

ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ ΠΑΡΑΓΩΓΗΣ ΚΑΙ ΔΙΟΙΚΗΣΗΣ

ΣΧΕΔΙΑΣΜΟΣ και ΒΕΛΤΙΣΤΟΠΟΙΗΣΗ της ΕΦΟΔΙΑΣΤΙΚΗΣ ΑΛΥΣΙΔΑΣ

Απαλλακτική Εργασία

Θέμα: A2B2Δ4Ε3

Αλγόριθμος Βελτιστοποίησης Σμήνους Σωματιδίων
(Particle Swarm Optimization)

Υπό

ΠΑΣΧΑΛΙΔΗΣ ΜΑΡΙΟΣ

A.M.: 2015010115

Χανιά, 2019

ΠΕΡΙΕΧΟΜΕΝΑ

Περίληψη άσκησης.....	3
Λειτουργία Προγράμματος.....	4
Βασικό Πρόγραμμα (Particle_Swarm_Optimization)	5
Συνάρτηση (PARTICLES).....	6-7
Συνάρτηση (Three_Opt_1_1_exchange).....	8-10
Συνάρτηση (Optimization).....	10-12
Συνάρτηση (transform).....	12-13
Συνάρτηση (route_number_reduction).....	14-16
Συνάρτηση (PLOTS).....	16
Επιλογή Τιμών στις Παραμέτρους.....	16-18
ΑΠΟΤΕΛΕΣΜΑΤΑ.....	19-24

Περίληψη

Στην εργασία αυτή δημιουργείται πρόγραμμα που θα εφαρμόσει τον 1. Αλγόριθμο της Διαδικασίας Εισαγωγής Κόμβων για την δημιουργία αρχικών λύσεων, τον αλγόριθμο τοπικής αναζήτησης 2. 3-Opt, 1-1exchange, και τον 3. Αλγόριθμο Βελτιστοποίησης Σμήνους σωματιδίων ως μεθευρετικό αλγόριθμο εμπνευσμένο από τη φύση. Οι παραπάνω αλγόριθμοι θα εφαρμοστούν στο πλαίσιο του Ανοιχτού Προβλήματος Δρομολόγησης Οχημάτων και στα δεδομένα του αρχείου E3 που παρέχεται μαζί με το πρόγραμμα και την αναφορά.

Λειτουργία Προγράμματος

Στο πρόγραμμα που παρατίθεται, εφαρμόζεται αρχικά ο Αλγόριθμος της Διαδικασίας Εισαγωγής Κόμβων με κατάλληλη τροποποίηση εισάγοντάς του την τυχειότητα που είναι απαραίτητη για την εφαρμογή του βασικού αλγορίθμου βελτιστοποίησης σωματιδίων σμήνους. Ο παραπάνω αλγόριθμος εφαρμόζεται από τη συνάρτηση με όνομα **particles.m** η οποία καλείται στο βασικό αρχείο με όνομα **Particle_Swarm_Optimization.m**. Στη συνέχεια, στις αρχικές λύσεις που δημιουργήθηκαν, για βελτίωσή τους εφαρμόζεται τοπική αναζήτηση με τον αλγόριθμο 3-opt, 1-1 exchange την οποία εφαρμόζει η συνάρτηση με όνομα **Three_Opt_1_1_exchange.m** και οποία επίσης καλείται στο βασικό πρόγραμμα. Τέλος, στις λύσεις εφαρμόζεται ο αλγόριθμος βελτιστοποίησης σωματιδίων σμήνους από τη συνάρτηση με όνομα **Optimization.m**. Για τη σωστή λειτουργία του προγράμματος χρησιμοποιούνται επίσης οι παρακάτω συναρτήσεις οι οποίες δημιουργήθηκαν για το συγκεκριμένο πρόβλημα και είναι οι εξής:

- **F.m** : Εφαρμόζει τη λειτουργία της (Fitness Function) επιστρέφει δηλαδή για κάποια διαδρομή ή σύνολο διαδομών την απόδοσή τους σε απόσταση ή χρόνο.
- **transform.m** : Εφαρμόζει τον απαραίτητο μετασχηματισμό των λύσεων για την εφαρμογή του αλγορίθμου βελτιστοποίησης σωματιδίων σμήνους ή τον αντίστροφο μετασχηματισμό τους ώστε να μπορούν να αξιολογηθούν
- **route_number_reduction.m** : Μπορεί και μειώνει το σύνολο των διαδρομών σε μία λύση.
- **plots.m** : Τυπώνει τις διαδρομές της λύσης σε κατάλληλο γράφημα.

Η λειτουργία του βασικού προγράμματος αλλά και των συναρτήσεων θα μελετηθεί πιο αναλυτικά πιο κάτω.

Βασικό Πρόγραμμα (Particle_Swarm_Optimization)

Μέχρι και την γραμμή 44 του κώδικα διαβάζονται τα δεδομένα απο το αρχείο E3.txt και αποθηκεύονται σε μεταβλητές ώστε να χρησιμοποιηθούν στη συνέχεια. Υπολογίζονται οι αποστάσεις των κόμβων μεταξύ τους στον πίνακα *distances* από τις συντεταγμένες (coordinates) των κόμβων και αποθηκεύεται ο αριθμός των λύσεων που ζητήθηκε να δημιουργηθούν στην μεταβλητη *Solutions*.

Στις γραμμές 47 έως και 58 δημιουργούνται οι αρχικές λύσεις (10 στο σύνολο) μέσω της συνάρτησης **particles** και αποθηκεύονται στη λίστα *Swarm*. Στην συνάρτηση, εκτός από τα απαραίτητα δεδομένα που δίνονται σαν ορίσματα επιλέγεται επίσης και το ποσοστό των κόμβων που θα επιλεγούν τυχαία ώστε να υπάρχει η απαραίτητη διαφοροποίηση μεταξύ των λύσεων και να εφαρμοστεί σωστά ο αλγοριθμος βελτιστοποίησης σμήνους σωματιδίων. Στη συνέχεια τυπώνονται οι λύσεις και αποθηκεύονται οι αποδόσεις τους στον πίνακα *fitness*.

Στις γραμμές 59 έως και 67 εφαρμόζεται σε κάθε λύση η τοπική αναζήτηση μέσω της συνάρτησης **Three_Opt_1_1_exchange** και αποθηκεύονται αντίστοιχα οι αποδόσεις των βελτιωμένων λύσεων στον *fitness* και τυπώνονται οι διαδρομές της κάθε λύσης.

