Πολυτεχνείο Κρήτης

Τμήμα Μηχανικών Παραγωγής και Διοίκησης

Μεταπτυχιακό Δίπλωμα Ειδίκευσης στην Επιχειρησιακή Έρευνα

Μεταπτυχιακή Εργασία

«Εφαρμογή του αλγορίθμου βελτιστοποίησης αποικίας μελισσών σε προβλήματα δρομολόγησης»

Μαραγκουδάκης Ανδρέας

Επιβλέπων καθηγητής: Λέκτορας Μ.Π.Δ. Ιωάννης Μαρινάκης

Χανιά, Φεβρουάριος 2013

Ευχαριστίες

Η επιστροφή μου στο Πολυτεχνείο μετά από έντεκα χρόνια, για μεταπτυχιακές σπουδές, κάθε άλλο παρά μία εύκολη απόφαση ήταν. Η προτροπή ενός παλιού μου φίλου και συμφοιτητή έπαιξε σημαντικό ρόλο στο να πάρω αυτή την απόφαση. Πίστεψε από την αρχή αυτής της προσπάθειας στις δυνατότητές μου και αυτό μου έδωσε δύναμη αλλά και κουράγιο όχι μόνο στο νέο αυτό ξεκίνημα, αλλά και σε όλη την διάρκεια των σπουδών μου, όπου είχα να αντιμετωπίσω διάφορες δυσκολίες. Μη ξεχνάτε οτι έπρεπε να αναπληρώσω τις γνώσεις μου και να γεφυρώσω τα κενά που είχαν δημιουργηθεί μετά από τόσα χρόνια απουσίας από την εκπαίδευση.

Για το λόγο αυτό θα ήθελα να ευχαριστήσω τον φίλο μου, πρώην συμφοιτητή μου και νυν Λέκτορα του τμήματος Μηχανικών Παραγωγής και Διοίκησης στο Πολυτεχνείο Κρήτης, Μαρινάκη Ιωάννη, ο οποίος ήταν και ο επιβλέπων μου στην μεταπτυχιακή εργασία που έχετε στα χέρια σας.

Επίσης θα ήθελα να ευχαριστήσω τη σύζυγό μου, που ήταν ο πιο θερμός υποστηρικτής αυτής της προσπάθειας, για τη συμπαράστασή της και την υπομονή της, όλη αυτή την περίοδο των τριών χρόνων.

Περιεχόμενα

Εισαγωγή

Στην εργασία αυτή εξετάζεται η χρηστικότητα της μεθευρετικής μεθόδου βελτιστοποίησης τεχνητής αποικίας μελισσών (Artificial Bee Colony optimization algorithm, ABC) στην επίλυση του προβλήματος δρομολόγησης οχημάτων με στοχαστική ζήτηση (Vehicle Routing Problem with Stochastic Demand, VRPSD).

Για τους σκοπούς της εργασίας αυτής υλοποιείται η μέθοδος ABC με διαφορετικές τιμές παραμέτρων, και τα αποτελέσματα που προκύπτουν, συγκρίνονται με αυτά των παρακάτω μεθόδων:

- Αλγόριθμος βελτιστοποίησης σμήνους σωματιδίων (**P**article **S**warm **O**ptimization, **PSO**)
- Αλγόριθμος βελτιστοποίησης σμήνους μελισσών (**B**ee **S**warm **O**ptimization, **BSO**)
- Αλγόριθμος βελτιστοποίησης ζευγαρώματος μπάμπουρων (**B**umble **B**ees **M**ating **O**ptimization, **BBMO**)
- Αλγόριθμος βελτιστοποίησης ζευγαρώματος μελισσών (**H**oney **B**ees **M**ating **O**ptimization, **HBMO**)
- Αλγόριθμος βελτιστοποίησης σμήνους πυγολαμπίδων (**G**lowworm **S**warm based **O**ptimization algorithm, **GSO**)
- Αλγόριθμος βελτιστοποίησης επιλογής κλώνων (**C**lonal **S**election **A**lgorithm, **CSA**)
- Γενετικός αλγόριθμος (**G**enetic **A**lgorithm, **GA**)
- Αλγόριθμος Διαφορικής Εξέλιξης (**D**ifferential **E**volution algorithm, **DE**)

Στο Κεφάλαιο 1 παρουσιάζεται το πρόβλημα δρομολόγησης οχημάτων (Vehicle Routing Problem, VRP) και η προέκταση αυτού, που αποτελεί το πρόβλημα δρομολόγησης οχημάτων με στοχαστική ζήτηση (VRPSD). Επίσης, στο τέλος του κεφαλαίου 1 γίνεται μία σύντομη αναφορά στις ερευνητικές μελέτες που έχουν γίνει μέχρι σήμερα στην επίλυση προβλημάτων VRPSD. Στο Κεφάλαιο 2 αρχικά γίνεται μία γενική αναφορά στους μεθευρετικούς αλγόριθμους και στη συνέχεια ακολουθεί ανάλυση μερικών από αυτούς, που είναι εμπνευσμένοι από τα σμήνη μελισσών. Συνεχίζοντας, στο Κεφάλαιο 3 αναλύεται ο αλγόριθμος τεχνητής αποικίας μελισσών

(ABC) καθώς και η εφαρμογή του σε προβλήματα δρομολόγησης οχημάτων με στοχαστική ζήτηση. Στο Κεφάλαιο 4 παρουσιάζονται τα δεδομένα των προβλημάτων που επιλύουμε στην εργασία μας, καθώς και τα αποτελέσματα που μας δίνει ο υβριδικός αλγόριθμος ABC σε αυτά. Στο επόμενο κεφάλαιο, το Κεφάλαιο 5, γίνεται μία σύντομη περιγραφή των αλγόριθμων που έχουν επιλύσει τα συγκεκριμένα προβλήματα της εργασίας μας, οι οποίοι δεν περιγράφηκαν στο κεφάλαιο 2, και ακολουθεί η σύγκριση των αποτελεσμάτων με πίνακες, γραφήματα και αναλύσεις. Στο Κεφάλαιο 6, που είναι και το τελευταίο, αναφέρονται τα συμπεράσματά μας.

Κεφάλαιο 1.

Περιγραφή προβλήματος δρομολόγησης οχημάτων (Vehicle Routing Problem,VRP)

1.1 Εισαγωγή

Στο κεφάλαιο αυτό θα περιγραφεί το πρόβλημα δρομολόγησης οχημάτων (Vehicle Routing Problem, VRP) και το πρόβλημα δρομολόγησης οχημάτων με στοχαστική ζήτηση (Stochastic Vehicle Routing Problem, SVRP).

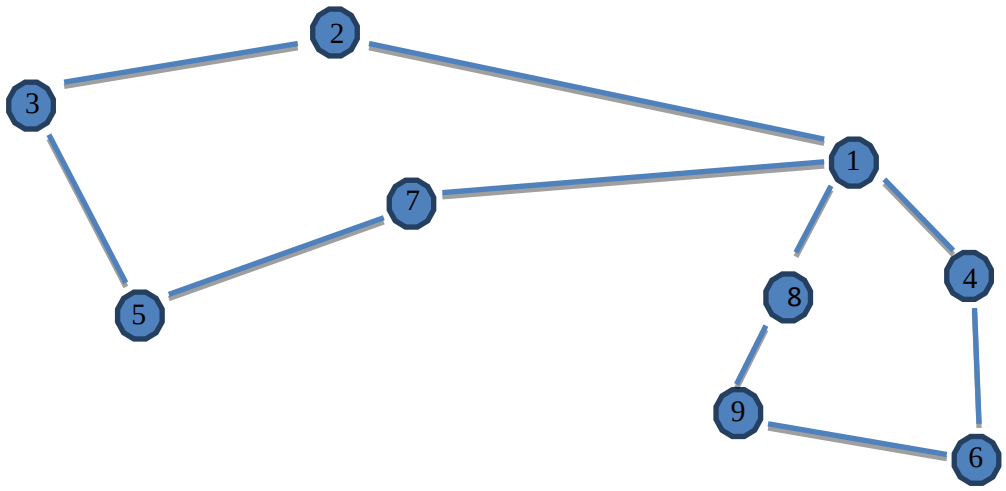
1.2 Πρόβλημα δρομολόγησης οχημάτων (Vehicle Routing Problem, VRP)

Το πρόβλημα δρομολόγησης οχημάτων (VRP) παρουσιάζεται συνεχώς στην καθημερινή μας ζωή. Αυτό που αντιμετωπίζουμε σε τέτοιου είδους προβλήματα, είναι να καθορίζουμε με βέλτιστο τρόπο τις διαδρομές που πρέπει να κάνει ένας στόλος οχημάτων για να εξυπηρετήσει ένα πλήθος πελατών.

Οι Dantzig και Ramser ήταν οι πρώτοι που ασχολήθηκαν με τέτοιου είδους προβλήματα, πριν από περισσότερα από σαράντα χρόνια. Το πρόβλημα που είχαν επιλύσει αφορούσε τη διανομή πετρελαίου σε διάφορους σταθμούς. Από τότε μέχρι και σήμερα, πάρα πολλοί ήταν οι ερευνητές που πρότειναν διάφορους αλγορίθμους για τέτοιου είδους προβλήματα.

Ένα τυπικό πρόβλημα δρομολόγησης οχημάτων είναι ένα πρόβλημα σχεδιασμού βέλτιστης διαδρομής βάσει των διαθέσιμων οχημάτων, της χωρητικότητας του κάθε οχήματος και των πελατών που πρέπει να εξυπηρετηθούν. Οι διαδρομές πρέπει να σχεδιαστούν έτσι ώστε το όχημα να επισκέπτεται μία φορά τον κάθε πελάτη, οι διαδρομές να ξεκινούν και να καταλήγουν στην αποθήκη, η συνολική ζήτηση των πελατών της κάθε διαδρομής να μην ξεπερνάει την χωρητικότητα του οχήματος και σε μία διαδρομή θα πρέπει απαραίτητα να περάσει το όχημα τουλάχιστον από ένα πελάτη (Toth και Vigo, 2002).

Στο σχήμα 1 φαίνεται μια λύση του VRP, δηλαδή οι διαδρομές που θα κάνει ένα όχημα. Ο κόμβος 1 είναι η αποθήκη και οι κόμβοι 2, 3, 4, 5, 6, 7, 8, 9 είναι οι πελάτες.



Σχήμα 1: Λύση ενός προβλήματος VRP.

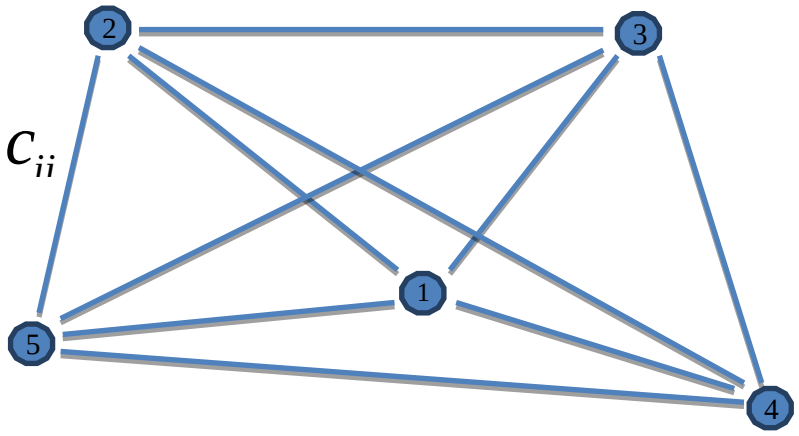
1.2.1 Χαρακτηριστικά του προβλήματος δρομολόγησης οχημάτων

Το οδικό δίκτυο, οι πελάτες, οι αποθήκες, τα οχήματα και οι οδηγοί είναι τα κύρια χαρακτηριστικά ενός προβλήματος δρομολόγησης (Toth και Vigo, 2002).

Οδικό δίκτυο

Το οδικό δίκτυο, το οποίο χρησιμοποιείται για τη μεταφορά των προϊόντων συνήθως περιγράφεται με ένα γράφημα. Τα τόξα του γραφήματος αναπαριστούν τμήματα δρόμου, ενώ οι κορυφές του, αντιστοιχούν σε διασταύρωση δρόμων και στις τοποθεσίες της αποθήκης και των πελατών. Τα τόξα, μπορεί να έχουν μια κατεύθυνση ή να μην έχουν, ανάλογα με το αν υπάρχει η δυνατότητα να μεταφερθούν τα προϊόντα προς τη μία κατεύθυνση του δρόμου ή και από τις δύο, αντίστοιχα. Σε κάθε τόξο αντιστοιχεί και ένα κόστος (c_{ij}, όπου i, j είναι η αρίθμηση των κόμβων) που αυτό μπορεί να αντιπροσωπεύει το μήκος του τόξου, το χρονικό διάστημα που κάνει το όχημα να διασχίσει το δρόμο ή μπορεί να είναι και κάτι άλλο, ανάλογα με τη μοντελοποίηση του προβλήματος. Παρακάτω ακολουθεί γράφημα που αναπαριστά

όλες τις δυνατές διαδρομές σε ένα οδικό δίκτυο, με μια αποθήκη (κόμβος 1) και τέσσερις πελάτες (κόμβοι 2, 3, 4, 5).



Σχήμα 2: Όλες οι δυνατές διαδρομές σε ένα οδικό δίκτυο.

Πελάτες

Τα βασικά χαρακτηριστικά των πελατών είναι τα εξής:

- η κορυφή του γραφήματος στην οποία βρίσκεται ο πελάτης
- το πλήθος των προϊόντων (ζήτηση) τα οποία πρέπει να διανεμηθούν ή να συλλεχθούν από τον πελάτη
- οι χρονικές περίοδοι (time windows) μέσα στις οποίες μπορεί να εξυπηρετηθεί ο πελάτης
- ο χρόνος που απαιτείται για να διανεμηθούν ή να συλλεχθούν τα προϊόντα του πελάτη
- το υποσύνολο των διαθέσιμων οχημάτων που μπορούν να εξυπηρετήσουν τον πελάτη

Αποθήκες

Οι διαδρομές που κάνει ένα όχημα για να εξυπηρετήσει τους πελάτες μπορούν να αρχίζουν και να τελειώνουν σε μία ή περισσότερες αποθήκες. Οι αποθήκες όπως έχουμε προαναφέρει αναπαρίστανται σαν κορυφές του γραφήματος. Η κάθε αποθήκη χαρακτηρίζεται από το πλήθος και το είδος των οχημάτων που σχετίζονται με αυτή και από το πλήθος των προϊόντων που μπορεί να αποθηκεύσει.

Οχήματα

Τα οχήματα χρησιμοποιούνται για τη μεταφορά των προϊόντων. Το είδος του οχήματος που επιλέγεται κάθε φορά εξαρτάται από τις απαιτήσεις των πελατών. Τα βασικά χαρακτηριστικά τους είναι τα εξής:

- η αποθήκη στην οποία γυρνάει το όχημα όταν ολοκληρώσει τη διαδρομή του, και η πιθανότητα τελικά να μη γυρίσει σε αυτή την αποθήκη αλλά σε κάποια άλλη
- η χωρητικότητα του οχήματος, που είναι το μέγιστο βάρος ή η ποσότητα ή το πλήθος των παλετών που μπορεί να φορτώσει
- η υποδιαίρεση του οχήματος σε τμήματα όπου το καθένα θα χαρακτηρίζεται από τη χωρητικότητα του και το είδος των προϊόντων που μπορεί να μεταφέρει
- τα μηχανήματα που είναι διαθέσιμα για τη φόρτωση και εκφόρτωση των προϊόντων
- το σύνολο των τόξων του γραφήματος που μπορεί να διασχίσει το όχημα
- τα κόστη που σχετίζονται με τη χρήση του οχήματος

Οδηγοί

Οι οδηγοί των οχημάτων θα πρέπει να ικανοποιούν κάποιους περιορισμούς που έχουν να κάνουν με το ημερήσιο χρονικό διάστημα που δουλεύουν, το πλήθος και τη διάρκεια των διαλλειμάτων που κάνουν εν ώρα εργασίας, υπερωρίες.

1.2.2 Στόχοι και περιορισμοί ενός προβλήματος δρομολόγησης οχημάτων

Οι στόχοι ενός προβλήματος δρομολόγησης οχημάτων ποικίλλουν. Παρακάτω αναφέρουμε κάποιους από τους στόχους που μπορούν να τεθούν σε ένα τέτοιο πρόβλημα (Μαρινάκης και Μυγδαλάς, 2008).

- ελαχιστοποίηση κόστους διαδρομής
- ελαχιστοποίηση οχημάτων
- ελαχιστοποίηση χρόνου παράδοσης των προϊόντων
- ελαχιστοποίηση παραπονεμένων πελατών

Υπάρχουν διάφοροι περιορισμοί τους οποίους μπορούμε να εντάξουμε στη μοντελοποίηση ενός προβλήματος δρομολόγησης οχημάτων. Κάποιοι από τους περιορισμούς είναι οι ακόλουθοι (Μαρινάκης και Μυγδαλάς, 2008):

- για κάθε διαδρομή που κάνει ένα όχημα, η ποσότητα των προϊόντων που μεταφέρει δε θα πρέπει να ξεπερνά τη χωρητικότητα του οχήματος
- οι πελάτες μπορούν να εξυπηρετηθούν μέσα σε συγκεκριμένα χρονικά διαστήματα
- οι οδηγοί των οχημάτων μπορούν να δουλέψουν κάποιες συγκεκριμένες ώρες
- κάποιοι πελάτες να θέλουν μόνο διανομή ή μόνο παραλαβή προϊόντων και άλλοι να θέλουν και τα δύο

1.3. Πρόβλημα δρομολόγησης οχημάτων με στοχαστική ζήτηση (Vehicle Routing Problem with Stochastic Demand, VRPSD)

Το Πρόβλημα δρομολόγησης οχημάτων με στοχαστική ζήτηση (VRPSD) ανήκει στην κατηγορία των προβλημάτων γνωστά ως Stochastic VRPs (SVRPs) (Leonora Bianchi, Monaldo Mastrolilli, Mauro Birattari, Max Manfrin, Marco Chiarabdini, Luis Paquete, Olivia Rossi-Doria). Σε αυτήν την κατηγορία προβλημάτων στοιχεία του προβλήματος όπως το σύνολο των πελατών, η ζήτηση των πελατών ή ο χρόνος της διαδρομής είναι στοχαστικές μεταβλητές. Χαρακτηριστικό αυτών των προβλημάτων είναι ότι έχουν ένα στοιχείο ντετερμινιστικό. Στο πρόβλημα το δικό μας η ζήτηση των πελατών είναι η στοχαστική μεταβλητή και η δυσκολία σε αυτού του είδους προβλημάτων είναι η αντικειμενική συνάρτηση που θα χρησιμοποιήσουμε για να βρεθεί το κόστος της κάθε διαδρομής (Σπανού, 2010).

Κάποιες από τις εφαρμογές ενός προβλήματος VRPSD είναι οι εξής:

- διανομή πετρελαίου
- συλλογή σκουπιδιών
- συλλογή γαλακτοκομικών προϊόντων από διάφορους παραγωγούς
- διανομή προϊόντων σε παντοπωλεία
- τα σχολικά λεωφορεία και γενικά τα λεωφορεία

1.3.1 Μοντελοποίηση του προβλήματος δρομολόγησης οχημάτων με στοχαστική ζήτηση

Το πρόβλημα δρομολόγησης οχημάτων με στοχαστική ζήτηση αποτυπώνεται από ένα γράφημα της μορφής G{ V, A, D }, όπου:

- V{ 0, 1, …, n } είναι το σύνολο των κόμβων (πελάτες)
- A{ (i, j) : i, j$\in$ V, i$\neq$ j } είναι τα τόξα που συνδέουν τους κόμβους (πελάτες)
- D{ $d_{i,j}$: i, j$\in$ V, i $\neq$j } είναι το κόστος μετάβασης από τον i κόμβο (πελάτη) στον j κόμβο (πελάτη).

Ένα όχημα με χωρητικότητα Q θα πρέπει να ικανοποιήσει τη ζήτηση των πελατών, ελαχιστοποιώντας το κόστος της διαδρομής. Η ζήτηση του κάθε πελάτη (ξ_i i=1,…,n) είναι ανεξάρτητα κατανεμημένη στοχαστική μεταβλητή με γνωστή κατανομή, ακολουθεί διακριτή κατανομή $p_{i,j} = Prob(\xi_t = k)$, $k(= 0,1,2,...,K) \leq Q$, δεν θα πρέπει να ξεπερνά τη χωρητικότητα του οχήματος και γίνεται γνωστή μόνο όταν το όχημα φτάσει στον πελάτη.

Για την εύρεση της διαδρομής που θα κάνει το όχημα, θεωρούμε μια αρχική διαδρομή η οποία ξεκινάει από την αποθήκη. Ανάλογα με τη ζήτηση του επόμενου πελάτη αποφασίζουμε αν το όχημα θα επιστρέψει στην αποθήκη ή θα συνεχίσει στον επόμενο πελάτη. Μερικές φορές παρόλο που η αναμενόμενη ζήτηση του επόμενου πελάτη είναι μικρότερη από το απόθεμα του οχήματος, επιλέγουμε να γυρίσει το όχημα στην αποθήκη για ανεφοδιασμό. Αυτή η πράξη ονομάζεται ‘προληπτικός ανεφοδιασμός’ και ο σκοπός της είναι να αποφευχθεί το ρίσκο να πάει το όχημα στον επόμενο πελάτη και να μη μπορεί να τον ικανοποιήσει, καθώς αυτό θα δημιουργούσε ένα επιπλέον κόστος, αφού θα έπρεπε το όχημα να επιστρέψει στην αποθήκη για ανεφοδιασμό και να επιστρέψει πάλι πίσω στον ίδιο πελάτη.

Παρακάτω φαίνεται η αντικειμενική συνάρτηση με την οποία βρίσκουμε την αναμενόμενη απόσταση (κόστος) της διαδρομής που θα κάνει το όχημα.

$$f_j(q) = \min \begin{cases} f_j^p \\ f_j^r \end{cases} \qquad (1)$$

$$f_j^p = c_{j,j+1} + \sum_{k,k\le q} f_{j+1}(q-k) * p_{j+1,k} + \sum_{k,k>q} \left[2 * c_{j+1,0} + f_{j+1}(q+Q-k) * p_{j+1,k} \right] \qquad (2)$$

$$f_j^r = c_{j,o} + c_{0,j+1} + \sum_{k=1}^{K} f_{j+1}(Q-k) * p_{j+1,k} \qquad (3)$$

υπό τον περιορισμό,

$$f_n(q) = c_{n,0} \quad q \in \mathrm{Ln} \quad (4)$$

Όπου s=(0,1,…,n) είναι μία αρχική διαδρομή που εμείς έχουμε θεωρήσει, q είναι το φορτίο που έχει το όχημα μετά την εξυπηρέτηση του πελάτη j, $f_j(q)$ είναι το αναμενόμενο κόστος της μετάβασης στον επόμενο πελάτη j, f_j^p είναι το αναμενόμενο κόστος της διαδρομής όταν το όχημα δε γυρίζει στην αποθήκη αλλά πάει στον επόμενο πελάτη και f_j^r το αναμενόμενο κόστος όταν το όχημα γυρίζει στην αποθήκη για ανεφοδιασμό (Ι.Μαρινάκης και συνεργάτες, 2011).

Η τυχαιότητα της ζήτησης των πελατών θα μπορούσε να οδηγήσει σε μή εφικτές λύσεις, καθώς υπάρχει το ενδεχόμενο η τελική ζήτηση να υπερβαίνει της χωρητικότητας του οχήματος. Αυτή η κατάσταση είναι γνωστή ως ‘αποτυχία διαδρομής’ (route failure) και όταν αυτό συμβαίνει, θα πρέπει να γίνουν κάποιες διορθωτικές κινήσεις, ώστε να οδηγηθούμε και πάλι σε εφικτές λύσεις. Για το λόγο αυτό επιλέγουμε ένα κατώφλι h_j, έτσι ώστε όταν το φορτίο του οχήματος μετά την εξυπηρέτηση του πελάτη j είναι μεγαλύτερο ή ίσο με h_j θα πρέπει το όχημα να παέι στον επόμενο πελάτη, ενώ αντίθετα αν το φορτίο είναι μικρότερο του h_j θα πρέπει το όχημα να επιστρέψει στην αποθήκη για ανεφοδιασμό. Στους αλγορίθμους της εργασίας αυτής, η ζήτηση των πελατών δεν παίρνει οποιαδήποτε τιμή. Υπάρχουν τρείς πιθανές τιμές για τη ζήτηση:

- είτε θα έχει απόκλιση από την πραγματική ζήτηση το πολύ συν πλην ένα
- είτε θα έχει απόκλιση από την πραγματική ζήτηση το πολύ συν πλην δύο
- είτε θα έχει μηδενική απόκλιση από την πραγματική ζήτηση.

1.3.2 Ερευνητικές μελέτες που έχουν γίνει μέχρι σήμερα στην επίλυση του προβλήματος δρομολόγησης οχημάτων με στοχαστική ζήτηση (VRPSD)

Πολλοί ερευνητές έχουν ασχοληθεί με την επίλυση του προβλήματος δρομολόγησης οχημάτων με στοχαστικές παραμέτρους. Οι σημαντικότερες από αυτές, αναφέρονται παρακάτω (Ιορδανίδου Γ.Ρ. 2012):

- Ο Tillman το 1969 ήταν ο πρώτος που πρότεινε έναν αλγόριθμο για το VRPSD.
- Οι Matthew Protonotarios, George Mourkousis, Ioannis Vyridis και Theodora Varvarigou ασχολήθηκαν με το VRPSD, όπου θέλησαν να ελαχιστοποιήσουν τα κόστη μεταφοράς και να μεγιστοποιήσουν την ικανοποίηση των πελατών, σε μεγάλης κλίμακας προβλήματα, έχοντας ως περιορισμούς τη χωρητικότητα των οχημάτων, τα χρονικά παράθυρα για την εξυπηρέτηση των πελατών και τις ώρες εργασίας ανά ημέρα των οδηγών. Η ζήτηση των πελατών είναι στοχαστική.
- Ο Z.G. Guo και ο K.L. Mak επέλυσαν με γενετικό αλγόριθμο ένα VRPSD με στοχαστικές παραμέτρους τη ζήτηση των πελατών και την παρουσία των πελατών στο σημείο παραλαβής.
- Οι Lars M. Hvattum, Arne Lokketangen και Gilbert Laporte με στοχαστικές παραμέτρους τους πελάτες και τη ζήτηση προτείνουν έναν αλγόριθμο επίλυσης και στόχος τους είναι η ελαχιστοποίηση των οχημάτων και του κόστους των διαδρομών.
- Οι Shangyao Yan, Chin-Jen Chi, Ching-Hui Tang εφάρμοσαν μια τεχνική προσομοίωσης μαζί με στρατηγικές βασισμένες σε συνδέσεις και μονοπάτια για να αναπτύξουν δύο ευρετικούς αλγορίθμους για την επίλυση του VRPSD.
- Οι Leonora Bianchi, Mauro Birattari, Marco Chiarandini, Max Manfrin, Monaldo Mastrolilli, Luis Paquete, Olivia Rossi-Doria και Tommaso Schiavinoto προτείνουν υβριδικούς μεθευρετικούς αλγορίθμους για την επίλυση του VRPSD πάνω στους αλγορίθμους προσομοιωμένης ανόπτησης, περιορισμένης αναζήτησης, επαναλαμβανόμενης τοπικής αναζήτησης, βελτιστοποίησης αποικίας μυρμηγκιών και εξελικτικούς αλγορίθμους.

- Οι Dag Haugland, Sin C. Ho, Gilbert Laporte επιλύουν το VRPSD με στοχαστική ζήτηση, με τον αλγόριθμο περιορισμένης αναζήτησης και με πολυεναρκτήριο ευρετικό αλγόριθμο.
- Οι Xiangyong Li, Peng Tian, Stephen C.H. Leung επιλύουν το VRPSD με χρονικά παράθυρα και στοχαστικές παραμέτρους το χρόνο που κάνει το όχημα, τις διανομές και το χρόνο εξυπηρέτησης των πελατών, με τον αλγόριθμο Περιορισμένης Αναζήτησης.
- Οι Chang-Shi Liu, Ming-Yong Lai είχαν σαν στοχαστική παράμετρο τη ζήτηση των πελατών και προσεγγίστηκε το πρόβλημα με βελτιωμένο εξελικτικό αλγόριθμο.
- Ο Marc Reimann με τη ζήτηση των πελατών σαν στοχαστική παράμετρο χρησιμοποίησε τον αλγόριθμο βελτιστοποίησης αποικίας μυρμηγκιών για να λύσει το πρόβλημα.
- Οι Minis I., Tatarakis A. προτείνουν έναν αλγόριθμο δυναμικού προγραμματισμού για να καθορίσουν το ελάχιστο κόστος δρομολόγησης ενός οχήματος το οποίο διανέμει και συλλέγει προϊόντα με στοχαστική ζήτηση.
- Οι Juan A., Faulin J., Grasman S., Riera D., Marull J., Mendez C. χρησιμοποιούν ένα μετασχηματισμό του στοχαστικού προβλήματος δρομολόγησης οχημάτων σε ένα μικρό σύνολο από περιορισμένης χωρητικότητας προβλήματα δρομολόγησης οχημάτων.
- Οι Lei H., Laporte G., Guo B. επιλύουν το πρόβλημα δρομολόγησης οχημάτων περιορισμένης χωρητικότητας με στοχαστικές απαιτήσεις και χρονικά παράθυρα. Για την επίλυση αυτού του προβλήματος προτείνεται ένας προσαρμοστικός ευρετικός αλγόριθμος με μεγάλη γειτονιά αναζήτησης.
- Οι Goodson J.C., Ohlmann J.W., Thomas B.W. εφαρμόζουν στα γενικής κατηγορίας προβλήματα δρομολόγησης οχημάτων, δομές γειτονιάς με ευρετική αναζήτηση.
- Οι Juan A.A., Faulin J., Jorba J., Caceres J., Marques J.M. συζητάνε για το πώς παράλληλα και κατανεμημένα υπολογιστικά συστήματα μπορούν να εφαρμοστούν αποτελεσματικά για να λύσουν προβλήματα δρομολόγησης οχημάτων με στοχαστική ζήτηση.

Κεφάλαιο 2.

Μεθευρετικές μέθοδοι επίλυσης συνδυαστικών προβλημάτων βελτιστοποίησης.

2.1 Εισαγωγή

Στο κεφάλαιο αυτό αρχικά θα περιγράψουμε τους μεθευρετικούς αλγορίθμους που χρησιμοποιήθηκαν στην εργασία αυτή, με σκοπό τη βελτιστοποίηση διαφόρων δεδομένων προβλημάτων δρομολόγησης οχημάτων με στοχαστική ζήτηση. Στη συνέχεια θα περιγράψουμε τον Αλγόριθμο υπολογισμού της αντικειμενικής συνάρτησης (κόστος μίας λύσης) και τέλος, τους αλγόριθμους τοπικής αναζήτησης που χρησιμοποιήσαμε για την επίλυση των προβλημάτων της εργασίας μας.

2.2 Μεθευρετικοί αλγόριθμοι

Οι μεθευρετικοί αλγόριθμοι είναι αλγόριθμοι επίλυσης που συνδυάζουν διαδικασίες τοπικής αναζήτησης και υψηλότερου επιπέδου στρατηγικές για να δημιουργήσουν μια διαδικασία που είναι ικανή να ξεφεύγει από τοπικά ελάχιστα. Οι μεθευρετικοί αλγόριθμοι συνήθως χρησιμοποιούν σαν υποδιαδικασίες, παραδοσιακούς ευρετικούς αλγόριθμους. Σε πολλές περιπτώσεις, επιτρέπουν την εύρεση μή εφικτών λύσεων με σκοπό να ξεπεραστεί μία λύση που ενδεχομένως να αποτελεί τοπικό ελάχιστο, ώστε η τελική λύση που θα βρεθεί να είναι και η βέλτιστη δυνατή του προβλήματος. Με άλλα λόγια, διερευνάται το πεδίο της λύσης με λιγότερους περιορισμούς, με στόχο να βρεθεί η καλύτερη λύση.

Τα βασικά χαρακτηριστικά των μεθευρετικών αλγορίθμων είναι οτι είναι προσαρμοστικοί, μπορούν να μεταφερθούν εύκολα σε παράλληλη μορφή και κατά κανόνα μοντελοποιούν ένα φαινόμενο που υπάρχει στη φύση. Κάποιοι από αυτούς είναι οι εξής (Ιωάννης Μαρινάκης, Αθανάσιος Μυγδαλάς, 2008):

- Προσομοιωμένη ανόπτηση (Simulated Annealing)
- Περιορισμένη αναζήτηση (Tabu Search)

- Αλγόριθμος βελτιστοποίησης σμήνους σωματιδίων (Particle Swarm Optimization)
- Γενετικοί αλγόριθμοι (Genetic Algorithms)
- Εξελικτικοί αλγόριθμοι (Evolutionary Algorithms)
- Νευρωνικά δίκτυα (Neural Nets)
- Αλγόριθμοι βελτιστοποίησης αποικίας μυρμηγκιών (Ant Colony Optimization)
- Αλγόριθμος διασκορπισμένης αναζήτησης (Scatter Search)
- Διαδικασία άπληστης τυχαιοποιημένης προσαρμοστικής αναζήτησης (Greedy Randomized Adaptive Search Procedure)
- Αλγόριθμος διαφορικής εξέλιξης (Differential Evolution)

Οι μεθευρετικοί αλγόριθμοι χωρίζονται σε δύο κατηγορίες: στους εξελικτικούς αλγόριθμους (evolutionary algorithms) και στους αλγόριθμους που βασίζονται στην ευφυΐα του σμήνους (swarm intelligence based algorithms).

2.3 Εξελικτικοί αλγόριθμοι

Οι εξελικτικοί αλγόριθμοι προέρχονται από το χώρο της Βιολογίας και αποτελούν τεχνικές αναζήτησης και βελτιστοποίησης. Χρησιμοποιούν την ιδέα της φυσικής επιλογής και της επιβίωσης του καλύτερου, σύμφωνα με τον Δαρβίνο. Οι εξελικτικοί αλγόριθμοι χρησιμοποιούνται στα προβλήματα βελτιστοποίησης για την εύρεση καλύτερων λύσεων. Οι μεθοδολογίες των εξελικτικών αλγόριθμων είναι οι Γενετικοί αλγόριθμοι (Genetic Algorithms), ο Γενετικός Προγραμματισμός (Genetic Programming), ο Εξελικτικός Προγραμματισμός (Evolutionary Programming) και η Εξελικτική Στρατηγική (Evolution Strategy). Χωρίς αμφιβολία, οι Γενετικοί Αλγόριθμοι αποτελούν την πιο ευρέως χρησιμοποιούμενη τεχνική.

2.4 Αλγόριθμοι που είναι βασισμένοι στην ευφυΐα του σμήνους

Οι αλγόριθμοι που βασίζονται στην ευφυΐα του σμήνους είναι εμπνευσμένοι από την φύση. Η ευφυΐα του σμήνους είναι ένα υπολογιστικό πρότυπο εμπνευσμένο από την φύση και πιο συγκεκριμένα από την συμπεριφορά διαφόρων κοινωνικών οργανισμών όπως είναι τα μυρμήγκια, οι μέλισσες, τα ψάρια, τα πουλιά. Μιμείται αυτούς τους οργανισμούς αφού ο τρόπος ζωής τους μπορεί να δώσει λύσεις σε διάφορα υπολογιστικά προβλήματα. Οι διάφορες κοινωνίες οργανισμών συνεργάζονται μεταξύ τους για την ολοκλήρωση κοινών σκοπών, όπως είναι η συλλογή της τροφής το ψ ή η κατασκευή της φωλιάς το ψ. Η ευφυΐα του σμήνους αποτελεί συχνό αντικείμενο έρευνας τα τελευταία χρόνια και ορίζεται ως «…οποιαδήποτε απόπειρα σχεδίασης αλγορίθμων ή κατανεμημένων μηχανισμών επίλυσης προβλημάτων εμπνευσμένη από τη συλλογική συμπεριφορά αποικιών κοινωνικών εντόμων και άλλων ζωϊκών κοινοτήτων…» (Bonabeau και συνεργάτες,1999). Αν και ο Bonabeau και οι συνεργάτες του εστίασαν αποκλειστικά στους κοινωνικούς οργανισμούς, ο όρος "σμήνος" χρησιμοποιείται πιο γενικά για την αναφορά σε οποιοδήποτε περιορισμένο σύνολο αλληλεπιδρώντων πρακτόρων ή ατόμων. Το πιο κλασικό παράδειγμα είναι αυτό των μελισσών, ωστόσο η μεταφορά μπορεί να γενικευθεί εύκολα και σε άλλα συστήματα παρόμοιας αρχιτεκτονικής, όπως π.χ. ένα 'ανοσοποιητικό σύστημα' (De Castro, Von Zuben, 1999) μπορεί να θεωρηθεί οτι είναι ένα σμήνος κυττάρων, και ένα 'πλήθος' θα μπορούσε να χαρακτηριστεί ως ένα σμήνος ανθρώπων (Vesterstrøm, Riget 2002).

Στην εργασία αυτή, για την επίλυση των προβλημάτων μας έχει χρησιμοποιηθεί μία σειρά αλγορίθμων βασισμένων στην ευφυΐα του σμήνους και κυρίως, στη συμπεριφορά των μελισσών, όπως θα περιγραφούν στη συνέχεια.

Αυτοί είναι:

- Αλγόριθμος βελτιστοποίησης σμήνους σωματιδίων (**P**article **S**warm **O**ptimization, **PSO**)
- Αλγόριθμος βελτιστοποίησης σμήνους μελισσών (**B**ee **S**warm **O**ptimization, **BSO**)
- Αλγόριθμος βελτιστοποίησης αποικίας μελισσών (**B**ee **C**olony **O**ptimization, **BCO**)

- Αλγόριθμος βελτιστοποίησης ζευγαρώματος μπάμπουρων (**B**umble **B**ees **M**ating **O**ptimization, **BBMO**)
- Αλγόριθμος βελτιστοποίησης ζευγαρώματος μελισσών (**H**oney **B**ees **M**ating **O**ptimization, **HBMO**)
- Αλγόριθμος τεχνητής αποικίας μελισσών (**A**rtificial **B**ee **C**olony optimization algorithm, **ABC**)
- Αλγόριθμος βελτιστοποίησης σμήνους πυγολαμπίδων (**G**lowworm **S**warm based **O**ptimization algorithm, **GSO**)
- Αλγόριθμος βελτιστοποίησης επιλογής κλώνων (**C**lonal **S**election **A**lgorithm, **CSA**)

Στη συγκεκριμένη εργασία ασχοληθήκαμε με την επίλυση του προβλήματος δρομολόγησης οχημάτων με στοχαστική ζήτηση, με τη χρήση του αλγορίθμου Τεχνητής Αποικίας Μελισσών. Στη συνέχεια, περιγράφονται οι αλγόριθμοι που έχουν προταθεί στη βιβλιογραφία και βασίζονται σε προσομοίωση κάποιων διαδικασιών που εκτελούν οι μέλισσες, κατά τη διάρκεια της ζωής τους. Δε θα γίνει αναλυτική παρουσίαση του αλγορίθμου Τεχνητής Αποικίας Μελισσών, ο οποίος θα παρουσιαστεί αναλυτικά στο επόμενο κεφάλαιο.

2.4.1 Αλγόριθμος βελτιστοποίησης σμήνους μελισσών (Bee Swarm Optimization, BSO)

Το 1946, ο Karl Von Fris, καθώς αποκωδικοποιούσε τη 'γλώσσα' των μελισσών, παρατήρησε ότι οι μέλισσες επικοινωνούν κατά την επιστροφή τους στην κυψέλη, μέσω του χορού για την απόσταση, την κατεύθυνση και την πληθωρικότητα μίας πηγής τροφής.

