ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ ΠΑΡΑΓΩΓΗΣ ΚΑΙ ΔΙΟΙΚΗΣΗΣ

Επίλυση προβλήματος προγραμματισμού παράγωγης κατά παραγγελία (JOB SHOP SCHEDULING) με τη χρήση του μεθευρετικού αλγορίθμου περιορισμένης αναζήτησης (Tabu Search)

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ιερωνυμάκης Κωνσταντίνος

Τριμελής Επιτροπή:

Επίκουρος Καθηγητής Μαρινάκης Ιωάννης (Επιβλέπων)

Καθηγητής Σταυρουλάκης Γεώργιος

Καθηγητής Αντωνιάδης Αριστομένης

Χανιά, Μάρτιος 2014

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω τον επιβλέποντα καθηγητή μου κ. Μαρινάκη Ιωάννη για την άψογη συνεργασία που είχαμε και για το χρόνο που μου αφιέρωσε .

Τέλος, θα ήθελα να ευχαριστήσω τους γονείς μου Δημήτρη και Αρετή, καθώς επίσης και τον αδερφό μου Χάρη, που με υπομονή και κουράγιο προσέφεραν την απαραίτητη ηθική συμπαράσταση για την ολοκλήρωση των προπτυχιακών σπουδών μου και της Διπλωματικής εργασίας.

Επίσης να ευχαριστήσω την τριμελής επιτροπή τον κ. Σταυρουλάκη και τον κ. Αντωνιάδη για το χρόνο τους και για τη προσοχή του κατά τη διάρκεια της παρουσίασης.

Τέλος θα ήθελα να ευχαριστήσω τους Φίλους και Συμφοιτητές μου που με στήριξαν καθ' όλη την διάρκεια της συγγραφής της παρούσας εργασίας.

Την παρούσα εργασία την αφιερώνω στην οικογένεια μου.

Ιερωνυμάκης Κωνσταντίνος

Χανιά, Μάρτιος 2014

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΕΡΙΕΧΟΜΕΝΑ ΕΙΚΩΝΩΝ

ΠΕΡΙΕΧΟΜΕΝΑ ΠΙΝΑΚΩΝ

ΕΙΣΑΓΩΓΗ

Στην παρούσα διπλωματική εργασία εξετάζουμε ένα κλασσικό πρόβλημα της εφοδιαστικής αλυσίδας, το πρόβλημα προγραμματισμού παραγωγής. Πιο συγκεκριμένα θα επιλύσουμε το πρόβλημα παράγωγης κατά παραγγελία (job shop scheduling) με το μεθευρετικό αλγόριθμο περιορισμένης αναζήτησης (tabu search).

Αρχικά θα αναλύσουμε τι είναι προγραμματισμός παράγωγης προσδιορίζοντας το ρόλο και τη χρησιμότητα του. Θα ορίσουμε μεταβλητές και εξισώσεις που χρησιμοποιήσαμε στο πρόγραμμα μας. Έπειτα θα διαχωρίσουμε τα ειδή του προγραμματισμού παράγωγης και θα δώσουμε έμφαση στο προγραμματισμό παράγωγης κατά παραγγελία (jobshop scheduling) που είναι και το κύριο κομμάτι που θα ασχοληθούμε .

Ο συγκεκριμένος κλάδος του προγραμματισμού παράγωγης θα συνδυαστεί με τον αλγόριθμο προορισμένης αναζήτησης (tabu search) , συνεπώς θα δώσουμε έμφαση στις στρατηγικές και τους αλγορίθμους που χρησιμοποιούνται στο προγραμματισμό παράγωγης . Θα αναλύσουμε τα ειδή των αλγορίθμων και θα επικεντρωθούμε στου εφευρετικούς όπου και ανήκει ο αλγόριθμος περιορισμένης αναζήτησης. Στη συνεχεία θα παρουσιάσουμε με διάγραμμα ροής και ψευδογλώσσα το πρόγραμμα μας και θα το κατηγοριοποιήσουμε σύμφωνα με τις βασικές συναρτήσεις που χρησιμοποιήσαμε.

Σημαντικό μέρος της πτυχιακής εργασίας έχουν και τα δεδομένα που συλλέξαμε τα οποία παρουσιάζονται κατηγοριοποιημένα κάνοντας παράλληλα αναφορά στα πρόσωπα που ανήκουν. Έπειτα παρουσιάζεται με τη μορφή πινάκων και διαγραμμάτων η σύγκριση των αποτελεσμάτων που μας έδωσε το πρόγραμμα μας σε σχέση με τα βέλτιστα αποτελέσματα σε κάθε ομάδα δεδομένων. Προσπάθησα στις παρακάτω σελίδες να παρουσιαστεί ποιοτικά και ποσοτικά όσο πιο αναλυτικά γίνετε οι αποκλίσεις που είχαμε αλλά και τα καλύτερα αποξέσματα που εξάγαμε . Παρουσιάζονται οι τελικοί πινάκες με τους συνδυασμούς εργασιών-μηχανών που πρέπει να πραγματοποιηθούν ώστε να ελαχιστοποιήσουμε το χρόνο και αντίστοιχα το κόστος παράγωγης. Εξάγω κατά τη γνώμη μου σημαντικά συμπεράσματα για τη χρησιμότητα αλλά και την

αποτελεσματικότητα του προγράμματος για τη λύση του προβλήματος παράγωγης που μου ανατέθηκε.

Τέλος η παρακάτω εργασία προσπαθεί να δώσει μια σφαιρική εικόνα για το πώς προβλήματα καθημερινής φύσης μπορούν να μοντελοποιηθούν ούτος ώστε να χρησιμοποιήσουν κατά το βέλτιστο μια επαναληπτική διαδικασία που θα εξάγει χρήσιμες λύσεις.

ΚΕΦΑΛΑΙΟ 1

ΠΡΟΒΛΗΜΑ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ ΠΑΡΑΓΩΓΗΣ

1.1 Ορισμός προγραμματισμού παραγωγής (scheduling)

Ο προγραμματισμός παραγωγής έχει σημαντικό ρόλο στο τομέα της παροχής υπηρεσιών. Για να βελτιστοποιήσουμε την απόδοση του συστήματος είναι απαραίτητη η εφαρμογή χρονοδιαγράμματος και η σωστή χρήση του. Ως εκ τούτου ο κατασκευαστής ή ο πάροχος υπηρεσιών μπορεί να διατηρήσει χαμηλότερο κόστος, προκειμένου να ενισχύσει την υπεροχή του έναντι του έντονου ανταγωνισμού στο σημερινό περιβάλλον.

Ο όρος του προγραμματισμού έχει δύο σημασίες στη βιβλιογραφία. Ο πρώτος αναφέρεται στη λειτουργία, ενώ ο δεύτερος σχετίζεται με τη θεωρία. Στον πρώτο οι διαχειριστές αναζητούν τις απαντήσεις σε αυτά τα ερωτήματα:

- Τι προϊόν ή υπηρεσία παράγεται;
- Ποια θα είναι η κλίμακα της παραγωγής;
- Ποιοι πόροι θα χρησιμοποιηθούν;

Αρκετά μοντέλα μπορεί να χρησιμοποιηθούν για να βρεθούν οι απαντήσεις που απαιτούνται . Το παλαιότερο και ίσως καλύτερο γνωστό μοντέλο είναι το διάγραμμα Gantt. Το διάγραμμα Gantt αποτελείται από οριζόντιες γραμμές οι οποίες αντιπροσωπεύουν θέσεις εργασίας. Τα μήκη των ράβδων δείχνουν την διάρκεια των θέσεων εργασίας. Η Θεωρία του προγραμματισμού εστιάζεται κυρίως στα μαθηματικά μοντέλα και τις τεχνικές επίλυσης που θα βοηθήσουν στη αποδοτικότερη λειτουργία του συστήματος[1].

Ο προγραμματισμός της παραγωγής είναι η διαδικασία οργάνωσης των πόρων για την εκτέλεση όλων των δραστηριοτήτων που απαιτούνται. Είναι μια από τις πιο σημαντικές διαδικασίες λήψης αποφάσεων στη διαχείριση των επιχειρήσεων δεδομένου ότι αποτελεί μια σημαντική

βάση για τον προγραμματισμό των δραστηριοτήτων τους. Επιπλέον έχει ένα ευρύ πεδίο εφαρμογής που καλύπτει το σχεδιασμό των έργων, τη διαχείριση καταστημάτων, τα ωρολόγια προγράμματα, τη δρομολόγηση της μεταφοράς οχημάτων, κλπ.

Το πρόβλημα προγραμματισμού κατατάσσεται σε τέσσερις κατηγορίες με βάση τη διχοτόμηση του στατικού εναντίον του δυναμικού και του στοχαστικού εναντίον του ντετερμινιστικού. Η αντικειμενική συνάρτηση ορίζεται με βάση την απόδοση πολλών παραμέτρων . Τα μέτρα αυτά μπορεί να είναι ο χρόνος ροής, η πρωιμότητα, η καθυστέρηση, η αργοπορία, ο αριθμός των καθυστερημένων εργασιών, κλπ.

Οι ακόλουθες παράμετροι είναι οι βάσεις για τον υπολογισμό της αντικειμενικής συνάρτησης και έχουν ως εξής:

- Ο χρόνος επεξεργασίας (t_i): Χρονικό διάστημα που απαιτείται για την εργασία i μέχρι ολοκληρωθεί.
- Χρόνος προετοιμασίας (r_i): Χρονικό σημείο στο οποίο οι εργασίες είναι έτοιμες να υποστούν επεξεργασία .
- Ημερομηνία καταβολής (d_i): Το αργότερο χρονικό διάστημα καταβολής της εργασίας.

Η ακόλουθη παράμετρος βρίσκεται μετά το προσδιορισμό του πλήρες έργου:

- Χρόνος ολοκλήρωσης (C_i): Χρονικό σημείο στο οποίο η εργασία έχει ολοκληρωθεί

Οι ακόλουθες παράμετροι είναι βασικές ποσοτικών μέτρων βασιζόμενες στον χρόνο που απαιτείται για την αξιολόγηση του χρονοδιαγράμματος:

- Χρόνο ροής (F_i): Μήκος του χρόνου εργασίας i που ξοδεύει στο σύστημα.
- Αργοπορία (L_i): Χρονικό διάστημα όπου η ολοκλήρωση της εργασίας i υπερβαίνει την ημερομηνία λήξης.

Οι παράμετροι αυτοί υπολογίζονται ως εξής:

$F_i = C_i - R_i$

$L_i = C_i - d_i$

Η αργοπορία μπορεί να είναι αρνητική, μηδέν ή θετική. Μη αρνητικές τιμές δείχνουν καλές επιδόσεις του χρονοδιαγράμματος. Ωστόσο οι αρνητικές τιμές ευθύνονται για την κακή απόδοση του. Η βραδύτητα ορίζεται ως εξής:

$T_i = \max \{0 , L_i\}$

Ομοίως η πρωιμότητα μπορεί να οριστεί :

$E_i = \max \{0 , - L_i\}$

Τα Δρομολόγια αξιολογούνται χρησιμοποιώντας διάφορα μέτρα απόδοσης, τα μέτρα αυτά είναι συνήθως με βάση τους χρόνους ολοκλήρωσης. Ας υποθέσουμε ότι έχουμε n θέσεις εργασίας, τα πιο κοινά μέτρα μπορούν να οριστούν ως εξής[2]:

Ο μέσος χρόνος ροής : $F = \sum F_i / n$

Η μέση βραδύτητα: $T = \sum T_i / n$

Μέγιστος χρόνος ροής: $Fmax = \max\{F_i\}$

Μέγιστη καθυστέρηση: $Lmax = \max\{L_i\}$

Μέγιστης καθυστέρησης: $Tmax = \max\{T_i\}$

Makespan: $Cmax = \max\{C_i\}$

1.2 Είδη προγραμματισμού παραγωγής

Περά από τη παραγωγή κατά παραγγελία (job shop) την όποιο θα αναλύσουμε στη συνεχεία δυο είναι τα βασικά ειδή προγραμματισμού παράγωγης τα οποία έχουν αναπτυχθεί και χρησιμοποιούνται για τη βελτίωση του συστήματος.

Προγραμματισμός παράγωγης συνεχούς ροής (FLOW SHOP)

Προβλήματα προγραμματισμού συνεχούς ροής είναι εύκολο να διατυπωθούν ακόμη και αν είναι εξαιρετικά πολύπλοκα. Στα συγκεκριμένα προβλήματα υπάρχει ένα σύνολο από n παραγγελίες προϊόντων για την παραγωγή. Αυτά είναι συνήθως εργασίες. Η παραγωγή αποτελείται από m μηχανές που είναι τοποθετημένες σε σειρά. Κάθε εργασία επισκέπτεται κάθε μηχάνημα σε τάξη. Αυτή η σειρά μπορεί να είναι χωρίς απώλειες μηχανή 1, μηχανή 2 και ούτω καθ εξής μέχρι τη μηχανή m. Σαν αποτέλεσμα αυτού κάθε εργασία j, $j \in N$ έχει συγκεκριμένη αλληλουχία και εκτελείται σε ένα μηχάνημα i, $i \in M$.

Ο χρόνος επεξεργασίας των καθηκόντων αναφέρεται ως $p_{j,i}$ ο οποίος δηλώνει ουσιαστικά το μη αρνητικό γνωστό χρόνο επεξεργασίας της εργασίας j στη μηχανή i.

Το παραπάνω πρόβλημα συνίσταται στην εύρεση μιας παραγωγικής ακολουθίας των n εργασιών στις μηχανές m, έτσι ώστε να βελτιστοποιηθεί ένα δεδομένο κριτήριο απόδοσης. Ο συνολικό αριθμό των εφικτών λύσεων προέρχεται από τους πιθανούς συνδυασμούς των εργασιών στις μηχανές. Για κάθε μηχανή έχουμε n! δυνατές εργασίες μετάθεσης.

Ανοιχτό πρόβλημα προγραμματισμού ροής (OPEN SHOP)

Το πρόβλημα ανοιχτού προγραμματισμού ροής είναι μια ειδική περίπτωση του γενικού προβλήματος προγραμματισμού στο οποίο:

• κάθε εργασία i αποτελείται από m λειτουργίες, Oij (j = 1 , ... , m), όπου Oij πρέπει να υποβάλλονται σε επεξεργασία στη μηχανή Mj.

• δεν υπάρχουν σχέσεις προτεραιότητας μεταξύ των διαδικασιών.

Στη συγκεκριμένη μέθοδο δεν τίθενται περιορισμοί για τη σειρά με την οποία θα επεξεργαστούν οι εργασίες. Είναι εύκολο να φανταστούμε καταστάσεις όπου τα καθήκοντα που συνθέτουν μια εργασία μπορεί να εκτελούνται με οποιαδήποτε σειρά έστω και αν δεν είναι δυνατόν να προβούν σε περισσότερες από μία εργασίες παράλληλα.

