ΠΟΛΥΤΕΧΝΕΙΟ ΚΡΗΤΗΣ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΑΡΑΓΩΓΗΣ ΚΑΙ ΔΙΟΙΚΗΣΗΣ

Ένας μεθευρετικός αλγόριθμος με απαγόρευση κινήσεων για την επίλυση του προβλήματος παραγωγής συνεχούς ροής

Διπλωματική εργασία

Κουρουνιώτης Γιώργος

Χανιά, 2013

Επιβλέπων καθηγητής : Μαρινάκης Γιάννης

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα καταρχήν να ευχαριστήσω όλους όσους συνέβαλαν με οποιονδήποτε τρόπο στην εκπόνηση αυτής της διπλωματικής εργασίας και ιδιαίτερα την κ.Αθανασία Χάρη για τις πολύτιμες συμβουλές της γύρω από την εκπόνηση διπλωματικών εργασιών. Επίσης θα πρέπει να ευχαριστήσω θερμά τον κύριο Γιάννη Μαρινάκη για την επίβλεψη αυτής της διπλωματικής εργασίας. Έπειτα ευχαριστώ όλους όσους υπήρξαν γύρω μου όλα αυτά τα οχτώ χρόνια που διέμεινα στα Χανιά.

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΕΡΙΛΗΨΗ

Στην παρούσα διπλωματική εργασία προσεγγίζεται το πρόβλημα του χρονικού προγραμματισμού εργασιών παραγωγής συνεχούς ροής (flow-shop system) με την εφαρμογή του μεθευρετικού αλγορίθμου τοπικής αναζήτησης με απαγόρευση κινήσεων (αλγόριθμος Tabu Search). Τα προβλήματα χρονικού προγραμματισμού παρουσιάζουν ιδιαίτερο ενδιαφέρον καθώς πηγάζουν από τον χώρο της βιομηχανικής παραγωγής ενώ παράλληλα χαρακτηρίζονται ως NP-hard προβλήματα, για τα οποία δεν υπάρχει γνωστός αποδοτικός αλγόριθμος που να δίνει βέλτιστη λύση. Στόχος του χρονικού προγραμματισμού είναι η «τακτοποίηση» συνόλου εργασιών κατά τέτοιο τρόπο ώστε να βελτιστοποιούνται ορισμένα κριτήρια, όπως ο χρόνος. Στην εν λόγω εργασία εξετάζεται ένα σύστημα παραγωγής συνεχούς ροής ενώ μοναδικό κριτήριο βέλτιστου είναι η ελαχιστοποίηση του χρόνου περάτωσης του προγράμματος εργασιών. Για την επίλυση του παραπάνω προβλήματος αναπτύσσεται ο αλγόριθμος τοπικής αναζήτησης με απαγόρευση κινήσεων για εύρεση υποβέλτιστων λύσεων.

ABSTRACT

In this thesis a flow-shop production sequencing problem is being analyzed in an attempt to find an optimal solution by the use of a particular metaheuristic algorithm, titled as Tabu Search Algorithm. Production scheduling problems are of particular interest as derived from the fields of industrial production, while they are classified as NP-hard problems for which there has not been discovered any algorithm that would efficiently give an optimal solution. The objective of a scheduling production is to «regulate» a set of actions in such a way that certain criteria are optimized, such as the criterion of time. In this project a flow-shop scheduling system is examined and a single case is dealt with, the objective of which is to minimize the completion time of a set of actions. For this purpose an algorithm of local search with forbidden (tabu) movements is applied in order to search for suboptimal solutions.

ΕΙΣΑΓΩΓΗ

Στα τέλη του 20^{ου} και στις αρχές του 21^{ου} αιώνα, η ανακάλυψη νέων προηγμένων τεχνολογιών βρίσκει εφαρμογή στον χώρο της βιομηχανίας με την χρήση πολλαπλών, τεχνολογικά τελειοποιημένων, μηχανών. Ταυτοχρόνως η συνεχής αύξηση του πληθυσμού με την αντίστοιχη αύξηση της ζήτησης προϊόντων σε αντιπαράθεση με τον ολοένα αυξανόμενο ανταγωνισμό στον χώρο της βιομηχανίας και των υπηρεσιών καθιστά ως επιτακτική ανάγκη τον χρονικό προγραμματισμό ενός συστήματος παραγωγής (scheduling of production processes).

Ο χρονικός προγραμματισμός συστημάτων παραγωγής στοχεύει στην βελτιστοποίηση της αποδοτικότητας ενός συνόλου διεργασιών παραγωγής με σκοπό την αύξηση των οικονομικών εισροών στο σύστημα, είτε λόγω αύξησης της παραγωγής σε δεδομένο χρόνο, είτε λόγω μείωσης της ανθρώπινης εργασίας. Αντίθετα, οι μεγάλες καθυστερήσεις σε ένα συστήματα παραγωγής αποτελούν τροχοπέδη για τα οικονομικά αποτελέσματα. Σε γενικές γραμμές, οι αντικειμενικοί σκοποί του χρονικού προγραμματισμού της παραγωγής είναι η αποτελεσματική χρησιμοποίηση μηχανών και προσωπικού και η ελαχιστοποίηση του χρόνου αναμονής πελατών, αποθήκευσης και χρόνου εκτέλεσης.

Στην παρούσα διπλωματική εργασία εξετάζεται ο χρονοπρογραμματισμός εργασιών παραγωγής συνεχούς ροής. Το πρόβλημα προγραμματισμού παραγωγής συνεχούς ροής (flow-shop sequencing problem) χαρακτηρίζεται ως NP-hard πρόβλημα, για το οποίο δεν υπάρχει γνωστός αποδοτικός αλγόριθμος που να δίνει βέλτιστη λύση και είναι σχεδόν αδύνατον να βρεθεί. Ως εκ τούτου, εφαρμόζονται τεχνικές μεθευρετικών αλγορίθμων για την εύρεση μιας καλής λύσης στο πρόβλημα βελτιστοποίησης καθώς η υπολογιστικής ισχύς είναι περιορισμένη και δεν επαρκεί για την εύρεση της βέλτιστης λύσης. Από το σύνολο των μεθευρετικών αλγορίθμων εφαρμόζεται, στις επόμενες ενότητες, ο αλγόριθμος τοπικής αναζήτησης με απαγόρευση κινήσεων, επιστημονικά αναγνωρισμένος ως Tabu Search algorithm.

Η πρώτη ενότητα αναφέρεται στο θεωρητικό μέρος, όπου αναλύονται οι ορισμοί βασικών εννοιών για την κατανόηση των τεχνικών που χρησιμοποιήθηκαν ενώ παράλληλα παρουσιάζονται οι βιβλιογραφικές αναφορές που υποστηρίζουν το πρόβλημα της παραγωγής συνεχούς ροής με χρήση του αλγόριθμου tabu search.

Στην δεύτερη ενότητα αναπτύσσεται η μεθοδολογία που εφαρμόστηκε για την εύρεση υποβέλτιστης λύσης. Περιγράφεται αναλυτικά το πρόβλημα της παραγωγής συνεχούς ροής, κατασκευάζεται ο αλγόριθμος τοπικής αναζήτησης με απαγόρευση θέσεων, δίνεται η μαθηματική συνάρτηση και γίνεται αναφορά στα δεδομένα εισόδου (data input). Οι αλγόριθμοι έχουν προγραμματιστεί σε γλώσσα προγραμματισμού c.

Στην τρίτη ενότητα δίνονται τα αποτελέσματα του προβλήματος με μορφή πινάκων. Παρατίθενται τα αποτελέσματα του αλγορίθμου με διάφορες στρατηγικές και συγκρίνονται με τα ανώτερα όρια σύμφωνα με Taillard και σχολιάζεται η αποτελεσματικότητα του αλγορίθμου.

1η ΕΝΟΤΗΤΑ-ΘΕΩΡΗΤΙΚΟ ΜΕΡΟΣ

1.1. Χρονικός Προγραμματισμός συστημάτων παραγωγής

Ο χρονικός προγραμματισμός στοχεύει στον προγραμματισμό ενός συνόλου εργασιών με σκοπό την βελτιστοποίηση ορισμένων κριτηρίων και την ικανοποίηση συγκεκριμένων περιορισμών.

Η πλειονότητα των προβλημάτων χρονικού προγραμματισμού παραγωγής παρουσιάζει ιδιαίτερη πολυπλοκότητα στην επίλυσή τους για την εύρεση της ολικής βέλτιστης λύσης. Στις περισσότερες περιπτώσεις, όταν το μέγεθος του προβλήματος αυξάνεται απαιτείται μεγάλο χρονικό διάστημα για την επίλυσή του και μερικές φορές η επίλυση είναι πρακτικά αδύνατη. Προβλήματα για τα οποία δεν υπάρχει γνωστός αποδοτικός αλγόριθμος που να δίνει βέλτιστη λύση και είναι σχεδόν αδύνατο να βρεθεί, ανήκουν στην κατηγορία των NP-hard προβλημάτων σύμφωνα με την ορολογία που διατύπωσε ο Knuth το 1974 [1].

Για την διευκόλυνση στη λήψη των σωστών αποφάσεων, στις περιπτώσεις όπου η υπολογιστική πολυπλοκότητα θεωρείται ως κυρίαρχο χαρακτηριστικό του προβλήματος, έχουν αναπτυχθεί διάφορες τεχνικές βελτιστοποίησης.

Για την δημιουργία και την εφαρμογή τεχνικών βελτιστοποίησης, κάθε πρόβλημα χρονικού προγραμματισμού ορίζεται από μια σειρά παραμέτρων, με βάση των οποίων ζητείται η καλύτερη δυνατή τιμή μιας συνάρτησης κόστους. Η τιμή αυτή αναφέρεται στη καλύτερη δυνατή επιλογή των δεδομένων κριτηρίων που μπορεί να αναφέρονται στη μείωση του χρόνου περάτωσης των εργασιών στις μηχανές, στην αποτελεσματικότερη αξιοποίηση της διαθέσιμης δυναμικότητας, στην γρηγορότερη εξυπηρέτηση των πελατών, κα.

Επιπλέον στο πρόβλημα του χρονικού προγραμματισμού πρέπει να ληφθούν υπόψη και οι περιορισμοί του συστήματος που αναφέρονται στη δυναμικότητα (διαθέσιμος παραγωγικός εξοπλισμός, ανθρώπινο δυναμικό), την ακολουθία της διεργασιών που ορίζει η υπάρχουσα τεχνολογία, τις απαιτήσεις για τη συντήρηση του εξοπλισμού, κ.α.

Σε γενικές γραμμές οι στόχοι του χρονικού προγραμματισμού σε ένα σύστημα παραγωγής είναι οι εξής [2]:

- Συνέπεια στις ημερομηνίες παράδοσης μιας παραγγελίας

- Ελαχιστοποίηση του χρόνου παραγωγής μιας παραγγελίας
- Ελαχιστοποίηση του χρόνου προετοιμασίας του εξοπλισμού
- Ελαχιστοποίηση των εκκρεμούντων εργασιών στο σύστημα
- Μεγιστοποίηση αξιοποίησης εξοπλισμού και/ή ανθρώπινου δυναμικού

1.1.1. Κατηγοριοποίηση προβλημάτων χρονικού προγραμματισμού

Τα προβλήματα χρονικού προγραμματισμού μπορούν να ταξινομηθούν με βάση τη συγκρότηση της διαδικασίας παραγωγής [3]. Διακρίνονται έξι περιπτώσεις [4]:

- *Μοναδική μηχανή (single machine)*

  Κάθε εργασία που αποτελείται από μία κατεργασία αντιστοιχεί στη μία και μοναδική μηχανή. Συνιστά το απλούστερο πρόβλημα χρονικού προγραμματισμού
- *Πανομοιότυπες παράλληλες μηχανές (Identical parallel machines)*

  Υπάρχουν m πανομοιότυπες μηχανές σε παράλληλη διάταξη. Η εργασία j απαιτεί μία κατεργασία και μπορεί να υλοποιηθεί από οποιαδήποτε από τις m μηχανές.
- *Παράλληλες μηχανές με διαφορετική ταχύτητα (Parallel Machines with Different Speeds)*

  Υπάρχουν m μηχανές σε παράλληλη διάταξη με διαφορετικές ταχύτητες
- Ασυσχέτιστες παράλληλες μηχανές (Unrelated parallel machines)

  Περίπτωση πιο γενικευμένη από ό, τι η παραπάνω, όπου υπάρχουν m παράλληλες μηχανές με ταχύτητα που σχετίζεται με την εργασία που υλοποιείται στην κάθε μία.
- *Συνεχούς ροής Παραγωγή (Flow-shop)*

  Υπάρχουν m μηχανές σε σειρά. Κάθε εργασία αποτελείται από παραπάνω από μία κατεργασίες. Κάθε κατεργασία αντιστοιχεί σε μία μηχανή με σταθερή σειρά διέλευσης.
- *Ανοιχτή παραγωγή (open-shop)*

  Κάθε εργασία αποτελείται από επιμέρους διεργασίες καθεμία από τις οποίες εκτελείται σε ορισμένη μηχανή. Η σειρά εκτέλεσης των διεργασιών είναι διαφορετική για κάθε εργασία και μια εργασία μπορεί να έχει παραπάνω από μία εναλλακτικές σειρές εκτέλεσης διεργασιών.

