Πολυτεχνείο Κρήτης
Σχολή Μηχανικών Παραγωγής και Διοίκησης

Αλγόριθμος βελτιστοποίησης ζευγαρώματος μελισσών για την επίλυση του προβλήματος χρονοπρογραμματισμού εργασιών.

Διπλωματική Εργασία
Κουρομιχελάκης Γεώργιος

Επιβλέπων
Ιωάννης Μαρινάκης

Χανιά, Οκτώβριος 2022

Ευχαριστίες

Αρχικά, θα ήθελα να ευχαριστήσω τον κ. Μαρινάκη για την συνεργασία και την βοήθεια που μου παρείχε σε όλο το διάστημα εκπόνησης της διπλωματικής μου εργασίας. Επίσης, θα ήθελα να ευχαριστήσω την οικογένειά μου η οποία αποτέλεσε την πιο σημαντική πηγή στήριξης μου όλα αυτά τα χρόνια. Τέλος, μαζί με τα εφόδια που αποκόμισα κέρδισα και μοναδικούς φίλους που αποτέλεσαν ένα σημαντικό κεφάλαιο στήριξης όλα αυτά τα χρόνια.

Περίληψη

Τα προβλήματα χρονοπρογραμματισμού εργασιών, αφορούν προβλήματα που χρήζουν εύρεσης της βέλτιστης-ελάχιστης χρονικής διάρκειας για την τέλεση μίας σειράς διεργασιών n από έναν αριθμό μηχανών m. Στα προβλήματα αυτά, πρέπει όλες οι διεργασίες να περνάνε με την ίδια σειρά από τις μηχανές και καμία από τις μηχανές δεν μπορεί να εκτελεί παραπάνω από μία διεργασία ταυτόχρονα ή να ξεκινήσει μία καινούργια πριν ολοκληρωθεί η προηγούμενη. Προβλήματα σαν αυτά, είναι αντικείμενο μελέτης στον τομέα της εφοδιαστικής αλυσίδας στην πραγματική παραγωγή. Η εξέλιξη και η ανάπτυξη ευρετικών αλγορίθμων με την βοήθεια τεχνητής νοημοσύνης για την επίλυση των προβλημάτων χρονοπρογραμματισμού εργασιών είναι το αντικείμενο μελέτης της συγκεκριμένης εργασίας. Ο αλγόριθμος βελτιστοποίησης που θα αναπτυχθεί είναι ένας ευρετικός αλγόριθμος, εμπνευσμένος από την φύση και πιο συγκεκριμένα από τον τρόπο που ζευγαρώνει η βασίλισσα των μελισσών. Ο συγκεκριμένος αλγόριθμος, μιμείται ουσιαστικά την διαδικασία μέσω της οποίας η βασίλισσα η οποία αποτελεί την αρχική βέλτιστη λύση ξεκινάει την αναπαραγωγική της πτήση, κατά τη διάρκεια της οποίας θα δεσμεύσει στην σπερματοθήκη της, σπέρμα από τους κηφήνες με τους οποίους θα αναπαραχθεί. Όταν η ενέργεια της βασίλισσας μέλισσας πέσει χαμηλότερα από το ενεργειακό της κατώφλι, η βασίλισσα επιστρέφει στην κυψέλη και ξεκινάει την δημιουργία απογόνων- πιθανών λύσεων. Όπως και στην πραγματικότητα που η μέλισσα μπορεί να παρέμβει στον γενότυπο του κάθε απόγονου με προοπτική την ενδυνάμωση του, έτσι και ο αλγόριθμος που θα αναπτύξω θα μπορεί να βελτιώσει την ποιότητα του απόγονου με την προοπτική να μπορεί να κάνει καλύτερη τοπική αναζήτηση και άρα να δίνει πιο ανταγωνιστικές λύσεις. Τέλος, ο αλγόριθμος θα δέχεται την ύπαρξη μίας και μόνο βασίλισσας-βέλτιστης λύσης, αντικαθιστώντας την παλιά με την νέα βασίλισσα όπως γίνεται και στις πραγματικές κυψέλες που μπορεί να υπάρχει μία μονάχα βασίλισσα.

Λέξεις Κλειδιά

Αλγόριθμος Βελτιστοποίησης Ζευγαρώματος Μελισσών, Πρόβλημα Χρονοπρογραμματισμού Εργασιών, Nawaz-Enscore-Ham, Αλγόριθμοι Νοημοσύνης Σμήνους.

Περιεχόμενα

1. Εισαγωγή

Στο σύγχρονο παγκοσμιοποιημένο σύστημα παραγωγής οι βιομηχανίες και οι παραγωγικές αλυσίδες κάνουν όλο και πιο εκτεταμένη υπολογιστική έρευνα πάνω στην εφοδιαστική αλυσίδα καθώς και στον χρονοπρογραμματισμό των εργασιών τους. Με σκοπό την βελτιστοποίηση ως προς το κόστος και την ταχύτητα της παραγωγικής διαδικασίας, έχουν αναπτυχθεί προβλήματα που προσομοιάζουν την παραγωγική διαδικασία και τις απαιτήσεις της.

Στην παρούσα εργασία παρουσιάζονται τα αποτελέσματα αλγορίθμου εμπνευσμένου από την φύση για την επίλυση του προβλήματος χρονοπρογραμματισμού εργασιών. Οι όλο και αυξανόμενες ανάγκες στην παραγωγή καθώς και η ανάγκη της εύρεσης της φτηνότερης άρα και συντομότερης παραγωγικής διαδικασίας, καθιστά τα σύγχρονα προβλήματα βελτιστοποίησης την αιχμή του δόρατος για τις ανάγκες των ιθύνοντων αλλά και των εργατών στην παραγωγική διαδικασία.

Ο αλγόριθμος που επιλέχθηκε για την επίλυση του προβλήματος χρονοπρογραμματισμού είναι μία σύγχρονη προσέγγιση στην επίλυση των προβλημάτων βελτιστοποίησης. Βασίζεται στην νοημοσύνη σμήνους και μιμείται διεργασίες οντοτήτων της φύσης ανάλογα με τα χαρακτηριστικά του προβλήματος που σχεδιάστηκε για να επιλύσει. Οι δύο βασικές κατηγορίες αλγορίθμων εμπνευσμένων από την φύση είναι οι αλγόριθμοι που προσομοιάζουν την συλλογή τροφής και αυτοί που προσομοιάζουν την αναπαραγωγή και την δημιουργία απογόνων.

Στο πρώτο κεφάλαιο της εργασίας θα παρουσιαστεί το πρόβλημα χρονοπρογραμματισμού εργασιών καθώς και τα χαρακτηριστικά του. Ακόμα θα παρουσιαστεί και η μαθηματική μοντελοποίηση του προβλήματος. Στο δεύτερο κεφάλαιο θα παρουσιαστούν ο γενετικός αλγόριθμος και οι μεταευρετικοί αλγόριθμοι που εφαρμόσθηκαν καθώς και πληροφορίες σχετικά με τον τρόπο εφαρμογής τους. Στο επόμενο κεφάλαιο έγινε παρουσίαση της μεθόδου τοπικής αναζήτησης που χρησιμοποιήθηκε για τη βελτίωση των αποτελεσμάτων που παρήχθησαν από την βασική εφαρμογή των αλγορίθμων. Τέλος, στο τελευταίο κεφάλαιο έγινε παρουσίαση των τελικών αποτελεσμάτων και σύγκριση των σχετικών αποτελεσμάτων με τα αντίστοιχα της βιβλιογραφίας, καθώς και μια απεικόνιση των αποκλίσεων των αποτελεσμάτων που βρέθηκαν σε σχέση με τα πρότυπα.

2. Διαχείριση εφοδιαστικής αλυσίδας

Ο κλάδος της διαχείρισης της εφοδιαστικής αλυσίδας είναι ο επιστημονικός κλάδος που ασχολείται με την μελέτη τον σχεδιασμό και την ανάλυση όλων των ενεργειών που αφορούν αρχικά την προμήθεια και την διαχείριση των πόρων στην αποθήκη, στη συνέχεια την παραγωγή μελετώντας οικονομοτεχνικά την σχέση κόστους απόδοσης, και τέλος στο κομμάτι της διανομής στο πελάτη. Ο αντικειμενικός σκοπός του κλάδου αυτού είναι η ελαχιστοποίηση του κόστους παραγωγής χωρίς να επηρεαστεί η ποιότητα και συγχρόνως η άρτια διαχείριση της αποθήκης καθώς και η έγκαιρη διανομή του προϊόντος στους πελάτες. Όλα αυτά επιτυγχάνονται με τον μέγιστο σεβασμό στον εργαζόμενο αλλά και στον πελάτη, καθώς και με συνεχή επιμόρφωση πάνω στις νέες τεχνολογίες και ανάγκες της εκάστοτε εποχής. Ο κλάδος που περιγράφεται παραπάνω ενέχεται στον όρο Logistics.

2.1 Τα logistics στην παραγωγή

Όπως περιεγράφηκε και παραπάνω ο κλάδος των logistics έχει εξέχουσα θέση σε όλα τα στάδια του σχεδιασμού, παραγωγής και διακίνησης προϊόντων, αφού έχει εποπτικό και βελτιωτικό ρόλο σε μία σειρά πραγμάτων που θα αναφερθούν παρακάτω.

- Κατανόηση απαιτήσεων: στο στάδιο αυτό το τμήμα των logistics είναι το υπεύθυνο για την κατανόηση των απαιτήσεων που έχει ο πελάτης ή η γραμμή παραγωγής έτσι ώστε να μπορεί να γίνει ακριβής μελέτη για τον υπολογισμό των πόρων που θα χρειαστούν.
- Σχεδιομέλετη: στο στάδιο αυτό γίνεται ο σχεδιασμός του προϊόντος έτσι ώστε να γίνουν απόλυτα κατανοητές οι ανάγκες και οι προδιαγραφές του. Στη συνέχεια μέσω δοκιμών γίνεται η τεχνοοικονομική μελέτη υλικών και τελικά γίνεται η συνολική αξιολόγηση του παραγόμενου αγαθού.
- Εφοδιασμός: στο στάδιο αυτό το τμήμα των logistics θα φροντίσει να παραλάβει τον κατάλληλο εξοπλισμό και πόρους για την παραγωγή στην βέλτιστη τιμή και στον καλύτερο χρονικό ορίζοντα. Στο στάδιο αυτό, επιπλέον θα πρέπει να γίνει η απαραίτητη επιμόρφωση του προσωπικού καθώς και προσλήψεις στα πόστα που είναι αναγκαίο.
- Κάλυψη αναγκών και συντήρηση: στο συγκεκριμένο στάδιο θα γίνει κάλυψη τον αναγκών σε εξοπλισμό, υλικά και πόρους στους

εργάτες καθώς και συντήρηση στις μηχανές αλλά και επικαιροποίηση στις συμφωνίες μεταξύ της γραμμής παραγωγής και της γραμμής ανεφοδιασμού.

3. Χρονοπρογραμματισμός εργασιών

3.1 Κατηγορίες διεργασιών

- Συνεχείς/Διακριτές: Συνεχείς είναι οι διεργασίες οι οποίες λειτουργούν σε συνεχή χρονική κλίμακα όπως είναι για παράδειγμα η εξόρυξη του πετρελαίου ή μία μονάδα επεξεργασίας λυμάτων, ενώ διακριτές εννοούμε τις διεργασίες οι οποίες απαιτούν την δημιουργία κάποιου αντικειμένου (έπιπλα, υπολογιστές).
- Χρονικά εξαρτημένες/Ανεξάρτητες: Χρονικά εξαρτημένες είναι οι διεργασίες οι οποίες έχουν χρονικούς περιορισμούς όπως η ύπαρξη βάρδιας ή το ωράριο λειτουργίας μιας βιομηχανίας ή ακόμα και η ανάγκη για την λήξη μιας προηγούμενης διεργασίας για την έναρξη της επόμενης. Ανεξάρτητες είναι οι διεργασίες που δεν έχουν χρονικούς περιορισμούς.
- Χωρικά εξαρτημένες/Ανεξάρτητες: Χωρικά εξαρτημένες είναι οι διεργασίες που μπορούν να γίνουν αυστηρά σε συγκεκριμένο χώρο ή περιβάλλον (νοσοκομεία, μονάδες παραγωγής), ενώ οι χωρικά ανεξάρτητες είναι οι διεργασίες που δεν είναι απαραίτητη η εκτέλεση τους σε κάποιο συγκεκριμένο χώρο ή περιβάλλον (προγραμματισμός, διαφήμιση).
- Διακεκομμένες/Συνεχείς: Διακεκομμένες είναι οι διεργασίες που μπορούν να διακοπούν κατά την διάρκεια εκτέλεσης τους με δυνατότητα περάτωσης τους σε δεύτερο χρόνο (επισκευές μηχανών, προγραμματισμός), ενώ συνεχείς είναι όλες αυτές οι διεργασίες που δεν μπορούν να διακοπούν μέχρι την περάτωση τους (ενοικίαση σπιτιού).
- Εφάπαξ/ Επαναλαμβανόμενες: Εφάπαξ είναι οι διεργασίες οι οποίες γίνονται μία φορά και δεν υπάρχει δυνατότητα να επαναληφθούν (υπογραφή μηχανικού πριν το έργο, ένα σημαντικό χειρουργείο), ενώ επαναλαμβανόμενες είναι οι διεργασίες οι οποίες βελτιώνονται μέσω ανάδρασης περιοδικά. Οι διεργασίες αυτές κατά το πέρας της εκάστοτε περιόδου επιστρέφουν στην αρχική τους κατάσταση και ξεκινούν εκ νέου έναν καινούργιο κύκλο (συντήρηση μηχανών).

3.2 Το πρόβλημα χρονοπρογραμματισμού εργασιών

Το Πρόβλημα του χρονοπρογραματισμού εργασιών (Flowshop scheduling problem) παρουσιάστηκε για πρώτη φορά από τον Johnson το 1954 χωρίς να χρησιμοποιηθεί ακόμα ο όρος «χρονοπρογραμματισμός» αλλά περιγράφοντας ένα υπαρκτό πρόβλημα παραγωγής της εποχής του που αφορούσε τρείς μηχανές. Σήμερα αποτελεί ένα από τα πιο διαδεδομένα επιλύσιμα προβλήματα της παραγωγής με πολλές χιλιάδες δημοσιεύσεις και αναλύσεις. Η πρώτη φορά που συναντάται σε δημοσίευση το «Flowshop scheduling problem» είναι το 1960 από τον Heller και το 1965 από τους Ignall και Schrage.

Στα 60 περίπου χρόνια μελέτης του συγκεκριμένου προβλήματος έχουν υπάρξει πολλές και διάφορες προσπάθειες επίλυσης του, βασισμένες στα μέσα και στις τεχνικές της εκάστοτε εποχής. Η πρώτη προσπάθεια μαθηματικού προγραμματισμού έγινε από τον Wagner το 1959 όπου η έλλειψη ισχυρών υπολογιστικών μηχανών και προγραμμάτων, καθώς και η NP-δυσκολία του προβλήματος, αποτελούσαν την αιτία για την οποία ήταν αδύνατο να χρησιμοποιηθούν παραδείγματα με πολλές διεργασίες και μηχανές. Με την πάροδο του χρόνου καθώς και με την εξέλιξη της τεχνολογίας έγιναν απόπειρες επίλυσης του συγκεκριμένου προβλήματος με την βοήθεια γενετικών αλγορίθμων και τεχνητής νοημοσύνης, με παραδείγματα τους Brown, Scherer το 1995, και Ruiz, Maroto, Alcaraz το 2005 οι οποίοι δημιούργησαν ευρετικούς και μεταευρετικούς αλγόριθμους οι οποίοι έδιναν καλύτερα αποτελέσματα.