Οι μέλισσες της ίδιας αποικίας επισκέπτονται περισσότερες από δέκα πιθανές περιοχές εκμετάλλευσης, ωστόσο, εστιάζουν στην εκμετάλλευση μόνο ενός μικρού συνόλου από αυτές, τις πλουσιότερες και με την ευκολότερη πρόσβαση. Πολλαπλές παρατηρήσεις, οδήγησαν στο συμπέρασμα ότι οι μέλισσες μίας αποικίας μπορεί να μεταφέρουν την προσοχή τους από μία περιοχή σε μία άλλη.

Κατά την εμπειρία τους το 1991, οι Seely, Camazine και Sneyd έδειξαν ότι όταν δίνεται σε μία αποικία μελισσών η επιλογή μεταξύ δύο αντιδιαμετρικά τοποθετημένων πηγών τροφής με πολύ άνιση συγκέντρωση ζάχαρης, αυτή

συγκεντρώνει τη προσοχή της στη πλουσιότερη πηγή. Στο φαινόμενο αυτό, το σμήνος ακολουθεί τη μέλισσα με τον πιο έντονο χορό, που υποδεικνύει και την περιοχή με την πλουσιότερη πηγή τροφής (Bonabeau και συνεργάτες, 1999).

Ο μεθευρετικός αλγόριθμος BSO είναι εμπνευσμένος από την παραπάνω συλλογική συμπεριφορά των μελισσών. Χειρίζεται τεχνητές μέλισσες για την μίμηση της πραγματικής διαδικασίας συλλογής τροφής των μελισσών στην επίλυση προβλημάτων. Αρχικά, μία μέλισσα ονόματι *Bee Init* βρίσκει μία λύση που να παρουσιάζει καλά χαρακτηριστικά, τα *Sref*, από την οποία οι άλλες λύσεις του χώρου εύρεσης καθορίζονται βάσει συγκεκριμένης στρατηγικής. Το σύνολο αυτών των λύσεων καλείται *Search Area*. Έπειτα, κάθε μέλισσα θα θεωρήσει μία λύση του *Search Area* ως σημείο εκκίνησης της έρευνας. Με την ολοκλήρωση της έρευνας, κάθε μέλισσα επικοινωνεί στους συναδέλφους της, μέσω του χορού (δομή *Dance*), τη καλύτερη λύση που 'επισκέφτηκε'. Μία από τις λύσεις της λίστας γίνεται η νέα λύση αναφοράς για τη νέα επανάληψη της διαδικασίας. Για την αποφυγή σχηματισμού κύκλων, οι λύσεις αναφοράς αποθηκεύονται σε μία λίστα, η οποία είναι γνωστή και ως λίστα *taboo*.

Η επιλογή της λύσης αναφοράς γίνεται αρχικά βάση του κριτηρίου ποιότητας. Ωστόσο, αν έπειτα από μία πάροδο χρόνου, το σμήνος δεν παρατηρήσει βελτίωση της ποιότητας, ενσωματώνει ένα δεύτερο κριτήριο διαφοροποίησης ώστε να ξεφύγει από την περιοχή στην οποία κατά πάσα περίπτωση έχει παγιδευτεί.

2.4.2 Αλγόριθμος βελτιστοποίησης αποικίας μελισσών (Bee Colony Optimization, BCO)

Οι Lucic και Teodorovic (2001) ήταν μεταξύ των πρώτων που χρησιμοποίησαν τις βασικές αρχές της συλλογικής ευφυΐας των μελισσών για να επιλύσουν συνδυαστικά προβλήματα βελτιστοποίησης. Ο BCO είναι ένας πληθυσμιακά βασισμένος αλγόριθμος, όπου ο πληθυσμός των τεχνητών μελισσών αναζητεί τη βέλτιστη λύση. Οι τεχνητές μέλισσες αντιπροσωπεύουν πράκτορες που λειτουργούν συλλογικά για την επίλυση πολύπλοκων συνδυαστικών προβλημάτων. Κάθε μέλισσα παράγει μία λύση του προβλήματος με τον αλγόριθμο να αποτελείται από δύο εναλλασσόμενες φάσεις: το "προς τα μπροστά πέρασμα" (*forward pass*) και το "προς τα πίσω πέρασμα" (*backward pass*). Σε κάθε "προς τα μπροστά πέρασμα" κάθε μέλισσα εξερευνά το χώρο εύρεσης λύσεων, εκτελώντας έναν αριθμό προκαθορισμένων

βημάτων για τη δημιουργία λύσης ή τη βελτίωση υπάρχουσας λύσης, οδηγώντας έτσι σε μία νέα, καλύτερη λύση. Αφού έχουν δημιουργήσει νέες μερικές λύσεις, οι μέλισσες επιστρέφουν στη φωλιά τους για την εκτέλεση του "προς τα πίσω περάσματος". Στη φάση αυτή, όλες οι μέλισσες μοιράζονται πληροφορίες για τις μερικές λύσεις που έχουν δημιουργήσει.

Στη φύση, οι μέλισσες εκτελούν έναν ιεροτελεστικό χορό για την ενημέρωση των άλλων μελισσών όσον αφορά την ποσότητα τροφής που έχουν συλλέξει και την απόσταση της πηγής της τροφής από την κυψέλη. Στον αλγόριθμο BCO δημοσιοποιείται η ποιότητα της λύσης, δηλαδή η τιμή της αντικειμενικής συνάρτησης. Κατά το "προς τα πίσω πέρασμα", κάθε μέλισσα επιλέγει με μία συγκεκριμένη πιθανότητα αν θα χορέψει για τη προσέλκυση και άλλων μελισσών πριν την επιστροφή στη μερική λύση που έχει βρει, ή αν θα παρατήσει τη λύση αυτή. Μέλισσες με καλύτερη τιμή της αντικειμενικής έχουν μεγαλύτερη πιθανότητα να συνεχίσουν με την ίδια λύση. Κάθε μη αντιστοιχισμένη μέλισσα επιλέγει με σταθμισμένη τυχαιότητα ποια μερική λύση θα ακολουθήσει.

Κατά το δεύτερο "προς τα εμπρός πέρασμα", οι μέλισσες αναπτύσσουν τις μερικές λύσεις που έχουν δημιουργήσει προηγουμένως, κατά ένα προκαθορισμένο αριθμό κόμβων, και έπειτα εκτελούν ξανά ένα πέρασμα προς τα πίσω και επιστρέφουν στη κυψέλη. Στη κυψέλη, οι μέλισσες λαμβάνουν πάλι μέρος σε μία διαδικασία λήψης απόφασης, λαμβάνουν μία απόφαση, εκτελούν ένα τρίτο "προς τα εμπρός πέρασμα", συνεχίζοντας έτσι τη διαδικασία. Οι δύο φάσεις εκτελούνται επαναληπτικά μέχρι την ικανοποίηση κάποιου κριτηρίου τερματισμού. Τέτοια κριτήρια θα μπορούσαν να είναι ο μέγιστος αριθμός επαναλήψεων, ή ο μέγιστος αριθμός επαναλήψεων χωρίς βελτίωση.

2.4.3 Αλγόριθμος βελτιστοποίησης ζευγαρώματος μελισσών (Honey Bees Mating Optimization, HBMO)

Ο Αλγόριθμος Βελτιστοποίησης Ζευγαρώματος Μελισσών προτάθηκε από τον Abass το 2001 και αναπαριστά τη διαδικασία ζευγαρώματος της βασίλισσας μέλισσας της κυψέλης. Πριν την παρουσίαση του αλγορίθμου ας δούμε μερικά χαρακτηριστικά των μελισσών. Οι μέλισσες είναι κοινωνικά έντομα και δουλεύουν ομαδικά για να κατασκευάζουν τις κυψέλες όπου ζουν μέσα. Μια αποικία από μέλισσες συνήθως αποτελείται από μία βασίλισσα, από μηδέν έως μερικές χιλιάδες κηφήνες (εξαρτάται

από τη χρονική στιγμή κατά τηδιάρκεια της σεζόν) και συνήθως από 10000-60000 εργάτριες. Η δουλειά της βασίλισσας είναι να γεννάει αυγά, ενώ μπορεί να γεννήσει μέχρι 1500 αυγά την ημέρα και ζει συνήθως από 5 μέχρι 6 χρόνια. Μόνο η βασίλισσα μπορεί να φάει βασιλικό πολτό, πράγμα που την κάνει μεγαλύτερη από όλες τις άλλες μέλισσες στην κυψέλη. Οι κηφήνες παίζουν το ρόλο του 'αρσενικού' στις κυψέλες αφού ο κύριος ρόλος τους είναι να γονιμοποιούν τη βασίλισσα. Στο τέλος της σεζόν οι κηφήνες φεύγουν από την κυψέλη για να πεθάνουν. Οι κηφήνες ποτέ δεν ζουν πάνω από 6 μήνες. Οι εργάτριες κάνουν όλες τις δουλειές που χρειάζεται για να λειτουργήσει η κυψέλη. Στην ουσία φυλούν την κυψέλη και είναι στείρα θηλυκά. Όταν είναι νέες μένουν στην κυψέλη και κάνουν όλες τις κατασκευαστικές δουλειές, ακόμα και τη φροντίδα των νεογνών. Οι μεγαλύτερες εργάτριες βγαίνουν έξω από την κυψέλη και ψάχνουν να βρουν νέκταρ, νερό, γύρη και γενικά οτιδήποτε χρειάζεται η κυψέλη. Οι εργάτριες που θα γεννηθούν στην αρχή της σεζόν θα ζήσουν περίπου 6 βδομάδες ενώ αυτές που θα γεννηθούν το φθινόπωρο θα ζήσουν μέχρι την επόμενη άνοιξη.

Στη διαδικασία ζευγαρώματος των μελισσών, η βασίλισσα ζευγαρώνει κατά τη διάρκεια των 'πτήσεων ζευγαρώματος' αρκετά μακριά από την κυψέλη. Η βασίλισσα ξεκινάει με ένα χορό που πραγματοποιείται στην κυψέλη και στη συνέχεια πραγματοποιεί μια 'πτήση ζευγαρώματος' κατά την οποία οικηφήνες την ακολουθούν και ζευγαρώνουν μαζί της στον αέρα. Η βασίλισσα καταδιώκεται από ένα μεγάλο σμήνος από κηφήνες. Η γονιμοποίηση τελειώνει με το θάνατο του κηφήνα και με τη βασίλισσα να έχει πάρει το σημάδι ότι ζευγάρωσε με τον κηφήνα. Η βασίλισσα μπορεί να ζευγαρώσει πολλές φορές κατά τη διάρκεια μιας πτήσης αλλά ο κηφήνας μονάχα μία φορά. Σε κάθε ζευγάρωμα με διαφορετικό κηφήνα, το σπέρμα του κηφήνα αποθηκεύεται στην σπερματοθήκη της βασίλισσας για να δημιουργήσει το γενετικό υλικό της αποικίας. Κάθε φορά που η βασίλισσα γεννάει γονιμοποιημένα αυγά χρησιμοποιεί μια μείξη από το σπέρμα που έχει αποθηκεύσει από όλους τους κηφήνες στη σπερματοθήκη της.

Μπορεί να θεωρηθεί ότι πρόκειται απλά για ένα γενετικό αλγόριθμο με τη χρήση ενός πολύ ισχυρού υπερ-γονέα (τη βασίλισσα). Αλλά φυσικά αυτή η μέθοδος δεν είναι απλά ένας γενετικός αλγόριθμος. Οι δύο βασικές διαφορές είναι ότι η βασίλισσα κινείται τυχαία στο χώρο και επιλέγει, όπως θα δούμε και παρακάτω, βάσει της ταχύτητάς της και της ενέργειάς της με ποιον κηφήνα θα ζευγαρώσει. Ακόμα και αν η

βασίλισσα ζευγαρώσει με ένα κηφήνα, δεν δημιουργεί κατευθείαν ένα νεογνό, αλλά αποθηκεύει το γονότυπό του (μέρος της λύσης του) στη σπερματοθήκη της και το κάθε νεογνό (απόγονος) δημιουργείται μόνο όταν η 'πτήση ζευγαρώματος' έχει ολοκληρωθεί. Άλλη μία διαφορά από τους γενετικούς αλγόριθμους είναι ότι οι απόγονοι δεν δημιουργούνται από μία λύση (έναν κηφήνα) αλλά παίρνουν χαρακτηριστικά από πολλούς κηφήνες και από τη βασίλισσα.

Το πρώτο βήμα που πρέπει να γίνει για την υλοποίηση του Αλγορίθμου Βελτιστοποίησης Ζευγαρώματος Μελισσών είναι να δημιουργηθεί ο πληθυσμός των μελισσών ο οποίος αποτελεί την αρχική κυψέλη. Η καλύτερη μέλισσα αποτελεί τη βασίλισσα και όλες οι υπόλοιπες τους κηφήνες, ενώ οι μέλισσες εργάτριες είναι μέθοδοι τοπικής αναζήτησης. Θα πρέπει να οριστεί ένας αριθμός ο οποίος θα ορίζει το μέγεθος της σπερματοθήκης της βασίλισσας. Η 'πτήση ζευγαρώματος' τελειώνει όταν γεμίσει η σπερματοθήκη της βασίλισσας. Επίσης θα πρέπει να οριστεί ο αριθμός των βασιλισσών και ο αριθμός των νεογνών. Όταν ξεκινάει η πτήση της βασίλισσας στην ουσία η βασίλισσα κινείται στο χώρο λύσεων με κάποια ταχύτητα και ζευγαρώνει με κάποια πιθανότητα με έναν κηφήνα. Στην αρχή της πτήσης η βασίλισσα έχει μια ενέργεια και όταν γυρίσει στην κυψέλη αυτή η ενέργεια βρίσκεται στο διάστημα από μηδέν έως το μέγεθος της σπερματοθήκης. Παρακάτω φαίνεται ο τύπος για την πιθανότητα να ζευγαρώσει ένας κηφήνας με τη βασίλισσα.

$$Prob(D) = e^{\frac{-\Delta(f)}{Speed(t)}} \qquad (5)$$

όπου $Prob(D)$ αντιπροσωπεύει την πιθανότητα να ζευγαρώσει ο κηφήνας D με τη βασίλισσα, Δ(f) είναι η διαφορά στην τιμή της συνάρτησης ποιότητας της βασίλισσας και του κηφήνα D και Speed(t) είναι η ταχύτητα της βασίλισσας τη χρονική στιγμή t. Η πιθανότητα του ζευγαρώματος είναι υψηλή όταν η βασίλισσα βρίσκεται στο ξεκίνημα της 'πτήσης ζευγαρώματος' ή όταν η συνάρτηση ποιότητας του κηφήνα είναι περίπου τόσο καλή όσο της βασίλισσας. Μετά από κάθε μετάβαση της βασίλισσας στο χώρο, η ταχύτητά της και η ενέργειά της μειώνονται βάσει των τύπων (Abbass 2001aκαι b):

$$Speed(t+1) = a \times Speed(t) \qquad (6)$$

$$energy(t+1) = energy(t) - step \qquad (7)$$

όπου α είναι ένας παράγοντας στο διάστημα (0,1) και αντιπροσωπεύει το ποσό που μειώνεται η ταχύτητα σε κάθε μετάβαση. Αν το α πάρει πολύ μεγάλη τιμή η ταχύτητα μειώνεται αργά, αντίθετα, αν το α πάρει μικρή τιμή η ταχύτητα μειώνεται πολύ γρήγορα. Το step αντιπροσωπεύει το ποσό που μειώνεται η ενέργεια σε κάθε μετάβαση. Υπάρχουν υλοποιήσεις του αλγορίθμου που αντί για την παραπάνω εξίσωση ενέργειας χρησιμοποιούν (Magda Marinakis και συνεργάτες, 2010 και Magda Marinakis και συνεργάτες, 2008) ένα τύπο της μορφής:

$$energy(t+1) = a \times energy(t) \qquad (8)$$

ώστε η μείωση στην ταχύτητα και στην ενέργεια να είναι ανάλογες. Αρχικά, η ταχύτητα και η ενέργεια της βασίλισσας αρχικοποιούνται τυχαία. Ένας αριθμός από 'πτήσεις ζευγαρώματος' πραγματοποιούνται. Στο ξεκίνημα της πτήσης η ταχύτητα είναι συνήθως μεγάλη και έτσι η βασίλισσα κάνει πολύ μεγάλα βήματα στο χώρο. Καθώς η ενέργεια της βασίλισσας μειώνεται, η ταχύτητά της μειώνεται, πράγμα που έχει ως αποτέλεσμα οι δυνατότητες αναζήτησης που έχει η βασίλισσα να μειώνονται. Ο γονότυπος των κηφήνων που έχει αποθηκευτεί στη σπερματοθήκη της βασίλισσας διασταυρώνεται με της βασίλισσας με τη χρήση οποιουδήποτε τελεστή διασταύρωσης που έχει την ικανότητα να μπορεί να χρησιμοποιήσει παραπάνω από δύο γονείς.

Ο ρόλος των εργατριών περιορίζεται σε μια απλή διαδικασία τοπικής αναζήτησης, που στην ουσία παίζει το ρόλο του ταΐσματος των νεογνών με βασιλικό πολτό με στόχο να βρεθεί μια καλύτερη βασίλισσα. Για αυτό το λόγο οι εργάτριες δεν είναι ξεχωριστά μέλη του πληθυσμού αλλά χρησιμοποιούνται ως διαδικασίες τοπικής αναζήτησης με στόχο να βελτιωθούν οι λύσεις που βρέθηκαν από τη διαδικασία ζευγαρώματος της βασίλισσας. Αν χρησιμοποιηθεί μία μόνο διαδικασία τοπικής αναζήτησης τότε δεν εκμεταλλευόμαστε το γεγονός ότι κάθε μία από τις εργάτριες έχει διαφορετικές ικανότητες και η επιλογή μιας εργάτριας είναι πολύ σημαντική για τη βελτίωση της λύσης του νεογνού. Οι διαφορετικές ικανότητες των εργατριών αντιστοιχούν σε διαφορετικές διαδικασίες τοπικής αναζήτησης, ανάλογα με το πρόβλημα που επιλύουμε. Έτσι, στο ξεκίνημα του αλγορίθμου θα πρέπει να καθορίσουμε το σύνολο των διαδικασιών τοπικής αναζήτησης και να αντιστοιχίσουμε κάθε μία εργάτρια σε μία διαδικασία τοπικής αναζήτησης ή σε συνδυασμό παραπάνω από μίας διαδικασίας τοπικής αναζήτησης. Η επιλογή από το νεογνό για το ποια εργάτρια θα το θρέψει, δηλαδή ποια μέθοδος τοπικής αναζήτησης θα εφαρμοστεί,

μπορεί να γίνει με τυχαίο τρόπο ή με μια ποιο συγκεκριμένη διαδικασία. Για παράδειγμα το καλύτερο νεογνό να τραφεί από την ισχυρότερη εργάτρια (πιο αποτελεσματική, θεωρητικά, διαδικασία τοπικής αναζήτησης).

Όταν βρεθεί ένα νεογνό που έχει καλύτερη λύση από τη βασίλισσα τότε την αντικαθιστά και η τρέχουσα βασίλισσα φεύγει από την κυψέλη. Όλα τα υπόλοιπα νεογνά θα είναι οι κηφήνες στην επόμενη 'πτήση ζευγαρώματος' της νέας βασίλισσας. Ο πληθυσμός των κηφήνων δεν μεταβάλλεται αφού όπως αναφέραμε και παραπάνω αν ένας κηφήνας επιλεγεί για ζευγάρωμα με τη βασίλισσα σε μία πτήση ζευγαρώματος, αμέσως μετά πεθαίνει, άρα διαγράφεται από το συνολικό πληθυσμό. Αν τώρα ο αριθμός των απογόνων είναι μεγαλύτερος από τον αριθμό των κηφήνων που διαγράφηκαν σε μια πτήση ζευγαρώματος τότε επιλέγεται ίσος αριθμός κηφήνων για την επόμενη πτήση όσο ήταν ο αρχικός πληθυσμός.

2.4.4 Αλγόριθμος βελτιστοποίησης ζευγαρώματος μπάμπουρων (Bumble Bees Mating Optimization, BBMO)

Ο Αλγόριθμος Βελτιστοποίησης Ζευγαρώματος Μελισσών προτάθηκε από τον Ι.Μαρινάκη και τους συνεργάτες του και αναπαριστά τη διαδικασία ζευγαρώματος της βασίλισσας των μπάμπουρων στην κυψέλη. Πριν την παρουσίαση του αλγορίθμου ας δούμε μερικά χαρακτηριστικά των μπάμπουρων. Οι μπάμπουρες είναι κοινωνικά έντομα που δημιουργούν αποικίες που αποτελούνται από τη βασίλισσα, πολλές εργάτριες και τους κηφήνες. Οι βασίλισσες είναι τα μόνα μέλη από την κυψέλη που επιβιώνουν από τη μία χρονιά στην άλλη, αφού περνούν το χειμώνα σε χειμερία νάρκη σε καλά προστατευμένο μέρος. Καθώς οι βασίλισσες ξυπνούν από τη χειμερία νάρκη συλλέγουν γύρη και νέκταρ και όταν βρουν το κατάλληλο μέρος, προετοιμάζουν την κυψέλη όπου θα αποθηκεύσουν τροφή και προετοιμάζουν τα κελιά που θα γεννήσουν τα αυγά τους (Goulson, 2009).

Η βασίλισσα μπορεί να γεννήσει γονιμοποιημένα και μη-γονιμοποιημένα αυγά. Τα γονιμοποιημένα αυγά έχουν χρωμοσώματα από τη βασίλισσα και από ένα ή περισσότερους κηφήνες, με τους οποίους είχε γονιμοποιηθεί τον προηγούμενο χρόνο, και παράγουν τις εργάτριες. Τα μη-γονιμοποιημένα περιλαμβάνουν χρωμοσώματα μόνο από τη βασίλισσα και παράγουν τους κηφήνες. Μετά τη δημιουργία των πρώτων εργατριών, η βασίλισσα δεν ασχολείται με την αναζήτηση τροφής και

παραμένει στην κυψέλη απλά γεννώντας καινούρια αυγά. Πλέον κάποιες από τις εργάτριες αναλαμβάνουν τη διαδικασία αναζήτησης τροφής εκτός της κυψέλης και κάποιες άλλες παραμένουν στην κυψέλη για να βοηθήσουν τη βασίλισσα στην ανατροφή των νεογνών. Οι εργάτριες είναι ικανές να παράγουν απλοειδή αυγά χωρίς τη διαδικασία γονιμοποίησης όταν η ικανότητα της βασίλισσας να παράγει αυγά αρχίσει να μειώνεται. Από αυτά τα αυγά μπορούν να προέλθουν μόνο κηφήνες (Goulson, 2009).

Μετά που οι κηφήνες θα φύγουν από τη φωλιά, αναδεικνύονται από τα αυγά που έχει γεννήσει η βασίλισσα οι καινούριες βασίλισσες. Όταν οι κηφήνες και οι νέες βασίλισσες φύγουν από την κυψέλη, η αποικία αρχίζει να εκφυλίζεται. Η παλιά βασίλισσα σταματάει να γεννάει αυγά και αρχίζει να γίνεται αδύναμη από τα γηρατειά. Οι εργάτριες συνεχίζουν να ψάχνουν για τροφή αλλά πλέον μόνο για τον εαυτό τους. Μακριά από την κυψέλη οι νέες βασίλισσες και οι κηφήνες αναζητούν νέκταρ και γύρη και περνούν τις νύχτες σε λουλούδια ή σε τρύπες. Η βασίλισσα ζευγαρώνει με κάποιον ή κάποιους κηφήνες, το σπέρμα αποθηκεύεται στην σπερματοθήκη και η βασίλισσα αναζητά ένα μέρος για να ξεχειμωνιάσει. Στους μπάμπουρες υπάρχουν τριών διαφορετικών ειδών διαδικασίες ζευγαρώματος. Στην πρώτη διαδικασία, ένας κηφήνας κάθεται σε κάποιο ψηλό σημείο και παρατηρεί το χώρο περιμένοντας τη βασίλισσα να περάσει από κάποιο κοντινό σημείο και να την κυνηγήσει με στόχο να ζευγαρώσουν. Η δεύτερη συμπεριφορά είναι όταν ο κηφήνας δημιουργεί ένα μονοπάτι που έχει ορισμένη οσμή το οποίο το μαρκάρει με μία φερομόνη και με αυτό τον τρόπο μία βασίλισσα έλκεται από το μονοπάτι και ακολουθεί το ίχνος μέχρι να βρει τον κηφήνα. Η τρίτη συμπεριφορά είναι όταν ο κηφήνας περιμένει έξω από την κυψέλη μία βασίλισσα να φύγει οπότε την ακολουθεί.

Στον αλγόριθμο βελτιστοποίησης ζευγαρώματος μπάμπουρων υπάρχουν τριών ειδών μπάμπουρες στην αποικία, οι βασίλισσες, οι εργάτριες και οι κηφήνες. Αρχικά, ένας αριθμός από μπάμπουρες επιλέγονται τυχαία. Κάθε μπάμπουρας αντιστοιχεί σε μία λύση του προβλήματος. Άρα το σύνολο των μπάμπουρωνείναι το σύνολο των λύσεων του προβλήματος. Έστω ότι με **n** συμβολίζουμε το συνολικό αριθμό των μεταβλητών. Οι μπάμπουρες αναπαρίστανται με διανύσματα μεγέθους **n**. Ανάλογα με το πρόβλημα που έχουμε να επιλύσουμε οι μπάμπουρες παίρνουν και αντίστοιχες αρχικές τιμές. Δηλαδή, αν έχουμε ένα πρόβλημα με συνεχείς τιμές στο διάστημα (0,1) τότε οι αρχικές τιμές των λύσεων είναι τυχαίες τιμές σε αυτό το διάστημα (Ι.Μαρινάκης και

συνεργάτες, 2010). Αν έχουμε ένα πρόβλημα για παράδειγμα επιλογής χαρακτηριστικών όπου το 1 συμβολίζει ότι το χαρακτηριστικό έχει επιλεγεί και το 0 ότι δεν έχει επιλεγεί, τότε οι μπάμπουρες παίρνουν ακέραιες τιμές 0 ή 1 (Ι.Μαρινάκης και συνεργάτες, 2009). Ενώ αν έχουμε ένα πρόβλημα δρομολόγησης όπου μία λύση αναπαρίσταται με μία διαδρομή τότε η κάθε μία λύση αναπαρίσταται με τη διαδρομή που αντιστοιχεί στη λύση (Ι.Μαρινάκης, Μ.Μαρινάκη, 2010). Στη συνέχεια η τιμή της αντικειμενικής συνάρτησης του κάθε μπάμπουρα υπολογίζεται και ο καλύτερος από τους μπάμπουρες γίνεται η βασίλισσα. Στη φάση αρχικοποίησης του αλγορίθμου όλοι οι άλλοι μπάμπουρες γίνονται κηφήνες. Η βασίλισσα επιλέγει τους κηφήνες που θα χρησιμοποιηθούν για το ζευγάρωμα με τη χρήση της δεύτερης συμπεριφοράς που περιγράψαμε προηγούμενα, που υποθέτει ότι οι καλύτεροι κηφήνες αφήνουν περισσότερη φερομόνη στομονοπάτι τους και ότι η βασίλισσα επιλέγει τα πιο ελπιδοφόρα μονοπάτια. Ένας απλός τρόπος για να γίνει αυτή η διαδικασία είναι με την ταξινόμηση των τιμών των αντικειμενικών συναρτήσεων όλων των κηφήνων. Κάθε φορά που η βασίλισσα ζευγαρώσει με ένα κηφήνα, ο γονότυπος του κηφήνα αποθηκεύεται στη σπερματοθήκη της μέχρι να επιτευχθεί ένας μέγιστος αριθμός από ζευγαρώματα (ο μέγιστος αριθμός από ζευγαρώματα είναι μία μεταβλητή που καθορίζει ο χρήστης).

Στη συνέχεια, η βασίλισσα ψάχνει να βρει ένα μέρος για να πέσει σε χειμερία νάρκη και στον επόμενο χρόνο (ο ένας χρόνος αντιστοιχεί σε μία επανάληψη) βρίσκει ένα μέρος για να δημιουργήσει την κυψέλη της και να αρχίσει να γεννάει αυγά. Όπως αναφέραμε προηγούμενα, υπάρχουν τριών ειδών μπάμπουρες που μία βασίλισσα μπορεί να γεννήσει: νέες βασίλισσες, εργάτριες και κηφήνες. Τα πρώτα δύο είδη δημιουργούνται με διασταύρωση του γονότυπου της βασίλισσας με το γονότυπο των κηφήνων με τη χρήση ενός οποιουδήποτε τελεστή διασταύρωσης. Για να εκμεταλλευτούμε τη δυνατότητα που έχει η βασίλισσα να δημιουργήσει μία λύση χρησιμοποιώντας στοιχεία από πολλούς κηφήνες συνήθως χρησιμοποιείται ο ακόλουθος τελεστής διασταύρωσης, όπου τα σημεία επιλέγονται τυχαία από τη βασίλισσα και τυχαία από ένα από τους κηφήνες.

Ο τελεστής διασταύρωσης που χρησιμοποιείται πιο πολύ είναι ο εξής: ορίζουμε μια παράμετρο C_{r1} η οποία καθορίζει την αναλογία των σημείων που θα επιλεχθούν από τον κηφήνα και από τη βασίλισσα, συγκρίνουμε έναν τυχαίο αριθμό μεταξύ του

διαστήματος (0,1) με το C_{r1} και αν το C_{r1} είναι μεγαλύτερο ή ίσο από τον τυχαίο αριθμό, το αντίστοιχο στοιχείο επιλέγεται από τη βασίλισσα ειδάλλως επιλέγεται από τις λύσεις των κηφήνων τυχαία. Η λύση ενός απογόνου i υπολογίζεται από τον παρακάτω τύπο:

$$b_{ij}(t) = \begin{cases} q_j(t), \text{ εάν } C_j(0,1) \le d_1 \text{ αλλιώς } d_{kj}(t) \end{cases} \quad (9)$$

όπου t είναι ο αριθμός της επανάληψης, j η διάσταση του προβλήματος, $q_j(t)$ η λύση της βασίλισσας και $d_{kj}(t)$ οι λύσεις των κηφήνων.

Οι ισχυρότεροι από τους απόγονους επιλέγονται για νέες βασίλισσες και οι υπόλοιπες επιλέγονται για εργάτριες. Οι νέες βασίλισσες επιλέγονται να είναι ίσες με το μέγιστο αριθμό των βασιλισσών (παράμετρος του αλγορίθμου). Αρχικά οι νέες βασίλισσες τρέφονται μόνο από τις παλιές βασίλισσες και στη συνέχεια τόσο από τις παλιές βασίλισσες όσο και από τις εργάτριες. Όσο προχωράει η διαδικασία της τροφής τρέφονται όλο και περισσότερο από τις εργάτριες και σχεδόν καθόλου από τις βασίλισσες. Αυτή η διαδικασία εφαρμόζεται προσπαθώντας να προσομοιώσουμε την πραγματική συμπεριφορά στη φύση των μπάμπουρων που αναφέρει ότι αρχικά τρέφονται από τη βασίλισσα και στη συνέχεια τρέφονται και από τους δύο μέχρι να αρχίσει η βασίλισσα να γίνεται αδύναμη οπότε δεν μπορεί να προσφέρει σε τροφή. Όλη αυτή η διαδικασία πραγματοποιείται με την παρακάτω εξίσωση η οποία λειτουργεί ως φάση τοπικής αναζήτησης όπου κάθε νέα βασίλισσα επιλέγει ποιος θα τη θρέψει.

$$nq_{ij} = nq_{ij} + \left(b_{\max} - \frac{(b_{\max} - b_{\min}) * lsi}{lsi_{\max}} \right) * \left(nq_{ij} - q_j \right) + \frac{1}{M} * \sum_{k=1}^{M} \left(b_{\min} - \frac{(b_{\min} - b_{\max}) * lsi}{lsi_{\max}} \right) * (nq_{ij} - w_{kj}) \quad (10)$$

όπου nq_{ij} είναι η λύση της νέας βασίλισσας i, q_j η λύση της παλιάς βασίλισσας, w_{kj} η λύση των εργατριών, M ο αριθμός των εργατριών που κάθε νέα βασίλισσα επιλέγει να τη θρέψουν και διαφέρει για κάθε βασίλισσα, και $b_{\max}$, $b_{\min}$ δύο παράμετροι οι οποίες παίρνουν τιμές στο διάστημα (0,1) και από αυτές εξαρτάται εάν η βασίλισσα θα τραφεί από την παλιά βασίλισσα, από τις εργάτριες ή και από τις δύο, lsi η τρέχουσα επανάληψη της τοπικής αναζήτησης και $lsi_{\max}$ ο μέγιστος αριθμός επαναλήψεων για την τοπική αναζήτηση. Αρχικά, οι καινούριες βασίλισσες

τρέφονται μόνο από την παλιά βασίλισσα αλλά καθώς αυξάνονται οι επαναλήψεις της τοπικής αναζήτησης, μόνο οι εργάτριες τρέφουν τις καινούριες βασίλισσες. Η κατάλληλη επιλογή των τιμών $b_{\max}$ και $b_{\min}$ ελέγχει τη διαδικασία τροφής. Για να επιτευχθεί σωστά η διαδικασία που παρουσιάζουμε είναι απαραίτητη η επιλογή για το $b_{\max}$ μιας μεγάλης τιμής και για το $b_{\min}$ μιας τιμής σχεδόν ίσης με το 0. Στη συνέχεια, οι νέες βασίλισσες εγκαταλείπουν την κυψέλη.

Οι κηφήνες παράγονται με μετάλλαξη του γονότυπου των παλιών βασιλισσών ή με μετάλλαξη του γονότυπου των εργατριών. Ανάλογα με το πρόβλημα χρησιμοποιείται διαφορετικός τελεστής μετάλλαξης. Για συνεχή προβλήματα ή για δυαδικά προβλήματα θα μπορούσε να χρησιμοποιηθεί μία τυχαία αλλαγή σε κάποιο ή κάποια από τα στοιχεία του διανύσματος. Όμως, αν είχαμε ένα πιο δύσκολο πρόβλημα, όπως για παράδειγμα το πρόβλημα δρομολόγησης οχημάτων στη θέση της μετάλλαξης θα μπορούσε να χρησιμοποιηθεί μία διαδικασία τοπικής αναζήτησης. Ο αριθμός των κηφήνων ανά αποικία υπολογίζεται απότηνακόλουθη εξίσωση:

number of drones per colony = total number of drones / number of queens (11)

Οι κηφήνες, στη συνέχεια, φεύγουν από την κυψέλη και αναζητούν νέες βασίλισσες για να ζευγαρώσουν. Καθώς οι κηφήνες πετούν μακριά από την κυψέλη, κινούνται σε σχηματισμό σμήνους με στόχο να βρουν το πιο ελπιδοφόρο μέρος για να τους βρουν οι νέες βασίλισσες από τα μαρκαρισμένα μονοπάτια τους. Η κίνηση των κηφήνων μακριά από την κυψέλη υπολογίζεται από την ακόλουθη εξίσωση:

$$d_{ij} = d_{ij} + a*(d_{kj} - d_{lj}) \qquad (12)$$

όπου d_{ij}, d_{kj}, d_{lj} οι λύσεις των κηφήνων i, k, l και α μια παράμετρος από την οποία εξαρτάται το ποσοστό που ο κηφήνας i επηρεάζεται από τους κηφήνες k και l. Η νέα βασίλισσα επιλέγει τους κηφήνες με τη διαδικασία που περιγράψαμε προηγούμενα. Στην επόμενη επανάληψη, οι καλύτερα γονιμοποιημένες βασίλισσες επιβιώνουν και όλα τα υπόλοιπα μέλη του πληθυσμού πεθαίνουν.

Κεφάλαιο 3.

Ο αλγόριθμος τεχνητής αποικίας μελισσών (Artificial Bee Colony optimization algorithm, ABC) και η εφαρμογή του σε προβλήματα δρομολόγησης οχημάτων με στοχαστική ζήτηση.

3.1. Εισαγωγή

Στο κεφάλαιο αυτό θα περιγραφεί αρχικά ο μεθευρετικός αλγόριθμος τεχνητής αποικίας μελισσών, ενώ στη συνέχεια θα περιγράψουμε τον αλγόριθμο που εμείς δημιουργήσαμε και χρησιμοποιήσαμε για την επίλυση διαφόρων δεδομένων προβλημάτων δρομολόγησης οχημάτων με στοχαστική ζήτηση.

3.2 Αλγόριθμος τεχνητής αποικίας μελισσών (Artificial Bee Colony optimization algorithm, ABC)

Ο Αλγόριθμος Τεχνητής Αποικίας Μελισσών (Karaboga, Basturk, 2007 και 2008) είναι ένας αλγόριθμος βελτιστοποίησης που βασίζεται στην συμπεριφορά ενός σμήνους μελισσών. Εφαρμόζεται κυρίως σε προβλήματα συνεχούς βελτιστοποίησης και προσομοιώνει τη διαδικασία του ‘χορού των μελισσών’ (waggled dance) που ένα σμήνος από μέλισσες πραγματοποιεί κατά τη διαδικασία αναζήτησης τροφής. Αυτό που συμβαίνει στην πραγματικότητα είναι ότι οι μέλισσες φεύγουν από την κυψέλη για την αναζήτηση τροφής και όταν βρουν την τροφή, επειδή δεν μπορούν να την συλλέξουν και να την μεταφέρουν μόνες τους στην κυψέλη, επιστρέφουν στην κυψέλη, ξεφορτώνουν το νέκταρ που πήραν μαζί τους και ενημερώνουν τις υπόλοιπες μέλισσες για την πηγή τροφής που βρήκαν. Η ενημέρωση γίνεται με ειδικές κινήσεις, οι οποίες επιστημονικά ονομάζονται ‘χορός εντός της κυψέλης’, με σκοπό την ανταλλαγή πληροφοριών μεταξύ τους. Οι κυριότερες πληροφορίες είναι το πόσο πλούσια σε τροφή (νέκταρ) είναι η πηγή που βρήκαν και πόσο απέχει από την κυψέλη. Σκοπός του ‘χορού’ είναι να ενημερώσουν και να πείσουν όσο το δυνατόν περισσότερες μέλισσες να τις ακολουθήσουν στις πηγές τροφής. Η κάθε μέλισσα χορεύει σε όσο το δυνατόν περισσότερους χώρους της κυψέλης για να πείσει όσο περισσότερες μέλισσες μπορεί. Επιστημονικές μελέτες έχουν δείξει ότι όταν οι

μέλισσες χορεύουν, η κατεύθυνση που έχουν δείχνει την κατεύθυνση της πηγής τροφής σε σχέση με τον ήλιο, η ένταση των κινήσεων δείχνει την απόσταση της πηγής τροφής ενώ η χρονική διάρκεια του χορού δείχνει την ποσότητα της τροφής της πηγής.

Στον αλγόριθμο υπάρχουν τρεις ομάδες μελισσών: οι εξερευνήτριες (the employed bees), οι οποίες είναι μέλισσες που βρίσκουν την πηγή τροφής (πιθανή λύση του προβλήματος) από ένα προκαθορισμένο σύνολο από πιθανές πηγές τροφής και μοιράζονται αυτή την πληροφορία (χορός μελισσών) με τις άλλες μέλισσες στην κυψέλη, οι θεατές μέλισσες (the onlookers bees) οι οποίες είναι μέλισσες που περιμένουν στην κυψέλη και με βάση την πληροφορία που παίρνουν από τις εργάτριες αναζητούν μία καλύτερη πηγή τροφής στη γειτονιά που βρίσκεται η τροφή που τους υπέδειξαν οι εργάτριες και τέλος, οι ανιχνεύτριες μέλισσες (the scout bees) οι οποίες στην ουσία είναι εξερευνήτριες που η πηγή τροφής τους έχει τελειώσει και αναζητούν τυχαία στο χώρο λύσεων μία καινούρια πηγή τροφής. Ο αλγόριθμος της τεχνητής αποικίας μελισσών είναι ο πιο γνωστός αλγόριθμος που βασίζεται στη συμπεριφορά των μελισσών κατά τη διάρκεια της αναζήτησης τροφής και έχει εφαρμοστεί σε πολλές μελέτες (π.χ. Baykasoglu και συνεργάτες, 2007, Karaboga, Akay, 2009a και b, Ozbakir και συνεργάτες, 2010, Sabat και συνεργάτες, 2010).