1.1.2. Κριτήριο βέλτιστου

Το κριτήριο βέλτιστου είναι από τα σημαντικότερα στοιχεία του χρονικού προγραμματισμού. Τα περισσότερα κριτήρια που αναφέρονται στην βιβλιογραφία ανήκουν στην κατηγορία των λεγόμενων κανονικών (regular) κριτηρίων.

Ένα κριτήριο βέλτιστου λέγεται κανονικό όταν ικανοποιεί τις παρακάτω τρεις συνθήκες [5]:

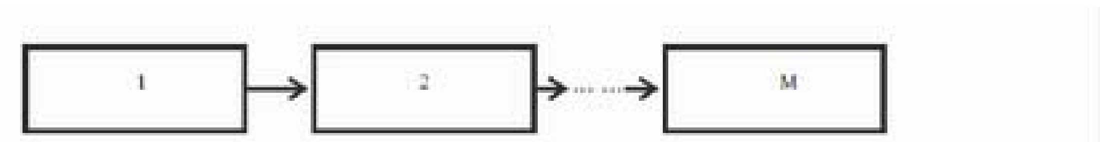
- Είναι συνάρτηση του χρόνου περάτωσης των εργασιών
- Η τιμή του πρέπει να ελαχιστοποιηθεί
- Η τιμή του αυξάνει μόνο εάν ένας από τους χρόνους περάτωσης των εργασιών αυξηθεί.

Στην κατηγορία των regular optimal criteria ανήκουν: ο χρόνος περάτωσης του προγράμματος, ο μέσος χρόνος περάτωσης των εργασιών, η μέγιστη καθυστέρηση στην περίπτωση που υπάρχουν προθεσμίες παράδοσης, η μέση καθυστέρηση και το πλήθος των καθυστερημένων εργασιών.

1.1.3. Σύστημα παραγωγής συνεχούς ροής

Το πρόβλημα χρονικού προγραμματισμού συστημάτων παραγωγής συνεχούς ροής συνιστά ένα σημαντικό πρόβλημα χρονοπρογραμματισμού και έχει ερευνηθεί εκτενώς από τότε που προτάθηκε πρώτα από τον Johnson το 1954 [6]

Τα συστήματα παραγωγής συνεχούς ροής (flow-shop) χρησιμοποιούνται για μαζική παραγωγή περιορισμένης ποικιλίας τυποποιημένων προϊόντων, όπως για παράδειγμα τα προϊόντα διατροφής. Ένα σύστημα flow-shop αποτελείται από m μηχανές (επεξεργαστές), όπου η εκτέλεση κάθε παραγγελίας n (έργου) περιλαμβάνει μέχρι m κατεργασίες (επεξεργασίες), μία σε κάθε μηχανή (επεξεργαστή).



Εικόνα 1: Σύστημα παραγωγής συνεχούς ροής

Υπάρχουν δύο κατηγορίες συστημάτων flow-shop. Η πρώτη ονομάζεται «καθαρό» σύστημα flow-shop, όπου όλες οι παρτίδες παραγωγής ενός προϊόντος για να εκτελεστούν περνούν από όλες τις μηχανές (επεξεργαστές), και η δεύτερη ονομάζεται «γενικό» flow-shop, όπου κάθε παρτίδα (έργο), αν και ακολουθεί την ίδια κατεύθυνση μέσα στο σύστημα, δεν περνάει υποχρεωτικά από όλες τις μηχανές (επεξεργαστές). Στην τελευταία περίπτωση όπου κάποιες εργασίες δεν απαιτείται να περάσουν από όλες τις μηχανές πάλι υπάρχει συνεχής ροή εφόσον η σειρά με την οποία επισκέπτονται τις υπόλοιπες μηχανές είναι η καθορισμένη.

Παράδειγμα καθαρού συστήματος flow-shop είναι μια γραμμή συναρμολόγησης, όπου η παραγωγή πραγματοποιείται από φάση σε φάση στην ίδια πάντα κατεύθυνση και περνώντας από όλους τους σταθμούς παραγωγής. Σε άλλες περιπτώσεις, όπως στην περίπτωση παραγωγής ενδυμάτων, η ίδια διαδοχή κατεργασιών (επεξεργασιών) απαιτείται για ένα μεγάλο αριθμό κομματιών μιας παραγγελίας, ενώ η διαδοχή αυτή μπορεί να αλλάζει από παραγγελία σε παραγγελία. Ένα τέτοιο σύστημα θεωρείται γενικό σύστημα flow-shop.

Για την κατηγοριοποίηση του προβλήματος παραγωγής συνεχούς ροής χρησιμοποιούμε τον παρακάτω συμβολισμό, όπως χρησιμοποιήθηκε από τους Graham et al. το 1979 [7], κατά τον οποίο το πρόβλημα χωρίζεται σε τρία πεδία:

α/β/γ   όπου

α: η δομή του προβλήματος

β: οι περιορισμοί στα δεδομένα του προβλήματος, για παράδειγμα μία εργασία δεν μπορεί να υποστεί επεξεργασία σε μία μηχανή αν βρίσκεται ακόμα βρίσκεται υπό επεξεργασία στην προηγούμενη μηχανή

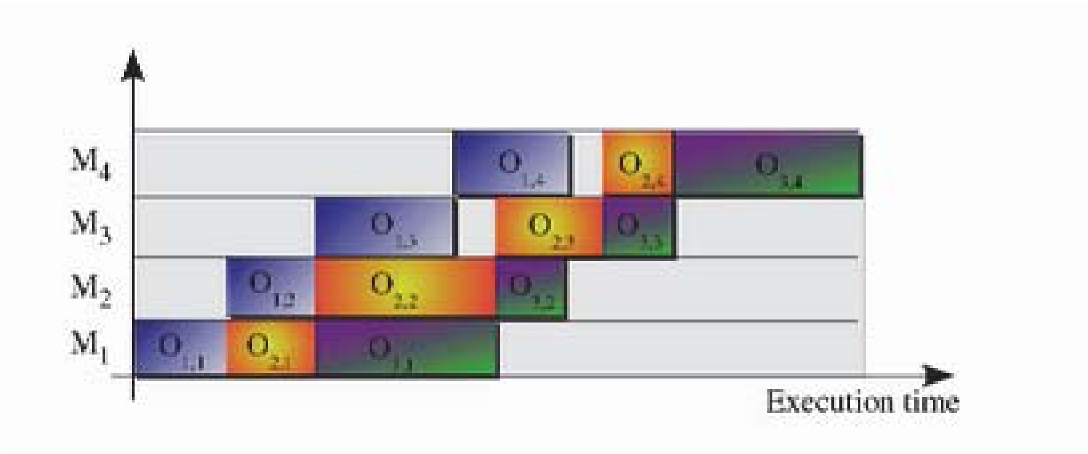
γ: το κριτήριο βέλτιστου, δηλαδή το αντικείμενο βελτιστοποίησης, π.χ. χρόνος περάτωσης

Στην παρούσα εργασία θα εξεταστεί ένα πρόβλημα αντιμετάθεσης χρονικού προγραμματισμού συστήματος συνεχούς ροής (Permutation Flow-Shop Scheduling Problem - PFSSP) με σκοπό την εφαρμογή ενός βέλτιστου χρονοπρογραμματισμού για N εργασίες σε M μηχανές.

Αναλυτικότερα, το σύστημα παραγωγής συνεχούς ροής αποτελείται από n εργασίες-jobs (i=1,2,….,n) και από m μηχανές-machines σε σειρά (j=1,2,….,m). Επιπλέον κάθε i εργασία αποτελείται από m κατεργασίες-operations και η $j^{οστή}$ κατεργασία πρέπει να υλοποιηθεί στην j μηχανή. Έτσι μία εργασία μπορεί να ξεκινήσει στη μηχανή j εφόσον έχει ολοκληρωθεί προηγουμένως στην j-1 μηχανή και η μηχανή j είναι ελεύθερη. Ο χρόνος επεξεργασίας της κάθε κατεργασίας (operation) σε κάθε μηχανή ορίζεται ίσος με $p_{i,j}$.

Μπορούμε να συμβολίσουμε τις παραπάνω προϋποθέσεις ως εξής:

- $J_i=(O_{i,1}, O_{i,2},\ldots\ldots O_{i,m})$, όπου η $O_{i,j}$ αντιπροσωπεύει την $j^{οστή}$ κατεργασία της εργασίας J_i
- Η $O_{i,j}$ κατεργασία πρέπει να υποστεί επεξεργασία στην M_j μηχανή
- Για κάθε κατεργασία υπάρχει ο αντίστοιχος χρόνος $p_{i,j}$.



Εικόνα 2: Σχηματική απεικόνιση συστήματος παραγωγής συνεχούς ροής

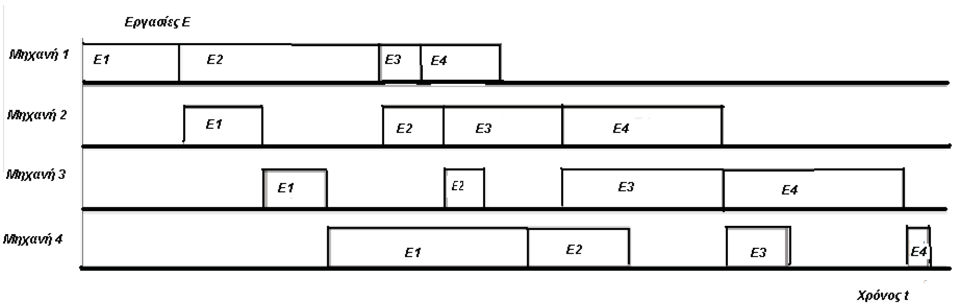
Σε ένα πρόβλημα αντιμετάθεσης παραγωγής συνεχούς ροής (Permutation Flow-Shop Scheduling Problem - PFSSP) η αλληλουχία των εργασιών προς επεξεργασία είναι η ίδια σε κάθε μηχανή. Αν η μία εργασία βρίσκεται στην $i^{οστή}$ θέση στη μηχανή 1, τότε αυτή η εργασία θα βρίσκεται σε αυτή τη θέση σε όλες τις μηχανές.

Τα παραπάνω μπορούν να εξηγηθούν στα δύο διαγράμματα που ακολουθούν που συνιστούν σχηματική απεικόνιση του προβλήματος που περιγράφεται στον πίνακα 1.

Πίνακας 1: Πρόβλημα PFSSP με 4 Μηχανές και 4 Εργασίες

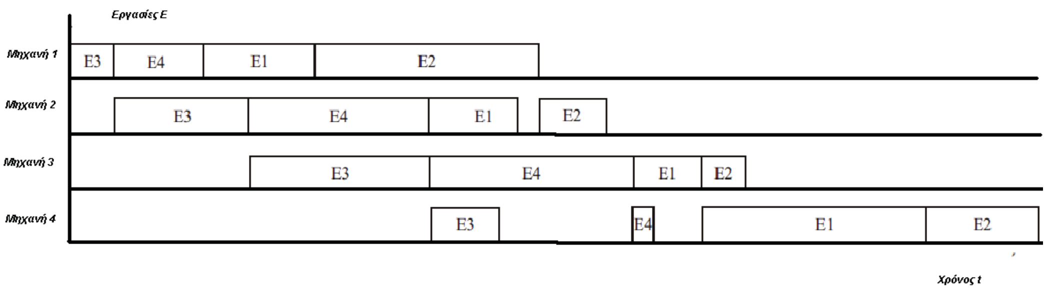
Εργασίες / Μηχανές	E1	E2	E3	E4
M1	$O_{1,1}$	$O_{2,1}$	$O_{3,1}$	$O_{4,1}$
M2	$O_{1,2}$	$O_{2,2}$	$O_{3,2}$	$O_{4,2}$
M3	$O_{1,3}$	$O_{2,3}$	$O_{3,3}$	$O_{4,3}$
M4	$O_{1,4}$	$O_{2,4}$	$O_{3,4}$	$O_{4,4}$

Οι μηχανές έχουν μία καθορισμένη σειρά: M1, M2, M3 και M4. Όταν εκτελεσθεί και η τελευταία διεργασία τότε ολοκληρώνεται η εκάστοτε εργασία. Δηλαδή όταν περάσουν όλες οι εργασίες και από την τελευταία μηχανή, την M4, θα έχουν ολοκληρωθεί όλες οι εργασίες.



Εικόνα 3: Διάγραμμα Gantt για σειρά εκτέλεσης εργασιών Ε1,Ε2,Ε3, Ε4

Στο επόμενο διάγραμμα απεικονίζονται πάλι οι χρονικές διάρκειες των διεργασιών, όμως αυτή τη φορά η σειρά πραγματοποίησης των εργασιών είναι διαφορετική: Ε3, Ε4, Ε1, Ε2.



Εικόνα 4: Διάγραμμα Gantt για σειρά εκτέλεσης εργασιών Ε3,Ε4,Ε1,Ε2

Στην παρούσα εργασία του κριτήριο του βέλτιστου είναι η ελαχιστοποίηση του χρόνου περάτωσης της τελευταίας εργασίας (makespan). Στόχος του προβλήματος είναι η αναζήτηση της βέλτιστης μετάθεσης (permutation), δηλαδή αυτής της αλληλουχίας των εργασιών που θα μειώσει σε έναν καλό βαθμό τον χρόνο περάτωσης των εργασιών [8].