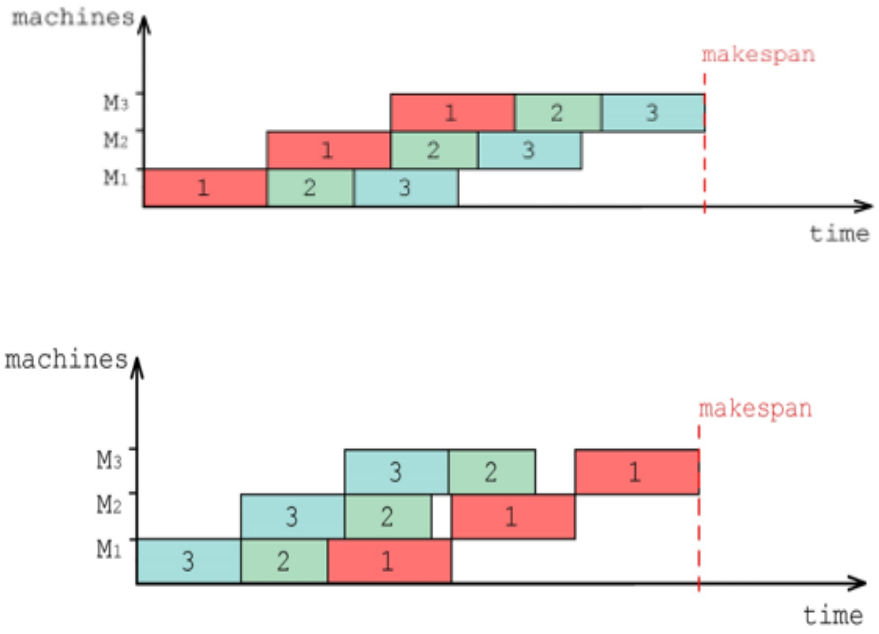
Ο βασικός στόχος του προβλήματος χρονοπρογραμματισμού εργασιών είναι η ελαχιστοποίηση του συνολικού χρόνου (makespan) που θα χρειαστεί για να περάσουν ένα σύνολο εργασιών n από ένα σύνολο μηχανών m. Με αυτόν τον τρόπο επιτυγχάνεται η όσο το δυνατόν περισσότερο αποτελεσματική χρήση μηχανών-εργατών καθώς και η ελαχιστοποίηση της αναμονής των πελατών και αποθήκης. Το πρόβλημα αυτό μετράει τον απαραίτητο χρόνο που θα χρειαστούν οι διαδικασίες για να περάσουν σειριακά από τις μηχανές, χωρίς δηλαδή να υπάρχει η δυνατότητα να ακολουθηθεί μια διαφορετική σειρά με την οποία θα εισάγονται στις μηχανές και χωρίς να μπορεί να εκτελέσει μία μηχανή παραπάνω από μία διεργασία.

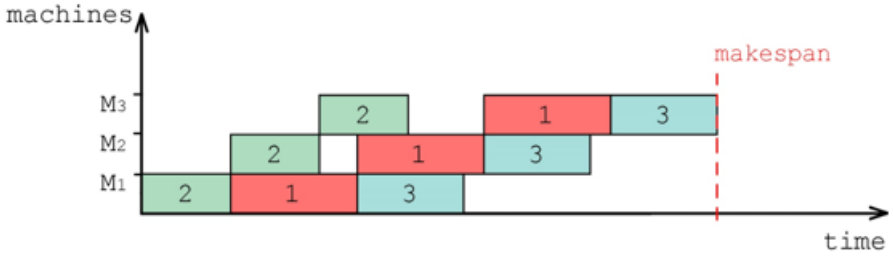
3.3 Χαρακτηριστικά του προβλήματος χρονοπρογραμματισμού εργασιών

Το πρόβλημα αντιμετάθεσης χρονικού προγραμματισμού εργασιών (Permutation Flow-Shop Scheduling Problem) όπως αναφέρθηκε είναι ένα πρόβλημα χρονικού προγραμματισμού N εργασιών σε M μηχανές. Πιο συγκεκριμένα το πρόβλημα αποτελείται από (i=1,2,…n) εργασίες και από (j=1,2,…m) μηχανές και σκοπός μας είναι να βρούμε τη βέλτιστη χρονική διάρκεια (makespan) που όλες οι εργασίες θα περάσουν από όλες τις μηχανές με την διαδικασία της αντιμετάθεσης εργασιών.

Κάθε i_n εργασία μπορεί να ξεκινήσει στην επόμενη J_m μηχανή αρκεί να έχει τελειώσει την διεργασία της στην J_{m-1} μηχανή και η νέα μηχανή να μην είναι δεσμευμένη από κάποια άλλη διεργασία. Ο εκάστοτε χρόνος κατεργασίας μιας διεργασίας από μία μηχανή συμβολίζεται ως $p_{i,j}$.

Στο πρόβλημα αντιμετάθεσης χρονικού προγραμματισμού εργασιών κάθε εργασία που περνάει από την οποιαδήποτε μηχανή στην $i_{οστη}$ θέση κατά σειρά, θα συνεχίσει να έχει τη θέση αυτή μέχρι να περάσει από όλες τις μηχανές.





Επομένως το πρόβλημα έχει τα εξής χαρακτηριστικά:

- Η σειρά των μηχανών από την οποία θα πρέπει να περάσουν οι διεργασίες είναι ορισμένη εξ αρχής (M1, M2,…Mn).
- Η αντιμετάθεση των εργασιών αποσκοπεί στην εύρεση της βέλτιστης σειράς με την οποία θα περάσουν οι διεργασίες από τις μηχανές στον μικρότερο χρόνο.
- Ο χρόνος που χρειάζεται η εκάστοτε διεργασία για να περάσει από κάθε μία από τις μηχανές είναι γνωστός.
- Καμία μηχανή δεν μπορεί να πραγματοποιήσει παραπάνω από μια διεργασία ταυτόχρονα.
- Ο συνολικός χρόνος (makespan) θα είναι η χρονική στιγμή που η τελευταία διεργασία θα περάσει από την τελευταία μηχανή.
- Οι χρόνοι εξάρμωσης των μηχανών είναι προσμετρημένοι στον χρόνο κατεργασίας της κάθε διεργασίας.

Τα προβλήματα χρονοπρογραμματισμού μπορούν να είναι από πολύ απλά με λίγες διεργασίες και μηχανές όπως στο παράδειγμα παραπάνω και να έχει δυνατότητα λύσης στο χαρτί, αλλά μπορεί να είναι μέχρι πολύ σύνθετο και να είναι απαραίτητη η χρήση υπολογιστικού συστήματος για την επίλυση τους. Για την επίλυση του στην εργασία αυτή αναπτύχθηκε ο αλγόριθμος ζευγαρώματος μελισσών (HBMO) ο οποίος είναι ένας γενετικός αλγόριθμος εμπνευσμένος από τη φύση και είναι μία πρωτοποριακή μέθοδος επίλυσης προβλημάτων Logistics.

4. Γενετικοί αλγόριθμοι και εφαρμογές στην μηχανική

Οι τεχνολογικές επαναστάσεις και τα νέα «προβλήματα» που γεννήθηκαν, δημιούργησαν την ανάγκη για ανεύρεση νέων μεθόδων για την επίλυση τους. Τα σύγχρονα προβλήματα βελτιστοποίησης όπως το (PFSSP) είναι ένα από τα χιλιάδες προβλήματα βελτιστοποίησης που

προσεγγίζονται όλο και περισσότερο με την βοήθεια τέτοιων αλγορίθμων. Η χρήση γενετικών αλγορίθμων έχει σαφή πλεονεκτήματα σε σχέση με παλαιότερες μεθόδους (γραμμικός, μη γραμμικός, δυναμικός προγραμματισμός), καθώς είναι πολύ πιο φτηνοί και σύντομοι και επιπλέον όπως έχει αποδειχθεί παρέχουν και καλύτερα αποτελέσματα.

Η μικρότερη αποτελεσματικότητα των μεθόδων αυτών, έγκειται στο γεγονός ότι οι κλασσικές μέθοδοι βελτιστοποίησης παραμετροποιημένων προβλημάτων αρκούνται στο πρώτο τοπικό ελάχιστο που θα βρουν στην «γειτονιά» γύρω από το αρχικό σημείο αναζήτησης λύσεων. Στα σύνθετα-σύγχρονα προβλήματα που τα τοπικά ελάχιστα είναι πάρα πολλά, μέθοδοι σαν αυτές μόνο τυχαία μπορούν να προσεγγίζουν την ολικά βέλτιστη λύση.

Οι σύγχρονες μέθοδοι επίλυσης τέτοιων προβλημάτων έχουν πλεονέκτημα απέναντι στις παλιές μαθηματικές μεθόδους καθώς έχουν την δυνατότητα υπακοής στους κανόνες και τους περιορισμούς του εκάστοτε προβλήματος, αλλά συγχρόνως ερευνούν λύσεις με τυχαιότητα, γεγονός που αφενός αυξάνει τις «γειτονίες» αναζήτησης λύσεων και αφετέρου φέρνει όλο και πιο κοντινά αποτελέσματα στα επιθυμητά. Οι γενετικοί αλγόριθμοι συνήθως μιμούνται φυσικά φαινόμενα, και προσομοιάζουν τα φαινόμενα αυτά με τις λύσεις του εκάστοτε προβλήματος.

4.1 Οι γενετικοί ως ευρετικοί αλγόριθμοι

Οι γενετικοί αλγόριθμοι ανήκουν στην κατηγορία των ευρετικών μεθόδων πλήρους αναζήτησης (global search heuristics) και σκοπός τους είναι η εύρεση όσο το δυνατόν πιο κοντινών λύσεων στην βέλτιστη. Δομούνται με την μοντελοποίηση του προβλήματος σε ένα πολύπλοκο πληθυσμό ζωντανών οργανισμών και μέσω κάποιον μιμήσεων των συνηθειών των ζωντανών οργανισμών αυτών, καταφέρνουν να αναζητούν όλο και καλύτερες λύσεις μέχρι να βρεθεί η βέλτιστη. Οι γενετικοί αλγόριθμοι λειτουργούν πάνω στους εξής τρεις μηχανισμούς:

- Αναπαραγωγή
- Διασταύρωση
- Μετάλλαξη

Σε αυτά τα στάδια ο αλγόριθμος αρχικά «αναπαράγεται» δίνοντας νέες πιθανές λύσεις τυχαία κατανεμημένες στον χώρο, στην συνέχεια μέσω

της διασταύρωσης δημιουργούνται απόγονοι-λύσεις με χαρακτηριστικά όμοια με τους προγόνους τους και τελικά γίνονται τυχαίες μεταλλάξεις μεταβάλλοντας τους υπάρχοντες συνδυασμούς αλληλουχιών που ήδη υπάρχουν.

Ο μηχανισμοί αυτοί σε συνδυασμό με την δυνατότητα των γενετικών αλγορίθμων να διαχωρίζουν τους «καλούς» από τους «κακούς» απογόνους, δίνουν όλο και πιο αξιόλογους γόνους-λύσεις αφού μπορούν να αξιολογήσουν τις αλληλουχίες των βέλτιστων χαρακτηριστικών από τις λιγότερο βέλτιστες.

Σε ένα πρόβλημα όπως του χρονοπρογραμματισμού εργασιών οι μηχανισμοί δημιουργίας λύσεων είναι οι εξής: αρχικά η τυχαία κατανομή των εργασιών σε ένα διάνυσμα (δημιουργία τυχαίων λύσεων), η πρόσμιξη κομματιών από τα διανύσματα των γονέων-λύσεων στον απόγονο-λύση (διασταύρωση) και η τυχαία παραλλαγή του διανύσματος κατά την δημιουργία νέας λύσης (μετάλλαξη). Η αξιολόγηση των λύσεων που θα προκύψουν θα γίνεται κάθε φορά με κριτήριο το makespan, δηλαδή τον συνολικό χρόνο που θα δαπανηθεί για την περάτωση όλων των εργασιών από όλες τις μηχανές με την σειρά του κάθε διανύσματος λύσης.

5. Αλγόριθμος ζευγαρώματος μελισσών-honey bee mating optimization (HBMO)

Ο αλγόριθμος ζευγαρώματος μελισσών είναι ένας από τους πιο διαδεδομένους γενετικούς αλγορίθμους βελτιστοποίησης πολύπλοκων προβλημάτων όπως το πρόβλημα χρονοπρογραμματισμού εργασιών (PFSSP) όπου είναι και το θέμα της συγκεκριμένης εργασίας. Ο αλγόριθμος προσομοιώνει την διαδικασία μέσα από την οποία η βασίλισσα των μελισσών ζευγαρώνει με τους κηφήνες για να αναπαραχθεί.

Αρχικά πρέπει να αναλυθεί η διαδικασία η οποία ακολουθείται στον πραγματικό κόσμο από την μέλισσα για την δημιουργία δυνατών απογόνων.

Σε κάθε μία κυψέλη πάντα επιβιώνει μόνο μία βασίλισσα μέλισσα (Queen) και ουσιαστικά αποτελεί την πιο σημαντική μέλισσα ανάμεσα σε όλες τις άλλες αφού αυτή θα είναι η υπεύθυνη για την αναπαραγωγή της κυψέλης αλλά και για την παραγωγή του βασιλικού πολτού που θα θρέψει τους απογόνους (Broods). Τα υπόλοιπα μέλη της κυψέλης είναι οι

κηφήνες (Drones), οι οποίοι είναι αρσενικές μέλισσες και είναι υπεύθυνες για το ζευγάρωμα της βασίλισσας και τέλος είναι οι εργάτριες (Workers) οι οποίες είναι στείρες θηλυκές μέλισσες και είναι υπεύθυνες για την φροντίδα της κυψέλης. Οι εργάτριες δεν έχουν κάποιο ρόλο στην αναπαραγωγή της κυψέλης επομένως στην υλοποίηση του αλγόριθμου θα παίξουν ρόλο μόνο στο κομμάτι της τοπικής αναζήτησης (local search θα αναλυθεί παρακάτω). Τα τρία είδη μελισσών-λύσεων που θα χρησιμοποιηθούν είναι αρχικά η βασίλισσα (Queen-βέλτιστη λύση), οι κηφήνες (drones-τυχαίες λύσεις), οι απόγονοι (broods-νέες λύσεις).

Για την δημιουργία απογόνων η βασίλισσα ξεκινάει την πτήση της κατά την διάρκεια της οποίας θα ζευγαρώσει με 8-18 κηφήνες. Η πιθανότητα να γίνει επιτυχημένα η γονιμοποίηση της είναι ανάλογη της ενέργειας της βασίλισσας και των κηφήνων, καθώς και της πτητικής ταχύτητας με την οποία η βασίλισσα πετάει. Η βασίλισσα θα επιστρέψει στην κυψέλη μόνο όταν η ενέργεια της πέσει κάτω από ένα ορισμένο ενεργειακό κατώφλι ή όταν η σπερματοθήκη της γεμίσει. Όταν η βασίλισσα θα γονιμοποιηθεί επιτυχώς, θα αποθηκεύσει στην σπερματοθήκη της το σπέρμα από όλους τους κηφήνες που κατάφεραν να την γονιμοποιήσουν και με τον κατάλληλο τρόπο θα χρησιμοποιήσει τους πιο δυνατούς γενότυπους για την δημιουργία απογόνων.

