ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΑΡΑΓΩΓΗΣ ΚΑΙ ΔΙΟΙΚΗΣΗΣ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Επίλυση του προβλήματος χρονοπρογραμματισμού φορτηγών για την αποβάθρα εμπορευματοκιβωτίων με χρήση του αλγορίθμου βελτιστοποίησης σμήνους σωματιδίων

Γεώργιος Χριστόπουλος

Επιβλέπων καθηγητής: Δρ. Ιωάννης Μαρινάκης

Χανιά, 2025

Περίληψη

Την σημερινή εποχή της παγκοσμιοποίησης η ανάπτυξη και βελτιστοποίηση της εφοδιαστικής αλυσίδας καθώς και των logistics είναι πιο αναγκαία όσο ποτέ, ιδιαίτερα στον τομέα της ναυτιλίας που ο ανταγωνισμός είναι πιο σκληρός και ανελέητος. Τα λιμάνια τα τελευταία χρόνια δουλεύουν στην πλήρης ,ή σχεδόν πλήρης, χωρητικότητα τους για να μπορέσουν να εξυπηρετήσουν το αυξημένο φόρτο εργασίας, οπότε είναι απαραίτητη η βελτιστοποίηση τον επιμέρους διαδικασιών τους για την απελευθέρωση πολυτίμου χρόνου και χώρου. Σε αυτή την εργασία θα μελετήσουμε το πρόβλημα της δρομολόγησης στις αποβάθρες των λιμανιών που αποτελεί ένα από τα σημαντικότερα προβλήματα για τον προγραμματισμό και διαχείριση λιμανιών. Αυτό το πρόβλημα αφορά τη βελτιστοποίηση της δρομολόγησης των φορτηγών που μεταφέρουν κοντέινερ στις αποβάθρες των λιμανιών μεταξύ των αποθηκευτικών χώρων και πλοίων. Ο στόχος του προβλήματος είναι η μείωση του χρόνου αναμονής, της απόστασης που πρέπει να διανύσουν τα φορτηγά καθώς και του λειτουργικού κόστους. Για την λύση του προβλήματος θα χρησιμοποιηθεί ο αλγόριθμος βελτιστοποίησης σμήνους σωματιδίων (Particle Swarm Optimization - PSO) ο οποίος είναι εμπνευσμένος από τη συλλογική συμπεριφορά σμήνος πουλιών και κοπαδιών ψαριών. Τα σωματίδια αναζητούν την καλύτερη λύση επηρεαζόμενα και από την προσωπική τους καλύτερη θέση αλλά και από τη συλλογική καλύτερη θέση. Με την χρήση του PSO θα βρεθεί η βέλτιστη δρομολόγηση των φορτηγών μειώνοντας τον χρόνο αναμονής καθώς και την απόσταση που θα πρέπει να διανύσουν τα φορτηγά επιτυγχάνοντας την αύξηση της αποδοτικότητας των λιμανιών.

Ευχαριστίες

Θα ήθελα να ευχαριστήσω πολύ τον επιβλέπων καθηγητή μου, Δρ. Ιωάννη Μαρινάκη για την καθοδήγηση του, καθώς και την οικογένεια μου, που με στήριξε όλα αυτά τα χρονιά και ήταν πάντα δίπλα μου.

Περιεχόμενα

Κεφάλαιο 1

1.1 Εισαγωγή

Στην εποχή της παγκοσμιοποίησης οι όγκοι μεταφορών αυξάνονται διαρκώς, αυτό καθιστά την εύρυθμη λειτουργία των λιμανιών κρίσιμο παράγοντα για την αποδοτικότητα και σωστή λειτουργία της εφοδιαστικής αλυσίδας. Τα λιμάνια ως βασικοί κομβόι διακίνησής εμπορευμάτων απαιτούνται να διαχειριστούν με υψηλή ακρίβεια και ταχύτητα πολύπλοκες εργασίες, όπως η φόρτωση και εκφόρτωση πλοίων, η προσωρινή αποθήκευση εμπορευματοκιβωτίων καθώς και τη μεταφορά τους εντός του λιμανιού. Εκτός από τη σύνδεση θαλάσσιων και χερσαίων μεταφορών, τα λιμάνια εκτελούν συνθέτες διαδικασίες όπως η ταξινόμηση, η καταγραφή, η προσωρινή αποθήκευση και η δρομολόγηση κοντέινερ. Η ανάγκη για μείωση του κόστους και ελαχιστοποίηση των καθυστερήσεων έχει οδηγήσει σε μια νέα εποχή λειτουργίας των λιμανιών οπού η αυτοματοποίηση και η βέλτιστη αξιοποίηση πόρων παίζουν πρωταγωνιστικό ρολό. Ένα από τα πιο βασικά προβλήματα που πρέπει να βελτιστοποιήσουν τα λιμάνια για την αύξηση της αποδοτικότητας τους είναι η δρομολόγηση των φορτηγών στις αποβάθρες εμπορευματοκιβώτιων, τα οποία είναι υπεύθυνα για την μεταφορά των εμπορευματοκιβώτιων μεταξύ των σημείων φόρτωσης και εκφόρτωσης καθώς και των αποθηκευτικών χώρων. Η μη αποδοτική διαχείριση των φορτηγών αυτών μπορεί να οδηγήσει σε αυξημένες καθυστερήσεις ολόκληρης της παραγωγικότητας του λιμανιού. Το πρόβλημα του χρονοπρογραμματισμού φορτηγών για τις αποβάθρες εμπορευματοκιβώτιων αποτελεί ένα δύσκολο συνδυαστικό πρόβλημα το οποίο

έχει πολλές παραμέτρους όπως για παράδειγμα: χρονικά παράθυρα εξυπηρέτησης, αρχικές θέσεις των φορτηγών, διακριτές εργασίες φόρτωσης και εκφόρτωσης καθώς και αλλά που θα δούμε σε μετέπειτα κεφάλαιο.

Η πολυπλοκότητα του προβλήματος καθιστά αναποτελεσματικές τις κλασικές μεθόδους επίλυσης ιδιαίτερα για προβλήματα με ρεαλιστική κλίμακα. Για αυτό τον λόγο, χρησιμοποιούνται ευρετικές και μεθευρετικές μέθοδοι σαν τον αλγόριθμο βελτιστοποίησης σμήνους σωματιδίων με τον οποίο θα ασχοληθούμε σε αυτή την εργασία.

1.2 Μεθοδολογία

Ο αλγόριθμος υλοποιήθηκε σε γλώσσα προγραμματισμού C++. Τα δεδομένα του προβλήματος προέρχονται από έναν αλγόριθμο που μου προώθησε ο υπεύθυνος καθηγητής μου Δρ. Ιωάννης Μαρινάκης και βασίζεται σε πραγματικά δεδομένα αλλά μικρότερης κλίμακας. Η λύση του προβλήματος προσεγγίστηκε με τον αλγόριθμο βελτιστοποίησης σμήνους σωματιδίων.

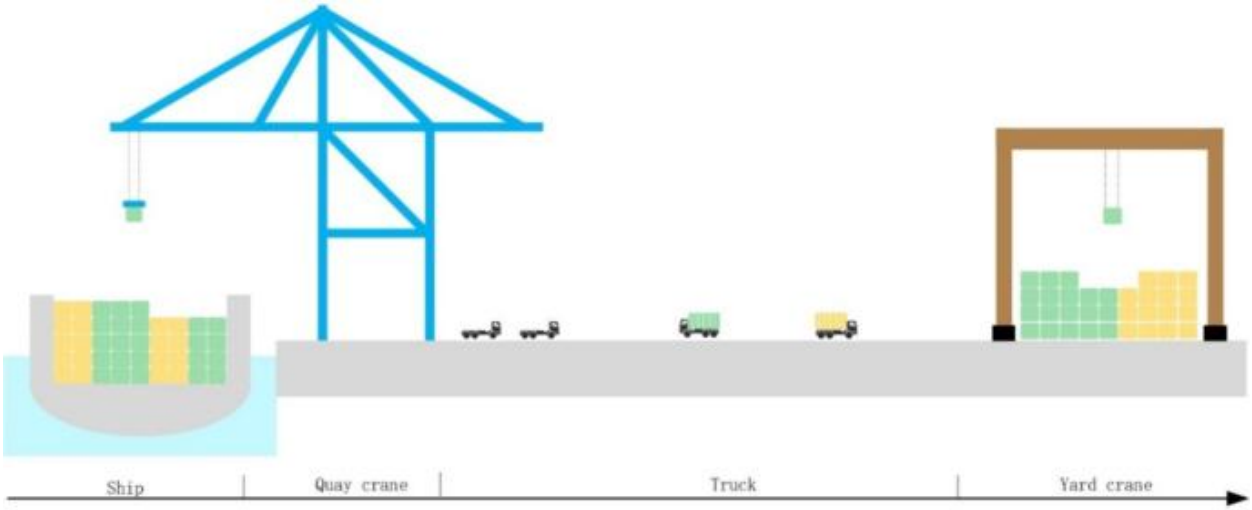
Κεφάλαιο 2

2.1 Εισαγωγή στη γενική λειτουργεία των τερματικών σταθμών εμπορευματοκιβωτίων

Πριν εμβαθύνουμε περισσότερο στο πρόβλημα χρονοπρογραμματισμού των φορτηγών θα εξηγήσουμε την γενική λειτουργία των τερματικών σταθμών εμπορευματοκιβωτίων (container terminal) ώστε ο αναγνώστης να έχει μια καλύτερη ιδέα για το πρόβλημα. Αρχικά, οι τερματικοί σταθμοί εμπορευματοκιβωτίων χωρίζονται σε δυο πλευρές, την πλευρά της θάλασσας (sea side) και την πλευρά της γης (land side).

Στην πλευρά της θάλασσας όπως είναι προφανές οι μόνες εργασίες που εκτελούνται είναι η φόρτωση και εκφόρτωση των πλοίων. Η διαδικασία αυτή γίνεται με τη βοήθεια γερανών αποβάθρας (quay crane).

Στην πλευρά της γης έχουμε περισσότερες εργασίες. Φορτηγά μεταφέρουν τα εμπορευματοκιβώτια από τους γερανούς αποβάθρας στους γερανούς αυλής (yard crane) οι οποίοι με την σειρά τους στοιβάζουν τα εμπορευματοκιβώτια σε χώρους προσωρινής και μη προσωρινής αποθήκευσης. Εκεί γίνεται η απαραίτητη καταμέτρηση, αρχειοθέτηση και δρομολόγηση των εμπορευματοκιβωτίων.



Σχήμα 2.2: Πορεία εμπορευματοκιβωτίων από το καράβι στην αποθήκη πηγή (https://www.nature.com/articles/s41598-025-91140-7)

2.2 Το προβλήματα χρονοπρογραμματισμού φορτηγών για την αποβάθρα εμπορευματοκιβωτίων

Όπως είδαμε στην προηγουμένη ενότητα το πρόβλημα χρονοπρογραμματισμού φορτηγών για την αποβάθρα εμπορευματοκιβωτίων δεν είναι τίποτε άλλο από την δρομολόγηση φορτηγών από τον γερανό αποβάθρας στον γερανό αυλής, και ανάποδα. Μπορεί να ακούγεται απλή διαδικασία αλλά στην πράξη δεν είναι καθόλου, καθώς έχουμε περιορισμένο αριθμό φορτηγών για να εξυπηρετήσουν πολλά εμπορευματοκιβώτια με διαφορετικό χρονικό παράθυρο εξυπηρέτησης το καθένα.