Αρχικά, ένα σύνολο από πηγές τροφής (πιθανές λύσεις) επιλέγονται τυχαία από τις εξερευνήτριες μέλισσες και η διαθέσιμη ποσότητα νέκταρ τους υπολογίζεται (τιμή της αντικειμενικής συνάρτησης). Αν το πρόβλημα είναι συνεχές τότε οι αρχικές τιμές καθορίζονται τυχαία στο πεδίο τιμών που το πρόβλημα θα πρέπει να επιλυθεί. Αν όμως, έχουμε πρόβλημα συνδυαστικής βελτιστοποίησης, πρέπει να υπάρχει ένας τρόπος για να μετατραπούν οι τιμές από συνεχείς σε διακριτές. Στο άρθρο (Ι.Μαρινάκης, Μ.Μαρινάκη, Ν.Ματσατσίνης, 2009) έχει προταθεί ένας τρόπος όμοιος με τον τρόπο που έχει προταθεί για να μετατραπούν οι τιμές των μεταβλητών σε έναν αλγόριθμο βελτιστοποίησης σμήνους σωματιδίων (Shi, Eberhart, 1998).

Χρησιμοποιείται η εξίσωση:

$$sig(x_{ij}) = \frac{1}{1+\exp(-x_{ij})} \qquad (13)$$

και στη συνέχεια οι τιμές των πηγών τροφής διακριτοποιούνται σύμφωνα μετην ακόλουθη εξίσωση:

$$y_{ij} = \begin{cases} 1, & \varepsilon\acute{\alpha}\nu \ rand1 < sig(x_{ij}) \\ 0, & \varepsilon\acute{\alpha}\nu \ rand1 \geq sig(x_{ij}) \end{cases} \qquad (14)$$

όπου το x_{ij} είναι η λύση (πηγή τροφής), i = 1, ...,N (N είναι ο αριθμός των πηγών τροφής), j=1,...,d (d είναι η διάσταση του προβλήματος), y_{ij} είναι η μετασχηματισμένη ακέραια λύση (αυτή η μεταβλητή χρησιμοποιείται μόνο στην περίπτωση που θέλουμε να λύσουμε κάποιο πρόβλημα με τη χρήση δυαδικών τιμών) και rand1 είναι ένας τυχαίος αριθμός στο διάστημα (0,1).

Στη συνέχεια η τιμή της αντικειμενικής συνάρτησης υπολογίζεται και σε κάθε μία πηγή τροφής αντιστοιχίζεται μία εξερευνήτρια μέλισσα. Οι εξερευνήτριες μέλισσες επιστρέφουν στην κυψέλη και πραγματοποιούν τον αποκαλούμενο χορό της μέλισσας (waggle dance) με στόχο να ενημερώσουν τις μέλισσες που έχουν παραμείνει στην κυψέλη, τις θεατές μέλισσες, σε ποια σημεία βρίσκονται οι πηγές τροφής. Στο αρχικό άρθρο που προτάθηκε ο αλγόριθμος (Karaboga, Basturk, 2007) προτείνεται ο αριθμός των εξερευνητριών και ο αριθμός των θεατών μελισσών να είναι ίδιος, όμως αυτό δεν είναι δεσμευτικό και εξαρτάται από το πρόβλημα που θέλουμε να επιλύσουμε. Στη συνέχεια, οι θεατές μέλισσες επιλέγουν την πηγή τροφής που θα επισκεφθούν βασιζόμενες στην πληροφορία που λαμβάνουν για το νέκταρ της κάθε πηγής από τη διαδικασία του χορού της μέλισσας.

Στον επόμενο τύπο φαίνεται η πιθανότητα να επιλεγεί μια πηγή τροφής:

$$p_i = \frac{f_i}{\sum_{n=1}^{N} f_n} \qquad (15)$$

όπου f_i η τιμή της αντικειμενικής συνάρτησης για κάθε πηγή τροφής i.

Στη συνέχεια, οι εξερευνήτριες και οι θεατές μέλισσες τοποθετούνται στις επιλεγμένες πηγές, αντιστοιχίζοντας στην κάθε πηγή τροφής μία μέλισσα. Για να παραχθεί μία καινούρια πηγή τροφής χρησιμοποιείται η ακόλουθη εξίσωση (Karaboga, Basturk, 2007 και 2008):

$$x'_{ij} = x_{ij} + rand2(x_{ij} - x_{kj}) \qquad (16)$$

όπου x'_{ij} η νέα πιθανή πηγή τροφής, k μια άλλη πηγή τροφής και rand2 ένας τυχαίος αριθμός στο διάστημα (0,1). Στη συνέχεια υπολογίζεται η τιμή της αντικειμενικής συνάρτησης της κάθε πηγής τροφής. Αν για κάποια πηγή τροφής βρεθεί από τις θεατές μέλισσες με τις κινήσεις τοπικής αναζήτησης μία καλύτερη πηγή τροφής τότε η πηγή τροφής αντικαθίσταται στην μνήμη των μελισσών από την καλύτερή της.

Θα πρέπει να τονιστεί ότι αν υπάρχουν πολλές μέλισσες σε μία πηγή τροφής τότε από τις κινήσεις τοπικής αναζήτησης που πραγματοποιεί η κάθε μέλισσα, η συγκεκριμένη πηγή τροφής έχει πολύ μεγαλύτερες ικανότητες αναζήτησης σε διαφορετικά σημεία του χώρου λύσεων. Όταν λέμε ότι η πηγή τροφής έχει μεγαλύτερες ικανότητες αναζήτησης σημαίνει ότι η συνάρτηση ποιότητας της συγκεκριμένης πηγής τροφής έχει καλύτερη τιμή. Στη συνέχεια, όλες οι μέλισσες επιστρέφουν ξανά στην κυψέλη και η διαδικασία ξεκινάει από την αρχή με το χορό της μέλισσας. Εάν για έναν αριθμό επαναλήψεων η λύση δεν μπορεί να βελτιωθεί, τότε αυτή η πηγή τροφής θεωρείται ότι έχει εξαντληθεί και μία ανιχνεύτρια μέλισσα τοποθετείται σε μία καινούρια τυχαία θέση μέσα στο χώρο λύσεων (μία καινούρια πηγή τροφής). Για να βρίσκεται μέσα σε κάποιο προκαθορισμένο πεδίο τιμών η καινούρια τυχαία θέση (με στόχο να μπορεί να οδηγήσει πιο γρήγορα σε μία καλή λύση) μπορούμε να υπολογίσουμε την καινούρια θέση από τον τύπο:

$$x'_{ij} = x_{\min,j} + rand3(x_{\max,j} - x_{\min,j}) \qquad (17)$$

όπου $x_{\min,j}$ και $x_{\max,j}$ είναι η ελάχιστη και η μέγιστη τιμή που μπορεί να πάρει μία μεταβλητή στο πεδίο τιμών και rand3 ένας τυχαίος αριθμός στοδιάστημα (0,1).

Αυτό που πρέπει να τονιστεί για το συγκεκριμένο αλγόριθμο είναι ότι οι μέλισσες δεν είναι λύσεις όπως ήταν σε κάποιους άλλους αλγόριθμους που έχουμε μελετήσει ως τώρα. Οι μέλισσες είναι διαδικασίες που εφαρμόζονται σε κάποια λύση (πηγή τροφής) με στόχο να την βελτιώσουν. Άρα αν σε κάποια πηγή τροφής εμφανιστεί οτι θα ακολουθήσουν 10 μέλισσες την εξερευνήτρια που βρήκε την πηγή τροφής τότε σημαίνει ότι θα εφαρμοστούν 10 προσπάθειες για εύρεση νέας πηγής τροφής. Έτσι, ένα βασικό σημείο που πρέπει να προσεχθεί στον αλγόριθμο είναι το πλήθος των πηγών τροφής να είναι μεγαλύτερο από το σύνολο των εξερευνητριών ώστε να μπορούν να τοποθετηθούν πάνω στις πηγές τροφής χωρίς να περισσέψει κάποια μέλισσα.

3.3 Υβριδικός Αλγόριθμος τεχνητής αποικίας μελισσών για την επίλυση προβλημάτων δρομολόγησης οχημάτων με στοχαστική ζήτηση.

Αρχικά, ορίζουμε στη μεταβλητή dem, την τιμή της απόκλισης από τη ζήτηση και στη μεταβλητή problima, ποιό παράδειγμα θα τρέξουμε. Στη συνέχεια παίρνουμε τα δεδομένα του παραδείγματος από ένα αρχείο txt που είναι, το πλήθος των κόμβων **n**, η χωρητικότητα του οχήματος **Q**, το μέγιστο μήκος διαδρομής, που είναι ένας μεγάλος αριθμός, (πράγμα που σημαίνει ότι ο περιορισμός μέγιστου μήκους διαδρομής δεν επηρεάζει την επίλυση του προβλήματος), το χρόνο επισκευής του οχήματος που είναι ίσος με μηδέν, τις συντεταγμένες **x** και **y** των κόμβων (X,Y) και τη ζήτηση του κάθε κόμβου **d.** Έπειτα, υπολογίζουμε τις αποστάσεις των κόμβων και τις αποθηκεύουμε σε έναν πίνακα, οι γραμμές και οι στήλες του οποίου, είναι ίσες με το πλήθος των κόμβων. Οι αποστάσεις υπολογίζονται σύμφωνα με τον τύπο:

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \qquad (18)$$

όπου (x_1 , y_1) και (x_2 , y_2) η θέση δύο κόμβων στο επίπεδο. Στην διαγώνιο του πίνακα βάζουμε ένα μεγάλο αριθμό έτσι ώστε να μην επιλεγεί ποτέ αυτή η μετάβαση, διότι δεν υφίστανται στο πρόβλημα μας η 'μετακίνηση' από έναν κόμβο στον εαυτό του.

Συνεχίζουμε καθορίζοντας τις παραμέτρους του αλγορίθμου που είναι οι εξής:

- ο μέγιστος αριθμός επαναλήψεων του αλγορίθμου
- το πλήθος των πηγών τροφής
- το πλήθος των εξερευνητικών μελισσών
- το πλήθος των θεατών μελισσών
- δύο τυχαίους αριθμούς στο διάστημα (0,1).

Στη συνέχεια αρχικοποιούμε τις πηγές τροφής (λύσεις), υπολογίζουμε το κόστος της κάθε πηγής τροφής και τις ταξινομούμε σε αύξουσα σειρά.

Το κόστους της διαδρομής κάθε λύσης υπολογίζεται από το τέλος της διαδρομής προς την αρχή. Αρχικά, αποθηκεύουμε στη μεταβλητή p την πιθανότητα να πάρει η ζήτηση μια τιμή, η οποία (πιθανότητα) είναι άμεσα εξαρτημένη από την τιμή της απόκλισης. Έστω για παράδειγμα οτι η πραγματική ζήτηση είναι 10 και η απόκλιση είναι ±1. Η πιθανότητα p είναι ίση με 1/3 διότι η ζήτηση μπορεί να πάρει τρείς τιμές

{9, 10,11}, και προφανώς η πιθανότητα να πάρει μια από αυτές τις τιμές είναι η ίδια. Το κόστος από τον τελευταίο κόμβο στην αποθήκη δεν εξαρτάται από τη ζήτηση, συνεπώς μπορούμε να το υπολογίσουμε αμέσως. Στη συνέχεια, παίρνουμε ανά δύο τους κόμβους από το τέλος της διαδρομής προς την αρχή και υπολογίζουμε το κόστος της διαδρομής από τον ένα κόμβο στον άλλο, (α) για την περίπτωση που γυρίσει στην αποθήκη το όχημα για ανεφοδιασμό και (β) για την περίπτωση που δε γυρίσει στην αποθήκη και πάει κατευθείαν στον πελάτη. Το κόστος της διαδρομής υπολογίζεται από την αντικειμενική συνάρτηση:

$$f_j^p = c_{j,j+1} + \sum_{k,k\le q} f_{j+1}(q-k)^* p_{j+1,k} + \sum_{k,k>q} \left[2^* c_{j+1,0} + f_{j+1}(q+Q-k)^* p_{j+1,k} \right] \qquad (19)$$

$$f_j^r = c_{j,o} + c_{0,j+1} + \sum_{k=1}^{K} f_{j+1}(Q-k)^* p_{j+1,k} \qquad (20)$$

υπό τον περιορισμό,

$$f_n(q) = c_{n,0} \quad q \in \mathrm{Ln} \qquad (21)$$

Όπως έχει αναλυτικά αναφερθεί στο κεφάλαιο **1.3.1**. Αφού υπολογίσουμε τα δύο αυτά κόστη, το όχημα επιλέγει να γυρίσει ή όχι στην αποθήκη ανάλογα με το ποιό κόστος από τα δύο είναι το μικρότερο.

$$f_j(q) = \min \begin{cases} f_j^p \\ f_j^r \end{cases}$$

<u>Επαναλήψεις αλγορίθμου</u>

Βελτιώνουμε τις τριάντα καλύτερες πηγές τροφής με τους αλγορίθμους τοπικής αναζήτησης 2-opt και 3-opt (οι αλγόριθμοι 2-opt και 3-opt περιγράφονται αναλυτικά στα κεφάλαια **3.4.1** και **3.4.2** αντίστοιχα) και τις ταξινομούμε πάλι σε αύξουσα σειρά. Εφαρμόζουμε τον αλγόριθμο path relinking (ο αλγόριθμος path relinking περιγράφεται αναλυτικά στο κεφάλαιο **3.4.3**) για την καλύτερη λύση και μια τυχαία λύση από τις είκοσι πέντε καλύτερες για να βελτιώσουμε την καλύτερη λύση. Στην περίπτωση που βρεθεί καλύτερη λύση, βελτιώνουμε περαιτέρω και την καλύτερη λύση που βρήκαμε και την τυχαία λύση με την οποία κάναμε το path relinking, με τον αλγόριθμο 2-opt και ταξινομούμε και πάλι τις λύσεις σε αύξουσα σειρά. Στη συνέχεια μετατρέπουμε τις πηγές τροφής σε συνεχή μορφή διαιρώντας κάθε στοιχείο της

λύσης με το μέγιστο στοιχείο της λύσης. Ο λόγος που τις μετατρέπουμε σε συνεχή μορφή είναι για να τις χρησιμοποιήσουμε αμέσως μετά στον τύπο για τη δημιουργία νέων λύσεων. Δημιουργούμε νέες πηγές τροφής και εάν το κόστος τους είναι καλύτερο από το προηγούμενο τις κρατάμε. Βελτιώνουμε την καλύτερη λύση με τον αλγόριθμο 2-opt και τον αλγόριθμο path relinking για την καλύτερη λύση και μια τυχαία λύση από τις νέες που δημιουργήσαμε. Αν με τις παραπάνω βελτιώσεις έχει βρεθεί κάποια καλύτερη λύση, τότε ταξινομολύμε τις λύσεις σε αύξουσα σειρά. Υπολογίζουμε νέες λύσεις εάν για δέκα συνεχόμενες επαναλήψεις του αλγορίθμου το κόστος τους δεν έχει βελτιωθεί και εάν βρεθεί κάποια καλύτερη την αντικάθιστουμε με την παλιά. Στην περίπτωση που έστω και για μια λύση βρέθηκε καλύτερη ταξινομούμε τις λύσεις σε αύξουσα σειρά. Εφαρμόζουμε τον αλγόριθμο path relinking για την καλύτερη λύση και μια τυχαία πηγή τροφής για να βελτιώσουμε την καλύτερη λύση. Στην περίπτωση που βρεθεί καλύτερη λύση, βελτιώνουμε περαιτέρω και την καλύτερη λύση που βρήκαμε και την τυχαία λύση με την οποία κάναμε το path relinking. Αν βρέθηκε καλύτερη λύση από το προηγούμενο path relinking ταξινομούμε τις λύσεις σε αύξουσα σειρά. Βελτιώνουμε τις τριάντα καλύτερες λύσεις με τον αλγόριθμο 2-opt και 3-opt. Αν έστω και μια λύση βελτιώθηκε ταξινομούμε τις λύσεις σε αύξουσα σειρά. Εφαρμόζουμε τον αλγόριθμο path relinking για την καλύτερη λύση και μια τυχαία λύση για να βελτιώσουμε την καλύτερη λύση και ξαναταξινομούμε τις λύσεις αν βρεθεί καλύτερη λύση. Με τους αλγορίθμους 2-opt και 3-opt βελτιώνουμε την καλύτερη λύση και ταξινομούμε τις λύσεις αν χρειάζεται. Σε αυτό το σημείο κοιτάμε εάν στην επανάληψη που βρισκόμαστε βελτιώθηκε το κόστος της καλύτερης λύσης και αν βελτιώθηκε κρατάμε τον αριθμό της επανάληψης έτσι ώστε στο τέλος των επαναλήψεων να ξέρουμε σε ποιά επανάληψη έγινε η τελευταία βελτίωση στην τιμή του κόστους. Μετά γίνεται η άθροιση των επαναλήψεων για τις οποίες δε γίνεται βελτίωση του κόστους στις δέκα καλύτερες πηγές τροφής. Τέλος αυξάνεται ο μετρητής των επαναλήψεων.

Κριτήριο σύγκλισης αλγορίθμου

Ο αλγόριθμος περατώνεται εφόσον ολοκληρωθεί το πλήθος των επαναλήψεων του. Ο αλγόριθμος επιστρέφει το βέλτιστο κόστος, τη βέλτιστη διαδρομή, την επανάληψη στην οποία βρέθηκε το βέλτιστο κόστος και το διάγραμμα της βέλτιστης διαδρομής στο επίπεδο, στο οποίο δε φαίνεται πότε το όχημα γυρίζει στην αποθήκη για να είναι το διάγραμμα πιο ευδιάκριτο.

Ψευδοκώδικας

Επιλογή της απόκλισης από τη ζήτηση (dem) και παραδείγματος που θα επιλυθεί

Αποθήκευση δεδομένων από αρχείο txt

Υπολογισμός του κόστους μεταβάσεων από κόμβο σε κόμβο

Καθορισμός παραμέτρων

Αρχικοποίηση πηγών τροφής

Ταξινόμηση πηγών τροφής σε αύξουσα σειρά με βάση το κόστος τους

Do until (δεν έχει φτάσει ο μέγιστος αριθμός επαναλήψεων)

Βελτίωση 30 καλύτερων πηγών τροφής με τον αλγόριθμο two opt και three opt

if βελτιώθηκε έστω και μια πηγή τροφής

Ταξινόμηση πηγών τροφής σε αύξουσα σειρά με βάση το κόστος τους

endif

Βελτίωση καλύτερης πηγής τροφής με Path Relinking

Βελτίωση 30 καλύτερων πηγών τροφής με τον αλγόριθμο three opt

if βελτιώθηκε έστω και μια πηγή τροφής

Ταξινόμηση πηγών τροφής σε αύξουσα σειρά με βάση το κόστος τους

endif

Μετατροπή πηγών τροφής σε συνεχή μορφή

Δημιουργία νέων πηγών τροφής

Βελτίωση καλύτερης πηγής τροφής από τις νέες με τον αλγόριθμο two opt

Βελτίωση καλύτερης πηγής τροφής με Path Relinking

if βελτιώθηκε έστω και μια πηγή τροφής

Ταξινόμηση πηγών τροφής σε αύξουσα σειρά με βάση το κόστος τους

endif

Δημιουργία νέων λύσεων

if βελτιώθηκε έστω και μια πηγή τροφής

Ταξινόμηση πηγών τροφής σε αύξουσα σειρά με βάση το κόστος τους

endif

Βελτίωση καλύτερης πηγής τροφής με Path Relinking

if βελτιώθηκε έστω και μια πηγή τροφής

Ταξινόμηση πηγών τροφής σε αύξουσα σειρά με βάση το κόστος τους

endif

Βελτίωση 30 καλύτερων πηγών τροφής με τον αλγόριθμο two opt και three opt

if βελτιώθηκε έστω και μια πηγή τροφής

Ταξινόμηση πηγών τροφής σε αύξουσα σειρά με βάση το κόστος τους

endif

Βελτίωση καλύτερης πηγής τροφής με Path Relinking

if βελτιώθηκε έστω και μια πηγή τροφής

Ταξινόμηση πηγών τροφής σε αύξουσα σειρά με βάση το κόστος τους

endif

Βελτίωση καλύτερης πηγής τροφής με τον αλγόριθμο two opt

Βελτίωση όλων των πηγών τροφής από τις νέες με τον αλγόριθμο two opt

if βελτιώθηκε έστω και μια πηγή τροφής

Ταξινόμηση πηγών τροφής σε αύξουσα σειρά με βάση το κόστος τους

endif

Αποθήκευση επανάληψης στην οποία βρέθηκε καλύτερη λύση

Άθροιση επαναλήψεων για τις οποίες δε γίνεται βελτίωση του κόστους για τις δέκα καλύτερες λύσεις

enddo

Επιστροφή βέλτιστου σωματιδίου, βέλτιστου κόστους, επανάληψης που βρέθηκε το βέλτιστο και διαγράμματος βέλτιστου σωματιδίου

3.4 Αλγόριθμοι τοπικής αναζήτησης

Στην Επιστήμη Υπολογιστών, ένας αλγόριθμος τοπικής αναζήτησης *(local search algorithm)* είναι μία μεθευρετική *(metaheuristic)* μέθοδος για την επίλυση υπολογιστικά δύσκολων προβλημάτων βελτιστοποίησης. Η τοπική αναζήτηση μπορεί να χρησιμοποιηθεί σε προβλήματα που μπορούν να μοντελοποιηθούν ως την εύρεση μίας λύσης που να μεγιστοποιεί κάποιο κριτήριο ανάμεσα σε έναν αριθμό υποψήφιων λύσεων. Οι αλγόριθμοι τοπικής αναζήτησης μετακινούνται από λύση σε λύση στο χώρο των υποψήφιων λύσεων (χώρος αναζήτησης, *searchspace*) μέσω τη εφαρμογής τοπικών αλλαγών, ώσπου να ικανοποιηθεί κάποιο κριτήριο τερματισμού.

Ένας αλγόριθμος τοπικής αναζήτησης ξεκινάει από μία υποψήφια λύση και μετακινείται επαναληπτικά προς μία γειτονική λύση. Αυτό είναι δυνατόν μόνο αν έχει οριστεί μία σχέση γειτνίασης στο χώρο αναζήτησης λύσεων. Η επιλογή της γειτονικής λύσης προς την οποία θα μετακινηθεί γίνεται λαμβάνοντας υπόψη τις πληροφορίες για τη γειτονιά λύσεων της τρέχουσας, γι’ αυτό και ο όρος ‘τοπική αναζήτηση’. Όταν η επιλογή της γειτονικής λύσης γίνεται μέσω της τοπικής μεγιστοποίησης του εκάστοτε κριτηρίου, η μεθευρετική αυτή μέθοδος παίρνει το όνομα αναρρίχηση λόφων *(hill climbing)*. Όταν δεν παρίστανται καλύτερες λύσεις στη γειτονιά, ο αλγόριθμος τοπικής αναζήτησης έχει κολλήσει σε τοπικά βέλτιστο σημείο.

Παρακάτω θα δοθούν επιγραμματικά τρεις αλγόριθμοι τοπικής αναζήτησης, όπως χρησιμοποιήθηκαν για τους σκοπούς της εργασίας.

3.4.1 Αλγόριθμος 2-opt

Για να εφαρμόσουμε τον αλγόριθμο τοπικής αναζήτησης 2-opt (Leonora Bianchi, Marco Dorigo, Luca Maria Gambardella 2003), θα πρέπει να έχουμε δημιουργήσει με κάποιο τρόπο μία αρχική λύση, πάνω στην οποία θα εφαρμοστεί ο αλγόριθμος. Συνήθως δημιουργούμε μία τυχαία αρχική λύση, μία διαδρομή δηλαδή που αποτελείται από συνεχόμενους κόμβους. Στη συνέχεια επιλέγουμε δύο τυχαίους κόμβους της λύσης αυτής και αλλάζουμε τη φορά των κόμβων που βρίσκονται ανάμεσα τους, διατηρώντας την υπόλοιπη διαδρομή αναλλοίωτη.

Παράδειγμα 2-opt

Λύση πριν την εφαρμογή 2-opt:

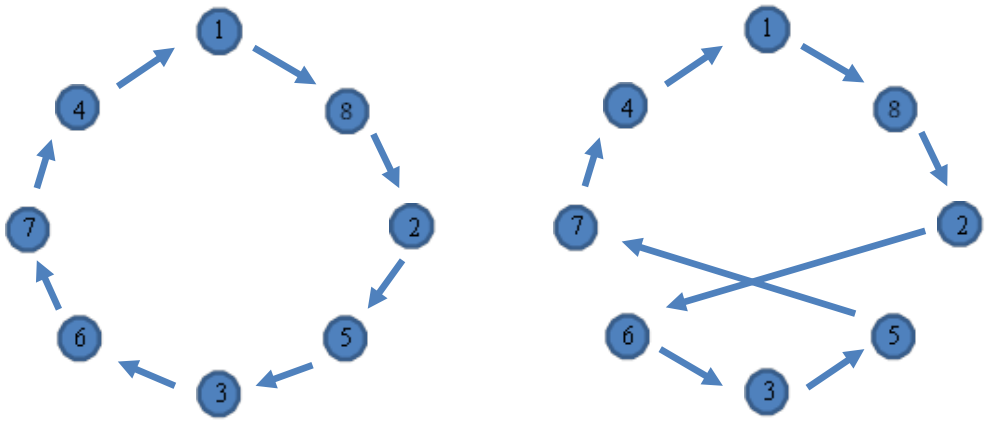
1 8 2 5 3 6 7 4

Επιλέγουμε τον τρίτο κόμβο (2) και τον έβδομο κόμβο (7) της παραπάνω διαδρομής.

Λύση μετά την εφαρμογή 2-opt:

1 8 2 **6 3 5** 7 4

Στο παρακάτω σχήμα φαίνονται οι αλλαγές που πραγματοποιούνται κατά την εφαρμογή του αλγοριθμου 2-opt στο συγκεκριμένο παράδειγμα που αναφέραμε. Στα αριστερά είναι η λύση πριν την εφαρμογή, ενώ αντίστοιχα στα δεξιά είναι η λύση μετά την εφαρμογή.



Σχήμα 1. Εφαρμογή του αλγόριθμου τοπικής αναζήτησης 2-opt

3.4.2 Αλγόριθμος 3-opt

Η μέθοδος 3-opt είναι όμοια με την 2-opt, αλλά εδώ επιλέγουμε τρεις τυχαίους κόμβους της λύσης αντί για δύο. Στη συνέχεια, αλλάζουμε τη φορά των κόμβων που βρίσκονται μεταξύ των δύο πρώτων κόμβων που επιλέξαμε και στη διαδρομή που προκύπτει αλλάζουμε τη φορά των κόμβων μεταξύ του δεύτερου και του τρίτου επιλεγμένου κόμβου, διατηρώντας πάντα, όπως και στη μέθοδο 2-opt την υπόλοιπη διαδρομή αναλλοίωτη.

Παράδειγμα 3-opt:

Λύση πριν την εφαρμογή 3-opt:

1 8 2 5 3 6 7 4

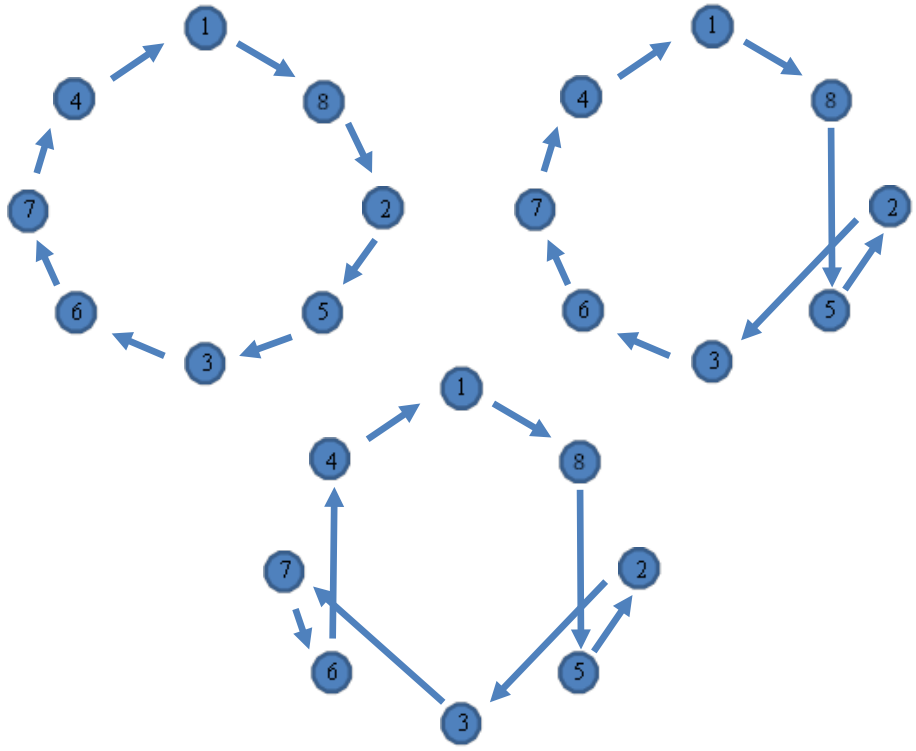
Επιλέγουμε το δεύτερο κόμβο (8), τον πέμπτο κόμβο (3) και τον όγδοο κόμβο (4) της παραπάνω διαδρομής. Στην πρώτη αλλαγή των κόμβων μεταξύ του δεύτερου και του πέμπτου κόμβου, προκύπτει η διαδρομή:

1 8 5 2 3 6 7 4

Λύση μετά την εφαρμογή 3-opt:

1 8 **5 2** 3 **7 6** 4

Στο παρακάτω σχήμα φαίνονται οι αλλαγές που πραγματοποιούνται κατά την εφαρμογή του αλγοριθμου 3-opt στο συγκεκριμένο παράδειγμα που αναφέραμε. Πάνω αριστερά βλέπουμε την αρχική λύση, ενώ αντίστοιχα πάνω δεξιά είναι η λύση μετά το πρώτο βήμα του αλγορίθμου. Στη συνέχεια απεικονίζεται η λύση μετά την εφαρμογή του αλγορίθμου.



Σχήμα 2. Εφαρμογή του αλγόριθμου τοπικής αναζήτησης 3-opt

3.4.3 Αλγόριθμος path relinking

Στον αλγόριθμο path relinking παίρνουμε δύο λύσεις και από αυτές δημιουργούμε νέες λύσεις, συνδυάζοντας αυτές τις δύο λύσεις, υπολογίζοντας κάθε φορά και το κόστος τους. Παίρνουμε τη δεύτερη λύση και αλλάζουμε κάθε φορά τη θέση δύο κόμβων, μέχρι να γίνει ίδια με την πρώτη λύση. Συγκεκριμένα για κάθε κόμβο της πρώτης λύσης, από το δεύτερο και μετά, τον τοποθετούμε στην δεύτερη λύση, στην ίδια θέση που βρισκόταν στην πρώτη λύση. Μετά από αυτή τη διαδικασία θα έχουμε τόσες λύσεις όσες και το πλήθος των κόμβων της διαδρομής μείον δύο. Στο τέλος συγκρίνουμε το κόστος της πρώτης λύσης με τα κόστη των νέων λύσεων και αν κάποια από τις νέες λύσεις είναι καλύτερη, αντικαθιστούμε την αρχική λύση με τη νέα λύση.

Παράδειγμα path relinking:

Λύσεις πριν την εφαρμογή path relinking:

$1^{η}$ λύση: 1 8 2 5 3 6 7 4

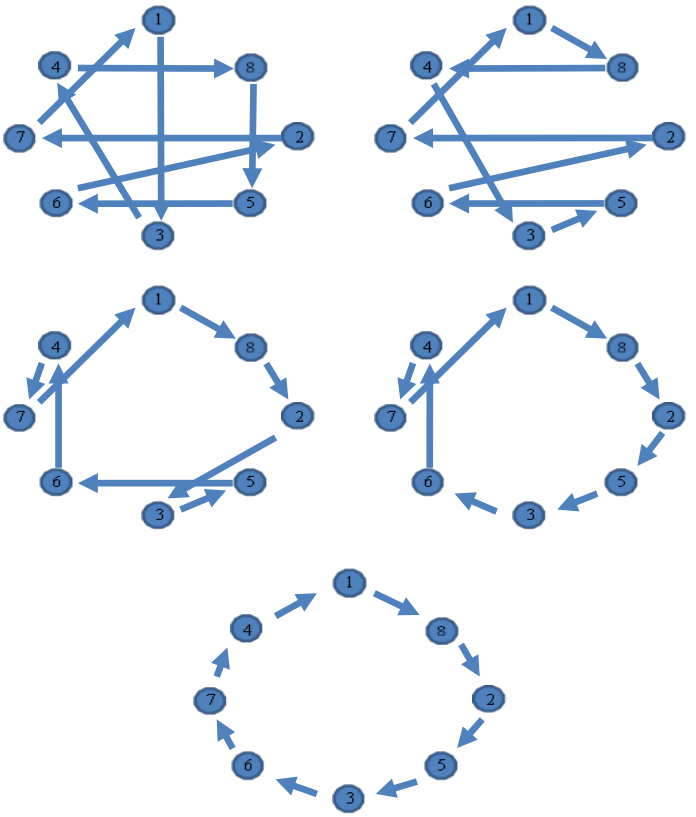
$2^{η}$ λύση: 1 3 4 8 5 6 2 7

Οι λύσεις που θα δημιουργήσει η μέθοδος path relinking είναι:

1 **8** 4 **3** 5 6 2 7
1 8 **2** 3 5 6 **4** 7
1 8 2 **5** **3** 6 4 7
1 8 2 5 **3** 6 4 7
1 8 2 5 3 **6** 4 7
1 8 2 5 3 6 **7** **4**

Όπως παρατηρούμε, στο συγκεκριμένο παράδειγμα, προκύπτουν κάποιες όμοιες λύσεις. Αυτό συμβαίνει γιατί στο τρίτο βήμα της εφαρμογής, τοποθετώντας τον τέταρτο κόμβο στην νέα θέση, δημιουργείται μία λύση στην οποία οι επόμενοι δύο κόμβοι (ο πέμπτος και ο έκτος) είναι στην ίδια θέση που εμφανίζονται και στην πρώτη λύση, οπότε στα επόμενα δύο βήματα του αλγορίθμου, δεν υπάρχει αλλαγή θέσεων για τους κόμβους της λύσης.

Στο παρακάτω σχήμα φαίνονται οι αλλαγές βήμα προς βήμα που πραγματοποιούνται κατά την εφαρμογή του αλγοριθμου path relinking στο συγκεκριμένο παράδειγμα που αναφέραμε. Πάνω αριστερά ξεκινάμε με την δεύτερη λύση, ενώ μετά την ολοκλήρωση της εφαρμογής καταλήγουμε στην πρώτη λύση.



Σχήμα 3. Εφαρμογή του αλγόριθμου τοπικής αναζήτησης path relinking

Κεφάλαιο 4.

Εφαρμογή του αλγόριθμου ABC σε προβλήματα δρομολόγησης οχημάτων με στοχαστική ζήτηση

4.1 Εισαγωγή

Στο κεφάλαιο αυτό θα παρουσιάσουμε αρχικά τα προβλήματα δρομολόγησης οχημάτων με στοχαστική ζήτηση που επιλύσαμε με τον αλγόριθμο ABC και στη συνέχεια θα ακολουθήσουν πίνακες και διαγράμματα με τα αποτελέσματα.

4.2 Παρουσίαση των προβλημάτων

Καθώς σε επόμενο κεφάλαιο τα αποτελέσματα του αλγόριθμου ABC, θα συγκριθούν με αυτά άλλων αλγορίθμων για τα ίδια προβλήματα, θεωρήσαμε χρήσιμο να κάνουμε χρήση των ίδιων συντομογραφιών που έχουν χρησιμοποιηθεί σε προηγούμενες εργασίες για να υπάρχει απόλυτη ταύτιση. Έτσι, το πρόβλημα/παράδειγμα θα αναφέρεται ως par, το πλήθος των κόμβων θα συμβολίζεται με n, η χωρητικότητα του οχήματος με Q, ενώ η απόκλιση από τη ζήτηση, θα αναφέρεται ως dem. Τα προβλήματα δρομολόγησης οχημάτων με στοχαστική ζήτηση που επιλύσαμε με τον αλγόριθμο ABC, είναι επτά στον αριθμό και είναι προβλήματα τα οποία έχουν επιλυθεί και με άλλους αλγόριθμους, όπως αναφέραμε αναλυτικά στο κεφάλαιο **2.4.**Το κάθε πρόβλημα, το επιλύσαμε, αναφορικά με την απόκλιση από τη ζήτηση, για dem={0,1,2}. Τα δεδομένα των προβλημάτων, όπως είναι ο αριθμός κόμβων, οι συντεταγμένες αυτών, η χωρητικότητα του οχήματος, η ζήτηση, καθώς και οι βέλτιστες τιμές που έχουν βρεθεί μέχρι σήμερα, παρουσίαζονται παρακάτω.

Παράδειγμα (par)	Αριθμός κόβμων	Χωρητικότητα οχήματος	Απόκλιση από τη ζήτηση (dem)	Βέλτιστο κόστος
1	51	116	0	524.61
			1	528.57
			2	531.54
2	76	114	0	836.07
			1	840.35
			2	848.73
3	101	120	0	837.76
			1	836.88
			2	842.45
4	151	120	0	1074.40
			1	1068.20
			2	1065.40
5	200	120	0	1379.80
			1	1370.20
			2	1387.00
6	121	120	0	1054.10
			1	1075.30
			2	1095.30
7	101	120	0	819.56
			1	851.47
			2	858.85

Πίνακας 1. Τα δεδομένα των προβλημάτων/παραδειγμάτων

Στους παρακάτω πίνακες καταγράφονται οι συντεταγμένες των κόμβων των προβλημάτων:

Παράδειγμα 1.

Κόμβος	X	Y	Ζήτηση
1	30	40	0
2	37	52	7
3	49	49	30
4	52	64	16
5	20	26	9
6	40	30	21
7	21	47	15
8	17	63	19
9	31	62	23
10	52	33	11
11	51	21	5
12	42	41	19
13	31	32	29
14	5	25	23
15	12	42	21
16	36	16	10
17	52	41	15
18	27	23	3
19	17	33	41
20	13	13	9
21	57	58	28
22	62	42	8
23	42	57	8
24	16	57	16
25	8	52	10

Κόμβος	X	Y	Ζήτηση
26	7	38	28
27	27	68	7
28	30	48	15
29	43	67	14
30	58	48	6
31	58	27	19
32	37	69	11
33	38	46	12
34	46	10	23
35	61	33	26
36	62	63	17
37	63	69	6
38	32	22	9
39	45	35	15
40	59	15	14
41	5	6	7
42	10	17	27
43	21	10	13
44	5	64	11
45	30	15	16
46	39	10	10
47	32	39	5
48	25	32	25
49	25	55	17
50	48	28	18
51	56	37	10

Παράδειγμα 2.