5.1 Μοντελοποίηση του αλγορίθμου ζευγαρώματος μελισσών

Για την ανάπτυξη του αλγορίθμου ζευγαρώματος μελισσών, θα πρέπει να γίνει μαθηματική μοντελοποίηση της σχέσης που εκφράζει την ενέργεια και την ταχύτητα της βασίλισσας κατά την πτήση καθώς και την σχέση που περιγράφει την πιθανότητα να γίνει επιτυχημένα γονιμοποίηση της βασίλισσας από ένα κηφήνα. Οι σχέσεις αυτές παρουσιάζονται και εξηγούνται παρακάτω:

$$\boldsymbol{Prob(D) = e^{\left[-\frac{\Delta(f)}{Speed(t)}\right]}} \qquad \textbf{Σχέση 1}$$

$$\boldsymbol{Speed(t+1) = \alpha * Speed(t) \quad \alpha \in (0,1)} \qquad \textbf{Σχέση 2}$$

$$\boldsymbol{Energy(t+1) = \alpha * energy(t) \;\; \alpha \in (0,1)} \qquad \textbf{Σχέση 3}$$

- Η σχέση 1 περιγράφει την πιθανότητα να αντληθεί το σπέρμα ενός κηφήνα στην σπερματοθήκη της βασίλισσας και είναι μια εκθετική συνάρτηση η οποία στο εκθετικό της μέρος έχει τον όρο Δ(f) ο οποίος είναι ίσος με την διαφορά της καταλληλόλητας της

βασίλισσας και του κηφήνα σε απόλυτη τιμή (fitness value θα αναλυθεί παρακάτω) και τον όρο Speed(t) ο οποίος είναι η ταχύτητα της βασίλισσας στον χρόνο t. Η ύπαρξη του αρνητικού πρόσημου στον εκθέτη δηλώνει ότι η πιθανότητα να γονιμοποιηθεί η βασίλισσα είναι αντιστρόφως ανάλογη με τον χρόνο.

- Η σχέση 2 περιγράφει την μείωση της ταχύτητας πτήσης της βασίλισσας στην πάροδο του χρόνου, η οποία κάθε χρονική στιγμή t είναι ίση με το γινόμενο της ταχύτητας την χρονική στιγμή t-1 και του σταθερού όρου α, ο οποίος ορίζεται από τον χρήστη και είναι μεταξύ 0 και 1.
- Η σχέση 3 περιγράφει την μείωση της ενέργειας πτήσης της βασίλισσας στην πάροδο του χρόνου, η οποία κάθε χρονική στιγμή t είναι ίση με το γινόμενο της ενέργειας την χρονική στιγμή t-1 και του σταθερού όρου α, ο οποίος ορίζεται από τον χρήστη και είναι μεταξύ 0 και 1.

5.2 Ψευδοκώδικας της main συνάρτησης Calculate-Time

Algorithm: Calculate_Time

```
1.  Initialize Population of Bees (N)
2.  Read an excel with the tasks and machines
3.  For i=1:N
4.      Sort tasks randomly
5.      Call create_solution algorithm
6.      Save all solutions
7.  End for
8.  Minimum time is Queen time
9.  Minimum order is Queen order
10. All the other solutions are Drones
11. Define mating flights (M)
12. Define reduction factor α
13. For i=1:M
14.     Initialize size of spermatheca
15.     Initialize energy of the Queen
```

```
Initialize speed of the Queen
Call the fitness function for Drones and Queen
While energy>threshold & spermatheca is not full
    Randomly select a drone
    If prob>threshold
        Add sperm to spermatheca
        Spermatheca=spermatheca -1
    End if
Reduce queen energy and speed
Energy=α*energy
Speed=α*speed
End for
Check the size of spermatheca (R)
For i=1:R
  Select a Drone sperm
  Call the crossover operator to create a brood
  Call the local_search function to improve the brood
  If crossover brood time< local_search brood time
     Best brood order=crossover brood order
     Best brood time= crossover brood time
  End if
     If Best brood time<Queen time
       Queen time= best brood time
       Queen order=best brood order
  Else
     If best brood time<= drone time
       Replace the drone with the brood
Call the Create_solution function for the new Queen
Return Queen order
Return Queen time
```

5.3 Επεξήγηση του main αλγορίθμου Calculate-Time

Η main συνάρτηση είναι η κύρια συνάρτηση μέσα από την οποία υλοποιείται και επιλύεται το πρόβλημα χρονοπρογραμματισμού εργασιών και πάνω σε αυτήν καλούνται ένα σύνολο άλλων συναρτήσεων που υλοποιούν και μοντελοποιούν τον αλγόριθμο ζευγαρώματος μελισσών. Η main συνάρτηση ονομάστηκε Calculate_Time καθώς η βελτιστοποίηση του προβλήματος βασίζεται στην εύρεση του μικρότερου makespan μεταξύ των διάφορων συνδυασμών των εργασιών που πρέπει να πραγματοποιηθούν. Οι συναρτήσεις που καλούνται για την υλοποίηση του αλγορίθμου είναι η Create_solution, η Crossover, η Fitness και η Local_Search οι οποίες θα αναλυθούν παρακάτω.

Ο αλγόριθμος ξεκινά με τον ορισμό του σμήνους μελισσών-τυχαίων λύσεων που θα χρησιμοποιηθεί (N) και διαβάζει από ένα αρχείο excel τον πίνακα που απεικονίζει τον αριθμό των μηχανών και των διεργασιών που πρέπει να πραγματοποιηθούν καθώς και τους χρόνους που χρειάζονται οι εκάστοτε διεργασίες για να περατωθούν. Η παρακάτω εικόνα παρουσιάζει ένα παράδειγμα 4 μηχανών και 5 διεργασιών.

	Task 1	Task 2	Task 3	Task 4	Task 5
Machine 1	$p_{1.1}$	$p_{1.2}$	$p_{1.3}$	$p_{1.4}$	$p_{1.5}$
Machine 2	$p_{2.1}$	$p_{2.2}$	$p_{2.3}$	$p_{2.4}$	$p_{2.5}$
Machine 3	$p_{3.1}$	$p_{3.2}$	$p_{3.3}$	$p_{3.4}$	$p_{3.5}$
Machine 4	$p_{4.1}$	$p_{4.2}$	$p_{4.3}$	$p_{4.4}$	$p_{4.5}$

Στην συνέχεια γίνεται μια επαναληπτική διαδικασία for η οποία αφού πρώτα γίνει τυχαία η ταξινόμηση των διεργασιών καλεί την συνάρτηση Create_Solution, ώστε να υπολογιστεί το makespan (Time) για κάθε μία από τις τυχαίες λύσεις και αυτή με το μικρότερο θα είναι η βασίλισσα (queen) ενώ όλες οι υπόλοιπες θα είναι οι κηφήνες (drones). Η συνάρτηση Create_Solution θα υπολογίσει και θα αποθηκεύσει επιπλέον τον πίνακα που δείχνει τις προσαυξήσεις του χρόνου κατά την διάρκεια περάτωσης των διεργασιών στην εκάστοτε σειρά (Solution) καθώς και τις N-σειρές λύσεων που υπάρχουν (Order). Εδώ τελειώνει η επαναληπτική διαδικασία.

Μετά το τέλος της επαναληπτικής διαδικασίας, ορίζω την σειρά που έδωσε το μικρότερο makespan ως «σειρά της βασίλισσας» (Queen_order), και το makespan αυτό ως «χρόνο βασίλισσας» (Queen_Time). Όλες τις υπόλοιπες λύσεις της αποθήκευσα ως «άλλες λύσεις» (Other_solutions), τους υπόλοιπους χρόνους ως «άλλους χρόνους» (Other_time).

Πλέον εφόσον υπάρχει βασίλισσα και κηφήνες μπορεί να ξεκινήσει η διαδικασία του ζευγαρώματος και η παραγωγή νέων λύσεων-απογόνων. Πριν ξεκινήσει η διαδικασία δημιουργίας απογόνων πρέπει να οριστεί ο

αριθμός πτήσεων ζευγαρώματος που θα κάνει η βασίλισσα κατά την διάρκεια εκτέλεσης του προγράμματος (M), καθώς και ο συντελεστής μείωσης της ενέργειας και της ταχύτητάς της (α). Στην συνέχεια, ξεκινάει μια επαναληπτική διαδικασία for για κάθε μία από τις πτήσεις (M) που θα κάνει η βασίλισσα και ο υποψήφιος κηφήνας κάθε φορά. Στην for loop αυτή αφού πρώτα οριστεί το μέγεθος της σπερματοθήκης της βασίλισσας (sp), η ενέργεια (energy) και η ταχύτητά της (speed), καλείται η συνάρτηση fitness η οποία θα δείξει την καταλληλόλητα της βασίλισσας και του κηφήνα για αναπαραγωγή. Στην συνέχεια για τον κάθε κηφήνα θα υπολογιστεί η πιθανότητα να γίνει επιτυχημένα η αναπαραγωγή σύμφωνα με την σχέση που αναφέρθηκε παραπάνω $Prob(D) = e^{\left[-\frac{\Delta(f)}{Speed(t)}\right]}$, εάν η σχέση αυτή είναι μεγαλύτερη από 0,5, τότε η βασίλισσα θα δεσμεύσει το σπέρμα του κηφήνα στην σπερματοθήκη της μειώνοντας τον χώρο της σπερματοθήκης της κατά 1 θέση ανά επιτυχημένη αναπαραγωγή. Εάν η πιθανότητα είναι μικρότερη του 0,5 τότε η αναπαραγωγή δεν είναι επιτυχημένη και η βασίλισσα προσεγγίζει τον επόμενο κηφήνα. Οι πτήσεις αυτές θα διακοπούν όταν η ενέργεια της βασίλισσας πέσει κάτω από ένα ορισμένο ενεργειακό κατώφλι ή εάν η σπερματοθήκη της γεμίσει. Σε κάθε μία πτήση η ταχύτητα και η ενέργεια της μειώνονται αναλογικά με τον όρο (α) που προαναφέρθηκε.

Στην επόμενη φάση η βασίλισσα ξεκινάει την παραγωγή απογόνων-λύσεων. Αυτή η διαδικασία υλοποιείται μέσω μιας επαναληπτικής διαδικασίας for κατά την οποία επιλέγεται το σπέρμα του κάθε κηφήνα και μέσω της συνάρτησης crossover που καλείται δημιουργείται ένας απόγονος-λύση (brood) κάθε φορά. Σε αυτό το σημείο εμφανίζεται και ο τελευταίος μηχανισμός του αλγορίθμου όπου είναι η λειτουργία που επιτελούν οι εργάτριες στην κυψέλη, δηλαδή η φροντίδα-βελτίωση των απογόνων. Αυτό αλγοριθμικά υλοποιήθηκε καλώντας την συνάρτηση Local_Search η οποία μέσω ενός μηχανισμού που θα αναλυθεί παρακάτω βελτιώνει την ποιότητα του απογόνου βελτιστοποιώντας την τιμή του makespan. Η main συνάρτηση τελειώνει συγκρίνοντας κάθε έναν απόγονο που δημιουργείται από την συνάρτηση crossover με την βελτιωμένη έκδοσή του η οποία δημιουργήθηκε από την συνάρτηση Local_Search. Ο απόγονος με το μικρότερο makespan θα δώσει τον καλύτερο χρόνο απογόνου (Best_Brood_Time) και την καλύτερη σειρά απογόνου (Best_Brood_Order), εάν ο απόγονος αυτός έχει μικρότερο makespan από την βασίλισσα τότε θα την αντικαταστήσει. Εάν τέλος ο καλύτερος απόγονος έχει μεγαλύτερο makespan από το αντίστοιχο της βασίλισσας λύσης, τότε γίνεται έλεγχος εάν υπάρχει κάποιος drone με μεγαλύτερο συνολικό χρόνο από τον best_brood όπου σε αυτήν την περίπτωση τον αντικαθιστά.

5.4 Ψευδοκώδικας της συνάρτησης Create_Solution

Algorithm: Create_Solution

```
1.   Sorting the tasks as per the order given
2.   Calculate the number of machines and tasks [m,t]
3.   Create an empty matrix in the size of tasks and machines found
     above time=zeros(m,t)
4.   The (1,1) element in the matrix will be the time of the first task
5.     For i=2:m
6.       Time(i,1)=time(i-1,1)+task_time(i,1)
7.     End for
8.     For i=2:t
9.       Time(1,i)=time(1,i-1)+task_time(1,i)
10.    End for
11.    For i=2:m
12.     For j=2:t
13.       Max_value=max(time(i,j-1),time(i-1,j)
14.         Time(i,j)=max_value+task_time(i,j)
15.    End for
16.     End for
```

5.5 Επεξήγηση του αλγορίθμου Create_solution

Η συνάρτηση αυτή, μοντελοποιεί την διαδικασία επίλυσης του προβλήματος χρονοπρογραμματισμού εργασιών και δίνει ως αποτέλεσμα τον συνολικό χρόνο (makespan) που θα απαιτηθεί για την περάτωση των εργασιών στη σειρά που έχουν γραφεί. Ως input παίρνει τον χρόνο περάτωσης διεργασιών (Task_Machine) και την σειρά με την οποία θα περάσουν από τις μηχανές (Order), και ως output δίνει τον συνολικό χρόνο (Time) και τον πίνακα συνολικών χρόνων (Solution).

Η συνάρτηση Create_solution είναι αυτή που υπολογίζει κάθε φορά το συνολικό makespan καθώς και τον πίνακα προσαύξησης των χρόνων,

ανάλογα με την σειρά των διεργασιών (Order) και τους χρόνους (Task_Machine) που δίνονται σε κάθε μία από τις επαναληπτικές διαδικασίες. Ο αλγόριθμος αυτός υλοποιεί ουσιαστικά την μέθοδο επίλυσης του προβλήματος χρονοπρογραμματισμού εργασιών όπως αυτή γίνεται στο χαρτί.

Ο αλγόριθμος αυτός φτιάχτηκε έτσι ώστε να μπορεί να επιλύσει το πρόβλημα με οποιαδήποτε σειρά και αν δοθούν οι διεργασίες καθώς και με όσο πλήθος διεργασιών ή μηχανών και αν υπάρξουν. Ξεκινάει ταξινομώντας τους χρόνους (Task_Machine) με την σειρά (Order) που έχει δοθεί και στην συνέχεια δημιουργεί ένα πίνακα [m,t] με μέγεθος ίσο με τον αριθμό μηχανών και διεργασιών (machines, tasks). Στην συνέχεια δημιούργησα έναν μηδενικό πίνακα με μέγεθος (m,t), ο οποίος ονομάστηκε Total_Time και ο οποίος μετά την ολοκλήρωση του αλγορίθμου θα έχει ως στοιχεία τους χρόνους όπως αυτοί προσαυξάνονται κατά την σειριακή είσοδο και έξοδο των διεργασιών από τις μηχανές και το τελευταίο στοιχείο του θα ορίζει το maksespan. Παρακάτω θα αναλυθεί η μεθοδολογία συμπλήρωσης του πίνακα Total_Time.

Το στοιχείο (1,1) θα είναι ο χρόνος που χρειάζεται η πρώτη διεργασία για να περάσει από την πρώτη μηχανή. Η συμπλήρωση της πρώτης στήλης έγινε με μία επαναληπτική διαδικασία for η οποία σε κάθε βήμα αθροίζει τον προηγούμενο συνολικό χρόνο που χρειάστηκε η διαδικασία για να περατωθεί από την προηγούμενη μηχανή m, με τον χρόνο που χρειάζεται για να περατωθεί από την παρούσα κάθε φορά μηχανή. Η διαδικασία αυτή σταματάει όταν η πρώτη διεργασία περάσει από όλες τις μηχανές. Για την συμπλήρωση της πρώτης σειράς έγινε αντίστοιχα μία επαναληπτική διαδικασία for η οποία σε κάθε της βήμα αθροίζει τον προηγούμενο συνολικό χρόνο που χρειάστηκε η κάθε διαδικασία t για να περάσει από την πρώτη μηχανή, με τον χρόνο της παρούσας κάθε φορά διεργασίας που έχει σειρά. Η διαδικασία αυτή σταματάει όταν όλες οι διεργασίες περάσουν από την πρώτη μηχανή.