Μπορούμε να διατυπώσουμε το πρόβλημα χρονοπρογραμματισμού φορτηγών για την αποβάθρα εμπορευματοκιβωτίων ως εξής:

Έστω ότι έχουμε ένα σύνολο από n εργασίες, οι οποίες χωρίζονται σε εργασίες φόρτωσης και εργασίες εκφόρτωσης εμπορευματοκιβωτίων. Οι εργασίες φόρτωσης στέλνουν τα φορτηγά σε έναν γερανό αποβάθρας να παραλάβουν το

εμπορευματοκιβώτιο και να το μεταφέρουν σε έναν γερανό αυλής για να το τοποθέτηση σε κάποια αποθήκη. Αντίθετα οι εργασίες εκφόρτωσης παραλαμβάνουν το εμπορευματοκιβώτιο από τις αποθήκες και τους γερανούς αυλής και κατευθύνονται προς τους γερανούς αποβάθρας για να το φορτώσουν σε κάποιο καράβι. Η κάθε εργασία πέρα από τον τύπο της, περιλαμβάνει και άλλα δεδομένα όπως η νωρίτερη έναρξη της δουλείας (esj), το χρονικό παράθυρο εξυπηρέτησης της (ltw) άλλα και τις συντεταγμένες της τοποθεσίας της (pujl για τις εργασίες φόρτωσης και dojl για τις εργασίες εκφόρτωσης). Στην κατοχή μας διαθέτουμε m φορτηγά που ξεκινούν από διαφορετικές αρχικές θέσεις (stt) και κινούνται με σταθερή δεδομένη ταχύτητα (ts). Κάθε εργασία πρέπει να εξυπηρετηθεί από ένα φορτηγό εντός του χρονικού της παραθύρου αλλιώς η εργασία χάνεται.

2.3 Μοντελοποίηση του προβλήματος

Σκοπός της εργασίας είναι να ελαχιστοποιήσουμε τον συνδυασμό της συνολικής καθυστέρησης των εργασιών και της συνολικής απόστασης που θα πρέπει να διανύσουν τα φορτηγά (2,1). Καταρχήν ας ορίσουμε τις παραμέτρους:

- i, j: δείκτες των εργασιών, με $i \neq j$
- p, q: δείκτες τοποθεσιών
- N: αριθμός εμπορευματοκιβώτιων προς μεταφορά
- M: αριθμός διαθέσιμων φορτηγών
- K: σύνολο διαθέσιμων αποθηκών
- J^+: σύνολο εργασιών φόρτωσης
- J^-: σύνολο εργασιών εκφόρτωσης
- w_i: χρόνος έναρξης της εργασίας j
- $[a_i, b_i)$: χρονικό παράθυρο της εργασίας i, με a_i την ώρα έναρξης και b_i την ώρα λήξης
- c_i: χρόνος ολοκλήρωσης της εργασίας i
- $d_i = \max(0, c_i - b_i)$: καθυστέρηση της εργασίας i

- a_1, a_2: βάρη στην αντικειμενική συνάρτηση
- τ_{pq}: χρόνος μετακίνησης μεταξύ των σημείων p και q
- t_i: χρόνος εξυπηρέτησης της εργασίας i
- L_m: αρχική θέση φορτηγού m
- r_m: χρόνος εκκίνησης φορτηγού m
- P_i: θέση παραλαβής εργασίας i
- Q_i: θέση παράδοσης εργασίας i
- S_{ij}: Χρόνος μετάβασης του φορτηγού από το τέλος της εργασίας i στην αρχή της εργασίας j
- X_{ijm}: 1, αν το φορτηγό m εκτελεί την εργασία j μετρά την εργασία i
  0, διαφορετικά
- Y_{im}: 1, αν η εργασίας i εκτελείται από το φορτηγό m
  0, διαφορετικά

Η αντικειμενική συνάρτηση έχει τύπο:

$$\min F = a_1 \times \sum_{i \in M} d_i + a_2 \left(\sum_{i \in N} t_i + \sum_{i,j \in N} S_{ij} \times X_{ijm} \right) \quad (2,1)$$

Υπό

$$\sum_{m \in M} Y_{im} = 1 \quad \forall i \in N \quad (2,2)$$

$$\sum_{j=1, j \neq i}^{N+1} X_{ijm} \leq Y_{im} \quad \forall i \in N, \ \forall m \in M \quad (2,3)$$

$$w_i + S_{ij} + t_i \leq K \times (1 - X_{ijm}) + w_j \quad \forall i, j \in N, \forall m \in M \quad (2,4)$$

$$a_i + d_i \leq c_i \quad \forall i \in N \quad (2,5)$$

$$r_m + \tau_{L_m P_i} + d_i \leq c_i \quad \forall i \in N, \forall m \in M \quad (2,6)$$

$$X_{ijm}, Y_{im} \in \{0, 1\} \quad (2,7)$$

$$w_i, t_i, S_{ij}, d_i \geq 0 \quad (2,8)$$

Ο περιορισμός (2,2) καθορίζει πως κάθε εργασία εξυπηρετείται από ακριβός ένα φορτηγό. Ο περιορισμός (2,3) καθορίζει την σειρά τον εργασιών για το ίδιο φορτηγό. Ο περιορισμός (2,4) καθορίζει την χρονική αλληλουχία μεταξύ των εργασιών. Ο περιορισμός (2,5) καθορίζει την καθυστέρηση της κάθε εργασίας. Ο περιορισμός (2,6) καθορίζει τον χρόνο παράδοσης από την αρχική θέση. Ο περιορισμός (2,7) καθορίζει πως οι συγκεκριμένες μεταβλητές παίρνουν τιμές μηδέν ή ένα. Και τέλος 0 περιορισμός (2,8) καθορίζει πως οι συγκεκριμένες μεταβλητές είναι θετικές καθώς μιλάμε για χρόνο.

Επίσης πρέπει να αναφέρουμε πως οι απόσταση μεταξύ των σημείων j και i βγαίνει από τον τύπο του πυθαγορείου θεωρήματος. Αρά για δυο σημεία i και j με συντεταγμένες (x_i, y_i) και (x_j, y_j) αντίστοιχα έχουμε:

$$\text{distance (i, j)} = \sqrt{(x_j + x_i)^2 + (y_j + y_i)^2} \quad (2,9)$$

Και ο χρόνος τις διαδρομής βγαίνει αν διαιρέσουμε την απόσταση τον δυο σημείων με την ταχύτητα των φορτηγών ts.

$$\tau_{ij} = \frac{distance\ (i,j)}{ts}$$

Κεφάλαιο 3

3.1 Αλγόριθμοι βελτιστοποίησης

Στην εφοδιαστική αλυσίδα συναντάται συχνά το πρόβλημα υπερμεγέθους όγκου πληροφορίας. Αυτό καθιστά ακριβείς μεθόδους όπως ο γραμμικός και δυναμικός προγραμματισμός ως μη αποτελεσματικές για την χρήση σε προβλήματα logistics καθώς, για μεγάλα προβλήματα απαιτούν τεράστια υπολογιστική ισχύ και χρόνο για να καταλήξουν στην βέλτιστη λύση. Για αυτόν τον λόγο οι περισσότεροι που ασχολούνται με αυτόν τον κλάδο χρησιμοποιούν ευρετικούς και μεθευρετικούς αλγορίθμους. Οι δυο αυτές μέθοδοι δεν βρίσκουν βέλτιστη λύση αλλά την προσεγγίζουν με ικανοποιητική ακρίβεια. Ακόμα και αν υπάρχει βέλτιστη λύση με αυτές τις μεθόδους δεν εγγυάται πως θα βρεθεί. Για αυτό θεωρούνται προσεγγιστικές και μη ακριβείς μέθοδοι. Με αυτόν τον τρόπο θυσιάζοντας ακρίβεια μειώνουν σημαντικά την απαραίτητη υπολογιστική ισχύ που απαιτείται για να δοθεί μια λύση κάνοντας τες μονόδρομο για μεγάλα προβλήματα με πολλά δεδομένα ή παραμέτρους ή για προβλήματα που είναι απαραίτητο να δοθεί μια λύση σε λογικό χρόνο.

Οι ευρετικοί χωρίζονται στις εξής κατηγορίες:

- Αλγόριθμος απληστίας
- Προσεγγιστικοί αλγόριθμοί
- Αλγόριθμοι τοπικής αναζήτησης

Στην παρούσα διπλωματική θα παρουσιάσουμε τους αλγορίθμους απληστίας και επίσης τους αλγορίθμους τοπικής αναζήτησης

Οι μεθευρετικοί αλγόριθμοι δημιουργήθηκαν για να μπορέσει μια λύση να ξεφύγει από το τοπικό ελάχιστο και χρειάζονται μια αρχική λύση ή λύσεις για να μπορέσουν να δουλέψουν. Πολύ από αυτούς τους αλγορίθμους είναι εμπνευσμένοι από την φύση. Αξιοσημείωτο είναι επίσης να αναφέρουμε πως τα τελευταία χρονιά οι περισσότεροι αλγόριθμοι που αναπτύσσονται ανήκουν σε αυτήν την κατηγορία.

Οι μεθευρετικοί αλγόριθμοι που θα παρουσιάσουμε είναι:

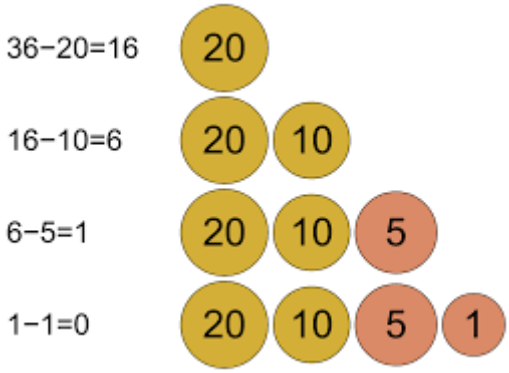
- Αλγόριθμος προσομοιωμένης ανόπτησης
- Αλγόριθμος βελτιστοποίησης αποικίας μυρμηγκιών
- Αλγόριθμος βελτιστοποίησης σμήνους σωματιδίων

3.2 Αλγόριθμος απληστίας

Ο αλγόριθμος απληστίας σε κάθε βήμα που κάνει επιλεγεί μια τοπική λύση του προβλήματος στοχεύοντας αυτή η τοπική λύση να είναι και η ολική βέλτιστη. Ο αλγόριθμος λειτουργεί λαμβάνοντας υπόψιν του σε κάθε βήμα την βέλτιστη επιλογή για εκείνη την χρονική στιγμή αγνοώντας τις επιπτώσεις που μπορεί να έχει στο τέλος. Επίσης ο αλγόριθμος χαρακτηρίζεται ως μυωπικός γιατί βλέπει μόνο μπροστά. Ο αλγόριθμος απληστίας χρησιμοποιείτε σε περιπτώσεις που η εύρεση της ολικής βέλτιστης λύσης είναι πολύ χρονοβόρα ή αδύνατη.

Ας κάνουμε ένα παράδειγμα για να γίνει πιο κατανοητός ο αλγόριθμος απληστίας. Το πρόβλημα των κερμάτων ορίζεται ως έξεις: Έχουμε ένα σετ νομισμάτων που απαρτίζεται από νομίσματα των 1, 5, 10, 20 και μας ζητείται να δώσουμε το ακριβές ποσό τον 36. Το κάθε είδος νομίσματος το έχουμε σε πολύ μεγάλες ποσότητες έτσι δεν έχουμε κάποιον περιορισμό στο πόσες φορές μπορούμε να δώσουμε ένα κέρμα μιας συγκεκριμένης τιμής και μας ζητείται να δώσουμε όσο το δυνατόν λιγότερα νομίσματα μπορούμε. Ο αλγόριθμος στο

πρώτο του βήμα θα διαλέξει το μεγαλύτερο νόμισμα που δεν υπερβαίνει την τιμή 36. Έτσι θα διαλέξει το 20 και η νέα τιμή που θα πρέπει να φτάσει θα είναι το 16. Για την τιμή 16 το μεγαλύτερο νόμισμα που διαθέτουμε είναι το 10, που διαλέγεται και μας μένει υπόλοιπο 6 που με την σειρά του θα διαλέξει κέρμα με τιμη 5. Τέλος το υπολιπόν μας είναι 1 και με κέρμα με τιμή 1 έχουμε λύσει το πρόβλημα με τελική απάντηση (20 + 10 + 5 + 1).



Εικόνα 3.2: Απεικόνιση του λυμένου προβλήματος των κερμάτων με χρήση το αλγορίθμου απληστίας
πηγή (https://en.m.wikipedia.org/wiki/File:Greedy_algorithm_36_cents.svg)

Εδώ αξίζει να επισημάνουμε πως ενώ στο παραπάνω πρόβλημα ο αλγόριθμος απληστίας βρήκε την ολική βέλτιστη λύση, δεν είναι όμως απαραίτητο πως σε όλα τα προβλήματα η επιλογή της τοπικής βέλτιστης λύσης θα οδηγήσει στην ολική βέλτιστη λύση.