1.3 Πρόβλημα παραγωγής κατά παραγγελία (Job shop)

Το Πρόβλημα παραγωγής κατά παραγγελία είναι ένα από τα πιο χαρακτηριστικά και πολύπλοκα προβλήματα στο τομέα του προγραμματισμού παράγωγης . Ο σκοπός του είναι να διαθέσει n θέσεις εργασίας σε m μηχανές προκειμένου να βελτιστοποιηθεί το μετρό απόδοσης του συστήματος [3]. Παραδοσιακά υπάρχουν τρεις προσεγγίσεις για την επίλυση του, οι κανόνες προτεραιότητας, η συνδυαστικής βελτιστοποίησης και οι περιορισμοί ανάλυσης[4]. Ο προγραμματισμός παράγωγης κατά παραγγελία έχει τη δυνατότητα παραγωγής προϊόντων με διαφορετικό χρόνο εκτέλεσης για κάθε εργασία. Λόγω των διαφορετικών λειτουργιών για κάθε προϊόν και των απαιτήσεων των μηχανών κατά την επεξεργασία του είναι αρκετά δύσκολο να βρούμε μια αποτελεσματική λύση προγραμματισμού.

Το πρόβλημα παραγωγής κατά παραγγελία ανήκει στη κατηγορία προβλημάτων της συνδυαστικής βελτιστοποίησης γνωστά ως NP - Hard 1. Τις τελευταίες τρεις δεκαετίες πολλοί ερευνητές έχουν δήξει ενδιαφέρον για τέτοια προβλήματα. Το συγκεκριμένο πρόβλημα είναι ένα από τα πιο σημαντικά θέματα στον σχεδιασμό της παράγωγης όπως επίσης είναι σημαντικό γιατί καθορίζει τη διαδικασία, τους χάρτες και τις δυνατότητες για την πλειονότητα των βιομηχανιών. Στη περίπτωση μας έχουμε n θέσεις εργασιών και m μηχανές, υπάρχουν (n1) ! (n2) ! . . . (nm) ! αλληλουχίες όπου nκ είναι ο αριθμός των ενεργειών που πρέπει να γίνουν από k μηχανές.

Φυσικά, δεν είναι όλες οι λύσεις εφικτές. Η καλύτερη αλληλουχία πρέπει να ικανοποιήσει τους περιορισμούς και να βελτιστοποιεί ένα ή περισσότερα κριτήρια. Ωστόσο είναι αδύνατο να αξιολογηθούν όλες οι λύσεις σε ένα εύλογο χρονικό διάστημα. Έτσι ένας μεγάλος αριθμός από αλγόριθμους έχουν αναπτυχθεί ώστε να επιτευχθούν οι κατάλληλες αλληλουχίες εργασιών σε μια διαδικασία κατασκευής. Αυτοί οι αλγόριθμοι μπορούν να ταξινομηθούν σε στατικούς και δυναμικούς με βάση την ώρα της απόφασης.

Σε στατικά προβλήματα η προτεραιότητα των θέσεων εργασίας εντοπίζεται κατά την επεξεργασία. Από την άλλη πλευρά σε μια δυναμική κατάσταση η αλληλουχία που προσδιορίζει τις εργασίες μπορεί να αλλάξει από το ένα μηχάνημα στο άλλο βασιζόμενο σε

διαφορετικές καταστάσεις. Δυναμικοί κανόνες είναι πιο συχνοί από στατικούς στο πραγματικό κόσμο της βιομηχανίας.

Ας υποθέσουμε ότι υπάρχουν m μηχανές και n εργασίες σε ένα πρόβλημα παραγωγής κατά παραγγελία. Κάθε πράξη σε μια εργασια έχει δική της επεξεργασία χρόνου. Σε αυτό το πρόβλημα, η λειτουργία k κάθε εργασίας γίνεται για τη μηχανή m. Αυτό το πρόβλημα μπορεί να έχει κάποιους περιορισμούς στην αρχή και στο τέλος κάθε εργασίας.

Ο περιορισμός του χρόνου έναρξης υπάρχει επειδή η ώρα έναρξης δεν μπορεί να είναι νωρίτερα από ένα ορισμένο χρονικό σημείο. Ο περιορισμός του χρόνου παράδοσης σημαίνει ότι επιβάλετε ποινή για παράδοση νωρίτερα ή αργότερα από την προκαθορισμένο χρόνο. Οι δυο ποινές μπορεί να είναι διαφορετικές ανάλογος τις θέσεις εργασίας. Ο στόχος της επίλυσης αυτού του προβλήματος είναι να προσδιορίσει τις αλληλουχίες εργασιών και μηχανημάτων προκειμένου να βελτιστοποιηθεί η απόδοση του συστήματος.

Οι ακόλουθοι στόχοι μπορεί να πραγματοποιηθούν σε προβλήματα προγραμματισμού:

• Μεγιστοποίηση της χρήσης των συστημάτων ή των πόρων.

• Ελαχιστοποίηση της αδράνειας σε κάθε μηχάνημα που προκαλείται από την πρωιμότητα της διαδικασίας λειτουργίας ή την αργοπορία στη λήψη των θέσεων εργασίας από μια μηχανή.

Αυτοί οι δύο στόχοι είναι παράλληλοι και σε ορισμένες περιπτώσεις η βέλτιστη λύση αλλάζει ανάλογα με το είδος του εκάστοτε στόχου. Η υπόθεση αυτή είναι πιο εμφανής όταν υπάρχουν διαφορετικές κυρώσεις για κάθε μηχάνημα.

Προβλήματα προγραμματισμού μπορεί να καταταχτούν σύμφωνα με τα κύρια κριτήρια, όπως τις απαιτήσεις, την πολυπλοκότητα της διαδικασίας, τα κριτήρια προγραμματισμού και το περιβάλλον προγραμματισμού. Σύμφωνα με το πρώτο κριτήριο δηλαδή τις απαιτήσεις υπάρχουν δύο στάδια: ανοικτού και κλειστού production shop που βασίζονται στις απαιτήσεις της παραγωγής.

Το δεύτερο κριτήριο εξαρτάται από τα στάδια τις διαδικασίες και τον αριθμός των σταθμών εργασίας. Αυτά μπορεί να κατηγοριοποιηθούν ως εξής:

- ένα στάδιο, μία διεργασία
- ένα στάδιο, πολλές διεργασίες
- πολλαπλά στάδια, γραμμή παραγωγής
- σε πολλαπλά στάδια παραγωγή κατά παραγγελία.

Στον πρώτο τύπο υπάρχει ένας επεξεργαστής και ένα στάδιο. Η δεύτερη κατηγορία αφόρα ένα στάδιο και πολλαπλούς επεξεργαστές. Στην τρίτη υποκατηγορία κάθε εργασία αποτελείται από πολλές διεργασίες. Στην τέταρτη υποκατηγορία είναι δυνατόν να εκχωρήσουμε έναν αριθμό μηχανημάτων και τη διαδρομή των εργασιών σε ένα έργο.

ΚΕΦΑΛΑΙΟ 2

ΕΞΕΛΙΚΤΙΚΕΣ ΣΤΡΑΤΗΓΙΚΕΣ ΚΑΙ ΑΛΓΟΡΙΘΜΟΙ

2.1 Ανάλυση αλγορίθμων (κύριοι αλγόριθμοι)

Μεθευρετικοί αλγόριθμοι όπως η λίστα περιορισμένης αναζήτησης (tabu search), οι γενετικοί αλγόριθμοι, η προσομοιωμένη ανόπτηση, νευρωνικά δίκτυα κλπ, έχουν χρησιμοποιηθεί για την επίλυση σημαντικών προβλημάτων συνδυαστικής βελτιστοποίησης. Στόχος αυτών των αλγορίθμων είναι η αναζήτηση του χώρου λύσεων πιο αποτελεσματικά από τις συμβατικές μεθόδους.

Οι πρώτοι γενετικοί αλγόριθμοι αναπτύχθηκαν και περιγράφηκαν πρώτα από τον J.Holland το 1960[5] και αργότερα από τον Goldberg το 1989[6]. Οι γενετικοί αλγόριθμοι προσομοιώνουν τη διαδικασία της εξέλιξης που συναντούμε στη φύση. Ένας πληθυσμός εξελίσσεται με την πάροδο των γενεών προκειμένου να προσαρμοστεί στο περιβάλλον του και να αντιμετωπίσει καλύτερα τα διάφορα προβλήματα που παρουσιάζονται. Ένας γενετικός αλγόριθμος είναι μια επαναληπτική στοχαστική διαδικασία όπου ένας σταθερός από άποψη πλήθους πληθυσμός πολλαπλασιάζεται για τη δημιουργία καλύτερων γενεών. Ο πληθυσμός των γενετικών αλγορίθμων αποτελείται από λύσεις του προβλήματος κατάλληλα κωδικοποιημένες σε μορφή χρωμοσωμάτων. Αντίθετα με τη φύση κάθε χρωμόσωμα αντιπροσωπεύει και μια λύση. Ξεκινώντας από ένα αρχικό πληθυσμό από εφικτές λύσεις οι λύσεις αυτές αξιολογούνται με συγκεκριμένο κριτήριο και με κατάλληλες διαδικασίες επιλογής και αναπαραγωγής που θα δούμε παρακάτω, πολλαπλασιάζονται δύο από αυτές (γονείς) και δημιουργούν δύο νέες λύσεις (απόγονους) μέχρι να δημιουργηθεί μια νέα γενιά χρωμοσωμάτων, όμοια σε πλήθος με την προηγούμενη. Ιδεατά κάποια από αυτές τις λύσεις θα είναι καλύτερη από την καλύτερη λύση της προηγούμενης γενιάς. Έτσι μέσα στον πληθυσμό θα υπάρχουν καλύτερα γονίδια τα οποία με τη σειρά τους θα οδηγήσουν στην δημιουργία ακόμα καλύτερων γενεών. Επιπλέον έχουμε και τη διαδικασία της μετάλλαξης. Ένας μικρός αριθμός του πληθυσμού μεταλλάσσεται τυχαία με σκοπό να εισαχθούν νέα γονίδια στον

πληθυσμό και να αποτρέψουν τη λύση να είναι υποβέλτιστη. Η διαδικασία αυτή επαναλαμβάνεται 10 μέχρι να γίνει αληθής μια συνθήκη εξόδου οπότε και τερματίζεται ο αλγόριθμος.

Οι μιμητικοί αλγόριθμοι αντιπροσωπεύουν έναν από τους πρόσφατα αναπτυσσόμενους τομείς έρευνας της εξελικτικής υπολογιστικής. Ο όρος μιμητικοί αλγόριθμοι χρησιμοποιείται ευρέως για την περιγραφή μεθόδων που αποτελούνται από τη συνεργεία μιας εξελικτικής ή πληθυσμιακής προσέγγισης με ξεχωριστή μάθηση για κάθε υποκείμενο ή διαδικασίες τοπικής βελτίωσης για την αναζήτηση του χώρου λύσεων του προβλήματος. Η τοπική αναζήτηση (local search) είναι μια μεθευρετική μέθοδος για την επίλυση δύσκολων και πολύπλοκων υπολογιστικά προβλημάτων βελτιστοποίησης. Γενικά μπορεί να περιγραφεί σαν μια μέθοδος που βρίσκεται μια λύση όπου μεγιστοποιείται ένα κριτήριο από ένα σύνολο πιθανών λύσεων. Οι αλγόριθμοι τοπικής αναζήτησης κινούνται από λύση σε λύση στο χώρο των πιθανών λύσεων (χώρος λύσεων του προβλήματος) μέχρι να βρεθεί μια λύση που ορίζεται ως βέλτιστη ή ξεπεραστεί κάποιος περιορισμός χρόνου εκτέλεσης[7].

2.2 Μεθευρετικοί αλγόριθμοι

Οι μεθευρετικοί αλγόριθμοι είναι αλγόριθμοι επίλυσης που συνδυάζουν διαδικασίες τοπικής αναζήτησης και υψηλότερου επιπέδου στρατηγικές για να δημιουργήσουν μια διαδικασία που είναι ικανή να ξεφύγει από κάποιο τοπικό ελάχιστο. Σε αυτή την κατηγορία αλγορίθμων εξερευνάται το πεδίο της λύσης με σκοπό να βρεθεί μια καλύτερη λύση.

Τα βασικά χαρακτηριστικά των μεθευρετικών αλγορίθμων είναι τα ακόλουθα [8]:

- μοντελοποιούν ένα φαινόμενο που υπάρχει στη φύση
- μπορούν να μεταφερθούν εύκολα σε παράλληλη μορφή
- είναι προσαρμοστικοί αλγόριθμοι

Κάποιοι από τους μεθευρετικούς αλγορίθμους είναι οι εξής:

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

2.3 Λίστα περιορισμένης αναζήτησης (Tabu Search)

Η λίστα περιορισμένης αναζήτησης (Tabu Search) ως μεθευρετικός αλγόριθμος χρησιμοποιεί ένα μηχανισμό για την αποφυγή του εγκλωβισμού σε τοπικό ελάχιστο και την αποδοτική αναζήτηση στο χώρο λύσεων. Η διαφορά με τους υπόλοιπους μεθευρετικούς αλγορίθμους είναι ότι αυτός ο μηχανισμός δεν χρησιμοποιεί την τύχη όπως είδαμε στους γενετικούς και τους μιμητικούς αλλά την απομνημόνευση των κινήσεων που έχουν γίνει στο παρελθόν. Κατά την εκτέλεση του TS γίνεται μετακίνηση από μια λύση στην καλύτερη γειτονική ακόμα και αν αυτό σημαίνει χειροτέρευση σε σχέση με την καλύτερη τρέχουσα. Αυτή η στρατηγική επιτρέπει στην αναζήτηση να ξεφύγει από το τρέχον τοπικό βέλτιστο και να ερευνήσει άλλες περιοχές του χώρου λύσεων. Για να αποφευχθεί η μετάβαση σε κάποια λύση που έχει ήδη ελεγχθεί όλες οι πρόσφατες κινήσεις που έχουν γίνει για την μετάβαση από μια λύση σε μια άλλη καταγράφονται σε μια λίστα περιορισμένου μεγέθους. Αυτό έχει σαν αποτέλεσμα την αποφυγή επανάληψης των ίδιων κινήσεων ξανά και ξανά, εξαλείφοντας τον κίνδυνο προσκόλλησης σε συγκεκριμένη περιοχή του χώρου λύσεων[9].

Πιο αναλυτικά, ο αλγόριθμος TS χρησιμοποιεί δύο στρατηγικές για την αποδοτική αναζήτηση του χώρου λύσεων. Αυτές οι στρατηγικές είναι η εντατικοποίηση (intensification) της αναζήτησης γύρω από ένα τοπικό βέλτιστο και την διάχυση (diversification) της έρευνας σε νέες περιοχές του εφικτού συνόλου. Ο συγκεκριμένος αλγόριθμος είναι μια τεχνική καθοδήγησης (guiding technique) της τοπικής αναζήτησης βασισμένη σε προσαρμοστική μνήμη, δίνοντας στην περιορισμένη αναζήτηση της ιδιότητες μιας μεθευρετικής μεθόδου. Είναι μέθοδος προσδιοριστικού (deterministic) χαρακτήρα που προσπαθεί να επιτύχει ότι θεωρητικά υπόσχεται η προσομοιωμένη ανόπτηση (simulated annealing) που είναι μέθοδος τυχαιοποίησης (στοχαστικού χαρακτήρα).