Κόμβος	X	Y	Ζήτηση
1	40	40	0
2	22	22	18
3	36	26	26
4	21	45	11
5	45	35	30
6	55	20	21
7	33	34	19
8	50	50	15
9	55	45	16
10	26	59	29
11	40	66	26
12	55	65	37
13	35	51	16
14	62	35	12
15	62	57	31
16	62	24	8
17	21	36	19
18	33	44	20
19	9	56	13
20	62	48	15
21	66	14	22
22	44	13	28
23	26	13	12
24	11	28	6
25	7	43	27

Κόμβος	X	Y	Ζήτηση
26	17	64	14
27	41	46	18
28	55	34	17
29	35	16	29
30	52	26	13
31	43	26	22
32	31	76	25
33	22	53	28
34	26	29	27
35	50	40	19
36	55	50	10
37	54	10	12
38	60	15	14
39	47	66	24
40	30	60	16
41	30	50	33
42	12	17	15
43	15	14	11
44	16	19	18
45	21	48	17
46	50	30	21
47	51	42	27
48	50	15	19
49	48	21	20
50	12	38	5

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

Παράδειγμα 3.

Κόμβος	X	Y	Ζήτηση
1	35	35	0
2	41	49	10
3	35	17	7
4	55	45	13
5	55	20	19
6	15	30	26
7	25	30	3
8	20	50	5
9	10	43	9
10	55	60	16
11	30	60	16
12	20	65	12
13	50	35	19
14	30	25	23
15	15	10	20
16	30	5	8
17	10	20	19
18	5	30	2
19	20	40	12
20	15	60	17
21	45	65	9
22	45	20	11
23	45	10	18
24	55	5	29
25	65	35	3
26	65	20	6
27	45	30	17
28	35	40	16
29	41	37	16
30	64	42	9
31	40	60	21
32	31	52	27
33	35	69	23
34	53	52	11
35	65	55	14

Κόμβος	X	Y	Ζήτηση
36	63	65	8
37	2	60	5
38	20	20	8
39	5	5	16
40	60	12	31
41	40	25	9
42	42	7	5
43	24	12	5
44	23	3	7
45	11	14	18
46	6	38	16
47	2	48	1
48	8	56	27
49	13	52	36
50	6	68	30
51	47	47	13
52	49	58	10
53	27	43	9
54	37	31	14
55	57	29	18
56	63	23	2
57	53	12	6
58	32	12	7
59	36	26	18
60	21	24	28
61	17	34	3
62	12	24	13
63	24	58	19
64	27	69	10
65	15	77	9
66	62	77	20
67	49	73	25
68	67	5	25
69	56	39	36
70	37	47	6

Κόμβος	X	Y	Ζήτηση
71	37	56	5
72	57	68	15
73	47	16	25
74	44	17	9
75	46	13	8
76	49	11	18
77	49	42	13
78	53	43	14
79	61	52	3
80	57	48	23
81	56	37	6
82	55	54	26
83	15	47	16
84	14	37	11
85	11	31	7
86	16	22	41
87	4	18	35
88	28	18	26
89	26	52	9
90	26	35	15
91	31	67	3
92	15	19	1
93	22	22	2
94	18	24	22
95	26	27	27
96	25	24	20
97	22	27	11
98	25	21	12
99	19	21	10
100	20	26	9
101	18	18	17

Παράδειγμα 4.

Κόμβος	X	Y	Ζήτηση
1	35	35	0
2	41	49	10
3	35	17	7
4	55	45	13
5	55	20	19
6	15	30	26
7	25	30	3
8	20	50	5
9	10	43	9
10	55	60	16
11	30	60	16
12	20	65	12
13	50	35	19
14	30	25	23
15	15	10	20
16	30	5	8
17	10	20	19
18	5	30	2
19	20	40	12
20	15	60	17
21	45	65	9
22	45	20	11
23	45	10	18
24	55	5	29
25	65	35	3
26	65	20	6
27	45	30	17
28	35	40	16
29	41	37	16
30	64	42	9
31	40	60	21
32	31	52	27
33	35	69	23
34	53	52	11
35	65	55	14

Κόμβος	X	Y	Ζήτηση
36	63	65	8
37	2	60	5
38	20	20	8
39	5	5	16
40	60	12	31
41	40	25	9
42	42	7	5
43	24	12	5
44	23	3	7
45	11	14	18
46	6	38	16
47	2	48	1
48	8	56	27
49	13	52	36
50	6	68	30
51	47	47	13
52	49	58	10
53	27	43	9
54	37	31	14
55	57	29	18
56	63	23	2
57	53	12	6
58	32	12	7
59	36	26	18
60	21	24	28
61	17	34	3
62	12	24	13
63	24	58	19
64	27	69	10
65	15	77	9
66	62	77	20
67	49	73	25
68	67	5	25
69	56	39	36
70	37	47	6

Κόμβος	X	Y	Ζήτηση
71	37	56	5
72	57	68	15
73	47	16	25
74	44	17	9
75	46	13	8
76	49	11	18
77	49	42	13
78	53	43	14
79	61	52	3
80	57	48	23
81	56	37	6
82	55	54	26
83	15	47	16
84	14	37	11
85	11	31	7
86	16	22	41
87	4	18	35
88	28	18	26
89	26	52	9
90	26	35	15
91	31	67	3
92	15	19	1
93	22	22	2
94	18	24	22
95	26	27	27
96	25	24	20
97	22	27	11
98	25	21	12
99	19	21	10
100	20	26	9
101	18	18	17
102	37	52	7
103	49	49	30
104	52	64	16
105	20	26	9
106	40	30	21
107	21	47	15
108	17	63	19
109	31	62	23
110	52	33	11

Κόμβος	X	Y	Ζήτηση
111	51	21	5
112	42	41	19
113	31	32	29
114	5	25	23
115	12	42	21
116	36	16	10
117	52	41	15
118	27	23	3
119	17	33	41
120	13	13	9
121	57	58	28
122	62	42	8
123	42	57	8
124	16	57	16
125	8	52	10
126	7	38	28
127	27	68	7
128	30	48	15
129	43	67	14
130	58	48	6
131	58	27	19
132	37	69	11
133	38	46	12
134	46	10	23
135	61	33	26
136	62	63	17
137	63	69	6
138	32	22	9
139	45	35	15
140	59	15	14
141	5	6	7
142	10	17	27
143	21	10	13
144	5	64	11
145	30	15	16
146	39	10	10
147	32	39	5
148	25	32	25
149	25	55	17
150	48	28	18
151	56	37	10

Παράδειγμα 5.

Κόμβος	X	Y	Ζήτηση
1	35	35	0
2	41	49	10
3	35	17	7
4	55	45	13
5	55	20	19
6	15	30	26
7	25	30	3
8	20	50	5
9	10	43	9
10	55	60	16
11	30	60	16
12	20	65	12
13	50	35	19
14	30	25	23
15	15	10	20
16	30	5	8
17	10	20	19
18	5	30	2
19	20	40	12
20	15	60	17
21	45	65	9
22	45	20	11
23	45	10	18
24	55	5	29
25	65	35	3
26	65	20	6
27	45	30	17
28	35	40	16
29	41	37	16
30	64	42	9
31	40	60	21
32	31	52	27
33	35	69	23
34	53	52	11
35	65	55	14

Κόμβος	X	Y	Ζήτηση
36	63	65	8
37	2	60	5
38	20	20	8
39	5	5	16
40	60	12	31
41	40	25	9
42	42	7	5
43	24	12	5
44	23	3	7
45	11	14	18
46	6	38	16
47	2	48	1
48	8	56	27
49	13	52	36
50	6	68	30
51	47	47	13
52	49	58	10
53	27	43	9
54	37	31	14
55	57	29	18
56	63	23	2
57	53	12	6
58	32	12	7
59	36	26	18
60	21	24	28
61	17	34	3
62	12	24	13
63	24	58	19
64	27	69	10
65	15	77	9
66	62	77	20
67	49	73	25
68	67	5	25
69	56	39	36
70	37	47	6

Κόμβος	X	Y	Ζήτηση
71	37	56	5
72	57	68	15
73	47	16	25
74	44	17	9
75	46	13	8
76	49	11	18
77	49	42	13
78	53	43	14
79	61	52	3
80	57	48	23
81	56	37	6
82	55	54	26
83	15	47	16
84	14	37	11
85	11	31	7
86	16	22	41
87	4	18	35
88	28	18	26
89	26	52	9
90	26	35	15
91	31	67	3
92	15	19	1
93	22	22	2
94	18	24	22
95	26	27	27
96	25	24	20
97	22	27	11
98	25	21	12
99	19	21	10
100	20	26	9

Κόμβος	X	Y	Ζήτηση
101	18	18	17
102	37	52	7
103	49	49	30
104	52	64	16
105	20	26	9
106	40	30	21
107	21	47	15
108	17	63	19
109	31	62	23
110	52	33	11
111	51	21	5
112	42	41	19
113	31	32	29
114	5	25	23
115	12	42	21
116	36	16	10
117	52	41	15
118	27	23	3
119	17	33	41
120	13	13	9
121	57	58	28
122	62	42	8
123	42	57	8
124	16	57	16
125	8	52	10
126	7	38	28
127	27	68	7
128	30	48	15
129	43	67	14
130	58	48	6

Κόμβος	X	Y	Ζήτηση
131	58	27	19
132	37	69	11
133	38	46	12
134	46	10	23
135	61	33	26
136	62	63	17
137	63	69	6
138	32	22	9
139	45	35	15
140	59	15	14
141	5	6	7
142	10	17	27
143	21	10	13
144	5	64	11
145	30	15	16
146	39	10	10
147	32	39	5
148	25	32	25
149	25	55	17
150	48	28	18
151	56	37	10
152	22	22	18
153	36	26	26
154	21	45	11
155	45	35	30
156	55	20	21
157	33	34	19
158	50	50	15
159	55	45	16
160	26	59	29
161	40	66	26
162	55	65	37
163	35	51	16
164	62	35	12
165	62	57	31

Κόμβος	X	Y	Ζήτηση
166	62	24	8
167	21	36	19
168	33	44	20
169	9	56	13
170	62	48	15
171	66	14	22
172	44	13	28
173	26	13	12
174	11	28	6
175	7	43	27
176	17	64	14
177	41	46	18
178	55	34	17
179	35	16	29
180	52	26	13
181	43	26	22
182	31	76	25
183	22	53	28
184	26	29	27
185	50	40	19
186	55	50	10
187	54	10	12
188	60	15	14
189	47	66	24
190	30	60	16
191	30	50	33
192	12	17	15
193	15	14	11
194	16	19	18
195	21	48	17
196	50	30	21
197	51	42	27
198	50	15	19
199	48	21	20
200	12	38	5

Παράδειγμα 6.

Κόμβος	X	Y	Ζήτηση
1	10	45	0
2	25	1	25
3	25	3	7
4	31	5	13
5	32	5	6
6	31	7	14
7	32	9	5
8	34	9	11
9	46	9	19
10	35	7	5
11	34	6	15
12	35	5	15
13	47	6	17
14	40	5	13
15	39	3	12
16	36	3	18
17	73	6	13
18	73	8	18
19	24	36	12
20	76	6	17
21	76	10	4
22	76	13	7
23	78	3	12
24	78	9	13
25	79	3	8
26	79	5	16
27	79	11	15
28	82	3	6
29	82	7	5
30	90	15	9

Κόμβος	X	Y	Ζήτηση
31	84	3	11
32	84	5	10
33	84	9	3
34	85	1	7
35	87	5	2
36	85	8	4
37	87	7	4
38	86	41	18
39	86	44	14
40	86	46	12
41	85	55	17
42	89	43	20
43	89	46	14
44	89	52	16
45	92	42	10
46	92	52	9
47	94	42	11
48	94	44	7
49	94	48	13
50	96	42	5
51	99	46	4
52	99	50	21
53	83	80	13
54	83	83	11
55	85	81	12
56	85	85	14
57	85	89	10
58	87	80	8
59	87	86	16
60	90	77	19

Κόμβος	X	Y	Ζήτηση
61	90	88	5
62	93	82	17
63	93	84	7
64	93	89	16
65	94	86	14
66	95	80	17
67	99	89	13
68	37	83	17
69	50	80	13
70	35	85	14
71	35	87	16
72	44	86	7
73	46	89	13
74	46	83	9
75	46	87	11
76	46	89	35
77	48	83	5
78	50	85	28
79	50	88	7
80	54	86	3
81	54	90	10
82	10	35	7
83	10	40	12
84	18	30	11
85	17	35	10
86	16	38	8
87	14	40	11
88	15	42	21
89	11	42	4
90	18	40	15

Κόμβος	X	Y	Ζήτηση
91	21	39	16
92	20	40	4
93	18	41	16
94	20	44	7
95	22	44	10
96	16	45	9
97	20	45	11
98	25	45	17
99	30	55	12
100	20	50	11
101	22	51	7
102	18	49	9
103	16	48	11
104	20	55	12
105	18	53	7
106	14	50	8
107	15	51	6
108	16	54	5
109	28	33	12
110	33	38	13
111	30	50	7
112	13	40	7
113	15	36	8
114	18	31	11
115	25	37	13
116	30	46	11
117	25	52	10
118	16	33	7
119	25	35	4
120	5	40	20
121	5	50	13

Παράδειγμα 7.

Κόμβος	X	Y	Ζήτηση
1	40	50	0
2	45	68	10
3	45	70	30
4	42	66	10
5	42	68	10
6	42	65	10
7	40	69	20
8	40	66	20
9	38	68	20
10	38	70	10
11	35	66	10
12	35	69	10
13	25	85	20
14	22	75	30
15	22	85	10
16	20	80	40
17	20	85	40
18	18	75	20
19	15	75	20
20	15	80	10
21	30	50	10
22	30	52	20
23	28	52	20
24	28	55	10
25	25	50	10
26	25	52	40
27	25	55	10
28	23	52	10
29	23	55	20
30	20	50	10
31	20	55	10
32	10	35	20
33	10	40	30
34	8	40	40
35	8	45	20

Κόμβος	X	Y	Ζήτηση
36	5	35	10
37	5	45	10
38	2	40	20
39	0	40	30
40	0	45	20
41	35	30	10
42	35	32	10
43	33	32	20
44	33	35	10
45	32	30	10
46	30	30	10
47	30	32	30
48	30	35	10
49	28	30	10
50	28	35	10
51	26	32	10
52	25	30	10
53	25	35	10
54	44	5	20
55	42	10	40
56	42	15	10
57	40	5	30
58	40	15	40
59	38	5	30
60	38	15	10
61	35	5	20
62	50	30	10
63	50	35	20
64	50	40	50
65	48	30	10
66	48	40	10
67	47	35	10
68	47	40	10
69	45	30	10
70	45	35	10

Κόμβος	X	Y	Ζήτηση
71	95	30	30
72	95	35	20
73	53	30	10
74	92	30	10
75	53	35	50
76	45	65	20
77	90	35	10
78	88	30	10
79	88	35	20
80	87	30	10
81	85	25	10
82	85	35	30
83	75	55	20
84	72	55	10
85	70	58	20
86	68	60	30
87	66	55	10
88	65	55	20
89	65	60	30
90	63	58	10
91	60	55	10
92	60	60	10
93	67	85	20
94	65	85	40
95	65	82	10
96	62	80	30
97	60	80	10
98	60	85	30
99	58	75	20
100	55	80	10
101	55	85	20

4.3 Παρουσίαση των αποτελεσμάτων του υβριδικού αλγόριθμου Τεχνητής Αποικίας Μελισσών

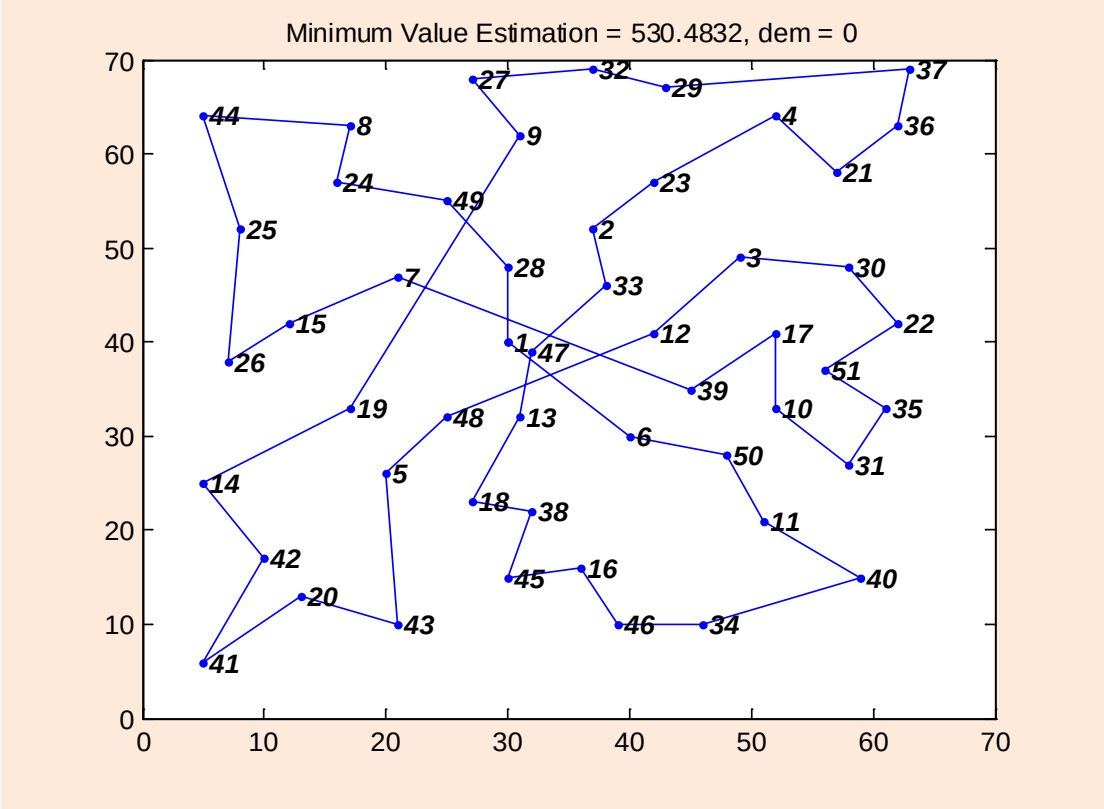
Παράδειγμα 1.

Για το παράδειγμα 1 (par1), έχουμε 51 κόμβους (n=51) καιη χωρητικότητα του οχήματος είναι ίση με 160 (Q=160). Αρχικά 'τρέξαμε' το πρόβλημα για dem=0, δηλαδή χωρίς απόκλιση από από την πραγματική ζήτηση. Τα πέντε καλύτερα αποτελέσματα, παρουσιάζονται στον παρακάτω πίνακα. Στην πρώτη στήλη εμφανίζεται το βέλτιστο κόστος που έχει σημειωθεί μέχρι σήμερα, ενώ στη δεύτερη τα αποτελέσματα που μας έδωσε ο αλγόριθμος ABC , στην τρίτη η επανάληψη στην οποία βρέθηκαν τα αποτελέσματα, ενώ στην τελευταία στήλη αναγράφεται η απόκλιση της κάθε λύσης από τη βέλτιστη, σε ποσοστό επί τοις εκατό.

par1 dem=0	**Αλγόριθμος ABC**	**Επανάληψη**	**Απόκλιση (%)**
Best Cost 524.6111	532.5864	44	1.52
	537.4482	28	2.45
	530.4832	**15**	**1.12**
	538.9778	36	2.74
	540.1124	55	2.95
Μέσος όρος	535.9216	36	2.16

Πίνακας 1. Αποτελέσματα για το par1,dem=0.

Όπως παρατηρούμε, καλύτερη τιμή που βρήκαμε με τον αλγόριθμο μας, είναι BC_{ABC}=530.4832 και η απόκλιση είναι ίση με 1,12% από τη βέλτιστη λύση. Στο σχήμα 1 αναπαριστάται η διαδρομή της συγκεκριμένης λύσης.



Σχήμα 1. Βέλτιστη διαδρομή του αλγορίθμου ABCγια το par1, dem=0

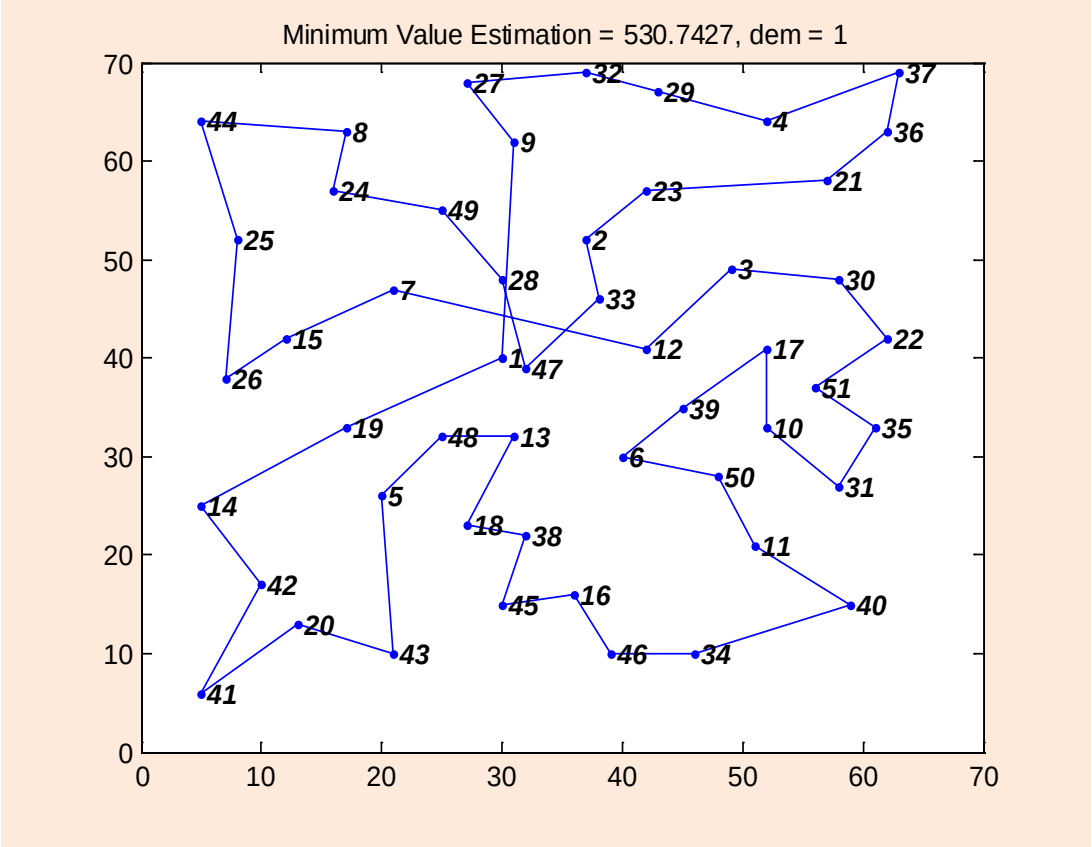
Ακολουθεί ο πίνακας με τις μεταβάσειςαπό κόμβο σε κόμβο, ξεκινώντας από την αποθήκη (κόμβος 1) και τερματίζοντας πάλι εκεί:

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

Συνεχίζοντας στο πρόβλημα 1 για dem=1 , δηλαδή η απόκλιση από τη ζήτηση είναι ±1, προέκυψαν τα παρακάτω αποτελέσματα:

par1 dem=1	Αλγόριθμος ABC	Επανάληψη	Απόκλιση (%)
Best Cost 528.5700	534.1226	25	1.05
	544.4880	24	3.01
	539.2332	44	2.02
	530.7427	**38**	**0.41**
	552.8448	63	4.59
Μέσος όρος	540.2863	39	2.22

Πίνακας 2. Αποτελέσματα για το par 1, dem=1.



Σχήμα 2. Βέλτιστη διαδρομή του αλγορίθμου ABCγια το par1, dem=1

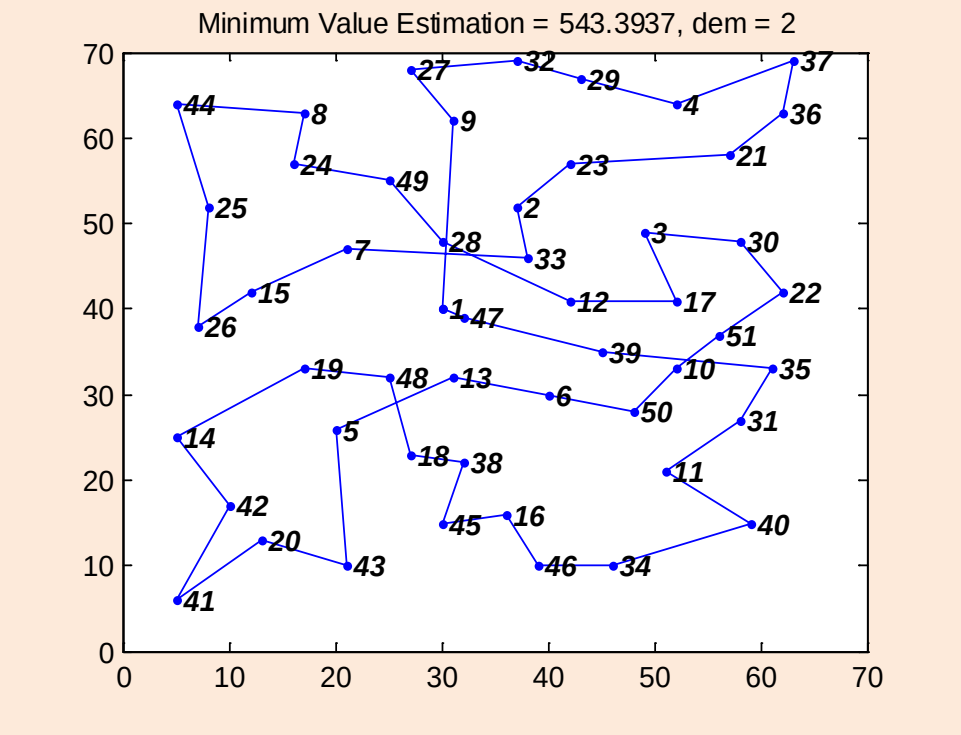
Αντίστοιχα, οι μεταβάσεις από κόμβο σε κόμβο:

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

Τελειώνοντας με το πρόβλημα 1, επιλύσαμε για dem=2, δηλαδή η απόκλιση από τη ζήτηση είναι ±2 και προέκυχαν τα παρακάτωαποτελέσματα:

par1 dem=2	**Αλγόριθμος ABC**	**Επανάληψη**	**Απόκλιση (%)**
Best Cost 531.5400	**543.3937**	**38**	**2.23**
	553.9197	35	4.21
	561.1554	41	5.57
	552.8227	56	4.00
	562.1997	53	5.77
Μέσος όρος	554.6982	45	4.36

Πίνακας 3. Αποτελέσματα για το par 1, dem=2.



Σχήμα 3. Βέλτιστη διαδρομή του αλγορίθμου ABCγια το par1, dem=2

Οι μεταβάσεις από κόμβο σε κόμβο για τη βέλτιστη λύση του αλγόριθμου ABC για το par 1, dem=2:

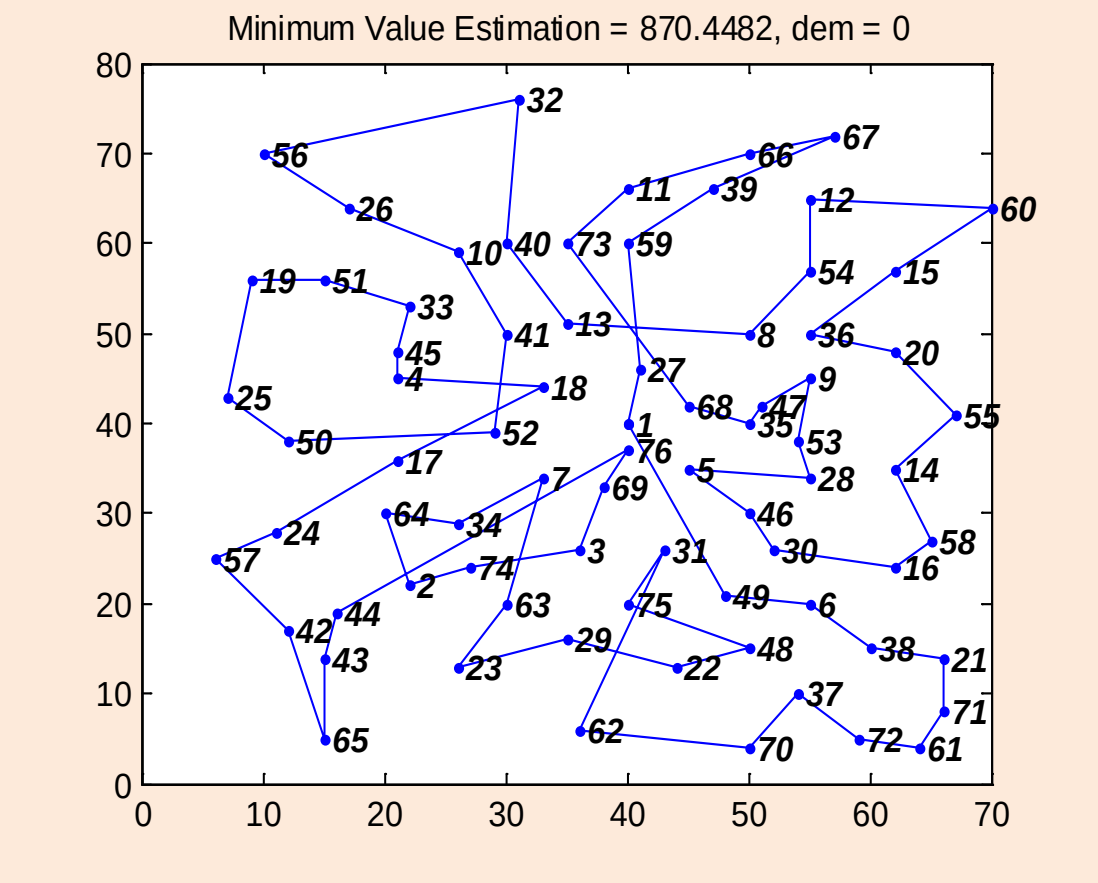
1	47	39	35	31	11	40	34	46	16
45	38	18	48	19	14	42	41	20	43
5	13	6	50	10	51	22	30	3	17
12	28	49	24	8	44	25	26	15	7
33	2	23	21	36	37	4	29	32	27
9	1								

Αντίστοιχα έχουμε δουλέψει και για τα υπόλοιπα προβλήματα, τα αποτελέσματα των οποίων παραθέτουμε παρακάτω:

Παράδειγμα 2.

par2 dem=0	Αλγόριθμος ABC	Επανάληψη	Απόκλιση (%)
Best Cost 836.0700	**873.9039**	**77**	**4.53**
	874.0151	139	4.54
	875.9141	53	4.77
	870.4792	141	4.12
	880.9373	154	5.37
Μέσος όρος	875.0499	113	4.66

Πίνακας 4. Αποτελέσματα για το par2, dem=0.



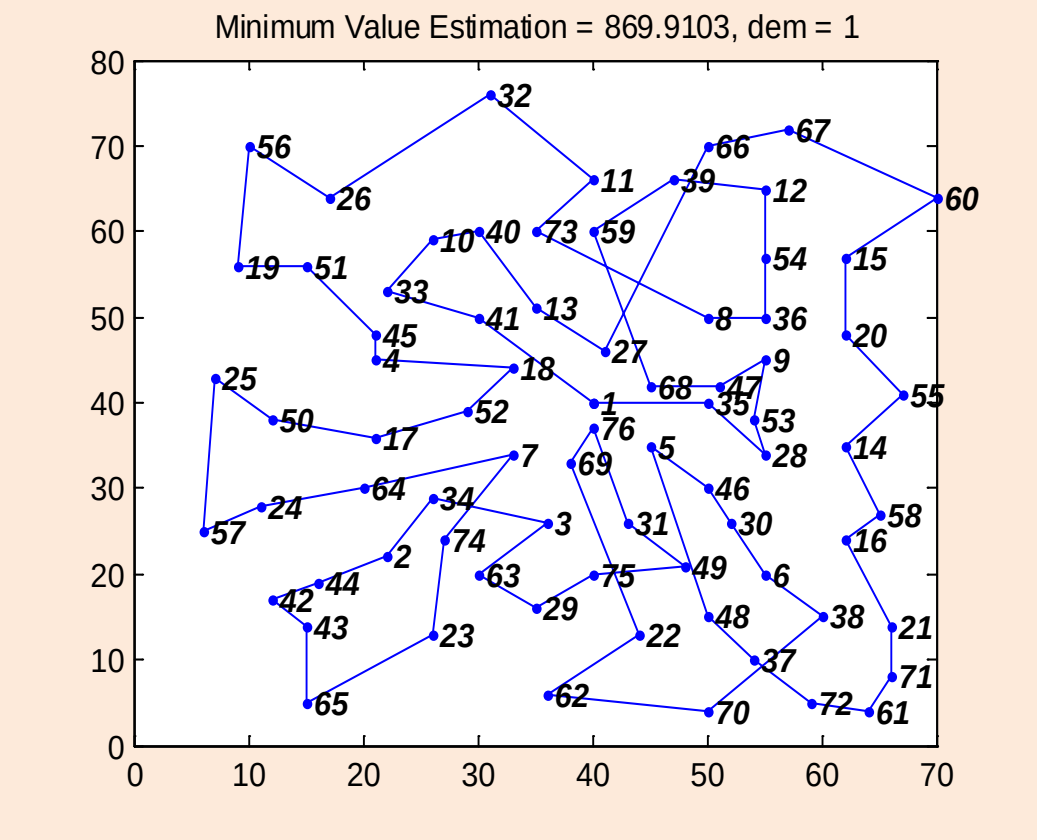
Σχήμα 4. Βέλτιστη διαδρομή του αλγορίθμου ABCγια το par2, dem=0.

Οι μεταβάσεις από κόμβο σε κόμβο για τη βέλτιστη λύση του αλγόριθμου ABC για το par 2, dem=0:

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

par2 dem=1	**Αλγόριθμος ABC**	**Επανάληψη**	**Απόκλιση (%)**
Best Cost 840.3502	891.6796	52	6.11
	872.2603	53	3.80
	884.7081	122	5.28
	869.9103	**60**	**3.52**
	889.5186	80	5.85
Μέσος όρος	881.6154	73	4.91

Πίνακας 5. Αποτελέσματα για το par2, dem=1.



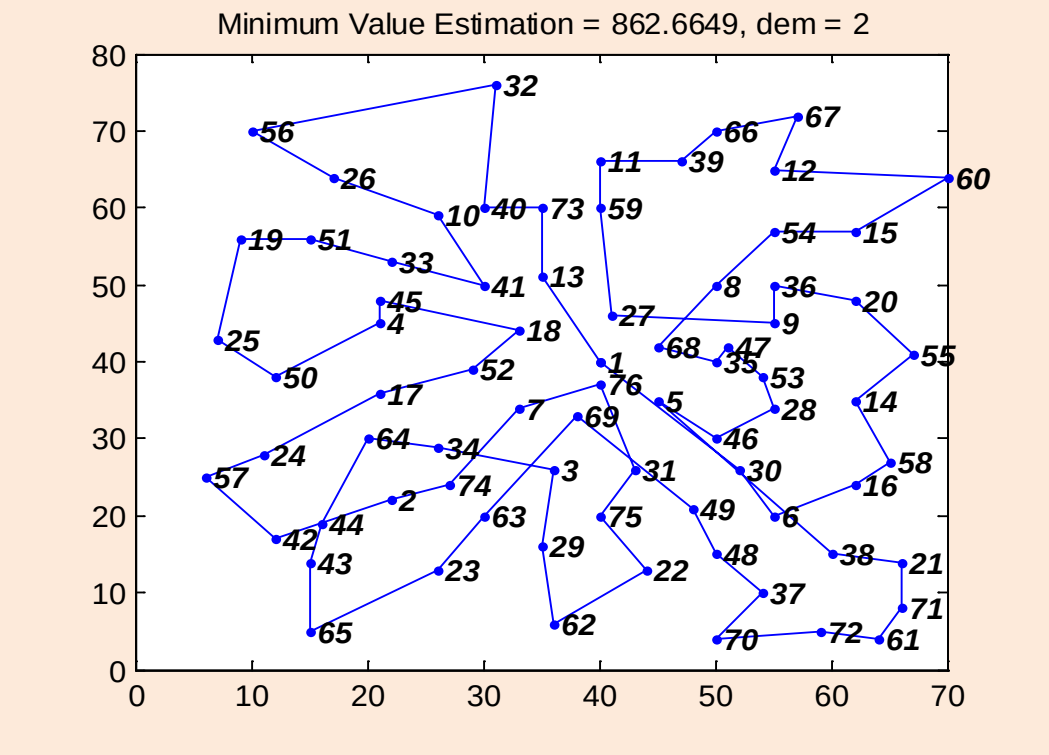
Σχήμα 5. Βέλτιστη διαδρομή του αλγορίθμου ABCγια το par2, dem=1.

Οι μεταβάσεις από κόμβο σε κόμβο για τη βέλτιστη λύση του αλγόριθμου ABC για το par 2, dem=1:

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

par2 dem=2	Αλγόριθμος ABC	Επανάληψη	Απόκλιση (%)
Best Cost 848.7300	894.7748	76	5.43
	862.6649	**91**	**1.64**
	891.1128	96	4.99
	872.8495	83	2.84
	895.3611	76	5.49
Μέσος όρος	883.3526	84	4.08

Πίνακας 6. Αποτελέσματα για το par2, dem=2.



Σχήμα 6. Βέλτιστη διαδρομή του αλγορίθμου ABCγια το par2, dem=2.

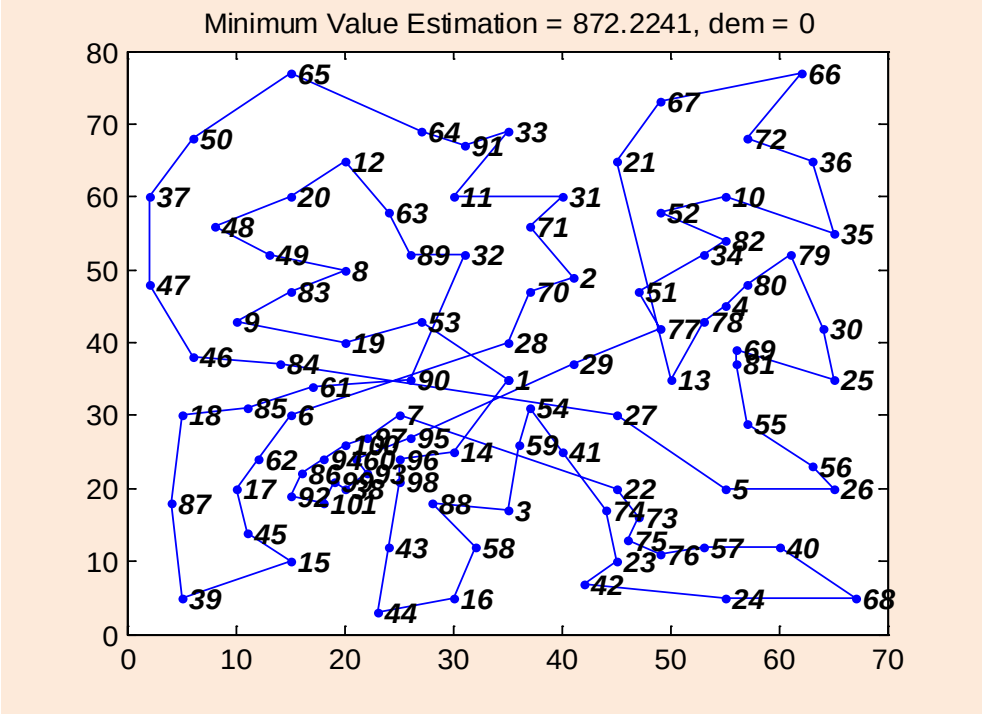
Οι μεταβάσεις από κόμβο σε κόμβο για τη βέλτιστη λύση του αλγόριθμου ABC για το par 2, dem=2:

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

Παράδειγμα 3.

par3 dem=0	Αλγόριθμος ABC	Επανάληψη	Απόκλιση (%)
Best Cost 837.7600	893.8525	107	6.70
	875.5153	135	4.51
	892.9150	106	6.58
	887.2707	60	5.91
	872.2241	**67**	**4.11**
Μέσος όρος	884.3555	95	5.56

Πίνακας 7. Αποτελέσματα για το par3, dem=0.