3.3 Αλγόριθμοί τοπικής αναζήτησης

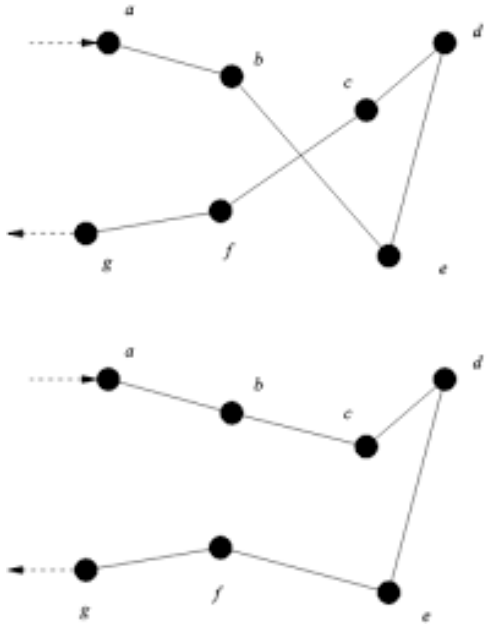
Οι αλγόριθμοι τοπικής αναζήτησης βασίζονται στην αρχαιότερη μέθοδο βελτιστοποίησης, στην μέθοδο δοκιμής και σφάλματος. Οι αλγόριθμοι τοπικής αναζήτησης προσπαθούν μέσω από μια αρχική λύση να βρούνε βέλτιστη λύση

στη γειτονική περιοχή των εφικτών λύσεων. Ως γειτονιά λύσεων ορίζεται το σύνολο όλων των εφικτών λύσεων με μικρή διαφορά από την κεντρική λύση. Όπως είναι προφανές η ποιότητα της αρχικής λύσης παίζει σημαντικό ρολό στην αποτελεσματικότητα της μεθόδου καθώς αν η αρχική λύσει αποκλίνει πολύ από το ολικό βέλτιστο ο αλγόριθμος θα αργήσει πολύ να την προσεγγίσει. Μια σημαντική διαφορά σε σχέση με τους υπολοίπους ευρετικούς αλγόριθμους είναι πως δεν παράγουν οι ίδιοι την αρχική λύση που θα χρησιμοποιήσουν. Για αυτό τον λόγο συνήθως συνδυάζονται με κάποιον άλλον ευρετικό αλγόριθμο για να βελτιστοποιήσουν την λύση του. Η χρήση κάποιου αλλού αλγορίθμου όμως δεν είναι απαραίτητη καθώς μπορεί να δοθούν τυχαίες αρχικές λύσεις. Στα επόμενα υπό κεφάλαια θα παρουσιάσουμε τρεις αλγορίθμους τοπικής αναζήτησης:

- Αλγόριθμος 2opt
- Αλγόριθμος 1-0 επανατοποθέτησης
- Αλγόριθμος 1-1 ανταλλαγής

3.3.1 Αλγόριθμος 2opt

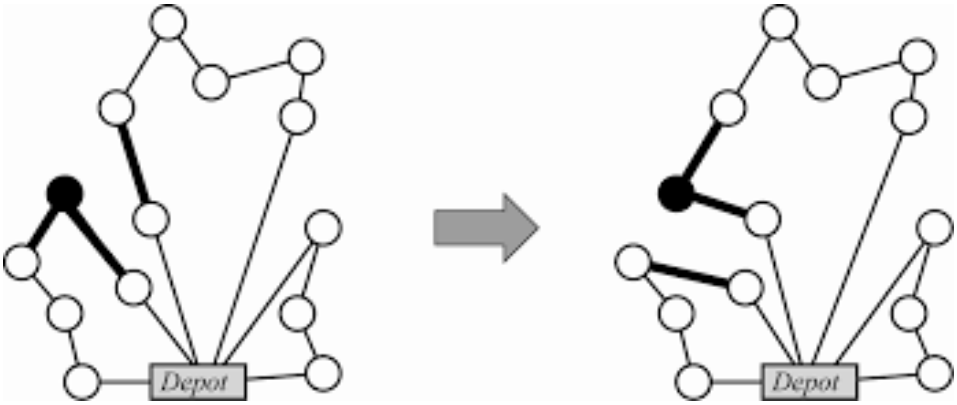
Ο 2opt αλγόριθμος τοπικής αναζήτησης προτάθηκε από τον Croes. Η βασική ιδέα είναι η αφαίρεση δυο ακμών από μια υπάρχουσα διαδρομή και η επανασύνδεση τους με διαφορετικό τρόπο ώστε να εξαλειφθούν οι διασταυρώσεις και να μειωθεί η συνολική απόσταση. Αυτή η διαδικασία επαναλαμβάνεται ώσπου να μην μπορούν να γίνουν περεταίρω βελτιστοποίησης στην λύση.



Εικόνα 3.3.1: παράδειγμα αλγορίθμου 2opt
πηγή (https://link.springer.com/chapter/10.1007/978-3-030-73882-2_98)

3.3.2 Αλγόριθμος 1-0 επανατοποθέτησης

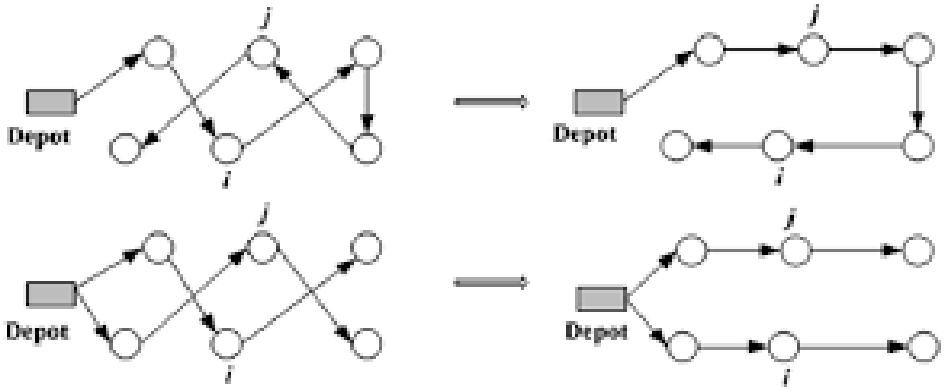
Ο αλγόριθμος 1-0 επανατοποθέτησης προτάθηκε από τον Waters. Η βασική του λειτουργία είναι η αποκοπή ενός κόμβου από μια διαδρομή και η επανατοποθέτηση του σε μια άλλη με σκοπό τη μείωση του κόστους. Αν η νέα λύση έχει καλύτερο κόστος ορίζεται ως η νέα βέλτιστη και η διαδικασία επαναλαμβάνεται έως ότου να μην μπορεί να γίνει περεταίρω βελτιστοποίηση.



Εικόνα 3.3.2: παράδειγμα αλγορίθμου 1-0 επανατοποθέτησης
πηγή (https://www.gerad.ca/~alainh/FinkeAnglais.pdf)

3.3.3 Αλγόριθμος 1-1 ανταλλαγής

Ο αλγόριθμος 1-1 ανταλλαγής προτάθηκε από τον Waters. Μοιάζει πάρα πολύ με τον αλγόριθμο 1-0 επανατοποθέτησης με την κυριά διαφορά τους να είναι πως στον 1-1 ανταλλαγής διαλέγονται δυο κομβόι αντί για έναν για να ανταλλάξουν τις διαδρομές τους. Οι κομβόι μπορεί να είναι και από την ίδια διαδρομή ή από διαφορετικές. Αν η ανταλλαγή αυτών τον κόμβων μειώσει το συνολικό κόστος και οδηγήσει σε καλύτερη λύση, αυτή γίνεται αυτόματος και η νέα βέλτιστη. Η διαδικασία επαναλαμβάνεται ως όπου να μην μπορούν να γίνουν περαιτέρω βελτιστοποιήσεις.



Εικόνα 3.3.3: παράδειγμα αλγορίθμου 1-1 ανταλλαγής
πηγή
(https://www.researchgate.net/publication/257551274_A_memetic_algorithm_for_the_open_capacitated_arc_routing_problem/figures?lo=1&utm_source=google&utm_medium=organic)

3.4 Αλγόριθμος προσομοιωμένης ανόπτησης

Ο αλγόριθμος προσομοιωμένης ανόπτησης έχει εμπνευστεί από τη διαδικασία της ανόπτησης στη μεταλλουργία. Η ανόπτησης είναι η διαδικασία όπου θερμαίνουμε ένα μέταλλο και στη συνέχεια ψύχεται αργά για να επιτευχθεί η κατάλληλη κρυσταλλική δομή. Με αυτή την λογική λειτουργεί και ο αλγόριθμος. Ξεκινώντας από μια αρχική λύση ο αλγόριθμος τροποποιεί τη λύση και την ελέγχει. Οι τροποποιήσεις είναι μικρές αλλαγές στην τωρινή λύση και σε περίπτωση που η νέα λύση είναι καλύτερη από την προηγουμένη γίνεται η νέα λύση. Αν όμως η νέα λύση είναι χειρότερη από την τωρινή ο αλγόριθμος μπορεί και πάλι να την δεχθεί με μια ορισμένη πιθανότητα που μειώνεται με την πάροδο του χρόνου. Η συνάρτηση που ορίζει την τιμή της πιθανότητας αποδοχής χειρότερης λύσης ονομάζεται θερμοκρασία. Η πιθανότητα αποδοχής όπως γίνεται και με την θερμοκρασία στα μέταλλα ξεκινάει από υψηλές τιμές και όσο κυλάει ο χρόνος μικραίνει. Αυτό γίνεται για να ξεκολλήσει ο αλγόριθμος από τοπικές βέλτιστες λύσεις και να αρχίσει να συγκλίνει σε ολικές βέλτιστες λύσεις.

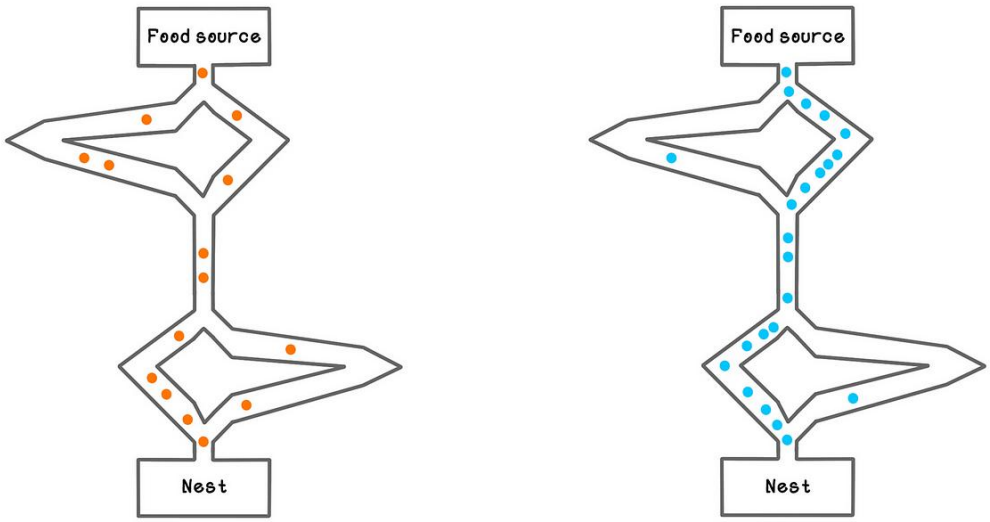
3.5 Αλγόριθμος βελτιστοποίησης αποικίας μυρμηγκιών

Ο αλγόριθμος βελτιστοποίησης αποικίας μυρμηγκιών έχει εμπνευστεί από τον τρόπο με τον οποίο τα μυρμήγκια συνεργάζονται και επικοινωνούν μεταξύ τους για να βρουν την μικρότερη διαδρομή αναμεσά στην φώλια τους και την τροφή. Τα μυρμήγκια όταν ταξιδεύουν αφήνουν από πίσω τους φερομόνη. Με αυτό τον τρόπο τα πρώτα μυρμήγκια που θα ξεκινήσουν θα κινηθούν σε διαφορετικές κατευθύνσεις το καθένα με σκοπό να βρουν την τροφή. Τα επόμενα μυρμήγκια που θα ξεκινήσουν να βρουν την τροφή θα ακολουθήσουν τα μονοπάτια φερομόνης που αφήσαν τα πρώτα. Επειδή η βέλτιστη διαδρομή που βρήκε ένα μυρμήγκι είναι μικρότερη σε σχέση με τις υπόλοιπες η συγκέντρωση φερομόνης σε αυτό το μονοπάτι θα είναι μεγαλύτερη από τα υπόλοιπα. Σαν

αποτέλεσμα μετά από έναν αριθμό επαναλήψεων επειδή η περισσότερη φερομόνη προσελκύει περισσότερα μυρμήγκια, αυτά ακολουθούν μόνο αυτό το μονοπάτι. Με παρόμοιο τρόπο δουλεύει και ο αλγόριθμος. Κάθε σωματίδιο-μυρμήγκι επιλεγεί την επόμενη του διαδρομή με βάση μια πιθανότητα που βασίζεται σε δυο παράγοντες:

- Την ποσότητα φερομόνης σε κάθε εναλλακτική
- Μια συνάρτηση επιθυμίας που επηρεάζεται από παράγοντες όπως το κόστος ή η απόσταση

Έτσι όσο περισσότερη φερομόνη υπάρχει σε μια διαδρομή και η συνάρτηση επιθυμίας βγάζει ικανοποιητικά αποτελέσματα τόσο πιο πιθανό είναι ένα σωματίδιο-μυρμήγκι να την επιλέξει. Αφού όλα τα μυρμήγκια έχουν κατασκευάσει μια διαδρομή ο αλγόριθμος αξιοποιεί την ποιότητα της κάθε λύσης και ενισχύει με παραπάνω φερομόνη τις βέλτιστες διαδρομές και ταυτόχρονα μειώνει μέσω της εξάτμισης της λιγότερο αποδοτικές λύσεις. Αυτή η διαδικασία επαναλαμβάνεται για έναν αριθμό επαναλήψεων με σκοπό η τελική λύση να συγκλίνει στην ολική βέλτιστη.



