ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΑΡΑΓΩΓΗΣ ΚΑΙ ΔΙΟΙΚΗΣΗΣ

ΖΑΧΑΡΙΟΥΔΑΚΗΣ ΔΙΟΝΥΣΙΟΣ

Α.Μ:2010010079

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΕΠΙΛΥΣΗ ΤΟΥ ΠΡΟΒΛΗΜΑΤΟΣ ΧΡΟΝΟΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ ΣΥΝΕΧΟΥΣ ΡΟΗΣ ΜΕ ΤΗΝ ΧΡΗΣΗ ΤΟΥ ΑΛΓΟΡΙΘΜΟΥ ΒΕΛΤΙΣΤΟΠΟΙΗΣΗΣ ΣΜΗΝΟΥΣ ΠΥΓΟΛΑΜΠΙΔΩΝ

ΕΠΙΒΛΕΠΩΝ:ΙΩΑΝΝΗΣ ΜΑΡΙΝΑΚΗΣ

ΧΑΝΙΑ 2016

ΕΥΧΑΡΙΣΤΙΕΣ

Με την παρούσα διπλωματική εργασία ολοκληρώνονται οι σπουδές μου σε προπτυχιακό σε προπτυχιακό επίπεδο στον Πολυτεχνείο Κρήτης και ως εκ τούτου θα ήθελα να ευχαριστήσω την οικογένεια μου για την βοήθεια και την στήριξη, ώστε να καταφέρω να κλείσω αυτόν τον κύκλο σπουδών.

Επιπλέον θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή της σχολής ΜΠΔ, κύριο Ιωάννη Μαρινάκη, τόσο για τις συμβουλές και την πολύτιμη βοήθεια του για την ολοκλήρωση της παρούσας διπλωματικής, όσο και για την άψογη συνεργασία που είχαμε.

Contents

ΠΕΡΙΛΗΨΗ

Η παρούσα διπλωματική εργασία πραγματεύεται το πρόβλημα του χρονικού προγραμματισμού εργασιών παραγωγής συνεχούς ροής. Στόχος προβλημάτων χρονικού προγραμματισμού είναι η ελαχιστοποίηση του χρόνου εκπόνησης διάφορων εργασιών σε συγκεκριμένες μηχανές. Σε αυτήν την εργασία χρησιμοποιήθηκε ο αλγόριθμος βελτιστοποίησης σμήνους πυγολαμπίδων(Glowworm Swarm Based Optimization Algorithm), εμπνευσμένος από φυσικές διαδικασίες . Χρησιμοποώντας αυτόν τον αλγόριθμο και τοπική αναζήτηση στο τέλος, προσπαθήσαμε να βρούμε την σειρά εργασιών που θα μας δίνει τον καλύτερο δυνατό χρόνο φτάνοντας όσο πιο κοντά μπορούμε στα παγκόσμια βέλτιστα.

1η ΕΝΟΤΗΤΑ-ΘΕΩΡΗΤΙΚΟ ΜΕΡΟΣ

1.1:Εισαγωγή στην εφοδιαστική αλυσίδα

Οι στόχοι και το περιεχόμενο της εφοδιαστικής αλυσίδας συνοψίζονται σε μία ρήση: Να παραδωθεί η σωστή ποσότητα του σωστού προϊόντος στον σωστό πελάτη στο σωστό τόπο και χρόνο σε σωστή στιγμή.

1.1.1: Ιστορική εξέλιξη

Η χρήση τη εφοδιαστικής(Logistics), έχοντας έννοια παρόμοια με αυτή που της αποδίδουμε σήμερα, ξεκινά από την εποχή των ρωμαϊκών χρόνων, που ήταν αναγκαία για την ύπαρξη οδών, για την επιτυχή και ταχεία μεταφορά διαταγών, υλικών ανδρών, σε διάφορα σημεία των αυτοκρατορικών συνόρων, ώστε να αντιμετωπιστούν αποτελεσματικά οι βαρβαρικές επιδρομές. Επίσης, ήταν αναγκαία η καλή διαχείρηση των αποθηκών, των αποθεμάτων, των μεταφορών, κ.ά, από τότε που οι άνθρωποι ξεκίνησαν να ασχολούνται ουσιαστικά με το εμπόριο. Στην συνέχεια, κατά την διάρκεια του δευτέρου παγκοσμίου πολέμου μέχρι και πρόσφατα στον πόλεμο του κόλπου και την εισβολή στο Ιράκ, αναπτύχθηκε η στατριωτική εφοδιαστική(military logistics). Ακόμη, χρησιμοποιούνταν στον πολιτικό τομέα σε επίπεδο διακυβέρνησης αλλά και σε ιδιωτικό επιχειρησιακό επίπεδο. Από τα τέλη του 1950 και τις αρχές του 1960, προσεγγίζεται και κρίνεται αναγκαία η ολοκληρωμένη εφοδιαστική διαχείρηση, που φτάνει μέχρι σήμερα. Η ολοκληρωμένη εφοδιαστική διαχείρηση, έχει ως στόχο την επίλυση πιθανόν συγκρούσεων μεταξύ των στόχων διαφόρων τομέων μέσα στην ίδια την επιχείρηση, ώστε να είναι ικανοποιητική και αποτελεσματική για την επιχείρηση σαν σύνολο. Αυτό απαιτεί την ανάλυση του ολικού εφοδιαστικού κόστους(total cost analysis), το οποίο θα περιέχει τις δαπάνες που σχετίζονται με τους στόχους από την αρχική φάση της έρευνας της ιδέας ενός προϊόντος μέχρι το τέλος της χρήσιμης ζωής του, το οποίο αυτό διάστημα ονομάζεται κόστος κύκλου ζωής (life cycle cost). Σήμερα, η ανάπτυξη ποσοτικών μεθόδων της επιχειρησιακής έρευνας, η πρόοδος της τεχνολογίας των υπολογιστών, η ανεπτυγμένη τεχνολογία επικοινωνιών και της πληροφορικής, επιτρέπουν την χρήση της εφοδιαστικής. Μίας εφοδιαστικής, η οποία κρίνεται απαραίτητη για την ισορροπία μεταξύ απόδοσης και κόστους ώστε να βελτιώνονται οι στόχοι της επιχείρησης.[1]

Άλλοι όροι όπως:[1]

- επιχειρηματική εφοδιαστική(business logistics)
- διαχείρηση καναλιών διανομής(channel distribution management)
- διαχείρηση διανομών(distribution management)
- διαχείρηση υλικού(materials management)
- φυσική διανομή(physical distribution)
- συστήματα ταχείας απόκρισης(quick-response systems)
- διαχείρηση εφοδιαστικής αλυσίδας(supply chain management)

όλοι αναφέρονται στην εφοδιαστική.

1.1.2: Ο όρος εφοδιαστική αλυσίδα

Όσο περνάνε τα χρόνια τόσο περισσότερο οδηγούμαστε προς την παγκοσμιοποίηση και οι πελάτες έχουν όλο και μεγαλύτερες απαιτήσεις. Έτσι, η δημιουργία μίας εφοδιαστικής αλυσίδας, από κάθε επιχείρηση, ικανή να καλύπτει τις ανάγκες των πελατών και να εξασφαλίζει αύξηση των μεριδιών της αγοράς και της κερδοφορίας είναι αναγκαία.

Με τον όρο **εφοδιαστική αλυσίδα** δεν εννούμε μόνο την ροή των υλικών από τον προμηθευτή πρώτων υλών ή τον κατασκευαστή μέχρι τον τελικό καταναλωτή, αλλά και την ροή πληροφοριών μεταξύ των μελών της ίδιας αλυσίδας. Η διαχείρησή της γίνεται σε δύο στάδια:[1]

- **Στάδιο προγραμματισμού:** στο στάδιο αυτό αναλύονται όλα τα δεδομένα που αφορούν τις προμήθειες, την αποθεματοποίση, τις πωλήσεις, γίνονται προβλέψεις και ο σχεδιασμός του πλάνου δράσης στην συνέχεια.
- **Στάδιο εκτέλεσης:** στο στάδιο αυτό εκτελείται το πλάνο που είχε σχεδιαστεί κατά τον προγραμματισμό και παρακολουθείται η εξέλιξή του.

Για την αντιμετώμιση των προβλημάτων εφοδιαστικής αλυσίδας προϋποθέτει την χρήση της σύγχρονης τεχνολογίας της πληρφορικής, των επικοινωνιών κ.ά και απαιτεί για την λήψη αποφάσεων λαμβάνοντας υπόψιν τις απαιτούμενες προϋποθέσεις, το ανθρώπινο δυναμικό, την ύπαρξη και την οργάνωση του πληροφιορικού συστήματος. Επομένως, για την επίλυση των

προβλημάτων ακολουθείται μία ολοκληρωμένη μεθοδολογία εφοδιαστικής αλυσίδας, η οποία αποτελείται από τα εξής βήματα:

1. Εντοπισμός του προβλήματος, ανάλυση των παραγόντων που το επηρεάζουν, των περιορισμών του, προσδιορισμος των στόχων και του πλάνου.
2. Μαθηματική (γραμμικός ή ακεραιος προγραμματισμός) ή συστημική (προσομοίωση ή μοντελοποιήση επιχειρησιακών διαδικασιών) προσέγγιση του προβλήματος για την απεικόνισή του.
3. Ανάπτυξη τεχνικών επίλυσης του προβλήματος μέσω μαθηματικού προγραμματισμού, ευρετικών ή γενετικών αλγορίθμων κλπ.
4. Υλοποίηση των παραπάνω τεχνικών σε κατάλληλη πλατφόρμα ανάλογα με τον εξοπλισμό που διαθέτει η κάθε εταιρεία.
5. Σχεδιασμός και σύνθεση των υποστηρικτικών συστημάτων που επιτρέπουν την διεπαφή των μάνατζερ και των συστημάτων που περιέχουν τα δεδομένα, ώστε να λειφθούν οι αποφάσεις.

Μια εφοδιαστική αλυσίδα αποτελείται από κατασκευαστές, προμηθευτές, χώρους αποθήκευσης, κεντρα διανομών, μεταφορείς, πωλητές λιανικής, πελάτες, τις πρώτες ύλες και τα αποθέματα κατά την διαδικασία της παραγωγής. Κάθε ένα απ' αυτά είναι και ένα στάδιο της εφοδιαστικής αλυσίδας, που εκτελεί διάφορες διαδικασίες και αλληλεπιδρά με τα υπόλοιπα.

Επιπλέον, η εφοδιαστική αλυσίδα αποτελείται από διάφορες δραστηριότητες, μερικές εκ των οποίων είναι οι εξής:[1]

- Συσκευασία
- Τροφοδοσία
- Διαχείρηση υλικού
- Διαχείρηση αποθηκών και κέντρων διανομών
- Πρόβλεψη ζήτησης
- Χωροθέτηση εγκαταστάσεων
- Μεταφορές και διανομές

1.2.3: Ανάλυση ολικού κόστους

Οι εφοδιαστικές δαπάνες δημιουργούνται σε όλη την διάρκεια της εφοδιαστικής διαδικασίας, οι οποίες είναι οι δαπάνες που προέρχονται από την εξυπηρέτηση του πελάτη, από τις μεταφορές και τις διανομές, από την

εκπαίδευση του προσωπικού, από την διαχείριση των εγκαταστάσεων, την επεξεργασία των παραγγελιών, των πληροφοριών, των επικοινωνιών και την διαχείριση των αποθεμάτων.

Παρακάτω θα αναφερθούμε συνοπτικά στις κυριότερες από αυτές τις δαπάνες:[1]

1. **Κόστος από την εξυπηρέτηση του πελάτη**: Αφορά τα έξοδα υποστήριξης και εξυπηρέτησης των πελατών. Τα έξοδα για ανταλλακτικά και επιδιορθώσεις, την διαχείριση επιστρεφόμενων προϊόντων καθώς και τους ανικανοποίητους πελάτες που οδηγούν σε κόστος απώλειας πωλήσεων.
2. **Κόστος μεταφορών και διανομών**: Είναι το κόστος που προκύπτει από την διανομή του προϊόντος στον πελάτη, τις μεταφορές των πρώτων υλών, την μεταφορά στις αποθήκες, κ.ά. Επηρεάζεται από τις αποστάσεις, το βάρος και τον όγκο των προϊόντων, το είδος του οχήματος κ.ά.
3. **Κόστος διαχείρησης αποθηκών και κέντρων διανομών**: Είναι ένα κόστος που προκύπτει από τις δραστηριότητες σ' αυτές τις εγκαταστάσεις, τον τόπο χωροθέτησής τους, τον αριθμό τους κ.ά.
4. **Κόστος διαχείρησης παραγγελιών και πληροφοριών**: Αποτελείται από τα έξοδα για την επεξεργασία των παραγγελιών, την πρόβλεψη της ζήτησης, την υιοθέτηση και την λειτουργία τεχνολογίας πληροφορικής και επικοινωνιών, κ.ά.
5. **Κόστος αποθεμάτων**: Αφορά τον ελέγχο, την διαχείρηση των αποθεμάτων και την συσκευασία. Ακόμη αφορά τα μεταβαλλόμενα έξοδα, όπως το κόστος κεφαλαίου που αντιστοιχεί στην επιστροφή κεφαλαίου εάν η επιχείρηση δεν το είχε δεσμεύσει στ' αποθέματα, το κόστος υπηρεσιών αποθέματος, που είναι τα έξοδα ασφάλισης και φορολόγησης, το κόστος χώρου αποθήκευσης και το κόστος αποθεματικού κινδύνου, που αναφέρεται στον κίνδυνο αχρηστίας, ζημιάς ή καταστροφής του αποθέματος ή ακόμα και κλοπής.
6. **Κόστος παρτίδων**: Σχετίζεται με τα έξοδα παραγωγής και αγοράς των πρώτων υλών, τα οποία μεταβάλλονται ανάλογα με το μέγεθος και την συχνότητα της παραγγελίας. Τα έξοδα αυτά περιλαμβάνουν το κόστος προετοιμασίας, το κόστος από την διαχείρηση υλικού, το κόστος διαφοροποίησης τιμών, το κόστος παραγγελιών, κ.ά.