Έτσι τα βασικά χαρακτηριστικά της μεθόδου μπορούν να συνοψιστούν στα εξής τέσσερα σημεία:

• Μνήμη "Η" βασισμένη στην ιστορία της τοπικής αναζήτησης σε αντίθεση προς την τοπική αναζήτηση που δεν διαθέτει μνήμη.

• Περιορισμοί Tabu που βασιζόμενοι στην μνήμη απαγορεύουν συγκεκριμένες κινήσεις (κυρίως στοιχειώδεις πράξεις σε λύσεις για μετάβαση σε άλλες):

- Κριτήρια φιλοδοξίας (aspiration) που επιτρέπουν υπέρβαση των περιορισμών της μεθόδου π.χ. εάν η χρήση κάποιων απαγορευμένων κινήσεων παρήγαγαν λύση καλύτερη από όλες που έχουν παραχθεί μέχρι εκείνη τη στιγμή, π.χ $f(Ʊ,x) \neq f(\chi)$.
- Κριτήρια διάχυσης (diversification) που επιτρέπουν την επιβολή ιδιοτήτων λύσεων που ιστορικά παρήγαγαν καλά αποτελέσματα και την διάχυση της αναζήτησης σε νέες περιοχές εφικτών λύσεων: Η $N(Ʊ, x^{(k)})$ μπορεί να περιέχει λύσεις εκτός $N(x^{(k)})$.

Το κυριότερο χαρακτηριστικό που ενσωματώθηκε στον αλγόριθμο που κατασκευάσαμε από την μέθοδο περιορισμένης αναζήτησης ήταν αυτό της μνήμης με τη μορφή μιας λίστας περιορισμένης αναζήτησης (Tabu List) και των tabu περιορισμών. Η χρήση αυτού του χαρακτηριστικού έγινε με τη δημιουργία μιας δομής δεδομένων τύπου ουράς όπου κάθε στοιχείο της περιέχει τα τρία διανύσματα με τα οποία κωδικοποιούμε την κάθε λύση. Η

λίστα αρχικά είναι άδεια και κάθε φορά που ο γενετικός αλγόριθμος πραγματοποιεί την διαδικασία διασταύρωσης δύο λύσεων (ενός ή δύο σημείων) ελέγχει εάν κάποια από τις λύσεις που προέκυψαν υπάρχει στη λίστα περιορισμένης αναζήτησης. Στην περίπτωση που συμβαίνει αυτό οι δύο νέες λύσεις-απόγονοι απορρίπτονται και επιλέγονται από την αρχή δύο νέοι γονείς προς διασταύρωση και γίνεται επανάληψη της διαδικασίας. Στην αντίθετη περίπτωση οι δύο νέες λύσεις προστίθενται στο τέλος της λίστας. Εάν η λίστα είναι γεμάτη τότε οι πιο παλιές λύσεις της λίστας διαγράφονται. Με αυτό τον τρόπο η λίστα διατηρεί ένα σταθερό μέγεθος κατά την εκτέλεση του αλγορίθμου δηλαδή ο ορίζοντας των κινήσεων που "βλέπει" είναι περιορισμένος. Το μέγεθος της λίστας προσαρμόζεται σύμφωνα με το μέγεθος του προβλήματος και ο μέγιστος αριθμός των στοιχείων που μπορεί να περιέχει ορίζεται στο τετραπλάσιο πλήθος στοιχείων από αυτό των πελατών[10].

Ο έλεγχος αυτός γίνεται μόνο κατά τη διαδικασία διασταύρωσης του διανύσματος δρομολόγησης. Αυτό συμβαίνει επειδή πάρα πολλές καλές λύσεις μοιράζονται τα διανύσματα αποθηκών και δείκτη ενώ μια νέα λύση δεν είναι ολοκληρωμένη πριν από τη δημιουργία του νέου διανύσματος δρομολόγησης. Έτσι η εισαγωγή των διανυσμάτων σε ξεχωριστές λίστες δεν έχει νόημα αφού μόνο με τη χρήση και των τριών μπορεί να περιγραφεί μια λύση ολοκληρωμένα, ενώ πριν ολοκληρωθεί η διαδικασία της διασταύρωσης των διανυσμάτων που περιέχουν τη σειρά επίσκεψης των πελατών δεν υπάρχουν σχηματισμένες νέες λύσεις και στις δύο μεθόδους επίλυσης (δύο φάσεις ταυτόχρονα ή ξεχωριστά).

Ο τρόπος με τον οποίο απορρίπτονται ή γίνονται αποδεκτές οι λύσεις που προκύπτουν από τις μεθόδους τοπικής αναζήτησης μοιάζει αρκετά με τον τρόπο που λειτουργεί το κριτήριο φιλοδοξίας στον Tabu Search. Οι λύσεις που προκύπτουν από αυτές τις μεθόδους γίνονται αποδεκτές μόνο εάν δίνουν καλύτερο κόστος σε σχέση με τις αρχικές οπότε δεν έχει νόημα ο έλεγχος στην Tabu List ενώ στην περίπτωση που δίνουν χειρότερο κόστος σε σχέση με πριν απορρίπτονται ούτως ή άλλως με τις αρχικές να υπάρχουν ήδη στη λίστα[11].

Η υλοποίηση της λίστας περιορισμένης αναζήτησης έγινε με τη χρήση δύο βοηθητικών συναρτήσεων και ενός πίνακα (array) σταθερού μεγέθους με μια μεταβλητή δείκτη η οποία μας δείχνει το τρέχον στοιχείο - κεφαλή της

λίστας. Αρχικά η λίστα είναι κενή. Με τη βοήθεια μιας συνάρτησης εισαγωγής στοιχείου, εκχωρείται το στοιχείο που δίνουμε στο σημείο που μας δείχνει η μεταβλητή και η τιμή της αυξάνεται κατά μια μονάδα έτσι ώστε να δείχνει την επόμενη θέση στη λίστα. Όταν αυτή η μεταβλητή πάρει τη μέγιστη τιμή της (μέγεθος λίστας) τότε της δίνεται η κατάλληλη τιμή ώστε να δείχνει στο πρώτο στοιχείο του πίνακα. Με αυτό τον τρόπο η μεταβλητή δείχνει πάντα το στοιχείο που βρίσκεται τον περισσότερο χρόνο στη λίστα δημιουργώντας μια δομή δεδομένων τύπου ουράς όπου επικρατεί η συνθήκη first in first out. Μπορούμε να δούμε πως λειτουργεί με μορφή ψευδοκώδικα:

method tabulistinsert

input tabulist, tbsize, tbstart, item

tabulist(tbstart) ←item

if tbstart = tbsize, **then**

tbstart ← 1

else

tbstart ←tbstart + 1

end if

return tabulist, tbstart

Η αναζήτηση στη λίστα γίνεται επίσης με τη χρήση μιας βοηθητικής συνάρτησης. Στη συνάρτηση δίνεται το στοιχείο προς αναζήτηση και στη συνέχεια γίνεται αναζήτηση στη λίστα για την ύπαρξη αυτού του στοιχείου. Αν η αναζήτηση ήταν επιτυχής και βρέθηκε το ζητούμενο στοιχείο η συνάρτηση επιστρέφει την τιμή 1, ενώ σε διαφορετική περίπτωση την τιμή 0. Εδώ παρουσιάζεται η συνάρτηση αναζήτησης σε μορφή ψευδοκώδικα:

method tabulistsearch

input tabulist, tbsize, searchitem

value ←0

for i: 1 → tbsize

if tabulist(i) = searchitem, **then**

returnvalue ←1

exit for loop

end if

end for

return value // returns value 1 if search item in list, 0 otherwise.

Στη συνέχεια παρουσιάζεται σε μορφή ψευδοκώδικα ο τρόπος με τον οποίο συνδυάζεται η λίστα περιορισμένης αναζήτησης στον αλγόριθμο κατά τη διαδικασία διασταύρωσης λύσεων.

method Tabu List implementation

input oldgeneration, tabulist, tbstart, populationsize

i ←1

do

parent1 ←roulette(oldgeneration)

```
        parent2 ←roulette(oldgeneration)
        offspring1, offspring2 ←crossover(parent1, parent2)
        if tabulistsearch(offspring1) = 0, then //Not in list
           if  tabulistsearch(offspring2) = 0,  then //Not in list
              tabulistinsert(offspring1)
              newgeneration(i) ←offspring1
              tabulistinsert(offspring2)
              newgeneration(i+1) ← offspring2
              i ←i+2
           end if
        end if
    while  (i < populationsize)
    return newgeneration
```

Για προβλήματα με μεγάλο πλήθος πελατών η Tabu List μπορεί να πάρει πολύ μεγάλο μέγεθος. Ακόμη τα προβλήματα αυτά χρησιμοποιούν μεγαλύτερα διανύσματα λόγω του μεγάλου αριθμού των πελατών που πρέπει να εξυπηρετήσουν. Έτσι είναι επόμενο η αναζήτηση στη λίστα περιορισμένης αναζήτησης να καθυστερεί τον αλγόριθμο σε βάρος της παραγωγής νέων γενιών και έτσι δεν είναι εξασφαλισμένη η επίτευξη καλύτερων αποτελεσμάτων σε σχέση με την περίπτωση που δεν γίνεται χρήση της. Όπως θα δούμε παρακάτω η επιτυχία ή όχι του συνδυαστικού αλγορίθμου εξαρτάται από τη φύση του προβλήματος.

ΚΕΦΑΛΑΙΟ 3

ΠΕΡΙΓΡΑΦΗ ΑΛΓΟΡΙΘΜΟΥ ΕΠΙΛΥΣΗΣ

3.1 Ανάλυση της διαδικασίας ανάπτυξης του αλγορίθμου επίλυσης

Σε αυτό το κεφάλαιο θα παρουσιάσουμε τον αλγόριθμο σε βήματα δείχνοντας τη πορεία που ακολουθήσαμε **.**

Η υλοποίηση του αλγορίθμου για την επίλυση του προβλήματος job shop scheduling έγινε με τη χρήση της μαθηματικής γλώσσας προγραμματισμού Matlab. Το matlab είναι ένα αριθμητικό υπολογιστικό περιβάλλον και τετάρτης γενεάς γλώσσα προγραμματισμού .

Ανεπτυγμένο από την εταιρία Math works το Matlab επιτρέπει το χειρισμό πινάκων, συναρτήσεων και γενικά αριθμητικών δεδομένων .

Το πρόγραμμα αποτελείται από 1 αρχείο συνάρτησης. Πριν την έναρξη της βελτιστοποίησης ζητείται από το χρήστη η εισαγωγή ενός αρχείου μορφής Microsoft office excel worksheet(.xls)

ΣΥΝΑΡΤΗΣΗ

Αρχικά δημιουργήσαμε τη συνάρτηση η οποία δέχεται τον πίνακα δεδομένων μέσα στον οποίο υπάρχουν δύο ίσοι πίνακες με γραμμές που αποτελούν της μηχανές και στήλες της εργασίες που θα πραγματοποιηθούν. Ο πρώτος πίνακας θα περιέχει τους χρόνους ενώ ο δεύτερος της προτεραιότητες που πρέπει να ακολουθήσουν.

Πίνακας 1: Πινάκας χρονών

5	8	9	15
20	30	40	60
80	100	120	140

Πίνακας 2: Πινάκας προτεραιοτήτων

3	2	1	2
2	3	3	1
1	1	2	3

Στη συνέχεια η συνάρτηση θα μας δίνει ένα τυχαίο πίνακα μεγέθους όσο οι δύο παραπάνω στον οποίο θα υπάρχει μια αρχική τυχαία λύση. Αυτή η τυχαία λύση θα διορθώνετε σύμφωνα με το πίνακα προτεραιοτήτων.

Πίνακας 3: Τυχαίος πινάκας

5	8	1	4
7	9	3	11
6	10	2	12

Πίνακας 4: Τυχαίος διορθωμένος πινάκας

7	9	1	11
9	10	3	4
5	8	2	12

Επόμενο βήμα στη συνάρτηση είναι ο υπολογισμός του makespan. Για να το υπολογίσουμε χρησιμοποιούμε το ταξινομημένο τυχαίο πίνακα αλλά και το πίνακα με τους χρόνους η διαδικασία υπολογισμού είναι η εξής:

Δημιουργούμε ένα μηδενικό πίνακα μεγέθους (nxm)

Ξεκινάμε από το πρώτο στοιχείο του ταξινομημένου τυχαίου πηγαίνοντας στο αντίστοιχο του μηδενικού και ελέγχουμε για αυτή τη στήλη και αυτή τη γραμμή αν υπάρχει κάποιο μεγαλύτερο, αν όχι τότε προσθέτουμε σε αυτή τη θέση το στοιχείο που υπάρχει στην αντίστοιχη του πίνακα των χρόνων αλλιώς στη μεγαλύτερη τιμή που θα βρούμε κατά τον έλεγχο.

Η μεγαλύτερη τιμή που θα υπάρχει στο πίνακα μετά την ολοκλήρωση της διαδικασίας είναι το makespan που ψάχνουμε.

Πίνακας 5: Κατά τη διάρκεια του γεμίσματος

0	0	9	0
0	0	169	229
209	0	129	0

Πίνακας 6: Ο πινάκας αφού έχει γεμίσει

254	317	9	332
249	347	169	229
209	309	129	472

Το παραπάνω παράδειγμα μας δίνει makespan 472 το όποιο αποθηκεύω προσωρινά σαν καλύτερο αποτέλεσμα μέχρι η tabu search να μας δώσει καλύτερη τιμή.

Στη συνεχεία παρουσιάζω τη συνάρτηση που γράψαμε στη matlab με μορφή ψευδογλωσσας.

ΣΥΝΑΡΤΗΣΗ

```
ΔΙΑΒΑΣΕ num_of_machines

ΔΙΑΒΑΣΕ num_of_jobs


Final ← TXL



Για i από 1 μέχρι num_of_jobs με βήμα 1

    temp_txl ← TXL(:,i)

    temp_pr ← PR(:,i)

    temp_txl ← sort(temp_txl)

    final(:,I )← temp_txl(temp_pr)

Τέλος_επανάληψης



C ← 0



Για i από 1 μέχρι num_of_machines*num_of_jobs με βήμα 1

    [x,y] ← find(final==i);
```

```
temp_grami ← C(x,:);

temp_stili ← C(:,y);

temp_value ← max(max(temp_grami),max(temp_stili));

C(x,y) ←temp_value+XR(x,y);
```

Τέλος_επανάληψης

ΚΥΡΙΩΣ ΠΡΟΓΡΑΜΜΑ (TABU SEARCH)

Στη συνέχεια του προγράμματος προσπαθώ μέσω επαναληπτικής διαδικασίας να έχω καλύτερο αποτέλεσμα δηλαδή μικρότερο makespan .