After 4 minutes

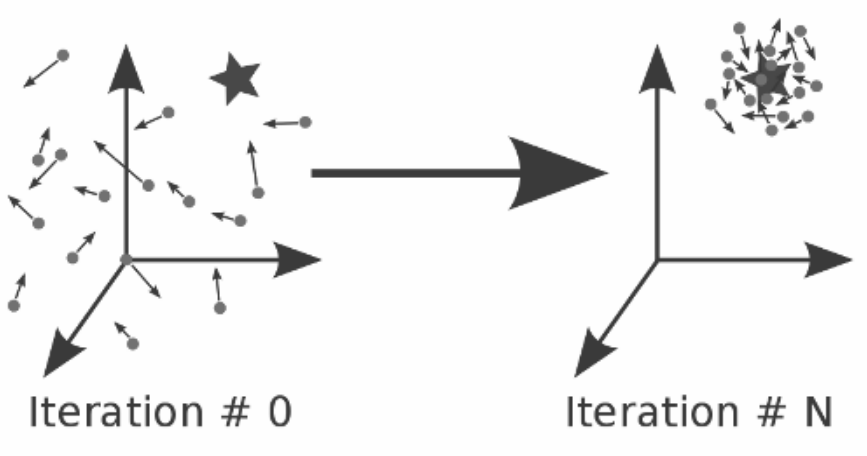
After 8 minutes

Grokking Artificial Intelligence Algorithms

Εικόνα 3.5: παράδειγμα αλγορίθμου βελτιστοποίησης αποικίας μυρμηγκιών
πηγή (https://wikidocs.net/202286)

03.6 Αλγόριθμος βελτιστοποίησης σμήνους σωματιδίων

Ο αλγόριθμος βελτιστοποίησης σμήνους σωματιδίων προτάθηκε από τους Kennedy και Eberhart το 1995. Η πηγή έμπνευσης του αλγορίθμου προήλθε από την παρατήρηση της κοινωνικής συμπεριφοράς κάποιων οργανισμών όπως το πέταγμα των πουλιών σε μορφή σμήνους και την ομαδική κίνηση των ψαριών σε κοπάδια. Η αρχική ιδέα ήταν να προσομοιωθεί γραφικά η κίνηση που κάνει ένα σμήνος από πουλιά και να καταγράφουν οι κανόνες που κάνουν το σμήνος να μην χάνει τον σχηματισμό του όταν αλλάζει ξαφνικά η κατεύθυνση του. Χαρακτηριστικό του αλγορίθμου βελτιστοποίησης σμήνους σωματιδίων είναι πως κάθε σωματίδιο ανανεώνει την θέση του με βάση την δικιά του προσωπική εμπειρία και την γενική εμπειρία ολοκλήρου του σμήνους, καθώς κάθε σωματίδιο επικοινωνεί με το σύνολο. Αυτό είναι το βασικό πλεονέκτημα το συγκεκριμένου αλγορίθμου συν το ότι όλες οι πληροφορίες δεν χάνονται διότι υπάρχει μνήμη. Με αυτό τον τρόπο σε κάθε επανάληψη του αλγορίθμου τα σωματίδια κινούνται προς καλύτερες λύσεις που είχαν βρει τα ιδιά αλλά και συνολικά όλο το σμήνος μαζί.



Εικόνα 3.6: παράδειγμα αλγορίθμου βελτιστοποίησης σμήνους σωματιδίων πηγή (https://esa.github.io/pagmo2/docs/cpp/algorithms/pso.html)

3.6.1 Περιγραφή αλγορίθμου βελτιστοποίησης σμήνους σωματιδίων

Αρχικά δημιουργείται ένας αριθμός από σωματίδια που αντιπροσωπεύουν μια πιθανή λύση του προβλήματος. Το κάθε σωματίδιο κατέχει μια θέση στο χώρο λύσεων και κινείται με μια δεδομένη ταχύτητα.

Η θέση που έχει το κάθε σωματίδιο αντιπροσωπευθεί μια συγκεκριμένη λύση στο πρόβλημα και αναπαρίσταται με ένα n διαστάσεων διάνυσμα στο χώρο των λύσεων x_{ij} με i = 1, 2, …., N και j = 1, 2, …., n όπου N το μέγεθος του πληθυσμού των σωματιδίων και n ο αριθμός των διαστάσεων του διανύσματος. Η απόδοση των σωματιδίων εκτιμάται από μια προκαθορισμένη συνάρτηση ποιότητας fitness function $f(x_{ij})$. Η ταχύτητα u_{ij} αντιπροσωπεύει τις μεταβολές που θα πραγματοποιηθούν για να κινηθούν τα σωματίδια από τη μια θέση στην άλλη. Προς τα που θα κινηθούν τελικά τα σωματίδια υπολογίζεται από την δυναμική αλληλεπίδραση της προσωπικής εμπειρίας ενός σωματιδίου αλλά και ολοκλήρου του σμήνους. Το κάθε σωματίδιο έχει τρεις πιθανότητες για να αλλάξει την τροχιά του κατά την διάρκεια της κίνησης του. Η πρώτη είναι να συνεχίσει να κινείται προς την κατεύθυνση της προηγουμένης του κίνησης, η δεύτερη είναι να κινηθεί προς την βέλτιστη θέση που είχε βρει κατά την

διάρκεια των επαναλήψεων $pbest_{ij}$ και η τρίτη είναι να κινηθεί προ τη βέλτιστη θέση που είχε βρει ένα οποιοδήποτε σωματίδιο στο σμήνος $gbest_j$. Οι εξισώσεις που ορίζουν την ταχύτητα και την θέση των σωματιδίων είναι:

$$u_{ij}(t+1) = u_{ij}(t) + c_1 \times rand_1 \times (pbest_{ij} - x_{ij}(t)) + c_2 \times rand_2 \times (gbest_j - x_{ij}(t)) \quad (3,1)$$

$$x_{ij}(t+1) = x_{ij}(t) + u_{ij}(t+1) \quad (3,2)$$

όπου t είναι ο μετρητής των επαναλήψεων, c_1 και c_2 είναι οι μεταβλητές επιτάχυνσης και $rand_1$ και $rand_2$ είναι δυο τυχαίες μεταβλητές στο διάστημα [0, 1]. Οι μεταβλητές επιτάχυνσης c_1 και c_2 καθορίζουν το ποσό μακριά μπορεί να κινηθεί ένα σωματίδιο κατά τη διάρκεια μιας επανάληψης. Μικρές τιμές των c_1 και c_2 επιτρέπουν στα σωματίδια να βρεθούν πολύ μακριά από τη στενευμένη περιοχή πριν βρεθούν κοντά σε τοπικό ελάχιστο, αντίθετα με μεγάλες τιμές έχουμε μια απότομη κίνηση προς τις στενευμένες περιοχές. Συνήθως και οι δυο μεταβλητές παίρνουν την τιμή 2 ωστόσο σε κάποια προβλήματα διαφορετικές τιμές στα c_1 και c_2 οδηγούν σε καλύτερα αποτελέσματα.

Για προβλήματα ελαχιστοποίησης η βέλτιστη θέση $pbest_{ij}$ ενός σωματιδίου υπολογίζεται από την εξίσωση:

$$pbest_{ij} = \begin{cases} x_{ij}\ (t+1), \text{εάν } (f\left(x_{ij}(t+1)\right) < f\left(\left(x_{ij}(t)\right)\right)) \\ pbest_{ij}\ \alpha\lambda\lambda\iota\acute{\omega}\varsigma \end{cases} \quad (3,3)$$

Για προβλήματα μεγιστοποίησης η βέλτιστη θέση $pbest_{ij}$ ενός σωματιδίου υπολογίζεται από την εξίσωση:

$$pbest_{ij} = \begin{cases} x_{ij}\ (t+1), \text{εάν } (f\left(x_{ij}(t+1)\right) > f\left(\left(x_{ij}(t)\right)\right)) \\ pbest_{ij}\ \alpha\lambda\lambda\iota\acute{\omega}\varsigma \end{cases} \quad (3,4)$$

η βέλτιστη θέση όλου του σμήνους τη χρονική στιγμή t υπολογίζεται από την εξίσωση:

$$gbest_j \in \{pbest_{1j}, pbest_{2j}, ..., pbest_{Nj} | f(gbest_j)\} =$$

$$\min \{f(pbest_{1j}), f(pbest_{2j}), ..., f(pbest_{Nj})\} \quad (3,5)$$

Η εξίσωση ταχύτητας (3,1) αποτελείται από τρεις όρους που επηρεάζουν το τελικό αποτέλεσμα μια χρονική στιγμή t +1:

- Ο πρώτος όρος είναι η προηγουμένη ταχύτητα $u_{ij}(t)$ του σωματιδίου και χρησιμοποιείται σαν μνήμη των προηγουμένων ταχυτήτων που είχε πάρει το σωματίδιο την προηγουμένη χρονική στιγμή. Λόγο της μνήμης το σωματίδιο δεν μπορεί να αλλάξει απότομα την κατεύθυνση του κατά τη διάρκεια μιας επανάληψης.
- Ο δεύτερος όρος ($c_1 \times rand_1 \times (pbest_{ij} - x_{ij}(t))$) ο οποίος λέγεται και γνωστικός όρος χρησιμοποιείται για να ποσοτικοποιήσει την απόδοση κάθε σωματιδίου σε σχέση με τις αποδόσεις του κατά το παρελθόν. Στην ουσία αυτός ο όρος παίζει τον ρολό της επαναφοράς του σωματιδίου σε προηγούμενες θέσεις που ήταν πολύ πιο αποδοτικές για το ίδιο το σωματίδιο.
- Ο τρίτος όρος ($c_2 \times rand_2 \times (gbest_j - x_{ij}(t))$) ο οποίος λέγεται και κοινωνικός όρος χρησιμοποιείται για να ποσοτικοποιήσει την απόδοση κάθε σωματιδίου σε σχέση με τις αποδόσεις ολοκλήρου του σμήνους. Στην ουσία αυτός ο όρος παίζει τον ρολό της αναζήτησης του σωματιδίου με την καλύτερη απόδοση στο σμήνος.

Για να επιτευχθεί επιτυχή σύγκλιση του αλγορίθμου είναι απαραίτητη η σωστή αρχικοποίηση των σωματιδίων, δηλαδή η επιλογή αρχικών λύσεων. Ο συνηθέστερος τρόπος αρχικοποίησης γίνεται δίνοντας τυχαίες θέσεις στα

σωματίδια στον χώρο των λύσεων. Στον αλγόριθμο μας με αυτό τον τρόπο αρχικοποιήσαμε τις λύσεις μας.