Στις υπόλοιπες γραμμές 68 έως 82 εφαρμόζεται από τη συνάρτηση **Optimization** ο αλγόριθμος βελτιστοποίησης σωματιδίων σμήνους στην οποία δίνονται αρχικά τα απαραίτητα ορίσματα στα *c1* και *c2* όπου από τη βιβλιογραφία για βελτιστοποίηση της λειτουργίας του αλγορίθμου επιλέγεται η τιμή 2. Στη συνέχεια τυπώνονται οι λύσεις και αποθηκεύονται οι αποδόσεις τους στον πίνακα *fitness*.

Τέλος, στις γραμμές 83 έως το τέλος εφαρμόζεται τροποποίηση από το χρήστη στις τελικές λύσεις. Δηλαδή εφαρμόζεται πάνω σε αυτές όσες φορές επιλέξει τοπική αναζήτηση και λόγω της μορφής του προβλήματος, επιλέγεται ο επιθυμητός αριθμός διαδρομών ο οποίος δεν θα χειροτερέψει σε μεγάλο βαθμό την απόδοση της λύσης. Η επιλογή των τιμών θα αναλυθεί πιο κάτω.

Συνάρτηση (PARTICLES)

Η συνάρτηση **particles.m** δέχεται σαν ορίσματα τα εξής:

1. percentage: Είναι το ποσοστό των κόμβων που θα επιλεγούν τυχαία.
2. N: Ο αριθμός των συνολικών κόμβων (συμπεριλαμβανομένης και της αποθήκης)
3. distances: Είναι ο πίνακας των αποστάσεων των κόμβων μεταξύ τους.
4. demand : Είναι διάνυσμα που περιέχει τη ζήτηση κάθε κόμβου.
5. maxStorage : Είναι η μέγιστη χωρητικότητα.
6. time_distances: Είναι οι χρονικές αποστάσεις των κόμβων μεταξύ τους.
7. maxTime: Είναι ο μέγιστος χρόνος παραμονής του οχήματος στη διαδρομή.
8. serviceTime: Είναι ο χρόνος εξυπηρέτησης.

Στην συνάρτηση, αρχικά, στις γραμμές 4-13 αρχικοποιείται το διάνυσμα λύσης με την πρώτη διαδρομή δίνοντας πρώτο και δεύτερο στοιχείο αντίστοιχα την αποθήκη και τον κοντινότερό της κόμβο αντίστοιχα.

Για την δημιουργία των αρχικών λύσεων εφαρμόζεται ο *_Αλγορίθμος της Διαδικασίας Εισαγωγής Κόμβων* όπως παρουσιάζεται στις σημειώσεις. Στις γραμμές 14-73 επαναλαμβάνεται ο βρόγχος του αλγορίθμου.

Στις γραμμές 15-28 επιλέγεται ο επόμενος κόμβος, από τους διαθέσιμους, που θα χρησιμοποιηθεί. Ο κόμβος αυτός επιλέγεται είτε τυχαία αν δοθεί τιμή στην τυχαία μεταβλητή p μικρότερη ή ίση του επιλεγμένου ποσοστού είτε επιλέγεται ο κόμβος, από τους διαθέσιμους, που θα προκαλέσει τη μεγαλύτερη μείωση της απόστασης μεταξύ δύο διαδοχικών στις ήδη διαμορφωμένες διαδρομές δηλαδή τον κόμβο k που θα ελαχιστοποιήσει τη συνάρτηση $c_{ik} + c_{kj} - c_{ij}$.

Στη συνέχεια, στις γραμμές 32-77, δημιουργείται ο βρόγχος εισαγωγής του επιλεγμένου κόμβου στην κατάλληλη θέση στο διάνυσμα των διαδρομών. Αρχίζει βρίσκοντας τη θέση στο διάνυσμα λύσεις στην οποία αν τοποθετηθεί ο

επιλεγμένος κόμβος θα προκαλέσει μικρότερη αύξηση της απόστασής της με όμοια διαδικασία με την οποία ακολουθήθηκε στην επιλογή του κόμβου(όχι τυχαία). Στην παραπάνω διαδικασία δεν ελέγχθηκε η τοποθέτηση του κόμβου στο τέλος της κάθε διαδρομής οπότε στις γραμμές 49-56 ελέγχεται αν η τοποθέτηση του κόμβου στο τέλος της διαδρομής που ανήκει ο επιλεγμένη θέση είναι προτιμότερο να επιλεγεί. Αυτό γίνεται αν c_{ik} ή $c_{kj} > c_{(\text{τέλος διαδρομής})k}$.

Έπειτα γίνεται έλεγχος αν η τοποθέτηση του κόμβου στην επιλεγμένη διαδρομή είναι εφικτή ελέγχοντας αν η αύξηση στη συνολική ζήτηση της διαδρομής και συνολικού χρόνου παραμονής σε αυτήν εξακολουθεί να βρίσκεται εντός των μέγιστων ορίων (μέγιστη ζήτηση και μέγιστος χρόνος παραμονής σε διαδρομή). Εάν είναι εφικτή η τοποθέτηση του κόμβου τότε τοποθετείται στην επιλεγμένη θέση και προκαλείται έξοδος από τον βρόγχο. Ενώ αν δεν είναι εφικτή, αποθηκεύεται ως μη εφικτή η διαδρομή στο διάνυσμα *violating* και επαναλαμβάνεται η διαδικασία του βρόγχου χωρίς τη δυνατότητα επιλογής των μη εφικτών διαδρομών.

Τέλος εάν σε όλες τις διαδρομές που έχουν ήδη δημιουργηθεί δεν είναι εφικτό να τοποθετηθεί ο κόμβος τότε τοποθετείται ο κόμβος σε καινούρια διαδρομή. Ο έλεγχος αυτός και η δημιουργία της καινούριας διαδρομής γίνεται στις γραμμές 72-76 στον οποίο επίσης προκαλείται έξοδος από τον βρόγχο.

Μετά από κάθε έξοδο από τον βρόγχο εισαγωγής του κόμβου στις διαδρομές αφαιρείται ο κόμβος που επιλέχθηκε στην αρχή από τους διαθέσιμους για επιλογή ώστε να μην ξαναχρησιμοποιηθεί.