Στόχος, λοιπόν, είναι όπως αναφέρθηκε και στην αρχή η βελτιστοποίηση του συνολικού κόστους που επιτυγχάνεται με την υποβελτιστοποίηση μίας ή περισσότερων δραστηριοτήτων, λαμβάνοντας υπόψιν τις μεταξύ τους σχέσεις και αλληλεπιδράσεις και όχι με την μεμονωμένη βελτιστοποίηση του εκάστοτε κόστους.

1.2.4: Κριτήρια απόδοσης της εφοδιαστικής αλυσίδας

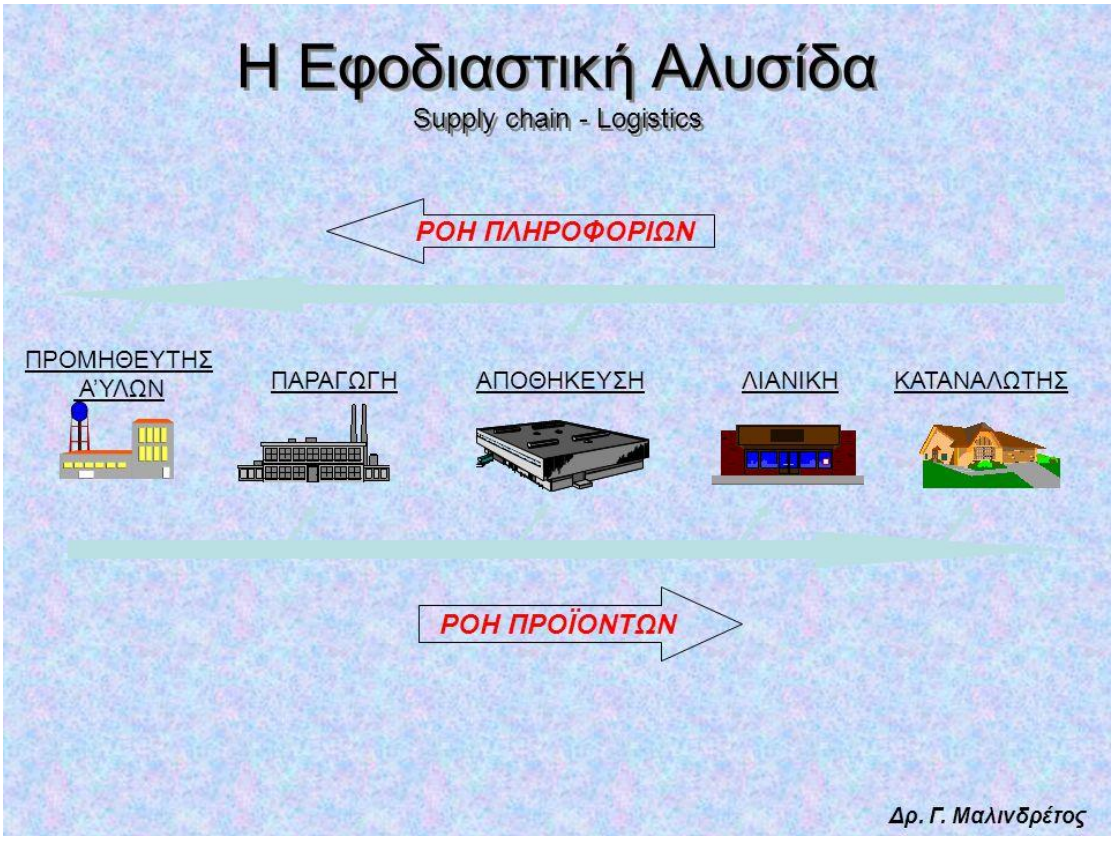
Για την σωστή και αποτελεσματική απόδοση της εφοδιαστικής αλυσίδας πρέπει να ληφθούν υπόψιν τα εξής κριτήτια:[1]

- Το ολικό κόστος
- Το ολικό κέρδος
- Ο χρονικός κύκλος

Σκοπός της εφοδιαστικής αλυσίδας είναι η ελαχιστοποίηση του ολικού κόστους, δηλαδή του συστήματος συνολικά και όχι κάθε τμήματος ξεχωριστά. Με τον ίδιο τρόπο επιδιώκει την μεγιστοποίηση του ολικού κέρδους. Ο χρονικός κύκλος αναφέρεται στον συνολικό χρόνο που απαιτείται για να ολοκληρωθέι η διαδικασία από τις πρώτες ύλες μέχρι το έτοιμο προϊόν που καταλήγει στον πελάτη.

Παρόλ' αυτά δεν είναι πάντα εύκολη η προσέγγισή της σαν ένα σύστημα για δύο λογους:[1]

- Διάφορα τμήματά της μπορεί να έχουν διαφορετικούς στόχους, που πιθανώς να έρχονται και σε σύγκρουση μεταξύ τους.
- Η εφοδιαστική αλυσίδα είναι ένα δυναμικό σύστημα που εξελίσσεται στο χρόνο. Για παράδειγμα η προσφορά, η ζήτηση, τα αποθέματα κ.ά αλλάζουν με τον χρόνο με αποτέλεσμα να δημιουργούνται πιθανώς κάποια προβλήματα στο σύστημα.



Εικόνα 1: Εικονική αναπαράσταση της εφοδιαστικής αλυσίδας

1.2:Χρονικός Προγραμματισμός συστημάτων παραγωγής

Ο χρονικός προγραμματισμός της παραγωγής επιδιώκει τον προγραμματισμό ενός συνόλου εργασιών, ικανοποιώντας διάφορους περιορισμούς, με σκοπό την βελτιστοποίηση ορισμένων κριτηρίων, όπως η ελαχιστοποίηση του χρόνου και του κόστους παραγωγής κλπ.

Τα περισσότερα προβλήματα χρονικού προγραμματισμού παραγωγής παρουσιάζουν ιδιαίτερη πολυπλοκότητα, με αποτέλεσμα να είναι δύσκολη η εύρεση της ολικής βέλτιστης λύσης. Σε αρκετές περιπτώσεις, όσο αυξάνεται το μέγεθος του προβλήματος(περισσότερες εργασίες, μηχανές) , λογικό είναι να αυξάνεται ανάλογα και ο χρόνος επίλυσης του προβλήματος. Σύμφωνα με την ορολογία που διατύπωσε ο Knuth το 1974 [2], για προβλήματα στα οποία δεν είναι γνωστός κάποιος αποδοτικός αλγόριθμος, ο οποίος να δίνει βέλτιστη λύση και είναι σχεδόν αδύνατο να βρεθεί, ανήκουν στην κατηγορία NP-hard προβλημάτων.

Για τις περιπτώσεις που παρουσιάζουν μεγάλη πολυπλοκότητα, έχουν αναπτυχθεί διάφορες τεχνικές βελτιστοποίησης, με την χρήση μοντέλων γραμμικού προγραμματισμού, ώστε να γίνεται πιο αποτελεσματικά η σωστή λήψη αποφάσεων. Κάθε πρόβλημα ορίζεται από μία σειρά παραμέτρων, οι οποίες αποτελούν την βάση για την εύρεση της καλύτερης δυνατής λύσης της συνάρτησης. Για παράδειγμα η καλύτερη δυνατή επιλογή μιας σειράς εργασιών που θα περάσουν από κάποιες μηχανές, λαμβάνοντας υπόψιν τους περιορισμούς αν υπάρχουν (διαθέσιμος εξοπλισμός, ανθρώπινο δυναμικό, συντήρηση εξοπλισμού κ.ά), ώστε να βρεθεί ο ελάχιστος χρόνος περάτωσης τους στις μηχανές.

1.2.1: Κατηγοριοποίηση προβλημάτων χρονικού προγραμματισμού

Υπάρχουν τρεις γενικές κατηγορίες συστημάτων παραγωγής για τα shop scheduling problems: [3]

1. **Συστήματα παραγωγής κατά παραγγελία (job shop):**Κάθε εργασία n αποτελείται από επιμέρους διεργασίες, οι οποίες εκτελούνται σε μία μηχανή m. Η σειρά εκτέλεσης των διεργασιών είναι διαφορετική για κάθε εργασία.
2. **Συστήματα παραγωγής συνεχούς ροής (flow shop):**Κάθε εργασία n αποτελείται από επιμέρους διεργασίες, οι οποίες εκτελούνται σε μία μηχανή m. Η σειρά εκτέλεσης των διεργασιών είναι ίδια για όλες τις εργασίες.
3. **Συστήματα ανοιχτής παραγωγής(open shop):** Κάθε εργασία n αποτελείται από επιμέρους διεργασίες, οι οποίες εκτελούνται σε μία μηχανή m. Η σειρά εκτέλεσης των διεργασιών είναι διαφορετική για κάθε εργασία και μία εργασία μπορεί να έχει παραπάνω από μία εναλλακτικές σειρές εκτέλεσης διεργασιών

Ακόμη μία κατηγοριοποίηση των προβλημάτων χρονικού προγραμματισμού, μπορεί να γίνει βάση των μηχανών ως εξής: [4], [5]

- **Μοναδική μηχανή (Single machine):** Αποτελεί το πιο απλό πρόβλημα από τις παρακάτω περιπτώσεις. Κάθε εργασία που αποτελείται από μία διεργασία αντιστοιχεί στην μοναδική μηχανή.
- **Πανομοιότυπες παράλληλες μηχανές (Identical parallel machines):** Έχουμε m πανομοιότυπες παράλληλα διατεταγμένες μηχανές. Κάθε εργασία απαιτεί μία διεργασία που μπορεί να υλοποιηθεί σε οποιαδήποτε από τις m μηχανές.

- **Παράλληλες μηχανές με διαφορετική ταχύτητα (Parallel machines with different speeds):** Παρόμοια με την προηγούμενη κατηγορία. Σε παράλληλη διάταξη μηχανές m που λειτουργούν σε διαφορετικές ταχύτητες.
- **Ασυσχέτιστες παράλληλες μηχανές (Unrelated parallel machines):** Αποτελεί την πιο γενική περίπτωση από όλες τις παραπάνω. Οι μηχανές m είναι σε παράλληλη διάταξη και η ταχύτητα κάθε μηχανής με την εκάστοτε διεργασία που εκπονεί.

Υπάρχουν βέβαια και άλλες κατηγορίες προβλημάτων χρονικού προγραμματισμού, ανάλογα με το κριτήριο που επιλέγουμε να τα κατηγοριοποιήσουμε.

1.2.2: Λειτουργίες και στόχοι χρονικού προγραμματισμού

Οι λειτουργίες που πρέπει να εκτελεστούν για την εφαρμογή του προγραμματισμού, συνοψίζονται ως εξής: [6]

- Ανάθεση παραγγελιών, εξοπλισμού και ανθρωπίνων πόρων στο κέντρο εργασίας.
- Καθορισμός (δρομολόγηση) της σειράς εκτέλεσης των εργασιών, θέτοντας προτεραιότητες.
- Κατανομή της εκτέλεσης των εργασιών (dispatching).
- Έλεγχος της παραγωγικής διαδικασίας (shop-floor control), που περιλαμβάνει:
  α)Ανάλυση και έλεγχος της εξέλιξης των εργασιών κατά την διάρκεια της εκτέλεσής τους.
  β)Επίσπευση καθυστερημένων και κρίσιμων εργασιών.

Οι στόχοι για έναν σωστό χρονικό προγραμματισμό είναι οι παρακάτω:

- Τήρηση των προθεσμιών για τις ημερομηνίες παράδοσεις(Due dates).
- Ελαχιστοποίηση του μέσου χρόνου παραγωγής μιας παρτίδας ή μιας παραγγελίας(Lead time).
- Ελαχιστοποίηση του χρόνου ή του κόστους προετοιμασίας και ρύθμισης του εξοπλισμού του κέντρου(Setup time).
- Ελαχιστοποίηση των εκκρεμούντων εργασιών στο σύστημα(Work in progress-WIP).
- Μεγιστοποίηση χρησιμοποίησης εξοπλισμού ή ανθρωπίνου δυναμικού(Machine or labor utilization).

1.2.3:Σύστημα παραγωγής συνεχούς ροής(Flow shop)

Το σύστημα συνεχούς ροής (Flow shop) αποτελεί ένα βασικό πρόβλημα για τον χρονικό προγραμματισμό συστημάτων. Τα συστήματα συνεχούς ροής έχουν μελετηθεί εκτενώς και ο πρώτος που πρότεινε ένα τέτοιο σύστημα ήταν ο Johnson (1954) [7]. Τα συστήματα αυτά χρησιμοποιούνται για μαζική παραγωγή περιορισμένης ποικιλίας τυποποιημένων προϊόντων και ο μηχανικός εξοπλισμός είναι τοποθετημένος στη σειρά.

Για την κατηγοριοποίηση των προβλημάτων flow shop, χρησιμοποιήθηκε από τον Graham et al(1979) [8], ο παρακάτω συμβολισμός: α/β/γ

όπου,

α: Αφορά την δομή του προβλήματος

β: Αναφέρεται στους περιορισμούς στα δεδομένα του προβλήματος

γ: Υποδεικνύει το κριτήριο βέλτιστου, δηλαδή τον παράγοντα που θέλου να βελτιστοποιήσουμε.