Η ταχύτητα αρχικοποιείται σχεδόν πάντα με την τιμή μηδέν. Οι αρχικές τιμές του pbest είναι ίσες με τις αρχικές θέσεις των σωματιδίων. Ο αλγόριθμος σταματά όταν ολοκληρωθεί το σύνολο τον επαναλήψεων.

Οι μεταβλητές επιτάχυνσης c_1 και c_2 και οι τυχαίες μεταβλητές $rand_1$ και $rand_2$ ελέγχουν την επιρροή του γνωστικού και κοινωνικού παράγοντα στη ταχύτητα του σωματιδίου. Εάν $c_1 = c_2 = 0$ τα σωματίδια κινούνται μόνο προς την κατεύθυνση της ταχύτητας τους. Εάν το $c_1 > 0$ και το $c_2 = 0$ τότε το κάθε μέλος του πληθυσμού επηρεάζεται μόνο από τις προηγούμενες κινήσεις του και κινείται ανεξάρτητα από τα άλλα σωματίδια του σμήνους. Στην αντίθετη περίπτωση όπου $c_1 = 0$ και το $c_2 > 0$ ολόκληρο το σμήνος κυνηγάει το βέλτιστο σωματίδιο. Ο στόχος είναι να βρεθεί η κατάλληλη ισορροπία μεταξύ του c_1 και του c_2. Τις περισσότερες φορές δίνεται $c_1 = c_2$, που σημαίνει ότι το σωματίδιο το έλκουν και οι δύο παράγοντες ισόποσα. Συνήθης πρακτική είναι να δίνονται μεταβλητές τιμές στα c_1 και c_2 ούτως ώστε να μεταβάλλεται η επιρροή των δύο παραγόντων κατά τη διάρκεια των επαναλήψεων. Αν θεωρήσουμε ότι το $c_{1,min}$, $c_{1,max}$, $c_{2,min}$, $c_{2,max}$ είναι οι μέγιστες και ελάχιστες τιμές που θα μπορούν να πάρουν τα c_1, c_2 αντίστοιχα, τότε:

$$c_1 = c_{1,min} + \frac{c_{1,min} - c_{1,max}}{iter_{max}} \times t \quad (3,6)$$

$$c_2 = c_{2,min} + \frac{c_{2,min} - c_{2,max}}{iter_{max}} \times t \quad (3,7)$$

όπου τ είναι ο αριθμός της τρέχουσας επανάληψης και $iter_{max}$ ο μέγιστος αριθμός των επαναλήψεων. Στις πρώτες επαναλήψεις του αλγορίθμου οι τιμές c_1 και c_2 είναι μικρές άρα υπάρχει μεγάλη ελευθέρια κίνησης μέσα στον χώρο των λύσεων για να βρεθεί πιο γρηγορά η βέλτιστη λύση.

3.6.2 Παραλλαγές του αλγορίθμου βελτιστοποίησης σμήνους σωματιδίων

Για να βελτιωθεί ο αλγόριθμος βελτιστοποίησης σμήνους σωματιδίων έχουν προταθεί πολλές παραλλαγές, τρεις από αυτές είναι:

1) Ο Shi και Eberhart προτείναν μια παραλλαγή του αλγορίθμου, η οποία χρησιμοποιεί ένα βάρος αδράνειας w. Ο ρόλος του βάρους αδράνειας είναι να ελέγχει την επιρροή των προηγουμένων ταχυτήτων στην εξίσωση της τρέχουσας ταχύτητας. Η νέα εξίσωση της ταχύτητάς είναι:

$$u_{ij}(t+1) = w \times u_{ij}(t) + c_1 \times rand_1 \times (pbest_{ij} - x_{ij}(t)) + c_2 \times rand_2 \times (gbest_j - x_{ij}(t)) \quad (3,8)$$

όπου

$$w = w_{max} - \frac{w_{max} - w_{min}}{iter_{max}} \times t \quad (3.9)$$

Τα w_{max} και w_{min} είναι η μέγιστη και ελάχιστη τιμή του βάρους αδράνειας και $iter_{max}$ ο μέγιστος αριθμός επαναλήψεων.

2) Με σκοπό να μην υπάρχει απομάκρυνση από το ολικό βέλτιστο, να εξαφανιστεί η σύγκλιση του αλγορίθμου και να εξαλειφθούν οι παράμετροι που περιορίζουν την ταχύτητα των σωματιδίων, χρησιμοποιείται ένας παράγοντας περιορισμού στις ταχύτητες των σωματιδίων. Η νέα εξίσωση της ταχύτητάς είναι:

$$u_{ij}(t+1) = x \times u_{ij}(t) + c_1 \times rand_1 \times (pbest_{ij} - x_{ij}(t)) + c_2 \times rand_2 \times (gbest_j - x_{ij}(t)) \quad (3,10)$$

όπου

$$x = \frac{2}{|2 - c - \sqrt{c^2 - 4 \times c}|} \text{ και } c = c_1 + c_2,\ c > 4 \quad (3,11)$$

3) Ο Kennedy δοκίμασε δύο διαφορετικά μοντέλα για τον υπολογισμό της ταχύτητας που δεν χρησιμοποιούν και τους τρείς παράγοντες που περιγράψαμε στο προηγούμενο κεφάλαιο.

   Το πρώτο το ονόμασε γνωστικό μόνο μοντέλο και δεν χρησιμοποιεί καθόλου τον κοινωνικό παράγοντα στην εξίσωση της ταχύτητας. Η νέα εξίσωση της ταχύτητάς είναι:

   $u_{ij}(t + 1) = u_{ij}(t) + c_1 \times rand_1 \times (pbest_{ij} - x_{ij}(t))$ (3,12)

   Το δεύτερο το ονόμασε κοινωνικό μόνο μοντέλο και δεν χρησιμοποιεί καθόλου τον γνωστικό παράγοντα στην εξίσωσης της ταχύτητας. Η νέα εξίσωση της ταχύτητας είναι:

   $u_{ij}(t + 1) = u_{ij}(t) + c_2 \times rand_2 \times (gbest_j - x_{ij}(t))$ (3,13)

Κεφάλαιο 4

Στην παρούσα διπλωματική όπως έχουμε ήδη αναφέρει έχουμε να αντιμετωπίσουμε το πρόβλημα χρονοπρογραμματισμού φορτηγών για την αποβάθρα εμπορευματοκιβωτίων. Θα γίνει μια προσέγγιση με την χρήση του αλγορίθμου βελτιστοποίησης σμήνους σωματιδίων με τυχαίες αρχικές λύσεις. Στο παρόν κεφάλαιο θα γίνει αναφορά στα δεδομένα που χρησιμοποιήθηκαν καθώς και ανάλυση στην υλοποίηση του κώδικα.

4.1 Δεδομένα προβλήματος

Όπως αναφέραμε και στο κεφάλαιο 1.3 τα δεδομένα που θα χρησιμοποιήσουμε για το πρόβλημα μας προέρχονται από έναν αλγόριθμο γραμμένο σε γλώσσα προγραμματισμού c++. Τον συγκεκριμένο αλγόριθμο μας τον παρείχε ο υπεύθυνος καθηγητής Δρ. Ιωάννης Μαρινάκης. Ο αλγόριθμος κάθε φορά που εκτελείτε βγάζει διαφορετικά δεδομένα έτσι ώστε να μπορέσει να καλύψει πολλά διαφορετικά ρεαλιστικά σενάρια του συγκεκριμένου προβλήματος. Θα τρέξομε τον αλγόριθμο δυο φορές για να λύσουμε δυο διαφορετικά σετ δεδομένων. Τα τυχαία δεδομένα που πήραμε τρέχοντας τον αλγόριθμο την πρώτη φορά είναι:

n	8
m	3
ts	11,1

a_1	0,6
a_2	0,6
lc	1
dc	7
storage	8
lc_vector	3
dc_vector	1, 2, 4, 5, 6, 7, 8
stt	0, 0, 0
esj	1465, 360, 91, 1003, 416, 499, 1444, 812
ltw	277, 202, 224, 225, 374, 261, 455, 435
stl	(726, 1460), (1411, 116), (290, 160)
pujl	(1104, 1435)
dojl	(1027, 1204), (296, 959), (1053, 61), (196, 117), (288, 385), (1305, 1211)
sc	(1147, 912), (184, 1249), (247, 862), (1430, 76), (1031, 138), (1268, 1118), (769, 24), (1307, 1106)

Ενώ την δεύτερη είναι:

n	8
m	3
ts	11,1
a_1	0,6
a_2	0,6
lc	6
dc	2
storage	8
lc_vector	1, 2, 3, 5, 6, 8
dc_vector	4, 7
stt	0, 0, 0

esj	683, 593, 579, 224, 1134, 738, 1207, 37
ltw	352, 303, 419, 325, 294, 463, 314, 271
stl	(695, 202), (1221, 1401), (131, 1385)
pujl	(1194, 1489), (936, 2), (470, 1281), (1242, 202), (256, 275), (962, 576)
dojl	(977, 599), (746, 880)
sc	(1148, 1129), (1280, 1301), (905, 1246), (1124, 491), (1021, 934), (952, 893), (606, 1236), (998, 881)

Οπού

- n: αριθμός εργασιών
- m: αριθμός φορτηγών
- ts: ταχύτητά φορτηγών
- a_1, a_2: βάρη στην αντικειμενική συνάρτηση
- lc: αριθμός εργασιών φόρτωσης
- dc: αριθμός εργασιών εκφόρτωσης
- storage: αριθμός αποθηκών
- lc_vector: ποιες εργασίες συγκεκριμένα είναι φόρτωσης
- dc_vector: ποιες εργασίες συγκεκριμένα είναι εκφόρτωσης
- stt: ώρα έναρξης των φορτηγών
- esj: έναρξη χρονικού παραθύρου της κάθε εργασίας
- ltw: χρονική διάρκεια του χρονικού παραθύρου της κάθε εργασίας
- stl: αρχικές συντεταγμένες φορτηγών
- pujl: συντεταγμένες εργασιών φόρτωσης

- dojl: συντεταγμένες εργασιών εκφόρτωσης
- sc: συντεταγμένες αποθηκών

Εδώ οφείλουμε να αναφέρουμε πως έξι από τα παραπάνω δεδομένα παραμένουν σταθερά όσες φορές και αν τρέξουμε τον κώδικα. Ίδια παραμένουν: ο αριθμός των εργασιών n, ο αριθμός των φορτηγών m, η ταχύτητα των φορτηγών ts, τα βάρη της αντικειμενικής συνάρτησης a_1, a_2 καθώς και η ώρα έναρξης των φορτηγών stt.

Για τον υπολογισμό της απόστασης αναμεσά σε δυο σημεία χρησιμοποιείται ο τύπος της ευκλείδειας απόστασης όπως αναφέρθηκε στο κεφάλαιο 2 (συνάρτηση 2,9). Επίσης θεωρούμε πως οι διαδρομές που ακολουθούν τα φορτηγά είναι ευθείες.

Για τις εργασίες φόρτωσης το φορτηγό πρέπει πρώτα να πάει στις συντεταγμένες της εργασίας να παραλάβει το φορτίο και έπειτα να κατευθυνθεί σε κάποια αποθήκη για να το αποθηκεύσει. Στις εργασίες εκφόρτωσης το φορτηγό πηγαίνει πρώτα σε μια αποθήκη να παραλάβει το φορείο και ελέγχει αν την χρονική στίμη που θα φτάσει στο σημείο εκφόρτωσης δεν έχει κλείσει το χρονικό παράθυρο της συγκεκριμένης εργασίας. Αν το παράθυρο έχει κλείσει το φορτηγό δεν παραλαμβάνει το φορτίο και πηγαίνει στην αμέσως επόμενη εργασία. Αν όμως το παράθυρο δεν έχει κλείσει το φορτηγό παραλαμβάνει το φορτίο και πηγαίνει να το παραδώσει στις συντεταγμένες εκφόρτωσης. Όταν μια εργασία δεν εκτελεστεί για ακριβώς αυτό τον λόγο τότε προστίθεται στην αντικειμενική συνάρτηση ένα πέναλτι που αυξάνει σημαντικά το κόστος της. Αυτό γίνεται για να αποφευχθεί η τελική λύση που βρήκαμε να περιέχει εργασίες οι οποίες δεν εκτελέστηκαν ποτέ. Ακόμη η επιλογή της αποθήκης για παραλαβή ή αποθήκευση γίνεται με τυχαίο τρόπο για να προσομοιωθούν οι συνθήκες που καλούνται τα λιμάνια να αντιμετωπίσουν καθώς υπάρχουν πολλές αλλαγές του τελευταίου λεπτού και τίποτα δεν μπορεί να είναι προγραμματισμένο προ πολλού.