Για την συμπλήρωση των υπόλοιπων στοιχείων του πίνακα χρησιμοποιήθηκαν δύο επαναληπτικές διαδικασίες for, μία η οποία σκάναρε οριζόντια τις σειρές του πίνακα και μία που σκάναρε κάθετα τις στήλες του. Στην for αυτή για κάθε ζεύγος m,t αρχικά ελέγχεται ποιος χρόνος είναι μεγαλύτερος ανάμεσα στην απελευθέρωση της μηχανής m από την t-1 διεργασία και της ολοκλήρωσης της t από την m-1 μηχανή. Το εκάστοτε σημείο m,t θα έχει ως τιμή το άθροισμα του μεγαλύτερου χρόνου που βρέθηκε προηγουμένως με τον χρόνο περάτωσης της t από την m.

Έτσι συμπληρώνονται ένα προς ένα τα στοιχεία του πίνακα μέχρι να συμπληρωθεί ολόκληρος. Το makespan θα αποθηκευτεί ως Time και ο συνολικός πίνακας ως Solution.

5.6 Ψευδοκώδικας της συνάρτησης Crossover

Algorithm: Crossover

```
While length(unique(Brood_Order))~=length(Queen_Order)
    N= number of tasks in Queen_order
    B1=random breaking point between [1,N-1]
    B2=random breaking point between [1,N-1]
     While B1==B2
          B2=random breaking point between [1,N-1]
     End
     Min_break=min(B1,B2)
     Max_break=max(B1,B2)
         Queen_Part1= Queen_order(1,Min_break)
         Queen_Part2= Queen_order(Min_break+1,Max_break)
         Queen_Part3= Queen_order(Max_break+1,end)
         Drone_Part1 = Drone_Order(1:Min_Break)
         Drone_Part2 = Drone_Order(Min_Break+1:Max_Break)
         Drone_Part3 = Drone_Order(Max_Break+1:end)
Create offsprings by the parts created before
      Offspring1= [Queen_Part1 Drone_Part2 Queen_Part3]
      Offspring2= [Drone_Part1 Queen_Part2 Drone_Part3]
Initialize the three parts of the offsprings
         Upper_New1 = New1(1:Min_Break);
         Upper_New2 = New2(1:Min_Break);
         Center_New1 = New1(Min_Break+1:Max_Break);
         Center_New2 = New2(Min_Break+1:Max_Break);
         Lower_New1 = New1(Max_Break+1:end);
         Lower_New2 = New2(Max_Break+1:end);
Find values which needs to be swapped in both offsprings
```

```
For i=1:col
Swap values in offsprings as per values found out above
        Upper_New1=swap(2,i);
        Upper_New2=swap(1,i);
        Lower_New1=swap(2,i);
        Lower_New2=swap(1,i);
End for
    Brood1 = [Upper_New1 Center_New1 Lower_New1];
    Brood2 = [Upper_New2 Center_New2 Lower_New2];
Find makespan (Brood_Time) for both offsprings
    create_solution(Task_Machine,Brood1);
    create_solution(Task_Machine,Brood2);
    Brood_Time = min(Makespan1,Makespan2);
  If Brood_Time=Makespan1
        Brood_Order = Order1;
  End if
 Else if Brood_Time=Makespan2
        Brood_Order = Order2;
  End else if
End while
```

5.7 Επεξήγηση του αλγορίθμου Crossover

Η συνάρτηση Crossover μοντελοποιεί την διαδικασία ζευγαρώματος της βέλτιστης λύσης (Queen) και του εκάστοτε κηφήνα (drone) που επιτυγχάνει ζευγάρωμα μαζί της. Υπάρχουν πολλές μέθοδοι δημιουργίας απογόνων λύσεων από δύο αρχικές λύσεις αλλά στη συγκεκριμένη εργασία χρησιμοποιήθηκε η μέθοδος που αποκαλείται μερικώς χαρτογραφημένο Crossover (Partially mapped Crossover).

Η συγκεκριμένη μέθοδος Crossover αναπτύχθηκε από τους Goldberg και Lingle (1985) για την επίλυση του προβλήματος Πλανόδιου Πωλητή (TSP) και είναι πλέον η πιο διαδεδομένη μέθοδος crossover στα προβλήματα που επιλύονται από γενετικούς αλγόριθμους. Ο τρόπος με τον οποίο λειτουργεί ο συγκεκριμένος αλγόριθμος, είναι ότι ο γενότυπος τον δύο γονέων «σπάει» τυχαία σε δύο κόμβους, δημιουργώντας έτσι τα χρωμοσώματα από τα οποία θα δημιουργηθούν οι απόγονοι. Οι γενότυποι των απογόνων θα αποτελούνται από την πρόσμιξη των χρωμοσωμάτων των αρχικών γονέων λύσεων. Παρακάτω θα αναλυθεί ο αλγόριθμος όπως γράφηκε στο περιβάλλον της Matlab.

Ο αλγόριθμος που έφτιαξα ως input έπαιρνε την σειρά διεργασιών της βέλτιστης λύσης (Queen_Order), την σειρά διεργασιών της τυχαίας λύσης με την οποία θα διασταυρωθεί η βέλτιστη (Drone_Order) καθώς και τους χρόνους που χρειάζεται η κάθε διεργασία για να περατωθεί από τις μηχανές (Task_Machine). Ως output ο αλγόριθμος δίνει την σειρά των διεργασιών που θα έχουν οι απόγονοι (Brood_Order), τον πίνακα που δείχνει την λύση του προβλήματος χρονοπρογραμματισμού εργασιών για τον γενότυπο της απογόνου λύσης (Brood_Solution), και τέλος το makespan των απογόνων (Brood_Time).

Ο αλγόριθμος ξεκινάει αρχικοποιώντας έναν μηδενικό πίνακα ο οποίος θα έχει το μέγεθος των αρχικών λύσεων και μέσα σε αυτόν θα δημιουργηθούν οι γενότυποι των νέων απογόνων. Το crossover ξεκινάει αρχικοποιώντας ως N τον αριθμό των διεργασιών της βέλτιστης λύσης και βρίσκει τυχαία δύο σημεία B1, B2 τα οποία θα δείχνουν τους κόμβους στους οποίους θα «σπάσουν» οι γενότυποι της βασίλισσας και του κηφήνα που θα ζευγαρώσουν. Επειδή τα B1 και B2 επιλέγονται τυχαία από τον υπολογιστή, εισήγαγα έναν έλεγχο ο οποίος εάν τα B1 και B2 είναι το ίδιο σημείο τότε ο αλγόριθος βρίσκει εκ νέου ένα νέο B2. Εφόσον έχουν οριστεί οι κόμβοι B1, B2 θα πρέπει να βρεθεί ποιος από τους δύο είναι ο μικρότερος ώστε να οριστούν τα Min_Break και Max_Break.

Αφού ορίστηκαν τα δύο παραπάνω σημεία οι γενότυποι της βασίλισσας και του κηφήνα θα χωριστούν σε τρία μέρη. Το πρώτα μέρη της βασίλισσας και του κηφήνα θα αποτελούνται από το πρώτο στοιχείο των διανυσμάτων Queen_Order και Drone_Order αντίστοιχα μέχρι και το σημείο Min_Break (1:Min_Break), το δεύτερο μέρος τους θα αποτελείται από το στοιχείο ακριβώς μετά το Min_Break μέχρι το Max_Break (Min_Break+1:Max_Break) και το τρίτο μέρος τους θα αποτελείται από το στοιχείο ακριβώς μετά το Max_Break μέχρι το τελευταίο στοιχείο των διανυσμάτων Queen_Order και Drone_Order αντίστοιχα (Max_Break+1:end).

Εφόσον έχουν οριστεί τα τρία μέρη των αρχικών λύσεων, ξεκινάει η διαδικασία του crossover κατά την οποία θα δημιουργηθούν δύο νέες λύσεις (Brood). Η πρώτη θα αποτελείται από το πρώτο και το τρίτο μέρος της βασίλισσας καθώς και από το μεσαίο μέρος του κηφήνα, ενώ η δεύτερη θα αποτελείται από το πρώτο και το τρίτο μέρος του κηφήνα και το μεσαίο της βασίλισσας.

Αφού δημιουργηθούν οι δύο νέες λύσεις θα πρέπει να ελεγχθεί ποιες διεργασίες επαναλαμβάνονται εντός των λύσεων. Για να γίνει αυτό χρησιμοποίησα τα μεσαία μέρη των δύο απογόνων λύσεων ώστε να δω

ποια νούμερα πρέπει να αντικατασταθούν στα δύο ακριανά μέρη.

5.8 Ψευδοκώδικας της συνάρτησης Fitness

Algorithm: Local_Search

```
1.    Get size of the Best in [m,n]
2.    Calculate Best_Time from Best in m and n
3.    For 1:Length of the Rest
4.        Calculate Rest_Time = Rest(m,n,i)
5.    End for
6.    Calculate Max_Time Equals to find max in Rest_Time
7.    Calculate Rest_Fitness Equals Max_Time - Rest_Time;
8.    Calculate Rest_Fitness = Rest_Fitness + 1;
9.    Calculate Best_Fitness = Max_Time - Best_Time;
10.   Calculate Best_Fitness = Best_Fitness + 1;
```

5.9 Επεξήγηση του αλγορίθμου Fitness

Οι γενετικοί αλγόριθμοι για την δημιουργία νέων απογόνων λύσεων από ήδη υπαρκτές λύσεις χρειάζονται μια διαδικασία η οποία θα ελέγχει την καταλληλόλητα των γονέων λύσεων να «ζευγαρώσουν». Η συνάρτηση Fitness που δημιούργησα, καλείται στην main συνάρτηση πριν την συνάρτηση Crossover έτσι ώστε να βρεθεί η καταλληλόλητα των Drone με τους οποίους θα ζευγαρώσει η καλύτερη λύση (Queen). Σκοπός της συγκεκριμένης συνάρτησης είναι να γίνει μια ταξινόμηση ανάμεσα στους drone με βάση το fitness value τους, δηλαδή με την καταλληλότητα τους για αναπαραγωγή.

Η συνάρτηση μου παίρνει ως input την καλύτερη λύση η οποία θα είναι η Queen (Best) και τις τυχαίες λύσεις με τις οποίες θα κληθεί να κάνει crossover η βέλτιστη λύση οι οποίες θα είναι οι Drone (Rest), επιπλέον ως output δίνει την (Best_Fitness) η οποία θα είναι η λύση με την μεγαλύτερη fitness value και θα έχει τις περισσότερες πιθανότητες να περάσει τον έλεγχο $Prob(D) = e^{\left[-\frac{\Delta(f)}{Speed(t)}\right]}$, και την Rest_Fitness η οποία θα είναι η λύση με την μικρότερη fitness value και θα έχει πρακτικά ελάχιστη πιθανότητα να επιλεγεί για crossover.

Ο μηχανισμός που χρησιμοποιήθηκε για την εύρεση των fitness value

είναι ο εξής˙ αρχικά υπολογίστηκε το makespan της βασίλισσας λύσης και των πιθανών τυχαίων λύσεων με σκοπό την εύρεση της λύσης με το μεγαλύτερο. Στη συνέχεια αφαίρεσα από όλα τα makespan την τιμή του μεγαλύτερου και πρόσθεσα σε όλα τα αποτελέσματα 1 με σκοπό να μην αποκλειστεί η λύση με το μεγαλύτερο makespan η οποία μετά την αφαίρεση θα έχει fitness value ίσο με 0 και άρα δεν θα περάσει τον έλεγχο πιθανότητας ποτέ. Από τις τιμές του fitness value που βρίσκονται είναι προφανές ότι η βασίλισσα θα έχει την μεγαλύτερη τιμή κατά απόλυτο νούμερο. Για τις υπόλοιπες λύσεις ισχύει ότι όσο μεγαλύτερο δείκτη καταλληλότητας βγάλουν τόσο μεγαλύτερη πιθανότητα έχουν να περάσουν τον έλεγχο πιθανότητας και να κάνουν επιτυχημένο Crossover.

5.10 Ψευδοκώδικας της συνάρτησης Local_Search

Algorithm: Local_Search

```
11.   Order (1,1) = first task of Current Order
12.   For i=2: length Current Order
13.       Order(1,i) = Curr_Order(1,i)
14.       Get Time and Order by create_solution(Curr_Task,Order)
15.           Call the Local minima function to calculate Best_Time
      by passing Curr_Task and Order
16.           If Time >= Best_Time
17.               Order associated with this time is the Best_Order
18.           End If
19.   End for
20.   End
```

5.11 Επεξήγηση του αλγορίθμου Local_Search

Ο αλγόριθμος τοπικής αναζήτησης είναι η προσομοίωση της λειτουργίας των εργατριών στην κυψέλη. Όπως προαναφέρθηκε, ο αλγόριθμος τοπικής αναζήτησης είναι το κομμάτι κώδικα αυτό, που είναι υπεύθυνο για την βελτίωση των νέων απογόνων λύσεων και την ελαχιστοποίηση του makespan στο συγκεκριμένο πρόβλημα. Ο κώδικας τοπικής αναζήτησης που δημιούργησα είναι βασισμένος στον αλγόριθμο ΝΕΗ και βασίζεται στην αντιμετάθεση διεργασιών στο πρόβλημα χρονοπρογραμματισμού εργασιών με σκοπό την ελαχιστοποίηση του

makespan.

Ο μηχανισμός με τον οποίο λειτουργεί ο αλγόριθμος τοπικής αναζήτησης που δημιούργησα είναι μία επαναληπτική διαδικασία κατά την οποία γίνεται εύρεση τοπικών ελαχίστων κατά την αντιμετάθεση των διεργασιών με σκοπό την εύρεση κοντινότερων makespan σε αυτά της βιβλιογραφίας. Για την εύρεση των τοπικών ελαχίστων ο αλγόριθμος της τοπικής αναζήτησης χρησιμοποιεί έναν άλλο αλγόριθμο που δημιούργησα ο οποίος ονομάστηκε Local_Minima και θα αναλυθεί η λειτουργία του παρακάτω.

Ο αλγόριθμος Local_Search πήρε ως input τους χρόνους περάτωσης των διεργασιών όπως αυτοί δόθηκαν από του αρχικούς πίνακες (Curr_Task) καθώς και την σειρά των διεργασιών (Curr_Order) πάνω στην οποία θα γίνεται κάθε φορά η τοπική αναζήτηση. Ως output ο αλγόριθμος επιστρέφει την βέλτιστη σειρά (Best_Order) καθώς και τον βέλτιστο χρόνο περάτωσης του προβλήματος (Best_Time). Ο αλγόριθμος ξεκινάει παίρνοντας το πρώτο στοιχείο από την δοθείσα σειρά και συνεχίζει μια επαναληπτική διαδικασία for στην οποία θα προστεθούν μία προς μία και οι υπόλοιπες διεργασίες.

Στην for αυτή ο αλγόριθμος θα πραγματοποιήσει δύο πολύ σημαντικές λειτουργίες για την μέθοδο ΝΕΗ που επέλεξα να υλοποιήσω, αφού η συγκεκριμένη μέθοδος σε αντίθεση με άλλους αλγορίθμους τοπικής αναζήτησης θεωρεί ως δεδομένα χειρότερη μία σειρά η οποία κατά την δημιουργία της έχει μεγαλύτερο makespan. Οι δύο λειτουργίες που πραγματοποιήθηκαν είναι πρώτον ο υπολογισμός του makespan για κάθε μια διεργασία που εντάχθηκε μέσω της for στην σειρά, και δεύτερον η αντιμετάθεση της κάθε νέας διαδικασίας με όλες τις προϋπάρχουσες διαδικασίες. Για να γίνουν τα παραπάνω πρώτα κλήθηκε η Create_Solution και μετά η Local_Minima για κάθε μία διεργασία που εντάσσεται στην for.