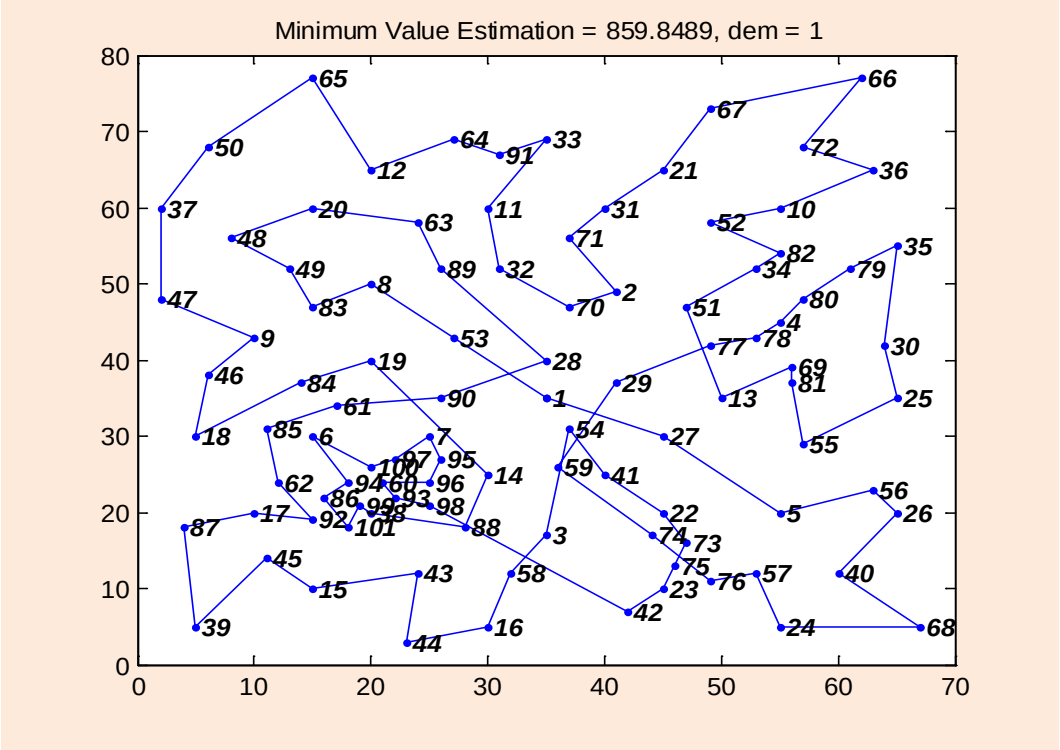
Σχήμα 7. Βέλτιστη διαδρομή του αλγορίθμου ABCγια το par3, dem=0.

Οι μεταβάσεις από κόμβο σε κόμβο για τη βέλτιστη λύση του αλγόριθμου ABC για το par3, dem=0:

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

par3 dem=1	Αλγόριθμος ABC	Επανάληψη	Απόκλιση (%)
Best Cost 836.8800	887.4306	78	6.04
	888.4829	129	6.17
	859.8489	**187**	**2.74**
	887.5961	161	6.06
	896.9060	196	7.17
Μέσος όρος	884.0529	150	5.64

Πίνακας 8. Αποτελέσματα για το par3, dem=1.



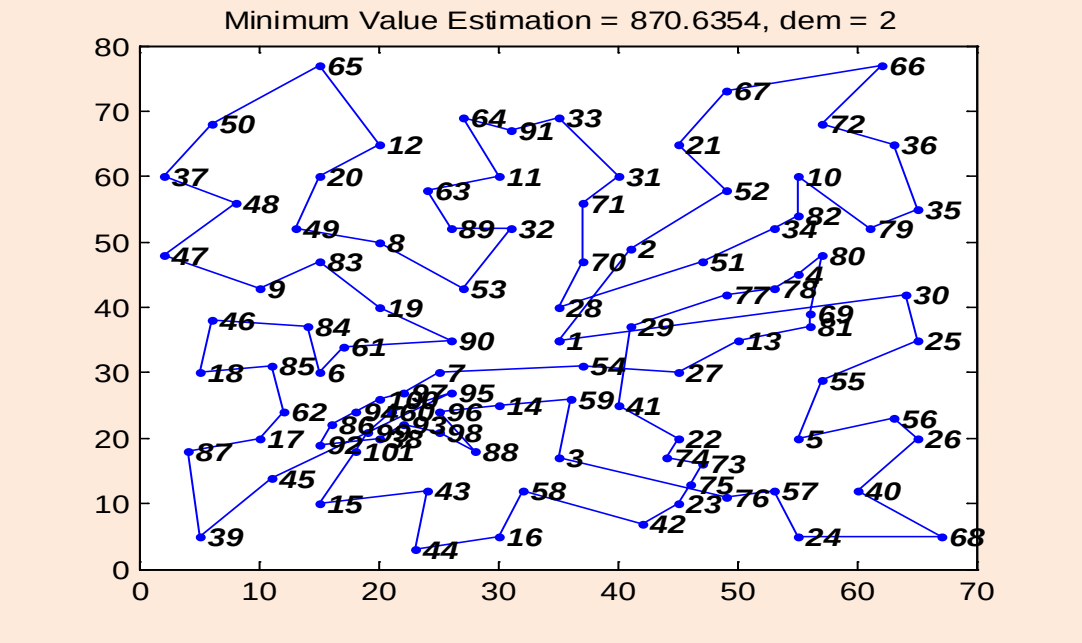
Σχήμα 8. Βέλτιστη διαδρομή του αλγορίθμου ABCγια το par3, dem=1.

Οι μεταβάσεις από κόμβο σε κόμβο για τη βέλτιστη λύση του αλγόριθμου ABC για το par3, dem=1:

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

par3 dem=2	Αλγόριθμος ABC	Επανάληψη	Απόκλιση (%)
Best Cost 842.4500	888.4710	102	5.46
	870.6354	**169**	**3.35**
	887.7762	199	5.38
	891.0809	186	5.77
	880.9159	152	4.57
Μέσος όρος	883.7759	162	4.91

Πίνακας 9. Αποτελέσματα για το par3, dem=2.



Σχήμα 9. Βέλτιστη διαδρομή του αλγορίθμου ABCγια το par3, dem=2.

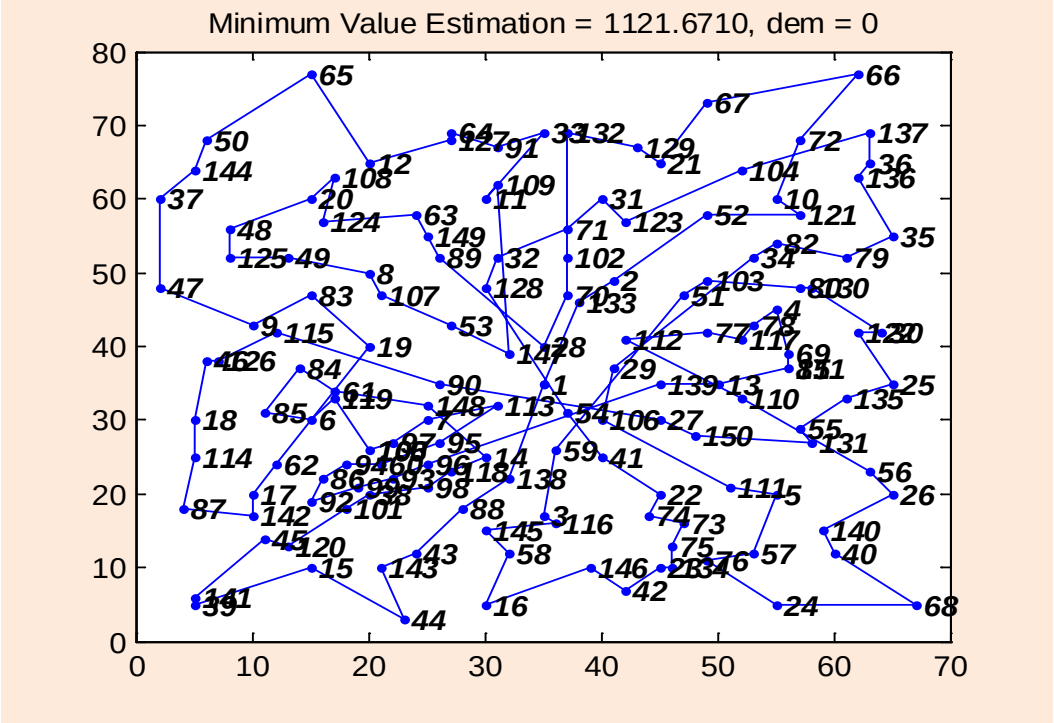
Οι μεταβάσεις από κόμβο σε κόμβο για τη βέλτιστη λύση του αλγόριθμου ABC για το par3, dem=2:

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

Παράδειγμα 4.

par4 dem=0	Αλγόριθμος ABC	Επανάληψη	Απόκλιση (%)
Best Cost 1074.4000	**1121.6710**	**199**	**4.40**
	1162.2311	191	8.17
	1163.9740	190	8.34
	1162.0530	191	8.16
	1147.8190	200	6.83
Μέσος όρος	1151.5496	194	7.18

Πίνακας 10. Αποτελέσματα για το par4, dem=0.



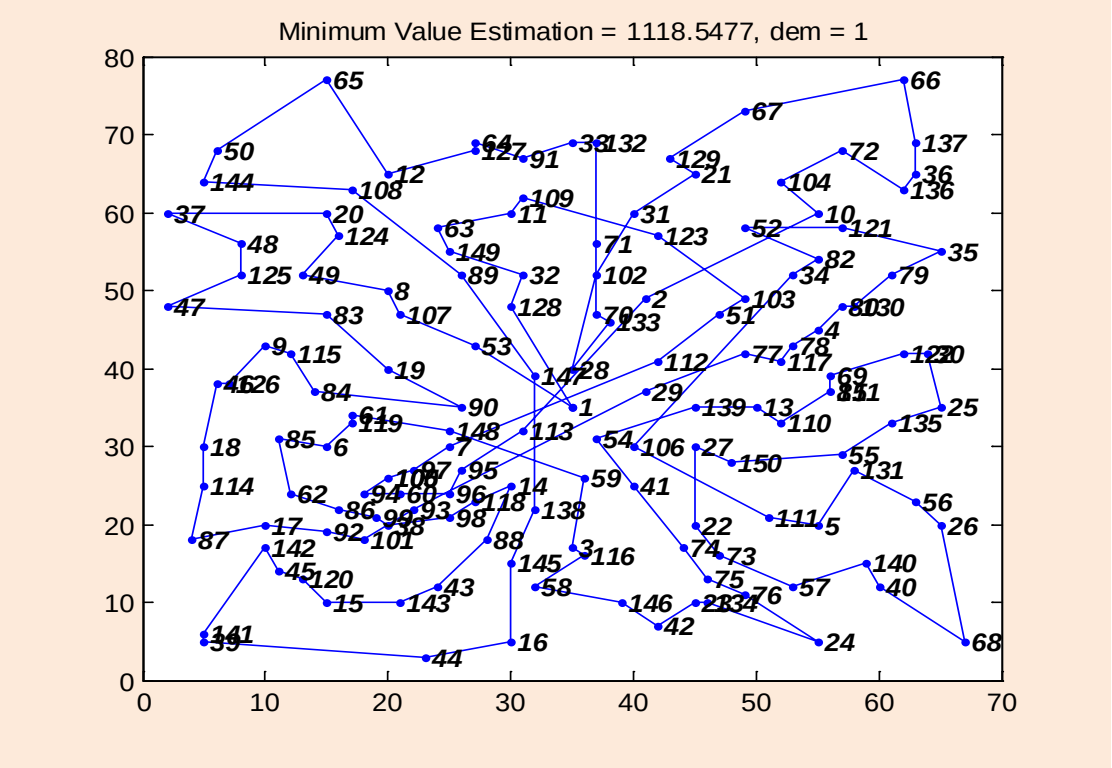
Σχήμα 10. Βέλτιστη διαδρομή του αλγορίθμου ABCγια το par4, dem=0.

Οι μεταβάσεις από κόμβο σε κόμβο για τη βέλτιστη λύση του αλγόριθμου ABC για το par4, dem=0:

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

par4 dem=1	Αλγόριθμος ABC	Επανάληψη	Απόκλιση (%)
Best Cost 1068.2000	1154.9610	300	8.12
	1169.7791	271	9.51
	1118.5480	**296**	**4.71**
	1157.2347	279	8.34
	1137.9454	282	6.53
Μέσος όρος	1147.6936	286	7.44

Πίνακας 11. Αποτελέσματα για το par4, dem=1.



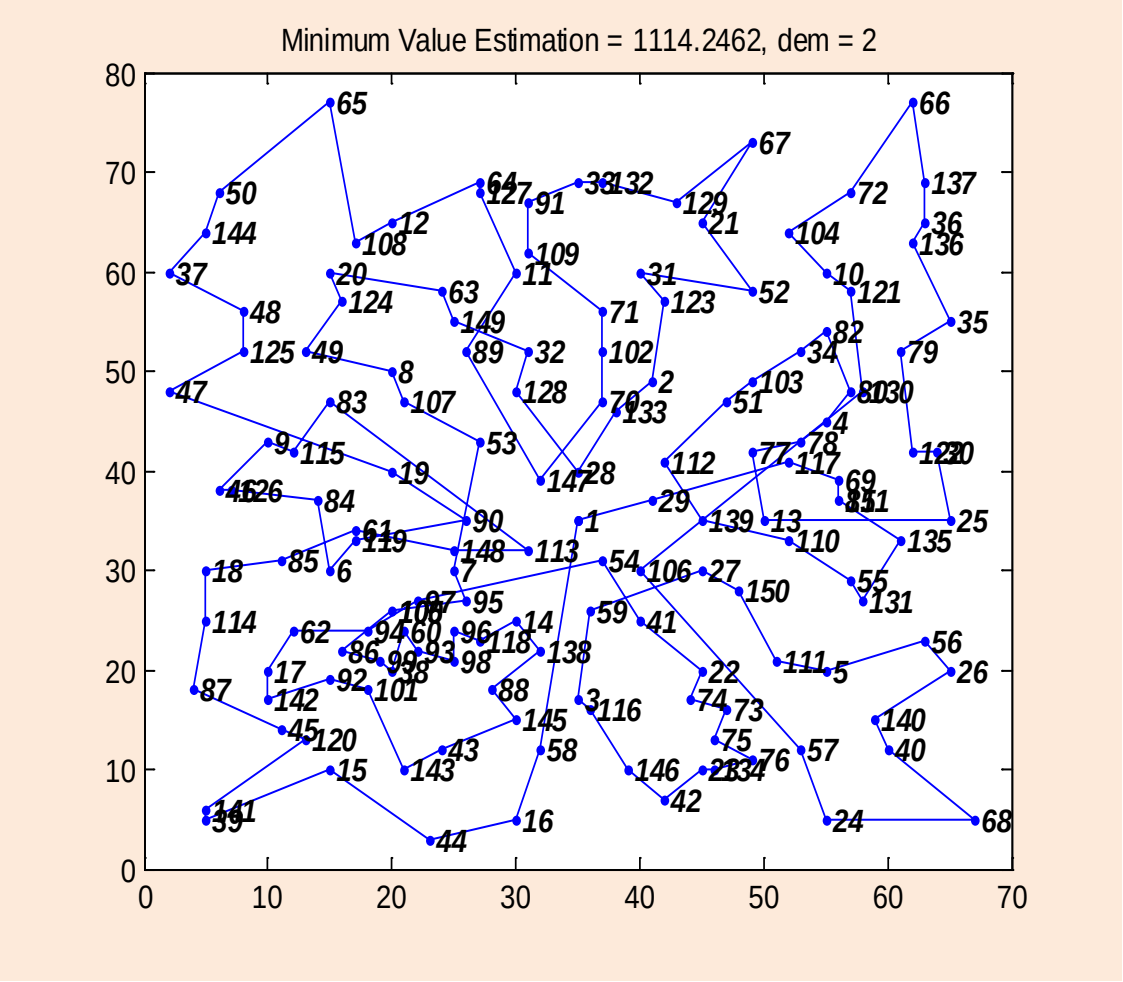
Σχήμα 11. Βέλτιστη διαδρομή του αλγορίθμου ABCγια το par4, dem=1.

Οι μεταβάσεις από κόμβο σε κόμβο για τη βέλτιστη λύση του αλγόριθμου ABC για το par4, dem=1:

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

par4 dem=2	Αλγόριθμος ABC	Επανάληψη	Απόκλιση (%)
Best Cost 1065.4000	1189.7421	187	11.67
	1196.8260	200	12.34
	1145.3900	267	7.51
	1136.4113	142	6.67
	1114.2460	**300**	**4.58**
Μέσος όρος	1156.5231	219	8.55

Πίνακας 12. Αποτελέσματα για το par4, dem=2.



Σχήμα 12. Βέλτιστη διαδρομή του αλγορίθμου ABCγια το par4, dem=2.

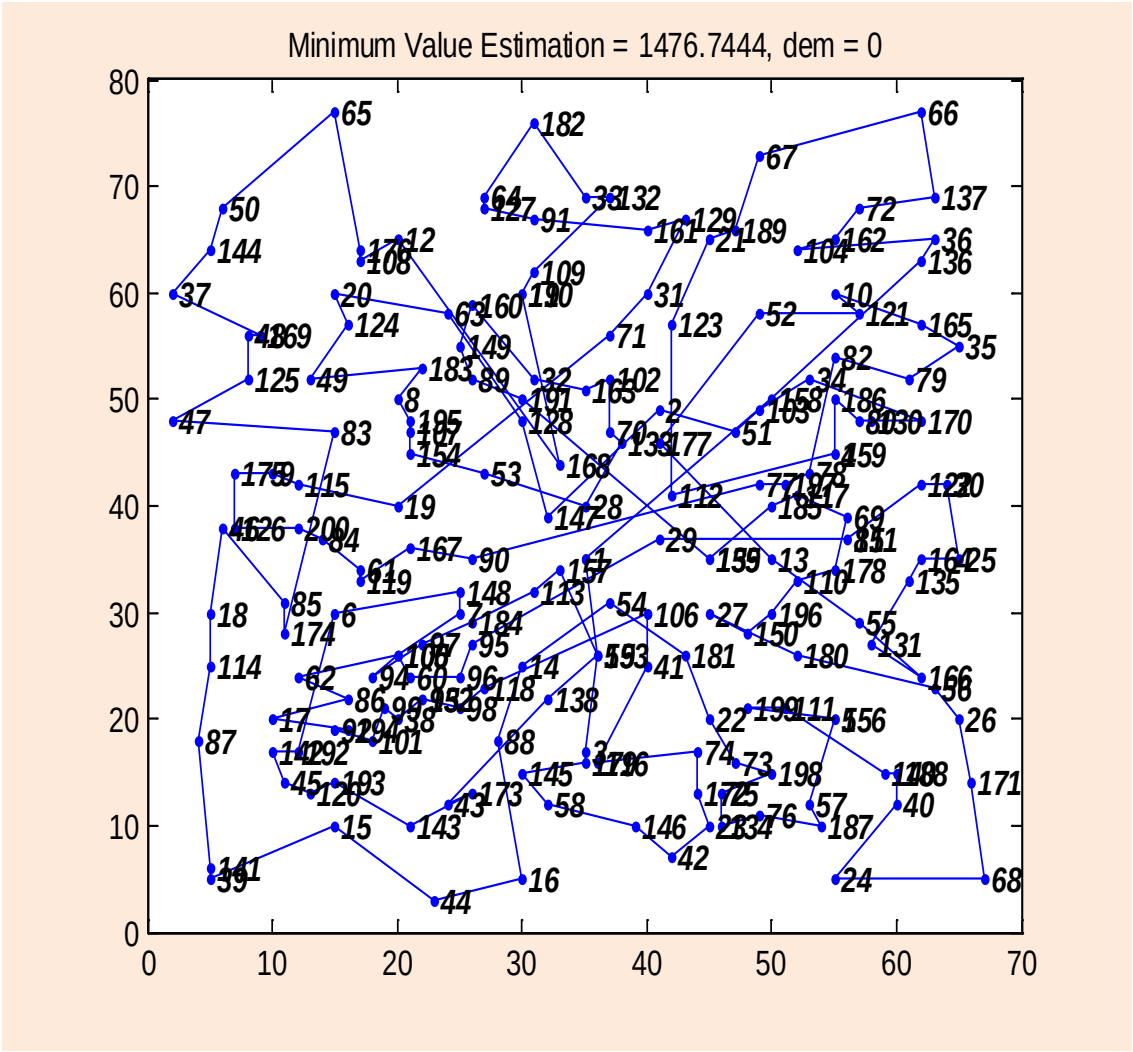
Οι μεταβάσεις από κόμβο σε κόμβο για τη βέλτιστη λύση του αλγόριθμου ABC για το par4, dem=2:

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

Παράδειγμα 5.

par5 dem=0	Αλγόριθμος ABC	Επανάληψη	Απόκλιση (%)
Best Cost 1379.8000	1497.9020	258	8.56
	1750.1040	252	26.84
	1476.7440	**300**	**7.03**
	1544.6203	241	11.95
	1541.8890	229	11.75
Μέσος όρος	1562.2519	256	13.22

Πίνακας 13. Αποτελέσματα για το par5, dem=0.



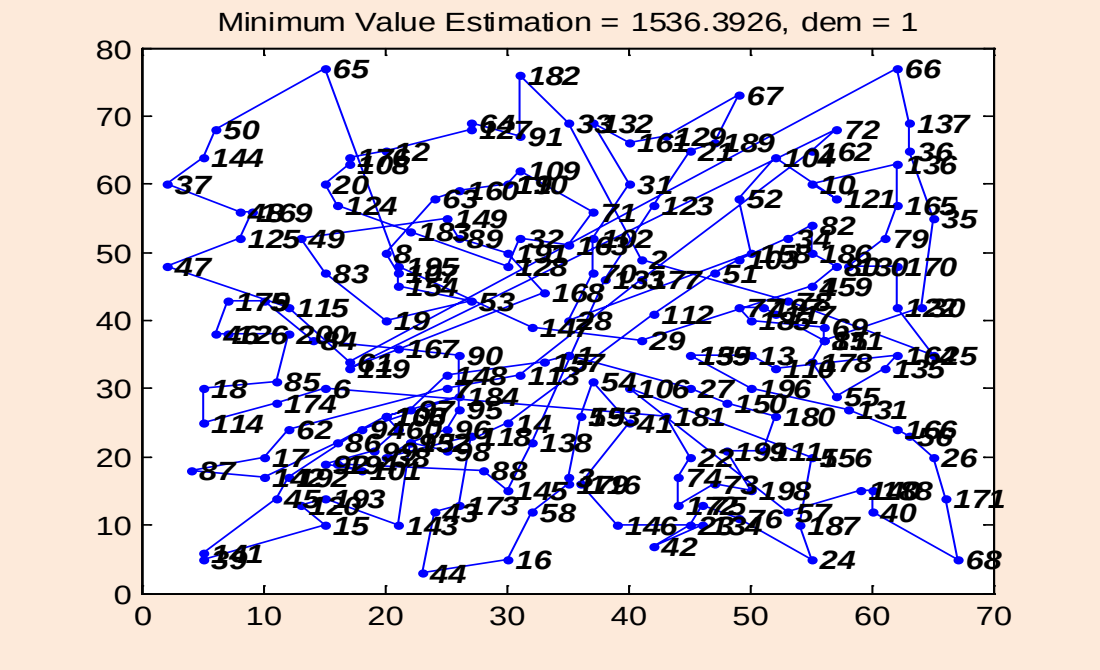
Σχήμα 13. Βέλτιστη διαδρομή του αλγορίθμου ABCγια το par5, dem=0.

Οι μεταβάσεις από κόμβο σε κόμβο για τη βέλτιστη λύση του αλγόριθμου ABC για το par5, dem=0:

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

par5 dem=1	Αλγόριθμος ABC	Επανάληψη	Απόκλιση (%)
Best Cost 1370.2000	1563.7124	300	14.12
	1536.3930	**153**	**12.13**
	1741.6584	246	27.11
	1622.4682	164	18.41
	1668.4187	285	21.76
Μέσος όρος	1626.5301	230	18.71

Πίνακας 14. Αποτελέσματα για το par5, dem=1.



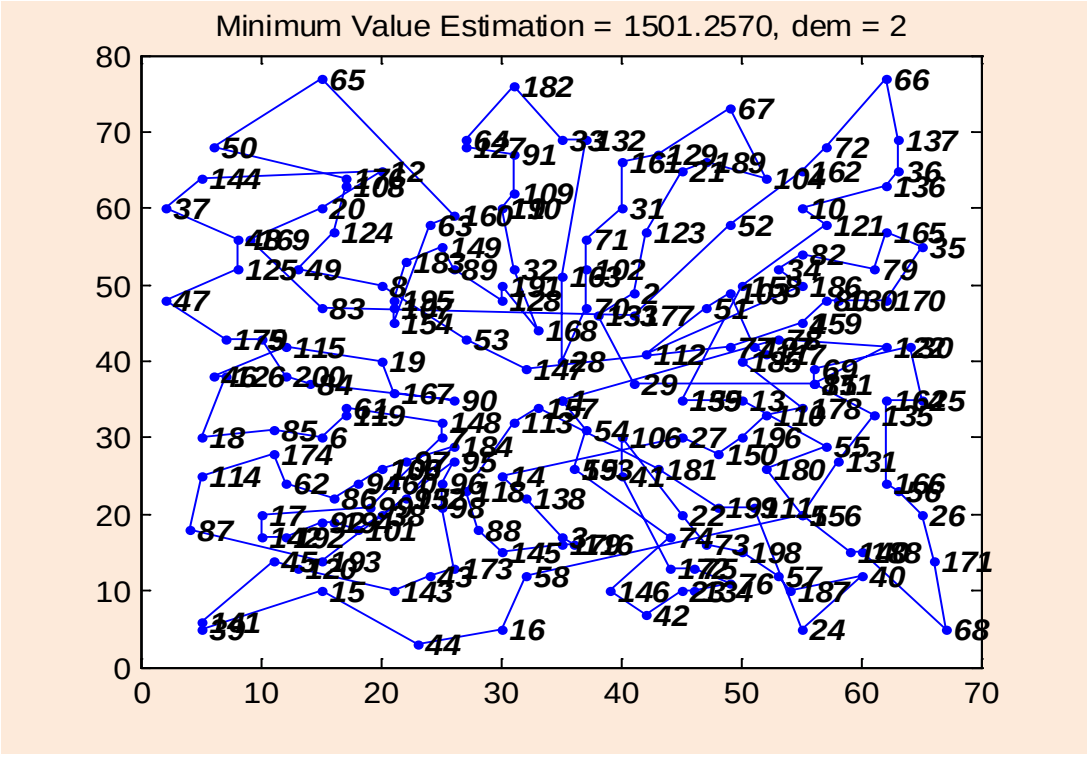
Σχήμα 14. Βέλτιστη διαδρομή του αλγορίθμου ABCγια το par5, dem=1.

Οι μεταβάσεις από κόμβο σε κόμβο για τη βέλτιστη λύση του αλγόριθμου ABC για το par5, dem=1:

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

par5 dem=2	Αλγόριθμος ABC	Επανάληψη	Απόκλιση (%)
Best Cost 1387.0000	1532.4110	233	10.48
	1561.0224	274	12.55
	1657.6845	246	19.52
	1501.2570	**291**	**8.24**
	1678.5002	283	21.02
Μέσος όρος	1586.1750	265	14.36

Πίνακας 15. Αποτελέσματα για το par5, dem=2.



Σχήμα 15. Βέλτιστη διαδρομή του αλγορίθμου ABCγια το par5, dem=2.

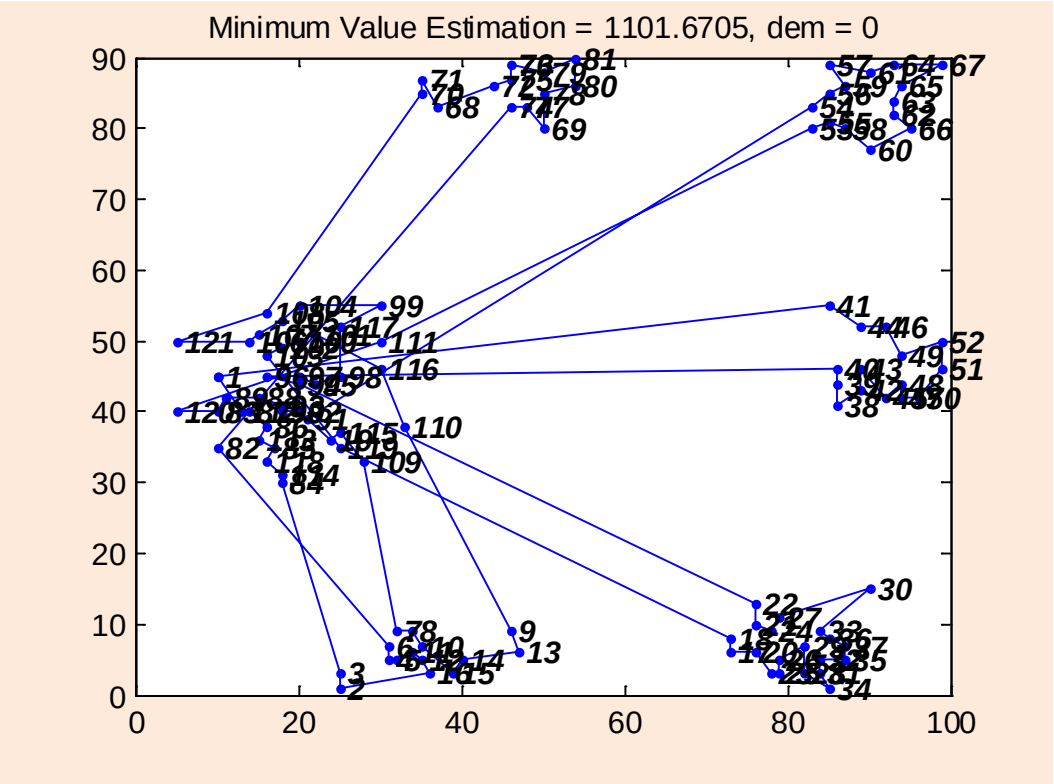
Οι μεταβάσεις από κόμβο σε κόμβο για τη βέλτιστη λύση του αλγόριθμου ABC για το par5, dem=2:

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

Παράδειγμα 6.

par6 dem=0	Αλγόριθμος ABC	Επανάληψη	Απόκλιση (%)
Best Cost 1054.1000	1196.3012	188	13.49
	1182.0625	201	12.14
	1160.8450	121	10.13
	1101.6710	**213**	**4.51**
	1192.5000	164	13.13
Μέσος όρος	1166.6759	177	10.68

Πίνακας 16. Αποτελέσματα για το par6, dem=0.



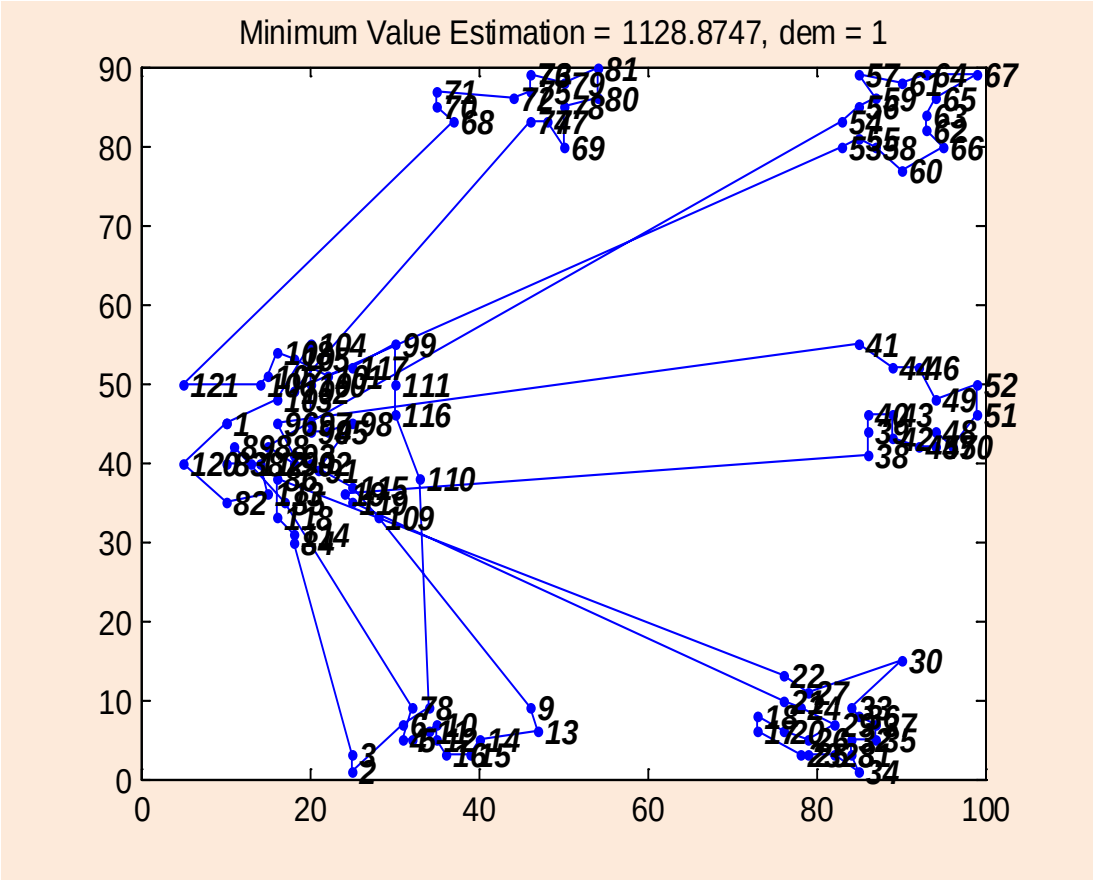
Σχήμα 16. Βέλτιστη διαδρομή του αλγορίθμου ABCγια το par6, dem=0.

Οι μεταβάσεις από κόμβο σε κόμβο για τη βέλτιστη λύση του αλγόριθμου ABC για το par6, dem=0:

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

par6 dem=1	Αλγόριθμος ABC	Επανάληψη	Απόκλιση (%)
Best Cost 1075.3000	**1128.8750**	**149**	**4.98**
	1230.7122	182	14.45
	1237.1889	144	15.06
	1206.6563	129	12.22
	1226.8561	234	14.09
Μέσος όρος	1206.0577	168	12.16

Πίνακας 17. Αποτελέσματα για το par6, dem=1.



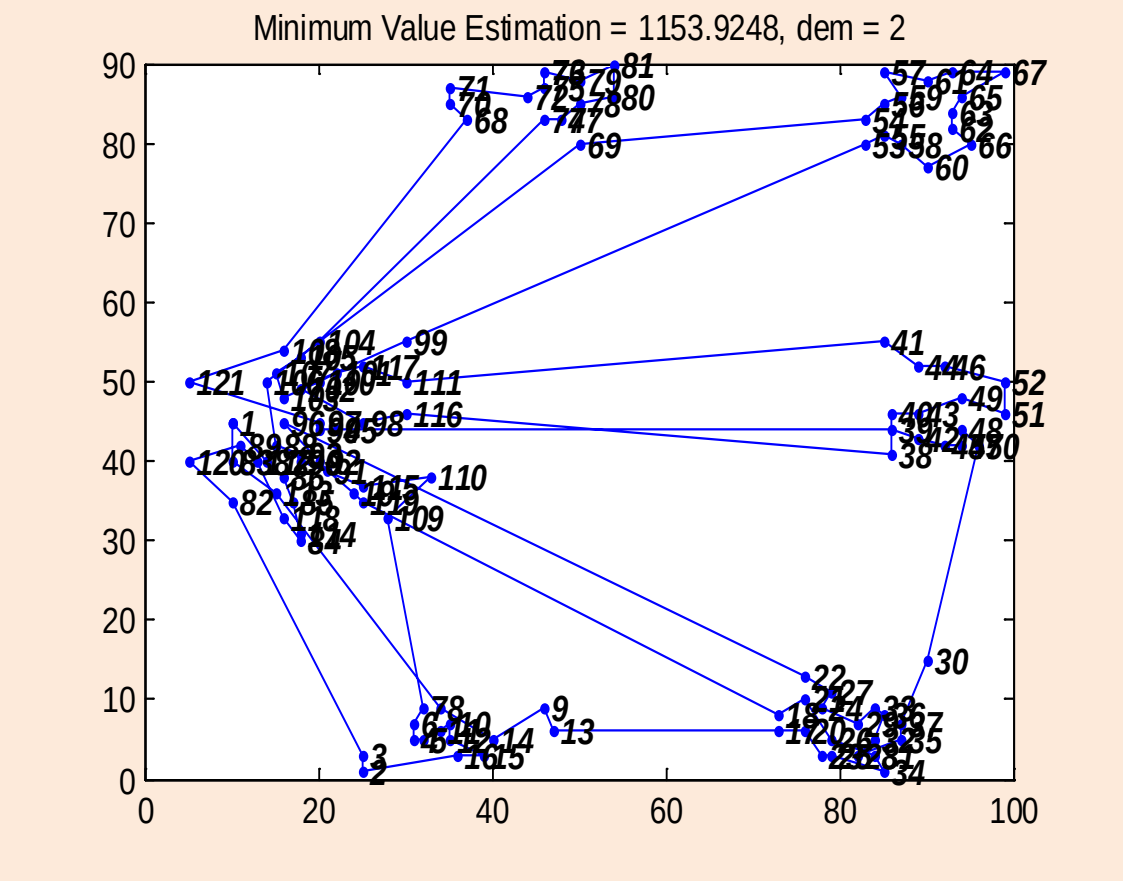
Σχήμα 17. Βέλτιστη διαδρομή του αλγορίθμου ABCγια το par6, dem=1.

Οι μεταβάσεις από κόμβο σε κόμβο για τη βέλτιστη λύση του αλγόριθμου ABC για το par6, dem=1:

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

par6 dem=2	Αλγόριθμος ABC	Επανάληψη	Απόκλιση (%)
Best Cost 1095.3000	1210.2731	144	10.50
	1175.8416	100	7.35
	1153.9248	157	5.35
	1139.5109	**42**	**4.04**
	1157.0316	150	5.64
Μέσος όρος	1167.3164	119	6.58

Πίνακας 18. Αποτελέσματα για το par6, dem=2.



Σχήμα 18. Βέλτιστη διαδρομή του αλγορίθμου ABCγια το par6, dem=2.