Έστω Π={Π1,Π2,…Πn} είναι η αλληλουχία των εργασιών. Η ολοκλήρωση μιας εργασίας J_i στη μηχανή M_j συμβολίζεται με $C(\pi_i,j)$ και ορίζεται ως ακολούθως:

$C(\pi_i,j)= \max \{C(\pi_{i-1},j), C(\pi_i,j-1)\}+p_{\pi i,j}$

Με i=1,2,….,n και j=1,2,……m

Ο χρόνος λήξης του προγράμματος μιας συγκεκριμένης αντιμετάθεσης ορίζεται ως $C_{max}(\pi)=C(\pi_n,m)$, δηλαδή ο χρόνος ολοκλήρωσης των εργασιών είναι ο χρόνος περάτωσης της τελευταίας εργασίας $J\pi_n$ στην τελευταία μηχανή M_m.

Το πρόβλημα παραγωγής συνεχούς ροής που μελετάται στην εργασία αυτή είναι της μορφής (βάσει του συμβολισμού των Graham et al [7]):

N/M/F,Permu/Cmax

Ο συμβολισμός N/M αναφέρεται στο πεδίο «α» της γενικής κατάταξης (βλέπε παραπάνω), όπου N εργασίες υλοποιούνται σε M μηχανές. Στο πεδίο των περιορισμών το F δηλώνει ότι πρόκειται για πρόβλημα παραγωγής συνεχούς ροής, ενώ το permu ότι οι λύσεις περιορίζονται σε προβλήματα μετάθεσης μόνο. Το Cmax αναφέρεται στο κριτήριο βέλτιστου που είναι η ελαχιστοποίηση του χρόνου περάτωσης των εργασιών.

Για την επίλυση του παραπάνω προβλήματος λαμβάνονται υπόψη κάποιες παραμέτρους όπως διατυπώνονται από την βιβλιογραφία [5,9]

- Μια ομάδα N ασυσχέτιστων εργασιών είναι διαθέσιμη για επεξεργασία τη χρονική στιγμή 0.
- Η κάθε εργασία απαιτεί M διεργασίες και η κάθε διεργασία απαιτεί διαφορετική μηχανή.
- Μια εργασία είναι διαθέσιμη σε μία μηχανή, μόνο αν η επεξεργασία της έχει ολοκληρωθεί στην αμέσως προηγούμενη. Όλες οι εργασίες είναι διαθέσιμες στην πρώτη μηχανή του συστήματος.
- Κάθε μηχανή μπορεί να επεξεργάζεται το πολύ μία εργασία σε δεδομένη χρονική στιγμή.

- Οι χρόνοι εξάρμωσης[1] (setup time) είναι «εξαρτημένοι ακολουθίας» (sequencing dependent) των μηχανών και συμπεριλαμβάνονται στους χρόνους επεξεργασίας.
- Οι χρόνοι επεξεργασίας των διεργασιών από τις μηχανές είναι γνωστοί από την αρχή.
- Όλες οι μηχανές είναι πάντα διαθέσιμες.
- Όταν ξεκινήσει η διεργασία συνεχίζει χωρίς καμία διακοπή.

Ο χρονικός προγραμματισμός σε παραγωγικά συστήματα flow-shop αφορά την εύρεση της βέλτιστης μεθόδου δρομολόγησης των εργασιών, δηλαδή τέτοιας που να ικανοποιεί στο μέγιστο βαθμό τα επιλεγμένα κριτήρια απόδοσης, λαμβάνοντας πάντα υπόψη τους υπάρχοντες περιορισμούς. Όμως, ακόμα και για σχετικά απλά προβλήματα δεν είναι εύκολο να βρεθούν βέλτιστες λύσεις. Η εφαρμογή των μεθευρετικών αλγορίθμων και κυρίως του αλγορίθμου Tabu Search είναι μία αποδοτική τεχνική για την εύρεση μιας καλής λύσης, όπως αναλύεται παρακάτω.

1.2. Ευρετικοί-Μεθευρετικοί Αλγόριθμοι

Η επίλυση ενός προβλήματος χρονικού προγραμματισμού γίνεται ολοένα και δυσκολότερη όσο αυξάνει το μέγεθος του προβλήματος και πολλές φορές το να προσπαθούμε να βρούμε την ολικά βέλτιστη λύση σε λογικό χρόνο είναι πρακτικά αδύνατο. Για να επιλυθούν προβλήματα αυτής της μορφής συχνά καταφεύγουμε σε διαφορετικές τεχνικές που μας οδηγούν σε μια σχεδόν βέλτιστη, αλλά ικανοποιητική λύση. Μια λύση ενός ευρετικού αλγορίθμου γίνεται αποδεκτή αν ικανοποιεί κάποια κριτήρια όπως η ποιότητα της λύσης, δηλαδή η απόκλισή της από τη βέλτιστη, η ευκολία απόκτησης μιας λύσης, η λογική πάνω στην οποία στηρίζονται οι κανόνες του ευρετικού αλγορίθμου που χρησιμοποιήθηκαν για να οδηγηθούμε στη λύση. Για κάθε πρόβλημα βελτιστοποίησης δεν υπάρχει μονάχα ένας ευρετικός αλγόριθμος που να δίνει τη βέλτιστη λύση, αλλά έχουν αναπτυχθεί αρκετοί αλγόριθμοι οι οποίοι συγκρινόμενοι μεταξύ τους, οδηγούν ολοένα και σε καλύτερες λύσεις.

[1] Χρόνος εξάρμωσης μιας μηχανής (setup time) είναι ο χρόνος προετοιμασίας που χρειάζεται η μηχανή. Αν εξαρτάται μόνο από το είδος της εργασίας που πρόκειται να εκτελεσθεί στη μηχανή λέγεται «ανεξάρτητος ακολουθίας» (sequence independent), ενώ αν εξαρτάται και από το είδος της εργασίας της οποίας η επεξεργασία προηγήθηκε στη μηχανή, λέγεται «εξαρτημένος ακολουθίας» (sequence dependent). Στην παρούσα εργασία υποθέτουμε ότι οι χρόνοι εξάρμωσης περιλαμβάνονται στον χρόνο επεξεργασίας και δεν λαμβάνονται υπόψη.

Ένα άλλο θέμα που οφείλεται να εξεταστεί είναι η ποιότητα της λύσης των ευρετικών αλγορίθμων. Έχουν διατυπωθεί διάφορες μέθοδοι για τον σκοπό αυτό [10] καθώς και τεχνικές για την επιλογή αλγορίθμου [11]. Ένας κλασικός τρόπος για τη σύγκριση της αποδοτικότητας διαφορετικών αλγορίθμων είναι ο υπολογιστικός χρόνος που απαιτείται για την επίλυση του ίδιου προβλήματος και η εν τέλει η σύγκριση των αποτελεσμάτων τους. Το μεγαλύτερο πρόβλημα του συγκεκριμένου τρόπου είναι ότι το δείγμα που δοκιμάζεται δε μπορεί να είναι αντιπροσωπευτικό για όλες τις περιπτώσεις. Δεν σημαίνει δηλαδή ότι αν ένας αλγόριθμος βελτιστοποιεί ένα σετ δεδομένων ενός συγκεκριμένου προβλήματος θα συμβαίνει το ίδιο και με όλα τα πιθανά σετ δεδομένων.

1.2.1. Ιστορικό εξέλιξης

Το πρόβλημα αντιμετάθεσης χρονικού προγραμματισμού ενός συστήματος συνεχούς ροής (PFSSP) συνιστά πρόβλημα NP-hard όπως αποδεικνύεται από τους Garey et al. το 1976 [12]. Γι αυτό το λόγο οι προσπάθειες είναι συντονισμένες ως προς την κατεύθυνση αναζήτησης λύσεων υψηλής ποιότητας σε σχετικά μικρό «υπολογιστικό» χρόνο και όχι ως προς την κατεύθυνση εύρεσης βέλτιστων λύσεων [8].

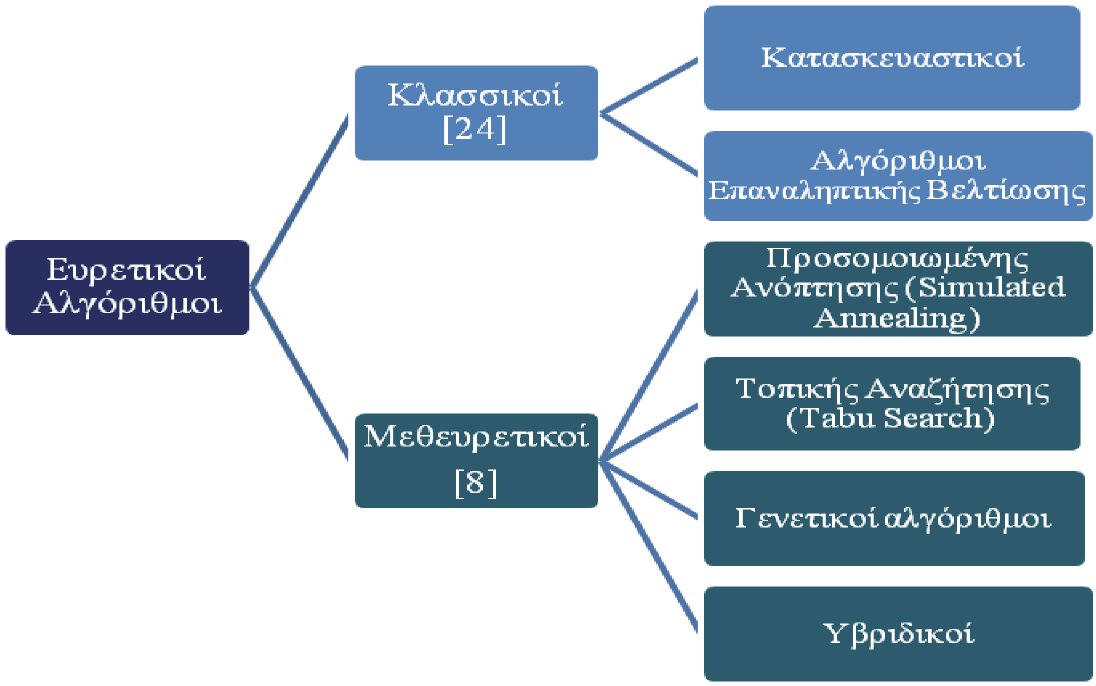
Οι ευρετικοί αλγόριθμοι που προτείνονται για την επίλυση του PFSSP προβλήματος διακρίνονται στους κατασκευαστικούς (constructive) και στους εξελικτικούς (improvement) αλγορίθμους [13]. Σε έναν κατασκευαστικό ευρετικό αλγόριθμο όταν η αλληλουχία των εργασιών καθοριστεί τότε οριστικοποιείται και δεν μπορεί να ανατραπεί. Στην βιβλιογραφία υπάρχουν πολλοί ερευνητές που έχουν μελετήσει τους κατασκευαστικούς ευρετικούς αλγορίθμους για την επίλυση του εν λόγω προβλήματος [14]. Οι τεχνικές αυτές αν και γρήγορες χαρακτηρίζονται από λύσεις χαμηλής ποιότητας. Η καλύτερη μέθοδος κατασκευαστικού ευρετικού αλγορίθμου είναι η NEH όπως προτάθηκε από τους Nawaz et al. το 1983 [15]. Από την άλλη πλευρά οι εξελικτικοί ευρετικοί αλγόριθμοι ξεκινούν με μία αρχική λύση και παρέχουν ένα σχήμα όπου αναζητείται η εξελιγμένη λύση μέσα από επαναληπτικές διαδικασίες. Σε αυτήν την κατηγορία ανήκουν οι μεθευριτικοί αλγόριθμοι. Τα τελευταία χρόνια πολλές διαφορετικές μέθοδοι μεθευρετικών αλγορίθμων έχουν εφαρμοστεί σε PFSSP προβλήματα με κριτήριο βέλτιστου την ελαχιστοποίηση του χρόνου περάτωσης των εργασιών (makespan).

Οι δημοφιλέστεροι αλγόριθμοι είναι οι εξής: (i) αλγόριθμος προσομοιωμένης ανόπτησης-simulated annealing [16,17], (ii) αλγόριθμος τοπικής αναζήτησης-tabu search [18,19,20], (iii) γενετικοί αλγόριθμοι-genetic algorithms [21] και (iv) υβριδικοί μεθευρετικοί-hybrid metaheuristics [22,23]. Οι μεθευρετικοί αλγόριθμοι αποδίδουν λύσεις καλύτερης ποιότητας και ο «υπολογιστικός» χρόνος είναι πολύ μικρότερος.