5.12 Ψευδοκώδικας της συνάρτησης Local_Minima

Algorithm: Local_Minima

```
1.      a = length of Order
2.      sort = order
3.      val= order(a)
4.   For (1:a-1)
```

```
5.          A = Order
6.          old = A(i)
7.          A(i) = val
8.          A(a) = old
9.          Sort(i+1,:) = A
10.     End For
11.     Variable Time = zeros(1 to a)
12.     For (1:a)
13.         Variable B = Sort(i,:)
14.         Call create_solution to calculate makespan
15.     End for
16.     Calculate Best_Time which is equals to the minimum of Time
17.     Calculate Best_Order and short it Sort(I,:)
```

5.13 Επεξήγηση του αλγορίθμου Local_Minima

Ο συγκεκριμένος αλγόριθμος όπως αναφέρθηκε και παραπάνω είναι αυτός ο οποίος θα κάνει αντιμετάθεση των διεργασιών και θα υπολογίσει τα τοπικά ελάχιστα από τα οποία θα επιλεγεί το μικρότερο ως η βέλτιστη λύση των απογόνων λύσεων (Brood). Είναι η μοναδική συνάρτηση που δεν καλείται στην main συνάρτηση (Calculate_Time), αλλά καλείται εντός της τοπικής αναζήτησης.

Ως input παίρνει την σειρά (Order) που έχει την κάθε φορά εντός της επαναληπτικής διαδικασίας στην οποία καλείται καθώς και τον χρόνο των διεργασιών όπως αυτοί αναγράφονται στο εκάστοτε παράδειγμα. Ως output έχει τις ίδιες μεταβλητές που έχει και ο αλγόριθμος της τοπικής αναζήτησης, δηλαδή την βέλτιστη σειρά (Best_Order) καθώς και τον βέλτιστο χρόνο περάτωσης του προβλήματος (Best_Time). Ο αλγόριθμος ξεκινάει με την αρχικοποίηση των μεταβλητών a ίσης με το πλήθος των διεργασιών, sort ίσης με το διάνυσμα που δείχνει την σειρά και val ίσης με τη τελευταία θέση της σειράς των διεργασιών σε κάθε επανάληψη.

Εφόσον οι μεταβλητές έχουν αρχικοποιηθεί, η συνάρτηση ξεκινάει μία επαναληπτική διαδικασία for, η οποία για κάθε μία από τις διεργασίες που εντάσσει η συνάρτηση Local_Search θα κάνει αντιμετάθεση της με όλες τις προϋπάρχουσες διεργασίες με σκοπό την εύρεση του ελάχιστου χρόνου υλοποίησης. Ο μηχανισμός που επέλεξα αλγοριθμικά για να βρω τα τοπικά ελάχιστα και να κάνω τοπική αναζήτηση είναι ο εξής˙ αρχικά η μεταβλητή old παίρνει την τιμή του A(i) σε κάθε επανάληψη της λούπας, στη συνέχεια αντιμεταθέτει την τιμή του A(i) στην θέση που δεσμεύει η μεταβλητή val επομένως σε κάθε επανάληψη το τελευταίο στοιχείο

εναλλάσσεται με όλα τα υπόλοιπα στοιχεία. Εφόσον γίνουν οι αντιμεταθέσεις μεταξύ των διεργασιών γίνεται ταξινόμηση μεταξύ τους έτσι ώστε να καταγραφούν όλοι οι πιθανοί συνδυασμοί των διεργασιών.

Σε αυτό το σημείο ο αλγόριθμος ξεκινάει μία επαναληπτική διαδικασία στην οποία υπολογίζει το makespan για κάθε έναν από τους συνδυασμούς που δημιουργήθηκαν αρχικά, έτσι ώστε να επιλέξει ως καλύτερη σειρά αυτήν με το μικρότερο makespan. Με αυτόν τον τρόπο, αφού έχει ολοκληρωθεί η υλοποίηση του αλγορίθμου Local_Minima, η Local_Search προσθέτει μία επιπλέον διεργασία στην συνάρτηση Local_Minima μέχρι να γίνει πλήρης αντιμετάθεση όλων των διεργασιών και να βρεθεί το ελάχιστο makespan για όλο τον γενότυπο μιας λύσης.

6 Αλγόριθμοι Τοπικής Αναζήτησης

6.1 Εισαγωγή

Οι αλγόριθμοι τοπικής αναζήτησης είναι η πιο διαδεδομένη μεταευρετική διαδικασία εύρεσης και βελτιστοποίησης ήδη υπαρχουσών λύσεων. Βασίζεται στην μέθοδο δοκιμής σφάλματος, και αποτελεί μία από τις πιο αποτελεσματικές μεθόδους εύρεσης καλύτερης λύσης στα προβλήματα βελτιστοποίησης. Ο ορισμός των βασικών παραμέτρων που επηρεάζουν τους αλγορίθμους αυτούς μελετήθηκαν και προσδιορίστηκαν από τον Van Breedam το 1994. Στο πρόβλημα χρονοπρογραμματισμού εργασιών έχουν εφαρμοστεί διάφοροι αλγόριθμοι τοπικής αναζήτησης οι οποίοι βασίζονται στην αναζήτηση λύσης είτε σε συγκεκριμένες «γειτονιές» λύσεων, είτε σε όλο το πεδίο λύσεων. Οι αλγόριθμοι αυτοί λόγω του μεγάλου υπολογιστικού τους κόστους εφαρμόζονται σε συγκεκριμένες λύσεις, καθώς η εφαρμογή τοπικής αναζήτησης σε όλο το σύνολο του αρχικού πληθυσμού λύσεων θα απαιτούσε τεράστιο χρόνο για την εκτέλεση του προγράμματος καθώς και τεράστια κάλυψη μνήμης.

Ο μηχανισμός λειτουργίας τέτοιων αλγορίθμων είναι η επιλογή μίας προϋπάρχουσας λύσης και η μετατροπή της με διάφορες μεθόδους έτσι ώστε να δημιουργηθεί μία νέα βελτιωμένη λύση. Η βελτίωση της ποιότητας των λύσεων βασίζεται στην στρατηγική του εκάστοτε αλγορίθμου τοπικής αναζήτησης που χρησιμοποιείται και ορίζει ουσιαστικά την κίνηση που θα ακολουθήσει ο αλγόριθμος προκείμενου να φτάσει σε νέα τοπικά ελάχιστα. Οι τρείς βασικές κινήσεις που εφαρμόζουν οι αλγόριθμοι τοπικής αναζήτησης είναι οι εξής:

- Ανταλλαγή (String Exchange): Ένας αριθμός διαδοχικών κόμβων ανταλλάσσεται μεταξύ δύο τμημάτων της λύσης.
- Διασταύρωση (Cross Exchange): Διαδοχικοί κόμβοι ανταλλάσσονται με τη διασταύρωση δύο τόξων από δύο τμήματα της λύσης.
- Επανατοποθέτηση (String Relocation): Επανατοποθέτηση μιας ακολουθίας κόμβων από ένα επιμέρους τμήμα της λύσης, σε ένα άλλο.

6.2 1-1 Exchange, 2-2 Exchange

Οι μέθοδοι τοπικής αναζήτησης αυτοί βασίζονται πάνω στην εναλλαγή κόμβων. Ο μηχανισμός με τον οποίο υλοποιείται η τοπική αναζήτηση στην μέθοδο 1-1 exchange και τελικά η δημιουργία νέων βελτιωμένων λύσεων είναι σταθερή αλλά όπως όλοι οι αλγόριθμοι τοπικής αναζήτησης, στο επίπεδο υλοποίησης σε κώδικα έχει παραλλαγές στον τρόπο γραφής της ανάλογα με της απαιτήσεις του προγραμματιστή.

Αρχικά, επιλέγεται ο πρώτος κόμβος και στη συνέχεια επιλέγεται ο δεύτερος κόμβος ο οποίος θα είναι αυστηρά διαφορετικός από τον πρώτο. Ύστερα πραγματοποιείται η εναλλαγή των δύο κόμβων αυτών. Με τον τρόπο αυτό δημιουργείται μία νέα λύση η οποία είναι αποτέλεσμα εναλλαγής δύο κόμβων από την παλιά λύση και μπορεί να υπολογιστεί το εάν αποτελεί βελτίωση της αρχικής λύσης. Παρακάτω παρουσιάζεται ένα παράδειγμα 6 κόμβων πριν και μετά την μέθοδο 1-1 exchange.

1	6	4	3	2	5

Στο συγκεκριμένο παράδειγμα έχουν επιλεγεί τα στοιχεία που βρίσκονται στις θέσεις 3 και 5 αντίστοιχα για εναλλαγή.

1	6	2	3	4	5

Γράφημα: Κίνηση 1-1 exchange εντός μίας διαδρομής

Η μέθοδος 2-2 exchange ακολουθεί την ίδια διαδικασία που περιγράφηκε παραπάνω με μόνη διαφορά ότι σε αυτή τη μέθοδο δεν πραγματοποιείται εναλλαγή δύο κόμβων αλλά δύο τόξων της λύσης. Παρακάτω παρουσιάζεται ένα παράδειγμα 8 κόμβων πριν και μετά την μέθοδο 2-2 exchange.

1	6	4	3	2	5	8	7

Στο συγκεκριμένο παράδειγμα έχουν επιλεγεί τα στοιχεία που

βρίσκονται στις θέσεις 3-4 και 7-8 αντίστοιχα για εναλλαγή.

1	6	8	7	2	5	4	3

Γράφημα: Κίνηση 2-2 exchange εντός μίας διαδρομής

6.3 1-0 Relocate, 2-0 Relocate

Οι μέθοδοι τοπικής αναζήτησης αυτοί βασίζονται πάνω στην διαγραφή και επανατοποθέτηση κόμβων σε μία λύση. Και αυτές οι μέθοδοι ακολουθούν κοινή στρατηγική κίνησης αλλά παρουσιάζουν παραλλαγές ανάλογα με τις απαιτήσεις του προγραμματιστή. Αρχικά επιλέγεται ο πρώτος κόμβος και στη συνέχεια επιλέγεται ο δεύτερος κόμβος ο οποίος θα είναι αυστηρά διαφορετικός από τον πρώτο. Ύστερα γίνεται η διαγραφή του πρώτου κόμβου και η επανατοποθέτηση του στοιχείου που άνηκε στον πρώτο κόμβο στην θέση που δείχνει ο δεύτερος κόμβος που επιλέχθηκε. Παρακάτω παρουσιάζεται ένα παράδειγμα 6 κόμβων πριν και μετά την μέθοδο 1-0 Relocate.

1	6	4	3	2	5

Στο συγκεκριμένο παράδειγμα έχει επιλεγεί το στοιχείο που βρίσκεται στη θέση 3 να διαγραφεί και να μεταφερθεί στη θέση του στοιχείου που βρισκόταν στη θέση 5 της αρχικής λύσης.

1	6	3	2	4	5

Γράφημα: Κίνηση 1-0 relocate εντός μίας διαδρομής

Η μέθοδος 2-0 relocate ακολουθεί την ίδια διαδικασία που περιγράφηκε παραπάνω με μόνη διαφορά ότι στον πρώτο κόμβο που θα επιλεγεί θα αποθηκευτεί και ο επόμενος κόμβος έτσι ώστε να δημιουργηθεί ένα τόξο. Στη συνέχεια επιλέγεται ο δεύτερος κόμβος ο οποίος θα αντικαταστήσει σε δεύτερο χρόνο το τόξο που δημιουργήθηκε. Παρακάτω παρουσιάζεται ένα παράδειγμα 8 κόμβων πριν και μετά την μέθοδο 2-0 Relocate.

1	6	3	2	4	5	8	7

Στο συγκεκριμένο παράδειγμα έχει επιλεγεί το στοιχείο που βρίσκεται στη θέση 2 να διαγραφεί και να μεταφερθεί στη θέση του στοιχείου που βρισκόταν στη θέση 6 της αρχικής λύσης.

1	2	4	5	8	6	3	7

Γράφημα: Κίνηση 2-0 relocate εντός μίας διαδρομής

6.4 2-opt

Ο 2-opt είναι ο πιο διαδεδομένος αλγόριθμος τοπικής αναζήτησης για προβλήματα logistics καθώς είναι ο μοναδικός αλγόριθμος που χρησιμοποιείται για την τροποποίηση μεμονωμένων διαδρομών. Ο μηχανισμός του αλγόριθμου είναι αρχικά η διαγραφή 2 τόξων και στη συνέχεια η διάταξη των κόμβων ανάμεσα στα δύο τόξα αυτά με νέα σειρά.

Η ανωτερότητα της συγκεκριμένης μεθόδου τοπικής αναζήτησης σε σχέση με τις άλλες μεθόδους έγκειται στο γεγονός ότι ο 1-opt ουσιαστικά ελέγχει που θα σπάσουν τα 2 τόξα με τον κατάλληλο τρόπο έτσι ώστε να απορριφθούν μονοπάτια με χειρότερες λύσεις. Παρακάτω παρουσιάζεται ένα παράδειγμα 6 κόμβων πριν και μετά την μέθοδο 2-opt.

1	2	4	5	8	6	3	7

Τα δύο τόξα που δημιουργήθηκαν είναι οι πρώτοι 2 κόμβοι και οι τελευταίοι 2 κατά σειρά. Μετά την εφαρμογή του 2 -opt τα εσωτερικά στοιχεία ανάμεσα στα δύο τόξα θα αναδιαταχθούν σε μία καινούργια βέλτιστη μορφή.

1	2	6	8	5	4	3	7

Γράφημα: Κίνηση 2 - Opt εντός μίας διαδρομής

6.5 Nawaz-Enscore-Ham (NEH)

Ο αλγόριθμος NEH προτάθηκε από τους Nawaz, Enscore και Ham το 1983 και είναι ένας από τους πιο διαδεδομένους και ποιοτικούς ευρετικούς αλγορίθμους ως προς τα αποτελέσματα που εξάγουν ειδικά στο πρόβλημα χρονοπρογραμματισμού εργασιών (PFSSP). Η τοπική αναζήτηση NEH βασίζεται στην λογική ότι οι διεργασίες με τον μεγαλύτερο χρόνο περάτωσης πρέπει να πραγματοποιηθούν όσο το δυνατόν γρηγορότερα. Επιπλέον επειδή κάνει αντιμετάθεση για κάθε έναν από τους κόμβους που εισάγονται σε κάθε μια από τις σειρές λύσεων, βρίσκει συνέχεια τα ζευγάρια που θα δώσουν πιο γρήγορα makespan απορρίπτοντας πολύ γρήγορα λύσεις που δεν φαίνονται από

την αρχή να είναι βέλτιστες.

Ο μηχανισμός υλοποίησης της τοπικής αναζήτησης NEH στο πρόβλημα χρονοπρογραμματισμού διεργασιών είναι ο εξής˙ αρχικά ο αλγόριθμος διαβάζει την λύση την οποία έχει κληθεί να βρει το τοπικό της ελάχιστο και στην συνέχεια προσθέτοντας μία διεργασία την φορά βρίσκει όλους τους πιθανούς συνδυασμούς που μπορούν να έχουν οι διεργασίες εντός της λύσης μέχρι την εύρεση του τοπικού ελάχιστου. Η τοπική αναζήτηση NEH είναι ευρέως γνωστή για την εφαρμογή της στην βελτιστοποίηση του PFSSP αφού έχει ανωτερότητα ως προς τον χρόνο και την ποιότητα των αποτελεσμάτων (σύγκριση 25 διαφορετικών αλγορίθμων για την επίλυση του προβλήματος χρονοπρογραμματισμού εργασιών Ruiz and Maroto (2005)).

