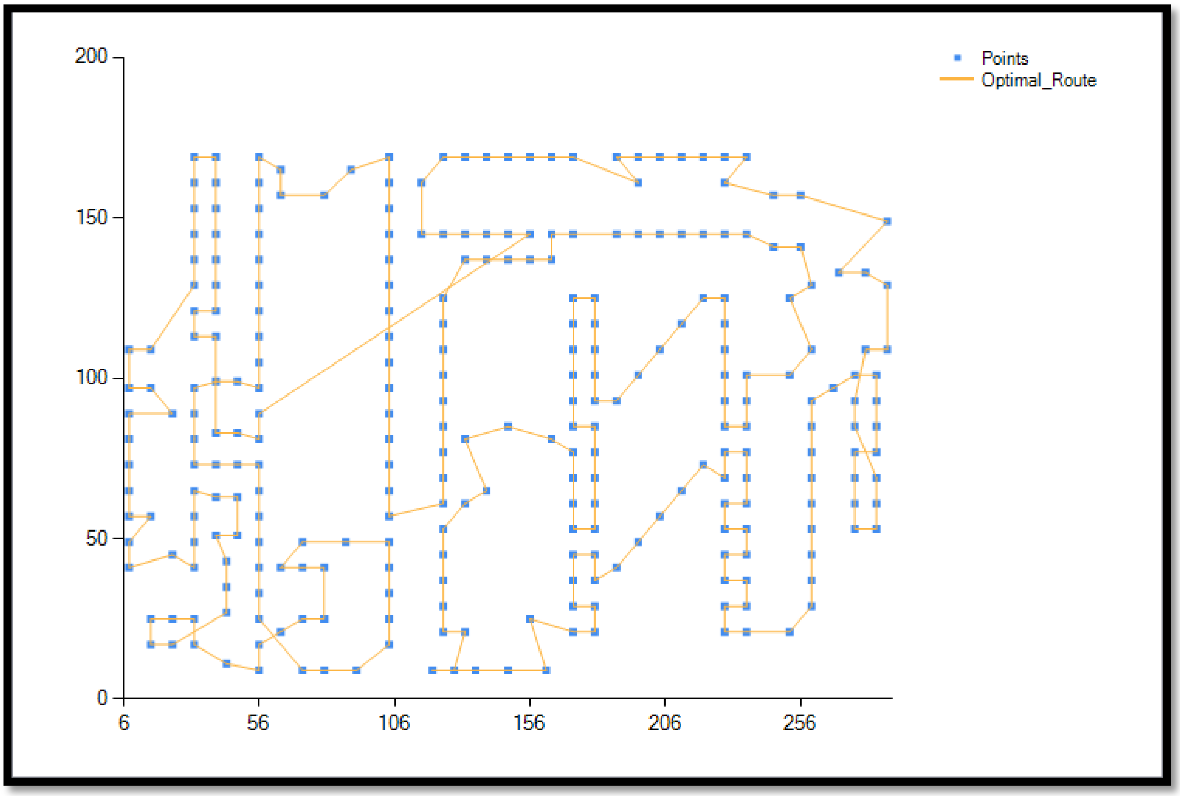


Σχήμα 5.19: **Η τελική διαδρομή**

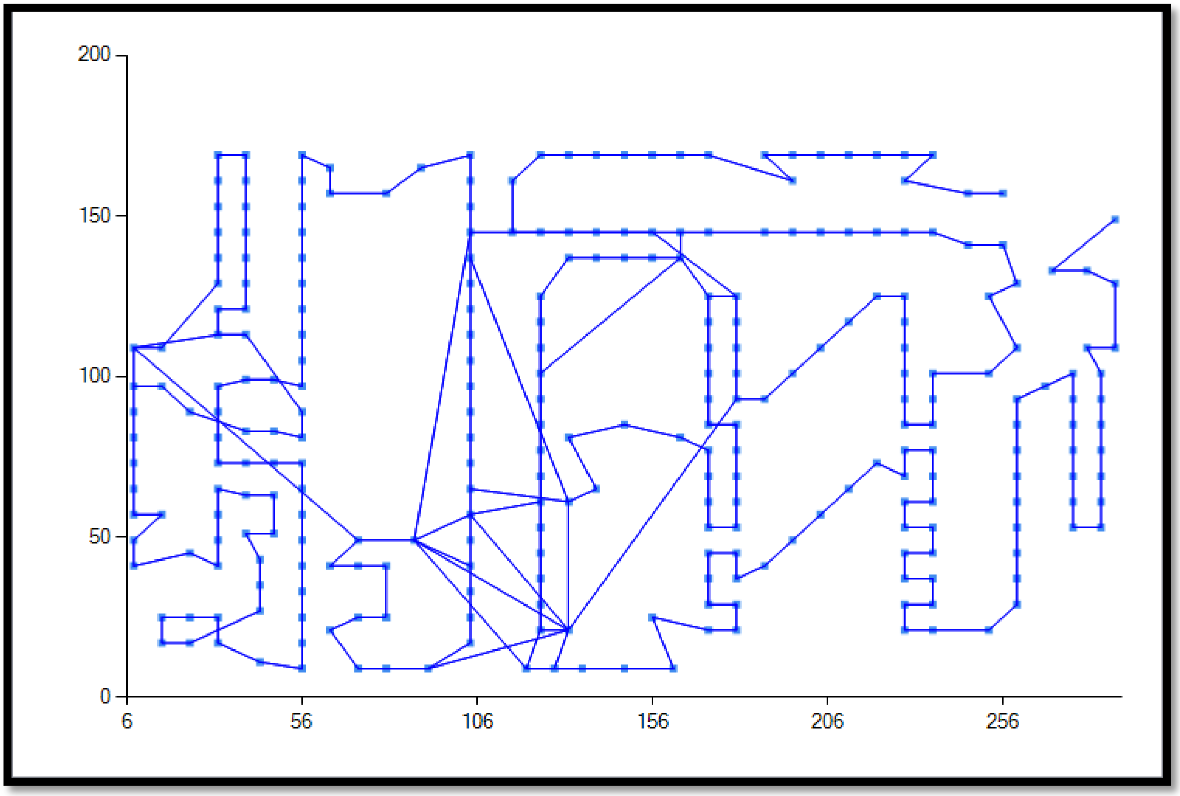


Σχήμα 5.20: **Η εξέλιξη των κόστων κάθε χρωμοσώματος**

A_280



Σχήμα 5.21 : **Η τελική διαδρομή**



Σχήμα 5.22: **Ο πίνακας γειτνίασης**

<u>Παρατήρηση:</u>

- ✓ Όπως , παρατηρούμε και παραπάνω , τα πολύ καλά αποτελέσματα , επιβεβαιώνονται και από τους πίνακες γειτνίασης .

Παράρτημα Α

Ακολουθούν τα δεδομένα κάθε παραδείγματος που χρησιμοποιήθηκε :

Eil_76

Rd_100

Pr_76

Eil_101

KroB_100

Ch_130

KroA_100

Ch_150

Berlin_52

A_280

Βιβλιογραφία

Ελληνική βιβλιογραφία

1. Ιωάννης Μαρινάκης-Μυγδαλάς Αθανάσιος: ‘Σχεδιασμός και Βελτιστοποίηση της Εφοδιαστικής Αλυσίδας’, Εκδόσεις Σοφία, 2008
2. Ιωάννης Μαρινάκης-Μυγδαλάς Αθανάσιος: ‘Συνδυαστική Βελτιστοποίηση’, Πολυτεχνείο Κρήτης , 2004
3. Ολυμπία Βαλάση: Διπλωματική εργασία ‘Βελτιστοποίηση κόστους στη αλυσίδα διανομής κατεψυγμένων προϊόντων της εταιρείας Αφοί Χιωτάκη Ε.Ρ.Ε ‘, 2007
4. Στρατάκη Χρυσάνθη: Μεθευρετικός Αλγόριθμος για την επίλυση του σύνθετου προβλήματος χωροθέτησης εγκαταστάσεων και δρομολόγησης οχημάτων .

Ξένη βιβλιογραφία

5. Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT) and Department of Computer Science and Operations Research, University de Montreal
6. International Conference on Computer Systems and Technologies - *CompSysTech’ 2005*
   Genetic operators crossover and mutation in solving the TSP problem
   Milena Karova Vassil Smarkov Stoyan Penev
7. H. Sengoku, I. Yoshihara, “A Fast TSP Solver Using GA on Java”, 1997.
8. J. Bang-Jensen, G. Gutin and A. Yeo, When the greedy algorithm fails. *Discrete Opti-mization* 1 (2004).
9. The Traveling Salesman Problem:
   A Case Study in Local Optimization by David S. Johnson1, Lyle A. McGeoch2
10. Christofides, Nicos, Eilon, Samuel, 1969. An algorithm for the vehicle-dispatching problem. Operational Research Quarterly 20 (3) 309-318.