1.2.2. Ταξινόμηση ευρετικών-μεθευρετικών αλγορίθμων

Οι ευρετικοί αλγόριθμοι ανήκουν στην ευρύτερη κατηγορία των προσεγγιστικών αλγορίθμων και διαχωρίζονται στις ακόλουθες υποκατηγορίες [8,24]:

- Κλασσικοί Ευρετικοί Αλγόριθμοι
- Μεθευρετικοί Αλγόριθμοι (Αλγόριθμοι Υπολογιστικής Ευφυΐας)



Εικόνα 5: Ταξινόμηση ευρετικών αλγορίθμων βάσει της βιβλιογραφίας [8,24]

Οι μεθευρετικοί αλγόριθμοι συνιστούν αναβαθμισμένοι μορφή των ευρετικών αλγορίθμων. Ο όρος μεθευρετικός αλγόριθμος χρησιμοποιήθηκε πρώτη φορά από τον Glover το 1986 [25] θέλοντας να περιγράψει «μια ανώτερη στρατηγική η οποία καθοδηγεί και τροποποιεί άλλους ευρετικούς στο να παράγουν λύσεις πέρα από αυτές που παράγονται κατά την έρευνα της ευνοϊκότερης τοπικής συνθήκης. Ένας μεθευρετικός αλγόριθμος αποτελεί μια ευφυή διαδικασία επαναληπτικής βελτίωσης η οποία χρησιμοποιεί μη-εξαρτημένους από το εξεταζόμενο πρόβλημα μηχανισμούς καθοδήγησης υποδεέστερων ευρετικών, με σκοπό την επίτευξη ευρωστίας (robustness), της ισορροπίας δηλαδή ανάμεσα στην ικανότητα παραγωγής υψηλής ποιότητας λύσεων σε συγκεκριμένα προβλήματα, από τη μία πλευρά, και στην ευελιξία που απαιτείται για την επιβίωση σε πολλά διαφορετικά περιβάλλοντα, από την άλλη (διαφορετικοί περιορισμοί, πόροι, κλπ.).

Ο στόχος της χρήσης των μεθευρετικών αλγορίθμων είναι ο ταχύς προσδιορισμός των περιοχών αυτών που «φέρουν» λύσεις υψηλής ποιότητας ενώ παράλληλα αποφεύγεται η αναζήτηση σε περιοχές που είτε έχουν εξερευνηθεί ήδη, είτε δεν «φέρουν» λύσεις υψηλής ποιότητας. Για την επίτευξη του παραπάνω στόχου γίνεται χρήση στρατηγικών εντατικοποίησης[2] και διαφοροποίησης[3] της αναζήτησης καθώς και απεγκλωβισμός από τα τοπικά ελάχιστα.

Συνοψίζοντας, οι μεθευρετικοί αλγόριθμοι επιλύουν περιπτώσεις NP-hard προβλημάτων εξερευνώντας συνήθως ένα μεγάλο πεδίο λύσεων, το οποίο σταδιακά μειώνεται με σκοπό να βρεθεί μια καλύτερη λύση. Επιπλέον, είναι εύκολοι στο σχεδιασμό και πολύ ευέλικτοι. Έτσι, τα τελευταία χρόνια οι περισσότεροι αλγόριθμοι που έχουν αναπτυχθεί για την επίλυση των προβλημάτων συνδυαστικής βελτιστοποίησης ανήκουν σε αυτή την κατηγορία [26].

1.2.3. Ανασκόπηση αλγορίθμων

[2] Ο όρος αναφέρεται στην επικέντρωση της αναζήτησης σε περιοχές του χώρου λύσεων με υψηλής ποιότητας λύσεις (επικεντρώνεται η αναζήτηση σε μικρά σημεία του χώρου έρευνας [24].

[3] Ο όρος αναφέρεται στην μετακίνηση της αναζήτησης σε ανεξερεύνητες περιοχές του χώρου των λύσεων όταν κρίνεται απαραίτητο (σκοπός να εξεταστούν τελείως διαφορετικές περιοχές του χώρου έρευνας [24].

I. Ο αλγόριθμος Johnson για δύο μηχανές

Το πρόβλημα της παραγωγής συνεχούς ροής αντιμετωπίστηκε πρώτη φορά από τον Johnson το 1954 [6] σε ένα πρόβλημα δύο μηχανών, κατά το οποίο οι N εργασίες πρέπει να εκτελεστούν σε δύο μηχανές ώστε να ελαχιστοποιηθεί ο συνολικός χρόνος επεξεργασίας τους. Ο Johnson απέδειξε ότι η βέλτιστη λύση του προβλήματος N/2/F/Cmax δίνεται από ένα μεταθετικό πρόγραμμα στο οποίο μία εργασία i προηγείται μιας εργασία j μόνο εάν

$$\min(p_{i1},p_{j2})\leq\min(p_{j1},p_{i2})$$

Όπως υποδεικνύει η παραπάνω πρόταση, στην αρχή του προγράμματος τοποθετούνται οι εργασίες που έχουν τους μικρότερους χρόνους επεξεργασίας στην πρώτη μηχανή, ώστε η δεύτερη μηχανή να αρχίσει νωρίς της επεξεργασία τους. Οι εργασίες που έχουν τους μικρότερους χρόνους επεξεργασίας στη δεύτερη μηχανή, τοποθετούνται στο τέλος του προγράμματος, ώστε η εκτέλεσή τους να ολοκληρωθεί γρήγορα [5].

Στο πρόβλημα N/3/F/Cmax, όπου ο προγραμματισμός των εργασιών γίνεται σε 3 μηχανές, ο Johnson έδειξε ότι ο αλγόριθμός του δίνει τη βέλτιστη λύση στην ειδική περίπτωση που οι χρόνοι επεξεργασίας των κατεργασιών στη δεύτερη μηχανή είναι μικρότεροι από όλους τους χρόνους επεξεργασίας των κατεργασιών στην πρώτη και στην τρίτη μηχανή. Επομένως αν ισχύει:

$$\min p_{i1} \geq \max p_{i2} \text{ ή } \min p_{i3} \geq \max p_{i2}$$

τότε η βέλτιστη λύση του προβλήματος δίνεται από ένα μεταθετικό πρόγραμμα, στο οποίο μία εργασία i προηγείται μιας εργασίας j μόνο αν

$$\min (p_{i1}+p_{i2},p_{j3}+p_{j2}) \leq \min (p_{j1}+p_{j2}, p_{i3}+p_{i2})$$

II. Επέκταση σε Μ μηχανές

Μετά τη δράση του Johnson πολλοί ερευνητές έπειτα στράφηκαν στον προγραμματισμό N εργασιών σε Μ μηχανές κατά βέλτιστο τρόπο, με κριτήριο βέλτιστου τον συνολικό χρόνο περάτωσης των εργασιών Cmax. Τα πρώτα αποτελέσματα της έρευνας του γενικότερου προβλήματος παραγωγής συνεχούς ροής ήταν οι αλγόριθμοι συνδυαστικής βελτιστοποίησης [27,28].

Ωστόσο, παρά τη σημαντική προσπάθεια, κανένα ουσιαστικό βήμα δεν πραγματοποιήθηκε προς την επέκταση των αποτελεσμάτων του Johnson στο πρόβλημα της παραγωγής συνεχούς ροής, καθώς αποδείχθηκε πως το πρόβλημα της μορφής N/M/F,permu/Cmax ανήκει στην κατηγορία των NP-πλήρων προβλημάτων. Ο τελευταίος χαρακτηρισμός του γενικότερου προβλήματος παραγωγής συνεχούς ροής ως δυσεπίλυτου οδήγησε την ερευνητική δράση στην ανάπτυξη ευρηματικών αλγορίθμων για την εύρεση προσεγγιστικών λύσεων. Η μέθοδος της τοπικής αναζήτησης με περιορισμό κινήσεων ανήκει στην κατηγορία των ευρηματικών μεθόδων εύρεσης υπο-βέλτιστων περιοχών, όπως θα αναπτυχθεί στο επόμενο κεφάλαιο.

1.3. Τοπική Αναζήτηση με Απαγόρευση Κινήσεων

1.3.1. Αλγόριθμος τοπικής αναζήτησης

Η μέθοδος της τοπικής αναζήτησης είναι μία ευρηματική μέθοδος εύρεσης υπο-βέλτιστων περιοχών. Οι τεχνικές τοπικής αναζήτησης (Local Search-LS) προϋποθέτουν την ύπαρξη μιας αρχικής λύσης στην οποία εφαρμόζονται διάφορες αλλαγές προκειμένου να βρεθεί μία καλύτερη λύση.

Η αρχή λειτουργίας της συνοψίζεται παρακάτω. Αρχικά επιλέγεται μια αρχική λύση από το χώρο αναζήτησης (γειτονιά λύσεων S_N) και ονομάζεται τρέχουσα. Στη συνέχεια εφαρμόζεται ένας μετασχηματισμός στην τρέχουσα λύση με αποτέλεσμα τη δημιουργία μιας νέας λύσης. Έπειτα γίνεται η σύγκριση της νέας με την τρέχουσα λύση και στην περίπτωση που η πρώτη είναι καλύτερη αντικαθιστά την τρέχουσα λύση, διαφορετικά απορρίπτεται. Η διαδικασία επαναλαμβάνεται συνήθως μέχρι να μην υπάρχει δυνατότητα περεταίρω βελτίωσης της λύσης ή μετά από κάποιο συγκεκριμένο αριθμό επαναλήψεων.

Αναλυτικότερα: Αν $Π_0$ είναι το πρόγραμμα που αποτελεί την αρχική λύση, τότε κατά την αναζήτηση εξετάζονται το προγράμματα μιας περιοχής $U(Π_0)$ του χώρου S_N, για να επιλεγεί ένα $Π_1$ πρόγραμμα με την ιδιότητα $Cmax(Π_1)<Cmax(Π_0)$. Η περιοχή $U(Π_0)$ ονομάζεται γειτονία του $Π_0$ και περιέχει προγράμματα τα οποία διαφέρουν από το πρόγραμμα $Π_0$, ως προς τις θέσεις ενός μικρού αριθμού εργασιών. Η επαναληπτική εξέταση των γειτονιών και η επιλογή του ελαχίστου καθεμιάς έχει ως αποτέλεσμα το σχηματισμό μιας ακολουθίας $\{Π_k\}$ προγραμμάτων, όπου $Π_{k+1}$ є $U(Π_k)$ και $Cmax(Π_{k+1})$ $<Cmax(Π_k)$. Η εκτέλεση της μεθόδου τερματίζεται όταν βρεθεί ένα τοπικό ελάχιστο Π* του χώρου S_N, οπότε η αναζήτηση δεν μπορεί πλέον να συνεχιστεί.

Ο αλγόριθμος της τοπικής αναζήτησης βελτιώνει την αρχική λύση, την οποία παράγει στο πρώτο βήμα, εξετάζοντας διαδοχικές γειτονιές προγραμμάτων. Επειδή, όμως, τα προγράμματα που υπάρχουν σε μια γειτονιά δε διαφέρουν σημαντικά μεταξύ τους, η βελτίωση που επιτυγχάνεται σε κάθε επανάληψη είναι συνήθως μικρή. Αυτό ενέχει τον κίνδυνο του εγκλωβισμού σε τοπικά ακρότατα.

Οι αλγόριθμοι τοπικής αναζήτησης εξετάζουν σε κάθε επανάληψη μια γειτονιά προγραμμάτων της τρέχουσας λύσης επιλέγοντας το ελάχιστο της γειτονιάς. Με τον τρόπο αυτό η λύση βελτιώνεται σταδιακά μέχρι να βρεθεί κάποιο τοπικό ελάχιστο. Στο σημείο αυτό η αναζήτηση εγκλωβίζεται στη γειτονιά του τοπικού ελάχιστου, καθώς καμία κίνηση προς καμία κατεύθυνση δεν βελτιώνει πλέον τη λύση. Στην περίπτωση αυτή η τοπική αναζήτηση συνήθως τερματίζεται και το τοπικό ελάχιστο αποτελεί και την τελική λύση. Αφού το πρόγραμμα που αποτελεί τη βέλτιστη λύση, είναι ένα από τα τοπικά ελάχιστα του χώρου S_N, η ποιότητα της τελικής λύσης που παράγουν οι αλγόριθμοι τοπικής αναζήτησης θα εξαρτάται από το πλήθος των τοπικών ελαχίστων [5]. Καθώς η δομή του χώρου S_N δεν είναι γνωστή, το πλήθος των τοπικών ελαχίστων που περιέχονται μέσα σε αυτόν δεν μπορεί να υπολογιστεί ακριβώς. Θεωρείται, όμως, ανάλογο του μεγέθους του χώρου αυτού, με αποτέλεσμα σε μεγάλου μεγέθους χώρους το πλήθος των τοπικών ελαχίστων αναμένεται να είναι σημαντικό. Ως εκ τούτου, ο τερματισμός της αναζήτησης με το πρώτο τοπικό ελάχιστο που εντοπίζεται δεν κρίνεται ικανοποιητικός και θεωρείται απαραίτητη η συνέχιση της αναζήτησης, ώστε να εξεταστεί ο μεγαλύτερος δυνατός αριθμός τοπικών ελαχίστων [29].

Για την αντιμετώπιση της παραπάνω δυσκολίας εφαρμόζεται ο αλγόριθμος τοπικής αναζήτησης με περιορισμό κινήσεων, επιστημονικά αναγνωρισμένος ως tabu search algorithm.