6.6 Εφαρμογή του NEH στις λύσεις του HBMO

Ο γενετικός αλγόριθμος που δημιούργησα βασίστηκε στην αρχή ότι υπάρχει πάντα μία μοναδική βέλτιστη λύση η οποία είναι η βασίλισσα και πολλές τυχαίες λύσεις κηφήνες, με τις οποίες θα δημιουργήσει νέες λύσεις. Η βελτίωση αυτών των λύσεων επιλέχθηκε να γίνει μέσω του ευρετικού αλγορίθμου NEH με στόχο την προσέγγιση όσο το δυνατόν περισσότερο τις τιμές των λύσεων της βιβλιογραφίας. Παραπάνω περιγράφηκε θεωρητικά η λειτουργία της μεθόδου αυτής και η ανωτερότητα της απέναντι σε άλλες μεθόδους τοπικής αναζήτησης, επομένως τώρα θα δείξω και αλγοριθμικά ποια είναι η λειτουργία της.

Στο παρακάτω παράδειγμα θα παρουσιαστεί η τοπική αναζήτηση NEH για το πρόβλημα χρονοπρογραμματισμού εργασιών 4 μηχανών και 5 διεργασιών για την σειρά περάτωσης [1 2 3 4 5].

	Task 1	Task 2	Task 3	Task 4	Task 5
Machine 1	15	12	8	7	8
Machine 2	17	17	12	20	11
Machine 3	15	20	19	21	19
Machine 4	24	12	11	9	18

Στο πρώτο βήμα θα επιλεγούν οι πρώτες δύο διεργασίες [1 2]

	Task 1	Task 2

Machine 1	15	12
Machine 2	17	17
Machine 3	15	20
Machine 4	24	12

Ο παραπάνω πίνακας είναι ο πίνακας που θα μπει ως όρισμα στην συνάρτηση Local_Minima και στην συνέχεια θα βρεθούν όλοι οι πιθανοί συνδυασμοί και τα makespan τους. Οι πιθανοί συνδυασμοί είναι οι εξής:

1. Task 1, Task 2
2. Task 2, Task 1

Τα makespan των συνδυασμών είναι τα εξής:

1. 88
2. 83

Εφόσον βρέθηκαν τα makespan ο αλγόριθμος θα επιλέξει την σειρά με τον μικρότερο χρόνο (Task 2, Task 1) και θα ενταχθεί στην διαδικασία η επόμενη διαδικασία, δηλαδή η Task 3. Οι πιθανοί συνδυασμοί είναι οι εξής:

1. Task 2, Task 1, Task 3
2. Task 2, Task 3, Task 1
3. Task 3, Task 1, Task 2

Τα makespan των συνδυασμών είναι τα εξής:

1. 99
2. 107
3. 91

Ο αλγόριθμος θα επιλέξει την σειρά με τον μικρότερο χρόνο (Task 3, Task 1, Task 2) και θα εισάγει την επόμενη διαδικασία, δηλαδή η Task 4. Οι νέοι πιθανοί συνδυασμοί είναι οι εξής:

1. Task 3, Task 1, Task 2, Task 4
2. Task 3, Task 1, Task 4, Task 2
3. Task 3, Task 4, Task 2, Task 1
4. Task 4, Task 1, Task 2, Task 3

Τα makespan των συνδυασμών είναι τα εξής:

1. 107
2. 113
3. 120
4. 113

Η σειρά που επιλέχθηκε είναι η (Task 3, Task 1, Task 2, Task 4) και θα εισαχθεί η Task 5. Οι νέοι πιθανοί συνδυασμοί είναι οι εξής:

1. Task 3, Task 1, Task 2, Task 4, Task 5
2. Task 3, Task 1, Task 2, Task 5, Task 4
3. Task 3, Task 1, Task 5, Task 4, Task 2
4. Task 3, Task 5, Task 2, Task 4, Task 1
5. Task 5, Task 1, Task 2, Task 4, Task 3

Τα makespan των συνδυασμών είναι τα εξής:

1. 135
2. 126
3. 127
4. 138
5. 128

Όταν έχουν ερευνηθεί όλοι συνδυασμοί ο αλγόριθμος όντας ένας αλγόριθμος εύρεσης τοπικών ελαχίστων θα επιλέξει την σειρά με το ελάχιστο makespan η οποία είναι η σειρά [3 1 2 5 4] με συνολικό χρόνο 126. Στο συγκεκριμένο παράδειγμα η μέθοδος NEH βελτιστοποίησε τον συνολικό χρόνο κατά 20 μονάδες χρόνου καθώς όταν έτρεξα τον αλγόριθμο Calculate_Time για την σειρά [1 2 3 4 5] το makespan ήταν ίσο με 146.

6.7 Χαρακτηριστικά Διαδικασίας Εφαρμογής Αλγορίθμων

Για την κατανόηση και την δημιουργία των αλγορίθμων αρχικά έγραψα ψευδοκώδικες για κάθε μία από τις απαιτούμενες λειτουργίες και στην συνέχεια τους υλοποίησα σε περιβάλλον MATLAB. Οι εφαρμογές των αλγορίθμων έγιναν σε υπολογιστή με επεξεργαστή Intel(R) Core(TM) i3-5005U CPU @ 2.00GHz και στην έκδοση R2016a της MATLAB. Η εγκατεστημένη μνήμη του υπολογιστή είναι 8 GB. Οι αλγόριθμοι έλυσαν το πρόβλημα χρονοπρογραμματισμού εργασιών για 120 βασικά παραδείγματα (Taillard Benchmark Instances) διάφορων μεγεθών. Για την μελέτη και περαιτέρω κατανόηση και ανάλυση των αποτελεσμάτων, οι κώδικες δοκιμάστηκαν για μεγαλύτερα μεγέθη σπερματοθήκης με σκοπό την ύπαρξη περισσότερων crossover καθώς και για μικρότερο ενεργειακό κατώφλι με σκοπό την υπέρβαση του ελέγχου πριν το crossover και για μικρότερες τιμές ενέργειας της Βασίλισσας.

7 Αποτελέσματα Εφαρμογής του HBMO στο PFSSP

7.1Ταυτότητα Δεδομένων

Τα δεδομένα τα οποία εισήγαγα στο Excel, πάρθηκαν από τα Benchmark Instances του Taillard (1993). Τα παραδείγματα αυτά ήταν συστάδες των 20, 50, 100, 200, 500 διεργασιών και 5, 10, 20 μηχανών. Κάθε συστάδα είχε 10 παραδείγματα από κάθε συνδυασμό $m \times n$ και οι συνδυασμοί ήταν οι εξής: $20 \times 5, 20 \times 10, 20 \times 20, 50 \times 5, 50 \times 10, 50 \times 20, 100 \times 5, 100 \times 10, 100 \times 20, 200 \times 10, 200 \times 20, 500 \times 20$. Για μεγαλύτερη κατανόηση της ποιότητας των αποτελεσμάτων οι αλγόριθμοι έτρεξαν για διαφορετικούς συνδυασμούς αρχικών τυχαίων λύσεων, «αναπαραγωγικών πτήσεων» της βασίλισσας και διαφορετικές τιμές στην σπερματοθήκη και το κατώφλι ενέργειας της βασίλισσας. Στα παραδείγματα μεγάλου μεγέθους (100, 200,500) διεργασιών, προτίμησα να δώσω αρκετές επαναλήψεις- πτήσεις και σχετικά μεγάλες τιμές στην σπερματοθήκη και στην ενέργεια της βασίλισσας σε πλαίσια όμως που ο αλγόριθμος θα τερμάτιζε σε σχετικά φυσιολογικό χρόνο. Στα παραδείγματα μικρότερου μεγέθους που η υλοποίηση ήταν γρηγορότερη δηλαδή στα παραδείγματα (20,50) διεργασιών έκανα περισσότερους συνδυασμούς πτήσεων-αρχικών λύσεων-σπερματοθήκης- ενέργειας όπως αναλύεται και παρακάτω.

7.2 Υπολογιστικά Αποτελέσματα 20 και 50 διεργασιών

Στον παρακάτω πίνακα παρουσιάζονται τα αποτελέσματα του αλγορίθμου για τα παραδείγματα 20 και 50 διεργασιών με την εξής αρχικοποίηση παραμέτρων: αρχικές τυχαίες λύσεις (N=100), αριθμός επαναλήψεων-πτήσεων για δημιουργία λύσεων μέσω crossover (M=50), μέγεθος σπερματοθήκης (sp=25) και ενεργειακό κατώφλι (energy=7). Στην πρώτη στήλη φαίνεται ο αριθμός του Benchmark Instance που επιλέχθηκε σε κάθε μία περίπτωση, στην δεύτερη στήλη ο συνδυασμός διεργασιών μηχανών, στην τρίτη το makespan που βρέθηκε μετά την εκτέλεση του αλγορίθμου, στην τέταρτη βρίσκονται οι βιβλιογραφικές βέλτιστες τιμές του makespan (PSO) και στην τελευταία οι αποκλίσεις των δικών μου τιμών με τις αντίστοιχες της βιβλιογραφίας.

n	$m \times n$	Makespan (HBMO)	Makespan (BKS)	ω(%)
BI1	20×5	1297	1278	1,486

BI2	20×5	1366	1359	0,515
BI3	20×5	1098	1081	1,572
BI4	20×5	1304	1293	0,850
BI5	20×5	1250	1235	1,214
BI6	20×5	1210	1195	1,255
BI7	20×5	1251	1239	0,968
BI8	20×5	1217	1206	0,912
BI9	20×5	1273	1230	3,495
BI10	20×5	1127	1108	1,714
BI11	20×10	1649	1582	4,235
BI12	20×10	1707	1659	2,893
BI13	20×10	1593	1496	6,484
BI14	20×10	1422	1377	3,268
BI15	20×10	1475	1419	3,946
BI16	20×10	1455	1397	4,151
BI17	20×10	1542	1484	3,908
BI18	20×10	1614	1538	4,941
BI19	20×10	1640	1593	2,950
BI20	20×10	1651	1591	3,771
BI21	20×20	2383	2297	3,744
BI22	20×20	2182	2099	3,954
BI23	20×20	2393	2326	2,880
BI24	20×20	2272	2223	2,204
BI25	20×20	2378	2291	3,797
BI26	20×20	2330	2226	4,672
BI27	20×20	2361	2273	3,871
BI28	20×20	2241	2200	1,863
BI29	20×20	2297	2237	2,682
BI30	20×20	2292	2178	5,234

Πίνακας Αποτελεσμάτων 1: Παραδείγματα 20 διεργασιών

Στα παραδείγματα 20 διεργασιών υπήρξε αρκετά καλή αξιοπιστία στα αποτελέσματα, καθώς οι συνδυασμοί είναι αρκετά λιγότεροι και επομένως η τοπική αναζήτηση ήταν πιο εύκολο να πλησιάσει το ολικό ελάχιστο. Η χαμηλότερη απόκλιση για τις συγκεκριμένες τιμές των παραμέτρων που ορίστηκαν στην αρχή ήταν 0,515% και η μεγαλύτερη 6,484%. Κατά μέσο όρο η απόκλιση για το συγκεκριμένο σετ των Benchmark Instances από την βιβλιογραφία ήταν 2,981%.

n	$m \times n$	Makespan (HBMO)	Makespan (BKS)	ω(%)
BI31	50×5	2737	2724	0,477
BI32	50×5	2854	2834	0,706
BI33	50×5	2651	2621	1,145
BI34	50×5	2762	2751	0,400
BI35	50×5	2866	2863	0,105
BI36	50×5	2852	2829	0,813
BI37	50×5	2732	2725	0,257
BI38	50×5	2707	2683	0,895
BI39	50×5	2595	2552	1,685
BI40	50×5	2782	2782	0,000
BI41	50×10	3240	2991	8,325
BI42	50×10	3094	2867	7,918
BI43	50×10	3027	2839	6,622
BI44	50×10	3227	3063	5,354
BI45	50×10	3171	2976	6,552
BI46	50×10	3181	3006	5,822
BI47	50×10	3240	3093	4,753
BI48	50×10	3170	3037	4,379
BI49	50×10	3058	2897	5,557
BI50	50×10	3254	3065	6,166
BI51	50×20	4165	3850	8,182
BI52	50×20	3998	3704	7,937
BI53	50×20	3958	3640	8,736
BI54	50×20	4063	3720	9,220
BI55	50×20	3927	3610	8,781
BI56	50×20	3983	3681	8,204
BI57	50×20	4072	3704	9,935
BI58	50×20	4029	3691	9,157
BI59	50×20	4090	3743	9,271
BI60	50×20	4056	3756	7,987

Πίνακας Αποτελεσμάτων 2: Παραδείγματα 50 διεργασιών

Στα παραδείγματα 50 διεργασιών όπως φαίνεται και από τον παραπάνω πίνακα υπάρχει αντίστοιχα καλή αξιοπιστία στα αποτελέσματα, καθώς και σε αυτές τις περιπτώσεις ο αλγόριθμος της τοπικής πλησίασε το ολικό ελάχιστο σε αρκετές περιπτώσεις. Η χαμηλότερη απόκλιση σε αυτό το σετ ήταν 0% καθώς στο BI40 βρέθηκε ακριβώς το ίδιο makespan με την βιβλιογραφία και η μεγαλύτερη απόκλιση ήταν ίση με 9,935%. Κατά μέσο όρο η απόκλιση για το συγκεκριμένο σετ των Benchmark Instances από

την βιβλιογραφία ήταν 5,178 %.

7.3 Υπολογιστικά Αποτελέσματα 100, 200 και 500 διεργασιών

Ο παρακάτω πίνακας παρουσιάζει τα αποτελέσματα του αλγορίθμου για τα παραδείγματα 100 διεργασιών με διεργασιών με την εξής αρχικοποίηση παραμέτρων: αρχικές τυχαίες λύσεις (N=501), αριθμός επαναλήψεων-πτήσεων για δημιουργία λύσεων μέσω crossover (M=100), μέγεθος σπερματοθήκης (sp=50) και ενεργειακό κατώφλι (energy=7). Ο λόγος που επέλεξα να αυξήσω τον αριθμό της σπερματοθήκης της βασίλισσας σε σχέση με τα προηγούμενα παραδείγματα είναι ότι παρατήρησα όταν οι διεργασίες είναι πολλές και οι αναπαραγωγικές πτήσεις και η σπερματοθήκη είναι μικρή, το makespan είναι λογικό να έχει μεγαλύτερη απόκλιση από αυτό της βιβλιογραφίας, αφού μπορεί η λύση που θα προκύψει να απέχει αρκετά από το ολικό ελάχιστο.

n	$m \times n$	Makespan (HBMO)	Makespan (BKS)	ω(%)
BI61	100×5	5493	5493	0,000
BI62	100×5	5290	5268	0,418
BI63	100×5	5219	5175	0,850
BI64	100×5	5023	5014	0,179
BI65	100×5	5255	5250	0,095
BI66	100×5	5146	5135	0,214
BI67	100×5	5262	5246	0,305
BI68	100×5	5137	5094	0,844
BI69	100×5	5473	5448	0,459
BI70	100×5	5346	5322	0,451
BI71	100×10	5977	5770	3,588
BI72	100×10	5628	5349	5,216
BI73	100×10	5817	5676	2,484
BI74	100×10	5991	5781	3,633
BI75	100×10	5690	5467	4,079
BI76	100×10	5430	5303	2,395
BI77	100×10	5753	5595	2,824
BI78	100×10	5771	5617	2,742
BI79	100×10	5971	5871	1,703
BI80	100×10	5903	5845	0,992
BI81	100×20	6713	6202	8,239
BI82	100×20	6697	6183	8,313

BI83	100×20	6721	6271	7,176
BI84	100×20	6756	6269	7,768
BI85	100×20	6878	6314	8,933
BI86	100×20	6828	6364	7,291
BI87	100×20	6893	6268	9,971
BI88	100×20	7020	6401	9,670
BI89	100×20	6774	6275	7,952
BI90	100×20	6882	6434	6,963