Συνάρτηση (Three_Opt_1_1_exchange)

Η συνάρτηση **Three_Opt_1_1_exchange.m** δέχεται σαν ορίσματα τα εξής:

1. **x**: Είναι η αρχική λύση στην οποία θα εφαρμοστεί ο αλγόριθμος.
2. **distances**: Είναι ο πίνακας των αποστάσεων των κόμβων μεταξύ τους.
3. **time_distances**: Είναι οι χρονικές αποστάσεις των κόμβων μεταξύ τους.
4. **maxTime**: Είναι ο μέγιστος χρόνος παραμονής του οχήματος στη διαδρομή.
5. **serviceTime**: Είναι ο χρόνος εξυπηρέτησης.
6. **demand** : Είναι διάνυσμα που περιέχει τη ζήτηση κάθε κόμβου.
7. **maxStorage** : Είναι η μέγιστη χωρητικότητα.

Στην συνάρτηση, αρχικά, στις γραμμές 7-10 βρίσκονται τα σημεία στο διάνυσμα λύσης όπου υπάρχει η τιμή 1 η οποία δηλώνει την αποθήκη και τοποθετούνται στο διάνυσμα *aces* το οποίο θα χρησιμοποιηθεί για να διαχωριστούν οι διαδρομές μεταξύ τους και να μπορούν να παρθούν από το διάνυσμα λύσεων, να εφαρμοστεί σε αυτές ο αλγόριθμος και να τοποθετηθούν οι βελτιωμένες διαδρομές πίσω στο διάνυσμα λύσεων. Στο τέλος του *aces* αποθηκεύεται το τέλος του διανύσματος λύσης το οποίο δηλώνει και το τέλος της τελευταίας διαδρομής.

Στη συνέχεια εκτελείται ο αλγόριθμος One-One Exchange πριν εφαρμοστεί ο Three Opt. Ο αλγόριθμος One-One Exchange πραγματοποιείται στην γραμμή 11 μέσω της συνάρτησης **one_one_exchange.m** η οποία και καλείται δίνοντάς της τα κατάλληλα ορίσματα (ίδια με της *Three_Opt_1_1_exchange.m*). Ο αλγόριθμος 1-1 exchange πραγματοποιείται με σκοπό την εύρεση καλύτερου αποτελέσματος του συνόλου των διαδρομών με την αποκοπή του μεγαλύτερου τόξου και την αντικατάστασή του από ένα μικρότερο, ή στην περίπτωση που δεν επιτευχθεί μείωση της συνολικής απόστασης των διαδρομών επιχειρείται η τυχαία τοποθέτησή του σε κάποιο άλλο σημείο με τον ίδιο σκοπό, με όριο τις 25 προσπάθειες.

Στις υπόλοιπες γραμμές 14-106, στο σύνολο των διαδρομών που προκύπτουν από τον *one_one_exchange.m*, δημιουργείται βρόγχος στον οποίο κάθε επανάληψη αντιστοιχεί σε κάθε διαδρομή. Αρχικά στις γραμμές 15-21 αποθηκεύεται στο

διάνυσμα *route* η διαδρομή μέσω του *aces* και μετά αρχικοποιούνται οι μεταβλητές *bestroute* και *bestroutdistance* στις οποίες θα αποθηκεύονται το καλύτερο αποτέλεσμα (διαδρομή) του αλγορίθμου και η συνολική απόστασή της αντιστοίχα. Στην αρχικοποίησή τους δίνονται τα δεδομένα της αρχικής διαδρομής από το διάνυσμα λύσης x .

Στις γραμμές 31-90 πραγματοποιείται ο αλγόριθμος 3 – opt στις διαδρομές που έχουν μέγεθος μεγαλύτερο του 4 (συμπεριλαμβανομένης και της αποθήκης) στις οποίες είναι εφικτό να εφαρμοστεί σωστά ο αλγόριθμος.

Οπότε στις γραμμές 34-40 επιλέγονται τα τόξα που θα κοπούν τυχαία και μέσω του βρόγχου (34-36) εξασφαλίζεται ότι δεν θα επιλεγεί κάποιο τόξο παραπάνω από μία φορά.

Στη συνέχεια και στις γραμμές 41-54 δημιουργούνται όλα τα πιθανά κομμάτια που δημιουργούνται μετά την κοπή των επιλεγμένων τόξων στη διαδρομή. Τα κομμάτια αυτά αποθηκεύονται στη λίστα N και στο τμήμα 1 αποθηκεύονται τα κομμάτια όπως υπήρχαν στη διαδρομή ενώ στο τμήμα 2 αποθηκεύονται τα ανάστροφα αυτών.

Έπειτα στις γραμμές 56-89 δημιουργήθηκε βρόγχος οποίος σε 20 επαναλήψεις (όπως ζητήθηκε στην εκφώνηση της εργασίας) θα ελέγχει διαφορετικούς συνδυασμούς των δημιουργημένων κομματιών και θα αποθηκεύεται ο συνδυασμός με την μικρότερη απόσταση. Στο βρόγχο των γραμμών 62-69 δημιουργείται ένας καινούριος συνδυασμός των κομματιών μέσω των διανυσμάτων $k-cmbns$ και $j-cmbns$ τα οποία δηλώνουν τη σειρά των θέσεων στη λίστα N και των των τμημάτων της αντίστοιχα.

Στη συνέχεια στις γραμμές 71-87 γίνεται ο έλεγχος εάν έχει ήδη πραγματοποιηθεί ο συνδυασμός που δημιουργήθηκε. Εάν έχει χρησιμοποιηθεί επαναλαμβάνεται ο βρόγχος χωρίς να μετρηθεί σαν επανάληψη, ενώ αν είναι καινούριος ο συνδυασμός δημιουργείται η διαδρομή τοποθετώντας τα κατάλληλα κομμάτια με την σειρά που δείχνει ο συνδυασμός. Έπειτα ελέγχεται εάν η διαδρομή αυτή είναι εφικτή (δαλαδή εάν η συνολική χρονική της απόσταση είναι μικρότερη της μέγιστης ενώ δεν χρειάζεται ο έλεγχος της συνολικής ζήτησης διότι οι κόμβοι της διαδρομής δεν αλλάζουν), και εάν η συνολική της απόσταση είναι μικρότερη είναι μικρότερη από αυτήν της μέχρι τότε καλύτερης τότε αποθηκεύεται αυτή ως

καλύτερη. Τέλος αποθηκεύεται ως ελεγμένος ο συνδυασμός στα διανύσματα k_past και j_past και ανεβαίνει κατά 1 ο μετρητής των επαναλήψεων.