1.3.2 Αλγόριθμος Tabu Search-TS

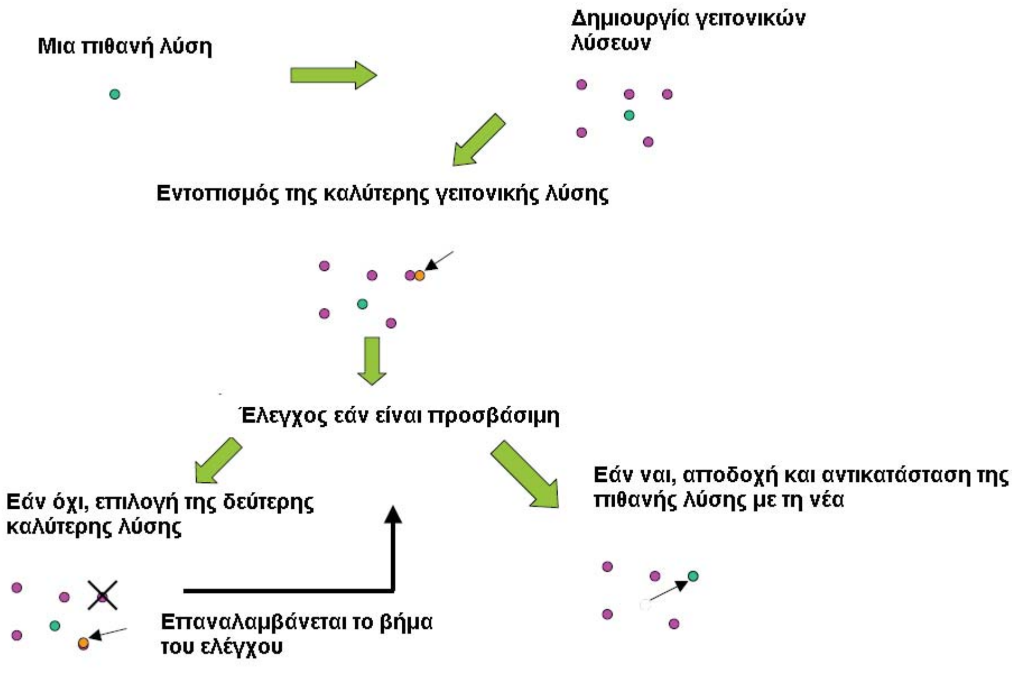
Ο αλγόριθμος Tabu Search είναι ένας μεθευρετικός αλγόριθμος ο οποίος χρησιμοποιεί την τεχνική της τοπικής αναζήτησης, εμπλουτίζοντας τη λειτουργία της, με σκοπό την επίλυση πολλών συνδυαστικών προβλημάτων βελτιστοποίησης, κυρίως στο πεδίο του χρονοπρογραμματισμού. Ο αλγόριθμος Tabu Search βελτιώνει τις λειτουργίες της μεθόδους τοπικής αναζήτησης χρησιμοποιώντας «δομές μνήμης-memory structures» στις οποίες καταχωρούνται οι ήδη εντοπισμένες λύσεις.

Ο αλγόριθμος tabu search εισήχθη πρώτα από τον Glover το 1986 [25]. Ο αλγόριθμος TS εξετάζει μια σειρά ακολουθιών από λύσεις και μετακινείται στην καλύτερη γειτονιά της πρόσφατα εντοπισμένης λύσης. Για να αποφευχθούν οι κύκλοι, οι λύσεις οι οποίες πρόσφατα εξετάστηκαν χαρακτηρίζονται ως απαγορευμένες ή «tabu» για έναν αριθμό από επαναλήψεις.

Ο Thaillard το 1993 [18] εισάγει ένα νέο γνώρισμα κατά το οποίο το κυρίως πρόβλημα αποδομείται σε ανεξάρτητα υπό-προβλήματα ώστε ο αλγόριθμος να παραλληλοποιείται σε πολλαπλούς επεξεργαστές. Κάθε υπό-πρόβλημα επιλύεται σε έναν διαφορετικό επεξεργαστή πριν να ομαδοποιηθούν οι ξεχωριστές αναζητήσεις για τη δημιουργία μίας νέας λύσης στο αρχικό πρόβλημα. Έπειτα, η νέα λύση αποδομείται εκ νέου και η διαδικασία επαναλαμβάνεται για συγκεκριμένο αριθμό επαναλήψεων. Η τυχαία επιλογή των συνιστωσών από την διαδικασία της αποδόμησης εξασφαλίζει ότι ο αλγόριθμος θα δώσει διαφορετικές λύσεις από τη μία εκτέλεση στην άλλη.

Η τοπική αναζήτηση με περιορισμό κινήσεων ξεκινάει με τον εντοπισμό ενός τοπικού ελαχίστου. Για την αποφυγή των κύκλων γύρω από το τοπικό ελάχιστο μίας γειτονιάς, η μέθοδος καταγράφει τις πρόσφατες κινήσεις (ή λύσεις) σε μία ή περισσότερες tabu λίστες. Ο αριθμός των κινήσεων στη λίστα καθορίζεται από το μέγεθος της tabu λίστας, που συμβολίζεται με T.

Στην εκκίνηση της μεθόδου πραγματοποιείται μια χονδροειδής εξέταση στο χώρο των λύσεων, που ονομάζεται «διαφοροποίηση-diversification», αλλά καθώς εντοπίζονται υποψήφιες τοποθεσίες (candidate locations), η αναζήτηση επικεντρώνεται στην εύρεση τοπικών βέλτιστων λύσεων στα πλαίσια της διαδικασίας της «εντατικοποίησης-intensification».



Εικόνα 6: Η βασική δομή της μεθόδου Tabu Search

1.3.2 Ο αλγόριθμος Tabu Search σε πρόβλημα παραγωγής συνεχούς ροής

Η εφαρμογή της μεθόδου TS στο πρόβλημα αντιμετάθεσης της παραγωγής συνεχούς ροής ορίζεται συνήθως έτσι ώστε ο αλγόριθμος να ξεκινάει από μία αρχική ακολουθία εργασιών και να μετακινείται διαδοχικά ανάμεσα από γειτονικές ακολουθίες. Σε κάθε επανάληψη, μία κίνηση πραγματοποιείται σε μία καλύτερη ακολουθία που υπάρχει μέσα στη γειτονιά της πρόσφατης ακολουθίας που είχε επιλεγεί χωρίς απαραίτητα να συνιστά μία βέλτιστη λύση. Σε μεγαλύτερα προβλήματα, όπου ο υπολογιστικός χρόνος για τον εντοπισμό της καλύτερης λύσης από όλες τις γειτονικές της είναι πολύ μεγάλος, δημιουργείται μία λίστα με υποψήφιες διαδικασίες για να περιορίσουν τις εναλλακτικές. Η μέθοδος απαγορεύει ακολουθίες με συγκεκριμένα γνωρίσματα αποσκοπώντας στην αποφυγή κύκλων και στην καθοδήγηση της αναζήτησης σε ανεξερεύνητες περιοχές των χώρου των λύσεων [30].

2η ΕΝΟΤΗΤΑ-ΜΕΘΟΔΟΛΟΓΙΑ

2.1 Δομή προγράμματος –Παρουσίαση αλγορίθμου

Στην παρούσα ενότητα θα δούμε τον αλγόριθμο.

Αποτελείται από τον κορμό και από τρεις συναρτήσεις.

ΣΥΝΑΡΤΗΣΗ 1

Η πρώτη συνάρτηση δέχεται ως όρισμα ένα πίνακα χρόνων, στον οποίο είναι αποθηκευμένοι οι χρόνοι που κάνει κάθε εργασία σε κάθε μηχανή, και επιστρέφει τον συνολικό χρόνο διεκπεραίωσης όλων των εργασιών. Αυτό για να γίνει χρησιμοποιείται ένας νέος πίνακας στον οποίο αποθηκεύεται ο χρόνος περάτωσης μιας εργασίας, έχοντας ξεκινήσει να μετράμε από την αρχή, δηλαδή τη στιγμή έναρξης της πρώτης εργασίας.

Ειδικότερα:

```
Για κάθε μηχανή i
        Για κάθε εργασία j
                Αν  στοιχείο [i][j-1] < στοιχείο[i-1][j]
                        στοιχείο[i][j]=στοιχείο[i-1][j]+χρόνος[i][j];
                Αλλιώς  στοιχείο[i][j]=στοιχείο[i][j-1]+χρόνος[i][j];
```

Έτσι δημιουργείται ο παραπάνω πίνακας, και ο συνολικός χρόνος, δηλαδή η επιστρεφόμενη τιμή της συνάρτησης είναι το τελευταίο στοιχείο του πίνακα.

ΣΥΝΑΡΤΗΣΗ 2

Η δεύτερη συνάρτηση δέχεται ως ορίσματα ένα διάνυσμα που περιγράφει μια σειρά εργασιών καθώς και έναν πίνακα με τους χρόνους της κάθε εργασίας στην κάθε μηχανή και επιστρέφει τον συνολικό χρόνο διεκπεραίωσης όλων των εργασιών αν περάσουν από τις μηχανές με την δοθείσα ως όρισμα σειρά. Αυτό γίνεται αλλάζοντας τον δοθέντα ως όρισμα πίνακα έτσι ώστε να ταιριάζει με την σειρά που επίσης έχει δοθεί ως όρισμα. Ύστερα καλείται η πρώτη συνάρτηση για αυτόν τον πίνακα και επιστρέφεται ο ζητούμενος χρόνος.

ΣΥΝΑΡΤΗΣΗ 3

Η συνάρτηση αυτή δέχεται ως ορίσματα ένα διάνυσμα που περιγράφει μία σειρά εργασιών καθώς και έναν πίνακα με τους χρόνους της κάθε εργασίας στην κάθε μηχανή, όπως στη δεύτερη συνάρτηση.

Στο διάνυσμα που περιγράφει την δοθείσα σειρά εργασιών, εφαρμόζεται ανταλλαγή 1-1. Δηλαδή κάθε στοιχείο του διανύσματος δίνει την θέση του σε ένα άλλο και παίρνει τη θέση του δεύτερου, ελέγχοντας κάθε φορά τον συνολικό χρόνο διεκπεραίωσης όλων των εργασιών χρησιμοποιώντας την δεύτερη συνάρτηση για την προκύπτουσα σειρά.

Σε περίπτωση που προκύψει καλύτερος χρόνος τα στοιχεία του διανύσματος που ανταλλαχθήκαν τοποθετούνται σε ένα διάνυσμα το οποίο έχει συγκεκριμένο μέγεθος. Το διάνυσμα αυτό είναι η tabu-list δηλαδή η λίστα με τις απαγορευμένες κινήσεις. Αυτό σημαίνει ότι στην επόμενη ανταλλαγή 1-1 δεν θα συμπεριληφθούν οι εργασίες που βρίσκονται στην tabu-list. Εξαιρούνται οι περιπτώσεις που πληρείται το κριτήριο φιλοδοξίας, δηλαδή η περίπτωση στην οποία η συμμετοχή μίας εργασίας, που βρίσκεται στην λίστα των απαγορευμένων κινήσεων, στην 1-1 ανταλλαγή οδηγεί σε καλύτερο αποτέλεσμα.

Μόλις ολοκληρωθεί η εκτέλεση της συνάρτησης επιστρέφεται ένας δείκτης που δείχνει στο διάνυσμα στο οποίο είναι αποθηκευμένο η βέλτιστη σειρά εργασιών.

Ειδικότερα:

```
Για κάθε εργασία i
    Για κάθε εργασία j
        Βάλε την i στην θέση της j
        Υπολόγισε χρόνο περάτωσης Ttemp
        Αν Ttemp < T
            i,j να μπουν στην λίστα απαγορευμένων κινήσεων
            T = Ttemp
        Αλλιώς
            Για κάθε στοιχείο της λίστας tabu
                Αν η εργασία I,J υπάρχει στη λίστα
                    Τότε βαλε την j στην θέση της j όπως ήταν πριν
```

ΒΑΣΙΚΟΣ ΚΟΡΜΟΣ ΠΡΟΓΡΑΜΜΑΤΟΣ

Για όσες επαναλήψεις θέλει ο χρήστης

Random σειρά εργασιών

Κάλεσμα της τρίτης συνάρτησης

Αν ο χρόνος tabu είναι καλύτερος από τον χρόνο

Χρόνος = χρόνος tabu

ΙΙ.2 Εφαρμογή σε βιβλιογραφικά δεδομένα

Οι αλγόριθμοι που περιγράφηκαν στην ενότητα 1 κωδικοποιήθηκαν σε γλώσσα προγραμματισμού C και εφαρμόστηκαν σε FSSP ποικίλων μεγεθών όπως δημοσιεύτηκαν από τον Taillard (1993) 5. O Taillard έδωσε τα ανώτερα και τα κατώτερα όρια της τιμής του makespan σε προβλήματα: 20x5, 20x10, 20x20, 50x5, 50x10 και 50x20 (εργασίες x μηχανές). Το ανώτερο όριο προσεγγίζεται από ευρετικούς αλγορίθμους. Όσο πιο κοντά στο ανώτερο όριο είναι η λύση που βρίσκει ο αλγόριθμος τόσο καλύτερη απόδοση έχει. Με αυτόν τον τρόπο θα γίνει η αξιολόγηση του αλγορίθμου που αναπτύσσεται στην παρούσα εργασία ως προς την ποιότητα της λύσης της. Ανά καιρούς έχουν δημοσιευτεί έρευνες οι οποίες αλλάζουν το ανώτερο όριο του Taillard, δηλαδή βρίσκουν καλύτερες λύσεις για κάποια προβλήματα (Pan et al, 2008).