Γενικά, ένα σύστημα flow shop χαρακτηρίζεται από τις εξής προϋποθέσεις: (Baker και Trietsch (2009)

- Μια ομάδα ασυσχέτιστων εργασιών είναι διαθέσιμη για επεξεργασία τη χρονική στιγμή 0.
- Κάθε εργασία αποτελείται από διαφορετικές διεργασίες, οι οποίες απαιτούν διαφορετική μηχανή.
- Οι χρόνοι εξάρμωσης(setup time) είναι εξαρτώμενοι των μηχανών και συμπεριλαμβάνονται στους χρόνους επεξεργασίας.
- Οι χρόνοι επεξεργασίας των διεργασιών στις μηχανές είναι γνωστοί από την αρχή.
- Όλες οι μηχανές είναι πάντα διαθέσιμες.
- Όταν ξεκινήσει μία διεργασία σε μία μηχανή συνεχίζει χωρίς καμία διακοπή.

Έτσι λοιπόν στα συστήματα αυτά έχουμε τις μηχανές 1,2,..,M σε σειρά και τις εργασίες 1,2,...,N. Κάθε εργασία αποτελείται από επιμέρους διεργασίες, οι οποίες ουσιαστικά αντιστοιχούν με τις μηχανές. Για να μπορεί μία εργασία να ξεκινήσει στην μηχανή j, πρέπει να έχει ολοκληρωθεί η επεξεργασία της στην προηγούμενη μηχανή, δηλαδή την j-1, και η μηχανή j να είναι ελεύθερη. Οι

χρόνοι επεξεργασίας κάθε διεργασίας σε κάθε μηχανή, συμβολίζεται ως p_{ij}, όπου I είναι οι εργασίες 1,2,...,N και j είναι οι μηχανές 1,2,...,M. Σε προβλήματα flow shop, σκοπός μας είναι η ελαχιστοποίηση της χρονικής διάρκειας του προγράμματος, δηλαδή του χρόνου περάτωσης όλων των διεργασίων σε όλες τις μηχανές. Συμβολίζεται ως $C_{max}=C(N,M)$. Επομένως, για να βρούμε τον ελάχιστο χρόνο που απαιτείται για την ολοκλήρωση του προγράμματος, πρέπει να συμπληρώσουμε τον πίνακα C. Αρχικά, γίνεται η αρχικοποίηση κατά την οποία στον πρώτο στοιχείο του πίνακα C καταχωρούμε τον χρόνο που απαιτείται για να ολοκληρωθεί η πρώτη εργασία στην πρώτη μηχανή και έπειτα πρέπει να συμπληρώσουμε την πρώτη στήλη και την πρώτη γραμμή του πίνακα.

Για την συμπλήρωση της πρώτης στήλης χρησιμοποιούμε την σχέση:

$C_{i1}=C(i-1,1)+p_{i,1}$, $i\geq 2$ (1)

Για την συμπλήρωση της πρώτης γραμμής χρησιμοποιούμε την σχέση:

$C_{1j}=C(1,j-1)+p_{1,j}$, $j\geq 2$ (2)

Έπειτα, συνεχίζουμε με την συμπλήρωση του υπόλοιπου πίνακα χρησιμοποιώντας την σχέση:

$C_{ij}=\max\{C_{i-1,j},C_{i,j-1}\}+p_{ij}$, $i\geq 2$, $j\geq 2$ (3)

Έτσι, τελειώντας με την συμπλήρωση του πίνακα C έχουμε στο στοιχείο C(N,M) το ελάχιστο δυνατό χρόνο που χρειάζεται για την περάτωση όλων των εργασιών απ' όλες τις μηχανές.

Στόχος μας, λοιπόν, είναι να βρούμε την καλύτερη δυνατή σειρά εργασιών (χρησιμοποιώντας δίαφορες τεχνικές και αλγορίθμους), η οποία θα μας δίνει το μικρότερο δυνατό αποτέλεσμα. [9]

1.3: ΓΕΝΕΤΙΚΟΙ ΑΛΓΟΡΙΘΜΟΙ

Ένας γενετικός αλγόριθμος είναι μία στοχαστική επαναληπτική διαδικασία που διατηρεί το μέγεθος του πλυθησμού σταθερό σε κάθε επανάληψη, η οποία ονομάζεται γενιά. Στόχος είναι η δημιουργία μίας νέας λύσης που προκύπτει από το ταίριασμα δύο παλιών λύσεων. Για την δημιουργία της νέας λύσης εφαρμόζονται δύο τελεστές, ένας δυαδικός που ονομάζεται

διαστάυρωση και μοναδιαίος που ονομάζεται μετάλλαξη. Στην διαστάυρωση από την ανταλλαγη στοιχείων μεταξύ δύο αρχικών λύσεων, που ονομάζονται γονείς, προκύπτουν δύο νέα άτομα, οι απόγονοι.

1.3.1: Η γέννηση των γενετικών αλγορίθμων

Η ανάγκη εύρεσης λύσεων σε δύσκολα προβλήματα αναζήτησης και συνδυαστικής βελτιστοποίησης, όπως η ελαχιστοποίηση συναρτήσεων και η μηχανική μάθηση, οδήγησε στην ανάπτυξη των γενετικών αλγορίθμων. Οι γενετικοί αλγόριθμοι μιμούνται την διαδικασία της εξέλιξης στη φύση, όπως και ο αλγόριθμος βελτιστοποίησης σμήνους πυγολαμπίδων, που χρησιμοποιήθηκε στην υλοποίηση της εργασίας.

Στους γενετικούς αλγορίθμους, αρχικά, δημιουργείται ένας πληθυσμός από αρχικές λύσεις, εκ των οποίων επιλέγονται οι δύο γονείς μέσω μίας διαδικασίας. Στη συνέχεια, στο αποτέλεσμα αυτής της διαδικασίας εφαρμόζεται τοπική αναζήτηση και η λύση αυτή είναι που επιστρέφεται στον πληθυσμό, ο απόγονος, για τον οποίο υπολογίζεται μία συνάρτηση που εκτιμά την ποιότητά του. Η όλη διαδικασία επαναλαμβάνεται με σκοπό να βρεθεί το καλύτερο δυνατό βέλτιστο.

Τα βασικά σημεία των γενετικών αλγορίθμων είναι:

1. Η κωδικοποίηση των λύσεων, έτσι ώστε να αντιστοιχούν σε κάποιο άτομο.
2. Μια συνάρτηση που να εκτιμά την ποιότητα του κάθε ενός από τα άτομα του πληθυσμού.
3. Αρχικοποίηση του πληθυσμου.
4. Επιλογή των τελεστών.
5. Τελεστές αναπαραγωγής.

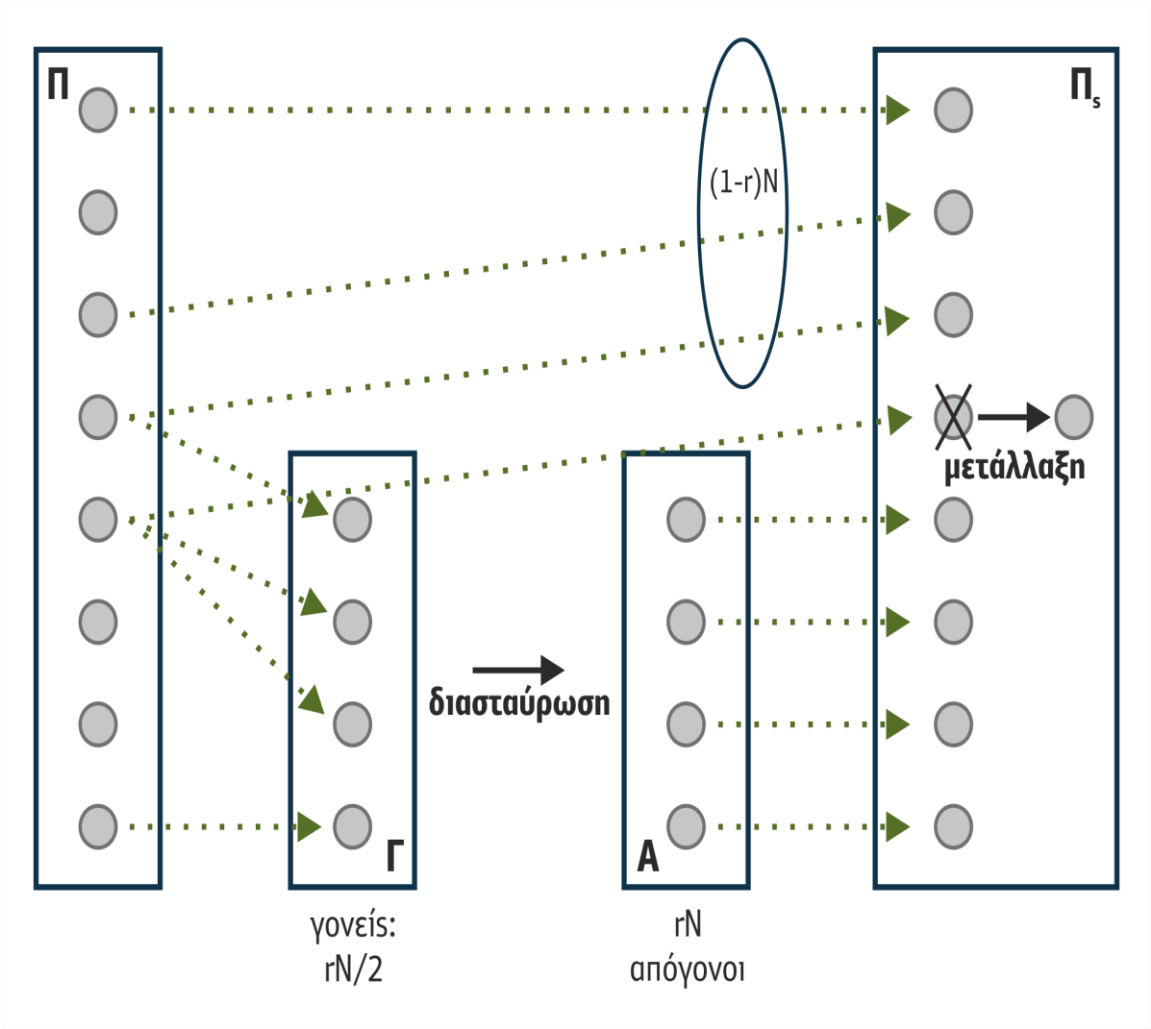
1.3.2: Η διαδικασία ενός γενετικού αλγορίθμου

Δύο βασικά ζητήματα πάνω σ' αυτούς του αλγορίθμους είναι το μέγεθος του αρχικού πληθυσμού και ο τρόπος που επιλέγονται τα άτομα που θα παίξουν τον ρόλο των γονέων. Σχετικά με το μέγεθος, εάν έχουμε πολύ μικρό πληθυσμό τότε δεν θα είναι αντιπροσωπευτικός όλων των δυνατών λύσεων και η λύση δεν θα είναι ικανοποιητική, ενώ αν είναι πολύ μεγάλος δεν θα είναι δυνατόν καμία μέθοδος να βρει γρήγορα και αποτελεσματικά λύση. Έτσι, επιλέγται κάτι ενδιάμεσα, ώστε να είναι εφικτό να βρεθεί λύση σχετικά

γρήγορα και να είναι ικανοποιητική. Όσων αφορά την επιλογή των γονέων υπάρχουν διάφοροι μέθοδοι. Μία απλή μέθοδος είναι η τυχαία επιλογή, η οποία όμως εγκαταλήφθηκε, διότι υπήρχε πιθανότητα να επιλέγουν τα χειρότερα άτομα του πληθυσμού ως γονείς, με αποτέλεσμα οι απόγονοι να έχουν κακή συνάρτηση ποιότητας. Στην συνέχεια, αναπτύχθηκε η κατάταξη της ποιότητας των γονέων, ώστε η επιλογή να γίνεται από τους καλύτερους και η διαδικασία επιλογής των δύο καλύτερων γονέων από δύο ομάδες που είχαν δημιουργηθεί τυχαία από τον αρχικό πληθυσμό (tournament selection). Όμως, η πιο εξελιγμένη μέθοδος επιλογής γονέων γίνεται με τον κανόνα της ρουλέτας (roulette wheel selection), όπου η επιλογή δεν είναι τυχαία, αλλά όσο καλύτερη ποιότητα έχει ένα άτομο, τόσο μεγαλύτερη και η πιθανότητα να επιλεγεί ως γονέας.

Εφόσον, γίνει η επιλογή των γονέων, ακολουθεί η δημιουργία των απογόνων, η οποία γίνεται είτε με την διαδικασία της διαστάυρωσης, είτε με την διαδικασία της μετάλλαξης, είτε και με τις δύο. Στην διασταύρωση, ο απόγονος δημιουργείται ή μόνο από έναν γονέα (asexual), ή από δύο γονείς που συμμετέχουν στην δημιουργία τουλάχιστον δύο απογόνων(sexual), ή περισσότεροι από δύο γονείς δημιουργούν δύο ή περισσότερους απογόνους (multi-recombination). Στην μετάλλαξη, κύριος στόχος είναι η εισαγωγή νέου γενετικού υλικού μέσα στον πληθυσμό, η οποίο πρέπει να γίνεται με προσοχή, ώστε να μην καταστραφούν καλές λύσεις. Η μετάλλαξη γίνεται με δύο τελεστές. Είτε με τον ομοιόμορφο τελεστή, όπου ο αριθμός και οι θέσεις των γονιδίων που θα γίνει η μετάλλαξη επιλέγονται τυχαία (uniform mutation), είτε με τον τελεστή που επιλέγονται δύο μόνο γονίδια και σ’ αυτά γίνεται τυχαία μετάλλαξη (inorder mutation).