Αρχικά στη τυχαία λύση που δημιούργησα στη συνάρτηση θα κάνω μια νέα αλλαγή μεταξύ δύο στοιχειών.

Στη συνέχεια θα διορθώσω το πίνακα σύμφωνα με αυτό των προτεραιοτήτων.

Υπολογίζω το καινούργιο makespan και το συγκρίνω με το προηγούμενο κρατώντας το μικρότερο.

Η παραπάνω διαδικασία επαναλαμβάνετε σε κάθε παράδειγμα για 10000 φορές έχοντας όμως ως περιορισμό ότι ανά 100 επαναλήψεις θα δημιουργώ ένα καινούριο τυχαίο πίνακα και θα το συγκρίνω με τον καλύτερο ήδη υπάρχων τυχαίο , αποφεύγοντας με αυτό το τρόπο να εγκλωβιστούμε σε τοπικό ελάχιστο.

TABU LIST

Η TABU LIST είναι η λίστα την οποία έχω δημιουργήσει στο πρόγραμμα μου , για να αποθηκεύω σε κάθε επανάληψη τα δύο στοιχεία που θα αλλάξω θέση στο πίνακα μου .

Και στη συγκεκριμένη περίπτωση όμως για να αποφύγω της ιδέες αλλαγές σε σύντομο χρονικό διάστημα έχω δημιουργήσει περιορισμό ο οποίος μου απαγορεύει στις 10 επόμενες επαναλήψεις να γίνει η αλλαγή ανάμεσα σε δύο ίδια στοιχεία .

Στη συνεχεία παρουσιάζω το κώδικα που γράψαμε στη matlab με μορφή ψευδογλωσσας:

```
num_of_machines ← stiles;

num_of_jobs ← grammes;

megethos ← num_of_machines*num_of_jobs

TX ← randperm(megethos)

TXL ← reshape(TX,num_of_machines,num_of_jobs)

original_TXL ← TXL

[C_best,current_best] ← find_makespan(original_TXL,XR,PR)

tabu_list ← zeros(10000,2)

tabu_list(1:megethos,1) ← randperm(megethos)
```

```
tabu_list(1:megethos,2) ← randperm(megethos)

Για i από megethos+1 μέχρι 10000 με βήμα 1

    tabu_list(i,1) ← floor(rand()*megethos)+1
    tabu_list(i,1) ← floor(rand()*megethos)+1
    tabu_list(i,2) ← floor(rand()*megethos)+1

    Όσο (tabu_list(i,1)==tabu_list(i,2)) επανέλαβε
        tabu_list(i,2) ← floor(rand()*megethos)+1
    Τέλος_επαναληψης


    tmp_place ← find(tabu_list(i-1-10:i-1,1)==tabu_list(i,1))

    Αν ~isempty(tmp_place) τοτε

        Αν tabu_list(tmp_place,2)==tabu_list(i,2) τοτε

            Όσο (tabu_list(i,2)==tabu_list(tmp_place,2)) επανέλαβε
                tabu_list(i,2) ← floor(rand()*megethos)+1
            τελος_επαναληψης

        Τελος_αν
     Τελος_αν
 Τελος_αν

 i ← 1
```

```
Όσο (i<10000) επανέλαβε

    TXL ← original_TXL
    idx1 ← find(TXL==tabu_list(i,1))
    idx2 ← find(TXL==tabu_list(i,2))
    TXL(idx2) ← tabu_list(i,1)
    TXL(idx1) ← tabu_list(i,2)

    [C,final] ← find_makespan(TXL,XR,PR)

    Αν max(max(C))<max(max(C_best)) τότε

        current_best ← final
        C_best ← C
        original_TXL ← TXL

    τελος_αν

    Αν (mod(i,100)==0) τότε

        TX ← randperm(megethos)
        Reshape (TX,num_of_machines,num_of_jobs)
        TXL_new ← reshape(TX,num_of_machines,num_of_jobs)
        [C_new,final_new] ← find_makespan(original_TXL,XR,PR)

        Αν max(max(C_new))<max(max(C_best)) τότε
            current_best ← final_new
```

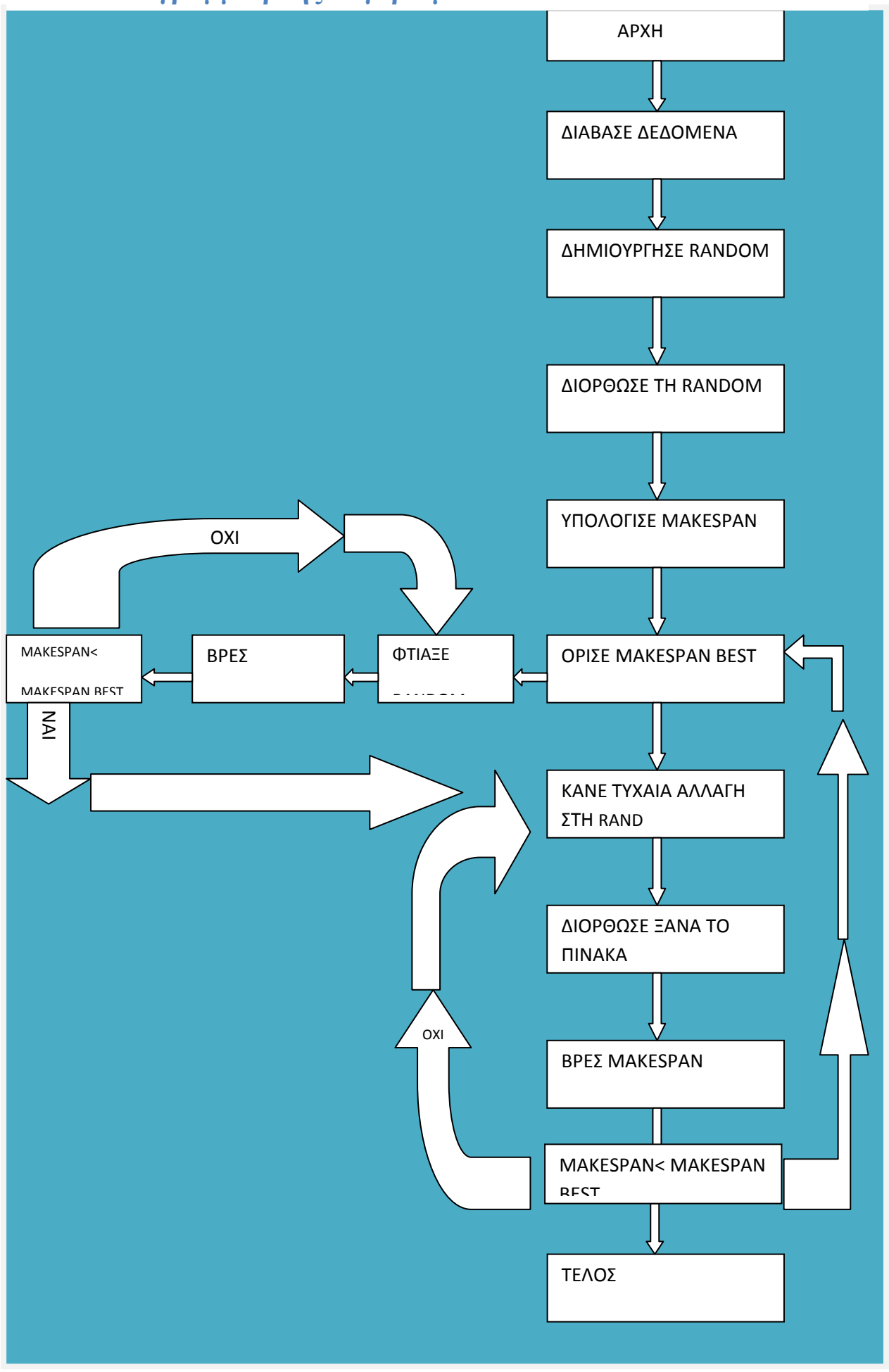
```
            C_best ← C_new
            original_TXL ← TXL_new

        τελος_αν
    Τελος_αν

    Εκτύπωσε i
    Εκτύπως max(max(C_best))
    i ← i+1;

Τελος_αν
```

3.2 Διάγραμμα ροής αλγορίθμου



ΚΕΦΑΛΑΙΟ 4

ΑΠΟΤΕΛΕΣΜΑΤΑ

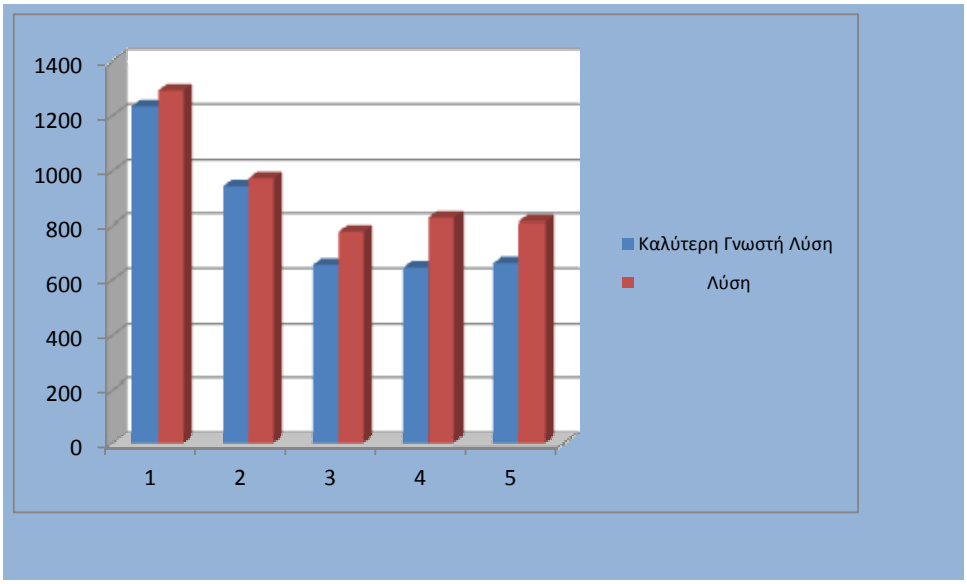
4.1 Αναλυτική παρουσίαση πρώτης σειράς δεδομένων

Η πρώτη σειρά δεδομένων είναι η Abz5- Abz9 είναι από J. Adams, E. Balas and D. Zawack (1988). Παρακάτω θα σας παρουσιάσω το κάθε παράδειγμα ξεχωριστά καθώς και τα αποτελέσματα της επαναληπτικής διαδικασίας.

Πίνακας 7: Πινάκας σύγκρισης

Παράδειγμα	**Μέγεθος**	**ΚαλύτερηΓνωστή Λύση**	**Λύση**
ABZ5	10×10	1234	1294
ABZ6	10×10	943	972
ABZ7	20×15	656	779
ABZ8	20×15	646	826
ABZ9	20×15	662	812

Γραφική απεικόνιση της σύγκρισης ανάμεσα στη καλύτερη λύση και στη λύσης της επαναληπτικής διαδικασίας.

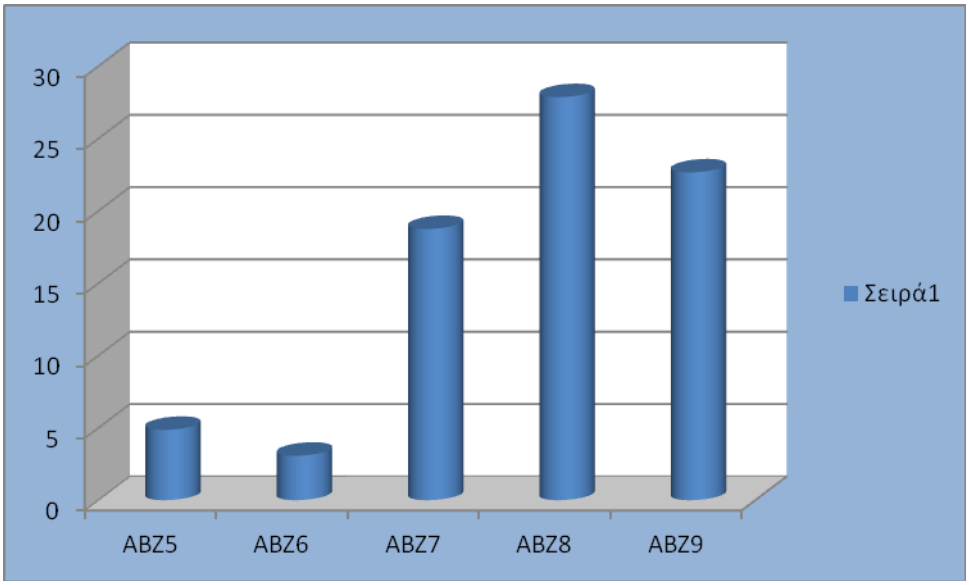


Εικόνα 1: Διάγραμμα σύγκρισης

Πίνακας 8: Πινάκας αποκλίσεων

ABZ5	**4.862237**
ABZ6	3.075292
ABZ7	18.75
ABZ8	27.86378
ABZ9	22.65861

Γραφική απεικόνιση της απόκλισης για ABZ5-ABZ9 :



Εικόνα 2: Διάγραμμα αποκλίσεων

Από το πινάκα μας και το γράφημα συμπεραίνουμε ότι καλύτερο αποτέλεσμα είχαμε ABZ 6 στο όποιο έχουμε και τη μικρότερη απόκλιση σε σχέση με τα υπόλοιπα παραδείγματα αυτής της σειράς.

ΠΑΡΑΔΕΙΓΜΑ: ABZ 5

Τα δεδομένα του προβλήματος παρουσιάζονται παρακάτω , οι γραμμές αντιπροσωπεύουν μηχανές και οι στήλες εργασίες. Το ακόλουθο παράδειγμα αφορά πρόβλημα 10 μηχανών και 10 εργασιών.