Αξίζει επίσης να αναφέρουμε πως στο συγκεκριμένο πρόβλημα δεν λαμβάνουμε υπόψιν μας τον χρόνο φόρτωσης και εκφόρτωσης. Οπότε κάθε φορτηγό με το που φτάσει στον προορισμό εκτελεί την εργασία ακαριαία.

4.2 Περιγραφή αλγορίθμου

Πριν ξεκινήσουμε με την περιγραφή του αλγορίθμου ας αναφερθούμε ξανά στο πρόβλημα που καλούμαστε να λύσουμε. Ο τελικός σκοπός είναι να ελαχιστοποιήσουμε τον συνδυασμό της συνολικής καθυστέρησης των εργασιών και της συνολικής απόστασης που θα πρέπει να διανύσουν τα φορτηγά. Οπότε πρέπει να βρεθεί η σωστή μοιρασιά των εργασιών στα διαθέσιμα φορτηγά και την σωστή σειρά προτεραιότητας τον εργασιών. Οι εργασίες όπως αναφέραμε στο προηγούμενο κεφάλαιο είναι οκτώ και χωρίστηκαν ως τρεις στα πρώτα δυο φορτηγά και δυο στο τελευταίο.

Ο αλγόριθμος που κατασκευάστηκε χωρίζεται σε τρία κομμάτια. Στο πρώτο κομμάτι δημιουργούνται οι τυχαίες αρχικές λύσεις που χρειάζεται ο αλγόριθμος βελτιστοποίησης σμήνους σωματιδίων. Στο δεύτερο κομμάτι υπολογίζεται η συνάρτηση κόστους για το κάθε σωματίδιο. Στο τρίτο και τελευταίο κομμάτι υλοποιείται ο αλγόριθμος βελτιστοποίησης σμήνους σωματιδίων.

Ο αλγόριθμος βελτιστοποίησης σμήνους σωματιδίων δουλεύει για συνεχή προβλήματα, στην δικιά μας περίπτωση το πρόβλημα είναι διακριτό καθώς, οι εργασίες και οι αποθήκες είναι διακριτές. Επόμενος πρέπει να τροποποιήσουμε τα δεδομένα για να μπορέσει να λειτουργήσει ο αλγόριθμος. Η μετατροπή γίνεται μετατρέποντας την σειρά των εργασιών από διακριτές τιμές σε συνεχή και αφού γίνουν οι απαραίτητες μετατροπές του αλγορίθμου βελτιστοποίησης σμήνους σωματιδίων τις μετατρέπουμε ξανά σε διακριτές. Ας παραθέσουμε το παράδειγμα μιας λύσης για να μπορέσει να γίνει καλύτερα κατανοητή η μετατροπή. Έστω ότι μια λύση είναι η (1, 2, 3, 4, 5, 6, 7, 8). Η μετατροπή θα γίνει διαιρώντας κάθε όρο της λύσης με την μεγαλύτερη τιμή της, δηλαδή στην δικιά μας περίπτωση με το 8 και το νέο διάνυσμα θα έχει την μορφή (0.125, 0.25,

0.375, 0.5, 0.625, 0.75, 0.875, 1). Τώρα αφού τα δεδομένα μας έχουν συνεχή μορφή μπορούμε να εκτελέσουμε τον αλγόριθμο βελτιστοποίησης σμήνους σωματιδίων και ας πούμε πως δίνει σαν αποτέλεσμα (0.49, 0.541, 1.5, 0.06, 0.32, 1.1, 0.8, 1.17). Για να τα μετατρέψουμε ξανά σε διακριτές τιμές θα ξεκινήσουμε από την μεγαλύτερη συνεχή τιμή και θα της δώσουμε την μεγαλύτερη διακριτή τιμή. Έτσι η τιμή 1.5 θα πάρει την τιμή 8 και με αυτό τον τρόπο θα συμπληρωθεί και το υπόλοιπο διάνυσμα (3, 4, 8, 1, 2, 6, 5, 7).

4.2.1 Δημιουργία τυχαίων αρχικών λύσεων

Το πρώτο κομμάτι του κώδικα αποτελεί το αρχικό στάδιο της υλοποίησης του αλγορίθμου βελτιστοποίησης σμήνους σωματιδίων και έχει ως στόχο τη δημιουργία των σωματιδίων. Κάθε σωματίδιο αντιπροσωπεύει μια πιθανή λύση του προβλήματος, δηλαδή μια σειρά εκτέλεσης εργασιών. Για την παραγωγή των λύσεων ο κώδικας ανακατεύει τυχαία τη λίστα τον εργασιών. Πιο συγκεκριμένα για κάθε σωματίδιο πραγματοποιείται μια σειρά τυχαίων ανταλλαγών μεταξύ θέσεων στο διάνυσμα των εργασιών με σκοπό την δημιουργία διαφορετικής αλληλουχίας. Αφού ολοκληρωθεί η διαδικασία ανακατέματος, αποθηκεύεται σαν αρχική λύση για ένα σωματίδιο. Η διαδικασία επαναλαμβάνεται έως ότου να δημιουργηθούν αρχικές λύσεις για όλα τα σωματίδια.

Ακολουθεί ο ψευδοκώδικας για τη δημιουργία αρχικών λύσεων.

Ψευδοκώδικας 1

```
Μεταβλητές εισόδου: n, N, jobs
Μεταβλητές εξόδου: particles
Αρχή
    Για i από 0 μέχρι N
        Για u από 0 μέχρι n
            Δημιούργησε έναν τυχαίο αριθμό randnumb από το 0 μέχρι n-1
            Αντάλλαξε τις θέσεις u και randnumb στο διάνυσμα jobs
        Τέλος επανάληψης
        Για u από 0 μέχρι n
            Τοποθέτησε τη τιμή του job[u] στο διάνυσμα particles[i][u]
        Τέλος επανάληψης
    Τέλος επανάληψης
Τέλος
```

Όπου:

- N: πλήθος σωματιδίων
- jobs: σειρά εργασιών
- particles: δυσδιάστατο διάνυσμα σωματιδίων

4.2.2 Υπολογισμός συνάρτησης κόστους

Αφού έχουμε δημιουργήσει τις αρχικές λύσεις πρέπει να υπολογίσουμε το κόστος του κάθε σωματιδίου. Η συνάρτηση κόστους (2,1) όπως αναφέραμε σε προηγούμενο κεφάλαιο υπολογίζεται από την καθυστέρηση έναρξης των εργασιών και από τον συνολικό χρόνο που τα φορτηγά μεταφέρονται από το ένα σημείο στο άλλο. Αρά πρέπει να υπολογιστεί για το κάθε φορτηγό ξεχωριστά ποια χρονική στιγμή ξεκινάει την κάθε εργασία που του έχει ανατεθεί για να δούμε αν υπάρχει καθυστέρηση ,αλλά και το χρόνο που σπαταλάει στο δρόμο

πηγαίνοντας από το ένα σημείο στο άλλο. Ο κώδικας αρχικά αναθέτει στα φορτηγά ποιες εργασίες θα εξυπηρετήσουν. Όπως αναφέραμε πιο πάνω το πρώτο φορτηγό θα εξυπηρετήσει τις τρεις πρώτες, το δεύτερο τις επόμενες τρεις και το τρίτο τις δυο τελευταίες. Στη συνέχεια ξεκινάει ο υπολογισμός του συνολικού χρόνου καθυστέρησης και του χρόνου διαδρομής των φορτηγών. Αρχικά ελέγχουμε αν η δουλειά που είναι να εξυπηρετηθεί είναι εργασία φόρτωσης ή εργασία εκφόρτωσης. Αν η εργασία είναι φόρτωσης υπολογίζεται από τον τύπο της ευκλείδειας απόστασης (2,6) η απαιτούμενη απόσταση αναμεσά στην θέση του φορτηγού και στην θέσης της εργασίας. Για να βρεθεί ο χρόνος που το φορτηγό σπατάλησε στη διαδρομή πρέπει να διαιρέσουμε την απόσταση που βρήκαμε με την ταχύτητα ts του φορτηγού. Υστέρα ελέγχεται σε τι κατάσταση βρίσκεται το χρονικό παράθυρο της εργασίας. Αν δεν έχει ανοίξει το φορτηγό περιμένει μέχρι να ανοίξει και έτσι η καθυστέρηση της εργασίας είναι μηδέν. Αν έχει ανοίξει αλλά δεν έχει τελειώσει, η καθυστέρηση ορίζεται ως η διαφορά της έναρξης της εργασίας με την χρονική στιγμή που φτάνει το φορτηγό. Και αν έχει κλείσει, στην καθυστέρηση προσθέτετε ένα πέναλτι που την αυξάνει σημαντικά. Όταν η εργασία εκτελείται μέσα στο χρονικό παράθυρό της διαλέγεται τυχαία μια αποθήκη, που το φορτηγό πηγαίνει για να αποθηκεύσει το εμπορευματοκιβώτιο. Οι αποθήκες δεν έχουν χρονικό παράθυρο καθώς είναι πάντα διαθέσιμες αλλά υπολογίζουμε τον χρόνο διαδρομής από την εργασία στην αποθήκη και τον προστεθούμε στον συνολικό χρόνο διαδρομής. Από την άλλη αν η εργασία είναι εκφόρτωσης διαλέγεται μια τυχαία αποθήκη για να φορτώσει το φορτηγό ένα εμπορευματοκιβώτιο και υπολογίζεται η απόσταση της με την θέση του φορτηγού. Σε αυτή την περίπτωση ελέγχουμε πρώτα αν το φορτηγό τη στιγμή που θα φτάσει στην εργασία εκφόρτωσης δεν έχει κλείσει το χρονικό παράθυρό της. Αυτό γίνεται για να μην φορτωθούμε το εμπορευματοκιβώτιο και επειδή η εργασία έχει κλείσει το φορτηγό θα πρέπει να το γυρίσει πίσω για να το αφήσει για να πάει στην επόμενη εργασία. Όπως ακριβώς και στις εργασίες φόρτωσης, αν το χρονικό παράθυρο δεν έχει ανοίξει η καθυστέρηση είναι μηδέν, αν έχει ανοίξει είναι η διαφορά έναρξης με την χρονική στιγμή που φτάνει το φορτηγό και αν έχει κλείσει προσθέτουμε το πέναλτι. Όταν το χρονικό παράθυρο δεν έχει

κλείσει προσθέτουμε στον συνολικό χρόνο διαδρομής τον χρόνο διαδρομής από την αποθήκη στην εργασία εκφόρτωσης. Αφού η διαδικασία έχει επαναληφθεί για όλα τα φορτηγά προσθέτουμε την συνολική καθυστέρηση των εργασιών και τον συνολικό χρόνο διαδρομής για όλα τα φορτηγά και με τον τύπο (2.1) βγαίνει το συνολικό κόστος του σωματιδίου. Η διαδικασία επαναλαμβάνεται για όλο το πλήθος των σωματιδίων

Ακολουθεί ο ψευδοκώδικας για τον υπολογισμό του κόστους.