Αφού δημιουργηθούν οι απόγονοι σειρά έχει η επιλογή του καινούριου πληθυσμού που θα αποτελέσει την νέα γενιά. Υπάρχουν πολλοί τρόποι για να γίνει η επιλογή. Ένας τρόπος είναι οι απόγονοι να αντικαταστήσουν όλους τους γονείς (replace oldest), το οποίο από θέμα βελτιστοποίησης δεν είναι και τόσο καλό. Άλλη διαδικασία είναι η κατάταξη των λύσεων βάση της συνάρτησης ποιότητας των γονέων και των απογόνων και η επιλογή να γίνει απ’ όλους. Ακόμη άλλες στρατηγικές που χρησιμοποιούνται είναι η αντικατάσταση των χειρότερων(replace worst), όπου οι απόγονοι αντικαθιστούν τους χειρότερους γονείς, η τυχαία αντικατάσταση (replace random), η δημιουργία ενός συνόλου από εκλεκτές λύσεις (elite solutions), οι οποίες εξαιρούνται από την επιλογή και συνεχίζουν από γεννιά σε γεννιά κ.ά.



Εικόνα 2: Αναπαράσταση διαδικάσιας γέννεσης απογόνων

Ένας γενετικός αλγόριθμος τερματίζει όταν ισχύει ένα από τα παρακάτω ή και συνδυασμός αυτών:

- Επιτεύχθεί ο μέγιστος αριθμός γεννιών (επαναλήψεων).
- Για ένα συσκεκριμένο αριθμό γεννιών δεν υπάρχει βελτίωση στην καλύτερη λύση του πληθυσμού.
- Δεν υπάρχει αλλαγή σε κανένα μέλος του πληθυσμού.
- Μία αποδεχτή λύση έχει βρεθεί.

Ακολουθεί ψευδοκώδικας για την διαδικασία των γενετικών αλγορίθμων:

Όνομα *Γενετιικοί Αλγόριθμοι*

Αρχικοποίηση

Επιλογή του τελεστή διαστάυρωσης

Επιλογή του τελεστή μετάλλαξης

Επιλογή του τελεστή επιλογής

Επιλογή του ποσοστού του πληθυσμού πάνω στο οποίο θα γίνει διαστάυρωση

Επιλογή του ποσοστού του πληθυσμού πάνω στο οποίο θα γίνει μετάλλαξη

Δημιουργία του αρχικού πληθυσμού

Υπολογισμός της αρχικής συνάρτησης ποιότητας για κάθε μέλος του πληθυσμού

Κύρια Φάση

***Do while** κριτήρια σταματήματος δεν ικανοποιούνται*

***Επιλογή** ατόμων από τον πληθυσμό για να γίνουν οι γονείς*

***Call** τελεστή διασταύρωσης για να παραχθούν οι απόγονοι*

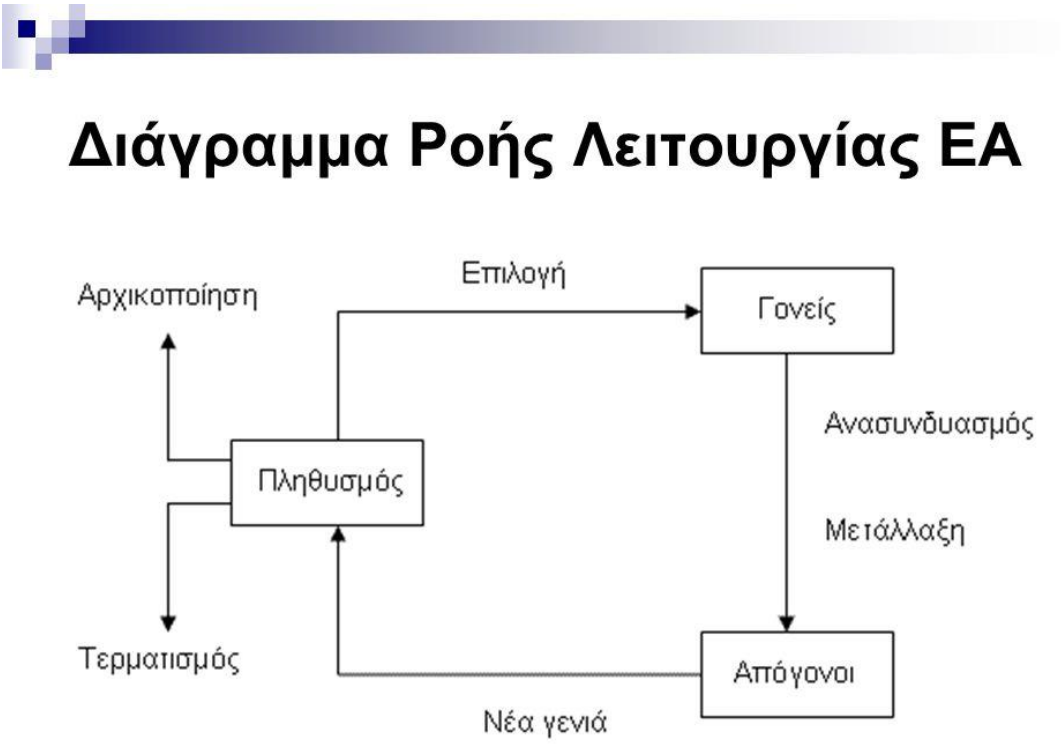
***Call** τελεστή μετάλλαξης*

Υπολογισμός της συνάρτησης ποιότητας για κάθε μέλους του πληθυσμού

***Αντικατέστησε** τον πληθυσμό της προηγούμενης γεννιάς με τους καταλληλότερους από ολόκληρο τον πληθυσμό*

Enddo

***Return** το καλύτερο άτομο-λύση.*



Εικόνα 3: Διάγραμμα ροής λειτουργίας γενετικών αλγορίθμων

1.3.3: Πλεονεκτήματα γενετικών αλγορίθμων

Τα βασικά πλεονεκτήματα των γενετικών αλγορίθμων είναι τα εξής:[9]

Μπορούν να επιλύουν δύσκολα προβλήματα γρήγορα και αξιόπιστα. Βασικό χαρακτηριστικό των γενετικών αλγορίθμων είναι η μεγάλη τους αποδοτικότητα, καθώς σε προβλήματα που έχουν δύσκολα προσδιαορισμένες λύσεις, φαίνεται να αντιμετωπίζονται καλύτερα από τους γενετικούς αλγορίθμους.

- **Είναι εύκολα επεκτάσιμοι και εξελίξιμοι**. Μπορείς εύκολα να κάνεις αλλαγές επεκτάσεις και μετεξελίξεις. Σε πολλές εφαρμογές έχουν αναφερθεί λειτουργίες των γενετικών αλγορίθμων που έχουν υποστεί αλλαγές προς όφελος της απόδοσης.
- **Μπορούν να συμμετέχουν σε υβριδικές μορφές**. Σε διάφορες ειδικές περιπτώσεις προβλημάτων είναι δυνατή η χρήση ενός υβριδικού γενετικού αλγορίθμου με μία άλλη μέθοδο.
- **Εφαρμόζονται σε πολύ περισσότερα πεδία από κάθε άλλη μέθοδο**, όπως στην οικονομία, στον σχεδιασμό μηχανών, στην

επίλυση μαθηματικών εξισώσεων, στην εκπαίδευση νευρωνικών δικτύων και πολλούς άλλους τομείς.

- **Δεν ενδιαφέρει η σημασία της υπό εξέτασης πληροφορίας**. Η μόνη αλληλεπίδραση του γενετικού αλγορίθμου με το περιβάλλον του είναι η αντικειμενική συνάρτηση, πράγμα που εγγυάται την επιτυχία του ανεξάρτητα από την σημασία του προβλήματος.
- **Δεν απαιτούν περιορισμούς στις συναρτήσεις που επεξεργάζονται**, σε σχέση με άλλες μεθόδους, πράγμα που τους καθιστά κατάλληλους για μεγάλο φάσμα προβλημάτων.

1.4: Αλγόριθμος Βελτιστοποίησης Σμήνους Πυγολαμπίδων (Glowworm Swarm Based Optimization Algorithm(GSO))

1.4.1: Χαρακτηριστικά ενός σμήνους

Όπως είναι κατανοητό και από τον τίτλο ο αλγόριθμος αυτός αναφέρεται σε ένα σμήνος. Κάθε σμήνος έχει κάποια χαρακτηριστικά είτε θετικά είτε αρνητικά που αφορούν την νοημοσύνη που παρουσιάζει το σμήνος.[10]

Τα θετικά χαρακτηριστικά ενός σμήνους είναι τα εξής:

- **Επεκτασιμότητα(scalability):** Τα σμήνη παρουσιάζουν διατήρηση της δομής στην ομάδα. Έχει παρατηρηθεί ότι τα έντομα λειτουργούν συλλογικά σε μεγάλο εύρος πληθυσμού και έχει διαπιστωθεί ότι ο συντονισμός και η συνεργασία λειτουργεί ανεξάρτητα από το αριθμό των ατόμων.
- **Ευελιξία(flexibility):** Υπάρχει δυνατότητα να προσθέτουν ή να αφαιρεθούν μονάδες από την ομάδα ή ακόμη να γίνει αναδιοργάνωση και ανακατανομή στην ομάδα.
- **Ευρωστία(robustness):** Με την έννοια ότι το τελικό συλλογικό σύστημα είναι εύρωστο, όχι μόνο στην μείωση των μονάδων (σφάλματα κλπ) αλλα και στην ελαχιστοποίηση στη σχεδίαση της μονάδας.
- **Αποκέντρωση (decentralization):** Οι υπομονάδες λειτουργοούν αυτόνομα, όπως και το περιβάλλον.
- **Αυτό-οργάνωση (self-organization):** Με την έννοια της εύρεσης λύσης, η οποία δεν είναι προκαθορισμένη αλλά προκύπτει.
- **Απλότητα (simplicity):** Οι διαμεσολαβητές σχεδιάζονται, προγραμματίζονται και ελέγχονται με μεγάλη ευκολία, έχουν την δυνατότητα εφαρμογής σε μικρές πλατφόρμες και να εκπαιδευτούν με

την χρήση γενετικών αλγορίθμων σε σχέση με άλλους αλγόριθμους μάθησης.

- **Ενοποίηση με το περιβάλλον (environmental integration):** Η δυνατότητα χρήσης των καταστάσεων που προσφέρει το περιβάλλον για τον έλεγχο του συστήματος (στιγματική συμπεριφορά). Για παράδειγμα, ενώ σε ένα συμβατικό σύστημα ο θόρυβος αποτελεί μεγάλο μειονέκτημα, το στιγματικό σύστημα το εκμεταλλέυεται για να εξυπηρετήσει κάποιους σκοπούς.

Παρά τα θετικά χαρακτηριστικά, η νοημοσύνη του σμήνους παρουσιάζει και κάποια αρνητικά γνωρίσματα:

- **Συμπεριφορά:** Δεν μπορούμε να εξάγουμε ασφαλή συμπεράσματα για την ατομική συμπεριφορά με βάση την συλλογική συμπεριφορά
- **Γνώση:** η έλλειψη καθολικής γνώσης για το περιβάλλον και για την λειτουργία της ομάδας μπορεί να θεωρηθεί αρνητικός παράγοντας σε κάποιες περιπτώσεις.
- **Σχεδιασμός:** Για τον σχεδιασμό τέτοιου είδους συστημάτων υπάρχουν πολύ λίγοι μηχανισμοί.
- **Δράση:** Η συμπεριφορά και η δράση του ατόμου μοιάζουν με θόρυβο ειδικά όταν η επιλογή δράσης είναι στοχαστική.
- **Παράμετροι:** Το σύστημα είναι ευαίσθητο στις παραμέτρους. Η παραμικρή αλλαγή σε οποιαδήποτε παράμετρο λειτουργίας του συστήματος μπορεί να επιφέρει μεγάλες αλλαγές στην συλλογική συμπεριφορά. Έτσι, πρέπει να γίνεται έλεγχος της συμπεριφοράς του συστήματος για διαφορετικά σύνολα παραμέτρων.

Στην εργασία αυτή με σκοπό την βελτίωση των αποτελεσμάτων σε διάφορα προβλήματα flow shop, χρησιμοποιήθηκε ο αλγόριθμος βελτιστοποίησης σμήνους πυγολαμπίδων. Ο αλγόριθμος αυτός ανήκει στους αλγόριθμους που είναι εμπνευσμένοι από φυσικές διαδικασίες, στην συγκεκριμένη περίπτωση στην συμπεριφορά που παρουσιάζει ένα σμήνος πυγολαμπίδων. Η ιδέα, λοιπόν, [10] βασίζεται στο γεγονός ότι όσο περισσότερο φώς (λουσιφερίνη) εκπέμπει μία πυγολαμπίδα, τόσο περισσότερο είναι ορατή και έλκει τις άλλες πυγολαμπίδες ή τα θηράματα. Άρα, οι πυγολαμπίδες θα κινηθούν προς την πυγολαμπίδα που εκπέμπει την μεγαλύτερη λουσιφερίνη. Έτσι, ο αλγόριθμος αυτός χωρίζεται σε τέσσερις φάσεις, τις οποίες θα περιγράψουμε αναλυτικά στην συνέχεια: [11]

1. Αρχικοποίηση
2. Ενημέρωση λουσιφερίνης
3. Μετακίνηση
4. Ενημέρωση εύρους γειτονιάς

Στην αρχικοποίηση, έχουμε ένα σύνολο από n οντότητες i=1,...,n που τοποθετούνται τυχαία στον χώρο λύσεων x_i, ώστε να έχουν όσο το δυνατόν μεγαλύτερη διασπορά. Κάθε πυγολαμπιδα i έχει μία αντικειμενική συνάρτηση $f(x_i(t))$ για κάθε τοποθεσία $x_i(t)$ στο χώρο λύσεων και μία τιμή λάμψης l_i την οποία γνωστοποιεί στους γειτονές της. Στην αρχή όλες οι πυγολαμπίδες έχουν την ίδια τιμή λάμψης L_0 ίση με μηδέν.