ΙΙ.3 Αξιολόγηση της αντικειμενικής συνάρτησης

Το κριτήριο προς ελαχιστοποίηση είναι το makespan, δηλαδή ο χρόνος περάτωσης της τελευταίας εργασίας. Η αντικειμενική συνάρτηση του προβλήματος είναι:

$$\min(C_{max}) = \min(\max(C_{iM})), \quad i = 1,2,\ldots,n$$

Όπου C_{max} ο χρόνος λήξης του προγράμματος (makespan) και C_{iM} ο χρόνος περάτωσης κάθε εργασίας στην τελευταία μηχανή M.

Υπολογίζω τον χρόνο περάτωσης της κάθε εργασίας και δημιουργώ τον πίνακα κόστους C του οποίου οι σειρές συμβολίζουν τις μηχανές και οι στήλες τις εργασίες. Στην πρώτη μηχανή οι εργασίες εκτελούνται σειριακά χωρίς να περιμένει η μηχανή για την επόμενη εργασία. Επίσης, η πρώτη εργασία εκτελείται χωρίς να περιμένει, σε όλες τις μηχανές. Για όλες τις υπόλοιπες περιπτώσεις όμως θα πρέπει να υπολογιστεί αν η μηχανή είναι διαθέσιμη και περιμένει την εργασία ή αν η εργασία είναι αναγκασμένη να περιμένει τη μηχανή:

$c_{i,j} = \max(c_{i-1,j}, c_{i,j-1}) + p_{i,j}$ (όπου $p_{i,j}$ οι χρόνοι της εργασίας j στην μηχανή i).

Ο πίνακας κόστους των δεδομένων ενός παραδείγματος με σειρά εργασιών E1, E2, E3, E4 θα είναι:

$$C \begin{vmatrix} 5 & 15 & 17 & 21 \\ 9 & 18 & 24 & 32 \\ 12 & 20 & 32 & 41 \\ 22 & 27 & 35 & 42 \end{vmatrix} =$$

Το στοιχείο προς ελαχιστοποίηση, δηλαδή το makespan, είναι το στοιχείο της τελευταίας σειράς και της τελευταίας γραμμής. Δηλαδή το makespan του παραδείγματος είναι Cmax = 42.

Έτσι λοιπόν η αξιολόγηση της αντικειμενικής συνάρτησης γίνεται με βάση τον πίνακα C. Για όλες τις λύσεις δημιουργείται ο πίνακας κόστους και με τα κριτήρια του αλγορίθμου Tabu Search γίνεται επιλογή των καλύτερων λύσεων.

Όπου M ο αριθμός των μηχανών, D ο αριθμός των εργασιών, array το διάνυσμα που ισούται με κάποιο μέλος του πληθυσμού, C ο πίνακας κόστους και p ο πίνακας δεδομένων (η χρονική διάρκεια που απαιτείται για κάθε εργασία σε κάθε μηχανή).

Ο αλγόριθμος για τον υπολογισμό της αντικειμενικής:

ΓΙΑ κάθε μηχανή i **ΕΠΕΝΑΛΑΒΕ**

ΓΙΑ κάθε εργασία j **ΕΠΑΝΑΛΑΒΕ**

ΑΝ i = 1 **ΚΑΙ** j = 1 **ΤΟΤΕ**

C[1][1] = p[1][array[1]]

ΑΛΛΙΩΣ ΑΝ i = 1 **ΚΑΙ** j ≠ 1 **ΤΟΤΕ**

C[1][j] = C[1][j-1] + p[1][array[j]]

ΑΛΛΙΩΣ ΑΝ i ≠ 1 **ΚΑΙ** j = 1 **ΤΟΤΕ**

C[i][1] = C[i-1][1] + p[i][array[1]]

ΑΛΛΙΩΣ ΑΝ i ≠ 1 **ΚΑΙ** j ≠ **ΚΑΙ** C[i-1][j] ≥ C[i][j-1] **ΤΟΤΕ**

C[i][j] = C[i-1][j] + p[i][array[j]]

ΑΛΛΙΩΣ

C[i][j] = C[i][j-1] + p[i][array[j]]

ΤΕΛΟΣ ΑΝ

ΤΕΛΟΣ ΕΠΑΝΑΛΗΨΗΣ

ΤΕΛΟΣ ΕΠΑΝΑΛΗΨΗΣ

makespan=C[M][D]

ΙΙ.4 Αρχικοποίηση

Ο αλγόρθμος tabu search βασίζεται σε μία ήδη υπάρχουσα σειρά εργασιών, πάνω στην οποία, με εναλλαγή εργασιών και απαγόρευση κινήσεων, επέρχεται η ελαχιστοποίηση της αντικειμενικής που στην προκειμένη είναι το ζητούμενο.

Αρχικά προγράμματα ονομάζονται τα προγράμματα που παράγονται με οποιοδήποτε τρόπο και απότελούν αρχικές λύσεις αλγορίθμων τοπικής αναζήτησης. Οι μέθοδοι παραγωγής αρχικών λύσεων είναι σημαντικές καθώς αυτά καθορίζουν σε μεγάλο βαθμό την ποιότητα των τελικών λύσεων.

Ο αλγόριθμος tabu search είναι ένας αλγόριθμος τοπικής αναζήτησης πράγμα που σημαίνει πως επικεντρώνεται σε ένα σημείο και προσπαθεί να ελαχιστοποιήσει την αντικειμενική γύρω από αυτό. Αυτό δεν είναι θεμιτό γιατί ο αλγόριθμος συνήθως παγιδεύεται σε κάποιο τοπικό ελάχιστο. Αυτό αποφεύγεται με διάφορα τεχνάσματα που οδηγούν τον αλγόριθμο εκτός της περιοχής του τοπικού ελάχιστου. Στην παρούσα υλοποίηση του αλγορίθμου η αρχική λύση καθορίζεται τυχαία. Αυτό πλεονεκτεί γιατί ανιχνεύεται το ζητούμενο ελάχιστο σε μεγαλύτερη γκάμα λύσεων. Ωστόσο με αυτόν τον τρόπο η ποιότητα των αρχικών λύσεων εξαρτάται άμεσα από το μέγεθος του χώρου των αρχικών προγραμμάτων. Όσο μεγαλύτερο το πλήθος των προγραμμάτων που περιέχονται στο χώρο αυτό τόσο πιθανότερο να υπάρχει κάποια αρχική λύση που να μας οδηγήσει σε κάποια καλή τελική λύση. Αυτό έχει μεγάλο υπολογιστικό κόστος αλλά έτσι ελέγχονται περιοχές που φαινομενικά είναι κακές όσον αφορά την ποιότητα τους αλλά μπορεί να περιλαμβάνουν καλές λύσεις.

ΙΙ.5 Γειτονιές Αναζήτησης

Η λειτουργία της μεθόδου τοπικής αναζήτησης απαιτεί τον χωρισμό του χώρου όλων των προγραμμάτων σε υποπεριοχές – γειτονιές αναζήτησης. Αυτές οι υποπεριοχές δημιουργούνται και σχετίζονται άμεσα με την μορφή της τοπικής αναζήτησης. Επιλέγεται δηλαδή ένα είδος κίνησης με την οποία μπορούμε από ένα πρόγραμμα να παράγουμε ένα καινούργιο. Με το ίδιο είδος κίνησης λειτουργώντας ως βρόγχος παράγουμε ένα νέο πρόγραμμα και αυτό συνεχίζεται μέχρι να σταματήσει ο βρόγχος ανάλογα με τις συνθήκες. Στον κατασκευασθείσα αλγόριθμο θα δημιουργούνται οι γειτονιές με ανταλλαγή 1-1.

ΙΙ.6 Εφαρμογή του Αλγορίθμου Αναζήτησης με Απαγόρευση Κινήσεων (Tabu Search)

Ο σκοπός της συγκεκριμένης ενότητας είναι να διερευνήσει τον αλγόριθμο tabu search και να σχολιάσει την απόδοση του σε προβλήματα του Taillard. Αφού παρουσιαστούν τα βασικά σημεία του αλγορίθμου και οι συναρτήσεις που τον απαρτίζουν, θα γίνει προσπάθεια προσδιορισμού των παραμέτρων και της κατάλληλης στρατηγικής του αλγορίθμου για το FSSP μέσα από στατιστική διαδικασία. Το κρισιμότερο σημείο στην απόδοση των αλγορίθμων τοπικής αναζήτησης είναι όπως προαναφέρθηκε η παγίδευσή τους σε γειτονιές τοπικών ελάχιστων. Οι αλγόριθμοι τοπικής αναζήτησης εξετάζουν σε κάθε επανάληψη μία γειτονιά προγραμμάτων της τρέχουσας λύσης επιλέγοντας το ελάχιστο της γειτονιάς. Με αυτό τον τρόπο η λύση βελτιώνεται σταδιακά έως να εντοπισθεί κάποιο τοπικό ελάχιστο. Στο σημείο αυτό η αναζήτηση εγκλωβίζεται στη γειτονιά του τοπικού ελάχιστου, καθώς καμία κίνηση προς καμία κατεύθυνση δεν βελτιώνει πλέον τη λύση. Στην περίπτωση αυτή η αναζήτηση τερματίζει και η τελική λύση αποτελείται από το τοπικό ελάχιστο. Η κατεύθυνση της αναζήτησης μπορεί να στραφεί σε τοπικά ελάχιστα με απαγόρευση μεμονωμένων κινήσεων. Συγκεκριμένα μία κίνηση η οποία οδηγεί σε καλύτερη λύση απαγορεύεται να τροποποιηθεί για ένα ορισμένο διάστημα εκτέλεσης του βρόγχου.

Η απαγόρευση μεμονωμένων κινήσεων ωστόσο δεν εξασφαλίζει ούτε τον απεγκλωβισμό από περιοχές τοπικών ελαχίστων ούτε αποτρέπει κύκλους κατά την αναζήτηση. Κρίνεται απαραίτητο για να αποφευχθούν άσκοποι κύκλοι η απαγόρευση συνόλου κινήσεων. Το μέγεθος του συνόλου αυτού θα εξεταστεί στην ενότητα «προσδιορισμός παραμέτρων και στρατηγικής» μαζί με άλλες παραμέτρους που μπορεί να επηρεάσουν την τελική λύση.
Στον παρακάτω πίνακα δίνονται τα αποτελέσματα της εφαρμογής του αλγορίθμου σε προβλήματα του Taillard. Οι παράμετροι σε πρώτη φάση επιλεχθήκαν τυχαία. Συγκεκριμένα το πλήθος τον αρχικών λύσεων πήρε την τιμή 4000, το μέγεθος της tabu list πήρε την τιμή 8 (δηλαδή οχτώ εργασίες βρίσκονται σε κάθε εκτέλεση του βρόγχου στην λίστα των απαγορευμένων κινήσεων – δεν μπορούν να αλλάξουν θέση, εκτός αν οδηγούν σε καλύτερο αποτέλεσμα) και η «εντατικοποίηση» την τιμή 5 (δηλαδή για κάθε εργασία θα εξεταστεί η ανταλλαγή της 1-1 με όλες της υπόλοιπες επί πέντε φορές, παράλληλα με την λειτουργία της λίστας των απαγορευμένων κινήσεων).

Είδος προβλήματος	Πρόβλημα	Ανώτερο όριο Taillard (2001)	Αποτελέσματα εφαρμογής αλγορίθμου TABU SEARCH

			Makespan	**Σειρά εργασιών**
20 x 5	Tai01	1278	1278	9 15 6 19 1 3 8 5 17 7 11 4 2 13 18 14 16 10 20 12
	Tai02	1359	1360	6 19 10 7 15 17 20 9 14 1 3 16 5 18 12 8 13 4 11 2
	Tai03	1081	1081	3 4 16 11 1 20 18 19 14 13 12 7 5 10 17 9 6 8 15 2
	Tai04	1293	1293	13 17 19 9 16 11 15 2 20 7 12 4 10 1 5 8 6 14 3 18
	Tai05	1236	1250	12 13 9 17 19 5 15 4 16 10 6 2 18 3 14 11 1 7 8 20
20 x 10	Tai01	1582	1591	18 5 9 12 17 19 15 3 4 2 10 6 14 8 20 11 13 7 1 16
20 x 10	Tai02	1659	1677	12 17 9 19 1 5 13 15 20 7 2 16 11 4 10 8 14 3 6 18
20 x 20	Tai01	2297	2319	16 8 7 18 15 10 12 13 5 9 14 11 6 17 1 20 2 4 3 19
50 x 5	Tai01	2724	2724	31 50 30 41 34 8 10 32 28 4 25 23 26 29 18 13 39 5 14 6 9 46 45 1 11 3 40 44 49 42 27 17 33 2 21 7 47 12 38 43 20 22 19 16 24 15 48 35 37 36
50 x 10	Tai01	3025	3087	18 20 22 19 44 14 49 36 31 2 43 7 15 11 12 6 32 29 28 42 46 27 38 13 30 1 8 3 21 41 50 26 33 4 17 5 25 10 35 23 16 9 40 47 37 34 24 48 45 39
50 x 20	Tai01	3875	3987	35 43 37 24 31 27 45 17 47 20 48 15 28 6 32 5 14 38 7 10 49 26 39 33 42 46 1 34 19 16 23 18 36 25 40 2 21 13 8 44 50 30 29 22 11 12 9 41 4 3

Παρατηρώ ότι στα προβλήματα 20 εργασιών τα αποτελέσματα είναι σχετικά κοντά με αυτά που δίνει ο Taillard, ενώ στα προβλήματα 50 εργασιών όχι. Αυτό είναι απόλυτα φυσιολογικό καθώς τα προβλήματα 20 εργασιών έχουν (20!) λύσεις δηλαδή 2432902008176640000, ενώ τα προβλήματα 50 εργασιών έχουν (50!) λύσεις δηλαδή 3.0414093201713378043612608166065e+64. Χρειάζονται λοιπόν αρκετά περισσότερες αρχικές λύσεις από τις 4000, πράγμα που θα φανεί σε επόμενη παράγραφο.