Πίνακας Αποτελεσμάτων 3: Παραδείγματα 100 διεργασιών

Όπως φαίνεται από τα αποτελέσματα του πίνακα η αρχικοποίηση των παραμέτρων που έγινε κατά την εκτέλεση του προγράμματος για 100 διεργασίες έδωσε αξιόπιστα αποτελέσματα καθώς και σε αυτήν την περίπτωση ο αλγόριθμος κατάφερε να έχει μικρές αποκλίσεις. Η μικρότερη απόκλιση βρέθηκε στο BI61 στο οποίο το makespan ήταν ίσο με το ενδεδειγμένο άρα ήταν ίση με 0% και η μεγαλύτερη απόκλιση ήταν ίση με 9,971%. Για το σετ των Benchmark Instances 100 διεργασιών ο μέσος όρος των αποκλίσεων ήταν ίσος με 3,858%.

n	$m \times n$	Makespan (HBMO)	Makespan (BKS)	ω(%)
BI91	200×10	10990	10862	1,178
BI92	200×10	10838	10480	3,416
BI93	200×10	11126	10922	1,868
BI94	200×10	10939	10889	0,459
BI95	200×10	10813	10524	2,746
BI96	200×10	10660	10329	3,205
BI97	200×10	10948	10854	0,866
BI98	200×10	10944	10730	1,994
BI99	200×10	10669	10438	2,213
BI100	200×10	10781	10675	0,993
BI101	200×20	12000	11195	7,191
BI102	200×20	12083	11203	7,855
BI103	200×20	12169	11281	7,872
BI104	200×20	12117	11275	7,468
BI105	200×20	12063	11259	7,141
BI106	200×20	12067	11176	7,972
BI107	200×20	12239	11360	7,738
BI108	200×20	12150	11334	7,200
BI109	200×20	11970	11192	6,951
BI110	200×20	12145	11288	7,592

Πίνακας Αποτελεσμάτων 4: Παραδείγματα 200 διεργασιών

Ο παραπάνω πίνακας αφορά τις αποκλίσεις των makespan για τα παραδείγματα 200 διεργασιών. Όπως είναι λογικό οι αποκλίσεις είναι μεγαλύτερες σε σχέση με τα παραδείγματα λιγότερων διεργασιών καθώς η αναζήτηση στο πεδίο των πιθανών λύσεων γίνεται όλο και δυσκολότερη όσο αυξάνονται οι απαιτούμενες διεργασίες που πρέπει να περατωθούν. Ωστόσο και σε αυτό το σετ υπήρξαν περιπτώσεις με ελάχιστη απόκλιση όπως το BI94 το οποίο είχε απόκλιση 0,459%. Η μεγαλύτερη απόκλιση ήταν ίση με 7,972% και ο μέσος όρος των αποκλίσεων ήταν ίσος με 4,695%.

n	$m \times n$	Makespan (HBMO)	Makespan (BKS)	ω(%)
BI111	500×20	27442	26059	5,307
BI112	500×20	27796	26520	4,811
BI113	500×20	27709	26371	5,074
BI114	500×20	27895	26456	5,439
BI115	500×20	27546	26334	4,602
BI116	500×20	27624	26477	4,332
BI117	500×20	27587	26389	4,540
BI118	500×20	27902	26560	5,053
BI119	500×20	27350	26005	5,172
BI120	500×20	27815	26457	5,133

Πίνακας Αποτελεσμάτων 5: Παραδείγματα 500 διεργασιών

Το σετ των 500 διεργασιών είναι αυτό με τις μεγαλύτερες συνολικά αποκλίσεις αφού όπως περιγράφηκε και προηγουμένως υπάρχουν αρκετοί παράγοντες που επηρεάζουν την ποιότητα του makespan που θα προκύψει από τον αλγόριθμο. Τα αποτελέσματα βέβαια που εξάγονται από τον παραπάνω πίνακα αποδεικνύουν ότι ο γενέτικος και ο μεθευρετικός αλγόριθμος που ανέπτυξα έκαναν έρευνα σε κοντινές «γειτονιές» του ολικού ελαχίστου. Η μικρότερη απόκλιση που βρέθηκε ήταν στο BI116 ίση με 4,332%, ενώ η μεγαλύτερη ίση με 5,439%. Ο μέσος όρος των αποκλίσεων ήταν ίσος με 4,946%.

7.4 Υπολογιστικά αποτελέσματα με νέα αρχικοποίηση των παραμέτρων

Για να προσεγγίσω ακόμα περισσότερο τις βέλτιστες τιμές της βιβλιογραφίας επανέλαβα την διαδικασία υλοποίησης του αλγορίθμου για κάθε ένα από τα Benchmark Instances αλλά για νέες τιμές των παραμέτρων αυτών. Οι τιμές που επέλεξα ήταν: N=10000, M=200 και sp=100. Η επιλογή των παραμέτρων έγινε με κριτήριο το ότι όσα περισσότερα crossover πραγματοποιηθούν, τόσο πιθανότερο είναι να βρεθουν μέσω της τοπικής αναζήτησης λύσεις που θα προσεγγίζουν τις βέλτιστες. Η ύπαρξη τόσο μεγάλων νούμερων στις παραμέτρους έκανε το πρόγραμμα εξαιρετικά χρονοβόρο κατά την εκτέλεση του ειδικά κυρίως στα παραδείγματα 200 και 500 διεργασιών.

n	$m \times n$	Makespan (HBMO)	Makespan (BKS)	ω(%)
BI1	20×5	1290	1278	0,939
BI2	20×5	1359	1359	0,000
BI3	20×5	1098	1081	1,573
BI4	20×5	1304	1293	0,851
BI5	20×5	1250	1235	1,215
BI6	20×5	1210	1195	1,255
BI7	20×5	1251	1239	0,969
BI8	20×5	1212	1206	0,498
BI9	20×5	1257	1230	2,195
BI10	20×5	1126	1108	1,625
BI11	20×10	1620	1582	2,402
BI12	20×10	1704	1659	2,712
BI13	20×10	1581	1496	5,682
BI14	20×10	1416	1377	2,832
BI15	20×10	1473	1419	3,805
BI16	20×10	1445	1397	3,436
BI17	20×10	1536	1484	3,504
BI18	20×10	1582	1538	2,861
BI19	20×10	1638	1593	2,825
BI20	20×10	1642	1591	3,206
BI21	20×20	2346	2297	2,133
BI22	20×20	2162	2099	3,001
BI23	20×20	2353	2326	1,161
BI24	20×20	2262	2223	1,754
BI25	20×20	2353	2291	2,706

BI26	20×20	2273	2226	2,111
BI27	20×20	2335	2273	2,728
BI28	20×20	2233	2200	1,500
BI29	20×20	2276	2237	1,743
BI30	20×20	2290	2178	5,142

Πίνακας Αποτελεσμάτων 6: Παραδείγματα 20 διεργασιών

Όπως φαίνεται από τον παραπάνω πίνακα τα αποτελέσματα είχαν ξανά μικρές αποκλίσεις από τις βέλτιστες. Η βελτίωση τους οφείλεται στο γεγονός ότι ο αλγόριθμος έκανε περισσότερες φορές crossover και δημιούργησε περισσότερους απογόνους άρα ο μεθευρετικός αλγόριθμος NEH πλησίασε περισσότερο τη λύση με βέλτιστο makespan. Η μικρότερη απόκλιση βρέθηκε στο BI12 στο οποίο το makespan ήταν ίσο με το ενδεδειγμένο άρα ήταν ίση με 0% και η μεγαλύτερη απόκλιση ήταν ίση με 5,142%. Για το σετ των Benchmark Instances 20 διεργασιών ο μέσος όρος των αποκλίσεων ήταν ίσος με 2,279%.

n	$m \times n$	Makespan (HBMO)	Makespan (BKS)	ω(%)
BI31	50×5	2729	2724	0,184
BI32	50×5	2845	2834	0,388
BI33	50×5	2648	2621	1,030
BI34	50×5	2762	2751	0,400
BI35	50×5	2864	2863	0,035
BI36	50×5	2840	2829	0,389
BI37	50×5	2732	2725	0,257
BI38	50×5	2704	2683	0,783
BI39	50×5	2597	2552	1,763
BI40	50×5	2783	2782	0,036
BI41	50×10	3173	2991	6,085
BI42	50×10	3073	2867	7,185
BI43	50×10	3027	2839	6,622
BI44	50×10	3209	3063	4,767
BI45	50×10	3163	2976	6,284
BI46	50×10	3164	3006	5,256
BI47	50×10	3228	3093	4,365
BI48	50×10	3130	3037	3,062
BI49	50×10	3038	2897	4,867
BI50	50×10	3206	3065	4,600
BI51	50×20	4089	3850	6,208

BI52	50×20	4011	3704	8,288
BI53	50×20	3974	3640	9,176
BI54	50×20	4016	3720	7,957
BI55	50×20	3946	3610	9,307
BI56	50×20	3964	3681	7,688
BI57	50×20	4053	3704	9,422
BI58	50×20	4019	3691	8,886
BI59	50×20	4045	3743	8,068
BI60	50×20	4054	3756	7,934

Πίνακας Αποτελεσμάτων 7: Παραδείγματα 50 διεργασιών

Ο παραπάνω πίνακας παρουσιάζει τα αποτελέσματα του αλγορίθμου για τα παραδείγματα των 50 διεργασιών και σε αυτήν την περίπτωση ο αλγόριθμος παρουσίασε σχεδόν σε όλα βελτιωμένα αποτελέσματα σε σχέση με τις πρώτες αρχικοποιήσεις των παραμέτρων. Η μικρότερη απόκλιση βρέθηκε στο BI40 η οποία ήταν ίση με 0,036% ενώ η μεγαλύτερη απόκλιση ήταν ίση με 9,422%. Για το σετ των Benchmark Instances 20 διεργασιών ο μέσος όρος των αποκλίσεων ήταν ίσος με 4,674%.

n	$m \times n$	Makespan (HBMO)	Makespan (BKS)	ω(%)
BI61	100×5	5493	5493	0,000
BI62	100×5	5283	5268	0,285
BI63	100×5	5284	5175	2,106
BI64	100×5	5021	5014	0,140
BI65	100×5	5255	5250	0,095
BI66	100×5	5137	5135	0,039
BI67	100×5	5247	5246	0,019
BI68	100×5	5105	5094	0,216
BI69	100×5	5467	5448	0,349
BI70	100×5	5467	5322	2,725
BI71	100×10	5892	5770	2,738
BI72	100×10	5516	5349	3,122
BI73	100×10	5761	5676	1,498
BI74	100×10	5939	5781	2,733
BI75	100×10	5664	5467	3,603
BI76	100×10	5364	5303	1,150
BI77	100×10	5688	5595	1,662
BI78	100×10	5789	5617	3,062
BI79	100×10	5905	5871	0,579

BI80	100×10	5871	5845	0,445
BI81	100×20	6326	6202	1,999
BI82	100×20	6659	6183	7,699
BI83	100×20	6656	6271	6,139
BI84	100×20	6476	6269	3,302
BI85	100×20	6691	6314	5,971
BI86	100×20	6782	6364	6,568
BI87	100×20	6640	6268	5,935
BI88	100×20	6802	6401	6,265
BI89	100×20	6613	6275	5,386
BI90	100×20	6850	6434	6,466

Πίνακας Αποτελεσμάτων 8: Παραδείγματα 100 διεργασιών

Παρόμοια βελτίωση παρατηρήθηκε και στα αποτελέσματα των σετ περισσότερων διεργασιών καθώς πλέον οι αποκλίσεις σε ορισμένες περιπτώσεις έγιναν αισθητά μικρότερες. Στο σετ συγκεκριμένα 100 διεργασιών όπως φαίνεται και παραπάνω για αριθμό m=5 είχε αποκλίσεις κοντά στο 0 για όλα τα παραδείγματα. Όντας ένα από τα μεγάλα σετ παραδειγμάτων και εδώ παρατηρείται ότι όσο αυξάνεται ο αριθμός των μηχανών η απόκλιση από τις τιμές της βιβλιογραφίας αυξάνεται. Η μικρότερη απόκλιση που βρέθηκε για τις νέες αρχικοποιήσεις στο συγκεκριμένο σετ είναι για το BI61 στο οποίο η απόκλιση είναι ίση με 0%, και η μεγαλύτερη είναι 7,699%. Ο μέσος όρος των τιμών των αποκλίσεων που βρέθηκαν είναι με 2,743%.

n	$m \times n$	Makespan (HBMO)	Makespan (BKS)	ω(%)
BI91	200×10	10973	10862	1,022
BI92	200×10	10967	10480	4,647
BI93	200×10	11045	10922	1,126
BI94	200×10	10928	10889	0,358
BI95	200×10	10759	10524	2,233
BI96	200×10	10558	10329	2,217
BI97	200×10	10923	10854	0,636
BI98	200×10	10899	10730	1,575
BI99	200×10	10652	10438	2,050
BI100	200×10	10785	10675	1,030
BI101	200×20	11844	11195	5,797
BI102	200×20	12083	11203	7,855

BI103	200×20	12095	11281	7,216
BI104	200×20	12037	11275	6,758
BI105	200×20	12112	11259	7,576
BI106	200×20	12024	11176	7,588
BI107	200×20	11972	11360	5,387
BI108	200×20	12003	11334	5,903
BI109	200×20	11774	11192	5,200
BI110	200×20	12002	11288	6,325

Πίνακας Αποτελεσμάτων 9: Παραδείγματα 200 διεργασιών

Και στον παραπάνω πίνακα παρατηρείται βελτίωση σε σχέση με τις πρώτες μετρήσεις βλέποντας συνολικά μια μείωση τον αποκλίσεων ειδικά στα παραδείγματα για m=10. Στα Benchmark Instances των 20 μηχανών οι αποκλίσεις είχαν σαφώς μεγαλύτερα ποσοστά, γεγονός που συμβαίνει αφού πλέον οι συνδυασμοί τους οποίους θα έπρεπε να ελέγξει ο μεθευρετικός αλγόριθμος στο πεδίο των λύσεων ήταν πάρα πολλοί. Η μικρότερη απόκλιση βρέθηκε στο BI94 και ήταν ίση με 0,358%. Η μεγαλύτερη απόκλιση ήταν ίση με 7,855% και ο μέσος όρος των αποκλίσεων ήταν ίσος με 4,124%.

n	$m \times n$	Makespan (HBMO)	Makespan (BKS)	ω(%)
BI111	500×20	27215	26059	4,436
BI112	500×20	27742	26520	4,608
BI113	500×20	27659	26371	4,884
BI114	500×20	27895	26456	5,439
BI115	500×20	27134	26334	3,038
BI116	500×20	27319	26477	3,180
BI117	500×20	27430	26389	3,945
BI118	500×20	27338	26560	2,929
BI119	500×20	27314	26005	5,034
BI120	500×20	27720	26457	4,774

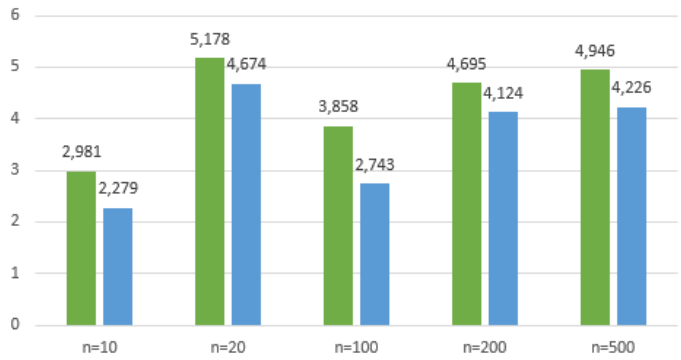
Πίνακας Αποτελεσμάτων 10: Παραδείγματα 500 διεργασιών

Το συγκεκριμένο σετ όπως ήταν και αναμενόμενο συνέχισε να έχει τις μεγαλύτερες αποκλίσεις λόγω πλήθους λύσεων. Η ύπαρξη περισσότερων crossover όμως έδωσε τη δυνατότητα στον αλγόριθμο να δημιουργήσει νέες λύσεις πιο κοντά στην βέλτιστη. Αυτός είναι ο λόγος που και σε αυτήν την περίπτωση τα makespan πλησίασαν την βιβλιογραφία και οι

αποκλίσεις μειώθηκαν. Η μικρότερη απόκλιση βρέθηκε στο BI118 ίση με 2,929%, ενώ η μεγαλύτερη ίση με 5,439%. Ο μέσος όρος των αποκλίσεων ήταν ίσος με 4,226%.