Η φάση ενημέρωσης της λάμψης, δηλαδή της λουσιφερίνης, εξαρτάται από την τιμή της αντικειμενικής συνάρτησης $f(x_i(t))$ στο σημείο που βρίσκεται η πυγολαμπίδα. Κατά την διάρκεια της ενημέρωσης κάθε πυγολαμπίδα προσθέτει στην προηγούμενή της ποσότητα μία τιμή ανάλογης της τωρινής θέσης στον χώρο λύσεων και αφαιρεί μία ποσότητα, η οποία δείχνει την εξασθένηση που παρουσιάζει η πυγολαμπίδα με τον χρόνο. Επομένως, η ενημέρωση της λουσιφερίνης γίνεται με την χρήση του εξής τύπου:

$$Li(t+1) = (1-\rho)Li(t) + \gamma(f(xi(t+1))) \qquad (4)$$

όπου, το ρ είναι η ποσότητα που μας δείχνει την εξασθένηση της λάμψης της πυγολαμπίδας με τον χρόνο και ανήκει στο διάστημα (0,1). Το γ είναι μία σταθερά που εξαρτάται από την τιμή της αντικειμενικής συνάρτησης σε σχέση με τον χρόνο.

Αφού γίνει η ενημέρωση της λουσιφερίνης σε όλες τις πυγολαμπίδες, κάθε μία πυγολαμπίδα αποφασίζει να κινηθεί προς μία άλλη πυγολαμπίδα, η οποία έχει μεγαλύτερη τιμή λάμψης από την δική της. Για την μετακίνησή της προς κάποιον γείτονα χρησιμοποιεί μία πιθανότητα, η οποία δίνεται από τον τύπο:

$$pij(t) = \frac{\mathrm{Lj(t)-Li(t)}}{\sum_{k\in Ni(t)} Lk(t)-Li(t)} \qquad (5)$$

Όπου $j\in N_i(t)=\{j : d_{ij}(t) < r^i_d(t), l_i(t) < l_j(t)\}$, είναι το σύνολο των γειτόνων του i την χρονική στιγμή t, και $d_{ij}(t)$ είναι η Ευκλείδια απόσταση ανάμεσα σε δύο πυγολαμπίδες i και j την χρονική στιγμή t. Ακόμη, r^i_d αναπαριστά την μεταβλητή ακτίνα γειτονιάς, στην οποία μπορεί να κινηθεί η κάθε πυγολαμπίδα, που σχετίζεται με την πυγολαμπίδα i την χρονική στιγμή t και παίρνει τιμές $0< r^i_d<r_s$. Αν συμβολίσουμε με r_0 την αρχική απόσταση που μπορεί μία πυγολαμπίδα να ψάξει για γείτονες τότε:

$r^i_d(t+1)=\min\{ r_s,\max\{0, r^i_d(t)+\beta(n_t-|N_i(t)|)\}\}$ (6)

όπου β είναι μία σταθερά και n_t είναι το πλήθος των γειτόνων που μπορεί να έχει μία πυγολαμπίδα.

Τελευτάια φάση αποτελεί η ενημέρωση του εύρους γειτονιάς. Έτσι, αν μία πυγολαμπίδα i αποφασίσει να κινηθεί προς μία άλλη πυγολαμπίδα j τότε η θέση της ενημερώνεται με την ακόλουθη σχέση:

$$xi(t+1) = xi(t) + s(\frac{xj(t)-xi(t)}{||xj(t)-xi(t)||}) \quad (7)$$

Όπου s>0 είναι το βήμα που κινείται μία πυγολαμπίδα προς την άλλη.

Ο λόγος που το μήκος αναζήτησης της γειτονιάς r^i_d είναι μεταβλητό και όχι σταθερό, είναι ότι πρέπει να αυξηθούν οι ικανότητες αναζήτησης του αλγορίθμου.

Αυτές είναι οι τέσσερις φάσεις, που ακολουθεί ο αλγόριθμος βελτιστοποίησης σμήνους πυγολαμπίδων με σκοπό την βελτίωση ενός αποτελέσματος. Στην συνέχεια ακολουθεί ένας ψευδοκώδικας και μία εικόνα που αναπαριστούν τον συγκεκριμένο αλγόριθμο: [12]

Αλγόριθμος *Βελτιστοποίηση Σμήνους Πυγολαμπίδων*

Καθορισμός των παραμέτρων που χρησιμοποιούνται στην κύρια φάση

Καθορισμός του μέγιστου αριθμού επαναλήψεων

Καθορισμός του αριθμού των πυγολαμίδων

Καθοριμσμός παραμέτρου s

Αρχικοποίηση

Δημιουργία του αρχικού πληθυσμού από τοποθεσίες πυγολαμπίδων

Αρχικοποίηση $L_i(0)=L_0$, $r^j_d(0)=r(0)$

Υπολογισμός της αντικειμενικής συνάρτησης για κάθε πυγολαμπίδα

Κύρια φάση

***Do while** ο μέγιστος αριθμός επαναλήψεων δεν έχει επιτευχθεί*

For** κάθε πυγολαμπίδα **do

Φάση ενημέρωσης της λάμψης με την χρήση της εξίσωσης (4)

Υπολογισμός της γειτονιάς αναζήτησης κάθε πυγολαμπίδας

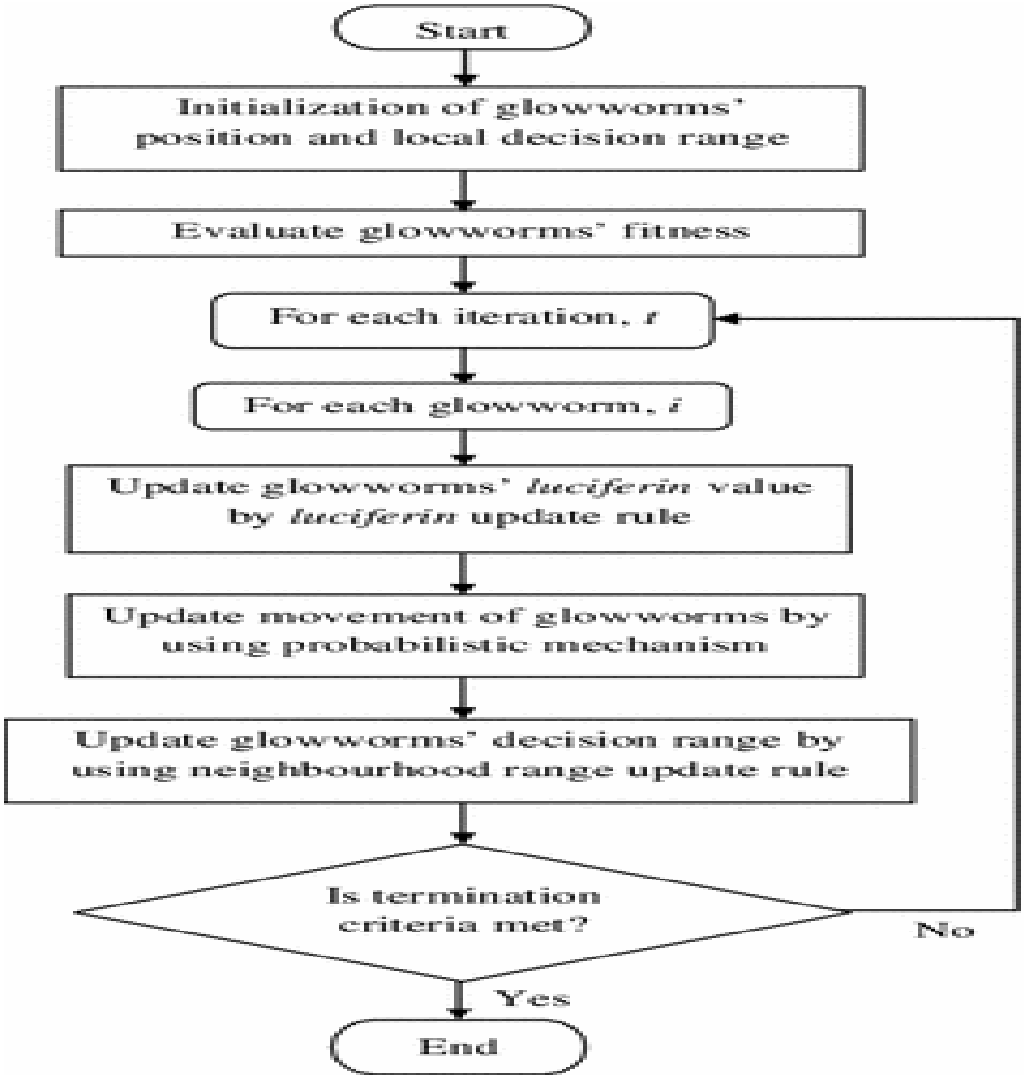
Ενημέρωση της θέσης κάθε πυγολαμπίδας βάσει της εξίσωσης (7)

Endfor

***Διατήρησε** τη βέλτιστη λύση από κάθε πυγολαμπίδα*

Enddo

***Επιστροφή** της καλύτερης πυγολαμπίδας(καλύτερη λύση)*



Εικόνα 4: Σχηματική απεικόνιση GSO

2η ΕΝΟΤΗΤΑ-ΜΕΘΟΔΟΛΟΓΙΑ

2.1: Εφαρμογή σε βιβλιογραφικά δεδομένα

Οι αλγόριθμοι που περιγράφηκαν στην προηγούμενη ενότητα κωδικοποιήθηκαν σε γλώσσα προγραμματισμού C και εφαρμόστηκαν σε διάφορα flow shop προβλήματα, με ποικιλία μεγεθών, όπως διαμοσιεύτηκαν από τον Taillard (1993). Ο Taillard έδωσε το ανώτερο και το κατώτερο όριο της τιμής του makespan σε προβλήματα: 20x5, 20x10, 20x20, 50x5, 50x10, 50x20, 100x5, 100x10, 100x20, 200x01, 200x20, 500x20 και κάθε ομάδα προβλημάτων αποτελείται από δέκα προβλήματα. Ο πρώτος αριθμός αντιπροσωπεύει τον αριθμό των εργασιών και ο δεύτερος των μηχανών. Το

ανώτερο όριο προσεγγίζεται από ευρετικούς αλγορίθμους. Όσο πιο κοντά στο ανώτερο όριο το αποτέλεσμα τόσο καλύτερη η απόδοση του αλγορίθμου. Κατά καιρούς έχουν δημοσιευτεί διάφορες έρευνες, οι οποίες αλλάζουν το ανώτερο όριο σε διάφορα προβλήματα του Taillard (Pan et al, 2008). Βάση αυτών των προβλημάτων θα γίνει η αξιολόγηση της συνάρτησης ποιότητας των αλγορίθμων που αναπτύχθηκαν σ' αυτήν την εργασία

.

2.2: Δομή προγράμματος-παρουσίαση αλγορίθμου

Παρακάτω θα γίνει η παρουσίασει του αλγορίθμου σε ψευδοκώδικα ανά τμήματα με επεξήγηση για το τι κάναμε σε κάθε τμήμα, έτσι ώστε να γίνει κατανοητός ο τρόπος που εργαστήκαμε.

2.2.1: Δήλωση μεταβλητών

Διάβασε *M(αριθμός μηχανών)*

Διάβασε *N(αριθμός εργασιών)*

Διάβασε *AL(αριθμός αρχικων λύσεων)*

Voithitiko(1, N) =0

Veltistos_Xronos(AL, 1) =0

makespan(M, N) = 0

makespan2(M, N) = 0

makespan3(M, N) = 0

B(AL, 1) =0

L(AL, 1) =0

SeiraTest(1, N) =0

C(M, N) =0

D(M, N) =0

G(M, N) =0

SeiraErgasiwn(AL, N) =0

SeiraErgasiwn2(AL, N) =0

VeltistiDiadromi(1, N) =0

g=0.9;

p=rand;

Veltisto=100000000;

A=xlsread('Efodiastikii.xlsx')

Το κομμάτι αυτό, γίνεται στην αρχή κάθε αλγορίθμου. Αποτελεί την δήλωση και την αρχικοποίηση όλων των μεταβλητών που θα χρησιμοποιηθούν στο πρόγραμμα. Αρχικά ζητείται από τον χρήστη να δώσει τιμή για τον αριθμό των μηχανών, των εργασιών και για τις αρχικές λύσεις που επιθυμούμε να δημιουργήσουμε. Έπειτα, γίνεται η αρχικοποίηση κάποιων πινάκων δίνοντας την τιμή μηδέν και κάποιων μεταβλητών δίνοντας μία συγκεκριμένη τιμή, η οποία εξυπηρετεί την υλοποίηση του προγράμματος. Όπως, για παράδειγμα η εκχώρηση της τιμής 100000000 στην μεταβλητή *Veltisto*, που μας δίνει την δυνατότητα η πρώτη λύση που θα βρούμε να αντικατασταθεί στην μεταβλητή *Veltisto,* εφόσον η τιμή είναι πολύ μεγάλη(πράγμα που θα δείξουμε και παρακάτω στον ψευδοκώδικα). Τέλος, γίνεται η εισαγωγή των δεδομένων του εκάστοτε προβλήματος, τα οποία έχουμε αποθηκεύσει σε ένα αρχείο excel, στον πίνακα *A.*

2.2.2: Αρχικοιποίηση πληθυσμού

ΓΙΑ *k από 1 μέχρι AL*

SeiraTest=randperm(N)

ΓΙΑ *j από 1 μέχρι N*

SeiraErgasiwn(k, j)=SeiraTest(j)