Στις γραμμές 91-98 γίνεται ο έλεγχος εάν η διαδρομή έχει τρεις κόμβους όπου δεν μπορεί να πραγματοποιηθεί σωστά ο 3-opt οπότε για να βελτιστοποιηθεί η απόδοση της διαδρομής τοποθετούνται οι κόμβοι σύμφωνα με τον πλησιέστερο γείτονα καί αποθηκεύεται αυτή ως καλύτερη.

Τέλος στις γραμμές 100 μέχρι το τέλος τοποθετείται η καλύτερη διαδρομή που δημιουργήσαμε στη θέση της αρχικής στο διάνυσμα λύσης.

Η συνάρτηση **Three_Opt_1_1_exchange** καλείται στο βασικό πρόγραμμα 10 φορές, για κάθε αρχική λύση που δημιουργήθηκε από μία φορά.

Συνάρτηση (Optimization)

Η συνάρτηση **Optimization.m** δέχεται σαν ορίσματα τα εξής:

1. **Swarm**: Είναι το πλήθος των λύσεων μετά την εφαρμογή σε αυτές του 3opt.
2. **distances**: Είναι ο πίνακας των αποστάσεων των κόμβων μεταξύ τους.
3. **c1** και **c2**: Είναι οι σταθερές που απαιτούνται από τον αλγόριθμο βελτιστοποίησης σμήνους σωματιδίων στις οποίες έχει δοθεί η τιμή 2.
4. **demand** : Είναι διάνυσμα που περιέχει τη ζήτηση κάθε κόμβου.
5. **maxStorage** : Είναι η μέγιστη χωρητικότητα.
6. **time_distances**: Είναι οι χρονικές αποστάσεις των κόμβων μεταξύ τους.
7. **maxTime**: Είναι ο μέγιστος χρόνος παραμονής του οχήματος στη διαδρομή.
8. **serviceTime**: Είναι ο χρόνος εξυπηρέτησης.

Στην αρχή, στις γραμμές 2-20, γίνεται η αρχικοποίηση των απαραίτητων μεταβλητών οι οποίες είναι απαραίτητες για την πραγματοποίηση του αλγορίθμου. Υπολογίζονται οι τιμές των c και X με $c = c1+c2$ και $X = \frac{2}{|2-c-\sqrt{c^2-4c}|}$.

Στην μεταβλητή *pbest* θα αποθηκεύονται τα καλύτερα διανύσματα λύσης για κάθε λύση στις επαναλήψεις ενώ στην *gbest* θα αποθηκεύεται το καλύτερο διάνυσμα λύσης από όλα τα διανύσματα λύσης στις επαναλήψεις. Οπότε στην αρχικοποίησή τους τους δίνεται η τιμή των αρχικών που δέχεται σαν όρισμα η συνάρτηση και επιλέγεται το καλύτερο από αυτά (μικρότερη συνολική απόσταση) και αποθηκεύεται στην *gbest*. Τέλος αρχικοποιείται το διάνυσμα της ταχύτητας με την μεταβλητή *velocity* δίνοντάς του την τιμή μηδέν για κάθε στοιχείο του.

Στη συνέχεια και από την γραμμή 21 έως το τέλος εφαρμόζεται ο αλγόριθμος βελτιστοποίησης σωματιδίων σμήνους στις λύσεις από έναν βρόγχο που ολοκληρώνεται στις 20 επαναλήψεις (όπως ζητείται στην εκφώνηση).

Κατά την εφαρμογή του αλγορίθμου καλείται η συνάρτηση μετασχηματισμού των σωματιδίων (λύσεις) με όνομα **transform.m** η οποία εκτελεί ορθό μετασχηματισμό (*inv=no*) και ανάστροφο (*inv=yes*). Ο μετασχηματισμός αυτός είναι απαραίτητος για την εκτέλεση του αλγορίθμου και η λειτουργία του θα αναλυθεί παρακάτω.

Στην αρχή του βρόγχου, στις γραμμές 24 και 25 δίνονται τυχαίες τιμές στο διάστημα $[0,1]$ στις μεταβλητές *r1* και *r2*. Στη συνέχεια μετασχηματίζεται η καλύτερη λύση (*gbest*) σε έναν βρόγχο στο πλήθος των λύσεων γίνεται σε σειρά ο μετασχηματισμός της λύσης, της καλύτερης λύσης της στο χρόνο (*pbest*), βρίσκεται η ταχύτητα της λύσης (σωματίδιο) μέσω της εξίσωσης

$$u(t + 1, i) = X\{u(t, i) + c1r1(\text{αλύτ ρη λύση σωματιδίου} - \text{Λύση}) + c2r2(\text{Καλύτ ρη λύση σμήνους} - \text{Λύση})\}$$

και υπολογίζεται η νέα τιμή της λύσης από:

$$\text{Λύση}(t + 1) = u(t + 1, i) + \text{Λύση}(t)$$

Αφού επαναληφθεί η διαδικασία αυτή για κάθε λύση (σωμματίδιο) γίνεται ο ανάστροφος μετασχηματισμός των καινούριων σωμματιδίων και εάν η συνολική τους απόσταση είναι μικρότερη από αυτήν της μέχρι τότε αποθηκευμένης ως καλύτερης (*pbest*), τότε αποθηκεύεται αυτή στη θέση της. Τέλος γίνεται ο έλεγχος εάν κάποια από τις καινούριες λύσεις έχει μικρότερη συνολική απόσταση από την έως τότε αποθηκευμένη ως καλύτερη του σμήνους (*gbest*) και αν γίνει αυτό τότε παίρνει αυτή την θέση της.

Στο τέλος των παραπάνω βημάτων τελειώνει ο αλγόριθμος και ανεβαίνει κατά ένα ο μετρητής των επαναλήψεων έως τις 20 επαναλήψεις.