11. C.K.Y. Lin , R.C.W. Kwok, ‘ Multi-objective metaheuristics for a location-routing problem with multiple use of vehicles on real data and simulated data’, European Journal of operational research 175 (2006) 1833-1849.
12. Daskin, Mark S., 1995. Network and discrete location: models, algorithms and applications. John Wiley & Sons, Inc., New York.
13. Dorigo ,M., & Gambardella, L. M (1997) Ant colonies for the traveling salesman problem. BioSystems, 43 ,73-81
14. Dilek Tuzun *, Laura I. Burke, ‘ A two-phase tabu search approach to the location routing problem’ , European Journal of Operational Research 116 (1999) 87-99.
15. Gabor Nagy, Saïd Salhi, ‘Location-routing Issues models and methods’, European Journal of operational research 117 (2007) 649-672.
16. Gaskell, T. J., 1967. Bases for vehicle fleet scheduling. Operational Research Quarterly 18 (3), 281-295

17. GUNADI W. NURCAHYO, ROSE ALINDA ALIAS, SM MAMYAM SHAMSUDDIN & MOHD.NOORMD. SAP., Jurnal Antarabangsa (Teknologi Maklumat) 2(2002): 51-64,: 'Sweep Algorithm in Vehicle Routing Problem For Public Transport'.
18. Yannis Marinakis · Magdalene Marinaki, ' A Particle Swarm Optimization Algorithm with Path Relinking for the Location Routing Problem' , published online : 16 November 2007, Springer Science + Business Media B.V. 2007.

Ηλεκτρονικές διευθύνσεις

19. http://www.atlantisresearch.gr/?l=4&cat=72&page=929
20. http://www.go-online.gr/ebusiness/specials/article.html?article_id=1316
21. http://www.go-online.gr/ebusiness/specials/article.html?article_id=1315&PHPSESSID=9d4772ef1150ce3cf7b24a123665f660
22. http://www.go-online.gr/ebusiness/specials/article.html?article_id=1318
23. http://www.plant-management.gr/index.php?id=11
24. http://nemertes.lis.upatras.gr/dspace/bitstream/123456789/1208/1/Nimertis_Dimitrakopoulou.pdf