Πίνακας 9: Πινάκας προτεραιοτήτων

5	9	7	6	2	3	10	8	1	4
6	4	7	5	3	9	1	2	8	10
10	9	1	2	7	6	8	5	3	4
8	3	2	5	4	7	6	1	10	9
4	5	10	9	1	3	7	6	8	2
2	5	6	7	9	3	8	10	4	1
8	2	5	4	1	9	3	6	7	10
5	7	4	3	2	6	8	1	9	10
1	7	4	8	2	3	5	6	9	10
4	1	2	9	8	10	7	5	6	3

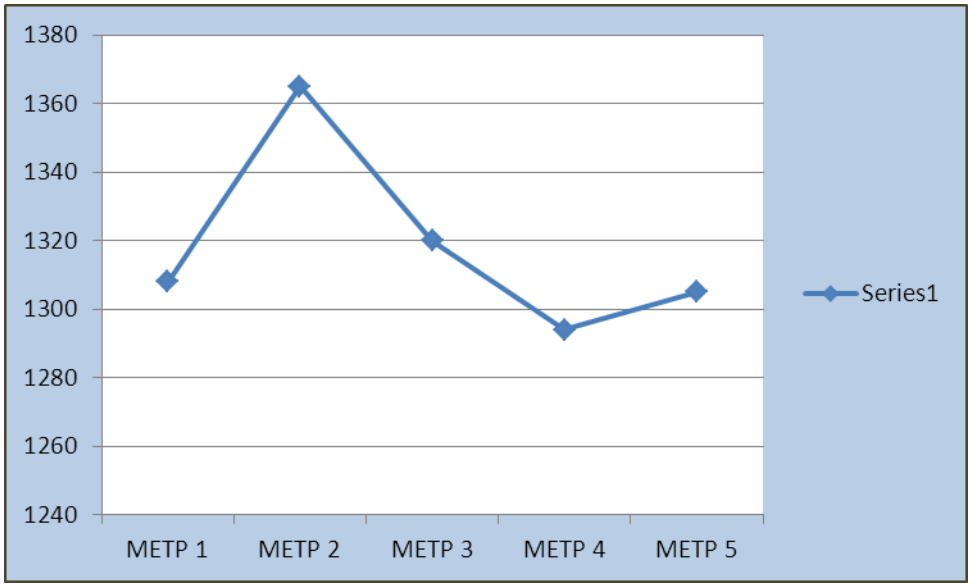
Πίνακας 10: Πινάκας χρονών

88	68	94	99	67	89	77	99	86	92
72	50	69	75	94	66	92	82	94	63
83	61	83	65	64	85	78	85	55	77
94	68	61	99	54	75	66	76	63	67
69	88	82	95	99	67	95	68	67	86
99	81	64	66	80	80	69	62	79	88
50	86	97	96	95	97	66	99	52	71
98	73	82	51	71	94	85	62	95	79
94	71	81	85	66	90	76	58	93	97
50	59	82	67	56	96	58	81	59	96

Τα αποτελέσματα της επαναληπτικής διαδικασίας του αλγορίθμου παρουσιάζονται στο πινάκα και στο γράφημα. Σε σχέση με τη καλύτερη γνωστή λύση έχουμε μια απόκλιση της τάξης του 4,8622%

Πίνακας 11: Πινάκας αποτελεσμάτων επαναληπτικής διαδικασίας

ΜΕΤΡ 1	ΜΕΤΡ 2	ΜΕΤΡ 3	ΜΕΤΡ 4	ΜΕΤΡ 5		ΜΕΣΟ ΟΡΟ
1308	1365	1320	1294	1305		1318.4



Εικόνα 3: Διάγραμμα αποτελεσμάτων επαναληπτικής διαδικασίας

Η καλύτερη τιμή είναι στη τέταρτη μέτρηση με makespan= 1294.

Ο πινάκας final μας εμφανίζει τη σειρά που θα εξακολουθήσουμε ώστε να περάσουν όλες οι εργασίες από όλες τις μηχανές.

Πίνακας 12: Πινάκας final

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

ΠΑΡΑΔΕΙΓΜΑ: ABZ 6

Το ακόλουθο παράδειγμα αφορά πρόβλημα 10 μηχανών και 10 εργασιών.

Πίνακας 13: Πινάκας προτεραιοτήτων

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

Πίνακας 14: Πινάκας χρόνων

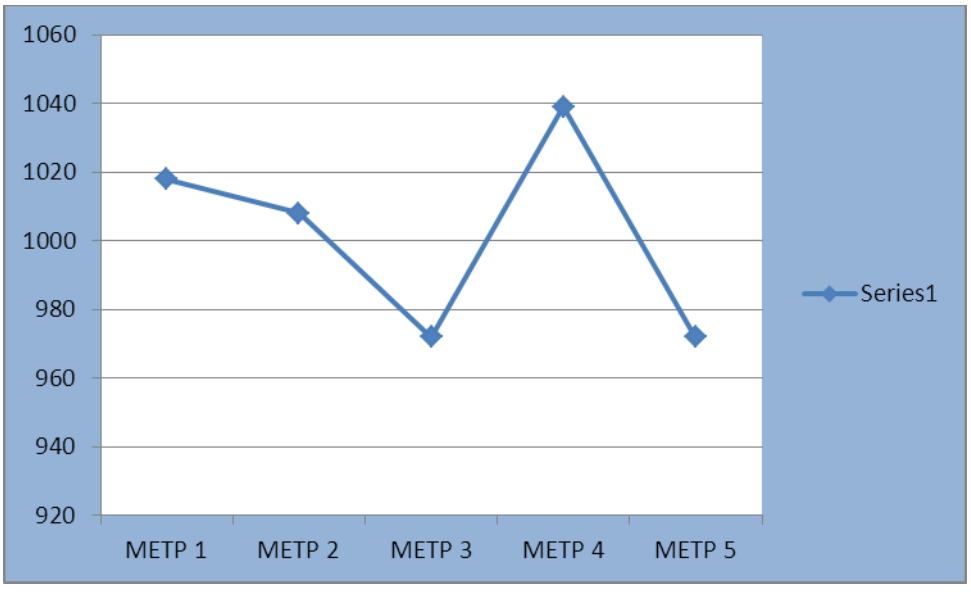
62	24	25	84	47	38	82	93	24	66
47	97	92	22	93	29	56	80	78	67
45	46	22	26	38	69	40	33	75	96
85	76	68	88	36	75	56	35	77	85
60	20	25	63	81	52	30	98	54	86
87	73	51	95	65	86	22	58	80	65

81	53	57	71	81	43	26	54	58	69
20	86	21	79	62	34	27	81	30	46
68	66	98	86	66	56	82	95	47	78
30	50	34	58	77	34	84	40	46	44

Τα αποτελέσματα της επαναληπτικής διαδικασίας του αλγορίθμου παρουσιάζονται στο πινάκα αλλά στο γράφημα . Η απόκλιση στο συγκεκριμένο παράδειγμα είναι 3,0752%.

Πίνακας 15: Πινάκας αποτελεσμάτων επαναληπτικής διαδικασίας

ΜΕΤΡ 1	ΜΕΤΡ 2	ΜΕΤΡ 3	ΜΕΤΡ 4	ΜΕΤΡ 5		ΜΕΣΟ ΟΡΟ
1018	1008	972	1039	972		1001.8



Εικόνα 4: Διάγραμμα αποτελεσμάτων επαναληπτικής διαδικασίας

Η καλύτερη τιμή είναι στη τρίτη , πέμπτη μέτρηση με makespan= 972

Ο πινάκας final μας εμφανίζει τη σειρά που θα εξακολουθήσουμε ώστε να περάσουμε από όλες τις εργασίες και όλες τις μηχανές με το μικρότερο δυνατό κόστος.

Πίνακας 16: Πινάκας final

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

ΠΑΡΑΔΕΙΓΜΑ: ABZ 7

Στο συγκεκριμένο πρόβλημα έχουμε 15 μηχανές και 20 εργασίες .

Τα δεδομένα του προβλήματος παρουσιάζονται παρακάτω .

Πίνακας 17: Πινάκας προτεραιοτήτων

2	3	9	4	0	6	8	7	1	5	11	14	13	10	12
6	3	12	11	1	13	10	2	5	7	0	8	14	9	4
6	0	13	7	2	3	12	10	9	1	5	11	8	4	14
9	6	1	7	12	4	0	5	11	10	14	2	3	8	13
11	13	1	6	7	5	8	9	3	10	2	0	14	12	4
8	11	14	1	7	6	5	10	3	2	4	9	13	12	0
13	3	6	8	12	4	2	9	14	5	10	11	1	0	7
5	4	6	9	3	10	8	14	13	2	1	12	11	7	0
13	11	8	3	5	4	9	0	14	6	2	10	7	12	1
12	1	5	14	7	0	3	13	8	11	4	10	2	9	6
10	5	12	1	7	8	14	4	3	9	13	11	6	2	0
0	11	5	13	4	8	9	14	2	7	10	1	3	12	6
12	5	2	8	3	13	0	6	7	11	14	1	4	9	10
8	1	14	12	4	5	6	10	0	7	11	2	13	9	3
9	14	3	12	0	11	2	5	1	8	4	13	10	6	7
9	14	7	4	0	5	8	13	10	3	2	1	11	6	12
3	5	2	12	7	1	8	6	11	4	10	14	9	13	0
12	10	0	2	5	1	11	8	14	3	7	9	13	4	6
5	10	7	14	13	11	9	3	1	2	12	6	8	4	0
9	8	12	1	5	3	11	13	0	7	14	4	6	2	10

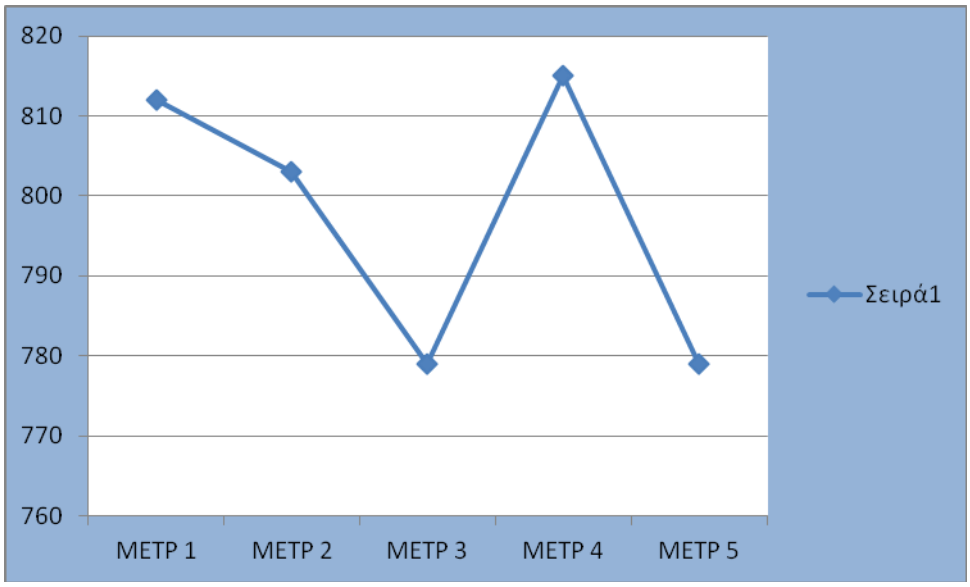
Πίνακας 18: Πινάκας χρονών

24	12	17	27	21	25	27	26	30	31	18	16	39	19	26
30	15	20	19	24	15	28	36	26	15	11	23	20	26	28
35	22	23	32	20	12	19	23	17	14	16	29	16	22	22
20	29	19	14	33	30	32	21	29	24	25	29	13	20	18
23	20	28	32	16	18	24	23	24	34	24	24	28	15	18
24	19	21	33	34	35	40	36	23	26	15	28	38	13	25
27	30	21	19	12	27	39	13	12	36	21	17	29	17	33
27	19	29	20	21	40	14	39	39	27	36	12	37	22	13
32	29	24	27	40	21	26	27	27	16	21	13	28	28	32
35	11	39	18	23	34	24	11	30	31	15	15	28	26	33
28	37	29	31	25	13	14	20	27	25	31	14	25	39	36
22	25	28	35	31	21	20	19	29	32	18	18	11	17	15
39	32	36	14	28	37	38	20	19	12	22	36	15	32	16
28	29	40	23	34	33	27	17	20	28	21	21	20	33	27
21	34	30	38	11	16	14	14	34	33	23	40	12	23	27
13	40	36	17	13	33	25	24	23	36	29	18	13	33	13
25	15	28	40	39	31	35	31	36	12	33	19	16	27	21
22	14	12	20	12	18	17	39	31	31	32	20	29	13	26
18	30	38	22	15	20	16	17	12	13	40	17	30	38	13
31	39	27	14	33	31	22	36	16	11	14	29	28	22	17

Τα αποτελέσματα της επαναληπτικής διαδικασίας του αλγορίθμου παρουσιάζονται στο πινάκα αλλά στο γράφημα. Η καλύτερη δυνατή λύση ισούται με 656 συνεπώς έχω απόκλιση 18,75%

Πίνακας 19: Πινάκας αποτελεσμάτων επαναληπτικής διαδικασίας

ΜΕΤΡ 1	ΜΕΤΡ 2	ΜΕΤΡ 3	ΜΕΤΡ 4	ΜΕΤΡ 5		ΜΕΣΟ ΟΡΟ
812	803	779	815	779		797.6



Εικόνα 5: Διάγραμμα αποτελεσμάτων επαναληπτικής διαδικασίας

Η καλύτερη τιμή είναι στη τρίτη , πέμπτη μέτρηση με makespan= 779

Ο πινάκας final μας εμφανίζει τη σειρά που θα εξακολουθήσουμε ώστε να περάσουμε από όλες τις εργασίες και όλες τις μηχανές με το μικρότερο δυνατό κόστος.

Πίνακας 20: Πινακας final

81	77	52	181	257	175	276	103	286	256	243	31	275	167	176	224	55	239	159	143
98	62	6	174	283	212	70	63	269	38	152	240	129	14	272	288	95	188	225	140
209	235	222	20	23	296	149	113	233	92	248	179	41	263	101	193	42	5	199	185
100	205	78	177	139	58	166	162	90	267	33	285	207	237	253	117	268	32	291	22
45	27	21	281	171	150	244	34	108	141	190	148	87	61	43	2	116	50	282	84
146	270	24	118	132	127	71	186	97	16	191	215	280	115	211	135	35	13	230	60
204	125	213	9	172	104	56	151	254	51	294	216	19	124	94	197	128	198	221	160
180	37	155	136	208	200	183	277	54	266	137	289	147	187	112	278	99	161	72	218
67	73	123	203	102	91	284	271	299	184	109	134	192	11	57	228	249	287	59	15
144	89	12	194	255	76	93	26	157	231	195	189	261	142	138	114	83	40	69	107
229	4	48	297	82	96	206	8	85	68	260	236	290	201	106	66	158	110	241	245
300	120	169	46	1	196	227	238	265	223	247	74	39	17	259	3	298	173	163	80
293	279	79	111	295	258	30	217	168	47	165	145	88	242	210	250	156	251	220	86
214	121	36	178	264	246	28	130	273	202	64	262	219	170	126	164	274	44	131	29
234	65	226	292	119	18	154	7	75	122	25	182	232	49	133	252	10	105	53	153

ΠΑΡΑΔΕΙΓΜΑ: ABZ 8

Στο πρόβλημα που ακολουθεί έχουμε 15 μηχανές και 20 εργασίες .