Συνάρτηση (transform)

Η συνάρτηση **transform.m** δέχεται σαν ορίσματα τα εξής:

1. *x*: Είναι το διάνυσμα λύσης (σε μορφή πριν τον μετασχηματισμό ή μετά τον μετασχηματισμό της).
2. *inv*: Δηλώνει εάν πρέπει να γίνει ορθός ή ανάστροφος μετασχηματισμός (παίρνει την τιμή “yes” για ανάστροφο και για “no” γίνεται ορθός).
3. *demand* : Είναι διάνυσμα που περιέχει τη ζήτηση κάθε κόμβου.
4. *maxStorage* : Είναι η μέγιστη χωρητικότητα.
5. *time_distances*: Είναι οι χρονικές αποστάσεις των κόμβων μεταξύ τους.
6. *maxTime*: Είναι ο μέγιστος χρόνος παραμονής του οχήματος στη διαδρομή.
7. *serviceTime*: Είναι ο χρόνος εξυπηρέτησης.

Ο ορθός μετασχηματισμός δηλώνει την μετατροπή των τιμών του διανύσματος λύσης από διακριτές (2 έως 51) σε συνεχείς στο διάστημα 0 έως 1. Κατά τον ορθό μετασχηματισμό επίσης αφαιρούνται όλες οι θέσεις που περιέχουν 1 ώστε να έχουν το ίδιο μέγεθος τα διανύσματα και να είναι δυνατό να πραγματοποιηθούν

πράξεις μεταξύ τους. Η μετατροπή αυτή επιτυγχάνεται διαιρώντας όλα τα στοιχεία του διανύσματος με το μεγαλύτερο (στο συγκεκριμένο πρόβλημα είναι το 51). Άρα ο αλγόριθμος βελτιστοποίησης σμήνους σωματιδίων στην συγκεκριμένη εργασία εφαρμόζεται στη σειρά που θα έχουν οι κόμβοι στο διάνυσμα λύσης με σκοπό να ομαδοποιηθούν με κάποιο τρόπο οι πιο κοντινοί μεταξύ τους ώστε όταν στον ανάστροφο μετασχηματισμό τοποθετηθούν σε διαδρομές να μειωθεί η συνολική απόσταση.

Ο ανάστροφος μετασχηματισμός δηλώνει την μετατροπή των συνεχών τιμών του διαστήματος $[0,1]$ σε διακριτές (2 έως 51). Αυτό επιτυγχάνεται κατατάσσοντας τα στοιχεία του διανύσματος σε αύξουσα σειρά και αντιστοιχίζοντάς τους διακριτές τιμές ξεκινώντας από το 2 (και τελειώνοντας στο συγκεκριμένο πρόβλημα στο 51). Στη συνέχεια αρχίζει η τοποθέτησή τους σε διαδρομές με τη σειρά που είχαν στο διάνυσμα συνεχών τιμών. Κατά την τοποθέτησή τους στη διαδρομή γίνεται έλεγχος παραβίασης των περιορισμών (χωρητικότητας και χρόνου παραμονής στη διαδρομή) και σε περίπτωση παραβίασής τους δημιουργείται καινούρια διαδρομή.

Στην συνάρτηση γίνεται ο έλεγχος της τιμής του inv και ανάλογα αυτής γίνεται η αντίστοιχη διαδικασία.

Στις γραμμές 2-27 πραγματοποιείται ο ανάστροφος μετασχηματισμός ακολουθώντας τη διαδικασία που περιγράφηκε παραπάνω ενώ στις γραμμές 2832 πραγματοποιείται ο ορθός μετασχηματισμός.

Συνάρτηση (route_number_reduction)

Η συνάρτηση **route_number_reduction.m** δέχεται σαν ορίσματα τα εξής:

1. **x**: Είναι το διάνυσμα λύσης.
2. **distances**: Είναι ο πίνακας των αποστάσεων των κόμβων μεταξύ τους.
3. **demand** : Είναι διάνυσμα που περιέχει τη ζήτηση κάθε κόμβου.
4. **maxStorage** : Είναι η μέγιστη χωρητικότητα.
5. **time_distances**: Είναι οι χρονικές αποστάσεις των κόμβων μεταξύ τους.
6. **maxTime**: Είναι ο μέγιστος χρόνος παραμονής του οχήματος στη διαδρομή.
7. **serviceTime**: Είναι ο χρόνος εξυπηρέτησης.

Το πρόβλημα που επιλύεται στην συγκεκριμένη εργασία είναι το Ανοιχτό Πρόβλημα Οδήγησης Οχημάτων (Open Vehicle Routing Problem) στο οποίο συνήθως μελετάται το ενδεχόμενο ενοικίασης οχημάτων, δηλαδή η κάθε επιπλέον διαδρομή επιβαρύνει σημαντικά το συνολικό κόστος. Εάν δηλαδή σε μία πιθανή λύση η μείωση του αριθμού των διαδρομών προκαλεί κάποια αύξηση της συνολικής απόστασης της λύσης, προκύπτει ότι η μείωση του κόστους που θα προκαλέσει η μείωση του αριθμού των διαδρομών θα είναι μεγαλύτερη από την αύξηση του κόστους που θα προκύψει από την συνολική αύξηση της απόστασης. Η λογική αυτή δεν είναι απόλυτη και θα ισχύει μέχρι ένα βαθμό, και επειδή δεν δίνονται στοιχεία κόστους απόστασης και διαδρομής στην εκφώνηση ο βέλτιστος αριθμός διαδρομών που επιλέγεται σε αυτήν την εργασία προκύπτει εμπειρικά και από επιλογή του χρήστη.

Με σκοπό επίλυσης του παραπάνω προβλήματος δημιουργήθηκε η συνάρτηση **route_number_reduction.m** η οποία επιτυγχάνει μείωση του αριθμού των διαδρομών στο διάνυσμα λύσης που δέχεται σαν όρισμα κατά ένα. Για επιπλέον μείωση των διαδρομών μπορεί να κληθεί τόσες φορές όσες η επιθυμητή μείωση. Δεν επιτεύχθηκε η λειτουργία της συνάρτησης σε όλα τα ενδεχόμενα οπότε σε περιπτώσεις υπερβολικά λίγων επιθυμητών διαδρομών, κοντά στις ελάχιστες επιτρεπόμενες από τον αριθμό των κόμβων και των περιορισμών χωρητικότητας και ελάχιστης παραμονής του οχήματος στη διαδρομή, παρουσιάζεται πρόβλημα και είναι αδύνατο να επιστρέψει την επιθυμητή λύση η συνάρτηση. Πρέπει λοιπόν

να χρησιμοποιηθεί η συνάρτηση συνετά και με λογική. Θα παρουσιαστούν παρακάτω οι επιλογές που προτείνονται στη συγκεκριμένη εργασία.