ΤΕΛΟΣ ΕΠΑΝΑΛΗΨΗΣ

ΓΙΑ *i από 1 μέχρι M*

    ΓΙΑ *j από 1 μέχρι N*

C(i, j)=A(i, SeiraTest(j))

ΤΕΛΟΣ ΕΠΑΝΑΛΗΨΗΣ

ΤΕΛΟΣ ΕΠΑΝΑΛΗΨΗΣ

makespan(1, 1)=C(1, 1)

ΓΙΑ *j από 2 μέχρι N*

makespan(1, j)= makespan(1, j-1) +C(1, j)

ΤΕΛΟΣ ΕΠΑΝΑΛΗΨΗΣ

ΓΙΑ *i από 2 μέχρι M*

makespan(i, 1)=makespan(i-1, j) +C(i, 1)

ΤΕΛΟΣ ΕΠΑΝΑΛΗΨΗΣ

ΓΙΑ *i από 2 μέχρι M*

ΓΙΑ *j από 2 μέχρι N*

ΑΝ *makespan(i-1, j)>makespan(i, j-1)* ***ΤΟΤΕ***

makespan(i, j)= makespan (i-1, j)+C(i, j)

ΑΛΛΙΩΣ

makespan(i, j)= makespan (i, j-1)+C(i, j)

ΤΕΛΟΣ ΑΝ

ΤΕΛΟΣ ΕΠΑΝΑΛΗΨΗΣ

ΤΕΛΟΣ ΕΠΑΝΑΛΗΨΗΣ

Veltistos_Xronos(k)=makespan(M, N)

*L(k)=(1-p)*0+g*Veltistos_Xronos(k)*

ΤΕΛΟΣ ΕΠΑΝΑΛΗΨΗΣ

Αφού, έγινε η δήλωση όλων των μεταβλητών και των πινάκων, που χρησιμοποιήθηκαν στο πρόγραμμα, το πρώτο βήμα ήταν να λύσουμε ένα κλασικό πρόβλημα flow shop. Αρχικά με την εντολή ***randperm(N)***, όπου N ο αριθμός των εργασιών δημιουργήσαμε ένα διάνυσμα, που περιέχει όλες τις εργασίες με τυχαία σειρά, το οποίο στην εκάστοτε επανάληψη αποθηκεύεται στην αντίστοιχη γραμμή ενός πίνακα. Έπειτα, δημιουργήθηκε ο πίνακας C

οποιός περιέχει τους χρόνους των εργασιών (με την σειρά που δημιουργήσαμε) στις μηχανές και ξεκινήσαμε να λύνουμε το flow shop πρόβλημα, όπως αναφέραμε και στην πρώτη ενότητα. Αρχικά, στο πρώτο στοιχείο του πίνακα makespan εκχωρήσαμε το πρώτο στοιχείο του πίνακα C, ώστε στην συνέχεια να συμπλήρώσουμε την πρώτη γραμμή και την πρώτη στήλη του πίνακαι makespan. Η πρώτη γραμμή συμπληρώθηκε χρησιμοποιώντας την εξίσωση (2) και η πρώτη στήλη χρησιμοποιώντας την εξίσωση (1) (από το δεύτερο στοιχείο και μετά εφόσον το πρώτο είναι ήδη καταχωρημένο). Τέλος, συμπληρώσαμε τον υπόλοιπο πίνακα makespan με την χρήση της εξίσωσης (3), όπως αναφέρθηκε στην πρώτη ενότητα. Αυτή η διαδικάσια έγινε τόσες φορές όσες και οι αρχικές λύσεις που επιλέξαμε. Έτσι, δημιουργήσαμε ένα πλήθος λύσεων, δηλαδή κάναμε την αρχικοποίηση του πληθυσμού, με διάφορες τυχαίες σειρές εργασιών και τους αντίστοιχους χρόνους περάτωσης των εργασιών από τις μηχανές, ενώ παράλληλα ενημερώσαμε την λουσιφερίνη της κάθε λύσης με την εξίσωση (4), ώστε στην συνέχεια να μπορούμε να εφαρμόσουμε την μέθοδο βελτιστοποίησης σμήνους πυγολαμπίδων.

2.2.3: Εφαρμογή της μεθόδου βελτιστοποίησης σμήνους πυγολαμπίδων

```
ΓΙΑ i από 1 μέχρι AL
    ΓΙΑ j από 1 μέχρι N
    SeiraErgasiwn2(i, j)=SeiraErgasiwn(i, j)
    ΤΕΛΟΣ ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ ΕΠΑΝΑΛΗΨΗΣ
ΓΙΑ i από 1 μέχρι 100
max=-10000000000
ΓΙΑ k από 1 μέχρι AL
    ΓΙΑ m από k+1 μέχρι k+4
        ΑΝ (m>AL)
        P=L(m-AL)-L(k);
        ΑΛΛΙΩΣ
        P=L(m)-L(k);
        ΤΕΛΟΣ ΑΝ
```

```
        ΑΝ (P>max)
        max=P;
          ΑΝ (m>AL)
          Pros=m-AL;
          ΑΛΛΙΩΣ
          Pros=m;
         ΤΕΛΟΣ ΑΝ
        ΤΕΛΟΣ ΑΝ
    ΤΕΛΟΣ ΕΠΑΝΑΛΗΨΗΣ
  ΓΙΑ n από k-4 μέχρι k-1
      ΑΝ (n<1)
      P=L(AL+n)-L(k);
      ΑΛΛΙΩΣ
      P=L(n)-L(k);
      ΤΕΛΟΣ ΑΝ
       ΑΝ (P>max)
        max=P;
          ΑΝ (n<1)
          Pros=AL+n;
          ΑΛΛΙΩΣ
          Pros=n;
          ΤΕΛΟΣ ΑΝ
        ΤΕΛΟΣ ΑΝ
    ΤΕΛΟΣ ΕΠΑΝΑΛΗΨΗΣ
Apo=k
```

Έτσι, λοιπόν, έχοντας δημιουργήσει τις αρχικές λύσεις που χρειαζόμασταν, αρχίσαμε να υλοποιούμε τον αλγόριθμο βελτιστοποίησης σμήνους πυγολαμπίδων. Στην αρχή δημιουργήσαμε ένα δεύτερο πίνακα με τις σειρές των εργασιών, όμοιο με τον πρώτο, στον οποίο θα γίνουν οι αλλαγές μετά την υλοποίηση του αλγοριθού. Έπειτα, ξεκίνησε η αναζήτηση στην γειτονιά, που αποφασίσαμε να είναι οι τέσσερις προηγούμενες και οι τέσσερις επόμενες λύσεις από αυτή που εξετάζουμε κάθες φορά, ώστε να βρούμε ποια λύση έχει μεγαλύτερη λουσιφερίνη και έλκει αυτήν που εξετάζεται. Οι αρχικές λύσεις θεωρείται σαν να έχουν σχηματίσει έναν κύκλο στο χώρο, οπότε για παράδειγμα αν έχουμε 50 αρχικές λύσεις, οι τέσσερις προηγούμενες της 1 είναι οι 50, 49, 48, 47, ενώ οι τέσσερις επόμενες της 50 είναι αντίστοιχα οι 1, 2, 3 και 4. Με αυτόν τον τρόπο επιλέχθηκε η γειτονιά αναζήτησης, ξεκινώντας από τις τέσσερις επόμενες, κρατώντας τον καλυτερο γείτονα(αυτός με την μεγαλύτερη λουσιφερίνη σε σχέση με αυτήν που εξετάζεται) υπολογίζοντας την διαφορά της λουσιφερίνης τους. Ακολούθησε η σύγκριση των τεσσάρων προηγούμενων λύσεων με τον καλύτερο από τις τέσσερις επόμενες, ώστε τελικά να κρατήσουμε τον καλύτερο γείτονα απ' όλη την γειτονιά και να κινηθούμε προς αυτόν.

```
AN (mod(N,2)==0) %Crossover των εργασιών με τον πλησιέστερο γείτονα
    ΓΙΑ j από (N/2+1) μέχρι N %Για N άρτιο
        SeiraErgasiwn2(Apo,j)=SeiraErgasiwn2(Pros,j);
    ΤΕΛΟΣ ΕΠΑΝΑΛΗΨΗΣ
ΑΛΛΙΩΣ
    ΓΙΑ j από round(N/2) μέχρι N%Για N περιττό
        SeiraErgasiwn2(Apo,j)=SeiraErgasiwn2(Pros,j);
    ΤΕΛΟΣ ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ ΑΝ

ΓΙΑ j από 1 μέχρι N %Δημιουργία βοηθητικού διανύσματος ώστε να βρούμε πόσες φορές εμφανίζεται η κάθες εργασία

Voithitiko(1,SeiraErgasiwn2(Apo,j))=Voithitiko(1,SeiraErgasiwn2(Apo,j))+1
ΤΕΛΟΣ ΕΠΑΝΑΛΗΨΗΣ
```

a=1;

ΟΣΟ *(a==1)* ***ΚΑΝΕ*** *%Μετατροπή της λύσης που πήραμε από το crossover σε εφικτή λύση με την βοήθεια του βοηθητικού διανύσματος*

ΓΙΑ *j από 1 μέχρι N*

ΑΝ *(Voithitiko(1,j)==2)*

Voithitiko(1,j)=1;

test2=j;

break

ΤΕΛΟΣ ΑΝ

ΤΕΛΟΣ ΕΠΑΝΑΛΗΨΗΣ

ΓΙΑ *j από 1 μέχρι N*

ΑΝ *(Voithitiko(1,j)==0)*

Voithitiko(1,j)=1;

test0=j;

break

ΤΕΛΟΣ ΑΝ

ΤΕΛΟΣ ΕΠΑΝΑΛΗΨΗΣ

ΓΙΑ *j από 1 μέχρι N*

ΑΝ *(SeiraErgasiwn2(Apo,j)==test2)*

SeiraErgasiwn2(Apo,j)=test0;

break

ΤΕΛΟΣ ΑΝ

ΤΕΛΟΣ ΕΠΑΝΑΛΗΨΗΣ

ΓΙΑ *j από 1 μέχρι N*

ΑΝ *(Voithitiko(1,j)==0)*

a=1;

break

ΑΛΛΙΩΣ

a=0;

ΤΕΛΟΣ ΑΝ

ΤΕΛΟΣ ΕΠΑΝΑΛΗΨΗΣ

ΤΕΛΟΣ ΕΠΑΝΑΛΗΨΗΣ

Μετά την επιλογή του καλύτερου γείτονα για την αρχική λύση που εξετάζουμε ξεκινάει η διαδικασία αλλαγής των λύσεων. Αυτό επιτυγχάνεται με το crossover μεταξύ των δύο αυτων λύσεων. Κατά την διαδικασία του crossover έχουμε τους δύο γονείς, την αρχική λύση που εξετάζουμε(πρώτος γονέας) και τον καλύτερο γείτονα ως προς αυτή (δεύτερος γονέας), μέσα από τους οποίους θα προκύψει ο απόγονος. Ο πρώτος γονέας δίνει τις πρώτες εργασίες μέχρι το μισό των συνολικών εργασιών, ενώ ο δεύτερος δίνει τις υπόλοιπες. Για παράδειγμα: (έστω δύο γονείς με 10 εργασίες συνολικά ο καθένας)

Πρώτος γονέας:

4	3	10	6	7	9	1	8	2	5

Δεύτερος γονέας:

1	7	4	2	5	6	10	8	3	9

Απόγονος που προκύπτει από το crossover:

4	3	10	6	7	6	10	8	3	9

Όμως, όπως φαίνεται και μέσα από το παραπάνω παράδειγμα είναι πολύ πιθανόν μετά από αυτή την διαδικασία να προκύψει μία λύση (απόγονος), η οποία δεν θα είναι εφικτή, θα έχει δηλαδή κάποιες εργασίες δύο φορές και άλλες τόσες εργασίες καμία φορά. Έτσι, με την βοήθεια ενός βοηθητικού διανύσματος, το οποίο στην κάθε θέση δείχνει η αντίστοιχη εργασία πόσες φορές εμφανίζεται στον απόγονο που δημιουργήσαμε. Ξέροντας, λοιπόν, ποιες εργασίες εμφανίζονται και πόσες φορές, αλλάζουμε την πρώτη εργασία που βρίσκουμε να εμφανίζεται δύο φορές με αυτην που βρίσκουμε πρώτη να μην εμφανίζεται καμία. Η διαδικασία επαναλαμβάνεται μέχρι το βοηθητικό διάνυσμα να έχει μόνο την τιμή ένα σε όλες τις θέσεις, δηλαδή όλες οι

εργασίες να εμφανίζονται μία φορά. Με αυτόν τον τρόπο τερματίζοντας αυτή η διαδικασία πλεόν ο απόγονος μας έχει μετατραπεί σε μία εφικτή λύση για την οποία μπορούμε να υπολογίσουμε την συνάρτηση ποιότητας.