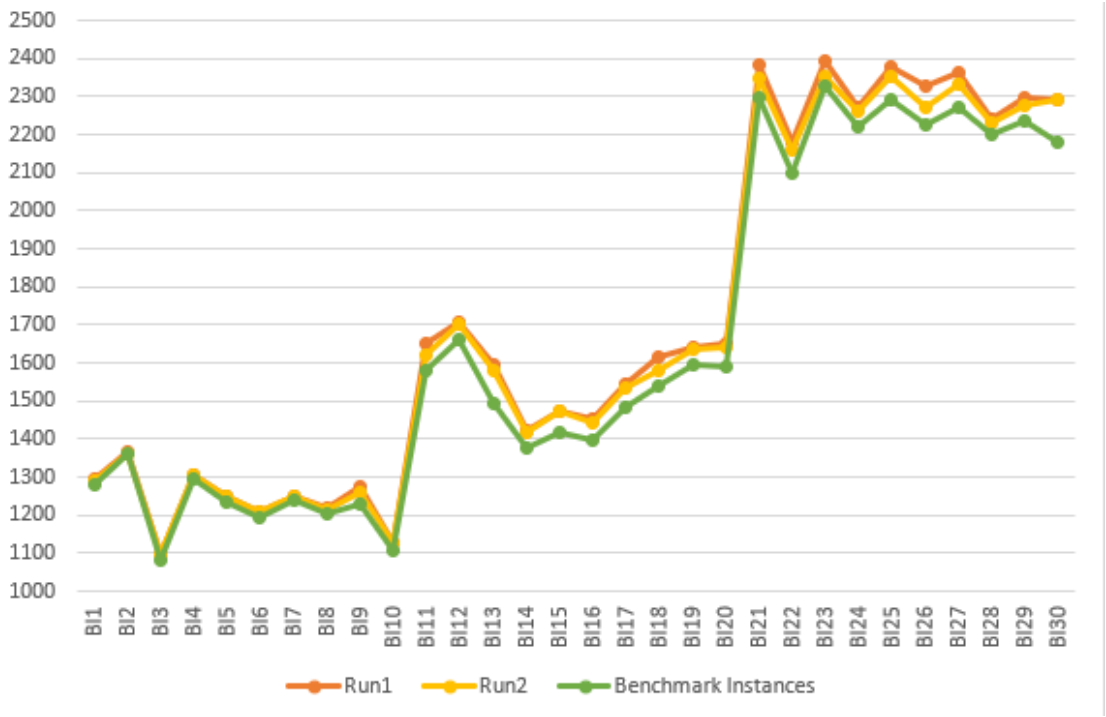
7.5 Επισκόπηση των Υπολογιστικών Αποτελεσμάτων

Για την ανάλυση των αποτελεσμάτων του αλγόριθμου ζευγαρώματος μελισσών για το πρόβλημα χρονοπρογραμματισμού εργασιών δημιούργησα τα παρακάτω παραδείγματα έτσι ώστε να φανούν και γραφικά τα ποιοτικά χαρακτηριστικά των αποτελεσμάτων σε σχέση με τα αποτελέσματα της βιβλιογραφίας.



Γράφημα 1: Αποκλίσεις των makespan ανά σετ. Πράσινο (1st Run), Μπλε (2nd Run)

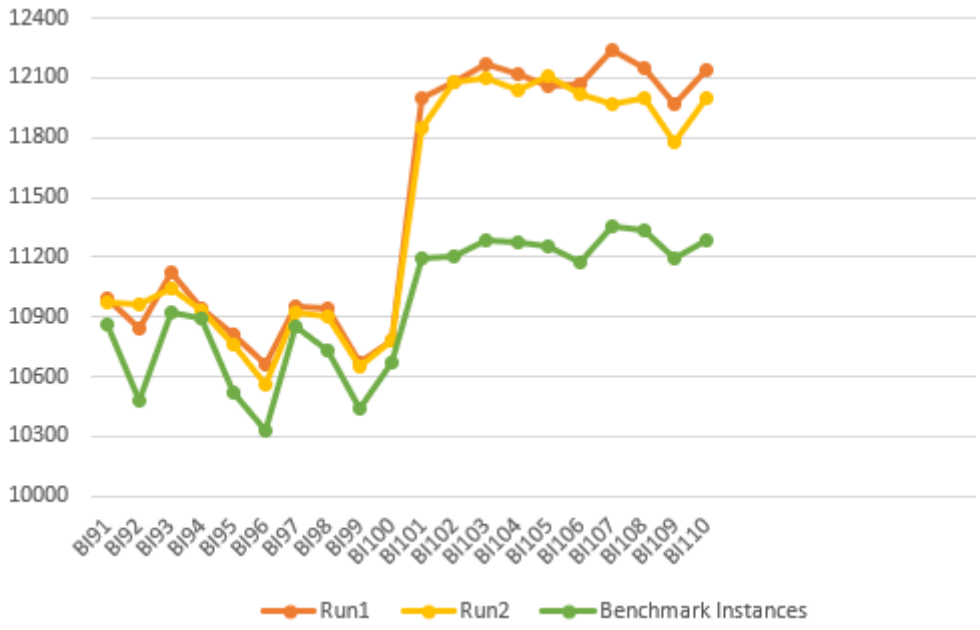
Το συγκεκριμένο γράφημα δείχνει την βελτίωση του αλγορίθμου μετά τις δεύτερες αρχικοποιήσεις που έκανα. Παρατηρείται ότι για τα παραδείγματα 10 και 100 διεργασιών, οι αποκλίσεις ήταν μικρότερες και στις δύο εκτελέσεις του αλγορίθμου που πραγματοποίηθηκαν. Η μικρότερη κατά μέσω όρο απόκλιση βρέθηκε στα Benchamark Instances 10 διεργασιών και ήταν ίση με 2,279% κατά την δεύτερη υλοποίηση του αλγορίθμου και η μεγαλύτερη στα Benchamark Instances 20 διεργασιών με ποσοστό 5,178% κατά την πρώτη υλοποίηση του αλγορίθμου.



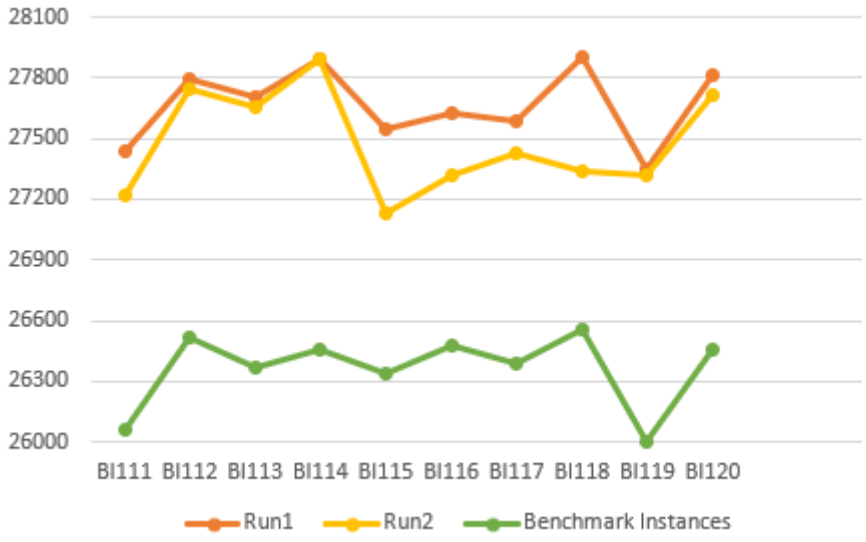
Γράφημα 2: Γραφική απεικόνιση αποτελεσμάτων για τα Benchmark Instances 20 διεργασιών

Γράφημα 3: Γραφική απεικόνιση αποτελεσμάτων για τα Benchmark Instances 50 διεργασιών

Γράφημα 4: Γραφική απεικόνιση αποτελεσμάτων για τα Benchmark Instances 100 διεργασιών



Γράφημα 5: Γραφική απεικόνιση αποτελεσμάτων για τα Benchmark Instances 200 διεργασιών



Γράφημα 6: Γραφική απεικόνιση αποτελεσμάτων για τα Benchmark Instances 500 διεργασιών

7.6 Συμπεράσματα

Στους παραπάνω πίνακες παρουσιάζονται γραφικά οι τιμές των makespan που βρέθηκαν για τα 120 Benchmark Instances στις δύο υλοποιήσεις του HBMO και της βιβλιογραφίας. Παρατηρείται ότι στα προβλήματα με περισσότερους κόμβους διεργασιών η διαφορά από τα βέλτιστα παρουσιάζει αύξηση. Και στα 5 σετ παραδειγμάτων αποδείχθηκε ότι η υλοποίηση του αλγορίθμου για μεγαλύτερο αριθμό επαναλήψεων crossover και για μεγαλύτερη σπερματοθήκη οι τιμές του makespan ήταν πιο κοντά στις βέλτιστες. Σε αυτό το πλαίσιο αξίζει να αναλυθούν τα αποτελέσματα της δεύτερης εκτέλεσης του προγράμματος.

Στα μικρά προβλήματα (20, 50 διεργασιών) η αποτελεσματικότητα των αλγορίθμων κρίνεται επαρκής. Η απόκλιση των τιμών για το σετ των 20 διεργασιών σε ποσοστό, κυμάνθηκε από 0% όπου ο αλγόριθμος βρήκε ακριβώς την βέλτιστη λύση, μέχρι και 5,142%. Αντίστοιχα για το σετ των 50 διεργασιών, οι αποκλίσεις κυμάνθηκαν από 0,036% στην βέλτιστη περίπτωση μέχρι και 9,422% στην χειρότερη.

Στα μικρά προβλήματα (20, 50 διεργασιών) η αποτελεσματικότητα των αλγορίθμων κρίνεται επαρκής. Η απόκλιση των τιμών για το σετ των 20 διεργασιών σε ποσοστό, κυμάνθηκε από 0% όπου ο αλγόριθμος βρήκε ακριβώς την βέλτιστη λύση, μέχρι και 5,142%. Αντίστοιχα για το σετ των 50 διεργασιών, οι αποκλίσεις κυμάνθηκαν από 0,036% έως και 9,422%.

Όσον αφορά τα μεγάλα προβλήματα (100, 200, 500 διεργασιών), όπως φαίνεται και από τα παραπάνω γραφήματα οι αποκλίσεις αυξάνονται αναλογικά με τον αριθμό των μηχανών που υπάρχουν στο κάθε Benchamerk Instance. Ο γενετικός αλγόριθμος που δημιούργησα ωστόσο, έδωσε επαρκή αποτελέσματα ακόμα και στους μεγάλους συνδυασμούς διεργασιών-μηχανών. Για το σετ των 100 διεργασιών οι αποκλίσεις κυμάνθηκαν από 0% στην καλύτερη περίπτωση, 7,699% στην χειρότερη. Αντίστοιχα για το σετ των 200 διεργασιών οι αποκλίσεις κυμάνθηκαν μεταξύ του 0,358% στο ελάχιστο και 7,855% στο μέγιστο. Τέλος, για το σετ των 500 διεργασιών, το οποίο είχε και την μεγαλύτερη πολυπλοκότητα σε ως προς το μέγεθος των λύσεων που κλήθηκε να ερευνήσει ο αλγόριθμος, τα αποτελέσματα ήταν εξαιρετικά ενθαρρυντικά αφού οι αποκλίσεις ξεκίνησαν από 2,929% και δεν υπερβήκανε το 5,439%. Πέραν από τις στατιστικές παρατηρήσεις που αφορούν τις διαφορές των τιμών από τις αντίστοιχες της βιβλιογραφίας, αξίζει να σημειωθούν ζητήματα που αφορούν την εφαρμογή των

αλγορίθμων.

Για την περαιτέρω βελτίωση των αποτελεσμάτων, προτείνεται ο προγραμματισμός του HBMO και του μεθευρετικού NEH σε άλλη γλώσσες προγραμματισμού με μεγαλύτερη ταχύτητα υλοποίησης. Η MATLAB παρουσίαζε αυξανόμενη επιβράδυνση όσο η κλίμακα των προβλημάτων αυξανόταν. Επίσης, η επιλογή του μεθευρετικού αλγορίθμου που επέλεξα, εξαιτίας της υλοποίησης στοχευμένης έρευνας ανάμεσα σε πάρα πολλούς συνδυασμούς λύσεων, κατανάλωνε αρκετό χρόνο κατά την διάρκεια εκτέλεσης του προγράμματος. Επιπλέον, παρατηρήθηκε ότι οι πολλαπλές επαναλήψεις κινήσεων τοπικής αναζήτησης είχαν ως αποτέλεσμα την μη βελτίωση των αποτελεσμάτων αλλά την επιμονή τους σε συγκεκριμένα τοπικά ελάχιστα. Επιπρόσθετα, από κατασκευής του ο αλγόριθμος λειτουργεί με αρχικοποιήσεις από τον χρήστη, επομένως συνιστάται για μεγαλύτερη ακρίβεια να μπαίνουν μεγάλοι αριθμοί στις τιμές της σπερματοθήκης έτσι ώστε να γεννιούνται περισσότερες λύσεις.

8 Βιβλιογραφία

[1] *Yannis Marinakis , Magdalene Marinaki, Georgios Dounias, Honey bees mating optimization algorithm for the Euclidean traveling salesman problem , Information Sciences* 181(20), (2011): 4684-4698.

[2] *Yannis Marinakis, Magdalene Marinaki, Particle swarm optimization with expanding neighborhood topology for the permutation flowshop scheduling problem , Soft Computing* 17(7), (2013): 1159-1173.

[3] *Weibo Liu, Yan Jin, Mark Price, A new improved NEH heuristic for permutation flowshop scheduling problems , International Journal of Production Economics* 193 (2017): 21-30.

[4] *Radosław Puka, Jerzy Duda , Adam Stawowy, Iwona Skalna, N-NEH+ algorithm for solving permutation flow shop problems, Computers & Operations Research* 132 (2021): 105296.

[5] *Rubén Ruiz, Concepción Maroto, Javier Alcaraz, Two new robust genetic algorithms for the flowshop scheduling problem, Omega* 34(5), (2006): 461-476.

[6] *Jatinder N.D. Gupta, Edward F., Stafford Jr., Flowshop scheduling research after five decades , European Journal of Operational Research* 169(3), (2006): 699-711

[7] *Vijendra Singh, Simran Choudhary, Genetic Algorithm for Traveling Salesman Problem: Using Modified Partially-Mapped Crossover Operator, 2009 International Multimedia, Signal Processing and Communication Technologies*. IEEE, 2009.

[8] *Quan-Ke Pana, Rubén Ruizb, Local search methods for the flow-shop scheduling problem with flow time minimization, European Journal of Operational Research* 222(1), (2012): 31-43.

[9] *Bruno de Athayde Prata, Victor Fernandez-Viagas, Jose M.Framinan, Carlos Diego Rodrigues, Matheuristics for the flowshop scheduling problem with controllable processing times and limited resource consumption to minimize total tardiness, Computers & Operations Research* (2022): 105880.