Στη συνάρτηση, αρχικά, μέχρι και την γραμμή 15 μέσω των θέσεων που περιέχουν την τιμή 1, όπως πραγματοποιήθηκε και σε άλλες συναρτήσεις, αναγνωρίζονται οι διαφορετικές διαδρομές και αποθηκεύονται στη λίστα *allroutes*.

Στο βρόγχο των σειρών 17-91 επιχειρείται η μείωση κατά ένα του αριθμού των διαδρομών. Αρχικά στη γραμμή 18 επιλέγεται η διαδρομή που θα αφαιρεθεί και της οποίας οι κόμβοι θα διανεμηθούν στις υπόλοιπες. Οι κόμβοι της αποθηκεύονται στο διάνυσμα *nodes* και στο *q* ταξινομούνται οι υπόλοιπες διαδρομές σε αύξουσα σειρά μεγέθους ώστε να επιχειρηθεί η τοποθέτηση των κόμβων του *nodes* σε αυτές σύμφωνα με αυτή τη σειρά.

Η εισαγωγή των κόμβων στις διαδρομές πραγματοποιείται στο βρόγχο των σειρών 32-77, όπου με την σειρά του διανύσματος *q* επιλέγεται η διαδρομή και στις γραμμές 39-45 επιλέγεται ο κόμβος του *nodes* που αν τοποθετηθεί στη διαδρομή θα προκαλέσει την μεγαλύτερη αύξηση της συνολικής ζήτησης χωρίς να ξεπεράσει τη μέγιστη.

Εάν υπάρχει κόμβος που να μην υπερβαίνει τον περιορισμό εισάγεται στον έλεγχο των γραμμών 46-72 και τοποθετείται στην κατάλληλη θέση στη διαδρομή. Επιλέγεται στις γραμμές 48-56 η κατάλληλη θέση δεξιά της οποίας είναι καλύτερο να τοποθετηθεί ο κόμβος στη διαδρομή μέσω της εξίσωσης $c_{ik} + c_{kj} - c_{ij}$ που εξηγήθηκε παραπάνω και αποθηκεύεται η θέση στη μεταβλητή *position*. Στη συνέχεια, στις γραμμές 58-62, ελέγχεται εάν είναι προτιμότερο να το τοποθετηθεί στο τέλος της διαδρομής, και τοποθετείται εκεί, αλλιώς τοποθετείται στην θέση που επιλέχθηκε προηγουμένως.

Στη συνέχεια, στις γραμμές 64-72 ελέγχεται εάν η τοποθέτηση του κόμβου προκάλεσε παραβίαση του περιορισμού ελάχιστου χρόνου παραμονής στη διαδρομή και εάν δεν παραβιάζεται αφαιρείται ο κόμβος που χρησιμοποιήθηκε από το διάνυσμα *nodes* αλλιώς γίνεται επαναφορά της διαδρομής στην αρχική της κατάσταση και επιχειρείται η ίδια διαδικασία μέσω του βρόγχου στην διαδρομή με το αμέσως μεγαλύτερο μέγεθος.

Τέλος, στις υπόλοιπες γραμμές γίνονται έλεγχοι αριθμού επαναλήψεων και ολοκλήρωσης ελέγχου σε όλες τις πιθανές επιλογές, και σε περίπτωση υπέρβασης

των περιορισμών επιστρέφεται το αρχικό διάνυσμα λύσης. Αλλιώς επιστρέφεται το μειωμένο κατά μια διαδρομή διάνυσμα λύσης.

Συνάρτηση (PLOTS)

Η συνάρτηση **plots.m** δέχεται σαν ορίσματα τα εξής:

1. x: Είναι η συντεταγμένες x των κόμβων.
2. y: Είναι οι συντεταγμένες y των κόμβων.
3. routes: Είναι το διάνυσμα λύσης.

Η συνάρτηση αυτή τυπώνει στο ίδιο γράφημα τις θέσεις των κόμβων της διαδρομής και την πορεία των διαδρομών (κάθε διαδρομή με διαφορετικό χρώμα) με ευθύγραμμα τμήματα να συνδέουν τον προηγούμενο με τον επόμενο κόμβο της διαδρομής.

Επιλογή Τιμών στις Παραμέτρους

Μετά την ολοκλήρωση των αλγορίθμων και των αντιστοιχων συναρτήσεων μελετήθηκε η συμπεριφορά των αποτελεσμάτων σε διάφορες τιμές των παραμέτρων (1) *percentage*: ποσοστό τυχαίας επιλογής επόμενων κόμβων, (2) *c1*: βάρος ροπής της ταχύτητας σωματιδίου προς την προσωπική καλύτερη επίδοση,

(3) c_2 : βάρος ροπής της ταχύτητας σωματιδίου προς την καλύτερη επίδοση όλου του σμήνους.