%΄Υπολογισμός ξανά του βέλτιστου χρόνου

***ΓΙΑ** i από 1 μέχρι M*

***ΓΙΑ** j από 1 μέχρι N*

D(i,j)=A(i,SeiraErgasiwn2(Apo,j));%Αλλαγή του A σε D με την νέα σειρά εργασιών

ΤΕΛΟΣ ΕΠΑΝΑΛΗΨΗΣ

makespan2(1,1)=D(1,1);

***ΓΙΑ** j από 2 μέχρι N %Δημιουργία της πρώτης γραμμής*

makespan2(1,j)=makespan2(1,j-1)+D(1,j);

ΤΕΛΟΣ ΕΠΑΝΑΛΗΨΗΣ

***ΓΙΑ** i από 2 μέχρι M%Δημιουργία της πρώτης στήλης*

makespan2(i,1)=makespan2(i-1,1)+D(i,1);

ΤΕΛΟΣ ΕΠΑΝΑΛΗΨΗ

%Συμπλήρωση του υπόλοιπου πίνακα makespan

***ΓΙΑ** i από 2 μέχρι M*

***ΓΙΑ** j από 2 μέχρι N*

***ΑΝ** (makespan2(i-1,j)>makespan2(i,j-1))*

makespan2(i,j)=makespan2(i-1,j)+D(i,j);

ΑΛΛΙΩΣ

makespan2(i,j)=makespan2(i,j-1)+D(i,j);

ΤΕΛΟΣ ΑΝ

ΤΕΛΟΣ ΕΠΑΝΑΛΗΨΗΣ

ΤΕΛΟΣ ΕΠΑΝΑΛΗΨΗΣ

%Σύγκριση του βέλτιστου

ΑΝ *(makespan2(M,N)<Veltistos_Xronos(k))*

Veltistos_Xronos(k)=makespan2(M,N);

ΓΙΑ *j από 1 μέχρι N*

SeiraErgasiwn(k,j)=SeiraErgasiwn2(k,j);

ΤΕΛΟΣ ΕΠΑΝΑΛΗΨΗΣ

ΤΕΛΟΣ ΑΝ

Σ' αυτό το κομμάτι ουσιαστικά επαναλαμβάνεται η ίδια διαδικασία που έγινε στην αρχή του προγράμματος. Λύνουμε, δηλαδή, ένα κλασικό πρόβλημα flow shop αποθηκευοντας τις νέες σειρές εργασιών και του αντίστοιχους συνολικούς χρόνους περάτωσης τους από τις μηχανές σε νέους πίνακες. Τέλος, συγκρινεται κάθε φορά το νέο αποτέλεσμα με τους παλιούς πίνακες και αν βρεθεί καλύτερος συνολικός χρόνος, τότε γίνεται αντικατάσταση της σειράς εργασιών και του χρόνου στον παλιό πίνακα, αλλιώς παραμένει ίδιος.

2.2.4: Τοπική Αναζήτηση

%Τοπική Αναζήτηση(αλλάζουμε τυχαία 2 εργασίες στον διάνυσμα)

ΓΙΑ *i από 1 μέχρι 10*

E1=randi(N);

ΓΙΑ *b από 1 μέχρι 100*

E2=randi(N);

ΑΝ *(E1~=E2)*

break

ΤΕΛΟΣ ΑΝ

ΤΕΛΟΣ ΕΠΑΝΑΛΗΨΗΣ

ΓΙΑ *j από 1 μέχρι N*

ΑΝ *(SeiraErgasiwn2(Apo,j)==E1)*

k1=j;

break

ΤΕΛΟΣ ΑΝ

ΤΕΛΟΣ ΕΠΑΝΑΛΗΨΗΣ

ΓΙΑ *j από 1 μέχρι N*

ΑΝ *(SeiraErgasiwn2(Apo,j)==E2)*

k2=j;

ΤΕΛΟΣ ΑΝ

ΤΕΛΟΣ ΕΠΑΝΑΛΗΨΗΣ

SeiraErgasiwn2([k1 k2])=SeiraErgasiwn2([k2 k1]);

%Υπολογισμός ξανά του βέλτιστου χρόνου μετά την αλλαγή

ΓΙΑ *i από 1 μέχρι M*

ΓΙΑ *j από 1 μέχρι N*

G(i,j)=A(i,SeiraErgasiwn2(Apo,j));%Allagi tou A se G me tin nea seira ergasiwn

ΤΕΛΟΣ ΕΠΑΝΑΛΗΨΗΣ

ΤΕΛΟΣ ΕΠΑΝΑΛΗΨΗΣ

makespan3(1,1)=G(1,1);

ΓΙΑ *j από 2 μέχρι N %Δημιουργία της πρώτης γραμμής*

makespan3(1,j)=makespan3(1,j-1)+G(1,j);

ΤΕΛΟΣ ΕΠΑΝΑΛΗΨΗΣ

ΓΙΑ *i από 2 μέχρι M%Δημιουργία της πρώτης στήλης*

makespan3(i,1)=makespan3(i-1,1)+G(i,1);

ΤΕΛΟΣ ΕΠΑΝΑΛΗΨΗΣ

%Συμπλήρωση του υπόλοιπου πίνακα makespan

ΓΙΑ *i από 2 μέχρι M*

ΓΙΑ *j από 2 μέχρι N*

ΑΝ *(makespan3(i-1,j)>makespan3(i,j-1))*

```
        makespan3(i,j)=makespan3(i-1,j)+G(i,j);
    ΑΛΛΙΩΣ
        makespan3(i,j)=makespan3(i,j-1)+G(i,j);
    ΤΕΛΟΣ ΑΝ
  ΤΕΛΟΣ ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ ΕΠΑΝΑΛΗΨΗΣ


%Σύγκριση του βέλτισου
ΑΝ (makespan3(M,N)<Veltistos_Xronos(k))
  Veltistos_Xronos(k)=makespan3(M,N);
  ΓΙΑ j από 1 μέχρι N
     SeiraErgasiwn(k,j)=SeiraErgasiwn2(k,j);
  ΤΕΛΟΣ ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ ΑΝ

ΤΕΛΟΣ ΕΠΑΝΑΛΗΨΗΣ

L(k)=(1-0.5)*L(k)+0.5*makespan3(M,N);%Ενημέρωση της λουσιφερίνης

      ΓΙΑ j από 1 μέχρι N
       Voithitiko(1,j)=0;
      ΤΕΛΟΣ ΕΠΑΝΑΛΗΨΗΣ
    max=-10000000000;
  ΤΕΛΟΣ ΕΠΑΝΑΛΗΨΗΣ
ΤΕΛΟΣ ΕΠΑΝΑΛΗΨΗΣ
```

Με σκοπό την περεταίρω βελτιστοποίηση των αποτελεσμάτων που έχουν βρεθεί μέχρι στιγμής, εφαρμόστηκε τοπική αναζήτηση. Έτσι με την εντολή ***randi(N)*** επιλέχθηκαν τυχαία δύο εργασίες, στις οποίες έγινε ανταλλαγή μέσα στον πίνακα που ήταν αποθηκευμένη η σειρά εργασιών, αφού ελέχθηκε ότι οι δύο εργασίες δεν είναι οι ίδιες. Έτσι δημιουργηθηκε μία νέα σειρά εργασιών για την οποία υπολογίστηκε ξανά ο συνολικός χρόνος περάτωσης των εργασιών από τις μηχανές. Η διαδικασία της τοπικής αναζήτησης έγινε δέκα φορές για κάθε σειρά εργασιών που είχε ο πίνακας που τις έχει αποθηκευμένες και όταν βρισκόταν καλύτερος χρόνος, τότε έπαιρνε την θέση των παλιών αποτελεσμάτων στους αρχικούς πίνακες. Τέλος, γίνεται η ενημέρωση της λουσιφερίνης ξανά με χρήση της σχέσης (4) και η αρχικοποίηση κάποιων μεταβλητών και πινάκων (μηδενισμός), ώστε να χρησιμοποιηθούν στην επόμενη επανάληψη. Η όλη διαδικασία επαναλαμβάνεται εκατό φορές με σκοπό τον υπολογισμό του συνολικού χρόνου σε όσες το δυνατόν περισσότερες σειρές εργασιών, ώστε να πλησιάσουμε όσο είναι δυνατόν στο βέλτιστο αποτέλεσμα για το συγκεκριμένο πρόβλημα.

2.2.5: Εμφάνιση Τελικού Αποτελέσματος

SunolikosXronos=Veltistos_Xronos(1);

ΓΙΑ *i από 2 μέχρι AL*

ΑΝ *(Veltistos_Xronos(i)<SunolikosXronos)*

SunolikosXronos=Veltistos_Xronos(i);

ΓΙΑ *j από 1 μέχρι N*

VeltistiDiadromi(j)=SeiraErgasiwn(i,j);

ΤΕΛΟΣ ΕΠΑΝΑΛΗΨΗΣ

ΤΕΛΟΣ ΑΝ

ΤΕΛΟΣ ΕΠΑΝΑΛΗΨΗΣ

ΤΥΠΩΣΕ *VeltistiDiadromi*

ΤΥΠΩΣΕ *SunolikosXronos*

Στο τελευταίο κομμάτι του προγράμματος, έχουμε έναν πίνακα ο οποίος έχει τόσες γραμμές όσες και οι αρχικές λύσεις που είχαμε επιλέξει να δημιουργήσουμε και σε κάθε γραμμή έχει μία διαφορετική σειρά εργασιών. Επίσης, έχουμε άλλον ένα πίνακα με τους αντίστοιχους συνολικούς χρόνους

περάτωσης όλων των εργασιών από όλες τις μηχανές για κάθε μία σειρά εργασιών. Για να βγάλουμε το καλύτερο αποτέλεσμα θέτουμε σαν βέλτιστο χρόνο, αυτό που έχει η πρώτη σειρά εργασιών και αρχίζουμε να τον συγκρίνουμε με τους υπόλοιπους. Κάθε φορά που βρίσκουμε κάποιον καλύτερο, τον αντικαθιστούμε στον βέλτιστο και αποθηκεύουμε και την αντίστοιχη σειρά εργασιών. Έτσι, τελειώνοντας από τις συγκρίσεις έχουμε το καλύτερο αποτέλεσμα που υπολογίστηκε απ' όλη αυτή την διαδικασία, το οποίο και εμφανίζουμε μαζί με την αντίστοιχη σειρά εργασιών που αντιπροσωπεύει αυτόν τον χρόνο.

2.3: Παρουσίαση Αποτελεσμάτων

Είδος Προβλήματος	Πρόβλημα	Τρέξιμο Νο 1	Τρέξιμο Νο 2	Τρέξιμο Νο 3	Μ.Ο	Ανώτερο Όριο	Απόκλιση	Σειρά Εργασιών
20x5	Tai001	1359	1367	1372	1366	1278	0,0634	9 8 10 1 2 14 6 15 17 7 19 16 11 4 5 13 3 12 18 20
20x5	Tai002	1403	1433	1431	1422,333	1359	0,0324	7 9 3 12 6 15 14 17 1 10 8 13 19 20 18 16 4 2 11 5
20x5	Tai003	1254	1304	1262	1273,333	1081	0,1600	16 5 13 3 11 7 18 1 6 10 12 19 15 2 8 14 9 4 20 17
20x5	Tai004	1480	1491	1449	1473,333	1293	0,1206	16 15 1 11 2 12 17 4 9 7 6 13 10 19 20 18 8 3 5 14
20x5	Tai005	1377	1377	1392	1382	1235	0,1150	16 18 3 15 19 9 7 14 8 4 5 12 2 13 6

								10 17 11 1 20
20x5	Tai006	1368	1398	1333	1366,333	1195	0,1155	17 1 13 6 9 11 7 16 18 12 4 10 3 5 15 20 2 8 14 19
20x5	Tai007	1390	1306	1342	1346	1239	0,0541	5 20 8 18 17 14 3 16 11 1 19 6 2 12 13 10 15 7 4 9
20x5	Tai008	1401	1369	1401	1390,333	1206	0,1352	14 17 4 6 19 16 10 5 20 9 1 13 3 8 2 12 11 15 18 7
20x5	Tai009	1447	1449	1411	1435,667	1230	0,1472	4 3 11 6 8 7 17 18 5 15 12 1 14 2 19 9 10 20 16 13
20x5	Tai010	1303	1220	1288	1270,333	1108	0,1011	5 16 11 12 1 10 3 14 8 6 17 15 7 13 19 20 18 2 4 9
20x10	Tai011	1846	1861	1889	1865,333	1582	0,1669	18 6 7 20 12 14 9 8 3 2 13 4 5 19 17 11 1 15 10 16
20x10	Tai012	1917	1909	1861	1895,667	1659	0,1218	17 12 11 7 5 9 10 1 19 13 14 6 3 20 15 2 18 4 16 8
20x10	Tai013	1778	1715	1764	1752,333	1496	0,1464	4 9 6 10 2 15 16 14 13 12 1 7 11 5

								3 8 17 19 18 20
20x10	Tai014	1667	1592	1637	1632	1377	0,1561	1 3 9 11 15 13 6 7 20 18 10 16 2 14 8 12 4 19 17 5
20x10	Tai015	1712	1652	1732	1698,667	1419	0,1642	8 12 20 9 13 14 5 10 16 6 11 1 18 2 3 15 17 4 19 7
20x10	Tai016	1692	1678	1606	1658,667	1397	0,1496	8 5 9 19 3 6 14 11 15 13 16 18 4 7 12 20 1 10 17 2
20x10	Tai017	1729	1765	1715	1736,333	1484	0,1557	19 7 9 12 4 5 10 18 6 14 2 1 3 16 15 20 8 13 11 17
20x10	Tai018	1794	1811	1788	1797,667	1538	0,1625	20 7 14 17 8 9 5 18 15 16 3 10 6 19 13 4 2 1 12 11
20x10	Tai019	1819	1822	1793	1811,333	1593	0,1255	2 14 6 11 7 20 19 1 17 5 16 4 18 8 12 3 13 15 10 9
20x10	Tai020	1794	1841	1869	1834,667	1591	0,1276	13 14 5 6 18 8 10 15 4 9 16 12 2 7 20 17 3 19 11 1
20x20	Tai021	2557	2617	2585	2586,333	2297	0,1132	18 6 4 16 9 20 19 13 10 14 11 12 2 3