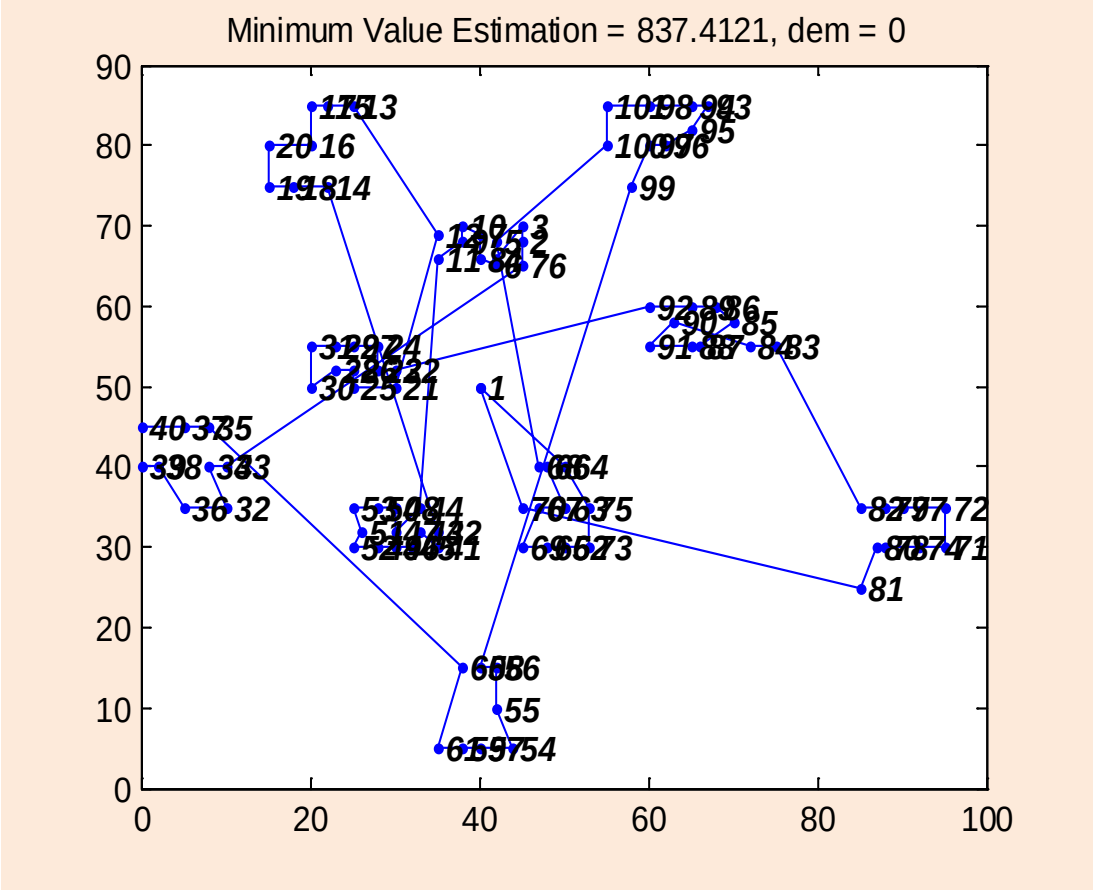
Οι μεταβάσεις από κόμβο σε κόμβο για τη βέλτιστη λύση του αλγόριθμου ABC για το par6, dem=2:

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

Παράδειγμα 7.

par7 dem=0	Αλγόριθμος ABC	Επανάληψη	Απόκλιση (%)
Best Cost 819.5575	872.3398	160	6.44
	849.6246	102	3.67
	837.4121	**193**	**2.18**
	888.0023	172	8.35
	893.7415	53	9.05
Μέσος όρος	868.2241	136	5.94

Πίνακας 19. Αποτελέσματα για το par7, dem=0.



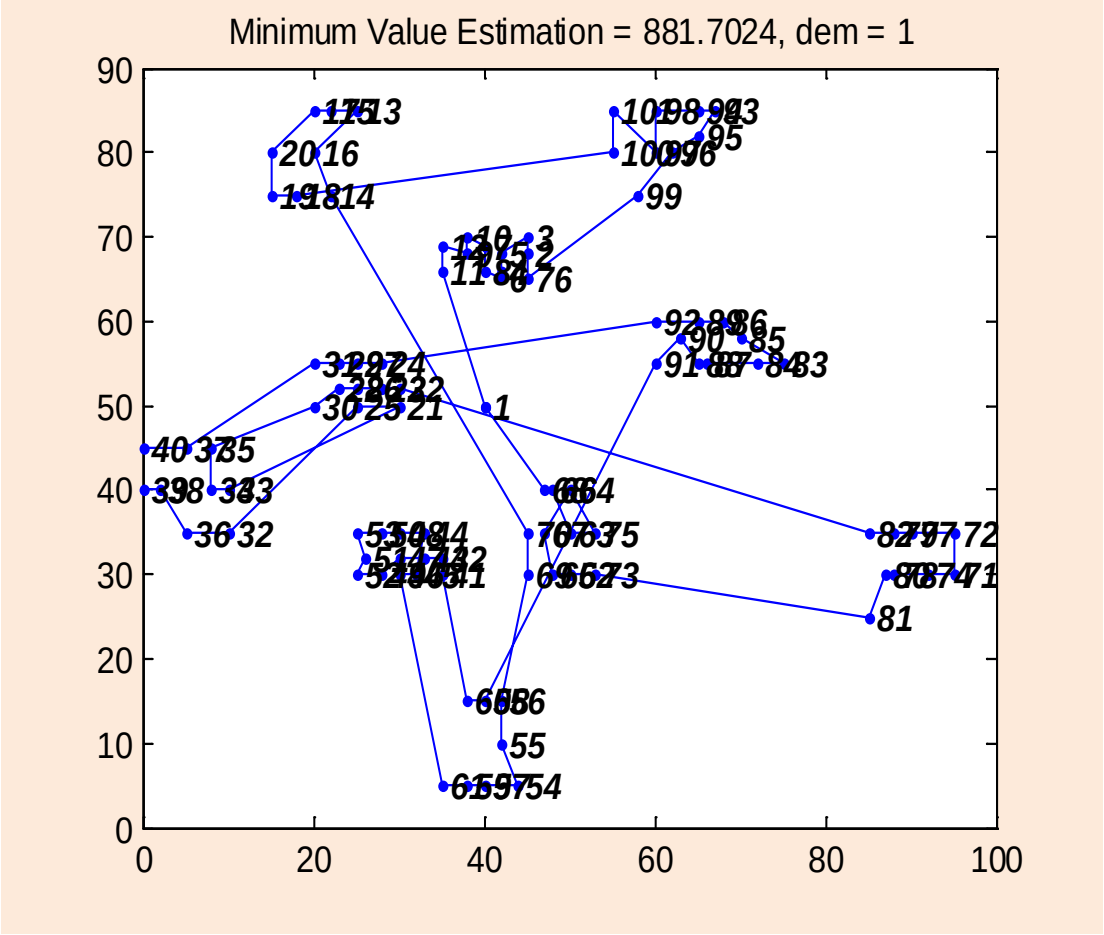
Σχήμα 19. Βέλτιστη διαδρομή του αλγορίθμου ABCγια το par7, dem=0.

Οι μεταβάσεις από κόμβο σε κόμβο για τη βέλτιστη λύση του αλγόριθμου ABC για το par7, dem=0:

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

par7 dem=1	Αλγόριθμος ABC	Επανάληψη	Απόκλιση (%)
Best Cost 851.4700	901.0223	166	5.82
	932.4402	202	9.51
	917.8733	178	7.80
	881.7024	**149**	**3.55**
	887.4125	163	4.22
Μέσος όρος	904.0901	172	6.18

Πίνακας 20. Αποτελέσματα για το par7, dem=1.



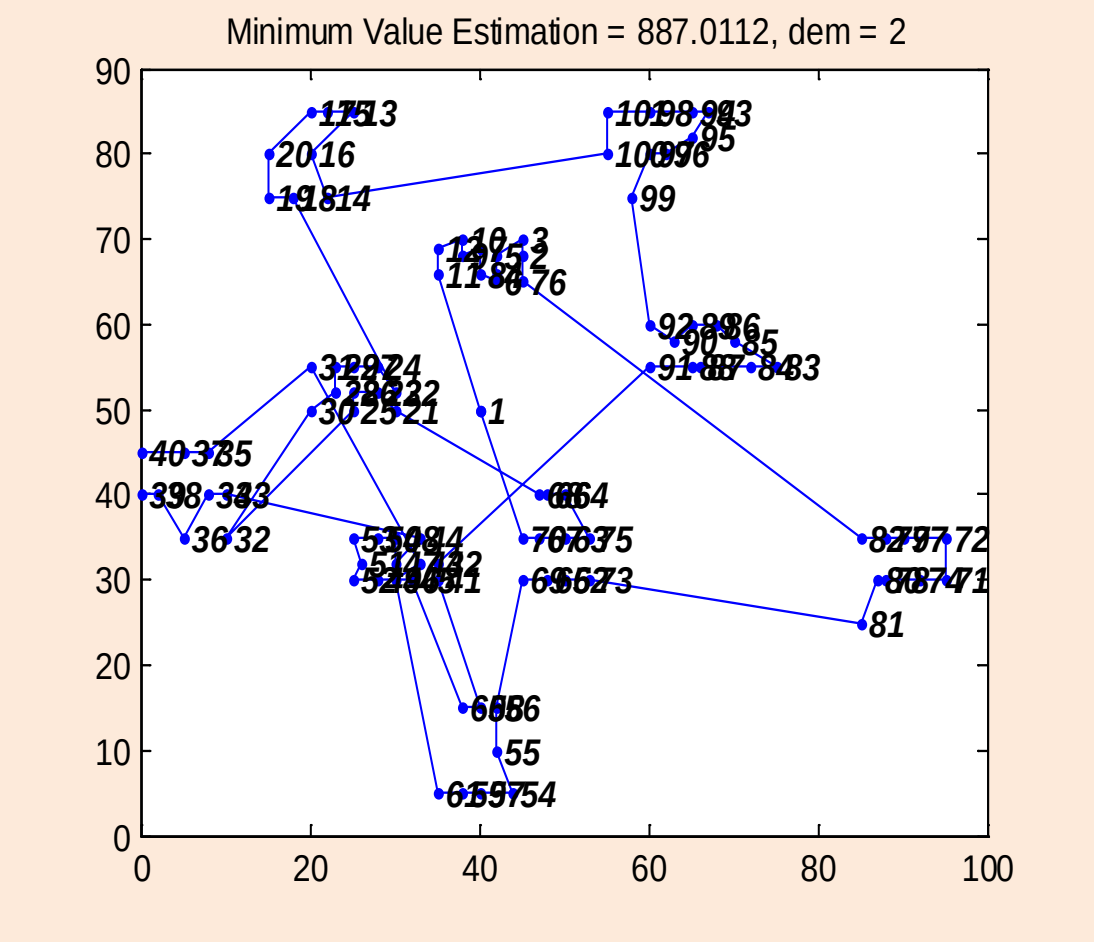
Σχήμα 20. Βέλτιστη διαδρομή του αλγορίθμου ABCγια το par7, dem=1.

Οι μεταβάσεις από κόμβο σε κόμβο για τη βέλτιστη λύση του αλγόριθμου ABC για το par7, dem=1:

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

par7 dem=2	Αλγόριθμος ABC	Επανάληψη	Απόκλιση (%)
Best Cost 858.8513	999.5831	141	16.39
	887.0112	**199**	**3.28**
	902.3020	143	5.06
	1007.2126	123	17.27
	955.4271	155	11.24
Μέσος όρος	950.3072	152	10.65

Πίνακας 21. Αποτελέσματα για το par7, dem=2.



Σχήμα 21. Βέλτιστη διαδρομή του αλγορίθμου ABCγια το par7, dem=2.

Οι μεταβάσεις από κόμβο σε κόμβο για τη βέλτιστη λύση του αλγόριθμου ABC για το par7, dem=2:

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

Κεφάλαιο 5.

Σύγκριση αποτελεσμάτων

5.1 Εισαγωγή

Στο κεφάλαιο αυτό, θα συγκρίνουμε τα αποτελέσματα που μας έδωσε ο αλγόριθμος Τεχνητής Αποικίας Μελισσών ABC, με τους αλγόριθμους οι οποίοι έχουν επιλύσει τα ίδια προβλήματα δρομολόγησης οχημάτων με στοχαστική ζήτηση. Οι αλγόριθμοι αυτοί είναι οι εξής: ο αλγόριθμος Διαφορικής Εξέλιξης DE και ο Γενετικός Αλγόριθμος GA (Σπανού, 2010), ο αλγόριθμος Βελτιστοποίησης Σμήνους Σωματιδίων PSO (Ιορδανίδου 2011), ο αλγόριθμος Βελτιστοποίησης Ζευγαρώματος Μελισσών HBMO, ο αλγόριθμος Βελτιστοποίησης Ζευγαρώματος Μπάμπουρων BBMO, ο αλγόριθμος Επιλογής Κλώνων CSA και ο αλγόριθμος Βελτιστοποίησης Σμήνους Πυγολαμπίδων GSO (Ιορδανίδου, 2012).

5.2 Περιγραφή αλγορίθμων

Πριν όμως προχωρήσουμε στις συγκρίσεις, ας περιγράψουμε πρώτα συνοπτικά, έναν προς έναν τους αλγόριθμους που αναφέραμε παραπάνω και δεν έχουμε αναλύσει στο κεφάλαιο **2.4** .

5.2.1 Αλγόριθμος Διαφορικής Εξέλιξης (Differential Evolution algorithm, DE)

Η Διαφορική Εξέλιξη (Differential Evolution, DE) είναι ένας στοχαστικός, βασιζόμενος σε πληθυσμό, αλγόριθμος που προτάθηκε από τους Storn και Price (Kenneth V.Price, Rainer M.Storn, Jouni A.Lampinen, 2005). Η Διαφορική Εξέλιξη έχει τα βασικά χαρακτηριστικά των εξελικτικών αλγορίθμων καθώς είναι ένας εξελικτικός αλγόριθμος. Έχει όμως και ένα αριθμό από διαφορές όπως το γεγονός ότι αυτή η μέθοδος εστιάζει στην απόσταση μεταξύ των μελών του πληθυσμού και στις διαφορετικές κατευθύνσεις που μπορεί να κινηθεί κάποιο μέλος του πληθυσμού. Για να εφαρμοστεί η διαφορική εξέλιξη θα πρέπει να έχουμε κωδικοποιήσει τις λύσεις με αναπαράσταση πραγματικού αριθμού ώστε να μπορούν να εφαρμοστούν οι τελεστές

μετάλλαξης. Στους εξελικτικούς αλγορίθμους όταν χρησιμοποιείται ο τελεστής διασταύρωσης εφαρμόζεται αρχικά πάνω σε δύο ή περισσότερους γονείς ενώ στη συνέχεια στον απόγονο ή στους απογόνους που θα δημιουργηθούν, εφαρμόζεται ένας τελεστής μετάλλαξης ο οποίος συνήθως μετακινεί τη λύση από ένα σημείο σε κάποιο άλλο με τη χρήση κάποιου βήματος που στηρίζεται σε κάποια κατανομή πιθανότητας. Υπάρχουν δύο βασικές διαφορές στη διαφορική εξέλιξη (Engelbrecht, 2007):

1. Ο τελεστής μετάλλαξης χρησιμοποιείται αρχικά για να παραχθεί ένα δοκιμαστικό διάνυσμα, το οποίο στη συνέχεια χρησιμοποιείται με κάποιο τελεστή διασταύρωσης για τη δημιουργία ενός απογόνου

2. Τα βήματα που γίνονται με τον τελεστή μετάλλαξης δεν υπόκεινται σε κάποια γνωστή κατανομή πιθανοτήτων αλλά επηρεάζονται από τις διαφορετικές τιμές στα γονίδια ανάμεσα σε μέλη του πληθυσμού.

Ο τελεστής μετάλλαξης παράγει ένα δοκιμαστικό διάνυσμα για κάθε μέλος του πληθυσμού με μετάλλαξη ενός διανύσματος στόχου, στο οποίο προστίθεται η διαφορά ανάμεσα στις τιμές των γονιδίων δύο ή περισσότερων ατόμων του πληθυσμού, πολλαπλασιασμένων με κάποιο βάρος. Το δοκιμαστικό διάνυσμα τότε θα χρησιμοποιηθεί από τον τελεστή διασταύρωσης για να δημιουργηθεί ο απόγονος.

Μετά την ολοκλήρωση της φάσης της μετάλλαξης εφαρμόζεται ένας τελεστής διασταύρωσης. Στις πρώτες εφαρμογές που εμφανίστηκαν με τη μέθοδο της διαφορικής εξέλιξης, η συνάρτηση ποιότητας του δοκιμαστικού διανύσματος και του γονέα συγκρίνονταν και αυτό που είχε καλύτερη τιμή επιλεγόταν για την επόμενη γενιά. Στη συνέχεια, για να βελτιωθεί η αποδοτικότητα της μεθόδου χρησιμοποιείται ένας τελεστής διασταύρωσης που ονομάζεται ‘δυωνυμικός τελεστής διασταύρωσης’ (binomial crossover) (Engelbrecht, 2007) ή ‘ομοιόμορφος τελεστής διασταύρωσης’ (uniform crossover) (Price και συνεργάτες, 2005). Σε αυτόν τον τελεστή διασταύρωσης τα γονίδια επιλέγονται τυχαία από το δοκιμαστικό διάνυσμα και από τον γονέα.

Μετά τον τελεστή διασταύρωσης, η συνάρτηση καταλληλότητας ενός απογόνου, υπολογίζεται και εάν είναι καλύτερο από την συνάρτηση καταλληλότητας του γονέα, τότε επιλέγεται για την επόμενη γενιά, αλλιώς ο γονέας επιβιώνει για μία ακόμα γενιά.

5.2.2 Γενετικός αλγόριθμος (GeneticAlgorithm, GA)

Οι αρχικοί βασικοί παράγοντες που υπάρχουν σε κάθε γενετικό αλγόριθμο και γενικότερα σε κάθε εξελικτικό αλγόριθμο είναι:

1. η κωδικοποίηση των λύσεων έτσι ώστε οι λύσεις να αντιστοιχούν σε κάποιο χρωμόσωμα ή χρωμοσώματα,

2. μια συνάρτηση που να εκτιμά την ποιότητα-καταλληλότητα του κάθε ενός από τα άτομα του πληθυσμού,

3. η αρχικοποίηση του πληθυσμού,

4. η επιλογή των τελεστών,

5. οι τελεστές αναπαραγωγής.

Κωδικοποίηση των λύσεων

Κάθε ένα άτομο του πληθυσμού αντιπροσωπεύει μια λύση του προβλήματος βελτιστοποίησης. Τα χαρακτηριστικά των ατόμων του πληθυσμού αναπαριστώνται από το χρωμόσωμα ή αλλιώς γονιδίωμα και αφορούν τις μεταβλητές του προβλήματος βελτιστοποίησης. Ένα πολύ σημαντικό χαρακτηριστικό στο 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 (Engelbrecht, 2007).

Επιλογή Αρχικού Πληθυσμού

Ο πιο συνήθης τρόπος για να δώσουμε τιμές σε κάθε άτομο του πληθυσμού, είναι με τη χρήση της τυχαίας επιλογής τιμών για κάθε μία μεταβλητή του ατόμου. Με αυτό τον τρόπο εξασφαλίζεται η ομοιόμορφη κατανομή του αρχικού πληθυσμού σε ολόκληρο το χώρο λύσεων.

Η δομή των γενετικών αλγορίθμων είναι τέτοια ώστε αν δεν καλυφθεί ολόκληρος ο χώρος λύσεων με τη δημιουργία των αρχικών λύσεων τότε τα μέρη του χώρου λύσεων που δεν θα καλυφθούν σε πολλές περιπτώσεις δεν θα υπάρχει η δυνατότητα να χρησιμοποιηθούν κατά τη διάρκεια της αναζήτησης.

Τελεστές Διασταύρωσης

Αφού γίνει η επιλογή των γονέων, η επόμενη διαδικασία που χρησιμοποιείται είναι η δημιουργία των απογόνων. Η δημιουργία των απογόνων πραγματοποιείται είτε με τη διαδικασία της διασταύρωσης, είτε με τη διαδικασία της μετάλλαξης είτε και με τις δύο διαδικασίες. Διασταύρωση είναι η δημιουργία δύο ή περισσότερων απογόνων χρησιμοποιώντας γενετικό υλικό από δύο ή περισσότερους γονείς.

Τελεστές Μετάλλαξης

Η δεύτερη διαδικασία που χρησιμοποιείται για την επιλογή των απογόνων είναι η διαδικασία της μετάλλαξης. Η μετάλλαξη είναι η διαδικασία όπου ένα υποσύνολο από μεταβλητές (γονίδια) επιλέγονται τυχαία και οι τιμές τους αλλάζουν. Ο κύριος στόχος της μετάλλαξης είναι η εισαγωγή νέου γενετικού υλικού στον πληθυσμό. Ο τελεστής μετάλλαξης πρέπει να εφαρμόζεται με προσοχή ώστε να μην καταστρέφει πολύ καλές λύσεις και για αυτό το λόγο συνήθως εφαρμόζεται σε ένα πολύ μικρό ποσοστό του πληθυσμού.

Επιλογή Καινούριου Πληθυσμού

Μετά την πραγματοποίηση των διαδικασιών μετάλλαξης και διασταύρωσης γίνεται η επιλογή του καινούριου πληθυσμού που θα αποτελέσει τη νέα γενιά. Μια πολύ καλή λύση είναι να χρησιμοποιήσουμε μια κατάταξη των λύσεων βάσει της συνάρτησης ποιότητας της λύσης, τόσο των γονέων όσο και των απογόνων, και από εκεί να επιλέξουμε το νέο πληθυσμό. Αυτή η στρατηγική ονομάζεται ‘επιβίωση των καλύτερων’.

Οι γενετικοί αλγόριθμοι δεν μπορούν να εφαρμοστούν χωρίς καμία τροποποίηση όταν χρειαστεί να επιλύσουν ένα πρόβλημα που έχει περιορισμούς. Σε αυτή την περίπτωση απαιτείται μία αλλαγή στη συνάρτηση καταλληλότητας ή και στη συμπεριφορά του ίδιου του αλγορίθμου. Ο πιο συνήθης τρόπος για να αντιμετωπιστεί το πρόβλημα είναι η εισαγωγή μίας συνάρτησης τιμωρίας στη συνάρτηση καταλληλότητας.

5.2.3 Αλγόριθμος βελτιστοποίησης σμήνους σωματιδίων (Particle Swarm Optimization, PSO)

Ο αλγόριθμος σμήνους σωματιδίων προτάθηκε από τους Kennedy και Eberhart (1995) για να προσομοιώσει την κοινωνική συμπεριφορά κάποιων οργανισμών όπως το πέταγμα των πουλιών σε σμήνος και την κίνηση των ψαριών σε κοπάδια. Η αρχική ιδέα ήταν να προσομοιωθεί η κίνηση των πουλιών που πετούν σε σχηματισμό και να αναλυθεί ο τρόπος που αλλάζουν αυτά κατεύθυνση, χωρίς να καταστρέφεται αυτός ο σχηματισμός.

Αρχικά δημιουργείται ένα σύνολο από σωματίδια όπου το καθένα αντιστοιχεί σε μία πιθανή λύση του προβλήματος. Το κάθε σωματίδιο κατέχει μία θέση στο χώρο των λύσεων και κινείται με μία δεδομένη ταχύτητα. Η απόδοση της θέσης, εκτιμάται από μία προκαθορισμένη συνάρτηση ποιότητας (*fitness function*), ενώ η ταχύτητα εκπροσωπεί τις μεταβολές που θα πραγματοποιηθούν για να κινηθεί το σωματίδιο από τη μία θέση στην άλλη. Προς τα πού θα κινηθεί τελικά το σωματίδιο, εξαρτάται από η δυναμική αλληλεπίδραση της δικής του εμπειρίας και της εμπειρίας όλου του σμήνους. Τρεις είναι οι πιθανές κατευθύνσεις που μπορεί να ακολουθήσει ένα σωματίδιο:

- να ακολουθήσει μία δική του διαδρομή
- να κινηθεί προς τη βέλτιστη θέση που είχε κατά τη διάρκεια των επαναλήψεων
- να κινηθεί προς τη θέση που είχε το βέλτιστο σωματίδιο

Ο συγκεκριμένος αλγόριθμος έχει αποδειχτεί πάρα πολύ χρήσιμος σε αρκετά προβλήματα λόγω της εύκολης υλοποίησής του και των πολύ καλών αποτελεσμάτων που έχει δώσει όπου έχει χρησιμοποιηθεί, ιδιαίτερα σε προβλήματα που έχουν συνεχείς μεταβλητές.

5.2.4 Αλγόριθμος βελτιστοποίησης σμήνους πυγολαμπίδων (Glowworm Swarm based Optimization algorithm, GSO)

Στον Αλγόριθμο Βελτιστοποίησης Σμήνους Πυγολαμπίδων (Krishnanand, Ghose, 2009 a και b) έχουμε **n** οντότητες i=1,…,n που αρχικά τοποθετούνται τυχαία στο χώρο λύσεων x_i. Κάθε μία οντότητα αποφασίζει για την κίνησή της με βάση την

ένταση του σήματος που δέχεται από τους γείτονές της. Αυτό είναι περίπου όμοιο με την λουσιφερίνη, την ουσία που προκαλεί λάμψη σε μια πυγολαμπίδα όταν αυτή θέλει να έλξει είτε άλλες πυγολαμπίδες είτε θηράματα. Όσο πιο έντονη είναι η λάμψη, τόσο δυνατότερη είναι η έλξη.

Για κάθε πυγολαμπίδα υπολογίζεται η τιμή της αντικειμενικής συνάρτησης $f(x_i(t))$ και η τιμή της λάμψης της l_i, για την οποία τιμή ενημερώνει τις γειτονικές πυγολαμπίδες. Η γειτονιά μιας πυγολαμπίδας ορίζεται από εκείνες τις πυγολαμπίδες που έχουν υψηλή τιμή λάμψης και η απόσταση τους r_d^i βρίσκεται στο διάστημα $0 < r_d^i < r_s$. Χρησιμοποιώντας μια πιθανότητα p_{ij} μία πυγολαμπίδα επιλέγει τους γείτονες της j και κινείται προς αυτούς. Αρχικά το σύνολο των πυγολαμπίδων έχει την ίδια ποσότητα λάμψης l_0. Σε κάθε επανάληψη του αλγορίθμου υπάρχει η φάση ενημέρωσης της λάμψης και η φάση της κίνησης. Η φάση ενημέρωσης της λάμψης εξαρτάται από την τιμή της αντικειμενικής συνάρτησης στο σημείο που βρίσκεται η πυγολαμπίδα. Κατά τη διάρκεια αυτής της φάσης κάθε πυγολαμπίδα, προσθέτει στην προηγούμενή της ποσότητα μία τιμή που είναι ανάλογη της τωρινής θέσης στο χώρο λύσεων. Επίσης αφαιρείται μία μικρή ποσότητα για να δείξει την εξασθένηση που έχει η πυγολαμπίδα με το χρόνο.

Κατά τη διάρκεια της φάσης της κίνησης κάθε πυγολαμπίδα αποφασίζει, χρησιμοποιώντας μία πιθανότητα, να κινηθεί προς μία πυγολαμπίδα που έχει τιμή λάμψης υψηλότερη από τη δική της. Έτσι, κάθε πυγολαμπίδα έλκεται από μία άλλη πυγολαμπίδα που λάμπει περισσότερο.

5.2.5 Αλγόριθμος Επιλογής Κλώνων (Clonal Selection Algorithm, CSA)

Ο Αλγόριθμος Επιλογής Κλώνων (De Castro, Von Zuben, 2000 και 2002) είναι ο σημαντικότερος αλγόριθμος της θεωρίας των τεχνητών ανοσοποιητικών συστημάτων, αφού είναι ο πλέον κατάλληλος σε σύγκριση με άλλους, για προβλήματα βελτιστοποίησης. Τα Τεχνητά Ανοσοποιητικά Συστήματα (Artificial Immune Systems, AIS) (Dasgupta, 1998 και De Castro, Timmis, 2002) εμπνέονται από τη λειτουργία του φυσικού ανοσοποιητικού συστήματος. Εξάγουν ιδέες και μεταφορές από τη λειτουργία του φυσικού ανοσοποιητικού συστήματος με σκοπό να τις

χρησιμοποιήσουν για την κατασκευή υπολογιστικών μοντέλων για να λύσουν πραγματικά προβλήματα. Η αντιστοίχιση της ορολογίας των φυσικών ανοσοποιητικών συστημάτων με ένα πρόβλημα βελτιστοποίησης είναι η ακόλουθη (Talbi, 2009):

- Το αντίσωμα, αντιστοιχεί στη λύση του προβλήματος
- Η συγγένεια (Affinity) αντιστοιχεί στην αντικειμενική συνάρτηση του προβλήματος
- Το αντιγόνο αντιστοιχεί στο ίδιο το πρόβλημα που πρέπει να επιλυθεί
- Η κλωνοποίηση αντιστοιχεί στη διαδικασία αναπαραγωγής των λύσεων
- Η σωματική μετάλλαξη ή υπερμετάλλαξη (Somatic mutation (hypermutation)) αντιστοιχεί στη διαδικασία της πολλαπλής μετάλλαξης μίας λύσης
- Η διαδικασία ωρίμανσης της συγγένειας (Affinity maturation) αντιστοιχεί στη διαδικασία επιλογής των βέλτιστων λύσεων
- Η διαδικασία διόρθωσης των υποδοχέων (Receptor editing) αντιστοιχεί στη διαδικασία διαφοροποίησης των λύσεων, δηλαδή στη διαδικασία διασποράς των λύσεων σε όλο το χώρο λύσεων του προβλήματος (Diversification).

Για να υλοποιηθεί ο Αλγόριθμος Επιλογής Κλώνων θα πρέπει πρώτα από όλα να γίνει η αρχικοποίηση του πληθυσμού των αντισωμάτων. Οι τιμές που θα πάρει το κάθε αντίσωμα εξαρτάται από το πρόβλημα που επιλύεται. Σε ένα πρόβλημα δρομολόγησης το αντίσωμα αντιστοιχεί σε μια διαδρομή. Έπειτα, για κάθε αντίσωμα υπολογίζεται η τιμή της αντικειμενικής συνάρτησης. Στη συνέχεια επιλέγουμε τις καλύτερες λύσεις του πληθυσμού οι οποίες κλωνοποιούνται και μεταλλάσσονται, για να δημιουργηθεί ο νέος πληθυσμός αντισωμάτων. Ακολούθως ένα μεγάλο μέρος των στοιχείων της λύσης ενός αντισώματος τα οποία επιλέγονται τυχαία μεταλλάσσεται, δηλαδή εφαρμόζουμε ένα τελεστή υπερμετάλλαξης. Έπειτα υπολογίζεται η τιμή της αντικειμενικής συνάρτησης για όλο τον πληθυσμό και επιλέγονται οι καλύτεροι κλώνοι για να πάρουν τη θέση κάποιων κλώνων του αρχικού πληθυσμού, ενώ παράλληλα, δημιουργούνται τυχαία αντισώματα για να αντικαταστήσουν κάποια από τα χειρότερα αντισώματα του αρχικού πληθυσμού.

5.3 Σύγκριση αποτελεσμάτων

Η σύγκριση των αποτελεσμάτων θα παρουσιαστεί σε δύο φάσεις. Στην πρώτη φάση, θα συγκριθούν τα αποτελέσματα που μας έδωσε ο αλγόριθμος ABC με αυτά των αλγόριθμων HBMO και BBMO, ενώ στη δεύτερη φάση, θα γίνουν οι συγκρίσεις των αποτελεσμάτων του ABC, με τα αποτελέσματα όλων των υπολοίπων (PSO, CSA,GSO,DE και GA). Αυτό γίνεται για να έχουμε ποιο συγκεκριμένη εικόνα για τις επιδόσεις που έχουν οι αλγόριθμοι που είναι εμπνευσμένοι από τις κοινωνίες των μελισσών (ABC, HBMO, BBMO) στην επίλυση προβλημάτων δρομολόγησης οχημάτων με στοχαστική ζήτηση και κατ'επέκταση, σε ποιο βαθμό οι επιδόσεις αυτές είναι ανταγωνίσιμες με αυτές των υπόλοιπων μεθευρετικών αλγόριθμων.

5.3.1 Σύγκριση αποτελεσμάτων των αλγόριθμων ABC, HBMO, BBMO

Με τη βοήθεια πινάκων και γραφημάτων, μπορούμε να εξάγουμε χρήσιμα συμπεράσματα όσον αφορά στην απόδοση του αλγόριθμου ABC σε σχέση με τις αποδόσεις των αλγορίθμων BBMO και HBMO. Αρχικά παραθέτουμε έναν συγκεντρωτικό πίνακα, στον οποίο έχουμε καταγράψει τις λύσεις που μας έδωσαν οι τρεις αυτοί αλγόριθμοι σε κάθε πρόβλημα και για κάθε απόκλιση από τη ζήτηση (για κάθε dem), καθώς και τις αποκλίσεις των τιμών αυτών από τη βέλτιστη λύση που έχει καταγραφεί μέχρι τώρα από το σύνολο των μεθευρετικών αλγορίθμων.

			HBMO		BBMO		ABC	
par	**dem**	**Best Cost**	**Best Cost**	**Var %**	**Best Cost**	**Var %**	**Best Cost**	**Var %**
1	0	524.6100	524.6100	0.00	524.6100	0.00	530.4832	1.12
	1	528.5700	528.6300	0.01	531.3900	0.53	530.7427	0.41
	2	531.5400	531.9400	0.08	531.5400	0.00	543.3937	2.23
2	0	836.0700	836.0700	0.00	851.4800	1.84	870.4792	4.12
	1	840.3502	842.7100	0.28	850.9200	1.26	869.9103	3.52
	2	848.7300	848.7300	0.00	852.9100	0.49	862.6649	1.64
3	0	837.7600	852.9400	1.81	837.7600	0.00	872.2241	4.11
	1	836.8800	836.8800	0.00	849.1700	1.47	859.8489	2.74
	2	842.4500	846.4000	0.47	850.7400	0.98	870.6354	3.35
4	0	1074.4000	1080.1000	0.53	1095.8000	1.99	1121.6710	4.40
	1	1068.2000	1087.7000	1.83	1093.7000	2.39	1118.5480	4.71
	2	1065.4000	1089.5000	2.26	1084.6000	1.80	1114.2460	4.58

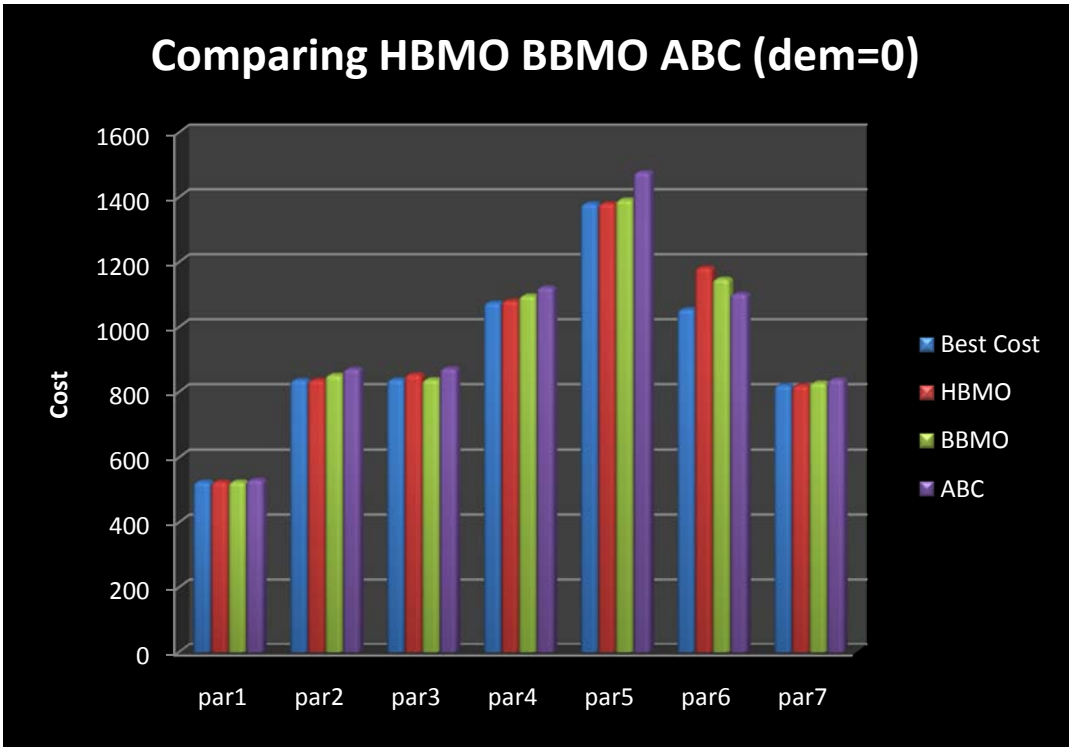
			HBMO		BBMO		ABC	
par	**dem**	**Best Cost**	**Best Cost**	**Var %**	**Best Cost**	**Var %**	**Best Cost**	**Var %**
5	0	1379.8000	1379.8000	0.00	1391.6000	0.86	1476.7440	7.03
	1	1370.2000	1378.2000	0.58	1406.7000	2.66	1536.3930	12.13
	2	1387.0000	1399.7000	0.92	1404.5000	1.26	1501.2570	8.24
6	0	1054.1000	1182.1000	12.14	1147.6000	8.87	1101.6710	4.51
	1	1075.3000	1170.7000	8.87	1155.5000	7.46	1128.8747	4.98
	2	1095.3000	1110.8000	1.42	1207.8000	10.27	1153.9250	5.35
7	0	819.5600	819.6000	0.00	827.5100	0.97	837.4121	2.18
	1	851.4700	856.4800	0.59	851.4700	0.00	881.7024	3.55
	2	858.8500	861.2000	0.27	873.3800	1.69	887.0112	3.28

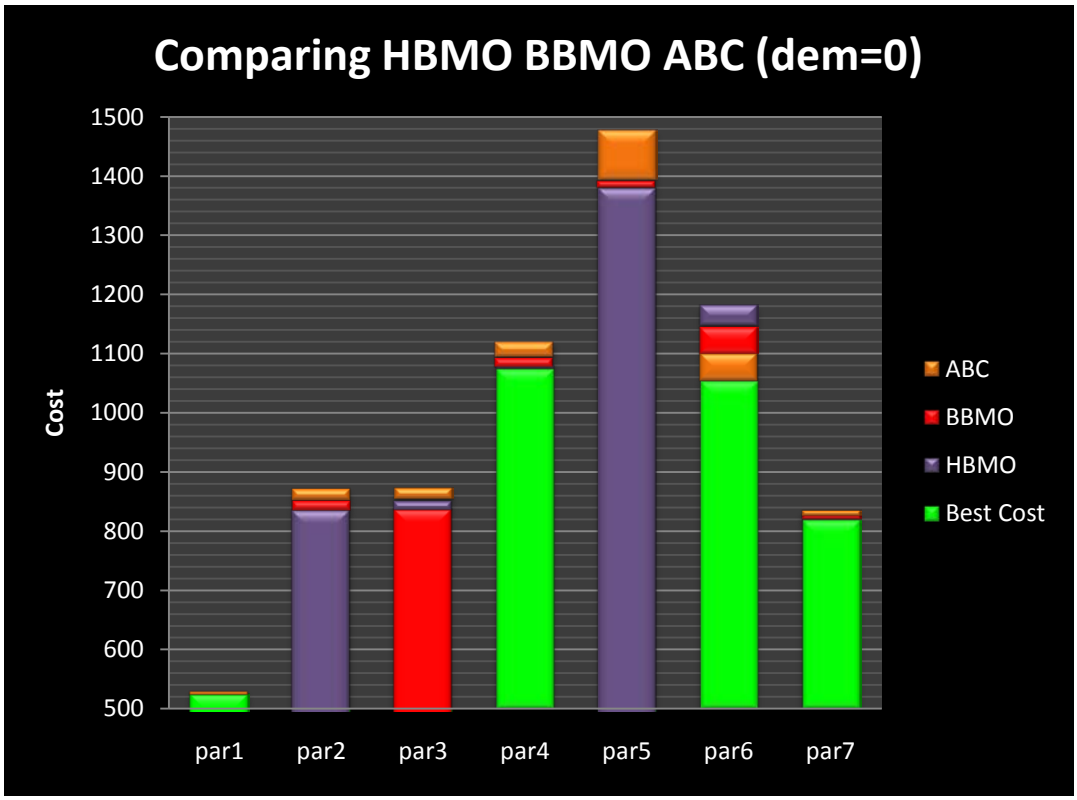
Πίνακας 1. Συνοπτικά τα αποτελέσματα και οι αποκλίσεις των αλγόριθμων HBMO, BBMO και ABC