1. *percentage*: Πολύ μεγάλη τιμή στην τυχαία επιλογή προσφέρει μεγάλη διαφοροποίηση μεταξύ των αρχικών λύσεων αλλά πολύ κακή απόδοση σε αυτές. Πολύ μικρή τιμή, από την άλλη προσφέρει καλύτερα αποτελέσματα αλλά πολύ μικρή διαφοροποίηση, κάτι που εμποδίζει σε μεγάλο βαθμό την απόδοση και λειτουργία του αλγορίθμου βελτιστοποίησης σμήνους σωματιδίων. Με γνώση των παραπάνω και ελέγχους στο διάστημα $[0,1]$ επιλέχθηκε η τιμή **0,3** η οποία προσφέρει αρκετή διαφοροποίηση ώστε να λειτουργεί σωστά ο βασικός αλγόριθμος αλλά και αρκετά καλή απόδοση των λύσεων.
2. c_1 : Η βιβλιογραφία αναφέρει ότι η βέλτιστη τιμή του c_1 ώστε να λειτουργεί σωστά ο αλγόριθμος είναι το 2. Λόγω όμως της μορφής του προβλήματος μετά τον υπολογισμό της νέας τιμής της λύσης, κατά τον ανάστροφο μετασχηματισμό της αλλοιώνεται η μορφή της λύσης χάνεται μεγάλο τμήμα της προόδου των λύσεων. Απαιτείται λοιπόν ισχυρότερη επίδραση του αλγορίθμου στις λύσεις. Παρατηρήθηκε λοιπόν ότι αύξηση της τιμής του c_1 προκαλούσε καλύτερευση των τελικών αποτελεσμάτων. Αυτό μπορεί να εξηγηθεί και θεωρητικά διότι μεγαλύτερη τιμή του c_1 σημαίνει ότι αυξάνεται το βάρος της προσωπικής καλύτερης επίδοσης στον υπολογισμό της ταχύτητας, με αποτέλεσμα στη διάρκεια των επαναλήψεων το σωματίδιο να ωθείται πιο ισχυρά προς την καλύτερη επίδοσή του και να μην του επιτρέπεται να κινείται στο χώρο με αποτέλεσμα χειροτέρευσης της απόδοσής του.
3. c_2 : Όμοια με το c_1 η βιβλιογραφία αναφέρει ως βλετιστη τιμή το 2. Ότι παρατηρήθηκε για το c_1 ισχύει και για το c_2 με τη διαφορά ότι το c_2 αναφέρεται στο βάρος της καλύτερης λύσης σε όλο το σμήνος. Είναι λογικό οπότε ότι είναι προτιμότερη η χρήση του c_2 παρότι έχουν παρόμοια συμπεριφορά. Με δοκιμή όλο και μεγαλύτερων τιμών παρατηρήθηκε μείωση της συνολικής απόστασης αλλά και των συνολικών διαδρομών στις τελικές λύσεις, όμως πολύ μεγάλες τιμές άρχιζαν να έχουν σαν αποτέλεσμα τη διακύμανση των τελικών αποτελεσμάτων, με πολύ καλύτερα αποτελέσματα κάποιες φορές αλλά και χειρότερα ακόμη και από αυτά που δεχόταν σαν είσοδο ο αλγόριθμος κάποιες άλλες φορές. Αφήνεται λοιπόν

στην κρίση του χρήστη η επιλογή της τιμής του $c2$. Σε αυτήν την εργασία προτείνεται η τιμή **2000** η οποία φαίνεται να δίνει σχετικά όμοια αποτελέσματα για να δεδομένα που δέχεται μετά και την εφαρμογή του `Sort` στις αρχικές λύσεις. Δεν έχει νόημα επίσης η πρόσδοση μεγάλης τιμής στο $c1$ και στο $c2$ διότι ακυρώνουν μεταξύ τους ότι θετικό προσέφερε η αύξηση της τιμής τους.

ΑΠΟΤΕΛΕΣΜΑΤΑ


Το αποτέλεσμα του διανύσματος διαδρομών της τελικής λύσης, που τυπώνεται ως τελική προτεινόμενη λύση, προκύπτει από τις παρακάτω διαδικασίες:

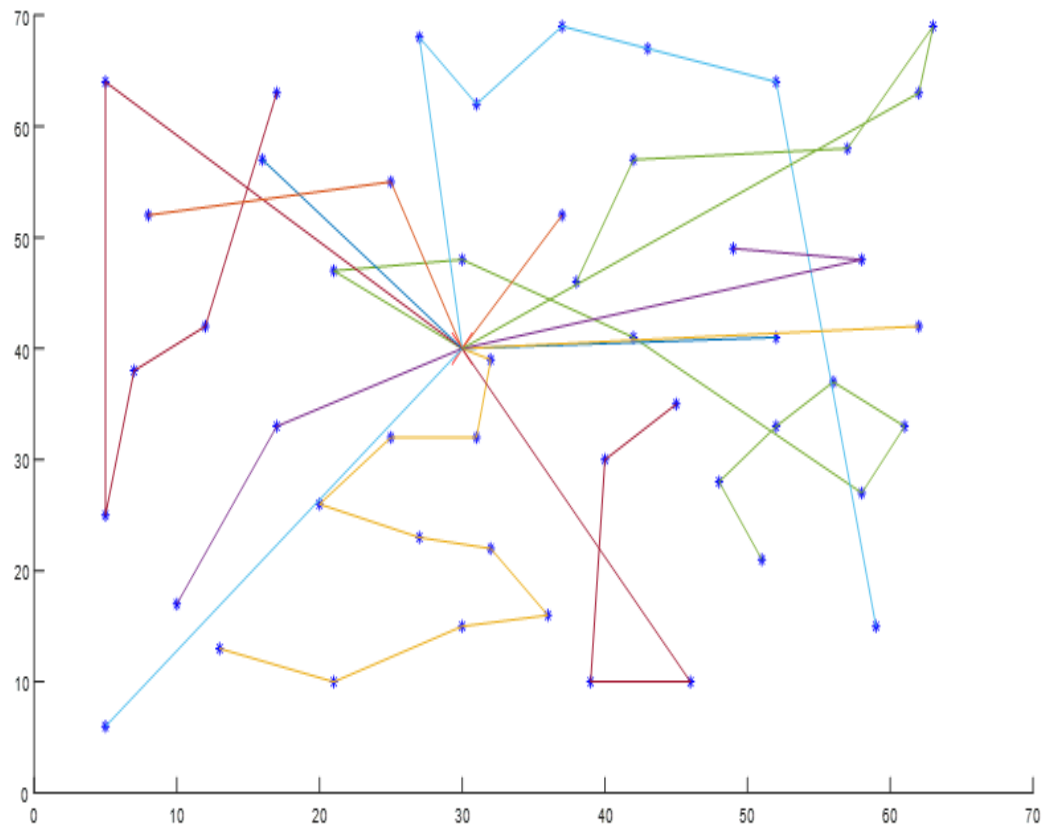
- Δημιουργία 10 αρχικών λύσεων από την συνάρτηση **particle.m** και για *percentage=0,3*.
- Εφαρμογή τοπικής αναζήτησης 3-opt, 1-1 exchange στις αρχικές λύσεις από μία φορά μέσω της συνάρτησης **Three_Opt_1_1_exchange.m**.
- Εφαρμογή του Αλγορίθμου Βελτιστοποίησης Σμήνους Σωματιδίων σε όλο το πλήθος λύσεων (σμήνος) για $c1=2$ και $c2=2000$ από τη συνάρτηση **Optimization.m**.
- Εφαρμογή 20 φορές τοπικής αναζήτησης 3-opt, 1-1 exchange σε κάθε λύση που προκύπτει στο τέλος του Αλγορίθμου Βελτιστοποίησης Σμήνους Σωματιδίων και επιλογή ως προτεινόμενη, αυτής με τη μικρότερη συνολική απόσταση.