								7 5 17 15 1 8
20x20	Tai022	2371	2442	2417	2410	2099	0,1296	7 13 20 18 11 8 14 10 1 4 12 16 3 2 5 19 15 9 17 6
20x20	Tai023	2612	2596	2613	2607	2326	0,1161	12 15 4 5 18 2 1 19 7 20 6 14 3 13 9 11 16 10 17 8
20x20	Tai024	2575	2536	2521	2544	2223	0,1341	4 6 8 17 2 4 15 1 20 18 12 7 5 14 11 3 10 17 13 9
20x20	Tai025	2612	2667	2631	2636,667	2291	0,1401	16 5 17 15 10 12 3 18 19 11 14 9 1 4 2 6 7 13 20 8
20x20	Tai026	2486	2527	2549	2520,667	2226	0,1168	4 14 1 19 15 18 17 13 9 6 8 3 2 11 7 10 5 12 20 16
20x20	Tai027	2558	2596	2551	2568,333	2273	0,1223	5 17 4 19 12 9 1 11 6 7 18 16 8 10 14 2 3 13 20 15
20x20	Tai028	2524	2520	2509	2517,667	2200	0,1405	3 2 20 5 7 8 14 15 12 10 16 11 17 4 6 19 1 9 13 18
20x20	Tai029	2536	2577	2576	2563	2237	0,1337	5 14 3 2 7 8 13 10 11 9 17 16 6 4 12

								19 18 1 15 20
20x20	Tai030	2523	2534	2517	2524,667	2178	0,1556	3 7 6 1 17 5 4 19 16 11 2 8 20 9 13 14 15 10 18 22
50x5	Tai031	2960	2956	2971	2962,333	2724	0,0852	
50x5	Tai032	3084	3096	3131	3103,667	2834	0,0882	
50x5	Tai033	2870	2860	2852	2860,667	2621	0,0881	
50x5	Tai034	3091	2969	3018	3026	2751	0,0792	
50x5	Tai035	3078	3047	3086	3070,333	2863	0,0643	
50x5	Tai036	3119	3138	3121	3126	2829	0,1025	
50x5	Tai037	2973	3020	3019	3004	2725	0,0910	
50x5	Tai038	2961	3017	2959	2979	2683	0,1029	
50x5	Tai039	2816	2866	2823	2835	2552	0,1034	
50x5	Tai040	3065	3052	3092	3069,667	2782	0,0971	
50x10	Tai041	3640	3532	3626	3599,333	2991	0,1809	
50x10	Tai042	3540	3482	3417	3479,667	2867	0,1918	

50x10	Tai043	3401	3477	3466	3448	2839	0,1980	
50x10	Tai044	3523	3572	3556	3550,333	3063	0,1502	
50x10	Tai045	3605	3551	3638	3598	2976	0,1932	
50x10	Tai046	3601	3570	3609	3593,333	3006	0,1876	
50x10	Tai047	3692	3575	3553	3606,667	3093	0,1487	
50x10	Tai048	3552	3543	3595	3563,333	3037	0,1666	
50x10	Tai049	3434	3553	3487	3491,333	2897	0,1854	
50x10	Tai050	3570	3574	3562	3568,667	3065	0,1622	
50x20	Tai051	4563	4660	4594	4605,667	3850	0,1852	
50x20	Tai052	4558	4426	4530	4504,667	3704	0,1949	
50x20	Tai053	4366	4377	4490	4411	3640	0,1995	
50x20	Tai054	4482	4510	4516	4502,667	3720	0,2048	
50x20	Tai055	4532	4466	4485	4494,333	3610	0,2371	
50x20	Tai056	4448	4431	4494	4457,667	3681	0,2037	
50x20	Tai057	4498	4510	4588	4532	3704	0,2144	

50x20	Tai058	4452	4435	4461	4449,333	3691	0,2016	
50x20	Tai059	4531	4547	4469	4515,667	3743	0,1940	
50x20	Tai060	4646	4498	4642	4595,333	3756	0,1976	
100x5	Tai061	5840	5862	5853	5851,667	5493	0,0632	
100x5	Tai062	5689	5661	5689	5679,667	5268	0,0746	
100x5	Tai063	5566	5619	5572	5585,667	5175	0,0756	
100x5	Tai064	5464	5339	5348	5383,667	5014	0,0648	
100x5	Tai065	5702	5628	5664	5664,667	5250	0,0720	
100x5	Tai066	5469	5520	5447	5478,667	5135	0,0750	
100x5	Tai067	5496	5702	5645	5614,333	5246	0,0477	
100x5	Tai068	5392	5491	5530	5471	5094	0,0585	
100x5	Tai069	5896	5904	5878	5892,667	5448	0,0789	
100x5	Tai070	5833	5781	5663	5759	5322	0,0641	
100x10	Tai071	6587	6681	6620	6629,333	5770	0,1416	
100x10	Tai072	6283	6246	6296	6275	5349	0,1677	

100x10	Tai073	6393	6346	6425	6388	5676	0,1180	
100x10	Tai074	6628	6688	6701	6672,333	5781	0,1465	
100x10	Tai075	6342	6267	6377	6328,667	5467	0,1463	
100x10	Tai076	6157	6219	6065	6147	5303	0,1437	
100x10	Tai077	6347	6400	6325	6357,333	5595	0,1305	
100x10	Tai078	6246	6378	6342	6322	5617	0,1120	
100x10	Tai079	6607	6623	6605	6611,667	5871	0,1250	
100x10	Tai080	6554	6509	6611	6558	5845	0,1136	
100x20	Tai081	7521	7485	7511	7505,667	6202	0,2069	
100x20	Tai082	7540	7567	7362	7489,667	6183	0,1907	
100x20	Tai083	7568	7474	7534	7525,333	6271	0,1918	
100x20	Tai084	7556	7494	7444	7498	6269	0,1874	
100x20	Tai085	7557	7573	7474	7534,667	6314	0,1837	
100x20	Tai086	7669	7555	7587	7603,667	6364	0,1871	
100x20	Tai087	7588	7635	7566	7596,333	6268	0,2071	

100x20	Tai088	7562	7724	7798	7694,667	6401	0,1814	
100x20	Tai089	7545	7550	7649	7581,333	6275	0,2024	
100x20	Tai090	7620	7786	7626	7677,333	6434	0,1843	
200x10	Tai091	12015	11774	11909	11899,33	10862	0,0840	
200x10	Tai092	11659	11793	11663	11705	10480	0,1125	
200x10	Tai093	11980	12020	12136	12045,33	10922	0,0969	
200x10	Tai094	11874	11744	11810	11809,33	10889	0,0785	
200x10	Tai095	11957	11691	11641	11763	10524	0,1061	
200x10	Tai096	11554	11600	11526	11560	10329	0,1159	
200x10	Tai097	11981	11992	11912	11961,67	10854	0,0975	
200x10	Tai098	11951	11931	12040	11974	10730	0,1119	
200x10	Tai099	11655	11814	11862	11777	10438	0,1166	
200x10	Tai100	11959	11768	11800	11842,33	10675	0,1024	
200x20	Tai101	13157	13136	13261	13184,67	11195	0,1734	
200x20	Tai102	13184	13381	13356	13307	11203	0,1768	

200x20	Tai103	13354	13287	13451	13364	11281	0,1778	
200x20	Tai104	13186	13324	13291	13267	11275	0,1695	
200x20	Tai105	13182	13262	13338	13260,67	11259	0,1708	
200x20	Tai106	13214	13324	13294	13277,33	11176	0,1824	
200x20	Tai107	13469	13502	13506	13492,33	11360	0,1857	
200x20	Tai108	13310	13275	13414	13333	11334	0,1713	
200x20	Tai109	13045	13215	13371	13210,33	11192	0,1656	
200x20	Tai110	13310	13509	13362	13393,67	11288	0,1791	
500x20	Tai111	29622	29511	29676	29603	26059	0,1325	
500x20	Tai112	30053	30163	30028	30081,33	26520	0,1323	
500x20	Tai113	29792	29649	29829	29756,67	26371	0,1243	
500x20	Tai114	29676	29904	29900	29826,67	26456	0,1217	
500x20	Tai115	29872	29911	29923	29902	26334	0,1344	
500x20	Tai116	30052	30271	29907	30076,67	26477	0,1295	
500x20	Tai117	29698	29703	29666	29689	26389	0,1242	

500x20	Tai118	29989	29883	29834	29902	26560	0,1233	
500x20	Tai119	29635	29706	29553	29631,33	26005	0,1364	
500x20	Tai120	29904	29812	29884	29866,67	26457	0,1268	

2.4: Συμπεράσματα και παρατηρήσεις

Τα συμπεράσματα και οι παρατηρήσεις προκύπτουν εκτέλεση του αλγορίθμου και την εξαγωγή των αποτελεσμάτων είναι τα εξής:

- Οι χρόνοι εκτέλεσης του αλγορίθμου ήταν πολύ μικροί της τάξεως των δευτερολέπτων. Ακόμη και στα προβλήματα που είχαμε 500 εργασίες και 20 μηχανές ο χρόνος εκτέλεσης παρέμενε στα 30-40 sec.
- Τα αποτελέσματα που προέκυψαν έπειτα από τρεις εκτελέσης του αλγορίθμου για το ίδιο πρόβλημα, ήταν στα ίδια επίπεδα χωρίς μεγάλες αποκλίσεις, πράγμα που δείχνει την σταθερότητα του αλγορίθμου στα αποτελέσματα.
- Σε κανένα πρόβλημα, αν και είχαμε σχετικά καλά αποτελέσματα, ο αλγόριθμος δεν βρήκε το βέλτστο αποτέλεσμα βάση του Taillard. Στα περισσότερα προβλήματα είχαμε μία απόκλιση της τάξης 12%-19%, κυρίως στα μεγάλα προβλήματα από εκατό εργασίες και πάνω. Ενώ σε λίγες περιπτώσεις είχαμε μικρότερες αποκλίσεις της τάξεως 3%-9%, σε προβλήματα με είκοσι και πενήντα εργασίες.
- Κύριο συμπέρασμα είναι ότι ο αλγόριθμος βελτιστοποίησης σμήνους πυγολαμπίδων λειτούργησε ικανοποιητικά σε γενικές γραμμές. Για την βελτίωση των αποτελεσμάτων,πιθανόν, θα μπορούσαμε να χρησιμοπιήσουμε περισσότερες αρχικές λύσεις στις οποίες θα εφαρμόζαμε τον αλγόριθμο, πράγμα που δικαιολογείται, εφόσον για κάθε πρόβλημα με N μηχανές υπάρχουν N! διαφορετικές λύσεις ή κάποιον άλλο πιο αποδοτικό αλγόριθμο για τα προβλήματα flow shop.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1]: Μαρινάκης, Ι., & Μυγδαλάς, Α. «Σχεδιασμός και βελτιστοποίηση της εφοδιαστικής αλυσίδας» Θεσσαλονίκη, Εκδόσεις Σοφία (2008)

[2]: Knuth D.E. «A terminological proposal» ACM SIGACT News, 6 (1974) 12-18

[3]: GENETIC ALGORITHMS FOR SHOP SCHEDULING PROBLEMS: A SURVEY Frank Werner∗ Otto-von-Guericke-Universität, Fakultät für Mathematik, 39106 Magdeburg, Germany

[4]: Scheduling of 2-operation jobs on a single machine to minimize the number of tardy jobs Radhika M. Yeleswarapu University of South Florida

[5]: Aeysha Shahzad «A single machine scheduling problem with individual job tardiness based objectives» LINA, University of Nantes

[6]: «Χρονοπρογραμματισμός και έλεγχος παραγωγής» Σημειώσεις μαθήματος με τίτλο Διοίκηση Παραγωγής και Συστημάτων Υπηρεσιών, Σχολή ΗΜΜΥ του Ε.Μ.Π., Αθήνα

[7]: Johnson, S.M. «Optimal two-and-three stage production schedules with setup times included» Naval Research Logistics Quarterly, 1 (1954) 61–68.

[8]: Graham R. L., Lawler E. L., Lenstra J. K., and Rinnooy Kan. A.H.G. «Optimization and approximation in deterministic sequencing and scheduling: a survey.» In Annals of Discrete Mathematics, 5 (1979) 287-326

[9]: <<Γενετικοί Αλγόριθμοι και Εφαρμογές>>, Σημειώσεις εργαστηρίου, τμήμα μηχανικών ηλεκτρονικών υπολογιστών και πληροφορικής,Πατρα 1999

[10]: << ΝΟΗΜΟΣΥΝΗ ΣΜΗΝΟΥΣ ΜΕΛΕΤΗ ΑΞΙΟΛΟΓΗΣΗ & ΥΛΟΠΟΙΗΣΗ ΑΛΓΟΡΙΘΜΩΝ ΓΙΑ ΤΟΝ ΕΛΕΓΧΟ ΜΗ ΕΠΑΝΔΡΩΜΕΝΩΝ>> Πτυχιακή εργασία ΚΑΨΟΥΛΗΣ ΙΩΣΗΦ, ΜΑΝΩΛΟΠΟΥΛΟΣ ΧΡΗΣΤΟΣ, ΜΕΛΚΩΝΗΣ ΑΝΤΩΝΗΣ,ΙΟΥΝΙΟΣ 2016

[11]: Yannis Marinakis, Magdalene Marinaki <<Particle swarm optimization with expanding neighborhood topology for the permutation flowshop scheduling problem>>

[12]: Nurezayana Zainal,Azlan Mohd Zain,Nor Haizan Mohamed Radzi,Muhamad Razib Othman (2016) <<Glowworm swarm optimization (GSO) for optimization of machining parameters>>

[13]: Yannis Marinakis, Magdalene Marinaki (2015) <<A Glowworm Swarm Optimization algorithm for the Vehicle Routing Problem with Stochastic Demands

[14]: Ιωάννης Μαρινάκης, Μαγδαληνή Μαρινάκη, Νικόλαος Φ. Ματσατσίνης, Κωνσταντίνος Ζοπουνίδης (2011) << Μεθευρετικοί και εξελικτικοί αλγόριθμοι σε προβληματα διοικητικής επιστήμης>> Αθήνα:Κλειδάριθμος