Τα δεδομένα του προβλήματος παρουσιάζονται παρακάτω:

Πίνακας 21: Πινάκας προτεραιοτήτων

0	9	2	13	10	8	6	4	11	12	1	14	3	5	7
9	10	14	5	4	6	0	7	13	11	1	3	2	12	8
3	8	11	13	10	14	5	9	0	1	4	2	7	6	12
14	3	2	12	4	13	1	6	9	7	10	0	5	8	11
12	14	1	7	8	11	6	4	0	2	3	10	5	13	9
10	12	8	14	9	6	5	4	2	11	7	3	13	0	1
6	8	0	12	10	9	2	4	3	13	14	11	5	7	1
11	13	10	8	9	14	5	0	2	6	12	1	3	4	7
8	6	5	3	9	0	2	4	12	10	11	7	14	13	1
1	7	14	9	5	6	8	3	10	0	4	12	11	13	2
11	0	13	6	5	3	14	10	7	8	2	1	12	4	9
6	3	12	5	4	1	0	13	8	2	10	11	14	9	7
14	13	1	8	2	3	9	4	0	6	12	5	10	7	11
8	10	11	9	5	12	7	4	2	13	14	3	6	1	0
8	0	9	1	10	7	13	3	5	11	2	14	6	12	4
8	12	3	13	4	1	6	0	10	7	9	2	5	11	14
14	3	0	10	5	7	8	6	12	9	4	2	11	1	13
11	3	14	6	8	12	13	5	1	10	7	4	9	2	0
11	2	12	9	14	7	4	3	5	6	10	1	0	13	8
2	12	5	8	6	9	14	11	3	13	7	1	4	0	10

Πίνακας 22: Πινάκας χρονών

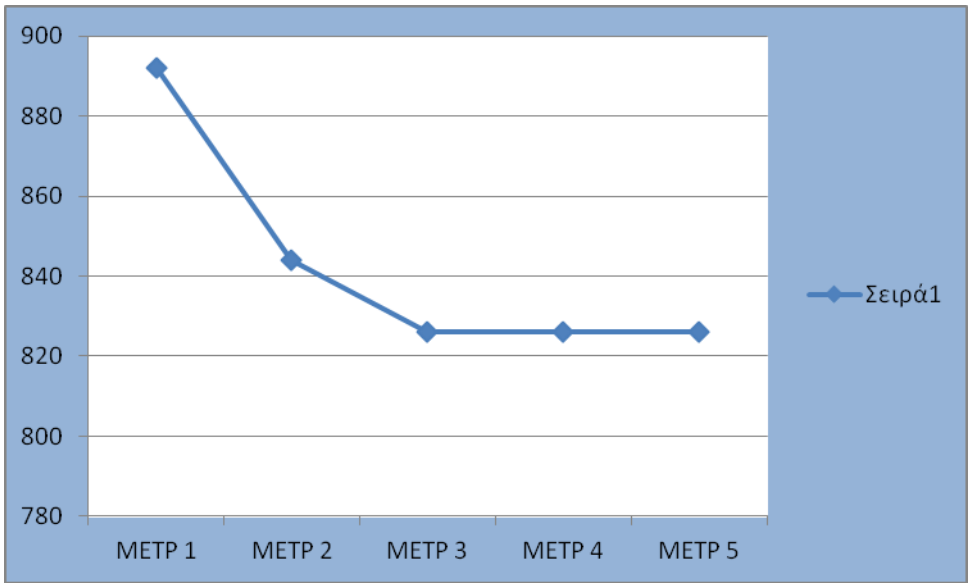
19	33	32	18	39	34	25	36	40	33	31	30	34	26	13
11	22	19	12	25	38	29	39	19	22	23	20	40	19	26
25	17	24	40	32	16	39	19	24	39	17	35	38	20	31
22	36	34	17	30	12	13	25	12	18	31	39	40	26	37
32	15	35	13	32	23	22	21	38	38	40	31	11	37	16
23	38	11	27	11	25	14	12	27	26	29	28	21	20	30
39	38	15	27	22	27	32	40	12	20	21	22	17	38	27
11	24	38	15	19	13	30	26	29	33	21	15	21	28	33
20	17	26	34	23	16	18	35	24	16	26	12	13	27	19
18	37	27	40	40	17	22	17	30	38	21	32	24	24	30
19	22	36	18	22	17	35	34	23	19	29	22	17	33	39
32	22	24	13	13	11	11	25	13	15	33	17	16	38	24
16	16	37	25	26	11	34	14	20	36	12	29	25	32	12
20	24	27	38	34	39	33	37	31	15	34	33	26	36	14
31	17	13	21	17	19	14	40	32	25	34	23	13	40	26
38	17	14	17	12	35	35	19	36	19	29	31	26	35	37
20	16	33	14	27	31	16	31	28	37	37	29	38	30	36
18	37	16	15	14	11	32	12	11	29	19	12	18	26	39
11	11	22	35	20	31	19	39	28	33	34	38	20	17	28
12	25	23	21	27	30	23	39	26	34	17	24	12	19	36

Η καλύτερη τιμή είναι στη τρίτη , τέταρτη μέτρηση με makespan= 826

Η απόκλιση στο συγκεκριμένο παράδειγμα είναι 27,8637%

Πίνακας 23: Πινάκας αποτελεσμάτων επαναληπτικής διαδικασίας

ΜΕΤΡ 1	ΜΕΤΡ 2	ΜΕΤΡ 3	ΜΕΤΡ 4	ΜΕΤΡ 5		ΜΕΣΟ ΟΡΟ
892	844	826	826	826		842.8



Εικόνα 6: Διάγραμμα αποτελεσμάτων επαναληπτικής διαδικασίας

Και ακολουθει ο πινακας final της matlab με το βελτιστο συνδιασμο.

Πίνακας 24: Πινάκας final

53	199	92	296	234	235	63	225	181	21	256	135	274	195	133	142	275	265	250
202	239	144	110	300	276	106	259	126	136	17	69	255	205	1	247	127	71	76
90	281	179	103	7	228	5	220	124	288	273	236	16	230	146	55	45	291	260
282	150	226	266	117	292	232	167	79	175	155	96	187	201	3	248	224	138	209
219	123	173	140	152	229	178	182	190	75	122	72	56	100	160	58	168	156	298
183	153	252	293	200	141	131	287	34	99	66	23	60	237	120	20	196	286	171
174	33	121	24	105	132	28	81	49	139	297	2	212	180	253	101	203	290	134
145	192	162	184	88	114	46	11	91	50	207	264	67	84	26	15	188	112	95
222	261	14	210	4	65	32	40	258	193	164	157	9	42	93	215	257	22	143
278	242	35	194	43	251	271	85	223	18	177	39	118	285	206	130	213	185	166
54	48	102	233	77	158	283	254	244	68	47	189	245	295	8	214	151	148	243
294	116	74	12	163	107	198	30	149	218	31	208	97	62	280	44	119	109	52
108	89	137	161	98	277	57	59	299	217	262	268	231	111	104	87	241	172	6
169	246	129	204	279	19	82	78	284	270	115	170	165	38	211	221	73	51	267
176	197	216	263	159	27	13	128	36	37	186	154	238	25	80	289	272	10	191

ΠΑΡΑΔΕΙΓΜΑ: ABZ 9

Τελευταίο παράδειγμα στην ομάδα ABZ είναι το ABZ 9 το όποιο αποτελείτε από 15 μηχανές και 20 εργασίες .

Πίνακας 25: Πινάκας προτεραιοτήτων

6	5	8	4	1	14	13	11	10	12	2	3	0	7	9
1	5	0	3	6	9	7	12	10	13	8	4	11	14	2
0	4	2	10	6	14	8	13	7	3	9	12	1	11	5
7	5	4	8	0	9	13	12	10	3	6	14	1	11	2
2	3	12	11	6	4	10	7	0	13	1	14	5	9	8
5	3	6	12	10	0	13	2	11	7	4	1	14	9	8
13	0	11	12	4	6	5	3	9	2	7	10	1	14	8
2	12	9	11	13	8	14	5	6	3	1	4	0	7	10
2	10	14	6	8	3	12	0	13	9	7	1	11	4	5
4	9	3	11	13	7	0	2	5	12	1	10	14	8	6
13	0	3	8	5	6	14	7	1	2	4	9	12	11	10
14	10	0	3	13	6	7	2	12	5	4	11	1	8	9
6	12	4	2	8	5	14	3	9	1	11	13	7	10	0
5	14	0	8	7	4	9	13	1	12	6	11	3	10	2
5	3	10	6	4	12	11	13	7	9	14	1	2	0	8
8	5	9	6	1	7	11	2	4	0	10	3	12	14	13
1	4	8	3	10	5	12	7	9	14	11	13	0	2	6
7	5	13	9	10	4	14	0	3	11	6	8	1	2	12
14	11	5	2	13	10	4	8	3	9	6	7	0	1	12
1	7	11	8	14	6	5	3	13	2	0	4	9	12	10

Πίνακας 26: Πινάκας χρόνων

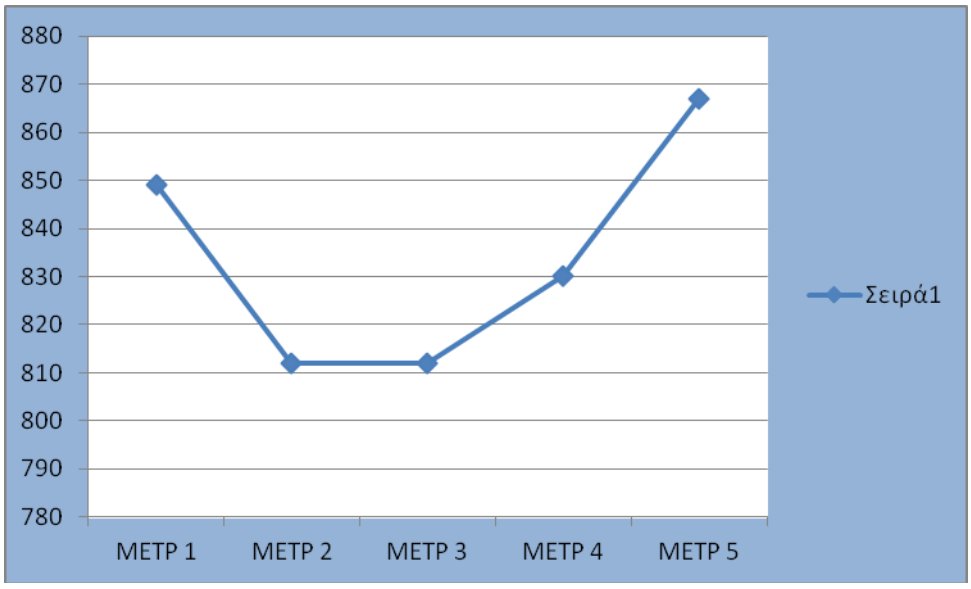
14	21	13	11	11	35	20	17	18	11	23	13	15	11	35
35	31	13	26	14	17	38	20	19	12	16	34	15	12	14
30	35	40	35	30	23	29	37	38	40	26	11	40	36	17
40	18	12	23	23	14	16	14	23	12	16	32	40	25	29
35	15	31	28	32	30	27	29	38	11	23	17	27	37	29
33	33	19	40	19	33	26	31	28	36	38	21	25	40	35
25	32	33	18	32	28	15	35	14	34	23	32	17	26	19
16	33	34	30	40	12	26	26	15	21	40	32	14	30	35
17	16	20	24	26	36	22	14	11	20	23	29	23	15	40
27	37	40	14	25	30	34	11	15	32	36	12	28	31	23
25	22	27	14	25	20	18	14	19	17	27	22	22	27	21
34	15	22	29	34	40	17	32	20	39	31	16	37	33	13
12	27	17	24	11	19	11	17	25	11	31	33	31	12	22
22	15	16	32	20	22	11	19	30	33	29	18	34	32	18
27	26	28	37	18	12	11	26	27	40	19	24	18	12	34
15	28	25	32	13	38	11	34	25	20	32	23	14	16	20
15	13	37	14	22	24	26	22	34	22	19	32	29	13	35
36	33	28	20	30	33	29	34	22	12	30	12	35	13	35
26	31	35	38	19	35	27	29	39	13	14	26	17	22	15
36	34	33	17	38	39	16	27	29	16	16	19	40	35	39

Η καλύτερη τιμή είναι στη δεύτερη , τρίτη μέτρηση με makespan= 812

Η απόκλιση στο συγκεκριμένο παράδειγμα είναι 22,6586%

Πίνακας 27: Πινάκας αποτελεσμάτων επαναληπτικής διαδικασίας

ΜΕΤΡ 1	ΜΕΤΡ 2	ΜΕΤΡ 3	ΜΕΤΡ 4	ΜΕΤΡ 5		ΜΕΣΟ ΟΡΟ
849	812	812	830	867		834



Εικόνα 7: Διάγραμμα αποτελεσμάτων επαναληπτικής διαδικασίας

Ο πινάκας final μας εμφανίζει τη σειρά που θα εξακολουθήσουμε ώστε να περάσουμε από όλες τις εργασίες και όλες τις μηχανές .

Πίνακας 28: Πινάκας final

156	53	14	150	47	102	281	58	60	116	292	293	121	92	83	171	19	164	268
148	99	95	128	59	82	24	216	173	177	23	254	260	299	67	119	33	106	241
178	2	79	112	238	120	192	155	290	86	94	26	45	3	210	182	103	245	118
132	66	232	166	225	284	230	190	114	202	237	137	11	170	105	159	31	186	55
6	130	187	15	88	219	78	246	144	233	167	286	135	133	71	75	218	188	249
280	197	288	174	68	36	96	146	73	158	181	200	97	85	270	163	37	93	229
247	151	205	239	189	287	81	258	201	38	300	207	296	176	242	220	267	276	100
228	252	282	209	123	57	72	91	22	62	198	126	27	273	291	107	89	12	169
224	214	203	179	13	240	172	122	222	129	48	272	208	18	110	113	140	77	70
236	277	80	111	259	160	46	63	161	223	90	196	10	265	147	25	298	199	194
17	184	231	139	42	87	134	44	143	52	145	153	255	98	294	217	251	136	141
65	69	266	244	262	41	180	64	32	185	256	264	285	235	21	108	289	183	154
1	227	39	76	74	295	28	20	191	278	275	84	125	49	35	243	4	50	9
162	279	261	206	152	212	283	127	101	175	271	213	226	195	7	297	29	61	30
193	54	157	109	142	204	149	168	104	138	269	250	5	40	124	253	56	221	248

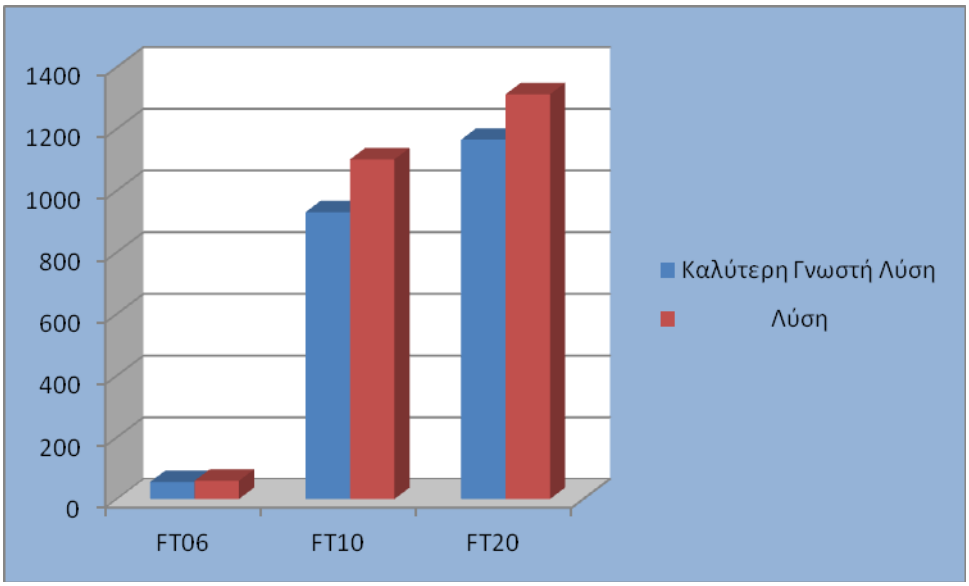
4.2 Παρουσίαση όλων των σειρών δεδομένων

Η επόμενη σειρά δεδομένων είναι ft06, ft10 και ft20 από H. Fisher, G.L. Thompson (1963). Στα 3 παραδείγματα παρουσιάζεται η καλύτερη γνωστή λύση , η καλύτερη δίκια μου λύση από τον αλγόριθμο όπως η απόκλιση ανάμεσα σε αυτές τις δυο.