Στη συνέχεια παρατίθενται τα αποτελέσματα και οι συνέπειες των παραπάνω διαδικασιών σε ένα τρέξιμο του προγράμματος με ενδεικτικά καλά αποτελέσματα:

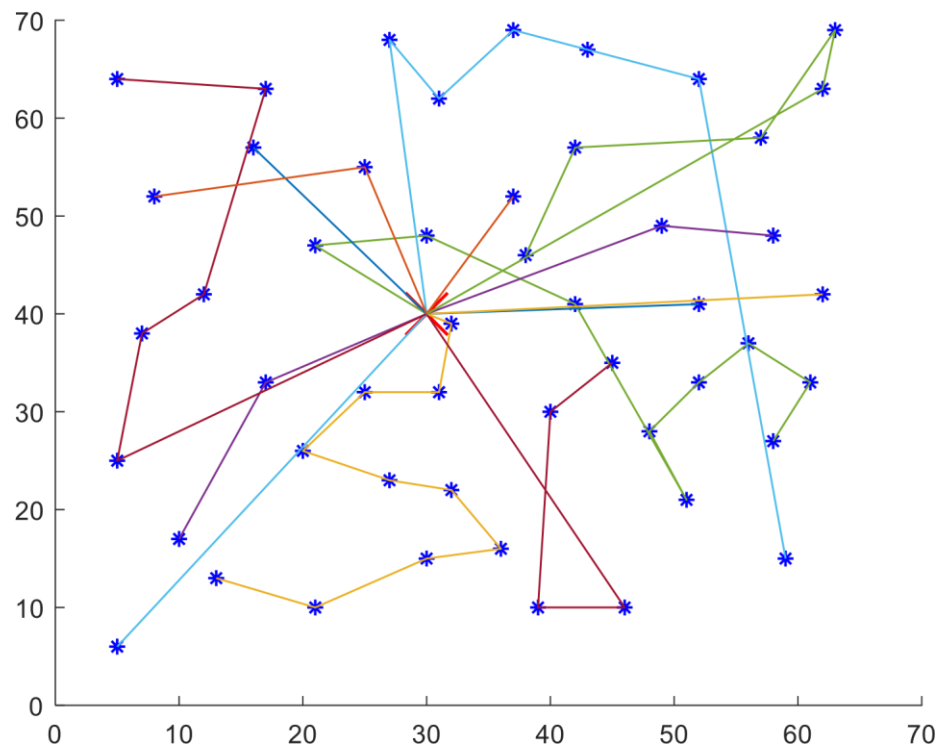
	Μέση συνολική απόσταση	Μέση Βελτίωση Συνολικής απόστασης
Αρχικές Λύσεις	749,8214	-----
3-Opt, 1-1exchange	721,5321	28,2893
Αλγόριθμος Βελτιστοποίησης Σμήνους Σωματιδίων	659,4988	62,03332
Τελικές Λύσεις	639,3498	20,1489
Τελική προτεινόμενη Λύση	605,3527	-----

Και το γράφημα των διαδρομών της τελικής προτεινόμενης λύσης που προκύπτει από κάθε συνάρτησι στο ίδιο τρέξιμο του προγράμματος είναι τα εξής:

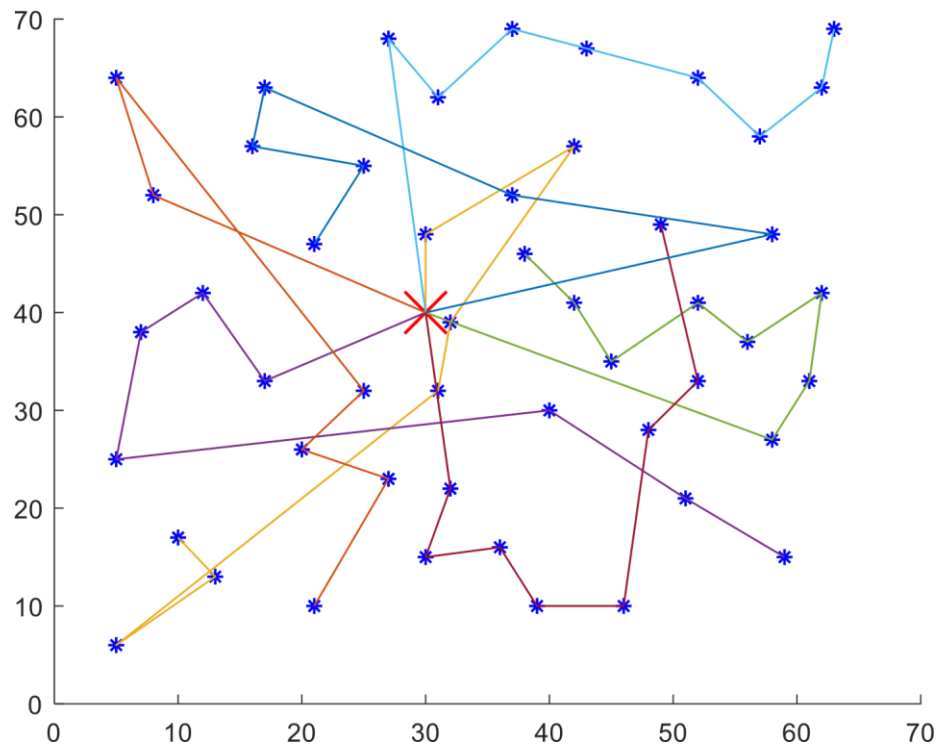
 Αρχική λύση (από particles.m)



Βελτιωμένη λύση μετά από τοπική αναζήτηση (από
Three_Opt_1_1_exchange.m)



✚ Λύση που προκύπτει από τον εξελικτικό αλγόριθμο βελτιστοποίησης σμήνους σωματιδίων (από Optimization.m)



✚ Τελική προτεινόμενη λύση.