II.7 Προσδιορισμός Παραμέτρων και Στρατηγικής

Ο προσδιορισμός των παραμέτρων είναι ένα από τα βασικά θέματα που πρέπει να διερευνηθούν, ενώ η επιλογή των κατάλληλων τιμών για τις παραμέτρους ελέγχου του Tabu Search εξαρτάται από το πρόβλημα.

Με γνώμονα τη στατιστική θα γίνει προσπάθεια προσδιορισμού της καλύτερης στρατηγικής με συγκεκριμένες παραμέτρους αλλά και με διάφορους συνδυασμούς παραμέτρων.

Μία παράμετρος του αλγορίθμου είναι το πλήθος των αρχικών λύσεων που θα εξεταστούν περαιτέρω με απαγόρευση κινήσεων. Αυτή η παράμετρος σχετίζεται άμεσα με το μέγεθος του προβλήματος καθώς όσο μεγαλώνει το πρόβλημα κρίνεται απαραίτητη η αναζήτηση σε περισσότερες αρχικές λύσεις. Στον παρακάτω πίνακα παρουσιάζονται τα αποτελέσματα του αλγορίθμου αλλάζοντας τον αριθμό των αρχικών λύσεων. Το μέγεθος της tabu list επιλέχθηκε να είναι ίσο με 10.

Είδος προβλήματος	**Πρόβλημα**	**Ανώτερο όριο Taillard (2001)**	**Makespan 4000**	**Makespan 8000**	**Makespan 12000**	**Σειρά εργασιών καλύτερου χρόνου**
20 x 5	Tai01	1278	1278	1278	1278	9 15 6 19 1 3 8 5 17 7 11 4 2 13 18 14 16 10 20 12
	Tai02	1359	1360	1360	1360	6 19 10 7 15 17 20 9 1 3 16 5 18 12 8 13 4 14 11 2
	Tai03	1081	1089	1089	1089	3 4 16 14 20 18 7 1 19 12 11 13 10 5 17 6 9 8 15 2
	Tai04	1293	1295	1295	1295	13 17 19 9 16 11 4 10 7 20 2 5 1 12 15 8 14 6 3 18
	Tai05	1236	**1235**	**1235**	**1235**	3 12 19 5 13 17 4 9 16 6 2 10 11 15

						14 7 18 1 20 8
20 x 10	Tai01	1582	1583	1583	1583	18 5 2 12 17 3 4 13 15 9 10 6 19 8 20 11 14 7 1 16
	Tai02	1659	1677	1675	1675	19 15 12 9 7 17 20 1 5 11 10 13 2 4 16 8 14 3 6 18
20 x 20	Tai01	2297	2311	2311	2311	16 18 8 13 10 9 14 11 5 6 12 7 15 1 17 20 2 4 3 19
50 x 5	Tai01	2724	2724	2724	2724	31 30 40 26 34 4 10 17 29 32 12 27 45 25 28 39 5 22 38 50 15 49 18 43 44 9 42 14 8 21 20 7 2 3 47 24 6 1 16 13 46 48 11 19 41 33 35 23 37 36
50 x 10	Tai01	3025	3102	3075	3075	18 44 20 34 33 12 42 2 43 19 15 3 49 25 31 11 13 36 6 14 37 30 9 4 41 5 35 8 46 32 21 47 48 38 29 10 17 16 28 22 50 26 40 23 7 1 45 27 24 39
50 x 20	Tai01	3875	4000	3989	3989	35 43 15 31 6 5 1 42 17 27 14 45 44 33 20 39 8 40 10 16 19 28 46 2 26 24 13 32 25 7 34 12 23 36 29 47 48 49 22 30 38 11 18 4 50 21 37 9 41 3

Σε αυτή την παράγραφο θα εξεταστεί αν και πως το μέγεθος της λίστας των απαγορευμένων κινήσεων μπορεί να επηρεάσει την τελική λύση. Θα συγκριθούν οι λύσεις με τα όρια που δίνει ο Taillard σε προβλήματα διαφορετικού μεγέθους. Συγκεκριμένα θα εξεταστούν προβλήματα με 20x5 , 20x10, 20x20, 50x5, 50x10 και 50x20, όπου ο πρώτος αριθμός αφορά τις εργασίες ενώ ο δεύτερος τις μηχανές. Τα μεγέθη της λίστας που θα εξεταστούν θα είναι τα 8,10,16,20 και 24. Δηλαδή οι απαγορευμένες κινήσεις θα προέρχονται από 4,5,8,10 και 12 εργασίες αντίστοιχα. Οι εργασίες αυτές δεν θα μετακινούνται ωσότου εξέλθουν από τη λίστα. Όπως έχει προηγουμένως αναφερθεί δύο εργασίες μπαίνουν ταυτόχρονα στη λίστα όταν η 1-1 εναλλαγή τους οδηγεί σε καλύτερο αποτέλεσμα δηλαδή σε μικρότερη makespan.

Είδος προβλήματος	**Πρόβλημα**	**Ανώτερο όριο Taillard**	**Makespan tabulist 8**	**Makespan tabulist 10**	**Makespan tabulist 16**	**Makespan tabulist 20**	**Makespan tabulist 24**	**Σειρά εργασιών καλύτερου χρόνου**
20 x 5	Tai01	1278	1278	1278	1278	1278	1278	9 15 6 19 1 3 8 5 17 7 11 4 2 13 18 14 16 10 20 12
	Tai02	1359	1360	1360	1360	1360	1359	6 10 17 7 19 15 3 9 18 12 14 20 1 2 13 5 16 11 8 4
	Tai03	1081	1081	1089	1087	1083	1088	3 4 16 11 1 20 18 19 14 13 12 7 5 10 17 9 6 8 15 2
	Tai04	1293	1293	1295	1293	1293	1293	13 17 19 9 16 11 15 2 20 7 12 4 10 1 5 8 6 14 3 18
	Tai05	1236	1250	**1235**	1236	1235	1235	12 4 13 17 19 9 5 3 16 6 2 10 11 15 14 7 18 1 20 8

20 x 10	Tai01	1582	1591	1583	1586	1586	1593	18 5 2 12 17 3 4 13 15 9 10 6 19 8 20 11 14 7 1 16
	Tai02	1659	1677	1677	1676	1678	1669	19 15 12 13 17 9 2 7 20 11 10 1 5 16 4 8 14 3 6 18
20 x 20	Tai01	2297	2319	2311	2316	2316	2315	16 18 8 13 10 9 14 11 5 6 12 7 15 1 17 20 2 4 3 19
50 x 5	Tai01	2724	2724	2724	2724	2724	2724	31 30 40 26 34 4 10 17 29 32 12 27 45 25 28 39 5 22 38 50 15 49 18 43 44 9 42 14 8 21 20 7 2 3 47 24 6 1 16 13 46 48 11 19 41 33 35 23 37 36
50 x 10	Tai01	3025	3087	3102	3082	3072	3075	18 22 6 3 33 20 44 15 38 49 43 36 8 42 37 34 10 1 31 30 29 13 40 12 41 46 9 32 35 47 4 21 19 16 23 11 17 2 27 14 5 45 28 26 25 50 7 24 48 39

50 x 20	Tai01	3875	3987	4000	3976	3974	3978	35 43 15 31 37 44 39 20 6 14 34 7 42 33 17 28 21 36 11 5 46 1 41 30 32 2 45 29 47 23 40 16 49 10 19 27 26 13 22 4 12 8 24 9 48 18 38 25 50 3

Ακολουθεί ο πίνακας με τις αποκλίσεις:

Είδος προβλήματος	Πρόβλημα	Ανώτερο όριο Taillard	Αποτέλεσμα tabu search	Απόκλιση
20 x 5	Tai01	1278	1278	0
	Tai02	1359	1359	0
	Tai03	1081	1081	0
	Tai04	1293	1293	0
	Tai05	1235	1235	0
20 x 10	Tai01	1582	1583	0,000632
	Tai02	1659	1669	0,006027
20 x 20	Tai01	2297	2311	0,006094
50 x 5	Tai01	2724	2724	0
50 x 10	Tai01	3025	3072	0,0155371
50 x 20	Tai01	3875	3974	0,0250322

ΣΥΜΠΕΡΑΣΜΑΤΑ

- Ο αλγόριθμος tabu search δίνει πολύ καλά αποτελέσματα σε σχέση με τον χρόνο εκτέλεσης του. Χαρακτηριστικό είναι ότι για τα προβλήματα 5 x 20 βρέθηκε το ανώτερο όριο σε πολύ σύντομο χρονικό διάστημα (3 λεπτά).

- Στα μεγάλα προβλήματα, δηλαδή όταν οι μηχανές και οι εργασίες είναι πολλές, το αποτέλεσμα δεν προσέγγισε το πάνω όριο του Taillard. Κρίνεται απαραίτητη η ύπαρξη ποιοτικότερων αρχικών λύσεων, πάνω στις οποίες θα εφαρμοστεί ο tabu search. Αυτό μπορεί να γίνει με αρκετά τεχνάσματα, όπως με την καταγραφή των τόξων που εμφανίζονται συχνότερα στις βέλτιστες λύσεις και εκ νέου εφαρμογή του αλγορίθμου σε αρχικές λύσεις που να περιέχουν αυτά τα τόξα σταθερά.

- Η εκτέλεση του αλγορίθμου με περισσότερες επαναλήψεις μπορεί να βελτιώσει το αποτέλεσμα ενός μεγάλου προβλήματος

- Το μέγεθος της λίστας των απαγορευμένων κινήσεων επηρεάζει την βέλτιστη λύση. Γίνεται ανίχνευση σε άλλους χώρους και ανιχνεύονται άλλα τοπικά ελάχιστα. Ωστόσο δεν μπορεί να καθοριστεί σχέση ανάμεσα στο μέγεθος της λίστας απαγορευμένων κινήσεων και στο μέγεθος του προβλήματος.

ΠΑΡΑΡΤΗΜΑ

Στους πίνακες που ακολουθούν βρίσκονται τα δεδομένα των προβλημάτων που μελετήθηκαν όπως τα δίνει ο Taillard.

Προβλήματα 20 x 5

```
Tai01
number of jobs, number of machines, initial seed, upper bound and
lower bound :
          20           5   873654221        1278        1232
processing times :
 54 83 15 71 77 36 53 38 27 87 76 91 14 29 12 77 32 87 68 94
 79  3 11 99 56 70 99 60  5 56  3 61 73 75 47 14 21 86  5 77
 16 89 49 15 89 45 60 23 57 64  7  1 63 41 63 47 26 75 77 40
 66 58 31 68 78 91 13 59 49 85 85  9 39 41 56 40 54 77 51 31
 58 56 20 85 53 35 53 41 69 13 86 72  8 49 47 87 58 18 68 28

Tai02
number of jobs, number of machines, initial seed, upper bound and
lower bound :
          20           5   379008056        1359        1290
processing times :
 26 38 27 88 95 55 54 63 23 45 86 43 43 40 37 54 35 59 43 50
 59 62 44 10 23 64 47 68 54  9 30 31 92  7 14 95 76 82 91 37
 78 90 64 49 47 20 61 93 36 47 70 54 87 13 40 34 55 13 11  5
 88 54 47 83 84  9 30 11 92 63 62 75 48 23 85 23  4 31 13 98
 69 30 61 35 53 98 94 33 77 31 54 71 78  9 79 51 76 56 80 72

Tai03
number of jobs, number of machines, initial seed, upper bound and
lower bound :
          20           5  1866992158        1081        1073
processing times :
 77 94  9 57 29 79 55 73 65 86 25 39 76 24 38  5 91 29 22 27
 39 31 46 18 93 58 85 58 97 10 79 93  2 87 17 18 10 50  8 26
 14 21 15 10 85 46 42 18 36  2 44 89  6  3  1 43 81 57 76 59
 11  2 36 30 89 10 88 22 31  9 43 91 26  3 75 99 63 83 70 84
 83 13 84 46 20 33 74 42 33 71 32 48 42 99  7 54  8 73 30 75

Tai04
number of jobs, number of machines, initial seed, upper bound and
lower bound :
          20           5   216771124        1293        1268
processing times :
 53 19 99 62 88 93 34 72 42 65 39 79  9 26 72 29 36 48 57 95
 93 79 88 77 94 39 74 46 17 30 62 77 43 98 48 14 45 25 98 30
 90 92 35 13 75 55 80 67  3 93 54 67 25 77 38 98 96 20 15 36
 65 97 27 25 61 24 97 61 75 92 73 21 29  3 96 51 26 44 56 31
 64 38 44 46 66 31 48 27 82 51 90 63 85 36 69 67 81 18 81 72
```

```
Tai05
number of jobs, number of machines, initial seed, upper bound and
lower bound :
          20           5   495070989        1236        1198
processing times :
 61 86 16 42 14 92 67 77 46 41 78  3 72 95 53 59 34 66 42 63
 27 92  8 65 34  6 42 39  2  7 85 32 14 74 59 95 48 37 59  4
 42 93 32 30 16 95 58 12 95 21 74 38  4 31 62 39 97 57  9 54
 13 47  6 70 19 97 41  1 57 60 62 14 90 76 12 89 37 35 91 69
 55 48 56 84 22 51 43 50 62 61 10 87 99 40 91 64 62 53 33 16
```