Πίνακας 29: Πινάκας σύγκρισης λύσεων

Παραδειγματα	Μέγεθος	Καλύτερη Γνωστή Λύση	Λύση
FT06	6×6	55	59
FT10	10×10	930	1101
FT20	20×5	1165	1312

Γραφική απεικόνιση και σύγκριση του παραπάνω πινάκα:

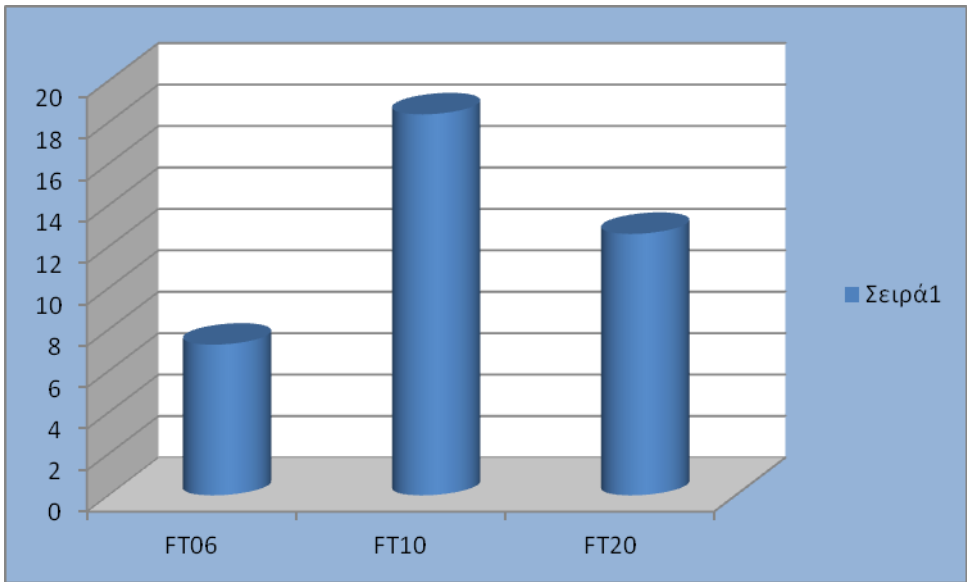


Εικόνα 8: Διάγραμμα σύγκρισης λύσεων

Πίνακας 30: Πινάκας αποκλίσεων

FT06	7,272727273
FT10	18,38709677
FT20	12,61802575

Γραφική απεικόνιση της απόκλισης για FT06, FT10, FT20:



Εικόνα 9: Διάγραμμα αποκλίσεων

Η επαναληπτική διαδικασία έδωσε καλύτερο αποτέλεσμα για το FT06 , αφού εκεί έχουμε μικρότερη απόκλιση.

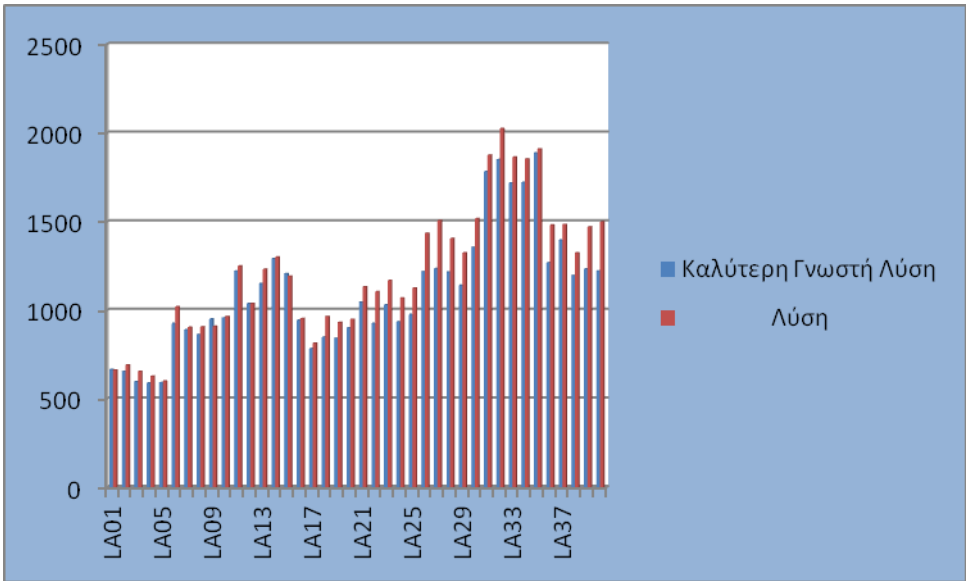
Η σειρά δεδομένων la02-la40 από S. Lawrence (1984). Στο παρακάτω πινάκα παρουσιάζονται τα αποτελέσματα των λύσεων , ενώ στο διαγράμματα η σύγκριση των καλυτέρων τιμών με αυτές που βρήκαμε από την αλγοριθμική ακολουθία.

Πίνακας 31: Πινάκας σύγκρισης λύσεων

Παράδειγμα	Μέγεθος	Καλύτερη Γνωστή Λύση	Λύση
LA02	10×5	655	692
LA03	10×5	597	656
LA04	10×5	590	628
LA05	10×5	593	601
LA06	15×5	926	1020
LA07	15×5	890	904
LA08	15×5	863	907
LA10	15×5	958	965
LA11	20×5	1222	1250
LA12	20×5	1039	1039
LA13	20×5	1150	1232
LA14	20×5	1292	1300
LA16	10×10	945	954
LA17	10×10	784	815
LA18	10×10	848	965
LA19	10×10	842	932
LA20	10×10	902	949
LA21	15×10	1046	1133
LA22	15×10	927	1105
LA23	15×10	1032	1168
LA24	15×10	935	1070
LA25	15×10	977	1126
LA26	20×10	1218	1434
LA27	20×10	1235	1507
LA28	20×10	1216	1406
LA29	20×10	1142	1324
LA30	20×10	1355	1517

LA31	30×10	1784	1876
LA32	30×10	1850	2026
LA33	30×10	1719	1864
LA34	30×10	1721	1854
LA35	30×10	1888	1910
LA36	15×15	1268	1482
LA37	15×15	1397	1484
LA38	15×15	1196	1324
LA39	15×15	1233	1470
LA40	15×15	1222	1499

Γραφική απεικόνιση και σύγκριση του παραπάνω πινάκα:



Εικόνα 10: Διάγραμμα σύγκρισης λύσεων

Πίνακας 32: Πινάκας αποκλίσεων

Παράδειγμα	Απόκλιση
LA02	5,648854962
LA03	9,882747069
LA04	6,440677966
LA05	1,349072
LA06	10,1511879
LA07	1,573033708
LA08	5,098493627
LA10	0,730688935
LA11	2,29132576
LA12	0
LA13	7,130434783
LA14	0,619295
LA16	0,95238095
LA17	3,954081633
LA18	13,79716981
LA19	10,6888361
LA20	5,210643016
LA21	8,317399618
LA22	19,201726
LA23	13,17829457
LA24	14,43850267
LA25	15,25076766
LA26	17,73399015

LA27	22,0242915
LA28	15,625
LA29	15,93695271
LA30	11,95571956
LA31	5,156950673
LA32	9,513513514
LA33	8,435136707
LA34	7,728065078
LA35	1,165254237
LA36	16,87697161
LA37	6,227630637
LA38	10,70234114
LA39	19,22141119
LA40	22,66775777

Γραφική απεικόνιση της απόκλισης από LA1-LA40:

Εικόνα 11: Διάγραμμα αποκλίσεων

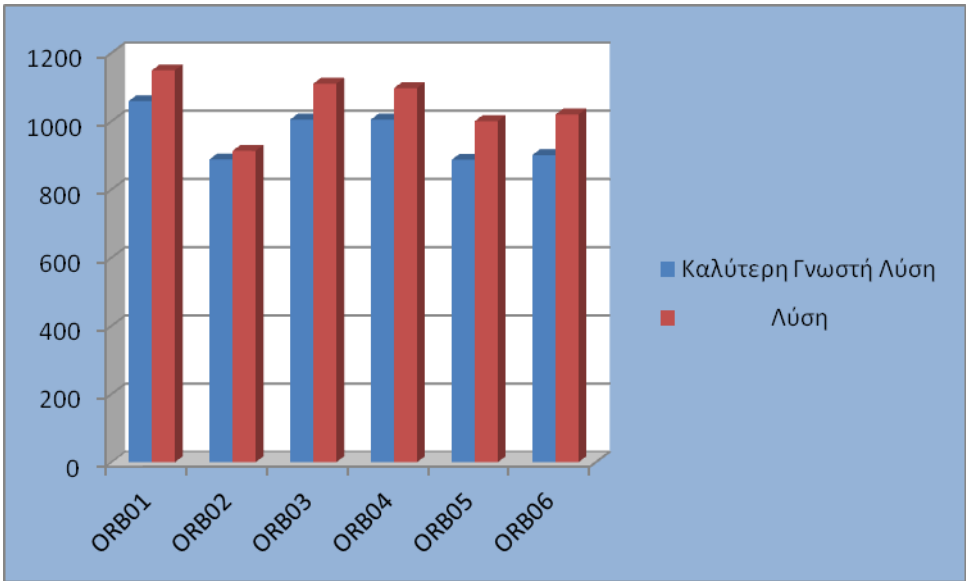
Υπάρχουν αρκετά παραδείγματα στα όποια παρατηρούμε ότι η απόκλιση τείνει στο 0. Αυτό σημαίνει ότι η επαναληπτική διαδικασία μας έδωσε καλά αποτελέσματα . Οι λύσεις που βρήκαμε είναι αρκετά κοντά στις καλύτερες γνώστες λύσεις.

Τα orb01-orb5 από D. Applegate, W. Cook (1991), είναι η επόμενη σειρά δεδομένων.

Πίνακας 33: Πινάκας σύγκρισης λύσεων

Παράδειγμα	Μέγεθος	Καλύτερη Γνωστή Λύση	Λύση
ORB01	10×10	1059	1149
ORB02	10×10	888	913
ORB03	10×10	1005	1110
ORB04	10×10	1005	1097
ORB05	10×10	887	1000

Γραφική απεικόνιση και σύγκριση του παραπάνω πινάκα:

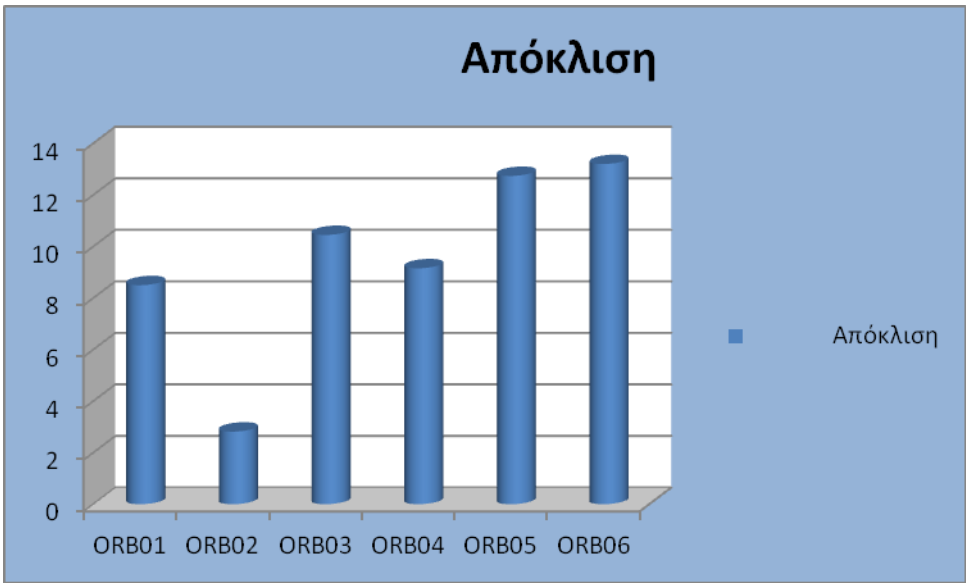


Εικόνα 12: Διάγραμμα σύγκρισης λύσεων

Πίνακας 34: Πινάκας αποκλίσεων

Παράδειγμα	Απόκλιση
ORB01	8,498583569
ORB02	2,815315315
ORB03	10,44776119
ORB04	9,154228856
ORB05	12,73957159
ORB06	13,20084379

Γραφική απεικόνιση της απόκλισης από ORB01-ORB06:



Εικόνα 13: Διάγραμμα αποκλίσεων

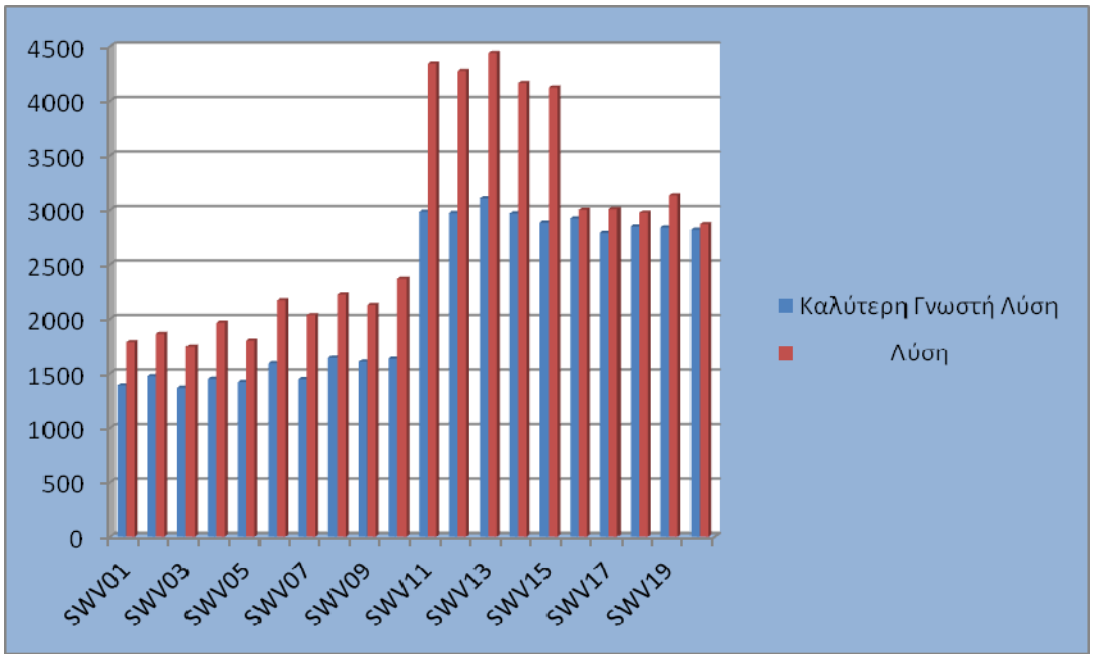
Οι τιμές της απόκλισης κυμανθήκαν περίπου στα ιδία επίπεδα εκτός από το ORB 01 όπου υπάρχει η μικρότερη απόκλιση συνεπές και το καλύτερο αποτέλεσμα.