Στη συνέχεια, έχουμε δημιουργήσει πίνακες και γραφήματα, έχοντας δώσει σε κάθε αλγόριθμο ένα διαφορετικό χρωματισμό, πράγμα που μας βοηθάει στην οπτικοποίηση των επιδόσεων των αλγορίθμων, σε όλα τα προβλήματα, για κάθε dem ξεχωριστά.

dem=0								
par1	**Best Cost**	**HBMO**		**BBMO**		**ABC**		
	524.61	**524.61**	**0.00%**	**524.61**	**0.00%**	**530.4832**	**1.12%**	
par2	**Best Cost**	**HBMO**		**BBMO**		**ABC**		
	836.07	**836.07**	**0.00%**	**851.48**	**1.84%**	**870.4792**	**4.12%**	
par3	**Best Cost**	**BBMO**		**HBMO**		**ABC**		
	837.76	**837.76**	**0.00%**	**852.94**	**1.81%**	**872.2241**	**4.11%**	
par4	**Best Cost**	**HBMO**		**BBMO**		**ABC**		
	1074.40	**1080.1**	**0.53%**	**1095.8**	**1.99%**	**1121.671**	**4.40%**	
par5	**Best Cost**	**HBMO**		**BBMO**		**ABC**		
	1379.80	**1379.8**	**0.00%**	**1391.6**	**0.86%**	**1476.744**	**7.03%**	
par6	**Best Cost**	**ABC**		**BBMO**		**HBMO**		
	1054.10	**1101.671**	**4.51%**	**1147.6**	**8.87%**	**1182.1**	**12.14%**	
par7	**Best Cost**	**HBMO**		**BBMO**		**ABC**		
	819.56	**819.6**	**0.00%**	**827.51**	**0.97%**	**837.41**	**2.18%**	

Πίνακας 2. Κατάταξη των αλγόριθμων σύμφωνα με την απόδοσή τους (dem=0)


Comparing HBMO BBMO ABC (dem=0)
Cost
1600
1400
1200
1000
800
600
400
200
0
par1
par2
par3
par4
par5
par6
par7
Best Cost
HBMO
BBMO
ABC


Comparing HBMO BBMO ABC (dem=0)
Cost
1500
1400
1300
1200
1100
1000
900
800
700
600
500
par1
par2
par3
par4
par5
par6
par7
ABC
BBMO
HBMO
Best Cost

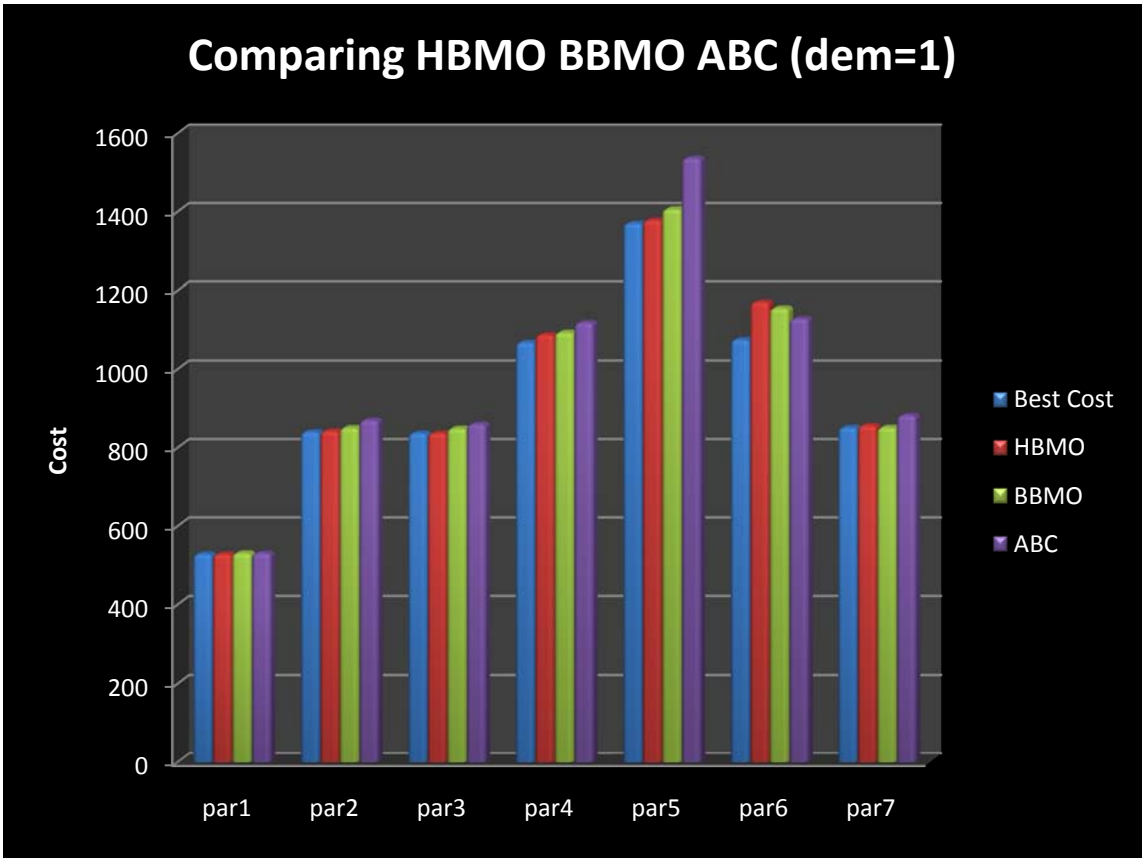
Σχολιασμός αποτελεσμάτων των αλγόριθμων HBMO, BBMO, ABC για dem=0

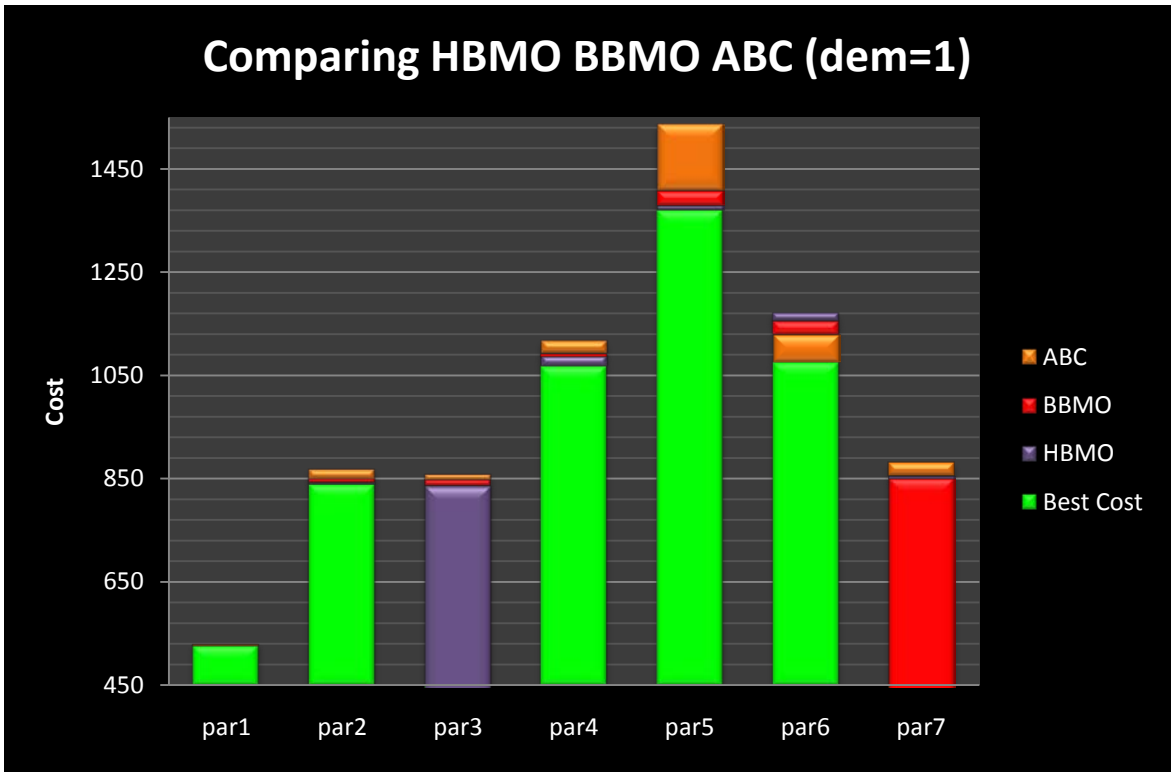
Ο αλγόριθμος HBMO δίνει τη βέλτιστη λύση για το par2 και par5, ενώ ο αλγόριθμος BBMO δίνει βέλτιστη λύση για το par3. Η ABC είχε σταθερά τη χειρότερη απόδοση με μοναδική εξαίρεση το par6 όπου και επιφέρει την καλύτερη λύση μεταξύ των τριών, όχι όμως τη βέλτιστη. Οι HBMO και BBMO εμφανίζουν παρόμοια μεταξύ τους αποτελέσματα. Η γενικότερα καλή τους απόδοση ανατρέπεται στο par6.

Συνεχίζουμε με τον ίδιο τρόπο για τα αποτελέσματα από την επίλυση των προβλημάτων με απόκλιση από τη ζήτηση ±1 (dem=1).

dem=1							
par1	Best Cost	HBMO		ABC		BBMO	
	528.57	528.63	0.01%	530.7427	0.41%	531.39	0.53%
par2	Best Cost	HBMO		BBMO		ABC	
	840.35	842.71	0.28%	850.92	1.26%	869.9103	3.52%
par3	Best Cost	HBMO		BBMO		ABC	
	836.88	836.88	0.00%	849.17	1.47%	859.8489	2.74%
par4	Best Cost	HBMO		BBMO		ABC	
	1068.20	1087.7	1.83%	1093.7	2.39%	1118.548	4.71%
par5	Best Cost	HBMO		BBMO		ABC	
	1370.20	1378.2	0.58%	1406.7	2.66%	1536.393	12.13%
par6	Best Cost	ABC		BBMO		HBMO	
	1075.30	1128.87	4.98%	1155.5	7.46%	1170.7	8.87%
par7	Best Cost	BBMO		HBMO		ABC	
	851.47	851.47	0.00%	856.48	0.59%	881.70	3.55%

Πίνακας 3. Κατάταξη των αλγοριθμων σύμφωνα με την απόδοσή τους (dem=1)


Comparing HBMO BBMO ABC (dem=1)
Cost
1600
1400
1200
1000
800
600
400
200
0
par1
par2
par3
par4
par5
par6
par7
Best Cost
HBMO
BBMO
ABC


Comparing HBMO BBMO ABC (dem=1)
Cost
1450
1250
1050
850
650
450
par1
par2
par3
par4
par5
par6
par7
ABC
BBMO
HBMO
Best Cost

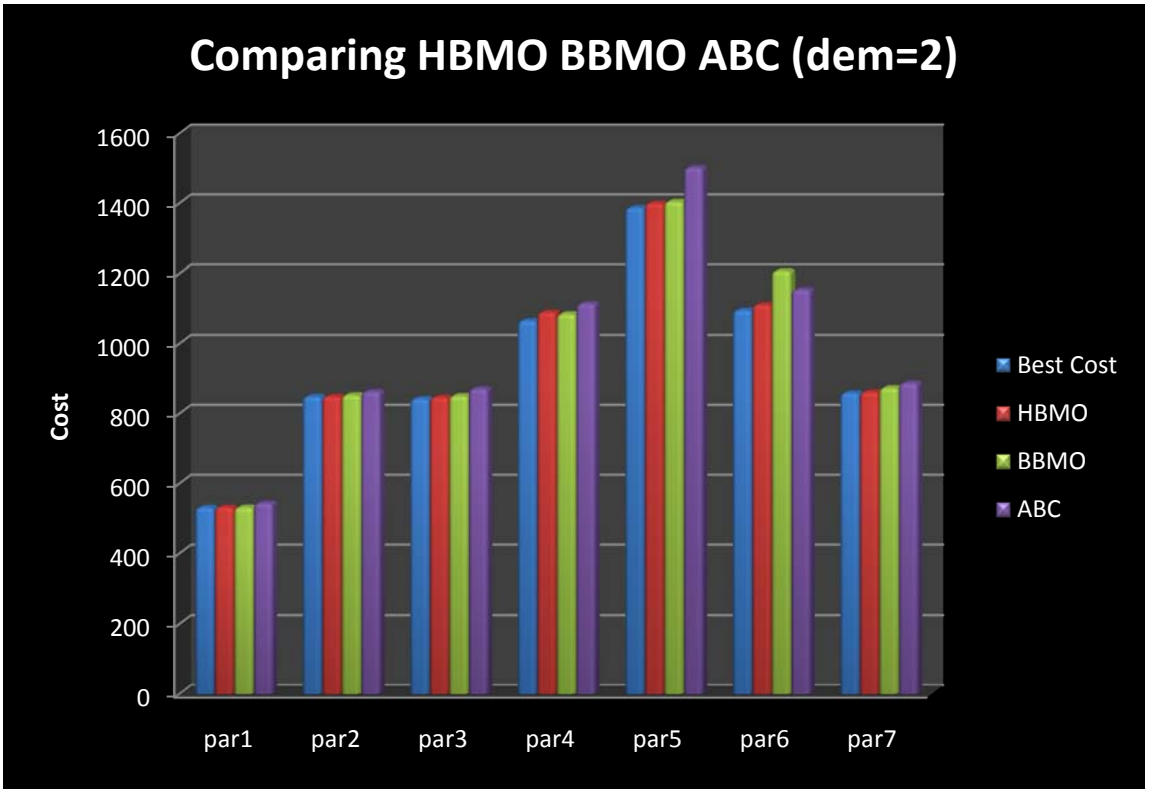
Σχολιασμός αποτελεσμάτων των αλγόριθμων HBMO, BBMO, ABC για dem=1

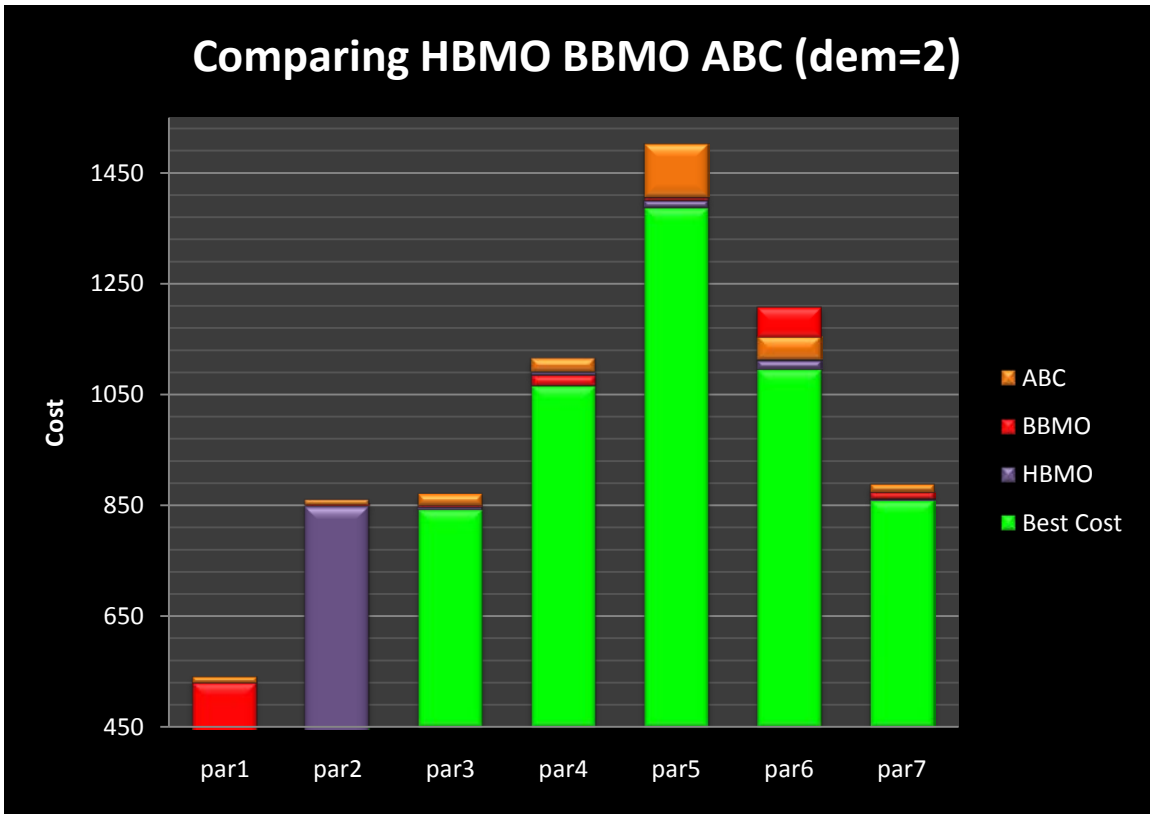
Ο αλγόριθμος HBMO δίνει τη βέλτιστη λύση για το par3, ενώ ο αλγόριθμος BBMO δίνει βέλτιστη λύση για το par7. Γενικότερα, από το par1 έως και το par4 όλες οι μέθοδοι βρήκαν λύσεις κοντά στο βέλτιστο, με τον ABC να υστερεί ελαφρώς στα par2, par3 και par4. Ο ABC παρουσιάζει αξιοπρεπή αποτελέσματα για το par1 έως και το par4 αλλά υστερεί σημαντικά στο par5 και λιγότερο, στο par7. Αντιθέτως, εμφάνισε τα καλύτερα αποτελέσματα για το par6 από τους άλλους δύο αλγόριθμους. Οι HBMO και BBMO παρουσιάζουν σταθερά καλά αποτελέσματα στα par1 έως par4 και στο par7. Στο par6 η απόδοση των δύο αλγορίθμων ήταν και πάλι κακή.

Ομοίως παρουσιάζουμε και τα αποτελέσματα από την επίλυση των προβλημάτων με απόκλιση από τη ζήτηση ±2 (dem=2).

dem=2							
par1	Best Cost	BBMO		HBMO		ABC	
	531.54	531.54	0.00%	531.94	0.08%	543.3937	2.23%
par2	Best Cost	HBMO		BBMO		ABC	
	848.73	848.73	0.00%	852.91	0.49%	862.6649	1.64%
par3	Best Cost	HBMO		BBMO		ABC	
	842.45	846.4	0.47%	850.74	0.98%	870.6354	3.35%
par4	Best Cost	BBMO		HBMO		ABC	
	1065.40	1084.6	1.80%	1089.5	2.26%	1114.246	4.58%
par5	Best Cost	HBMO		BBMO		ABC	
	1387.00	1399.7	0.92%	1404.5	1.26%	1501.257	8.24%
par6	Best Cost	HBMO		ABC		BBMO	
	1095.30	1110.8	1.42%	1153.93	5.35%	1207.8	10.27%
par7	Best Cost	HBMO		BBMO		ABC	
	858.85	861.2	0.27%	873.38	1.69%	887.01	3.28%

Πίνακας 4. Κατάταξη των αλγοριθμων σύμφωνα με την απόδοσή τους (dem=2)


Comparing HBMO BBMO ABC (dem=2)
Cost
1600
1400
1200
1000
800
600
400
200
0
par1
par2
par3
par4
par5
par6
par7
Best Cost
HBMO
BBMO
ABC


Comparing HBMO BBMO ABC (dem=2)
Cost
1450
1250
1050
850
650
450
par1
par2
par3
par4
par5
par6
par7
ABC
BBMO
HBMO
Best Cost

Σχολιασμός αποτελεσμάτων των αλγόριθμων HBMO,BBMO,ABC για dem=2

Ο αλγόριθμος HBMO δίνει τη βέλτιστη λύση για το par1, ενώ ο αλγόριθμος BBMO δίνει βέλτιστη λύση για το par2. Για τα par1 έως par4 όλες οι μέθοδοι παρουσίασαν αποδεχτές λύσεις. Ο αλγόριθμος ABC εμφάνισε τη χειρότερη λύση σε όλα τα προβλήματα πλην του par6 όπου ο BBMO εμφάνισε ιδιαίτερα κακό αποτέλεσμα.

5.3.2 Σύγκριση αποτελεσμάτων των αλγόριθμων ABC, PSO, CSA, GSO, DE και GA

Με τη βοήθεια πινάκων και γραφημάτων, μπορούμε να εξάγουμε χρήσιμα συμπεράσματα όσον αφορά στην απόδοση του αλγόριθμου ABC σε σχέση με τις αποδόσεις των αλγορίθμων PSO, CSA, DE και GA. Αρχικά παραθέτουμε έναν συγκεντρωτικό πίνακα, στον οποίο έχουμε καταγράψει τις λύσεις που μας έδωσαν οι έξι αυτοί αλγόριθμοι (για πρακτκούς λόγους στον πίνακα παρουσιάζονται ανά τρεις) σε κάθε πρόβλημα και για κάθε απόκλιση από τη ζήτηση (για κάθε dem), καθώς και τις αποκλίσεις των τιμών αυτών από τη βέλτιστη λύση που έχει καταγραφεί μέχρι τώρα από το σύνολο των μεθευρετικών αλγορίθμων.

			GA		DE		PSO	
par	**dem**	**Best Cost**	**Best Cost**	**Var %**	**Best Cost**	**Var %**	**Best Cost**	**Var %**
1	0	524.6100	542.6200	3.43	537.4200	2.44	524.6100	0.00
	1	528.5700	543.8000	2.88	538.3500	1.85	528.5700	0.00
	2	531.5400	540.5900	1.70	536.6900	0.97	533.1500	0.30
2	0	836.0700	867.8600	3.80	862.3900	3.15	842.6800	0.79
	1	840.3502	878.4500	4.53	862.7200	2.66	849.0500	1.04
	2	848.7300	886.8700	4.49	865.8600	2.02	854.4000	0.67
3	0	837.7600	850.6600	1.54	839.4000	0.20	839.3900	0.19
	1	836.8800	854.5100	2.11	866.1500	3.50	843.3500	0.77
	2	842.4500	854.1800	1.39	851.5200	1.08	842.4500	0.00
4	0	1074.4000	1137.2400	5.85	1095.1800	1.93	1086.7000	1.14
	1	1068.2000	1128.2100	5.62	1115.3700	4.42	1108.1000	3.74
	2	1065.4000	1141.1800	7.11	1139.0000	6.91	1117.0000	4.84
5	0	1379.8000	1511.2100	9.52	1495.2800	8.37	1480.7000	7.31
	1	1370.2000	1505.3700	9.86	1499.1700	9.41	1483.4000	8.26
	2	1387.0000	1497.8400	7.99	1489.5000	7.39	1487.4000	7.24
6	0	1054.1000	1061.4700	0.70	1055.8700	0.17	1054.1000	0.00
	1	1075.3000	1085.4900	0.95	1088.7400	1.25	1075.3000	0.00
	2	1095.3000	1117.1100	1.99	1098.7000	0.31	1095.3000	0.00
7	0	819.5600	835.1800	1.91	823.4700	0.48	819.5600	0.00
	1	851.4700	862.4900	1.29	859.7900	0.98	856.6800	0.61
	2	858.8500	864.4600	0.65	861.2800	0.28	858.8500	0.00

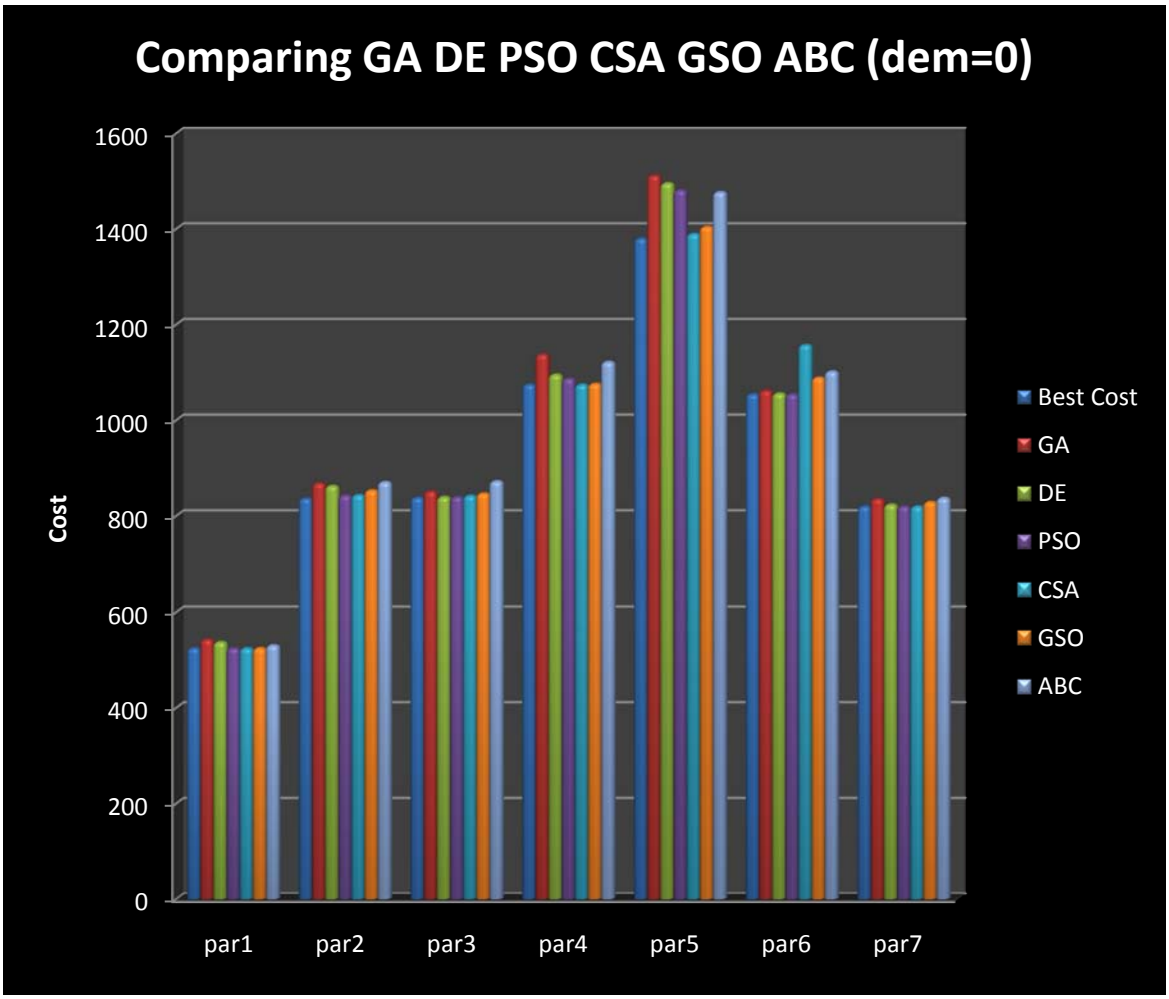
			CSA		GSO		ABC	
par	**dem**	**Best Cost**	**Best Cost**	**Var %**	**Best Cost**	**Var %**	**Best Cost**	**Var %**
1	0	524.6100	524.6100	0.00	524.6111	0.00	530.4832	1.12
	1	528.5700	530.4000	0.35	528.6329	0.01	530.7427	0.41
	2	531.5400	531.5400	0.00	532.9107	0.26	543.3937	2.23
2	0	836.0700	843.3400	0.87	853.1535	2.04	870.4792	4.12
	1	840.3502	844.6600	0.51	840.3502	0.00	869.9103	3.52
	2	848.7300	849.7900	0.12	860.6327	1.40	862.6649	1.64
3	0	837.7600	841.9900	0.50	846.5360	1.05	872.2241	4.11
	1	836.8800	840.3600	0.42	837.4868	0.07	859.8489	2.74
	2	842.4500	843.4000	0.11	848.0413	0.66	870.6354	3.35
4	0	1074.4000	1074.4000	0.00	1075.9000	0.14	1121.6710	4.40
	1	1068.2000	1068.2000	0.00	1081.8000	1.27	1118.5480	4.71
	2	1065.4000	1065.4000	0.00	1085.9151	1.93	1114.2460	4.58
5	0	1379.8000	1389.3000	0.69	1404.1000	1.76	1476.7440	7.03
	1	1370.2000	1370.2000	0.00	1387.0000	1.23	1536.3930	12.13
	2	1387.0000	1388.4000	0.10	1387.0000	0.00	1501.2570	8.24
6	0	1054.1000	1157.0000	9.76	1088.6000	3.27	1101.6710	4.51
	1	1075.3000	1172.7000	9.06	1097.5000	2.06	1128.8747	4.98
	2	1095.3000	1156.8000	5.61	1114.5000	1.75	1153.9250	5.35
7	0	819.5600	819.5600	0.00	828.8530	1.13	837.4121	2.18
	1	851.4700	860.2900	1.04	867.1657	1.84	881.7024	3.55
	2	858.8500	859.5300	0.08	859.7978	0.11	887.0112	3.28

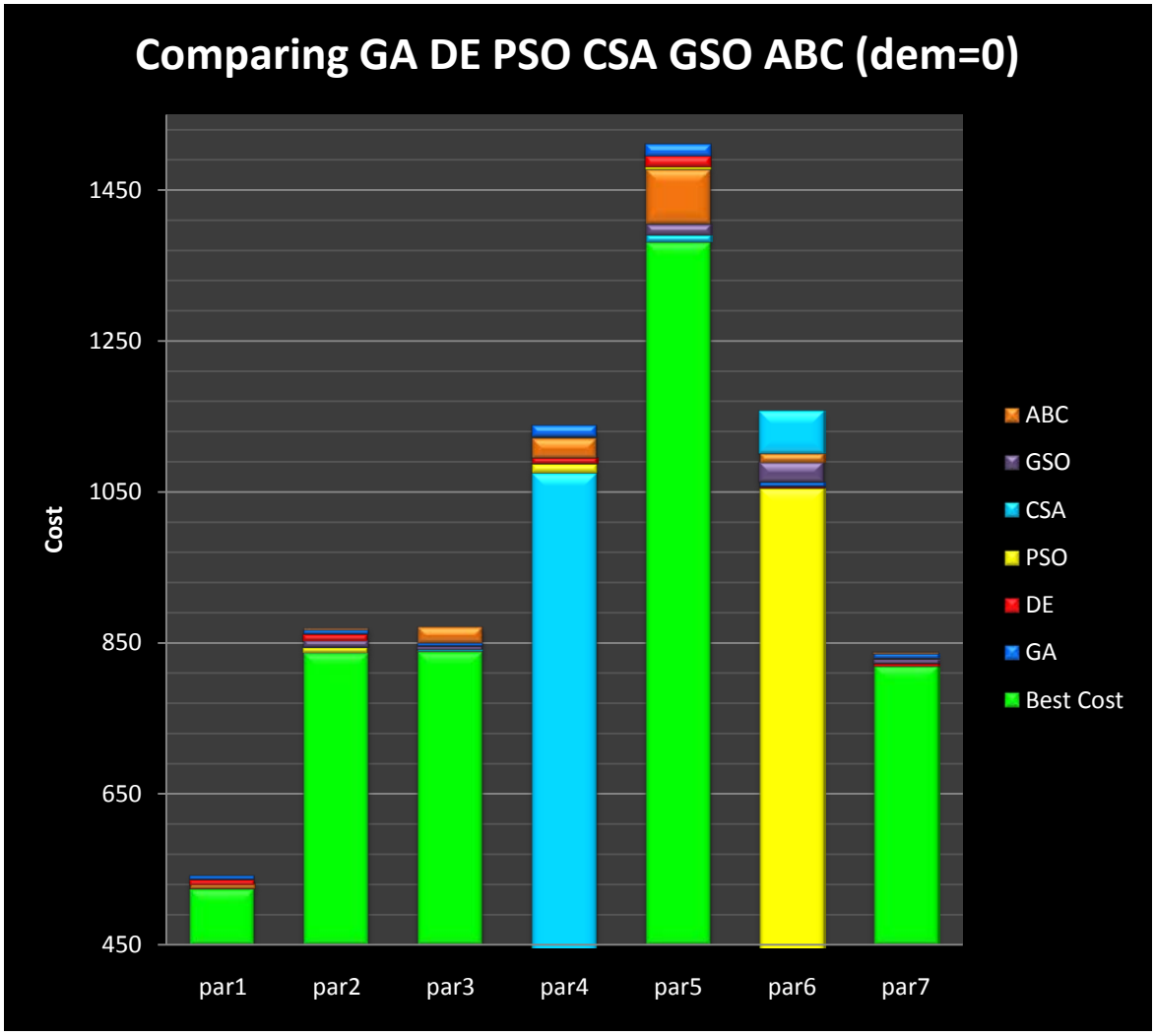
Πίνακας 5. Συνοπτικά τα αποτελέσματα και οι αποκλίσεις των αλγόριθμων GA, DE, PSO, CSA, GSO και ABC

Στη συνέχεια, έχουμε δημιουργήσει πίνακες και γραφήματα, έχοντας δώσει σε κάθε αλγόριθμο ένα διαφορετικό χρωματισμό, πράγμα που μας βοηθάει στην οπτικοποίηση των επιδόσεων των αλγορίθμων, σε όλα τα προβλήματα, για κάθε dem ξεχωριστά.

dem=0													
par1	Best Cost	PSO		CSA		GSO		ABC		DE		GA	
	524.61	524.61	0.00%	524.61	0.00%	524.61	0.00%	530.48	1.12%	537.42	2.44%	542.62	3.43%
par2	Best Cost	PSO		CSA		GSO		DE		GA		ABC	
	836.07	842.68	0.79%	843.34	0.87%	853.15	1.24%	862.39	2.34%	867.86	2.99%	870.48	4.12%
par3	Best Cost	PSO		DE		CSA		GSO		GA		ABC	
	837.76	839.39	0.19%	839.40	0.20%	841.99	0.31%	846.54	0.85%	850.66	1.34%	872.22	4.11%
par4	Best Cost	CSA		GSO		PSO		DE		ABC		GA	
	1074.40	1074.40	0.00%	1075.90	0.14%	1086.70	1.14%	1095.18	1.93%	1121.67	3.22%	1137.24	5.85%
par5	Best Cost	CSA		GSO		ABC		PSO		DE		GA	
	1379.80	1389.30	0.69%	1404.10	1.76%	1476.74	6.29%	1480.70	6.58%	1495.28	7.63%	1511.21	9.52%
par6	Best Cost	PSO		DE		GA		GSO		ABC		CSA	
	1054.10	1054.10	0.00%	1055.87	0.17%	1061.47	0.70%	1088.60	3.27%	1101.67	4.51%	1157.00	9.76%
par7	Best Cost	PSO		CSA		DE		GSO		GA		ABC	
	819.56	819.56	0.00%	819.56	0.00%	823.47	0.48%	828.85	1.13%	835.18	1.91%	837.41	2.18%

Πίνακας 6. Κατάταξη των αλγοριθμων σύμφωνα με την απόδοσή τους (dem=0)





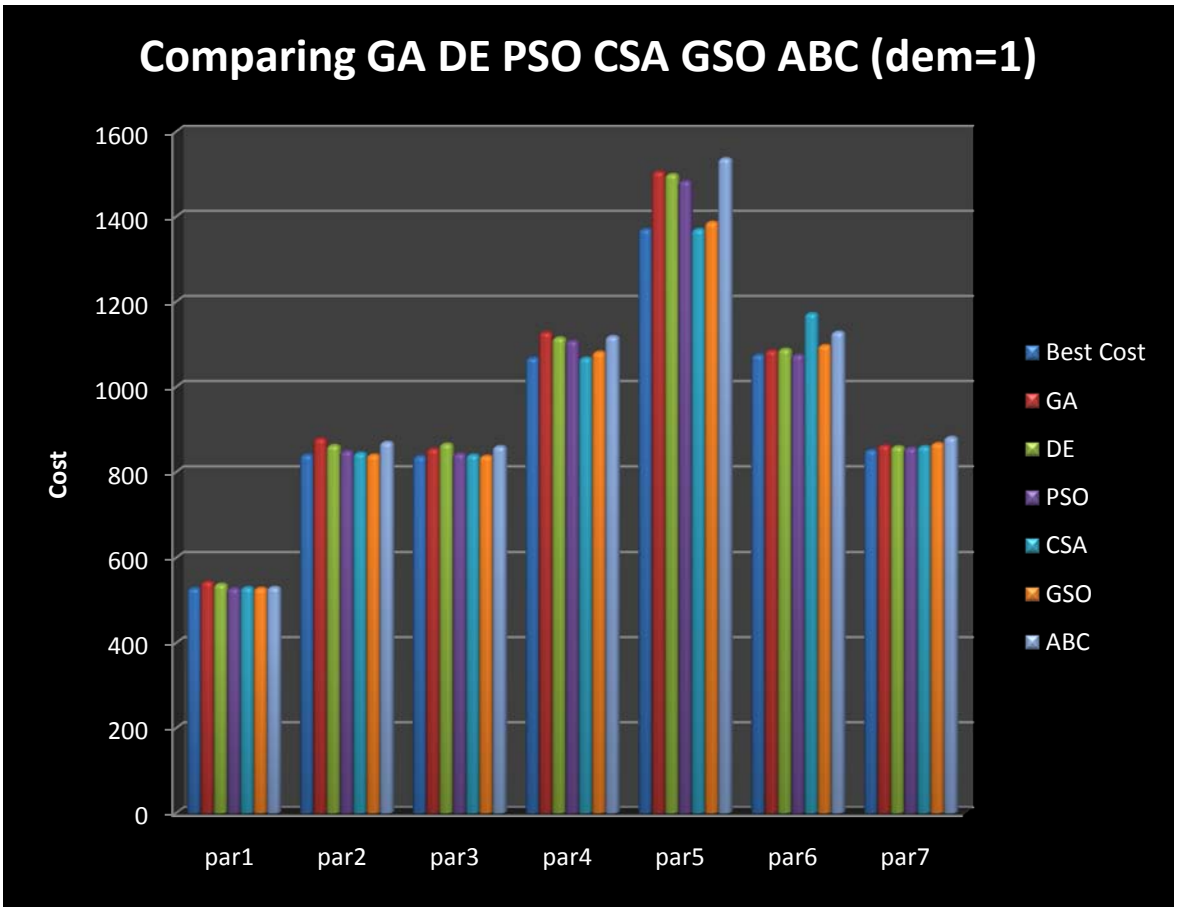
Σχολιασμός αποτελεσμάτων των αλγόριθμων GA, DE, PSO, CSA, GSO και ABC για dem=0

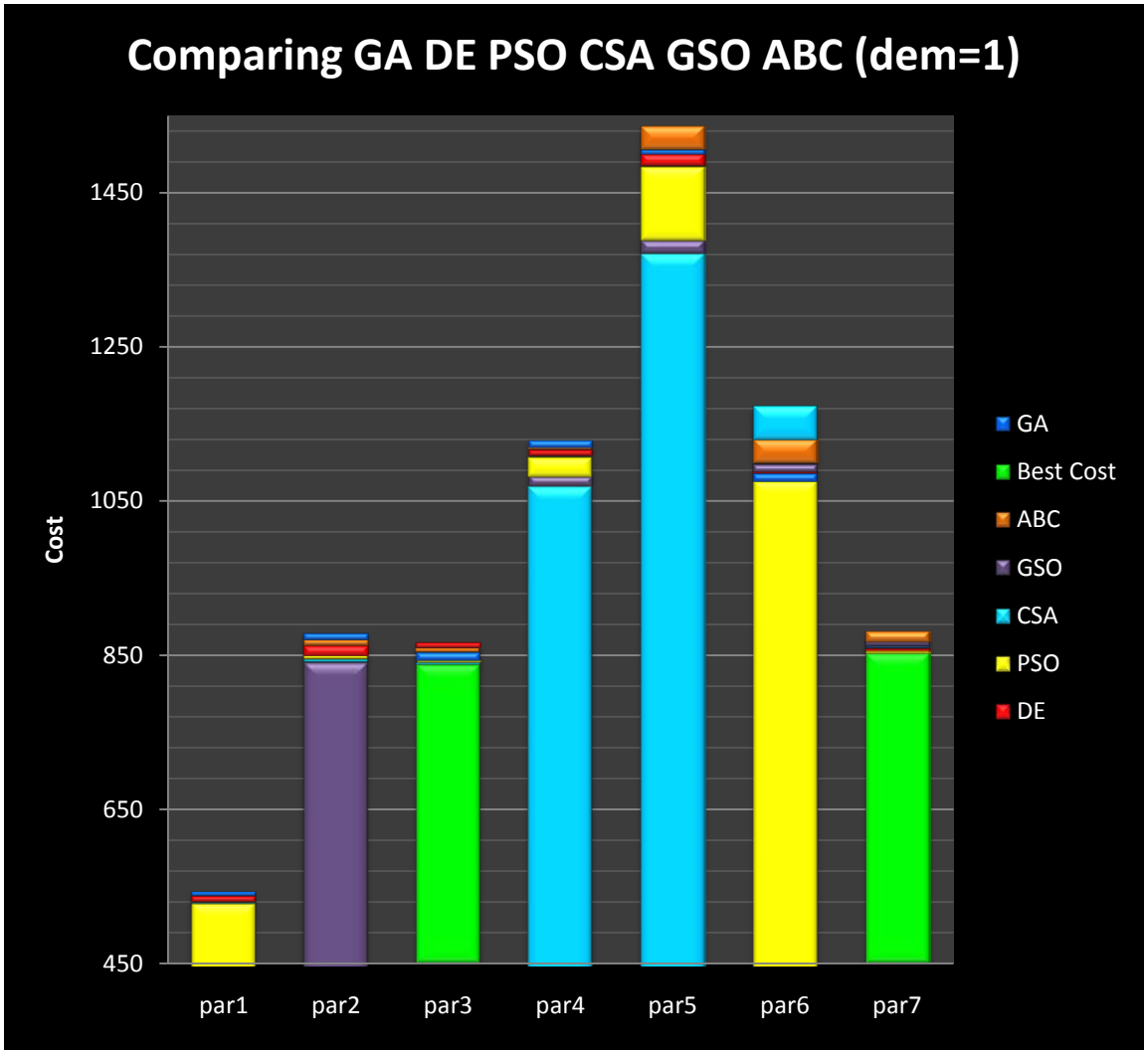
Ο αλγόριθμος CSA δίνει τη βέλτιστη λύση για το par4, ενώ ο αλγόριθμος PSO δίνει βέλτιστη λύση για το par6. Γενικότερα, οι μέθοδοι παρουσίασαν παρόμοια αποτελέσματα για τα προβλήματα par1, par2, par3 με τις λύσεις του αλγόριθμου ABC να είναι λίγο χειρότερες για τα par2 και par3. Στο par4 οι αλγόριθμοι GA και ABC είχαν τα χειρότερα αποτελέσματα. Στο par5 όλες οι μέθοδοι πλην των CSA και GSO παρουσίασαν κακές λύσεις. Στο par6 υπήρξε ιδιαίτερα κακή απόδοση από την CSA και σε μικρότερο βαθμό, από τις GSO και ABC. Στο par7, τέλος, όλες οι μέθοδοι εμφάνισαν παρόμοια, αποδεκτά αποτελέσματα.