Προβλήματα 20 x 10

```
Tai01
number of jobs, number of machines, initial seed, upper bound and
lower bound :
          20          10   587595453        1582        1448
processing times :
 74 21 58  4 21 28 58 83 31 61 94 44 97 94 66  6 37 22 99 83
 28  3 27 61 34 76 64 87 54 98 76 41 70 43 42 79 88 15 49 72
 89 52 56 13  7 32 32 98 46 60 23 87  7 36 26 85  7 34 36 48
 60 88 26 58 76 98 29 47 79 26 19 48 95 78 77 90 24 10 85 55
 54 66 12 57 70 82 99 84 16 41 23 11 68 58 30  5  5 39 58 31
 92 11 54 97 57 53 65 77 51 36 53 19 54 86 40 56 79 74 24  3
  9  8 88 72 27 22 50  2 49 82 93 96 43 13 60 11 37 91 84 67
  4 18 25 28 95 51 84 18  6 90 69 61 57  5 75  4 38 28  4 80
 25 15 91 49 56 10 62 70 76 99 58 83 84 64 74 14 18 48 96 86
 15 84  8 30 95 79  9 91 76 26 42 66 70 91 67  3 98  4 71 62
```

```
Tai02
number of jobs, number of machines, initial seed, upper bound and
lower bound :
          20          10  1401007982        1659        1479
processing times :
 80 13 64 77 17 78 82  4 72 93 68 25 67 80 43 93 21 33 14 30
 59 83 85 85 70 35  2 76 46 72 69 46  3 57 71 77 33 49 59 82
 59 70 76 10 65 19 77 86 21 75 96  3 50 57 66 84 98 55 70 32
 31 64 11  9 32 58 98 95 25  4 45 60 87 31  1 96 22 95 73 77
 30 88 14 22 93 48 10  7 14 91  5 43 30 79 39 34 77 81 11 10
 53 19 99 62 88 93 34 72 42 65 39 79  9 26 72 29 36 48 57 95
 93 79 88 77 94 39 74 46 17 30 62 77 43 98 48 14 45 25 98 30
 90 92 35 13 75 55 80 67  3 93 54 67 25 77 38 98 96 20 15 36
 65 97 27 25 61 24 97 61 75 92 73 21 29  3 96 51 26 44 56 31
 64 38 44 46 66 31 48 27 82 51 90 63 85 36 69 67 81 18 81 72
```

Προβλήματα 20 x 20

```
number of jobs, number of machines, initial seed, upper bound and
lower bound :
          20          20   479340445        2297        1911
processing times :
 50 90 39 34 66 81 27 48 46 68 48 92 78 84 93 39 43  1 65 87
 78 56  9 43 84 73 66 38 83 57 97 52 77 13 12  2 65 93 39  1
 36 43 10 19 55 48 85 70 82 39 91 82 85 17  6 54 87 85  4 72
 85 88 60 98  4 99 53 21 33 53 63 18 45 29 43 41 80  4 31 19
```

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

Προβλήματα 50 x 5

```
number of jobs, number of machines, initial seed, upper bound and
lower bound :
          50           5  1328042058        2724        2712
processing times :
 75 87 13 11 41 43 93 69 80 13 24 72 38 81 83 88 26  6 89 67 70 30 89
30 68 21 78 46 99 10 17 23 83 47 86 18 67 46  4 14  4 20 88 50 84 58
93 76 50 30
 26 37 25 95 49 12 59 17 46 20 52 44 92 75 95 33 10 45  2 62 62 82 29
29 94 20 42 80 94 35  8 41 65  4 71 30 14 32 50 30 27 98 39 84 65 12
58 45 49 15
 48  4 92 92 72 45  5 98 93 17 79 11 16 89 81 92 45 61 39 28 94 87 23
1 55 91 67 91  4 60 38 25 90 93 13 65 25 34 47 98 91 11 46 50 77  5
14 47 80 45
 26 67  4 14 93 54 21 20  6 18 75 25 16 77 28 24 15 77 36 16 32 46 21
81 28 70 89 54 96 62 46 60 19 97 13  7 44  7 73 15 66 70 97 33 97 64
73 28  4 87
 77 94  9 57 29 79 55 73 65 86 25 39 76 24 38  5 91 29 22 27 39 31 46
18 93 58 85 58 97 10 79 93  2 87 17 18 10 50  8 26 14 21 15 10 85 46
42 18 36  2
```

Προβλήματα 50 x 10

```
number of jobs, number of machines, initial seed, upper bound and
lower bound :
          50          10  1958948863        3025        2907
processing times :
 46 52 79 45 97 10 44 24 85 75 66 49 95 61 19 47 84 13 11 19 98  2 85
44  7 73 19 69 12 73 85 23 53 16 88  8 26 42 58 63  7  2 44 38 24 76
85 61 32 90
 61 87 51 25 73 93 28 90 94 59 64  2 16 35 53 40 81 26 85  4  4 10 63
96 55 71 66 94  7 15 11 99 37 50 56 69 22 56 67 63 96 74  4 42 40 30
93 36 25 87
  3  1 58 85 33 71 58 56 64 43 48 69 96 35 82 53 64 11 61 36 53 87 88
10 32 38 25 24 90  7 11 49  2 76 17 32 39  9 83 69 67 28 88 23 91 71
3 26 41 96
```

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

Προβλήματα 50 x 20

```
number of jobs, number of machines, initial seed, upper bound and
lower bound :
          50          20  1539989115        3875        3480
processing times :
 52 95 42 75 44 57 89 53 84 62 91 14 95 89  4 95  2 97 68 20 33 51 98
8 85 86 73  4 40 98 12 59 44 46  2 41 28 83 28 21 80 71  4 60 34 55
53 96 37 37
 63 99 69 70 53 21 10 31 80 18  5 18 17 71 90 93 14 49 52  7 78 57 41
75 98 93 33 75 68 33 60 82 24 99  4 97 24 50 55 91 46 58 17 47 82  6
15 91 74 42
 82 21 79 95 46 23 40 95 87 37 24 24 65 62 19 67 66  6 65 59  2 67 82
90 30 63  5 93 53 85 81 73 34 74 13 78 35 20 16 48 12 11 80  9 24 76
32 35 66 48
 16 26 46 66 76 31 36  8 37 21  3 76 67  5 47 72 66 56 95 49 47 26 81
56 76 66 36 53 26 52 29 36 68 21 71 61 71 69 28 86 27 41 86 55 17 62
96 59 53 93
 63 55 59 35 21 59 78 25 30 38 78 79 58 44 38 76 70 72 85  8 10 84 42
67 20 24 75 23 33 60 20 75 83 26 92 29 39 14 74 66 86 10 27  8  7 97
84 56 61  9
 94 34 89 62 47 66 76 15 18 54 24 55 96 10 12 96 53 92 77  6 91 14 41
30 85 17 23 60 76 39 85 10 65 15 55 41 28 93 88 27 77 81 19 76 55 67
65  8 18 56
 79 21 93 32  8 45 37 78 26 98 17 25 21 28 68 24 62 89 60 64 38 90 87
1 99 34  9 22 74 14 14 84 75 37 32 29 32 89 12 47 19 97  7 12 43 89
14 33 56 57
 22  6 24 55 48 57 78  5 50 83 70 21 71 58 36 50 31 86 29 30 93 49 83
89 44 38 62 45 22 85 39 98 56 68 84 77 67 53 46 24 52 96  2 88 33 27
49 78 82 65
 80 13 64 77 17 78 82  4 72 93 68 25 67 80 43 93 21 33 14 30 59 83 85
85 70 35  2 76 46 72 69 46  3 57 71 77 33 49 59 82 59 70 76 10 65 19
77 86 21 75
 96  3 50 57 66 84 98 55 70 32 31 64 11  9 32 58 98 95 25  4 45 60 87
31  1 96 22 95 73 77 30 88 14 22 93 48 10  7 14 91  5 43 30 79 39 34
77 81 11 10
```

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

ΒΙΒΛΙΟΓΡΑΦΙΑ

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

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

[3]: Lenstra J.K, Rinnooy Kan A.H.G., Brucker P. «Complexity of machine scheduling problems» Annals of Discrete Mathematics, 1 (1977) 343-362

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

[5]: Παπαδάκης Δ. «Αλγόριθμοι τοπικής αναζήτησης με απαγόρευση κινήσεων και παραλληλοποίηση της αναζήτησης για την επίλυση του προβλήματος παραγωγής συνεχούς ροής» Master of science thesis, Τμήμα Επιστήμης Υπολογιστών, Κρήτη (1995) 15-16

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

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

[8]: Lin-Yu Tseng, Ya-Tai Lin «A hybrid genetic local search algorithm for the permutation flowshop scheduling problem» European Journal of Operational Research, 198 (2009) 84–92

[9]: Baker, K., & Trietsch, D. «Principles of sequencing and scheduling.» Hoboken - New Jersey: John Wiley & Sons (2009).

[10]: Αναγνωστόπουλος, Κ., & Κώτσικας, Λ. «Στατιστικές τεχνικές για την αξιολόγηση ευρετικών αλγορίθμων» 17ο Πανελλήνιο Συνέδριο Στατιστικής, Λευκάδα (2004) 67 -74

[11]: Allen, J., & Minton, S. «Selecting the right heuristic algorithm: runtime performance predictors» 11th Biennial Conference of the Canadian Society for Computational Studies of Intelligence on Advances in Artificial Intelligence (1996) 41-53

[12]: Garey, M.R., Johnson, D.S., Sethi, R. «The complexity of flowshop and jobshop scheduling» Mathematics of Operations Research, 1 (1976) 117–129

[13]: Campbell HG, Dudek RA, Smith ML «A heuristic algorithm for the n-job, m-machine sequencing problem» Management Science, 16 (1970) B 7-630

[14]: Kuo-Ching Ying, Ching-Jong Liao «An ant colony system for permutation flow-shop sequencing» Computers & Operations Research, 31 (2004) 791–801

[15]: Nawaz, M., Enscore, Jr. E., Ham, I. «A heuristic algorithm for the m-Machine, n-Job flow-Shop Sequencing Problem» Omega, 11 (1983) 91–95

[16]: Ogub, F.A., Simith, D.K. «Simulated annealing for the permutation flowshop problem» Omega 19 (1990) 64–67

[17]: Ishubuchi, M., Masaki S., Tanaka, H. «Modified simulated annealing for the flow shop sequencing problems» European Journal of Operational Research, 81 (1995) 388–398

[18]: Taillard, E. «Benchmarks for basic scheduling problems» European Journal of Operational Research, 64 (1993) 278–285

[19]: Nowicki, E., Smutnicki, C. «A fast tabu search algorithm for the permutation flow-shop problem» European Journal of Operational Research, 91 (1996) 160–175

[20]: Grabowski, J., Wodecki, M. «A very fast tabu search algorithm for the permutation flow shop problem with makespan criterion» Computers and Operational Research, 31 (2004) 1891–1909

[21]: Reeves, C.R., «A genetic algorithm for flowshop sequencing» Computers and Operational Research, 22 (1995) 5–13

[22]: Reeves, C.R., Yamada, T. «Genetic algorithm, path re-linking and the flowshop sequencing problem» Evolutionary Computation, 6 (1998) 45–60

[23]: Wang, C., Chu, C., Proth, J.-M. «Heuristic approaches for n/m/F//ΣCi scheduling problems» European Journal of Operational Research, 96 (1997) 636–644

[24]: Dr. Tarantilis C.D. «Εφαρμογές Διοικητικής Επιστήμης» Operation Research & Management Science, Οικονομικό Πανεπιστήμιο Αθηνών

[25]: Glover, F. «Future paths for integer programming and links to artificial intelligence» Computers and Operations Research, 13 (1986) 533 – 549

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

[27]: Dudek R.A., Teuton Jr. O.F. «Development of M-stage decision Rule for Scheduling n Jobs Through m Machines» Operations Research, 12 (1964) 471-497

[28]: Smith R.D., Dudek R.A. «A General Algorithm for Solution of the n-job,m-machine problem of Flow-Shop» Operations Research, 15 (1967) 71-82

[29]: Glover F. «Tabu Search- Part I» Journal on Computing, 1 (1989) 190-206

[30]: Ben-Daya M., Al-Fawzan M., «A Tabu Search Approach for the flow shop scheduling problem» European Journal of Operational Research, 109 (1998) 88-95