Ψευδοκώδικας 2

```
Μεταβλητές εισόδου: n, N, esj, ltw, a1, a2, stt, stl, sc, pujl, dujl
Μεταβλητές εξόδου:  fitness
Αρχή
    Για i από 0 μέχρι N
        Αρχικοποίηση θέσης φορτηγών, χρόνου φορτηγών,
        συνολική καθυστέρησή, συνολικό χρόνο διάδρομων
        Για u από 0 έως n
            Εάν u < 3
                Χρήση πρώτου φορτηγού
            Εάν u < 6
                Χρήση δευτέρου φορτηγού
            Αλλιώς
                Χρήση τρίτου φορτηγού
            Εάν η εργασία n είναι φόρτωσης
                Υπολογισμός απόστασης αναμεσά στην εργασία και το φορτηγό
                Υπολογισμός χρόνου διαδρομής και πρόσθεση στον συνολικό χρόνο διαδρομής
                Εάν το χρονικό παράθυρο έχει ανοίξει και δεν έχει κλείσει
                    Καθυστέρηση ← Άνω ορός χρονικού παραθύρου πλην ώρα άφιξης φορτηγού
                    Συνολική καθυστέρησή ← συνολική καθυστέρηση συν καθυστέρησή
                    Υπολογισμός απόστασης αναμεσά στην εργασία και μια τυχαία αποθήκη
                    Υπολογισμός χρόνου διαδρομής και πρόσθεση στον συνολικό χρόνο διαδρομής
                Εάν το χρονικό παράθυρο δεν έχει ανοίξει
                    Καθυστέρηση ← 0
                    Υπολογισμός απόστασης αναμεσά στην εργασία και μια τυχαία αποθήκη
```

Υπολογισμός χρόνου διαδρομής και πρόσθεση στον συνολικό χρόνο διαδρομής
Εάν το χρονικό παράθυρο έχει κλείσει
Συνολική καθυστέρησή ← συνολική καθυστέρησή συν πέναλτι
Εάν η εργασία n είναι εκφόρτωσης
Υπολογισμός απόστασης αναμεσά στο φορτηγό και μια τυχαία αποθήκη
Υπολογισμός χρόνου διαδρομής και πρόσθεση στον συνολικό χρόνο διαδρομής
Υπολογισμός απόστασης αναμεσά στην αποθήκη και την εργασία
Υπολογισμός χρόνου διαδρομής
Εάν το χρονικό παράθυρο έχει ανοίξει αλλά δεν έχει κλείσει
Συνολικός χρόνος διαδρομής ← χρόνος διαδρομής
Καθυστέρηση ← Άνω ορός χρονικού παραθύρου πλην ώρα άφιξης φορτηγού
Συνολική καθυστέρησή ← συνολική καθυστέρηση συν καθυστέρησή
Εάν το χρονικό παράθυρο δεν έχει ανοίξει
Συνολικός χρόνος διαδρομής ← χρόνος διαδρομής
Καθυστέρηση ← 0
Εάν το χρονικό παράθυρο έχει κλείσει
Συνολική καθυστέρησή ← συνολική καθυστέρησή συν πέναλτι
Τέλος επανάληψης
fitness ← a_1 × συνολική καθυστέρηση + a_2 × συνολικός χρόνος διαδρομής
Τέλος επανάληψης
Τέλος

4.2.3 Υλοποίηση αλγορίθμου βελτιστοποίησης σμήνους σωματιδίων

Έχοντας υπολογίσει τα κόστη των λύσεων μπορούμε να υλοποιήσουμε βελτιστοποίηση σμήνους σωματιδίων. Στην πρώτη επανάληψη του κώδικα θα πρέπει να ορίσουμε τα pbest των σωματιδίων και το gbest. Ως pbest ορίζονται οι αρχικές λύσεις των σωματιδίων και η λύση με το μικρότερο κόστος ορίζεται ως η gbest. Στις επόμενες επαναλήψεις συγκρίνονται τα νέα κόστη με τα κόστη των pbest και gbest και αν τα νέα είναι καλύτερα από αυτά τότε παίρνουν την θέση τους. Έπειτα για να μπορέσει να δουλέψει ο αλγόριθμος βελτιστοποίησης όπως αναφέραμε σε αυτό το κεφάλαιο, πρέπει να μετατρέπουμε τα διανύσματα

από διακριτά σε συνεχή. Θα διαιρέσουμε λοιπόν τα pbest, gbest και τα σωματίδια με τον αριθμό των σωματιδίων. Τώρα μπορούμε να χρησιμοποιήσουμε τις εξισώσεις (3,1) και (3,2) για να μας μετακινήσουν τα σωματίδια στο χώρο των λύσεων. Και τέλος μετατρέπουμε ξανά τα σωματίδια και τα pbest και gbest από συνέχει σε διακριτά.

Ακολουθεί ο ψευδοκώδικας της υλοποίησης του αλγορίθμου βελτιστοποίησης σμήνους σωματιδίων.

Ψευδοκώδικας 3

```
Μεταβλητές εισόδου: n, N, particles, fitness,
Μεταβλητές εξόδου: gbest, gbest_fitness
Αρχή
    Για i από 0 έως N
        Διαιρούμε τα σωματίδια με τον αριθμό των σωματιδίων n
        Διαιρούμε τα pbest με τον αριθμό των σωματιδίων n
    Τέλος επανάληψης
    Για i από 0 έως n
        Διαιρούμε το gbest με τον αριθμό των σωματιδίων n
    Τέλος επανάληψης
    Για i από 0 έως N
        Υπολογίζεται η νέα ταχύτητα με τον τύπο (3,1)
        Υπολογίζονται οι μετακινήσεις των σωματιδίων με τον τύπο (3,2)
    Τέλος επανάληψης
    Για i από 0 έως n
        Μετατροπή gbest από διακριτό σε συνεχή
    Τέλος επανάληψης
    Για i από 0 έως  N
        Μετατροπή σωματιδίων από διακριτά σε συνεχή
        Μετατροπή pbest από διακριτά σε συνεχή
        Εάν το νέο κόστος του σωματιδίου i είναι μικρότερο από το κόστος της pbest i
            Ανανέωσε την pbest και το κόστος της με το συγκεκριμένο σωματίδιο
```

Εάν το νέο κόστος του σωματιδίου i είναι μικρότερο από το κόστος της gbest
Ανανέωσε την gbest και το κόστος της με το συγκεκριμένο σωματίδιο
Τέλος επανάληψης
Τέλος

Κεφάλαιο 5

5.1 Επιλογή κατάλληλου αριθμού σωματιδίων και επαναλήψεων

Για την σωστή υλοποίηση του αλγορίθμου είναι απαραίτητο να ορίσουμε σωστά τον αριθμό των σωματιδίων και τον αριθμό των επαναλήψεων. Για μικρές τιμές και των δυο παραμέτρων ο αλγόριθμος μπορεί να μην προλάβει να προσεγγίσει ικανοποιητικά το ολικό βέλτιστο. Ενώ για μεγάλες τιμές αυξάνεται η απαιτούμενη υπολογιστική ισχύς χωρίς απαραίτητα να αυξάνεται και η ποιότητα της λύσης. Για να εξετάσουμε ποιες τιμές μας δίνουν τα καλύτερα αποτελέσματα θα χρησιμοποιήσουμε την μέθοδο του πειράματος, σφάλματος. Στην ουσία θα τρέξουμε τον αλγόριθμο και σε κάθε καινούργια επανάληψη του, θα αλλάζουμε μόνο την υπό εξέταση μεταβλητή και θα συγκρίνουμε τα μεταξύ τους αποτελέσματα. Για να βρούμε τις κατάλληλες παραμέτρους θα χρησιμοποιήσουμε τα δεδομένα του δευτέρου προβλήματος. Τα αποτελέσματα φαίνονται στον παρακάτω πίνακα:

	Επαναλήψεις →										
Σμήνος ↓	**1**	**10**	**20**	**30**	**40**	**50**	**60**	**70**	**80**	**90**	**100**
1	12446	603	518	582	574	605	690	384	602	464	555
10	6828	471	465	594	445	460	372	403	488	423	444
20	764	512	488	419	533	380	422	352	390	451	362
30	714	509	408	370	378	397	401	462	354	347	370
40	727	422	559	338	378	339	391	420	365	396	312

50	789	541	390	407	374	363	399	356	346	365	496
60	692	392	359	441	337	406	423	344	350	331	391
70	771	460	370	361	365	335	342	368	354	401	435
80	661	559	400	343	368	346	354	321	395	382	356
90	709	369	363	379	360	376	396	314	310	351	332
100	698	376	422	382	413	327	335	332	293	335	340

Όπως μπορούμε να δούμε από τον πίνακα όσο αναβαίνον οι επαναλήψεις, το κόστος μειώνεται. Την πιο έντονη μείωση την βλέπουμε στις 60 επαναλήψεις. Από εκεί και πέρα δεν παρατηρούμε σημαντικές διάφορες. Για το μέγεθος του σμήνους παρατηρούμε πως έχουμε έντονη μείωση μέχρι την τιμή 30. Από την τιμή 30 μέχρι και την 100 δεν παρατηρούμε κάποια ιδιαίτερη βελτίωση. Επομένως για την λύση του προβλήματος θα χρησιμοποιηθούν 60 επαναλήψεις με μέγεθος σμήνους ίσο με 30.

5.2 Παρουσίαση αποτελεσμάτων

Αφού βρήκαμε τον καλύτερο συνδυασμό επαναλήψεων και μεγέθους του σμήνους θα παρουσιαστεί η λύση του προβλήματος αρχικά για τα πρώτα δεδομένα και έπειτα για του δευτέρου. Οι παράμετροί που χρησιμοποιήθηκαν είναι:

Μέγεθος σμήνους	30
Επαναλήψεις	60
a_1	0.6
a_2	0.4
c_1	2
c_2	2

Πρώτο πρόβλημα

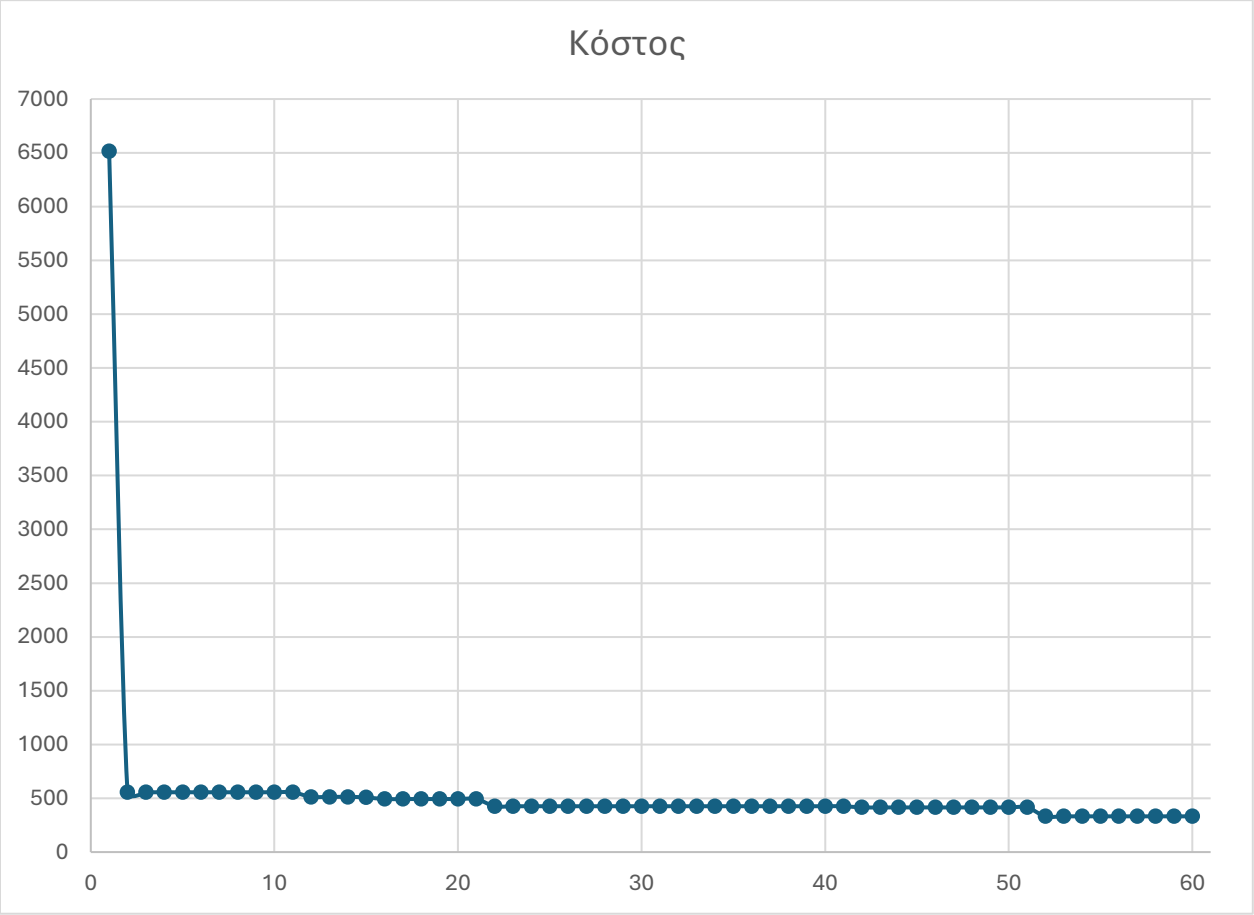
Στον παρακάτω πίνακα παρουσιάζεται η gbest για κάθε επανάληψη του αλγορίθμου με τα δεδομένα του πρώτου προβλήματος.