Συνεχίζουμε με τον ίδιο τρόπο για τα αποτελέσματα από την επίλυση των προβλημάτων με απόκλιση από τη ζήτηση ±1 (dem=1).

dem=1													
par1	Best Cost	PSO		GSO		CSA		ABC		DE		GA	
	528.57	528.57	0.00%	528.63	0.01%	530.40	0.35%	530.74	0.41%	538.35	1.85%	543.80	2.88%
par2	Best Cost	GSO		CSA		PSO		DE		ABC		GA	
	840.35	840.35	0.00%	844.66	0.51%	849.05	1.04%	862.72	2.66%	869.91	3.52%	878.45	4.53%
par3	Best Cost	GSO		CSA		PSO		GA		ABC		DE	
	836.88	837.49	0.07%	840.36	0.42%	843.35	0.70%	854.51	2.03%	859.85	2.74%	866.15	3.42%
par4	Best Cost	CSA		GSO		PSO		DE		ABC		GA	
	1068.20	1068.20	0.00%	1081.80	1.27%	1108.10	3.74%	1115.37	4.42%	1118.55	4.71%	1128.21	5.62%
par5	Best Cost	CSA		GSO		PSO		DE		GA		ABC	
	1370.20	1370.20	0.00%	1387.00	1.23%	1483.40	8.26%	1499.17	9.41%	1505.37	9.86%	1536.39	12.13%
par6	Best Cost	PSO		GA		DE		GSO		ABC		CSA	
	1075.30	1075.30	0.00%	1085.49	0.95%	1088.74	1.25%	1097.50	2.06%	1128.87	4.98%	1172.70	9.06%
par7	Best Cost	PSO		DE		CSA		GA		GSO		ABC	
	851.47	856.68	0.61%	859.79	0.98%	860.29	0.42%	862.49	0.68%	867.17	1.22%	881.70	3.55%

Πίνακας 7. Κατάταξη των αλγοριθμων σύμφωνα με την απόδοσή τους (dem=1)





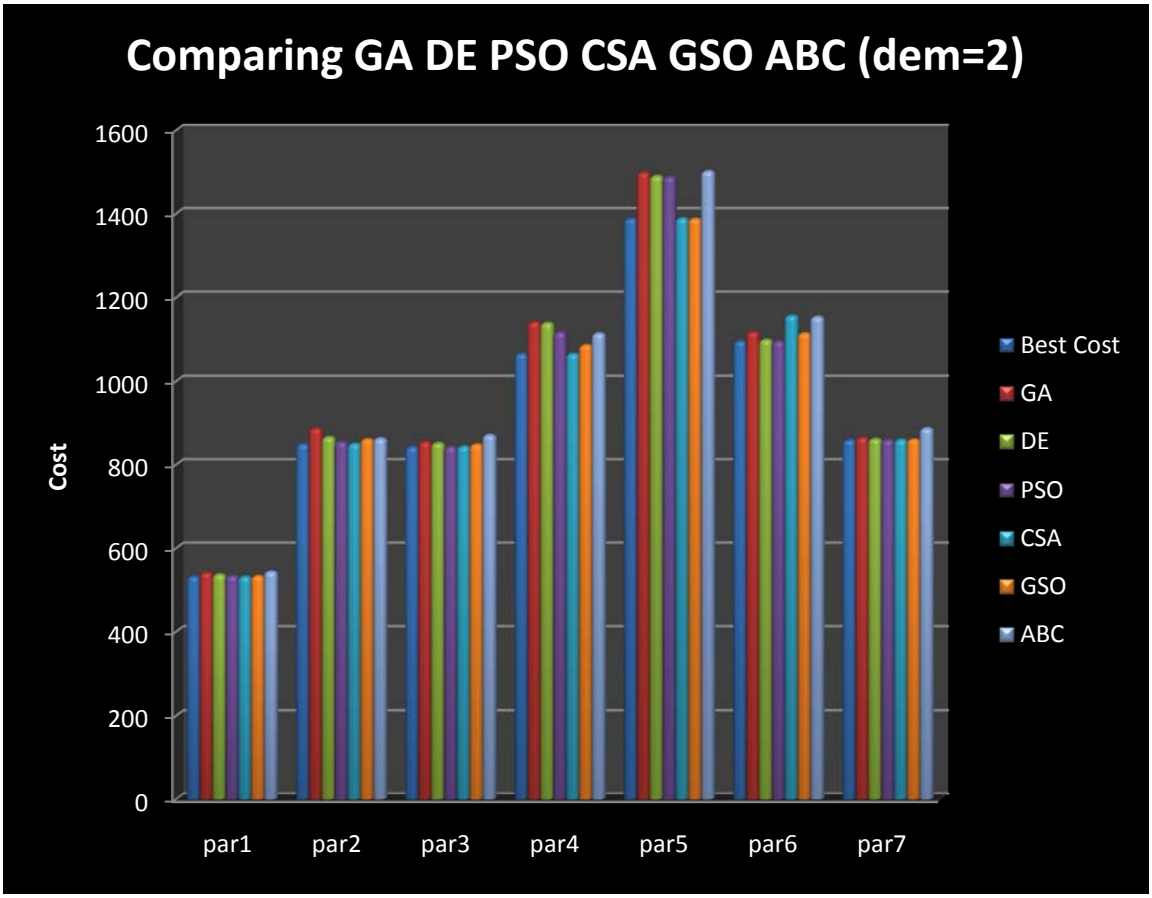
Σχολιασμός αποτελεσμάτων των αλγόριθμων GA, DE, PSO, CSA, GSO και ABC για dem=1

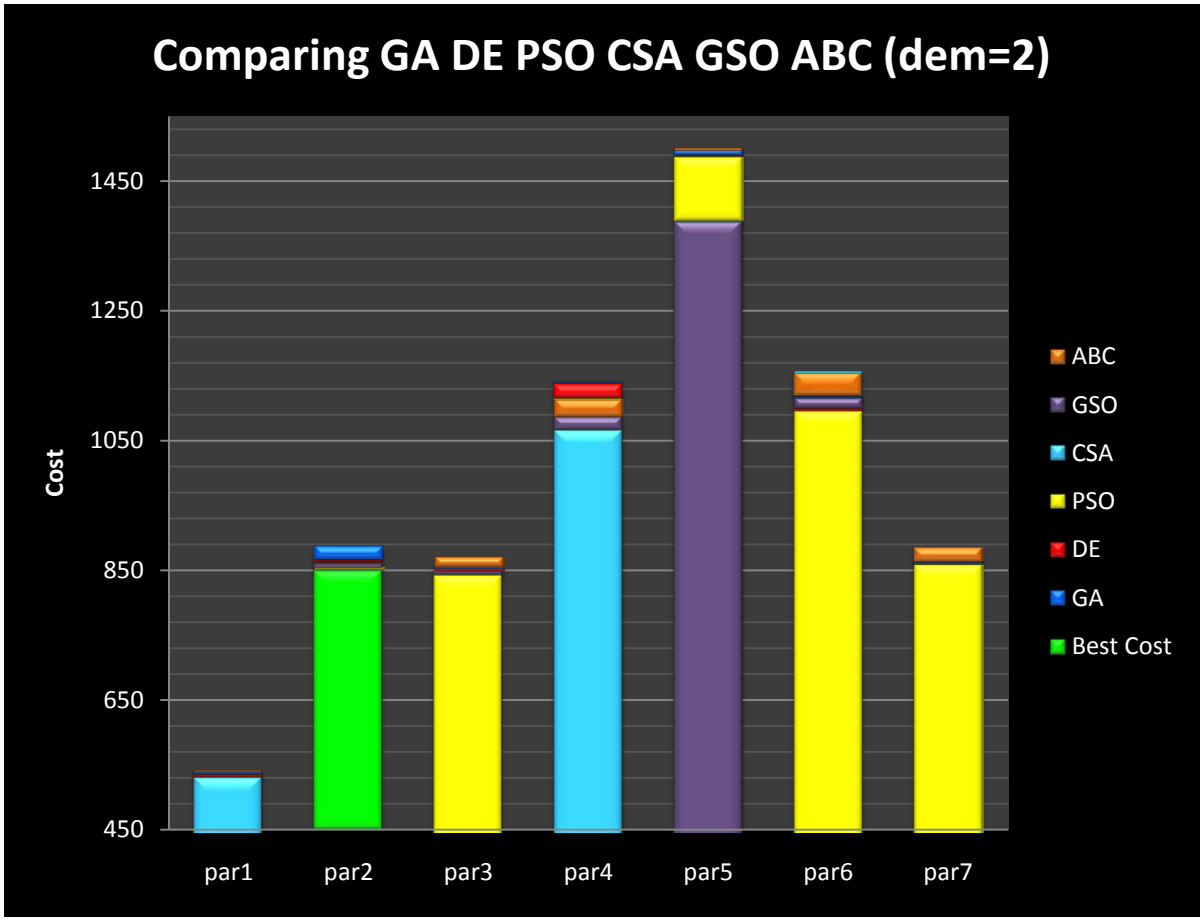
Ο αλγόριθμος PSO δίνει τη βέλτιστη λύση για τα par1 και par6, ο αλγόριθμος GSO δίνει βέλτιστη λύση για το par2, ενώ ο CSA δίνει βέλτιστα στα par4 και par5. Στο par1 όλες οι μέθοδοι πλησίασαν το βέλτιστο του PSO. Η ABC είχε κακή απόδοση σε όλες τις περιπτώσεις εκτός του par1. Στα par5 και par6 μάλιστα, το κόστος της λύσης ήταν ιδιαίτερα υψηλό. Γενικότερα, οι CSA και GSO ήταν σταθερά καλύτερες των άλλων μεθόδων, με εξαίρεση το par6 όπου παρουσιάζουν από τα χειρότερα αποτελέσματα.

Ομοίως παρουσιάζουμε και τα αποτελέσματα από την επίλυση των προβλημάτων με απόκλιση από τη ζήτηση ±2 (dem=2).

dem=2													
par1	Best Cost	CSA		GSO		PSO		DE		GA		ABC	
	531.54	531.54	0.00%	532.91	0.26%	533.15	0.30%	536.69	0.97%	540.59	1.70%	543.39	2.23%
par2	Best Cost	CSA		PSO		GSO		ABC		DE		GA	
	848.73	849.79	0.12%	854.40	0.67%	860.63	1.28%	862.66	1.52%	865.86	1.89%	886.87	4.49%
par3	Best Cost	PSO		CSA		GSO		DE		GA		ABC	
	842.45	842.45	0.00%	843.40	0.11%	848.04	0.66%	851.52	1.08%	854.18	1.39%	870.64	3.35%
par4	Best Cost	CSA		GSO		ABC		PSO		DE		GA	
	1065.40	1065.40	0.00%	1085.92	1.93%	1114.25	4.58%	1117.00	4.84%	1139.00	6.91%	1141.18	7.11%
par5	Best Cost	GSO		CSA		PSO		DE		GA		ABC	
	1387.00	1387.00	0.00%	1388.40	0.10%	1487.40	7.24%	1489.50	7.39%	1497.84	7.99%	1501.26	8.24%
par6	Best Cost	PSO		DE		GSO		GA		ABC		CSA	
	1095.30	1095.30	0.00%	1098.70	0.31%	1114.50	1.75%	1117.11	1.99%	1153.93	5.35%	1156.80	5.61%
par7	Best Cost	PSO		CSA		GSO		DE		GA		ABC	
	858.85	858.85	0.00%	859.53	0.08%	859.80	0.11%	861.28	0.28%	864.46	0.65%	887.01	3.28%

Πίνακας 8. Κατάταξη των αλγοριθμων σύμφωνα με την απόδοσή τους (dem=2)





Σχολιασμός αποτελεσμάτων των αλγόριθμων GA, DE, PSO, CSA, GSO και ABC για dem=2

Ο αλγόριθμος PSO θα μπορούσαμε να πούμε οτι 'δεσπόζει', αφού δίνει τη βέλτιστη λύση για τα par3, par6 και par7. Ακολουθεί ο αλγόριθμος GSO ο οποίος δίνει βέλτιστη λύση για το par5, ενώ ο CSA δίνει βέλτιστα στα par1 και par4. Στο par1 όλες οι μέθοδοι πλησίασαν ιδιαίτερα, το βέλτιστο του PSO, ενώ εμφανίζουν παρόμοιες λύσεις για τα par2 και par3. Στο par5 είναι άξιο αναφοράς οτι η απόκλιση του CSA από το βέλτιστο του GSO είναι μόλις 0,1%, σχεδόν μηδενική, ενώ οι υπόλοιπες μέθοδοι υστερούν σημαντικά. Οι ABC και CSA έχουν τα χειρότερα αποτελέσματα από όλες τις μεθόδους στο par6 με σημαντική διαφορά, της τάξης του 3,5%. Στο par7 όλες οι μέθοδοι είχαν αποτελέσματα κοντά στο βέλτιστο του PSO, πλην της ABC που υστέρησε ελαφρά.

5.4 Σύγκριση αποκλίσεων όλων των αλγόριθμων που χρησιμοποιήθηκαν για την επίλυση των προβλημάτων

Παρατηρώντας τα αποτελέσματα των αλγόριθμων, αλλά και τις αποκλίσεις τους από τα βέλτιστα, θεωρήσαμε χρήσιμο να δημιουργήσουμε τον παρακάτω πίνακα, όπου ταξινομούμε για κάθε έναν αλγόριθμο τις ποσοστιαίες αποκλίσεις του από τα βέλτιστα, για όλα τα προβλήματα. Με τον τρόπο αυτό, γίνεται ευδιάκριτη η απόδοση του κάθε αλγόριθμου ξεχωριστά, καθώς και το εύρος τιμών που παρουσιάζουν οι αποκλίσεις τους, πράγμα που μπορεί να αποτυπώνει και την αξιοπιστία τους, όσον αφορά στην χρήση τους για την επίλυση τέτοιου είδους προβλημάτων.

GA			DE			PSO			CSA		
par	**dem**	**Var %**	**par**	**dem**	**Var %**	**par**	**dem**	**Var %**	**par**	**dem**	**Var %**
7	2	0.65	**6**	0	0.17	**1**	0	0.00	**1**	0	0.00
6	0	0.70	**3**	0	0.20	**1**	1	0.00	**1**	2	0.00
6	1	0.95	**7**	2	0.28	**3**	2	0.00	**4**	0	0.00
7	1	1.29	**6**	2	0.31	**6**	0	0.00	**4**	1	0.00
3	2	1.39	**7**	0	0.48	**6**	1	0.00	**4**	2	0.00
3	0	1.54	**1**	2	0.97	**6**	2	0.00	**5**	1	0.00
1	2	1.70	**7**	1	0.98	**7**	0	0.00	**7**	0	0.00
7	0	1.91	**3**	2	1.08	**7**	2	0.00	**7**	2	0.08
6	2	1.99	**6**	1	1.25	**3**	0	0.19	**5**	2	0.10
3	1	2.11	**1**	1	1.85	**1**	2	0.30	**3**	2	0.11
1	1	2.88	**4**	0	1.93	**7**	1	0.61	**2**	2	0.12
1	0	3.43	**2**	2	2.02	**2**	2	0.67	**1**	1	0.35
2	0	3.80	**1**	0	2.44	**3**	1	0.77	**3**	1	0.42
2	2	4.49	**2**	1	2.66	**2**	0	0.79	**3**	0	0.50
2	1	4.53	**2**	0	3.15	**2**	1	1.04	**2**	1	0.51
4	1	5.62	**3**	1	3.50	**4**	0	1.14	**5**	0	0.69
4	0	5.85	**4**	1	4.42	**4**	1	3.74	**2**	0	0.87
4	2	7.11	**4**	2	6.91	**4**	2	4.84	**7**	1	1.04
5	2	7.99	**5**	2	7.39	**5**	2	7.24	**6**	2	5.61
5	0	9.52	**5**	0	8.37	**5**	0	7.31	**6**	1	9.06
5	1	9.86	**5**	1	9.41	**5**	1	8.26	**6**	0	9.76

HBMO			BBMO			GSO			ABC		
par	**dem**	**Var %**	**par**	**dem**	**Var %**	**par**	**dem**	**Var %**	**par**	**dem**	**Var %**
1	0	0.00	**1**	0	0.00	**1**	0	0.00	**1**	1	0.41
4	1	0.00	**1**	2	0.00	**2**	1	0.00	**1**	0	1.12
5	1	0.00	**3**	0	0.00	**5**	2	0.00	**2**	2	1.64
7	2	0.00	**7**	1	0.00	**1**	1	0.01	**7**	0	2.18
3	1	0.00	**2**	2	0.49	**3**	1	0.07	**1**	2	2.23
6	2	0.00	**1**	1	0.53	**7**	2	0.11	**3**	1	2.74
1	2	0.01	**5**	0	0.86	**4**	0	0.14	**7**	2	3.28
4	0	0.08	**7**	0	0.97	**1**	2	0.26	**3**	2	3.35
6	0	0.27	**3**	2	0.98	**3**	2	0.66	**2**	1	3.52
4	2	0.28	**2**	1	1.26	**3**	0	1.05	**7**	1	3.55
5	2	0.47	**5**	2	1.26	**7**	0	1.13	**3**	0	4.11
3	2	0.53	**3**	1	1.47	**5**	1	1.23	**2**	0	4.12
3	0	0.58	**7**	2	1.69	**4**	1	1.27	**4**	0	4.40
6	1	0.59	**4**	2	1.80	**2**	2	1.40	**6**	0	4.51
2	1	0.92	**2**	0	1.84	**6**	2	1.75	**4**	2	4.58
7	1	1.42	**4**	0	1.99	**5**	0	1.76	**4**	1	4.71
7	0	1.81	**4**	1	2.39	**7**	1	1.84	**6**	1	4.98
2	2	1.83	**5**	1	2.66	**4**	2	1.93	**6**	2	5.35
1	1	2.26	**6**	1	7.46	**2**	0	2.04	**5**	0	7.03
2	0	8.87	**6**	0	8.87	**6**	1	2.06	**5**	2	8.24
5	0	12.14	**6**	2	10.27	**6**	0	3.27	**5**	1	12.13

Πίνακας 9. Ταξινόμιση των ποσοστιαίων αποκλίσεων από τα βέλτιστα για κάθε αλγόριθμο

Κάποια χρήσιμα συμπεράσματα που προκύπτουν από τον πίνακα 9, παρατίθενται παρακάτω.

Τη μικρότερη μέγιστη απόκλιση μεταξύ των μεθόδων που συγκρίνουμε, είχε ο αλγόριθμος GSO με τιμή 3,27% (par6 dem=0), έχοντας σημαντική διαφορά από τη δεύτερη που παρουσίασε ο αλγόριθμος PSO, που ήταν στο 8,26% (par5 dem=1). Αντιθέτως, τη μεγαλύτερη μέγιστη απόκλιση την παρουσίασαν οι μέθοδοι HBMO και ABC με τιμές 12,14% (par5 dem=0) και 12,13% (par5 dem=1) αντίστοιχα. Οι υπόλοιπες μέγιστες αποκλίσεις κυμαίνονταν από 8% έως 10%.

Η μέθοδος GSO παρουσίασε τη μικρότερη μέση απόκλιση από το γνωστό βέλτιστο, ίση με 1,05%. Ακολουθεί η CSA με μέση απόκλιση 1,39%. Αντίστοιχα, τη μεγαλύτερη μέση απόκλιση την είχε ο αλγόριθμος ABC με τιμή 4,2% και ακολουθεί η GA με 3,78%.

Τις περισσότερες αποκλίσεις με τιμή κάτω του 1% τις εμφάνισε ο CSA (17 από τα 21 αποτελέσματα). Ακολουθούν οι HBMO και PSO με 15 και 14 αντίστοιχα. Τις περισσότερες αποκλίσεις με τιμή άνω του 1% τις είχε, με 20 στα 21 αποτελέσματα, ο αλγόριθμος ABC. Ακολούθησαν οι GA και DE με 18 και 14 αποτελέσματα αντίστοιχα.

Συνολικά:

Ο αλγόριθμος PSO μπορούμε να πούμε οτι παίρνει την πρωτιά, αφού σε 8 περιπτώσεις έχει τα καλύτερα αποτελέσματα, συνεπώς την βέλτιστη λύση. Ακολουθεί ο CSA με 7, ο HBMO με 6 ο BBMO με 4 και ο GSO, με τους υπόλοιπους αλγόριθμους να μην πετυχαίνουν να 'κυριαρχήσουν' σε καμία περίπτωση. Ο ABC είχε τη χειρότερη κατάταξη σε όλα τα κριτήρια που περιγράψαμε παραπάνω και έτσι, δεν θα δύναται να προταθεί ως αξιόπιστος τρόπος επίλυσης προβλημάτων VRPSD. Από τους υπόλοιπους αλγόριθμους, ο GSO δείχνει πιο ευσταθής, με τη μικρότερη μέγιστη απόκλιση και τη μικρότερη μέση απόκλιση. Ωστόσο, δεν κατορθώνει συνήθως τα καλύτερα αποτελέσματα. Ο CSA θα μπορούσε να προταθεί ως καλύτερη μέθοδος για την επίλυση τέτοιου είδους προβλημάτων, αν δεν υπήρχαν τα κακά αποτελέσματα για το par6, ανεξαρτήτως την τιμή που παίρνει το dem. Επίσης, καλά αποτελέσματα, με αντίστοιχα προβλήματα με τον CSA, είχαν οι αλγόριθμοι PSO και HBMO. Οι τρεις προαναφερόμενοι αλγόριθμοι, είχαν τις λιγότερες αποκλείσεις άνω το υ 1 % και τις περισσότερες καλύτερες λύσεις. Προτείνεται η χρήση ενός συνδυασμού των τριών μαζί με τον CSA για αποφυγή σημαντικών αποκλίσεων.

Κεφάλαιο 6.

Συμπεράσματα

Στην εργασία αυτή ασχοληθήκαμε με την επίλυση επτά προβλημάτων δρομολόγησης οχημάτων με στοχαστική ζήτηση με τον αλγόριθμο τεχνητής αποικίας μελισσών (Artificial Bee Colony optimization algorithm, ABC). Τα ίδια προβλήματα έχουν επιλυθεί και με άλλους μεθευρετικούς αλγόριθμους, όπως τον αλγόριθμο Βελτιστοποίησης Ζευγαρώματος Μελισσών HBMO, τον αλγόριθμο Βελτιστοποίησης Ζευγαρώματος Μπάμπουρων BBMO, τον αλγόριθμο Βελτιστοποίησης Σμήνους Σωματιδίων PSO, τον αλγόριθμο Επιλογής Κλώνων CSA, τον αλγόριθμο Βελτιστοποίησης Σμήνους Πυγολαμπίδων GSO, τον αλγόριθμο Διαφορικής Εξέλιξης DE και τον Γενετικό Αλγόριθμο GA. Στόχος μας ήταν να διαπιστώσουμε τις επιδόσεις του αλγόριθμου ABC στην επίλυση τέτοιου είδους προβλημάτων, αφού μπορούσαμε να συγκρίνουμε τα αποτελέσματά του, με των υπολοίπων αλγόριθμων, όπως αυτά έχουν καταγραφεί σε άλλες εργασίες. Ο αλγόριθμος ABC δε μπόρεσε να βελτιώσει κάποια λύση από αυτές που ήδη έχουν σημειωθεί, ενώ γενικά δεν επέδειξε σταθερές προσεγγίσεις στα εως τώρα βέλτιστα.

Βιβλιογραφία

Ελληνική

-Γεωργία-Ρουμπίνη Ιορδανίδου (2011), Επίλυση προβλήματος δρομολόγησης οχημάτων με στοχαστική ζήτηση με χρήση του Αλγορίθμου Βελτιστοποίησης Σμήνους Σωματιδίων, Προπτυχιακή εργασία στο Πολυτεχνείο Κρήτης, Τμήμα Μηχανικών Παραγωγής και Διοίκησης.

-Γεωργία-Ρουμπίνη Ιορδανίδου (2012), Επίλυση προβλήματος δρομολόγησης οχημάτων με στοχαστική ζήτηση με χρήση Αλγορίθμων Εμπνευσμένων από τη Φύση, Μεταπτυχιακή εργασία στο Πολυτεχνείο Κρήτης, Τμήμα Μηχανικών Παραγωγής και Διοίκησης.

-Ιωάννης Μαρινάκης, Αθανάσιος Μυγδαλάς (2008), Σχεδιασμός και Βελτιστοποίηση της Εφοδιαστικής Αλυσίδας, Εκδόσεις Σοφία.

-Ιωάννης Μαρινάκης, Μαγδαληνή Μαρινάκη, Νικόλαος Φ. Ματσατσίνης, Κωνσταντίνος Ζοπουνίδης (2011), Μεθευρετικοί και Εξελικτικοί Αλγόριθμοι σε Προβλήματα Διοικητικής Επιστήμης, Εκδόσεις Κλειδάριθμος.

-Σπανού Παρασκευή (2010), Ανάπτυξη εξελικτικού αλγορίθμου για το πρόβλημα δρομολόγησης οχημάτων με στοχαστική ζήτηση, Μεταπτυχιακή εργασία στο Πολυτεχνείο Κρήτης, Τμήμα Μηχανικών Παραγωγής και Διοίκησης.

Ξενόγλωσση

-Abbass H. A. (2001c). A Single Queen Single Worker Honey Bees Approach to 3SAT, The Genetic and Evolutionary Computation Conference, GECCO2001, San Francisco, USA.

-Abbass H. A. (2001d). An Agent Based Approach to 3SAT Using Marriage in Honey Bees Optimization, International Journal of Knowledge Based Intelligent Engineering Systems (KES), 6(2), 18.

-Afshar A., Bozog Haddad O., Marino M. A., Adams B. J. (2007), Honey Bee Mating Optimization (HBMO) Algorithm for Optimal Reservoir Operation, Journal of the Franklin Institute, 344, 452-462.

-B. Basturk, D. Karaboga (2006), An artificial bee colony (ABC) algorithm for numeric function optimization, in: IEEE Swarm Intelligence Symposium 2006, May

12–14, Indianapolis, IN, USA.

-Baykasoglu A., Ozbakor L., Tapkan P. (2007). Artificial Bee Colony Algorithm and its Application to Generalized Assignment Problem, Swarm Intelligence, Focus on Ant and Particle Swarm Optimization, Chan F.T.S., Tiwari M.K. (Editors), ITech Education and Publishing, 113-144.

-Chakraborty U.K., (2008), Advances in Differential Evolution, Studies in Computational Intelligence, 143, SpringerVerlag, Berlin, Heidelberg.

-Dasgupta D. (Editor) (1998), Artificial Immune Systems and their Application, Springer, Heidelberg.

-De Castro L.N., Von Zuben F. J. (2002), Learning and Optimization Using the Clonal Selection Principle, IEEE Transactions on Evolutionary Computation, Special Issue on Artificial Immune Systems, 6(3), 239-251.

-De Castro L.N., Timmis J. (2002), Artificial Immune Systems: A New Computational Intelligence Approach, Springer, Heidelberg.

-De Castro L.N., Von Zuben F. J. (2000), The Clonal Selection Algorithm with Engineering Applications, Proceedings of the Genetic and Evolutionary Computational Conference, 36-37.

-E. Bonabeau, M. Dorigo, G. Theraulaz (1999), Swarm Intelligence: From Natural to Artificial Intelligence, NY: Oxford University Press, NewYork.

-Eberhart R.C., and Kennedy J. (1995), A new optimizer using particle swarm theory, Proceedings of the Sixth International Symposium on Micro Machine and Human Science, Nagoya, Japan, 39–43. Piscataway, NJ: IEEE Service Center.

-Eberhart R.C., and Shi Y. (2000). Comparing inertia weights and constriction factors in particle swarm optimization. Proceedings of the 2000 Congress on Evolutionary Computation, 84–88. Piscataway, NJ: IEEE Service Center.

-Elbeltagia E., Hegazyb T., Grierson D. (2005), Comparison Among Five Evolutionary Based Optimization Algorithms, Advanced Engineering Informatics, 19, 43-53.

-Engelbrecht A. P. (2007), Computational Intelligence: An Introduction, Second Edition, John Wiley and Sons, England.

-Eusuff M.M., Lansey K.E. (2003), Optimization of Water Distribution Network Design Using the Shuffled Frog Leaping Algorithm, Journal of Water Resource Planning Management, 129(3), 210-225.

-Eusuff M.M., Lansey K.E., Pasha F. (2006). Shuffled Frog Leaping Algorithm: A Memetic Meta Heuristic for Discrete Optimization, Engineering Optimization, 38(2), 129-154.

-Fathian M., Amiri B., Maroosi A. (2007), Application of Honey Bee Mating Optimization Algorithm on Clustering, Applied Mathematics and Computation, 190, 1502-1513.

-Feoktistov V. (2006), Differential Evolution In Search of Solutions, Springer.

-Goulson D. (2009). Bumblebees: Behavior, Ecology, and Conservation, Oxford University Press, USA

-H.A. Abbass (2001a), A monogenous MBO approach to satisfiability, Proceeding of the International Conference on Computational, Intelligence for Modeling, Control and Automation, CIMCA'2001. Las Vegas, NV, USA.

-H.A. Abbass (2001b), Marriage in honey-bee optimization (MBO): a haplometrosis polygynous swarming approach, The Congress on Evolutionary Computation (CEC2001), Seoul, Korea, May 2001, pp. 207-214.

-Horng M.H. (2010). A Multilevel Image Thresholding Using the Honey Bee Mating Optimization, Applied Mathematics and Computation, 215, 3302-3310.

-J. Vesterstrøm, J. Riget (2002), Particle swarms extensions for improved local, multimodal and dynamic search in numerical optimization, M.Sc. Thesis, May.

-Karaboga D., Akay B. (2009), A Comparative Study of Artificial Bee Colony algorithm, Applied Mathematics and Computation, 214, 108-132.

-Karaboga D., Akay B. (2009), A Survey: Algorithms Simulating Bee Swarm Intelligence, Artificial Intelligence Review, DOI: 10.1007/s1046200991274

-Karaboga D., Basturk B. (2007), A Powerful and Efficient Algorithm for Numerical Function Optimization: Artificial Bee Colony (ABC) Algorithm, Journal of Global Optimization, 39, 459-471.

-Karaboga D., Basturk B. (2008), On the Performance of Artificial Bee Colony (ABC) Algorithm, Applied Soft Computing, 8, 687-697.

-Kennedy J. (1998), The behavior of particles, In V.W. Porto, N. Saravanan, D. Waagen, and A. E. Eiben (Eds.), Evolutionary Programming VII: Proceedings of the 7th Annual Conference on Evolutionary Programming. Berlin: Springer-Verlag.

-Kennedy J., Eberhart R.C. with Shi Y. (2001), Swarm Intelligence, The Morgan Kaufmann Series in Evolutionary Computation, ed. David B. Fogel.

-Kenneth V. Price, Rainer M. Storn, Jouni A. Lampinen (2005), Differential Evolution – A Practical Approach to Global Optimization.

-Krishnanand K.N., Ghose D. (2009), Glowworm Swarm Optimization: A New Method for Optimizing Multimodal Functions, International Journal of Computational Intelligence Studies, 1(1), 93-119.

-Krishnanand K.N., Ghose D. (2009). Glowworm Swarm Optimization for Searching Higher Dimensional Spaces. Innovations in Swarm Intelligence, SCI 248, Chee Peng Lim, Lakhmi C. Jain, and Satchidananda Dehuri (Editors), 61-76.

-L.N. De Castro, F.J. Von Zuben (1999), Artificial Immune Systems, Part I. Basic Theory And Applications, Technical Report No. Rt Dca 01/99, Feec/ Unicamp, Brazil.

-Lucic P., Teodorovic D. (2001), Bee System: modeling combinatorial optimization transportation engineering problems by swarm intelligence, In: Preprints of the TRISTAN IV Triennial Symposium on Transportation Analysis, Sao Miguel, Azores Islands, Portugal, 441-445.

-Marinaki M., Marinakis Y., Zopounidis C. (2010). Honey Bees Mating Optimization Algorithm for Financial Classification Problems, Applied Soft Computing, 10, 806-812.

-Marinakis Y. Marinaki M., Dounias G. (2008). Honey Bees Mating Optimization Algorithm for the Vehicle Routing Problem, Nature Inspired Cooperative Strategies for Optimization NICS 2007, Studies in Computational Intelligence, Krasnogor, N., Nicosia, G., Pavone, M., Pelta, D. (Editors), SpringerVerlag Berlin, 129, 139-148.

-Marinakis Y., Marinaki M. (2009), A Hybrid Honey Bees Mating Optimization Algorithm for the Probabilistic Traveling Salesman Problem, IEEE Congress on Evolutionary Computation (CEC 2009), Trondheim, Norway.

-Marinakis Y., Marinaki M., (2010). Bumble Bees Mating Optimization Algorithm for the Vehicle Routing Problem, Handbook of Swarm Intelligence Concepts, Principles and Applications, Series on Adaptation, Learning, and Optimization, Bijaya Ketan Panigrahi, Yuhui Shi and Meng Hiot Lim, (Editors), Springer Verlang, Berlin (accepted).

-Marinakis Y., Iordanidou G.R., Marinaki M. (2013). Particle Swarm Optimization for the Vehicle Routing Problem with Stochastic Demands.

-Marinakis Y., Marinaki M., Dounias, G. (2010a). A Hybrid Particle Swarm Optimization Algorithm for the Vehicle Routing Problem, Engineering Applications of Artificial Intelligence, 23, 463-472.

-Marinakis Y., Marinaki M., Dounias G. (2010b). Honey Bees Mating Optimization Algorithm for Large Scale Vehicle Routing Problems, Natural Computing, 9, 527.

-Marinakis Y., Marinaki M., Matsatsinis N. (2009). A Hybrid Discrete Artificial Bee Colony GRASP Algorithm for Clustering, 39th International Conference on Computers and Industrial Engineering, 6 8 July 2009, Troyes, France.

-Marinakis Y., Marinaki M., Matsatsinis N. (2010). A Bumble Bees Mating Optimization Algorithm for Global Unconstrained Optimization Problems, Nature Inspired Cooperative Strategies for Optimization NICSO 2010, Studies in Computational Intelligence Vol 284, Gonzalez, J. R. et al. (Editors), SpringerVerlag Berlin, 305-318.

-Onwubolu G.C., Davendra D. (2009). Differential Evolution: A Handbook for Global Permutation Based Combinatorial Optimization, Studies in Computational Intelligence, 175, SpringerVerlag Berlin, Heidelberg.

-Ozbakir L., Baykasoglu A., Tapkan P., (2010). Bees algorithm for Generalized Assignment Problem, Applied Mathematics and Computation, 215, 3782-3795.

-Paolo Toth Daniele Vigo (2002), The Vehicle Routing Problem, SIAM Monographs on Discrete Mathematics and Applications.

-Price K.V., Storn R.M., and Lampinen J.A. (2005), Differential Evolution: A Practical Approach to Global Optimization, Springer, Berlin.

-Rochat Y., Taillard E.D. (1995). Probabilistic Diversification and Intensification in Local Search for Vehicle Routing, Journal of Heuristics, 1, 147-167.

-Sabat S.L., Udgata S., Abraham A. (2010). Artificial Bee Colony Algorithm for Small Signal Model Parameter Extraction of MEFSET. Engineering Applications of Artificial Intelligence, doi:10.1016/j.engappai.2010.01.020.

-Shi Y., Eberhart R. (1998), A Modified Particle Swarm Optimizer. Proceedings of 1998 IEEE World Congress on Computational Intelligence, 69-73.

-Storn R. (1996). On the Usage of Differential Evolution for Function Optimization, Proceedings of the Biennial Conference of the North American Fuzzy Information Processing Society, 519-523.

-Storn R. (1999), System Design by Constraint Adaptation and Differential Evolution. IEEE Transactions on Evolutionary Computation 3, 1, 22-34.

-Storn R., Price K. (1997), Differential Evolution: A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces. Journal of Global Optimization 11, 4, 341-359

-Talbi E.G. (2009), Metaheuristics: From Design to Implementation, John Wiley and Sons, USA.

-Teo J., Abbass H. A. (2003), A True Annealing Approach to the Marriage in Honey Bees Optimization Algorithm., International Journal of Computational Intelligence and Applications, 3(2), 199-211.

-Teodorovic D., Lucic P., Markovic G., Dell' Orco M. (2006), Bee colony optimization: principles and applications, In: Reljin B., Stankovic S. (Eds.): Proceedings of the Eight Seminar on Neural Network Applications in Electrical Engineering - NEUREL 2006, University of Belgrade, Belgrade 151-156.