Η επόμενη σειρά δεδομένων είναι των Storer, Wu, και Vaccari.

Πίνακας 35: Πινάκας σύγκρισης λύσεων

Παράδειγμα	Μέγεθος	Καλύτερη Γνωστή Λύση	Λύση
SWV01	20×10	1392	1790
SWV02	20×10	1475	1865
SWV03	20×10	1370	1750
SWV04	20×10	1450	1965
SWV05	20×10	1421	1805
SWV06	20×15	1591	2166
SWV07	20×15	1447	2033
SWV08	20×15	1641	2230
SWV09	20×15	1605	2124
SWV10	20×15	1632	2371
SWV11	50×10	2983	4346
SWV12	50×10	2972	4279
SWV13	50×10	3104	4441
SWV14	50×10	2968	4160
SWV15	50×10	2885	4120
SWV16	50×10	2924	3001
SWV17	50×10	2794	3006
SWV18	50×10	2852	2974
SWV19	50×10	2843	3129
SWV20	50×10	2823	2871

Γραφική απεικόνιση και σύγκριση του παραπάνω πινάκα:



Εικόνα 14: Διάγραμμα σύγκρισης λύσεων

Πίνακας 36: Πινάκας αποκλίσεων

Παράδειγμα	Απόκλιση
SWV01	28,59195402
SWV02	26,44067797
SWV03	27,73722628
SWV04	35,51724138
SWV05	27,02322308
SWV06	36,14079195
SWV07	40,4975812
SWV08	35,89274832
SWV09	32,3364486

SWV10	45,28186275
SWV11	45,69225612
SWV12	43,97711978
SWV13	43,07345361
SWV14	40,16172507
SWV15	42,80762565
SWV16	2,633367
SWV17	7,587687903
SWV18	4,27769986
SWV19	10,05979599
SWV20	1,70031881

Γραφική απεικόνιση της απόκλισης από SWV01-SWV20:

Εικόνα 15: Διάγραμμα αποκλίσεων

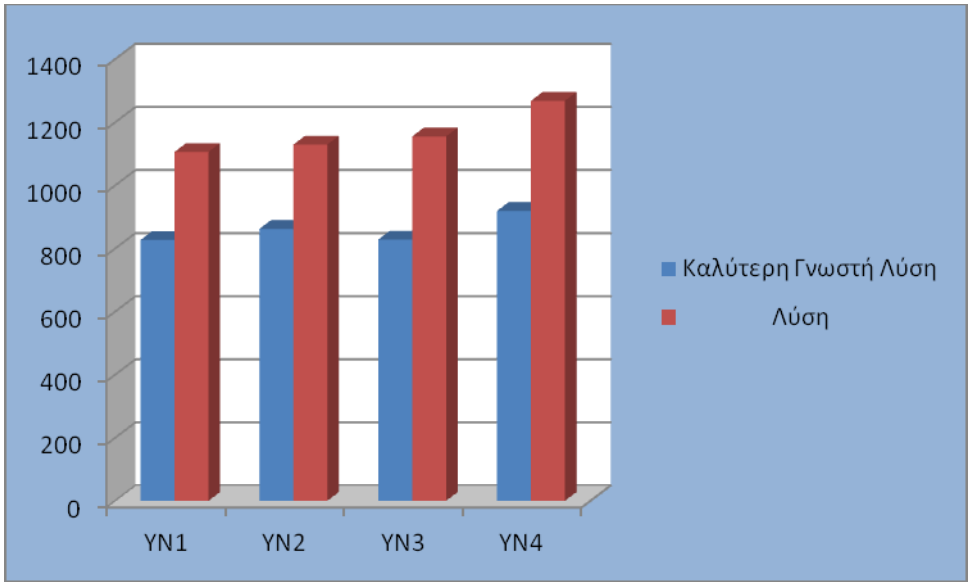
Παρατηρούμε ότι τα τελευταία 5 παραδείγματα αυτής της σειράς μας έδωσαν τη χαμηλότερη απόκλιση.

Η τελευταία σειρά αποτελείται από 4 παραδείγματα, yn1-yn4 των Yamada και Nakano:

Πίνακας 37: Πινάκας σύγκρισης λύσεων

Παράδειγμα	Μέγεθος	Καλύτερη Γνωστή Λύση	Λύση
YN1	20×20	827	1106
YN2	20×20	862	1129
YN3	20×20	828	1155
YN4	20×20	919	1268

Γραφική απεικόνιση και σύγκριση του παραπάνω πινάκα:

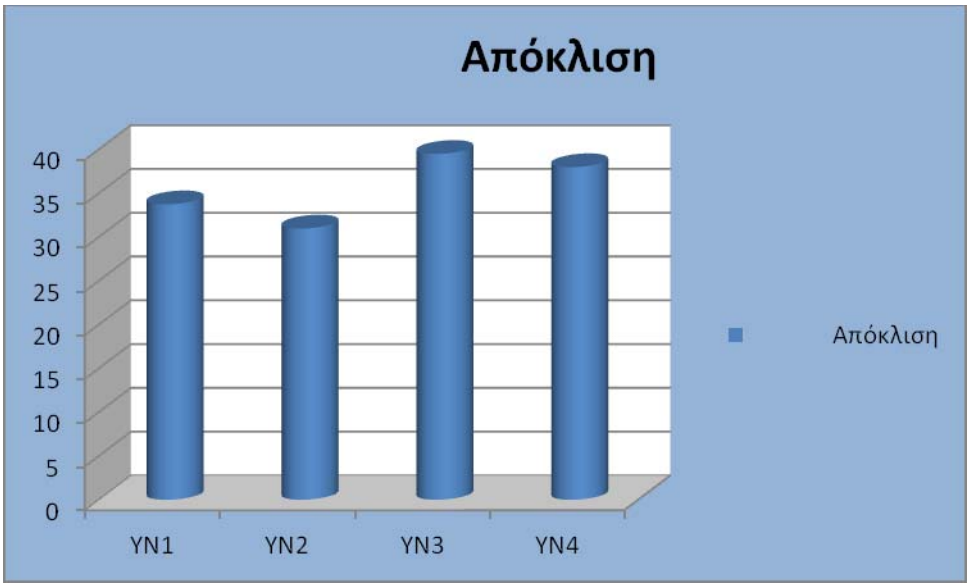


Εικόνα 16: Διάγραμμα σύγκρισης λύσεων

Πίνακας 38: Πινάκας αποκλίσεων

Παράδειγμα	Απόκλιση
YN1	33,73639661
YN2	30,97447796
YN3	39,49275362
YN4	37,97606094

Γραφική απεικόνιση της απόκλισης από YN1-YN4:



Εικόνα 17: Διάγραμμα αποκλίσεων

Διαπιστώνουμε ότι YN2 έχει τη μικρότερη απόκλιση άρα τη καλύτερη τιμή σε σχέση με τα υπόλοιπα παραδείγματα αυτής της σειράς.

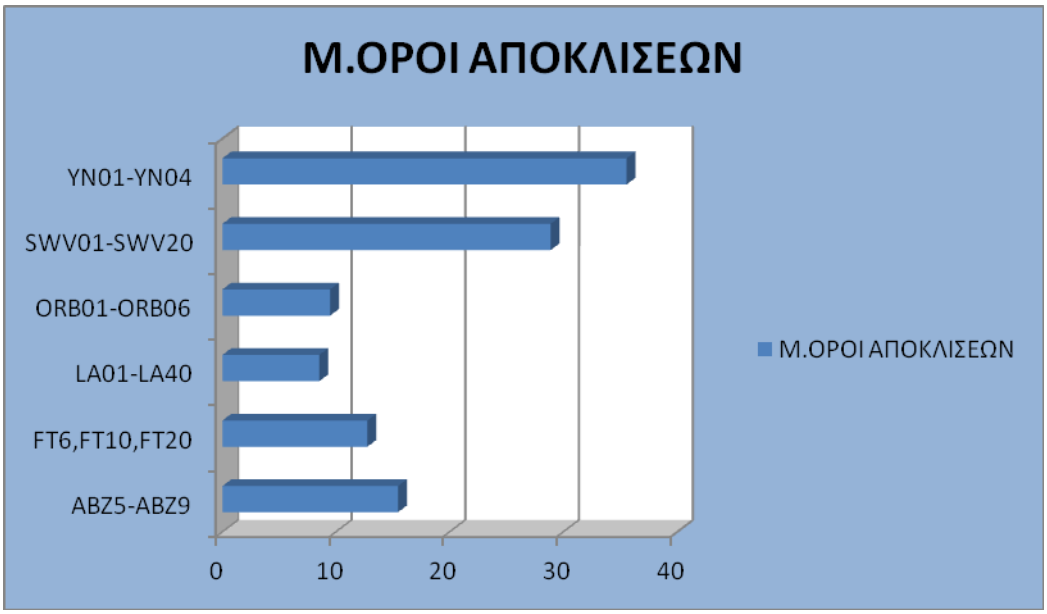
4.3 Συγκριτική ανάλυση αποτελεσμάτων

Εύκολα μπορούμε να παρατηρήσουμε ότι ένας κάλος τρόπος σύγκρισης των παραπάνω σειρών δεδομένων είναι με τη βοήθεια των αποκλίσεων που παρουσιάζουν. Στο παρακάτω πίνακα υπάρχουν οι μέσοι όροι των αποκλίσεων για όλες τις σειρές.

Πίνακας 39: Πινάκας μέσων ορών αποκλίσεων

	ABZ5-ABZ9	FT6,FT10,FT20	LA02-LA40	ORB01-ORB06	SWV01-SWV20	YN01-YN04
Μ. ΟΡΟΙ ΑΠΟΚΛΙΣΕΩΝ	15,44198312	12,75928327	7,947945328	9,476050719	28,66634251	35,54492228

Γραφική απεικόνιση μέσου όρου αποκλίσεων



Εικόνα 18: Διάγραμμα μέσων ορών αποκλίσεων

Και με τη βοήθεια της γραφικής απεικόνισης γίνετε αντιληπτό ότι στις σειρές δεδομένων LA02-LA40 , ORB01-ORB06 τα αποτελέσματα τις επαναληπτικής διαδικασίας είχαν τη μικρότερη απόκλιση σε σχέση με τις καλύτερες δυνατές λύσεις . Αντίθετα μεγαλύτερη απόκλιση εντοπίζουμε στις σειρές δεδομένων YN01-YN04 και SWV01-SWV20 ενώ οι σειρές (FT6, FT10,FT20) και ABZ5-ABZ9 κινήθηκαν σε ικανοποιητικά επίπεδα κοντά στο 15% .

ΚΕΦΑΛΑΙΟ 5

ΣΥΜΠΕΡΑΣΜΑΤΑ

Στην διπλωματική εργασία εξετάσαμε το πρόβλημα προγραμματισμού παραγωγής. Πιο συγκεκριμένα αναπτύξαμε τον πρόβλημα κατά παραγγελία (jobshop scheduling problem) συνδυάζοντας τον με το μεθευρετικό αλγόριθμο περιορισμένης αναζήτησης (tabu search). Η υλοποίηση του αλγορίθμου για την επίλυση του προβλήματος έγινε με τη χρήση της μαθηματικής γλώσσας προγραμματισμού Matlab.

Τα συμπεράσματα μας μπορούν να επικεντρωθούν σε τρεις κύριους τομείς :

- Όσον αφορά την αντιμετώπιση του προβλήματος ο αλγόριθμος ήταν αρκετά αποτελεσματικός αφού σε σχέση με τις καλύτερες λύσεις που μου δόθηκαν από τον επιβλέποντα καθηγητή η απόκλιση ήταν μικρή ενώ σε κάποια παραδείγματα σχεδόν μηδενική.

- Επίσης ο αλγόριθμος που χρησιμοποιήσαμε για τη παραπάνω μέθοδο συνδύασε εκτός από καλά αποτελέσματα και ταχύτητα, αφού η επαναληπτική διαδικασία για κάθε παράδειγμα δε διαρκούσε περισσότερο από ένα λεπτό βοηθώντας έτσι μείωση του χρόνου επίλυσης.

- Όπως παρουσιάζεται και παραπάνω η κάθε επαναληπτική διαδικασία εκτός από τις 10000 επαναλήψεις που πραγματοποιεί χρησιμοποιεί και άλλους περιορισμούς ώστε να αποφεύγετε ο εγκλωβισμός σε τοπικό ελάχιστο, συνεπώς η μέθοδος που χρησιμοποιήσαμε ελαχιστοποιεί τη πιθανότητα να μην έχουμε βέλτιστο αποτέλεσμα.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] R.W. Conway, W.L. Maxwell, Theory of Scheduling, Addison-Wesley, MA, 1967.

[2] Επίλυση προβλήματος προγραμματισμού εργασιών με την χρήση αλγορίθμων εμπνευσμένων από την φύση [πολυμέσα] / Βιττώριας Μιχαήλ ; επιβλ. καθηγ. Ιωάννης Μαρινάκης

[3] U.K. Chakraborty (Ed.): Comput. Intel. in Flow Shop and Job Shop Sched., SCI 230, pp. 261–300.

[4] Peter Brucker Scheduling Algorithms

[5] K.R. Baker, Introduction to Sequence and Scheduling, John Wiley, NY, 1974.

[6] Ι. Μαρινάκης και Α. Μυγδαλάς, Σχεδιασμός και βελτιστοποίηση της εφοδιαστικής αλυσίδας. Εκδόσεις "σοφία" α.ε., Θεσσαλονίκη, 2008.

[7] H. Yildiz, M. P. Johnson and S. Roehrig, A Genetic Algorithm for the Home-Delivered Meals Location-Routing Problem

[8] J. Perl and M.S. Daskin, A Warehouse Location-Routing Problem. Transpn. Res.-B Vol. 19B, No. 5, p. 381-396, 1985.

[9] Y. Marinakis and M. Marinaki, A Bilevel Genetic Algorithm for a real life location routing problem. International Journal of Logistics Research and Applications

[10] P.V. Hentenryck, Constant Satisfaction and Logic Programming, MIT Press, MA, 1989

[11] D. Dubois, H. Fargier, H. Prade, Fuzzy constraint in job shop scheduling, J. Intell. Manuf. 6 (1995) 215–234.