Επανάληψη	Διαδρομή	Κόστος
1η	3 8 2 6 4 1 5 7	6514
2η	5 8 7 3 6 1 2 4	556
3η	5 8 7 3 6 1 2 4	556
4η	5 8 7 3 6 1 2 4	556
5η	5 8 7 3 6 1 2 4	556
6η	5 8 7 3 6 1 2 4	556
7η	5 8 7 3 6 1 2 4	556
8η	5 8 7 3 6 1 2 4	556
9η	5 8 7 3 6 1 2 4	556
10η	5 8 7 3 6 1 2 4	556
11η	5 8 7 3 6 1 2 4	556
12η	3 6 8 5 4 1 2 7	513
13η	3 6 8 5 4 1 2 7	513
14η	3 6 8 5 4 1 2 7	513
15η	5 8 7 3 4 1 2 6	508
16η	2 6 8 3 5 1 4 7	493
17η	2 6 8 3 5 1 4 7	493
18η	2 6 8 3 5 1 4 7	493
19η	2 6 8 3 5 1 4 7	493
20η	2 6 8 3 5 1 4 7	493
21η	2 6 8 3 5 1 4 7	493
22η	3 5 7 2 4 1 6 8	426
23η	3 5 7 2 4 1 6 8	426
24η	3 5 7 2 4 1 6 8	426
25η	3 5 7 2 4 1 6 8	426
26η	3 5 7 2 4 1 6 8	426
27η	3 5 7 2 4 1 6 8	426
28η	3 5 7 2 4 1 6 8	426
29η	3 5 7 2 4 1 6 8	426
30η	3 5 7 2 4 1 6 8	426
31η	3 5 7 2 4 1 6 8	426
32η	3 5 7 2 4 1 6 8	426
33η	3 5 7 2 4 1 6 8	426

34η	3 5 7 2 4 1 6 8	426
35η	3 5 7 2 4 1 6 8	426
36η	3 5 7 2 4 1 6 8	426
37η	3 5 7 2 4 1 6 8	426
38η	3 5 7 2 4 1 6 8	426
39η	3 5 7 2 4 1 6 8	426
40η	3 5 7 2 4 1 6 8	426
41η	3 5 7 2 4 1 6 8	426
42η	2 6 8 3 5 1 4 7	415
43η	2 6 8 3 5 1 4 7	415
44η	2 6 8 3 5 1 4 7	415
45η	2 6 8 3 5 1 4 7	415
46η	2 6 8 3 5 1 4 7	415
47η	2 6 8 3 5 1 4 7	415
48η	2 6 8 3 5 1 4 7	415
49η	2 6 8 3 5 1 4 7	415
50η	2 6 8 3 5 1 4 7	415
51η	2 6 8 3 5 1 4 7	415
52η	5 6 8 3 2 1 4 7	332
53η	5 6 8 3 2 1 4 7	332
54η	5 6 8 3 2 1 4 7	332
55η	5 6 8 3 2 1 4 7	332
56η	5 6 8 3 2 1 4 7	332
57η	5 6 8 3 2 1 4 7	332
58η	5 6 8 3 2 1 4 7	332
59η	5 6 8 3 2 1 4 7	332
60η	5 6 8 3 2 1 4 7	332

Η λύση που έβγαλε ο αλγόριθμος είναι το πρώτο φορτηγό να εκτελέσει τις εργασίες 5 6 8, το δεύτερο τις 3 2 1 και το τρίτο τις 4 7. Το συνολικό κόστος ανέρχεται στις 332 μονάδες που είναι μια πολύ ικανοποιητική τιμή.

Στο επόμενο σχεδιάγραμμά φαίνεται πως ο αλγόριθμος μειώνει το κόστος για κάθε επανάληψη.



Δεύτερο πρόβλημα

Στον παρακάτω πίνακα παρουσιάζεται η gbest για κάθε επανάληψη του αλγορίθμου με τα δεδομένα του δευτέρου προβλήματος.

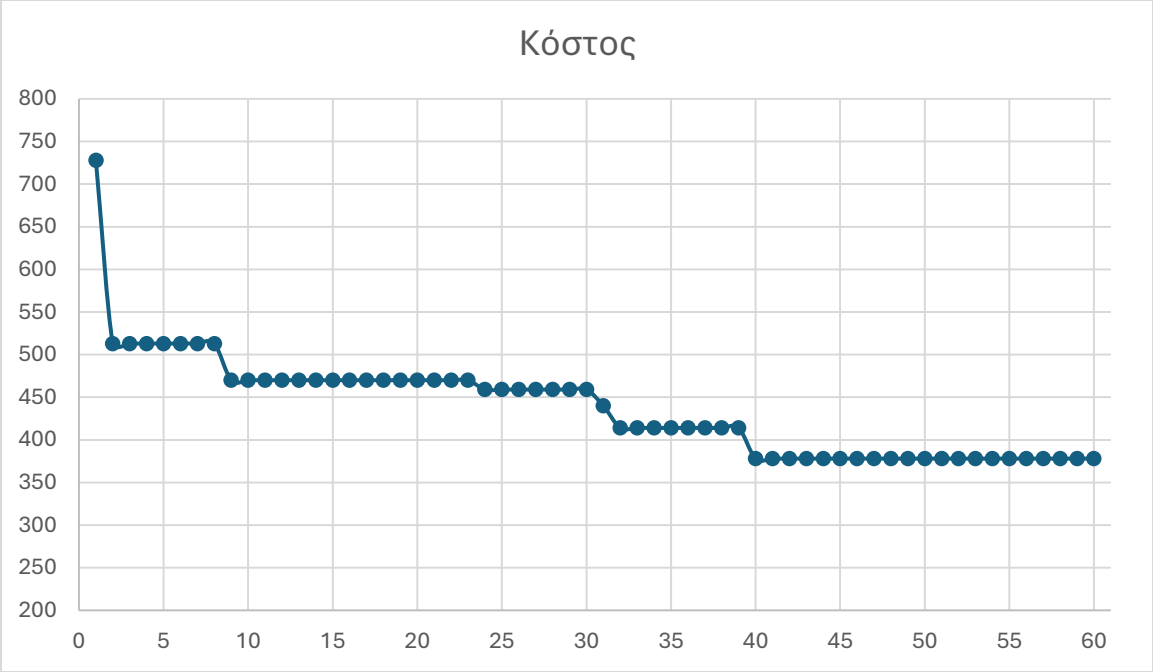
Επανάληψη	Διαδρομή	Κόστος
1η	3 2 1 4 8 5 6 7	728
2η	3 2 5 8 4 1 6 7	513
3η	3 2 5 8 4 1 6 7	513
4η	3 2 5 8 4 1 6 7	513
5η	3 2 5 8 4 1 6 7	513
6η	3 2 5 8 4 1 6 7	513

7ⁿ	3 2 5 8 4 1 6 7	513
8ⁿ	3 2 5 8 4 1 6 7	513
9ⁿ	4 2 7 8 3 1 6 5	470
10ⁿ	4 2 7 8 3 1 6 5	470
11ⁿ	4 2 7 8 3 1 6 5	470
12ⁿ	4 2 7 8 3 1 6 5	470
13ⁿ	4 2 7 8 3 1 6 5	470
14ⁿ	4 2 7 8 3 1 6 5	470
15ⁿ	4 2 7 8 3 1 6 5	470
16ⁿ	4 2 7 8 3 1 6 5	470
17ⁿ	4 2 7 8 3 1 6 5	470
18ⁿ	4 2 7 8 3 1 6 5	470
19ⁿ	4 2 7 8 3 1 6 5	470
20ⁿ	4 2 7 8 3 1 6 5	470
21ⁿ	4 2 7 8 3 1 6 5	470
22ⁿ	4 2 7 8 3 1 6 5	470
23ⁿ	4 2 7 8 3 1 6 5	470
24ⁿ	2 1 5 8 6 7 4 3	459
25ⁿ	2 1 5 8 6 7 4 3	459
26ⁿ	2 1 5 8 6 7 4 3	459
27ⁿ	2 1 5 8 6 7 4 3	459
28ⁿ	2 1 5 8 6 7 4 3	459
29ⁿ	2 1 5 8 6 7 4 3	459
30ⁿ	2 1 5 8 6 7 4 3	459
31ⁿ	4 2 7 8 3 1 6 5	440
32ⁿ	3 1 7 8 4 2 6 5	414
33ⁿ	3 1 7 8 4 2 6 5	414
34ⁿ	3 1 7 8 4 2 6 5	414
35ⁿ	3 1 7 8 4 2 6 5	414
36ⁿ	3 1 7 8 4 2 6 5	414
37ⁿ	3 1 7 8 4 2 6 5	414
38ⁿ	3 1 7 8 4 2 6 5	414
39ⁿ	3 1 7 8 4 2 6 5	414
40ⁿ	2 1 7 8 4 5 3 6	378
41ⁿ	2 1 7 8 4 5 3 6	378
42ⁿ	2 1 7 8 4 5 3 6	378

43η	2 1 7 8 4 5 3 6	378
44η	2 1 7 8 4 5 3 6	378
45η	2 1 7 8 4 5 3 6	378
46η	2 1 7 8 4 5 3 6	378
47η	2 1 7 8 4 5 3 6	378
48η	2 1 7 8 4 5 3 6	378
49η	2 1 7 8 4 5 3 6	378
50η	2 1 7 8 4 5 3 6	378
51η	2 1 7 8 4 5 3 6	378
52η	2 1 7 8 4 5 3 6	378
53η	2 1 7 8 4 5 3 6	378
54η	2 1 7 8 4 5 3 6	378
55η	2 1 7 8 4 5 3 6	378
56η	2 1 7 8 4 5 3 6	378
57η	2 1 7 8 4 5 3 6	378
58η	2 1 7 8 4 5 3 6	378
59η	2 1 7 8 4 5 3 6	378
60η	2 1 7 8 4 5 3 6	378

Η λύση που έβγαλε ο αλγόριθμος είναι το πρώτο φορτηγό να εκτελέσει τις εργασίες 2 1 7, το δεύτερο τις 8 4 5 και το τρίτο τις 3 6. Το συνολικό κόστος ανέρχεται στις 378 μονάδες που είναι μια πολύ ικανοποιητική τιμή.

Στο επόμενο σχεδιάγραμμά φαίνεται πως ο αλγόριθμος μειώνει το κόστος για κάθε επανάληψη.



5.3 Συμπεράσματα

Σε αυτή την διπλωματική εργασία κληθήκαμε να λύσουμε το πρόβλημα χρονοπρογραμματισμού φορτηγών για την αποβάθρα εμπορευματοκιβώτιων με την χρήση αλγορίθμου βελτιστοποίησης σμήνους σωματιδίων. Από τα αποτελέσματα συμπεραίνουμε πως ο αλγόριθμος βελτιστοποίησης σμήνους σωματιδίων ανταπεξέρχεται πολύ καλά στο συγκεκριμένο πρόβλημα δίνοντας ικανοποιητικές απαντήσεις σε πολύ μικρό χρονικό διάστημα.

Βιβλιογραφία

[1] Ιωάννης Μαρινάκης, Αθανάσιος Μυγδαλάς, « Σχεδιασμός και Βελτιστοποίηση της Εφοδιαστικής Αλυσίδας », Εκδόσεις ‘’σοφία’’, Θεσσαλονίκη 2008

[2] Ben Niu, Fangfang Zhang, Li Li, Lang Wu, Particle Swarm Optimization for Yard Truck Scheduling in Container Terminal with a Cooperative Strategy, 2017

[3] Ben Niu, Ting Xie, Lijing Tan, Ying Bi, Zhengxu Wang, Swarm intelligence algorithms for Yard Truck Scheduling and Storage Allocation Problems, 2016

[4] Ali Skaf, Sid Lamrous, Zakaria Hammoudan, Marie-Ange Manier, Integrated quay crane and yard truck scheduling problem at port of Tripoli-Lebanon, 2021

[5] Ιορδανίδου Γεωργία-Ρουμπίνη, «Επίλυση προβλήματος δρομολόγησης οχημάτων με στοχαστική ζήτηση με χρήση του Αλγορίθμου Βελτιστοποίησης Σμήνους Σωματιδίων (Particle Swarm Optimization-PSO), 2011